CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Conformal Predictive Decision Making

A Comparative Study with Bayesian and Point-Predictive Methods

Master's thesis in Computer science and engineering

Simon Lanngren
Martin Toremark

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's thesis 2025

Conformal Predictive Decision Making

A Comparative Study with Bayesian and Point-Predictive Methods

Simon Lanngren
Martin Toremark

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Conformal Predictive Decision Making
A Comparative Study with Bayesian and Point-Predictive Methods
Simon Lanngren
Martin Toremark

Supervisor: Johan Hallberg Szabadváry, Algorithma AB
Examiner: Marina Axelsson-Fisk, Department of Mathematical Sciences

Master's Thesis 2025
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2025

Conformal Predictive Decision Making
A Comparative Study with Bayesian and Point-Predictive Methods
Simon Lanngren
Martin Toremark
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

In many real-world settings, machine learning (ML) predictions serve as intermediate outputs used to inform decision-making. However, quantifying and accounting for uncertainty in these decisions remains a fundamental challenge. Conformal Predictive Decision Making (CPDM) is a framework for decision-making under uncertainty that leverages Conformal Predictive Distributions (CPDs) to optimize outcomes over a specified utility function. In this work, we evaluate two CPDM variants on synthetic datasets and compare their performance to two alternative approaches: Bayesian Decision Theory (BDT) and Point Predictive Decision Making (PPDM). Our proposed CPDM algorithm significantly outperforms the previously established one in most cases, while also offering computational advantages. It also showed greater robustness than BDT and PPDM in scenarios involving noisy data and skewed utility functions.

Keywords: Conformal Prediction, Conformal Predictive Decision Making, Bayesian Decision Theory, Decision making

Acknowledgements

We would like to extend our sincere thanks to our company supervisor, Johan Hallberg Szabadváry, for his invaluable guidance and support throughout this thesis. We are especially grateful to him and to the team at Algorithma for introducing us to the field of conformal prediction and for providing the opportunity to explore it in depth within a highly engaging and supportive environment. We also wish to thank our Chalmers supervisor, Professor Marina Axelsson-Fisk, for her feedback and for overseeing the academic aspects of this work.

Simon Lanngren & Martin Toremark, Gothenburg, May 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BDT	Bayesian Decision Theory
BRR	Bayesian Ridge Regression
CP	Conformal Predicition
CPD	Conformal Predictive Distribution
CPDM	Conformal Predictive Decision Making
CPS	Conformal Predictive System
DPS	Deterministic Predictive System
GPR	Gaussian Process Regression
ICPS	Inductive Conformal Predictive Systems
IID	Independent and Identically Distributed
KNN	k-Nearest Neighbors
KRR	Kernel Ridge Regression
KRRPM	Kernel Ridge Regression Prediction Machine
LSPM	Least Square Prediction Machine
ML	Machine Learning
NNPM	Nearest Neighbors Prediction Machine
OLS	Ordinary Least Squares
PDMS	Predictive Decision-Making System
PPDM	Point Predictive Decision Making
RBF	Radial Basis Function
RPS	Randomized Predictive System
RR	Ridge Regression
UQ	Uncertainty Quantification

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables that have been used throughout this thesis.

Examples and Decisions

$\boldsymbol{x}$	Object
y	Label
$\hat{y}$	Predicted Label
$\boldsymbol{y}$	Observed labels
z	Example
d	Decision
d^*	Optimal decision

Sets, Bags, and Sequences

$\mathbb{N}$	The set of positive integers, $\{1, 2, \dots\}$
$\mathbb{R}$	The set of real numbers
$\overline{\mathbb{R}}$	The extended real numbers, $\mathbb{R} \cup \{-\infty, \infty\}$
$\mathbb{R}^+$	The positive real numbers
X	Object space
Y	Label space
Z	Example space
$\{z_1, \dots, z_n\}$	Set (no repetitions of elements)
$\{\!\!\{z_1, \dots, z_n\}\!\!\}$	Bag (may contain repeated elements)
σ	Comparison bag
$(z_1, \dots, z_n)$	Sequence (parentheses may be omitted)
Z^n	The set of all length-n sequences of elements of Z

$Z^{(n)}$	The set of all bags of size n with elements of Z
Z^*	The set of all finite sequences of elements of Z
$Z^{(*)}$	The set of all finite bags of elements of Z
$\mathcal{H}$	Feature space
D	Set of available decisions
$\|A\|$	The cardinality, or size, of the set A

Parameters

$\boldsymbol{w}$	Regression weights
w_0	Intercept (bias)
$\hat{\boldsymbol{w}}$	Optimal regression weights
$\boldsymbol{\theta}$	Kernel parameters

Hyperparameters

k	Number of neighbors
γ	Regularization
ℓ	Length scale
λ	Prior precision on weights
β	Prior precision of Gaussian observation noise

Probability Theory

P	Probability distribution
Q_n	Predictive distribution
$\mathbb{E}$	Expectation
$\mathcal{N}$	Normal distribution
$\mathcal{U}$	Uniform distribution
τ	Random number
Σ	Covariance
σ^2	Variance
μ	Mean

ϵ	Normally distributed noise

Functions

$\mathcal{K}$	Kernel function
ϕ	Feature map
U	Utility function
A	Conformity measure
Q	Conformal Predictive System (CPS)
Q_n	Conformal Predictive Distribution (CPD)
$\Delta Q_n(y)$	The increase in Q_n at point y
U	Utility function
F	Predictive decision-making system (PDMS)
F^*	Optimal PDMS

Evaluation Metrics

R_F	The regret for the PDMS F
R_c	Cumulative regret
$\bar{U}$	Average utility

Other Notations

α	Conformity score
$\boldsymbol{X}$	Design matrix
$\boldsymbol{H}$	Hat matrix
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{K}$	Kernel matrix
$\boldsymbol{k}$	Vector of kernel evaluations for the test object
N_{k_i}	The k-neighborhood of example i
π	Permutation of $\{1, \ldots, n\}$

Contents

List of Figures

List of Tables

1

Introduction

Machine learning (ML) has advanced rapidly in recent decades, resulting in the development of numerous highly effective algorithms. Although much of the field has focused on designing learning algorithms that improve predictive accuracy, predictions alone are often insufficient. In many real-world applications, decisions must ultimately be made and predictions often serve as input to the decision-making process (Agrawal et al., 2019). We refer to this integration of prediction and action as predictive decision-making. In this setting, the goal extends beyond generating accurate predictions: it involves using predictions to select actions that lead to the most favorable outcomes. The preferences of possible outcomes are typically formalized using utility functions, allowing decisions to be guided by the principle of expected utility maximization (von Neumann and Morgenstern, 1953).

A central challenge in predictive decision-making is handling uncertainty. Point estimates from many standard ML models may not be sufficiently reliable for critical decisions (Nemani et al., 2023). This motivates the need for uncertainty quantification (UQ) to support more informed, robust, and risk-aware decision-making.

A powerful and widely used approach for quantifying uncertainty in ML is probabilistic prediction using Bayesian methods, which provides a principled way for refining probability estimates over time (Murphy, 2023). Although Bayesian probability estimates reflect uncertainty, the methods do not guarantee validity. In other words, informally, the posterior distribution does not necessarily capture the true uncertainty. In contrast, the conformal prediction (CP) framework provides valid prediction sets under the assumption of exchangeability in the data (Vovk et al., 2005). However, the set-valued outputs of conformal predictors lack the expressiveness of full probability distributions, making their integration into predictive decision making less straightforward. To address this limitation, Vovk et al. (2017) extended the CP framework to probabilistic regression using conformal predictive systems (CPSs), which produce conformal predictive distributions (CPDs). Importantly, these predictive distributions are valid. However, they do not come with any efficiency guarantees, that is, how narrow the distribution is in relation to the true distribution.

Access to CPDs allowed Vovk and Bendtsen (2018) to apply the CP framework to decision-making and develop a coherent, systematic theory for conformal predictive decision-making (CPDM). While there has been extensive research in Bayesian decision theory (BDT), CPDM remains largely unexplored. Since its introduction, no

further studies have examined it, particularly in comparison to alternative methods such as BDT. In their original paper, the authors applied their theory only to the simple Mushroom dataset from the UCI repository (UCI, 1981). Although their results were promising, they concluded that further research is needed to evaluate CPDM's practical usefulness.

1.1 Purpose and Aim

The purpose of this study is to expand the existing literature on CPDM. First, we compare the approach proposed by Vovk and Bendtsen (2018) to a modified version that aligns with the general principles of BDT. Second, we compare CPDM directly to both BDT and point predictive decision making (PPDM), which relies solely on point estimates rather than predictive distributions to inform decisions.

The aim is to evaluate the relative effectiveness of these approaches in a binary decision-making setting, highlighting their respective strengths and limitations. This evaluation will be conducted using synthetic datasets, varying factors such as data availability, noise levels, and the skewness of the utility functions. Furthermore, the approaches will be tested in both online and inductive settings. This analysis will provide insight into their comparative performance across different scenarios.

1.2 Research Questions

To better understand the practical usefulness of CPDM and its relative strengths and limitations compared to BDT and PPDM, this study addresses the following research questions.

- How does the modified version of CPDM compare in terms of decision-making performance and robustness to the original version proposed by Vovk and Bendtsen (2018)?
- How does the best-performing CPDM framework compare to BDT and PPDM in terms of decision-making performance and robustness across both inductive and online learning settings?

In both cases, performance will be evaluated across varying conditions, including the relationship between features and the target variable, the amount of available data, the level of noise in the data, and the skewness of the utility function.

1.3 Delimitations

To limit the scope, the following delimitations will be applied in this thesis:

- Decision-making will be limited to binary decisions, as defining utility functions for multi-class decisions is significantly more complex due to the increased difficulty of evaluating trade-offs between multiple possible outcomes.
- The evaluation between CPDM and the other methods will focus solely on decision-making performance, excluding other considerations such as computational efficiency.
- The underlying models used in the CPDM framework will be limited to non-Bayesian models. Specifically, Ridge Regression (RR), k-Nearest Neighbors (KNN), and Kernel Ridge Regression (KRR), or corresponding conformalized versions in the online setting, will be evaluated. This restriction is intended to maintain a clear distinction between CPDM and BDT, as incorporating Bayesian models into the CPDM framework would blur the conceptual boundaries between the two. Additionally, the selected models are of relatively low complexity, which makes them computationally feasible to evaluate in the online setting while still providing a focused yet diverse set of modeling approaches.
- The underlying models in the BDT framework will be limited to Bayesian Ridge Regression (BRR) and Gaussian Processes Regression (GPR). This restriction is mainly due to computational constraints in the online setting and to ensure a fair comparison with models of similar capacity in the CPDM setting.

2

Background

In this chapter, we present the theoretical background necessary to understand the methods used in this study. We begin with an overview of the CP framework, followed by a detailed outline of CPSs. We then describe predictive decision making and provide additional details on CPDM. Finally, we conclude with an overview of the ML models used in the experiments and their conformalized versions.

2.1 Conformal Prediction

CP is a nonparametric UQ framework that provides prediction sets with finite-sample validity, guaranteeing that the probability of error does not exceed a predefined significance level, and is applicable to nearly any underlying ML model (Vovk et al., 2022). The framework operates in a supervised learning setting, where we assume access to a set of objects $\boldsymbol{x}_i \in X$ that are labeled as $y_i \in Y$, where X and Y are fixed, non-empty measurable spaces, referred to as the *object space* and *label space*, respectively. The label space Y can be a finite set (for classification problems) or the set of real numbers (for regression problems). For notational convenience, we also define the Cartesian product $Z := X \times Y$, is called the *example space*, and let $z_i := (\boldsymbol{x}_i, y_i)$.

In supervised machine learning, the standard assumption is that the examples are randomly drawn from an unknown probability distribution P over a space of possible examples Z. More precisely, we assume the examples are independent and identically distributed (IID) according to P. Vovk et al. (2022) refer to this as the *randomness assumption*, and say that we are *learning under unconstrained randomness*, since we have no prior knowledge of P.

Most of the CP framework is concerned with learning under unconstrained randomness, and most of its results hold under the randomness assumption. However, in many cases we only need the slightly weaker *exchangeability assumption*, in which we assume the data is drawn from an *exchangeable distribution*. A probability distribution P on Z^n, where $n \in \mathbb{N}$, is *exchangeable* if

$$P(E) = P(\{(z_i, \ldots, z_n) \in Z^n : (z_{\pi(1)} \ldots z_{\pi(n)}) \in E\}), \tag{2.1}$$

for all measurable sets $E \subseteq Z^n$ and all permutations π of $\{1, \ldots, n\}$ (Vovk et al., 2022). In essence, the exchangeability assumption says that the distribution of

examples is invariant under any permutation, i.e., the distribution does not depend on the order of the examples.

Another important aspect of the CP framework is that it typically operates online, making sequential predictions based on all previous examples, unlike traditional ML, which operates offline by deriving a prediction rule from a fixed batch of training data before applying it to new examples (Vovk et al., 2022). In the online setting, our objective is formally to correctly predict the true label $y_n \in Y$ based on a sequence of past examples: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{n-1}, y_{n-1}) \in Z^*$, and a new object $\boldsymbol{x}_n \in X$.

Although the CP framework is most naturally formulated in an online setting, it can be adapted to resemble an offline protocol—an important consideration, as many real-world applications involve offline components. This adaptation is achieved through the use of the inductive version of CP, in which the underlying ML model is first trained as usual and then calibrated on a separate dataset to obtain the prediction sets (Vovk et al., 2022).

Vapnik (1998) distinguishes between two approaches to prediction: induction and transduction. In induction, we first derive a general prediction rule from examples (the inductive step) and then apply this rule to new objects to generate predictions (the deductive step). In contrast, transduction skips the need for a general rule, moving directly from examples to predictions. Thus, transduction aligns with the online framework, while induction corresponds to the offline framework.

2.2 Conformal Predictive Systems

CPS is an extension of CP to the probabilistic setting, allowing the derivation of CPDs that are valid under the assumption of exchangeability. Importantly, CPDs encode more information than standard CP intervals; in particular, a single CPD can produce multiple CP prediction sets. However, one limitation of CPDs is that they can only be created in the context of probabilistic regression, where $Y = \mathbb{R}$.

We begin our formal description of CPSs by introducing the notion of a *conformity measure*. Following Vovk et al. (2022), a conformity measure is a measurable function:

$$A : Z^{(*)} \times Z \to \overline{\mathbb{R}}, \tag{2.2}$$

where $Z^{(*)}$ denotes the set of all finite *bags* of elements from Z, and $\overline{\mathbb{R}}$ represents the extended real numbers. A bag is an unordered collection that may contain repeated elements, unlike a set. For any bag of examples $\sigma := \{\!|z_1, \ldots, z_{n-1}|\!\} \in Z^{(*)}$, and any example $z_n \in Z$, the conformity measure A assigns a numerical score indicating how well z_n conforms to σ. In our setting, we are only interested in values $A(\sigma, z_n)$ where $z_n \in \sigma$. Since bags are unordered, the conformity measure A is invariant under permutations of the elements in σ.

Given a conformity measure A, a *smoothed conformal transducer* is a function $Q : Z^n \times [0,1] \to [0,1]$ defined as:

$$
\begin{aligned}
Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), \tau) := \frac{1}{n} \left|\{i = 1, \ldots, n : \alpha_i^y < \alpha^y\}\right| \\
+ \frac{\tau}{n} \left|\{i = 1, \ldots, n : \alpha_i^y = \alpha^y\}\right|
\end{aligned}
\tag{2.3}
$$

where $z_1, \ldots, z_{n-1} \in Z^{n-1}$ is a training set, $\boldsymbol{x}_n \in X$ is a test object, $\tau \in [0, 1]$ is a random number independent of everything else, and distributed uniformly in $[0, 1]$. The numbers α_i^y and α^y are the *conformity scores* defined by:

$$
\begin{aligned}
\alpha_i^y &:= A(\sigma, z_i), \quad i = 1, \ldots, n-1, \\
\alpha^y &:= A(\sigma, (\boldsymbol{x}_n, y)), \\
\sigma &:= \wr z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y) \wr,
\end{aligned}
\tag{2.4}
$$

for each $y \in \mathbb{R}$ (Vovk and Bendtsen, 2018; Vovk et al., 2022).

Vovk et al. (2022) defines a function $Q : Z^n \times [0, 1] \to [0, 1]$ as a *randomized predictive system* (RPS) if it satisfies the following two conditions:

C1 The function $Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), \tau)$ is monotonically increasing in both $y \in \mathbb{R}$ and $\tau \in [0, 1]$, for all $(z_1, \ldots, z_{n-1}) \in Z^{n-1}$ and all $\boldsymbol{x}_n \in X$.

C2 The function $Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), \tau)$ satisfies the following limits:

$$
\begin{aligned}
\lim_{y \to -\infty} Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), 0) = 0, \\
\lim_{y \to \infty} Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), 1) = 1,
\end{aligned}
\tag{2.5}
$$

for all $(z_1, \ldots, z_{n-1}) \in Z^{n-1}$ and all $\boldsymbol{x}_n \in X$.

A function that is both a smoothed conformal transducer, determined by some conformity measure, and a randomized predictive system is called a *conformal predictive system* (CPS) (Vovk et al., 2022). Given a training set $(z_1, \ldots, z_{n-1}) \in Z^{n-1}$ and test object $\boldsymbol{x}_n \in X$, a CPS Q outputs the function:

$$
Q_n : (y, \tau) \in \mathbb{R} \times [0, 1] \mapsto Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), \tau), \tag{2.6}
$$

which is known as a *conformal predictive distribution* (CPD), a cumulative distribution function constructed from p-values. Under conditions **C1** and **C2**, Q_n is a monotonically increasing function in y that satisfies:

$$
\lim_{y \to -\infty} Q_n(y) = 0 \quad \text{and} \quad \lim_{y \to \infty} Q_n(y) = 1, \tag{2.7}
$$

ensuring that Q_n behaves like a proper cumulative distribution function, without requiring it to be right-continuous (Vovk and Bendtsen, 2018; Vovk et al., 2022).

Vovk et al. (2022) defined the *thickness* of a CPD Q_n as the infimum of all $\epsilon \geq 0$ such that, for all but finitely many $y \in \mathbb{R}$, the set:

$$\{Q_n(y, \tau) : \tau \in [0, 1]\} \tag{2.8}$$

has diameter at most ϵ; that is,

$$Q_n(y, 1) - Q_n(y, 0) \leq \epsilon. \tag{2.9}$$

The finitely many y-values where this condition fails are called *exception points*, and their total number is refereed to as the *exception size* of Q_n.

Vovk et al. (2022) define a smoothed conformal transducer Q to be *exactly valid* if the p-values
$Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), \tau)$ are uniformly distributed over $[0, 1]$, assuming that the examples $z_1, \ldots, z_{n-1}$ are generated by an exchangeable probability distribution on Z^{n-1}, and that $\tau \sim \mathcal{U}[0, 1]$ is sampled independently. Since every CPS is a smoothed conformal transducer, it follows that all CPSs are exactly valid. The validity property is achieved specifically through randomization, and the value of τ does not affect the p-values much.

2.2.1 Inductive Conformal Predictive Systems

In online settings, CPSs can be computationally intensive, particularly when dealing with large datasets. To address this, Vovk et al. (2022) introduced *inductive conformal predictive systems* (ICPSs). This enables the use of ICPSs within the CPDM framework. As is typical in the inductive setting, the training set $z_1, \ldots, z_{l-1}$ is split into a proper training set $z_1, \ldots, z_m$ and a calibration set $z_{m+1}, \ldots, z_{l-1}$.

An *inductive conformity measure* is defined as a measurable function:

$$A : Z^* \times Z \to \mathbb{R}, \tag{2.10}$$

where Z^* is the set of all finite sequences of elements of Z, that is invariant to permutations in its first argument. Given an inductive conformity measure A, an *inductive smoothed conformal transducer* $Q : Z^l \times [0, 1] \to [0, 1]$ is defined as:

$$\begin{aligned} Q(z_1, \ldots, z_{l-1}, (\boldsymbol{x}_l, y), \tau) := \frac{1}{l-m} \left|\{i = m+1, \ldots, l : \alpha_i < \alpha^y\}\right| \\ + \frac{\tau}{l-m} \left|\{i = m+1, \ldots, l : \alpha_i = \alpha^y\}\right| + \frac{\tau}{l-m} \end{aligned} \tag{2.11}$$

where $z_{m+1}, \ldots, z_{n-1} \in Z^{l-m-1}$ is a calibration set, $\boldsymbol{x}_l \in X$ is a test object, $\tau \in [0, 1]$ is a smoothing parameter, and α_i and α^y are the conformity scores defined by:

$$\begin{aligned} \alpha_i^y &:= A(z_1, \ldots, z_m, z_i), \quad i = m+1, \ldots, l-1, \\ \alpha^y &:= A(z_1, \ldots, z_m, (\boldsymbol{x}_l, y)), \end{aligned} \tag{2.12}$$

for each $y \in \mathbb{R}$.

Similar to a CPS, an ICPS is defined as a function that is an inductive smoothed conformal transducer, determined by some inductive conformity measure, and an RPS. Validity is automatically satisfied for an inductive smoothed conformal transducer, and means the p-values $Q(z_1, \ldots, z_{l-1}, (\boldsymbol{x}_l, y), \tau)$ are uniformly distributed over $[0, 1]$, provided the examples $z_1, \ldots, z_{l-1}, (\boldsymbol{x}_l, y)$ are generated by an exchangeable probability distribution on Z^l and $\tau \sim \mathcal{U}[0, 1]$ is generated independently of them.

2.3 Predictive Decision Making

Assume we are given a training set $z_1, \ldots, z_{n-1}$ of examples, where $z_i = (\boldsymbol{x}_i, y_i) \in Z$, and a finite set of available decisions D. We consider the decision problem of selecting the best decision $d \in D$ for a new object $\boldsymbol{x}_n \in X$, based on the information contained in the training set. Moreover, to guide the decision-making process, we assume the existence of a measurable *utility function* $U : Y \times D \to \mathbb{R}$, where $U(y, d)$ represents the von Neumann-Morgenstern utility associated with the decision $d \in D$ when the true label is y (von Neumann and Morgenstern, 1953).

The decision-making problem under consideration can be formalized using the concept of a *predictive decision-making system* (PDMS). Following Vovk and Bendtsen (2018), a PDMS is defined as a measurable function:

$$F : Z^{n-1} \times X \to D, \tag{2.13}$$

which recommends a decision d for a new test object $\boldsymbol{x}_n$ based on the training set $z_1, \ldots, z_{n-1}$. A PDMS may also be *randomized*, in which case it is defined as a measurable function $F : Z^{n-1} \times X \times [0, 1] \to D$ that takes an additional random number $\tau \in [0, 1]$ as input. The objective is to find a PDMS F that maximizes the expected utility; thus, the optimal PDMS F^* is given by:

$$F^* = \arg\max_F \mathbb{E}\Big[U(y, F(z_1, \ldots, z_{n-1}, \boldsymbol{x}_n, \tau))\Big], \tag{2.14}$$

where the expectation is taken with respect to the joint distribution (Murphy, 2023; Vovk and Bendtsen, 2018).

