Fast Online Value-Maximizing Prediction Sets with Conformal Cost Control

Zhen Lin [1] **Shubhendu Trivedi** [2] **Cao Xiao** [3] **Jimeng Sun** [1 4]

Abstract

Many real-world multi-label prediction problems involve set-valued predictions that must satisfy specific requirements dictated by downstream usage. We focus on a typical scenario where such requirements, separately encoding *value* and *cost*, compete with each other. For instance, a hospital might expect a smart diagnosis system to capture as many severe, often co-morbid, diseases as possible (the value), while maintaining strict control over incorrect predictions (the cost). We present a general pipeline, dubbed as `FavMac`, to maximize the value while controlling the cost in such scenarios. `FavMac` can be combined with almost any multi-label classifier, affording distribution-free theoretical guarantees on cost control. Moreover, unlike prior works, it can handle real-world large-scale applications via a carefully designed online update mechanism, which is of independent interest. Our methodological and theoretical contributions are supported by experiments on several healthcare tasks and synthetic datasets - `FavMac` furnishes higher value compared with several variants and baselines while maintaining strict cost control. Our code is available at https://github.com/zlin7/FavMac

1. Introduction

Multi-label classification, which aims to predict a *set* of possible labels for a particular input, is an important task in many domains such as computer vision and healthcare. The possible set of output labels reflects some underlying structure of the world. For instance, in healthcare contexts, patients are likely to exhibit a set of (related) symptoms and co-morbid conditions at the same time. Generating a good prediction set that adequately captures the co-occurrence of such labels is a problem that goes well beyond the more customary multi-class setting where one simply outputs the most likely class. Much research effort has concentrated on estimating the conditional probability $\mathbb{P}\{Y_k = 1|X\}$, where X is the input, and Y_k is the indicator of class k. However, examining prediction set generation when tied to a concrete downstream task has received considerably less attention. This aspect is nonetheless significant in real-world decision-making pipelines where the notion of a good prediction set is also dictated by how it is to be harnessed, by the concrete cost associated with incorrect predictions, and what value the predicted set brings. Stated differently, given a base classifier $\hat{\mathbf{p}}$, we could measure metrics like mean average precision (mAP) and quantify the quality of our classifier, but when do we exactly predict $\hat{Y}_k = 1$?

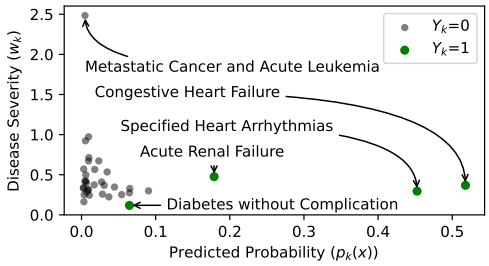

Figure 1. Severity vs the predicted probability of various diseases. Choosing a prediction set is a complex decision involving both. More details in Section 3.1.

The answer to this question depends on the goals and requirements of the specific downstream task under consideration. Often, these goals could be conflicting, have an attendant cost-value calculus, and thus could engender incurring necessary trade-offs. For example, in medical coding or disease detection tasks, we would like the prediction set to cover as many (true) disease diagnoses as possible for a patient, while also wanting to control the total number of incorrect diagnoses. Because the validation of any disease diagnosis may incur additional costs (e.g., expensive reviews) and (potentially invasive) unnecessary examinations. In such a use-case, a prediction set with more disease codes could possibly have higher *value* but also higher *cost*, which we

[1]Department of Computer Science, University of Illinois at Urbana-Champaign, USA [2]shubhendu@csail.mit.edu [3]Relativity [4]Carle Illinois College of Medicine, University of Illinois at Urbana-Champaign, USA. Correspondence to: Zhen Lin <zhenlin4@illinois.edu>, Jimeng Sun <jimeng@illinois.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

would like to control. Crucially, in such applications, we would also like to have provable guarantees on cost control.

Conformal prediction (CP) is a versatile and powerful framework to obtain coverage guarantees for prediction intervals or prediction sets. One of its most popular formulations, namely the split-conformal method, can be applied to black-box models such as large neural networks to furnish provable distribution-free finite-sample guarantees. Remarkably, the method only requires a bare-bones exchangeability assumption and a hold-out calibration set. A few recent works used CP techniques to construct prediction sets while controlling costs like the number of false positives (Bates et al., 2020; Fisch et al., 2022). However, the more general and practically relevant formulation involving the value-vs-cost trade-off remains unexplored. We explore such a formulation building on ideas from (Fisch et al., 2022).

Another challenge rarely addressed in prior works is the efficient online computation of conformal prediction. Most existing conformal prediction methods typically involve querying thresholds (e.g., quantiles) during inference. This becomes a problem in large-scale applications. For example, a health insurance company could have hundreds of millions of records, making real-time vanilla quantile queries infeasible. Such applications require additional design considerations to improve feasibility. Furthermore, in the healthcare context specifically, such a challenge is also inherent in the structure of the problem for reasons that go beyond the scale. In applications such as disease diagnosis and medical coding, we make predictions in a dynamic setting. The data for the general patient population is also constantly updating, thus changing the data distribution and the quantiles.

In this paper, we present a general and efficient framework to *maximize the value* associated with the prediction sets, while providing *provable cost control*. As part of our solution for large-scale settings, we also propose `QuantileTree`, an efficient data structure specifically designed to track and query *weighted quantiles*. Given the prevalence of weighted quantiles in the conformal prediction literature (Bates et al., 2020; Tibshirani et al., 2020; Guan, 2022), `QuantileTree` could be of independent interest to machine learning practitioners.

Summary of Contributions:

- We propose `FavMac`, or Fast online value-Maximizing prediction sets with conformal cost-control, where both the notions of value and cost are user-defined. To the best of our knowledge, `FavMac` is the first method to provide *expected cost control* and *violation risk control* for any *continuous* cost function.
- We recast the task as online weighted quantile computation, and propose an efficient new data structure (christened `QuantileTree`), reducing the complexity by a factor of N. `QuantileTree` can also be plugged into other (conformal prediction) contexts where an empirical distribution function (eCDF) is used.
- We conduct extensive experiments and ablation studies on several real-world healthcare problems to verify the effectiveness of `FavMac` over competing methods.

2. Related Works

Uncertainty Quantification: Uncertainty quantification (UQ) for modern machine learning models has attracted a lot of recent research attention. In the context of classification, one major task (especially for deep neural networks) is calibrating the predicted conditional probability (Guo et al., 2017; Murphy & Winkler, 1984; Vaicenavicius et al., 2019). Another major line of research uses prediction sets to encode predictive uncertainty (Sadinle et al., 2019; Zaffalon, 2002; Corani & Zaffalon, 2008). Classification with rejection (Hendrickx et al., 2021; Lin et al., 2022) could be viewed as a special case of such research since rejections are similar to an uninformative prediction set $\mathcal{Y}$. From the perspective of UQ, our work can be viewed as calibrating a cost function at the level of a prediction set.

Conformal Prediction: Conformal Prediction (CP) is a powerful tool that is able to provide finite-sample coverage guarantees (Vovk et al., 2005). In the context of (the more widely studied) multi-class classification, the typical guarantee is $\mathbb{P}\{Y \in S\} \geq 1 - \epsilon$, and the performance metric (the "value") is efficiency, or $\mathbb{E}[|S|]$ (Sadinle et al., 2019; Angelopoulos et al., 2021b). (Cauchois et al., 2022) study CP in the multi-label case, but unlike our task, it targets the coverage of all labels (namely zero cost). Our work (especially Theorem 4.6) should be considered a form of CP. Moreover, `QuantileTree` is applicable to other CP methods in an online setting.

Risk Controlling Prediction Sets: A few recent works on risk-controlling prediction sets are highly related to our work. (Fisch et al., 2022) define and propose a method for the false-positive control problem. Our work can be viewed as a generalization of (Fisch et al., 2022) with new challenges from online updates, value-cost trade-off, and more general cost functions. (Bates et al., 2020; Angelopoulos et al., 2021a) also perform violation control for general cost functions, but they focus on the *population* cost, providing a different guarantee than ours[1]. Another major difference is that these works do not have an explicit value-cost trade-off. We control the cost via an upper-bound[2] that creates a total ordering of all candidate sets (an equivalent form of which was first defined in (Fisch et al., 2022)) with nested level-

[1]The difference is similar to that between a confidence interval and a prediction interval.

[2]C^+, defined in Section 4.3.

sets, which is inspired by nested CP (Gupta et al., 2022). Nevertheless, nested CP focuses on covering the target Y entirely (i.e. the special case of zero cost).

Set-based Optimization: Our fundamental goal is to find prediction sets that have downstream task-dictated desirable properties. A lot of papers directly maximize a set-based utility function encoding a trade-off between value and cost, including F_β-measure (del Coz et al., 2009), discounted accuracy (Corani & Zaffalon, 2009), $\frac{1}{x}$-convex functions (Mortier et al., 2021) and many others (Zaffalon et al., 2012; Yang et al., 2017; Nguyen et al., 2018). In reality, it is hard to pinpoint a utility function that precisely specifies the trade-off, as reflected by the number of different choices. By decoupling the trade-off into value and cost functions, our setup avoids such ambiguities and permits explicit cost control [3] with theoretical guarantees for high-stakes applications. Set-based optimization also appears in combinatorial game theory problems like auction or allocation of indivisible goods (e.g. (Aziz et al., 2020; Igarashi & Peters, 2019)), which motivated our terminology of "value" and "cost" as our problem could be viewed as allocating classes to two agents - "predict" and "not predict". An obvious challenge is the combinatorially-hard integer programming for optimizing over general set functions. While a general panacea likely does not exist, techniques exist to simplify the search for special classes of functions, such as those in (Mortier et al., 2021), the (common) additive functions in this work, or the more general sub-modular functions for which submodular optimization (Krause & Golovin, 2014) could be useful. Orthogonal to our primary focus on conformal cost control, these strategies could however benefit from the prediction set selection detailed in Section 4.5.

Quantile Computation: A byproduct of this work is a data structure (`QuantileTree`) that speeds up a key online quantile query step (common in CP methods). Quantile computation/estimation has a long history in data science. However, most works study *approximate* quantile computation, with a focus on accuracy, space and memory efficiency, in the context of streaming or distributed computing (Chen & Zhang, 2020; Zhang & Wang, 2007; Cormode et al., 2005). Moreover, most do not estimate weighted quantiles, except the weighted quantile sketch in (Chen & Guestrin, 2016). In contrast, `QuantileTree` computes *exact weighted* quantiles, and can be used to represent any finite distribution.

3. Problem Setup

In this section, we formalize our problem with necessary notation and definitions. Recall our multi-label classification task with input $X \in \mathcal{X}$, and labels $Y \subseteq \mathcal{Y} = [K]$ (represented as K-dimension multi-hot vectors). We assume that a (black-box) base multi-label classifier $\hat{\mathbf{p}}$ is already trained, on a separate training split, with $\hat{p}_k(x)$ predicting $\mathbb{P}\{Y = k|X = x\}$. We make no assumptions about the underlying joint distribution of $Z = (X, Y)$, except for the usual *exchangeability assumption* on test set $[Z_1, Z_2, \ldots]$:

Definition 3.1. (**Exchangeability** (Vovk et al., 2005)) A sequence of random variables, $Z_1, Z_2, \ldots, Z_n \in \mathcal{Z}$ are exchangeable if for any permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$, and every measurable set $E \subseteq \mathcal{Z}^n$, we have

$$\mathbb{P}\{(Z_1, \ldots, Z_n) \in E\} = \mathbb{P}\{(Z_{\pi(1)}, \ldots, Z_{\pi(n)}) \in E\}. \quad (1)$$

Note that although we operate in an online setting, this assumption is reasonable in many settings where we can believe that the underlying distribution (e.g. the patient distribution) does not change. The theoretical results in this paper will be based on Definition 3.1. However, they can be used in consonance with recent works (Barber et al., 2022; Gibbs & Candès, 2022; Gibbs & Candes, 2021) to handle scenarios with mild distributional shifts as well.

