

**Uncertainty-Aware Systems for Human-AI Collaboration:
Generative and Conformal Models in Dynamic,
Resource-Constrained Environments**

Vasco Thomas Serrão Pearson

Thesis to obtain the Master of Science Degree in

Data Science and Engineering

Supervisors: Prof. Mário Alexandre Teles de Figueiredo
Dr. Jacopo Bono

Examination Committee

Chairperson: Profa. Maria do Rosário de Oliveira Silva
Supervisor: Prof. Mário Alexandre Teles de Figueiredo
Member of the Committee: Prof. Miguel Jorge Couceiro de Sousa Santos

October 2024

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to express my deepest gratitude to my supervisors, Prof. Mário Figueiredo and Dr. Jacopo Bono, for their invaluable guidance and support throughout the development of this thesis. I am also sincerely thankful to Jean Alves, whose tireless assistance and encouragement were always there when I needed them most.

I extend my appreciation to Feedzai, particularly to Pedro Bizarro, for backing this research project. To my colleagues at Feedzai, thank you for the warm welcome and for making me feel at home. The opportunity to work alongside each of you has been an invaluable part of my experience over these past few months.

I am deeply grateful to my parents, Ester and Gareth, my brother Samuel, and to Joana, for their support and encouragement every step of the way.

Lastly, to my family and friends—thank you for making this journey, and all others, such an enjoyable ride.

Abstract

Although machine learning models are now ubiquitous in high-stakes decision-making scenarios, full automation is prevented by these model's limitations: their performance is often constrained by their training data, making them ineffective in the face of distribution shifts, and their lack of transparency undermines human trust in these systems. Human-AI collaboration (HAIC) has been proposed as a solution to these issues, leveraging the complementary strengths of human experts to mitigate the models' limitations. Despite its promise, the leading framework for HAIC, Learning to Defer (L2D), has a critical shortcoming: it struggles in dynamic environments where machine learning models fail to generalize to out-of-distribution data, often producing inaccurate predictions. Therefore, estimating model uncertainty correctly and leveraging humans' adaptability becomes essential in these scenarios. In this thesis, we propose two uncertainty-aware methods to enhance HAIC systems in dynamic environments. The first enhances L2D by employing distance-aware models, combining machine learning outputs with a density function to enhance robustness, thus providing reliable uncertainty estimates. The second method uses density-based conformal prediction to assess epistemic uncertainty, deciding whether an instance should be processed through L2D or deferred directly to human experts via rejection learning. We further extend both methods to handle cost-sensitive scenarios, limited human predictions, and capacity constraints. Using constraint programming, we optimize the assignments to ensure that each decision-maker, human or machine, handles the instances they are most likely to classify correctly. Our results demonstrate that both approaches outperform state-of-the-art baselines, particularly in dynamic environments, while also improving calibration.

Keywords

human-AI collaboration, learning to defer, uncertainty estimation, conformal prediction

Resumo

Modelos de aprendizagem automática são amplamente usados em decisões de alto risco, mas têm limitações que impedem uma automação total. O seu desempenho é condicionado pelos dados de treino, tornando-os ineficazes perante alterações de distribuição, e a sua falta de transparência reduz a confiança humana. A colaboração Humano-IA (HAIC) é vista como solução, tirando proveito das forças complementares de especialistas humanos para mitigar as limitações dos modelos. Apesar do seu potencial, a principal abordagem para HAIC, o Learning to Defer (L2D), enfrenta dificuldades em ambientes dinâmicos, onde modelos têm dificuldades em generalizar para dados fora de distribuição, frequentemente com previsões excessivamente confiantes. Nesta tese, propomos duas abordagens para aprimorar os sistemas HAIC em cenários dinâmicos. A primeira refina L2D através de modelos sensíveis à distância, que combinam as previsões do modelo de aprendizagem automática com uma função de densidade, aumentando a robustez e fornecendo estimativas de incerteza mais fiáveis. A segunda abordagem usa conformal prediction com uma função de densidade para avaliar a incerteza epistémica, decidindo entre processar uma instância através de L2D ou delegá-la diretamente a especialistas humanos via rejection learning. Ambos os métodos são estendidos para cenários sensíveis a custos, com previsões humanas limitadas e restrições de capacidade. Utilizando programação por restrições, otimizamos as atribuições para garantir que cada decisor, humano ou máquina, lide com as instâncias que mais provavelmente classificarão corretamente. Os nossos resultados demonstram que ambas as abordagens superam os métodos atuais, especialmente em ambientes dinâmicos, e melhoram a calibração.

Palavras Chave

colaboração humano-IA, learning to defer, estimação de incerteza, conformal prediction

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Contributions	3
1.3	Thesis Outline	4
2	Background and Related Work	5
2.1	Machine Learning for Decision Making	5
2.2	Human-AI Collaboration	7
2.3	Rejection Learning and Uncertainty Estimation	8
2.3.1	Rejection learning	8
2.3.2	Aleatoric and Epistemic Uncertainty	9
2.3.3	Uncertainty Estimation Methods	11
2.4	Learning to Defer	21
2.5	Limitation of Learning to Defer	25
3	Enhancing Uncertainty Estimation in L2D with Density Softmax	29
3.1	Data	30
3.2	Density-softmax	31
3.3	Learning to Defer	33
3.3.1	Classifier	35
3.3.2	Rejector	36
3.4	Dealing with Epistemic Uncertainty in Expert Modeling	36
4	Conformal Prediction for Human-AI Collaboration	39
4.1	Density-Based Conformal Prediction	40
4.2	Learning to Defer and Rejection Learning	41
5	Cost-Sensitive Learning and Capacity Constraints	45
5.1	Cost-Sensitive Learning	45
5.2	Human Work Capacity Constraints	47

6	Experimental Setup for Financial Fraud Detection	53
6.1	Dataset	53
6.2	Misclassification Costs	54
6.3	Alert Data Setup	55
6.4	Noise injection	56
6.5	Synthetic Expert Decision Generation	57
6.6	Expert Properties	59
6.7	Data Availability and Capacity Constraints	59
6.8	Baselines	61
6.9	System Training	61
6.9.1	Density-Based Conformal Prediction	61
6.9.2	Density-Softmax	62
6.9.3	Learning to Defer	62
7	Results	63
7.1	L2D System Performance and Calibration	63
7.1.1	Classifier	63
7.1.2	Expert Decision Modeling	64
7.2	MLP and Feature Extraction	65
7.3	Density-Softmax	66
7.4	Density-Based Conformal Prediction	67
7.4.1	Kernel Density Estimation	67
7.4.2	Conformal Coverage	69
7.5	Assignment System Performance and Calibration	70
8	Conclusions and Future Work	77
8.1	Future Work	78
	Bibliography	78
A	Hyperparameter Selection	89
A.1	Alert Model	89
A.2	MLP	89
A.3	RealNVP models	90
A.4	Learning to Defer models	90
B	Extended Results	93
B.1	Assignment system extended results	93
B.2	Null set predictions	93

List of Figures

2.1	Rejection learning architectures [41].	9
2.2	Illustration of ambiguity and novelty rejection [41]	10
2.3	Context dependent aleatoric and epistemic uncertainty [43]	10
2.4	Sources of epistemic uncertainty [43]	11
2.5	Virtual ensemble [60].	14
2.6	Separating epistemic uncertainty from total uncertainty [60]	15
2.7	Density-Softmax model [10].	17
2.8	Comparison of various uncertainty estimation methods on a toy dataset [10].	18
2.9	Classification boundaries for density-based conformal prediction and baselines on the Iris dataset [37].	20
2.10	Performance plot of density-based conformal prediction as a function of α [37].	21
2.11	Joint training of the classifier and the rejector [64].	23
2.12	Charusaie et al.'s active learning approach for modeling human performance [11].	27
4.1	Assignment system.	40
5.1	Density-based conformal coverage.	50
6.1	Timeline for training and deployment of the alert model and assignment system.	55
6.2	ROC curve for the alert model.	56
6.3	Normalized weight feature vector.	59
6.4	Fraction of instances in which the row expert is correct and column expert is incorrect.	60
7.1	ROC and calibration curves for classifier h	64
7.2	Mean ROC-AUC and ECE for estimates of $\mathbb{P}(y_i = m_{j,i})$	64
7.3	t-SNE visualization of the encoded features.	65
7.4	t-SNE visualization of the density scores.	66
7.5	Density-Softmax Probabilities.	67

7.6	ROC and calibration curves for classifier h in the density-softmax approach.	68
7.7	Mean ROC-AUC and ECE for estimates of $\mathbb{P}(y_i = m_{j,i})$ in the density-softmax approach.	68
7.8	Conformal coverage.	69
7.9	Noisy data detection by conformal prediction.	70
7.10	Misclassification Cost vs Deferral Rate.	73
7.11	ECE vs Coverage Level.	75

List of Tables

7.1	Comparison of deferral strategies under different noise settings.	71
7.2	Comparison of the Density-Softmax and Learning to Defer strategies under different noise and deferral rate settings.	74
A.1	Alert Model: LightGBM hyperparameter search space	89
A.2	Feature Extraction MLP: Hyperparameter Search Space	90
A.3	RealNVP: Hyperparameter Search Space	90
A.4	ML classifier: LightGBM hyperparameter search space	91
A.5	Expert Models: LightGBM hyperparameter search space	91
B.1	Expected Misclassification Cost per 100 instances for each setting - part 1	94
B.2	Expected Misclassification Cost per 100 instances for each setting - part 2	95
B.3	Expected Misclassification Cost per 100 instances for each setting - part 3	96
B.4	Expected Misclassification Cost per 100 instances for each setting - part 4	97
B.5	Expected Misclassification Cost per 100 instances for each setting - part 5	98
B.6	Percentage of null set predictions	99

Acronyms

AI	artificial intelligence
BAF	bank-account-fraud
CNN	convolutional neural network
ECE	expected calibration error
FN	false negative
FNR	false negative rate
FP	false positive
FPR	false positive rate
GDPR	General Data Protection Regulation
GBDT	gradient boosted decision tree
KDE	kernel density estimation
L2D	learning to defer
L2D-Pop	learning to defer to a population
ML	machine learning
MCP	maximum class probability
MCD	Monte-Carlo dropout
MLP	multi-layer perceptron
OvA	one vs. all
OOD	out-of-distribution
RealNVP	real-valued non-volume preserving
ROC	receiver operating characteristic
ROC-AUC	area under the receiver operating characteristic curve

SGLB	stochastic gradient Langevin boosting
TPE	tree-structured parzen estimators
t-SNE	t-distributed Stochastic Neighbor Embedding
XAI	explainable artificial intelligence

Chapter 1

Introduction

1.1 Motivation

In recent years, advances in artificial intelligence (AI) and, particularly, machine learning (ML) have produced models that match or even surpass human experts' accuracy across a broad range of tasks. These technologies have been successfully deployed in many high-stakes decision-making areas such as finance, criminal justice, and healthcare. In finance, ML models are used in applications like credit scoring [50] and fraud detection [7], where they efficiently analyze large datasets to identify risks and anomalies. In criminal justice, AI models are employed to assess recidivism risk and influence sentencing decisions [31]. Similarly, in medicine, ML models assist in diagnosing diseases such as diabetic retinopathy [35], cardiac arrest [81], and skin cancer [24], often achieving diagnostic accuracy comparable to or surpassing that of experienced medical professionals [35]. The strengths of ML models—particularly their speed, scalability, and capacity to leverage vast amounts of data to make accurate predictions—make them relevant tools in these sectors.

Despite these strengths, ML models exhibit limitations that restrict their applicability in high-stakes domains, where decisions have the potential to significantly impact human lives. These models are highly reliant on the data used during training and may struggle to maintain performance when faced with data distributions that differ from those previously encountered [28, 65]. This lack of generalizability can lead to significant errors in dynamic environments with constantly evolving patterns. Additionally, widely adopted ML models such as neural networks are often opaque in their decision-making processes, with complex “black-box” algorithms providing little insight into how specific decisions are reached [76]. This lack of interpretability can lead to trust and accountability concerns, particularly in sensitive fields like healthcare and criminal justice. Consequently, deploying ML systems as fully automated decision-makers is often impractical or outright illegal [2] for high-stakes applications, where

accuracy and interpretability are critical.

To address these limitations, human-AI collaboration systems have been proposed, combining the complementary strengths of humans and ML models [18, 20]. AI systems excel at processing vast amounts of data quickly and consistently, while humans bring unique abilities to the table: they can access information beyond what is available to the model, adapt to new and evolving circumstances, engage in causal reasoning [32], and provide explanations for their decisions. This makes human decision-makers better equipped to handle tasks requiring flexibility and interpretability. Conversely, humans are slower and may be less consistent, especially when faced with complex statistical information, where ML models excel [17, 34]. Human-AI collaboration leverages these complementary strengths, creating decision-making systems that are potentially more accurate and resilient than either humans or machines alone.