In Bayesian decision theory, upon observing a test object $\boldsymbol{x}_n \in X$, the training set $z_1, \ldots, z_{n-1}$ and the test object $\boldsymbol{x}_n$ are treated as fixed while the corresponding label y of $\boldsymbol{x}_n$ is modeled as a random variable distributed according to the conditional probability distribution $P(y \mid \boldsymbol{x}_n)$, which is assumed to exist (Murphy, 2023). The expected utility for selecting the decision $d \in D$ given $\boldsymbol{x}_n$ is defined as:

$$\mathbb{E}_{P(y|\boldsymbol{x}_n)}\Big[U(y, F(z_1, \ldots, z_{n-1}, \boldsymbol{x}_n, \tau))\Big] = \int U(y, F(z_1, \ldots, z_{n-1}, \boldsymbol{x}_n, \tau)) \times P(y \mid \boldsymbol{x}_n)\, dy \tag{2.15}$$

and the optimal decision $d^*(\boldsymbol{x}_n)$ for a test object $\boldsymbol{x}_n$ is given by:

$$d^*(\boldsymbol{x}_n) := \arg\max_{d \in D} \mathbb{E}_{P(y|\boldsymbol{x}_n)}\Big[U(y, F(z_1, \ldots, z_{n-1}, \boldsymbol{x}_n, \tau))\Big], \tag{2.16}$$

where $d = F(z_1, \ldots, z_{n-1}, \boldsymbol{x}_n, \tau)$.

Vovk and Bendtsen (2018) used *regret* to measure how much worse a PDMS performs compared to optimal decisions. Given a training set $z_1, \ldots, z_{n-1}$, and an object $\boldsymbol{x}_n$, the regret of a PDMS F under a probability distribution P on Z is defined as:

$$R_F(z_1, \ldots, z_{n-1}, \boldsymbol{x}_n, \tau) := \max_{d \in D} \int U(y, d)\, P(dy \mid \boldsymbol{x}_n) - \int U\Big(y, F(z_1, \ldots, z_{n-1}, \boldsymbol{x}_n, \tau)\Big)\, P(dy \mid \boldsymbol{x}_n). \tag{2.17}$$

Of particular interest are PDMSs that show small regret, as well as those that are *asymptotically efficient*, meaning that $R_F(z_1, \ldots, z_{n-1}, \boldsymbol{x}_n, \tau) \to 0$ as $n \to \infty$.

2.3.1 Conformal Predictive Decision Making

In developing CPDM, Vovk and Bendtsen (2018) adopted the Bayesian approach for statistical decision-making. However, instead of using Bayesian models to produce posterior distributions, they used CPSs to derive CPDs. Specifically, they focused on CPDs with thickness $1/n$ and exception size at most $n-1$, meaning Q_n changes by at most $1/n$ as τ varies over $[0, 1]$, apart from exception points.

Vovk and Bendtsen (2018) defined the integral:

$$\int U(y, d) Q_n(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), \tau)\, dy := \sum_{y: \Delta Q_n(y) \neq 0} U(y, d) \Delta Q_n(y), \tag{2.18}$$

for any utility function $U : \mathbb{R} \times D \to \mathbb{R}$, where $\Delta Q_n(y) := Q_n(y+) - Q_n(y-)$ denotes the increase in the value of Q_n at the point y. They showed that any CPD Q_n is a piecewise constant function with at most $2(n-1)$ discontinuities, ensuring the sum in (2.18) is finite. To ensure that different integrals of the type (2.18) are comparable, they also strengthened the condition **C2** by requiring that the function $Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), \tau)$ also satisfies the following limits:

$$\begin{aligned} \lim_{y \to -\infty} Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), 1) &= \frac{1}{n}, \\ \lim_{y \to \infty} Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y), 0) &= \frac{n-1}{n}, \end{aligned} \tag{2.19}$$

for all $(z_1, \ldots, z_{n-1}) \in Z^{n-1}$ and all $\boldsymbol{x}_n \in X$. These limits make the total mass of Q_n equal to $(n-1)/n < 1$. Therefore, the resulting integrals are not expectations in the strict sense. However, because the loss in total mass is consistent across different integrals, they remain comparable. Moreover, these additional limits are often satisfied by many conformity measures, for instance, the standard regression conformity measure $A(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y)) := y - \hat{y}$, where $\hat{y}$ is the point prediction for y.

2.4 Machine Learning Models

In this subsection, we provide a detailed overview of the ML models we evaluate in the thesis, as well as their conformalized counterparts when applicable.

2.4.1 Ridge Regression

Suppose we have an object space $X = \mathbb{R}^p$, where each object consists of p attributes. Let the label space be $Y = \mathbb{R}$, indicating a regression problem. Given a training set $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$, linear regression aims to fit a linear model of the form:

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + w_0, \tag{2.20}$$

where $\boldsymbol{w} \in \mathbb{R}^p$ is the vector of regression coefficients and w_0 is the intercept (bias) parameter (Murphy, 2022). The standard fitting method is *ordinary least squares* (OLS), which minimizes the sum of squared residuals. However, *ridge regression* (RR) introduces ℓ_2 regularization, which penalizes the magnitude of the regression coefficients (Murphy, 2022). This helps mitigate multicollinearity and reduce overfitting, leading to a more stable and robust model, especially in high-dimensional settings (Hastie et al., 2009). Nevertheless, this benefit comes with a cost: the coefficients become less interpretable. They are shrunk toward zero in a way that depends on the correlation structure among predictors, making it more difficult to isolate the effect of individual variables (Hastie et al., 2009).

The RR solution $\hat{\boldsymbol{w}}$ is obtained by solving the following optimization problem:

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i - w_0)^2 + \gamma \|\boldsymbol{w}\|^2, \tag{2.21}$$

where $\gamma > 0$ is the regularization parameter, often selected via cross-validation (Murphy, 2022). OLS is a special case of RR when $\gamma = 0$. Note that the intercept term w_0 is excluded from regularization to maintain translation invariance. If it were penalized, adding a constant c to all target values y_i would no longer shift the predictions by c, as the penalty would restrict the intercept from adjusting freely (Hastie et al., 2009).

To express the RR fitting procedure more compactly, we define the design matrix $\boldsymbol{X} := (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^\top \in \mathbb{R}^{n \times p}$ and the label vector $\boldsymbol{y} := (y_1, y_2, \ldots, y_n)^\top \in \mathbb{R}^n$. Assuming the features have been standardized, Hastie et al. (2009) estimates the intercept separately as:

$$w_0 = \frac{1}{n} \sum_{i=1}^{n} y_i. \tag{2.22}$$

Then the ridge objective becomes:

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \, \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \gamma \|\boldsymbol{w}\|^2. \tag{2.23}$$

Expanding the objective function gives:

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \, \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{w}^\top \boldsymbol{X}^\top \boldsymbol{y} + \boldsymbol{w}^\top (\boldsymbol{X}^\top \boldsymbol{X} + \gamma \boldsymbol{I}) \boldsymbol{w}, \tag{2.24}$$

where $I \in \mathbb{R}^{p \times p}$ is the identity matrix. Taking the gradient with respect to $\boldsymbol{w}$ and setting it to zero yields the closed-form ridge solution:

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^\top \boldsymbol{X} + \gamma \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}. \tag{2.25}$$

Predictions for a new object $\boldsymbol{x}$ are then given by:

$$\hat{\boldsymbol{y}} = \boldsymbol{x}^\top \hat{\boldsymbol{w}} = \boldsymbol{x}^\top (\boldsymbol{X}^\top \boldsymbol{X} + \gamma \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}. \tag{2.26}$$

For the training data, the fitted values $\hat{y}_i$ for $i = 1, 2, \ldots, n$ can be expressed compactly as:

$$\hat{\boldsymbol{y}} := \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X} + \gamma \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y} = \boldsymbol{H}\boldsymbol{y}, \tag{2.27}$$

where $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is known as the *hat matrix*, since it "puts the hat" on $\boldsymbol{y}$ to yield the fitted values (Hastie et al., 2009).

The expression in (2.25) is the regularized counterpart to the *normal equations* obtained from OLS. While they provide a closed-form solution, RR is typically solved using more numerically stable methods such as *singular value decomposition* (SVD), as explicitly computing the matrix inverse can be expensive and unstable for large or ill-conditioned matrices (Murphy, 2022).

2.4.2 Least Squares Prediction Machine

The *Least Square Prediction Machine* (LSPM) is a CPS that has either OLS or RR as its underlying learning algorithm (Vovk et al., 2022). For simplicity, we present the derivation assuming OLS is used. In this case, the hat matrix simplifies to:

$$\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top. \tag{2.28}$$

Nevertheless, all results in this subsection also apply when RR is used instead.

Vovk et al. (2022) defined the LSPM based on regression residuals and introduced three variants: *ordinary*, *deleted*, and *studentized*. Among these, the studentized is the most useful, as it forms an RPS without requiring any assumptions about the influence of individual examples. In contrast, the ordinary and the deleted variants are not RPSs in general, as they may fail to produce CPDs that are monotonically increasing in y, unless high-leverage points are present.

We will refer to the studentized version simply as LSPM. Vovk et al. (2022) defined it using the studentized conformity measure:

$$A(\{z_1, \ldots, z_n\}, z_n) := \frac{y_n - \hat{y}_n}{\sqrt{1 - h_{n,n}}}, \tag{2.29}$$

where $\hat{y}_n$ is the prediction for y_n mad by the underlying regression algorithm, and $h_{n,n}$ is the nth diagonal element of the hat matrix corresponding to the examples $z_1, \ldots, z_n$. The authors showed that the CPD Q_n produced by the LSPM is monotonically increasing in y provided that $\max_{i=1,\ldots,n} h_{i,i} < 1$. This condition means that no training point is excessively influential. According to Vovk et al. (2022), it typically holds in practice, ensuring that the LSPM is an RPS and therefore also a CPS. If the condition is violated, some conformity scores become undefined, rendering the method inapplicable.

Vovk et al. (2022) represents the CPDs as intervals:

$$Q_n(y) := [Q_n(y, 0), Q_n(y, 1)], \tag{2.30}$$

mapping each potential label $y \in \mathbb{R}$ to a closed interval on $\mathbb{R}$. This formulation retains the same information as the original definition in (2.6), since any intermediate value $Q_n(y, \tau)$ can be recovered via a convex combination:

$$Q_n(y, \tau) = (1 - \tau)Q_n(y, 0) + \tau Q_n(y, 1). \tag{2.31}$$

To compute the CPDs produced by the LSPM, p-values are determined using the definition of a conformal transducer described (2.11). This involves solving the equality $\alpha_i^y = \alpha_n^y$ and inequality $\alpha_i^y < \alpha_n^y$, for $i = 1, 2, \ldots, n-1$. Vovk et al. (2022) showed that the difference $\alpha_n^y - \alpha_i^y$ can be expressed as:

$$\alpha_n^y - \alpha_i^y = \frac{y - \hat{y}_n}{\sqrt{1 - h_{n,n}}} - \frac{y_i - \hat{y}_i}{\sqrt{1 - h_{i,i}}} \tag{2.32}$$

$$= \frac{y - \sum_{j=1}^{n-1} h_{j,n} y_j - h_{n,n} y}{\sqrt{1 - h_{n,n}}} - \frac{y_i - \sum_{j=1}^{n-1} h_{i,j} y_j - h_{i,n} y}{\sqrt{1 - h_{i,i}}} \tag{2.33}$$

$$= \left(\sqrt{1 - h_{n,n}} + \frac{h_{i,n}}{\sqrt{1 - h_{i,i}}} \right) y - \left(\frac{\sum_{j=1}^{n-1} h_{j,n} y_j}{\sqrt{1 - h_{n,n}}} + \frac{y_i - \sum_{j=1}^{n-1} h_{i,j} y_j}{\sqrt{1 - h_{i,i}}} \right) \tag{2.34}$$

$$= B_i y - A_i, \tag{2.35}$$

where y is the label of the nth object x_n. The solutions to the equations $\alpha_i^y = \alpha_n^y$ are then given by $C_i := A_i / B_i,\ i = 1, \ldots, n-1$. As y increases, the CPD changes in value when y crosses one of these thresholds C_i. Sorting the values $C_1, \ldots, C_{n-1}$ in ascending order yields $C_{(1)} \leq \cdots \leq C_{(n-1)}$, and we define $C_{(0)} := -\infty$ and $C_{(n)} := \infty$. The CPD can then be expressed as:

$$Q_n(y) := \begin{cases} [\frac{i}{n}, \frac{i+1}{n}] & \text{if } y \in (C_{(i)}, C_{(i+1)}) \text{ for } i \in \{0, 1, \ldots, n-1\} \\ [\frac{i'+1}{n}, \frac{i''+1}{n}] & \text{if } y = C_{(i)} \text{ for } i \in \{1, \ldots, n-1\}, \end{cases} \tag{2.36}$$

where $i' := \min\{j : C_{(j)} = C_{(i)}\}$ and $i'' := \max\{j : C_{(j)} = C_{(i)}\}$. This CPD has thickness $1/n$ and an exception size of $n-1$, corresponding to the number of distinct C_i.

The complete procedure is summarized in Algorithm 1. We assume that A_i and B_i are defined for all $i = 1, \ldots, n-1$, and that all $B_i > 0$. These assumptions are almost automatically satisfied. Specifically, the quantities are defined as long as the denominators are non-zero, which occurs when $h_{i,i} < 1$ for all $i = 1, \ldots, n$. This condition coincides with the requirement that all conformity scores are well-defined on the augmented training dataset $z_1, \ldots, z_n$.

Algorithm 1: LSPM

Input: A training set $(\boldsymbol{x}_i, y_i) \in X \times \mathbb{R},\ i = 1, \ldots, n-1$, and a test object $\boldsymbol{x}_n \in X$

Set $\boldsymbol{X}$ to be the design matrix for the given n objects

Define the hat matrix $\boldsymbol{H}$ by (2.28)

for $i \in \{1, \ldots n-1\}$ **do**

  Define A_i and B_i as in (2.34)

  Set $C_i := A_i / B_i$

Obtain $C_{(1)}, \ldots, C_{(n-1)}$ by sorting $C_1, \ldots, C_{n-1}$ in ascending order

Set $C_{(0)} := -\infty$ and $C_{(n)} := \infty$

Output: The predictive distribution Q_n for y_n according to (2.36)

2.4.3 Kernel Ridge Regression

Kernel Ridge Regression (KRR) is the kernelized version of RR, utilizing the "kernel trick", meaning inner products are replaced with a *kernel function* (Vovk et al., 2022). In KRR, inputs from the object space X are implicitly mapped to a *Hilbert space* $\mathcal{H}$, often referred to as the *feature space*, via a kernel function, where RR is subsequently performed (Vovk et al., 2022). Since $\mathcal{H}$ is typically high-dimensional, and possibly infinite-dimensional, it allows the model to learn linear patterns in $\mathcal{H}$ that correspond to nonlinear relationships in X (Murphy, 2022).

Kernel functions define similarity between objects in kernel methods. Following, Murphy (2022), we consider *Mercer kernels*, which are symmetric functions $\mathcal{K} : X \times X \to \mathbb{R}^+$ satisfying:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) c_i c_j \geq 0, \tag{2.37}$$

for any training set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in X$, and any real coefficients $c_1, \ldots, c_n \in \mathbb{R}$. We further assume that $\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j) > 0$ to ensure that the equality in (2.37) hold only when $c_i = 0$ for all $i = 1, \ldots, n$.

As detailed in Murphy (2022), another way to represent a Mercer kernel is through its associated *Gram matrix*, often called the *kernel matrix*. Given a training set $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, the kernel matrix is defined as:

$$\boldsymbol{K} = \begin{bmatrix} \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & \mathcal{K}(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_1) & \ldots & \mathcal{K}(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}, \tag{2.38}$$

and the function $\mathcal{K}$ is a Mercer kernel if and only if its corresponding kernel matrix $\boldsymbol{K}$ is positive definite for any training set.

The most widely used kernel is the *radial basis function* (RBF) kernel, which measures similarity between two objects $\boldsymbol{x}, \boldsymbol{x}'$ using scaled Euclidean distance (Murphy, 2022). It is defined as:

$$\mathcal{K}(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right), \tag{2.39}$$

where ℓ is the length scale hyperparameter which controls the smoothness of the resulting function (Duvenaud, 2014). Specifically, it determines the distance over which differences between objects are expected to have a significant effect (Murphy, 2022).

Murphy (2022) describes that any positive definite matrix $\boldsymbol{K}$ admits an eigendecomposition of the form $\boldsymbol{K} = \boldsymbol{U}^\top \boldsymbol{\Lambda} \boldsymbol{U}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix of strictly positive eigenvalues $\lambda_i > 0$, and $\boldsymbol{U}$ is a matrix whose columns are the eigenvectors of $\boldsymbol{K}$. The elements of $\boldsymbol{K}$ can then be expressed as:

$$k_{i,j} = (\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{U}_{:i})^\top(\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{U}_{:j}), \tag{2.40}$$

where $\boldsymbol{\Lambda}^{\frac{1}{2}}$ denotes the diagonal matrix formed by takin the square root of each entry in $\boldsymbol{\Lambda}$, and $\boldsymbol{U}_{:i}$ denotes the ith column of $\boldsymbol{U}$. If we define the feature map $\phi(\boldsymbol{x}_i) = \boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{U}_{:i}$, the kernel elements can be expressed as:

$$k_{i,j} = \phi(\boldsymbol{x}_i)^\top\phi(\boldsymbol{x}_j) = \sum_m \phi_m(\boldsymbol{x}_i)\phi_m(\boldsymbol{x}_j), \tag{2.41}$$

where the sum runs over the components m of the feature vector $\phi(\boldsymbol{x}_i)$. In other words, the entries of the kernel matrix $\boldsymbol{K}$ correspond to inner products between feature vectors, implicitly defined by the eigenvectors of $\boldsymbol{K}$, in the feature space. This is an important result that is generalized to apply to kernel functions in *Mercers theorem*. Specifically, it allows us to compute inner products in feature spaces without explicitly mapping data into them. However, to apply the kernel trick, the model must be formulated entirely in terms of inner products between objects.

As presented in Vovk et al. (2022), the primal form of RR given in (2.26) is not formulated in terms of inner products between objects. However, this can be achieved by reformulating the procedure in its *dual form*. The "dualization" of RR can be concisely derived via the traditional statistical approach, utilizing the matrix equality:

$$\boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X} + \gamma\boldsymbol{I})^{-1} = (\boldsymbol{X}\boldsymbol{X}^\top + \gamma\boldsymbol{I})^{-1}\boldsymbol{X} \tag{2.42}$$

which can be easily verified by multiplying the left-hand side by $(\boldsymbol{X}\boldsymbol{X}^\top + \lambda\boldsymbol{I})$ and the right-hand side by $(\boldsymbol{X}^\top\boldsymbol{X} + \lambda\boldsymbol{I})$ (Vovk et al., 2022). Using the identity (2.42), the prediction of RR for an new object $\boldsymbol{x}$ can rewritten as:

$$\hat{y} = \boldsymbol{X}(\boldsymbol{X}^\top\boldsymbol{X} + \gamma\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y} = \boldsymbol{y}^\top(\boldsymbol{X}\boldsymbol{X}^\top + \gamma\boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{x}. \tag{2.43}$$

The new representation now only depends on the objects via the inner products between them. As shown by Vovk et al. (2022), this makes it possible to apply the kernel trick, allowing us to replace these inner products with evaluations of a kernel function. As a result, the prediction rule in (2.43) can be expressed as:

$$\hat{y} = \boldsymbol{y}^\top(\boldsymbol{K} + \gamma\boldsymbol{I})^{-1}\boldsymbol{k}, \tag{2.44}$$

where $\boldsymbol{K}$ is the kernel matrix, and $\boldsymbol{k} := (\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_i), \ldots, \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}_n))^\top$ is the vector of kernel evaluations between the test object $\boldsymbol{x}$ and the training inputs. This is the prediction rule for KRR, and if the RR is performed in the feature space, the corresponding hat matrix is given by:

$$\boldsymbol{H} := (\boldsymbol{K} + \gamma\boldsymbol{I})^{-1}\boldsymbol{K}. \tag{2.45}$$

2.4.4 Kernel Ridge Regression Prediction Machine

The *Kernel Ridge Regression Prediction Machine* (KRRPM) is a CPS that employs KRR as its underlying algorithm. As with the LSPM, it has three variants, but we focus on the studentized version, using the studentized conformity measure. Vovk et al. (2022) showed that KRRPM is an RPS, and hence a CPS, with the studentized conformity measure always being well-defined. Following the same convention as for LSPM, we will simply refer to the studentized KRRPM as KRRPM.

In our KRRPM derivation, we will not restrict ourselves to the case when $\gamma = 0$, since regularization is typically crucial in nonlinear settings. Vovk et al. (2022) describe that we need to solve the equations $\alpha_i^y = \alpha_n^y$, and $\alpha_i^y < \alpha_n^y$ for $i = 1, \ldots, n-1$ to obtain the p-values defined in (2.11), as we did for LSPM. This is done in the exact same way for KRRPM, as for LSPM, with the only difference being the definition of the hat matrix, which now should be of in the form given in (2.45). Although this definition is slightly more complex, we are in this case guaranteed to always have that $B_i > 0$ for all $i = 1, \ldots, n-1$. Apart from the difference in the hat matrix, the procedure is the same, and it is summarized in Algorithm 2.

Algorithm 2: KRRPM

Input: A training set $(\boldsymbol{x}_i, y_i) \in X \times \mathbb{R},\ i = 1, \ldots, n-1$, and a test object $\boldsymbol{x}_n \in X$

Set $\boldsymbol{X}$ to be the design matrix for the given n objects
Define the hat matrix $\boldsymbol{H}$ by (2.45)
Define the kernel matrix $\boldsymbol{K}$
for $i \in \{1, \ldots n-1\}$ **do**
    Define A_i and B_i as in (2.34)
    Set $C_i := A_i / B_i$
Obtain $C_{(1)}, \ldots, C_{(n-1)}$ by sorting $C_1, \ldots, C_{n-1}$ in ascending order
Set $C_{(0)} := -\infty$ and $C_{(n)} := \infty$

Output: The predictive distribution Q_n for y_n according to (2.36)

While Algorithm 2 allows us to obtain a nonlinear objective function, it is significantly more computationally expensive than LSPM, particularly for large datasets. This inefficiency primarily arises from the need to compute the hat matrix $\boldsymbol{H}$ for each new test object. Vovk et al. (2022) propose a method to improve the computational efficiency of Algorithm 2. However, we omit it here for brevity and because our main focus is on the overall understanding of the conformalization of KRR.