Our goal is to find a good prediction set S_{N+1} that maximizes a value metric $V(S_{N+1}; Y_{N+1})$ while controlling cost $C(S_{N+1}; Y_{N+1})$ (both user-defined) with provable guarantees. We assume that both V and C and bounded and take values between $[0, C_{max}]$ and $[0, V_{max}]$, for some constants C_{max} and V_{max}, respectively, a mild assumption given the finite domains. We assume that both V and C are *monotone* set functions:

Definition 3.2. A function $f(S)$ mapping a set in $2^{\mathcal{Y}}$ to $\mathbb{R}$ is *monotone* if $\forall S, S', S \subseteq S' \implies f(S) \leq f(S')$.

Definition 3.2 is very common for value functions in game theory (e.g. in fair allocation) (Brustle et al., 2020; Balcan et al., 2012). The monotone requirement can be removed without affecting our general algorithmic procedures, but it keeps the cost-value trade-off more intuitive. Finally, unlike (Karimi et al., 2020; Gendler et al., 2022), we assume any observation Y is non-noisy.

For cost control, there could be two possible goals: Controlling the **expected cost**, or controlling the probability of a **cost violation**, which we will henceforth refer to as "Expected Cost Control" and "Violation Control" respectively:

Definition 3.3. (Expected Cost Control) Given a target cost c, the expected cost control problem is:

$$\max_{S_{N+1} \in 2^{\mathcal{Y}}} \quad \mathbb{E}[V(S_{N+1}; Y_{N+1})] \quad (2)$$

$$\text{s.t.} \quad \mathbb{E}[C(S_{N+1}; Y_{N+1})] \leq c \quad (3)$$

Definition 3.4. (Violation Control) Given a target cost c and a violation probability target δ, the violation control problem is defined as:

$$\max_{S_{N+1} \in 2^{\mathcal{Y}}} \quad \mathbb{E}[V(S_{N+1}; Y_{N+1})] \quad (4)$$

$$\text{s.t.} \quad \mathbb{P}\{C(S_{N+1}; Y_{N+1}) > c\} \leq \delta \quad (5)$$

[3] Despite the name, *cost-sensitive learning* (Elkan, 2001) is similar to utility maximization without prediction sets: The trade-off is typically encoded in one objective function to minimize.

Remarks: Our notion of violation control (Definition 3.4) looks similar to that of risk control defined in (Angelopoulos et al., 2021a; Bates et al., 2020). However, apart from the additional goal to maximize value, there is a crucial difference: The equivalent of "cost" in their case is an *expectation* on the *population*, whereas Definition 3.4 bounds the cost on one instance (which has a large variance). Definitions 3.3 and 3.4 can also be viewed as generalizations of (Fisch et al., 2022) to general cost functions (instead of false positives), along with the additional value-maximization goal.

3.1. Examples of Cost and Value Function

Next, we present concrete examples of cost and value functions to ground our discussion. The most straightforward examples for value and cost could be the number of True or False Positives in the prediction set. Formally, we have:

$$V_{TP}(S; y) := |\{k : k \in S, k \in y\}| \tag{6}$$
$$C_{FP}(S; y) := |\{k : k \in S, k \notin y\}|. \tag{7}$$

Here, both V_{TP} and C_{FP} takes values in $\{0, 1, \ldots, K\}$. In many applications, however, this might not be enough.

Table 1. Notations used in this paper.

Symbol	Meaning
X, Y, Z	Random input X and label set Y; $Z = (X, Y)$.
$\mathcal{Y} = [K]$	The set of all K classes
$\hat{\mathbf{p}}$, $\mathbf{v}$, $\mathbf{r}$ v.s. $\hat{p}_k$, v_k, r_k	vector v.s. its k-th entry
$S \in 2^{\mathcal{Y}}$	Prediction set
$\mathcal{U}(X) \subseteq 2^{\mathcal{Y}}$	Universe to choose the prediction sets from
$V(S; Y)$	Value of set S (to maximize)
$C(S; Y) \in [0, C_{max}]$	Cost of set S (to control)
$\hat{C}(S; X)$, $\hat{V}(S; X)$	Cost and value proxy (estimates for C and V)
$C^+(t; Z)$	Maximum cost with proxy $\hat{C} < t$
$\hat{\mathbf{p}}(x) = [\hat{p}_k(x)]_{k=1}^K$	Probability prediction by the base classifier

Example in hospital coding: For a hospital quality-assurance system detecting missing diagnoses, S could be a set of disease codes. Diseases are typically grouped into hierarchical condition categories (HCC) codes, and assigned morbidity scores (risk adjustment factors). Some diseases are obviously more severe than others. As illustrated in Fig. 1, Metastatic Cancer and Acute Leukemia is 25+ times more severe than Diabetes without Complication[4]. The severity can be encoded in V or C via weights w_k:

$$V_{TPC}(S; y) := \sum_{k \in S} \mathbb{1}\{y_k = 1\} w_k \tag{8}$$
$$C_{FPC}(S; y) := \sum_{k \in S} \mathbb{1}\{y_k = 0\} w_k. \tag{9}$$

We refer to $V_{TP\underline{C}}$ and $C_{FP\underline{C}}$ as "continuous" functions[5]. C_{FPC} also has applications in health insurance plans and could be interpreted as incorrectly billed conditions, as w_k roughly corresponds to expected cost.

[4]According to the 2023 Midyear "Community NonDual Aged" factors released on `www.cms.gov`.

[5]Their ranges are technically discrete, but close to continuous for any practical value of K: There are 35 codes in `MIMIC`, so a `float32` number cannot encode all values of V_{TPC} and C_{FPC}.

4. Methodology

In this section, we lay out the exact methodology and the attendant theory for cost control, using ideas from conformal prediction. We will also describe how to improve the value of the final chosen prediction set. The computational details for efficient online updates are deferred to Section 5. The list of major notations can be found in Table 1.

4.1. Preliminary: Split Conformal False Positive Control

We begin by introducing (split) conformal prediction and showing how to use it to conservatively control C_{FP}. However, we first need to define the notion of a quantile function:

Definition 4.1. (Quantile Function Q) The β-quantile for a set of values $\{v_i\}_{i=1}^N$ is defined by the empirical CDF F:

$$Q(\beta; F) := \inf\{t : F(t) \geq \beta\} \text{ where } F(t) := \sum_{i=1}^{N} \frac{\Delta_{v_i}(t)}{N} \tag{10}$$

Here, $\Delta_v(t)$ is the indicator function $\mathbb{1}\{t \geq v\}$. When possible, we write $Q(\beta; \{v_i\}_{i=1}^N)$ for readability.

Theorem 4.2. *(Split Conformal Prediction (Papadopoulos et al., 2002)) Assuming exchangeability of $[Z_1, \ldots, Z_{N+1}]$, for a fixed measure g and any $\epsilon \in (0, 1)$:*

$$\mathbb{P}\Big\{g(Z_{N+1}) \leq Q\Big(1 - \epsilon; \{g(Z_i)\}_{i=1}^N \cup \{\infty\}\Big)\Big\} \geq 1 - \epsilon \tag{11}$$

Eq. (11) can be better understood as a statement on *rankings*. That is, due to exchangeability, the ranking of $g(Z_{N+1})$ among $\{g(Z_i)\}_{i=1}^{N+1}$ follows a uniform distribution.

Suppose for a moment that our goal is to control the false positive rate for class k only. We could focus on only the subpopulation where $k \notin Y$. If we define the *nonconformity score* for Z_i as $g(Z_i) := p_k(X_i)$, (Lei, 2014) gives us:

Theorem 4.3. *Assuming $[Z_1, \ldots, Z_{N+1}]$ are i.i.d., let*

$$k \in \hat{S}_{N+1} \Leftrightarrow p_k(x_{N+1}) > t_k \tag{12}$$
$$\text{s.t. } t_k = Q(1 - \epsilon; \{p_k(x_i) : y_{i,k} = 0\} \cup \{\infty\}). \tag{13}$$

Then, we have $\mathbb{P}\{k \in \hat{S}_{N+1} | k \notin Y_{N+1}\} \leq \epsilon$.

Theorem 4.3 can be used to conservatively control C_{FP}:

Theorem 4.4. *(The `ClassWise` method) Assuming $[Z_1, \ldots, Z_{N+1}]$ are i.i.d., let $\epsilon = \frac{c}{K}$ and S_{N+1} be defined like in Theorem 4.3, then $\mathbb{E}[C_{FP}(\hat{S}_{N+1}, Y_{N+1})] \leq c$.*

We defer all proofs to the Appendix. The intuition for `ClassWise` is that we evenly distribute the "cost budget" to each class, so in expectation the total cost $\leq c$. While

Theorem 4.4 does the job to control C_{FP}, it is conservative as long as $\exists k, \mathbb{P}\{k \in Y\} > 0$. Moreover, it cannot handle other cost functions and lacks a notion of value. Next, we will present `FavMac` to overcome these limitations.

4.2. Proxy Functions

In Section 4.1, we chose a threshold t_k for $1 - p_k(X)$ based on $\{Z_i\}_{i=1}^N$ to make a prediction for X. For more general cost control problems, we can find a proxy (or estimate) $\hat{C}$ for our cost function C. In many cases, $\hat{C}$ can be derived from the base prediction $\hat{\mathbf{p}}$. For instance, for C_{FP} and C_{FPC}, we could have the following cost proxies:

$$\hat{C}_{FP}(S; x) := \sum_{k \in S} (1 - \hat{p}_k(x)) \tag{14}$$

$$\hat{C}_{FPC}(S; x) := \sum_{k \in S} (1 - \hat{p}_k(x)) w_k. \tag{15}$$

For a generic, non-additive set function C, one possible proxy could be a Monte-Carlo estimate:

$$\hat{C}_{MC}(S; x) := \hat{\mathbb{E}}_{y \sim \hat{\mathbf{p}}(x)}[C(S; (x, y))]. \tag{16}$$

The value proxy $\hat{V}$ can be defined analogously to $\hat{C}$. Note that the sampling in Eq. (16) is trickier than it might seem because we cannot derive class correlations from $\hat{\mathbf{p}}$. One could also potentially estimate such correlational information, or directly train a $\hat{C}$ on the training set. Next, the cost proxy will be used to select the acceptable prediction sets.

4.3. The Expected Cost Control

To proceed, we fix a $\mathcal{U}(X) \subseteq 2^{\mathcal{Y}}$, the universe of all label sets that we would want to consider. For example, when $\mathcal{Y}$ is small, we could simply enumerate $\mathcal{U}(X) = 2^{\mathcal{Y}}$. The choice of $\mathcal{U}$ is discussed in Section 4.5. For both expected cost and violation control, we will create a total ordering of the subsets of $\mathcal{U}$, which allows us to pick a threshold like before to keep the "acceptable" sets. The rest of section 4, from here, elucidates these details and theoretical guarantees.

Define the maximum cost corresponding to a threshold t as:

$$C^+(t; z) := \max_{S \in \mathcal{U}(x)} \{C(S; z) : \hat{C}(S; x) < t\}. \tag{17}$$

Now, we create a total ordering via C^+, allowing us to pick a threshold T_c to control C^+ (thus C):

$$T_c = \sup \left\{ t : \frac{C_{max} + \sum_{i=1}^N C^+(t; Z_i)}{N+1} \leq c \right\} \tag{18}$$

Finally, we simply pick the expected value-maximizing set:

$$\hat{S}_{N+1} := \underset{S \in \mathcal{U}(x_{N+1}) : \hat{C}(S; x_{N+1}) < T_c}{\arg\max} \hat{V}(S; x_{N+1}). \tag{19}$$

We get a cost control guarantee similar to that in (Fisch et al., 2022), first proven in (Angelopoulos et al., 2021c):

Theorem 4.5. *Assuming exchangeable $\{Z_i\}_{i=1}^{N+1}$. For any $c \in (0, C_{max}]$, and $\hat{S}_{N+1}$ from Eq. (19):*

$$\mathbb{E}_{Z_1, \ldots, Z_{N+1}}[C(\hat{S}_{N+1}; Z_{N+1})] \leq c \tag{20}$$

4.4. The Violation Control

Analogously to the above, for the violation control, we could have the threshold $T_{c,\delta}$ and prediction set $\hat{S}_{N+1}$ as:

$$T_{c,\delta} = \sup \left\{ t : \frac{\sum_{i=1}^N \mathbb{1}\{C^+(t; Z_i) \leq c\}}{N+1} \geq 1 - \delta \right\}, \tag{21}$$

$$\hat{S}_{N+1} := \underset{S \in \mathcal{U}(X_{N+1}) : \hat{C}(S; X_{N+1}) < T_{c,\delta}}{\arg\max} \hat{V}(S; X_{N+1}). \tag{22}$$

Theorem 4.6. *Assuming exchangeable $\{Z_i\}_{i=1}^{N+1}$. For any $c \in (0, C_{max}]$, $\delta \in (0, 1)$, and $\hat{S}_{N+1}$ from Eq. (22):*

$$\mathbb{P}_{Z_1, \ldots, Z_{N+1}}\{C(\hat{S}_{N+1}; Z_{N+1}) \leq c\} \geq 1 - \delta \tag{23}$$

Note that the randomness is taken over $Z_1, \ldots, Z_{N+1}$. In particular, it means the guarantees hold with respect to a random X_{N+1}, and is only marginal. The proofs for both Theorem 4.5 and Theorem 4.6 (see Appendix) simply leverage the exchangeability of $\{Z_i\}_{i=1}^{N+1}$.