A fundamental challenge in designing human-AI collaboration systems is determining how to best assign each instance to either the AI model or a human expert to maximize the system's overall effectiveness. A simple approach, known as rejection learning [15], involves the ML model refraining from making predictions when it detects high uncertainty, thereby deferring these decisions to human experts. However, rejection learning only considers the ML model's confidence and does not take into account the potential strengths or weaknesses of the human decision-makers. To improve upon this, the learning to defer (L2D) framework was developed as a more sophisticated instance-assignment strategy [11, 39, 59, 63, 64]. L2D not only assesses the ML model's confidence in making a prediction but also estimates human experts' confidence levels, assigning each instance based on the estimated performance of each decision-maker, human or otherwise. This approach optimizes instance assignment by actively balancing the strengths of both human and AI decision-makers.

Despite its promise, L2D faces challenges in practical applications. A significant unresolved issue in L2D systems is their limited adaptability in dynamic environments, where data distributions evolve, rendering prior model assumptions inaccurate [28]. In performative prediction scenarios, where system outputs influence future data distributions—such as in fraud detection where fraudsters adapt to detection strategies—L2D systems may struggle to maintain performance [69]. Additionally, human experts' decision-making processes can change based on new information and experiences, further complicating the model's capacity to predict human accuracy. Therefore, to maintain robustness, a L2D system must dynamically adapt to these non-stationary elements, a necessity that current research has not adequately addressed.

Other challenges also hinder the broader applicability of L2D [55]. Many L2D methods require extensive datasets containing every experts' prediction on every instance, which are often unavailable due to constraints on human capacity or prohibitive data acquisition costs. Additionally, L2D systems typically do not account for human work capacity constraints when making assignments. In real-world

applications, human experts are limited in the rate of instances they can process, yet most current L2D methods lack mechanisms to ensure that assignments are feasible within these limitations. Furthermore, in cost-sensitive applications such as fraud detection, it is critical to account for the varying costs of misclassifications. Current research focuses only on accuracy maximization, disregarding scenarios where cost-sensitive objectives apply. This is particularly common in high-stakes decision-making scenarios such as medicine and fraud detection, where errors carry unequal consequences.

This dissertation addresses the main limitations of current L2D systems by developing two robust assignment frameworks for human-AI collaboration. Specifically, it focuses on handling dynamic environments by incorporating strategies that respond to data drift, while also overcoming challenges related to human work capacity constraints, data availability, and cost-sensitive decisions.

1.2 Contributions

This thesis makes three contributions to advancing human-AI collaboration systems for assignment and decision-making in dynamic environments. The first contribution is an enhanced, distance-aware L2D system designed to improve robustness in settings where data distributions may shift. By incorporating distance-aware models [10], this approach leverages density-based adjustments to account for distributional uncertainty, reducing overconfidence in unfamiliar regions, and enhancing overall calibration. This allows the assignment mechanism to make more informed deferral decisions by better estimating the confidence levels of both the ML model and human experts. Additionally, the system uses constraint programming to compute the optimal set of assignments, taking into account misclassification costs and human work capacity constraints.

The second contribution is a hybrid system that uses density-based conformal prediction [37, 62] to balance rejection learning with L2D in dynamic environments. This approach applies conformal prediction to identify instances likely to fall outside the training data distribution and defers them directly to human experts, using a rejection learning strategy. For in-distribution data, the system employs L2D to assign instances based on reliable correctness estimates for all decision-makers. Like the first system, this hybrid system handles work capacity constraints and cost-sensitive scenarios, using constraint programming to compute assignments that take into account the relative costs of errors and the human decision-makers' capacity.

The third contribution is an experimental study within a cost-sensitive fraud detection context to validate the effectiveness of the proposed methods. To create realistic testing scenarios, we generate synthetic fraud analysts with varying behaviors and feature dependencies, simulating real-world traits of human decision-making, including algorithmic bias [4]. In this human-AI collaboration setup, our proposed systems are tasked with assigning alerts flagged by an ML model. We evaluate our systems

under a wide array of conditions, introducing different magnitudes and types of noise in the test set to simulate data drift, varying the number of instances that can be deferred to the expert team, and altering the number of experts available. Additionally, we vary the data availability to simulate scenarios with limited expert labels at training time. Our evaluation compares the performance of the proposed methods against a state-of-the-art L2D method, as well as against rejection learning and random deferral baselines, demonstrating that our approaches outperform the baselines across a set of diverse, realistic settings. To summarize, our contributions are as follows:

- **Distance-aware L2D system:** a L2D approach incorporating distance-aware models to enhance calibration and robustness against out-of-distribution (OOD) data, with the ability to compute optimal assignments under misclassification costs and work capacity constraints.
- **Conformal prediction for human-AI collaboration:** a hybrid system that uses density-based conformal prediction to balance rejection learning and L2D for adaptive deferral to human experts under distribution shifts, while also accommodating cost and capacity constraints.
- **Empirical validation in a fraud detection setting:** an extensive experimental evaluation in a cost-sensitive fraud detection setting, featuring synthetic fraud analysts and a range of testing conditions. This study shows improved performance over baseline methods across diverse scenarios, including noise, data availability, and work capacity constraints.

1.3 Thesis Outline

This dissertation is organized as follows. In Chapter 2, we cover key concepts in human-AI collaboration, rejection learning, uncertainty estimation, and L2D, laying the foundation for the thesis. The next chapters focus on the two main contributions: Chapter 3 introduces distance-aware models to handle distribution shift in L2D, while Chapter 4 discusses using density-based conformal prediction to guide the deferral strategy between L2D and rejection learning. Chapter 5 addresses practical constraints by optimizing assignments based on cost and capacity limits. The methodology for testing the proposed models in a simulated fraud detection setting is detailed in Chapter 6. Chapter 7 presents and discusses the performance and calibration of the models. The thesis concludes with a summary of findings and directions for future research.

Chapter 2

Background and Related Work

2.1 Machine Learning for Decision Making

ML has rapidly evolved in recent years, due to its ability to transform vast datasets into actionable insights efficiently and at scale. This capability has rendered ML, particularly supervised learning, a valuable tool across various fields that depend on data-driven decisions. In supervised learning, the objective is to predict an output variable y from input features x . This involves learning a hypothesis h that models the relationship between x and y , from training data $\mathcal{D} = (x_1, y_1), \dots, (x_n, y_n)$, which consists of a collection of feature-outcome pairs.

Decision-making is a process in which individuals or organizations leverage available data and information to select the best among various options. This typically involves the collection of relevant data, the analysis of trends and patterns, and the application of reasoning to predict or classify outcomes. The application of supervised learning in this context is particularly valuable, as evidenced by its success across multiple domains. For example, in healthcare, convolutional neural networks (CNNs) have demonstrated the capability to detect melanoma from images with higher accuracy than dermatologists [9]. CNNs have also been shown to make more accurate referral recommendations in cases of retinal disease, highlighting their potential in clinical settings [25]. In the financial sector, Khandani et al. [50] show that ML can predict credit risk based on a client's financial status and spending history, while Awoyemi et al. [7] show its effectiveness in detecting credit card fraud.

Although ML has been successful across various decision-making domains, it has its drawbacks and limitations. One key limitation is that ML models can become obsolete due to distribution drifts [28]. Since an ML model relies solely on its training data, its performance can deteriorate significantly when the distribution of this data no longer reflects the real-world environment in which it is used. This creates a need for alternatives. In contrast, humans possess the ability to reason through causal relationships

[32] and are not confined to the data used during a training phase. This leads some researchers to argue that, in many circumstances, humans are far more adaptable and robust decision-makers than ML models.

Another significant limitation of ML models is their inability to provide explanations for their decisions [76]. In many decision-making contexts, it is crucial to be able to justify how a decision was reached and why it was considered the best option. Indeed, under the European Union General Data Protection Regulation (GDPR), individuals affected by automated algorithmic decisions have the right to an explanation. This right encompasses understanding the reasons behind a decision and the ability to appeal it. While simpler statistical models like linear or logistic regression can represent their decisions as a weighted sum of input features, the shift towards more complex models has led to what is often referred to as the “black box problem”. This term denotes the substantial challenges in understanding and explaining how these models obtain their predictions. This “opacity” raises increasing concerns about human trust in these models, as the lack of explainability can be a critical barrier to their adoption and acceptance, particularly in high-stakes scenarios like healthcare and finance.

Research into explainability, leading to the development of explainable artificial intelligence (XAI) methods, is driven by several key perspectives [76]. The regulatory perspective emphasizes compliance with laws like the GDPR, mandating that AI systems provide “right to explanation” to ensure fairness. From a scientific standpoint, explainability is crucial for uncovering new insights and understanding complex model behaviors. Industrially, there is a focus on balancing performance with transparency to maintain user trust and meet regulatory standards. The developmental perspective highlights the role of explainability in debugging and refining AI systems, helping developers to improve accuracy and fairness. Finally, the end-user and social perspective addresses the broader societal impacts, stressing the importance of trust and fairness in AI decisions, ensuring they are equitable and just.

The development of XAI has introduced techniques that allow for a better understanding of predictions made by black-box models. Notable among these are methods such as LIME [75] and SHAP [57], which help to clarify the influence of individual features on a model’s prediction. Despite their potential, the practical implementation of XAI methods is not without complications [19], as the process of explaining an AI’s decision-making process is not straightforward. An example of this is the use of criminal recidivism prediction algorithms in the United States. While these systems deliberately exclude variables like race, gender, employment, and residence to minimize biases and aim for objectivity, research shows that such systems still predict higher recidivism rates for black defendants [51], suggesting that outcomes are shaped by complex data interactions. This complexity makes it challenging for “explainers” to produce interpretations that are both accurate and comprehensible to the general public. As such, incorporating human decision-makers into the system provides a complementary path to fostering trust in AI systems. Rather than relying solely on automated explanation methods, human oversight can

introduce an added layer of accountability, ensuring that critical decisions are not made in isolation.

2.2 Human-AI Collaboration

Humans and AI possess distinct strengths and weaknesses in decision-making. ML models can quickly become outdated in the face of distribution shifts [28] and often lack the capability to explain their decision-making processes [76]. In contrast, humans have the ability to continuously learn and adapt to changes in their environment and, in some cases, can provide justifications for their decisions. However, humans are generally slower and less capable of processing large volumes of data when compared to ML models, which have demonstrated superior performance in several domains, such as detecting melanoma [9] and predicting recidivism rates [85].

Due to the complementary capabilities of humans and AI, several researchers advocate for a collaborative approach to enhance decision-making performance in some applications. A common method involves an *algorithm-in-the-loop* process, where an ML model provides insights to a human who ultimately makes the final decision [33]. Such systems are increasingly common in everyday settings, such as judges using risk assessments for sentencing decisions [51], banks employing models to manage credit risks [50], and caseworkers deciding whether to investigate potential child maltreatment based on hotline reports [18]. This process allows for human oversight, thus ensuring accountability and trust in these systems.

Building on the complementary strengths of humans and AI in decision-making, hybrid intelligence systems [20] have been proposed as a way to further optimize the combined strengths of humans and AI. By distributing tasks between algorithmic and human agents, these systems aim to solve complex problems more effectively than either humans or AI could independently. The challenge in these systems lies in determining the optimal instance distribution between human and AI agents.

One of the first proposed methods for distributing tasks between humans and AI is the *rejection learning* framework [13, 15]. In this approach, models are trained with the option to abstain from decision-making at the cost of a rejection loss. Consequently, the ML model enhances its performance on instances it does not reject, while the rejected instances are typically redirected to an alternative decision-maker, often a human. However, Madras et al. [59] critique this method for its lack of consideration of the downstream decision-maker, pointing out that it fails to account for situations where the human decision-maker may be equally uncertain. They argue that it would be more effective to allocate tasks to humans in scenarios where they are likely to outperform the ML model and vice-versa. To address this, they propose an approach called L2D, which assigns instances based on the estimated performance of each decision-maker, thereby optimizing the collaboration between human and AI strengths. We further delve into this approach in Section 2.4.

2.3 Rejection Learning and Uncertainty Estimation

2.3.1 Rejection learning

Standard ML models always produce predictions, regardless of their certainty or the potential for error. In real-world high-stakes applications, such as medical diagnosis [93] or autonomous driving [88], where incorrect predictions can lead to serious consequences, this characteristic can be problematic. It is essential for models to recognize their limitations and refrain from making predictions in cases of high uncertainty. The rejection of these instances leads to better performance of the models on instances selected for prediction. This branch of ML is known as *machine learning with a reject option*, or *rejection learning*, where models are designed to abstain from making predictions in cases where they are uncertain.