Another limitation of KRRPM, as noted by Vovk et al. (2022), is that its CPDs are not sensitive to the test object $\boldsymbol{x}_n$. Although different test objects produce CPDs located at different points, the shapes of these CPDs remain similar to the empirical distribution of residuals $y_i - \hat{y}_i$ of the training set. Thus, the shape of the CPD does not adapt to the specific test object $\boldsymbol{x}_n$.

2.4.5 K-Nearest Neighbors Regression

The material in this subsection is based on Vovk et al. (2022); Murphy (2022), as well as our own notation to make the definitions more precise.

K-Nearest Neighbors Regression (KNN) is one of the simplest regression algorithms available. It is a nonparametric, example-based method that makes predictions based on the similarity between data points. The main hyperparameter of KNN is $k \in \mathbb{N}$, which specifies how many of the examples $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_{i-1}, y_{i-1}), (\boldsymbol{x}_{i+1}, y_{i+1}), \ldots, (\boldsymbol{x}_n, y_n)$ are to be considered part of the "neighborhood" N_{k_i} for an example $(\boldsymbol{x}_i, y_i)$. Formally, the neighborhood N_{k_i} is defined as:

$$N_{k_i} := \{(\boldsymbol{x}_{\pi(1)}, y_{\pi(1)}), \ldots (\boldsymbol{x}_{\pi(k)}, y_{\pi(k)})\}, \tag{2.46}$$

where π is a permutation of $\{1, \ldots, n\}$ such that

$$d(\boldsymbol{x}_i, \boldsymbol{x}_{\pi(1)}) \leq \cdots \leq d(\boldsymbol{x}_i, \boldsymbol{x}_{\pi(n)}), \tag{2.47}$$

and d is a distance metric, typically the Euclidean distance. For simplicity, we assume that all distances are distinct to avoid ties in the neighborhood definition. The points included in N_{k_i} are the so-called "neighbors" of $\boldsymbol{x}_i$.

For regression tasks, the prediction is commonly made by averaging the target values of the neighbors, that is,

$$\hat{y}_i = \frac{1}{k} \sum_{j=1}^{k} y_{\pi(j)} \tag{2.48}$$

Alternatively, one may use the median of the neighbor values instead of the mean, which results in a more robust predictor that is less sensitive to outliers.

2.4.6 Nearest Neighbors Prediction Machine

The *Nearest Neighbors Prediction Machine* (NNPM) is a CPS that utilizes KNN as its underlying algorithm. Unlike KRRPM, NNPM adapts the shape of its predictive distribution based on the test object (Vovk et al., 2022). However, the resolution of its CPDs will depend on the number of neighbors k. Thus, we are mainly interested in cases where k is fairly large; otherwise, the CPDs will become very crude, characterized by a few large jumps.

The derivation of the algorithm for NNPM that we will present in this subsection will differ in notation compared to Vovk et al. (2022). This is because we want to be consistent with the notation in the previous subsection covering KNN. However, the algorithm we present is the same.

For the derivation of the algorithm for NNPM, Vovk et al. (2022) assumed that the object space X is a *metric space*, i.e., a space with a defined notion of distance

between points, e.g., Euclidean distance. Moreover, to avoid distractions, they assumed that all distances between objects $\boldsymbol{x}_i \in X$ are different and have distinct labels $y_i \in Y$.

Assuming a fixed $k \in \mathbb{N}$, Vovk et al. (2022) defined the following *deterministic predictive system* (DPS):

$$Q(z_1, \ldots, z_{n-1}, (\boldsymbol{x}_n, y)) := \frac{|\{(\boldsymbol{x}_j, y_j) \in N_{k_n} : y_j \leq y\}|}{k}, \tag{2.49}$$

where no τ is used for randomization and assuming that $n - 1 \geq k$ to ensure that the neighborhood N_{k_n} is defined. To conformalize this DPS, the authors used the following conformity measure:

$$A(\wr z_1, \ldots, z_n \int, z_i) := |\{(\boldsymbol{x}_j, y_j) \in N_{k_i} : y_j \leq y_i\}|. \tag{2.50}$$

In addition, they asserted that this conformity measure is monotonic with a minimum of 0 and a maximum of k when y varies. Vovk et al. (2022) subsequently showed that by a corollary they previously defined, this conformity measure produces a CPS. Since the corollary involves a lot of specific notation we do not use anywhere else and relies on another proof, we keep it short here and refer to the book for the details.

Vovk et al. (2022) classify the examples in the training set $z_1, \ldots, z_{n-1}$ in the following way:

- *Full neighbor*: An example z_i is considered a full neighbor if $\boldsymbol{x}_i$ is among the k nearest neighbors of $\boldsymbol{x}_n$, and $\boldsymbol{x}_n$ is among the k nearest neighbors of $\boldsymbol{x}_i$.
- *Single neighbor*: An example z_i is considered a single neighbor if $\boldsymbol{x}_i$ is among the k nearest neighbors of $\boldsymbol{x}_n$, but $\boldsymbol{x}_n$ is not among the k nearest neighbors of $\boldsymbol{x}_i$.
- *Semi-neighbor*: An example z_i is considered a single neighbor if $\boldsymbol{x}_i$ is not among the k nearest neighbors of $\boldsymbol{x}_n$, but $\boldsymbol{x}_n$ is among the k nearest neighbors of $\boldsymbol{x}_i$.
- *Non-neighbor*: An example z_i is a non-neighbor if it is neither a full neighbor, a single neighbor, nor a semi-neighbor.

In this classification context, the term neighbor refers to either a full neighbor or a single neighbor, and is thus used synonymously with being one of the k nearest neighbors of $\boldsymbol{x}_n$.

In the NNPM algorithm, Vovk et al. (2022) begin by defining N_{K_n}, which represent the set of all neighbors and semi-neighbors of $\boldsymbol{x}_n$ in the training set. They then proceed to sort the labels of the examples $(\boldsymbol{x}_l, y_l) \in N_{K_n}$ in ascending order: $y_{\pi(1)} \leq \cdots \leq y_{\pi(K)}$, and set $y_{\pi(0)} := -\infty$ and $y_{\pi(K+1)} := \infty$.

Next, Vovk et al. (2022) define the arrays α and H. An element α_i of the array α is the conformity score of example z_i with respect to the augmented training

set $z_1, \ldots, z_n$, where $z_n := (\boldsymbol{x}_n, y)$. Initially, the value of y is set to $-\infty$, and the conformity scores α_i are initialized as:

$$\alpha_i := |\{(\boldsymbol{x}_j, y_j) \in N_{k_i} : y_j \leq y_i\}|. \tag{2.51}$$

As the value of y increases toward ∞, the conformity scores α_i are updated accordingly. The array H is then defined as the histogram of the values in α, with:

$$H_l := |\{i \in \{\ 1, \ldots, n\} : \alpha_i = l\}|, \tag{2.52}$$

where $l \in \{1, \ldots, K\}$. The histogram H is updated after the conformity scores of α are updated.

Following this, Vovk et al. (2022) defines the values $L_0 := 0$ and $U_0 := H_0/n$. These values are used in the predictive distribution function, see (2.53). Then for all number of neighbors and semi-neighbors of $\boldsymbol{x}_n$, the arrays α and H are updated, and the values L_l and U_l computed, see Algorithm 3 for the details. Finally, a CPD of the form:

$$Q_n(y) := \begin{cases} [L_l, U_l] & \text{if } y \in (y_{\pi(l)}, y_{\pi(l+1)}) \text{ for } l \in \{0, 1, \ldots, K\}, \\ [L_{l-1}, U_l] & \text{if } y = y_{\pi(l)} \text{ for } l \in \{1, \ldots, K\} \end{cases} \tag{2.53}$$

is returned. The full procedure is summarized in Algorithm 3.

Algorithm 3: NNPM

Input: A training set $(\boldsymbol{x}_i, y_i) \in X \times \mathbb{R}$, $i = 1, \ldots, n-1$, and a test object $\boldsymbol{x}_n \in X$

Initialize N_{K_n} to the set of total number of neighbors and semi-neighbors
Sort the labels of the examples $(\boldsymbol{x}_l, y_l) \in N_{K_n}$ in ascending order, obtaining $y_{\pi(1)} \leq \cdots \leq y_{\pi(K)}$
Set $y_{\pi(0)} := -\infty$ and $y_{\pi(K+1)} := \infty$
Initialize the arrays α and H according to (2.51) and (2.52), respectively
Set $L_0 := 0$ and $U_0 := H_0/n$
for $l \in \{1, \ldots K\}$ **do**
    if $z_{\pi(l)}$ *is a neighbor* **then**
        H_{α_n} -= 1; α_n += 1; H_{α_n} += 1;
    if $z_{\pi(l)}$ *is a full neighbor or semi-neighbor* **then**
        $H_{\alpha_{\pi(l)}}$ -= 1; $\alpha_{\pi(l)}$ -= 1; $H_{\alpha_{\pi(l)}}$ += 1;
    Set $L_l := \sum_{l=0}^{\alpha-1} H_l/n$ and $U_l := \sum_{l=0}^{\alpha_n} H_l/n$

Output: The predictive distribution Q_n for y_n according to (2.53)

2.4.7 Bayesian Ridge Regression

Bayesian Ridge Regression (BRR) extends RR by treating model parameters probabilistically, with regularization parameters considered as random variables (Bishop, 2006). This results in a fully Bayesian formulation of linear regression that adapts to the data and enables UQ through the posterior distribution over model parameters.

In BRR, the regression weights, $\boldsymbol{w}$, are assumed to follow the distribution:

$$P(\boldsymbol{w} \mid \lambda) = \mathcal{N}(\mathbf{0}, \lambda^{-1}\boldsymbol{I}), \tag{2.54}$$

where λ is the prior precision on the weights.

The likelihood function for the observed targets $\boldsymbol{y}$ is given by:

$$P(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}, \beta) = \mathcal{N}(\boldsymbol{X}\boldsymbol{w}, \beta^{-1}\boldsymbol{I}), \tag{2.55}$$

where β is the precision of the Gaussian observation noise, assumed to be IID across data points (Bishop, 2006).

Given the Gaussian prior on $\boldsymbol{w}$ and the Gaussian likelihood, it can be shown that the posterior distribution over $\boldsymbol{w}$ is also Gaussian and is written as:

$$P(\boldsymbol{w} \mid \boldsymbol{y}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{2.56}$$

where the posterior mean and covariance are given as:

$$\boldsymbol{\Sigma}^{-1} = \lambda\boldsymbol{I} + \beta\boldsymbol{X}^\top\boldsymbol{X}, \tag{2.57}$$

$$\boldsymbol{\mu}_n = \beta\boldsymbol{\Sigma}\boldsymbol{X}^\top\boldsymbol{y}. \tag{2.58}$$

From the posterior, the resulting log-posterior, up to an additive constant, takes the form:

$$\log P(\boldsymbol{w} \mid \boldsymbol{y}) = -\frac{\beta}{2}\sum_{i=1}^{n}\left(y_i - \boldsymbol{w}^\top\boldsymbol{x}_i\right)^2 - \frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{w} + C \tag{2.59}$$

which shows that the Maximum A Posteriori (MAP) estimate of $\boldsymbol{w}$ corresponds to minimizing a penalized sum-of-squares error function (Bishop, 2006). This is equivalent to the ridge regression objective, where the regularization strength is given by $\gamma = \beta/\lambda$. Hence, RR can be interpreted as a special case of Bayesian inference where the prior precision λ and noise precision β are fixed and the posterior mean is used for prediction. In contrast, BRR infers these parameters and the full posterior, allowing for UQ.

In reality, we are more interested in predicting new target values than in the posterior distribution of the weights themselves (Bishop, 2006). To make predictions for a new

input $\boldsymbol{x}$, we compute the posterior predictive distribution by marginalizing over the posterior distribution of $\boldsymbol{w}$:

$$P(y \mid \boldsymbol{x}, \boldsymbol{X}, \boldsymbol{y}) = \int P(y \mid \boldsymbol{x}, \boldsymbol{w})\, P(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y})\, d\boldsymbol{w}. \tag{2.60}$$

Since both the likelihood $P(y \mid \boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(y \mid \boldsymbol{w}^\top \boldsymbol{x}, \beta^{-1})$ and the posterior $P(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{w} \mid \mu, \boldsymbol{\Sigma})$ are Gaussian, the result of this convolution is also Gaussian. The predictive distribution takes the form:

$$P(y \mid \boldsymbol{x}, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(y \mid \boldsymbol{x}^\top \mu, \sigma^2(\boldsymbol{x})), \tag{2.61}$$

where the predictive variance is given by:

$$\sigma^2(\boldsymbol{x}) = \frac{1}{\beta} + \boldsymbol{x}^\top \boldsymbol{\Sigma} \boldsymbol{x}. \tag{2.62}$$

The first term $1/\beta$ reflects the observation noise, while the second term accounts for model uncertainty through the posterior covariance $\boldsymbol{\Sigma}$ (Bishop, 2006).

In practice, the predictive mean $\boldsymbol{x}^\top \mu$ is often used directly as the point prediction. Since the predictive distribution is Gaussian and thus symmetric, the mean coincides with the median. Therefore, when we are interested in the integral over the cumulative predictive distribution, such as estimating the probability that y exceeds a threshold, the mean prediction is sufficient for inference, as it provides the same decision boundary as integrating over the full distribution.

Unlike RR, where the regularization strength γ is chosen manually or via cross-validation, BRR uses *empirical Bayes* or *type 2 maximum likelihood* to estimate β and λ from the data (Bishop, 2006). Tipping (2001) explains that this involves placing *Gamma priors* over the two parameters to ensure conjugacy:

$$\beta \sim \text{Gamma}(\beta_1, \beta_2), \tag{2.63}$$
$$\lambda \sim \text{Gamma}(\lambda_1, \lambda_2). \tag{2.64}$$

Under this hierarchical Bayesian formulation, the marginal likelihood is maximized with respect to $\log \beta$ and $\log \lambda$, which is convenient when assuming uniform hyperpriors on a logarithmic scale. This transforms the inference into an optimization problem over log-hyperparameters, simplifying the treatment of prior terms, which tend to vanish under flat priors (Tipping, 2001).

The resulting objective, ignoring terms independent of β and λ, is:

$$\begin{aligned} -\frac{1}{2}\Big[\log\left|\lambda^{-1}\boldsymbol{I} + \boldsymbol{X}^\top \boldsymbol{X}\right| + \boldsymbol{y}^\top \left(\lambda^{-1}\boldsymbol{I} + \boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{y} \Big] \\ + \sum_{i=0}^{n} (a \log \beta_i - b\beta_i) + c \log \lambda - d\lambda. \end{aligned} \tag{2.65}$$

The last three terms represent the additive prior terms for β and λ. When using Gamma priors, these can be retained to incorporate prior beliefs, but in practice they are often omitted by setting the Gamma shape and scale parameters to zero, i.e, adopting non-informative priors (Tipping, 2001).

The learning algorithm proceeds by iteratively updating β and λ to maximize this marginal likelihood. A more detailed derivation of the marginal likelihood expression and the strategy for the update rules is provided in Tipping (2001) Appendix A.

2.4.8 Gaussian Processes Regression

Gaussian Process Regression (GPR) is a non-parametric Bayesian approach to regression that models a distribution over functions directly (Rasmussen and Williams, 2006). Instead of assuming a fixed parametric form, GPR defines a prior over latent functions $f(\boldsymbol{x})$, where the function values are assumed to be jointly Gaussian. This prior is fully specified by a mean function $m(\boldsymbol{x})$ and a kernel $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$:

$$m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})], \tag{2.66}$$

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x}') - m(\boldsymbol{x}'))]. \tag{2.67}$$

The Gaussian process prior is written as:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')), \tag{2.68}$$

which implies that for any finite set of inputs, the corresponding function values follow a multivariate Gaussian distribution (Rasmussen and Williams, 2006). The function $f(\boldsymbol{x})$ represents the underlying latent function that we wish to learn and is assumed to generate the observed targets via:

$$y = f(\boldsymbol{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2), \tag{2.69}$$

where ε is IID Gaussian noise.

Given the training data $\boldsymbol{X}, \boldsymbol{y}$, the model is trained by maximizing the log marginal likelihood with respect to the specific kernel parameters $\boldsymbol{\theta}$:

$$\log p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^\top \boldsymbol{K}_{\boldsymbol{\theta}}^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}_{\boldsymbol{\theta}}| - \frac{n}{2}\log 2\pi, \tag{2.70}$$

where $\boldsymbol{K}_{\boldsymbol{\theta}} = \mathcal{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2\boldsymbol{I}$ is the kernel matrix evaluated on the training inputs, including the noise term (Rasmussen and Williams, 2006).

For a test input $\boldsymbol{x}$, the predictive distribution over the corresponding target value y is Gaussian:

$$P(y \mid \boldsymbol{x}, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(\mu, \sigma^2), \tag{2.71}$$

with

$$\mu = \boldsymbol{k}^\top \boldsymbol{K}^{-1} \boldsymbol{y}, \tag{2.72}$$

$$\sigma^2 = \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}^\top \boldsymbol{K}^{-1} \boldsymbol{k} + \sigma^2, \tag{2.73}$$

where $\boldsymbol{k} = k(\boldsymbol{X}, \boldsymbol{x})$ is the covariance vector between the training inputs and the test input (Rasmussen and Williams, 2006).

This formulation yields both a point prediction and an uncertainty estimate. Similarly to BRR, the predictive mean can be used directly in the application for this thesis, as the predictive distribution is symmetric, and therefore the mean coincides with the median.

3

Methodology

This section outlines the methodology used for the experiments in this thesis. We describe the datasets and preprocessing steps, the PDMSs under evaluation, the ML models and selection process, the applied utility functions, the evaluation metrics, and finally, the different experimental configurations. All the code for the experiments is available on the GitHub page[1] associated with this thesis.

3.1 Data and Preproccessing

We generated synthetic datasets using a linear model generator and a set of nonlinear benchmark functions originally proposed by Friedman (1991), all of which are available via `scikit-learn` (Pedregosa et al., 2011). This setup supports evaluation in both linear contexts and more complex scenarios involving nonlinearities and feature interactions, thus capturing a broad spectrum of real-world modeling challenges. A detailed description of the data-generating procedures is provided in the list below, where x_i represents the i:th feature, y the target variable, and ϵ the noise.

1. **Linear Dataset (`make_regression`)**

$$x_i \sim \mathcal{N}(0,1), \quad i = 1, \ldots, 5$$

$$y = \boldsymbol{w}^\top \boldsymbol{x} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

2. **Friedman #1 Dataset (`make_friedman1`)**

$$x_i \sim \mathcal{U}(0,1), \quad i = 1, \ldots, 5$$

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

3. **Friedman #2 Dataset (`make_friedman2`)**

$$x_1 \sim \mathcal{U}(0,100), \quad x_2 \sim \mathcal{U}(40\pi, 560\pi), \quad x_3 \sim \mathcal{U}(0,1), \quad x_4 \sim \mathcal{U}(1,11)$$

$$y = \left(x_1^2 + \left(x_2 x_3 - \frac{1}{x_2 x_4}\right)^2\right)^{1/2} + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

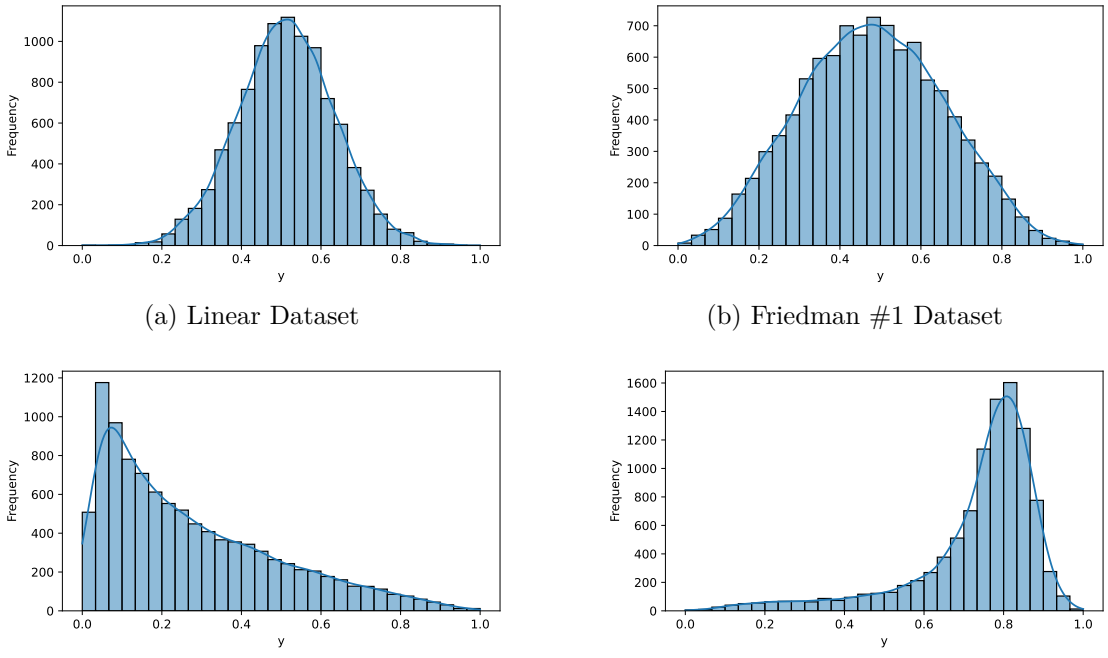
[1] https://github.com/simonlanngren/Conformal-Predictive-Decision-Making

4. **Friedman #3 Dataset (`make_friedman3`)**

$$y = \arctan\left(\frac{x_2 x_3 - \frac{1}{x_2 x_4}}{x_1}\right) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

The input features follow the same distributions as in Friedman #2.

An initial pool of 10,000 data points was generated for each data-generating process. The target variable in each dataset was scaled to the $[0, 1]$ range using min-max normalization. The target distributions after scaling are shown in Figure 3.1. Bootstrap samples were then drawn from these pools to construct the datasets used in each experimental run. A total of 100 bootstrap runs were performed for each inductive experiment and for the online experiments involving the linear dataset and linear models. The remaining experiments were carried out using 50 runs to maintain computational feasibility.



(a) Linear Dataset

(b) Friedman #1 Dataset

(c) Friedman #2 Dataset

(d) Friedman #3 Dataset

Figure 3.1: The target distributions of the four datasets used in this study.

The number of examples in each bootstrap sample varied between two configurations: a standard setting and a reduced-data setting. Each dataset, regardless of its size, was split in the same way into training and test sets. In the inductive setting, the training set was further divided into a proper training set and a calibration set, which was used to calibrate the ICPSs. The two data configurations are summarized in Table 3.1. Additionally, to evaluate robustness to noise in the data, the noise level σ^2 was varied across two settings, as detailed in Table 3.2.