4.5. Value-Maximization via the Choice of $\mathcal{U}$

The choice of $\mathcal{U}$ significantly impacts performance. To see why, suppose $\mathcal{U}(X)$ is an uninformative random subset of $2^{\mathcal{Y}}$ that does not depend on X. Since we only pick one element $\hat{S} \in \mathcal{U}$, but perform the cost control on the entire $\{S \in \mathcal{U} : \hat{C}(S; X) \leq T\}$ (see Eq. (19)), the gap between C^+ and $C(\hat{S})$ will be larger as $|\mathcal{U}|$ increases, leading to more conservative $\hat{S}$. On the flip side, as $\mathcal{U}$ expands, it also could potentially include a set with higher expected value. Using the information from X (and our knowledge of the value and cost functions) could potentially achieve higher value while satisfying cost control requirements.

The Full Universe A simple starting point, as suggested earlier, is $\mathcal{U}(X) = 2^{\mathcal{Y}}$. We denote this as $\mathcal{U}_{full}$. Clearly, $\mathcal{U}_{full}$ is only practical when K is small, as $|\mathcal{U}| = 2^K$. We consider more practical universes $\mathcal{U}$ below.

As many real-world value/cost functions are additive, we discuss the additive case before extending to the general case.

The Greedy Sequence (Additive) When K is large, we could employ a different strategy and add one class at a time and create a sequence of sets, by having $\mathcal{U}(x) = \{S_{(j)}(x)\}_{j=0}^K$ with $S_{(0)}(x) = \emptyset$ and

$$S_{(j+1)}(x) = S_{(j)}(x) \cup \{\sigma(j+1; x)\} \tag{24}$$

where σ is an ordering of $[K]$ that depends on x. A recent work on *false positive control* (Fisch et al., 2022) proposes to greedily add classes with the highest predicted probability:

$$\sigma_{\texttt{Prob}}(x) := argsort(-\hat{\mathbf{p}}(x)) \tag{25}$$

This happens to be suitable given their particular goal: Maximizing the true positive rate (TPR). To extend this idea, it might seem natural to simply replace the TPR-maximizing sequence with the value-maximizing sequence:

$$\sigma_{\texttt{Value}}(x) := argsort(-\hat{\mathbf{p}}(x) \odot \mathbf{v}) \tag{26}$$

where $\mathbf{v}$ denotes the value vector, with v_k being the value for class k, and $\odot$ being the Hadamard product. However, this is sub-optimal, because it fails to consider the trade-off between value and cost. Instead, we propose the **ratio-maximizing sequence**, where the ratio refers to the marginal value divided by the marginal cost proxy:

$$\sigma_{\texttt{Ratio}}(x) := argsort(-\mathbf{r}) \text{ s.t. } r_k := \hat{p}_k(x)v_k/\hat{C}_k(x). \tag{27}$$

Here, $\hat{C}_k(X)$ denotes the predicted marginal cost for class k. For example, for $\hat{C}_{FP}$ (Eq. (14)), $\hat{C}_{FP,k}(x) = 1-\hat{p}_k(x)$. Note that we dropped the dependency on S here because of the additive assumption.

The Greedy Ratio Sequence (General) The `Ratio` method described above could be further generalized to any cost and value functions:

$$\sigma'_{\texttt{Ratio}}(j+1, x) := \underset{k \notin S_{(j)}(x)}{\arg\max}\, r(k, S_{(j)}) \tag{28}$$

$$\text{where } r(k, S) := \frac{\hat{V}(S \cup \{k\}) - \hat{V}(S)}{\hat{C}(S \cup \{k\}) - \hat{C}(S)}. \tag{29}$$

`FavMac` by default uses Eq. (29), as well as the faster special case Eq. (27) when possible.

5. Fast Online Update with `QuantileTree`

We have presented the analytical form for the thresholds for both the expected cost and violation control in Eq. (18) and Eq. (21). However, both involve computing the supremum over a set of feasible values, which is costly, especially in an online setting where the thresholds need to be constantly re-computed. In this section, we will first cast the threshold-finding problem into a more general task of quantile finding, and then present an efficient implementation for the quantile finding problem (thus permitting computation of the thresholds dynamically). These two tricks reduce the complexity by a factor of N (sample size). We focus on computing T_c and defer $T_{c,\delta}$ to the Appendix as the latter is much simpler.

Suppose each sample corresponds to up to M $(= |\mathcal{U}(X)|)$ thresholds. There are $O(MN)$ potential thresholds. For each threshold t, evaluating $\sum_{i=1}^{N} C^+(t; Z_i)$ is $\Omega(MN)$ depending on the implementation. If we perform a binary search on the thresholds, *each online update* will cost at least $O(MN \log(MN))$, or roughly $O(N \log N)$ when $N \gg M$. This is too expensive when N is large - for example, tens of billions for a company processing insurance claims.

The first step towards an efficient algorithm is the observation that finding the threshold is similar to finding a weighted quantile of all thresholds recorded so far. Formally, given a set of values $v_1, \ldots, v_{N+1}$ with weights $w_1, \ldots, w_{N+1}$, the CDF F of the weighted distribution is (Guan, 2022):

$$F(t) := \sum_{i=1}^{N+1} w_i \Delta_{v_i}(t) \text{ where } \sum_{i=1}^{N+1} w_i = 1. \tag{30}$$

Recall the definition of Q as was provided in Eq. (10). Now, we formally establish the equivalence between T_c and a weighted quantile, in the theorem below:

Theorem 5.1. *For each data x_i, sort $\mathcal{U}(x_i)$ as $[S_{i,(1)}, \ldots, S_{i,(M)}]$ such that $\hat{c}(S_{i,(j)}) \leq \hat{c}(S_{i,(j+1)})$. Assuming $\forall x$, $\emptyset \in \mathcal{U}(x)$ and $C(\emptyset; Z) = \hat{C}(\emptyset; X) = 0$. Let*

$$c^+_{i,(j)} = \max_{j' \leq j} \{C(S_{i,(j')}; z_i)\} \tag{31}$$

$$w_{i,(j)} = \Big(\frac{c^+_{i,(j)} - c^+_{i,(j-1)}}{W}\Big)\mathbb{1}\{j > 1\}. \tag{32}$$

where $W = \sum_{i,j} w_{i,j}$ is a normalizing factor. Denote the weighted quantile as

$$T'_c := Q(\frac{c(N+1) - C_{max}}{W}; F = \sum_{i=1}^{N} \sum_{j} w_{i,(j)} \Delta_{\hat{c}_{i,(j)}}) \tag{33}$$

Then, T_c is the immediate successor of T'_c when $\frac{c(N+1)-C_{max}}{W}$ is in the domain of F in $\{\hat{c}(S_{i,(j)})\}_{i,j}$ (unlikely in practice). Otherwise, $T'_c = T_c$.

With Theorem 5.1, the problem becomes finding the weighted quantile[6] in an efficient manner. We will achieve this with a custom data structure, and our high-level idea is to combine a self-balancing binary search tree and a segmentation tree: The former allows for efficient updates of $\{c_{i,(j)}\}_{j=1}^{M}$, and the latter enables efficient query of the weighted quantile. The `INSERT` and `QUERY` operations are briefly described in Algorithm 1. We refer to this data structure as `QuantileTree`. To the best our knowledge, this is a new data structure to store and query any empirical CDF and is perhaps of independent interest[7]. Several technical details are deferred to the Appendix, including the (simpler) computation of $T_{c,\delta}$ and how to `DELETE` data points, which is important if in practice we need to account for distribution shift with a rolling (instead of expanding) window in calibrating our thresholds. Complete implementations are included in the supplementary material. Finally, we present the full pipeline of `FavMac` for expected cost control in Algorithm 2, with the slightly different version for violation control in the Appendix.

Complexity: The sorting of $[c_{i,(j)}]$ takes $O(M \log M)$ and each insertion into the tree takes $O(\log(MN))$. Thus, inserting a data point (x_i, y_i) takes $O(M \log(MN))$ in total

[6] Strictly speaking, we sometimes need to find its immediate successor. However, the chance of this is very low, and if we skip this step, we are only changing $\leq$ to $<$ in Eq. (3).

[7] Much effort has been denoted to estimating quantiles in very large-scale applications, but here we focus on the *exact* quantiles of any discrete CDF. See discussion in Section 2.

Algorithm 1 `QuantileTree` (short version)

Struct *Node*:
  `color` # For red/black marking
  `value` # Recording v_i (the threshold in Δ_{v_i})
  `sum` # The sum of all w_i under this node
  `weight` # w_i. `weight` (thus `sum`) is un-normalized.
Function `INSERT` (*value v, weight w*) :
  Insert a leaf Node(`value`=c, `weight`=w)
  Fix the tree to a valid red-black tree, while maintaining valid `sum` values.
End Function
Function `QUERY` (*quantile q*) :
  Compute the cumulative weight $cw =$`root.sum`$*q$.
  Find the `node`(cw) corresponding to cw using binary search on `sum`.
  return `node`(cw)`.value`
End Function

Algorithm 2 Expected Cost Control (online)

Input: data $\{(X_1, Y_1), (X_2, Y_2), \ldots\}$, value and cost function V and C
`tree` $\leftarrow$ a `QuantileTree` object.
for $N = 0, 1, \ldots$ **do**
  if $N >$ some burn-in period **then**
    $T_c \leftarrow$ `tree.QUERY`$(\frac{(N+1)\cdot c - C_{max}}{tree.root.sum})$
    Find the **greedy-ratio** $\mathcal{U}(x_{N+1})$ with Eqs. (24) and (29) and the relevant proxy $\hat{V}$ and $\hat{C}$ from Section 4.2.
    Pick $\hat{S}_{N+1}$ from $\mathcal{U}(x_{N+1})$ with Eq. (19) and T_c.
  end if
  Insert $\{(\hat{c}_{N,(j)}, w_{N,(j)})\}_{j=1}^{M}$ to `tree` (Theorem 5.1).
end for

(same for deletion). A query step takes $O(\log(MN))$. They add up to $O(M \log(MN))$ per update, which improves at least a factor of N over the binary search version with a complexity of $O(MN \log(MN))$ (or $O(\log N)$ vs $O(N \log N)$ if we consider $N \gg M$).

6. Experimental Evaluation

6.1. Datasets and Tasks

We consider the following datasets and tasks for evaluation.

- `MNIST`: A synthetic dataset created by superimposing MNIST (Lecun et al., 1998) images, with each class appearing with a probability of 0.4. The task is to detect all digits that appear in the image. Training/test split is 54000/10000.
- `MIMIC`: The MIMIC-III dataset (Johnson et al., 2016; Goldberger et al., 2000; Johnson et al., 2019) is collected from the critical care units of the Beth Israel Deaconess Medical Center and contains data from 38K patients. For each discharge report, we randomly mask out 30% of the HCC codes (and the associated ICD-9 codes) and remove the relevant diagnoses section in the discharge notes. Input include the remaining free-text notes and patient features pre-processed by (Harutyunyan et al., 2019). Training/test split is 27,895/8,570.
- `Claim`: Insurance claim prediction on a proprietary dataset from a large healthcare data provider in North America. For each claim date, we use the previous 26 weeks to predict any HCC code to appear in the next 26 weeks. Training/test split is 236,089/75,484.

For both `MIMIC` and `Claim`, we created a "(select)" version that only focuses on 10 frequent codes in order to evaluate $\mathcal{U}_{full}$ (which requires enumerating all 2^K sets). We create the synthetic `MNIST` to evaluate different methods on more complex (made-up) value functions, as well as when the label is not too sparse. ($\mathbb{E}[|Y|]$ ranges from 0.4 to 0.8 in `MIMIC` and `Claim`).