Rejection learning can be dated back to 1970, when Chow [13] first studied the trade-off between error rate and rejection rate. This approach has gained interest in recent years with a number of papers developing the framework [15, 29]. In rejection learning systems, the output space of the model is extended to include a new value $\textcircled{\text{R}}$ corresponding to abstaining to predict. This new output value can be seen as an additional class [15, 29]. Formally a model with a reject option $m : \mathcal{X} \rightarrow \mathcal{Y} \cup \{\textcircled{\text{R}}\}$,

$$m(x) = \begin{cases} \textcircled{\text{R}} & \text{if the prediction is rejected;} \\ h(x) & \text{if the prediction is accepted.} \end{cases} \quad (2.1)$$

can be represented by a pair (h, r) , where $h : \mathcal{X} \rightarrow \mathcal{Y}$ is the predictor and $r : \mathcal{X} \rightarrow \mathbb{R}$ is the rejector. At test time, the model m outputs $\textcircled{\text{R}}$ when the rejector r determines that the predictor h has high chance of making a mistake.

In their survey of rejection learning, Hendrickx et al. [41] identify three prevalent architectures of rejection learning systems (Figure 2.1): a separate rejector architecture, where the rejector functions independently of the predictor; a dependent rejector architecture, in which the rejector's decision is based on the predictor's output; and an integrated rejector architecture, which treats rejection as an additional class, merging the rejector and predictor into a single model.

Hendrickx et al. [41] categorize rejection into two distinct types. *Ambiguity rejection* enables the model to withhold predictions in scenarios where the target variable y is ambiguous. This can arise from the inherent properties of the relationship between features and label $P(Y|X)$, such as class overlap in specific regions of the instance space for classification tasks, or compromised training data (i.e. incorrectly labeled instances). Conversely, *novelty rejection* permits the model to refrain from predicting on instances that deviate significantly from the training data. During testing, if $P(X, Y)$ shifts away from the training distribution, the training data may become unrepresentative. Additionally, certain rare out-of-distribution examples might not be included in the training dataset due to their inherent rarity.

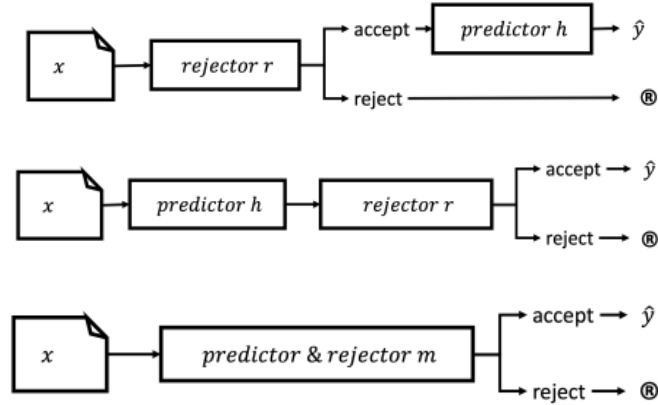


Figure 2.1: Different architectures of rejection learning systems [41]. Separated rejector (top); dependent rejector (middle); integrated rejector (bottom).

2.3.2 Aleatoric and Epistemic Uncertainty

Ambiguity rejection and novelty rejection are linked to the field of uncertainty quantification, which assesses the uncertainty in predictions made by ML models [27, 43]. Uncertainty quantification differentiates between *aleatoric uncertainty*, stemming from intrinsic randomness in the data (e.g. non-deterministic relationships between features and the target), and *epistemic uncertainty*, which corresponds to incomplete knowledge. An example of aleatoric uncertainty is the outcome of a coin toss, where the uncertainty is considered irreducible, that is, it cannot be reduced regardless of how much additional information is gathered. In contrast, epistemic uncertainty arises from a lack of knowledge and is thus reducible as more information becomes available. These forms of uncertainty are closely related with rejection learning. High aleatoric uncertainty leading to rejection is characteristic of ambiguity rejection scenarios (Figure 2.2 a)). Conversely, epistemic uncertainty can result in either novelty or ambiguity rejection. Novelty rejection (Figure 2.2 c)) occurs with out-of-distribution instances that differ significantly from the training data. However, when epistemic uncertainty is due to model uncertainty—uncertainty about the correct model to fit the data, thereby introducing high bias—it falls under ambiguity rejection (Figure 2.2 b)).

In ML, the distinction between aleatoric and epistemic uncertainties is often overlooked. However, in specific applications like rejection learning [13] or active learning [66], recognizing these two types of uncertainty can be essential. Senge et al. [77] highlight the importance of distinguishing these uncertainties within medical decision-making contexts, arguing that it allows for more reliable and interpretable decisions. Nguyen et al. [66] demonstrate that, in active learning, particularly in uncertainty sampling, epistemic uncertainty—which represents the reducible part of uncertainty—is more indicative of an instance’s utility than aleatoric uncertainty, which is irreducible.

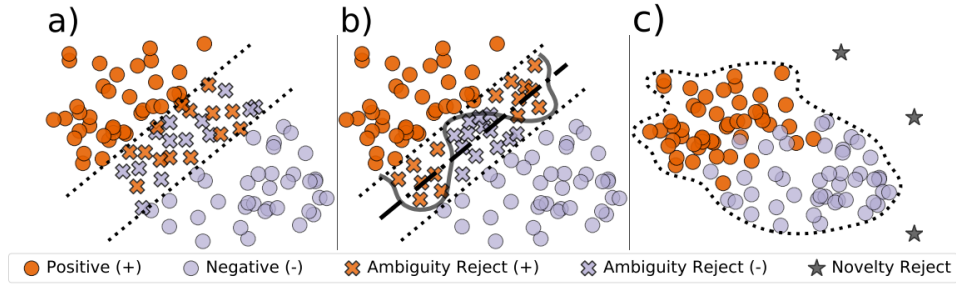


Figure 2.2: Illustration of ambiguity and novelty rejection [41]. The dotted lines represent the rejector, the dash-dotted line the fitted predictor h , and the solid line the ground-truth relation f . a) ambiguity rejection due to a non-deterministic relation between X and Y , b) ambiguity rejection due to the model bias, and c) an example of novelty rejection.

It is important to recognize that the concepts of aleatoric and epistemic uncertainty are context-dependent, defined within the framework $(\mathcal{X}, \mathcal{Y}, \mathcal{H}, \mathcal{P})$, where \mathcal{H} denotes the hypothesis space and \mathcal{P} represents the joint probability distribution over $\mathcal{X} \times \mathcal{Y}$. An illustrative example is shown in Figure 2.3, where two class distributions overlap in a low-dimensional space, leading to aleatoric uncertainty in specific regions of the feature space. This type of uncertainty can be mitigated by embedding the data in a higher-dimensional space, which may separate the classes and resolve the aleatoric uncertainty. Typically, moving data to a higher-dimensional space tends to decrease aleatoric uncertainty while increasing epistemic uncertainty, as the complexity of fitting a model increases and may require more data.

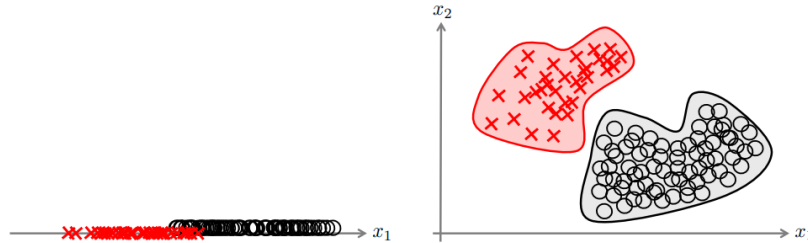


Figure 2.3: On the left, the classes overlap, causing aleatoric uncertainty. On the right this uncertainty is resolved by augmenting the feature space with an extra dimension [43].

Assuming a fixed setting $(\mathcal{X}, \mathcal{Y}, \mathcal{H}, \mathcal{P})$, uncertainty can stem from three distinct sources, with the first being aleatoric and the remaining two representing types of epistemic uncertainty:

1. *Aleatoric uncertainty* emerges if the relationship between \mathcal{X} and \mathcal{Y} is non-deterministic. Full knowledge of P does not eliminate uncertainty regarding the actual outcome y .
2. *Model uncertainty* arises when the best predictor within the hypothesis space h^* does not align with the pointwise Bayes predictor f^* . This misalignment, between h^* and f^* , highlights uncertainties related to choosing the correct model type to fit the data, and thus the choice of the hypothesis

space \mathcal{H} (Figure 2.4).

3. *Approximation uncertainty* refers to the gap between the hypothesis \hat{h} generated by the learning algorithm and the optimal hypothesis h^* . This gap is significantly influenced by the quality and quantity of the training data (Figure 2.4).

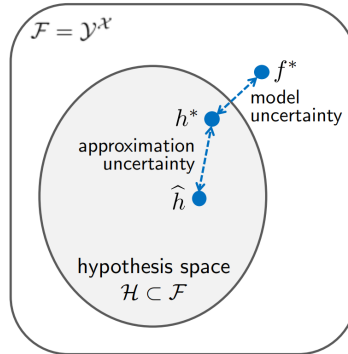


Figure 2.4: Illustration of the sources of epistemic uncertainty, where f^* represents the pointwise Bayes predictor, h^* indicates the optimal predictor within the hypothesis space, and \hat{h} denotes the predictor obtained through the learning algorithm. Model uncertainty is identified by the disparity between f^* and h^* , whereas approximation uncertainty is defined by the difference between \hat{h} and h^* [43].

2.3.3 Uncertainty Estimation Methods

An ideal rejection learning system must balance ambiguity and novelty rejection by creating a combined uncertainty metric to rank instances for rejection. ML models, while trained to predict correct classes on training data, often struggle with OOD data. Hein et al. [38] show that ReLU-based neural networks can give overconfident predictions on instances far from the training data. Similarly, k-nearest neighbors and decision trees assign high confidence to outliers because they assign the same probability as nearby instances or those in the same node of a tree. Uncertainty estimation methods seek to mitigate these issues by providing better confidence measures for the predictions of ML models.

Probabilistic Predictors and Calibration

Calibration ensures that a model's predicted probabilities correspond to the actual likelihood of events. For instance, if a probability estimate is a value between 0.6 and 0.7, we expect the empirical observed probability to be approximately between 60% and 70%. Good calibration is critical for providing reliable uncertainty estimates, particularly in high-stakes applications. One common metric for measuring calibration quality is the expected calibration error (ECE), which quantifies the difference between predicted probabilities and actual outcomes. It is computed by partitioning predictions into bins

based on their confidence and then comparing the average predicted probability to the actual frequency of positive outcomes in each bin:

$$ECE = \sum_{m=1}^M \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|, \quad (2.2)$$

where B_m is the set of samples in the m -th bin, $\text{acc}(B_m)$ is the accuracy of the samples in that bin, $\text{conf}(B_m)$ is the average confidence of predictions in the bin, and n is the total number of samples.

Many ML models function as probabilistic predictors, producing posterior probability estimates $p(y|x, h)$, rather than point predictions $h(x) \in \mathcal{Y}$. These probability estimates are often obtained by minimizing loss functions, such as the well-known cross-entropy loss, which are designed so that their minimization results in calibrated models, that is, models that produce accurate estimates of the “true” posterior probability distributions [30]. However, some models produce scores that are not “true” probabilities, such as the decision boundary margin in support vector machines or the relative vote counts in ensemble methods like bagging or boosting. To address this, calibration techniques are commonly used to convert these scores into better-calibrated probability estimates [36].

For probabilistic predictors, the model’s own confidence metrics can serve as a baseline for uncertainty estimation. Hendrycks and Gimpel [42] demonstrate that using the predicted class’s confidence — often measured by the maximum class probability (MCP) — can establish a simple benchmark for identifying OOD samples. However, this approach suffers from the previously mentioned propensity for overconfidence in OOD data, and primarily corresponds to aleatoric uncertainty.

Bayesian Inference

Bayesian inference presents a coherent probabilistic framework that allows for modeling uncertainty, and, for long, has been regarded as one of the preferred methods to perform uncertainty quantification. These methods provide insights into both epistemic and aleatoric uncertainty by learning a distribution over the hypothesis space. For instance, Bayesian neural networks [58] learn a probability distribution over the models’ weights and deep Gaussian processes [16], a variant of Bayesian neural networks, assume the weights follow a normal distribution and estimate their mean and variance from the training data. Since these models consider a probability distribution, instead of fixed values for the weights, their predictions are also probability distributions. Consequently, the uncertainty of each prediction can be quantified by taking said distribution’s variance. Depeweg et al. [21] propose differentiating aleatoric and epistemic uncertainties in Bayesian neural networks by calculating total uncertainty as the entropy of the predictive posterior $\mathcal{H}[p(y|x)]$ and aleatoric uncertainty as the expectation over the entropies of the predicted distributions $\mathbb{E}_{p(w|\mathcal{D})}[\mathcal{H}[p(y|x; w)]]$, with epistemic uncertainty being the difference between the two.