After data generation and bootstrap sampling for each run, an additional preprocessing step was performed by standard scaling all the features. To avoid data leakage in

the inductive setting, the scaler was fitted only on the proper training set, whereas in the online setting, it was fitted on the entire training set.

Setting	Train	Calibration	Test
Standard Data Configuration			
Online	25	—	75
Inductive	160	40	50
Limited Data Configuration			
Online	7	—	43
Inductive	64	16	20

Table 3.1: An overview of data splits used in the online and inductive experimental settings under different configurations.

Configuration	Linear	Friedman #1	Friedman #2	Friedman #3
Standard Noise	0.1	0.1	0.1	0.1
Noisy Configuration	10	5	100	0.5

Table 3.2: The noise levels σ^2 used across datasets under standard and noisy configurations.

3.2 Predictive Decision-Making Systems

This section outlines the PDMSs used for each of the approaches under consideration, in both the online and the inductive setting. The online PDMSs involving CPSs were implemented using the newly developed `online-cp` package (Szabadváry et al., 2025). The inductive PDMSs involving ICPSs, on the other hand, were implemented using the `crepes` package developed by Boström (2024), with the underlying ML-models implemented using `scikit-learn` (Pedregosa et al., 2011).

Vovk and Bendtsen (2018) provided a PDMS for CPDM, which they proved is asymptotically efficient. Their approach can be seen in Algorithm 4, which we will refer to as *CPDM v1*.

Algorithm 4: CPDM v1

Input: a training set $(\boldsymbol{x}_i, y_i) \in Z,\ i = 1, \ldots, n-1$, a test object $\boldsymbol{x}_n \in X$, a set of decisions D, and an utility function $U(d, y)$

for $d \in D$ **do**

- Create a new training sequence $(x_i, U(y_i, d)),\ i = 1, \ldots, n-1$;
- Find the CPD Q_n^d for this new training sequence;
- Compute the expected utility of d as $\int u Q_n^d(du)$;

Output: $d \in D$ with the highest expected utility

Algorithm 5: Online Decision Making

Input: a training set $(\boldsymbol{x}_i, y_i) \in Z,\ i = 1, \ldots, n-1$, a test object $\boldsymbol{x}_n \in X$, a set of decisions D, and an utility function $U(y, d)$

Find the predictive distribution Q_n for the test object $\boldsymbol{x}_n$ given the training set;

for $d \in D$ **do**

  Compute the expected utility of d as $\int U(y, d) Q_n(dy)$;

Output: $d \in D$ with the highest expected utility

Transforming the labels of the training set using a utility function, as shown in Algorithm 4, is not standard practice in BDT. Instead, the predictive distribution is typically based on the original training set, as outlined in Algorithm 5. In this setting, the predictive distribution Q_n can be obtained either as a posterior distribution from a Bayesian model or as a CPD from a CPS. We refer to the former simply as *BDT*, and the latter as *CPDM v2*.

We can easily formulate an inductive version of Algorithm 5, as shown in Algorithm 6 for CPDM and in Algorithm 7 for BDT.

Algorithm 6: Inductive Decision Making CPDM

Input: a proper training set $(\boldsymbol{x}_i, y_i) \in Z,\ i = 1, \ldots, m$, a calibration set $(\boldsymbol{x}_i, y_i) \in Z,\ i = m+1, \ldots, l-1$, a test object $\boldsymbol{x}_l \in X$, a set of decisions D, and an utility function $U(y, d)$

Train the ML model on the proper training set;

Find the predictive distribution Q_l for the test object $\boldsymbol{x}_l$ given the calibration set;

for $d \in D$ **do**

  Compute the expected utility of d as $\int U(y, d) Q_l(dy)$

Output: $d \in D$ with the highest expected utility

Algorithm 7: Inductive Decision Making BDT

Input: a training set $(\boldsymbol{x}_i, y_i) \in Z,\ i = 1, \ldots, l-1$, a test object $\boldsymbol{x}_l \in X$, a set of decisions D, and an utility function $U(y, d)$

Find the predictive distribution Q_l for the test object $\boldsymbol{x}_l$ given the training set;

for $d \in D$ **do**

  Compute the expected utility of d as $\int U(y, d) Q_l(dy)$

Output: $d \in D$ with the highest expected utility

In both the online and inductive settings, the PPDM methods are obtained by using the point estimates provided by the ML model, rather than a predictive distribution. Hence, we do not compute expected utility and instead use a threshold $t = 0.5$ to decide which decision to make: $d = 1$ if $t > 0.5$, $d = 0$ otherwise. We do not include explicit algorithms for brevity, as they can be easily discerned from Algorithm 5, 6, and 7.

3.3 Models and Model Selection

The models considered in this study, along with their tunable hyperparameters and their corresponding search spaces or optimization strategies, are summarized in Table 3.3. In this study, all kernel-based models used the same kernel, namely, the RBF kernel, to maintain comparability and consistency between methods. Regarding the model selection, the non-Bayesian models were tuned using grid search and 5-fold cross-validation, while the Bayesian models were tuned automatically during fitting. In the inductive setting for CPDM, hyperparameter tuning was performed once on the proper training data before the calibration procedure. However, in the online setting, hyperparameters were re-tuned at each iteration to allow the models to adapt to new data over time dynamically.

For the experimental part of this thesis, the parametric models RR (LSPM for online CPDM) and BRR will serve as the underlying models for the different PDMSs on the Linear dataset, while the non-parametric methods KNN (NNPM for online CPDM), KRR (KRRPM for online CPDM), and GPR will be used as underlying models on the nonlinear Friedman datasets (#1, #2, and #3) to capture their inherent nonlinearities better.

Model	**Tunable Parameters**	**Search Space / Bounds**
Non-bayesian models (for CPDM and PPDM)		
KNN/NNPM	k (number of neighbors)	{1, 3, 5, 7, 10, 15, 20}
RR/LSPM	γ (regularization)	{0.01, 0.1, 1.0, 10.0, 100.0}
KRR/KRRPM	γ (regularization)	{0.001, 0.01, 0.1, 1.0}
	ℓ (length scale)	{1, 10, 100}
Bayesian Models (for BDT)		
BRR	Gamma distribution priors	
	β_1, β_2 (shape, inverse scale)	Internally optimized
	λ_1, λ_2 (shape, inverse scale)	Internally optimized
GPR	ℓ (length scale)	$[10^{-10}, 10^{10}]$

Table 3.3: An overview of models, their hyperparameters, and search spaces/bounds.

3.4 Utility Functions

Since the study investigates binary decision-making, it is most intuitive to formulate the utility functions by assigning values to the components of a confusion matrix. We customized the utility function for each dataset based on its target distribution. Moreover, we defined three different utility configurations in each case to explore how varying the penalty for misclassification affects model performance. The *standard configuration* served as the default across datasets, while the *slightly skewed* and the *extremely skewed* configurations aimed to simulate more risk-sensitive settings where the penalty for minority-class misclassifications was increased. All utility

configurations are detailed in Table 3.4, and for all experiments, a threshold of $t = 0.5$ was used to convert the values of the predictive distributions into class labels.

	Linear / Friedman #1		**Friedman #2**		**Friedman #3**	
Actual \Pred.	**True**	**False**	**True**	**False**	**True**	**False**
Standard						
True	1	-5	2	-10	1	-5
False	-5	1	-5	1	-10	2
Slightly skewed						
True	1	-10	2	-10	1	-2
False	-2	1	-2	1	-10	2
Extremely skewed						
True	1	-100	2	-100	1	-2
False	-1	1	-2	1	-100	2

Table 3.4: Utility matrices for all datasets under different penalty configurations.

3.5 Evaluation Metrics

In the online learning setting, we assessed the performance of each model using cumulative regret. The cumulative regret R_c is defined as:

$$R_c = \sum_{i=1}^{n} \left(U(y_i, d_i^*) - U(y_i, d_i) \right), \tag{3.1}$$

where n is the number of test examples, $U(y_i, d_i^*)$ is the utility obtained by the optimal decision d_i^* for the true value y_i, and $U(y_i, d_i)$ is the utility obtained by the decision d_i selected by the model. The metric quantifies the total utility lost over time due to suboptimal choices, effectively measuring how well the system improves its decision-making over time.

In the inductive learning setting, performance was instead assessed using average utility over test cases, since the systems do not improve their policy over time. The average utility $\bar{U}$ is defined as:

$$\bar{U} = \frac{1}{n} \sum_{i=1}^{n} U(y_i, d_i). \tag{3.2}$$

3.6 Experimental Setup and Configurations

This thesis adopts a comparative experimental design to evaluate the performance of two versions of the CPDM framework, followed by a comparison with the BDT and PPDM approaches. The methodology is based on a systematic evaluation of these methods under varying experimental conditions. The comparison between CPDM v1 and v2 is referred to as *Experiment 1* in Table 3.5, while the comparison between the best performing CPDM variant, BDT, and PPDM, in both the online and inductive settings, is denoted *Experiment 2*.

To ensure consistency across evaluations, we designed a series of controlled experiments based on the conditions outlined in the previous sections. The configurations differ along three dimensions: data availability, noise level, and utility function skewness. These dimensions may differ between inductive and online learning settings. Table 3.5 summarizes all experimental setups.

Configuration	**Data**	**Noise Level**	**Utility Function**	**Experiments**
#1	Standard	Standard	Standard	1, 2
#2	Standard	Standard	Slightly skewed	1, 2
#3	Standard	Standard	Extremely skewed	1
#4	Standard	Noisy	Standard	1, 2
#5	Standard	Noisy	Slightly skewed	1, 2
#6	Standard	Noisy	Extremely skewed	1
#7	Limited	Standard	Standard	1, 2
#8	Limited	Standard	Slightly skewed	1, 2
#9	Limited	Standard	Extremely skewed	1
#10	Limited	Noisy	Standard	1, 2
#11	Limited	Noisy	Slightly skewed	1, 2
#12	Limited	Noisy	Extremely skewed	1

Table 3.5: Overview of the experimental configurations with varying data availability, noise levels, and utility functions.

4

Results

This section presents the empirical evaluation of the PDMSs across all configurations. Here, the most relevant results are highlighted; additional results are provided in Appendix A. The first part compares the performance of CPDM v1 and v2 in the online setting, while the second part compares the best-performing CPDM method against BDT and PPDM in both online and inductive learning settings. For brevity, we will refer to the different PDMSs in this chapter by their base model names only (e.g., BDT with BRR will be denoted simply as BRR, and CPDM v1 with KRRPM as KRRPM).

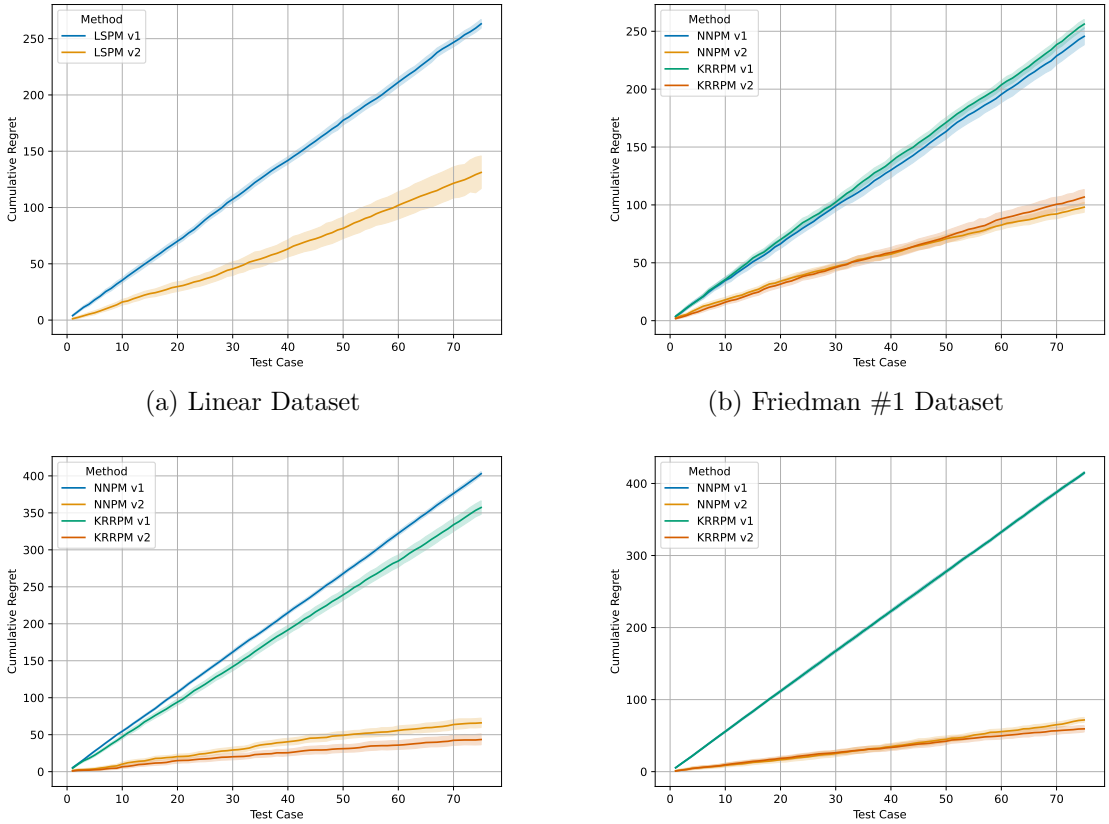
4.1 Comparison of CPDM Approaches

Each presented graph in this section shows cumulative regret over test cases with 95% confidence intervals based on 50 runs.

Figure 4.1 shows the cumulative regret for each dataset under Configuration #1. Across all datasets, CPDM v2 consistently outperformed CPDM v1 by achieving lower cumulative regret, indicating more effective decision-making. Additionally, the increase in cumulative regret of the v2 methods diminished slightly over time, while that of the v1 methods continued to be linear. This shows that the v2 methods learn somewhat over time, unlike the v1 methods.

Figure 4.2 compares CPDM v1 and v2 methods under Configuration #2, which introduces a slightly skewed utility function. The results were largely similar to those of Configuration #1, with CPDM v2 still significantly outperforming all corresponding CPDM v1 methods. However, one notable exception was observed for the Friedman #1 dataset: while NNPM v1 improved its performance relative to CPDM v2 methods, KRRPM v1 showed a significant performance decline. Nevertheless, both CPDM v2 methods still outperformed NNPM v1.

Figure 4.3 presents the results for each method under Configuration #3 with the extremely skewed utility function. For the Linear dataset, CPDM v1 slightly outperformed CPDM v2 and produced narrower confidence intervals; however, both methods struggled to learn. For the Friedman #1 dataset, all methods also struggled to learn and performed similarly, except for KRRPM v2, which showed slightly better performance. In Friedman #2, both v2 methods learned over time and performed equally well, while the v1 methods struggled, though NNPM v1 achieved



(a) Linear Dataset

(b) Friedman #1 Dataset

(c) Friedman #2 Dataset

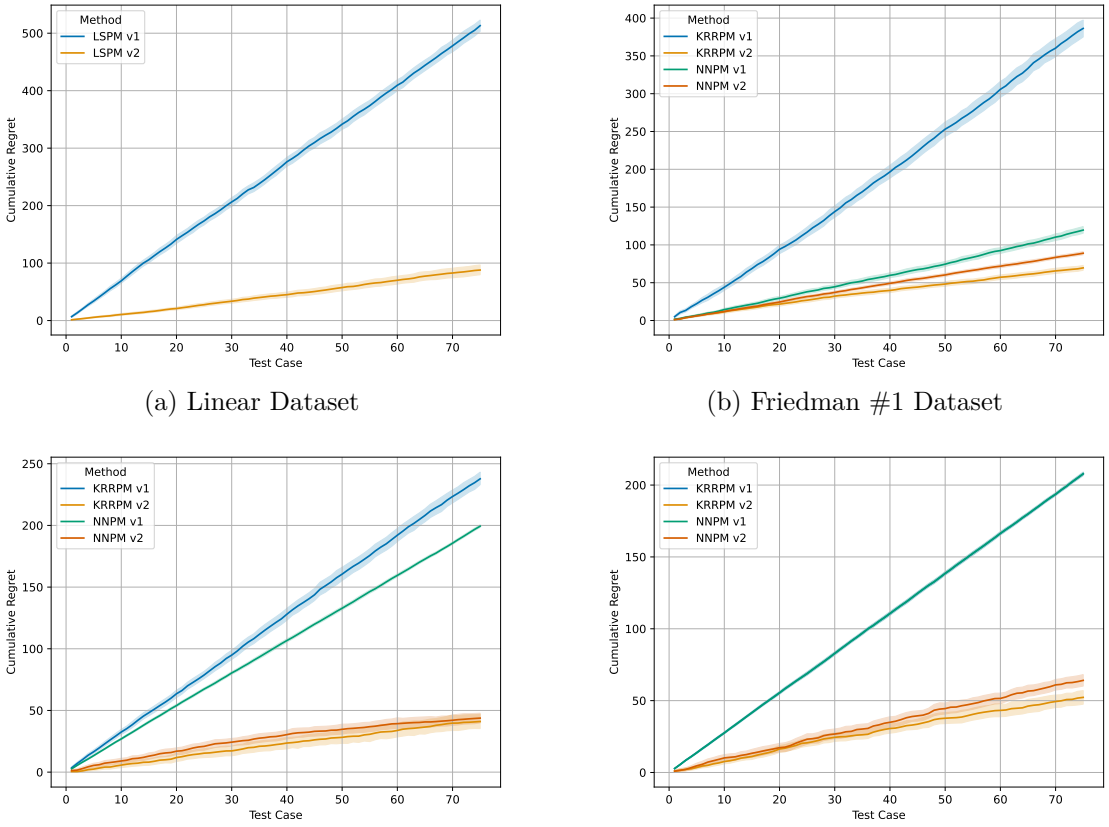
(d) Friedman #3 Dataset

Figure 4.1: Comparison of CPDM v1 and v2 in the online setting under Configuration #1.

acceptable performance, slightly below the v2 methods. For Friedman #3, v1 methods performed worse than their v2 counterparts and showed no signs of learning the data. The v2 methods also struggled to learn but demonstrated small improvements over time, albeit with considerably wider confidence intervals than their v1 counterparts.

For Configurations #4–#6, where noise was added to the data, the performance differences between CPDM v1 and v2 became less pronounced. However, CPDM v2 generally still achieved better performance. The only exception was Configuration #6, where all methods performed similarly, save for the Friedman #2 dataset, in which CPDM v1 outperformed CPDM v2. Another notable observation was made for the Friedman #1 dataset in Configuration #5: compared to Configuration #2, KNN-based methods now performed equally well.

In general, for Configurations #7–#9, where limited training data was used, the relative performance differences between methods remained similar to those in the standard data configurations. However, when comparing results from Configuration #9 with those of Configuration #3, CPDM v2 methods performed noticeably worse than CPDM v1 methods. Still, CPDM v2 methods performed similarly to CPDM v1 methods in most cases, except on the Linear dataset.



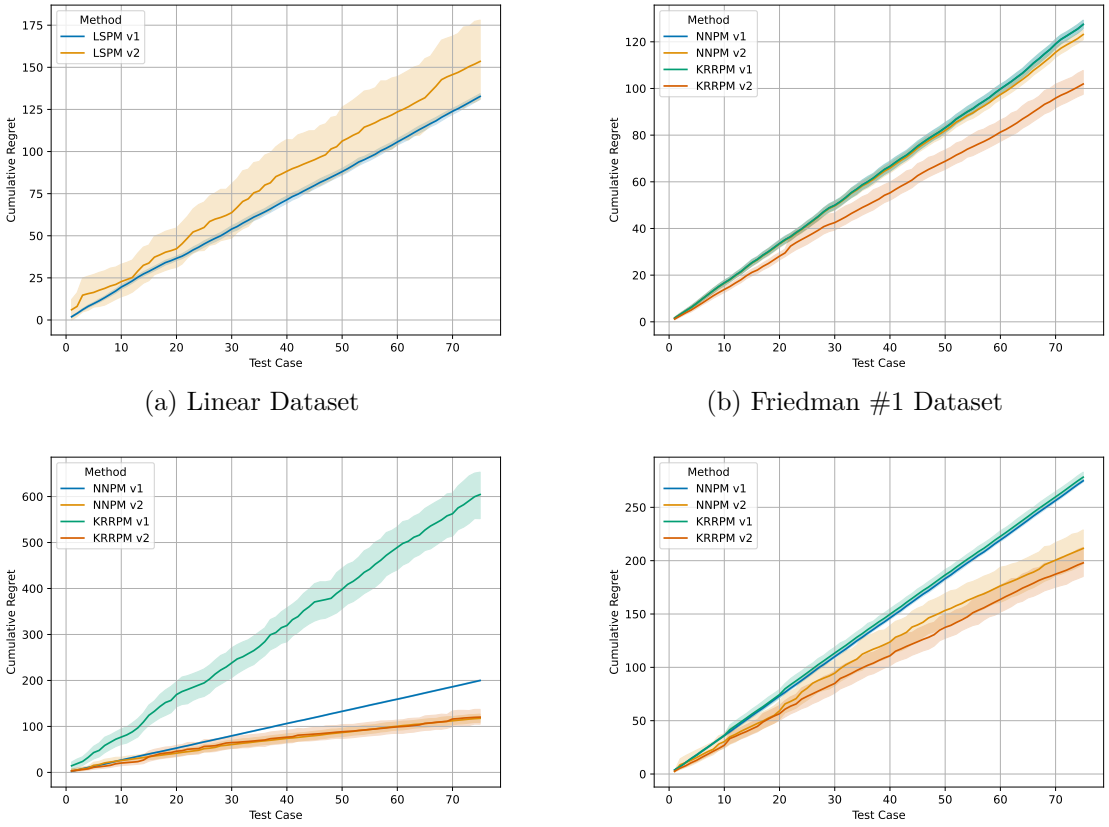
(a) Linear Dataset

(b) Friedman #1 Dataset

(c) Friedman #2 Dataset

(d) Friedman #3 Dataset

Figure 4.2: Comparison of CPDM v1 and v2 in the online setting under Configuration #2.



(a) Linear Dataset

(b) Friedman #1 Dataset

(c) Friedman #2 Dataset

(d) Friedman #3 Dataset

Figure 4.3: Comparison of CPDM v1 and v2 in the online setting under Configuration #3.

In Configurations #10–#12, where both noise was added and training data was limited, the performance gaps were again reduced, similar to Configurations #4–#6. For Configurations #11 and #12 a pattern was observed: the results were truncated earlier due to fewer iterations, meaning that CPDM v2 methods had insufficient learning examples to fully catch up to their v1 counterparts. For example, in Friedman #2, Configuration #12, CPDM v2 methods clearly learn the data, but the method did not match CPDM v1 performance as it did not have enough examples to converge.