Value and Cost Functions For `MIMIC` and `Claim`, we define V_{TP} and C_{FP} like in Section 3.1, as well as V_{TPC} and C_{FPC} with the corresponding risk adjustment factors. We let $w_k = k$ and define the same for `MNIST`[8].To illustrate the flexibility of `FavMac`, we also define a "general" (non-additive) value function for MNIST:

$$V_{GEN}(S; Y) = \prod_{k \in S \cap Y} \frac{k+5}{10} + \sum_{k \in S \cap Y} (k-5)^2. \quad (34)$$

For readability, all value and cost functions have been normalized to take values in $[0, 100]$.

Experiment Setup We set the target cost c from 5 to 50 with a stride of 5 (rounded for C_{FP}), as well as the commonly used $\delta = 0.1$. The base classifiers are deep neural networks, with architecture and training details in the Appendix. For C_{FP} control, we use the continuous values functions.For C_{FPC} control, we use all value functions: V_{TP}, V_{TPC} and V_{MULT}. V_{TP} with C_{FP} is skipped because it reduces to experiments in (Fisch et al., 2022). For each random seed, we randomly sample 3000 samples from the test set and use 1000 points as "burn-in". Mean and standard deviations for 10 seeds are reported. For `Claim`, we include a "sequential" version (`Claim-S`) in the Appendix with a temporal train/test split of 75/25 to simulate real-world applications.

6.2. Baselines

We compare our method with the following baselines (where applicable, as none can be applied to all experiments): `InnerSet`, a one-sided variant of (Cauchois et al., 2022) adapted to FP control in (Fisch et al., 2022); `ClassWise`, which controls class-wise false positive with $\epsilon = \frac{c}{K}$ (Theorem 4.4); `FPCP` (Fisch et al., 2022), equivalent to Eq. (25)

[8]We treat class 0 as 10 so it has nonzero weight.

but trains an additional DeepSets (Zaheer et al., 2017) as $\hat{C}_{FP}$. All these methods do not support flexible value functions, but where applicable they should provide (sometimes conservative) cost control.

We also include variants of `FavMac` with different $\mathcal{U}$: `Value` (Eq. (26)), `Prob` (Eq. (25)) and `Full` (using $\mathcal{U}_{full}$). Unless specified, `FavMac` refers to the variants with greedy ratio (Eqs. (27) and (29)). We want emphasize again that in controlling C_{FP}, `FavMac-Prob` behaves exactly the same as `FPCP`. However, we refer to it as `Prob` in C_{FPC} control, as `FPCP` does not apply to general cost functions and technically is too costly to compute in its original form (see Fig. 2). This name also highlights the cause of different performance - difference in $\mathcal{U}$.

Evaluation Metrics include the empirical mean of Excess Cost ($C(\hat{S}) - c$), Violation Frequency ($\mathbb{1}\{C(\hat{S}) > c\}$), and Value ($V(\hat{S})$), as well as Computation Time.

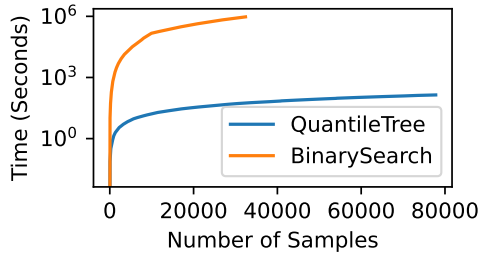

Figure 2. Computation time as a function of N, on `Claim-S`. As expected, `QuantileTree` gracefully handles large N, while BinarySearch does not finish.

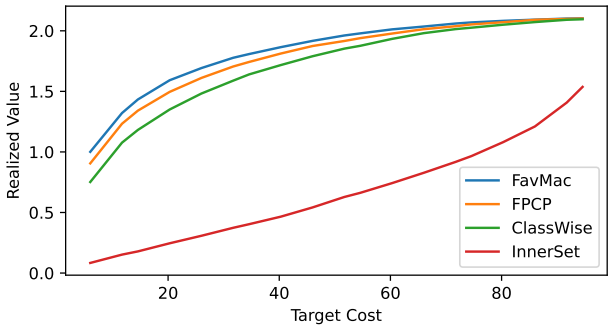

Figure 3. The V_{TPC} vs C_{FP} (value-cost) curve on `MIMIC`. Here we extend the target cost up to 95. `FavMac` achieves better tradeoffs than `FPCP`, while `InnerSet` and `ClassWise` are conservative, realizing lower values.

6.3. Results

Below we compare main results from the experiments, with full results in the Appendix due to space constraints. We use p-value=0.01 for all tables (paired t-test when possible).

Table 2. Mean value of the prediction sets with *expected* false positive control. The higher the better.

Value (↑)	`ClassWise`	`InnerSet`	`FPCP`	`FavMac`
`MIMIC` (select)	2.14±0.08	0.87±0.06	2.16±0.07	**2.26±0.08**
`MIMIC`	1.44±0.05	0.34±0.03	1.56±0.04	**1.64±0.05**
`Claim` (select)	1.83±0.11	1.45±0.06	2.02±0.10	**2.11±0.11**
`Claim`	1.17±0.07	0.68±0.04	1.18±0.07	**1.22±0.06**
`MNIST`	35.34±0.23	24.06±0.34	37.44±0.27	**38.20±0.26**
`MNIST` (GEN)	35.87±0.28	24.76±0.35	38.31±0.32	**39.64±0.34**

Table 3. Mean value of the prediction sets with false positive *violation* control. The higher the better.

Value (↑)	`InnerSet`	`FPCP`	`FavMac`
`MIMIC` (select)	0.30±0.04	2.08±0.07	**2.21±0.07**
`MIMIC`	0.14±0.02	1.41±0.04	**1.61±0.05**
`Claim` (select)	1.02±0.05	2.00±0.10	**2.09±0.11**
`Claim`	0.42±0.02	0.91±0.05	**1.21±0.06**
`MNIST`	16.12±0.54	36.39±0.26	**37.31±0.25**
`MNIST` (GEN)	16.84±0.57	37.26±0.30	**38.80±0.34**

The mean **Values** for false positive control are reported in Tables 2 and 3, where `FavMac` significantly outperforms baselines, giving up to 33% relative improvement against the best baseline, with p-value from 7e-14 to 1e-5. `ClassWise` and `InnerSet` are fundamentally conservative, as reflected in the low value their prediction sets achieve. Table 4 includes values for continuous cost control, for variants of `FavMac` using different $\mathcal{U}$ (and should be viewed as an ablation study). Note also how `Full` tends to perform badly despite searching through a much larger $\mathcal{U}$. Like noted earlier, this is due to larger gap between $C(\hat{S})$ and C^+ as $|\mathcal{U}|$ increases.

On the cost side, Table 5 shows the mean **Excess Cost**, defined by the mean cost minus target cost (averaged across all targets). Table 6 shows the mean **Violation Frequency**, namely the proportion of prediction sets with cost higher than target. `ClassWise` is conservative, verifying Section 4.1, and `InnerSet` is too as it controls the probability that *any* class is wrong. Like `FavMac`, `FPCP` controls C_{FP} well (but achieves lower values and cannot control general costs). Results on continuous costs are in the Appendix.

Finally, results for `FPCP` are computed using `QuantileTree`. As shown in Fig. 2, **Computation Time** is significantly improved by `QuantileTree`. Even with binary search on the thresholds, the original method still takes too long to finish.

7. Conclusion

We have formalized a new problem of value-maximization with cost control for multi-label prediction sets, and proposed a framework, `FavMac`, to solve this problem. Improving upon previous works, `FavMac` explicitly maximizes a value function for the prediction set, and is able

Table 4. Mean value of the prediction sets with continuous cost control by variants of `FavMac`. The difference is only the $\mathcal{U}$ being used. The `Ratio` version of `FavMac` consistently outperforms other variants. (`Prob` is equivalent to `Ratio` with V_{TPC}).

Expectation Control	Full	Prob	Value	Ratio
MIMIC (select,TP)	3.49±0.10	**3.63±0.10**	**3.63±0.10**	**3.63±0.11**
MIMIC (TP)	–	1.91±0.02	1.91±0.02	**1.95±0.03**
Claim (select,TP)	4.51±0.16	4.51±0.16	4.51±0.16	**4.55±0.16**
Claim (TP)	–	1.85±0.08	1.85±0.08	**1.88±0.09**
MNIST (TP)	34.62±0.26	36.80±0.29	36.80±0.29	**37.05±0.29**
MIMIC (select,TPC)	1.93±0.05	2.09±0.05	**2.12±0.06**	2.09±0.05
MIMIC (TPC)	–	**1.59±0.04**	1.50±0.04	**1.59±0.04**
Claim (select,TPC)	2.05±0.08	**2.08±0.09**	2.02±0.10	**2.08±0.09**
Claim (TPC)	–	**1.24±0.06**	1.18±0.06	**1.24±0.06**
MNIST (TPC)	33.87±0.22	**36.65±0.25**	35.99±0.26	**36.65±0.25**
MNIST (GEN)	36.54±0.27	37.60±0.31	38.39±0.34	**38.87±0.32**
Violation Control				
MIMIC (select,TP)	3.40±0.10	3.32±0.09	3.32±0.09	**3.43±0.10**
MIMIC (TP)	–	1.84±0.03	1.84±0.03	**1.89±0.03**
Claim (select,TP)	4.42±0.16	4.44±0.16	4.44±0.16	**4.45±0.16**
Claim (TP)	–	1.81±0.08	1.81±0.08	**1.84±0.08**
MNIST (TP)	31.10±0.22	32.76±0.26	32.76±0.26	**33.48±0.25**
MIMIC (select,TPC)	**1.86±0.05**	**1.86±0.05**	1.72±0.05	**1.86±0.05**
MIMIC (TPC)	–	**1.51±0.04**	1.38±0.04	**1.51±0.04**
Claim (select,TPC)	1.99±0.08	**2.02±0.08**	1.88±0.08	**2.02±0.08**
Claim (TPC)	–	**1.21±0.06**	1.14±0.06	**1.21±0.06**
MNIST (TPC)	30.02±0.22	**32.37±0.26**	30.76±0.29	**32.37±0.26**
MNIST (GEN)	33.17±0.27	33.46±0.30	34.40±0.36	**35.65±0.33**

Table 5. Excess cost, where values that are significant different from 0 (too conservative) are marked red.

Excess Cost	ClassWise	InnerSet	FPCP	FavMac
MIMIC (select)	-2.32±0.46	-26.58±0.13	0.02±0.15	-0.01±0.15
MIMIC	-0.91±0.36	-27.10±0.05	-0.04±0.08	-0.05±0.05
Claim (select)	-2.48±0.51	-23.40±0.26	-0.03±0.19	-0.04±0.20
Claim	-1.15±0.57	-24.40±0.19	-0.03±0.06	-0.05±0.05
MNIST	-13.14±0.25	-27.09±0.11	-0.08±0.10	-0.07±0.12
MNIST (GEN)	-13.14±0.25	-27.09±0.11	-0.08±0.10	-0.08±0.09

Table 6. Violation probability, where values that are significantly different from δ (too conservative) are marked red.

Violation ($\delta = 10\%$)	InnerSet	FPCP	FavMac
MIMIC (select)	0.13±0.02	10.05±0.49	10.13±0.32
MIMIC	0.01±0.00	10.00±0.48	9.95±0.55
Claim (select)	0.30±0.07	9.94±0.69	9.75±0.73
Claim	0.06±0.02	10.08±0.34	9.84±0.45
MNIST	0.10±0.03	9.88±0.42	9.91±0.39
MNIST (GEN)	0.10±0.03	9.88±0.42	10.08±0.48

to control, in either expectation or violation risk, any continuous cost function. Moreover, with the help of `QuantileTree`, a new custom data structure, `FavMac` can perform online updates efficiently for large-scale applications. Empirical evidence suggests that `FavMac` tends to achieve higher value while providing rigorous cost control. Since the trade-off between value and cost is common in many areas, we hope this paper will provide useful insights for various application areas, as well as serve as a foundation for potential future research in this task. For example, examining better $\mathcal{U}$ as well as different $\hat{C}$ and $\hat{V}$ for specific value and cost functions.