In spite of the aforementioned benefits, Bayesian models are known for their significant computational demands. To address this, Gal and Ghahramani [27] propose Monte-Carlo dropout (MCD) as an approximation of Bayesian inference within deep Gaussian processes, utilizing dropout combined with Monte-Carlo sampling. Their findings indicate that MCD effectively estimates uncertainty in deep learning models while maintaining manageable computational complexity and test accuracy. In this method, a neural network is trained with dropout regularization in the standard fashion. However, during prediction, dropout is still applied. The network then performs N stochastic forward passes for each input, creating an ensemble of predictions from the approximate deep Gaussian process. The final prediction is obtained by averaging these passes. The collected statistics from these multiple passes approximate those of the underlying deep Gaussian process, allowing the variance of these predictions to act as an estimate of point-wise uncertainty.

It is important to distinguish between point-wise uncertainty estimates, which focus on individual predictions, and the uncertainty about the predicted class itself. For example, if we apply MCD to two different instances and find that one has a mean prediction of 0.9 with a standard deviation of 0.05, while the other has a mean of 0.5 with a standard deviation of 0.01, the former exhibits higher point-wise uncertainty despite the latter being more uncertain about the predicted class. This distinction highlights a limitation of using only the variance from MCD to assess uncertainty without additional uncertainty measures. Geifman and El-Yaniv [29] compare MCD compared to the MCP baseline across three datasets: CIFAR-10, CIFAR-100, and ImageNet. Their evaluation using risk-coverage curves shows similar results for the CIFAR datasets, but MCP outperforms MCD on the ImageNet dataset, showing that relying solely on variance can misrepresent uncertainty. Corbière et al. [14] demonstrate that using the average of predictions from MCD's passes is a more effective uncertainty measure than variance. This method outperforms MCP on CIFAR-100, emphasizing that averaging predictions produces better uncertainty estimates than considering variance alone.

Ensembling Predictions

The concept of ensembling predictions for uncertainty estimation, as in MCD [27], has been revisited in several studies. Lakshminarayanan et al. [53] propose training multiple neural networks and averaging their softmax outputs to predict the class with the highest average probability. They achieve sufficient variability through random weight initialization and data shuffling, finding that bagging reduces performance. This aligns with Lee et al. [54], who showed that training deep ensembles on the full dataset with random initialization outperforms bagging, which reduces training data and increases the risk of networks converging to local minima. Although training multiple networks in parallel requires significant computational resources, their method surpasses MCD (using the mean of stochastic passes) on the MNIST and SVHN datasets. Additionally, their approach was tested on distinguishing between known

and unknown classes (e.g., MNIST versus NotMNIST and SVHN versus CIFAR-10), yielding lower confidence for OOD samples compared to single network predictions.

Shaker and Hüllermeier [79] suggest employing ensemble techniques to estimate the entropy measures defined by Depeweg et al. [21], representing the posterior $p(h|\mathcal{D})$ with an ensemble $\{h_1, \dots, h_M\}$. They estimate total uncertainty u_t as the entropy H of the average predictive distribution:

$$u_t(x) = H\left(\frac{1}{M} \sum_{i=1}^M p(y|h_i, x)\right) = - \sum_{y \in \mathcal{Y}} \left(\frac{1}{M} \sum_{i=1}^M p(y|h_i, x)\right) \log_2 \left(\frac{1}{M} \sum_{i=1}^M p(y|h_i, x)\right),$$

aleatoric uncertainty u_a as the average entropy of each individual model's predictions:

$$u_a(x) = \frac{1}{M} \sum_{i=1}^M H(p(y|h_i, x)) = -\frac{1}{M} \sum_{i=1}^M \sum_{y \in \mathcal{Y}} p(y|h_i, x) \log_2 p(y|h_i, x),$$

and epistemic uncertainty u_e as the difference between total and aleatoric uncertainty.

Malinin et al. [60] explore ensemble methods for gradient boosted decision tree (GBDT) models, recognizing the challenge posed by the limited randomness in gradient boosting compared to neural networks [53]. They propose three methods for creating GBDT ensembles. The first method involves using stochastic gradient boosting [26], where data is randomly subsampled at each iteration to generate independent models, introducing diversity within the ensemble. The second method, stochastic gradient Langevin boosting (SGLB) [86], integrates random Gaussian noise into the gradient updates to better explore the solution space. Both methods increase time and space complexity due to the need to train multiple models. To mitigate this, the third method creates a virtual ensemble by using truncated sub-models from a single GBDT model (Figure 2.5). This method differs from random forests, where trees are built independently, as the trees in a GBDT model are built sequentially. The virtual ensemble leverages this structure to form an ensemble without additional computational cost.

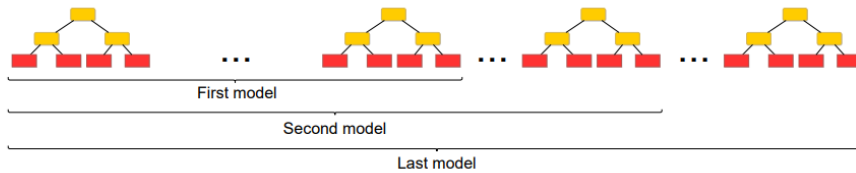


Figure 2.5: Virtual ensemble [60], where each tree is truncated sequentially from the root.

These three methods distinguish between epistemic and aleatoric uncertainty by following the same approach used by Shaker and Hüllermeier [79]. To estimate both types of uncertainty, the model parameters θ are assumed to be drawn from the posterior distribution $p(\theta|\mathcal{D})$, although this assumption is

fully satisfied only in the SGLB method. Total uncertainty is measured as the entropy of the predictive distribution of the target y conditioned on x and the dataset \mathcal{D} , while expected aleatoric uncertainty is the average entropy over the distribution of possible parameters θ . Epistemic uncertainty is then the difference between total and aleatoric uncertainty.

$$\underbrace{\mathcal{I}[y, \theta|x, \mathcal{D}]}_{\text{Epistemic Uncertainty}} = \underbrace{H[P(y|x, \mathcal{D})]}_{\text{Total Uncertainty}} - \underbrace{\mathbb{E}_{p(\theta|\mathcal{D})}[H[P(y|x; \theta)]]}_{\text{Expected Aleatoric Uncertainty}} \quad (2.3)$$

$$\approx H \left[\frac{1}{M} \sum_{m=1}^M P(y|x; \theta^{(m)}) \right] - \frac{1}{M} \sum_{m=1}^M H[P(y|x; \theta^{(m)})].$$

The authors test these methods on a synthetic dataset to evaluate their ability to distinguish between epistemic and aleatoric uncertainty. Figure 2.6 shows the results for an SGLB ensemble.

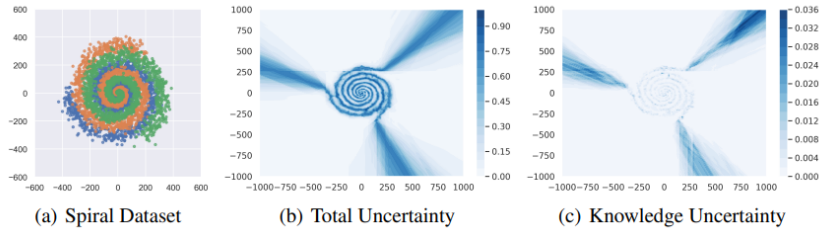


Figure 2.6: Separation of epistemic uncertainty (knowledge uncertainty) from total uncertainty in a synthetic dataset using a SGLB ensemble [60] Total uncertainty correctly detects class boundaries and ‘sectors’ of input space outside the training dataset. Estimates of knowledge uncertainty allow discrimination between out-of-domain regions and class boundaries

In their experiments using the CatBoost library [72] on UCI datasets, Malinin et al. [60] find that GBDT ensembles offer limited improvement in error detection over a single model, since GBDT is already an ensemble of trees. However, the ensembles provide valuable epistemic uncertainty estimates that are far more effective for OOD detection than measures of total uncertainty.

Reliable Classification

Senge et al. [77] were the first to differentiate between epistemic and aleatoric uncertainty in ML through their *reliable classification* approach. They introduce the normalized likelihood $\pi_{\mathcal{H}}$ for a hypothesis h in the hypothesis space \mathcal{H} as

$$\pi_{\mathcal{H}}(h) = \frac{L(h)}{L(h^{ml})} = \frac{L(h)}{\max_{h' \in \mathcal{H}} L(h')}, \quad (2.4)$$

where $L(h) = \prod_{i=1}^N p(y_i|h, x_i)$ is the likelihood of h , and h^{ml} is the hypothesis with the maximum likelihood within \mathcal{H} . For a particular instance x , they define the support $\pi(y|x)$ for the two classes $y \in \{0, 1\}$ as

$$\pi(1|x) = \sup_{h \in \mathcal{H}} \min[\pi_{\mathcal{H}}(h), p(1|h, x) - p(0|h, x)], \quad (2.5)$$

$$\pi(0|x) = \sup_{h \in \mathcal{H}} \min[\pi_{\mathcal{H}}(h), p(0|h, x) - p(1|h, x)]. \quad (2.6)$$

The value of $\pi(1|x)$ is high only if a plausible hypothesis significantly favors the positive class over the negative class, and conversely for $\pi(0|x)$. Using these supports, they define the degrees of epistemic uncertainty u_e and aleatoric uncertainty u_a as, respectively,

$$u_e(x) = \min[\pi(1|x), \pi(0|x)], \quad (2.7)$$

$$u_a(x) = 1 - \max[\pi(1|x), \pi(0|x)]. \quad (2.8)$$

Here, epistemic uncertainty reflects situations where both classes seem equally plausible, while aleatoric uncertainty measures the extent to which neither class is decidedly supported. It is important to note that the calculations for support and, consequently, uncertainty are complex and heavily reliant on the chosen hypothesis space \mathcal{H} .

Density Estimation

When employing learning methods with a large hypothesis space \mathcal{H} , such as neural networks or nearest neighbors, model uncertainty is typically negligible, leaving epistemic uncertainty primarily attributed to approximation uncertainty. Consequently, inference is essentially local, with predictions for a class y made in regions of the feature space where that class has been previously observed, aleatoric uncertainty occurring in regions with overlapping classes, and high epistemic uncertainty in areas of the feature space where no samples have been seen. Generative models that assess densities $p(x)$ are able to quantify epistemic uncertainty by differentiating areas of high and low density, the latter signifying greater epistemic uncertainty. This approach is similar to outlier detection, applying a density threshold to $p(x)$ to detect outliers. This concept ties into the framework of rejection learning, where, for instance, a classifier may decline predictions in low-density regions [70].

Bui and Liu [10] argue that, while methods like Bayesian neural networks, MCD, and ensembles have shown promise in uncertainty estimation, they are computationally intensive due to the need for multiple forward passes during inference or additional parameters. The authors also point out that these methods tend to produce overconfident predictions on OOD data (Figure 2.8), leading to miscalibration. Consequently, they present *density-softmax*, which combines density estimation with a classifier to improve the quality of uncertainty estimation in the presence of distribution shifts. Their density-softmax model includes 4 main components: a feature extractor, a density model on the latent feature space, a classifier, and a softmax layer. The density function is the key innovation that allows the model to

improve predictive uncertainty.

The model begins by pre-training the feature extractor f , which maps inputs into the latent space \mathcal{Z} . After this, the learned feature representations $z = f(x)$ are used to estimate a density function $p(z; \alpha)$, where α are the parameters of a normalizing flows model [74]. Normalizing flows are a class of generative models that transform a simple probability distribution into a more complex one by applying a series of invertible functions. This allows the model to estimate the marginal density in the latent space, assessing how likely a new input is based on the distribution of previously seen data. Finally, after combining the classifier with the likelihood value, the weights of the classifier are updated. During inference, the density score $p(z; \alpha)$ is multiplied by the logits from the classifier $g(z)$, and the result is passed through a softmax function to generate the final class probabilities.

The overall inference formula is:

$$p(y = i | x) = \frac{\exp(p(z; \alpha) \cdot (z^\top \theta_{g_i}))}{\sum_{j=1}^K \exp(p(z; \alpha) \cdot (z^\top \theta_{g_j}))}, \quad \text{for all } i \in \mathcal{Y}. \quad (2.9)$$

Here, $z^\top \theta_{g_i}$ represents the logits for class i and $p(z; \alpha)$ modulates these logits based on the density of the feature z in the latent space. This approach ensures that predictions in low-density regions (indicative of OOD data) reflect higher uncertainty.