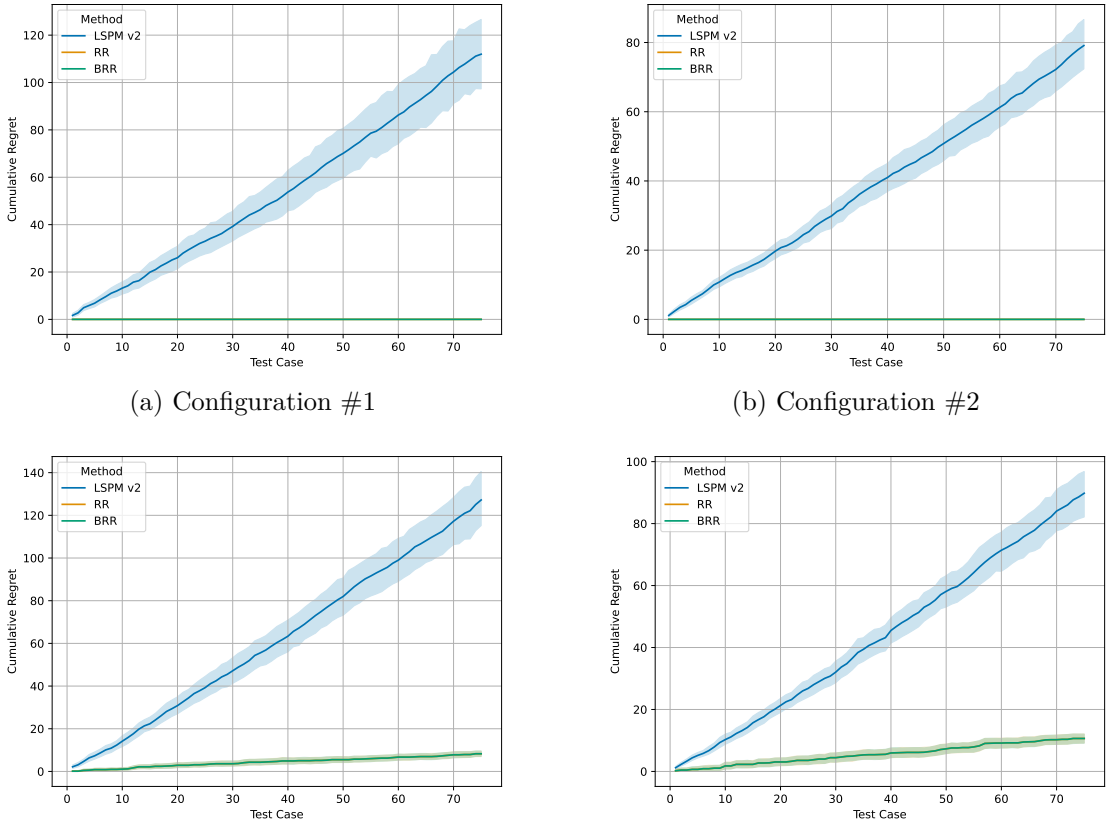
4.2 Comparison of CPDM, BDT, and PPDM

In the previous section, we observed that CPDM v2 generally outperformed CPDM v1, with the exception of the Linear dataset under an extremely skewed utility function, where its performance was slightly worse but still roughly equivalent. This suggests that certain edge cases may favor CPDM v1. However, such extreme utility functions are unlikely to occur in practice. Therefore, in this section, we focus our comparison on CPDM v2, BDT, and PPDM, excluding CPDM v1 from further analysis. In the online setting, each graph shows cumulative regret over test cases with 95% confidence intervals based on 50 runs. In the inductive setting, the corresponding metric is average utility over test cases based on 100 runs.

4.2.1 Online Setting

Figure 4.4 presents the results for the Linear dataset in the online learning setting across all configurations. The results showed that RR and BRR consistently outperformed LSPM v2, and in both Configurations #1 and #2, they demonstrated optimal performance. Although performance declined slightly when noise was introduced in Configurations #4 and #5, RR and BRR still maintained a clear advantage.

Figure 4.5 shows the results for the Friedman #1 dataset. In Configuration #1, the PPDM methods outperformed their CPDM v2 counterparts, while GPR achieved the best overall performance. All methods exhibited some learning, with GPR showing the most effective improvement. A similar pattern was observed for Configuration #2. However, in this case, the CPDM v2 methods outperformed their PPDM counterparts. In this configuration GPR still achieved the highest performance, though the performance gap to the other methods slightly decreased. Introducing noise into the data, as in Configurations #4 and #5, drastically reduced GPR's performance and its ability to learn. In Configuration #4, PPDM methods still outperformed CPDM v2. However, with the more skewed utility function in Configuration #5, CPDM v2 demonstrated significantly better performance. Overall, CPDM methods appeared more robust to noise and skewness, consistently outperforming PPDM under these conditions. In contrast, GPR performed substantially worse, struggling to adapt to the more complex environment.



(a) Configuration #1

(b) Configuration #2

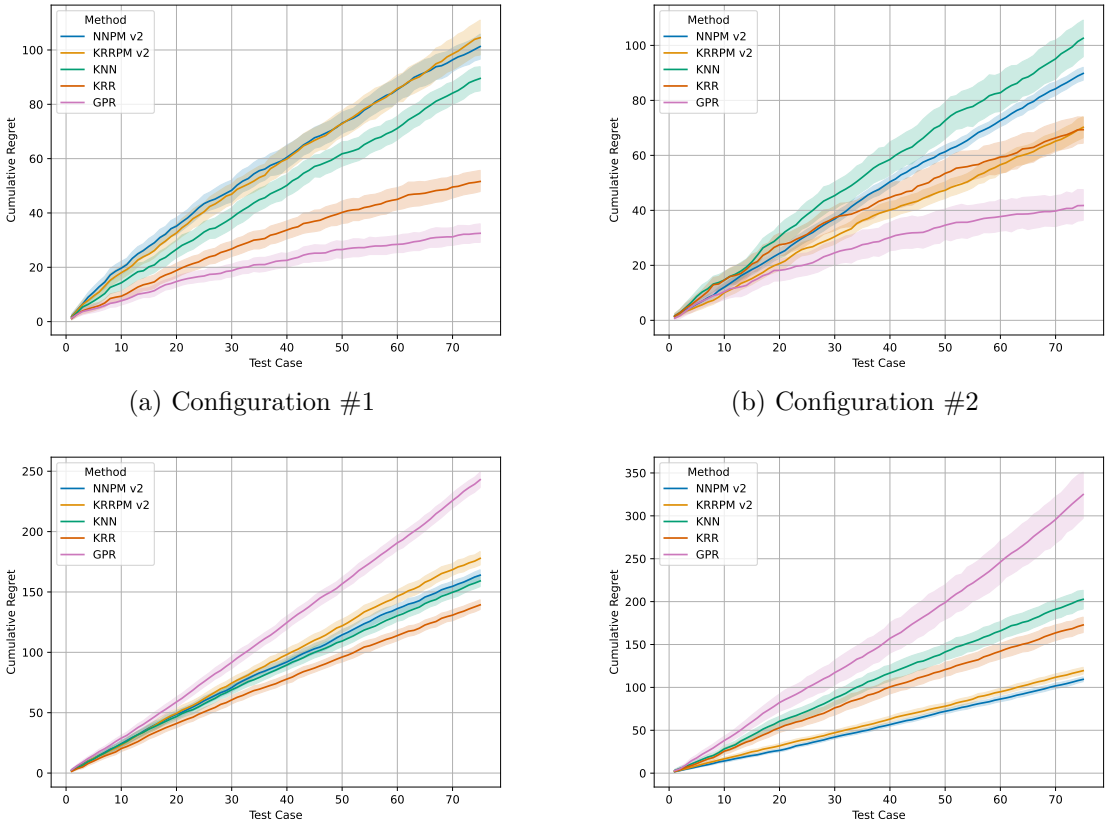
(c) Configuration #4

(d) Configuration #5

Figure 4.4: Online setting comparison for the Linear dataset of CPDM v2, PPDM, and BDT methods under different configurations.

Results for the Friedman #2 dataset are shown in Figure 4.6. On this dataset, GPR significantly outperformed all other methods in Configurations #1 and #2, fully learning the decision problem. Among the remaining methods, PPDM consistently outperformed its CPDM v2 counterparts, with KRR achieving the best performance by fully learning the problem in approximately 50 examples. The other methods showed some signs of learning, though not to the same extent as GPR and KRR. Under Configuration #2, CPDM v2 methods began to catch up with their PPDM counterparts; however, they still demonstrated worse performance.

In Configurations #4 and #5, when noise was introduced into the data, GPR continued to outperform the CPDM v2 methods overall, although it struggled more and was unable to fully learn the decision problem. Similarly, PPDM methods also experienced reduced performance but still outperformed CPDM v2. Notably, in both Configurations #4 and #5, KRR outperformed all other methods. As with the Friedman #1 dataset, CPDM v2 demonstrated greater robustness to noise and changes in the utility function. However, for this dataset, that robustness did not translate into superior overall performance as it did for Friedman #1.



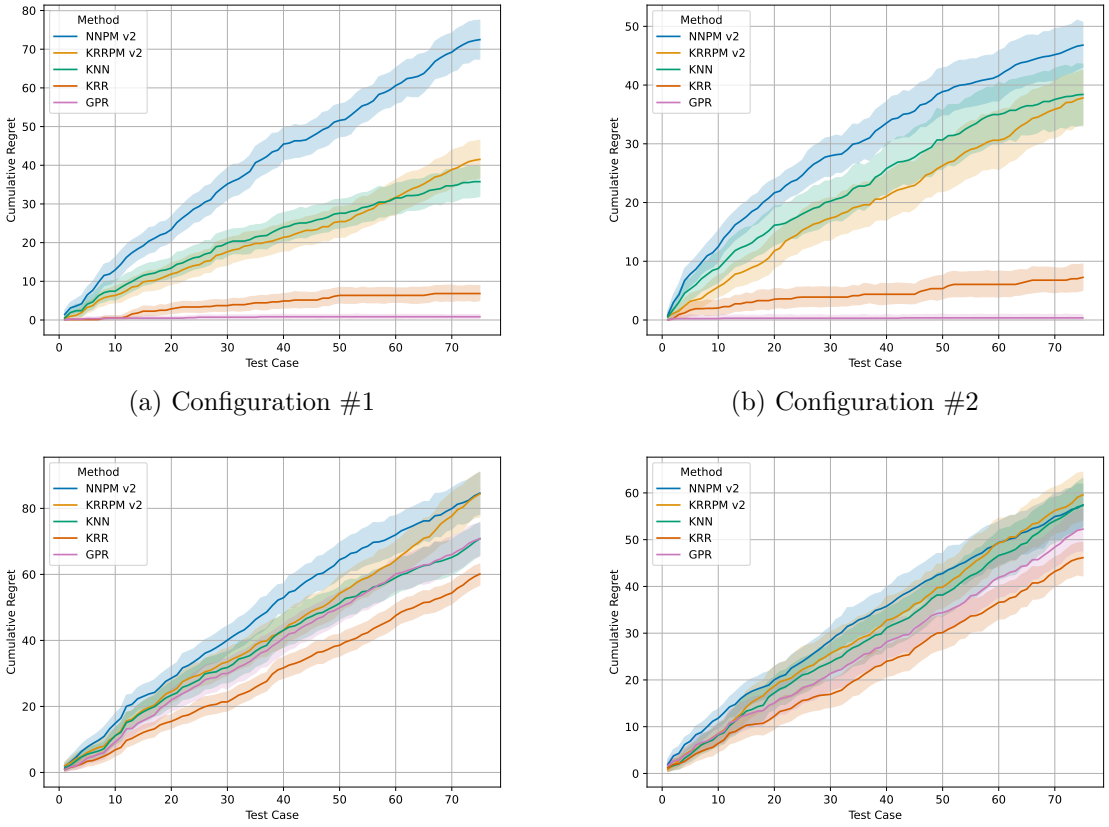
(a) Configuration #1

(b) Configuration #2

(c) Configuration #4

(d) Configuration #5

Figure 4.5: Online setting comparison for the Friedman #1 dataset of CPDM v2, PPDM, and BDT methods under different configurations.



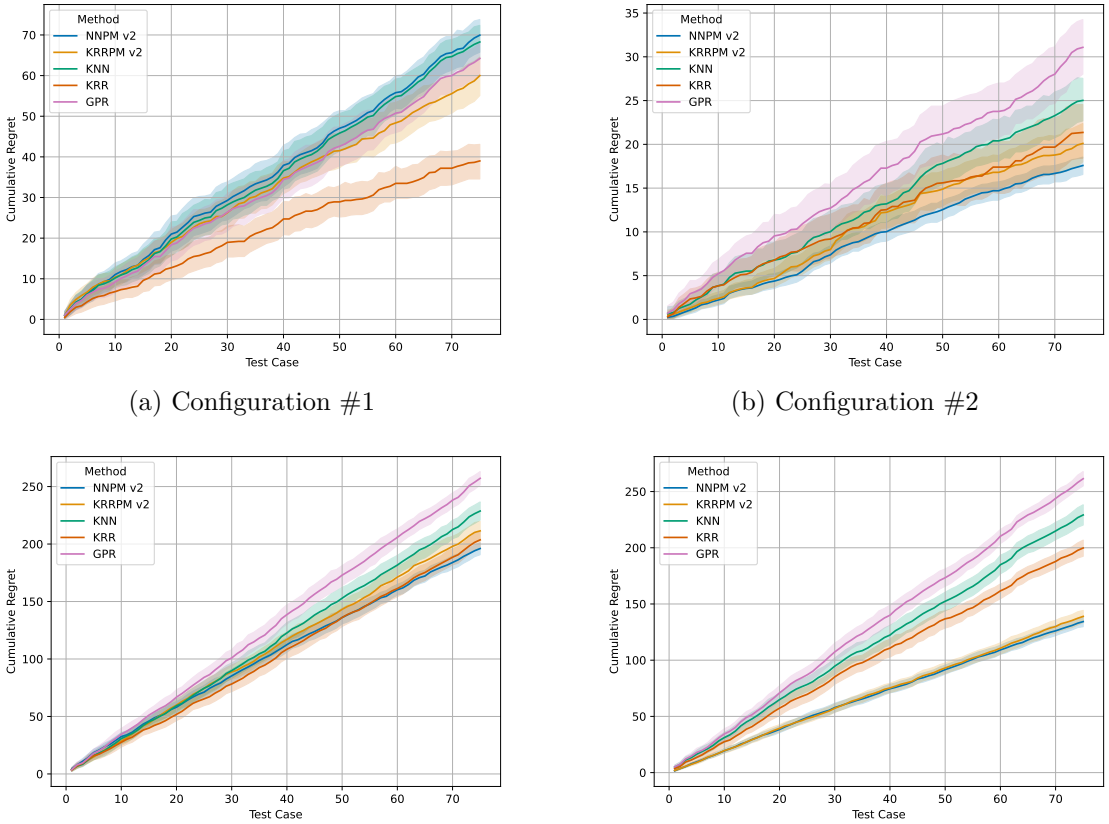
(a) Configuration #1

(b) Configuration #2

(c) Configuration #4

(d) Configuration #5

Figure 4.6: Online setting comparison for the Friedman #2 dataset of CPDM v2, PPDM, and BDT methods under different configurations.



(a) Configuration #1

(b) Configuration #2

(c) Configuration #4

(d) Configuration #5

Figure 4.7: Online setting comparison for the Friedman #3 dataset of CPDM v2, PPDM, and BDT methods under different configurations.

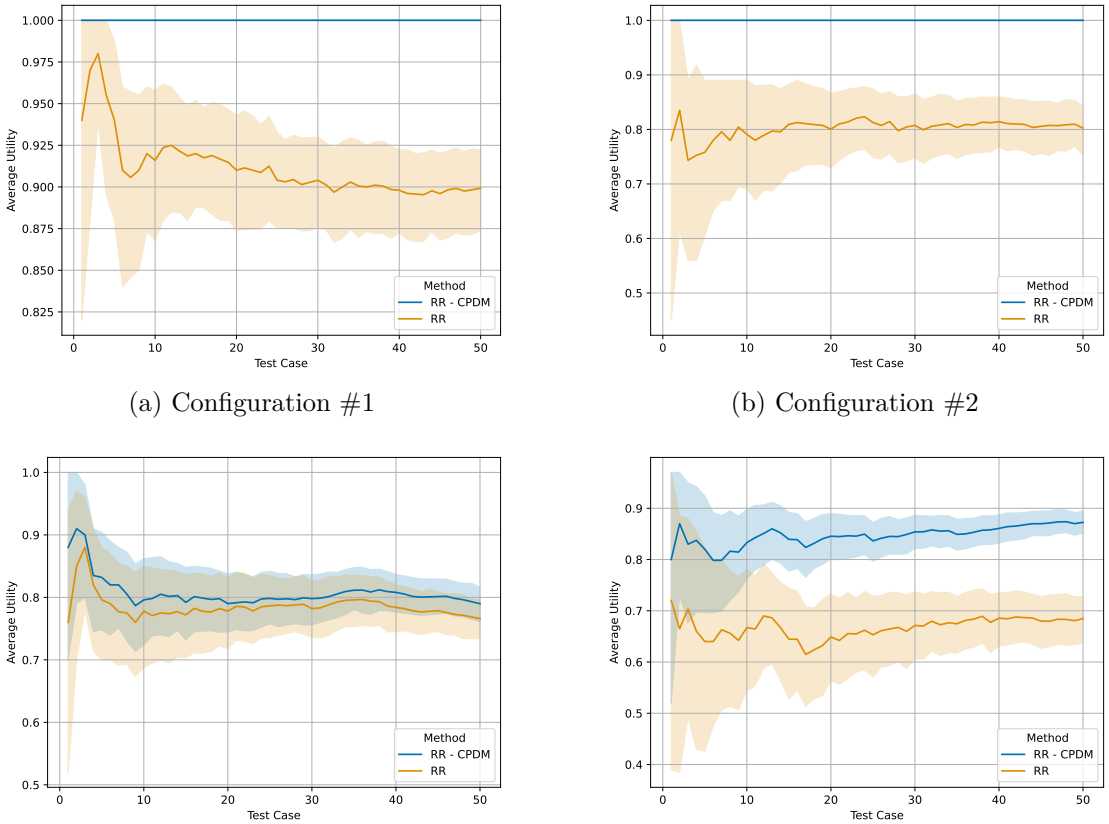
Results for the Friedman #3 dataset are shown in Figure 4.7. For this dataset, all methods performed similarly in Configuration #1, except for KRR, which clearly outperformed the others and was the only method showing signs of learning the problem. In Configuration #2, GPRs performance dropped significantly and became notably worse than that of the other methods. Here, CPDM v2 methods slightly outperformed their PPDM counterparts, though no method showed signs of learning the problem. This trend was even more pronounced in Configuration #4, where GPR again performed the worst. The remaining methods performed similarly, with CPDM v2 maintaining a slight edge over PPDM. In Configuration #5, CPDM v2 methods demonstrated superior performance compared to all others, though once again, none of the methods showed meaningful signs of learning the problem.

Compared to the configurations using standard data availability, the results of Configurations #7, #8, #10, and #11, where less training data was used, showed that the relative performance of the methods remained largely unchanged across all datasets. This indicates that data availability did not significantly affect the overall ranking of the methods.

However, one interesting observation emerged in the limited data setting: in Friedman #1, Configuration #11, compared to Configuration #5, KNN appears to learn the dataset more effectively in the limited data setting and significantly outperforms all other methods.

4.2.2 Inductive Setting

For the inductive setting on the Linear dataset, results are presented in Figure 4.8. In both Configurations #1 and #2, RR-CPDM achieved a perfect score on the test dataset, demonstrating strong generalization to simple problems when sufficient data was provided. BRR and RR showed similar performance, with a slight drop in average utility from Configuration #1 to #2. In Configuration #4, where noise was added to the data, all methods performed similarly. However, in Configuration #5, which combined noise and a slightly skewed utility function, RR-CPDM once again outperformed all other methods. Overall, RR-CPDM showed greater robustness and handled increased complexity better than the alternative models on the Linear dataset.



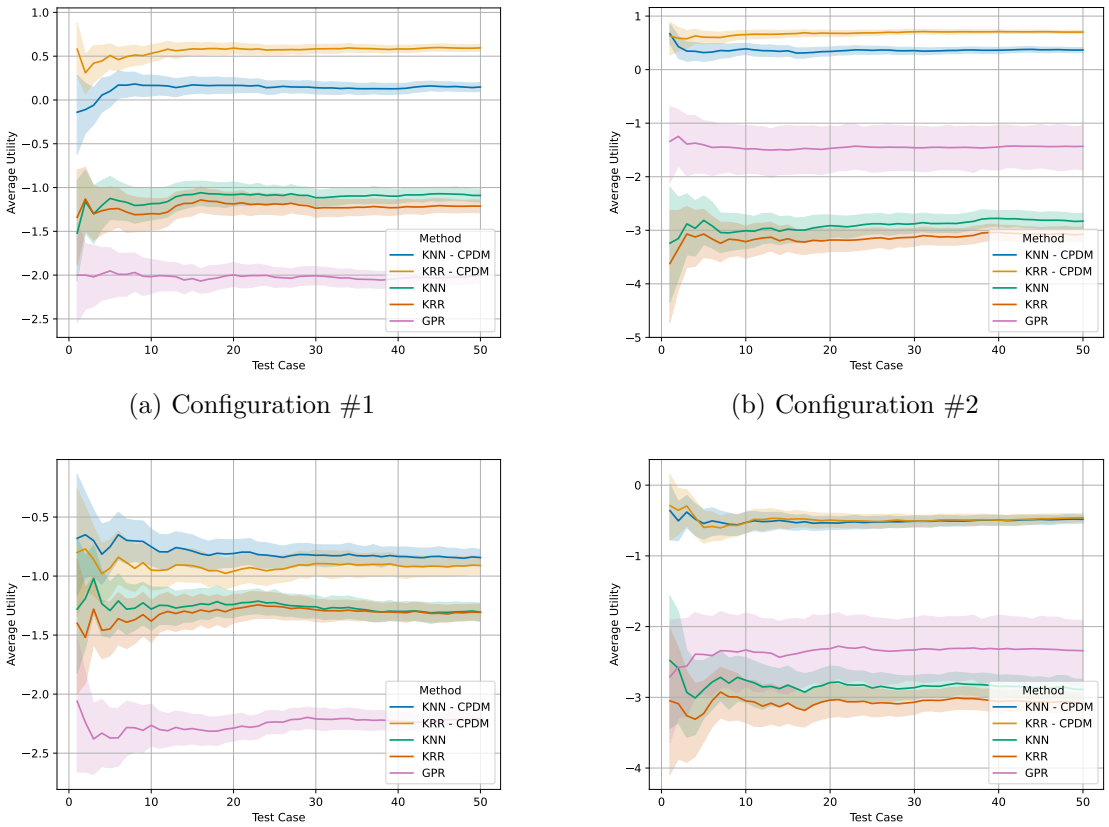
(a) Configuration #1

(b) Configuration #2

(c) Configuration #4

(d) Configuration #5

Figure 4.8: Inductive setting comparison for the Linear dataset of CPDM, PPDM, and BDT methods under different configurations.