Acknowledgements

This work was supported by NSF award SCH-2205289, SCH-2014438, IIS-1838042, NIH award R01 1R01NS107291-01.

References

Angelopoulos, A. N., Bates, S., Candès, E. J., Jordan, M. I., and Lei, L. Learn then test: Calibrating predictive algorithms to achieve risk control, 2021a. URL https://arxiv.org/abs/2110.01052.

Angelopoulos, A. N., Bates, S., Jordan, M., and Malik, J. Uncertainty sets for image classifiers using conformal prediction. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=eNdiU_DbM9.

Angelopoulos, A. N., Bates, S., and Lei, L. Risk control in expectation: A proof and counterexample, 2021c. URL http://angelopoulos.ai/publications/downloads/rce_note.pdf.

Aziz, H., Moulin, H., and Sandomirskiy, F. A polynomial-time algorithm for computing a pareto optimal and almost proportional allocation. *Operations Research Letters*, 48(5):573–578, 2020. ISSN 0167-6377. doi: https://doi.org/10.1016/j.orl.2020.07.005. URL https://www.sciencedirect.com/science/article/pii/S0167637720301024.

Balcan, M. F., Constantin, F., Iwata, S., and Wang, L. Learning valuation functions. In Mannor, S., Srebro, N., and Williamson, R. C. (eds.), *Proceedings of the 25th Annual Conference on Learning Theory*, volume 23 of *Proceedings of Machine Learning Research*, pp. 4.1–4.24, Edinburgh, Scotland, 25–27 Jun 2012. PMLR. URL https://proceedings.mlr.press/v23/balcan12b.html.

Barber, R. F., Candes, E. J., Ramdas, A., and Tibshirani, R. J. Conformal prediction beyond exchangeability, 2022. URL https://arxiv.org/abs/2202.13415.

Bates, S., Angelopoulos, A. N., Lei, L., Malik, J., and Jordan, M. I. Distribution-free, risk-controlling prediction sets. *arXiv preprint arXiv:2101.02703*, 2020.

Brustle, J., Dippel, J., Narayan, V. V., Suzuki, M., and Vetta, A. One dollar each eliminates envy. In *Proceedings of the 21st ACM Conference on Economics and Computation*, EC '20, pp. 23–39, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379755. doi: 10.1145/3391403.3399447. URL https://doi.org/10.1145/3391403.3399447.

Cauchois, M., Gupta, S., and Duchi, J. C. Knowing what you know: Valid and validated confidence sets in multiclass and multilabel prediction. *J. Mach. Learn. Res.*, 22 (1), jul 2022. ISSN 1532-4435.

Chen, T. and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

Chen, Z. and Zhang, A. A survey of approximate quantile computation on large-scale data. *IEEE Access*, 8:34585–34597, 2020.

Corani, G. and Zaffalon, M. Learning reliable classifiers from small or incomplete data sets: The naive credal classifier 2. *Journal of Machine Learning Research*, 9(20): 581–621, 2008. URL http://jmlr.org/papers/v9/corani08a.html.

Corani, G. and Zaffalon, M. Lazy naive credal classifier. In *Proceedings of the 1st ACM SIGKDD Workshop on Knowledge Discovery from Uncertain Data*, U '09, pp. 30–37, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605586755. doi: 10.1145/1610555.1610560. URL https://doi.org/10.1145/1610555.1610560.

Cormode, G., Korn, F., Muthukrishnan, S., and Srivastava, D. Effective computation of biased quantiles over data streams. In *21st International Conference on Data Engineering (ICDE'05)*, pp. 20–31, 2005. doi: 10.1109/ICDE.2005.55.

del Coz, J. J., Díez, J., and Bahamonde, A. Learning nondeterministic classifiers. *Journal of Machine Learning Research*, 10(79):2273–2293, 2009. URL http://jmlr.org/papers/v10/delcoz09a.html.

Elkan, C. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pp. 973–978. Lawrence Erlbaum Associates Ltd, 2001.

Fisch, A., Schuster, T., Jaakkola, T., and Barzilay, D. Conformal prediction sets with limited false positives. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 6514–6532. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/fisch22a.html.

Gendler, A., Weng, T.-W., Daniel, L., and Romano, Y. Adversarially robust conformal prediction. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=9L1BsI4wP1H.

Gibbs, I. and Candes, E. Adaptive conformal inference under distribution shift. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=6vaActvpcp3.

Gibbs, I. and Candès, E. Conformal inference for online prediction with arbitrary distribution shifts. *arXiv preprint arXiv:2208.08401*, 2022.

Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C. K., and Stanley, H. E. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation*, 2000. ISSN 15244539. doi: 10.1161/01.cir.101.23.e215.

Guan, L. Localized conformal prediction: a generalized inference framework for conformal prediction. *Biometrika*, 07 2022. ISSN 1464-3510. doi: 10.1093/biomet/asac040. URL https://doi.org/10.1093/biomet/asac040. asac040.

Guibas, L. J. and Sedgewick, R. A dichromatic framework for balanced trees. In *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, pp. 8–21. IEEE, 1978.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Gupta, C., Kuchibhotla, A. K., and Ramdas, A. Nested conformal prediction and quantile out-of-bag ensemble methods. *Pattern Recognition*, 127:108496, 2022. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.108496. URL https://www.sciencedirect.com/science/article/pii/S0031320321006725.

Harutyunyan, H., Khachatrian, H., Kale, D. C., Ver Steeg, G., and Galstyan, A. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):96, 2019. ISSN 2052-4463. doi: 10.1038/s41597-019-0103-9. URL https://doi.org/10.1038/s41597-019-0103-9.

Hendrickx, K., Perini, L., der Plas, D. V., Meert, W., and Davis, J. Machine learning with a reject option: A survey. *CoRR*, abs/2107.11277, 2021. URL https://arxiv.org/abs/2107.11277.

Huang, K., Altosaar, J., and Ranganath, R. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv:1904.05342*, 2019.

Igarashi, A. and Peters, D. Pareto-optimal allocation of indivisible goods with connectivity constraints. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 2045–2052, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 448–456. JMLR.org, 2015.

Johnson, A., Pollard, T., and Mark, R. Mimic-iii clinical database demo (version 1.4). https://archive.ics.uci.edu/ml/datasets/EEG+Database, 2019.

Johnson, A. E., Pollard, T. J., Shen, L., Lehman, L.-w. H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., and Mark, R. G. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3 (1):160035, 2016. ISSN 2052-4463. doi: 10.1038/sdata.2016.35. URL https://doi.org/10.1038/sdata.2016.35.

Karimi, D., Dou, H., Warfield, S., and Gholipour, A. Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis. *Medical Image Analysis*, 65:101759, 06 2020. doi: 10.1016/j.media.2020.101759.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krause, A. and Golovin, D. Submodular function maximization. *Tractability*, 3:71–104, 2014.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Lei, J. Classification with confidence. *Biometrika*, 101(4), 2014. ISSN 14643510. doi: 10.1093/biomet/asu038.

Lin, Z., Glass, L., Westover, M. B., Xiao, C., and Sun, J. Scrib: set-classifier with class-specific risk bounds for blackbox models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7497–7505, 2022.

Mortier, T., Wydmuch, M., Dembczyński, K., Hüllermeier, E., and Waegeman, W. Efficient set-valued prediction in multi-class classification. *Data Min. Knowl. Discov.*, 35 (4):1435–1469, jul 2021. ISSN 1384-5810. doi: 10.1007/s10618-021-00751-x. URL https://doi.org/10.1007/s10618-021-00751-x.

Murphy, A. H. and Winkler, R. L. Probability forecasting in meteorology. *Journal of the American Statistical Association*, 79:489–500, 1984.

Nguyen, V.-L., Destercke, S., Masson, M.-H., and Hüllermeier, E. Reliable multi-class classification based on pairwise epistemic and aleatoric uncertainty. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 5089–5095. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/706. URL https://doi.org/10.24963/ijcai.2018/706.

Papadopoulos, H., Proedrou, K., Vovk, V., and Gammerman, A. Inductive confidence machines for regression. In Elomaa, T., Mannila, H., and Toivonen, H. (eds.), *Machine Learning: ECML 2002*, pp. 345–356, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-36755-0.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Sadinle, M., Lei, J., and Wasserman, L. Least Ambiguous Set-Valued Classifiers With Bounded Error Levels. *Journal of the American Statistical Association*, 114(525), 2019. ISSN 1537274X. doi: 10.1080/01621459.2017.1395341.

Tibshirani, R. J., Barber, R. F., Candes, E. J., and Ramdas, A. Conformal prediction under covariate shift, 2020.

Vaicenavicius, J., Widmann, D., Andersson, C., Lindsten, F., Roll, J., and Schön, T. Evaluating model calibration in classification. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 3459–3467. PMLR, 16–18 Apr 2019.

URL https://proceedings.mlr.press/v89/vaicenavicius19a.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vovk, V., Gammerman, A., and Shafer, G. *Algorithmic learning in a random world*. Springer US, 2005. ISBN 0387001522. doi: 10.1007/b106715.

Yang, G., Destercke, S., and Masson, M.-H. The costs of indeterminacy: How to determine them? *IEEE Transactions on Cybernetics*, 47(12):4316–4327, 2017. doi: 10.1109/TCYB.2016.2607237.

Zaffalon, M. The naive credal classifier. *Journal of Statistical Planning and Inference*, 105 (1):5–21, 2002. ISSN 0378-3758. doi: https://doi.org/10.1016/S0378-3758(01)00201-4. URL https://www.sciencedirect.com/science/article/pii/S0378375801002014. Imprecise Probability Models and their Applications.

Zaffalon, M., Corani, G., and Mauá, D. Evaluating credal classifiers by utility-discounted predictive accuracy. *International Journal of Approximate Reasoning*, 53(8):1282–1301, 2012. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2012.06.022. URL https://www.sciencedirect.com/science/article/pii/S0888613X12000989. Imprecise Probability: Theories and Applications (ISIPTA'11).

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/f22e4747da1aa27e363d86d40ff442fe-Paper.pdf.

Zhang, Q. and Wang, W. A fast algorithm for approximate quantiles in high speed data streams. In *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*, pp. 29–29, 2007. doi: 10.1109/SSDBM.2007.27.

A. Proofs

A.1. Proof for Theorem 4.3

This is a relatively standard result and can be found, for instance, in (Lei, 2014). Hence we instead present a brief justification.

Proof: First of all, since $[Z_1, \ldots, Z_{N+1}]$ are assumed to be i.i.d., if we focus on the subset for which $Y_k = 0$, they are still i.i.d. (with the new distribution being the conditional distribution of $(X|_{Y_k=0}, Y|_{Y_k=0})$), thus exchangeable. If $Y_{N+1,k} = 1$, then the statement we want to prove is vacuously true. For simplicity, we denote the total number of samples with $Y_k = 0$ still as $N+1$, and will now assume all data have $Y_k = 0$ (as we have restricted to this distribution). Now, we could invoke the split conformal procedure from 4.2, and arrive at:

$$\mathbb{P}\{p_k(X_{N+1}) \leq Q(1-\epsilon; \{p_k(X_i)\} \cup \{\infty\}\} \geq 1-\epsilon \tag{35}$$
$$\implies \mathbb{P}\{p_k(X_{N+1}) \leq t\} \geq 1-\epsilon \tag{36}$$
$$\implies \mathbb{P}\{k \in S_{N+1}\} < \epsilon \tag{37}$$
$$\tag{38}$$

Note that if we are to assume exchangeability, we need to assume exchangeability of all conditional distributions (with different Y), making it not too different from just assuming the stronger assumption of i.i.d. □

A.2. Proof for Theorem 4.4

Proof:

$$\mathbb{E}[C_{FP}(S_{N+1}, Y_{N+1})] = \mathbb{E}[|\{k : k \in S_{N+1}, k \notin Y_{N+1}\}|] \tag{39}$$
$$= \mathbb{E}[\sum_{k\in[K]} \mathbb{1}\{k : k \in S_{N+1}, k \notin Y_{N+1}\}] \tag{40}$$
$$= \sum_{k\in[K]} \mathbb{P}\{k : k \in S_{N+1}, k \notin Y_{N+1}\} \tag{41}$$
$$= \sum_{k\in[K]} \mathbb{P}\{k \notin Y_{N+1}\}\mathbb{P}\{k : k \in S_{N+1} | k \notin Y_{N+1}\} \tag{42}$$
$$\leq \sum_{k\in[K]} \mathbb{P}\{k : k \in S_{N+1} | k \notin Y_{N+1}\} \tag{43}$$

Applying Theorem 4.3 to each k for Eq. (43) (with $\epsilon = \frac{c}{K}$), we have

$$\mathbb{E}[C_{FP}(S_{N+1}, Y_{N+1})] \leq \sum_{k\in[K]} \frac{c}{K} = c. \tag{44}$$

□

A.3. Proof for Theorem 4.5

Like (Fisch et al., 2022), the proof below uses a result from (Angelopoulos et al., 2021c). We modified the proof in (Angelopoulos et al., 2021c) so it is simpler and more self-contained.