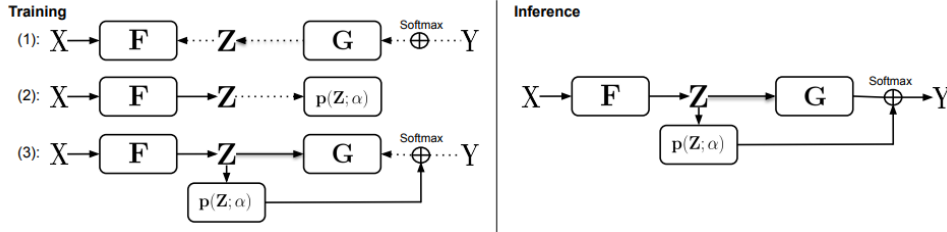


Figure 2.7: Architecture and training procedure of the density-softmax model [10], where f is the encoder, g is the classifier, and $p(Z, \alpha)$ is the density function. The dashed lines represent the backward pass, solid lines represent the forward pass, the circle with a cross represents the softmax layer. Initially, f and g are optimized through expected risk minimization. Then the density function $p(Z, \alpha)$ is estimated on the representation space \mathcal{Z} . Finally, the classifier g 's weights are refined to integrate the density function. At test time, the density function and the classifier are combined to make the prediction.

Density-softmax is evaluated on a toy dataset [56] to illustrate uncertainty visualization (Figure 2.8). In comparison to baselines for uncertainty estimation, density-softmax demonstrates strong distance awareness, providing uniform class probabilities and high uncertainty for OOD data. While SNGP [56] also exhibits distance awareness, it can produce overly confident predictions for OOD data (Figure 2.8) and is more computationally expensive. Density-softmax performs comparably to leading methods in terms of negative-log-likelihood and accuracy on perturbed datasets like CIFAR-10-C, CIFAR-100-C, and ImageNet-C. Moreover, when models trained on CIFAR-10 are tested on CIFAR-100, density-softmax exhibits higher entropy for OOD data and lower entropy for in-distribution data. Additionally, it achieves

better uncertainty estimates and calibration, with lower ECE, on models trained on CIFAR-10 and tested on CIFAR-10.1.

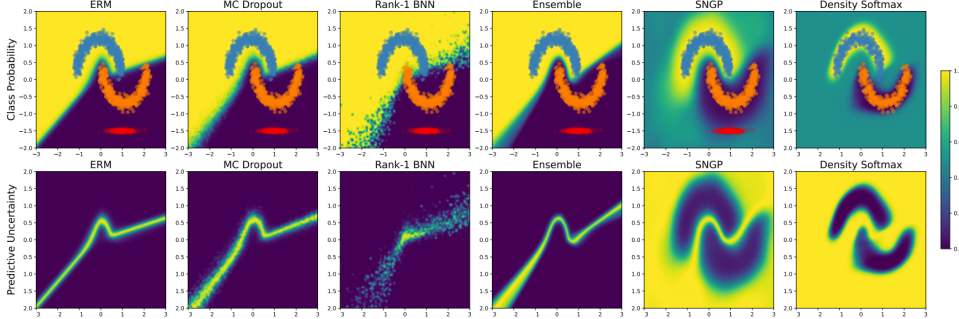


Figure 2.8: Comparison of various uncertainty estimation methods on a toy dataset [10]. Both Density-Softmax and SNGP [56] demonstrate the ability to generate distance-aware probabilities.

Hechtlinger et al. [37] take a different approach to density-based uncertainty estimation. They suggest fitting a generative model $p(x|y)$ for each class and merging density estimation with set-valued predictions via conformal prediction [91]. Three outcomes are possible: a null set prediction indicating high epistemic uncertainty; multiple class predictions signifying high aleatoric uncertainty; or a single class prediction, indicating both epistemic and aleatoric uncertainties are acceptably low. We discuss this approach in more detail below.

Conformal Prediction

Conformal prediction [6, 68, 78, 91] is a method that generates prediction sets rather than point predictions, which are sets of possible outputs that contain the true label with a predefined level of confidence. It was first introduced by Vovk et al. [91] as a transductive online learning method that directly derives prediction sets for each instance based on all previous instances. Papadopoulos et al. [68] then develop an inductive version, where a rule is derived from a set of predictions and is then applied to all new instances. In the following paragraphs, we will explain both the transductive and inductive versions of conformal prediction in detail.

Transductive conformal prediction, or full conformal prediction, is a method where, given $n + 1$ exchangeable¹ data points $(X_1, Y_1), \dots, (X_{n+1}, Y_{n+1})$, the model is presented with $(X_1, Y_1), \dots, (X_n, Y_n)$ and X_{n+1} and is tasked with predicting Y_{n+1} . The objective is to determine a set-valued function $\mathcal{C} : \mathcal{X} \rightarrow 2^{\mathcal{Y}}$ that satisfies

$$\mathbb{P}(Y_{n+1} \in \mathcal{C}(X_{n+1})) \geq 1 - \alpha, \quad (2.10)$$

¹Exchangeability refers to a set of random variables having a joint probability distribution that does not change when the variables are reordered. This means that the variables can be considered statistically equivalent in any order.

where α is a pre-defined error level. For each possible outcome $y \in \mathcal{Y}$, a model \hat{f}^y is trained on the augmented dataset $(X_1, Y_1), \dots, (X_{n+1}, y)$. This model must be invariant to permutations of the data. After calculating \hat{f}^y across all $y \in \mathcal{Y}$, a non-conformity score function $s_i^y = s(X_i, Y_i, \hat{f}^y)$ is computed for $i = 1, \dots, n$ and $s_{n+1}^y = s(X_{n+1}, y, \hat{f}^y)$, where higher scores reflect poorer agreement between x and y . A good example of such a score function is $s(X, Y, \hat{f}^y) = 1 - (\hat{f}^y(X))_Y$, assuming the output of \hat{f}^y are probability scores (e.g. its final layer is a softmax), where higher scores indicate greater uncertainty by reflecting lower predicted probabilities for class Y . The conformal quantile is then determined by

$$\hat{q}^y = \text{Quantile}\left(s_1^y, \dots, s_n^y; \frac{\lceil (n+1)(1-\alpha) \rceil}{n}\right).$$

The confidence set for Y_{n+1} contains all y values that align sufficiently with the data $(X_1, Y_1), \dots, (X_n, Y_n)$:

$$\mathcal{C}(X_{n+1}) = \{y : s_{n+1}^y \leq \hat{q}^y\}.$$

Vovk et al. [91] have theoretically validated that the defined set \mathcal{C} satisfies the coverage guarantee in Equation (2.10). The main drawback of this approach is the computational cost required to estimate \hat{f}^y and to compute $s(X_i, Y_i, \hat{f}^y)$ for every $y \in \mathcal{Y}$.

Inductive conformal prediction [68], also referred to as split conformal prediction, is a less computationally demanding alternative for calculating prediction sets. The method involves training a model \hat{f} on a given training set, reserving a portion of exchangeable data, called the *calibration data*. The goal remains to construct a set $\mathcal{C}(x)$ that guarantees

$$\mathbb{P}(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}})) \geq 1 - \alpha, \quad (2.11)$$

where $(X_{\text{test}}, Y_{\text{test}})$ represents a new data point from the same distribution, and α is the predefined error rate. Using \hat{f} and the calibration data, a non-conformity score $s_i = s(X_i, Y_i, \hat{f})$ is computed for each calibration data point, leading to the calculation of the conformal quantile as

$$\hat{q} = \text{Quantile}\left(s_1, \dots, s_n; \frac{\lceil (n+1)(1-\alpha) \rceil}{n}\right).$$

For a test instance X_{test} the prediction set is created according to

$$\mathcal{C}(X_{\text{test}}) = \{y : s(X_{\text{test}}, y) \leq \hat{q}\}.$$

Both the transductive and inductive methods ensure that conformal prediction provides sets that contain the true label with probability $1 - \alpha$, as opposed to offering a single point prediction.

Hechtlinger et al. [37] suggest a density-based conformal prediction method, which is a variation of

the inductive approach. Their strategy uses an estimate of $p(x|y)$, denoted $\hat{p}(x|y)$, as the scoring function in conformal prediction. Contrary to previously mentioned scoring functions, higher values of $\hat{p}(x|y)$ indicate stronger concordance between x and y , indicating that it is a conformity score, as opposed to the previously mentioned non-conformity score. Despite this difference, the conformal prediction process can be adapted by modifying the prediction sets to

$$\mathcal{C}(X_{\text{test}}) = \{y : s(X_{\text{test}}, y) \geq 1 - \hat{q}\}.$$

Hechtlinger et al. [37] define \hat{q}_y as the empirical $1 - \alpha$ quantile of the values $\hat{p}(X_1|y), \dots, \hat{p}(X_n|y)$, specifically:

$$\hat{q}_y = \sup \left\{ t : \frac{1}{n_y} \sum_{\{Z_i \in D_y^{\text{cal}}\}} \mathbb{I}(\hat{p}(x_i|y) \geq t) \geq 1 - \alpha \right\},$$

where n_y is the number of calibration set examples in class y , D_y^{cal} is the subset of calibration examples that belong to class y , and $\mathbb{I}(\hat{p}(x_i|y) \geq t)$ equals 1 when the estimated probability $\hat{p}(x_i|y)$ is greater than or equal to t , and 0 otherwise. For a novel test instance X_{test} , the prediction set is determined by

$$\mathcal{C}(X_{\text{test}}) = \{y \in Y : \hat{p}(X_{\text{test}}|y) \geq \hat{q}_y\}.$$

This formulation ensures that instances with a low likelihood of being observed under any class are classified as the null set \emptyset , reflecting epistemic uncertainty, while instances likely under multiple classes are assigned a set with various classes, indicative of aleatoric uncertainty. The authors also demonstrate the validity of their approach, proving that for a new observation $(X_{\text{test}}, Y_{\text{test}})$, the probability that Y_{test} falls within the prediction set $\mathcal{C}(X_{\text{test}})$ converges to $1 - \alpha$ as the minimum number of examples per class, $\min_y n_y$, approaches infinity.

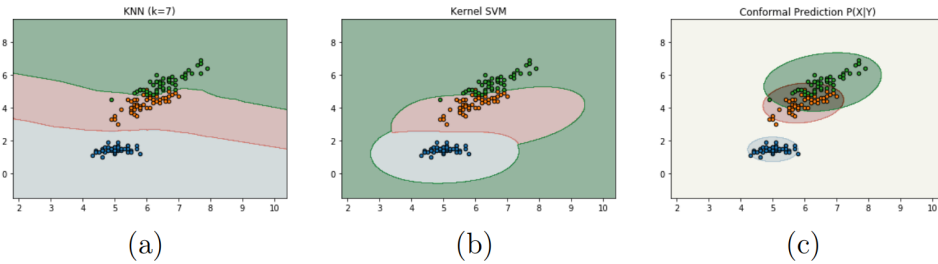


Figure 2.9: Classification boundaries for different methods on the Iris dataset [37]. In methods a) and b), the class boundaries cannot be justified in most of the OOD space. Conformal prediction (with $\alpha = 0.05$) outputs the null set in the white area representing epistemic uncertainty, and outputs multiple classes in the overlapping areas representing aleatoric uncertainty.

In their empirical analysis using the ImageNet dataset, the authors remove the last layer from the pre-trained Xception convolutional neural network [12] and use it as a feature extractor. Then, for each class,

they train a distinct kernel density estimator, which Lei, Robins, and Wasserman [46] have previously identified as optimal under certain conformal settings. The performance achieved with class-specific independent density estimators matched that of GoogLeNet [83], which is noteworthy given that the density estimators do not account for inter-class data relationships. They also discuss that an important aspect of their approach is the adjustability of α , which influences the conservativeness of the prediction sets: higher α values yield more frequent null-set predictions, while lower α values tend to produce prediction sets that include both labels (Figure 2.10).

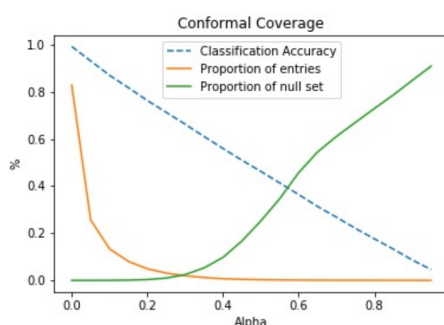


Figure 2.10: Performance plot for the conformal method on the ImageNet dataset. Accuracy decreases linearly as a function of α but affects the number of classes predicted per sample. [37]

Messoudi et al. [62] extend the use of density-based conformal prediction to multiple data types, including images, text, and tabular data. Through binary classification tasks across three datasets, their results demonstrate the method’s effectiveness in detecting noisy and OOD samples. This approach also proves capable of differentiating between aleatoric and epistemic uncertainty with a high degree of accuracy.