(a) Configuration #1

(b) Configuration #2

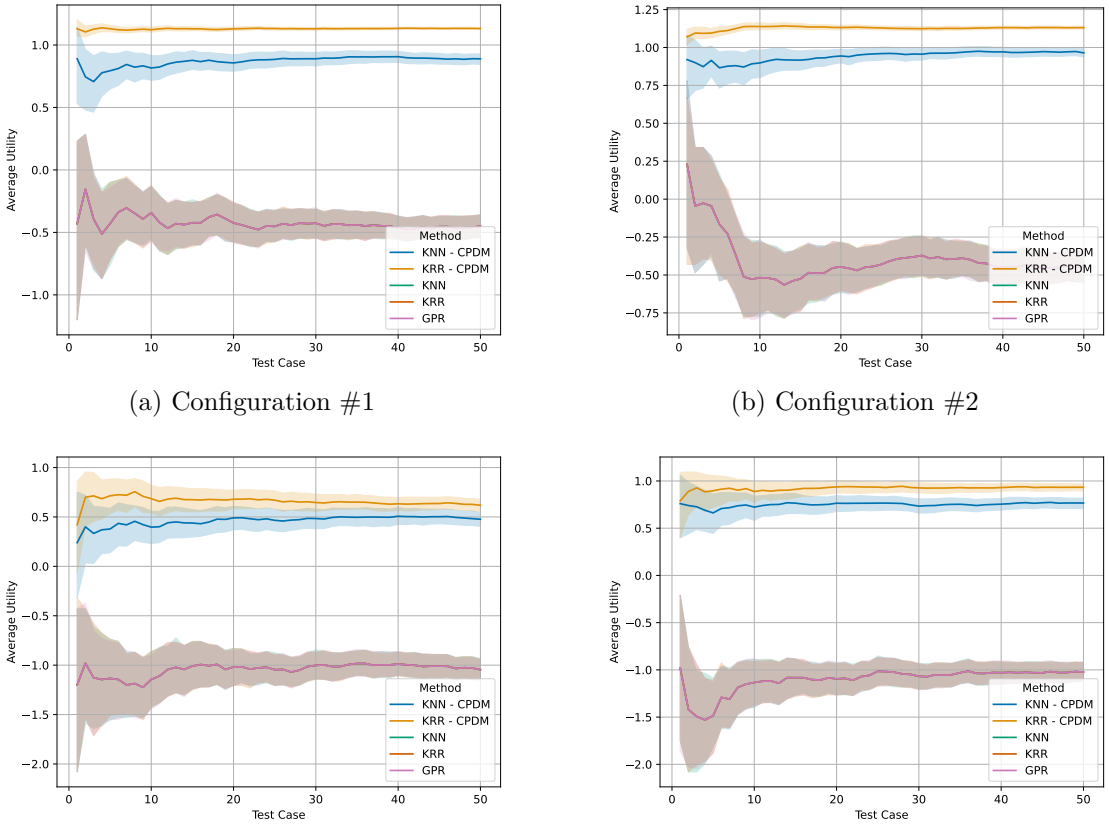
(c) Configuration #4

(d) Configuration #5

Figure 4.9: Inductive setting comparison for the Friedman #1 dataset of CPDM, PPDM, and BDT methods under different configurations.

Results for the Friedman #1 dataset in the inductive setting are shown in Figure 4.9. Here, CPDM methods clearly outperformed the others across all configurations. The performance of both PPDM methods and GPR varied across configurations, and in Configuration #4, PPDM methods performed just slightly worse compared to CPDM methods. For this configuration, KNN-CPDM also outperformed KRR-CPDM, suggesting better robustness to noise. In Configurations #2 and #5, GPR handled the slightly skewed utility function better than PPDM and in Configuration #5, both KNN-CPDM and KRR-CPDM performed similarly. Nevertheless, CPDM remained the top-performing approach overall.

A similar pattern to that observed for the Friedman #1 dataset in the inductive setting was seen for Friedman #2, with results shown in Figure 4.10. CPDM methods clearly outperformed all others across all configurations, with KRR-CPDM achieving the best performance. GPR and PPDM methods performed similarly across configurations in this setting. The relative performance among CPDM methods was also consistent, though the performance gap narrowed slightly in Configurations #4 and #5, where KNN closed in on KRR.



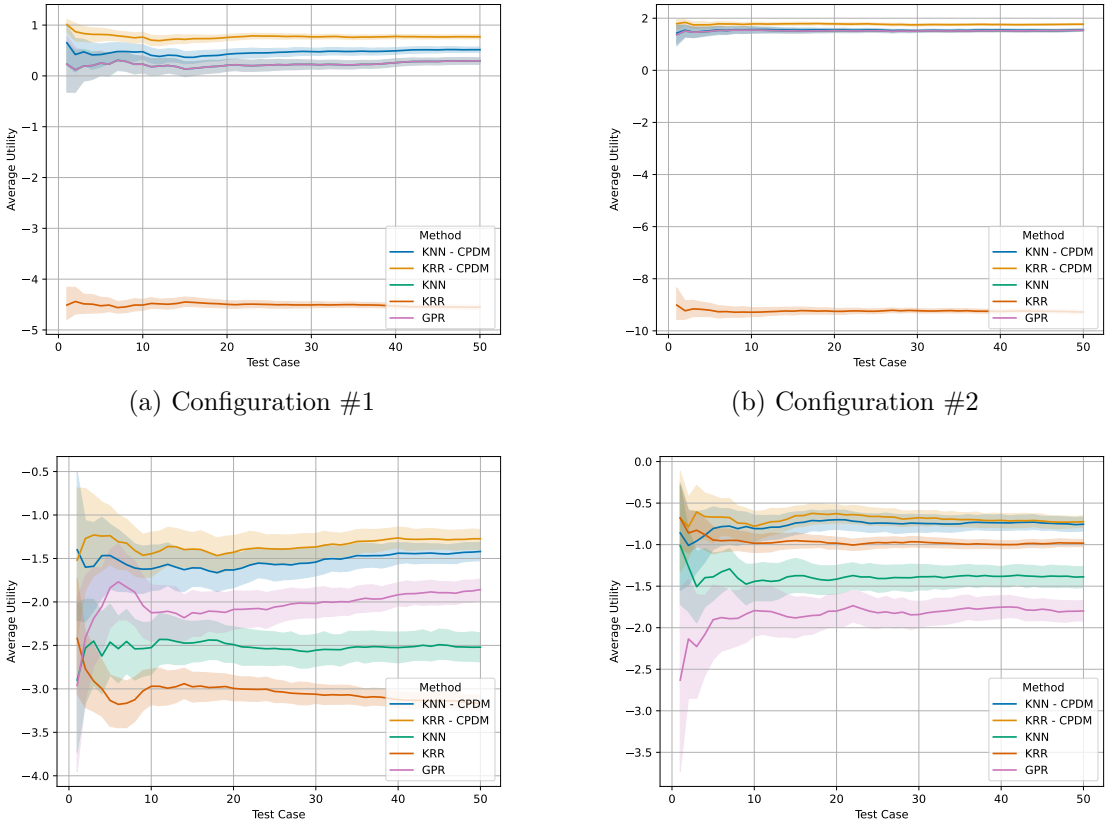
(a) Configuration #1

(b) Configuration #2

(c) Configuration #4

(d) Configuration #5

Figure 4.10: Inductive setting comparison for the Friedman #2 dataset of CPDM, PPDM, and BDT methods under different configurations.



(a) Configuration #1

(b) Configuration #2

(c) Configuration #4

(d) Configuration #5

Figure 4.11: Inductive setting comparison for the Friedman #3 dataset of CPDM, PPDM, and BDT methods under different configurations.

Results for the Friedman #3 dataset in the inductive setting are shown in Figure 4.11. In Configuration #1, CPDM methods slightly outperformed GPR, with KRR-CPDM achieving the highest average utility. In Configuration #2, GPR and KNN-CPDM performed similarly, while KRR-CPDM still performed marginally better than both. In both Configurations #1 and #2, PPDM methods performed significantly worse than all other methods. In Configurations #4 and #5, PPDM methods performed relatively better and even outperformed GPR in Configuration #5. Nevertheless, CPDM methods maintained the highest average utility in both configurations, with KRR-CPDM leading in Configuration #4 and both CPDM variants showing similar performance in Configuration #5.

Compared to the configurations using standard data availability, the results of Configurations #7, #8, #10, and #11, with reduced training data, showed that the relative ranking of the methods remained largely unchanged. However, in some cases, the performance gap were smaller. A few results are worth noting from the limited data configurations:

- **Friedman #1, Configuration #10:** Compared to Configuration #4, the PPDM methods performed more similarly to CPDM, suggesting that CPDMs advantage may diminish when training data is scarce for this dataset.
- **Friedman #1, Configuration #7:** Relative to Configuration #2, GPR showed improved performance, though it still did not outperform the CPDM methods.
- **Friedman #3, Configurations #7 and #8:** Compared to Configurations #1 and #2, the relative performance of the PPDM methods increased.

5

Discussion

In this chapter, we discuss the results from the previous chapter, dividing the discussion into three parts: first, a comparison of the two CPDM approaches; second, an analysis of the three frameworks on the Linear dataset; and third, a comparison of the frameworks on the nonlinear datasets. As in the previous chapter, we will refer to the different PDMSs by their base model names only, for brevity.

5.1 Comparison of CPDM Approches

Our results indicate that CPDM v2 generally outperforms CPDM v1 and demonstrates a greater ability to learn. This is particularly evident when the utility function is not extreme, which is common in most practical applications. However, CPDM v1 may offer advantages in rare scenarios where a highly unbalanced utility function is justified, such as in high-stakes decisions with potentially catastrophic consequences. Nevertheless, in such cases, predictive decision-making systems may not be reliable, limiting the practical usefulness of the CPDM v1 approach. Additionally, CPDM v2 provides a clear computational advantage: unlike CPDM v1, it does not require recomputing the CPD for each decision, making it more scalable.

One possible explanation for the performance difference is that in CPDM v1, transforming the targets before model training can obscure information about the underlying distribution, distorting the predictive structure and leading to less reliable estimates. In contrast, CPDM v2 applies the utility function after the predictive distribution is formed, which may avoid this distortion and result in more accurate estimates, ultimately improving generalization and decision quality. This probably also explains why the performance gap between the methods narrows when noise is introduced: CPDM v2 no longer benefits as strongly from leveraging the underlying structure of the data, as the added noise reduces the amount of meaningful information available for the model to exploit.

5.2 Comparison of CPDM, BDT, and PPDM on the Linear Dataset

On the Linear dataset, BRR and RR produced nearly identical results in both the online and inductive settings. This similarity is expected, as both models are based on the same underlying linear assumption, and since the predictive mean in BRR corresponds to the point prediction made by RR, assuming they have similar regularization. In our case, the only difference lies in how the regularization parameters are selected for the two methods, which might result in different linear models. This underscores how BRR can be viewed as a probabilistic generalization of RR.

When comparing CPDM v2 against BDT and PPDM on the Linear dataset, we observed that it did not achieve comparable performance in the online setting. This highlights the cost of achieving validity at the expense of efficiency. Therefore, in situations where the assumptions of linear regression are largely satisfied, CPDM is unlikely to be particularly useful. The validity guarantees come with an efficiency cost that degrades performance. This situation is discussed in Vovk et al. (2022) under the name *Burnaev-Wasserman programme*, whose main concern is how much we have to "pay" in efficiency for the gain in validity provided by CP. If the price is low, conformalizing may be viewed as a cheap insurance policy; otherwise, it may be viewed as being over insured, as appears to be the case here.

In the inductive setting, the results were more comparable across the different PDMSs, with CPDM outperforming the others in all configurations. One possible explanation for the improved performance of RR-CPDM in this setting is the increased amount of data—it may require a certain volume of data to reach the same level of performance as the other two methods. However, RR-CPDM v2 still performed comparably even when we reduced the training data, thus pointing towards another explanation. An investigation of the distributions revealed that the inductive CPDs were much narrower than the online CPDs, i.e., more efficient. Given that the linear regression assumptions are satisfied in the data, the increase in efficiency seems to allow the model to be more accurate in its decisions. In other words, compared to the online setting, we do not seem to have to "pay" an efficiency cost for the gain in validity provided by inductive CP.

5.3 Comparison of CPDM, BDT, and PPDM on the Nonlinear Dataset

When comparing CPDM v2 to BDT and PPDM on the nonlinear datasets in the online setting, we generally found that CPDM v2 handled noise more effectively. In percentage terms, its performance drop was always less than or equal to that of the other methods, indicating that CPDM v2 offers some robustness benefits. However, the other methods often outperformed CPDM v2 overall. Among the models, KNN consistently handled noise better than KRR, regardless of the framework. Notably, GPR was the worst-performing method in terms of noise robustness.

GPR's sensitivity to noise must be due to the model underestimating the noise level, causing it to fit the noise rather than capturing the true underlying trend, resulting in overfitting and poor generalization. The RBF kernel may not be expressive enough to account for the noise, and performance might have improved if the kernel had explicitly modeled it. Interestingly, KRR-based methods, which use the same kernel, did not exhibit the same level of degradation. This could be because hyperparameter tuning is simpler for KRR, constraining it to more robust configurations, whereas GPR has more flexibility in its hyperparameter search, which may lead to overfitting.

In terms of utility function skewness, we generally found that PPDM performed worse than the other PDMS. This is likely due to its lack of a predictive distribution, which appears to be particularly beneficial in such cases. For GPR, skewness had no major effect, except for a slight impact on the Friedman #3 dataset.

When combining increased noise with a skewed utility function, CPDM v2 significantly outperformed the other methods, with the exception of the Friedman #2 dataset. This suggests that the CPDM framework is especially effective in complex scenarios.

CPDM also outperformed the other methods in the inductive setting for all nonlinear datasets, with consistently better performance across configurations. Similarly, as for the Linear dataset in the inductive setting, the more efficient inductive CPDs are likely the explanation for the relatively stronger performance.

Data availability had little impact on the relative ranking of methods on the nonlinear datasets, regardless of the learning setting. However, in the inductive setting, there were indications that limited data availability slightly reduced the relative performance of CPDM, even though it remained the best-performing approach overall. This suggests that even in the limited data configurations, the calibration sets for the ICPSs were sufficiently large to provide informative predictive distributions for reliable decision-making. Nevertheless, the performance decline could potentially be mitigated by employing more advanced inductive conformal methods, such as *cross-conformal prediction* described in Vovk et al. (2022), which do not require a separate calibration set. However, this was outside the scope of the thesis.

6

Conclusion

In conclusion, CPDM v2 demonstrates both performance and computational advantages over CPDM v1, particularly when utility functions are not too extreme. It also showed greater robustness than BDT and PPDM in scenarios involving noisy data and skewed utility functions. However, in simpler online settings, where narrow parametric or Bayesian assumptions might be satisfied, CPDM v2 may introduce unnecessary overhead. In such cases, the validity guarantee costs too much in lost efficiency, limiting its practical benefit.

Importantly, CPDM v2 is not always superior to BDT; performance varies depending on the specific scenario. There are settings where BDT outperforms CPDM v2 and vice versa, and it remains unclear under which conditions one consistently outperforms the other. Therefore, it is important to evaluate both methods within the context of the specific application.

Based on the results presented in this paper, several natural directions for future work emerge. First, it would be valuable to investigate whether CPDM v2 also achieves asymptotic efficiency in predictive decision-making, as has been proven for CPDM v1 by Vovk and Bendtsen (2018). Establishing this property would strengthen the theoretical foundations of CPDM v2 and further justify its use in practical applications over CPDM v1. Second, it is important to evaluate the performance of CPDM in real-world applications, in both online and inductive settings, across varying amounts of data. This would provide a deeper understanding of the frameworks behavior in more complex scenarios, offer insights into the trade-off between validity guarantees and computational efficiency in online versus inductive contexts, and help identify the data threshold at which PPDM typically outperforms CPDM. Third, extending the current binary decision-making framework to a multi-class setting would broaden the applicability of CPDM to more realistic and diverse use cases.

Bibliography

(1981). Mushroom. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5959T.

Agrawal, A., Gans, J. S., and Goldfarb, A. (2019). Exploring the impact of artificial intelligence: Prediction versus judgment. *Information Economics and Policy*, 47:1–6. The Economics of Artificial Intelligence and Machine Learning.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York.

Boström, H. (2024). Conformal prediction in python with crepes. In *Proc. of the 13th Symposium on Conformal and Probabilistic Prediction with Applications*, pages 236–249. PMLR.

Duvenaud, D. (2014). *Automatic model construction with Gaussian processes*. PhD thesis, Apollo - University of Cambridge Repository.

Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1 – 67.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

Murphy, K. P. (2022). *Probabilistic Machine Learning: An introduction*. MIT Press.

Murphy, K. P. (2023). *Probabilistic Machine Learning: Advanced Topics*. MIT Press.

Nemani, V., Biggio, L., Huan, X., Hu, Z., Fink, O., Tran, A., Wang, Y., Zhang, X., and Hu, C. (2023). Uncertainty quantification in machine learning for engineering design and health prognostics: A tutorial. *Mechanical Systems and Signal Processing*, 205:110796.

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

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Szabadváry, J. H., Löfström, T., and Matela, R. (2025). online-cp: a python package for online conformal prediction, conformal predictive systems and conformal test martingales. Submitted Proceedings of the Fourteenth Symposium on Conformal and Probabilistic Prediction with Applications, under review.

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.*, 1:211244.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley.

von Neumann, J. and Morgenstern, O. (1953). *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, 3rd edition. First edition: 1944.

Vovk, V. and Bendtsen, C. (2018). Conformal predictive decision making. In Gammerman, A., Vovk, V., Luo, Z., Smirnov, E., and Peeters, R., editors, *Proceedings of the Seventh Workshop on Conformal and Probabilistic Prediction and Applications*, volume 91 of *Proceedings of Machine Learning Research*, pages 52–62. PMLR.

Vovk, V., Gammerman, A., and Shafer, G. (2005). *Algorithmic learning in a random world*, volume 29. Springer.

Vovk, V., Gammerman, A., and Shafer, G. (2022). *Algorithmic Learning in a Random World*. Second edition.

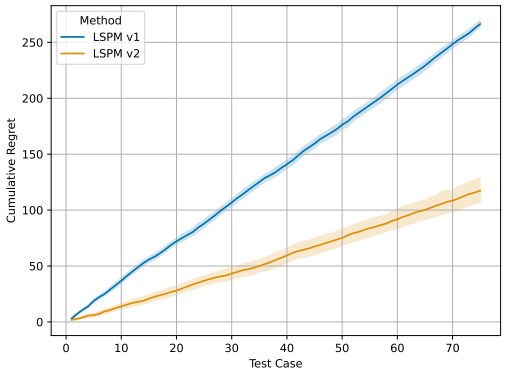
Vovk, V., Shen, J., Manokhin, V., and ge Xie, M. (2017). Nonparametric predictive distributions based on conformal prediction. In Gammerman, A., Vovk, V., Luo, Z., and Papadopoulos, H., editors, *Proceedings of the Sixth Workshop on Conformal and Probabilistic Prediction and Applications*, volume 60 of *Proceedings of Machine Learning Research*, pages 82–102. PMLR.

A

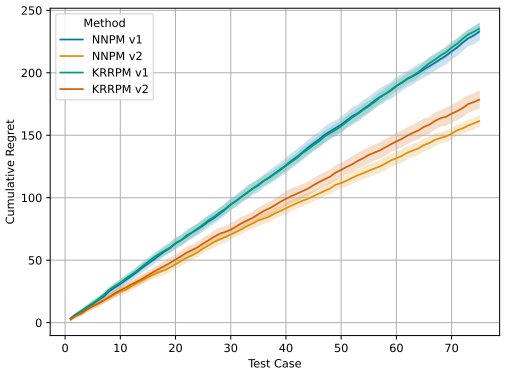
Additional Results

This appendix contains the figures not included in the result chapter of the thesis.

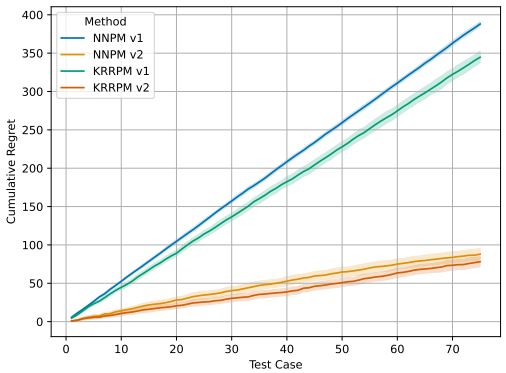
A.1 Comparison of CPDM v1 and v2, in limited data configurations



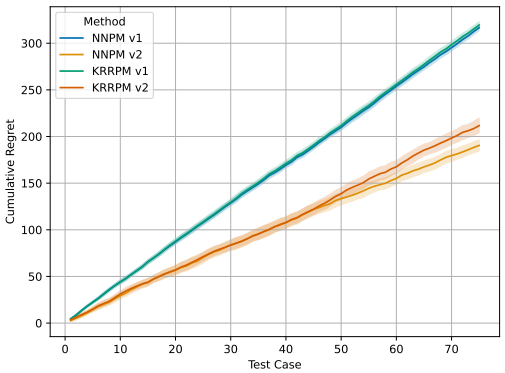
(a) Linear Dataset



(b) Friedman #1 Dataset

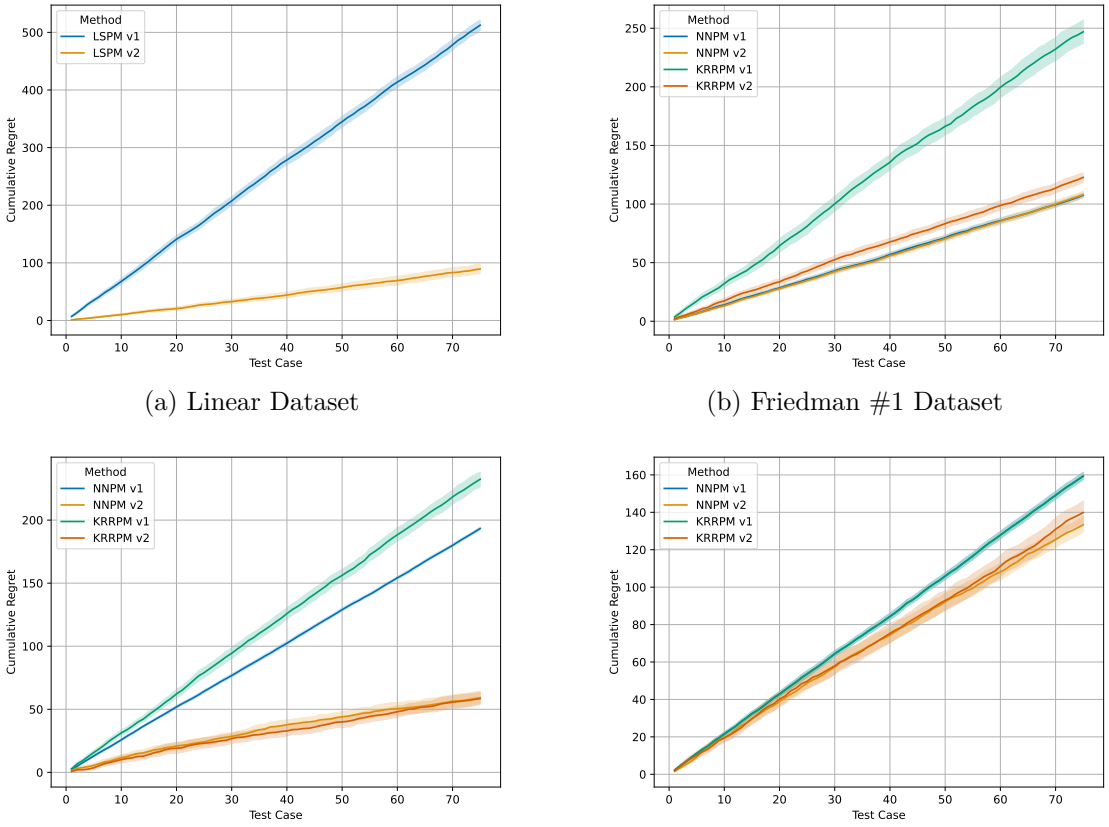


(c) Friedman #2 Dataset



(d) Friedman #3 Dataset

Figure A.1: Comparison of CPDM v1 and v2 in the online setting under Configuration #4.