Proof: Denote $C_i^+(\cdot) = C_i^+(\cdot, Z_i)$ for convenience. $[C_1^+, \ldots, C_{N+1}^+]$ are thus exchangeable non-decreasing bounded functions in $[0, C_{max}]$. Due to exchangeability,

$$\forall i \in [N+1], \forall t, \mathbb{E}[C_i^+(t)] \equiv \frac{\sum_{i=1}^{N+1} \mathbb{E}[C_i^+(t)]}{N+1} = \frac{\mathbb{E}[\sum_{i=1}^{N+1} C_i^+(t)]}{N+1}. \tag{45}$$

Let $c' = c - \frac{C_{max}}{N+1}$. We rewrite the definition in Eq. (18) for T_c as well as a new quantify T_c':

$$T_c = \sup\{t : \frac{1}{N+1} \sum_{i=1}^{N} C_i^+(t) \leq c'\} \tag{46}$$

$$T_c' = \sup\{t : \frac{1}{N+1} \sum_{i=1}^{N+1} C_i^+(t) \leq c\}. \tag{47}$$

(Note that if we are taking $\sup$ over empty sets T_c would be $-\infty$, and $\hat{S} =$ which will trivially satisfy all cost requirements.) Because

$$\frac{1}{N+1} \sum_{i=1}^{N} C_i^+(t) \leq c - \frac{C_{max}}{N+1} \implies \frac{1}{N+1} \sum_{i=1}^{N+1} C_i^+(t) \leq c, \tag{48}$$

we have $T_c' \geq T_c$, which means by definition of T_c':

$$\frac{\sum_{i=1}^{N+1} C_i^+(T_c)}{N+1} \leq c. \tag{49}$$

Invoking Eq. (45) for $i = N+1$ we have

$$\mathbb{E}[C_{N+1}^+(T_c)] \leq c. \tag{50}$$

Finally, by definition of C^+:

$$\mathbb{E}[C(\hat{S}_{N+1}; Z_{N+1})] \leq \mathbb{E}[\max_{S \in \mathcal{U}(X_{N+1})} \{C(S; Z_{N+1}) : \hat{C}(S; X_{N+1}) < T_c] \leq c. \tag{51}$$

Note that all expectations are taken over the randomness of all of $[Z_1, \ldots, Z_{N+1}]$. □

A.4. Proof for Theorem 4.6

Although there is a similar result in (Fisch et al., 2022) leveraging (Angelopoulos et al., 2021c), we will use a *different* proof approach here. We will define $T_{c,\delta}$ in a seemingly different way that makes the guarantee self-evident. Then, we will establish equivalence to Eq. (21). Note we defined $T_{c,\delta}$ like in Eq. (21) (which might look less familiar to readers versant in conformal prediction literature) in order to align with (Fisch et al., 2022) as well as Eq. (18) (and to make it clear why `QuantileTree` can be deployed similarly in both versions of cost control).

Proof: Denote random variable

$$T_i := \max_t \{t \in \mathcal{T}_i \cup \{\infty\} : C^+(t; Z_i) \leq c\} \tag{52}$$

$$\text{where } \mathcal{T}_i := \{\hat{C}(S; X_i) : S \in \mathcal{U}(X_i)\}. \tag{53}$$

It is clear that $[T_1, \ldots, T_{N+1}]$ are exchangeable because $[Z_1, \ldots, Z_{N+1}]$ are exchangeable. Note that $C^+(T_i; Z_i) \leq c$. Observe crucially that

- Since $C^+(t; Z_i)$ is non-decreasing in t, we have $C^+(t; Z_i) > c \geq C^+(T_i; Z_i) \implies t > T_i$.
- By the definition of T_i we also note that $t > T_i \implies C^+(t; Z_i) > c$ (if $T_i = \infty$ this is vacuously true).

Thus, we have $T_i = \sup\{t \in \mathbb{R} : C^+(t; Z_i) \leq c\}$, or equivalently:

$$t > T_i \Leftrightarrow C^+(t; Z_i) > c \tag{54}$$

If we choose $T_{c,\delta} = -Q(1-\delta; \{-T_i\}_{i=1}^N \cup \{\infty\})$, Theorem 4.2 directly gives us:

$$\mathbb{P}\{T_{N+1} \geq T_{c,\delta}\} = \mathbb{P}\{C^+(T_{c,\delta}; Z_{N+1}) \leq c\} \geq 1 - \delta \tag{55}$$

Finally, like before:

$$\mathbb{P}\{C(\hat{S}_{N+1}; Z_{N+1}) \leq c\} \geq \mathbb{P}\{\max_{S \in \mathcal{U}(X)} \{C(S; Z) : \hat{C}(S; X) < T_{c,\delta}\} \leq c\} \tag{56}$$

$$= \mathbb{P}\{C^+(T_{c,\delta}; Z_{N+1}) \leq c\} \geq 1 - \delta \tag{57}$$

Now, we only need to show that the $T_{c,\delta}$ defined above is the same as the one in Eq. (21). To see this:

$$-Q(1-\delta; \{-T_i\}_{i=1}^N \cup \{\infty\}) = -\inf\{t : \sum_{i=1}^N \mathbb{1}\{t \geq -T_i\} + \mathbb{1}\{t \geq \infty\} \geq (1-\delta)(N+1)\} \tag{58}$$

$$= \sup\{t : \sum_{i=1}^N \mathbb{1}\{t \leq T_i\} + \mathbb{1}\{t \leq -\infty\} \geq (1-\delta)(N+1)\} \tag{59}$$

$$= \sup\{t : \sum_{i=1}^N \mathbb{1}\{t \leq T_i\} \geq (1-\delta)(N+1)\} \tag{60}$$

Note again that $\mathbb{1}\{t \leq T_i\} = \mathbb{1}\{C^+(t; Z_i) \leq c\}$, which means

$$T_{c,\delta} = \sup\{t : \sum_{i=1}^N \mathbb{1}\{C^+(t; Z_i) \leq c\} \geq (1-\delta)(N+1)\} \tag{61}$$

□

A.5. Proof for Theorem 5.1

Proof: For simplicity, write $M = \frac{c(N+1)-C_{max}}{W}$, and define:

$$T_c' = \inf\{t : \sum_{i=1}^{N}\sum_{j} w_{i,(j)}\Delta_{\hat{c}_{i,(j)}} \geq M\} \tag{62}$$

$$= \inf\{t : \sum_{i=1}^{N}\sum_{j>1} \frac{(c^+_{i,(j)} - c^+_{i,(j-1)})}{W}\mathbb{1}\{t \geq \hat{c}_{i,(j)}\} \geq M\} \tag{63}$$

Note that $c^+_{i,(0)} = c_{i,(0)} = \hat{c}_{i,(0)} = 0$ for any i, which means

$$\sum_{j>1} \frac{(c^+_{i,(j)} - c^+_{i,(j-1)})}{W}\mathbb{1}\{t \geq \hat{c}_{i,(j)}\} = \frac{1}{W}\max\{c^+_{i,(j)} : \hat{c}_{i,(j)} \leq t\}. \tag{64}$$

Thus,

$$T_c' = \inf\{t : \sum_{i=1}^{N}\max\{c^+_{i,(j)} : \hat{c}_{i,(j)} \leq t\} \geq WM\} \tag{65}$$

Note the similarity between Eq. (65) and what we want below (i.e. Eq. (18)):

$$T_c = \sup\{t : \sum_{i=1}^{N} C^+(t; Z_i) \leq M\} = \sup\{t : \sum_{i=1}^{N}\max\{c^+_{i,(j)} : \hat{c}_{i,(j)} < t\} \leq WM\} \tag{66}$$

Rewrite the empirical weighted CDF as

$$F = \sum_{i=1}^{N}\sum_{j} w_{i,(j)}\Delta_{\hat{c}_{i,(j)}} = \sum_{l=1}^{m} w_l'\Delta_{t_l} \tag{67}$$

where $t_1 < t_2 < \ldots t_m$ are the distinct values in $\{\hat{c}_{i,j}\}$, and $w_l' = \sum_{(i,j):\hat{c}_{i,(j)}=t_l} w_{i,(j)}$. WLOG, assume $T_c' = t_{l^*}$. Also write a version that is not "inclusive" of the threshold, defined as

$$F'(t) = \sum_{i=1}^{N}\sum_{j} w_{i,(j)}\mathbb{1}\{t > \hat{c}_{i,(j)}\} = \sum_{l=1}^{m} w_l'\mathbb{1}\{t > t_l\}. \tag{68}$$

Note the sign in the indicator function is now $>$ instead of $\geq$. It is now clear that Eq. (65) and Eq. (66) can be written as:

$$T_c' = \inf\{t : F(t) \geq WM\} \tag{69}$$

$$T_c = \sup\{t : F'(t) \leq WM\} \tag{70}$$

Consider t_{l^*-1}, the immediate predecessor of T_c'. By definition of T_c' (Eq. (69)) we know

$$F(t_{l^*-1}) < WM. \tag{71}$$

Moreover, by definition of F and F' we have

$$F(T_c') - w_{l^*}' = F'(T_c') = F(t_{l^*-1}) < WM. \tag{72}$$

Following a similar logic, we know that $F'(t_{l^*+1}) = F(T_c')$.

- If $F(T_c') = WM$ exactly, then technically T_c is t_{l^*+1}. However, the probability of this happening is very low (if $\hat{C}$ is continuous, such as those constructed from $\hat{\mathbf{p}}$).
- Otherwise, $F(T_c') > WM$, which means $T_c = T_c'$.

□

B. Details on the Quantile Tracking Algorithms

B.1. Maintenance of `QuantileTree`

Our exact implementation can be found in the supplementary material. The base structure of `QuantileTree` is a red-black tree (Guibas & Sedgewick, 1978). Many implementations could be found online, and we based ours on `https://github.com/Bibeknam/algorithmtutorprograms`. Like shown in Algorithm 1, we add two fields to a `Node`: sum and weight. Since each node represents a point mass of the empirical CDF (corresponding to a value v), the weight is naturally the mass of this value. In other words, a `QuantileTree` with $\{\texttt{Node}(value = v_i, weight = w_i)\}$ represents the empirical CDF $\sum_i w_i \Delta_{v_i}$. (We use Δ_v instead of the more commonly used notation δ_v because δ is already used in Definition 3.4). Finally, we maintain a segmentation tree using `Node.sum`, which keeps track of the sum of weights of the sub-tree rooted at this node.

B.2. Full Algorithm with `DELETE`

Due to space constraint, we did not include the `DELETE` operation in Algorithm 1. In Algorithm 3, we present the full algorithm with this operation. Algorithm 4 explains the case for expected cost control.

Algorithm 3 `QuantileTree`

Struct *Node*:
- `color` # For red/black marking
- `value` # Recording v_i (the threshold in Δ_{v_i})
- `sum` # The sum of all w_i under this node
- `weight` # w_i. `weight` (thus `sum`) is un-normalized.

Function `INSERT` (*value* v, *weight* w):
- Insert a leaf Node(`value`=c,`weight`=w)
- Fix the tree to a valid red-black tree, while maintaining valid `sum` values.

End Function

Function `DELETE` (*value* v, *weight* w):
- Find node for v, denoted as $node(v)$
- **if** $node(v).weight = w$ **then**
  - Remove $node(v)$
  - Fix the tree to a valid red-black tree, while maintaining valid `sum` values
- **else**
  - $node(v).weight \leftarrow node(v).weight - w$
- **end**

End Function

Function `QUERY` (*quantile* q):
- Compute the cumulative weight $cw =$`root.sum`$*q$.
- Find the node(cw) corresponding to cw using binary search (via `sum`).
- **return** $node(cw).value$

End Function

B.2.1. Maintaining `Node.sum`

One technical detail we did not explain in Algorithm 1 is how to maintain valid `sum` while using the red-black tree.