By using the aforementioned techniques to measure an ML model’s prediction uncertainty, these can be integrated into a rejection learning system, as decisions to reject or accept predictions are dependent on the model’s confidence level. Section 2.4 focuses on an alternative strategy, where rejection decisions also consider the downstream decision-maker’s performance [59].

2.4 Learning to Defer

L2D was developed based on the critique of confidence-based deferral methods by Madras et al. [59]. In confidence-based deferral methods, a deferral rule that employs the confidence of the classifier determines whether or not to defer an instance to a more intricate classifier, often a human expert. In their proposal of L2D, Madras et al. [59] argue that confidence-based methods fail to consider the performance of the downstream decision-maker when the automated classifier abstains from predicting, and in some cases downstream humans can be just as uncertain as the model. For example, a human

decision maker might be a specialist who performs well only on a specific sub-group of the data. In such scenarios, confidence-based deferral may erroneously forward samples where the downstream human performs worse than the model [47].

Madras et al. [59] adapt the work of Cortes et al. [15] in their formulation of L2D. Cortes et al. [15] introduce the concept of a reject option directly into the learning algorithm by devising a specialized loss function (Equation (2.12)). Their objective is to fine-tune the balance between accuracy and rejection, encouraging the model to maximize its accuracy on the cases it chooses to classify. Simultaneously, the model faces a constant penalty, denoted by γ_{reject} , whenever it opts to reject an instance ($r(x) = 1$). The regular loss \mathcal{L} is downweighted by a factor of $1 - r(x)$, thus not penalizing the classifier for rejected predictions:

$$\mathcal{L}_{\text{reject}}(x, y, h, r) = \sum_i [(1 - r(x_i))\mathcal{L}(y_i, h(x_i)) + r(x_i)\gamma_{\text{reject}}]. \quad (2.12)$$

Madras et al. [59] extend this loss function to consider the performance of the downstream human decision-maker, thus creating L2D. In their formulation of the deferral system’s loss function,

$$\mathcal{L}_{\text{L2D}}(x, y, h, r) = \sum_i \mathbb{E}_{s \sim \text{Ber}(r(x))} [(1 - s_i)\mathcal{L}_{CE}(y_i, h(x_i)) + s_i(\mathcal{L}_{CE}(y_i, \hat{y}_i) + \gamma_{\text{reject}})] \quad (2.13)$$

a decision to defer results in the cross entropy loss between the human prediction \hat{y}_i and the true label, in addition to the constant rejection penalty γ_{reject} . If the decision is not deferred, the loss is computed as the cross entropy loss between the model prediction $h(x_i)$ and the true label. Minimizing this loss ensures that the rejector is penalized for deferring instances where the downstream decision maker made an incorrect prediction.

To differentiate the loss in training, it is necessary to consider continuous outputs for the functions $h(x)$, $r(x) \in [0, 1]$. Due to this, the authors suggest using a mixture of experts, where said experts are the human decision maker and the classifier. A deferral indicator variable s_i is introduced to denote who predicts on a particular instance. If $s_i = 1$, the human expert predicts, otherwise the ML classifier predicts. We can now parameterize r and h with neural networks and optimize the loss directly with gradient descent, estimating \mathcal{L}_{L2D} and its gradient by sampling s from a Bernoulli distribution (Equation (2.13)) for each instance. Training both h and r on the system loss allows each network to adapt to the behaviour of the other, optimizing the performance of the system as a whole.

Madras et al. [59] argue that incorrectly deferring examples can have significant fairness consequences, and thus focus on also developing a fairness-focused approach for L2D. For example, if the downstream human’s decisions are biased against a certain group, then it is preferable, from a fairness perspective, for the deferral system to defer less frequently on the instances pertaining to said group. For this purpose they build a regularized fair loss function that combines the error rate with a fairness

metric.

To evaluate their approach, Madras et al. [59] test three distinct scenarios, each representing a different type of decision-maker. They conduct experiments on two datasets (COMPAS [51] and Heritage Health [1]), simulating decision-maker data by training three distinct classifiers under the following conditions: the first classifier is trained with access to additional information correlated with the label, representing a high-accuracy decision-maker; the second is trained with inverse fairness objectives, introducing bias against a specific group; the third is a variation of the first, where some decisions are randomly flipped to simulate an inconsistent decision-maker. These settings are tested in parallel and compared against a rejection learning baseline. The results demonstrate that L2D consistently outperforms the rejection learning baseline, regardless of the type of decision-maker considered.

Since the introduction of L2D, several other authors have made significant contributions in improving and expanding this framework. Mozannar and Sontag [64] show that the mixture of experts loss proposed by Madras et al. [59] is not consistent, meaning that it does not necessarily converge to the Bayes optimal solution, even as the amount of training data approaches infinity. Therefore, they propose learning a single classifier with an additional class for deferral, trained on a consistent surrogate loss. This approach yields better results in practice when compared with the loss of Madras et al. [59], according to their experiments. Furthermore, Mozannar and Sontag emphasize the necessity of joint training for the classifier and rejector, arguing that it enables the classifier to adapt to the expertise of the human decision-makers, ensuring that the classifier is allowed to maximize its performance on instances that are not rejected (Figure 2.11).

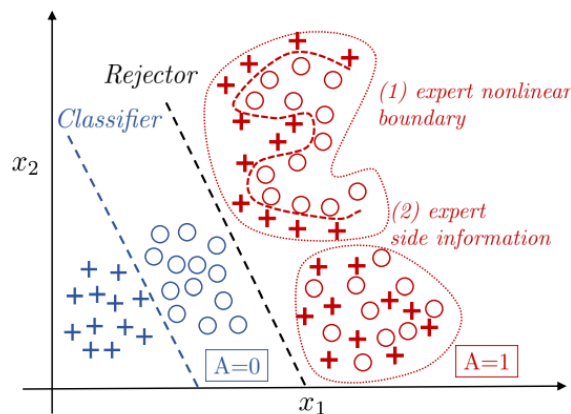


Figure 2.11: Mozannar and Sontag [64] argue that jointly training the main classifier and the rejector improves the overall system performance, as it allows the main classifier to focus on the instances that will be assigned to it.

Verma and Nalisnick [90] investigate the calibration of L2D systems, focusing on Mozannar and Sontag's [64] surrogate loss. They demonstrate both theoretically and practically that this loss is not calibrated with respect to expert correctness due to a degenerate parameterization. To address this lim-

itation, Verma and Nalisnick propose a one vs. all (OvA) approach, where the classifier and rejector are trained independently. This setup allows the classifier to focus on predicting correctly for all instances, whether they are likely to be deferred or not. Experimentally, they show that this OvA approach results in a better calibrated model, achieving improved calibration without sacrificing predictive performance.

In the OvA framework, the classification task is decomposed into K binary classification tasks, with each scoring function g_k trained to predict whether an instance belongs to class k or any other class. The rejector, represented by g_{\perp} , models whether or not the expert is correct. The final classification is determined by selecting the class with the maximum score: $\arg \max_{k \in \{1, \dots, K\}} (g_k(x))$, that is, the decision-maker which is most likely to be correct on a given instance. Verma and Nalisnick adapt Mozannar and Sontag’s approach and propose the following consistent surrogate loss for this framework:

$$\Psi_{\text{OvA}} = \Phi[g_{y_i}(x_i)] + \sum_{y' \in \mathcal{Y}, y' \neq y_i} \Phi[-g_{y'}(x_i)] + \Phi[-g_{\perp}(x_i)] + \mathbb{I}[\hat{y}_i = y_i](\Phi[g_{\perp}(x_i)] - \Phi[-g_{\perp}(x_i)]), \quad (2.14)$$

where, Φ is a proper binary surrogate loss, such as the logistic loss $\phi[g(x)] = \log(1 + \exp(-g(x)))$, which ensures that the posterior probability $p(y_i = 1 \mid x_i)$ can be recovered [73]. The inverse link function translates the raw scores from the classifier into calibrated probability estimates, which is vital for making accurate predictions and, consequently, properly deferring instances. The rejector’s function $g_{\perp}(x)$ maximizes the value when the expert is correct, ensuring that deferral occurs only in instances where it is beneficial. Overall, this formulation allows the system to make more accurate predictions while maintaining correct deferral behavior, achieving better calibration and outperforming Mozannar and Sontag’s approach.

Keswani et al. [49] expand upon the single expert L2D framework to incorporate multiple human decision-makers. In their approach, an entire team of decision-makers is regarded as a collective unit from which the system may select when deferring decisions. The authors return to a mixture of experts setting considered by Madras et al. [59], where each human decision-maker and the main classifier are considered as the experts in this framework. The system is structured as a multi-output neural network, with each output corresponding to an individual decision-maker, including the main classifier. Each decision-maker’s prediction is weighted in the final system decision, with the weight reflecting their influence. To compute the loss, they use a sigmoid transformation of the dot product between the output weights and the vector of individual predictions from each decision-maker. This allows for the integration of contributions of all decision-makers in the final prediction. At test time, the weights can be used to either determine the optimal overall prediction considering all decision-makers, or to identify the best individual decision-maker by ranking the output weights. The authors also revisit the topic of fairness within the L2D system, implementing minimax Pareto fairness to alleviate disparities across all groups [61].

Hemmer et al. [39] offer a similar approach to the one of Keswani et al. [49] for extending L2D to the multiple expert setting. Their key innovation is the use of a softmax function applied to the allocation system’s output ($\hat{y}_{L2D} = \text{softmax}(w) \cdot \hat{y}_{DM}$), which calculates the probability that a team member will provide the correct output, replacing the sigmoid function used in [49] ($\hat{y}_{L2D} = \sigma(w \cdot \hat{y}_{DM})$). Empirical comparisons between these two approaches demonstrate that Hemmer et al.’s method [39] surpasses the prior baseline established by Keswani et al. [49].

Given the lack of consistency and calibration in the aforementioned works [39, 49], Verma et al. [89] tackle this problem by adapting the work of Verma and Nalisnick [90] and the softmax surrogate loss of Mozannar and Sontag [64] to propose two consistent and calibrated loss functions for the multiple expert setting. The first one is based on the softmax parameterization in [64], being analogous to the surrogate loss presented in that work for the single expert setting, while the second is analogous to the OvA loss presented in [90] for the single expert setting. When studying the calibration of the system with respect to expert correctness (i.e. the system’s ability to estimate $\mathbb{P}(m_j = y|x)$, the probability that the j th expert will correctly predict the label for x), they demonstrate that the softmax-based loss causes miss-calibration of these estimates while the same does not happen for the OvA formulation.

2.5 Limitation of Learning to Defer

At Feedzai, implementing human-AI fraud detection systems comes with several challenges. Cost-sensitive scenarios are a frequent issue in fraud detection, where the cost of false positives (wrongly flagging legitimate transactions) is significantly higher than that of false negatives (missing fraudulent transactions). Managing capacity constraints is also critical, not just for Feedzai but for any application that employs human decision-makers, to ensure that these are not overwhelmed and are utilized optimally. In performative prediction scenarios, such as those encountered at Feedzai, where the system’s predictions can influence future data distributions and decision-making patterns, it is important to have a system that is adaptable to dynamic environments. Given the inherent ability of humans to learn and adapt, this presents a significant opportunity for human-AI collaboration systems to evolve continuously in response to changing environments.

Up until now, L2D has been showcased as an effective framework for optimizing decision-making involving humans and AI. However, the practical adoption of state-of-the-art methods faces significant challenges, as outlined by Leitão et al. [55]. A primary limitation is the need for human predictions for every training instance, which is not feasible for large datasets due to the high cost of obtaining human labels. This limitation is aggravated when considering the multiple expert setting, which requires every human’s prediction for every instance in training. Moreover, in existing deferral systems, human intervention typically occurs only for a subset of cases—usually those where the main classifier’s confi-

dence is low, which leads to a skew in the distribution of human-labeled data compared to the complete dataset. This distributional discrepancy hinders the application of imputation techniques to estimate human predictions for unlabeled data, thus complicating the training of a L2D system.

Jointly training the main classifier and the deferral model is highlighted as a benefit of L2D systems because it enables the main classifier to specialize on instances that are not deferred [59, 64]. However, this potentially benefit has the drawback of neglecting the classifier's effectiveness on cases that are likely to be deferred. In many fields, it's crucial for the classifier to perform well across all instances, not just non-deferred ones. This capability is especially important if the classifier is required to provide guidance to human decision-makers (e.g. a financial fraud detection classifier that raises alerts that are reviewed by financial analysts) or in situations where human experts are temporarily unavailable, thus necessitating reliable automated predictions. Verma et al. [89] adopt a different strategy by training the classifier and the rejector independently, which promotes accurate classification on all instances, whether they are likely to be deferred or not. While Mozannar et al. [63] highlight theoretical benefits of joint training, the OvA approach by Verma et al. [89] provides better performance through improved calibration over the adaptation of the softmax loss presented in [64], making it arguably more suitable for practical applications.