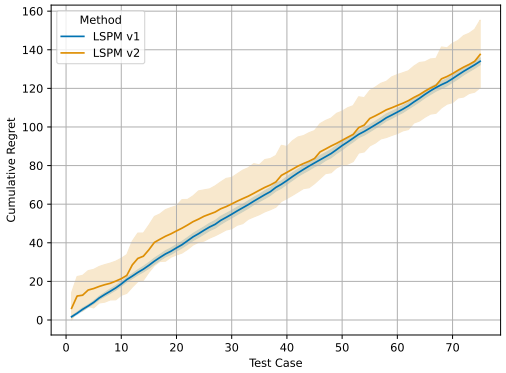
(a) Linear Dataset

(b) Friedman #1 Dataset

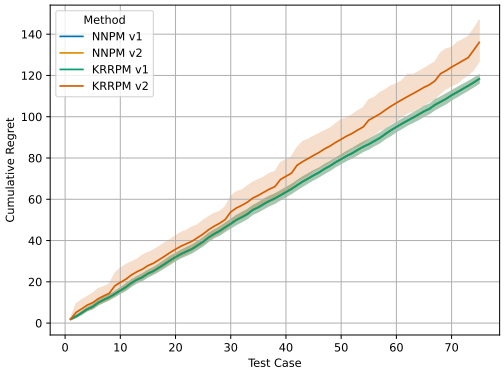
(c) Friedman #2 Dataset

(d) Friedman #3 Dataset

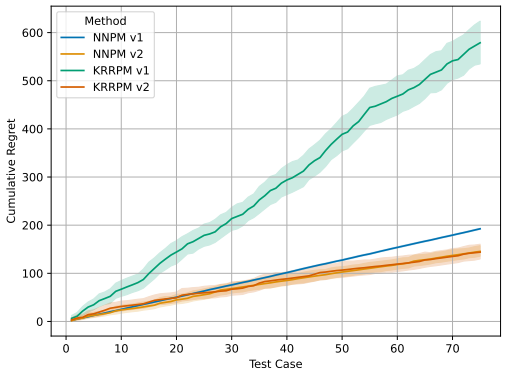
Figure A.2: Comparison of CPDM v1 and v2 in the online setting under Configuration #5.



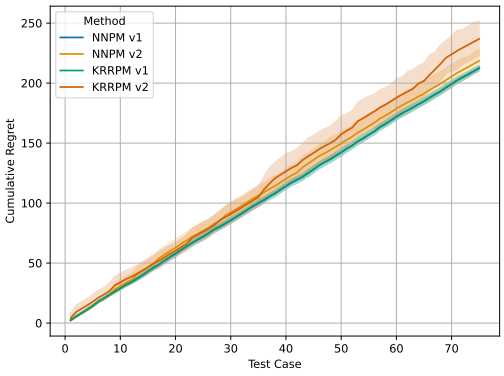
(a) Linear Dataset



(b) Friedman #1 Dataset

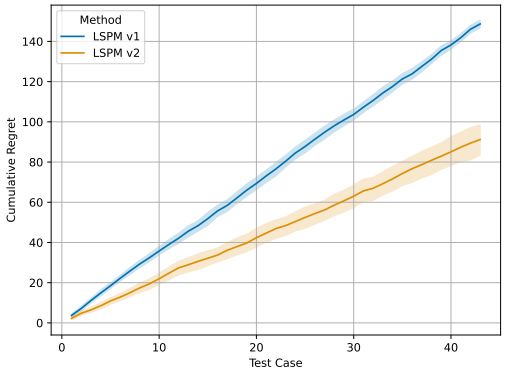


(c) Friedman #2 Dataset

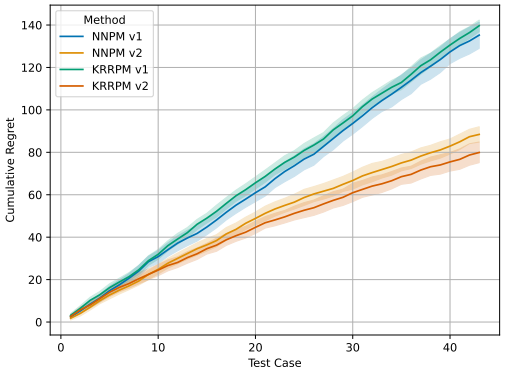


(d) Friedman #3 Dataset

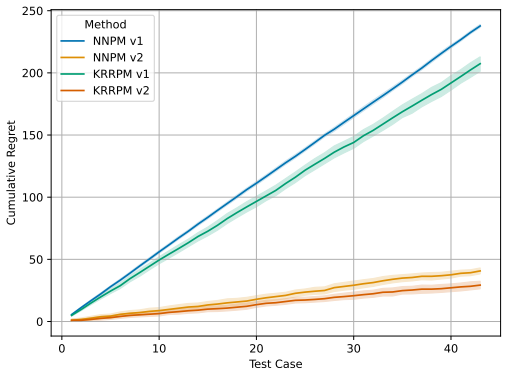
Figure A.3: Comparison of CPDM v1 and v2 in the online setting under Configuration #6.



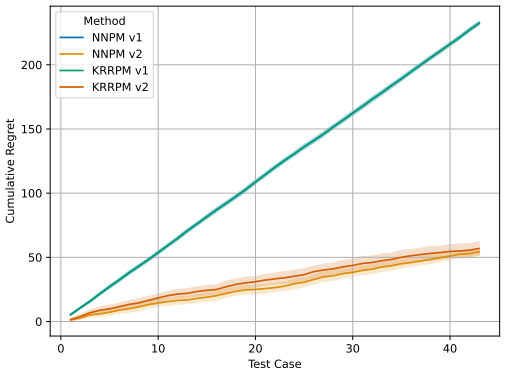
(a) Linear Dataset



(b) Friedman #1 Dataset

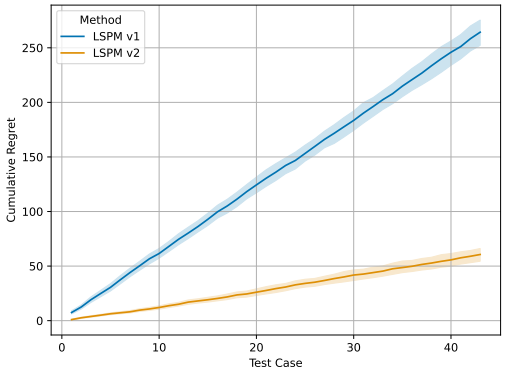


(c) Friedman #2 Dataset

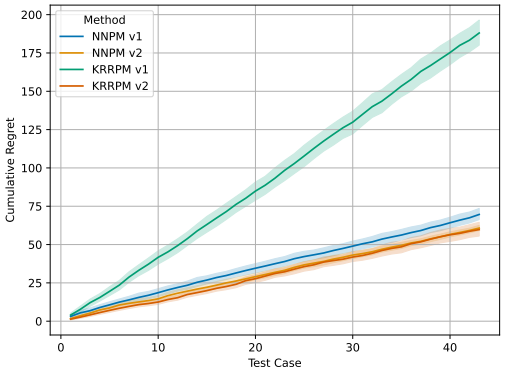


(d) Friedman #3 Dataset

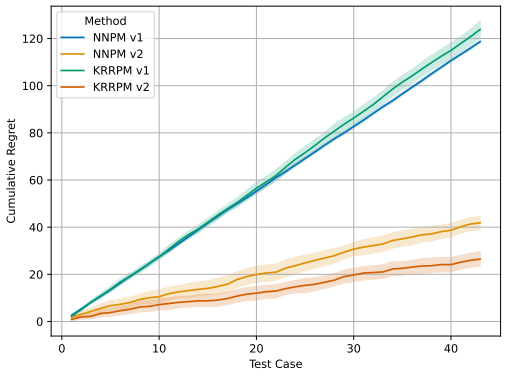
Figure A.4: Comparison of CPDM v1 and v2 in the online setting under Configuration #7.



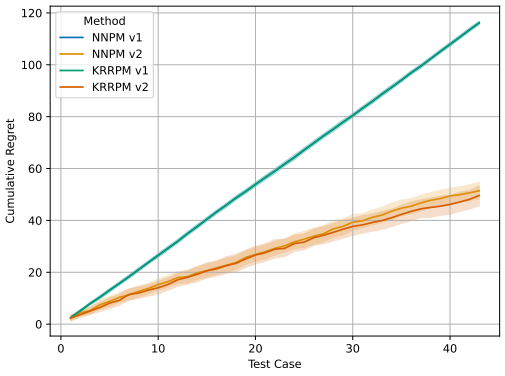
(a) Linear Dataset



(b) Friedman #1 Dataset

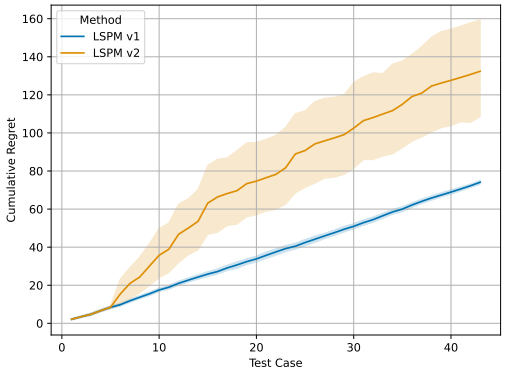


(c) Friedman #2 Dataset

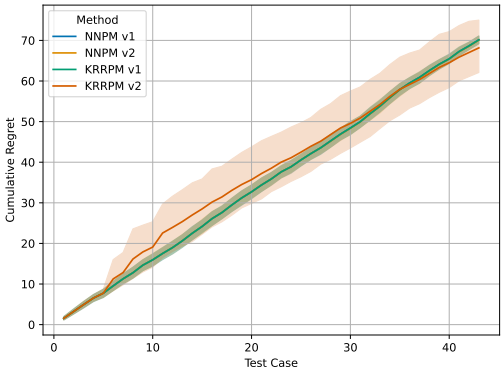


(d) Friedman #3 Dataset

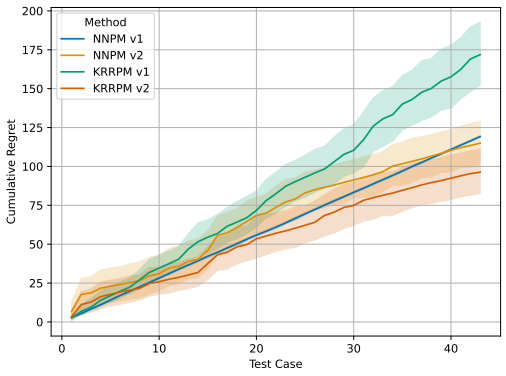
Figure A.5: Comparison of CPDM v1 and v2 in the online setting under Configuration #8.



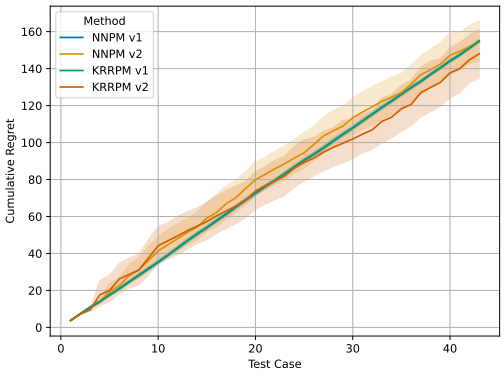
(a) Linear Dataset



(b) Friedman #1 Dataset

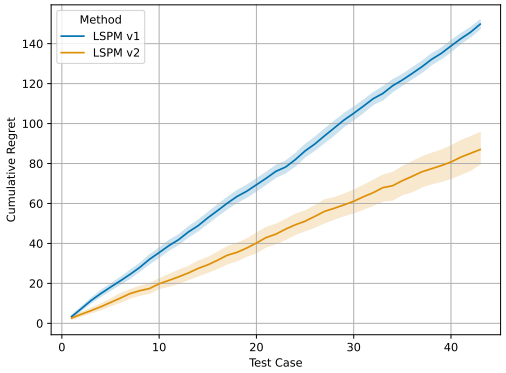


(c) Friedman #2 Dataset

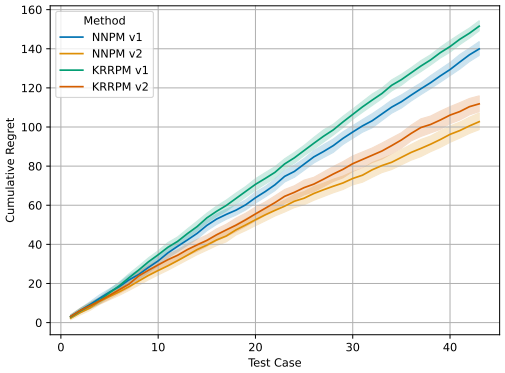


(d) Friedman #3 Dataset

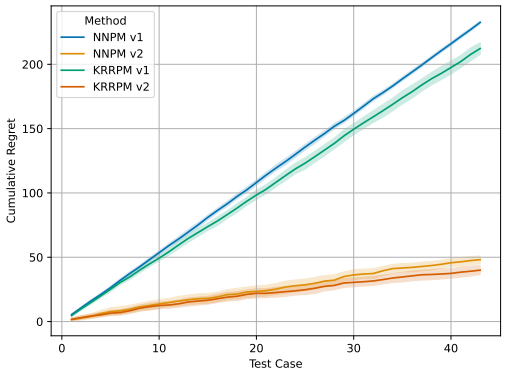
Figure A.6: Comparison of CPDM v1 and v2 in the online setting under Configuration #9.



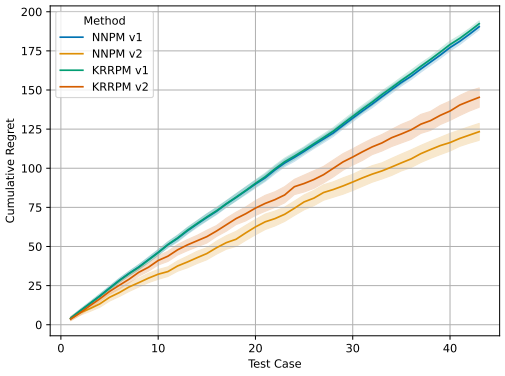
(a) Linear Dataset



(b) Friedman #1 Dataset

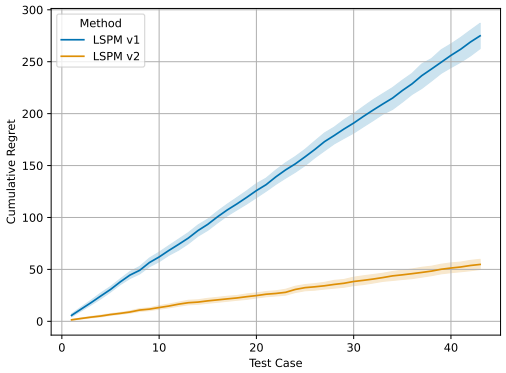


(c) Friedman #2 Dataset

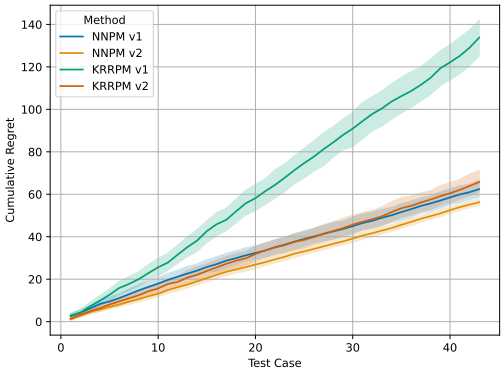


(d) Friedman #3 Dataset

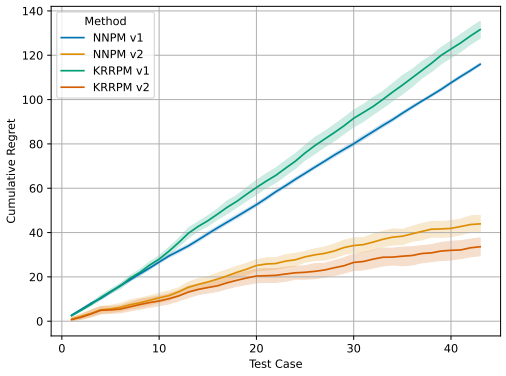
Figure A.7: Comparison of CPDM v1 and v2 in the online setting under Configuration #10.



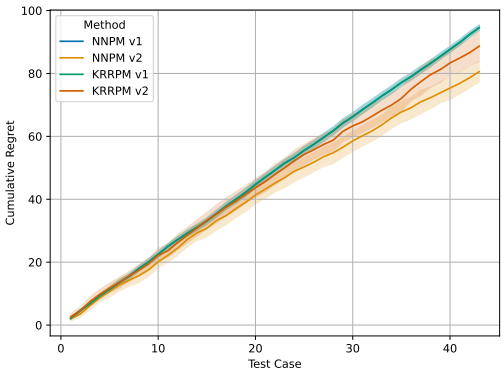
(a) Linear Dataset



(b) Friedman #1 Dataset

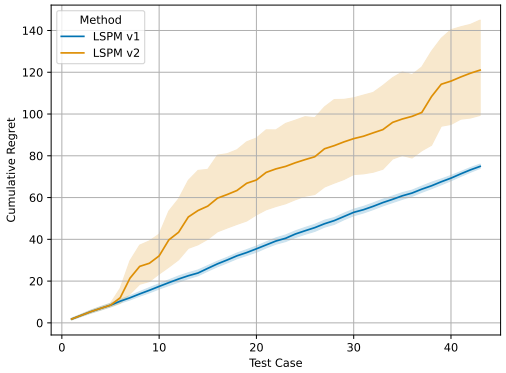


(c) Friedman #2 Dataset

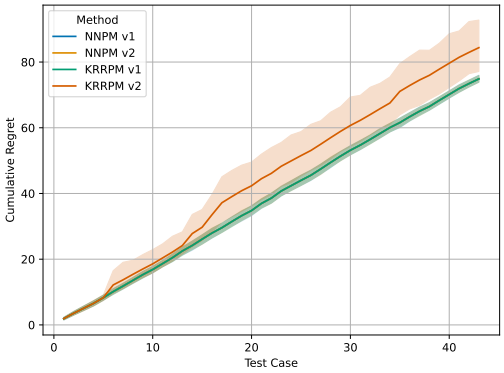


(d) Friedman #3 Dataset

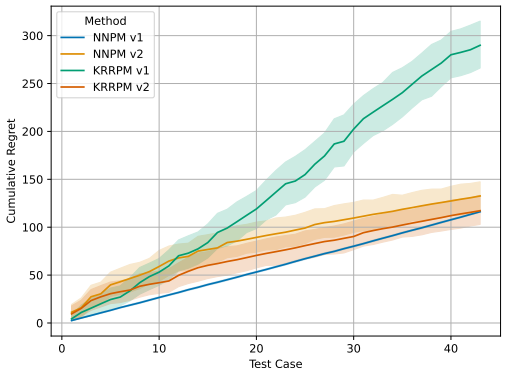
Figure A.8: Comparison of CPDM v1 and v2 in the online setting under Configuration #11.



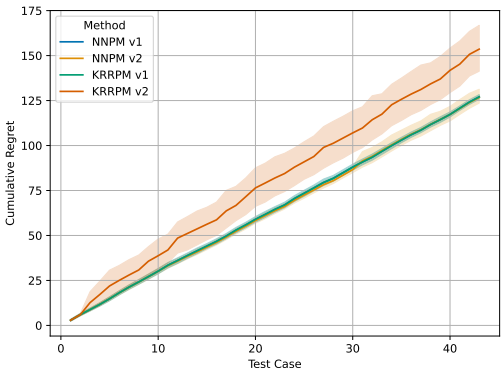
(a) Linear Dataset



(b) Friedman #1 Dataset



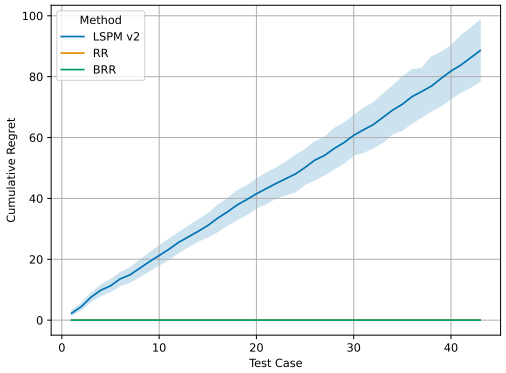
(c) Friedman #2 Dataset



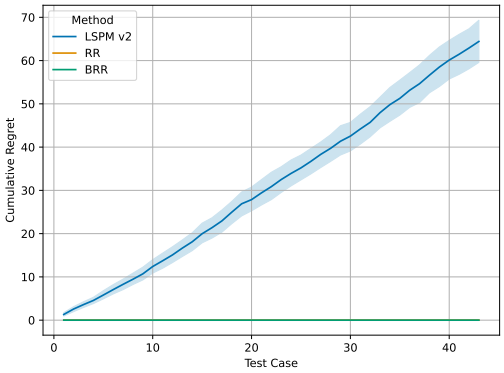
(d) Friedman #3 Dataset

Figure A.9: Comparison of CPDM v1 and v2 in the online setting under Configuration #12.