- During rotations, we only need to fix the recursive relation of `Node.sum = Node.left.sum + Node.weight + Node.right.sum` in each rotation operation, by computing the new sums and replace the `sum` after the rotation. For example, in a left rotation like:

```
x                y
 \       ->     /
  y            x
```

We need to compute

Algorithm 4 Expected Cost Control

`tree` ← a `QuantileTree` object
Function `INSERT` (*input* x_i, *label* y_i) **:**
  Denote $[\hat{c}_{i,(j)}]_{j=1}^M$ and $[w_{i,(j)}]_{j=1}^M$ as in Theorem 5.1.
  foreach $j = 1, \ldots, M$ **do**
    `tree.INSERT`($\hat{c}_{i,(j)}, w_{i,(j)}$)
  end
  Cache $[c_{i,(j)}]_{j=1}^M$ and $\{w_{i,j}\}_{j=1}^M$ (for `DELETION`).
End Function
Function `DELETE` (*input* x_i) **:**
  Retrieve $[\hat{c}_{i,(j)}]_{j=1}^M$ and $[w_{i,(j)}]_{j=1}^M$.
  foreach $j = 1, \ldots, M$ **do**
    `tree.DELETE`($c_{i,(j)}, w_{i,(j)}$)
  end
End Function
Function `COMPUTE_THRESHOLD` (*target cost* c) **:**
  return `tree.QUERY`($\frac{(N+1)\cdot c - C_{max}}{tree.root.sum}$)
End Function

```
//BEGIN OF CODE=========================================================
    new_x_sum = x.left.sum + y.left.sum + x.weight
    new_y_sum = new_x_sum + y.weight + y.right.sum
    ... //perform the rotation
    x.sum = new_x_sum
    y.sum = new_y_sum
//END OF CODE===========================================================
```

- Other than rotations, we only need to update the `sum` of all $O(log(N))$ nodes along the path to the node being inserted/deleted/updated.

B.3. Online Computation for $T_{c,\delta}$

In Algorithm 5, we present the computation of $T_{c,\delta}$ in an online fashion. Note that for this case, since it only involves counting integers (i.e. the weight is 1 for all thresholds), we could use existing python packages like `sortedcontainers.SortedList` to replace the self-balancing segmentation binary search tree.

Algorithm 5 Violation Control

`tree` ← a self-balancing segmentation binary search tree
Fix the cost target c
Function `INSERT` (*input* x_i, *label* y_i) **:**
  Let $t_i \leftarrow \inf_t\{C^+(t; (x_i, y_i) > c\}$
  `tree.INSERT`($t_i, weight = 1$)
  Cache t_i (for `DELETION`).
End Function
Function `DELETE` (*input* x_i) **:**
  Retrieve t_i.
  `tree.DELETE`($t_i, weight = 1$)
End Function
Function `COMPUTE_THRESHOLD` (*violation goal* δ) **:**
  Compute quantile to query $q = \frac{(N+1)\delta - 1}{tree.root.sum}$.
  return `tree.QUERY`(q)
End Function

B.4. An Example Application of `QuantileTree` to Existing Conformal Prediction Methods

In Algorithm 2, we saw how `QuantileTree` could be used to efficiently update the empirical CDF and look up $Q(\beta; F)$ for any β. We now look at a slightly different potential usage of `QuantileTree` in a recent paper (Bates et al., 2020), by efficiently evaluating F. (Note that $F = Q^{-1}$ for an continuous distribution.) We try to summarize the important steps in (Bates et al., 2020) here, but readers should refer to the original paper for a better understanding of their method. Like noted before, RCPS (Bates et al., 2020) controls the violation probability of the population risk. It achieves so by using concentration bounds to upper-bound the population risk, which is a function of a threshold parameter λ. The final prediction set is given as $\hat{S}(x) = \{k : \hat{p}_k(x) > \hat{\lambda}\}$, where $\hat{\lambda}$ is the chosen threshold.

In practice, λ could be searched from all applicable thresholds on the calibration set, and the bound is computed using the calibration set as well. Assuming the more tractable Hoeffding-Bentkus version is being used, to compute this upper-bound (for any λ), we need to compute the empirical risk $\hat{R}(\lambda)$. In (Bates et al., 2020), $\hat{R}(\lambda)$ only needs to be evaluated once, on the entire calibration set. As a result, one could cache the results $\hat{R}(\lambda)$ for each λ ($O(NK)$) and then perform a binary search over λ ($O(\log(NK))$), like what was done in the code published on `https://github.com/aangelopoulos/rcps`. Assuming we now want to update the threshold $\hat{\lambda}$ in an online fashion, which requires us to update $\hat{R}(\lambda)$ for *every possible* λ whenever the N-th calibration data point comes in, for $N = 1, 2, \ldots$. Actually recomputing $\hat{R}(\lambda)$ is obviously inefficient, but even if we only compute $R_{N+1}(\lambda)$ for each $\lambda_1, \ldots, \lambda_{NK}$ ($O(K^2N)$), and $R_i(\lambda_j)$ for $i \in [N]$ and $j \in [K]$ ($O(K^2N)$), it is still an $O(K^2N + NK\log(NK))$ operation per update, or $O(N \log N)$ assuming $N \gg K$ for simplicity). If we further use data structures like `sortedcontainers.SortedList` to maintain the sorted list of all $\{\lambda_{i,k}\}_{i\in[N],k\in[K]}$, the complexity for each update+query can reduce to $O(K^2N + \log(NK))$ (or $O(N)$ when $N \gg K$).

Instead, we could leverage the fact that $\hat{R}$ is monotonic decreasing in λ, and store the following CDF in a `QuantileTree`:

$$F = \sum_{i=1}^{N} \sum_{j=1}^{K} w_{i,(j)} \Delta_{-l_{i,(j)}} \tag{73}$$

$$\text{where } w_{i,(j)} = R_i(l_{i,(j)}) - R_i(l_{i,(j+1)}) \tag{74}$$

Here, we suppose the prediction (logits) are already sorted in a way that $j \leq j' \implies l_{i,(j)} \leq l_{i,(j')}$ (takes $O(K \log K)$), and that the risk of an empty set is zero[9] (i.e. $R_i(l_{i,K}) = 0$). Now, we could easily query $\hat{R}(\lambda) = F(-\lambda)$ in $O(\log NK)$ time, reducing the total time per update+query to $(K^2 + K\log K + (\log(NK))^2)$, or $O((\log N)^2)$ if $N \gg K$. An implementation of an online version of RCPS is also included in the supplementary material. We would like to emphasize that while `QuantileTree` is given, identifying the correct CDF to query (like in Eq. (73)) is crucial.

In fact, one can potentially improve the update time by first computing the max acceptable $\hat{R}$ given N and δ, denoted as $\hat{R}_{max}$, and then directly query $Q(\hat{R}_{max}; F)$, resulting in $O(K^2 + K\log K + \log\frac{B}{b} + \log(NK))$ time in total. Here $O(\log\frac{B}{b})$ is the complexity of finding the $\hat{R}_{max}$ by plugging in different risk values into the Hoeffding-Bentkus bound with a (continuous) binary search, with B denoting the width of the search interval and b denoting the tolerance. B is always less than or equal to the target risk, and b could be set to the minimum weight of the `QuantileTree` (which takes $O(1)$ to track per update). This version has a more similar flavor as Algorithm 2, in that it only query `QuantileTree` once per update. Depending on the specific distribution of interest, we might also see improved with this approach.

[9] If not, we just need to adjust the loss on a $\emptyset$ from all our estimate of $\hat{R}(\lambda)$.

C. Additional Experiment Details

In this section, we include additional experiment details, including results that cannot fit into the main paper.

C.1. Datasets

- `MNIST`: We first split the training set of the original MNIST (Lecun et al., 1998) into 90/10 train/validation split. We then create 54000 RGB images of shape $3 \times 48 \times 48$. For each image, we sample the 10 classes independently with probability 0.4. For each class, if it is to appear in the new image, we sample one original MNIST image from the training set and randomly assign it to a location and channel (the original MNIST is $1 \times 28 \times 28$). The 10000 new test images are generated the same way but with original images from the test set only.
- `MIMIC` (Johnson et al., 2016; Goldberger et al., 2000; Johnson et al., 2019): We select all admissions with an non-empty discharge report to simulate the task to predict missing diagnoses. For each admission, aggregate all ICD-9-CM diagnosis codes associated with the admission, map them to hierarchical condition category (HCC) code, and randomly delete the codes with a probability of 30%. We use the 3-digit version of ICD codes, and drop the bottom rare codes accounting for 3% of the total frequency. The deleted codes are used as target of prediction, and the un-deleted codes are taken as input to the model. Other input features used include patient features pre-processed like in (Harutyunyan et al., 2019), as well as clinical notes prepared using (Huang et al., 2019). We delete the typical section in clinical notes that list all diagnoses or disease name, so they do not give away the answer directly. We restricted the subset of HCC codes to the ones with at least 1000 patients, which is the same as using a code frequency cutoff of 0.8%. The precise list of HCC codes could be found in the code in our supplementary material. The risk factors are taken from `https://www.cms.gov/medicare/health-plans/medicareadvtgspecratestats/downloads/announcement2014.pdf`. Note that we present V_{TPC} and C_{FPC} normalized to $[0, 100]$. One could recover the total risk adjustment factor by using the C_{max}, which is 6.56 for `MIMIC` (select) and 16.48 for `MIMIC`.
- `Claim`: Insurance claim prediction on a proprietary dataset from a large healthcare data provider in North America. We kept only ICD-10-CM records (from 2015-10-01). Each claim will be treated as one sample. The input is all the ICD-10-CM codes in the past 26 weeks, and the target of prediction are the codes for this patient in the next 26 weeks. Like in `MIMIC`, we set the code frequency threshold at 0.8%. The precise list of HCC codes could be found in the code in our supplementary material. The risk factors are taken from `https://www.cms.gov/medicare/health-plans/medicareadvtgspecratestats/risk-adjustors/2023-model-software/icd-10-mappings` (2023 Midyear Software Model). For `Claim-S`, we used a temporal split of 75/25 (or the cutoff date of 2017-09-03) for training vs test set, resulting in 233679 samples for training and 77894 samples for testing. For `Claim`, the training set and test contains random but mutuallly exclusive patients. The total risk adjustment factor can be recovered by using C_{max} of 7.41 for `Claim` (select) and 14.87 for `Claim`.

C.2. Base Neural Networks

For each task, all methods presented in the paper are applied on the same base classifiers $\hat{\mathbf{p}}$, which are neural networks further described below:

- `MNIST`: We used a Convolutional Neural Network consisting of three `Conv-BatchNorm-ReLU-MaxPool` blocks (Ioffe & Szegedy, 2015; LeCun et al., 2015), with a fully connected layer at the end. We train the model with ADAM (Kingma & Ba, 2014) with a learning rate of 1e-2 and batch size of 128 for 50 epochs.
- `MIMIC`: We used a ClinicalBERT (Huang et al., 2019) to encode the clinical notes. Features processed by (Harutyunyan et al., 2019) (including some biomarkers) are fed into a multi-layer perceptron. Finally, embeddings (as well as HCC codes that are not deleted) are concatenated and fed into the final prediction layers. We train the model with ADAM (Kingma & Ba, 2014) with a learning rate of 1e-5 and batch size of 16 for 50 epochs.
- `Claim` (and `Claim-S`): We used a model with two layers of self-attention (Vaswani et al., 2017) to aggregate code embeddings to visit embeddings, as well as visit embeddings to an overall embedding, which is then concatenated with demographic information before being fed into the final prediction layer. We train the model with ADAM (Kingma & Ba, 2014) with a learning rate of 1e-5 and batch size of 128 for 20 epochs.

For FPCP we need to train a DeepSets-based cost proxy, so we will further perform a 80/20 split on the training data, using 80% to train the base classifier and 20% to train the cost proxy. The exact training scripts and model definitions could be found in our code. All experiments are carried out with PyTorch (Paszke et al., 2019).