Capacity constraints are another consideration that has often been overlooked in L2D research [5]. When faced with limited human resources, it's essential that decision-making is distributed in a way that maximizes overall accuracy while respecting the availability of human experts. This means that cases should be allocated based on both the expert's performance and availability. Addressing capacity constraints is a crucial aspect that requires more attention in the development of L2D frameworks, as it directly impacts the system's scalability and ensures that human expertise is utilized optimally, without leading to burnout or inefficiency.

A significant unsolved challenge in L2D systems is their lack of adaptability in dynamic environments, where data distributions can evolve over time, leading to concept drift [28]. Additionally, in performative prediction scenarios [69], the actions of the system might alter the environment itself, such as fraudsters modifying their tactics in response to fraud detection measures. Human decision-makers are also prone to changes in their decision-making processes, influenced by new knowledge, experiences, and external factors. To ensure robustness, a L2D system must be capable of adapting to these non-stationary elements, an area that current research has not yet adequately addressed.

Recent advancements in L2D research have aimed to address some of these practical constraints that limit the deployment of L2D systems. Charusaie et al. [11] address the issue of data efficient learning in the single expert setting by employing active learning strategies to reduce the number of human predictions necessary in training. Their method involves maintaining a version space of hypotheses that are consistent with the accumulated data. At each step, the algorithm identifies points of disagreement

within the hypothesis space—situations where two classifiers predict differently. The human expert is then asked to provide labels for these disputed instances, allowing the version space to be refined (Figure 2.12). This iterative process continues until the system achieves minimal loss.

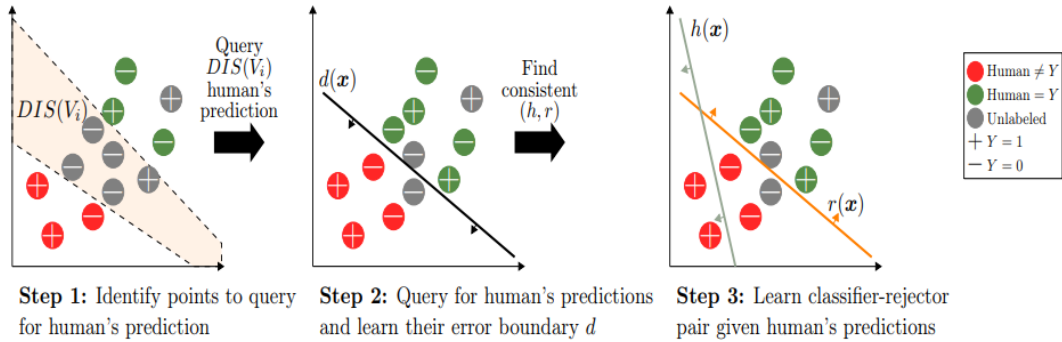


Figure 2.12: Illustration of Charusaie et al.'s approach [11]. At each round, they compute the disagreement set for the predictors of the human label disagreement, they then query the human for their prediction on these points. After they learn the expert error boundary, they then learn a consistent classifier-rejector pair.

Hemmer et al. [40] introduce a different approach to address the challenge of learning with limited expert predictions, incorporating semi-supervised learning techniques to impute missing human predictions, thus creating a dataset of both real and imputed labels to train the L2D system. Their imputation method involves first training an embedding model using ground truth labels to produce feature representations. These representations then inform the training of an expertise predictor model designed to approximate an expert's decision-making abilities. The expertise predictor is then used to generate synthetic expert predictions for instances that lack real expert annotations.

Both approaches discussed previously [11, 40] focus on the single expert scenario. To overcome the limitation of requiring human predictions for every instance and every expert in the multi-expert setting, Tailor et al. [84] introduce learning to defer to a population (L2D-Pop). This variant of L2D is designed to accurately defer decisions to human experts not observed during training, relying only on a small context set (a few exemplar data points) that characterizes the available expert. They employ meta-learning for L2D-Pop, utilizing the context set to adapt the model via a single forward pass. The adaptation leverages a deep sets architecture to encode the context set into a representation that reflects the expert's decision-making profile. This representation is combined with the input features to enhance the deferral decision. Additionally, they incorporate an attention mechanism to improve the adaptation process, focusing on the most pertinent elements of the context set for each specific decision. This approach is pertinent to deal with changes in an expert team, however it still requires an initial set of expert demonstrations to be able to train the system.

Alves et al. [5] address the overlooked aspect of cost-sensitive scenarios in existing L2D research, while at the same time echoing critiques from Leitão et al. [55] regarding the lack of consideration for

expert capacity constraints in L2D systems. They introduce a novel deferral strategy for managing assignments in cost-sensitive human-AI decision-making contexts that adhere to human capacity limits. Their proposed system comprises three components: a ML classifier that estimates the probability of the target class based on instance features; a human expertise model that models the probabilities of correctness of each of the experts in the team; and an assigner that optimally allocates assignments considering both capacity constraints and the probabilities of correctness of all decision-makers.

Since Alves et al. [5] is the only recent work to address both cost-sensitive scenarios and capacity constraints in L2D systems, we adapt their strategy for managing cost-sensitive scenarios and build upon their framework for handling capacity constraints. Their work will be discussed in further detail in Chapter 5. Additionally, we aim to extend their work by introducing a more robust L2D solution that is adaptable to dynamic environments—an area that has been largely unexplored in current research., while preserving the state-of-the-art advancements in L2D systems.

Chapter 3

Enhancing Uncertainty Estimation in L2D with Density Softmax

The primary objective of this thesis is to develop an assignment system that optimizes decision-making in a human-AI collaborative environment. This system aims to intelligently determine who should make a decision for each instance: the AI model or a human expert. In many existing systems, model uncertainty is often the sole criterion used to decide between the AI and human decision-makers [13, 15, 29]: if the model exhibits high uncertainty, the task is deferred to a human; otherwise, the AI proceeds with the decision. While this approach may seem practical, it overlooks critical factors such as the varying performance and potential biases of human decision-makers, as discussed in earlier chapters. To incorporate these human elements, L2D [59] has been proposed by several researchers as a method to train an assignment mechanism, aiming to improve overall system performance. However, current L2D research fails to consider the inability to adapt to dynamic environments. This limitation hinders the scalability and effectiveness of L2D systems in real-world applications. In this chapter, we present the first of two approaches to overcome this obstacle.

In this chapter, we introduce an L2D system enhanced with distance-aware models, designed to improve robustness against distribution shifts. Following the density-softmax approach proposed by Bui and Liu [10], we combine the output of our ML models with a density function trained on the same data to achieve distance awareness. By incorporating density estimation, we ensure that predictions made in low-density regions—where the model has seen fewer instances—are accompanied with higher uncertainty [10]. When using the OvA L2D approach proposed by Verma et al. [89] (which will be discussed in Section 3.3), this integration prevents the predictions of the correctness of each decision-maker (an ML model or human experts) from being overconfident on OOD data. Consequently, this approach not only reduces the likelihood of assigning decisions to suboptimal decision-makers in unfamiliar regions of

the feature space, but also improves the overall system calibration. By maintaining accurate predictions within the known data distribution and preventing overconfident predictions on OOD data, better calibration and reliability are ensured. This is particularly important when working with limited expert prediction data, as this requires training the models that assess expert correctness on a subset of the available training data.

In the next section, we provide an overview of the architecture and training process of the proposed system. We will demonstrate how it effectively mitigates overconfident predictions, ensuring robustness against shifts in data distribution, thereby improving the reliability of decision-making in human-AI collaborative environments.

3.1 Data

For simplicity, we start by considering a binary classification problem, although our approach can easily be generalized to multi-class problems. In this setup, each instance i is characterized by a feature vector x_i and a label $y_i \in \mathcal{Y} = \{0, 1\}$. The feature vector may also include additional information accessible to the experts, such as the ML model’s score. In cost-sensitive scenarios, we assume that each instance i has an associated misclassification cost c_i , which represents the cost incurred if the instance is classified incorrectly. Conversely, a correct classification does not incur any cost.

To model a realistic scenario, we assume that our dataset does not include predictions from every expert for every instance. Instead, we consider that expert j ’s prediction for instance i in our dataset, $m_{i,j} \in \mathcal{Y}$, is treated as its own unique instance within the training set. This means that if two distinct experts, j and k , both provide predictions for the same instance i , these will be treated as two separate instances in our training set: $\{x_i, y_i, m_{i,j}, c_i\}$ and $\{x_i, y_i, m_{i,k}, c_i\}$.

Consequently, our training set is defined as $S = \{x_i, y_i, m_{i,j}, c_i\}_{i=1, j=1}^{N, J}$. The objective is to estimate the probability of correctness associated with deferring instance i to expert j or to a ML model, with the goal of maximizing the overall probability of correctness under capacity constraints, consequently achieving a low misclassification cost.

It is important to note that the modeling assumptions and objectives outlined in this section are also applicable to the discussions in Chapter 4 and Chapter 5. The upcoming sections expand on the concepts introduced here, particularly in relation to how we handle uncertainty and cost-sensitive learning under expert capacity constraints.

3.2 Density-softmax

The density-softmax approach, as introduced by Bui and Liu [10], is composed of three main components: a feature extractor, a lightweight normalizing-flows density model on the feature space, and a classifier with a softmax output layer. During training, a feature extractor is trained using an empirical risk minimization objective. Once trained, an embedding of the input data is extracted, serving as input for the next step: density estimation.

To obtain estimates of the marginal density of the learned feature space, we use the real-valued non-volume preserving (RealNVP) model [22], belonging to a class of generative models referred to as normalizing flows models. These transform a simple, tractable distribution (usually a Gaussian) into a complex distribution through a sequence of invertible and differentiable transformations. The key advantage of normalizing flows is that they allow computing the exact data likelihood under the model, making them highly effective for tasks like density estimation.

The objective of the RealNVP model is to minimize the negative log-likelihood of the input data, which has been transformed into a simpler distribution. This is achieved through a series of transformations, where the likelihood of the transformed data is computed efficiently by taking into account both the likelihood of the transformed samples and the change in volume induced by the transformation. The log-likelihood can be obtained through the standard change of variable formula,

$$\log p_X(x) = \log p_Z(f(x)) + \log |\det(Df(x))|, \quad (3.1)$$

where $\log p_Z(f(x))$ is the log-likelihood of the data in the latent space (which is typically modeled as a Gaussian), and $\log |\det(Df(x))|$ is the log-determinant of $Df(x)$, the Jacobian of the transformation f , computed at x , which accounts for the change in volume during the transformation.

To make the computation of this Jacobian determinant efficient, RealNVP uses a specific design involving affine coupling layers. The core idea is to partition the input data into two subsets and apply a transformation only to one subset, conditioned on the other. The coupling layers alternate the subsets being transformed, ensuring that the entire transformation remains invertible. This clever partitioning leads to a Jacobian matrix with a triangular structure, which simplifies the computation of the determinant.

For a D -dimensional input x , the output y of an affine coupling layer is given by

$$y_{1:d} = x_{1:d}, \quad (3.2)$$

$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d}), \quad (3.3)$$

where s and t are scaling and translation functions, which can be arbitrarily complex, typically modeled

using neural networks. Because one subset of the data remains unchanged while the other undergoes an affine transformation, the Jacobian matrix for this transformation is triangular:

$$Df(x) = \begin{bmatrix} I_d & 0 \\ B & \text{diag}(\exp[s(x_{1:d})]) \end{bmatrix}, \quad (3.4)$$

where B is a block that does not affect the determinant of $Df(x)$. This structure allows computing the determinant of the Jacobian efficiently, as it reduces to the product of the diagonal elements:

$$\det(Df(x)) = \exp\left(\sum_j s(x_{1:d})_j\right). \quad (3.5)$$

Note that the forward transformation of the coupling layers leaves some components unchanged. This is overcome by composing coupling layers in an alternating pattern, such that the components that are left unchanged in one coupling layer are changed in the next. When combining coupling layers, the Jacobian determinant of the resulting function remains tractable due to the fact that the determinant of the composition of two transformations is simply the product of their respective Jacobians, as a simple consequence of the chain rule:

$$D(f_b \circ f_a)(x) = Df_b(f_a(x))D(f_a(x)), \quad (3.6)$$

thus

$$\det(D(f_b \circ f_a)(x)) = \det(Df_b(f_a(x)))\det(D(f_a(x))). \quad (3.7)$$

By structuring the transformation this way, RealNVP ensures that computing the log-determinant of the Jacobian is computationally feasible, even for high-dimensional data. This allows the model to compute exact log-likelihoods efficiently, making it a powerful tool for density estimation and other applications requiring a detailed understanding of data distributions.