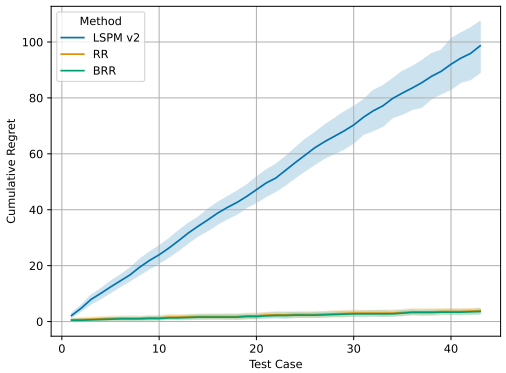
A.2 Online setting comparision of CPDM, BDT, and PPDM in limited data configurations



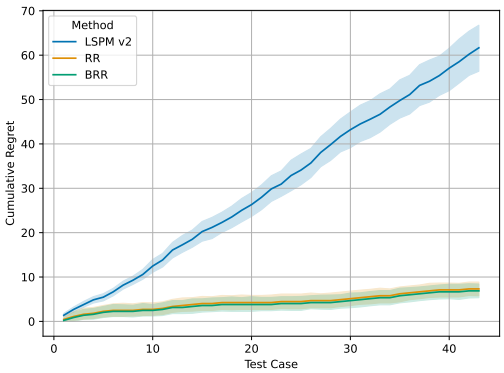
(a) Configuration #7



(b) Configuration #8

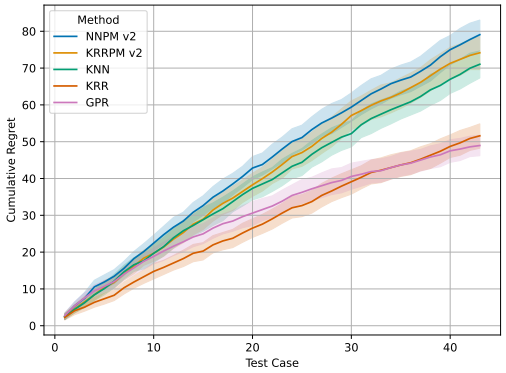


(c) Configuration #10

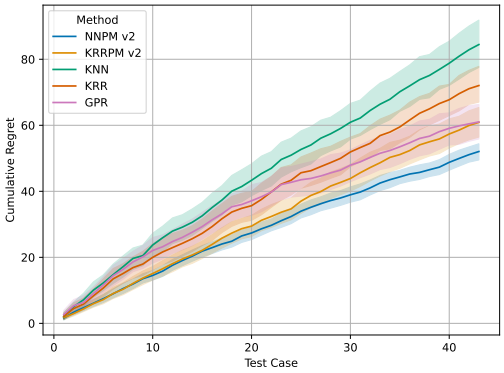


(d) Configuration #11

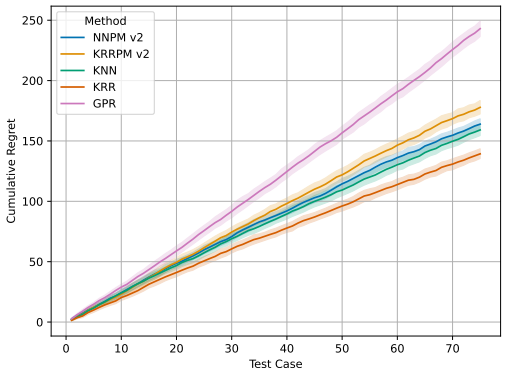
Figure A.10: Online setting comparison for the Linear dataset of CPDM v2, PPDM, and BDT methods under different noisy data configurations.



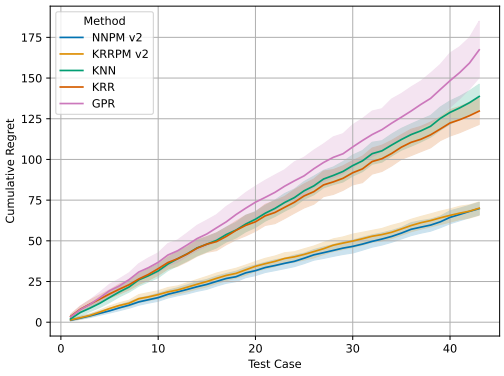
(a) Configuration #7



(b) Configuration #8

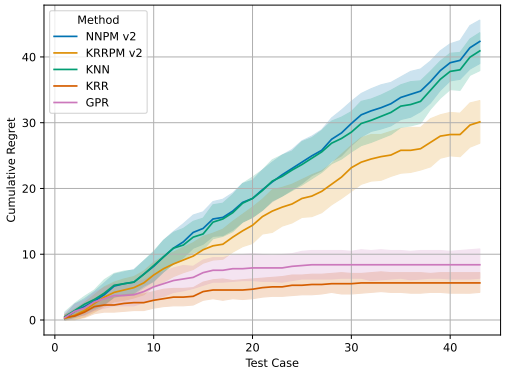


(c) Configuration #10

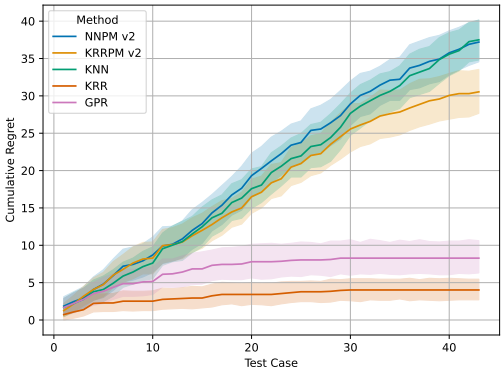


(d) Configuration #11

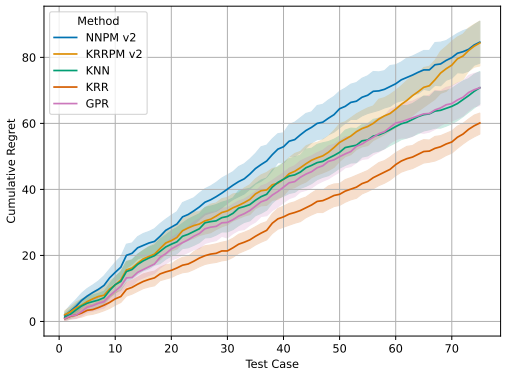
Figure A.11: Online setting comparison for the Friedman #1 dataset of CPDM v2, PPDM, and BDT methods under different noisy data configurations.



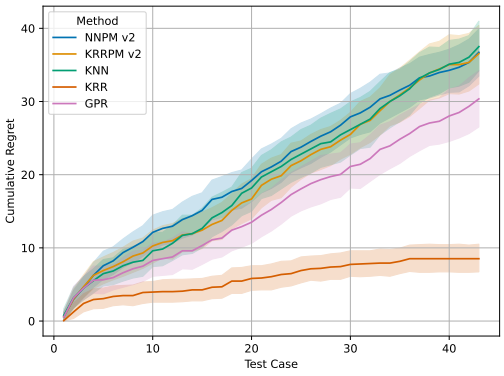
(a) Configuration #7



(b) Configuration #8

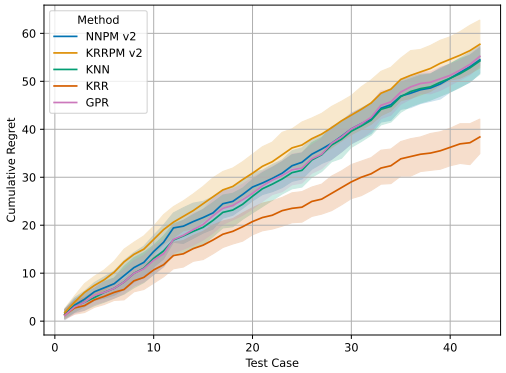


(c) Configuration #10

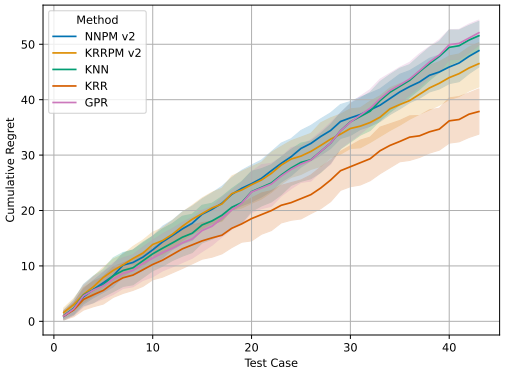


(d) Configuration #11

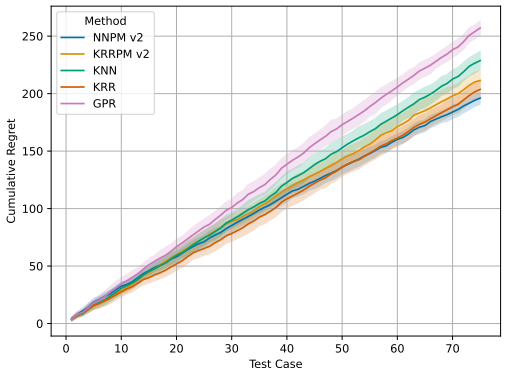
Figure A.12: Online setting comparison for the Friedman #2 dataset of CPDM v2, PPDM, and BDT methods under different noisy data configurations.



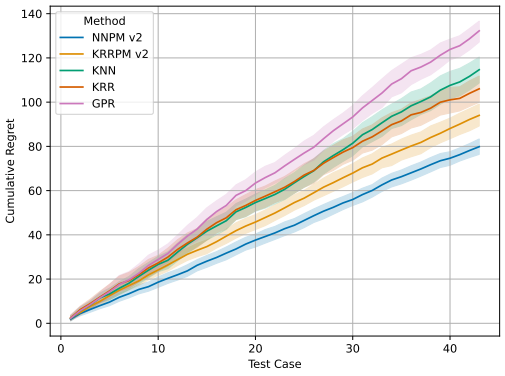
(a) Configuration #7



(b) Configuration #8



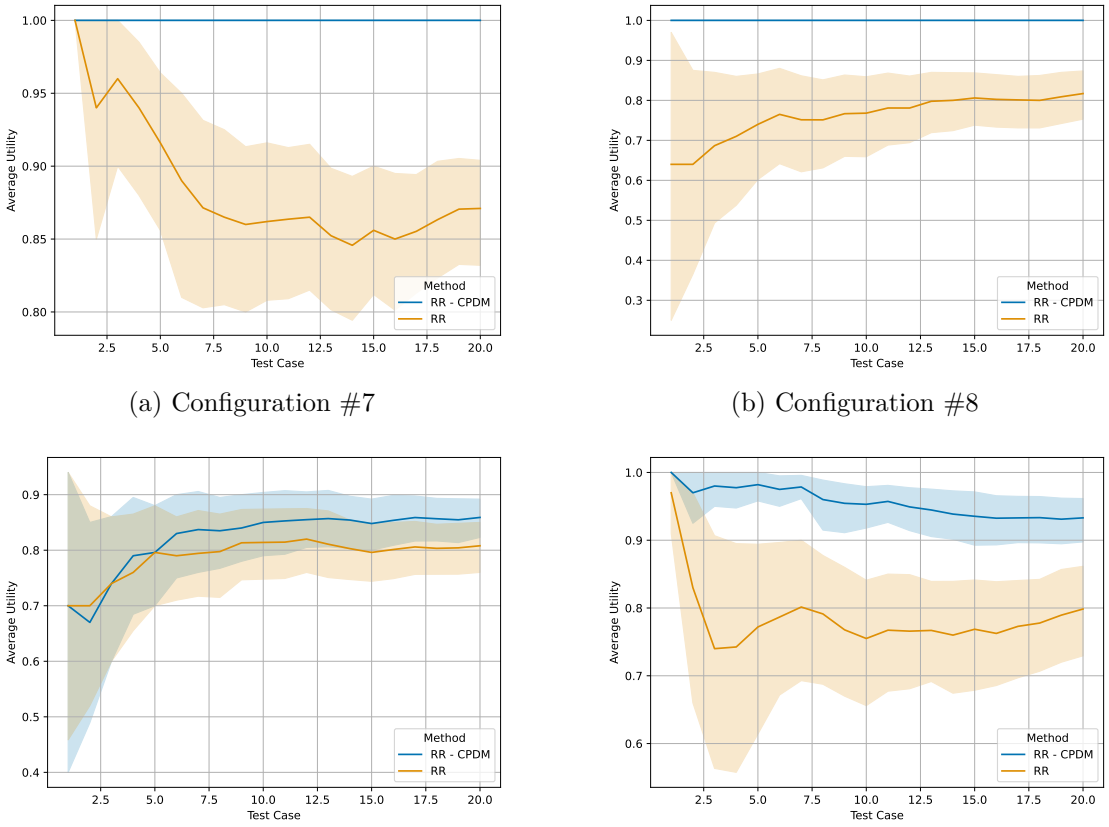
(c) Configuration #10



(d) Configuration #11

Figure A.13: Online setting comparison for the Friedman #3 dataset of CPDM v2, PPDM, and BDT methods under different noisy data configurations.

A.3 Inductive setting comparision of CPDM, BDT, and PPDM in limited data configurations



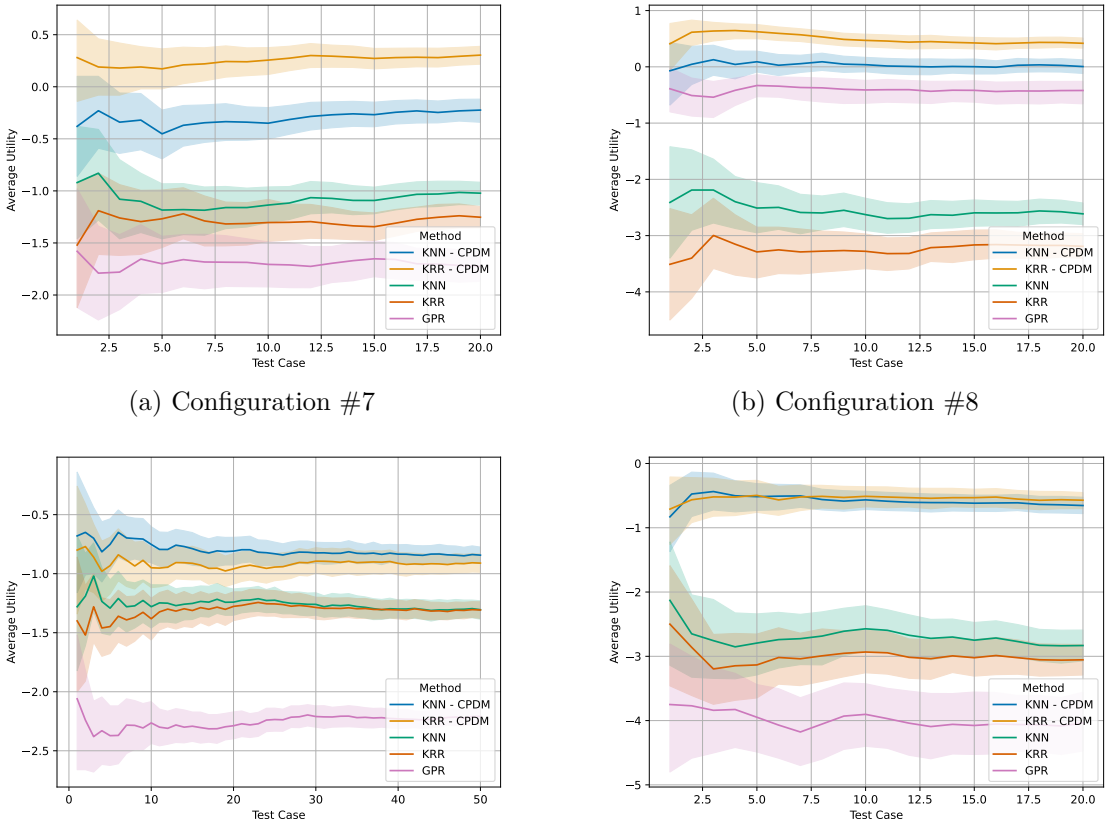
(a) Configuration #7

(b) Configuration #8

(c) Configuration #10

(d) Configuration #11

Figure A.14: Inductive setting comparison for the Linear dataset of CPDM, PPDM, and BDT methods under different noisy data configurations.



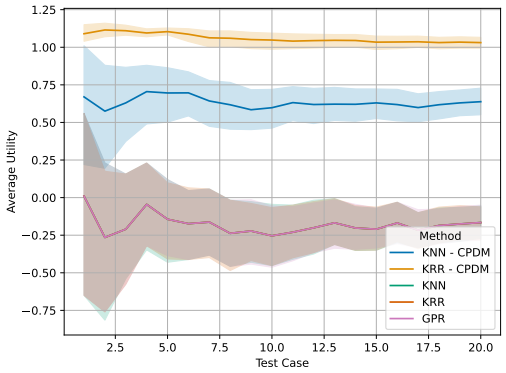
(a) Configuration #7

(b) Configuration #8

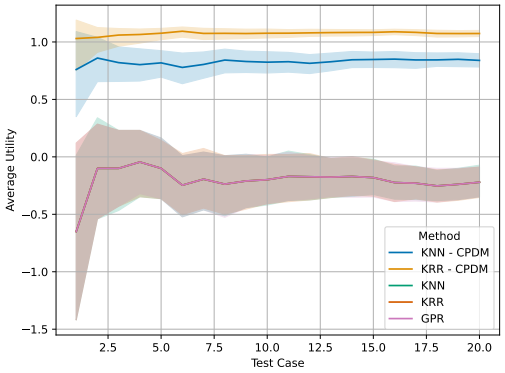
(c) Configuration #10

(d) Configuration #11

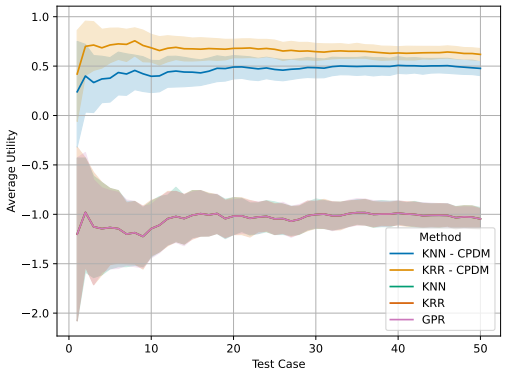
Figure A.15: Inductive setting comparison for the Friedman #1 dataset of CPDM, PPDM, and BDT methods under different noisy data configurations.



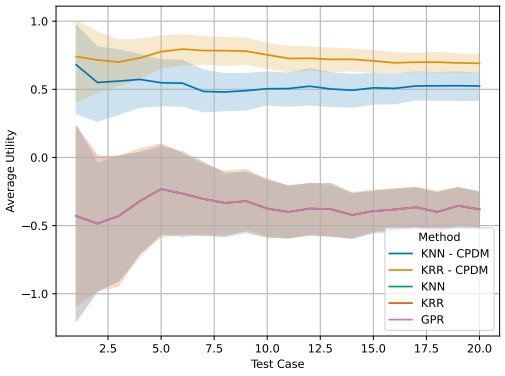
(a) Configuration #7



(b) Configuration #8

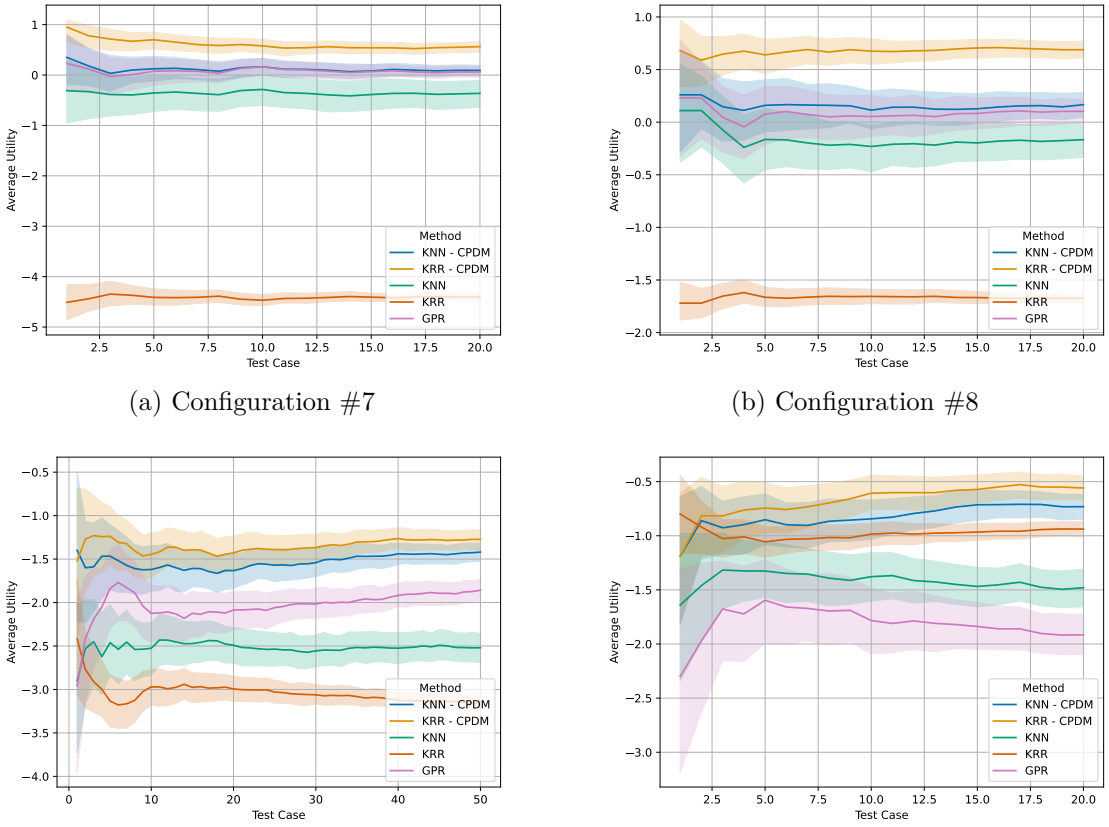


(c) Configuration #10



(d) Configuration #11

Figure A.16: Inductive setting comparison for the Friedman #2 dataset of CPDM, PPDM, and BDT methods under different noisy data configurations.



(a) Configuration #7

(b) Configuration #8

(c) Configuration #10

(d) Configuration #11

Figure A.17: Inductive setting comparison for the Friedman #3 dataset of CPDM, PPDM, and BDT methods under different noisy data configurations.