C.3. Additional Results

We include completed experiment results in Table 7 (Value for C_{FP} control), Table 8 (Value for C_{FPC} control), Table 9 (Excess Cost and Violation for C_{FP} control) and Table 10 (Excess Cost and Violation for C_{FPC} control). Methods in Table 8 and Table 10 could be considered ablation study of FavMac with different $\mathcal{U}$. Note that the "(NN)" suffix to method names means using the DeepSets-based cost proxy like FPCP to replace the $\hat{C}$ introduced in Section 4.2. The conclusions are essentially the same as those presented in the main paper.

Table 7. Mean value of the prediction sets with false positive control (expectation control as well as violation control). The higher the better.

			Variants of FavMac						
Expectation Control	ClassWise	InnerSet	Full	Prob (NN) (FPCP)	Value (NN)	Ratio (NN)	Prob	Value	Ratio
MIMIC (select)	2.137±0.081	0.869±0.064	2.186±0.074	2.159±0.071	2.241±0.078	2.245±0.081	2.188±0.066	**2.263±0.078**	2.261±0.078
MIMIC	1.443±0.053	0.338±0.029	–	1.564±0.044	1.607±0.047	1.610±0.047	1.587±0.046	**1.636±0.049**	**1.637±0.049**
Claim (select)	1.829±0.113	1.446±0.065	2.061±0.109	2.022±0.102	2.094±0.114	**2.106±0.116**	2.057±0.101	2.108±0.113	**2.113±0.112**
Claim	1.168±0.071	0.682±0.040	–	1.179±0.067	1.219±0.067	**1.223±0.068**	1.191±0.067	**1.218±0.064**	**1.221±0.065**
Claim-S (select)	1.892	1.523	2.147	2.043	2.165	2.170	2.100	2.186	**2.190**
Claim-S	1.224	0.747	–	1.209	1.264	1.266	1.242	1.285	**1.286**
MNIST	35.342±0.231	24.058±0.339	36.570±0.233	37.442±0.274	37.893±0.284	37.953±0.280	37.714±0.245	38.060±0.260	**38.203±0.259**
MNIST (GEN)	35.874±0.280	24.764±0.353	38.153±0.315	38.312±0.323	39.415±0.335	39.557±0.325	38.552±0.315	39.478±0.346	**39.637±0.337**
Violation Control									
MIMIC (select)	–	0.304±0.039	2.188±0.073	2.080±0.072	2.165±0.076	2.174±0.082	2.116±0.063	**2.211±0.074**	2.206±0.075
MIMIC	–	0.137±0.022	–	1.411±0.043	1.480±0.046	1.467±0.043	1.559±0.047	1.610±0.048	**1.611±0.048**
Claim (select)	–	1.016±0.049	2.067±0.110	1.997±0.098	2.071±0.114	**2.082±0.118**	2.027±0.096	2.081±0.110	**2.085±0.114**
Claim	–	0.420±0.020	–	0.910±0.052	0.955±0.054	0.963±0.053	1.172±0.066	1.203±0.063	**1.206±0.063**
Claim-S (select)	–	1.089	2.152	2.025	2.147	2.150	2.072	2.163	**2.167**
Claim-S	–	0.466	–	0.964	1.021	1.022	1.226	1.271	**1.272**
MNIST	–	16.120±0.538	35.262±0.212	36.390±0.264	36.926±0.293	36.974±0.304	36.695±0.219	37.131±0.240	**37.310±0.249**
MNIST (GEN)	–	16.836±0.573	36.886±0.298	37.256±0.303	38.488±0.322	38.630±0.341	37.509±0.295	38.614±0.335	**38.804±0.338**

Table 8. Mean value of the prediction sets with continuous cost (C_{FPC}) control (expectation control as well as violation control). The higher the better. The difference is only the $\mathcal{U}$ being used.

	Expectation Control				Violation Control			
	Full	Prob	Value	Ratio	Full	Prob	Value	Ratio
MIMIC (select,TP)	3.492±0.102	**3.628±0.104**	**3.628±0.104**	**3.628±0.109**	3.398±0.097	3.322±0.094	3.322±0.094	**3.427±0.102**
MIMIC (TP)	–	1.914±0.025	1.914±0.025	**1.948±0.026**	–	1.837±0.027	1.837±0.027	**1.891±0.025**
Claim (select,TP)	4.506±0.162	4.515±0.160	4.515±0.160	**4.555±0.163**	4.424±0.157	4.438±0.159	4.438±0.159	**4.451±0.161**
Claim (TP)	–	1.849±0.082	1.849±0.082	**1.878±0.086**	–	1.808±0.080	1.808±0.080	**1.839±0.083**
Claim-S (select,TP)	4.506	4.525	4.525	**4.545**	4.430	**4.462**	**4.462**	4.453
Claim-S (TP)	–	1.856	1.856	**1.877**	–	1.825	1.825	**1.847**
MNIST (TP)	34.625±0.263	36.801±0.287	36.801±0.287	**37.046±0.289**	31.103±0.225	32.760±0.264	32.760±0.264	**33.485±0.254**
MIMIC (select,TPC)	1.935±0.048	2.089±0.053	**2.122±0.062**	2.089±0.053	**1.855±0.046**	**1.856±0.046**	1.721±0.048	**1.856±0.046**
MIMIC (TPC)	–	**1.588±0.040**	1.499±0.040	**1.588±0.040**	–	**1.513±0.041**	1.376±0.036	**1.513±0.041**
Claim (select,TPC)	2.045±0.084	**2.080±0.087**	2.019±0.097	**2.080±0.087**	1.994±0.080	**2.020±0.082**	1.879±0.078	**2.020±0.082**
Claim (TPC)	–	**1.238±0.064**	1.181±0.060	**1.238±0.064**	–	**1.206±0.062**	1.141±0.058	**1.206±0.062**
Claim-S (select,TPC)	2.077	**2.115**	2.111	**2.115**	2.028	**2.051**	1.972	**2.051**
Claim-S (TPC)	–	**1.272**	1.243	**1.272**	–	**1.245**	1.204	**1.245**
MNIST (TPC)	33.871±0.221	**36.649±0.248**	35.987±0.256	**36.649±0.248**	30.024±0.220	**32.374±0.261**	30.759±0.290	**32.374±0.261**
MNIST (GEN)	36.542±0.272	37.598±0.312	38.386±0.341	**38.870±0.321**	33.170±0.269	33.458±0.302	34.398±0.358	**35.645±0.335**

Table 9. Excess cost and violation frequency for false positive control. Values lower than 0 (for excess cost) or $\delta = 0.1$ (for violation) are marked red. Note we do not mark `Claim-S` as we cannot repeat the experiment. However, the conclusion is similar to `Claim`.

			Variants of `FavMac`						
Excess Cost	`ClassWise`	`InnerSet`	`Full`	`Prob` (NN) (`FPCP`)	`Value` (NN)	`Ratio` (NN)	`Prob`	`Value`	`Ratio`
`MIMIC` (select)	-2.32±0.46	-26.58±0.13	-2.51±0.06	0.02±0.15	-0.01±0.15	-0.00±0.15	0.00±0.16	-0.02±0.16	-0.01±0.15
`MIMIC`	-0.91±0.36	-27.10±0.05	–	-0.04±0.08	-0.04±0.07	-0.04±0.07	-0.05±0.06	-0.05±0.05	-0.05±0.05
`Claim` (select)	-2.48±0.51	-23.40±0.26	-2.42±0.09	-0.03±0.19	-0.03±0.17	-0.02±0.18	-0.04±0.20	-0.03±0.19	-0.04±0.20
`Claim`	-1.15±0.57	-24.40±0.19	–	-0.03±0.06	-0.03±0.06	-0.03±0.06	-0.06±0.06	-0.05±0.05	-0.05±0.05
`Claim-S` (select)	-2.68	-24.00	-2.32	0.01	0.00	0.00	0.00	-0.00	-0.00
`Claim-S`	-1.21	-24.61	–	0.03	0.03	0.03	0.01	0.01	0.01
`MNIST`	-13.14±0.25	-27.09±0.11	-6.78±0.10	-0.08±0.10	-0.09±0.09	-0.09±0.10	-0.06±0.11	-0.08±0.11	-0.07±0.12
`MNIST` (GEN)	-13.14±0.25	-27.09±0.11	-6.76±0.11	-0.08±0.10	-0.10±0.13	-0.10±0.12	-0.06±0.11	-0.10±0.07	-0.08±0.09
Violation Frequency (in %)									
`MIMIC` (select)	–	0.13±0.02	7.32±0.37	10.05±0.49	10.07±0.44	10.01±0.44	10.21±0.40	10.08±0.27	10.13±0.32
`MIMIC`	–	0.01±0.00	–	10.00±0.48	10.00±0.46	10.05±0.52	9.96±0.60	9.94±0.55	9.95±0.55
`Claim` (select)	–	0.30±0.07	7.38±0.73	9.94±0.69	10.02±0.62	10.03±0.64	9.83±0.87	9.74±0.73	9.75±0.73
`Claim`	–	0.06±0.02	–	10.08±0.34	10.11±0.32	10.12±0.34	9.76±0.53	9.89±0.44	9.84±0.45
`Claim-S` (select)	–	0.30	7.32	9.97	10.01	9.97	9.91	10.01	9.96
`Claim-S`	–	0.07	–	10.22	10.27	10.26	10.18	10.21	10.22
`MNIST`	–	0.10±0.03	1.55±0.12	9.88±0.42	9.81±0.27	9.85±0.34	9.97±0.54	9.92±0.43	9.91±0.39
`MNIST` (GEN)	–	0.10±0.03	1.57±0.14	9.88±0.42	9.73±0.33	9.83±0.40	9.97±0.54	9.97±0.52	10.08±0.48

Table 10. Excess cost and violation frequency for continuous cost (C_{FPC}) control. Values lower than 0 (for excess cost) or $\delta = 0.1$ (for violation) are marked red. Note we do not mark `Claim-S` as we cannot repeat the experiment. However, the conclusion is similar to `Claim`.

	Expectation Control				Violation Control			
	`Full`	`Prob`	`Value`	`Ratio`	`Full`	`Prob`	`Value`	`Ratio`
`MIMIC` (select,TP)	-3.49±0.05	-0.00±0.12	-0.00±0.12	0.00±0.08	4.35±0.26	10.04±0.42	10.04±0.42	10.08±0.35
`MIMIC` (TP)	–	-0.04±0.07	-0.04±0.07	-0.06±0.06	–	9.84±0.50	9.84±0.50	9.87±0.53
`Claim` (select,TP)	-3.14±0.06	-0.05±0.08	-0.05±0.08	-0.05±0.10	5.65±0.44	9.92±0.59	9.92±0.59	9.77±0.62
`Claim` (TP)	–	-0.06±0.07	-0.06±0.07	-0.05±0.05	–	9.71±0.44	9.71±0.44	9.70±0.41
`Claim-S` (select,TP)	-3.15	0.01	0.01	0.02	5.11	9.84	9.84	9.82
`Claim-S` (TP)	–	0.01	0.01	0.01	–	9.99	9.99	10.02
`MNIST` (TP)	-7.46±0.10	-0.08±0.09	-0.08±0.09	-0.08±0.07	3.46±0.11	9.90±0.38	9.90±0.38	9.92±0.32
`MIMIC` (select,TPC)	-2.71±0.05	-0.00±0.12	-0.06±0.17	-0.00±0.12	5.56±0.31	10.04±0.42	9.93±0.30	10.04±0.42
`MIMIC` (TPC)	–	-0.04±0.07	-0.05±0.06	-0.04±0.07	–	9.84±0.50	9.91±0.56	9.84±0.50
`Claim` (select,TPC)	-2.35±0.07	-0.05±0.08	-0.09±0.13	-0.05±0.08	7.31±0.56	9.92±0.59	9.97±0.31	9.92±0.59
`Claim` (TPC)	–	-0.06±0.07	-0.04±0.05	-0.06±0.07	–	9.71±0.44	9.90±0.30	9.71±0.44
`Claim-S` (select,TPC)	-2.48	0.01	-0.04	0.01	6.59	9.84	10.00	9.84
`Claim-S` (TPC)	–	0.01	0.03	0.01	–	9.99	10.16	9.99
`MNIST` (TPC)	-6.94±0.09	-0.08±0.09	-0.11±0.11	-0.08±0.09	3.98±0.14	9.90±0.38	9.82±0.32	9.90±0.38
`MNIST` (GEN)	-7.44±0.10	-0.08±0.09	-0.10±0.12	-0.08±0.10	3.45±0.09	9.90±0.38	9.85±0.41	9.92±0.34