The RealNVP model outputs a log-likelihood for each data point x , $\log(p(f(x; \alpha)))$, which indicate how likely each point is to be sampled under the learned distribution, where α represents the parameters of the RealNVP model that control the transformation $z = f(x; \alpha)$ from the input space into a space where the features follow a simple, tractable distribution, usually, a standard Gaussian. To transform these log-likelihoods into likelihoods, we apply the exponential function $e^{\log(p(f(x; \alpha)))}$. However, directly exponentiating the log-likelihoods can lead to numerical issues, particularly when the log-likelihood values are large, causing a numeric overflow.

To address this, we follow the method described by Bui and Liu in Appendix B.1 [10]: the log-likelihoods are rescaled by dividing them by the maximum log-likelihood value obtained during training. This rescaling ensures that the log-likelihoods are within a manageable range, with a maximum of 1. After rescaling, we apply the exponential function to convert the log-likelihoods into likelihoods.

These likelihoods are then normalized to fall within the range $[0, 1]$, making them suitable as probability estimates.

In the final step, we combine the density scores (obtained from the likelihoods) with the logits from the classifier. During inference, the standard softmax function is modified to incorporate the density score $p(f(x; \alpha))$ into the logits. For a multi-class classification scenario, the predictive probability is given by

$$p(y = i|x) = \frac{\exp(p(f(x; \alpha)) \cdot (x^T \theta_{g_i}))}{\sum_{j=1}^K \exp(p(f(x; \alpha)) \cdot (x^T \theta_{g_j}))}, \quad (3.8)$$

where θ_g represent the weights of the classifier, $x^T \theta_{g_i}$ represents the logits for class i , and $p(f(x; \alpha))$ modulates these logits based on the density of the feature z in the latent space. In a binary classification setting, this is simplified by using the sigmoid function

$$p(y = 1|x) = \frac{1}{1 + \exp(-p(f(x; \alpha)) \cdot (x^T \theta_g))}. \quad (3.9)$$

This formulation ensures that for instances with low density scores (indicating they are likely OOD), the probability $p(y|x)$ will approach 0.5, reflecting high uncertainty. Conversely, for high-density scores (indicating in-distribution instances), the probability will align closely with the classifier's output, ensuring that the model's confidence is appropriate for the data at hand.

3.3 Learning to Defer

We build our L2D framework using the classifier-rejector approach used by Verma et al. [89], frequently employed in recent L2D work [59, 63, 64, 89, 90]. In this approach, the authors simultaneously learn a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ and a rejector $r : \mathcal{X} \cup \mathcal{E} \rightarrow \{0, 1, \dots, J\}$, where \mathcal{E} represents additional information available to the experts. The classifier h is responsible for making decisions, while the rejector r determines whether to rely on the classifier or to defer to one of the available human experts. Specifically, if $r(x_i) = 0$, the decision of classifier h on instance i is accepted, otherwise, if $r(x_i) = j$, the instance is assigned to expert j .

To guide the learning process, the 0-1 loss function \mathcal{L}_{0-1} proposed by Verma et al. [89] is

$$\mathcal{L}_{0-1}(h, r) = \mathbb{E}_{x, y, \{m_j\}_{j=1}^J} \left[\mathbb{I}[r(x) = 0] \mathbb{I}[h(x) \neq y] + \sum_{j=1}^J \mathbb{I}[r(x) = j] \mathbb{I}[m_j(x) \neq y] \right]. \quad (3.10)$$

In this formulation, the loss reflects the correctness of the decision-making process. When the classifier h makes the prediction (i.e., $r(x) = 0$), a loss of 1 is incurred if the prediction is incorrect. Similarly, when expert j is chosen to make the decision (i.e., $r(x) = j$), a loss of 1 is incurred if the expert's prediction $m_j(x)$ is incorrect. Verma et al. [89] demonstrate that the Bayes-optimal classifier h^* and rejector r^* ,

which minimize this loss, satisfy the following conditions:

$$h^*(x) = \arg \max_{y \in \mathcal{Y}} \mathbb{P}(y|x) \quad (3.11)$$

$$r^*(x) = \begin{cases} 0 & \text{if } \mathbb{P}(y = h^*(x)|x) > \mathbb{P}(y = m_j(x)|x) \forall j \in \{1, \dots, J\} \\ \arg \max_{j \in \{1, \dots, J\}} \mathbb{P}(y = m_j(x)|x) & \text{otherwise.} \end{cases} \quad (3.12)$$

However, while the 0-1 loss function provides a clear framework for decision-making, it is non-convex, making it challenging to optimize. To address this issue, Verma et al. [89] introduce a consistent convex surrogate for the 0-1 loss in their OvA approach. Minimizing that surrogate, leads to a classifier \hat{h} and a rejector \hat{r} that approximate the Bayes-optimal classifier-rejector pair.

In the OvA approach, the classifier is composed of K functions $g_k : \mathcal{X} \rightarrow \mathbb{R}$ for each class $k \in \{1, \dots, K\}$, each of which represents a binary classification task, estimating the likelihood that a given instance belongs to class k . Similarly, the rejector consists of J functions $g_{\perp,j} : \mathcal{X} \rightarrow \mathbb{R}$ corresponding to each expert j , which are related to the probability that expert j will make the correct decision. The OvA surrogate for the 0-1 loss function is obtained by combining the functions g_1, \dots, g_K and $g_{\perp,1}, \dots, g_{\perp,J}$ in the following manner:

$$\Psi_{\text{OvA}} = \Phi[g_y(x)] + \sum_{y' \in \mathcal{Y}, y' \neq y} \Phi[-g_{y'}(x)] + \sum_{j=1}^J \Phi[-g_{\perp,j}(x)] + \sum_{j=1}^J \mathbb{I}[m_j = y] (\Phi[g_{\perp,j}(x)] - \Phi[-g_{\perp,j}(x)]). \quad (3.13)$$

where $\Phi : \{\pm 1\} \times \mathbb{R} \rightarrow \mathbb{R}_+$ is a strictly proper binary surrogate loss, such as the logistic loss.

Verma et al. [89] show that the minimizer of the pointwise inner risk of this surrogate loss can be analyzed in terms of the pointwise minimizer of the risk for each of the $K + J$ underlying OvA binary classification tasks. They conclude that the minimizer of the pointwise inner Ψ_{OvA} -risk, g^* , consists of the minimizers of the inner Φ -risk for each individual binary classification problem, g_i^* .

In practical scenarios where not all experts' predictions are available for every instance, each binary classifier $g_{\perp,j}$ can be trained independently using the subset of the training data that contains expert j 's predictions, yielding the best possible estimates of the pointwise inner Φ -risk minimizers g_i^* based on the available data. This way, the OvA approach allows working in scenarios where only one expert prediction is available per instance, effectively addressing the data availability limitations that many L2D systems do not adequately consider.

Furthermore, this formulation allows us to enhance the system by combining each binary classifier with a density model, making the system distance-aware and improving uncertainty estimates on unseen data. By training a feature extractor and a RealNVP model on the same data as each classifier, we obtain density scores that capture the likelihood that a given instance belongs to the distribution characterized by the subset of the data on which each classifier is trained. These density scores are then combined

with the logits of each binary classifier before applying a sigmoid function. This results in a robust L2D system where the output probabilities are calibrated for in-distribution data and closer to random for OOD data. As a result, the system defers decisions more effectively by accounting for uncertainty in regions where the model has less confidence, leading to improved decision-making in collaborative environments.

3.3.1 Classifier

In the binary classification case, where $K = 2$, only one binary classifier is needed. This classifier \hat{h} , with scoring function g , is trained to predict the true label y_i by minimizing a proper binary composite loss function with a well-defined inverse link function ψ^{-1} . A common choice is the logistic loss whose inverse link function is given by $\psi^{-1}(g) = 1/(1 + e^{-g})$ [73]. The empirical estimator of the expected value of the logistic loss is given by

$$\mathcal{L}_{\log}(g) = \frac{1}{N} \sum_{i=1}^N [-y_i \log(\psi^{-1}(g(x_i))) - (1 - y_i) \log(1 - \psi^{-1}(g(x_i)))], \quad (3.14)$$

where $g(x_i)$ is the score for sample x_i .

By using a proper binary composite loss, we ensure that $\mathbb{E}[\mathcal{L}]$ is minimized when $\psi^{-1}(g(x)) = \mathbb{P}(y = 1|x)$ [73]. This means that minimization of this empirical loss converges to \hat{g} as $N \rightarrow \infty$, resulting in a scoring function that, when combined with a well-defined inverse link function, produces calibrated estimates of $P(y = 1|x)$. Therefore, we define \hat{h} as

$$\hat{h}(x) = \begin{cases} 1 & \text{if } \psi^{-1}(\hat{g}(x)) > 0.5 \\ 0 & \text{otherwise,} \end{cases} \quad (3.15)$$

ensuring that \hat{h} agrees with the Bayes-optimal classifier h^* in the large sample limit.

When extending this approach to cases with more than two classes ($K > 2$), the OvA approach involves training multiple binary classifiers. Specifically, for each class k in $\{1, \dots, K\}$, we train a separate binary classifier that distinguishes between class k and all other classes. Each classifier is trained in a similar fashion to the binary case described above, using a proper binary composite loss and an appropriate inverse link function. After training these K binary classifiers, we can then combine their outputs to make a final multi-class prediction. The predicted class for an instance x is the one corresponding to the classifier that gives the highest probability, ensuring that the resulting multi-class classifier generalizes the binary OvA approach to the multi-class setting. As discussed in Section 3.2, these binary classifiers can be combined with a density model to improve uncertainty estimation.

3.3.2 Rejector

The rejector in our L2D framework is composed of J functions $g_{\perp,j} : \mathcal{X} \rightarrow \mathbb{R}$, for $j \in \{1, \dots, J\}$, which estimate the probability that expert j will make the correct decision. These functions are trained to predict the correctness of the experts' decisions and are trained in the same manner as the classifier, ensuring consistency. Each $g_{\perp,j}$ is trained independently on the subset of the training data that contains expert j 's predictions. If additional information $e \in \mathcal{E}$ (e.g., ML model score) is available to the experts, it is also used by the scoring functions, thus $g_{\perp,j} : \mathcal{X} \cup \mathcal{E} \rightarrow \mathbb{R}$.

By training these functions, we obtain estimates $\mathbb{P}(y = m_j|x)$ for each $j \in \{1, \dots, J\}$, given by $\psi^{-1}(\hat{g}_{\perp,j}(x))$. The rejector \hat{r} is then defined as

$$\hat{r}(x) = \begin{cases} 0 & \text{if } \max\{\psi^{-1}(\hat{g}(x)), 1 - \psi^{-1}(\hat{g}(x))\} > \psi^{-1}(\hat{g}_{\perp,j}(x)), \forall j \in \{1, \dots, J\}, \\ \arg \max_{j \in \{1, \dots, J\}} \psi^{-1}(\hat{g}_{\perp,j}(x)) & \text{otherwise.} \end{cases} \quad (3.16)$$

This definition ensures that \hat{r} aligns with the Bayes-optimal rejector r^* , thus, the classifier-rejector pair, \hat{h} and \hat{r} , converge to the minimizers of \mathcal{L}_{0-1} , h^* and r^* , achieving optimal classification performance in theory.

As discussed in Section 3.2, each binary classifier within the rejector can be combined with a density model, allowing the rejector to provide well-calibrated and reliable estimates of expert correctness.

3.4 Dealing with Epistemic Uncertainty in Expert Modeling

Each binary classifier used in the rejector is designed to estimate the correctness of an expert's decision. By leveraging the density-softmax approach, these classifiers are combined with a density function, making them distance-aware. Consequently their predicted probabilities account for the likelihood of the instance belonging to the training distribution. In this way, the classifiers provide more reliable estimates of correctness, particularly when faced with OOD data, where uncertainty is higher.

There are two scenarios where the model may output an estimated probability of correctness close to random (0.5). The first occurs if no similar instances have been seen during training, leading to a very low density score. In this case, the density score downscales the logits to nearly zero, causing the sigmoid function's output to converge to 0.5 (as shown in Equation 3.9). This behavior reflects epistemic uncertainty, which could be reduced with more training data. The second scenario arises if an instance has a high density score (indicating it is in-distribution), yet the classifier still produces a sigmoid output of 0.5. In this case, the uncertainty is aleatoric, representing inherent noise or randomness in the data that cannot be reduced even with additional data.

For any given expert, if the model encounters an instance that was not present during training, it will output a probability of correctness close to 0.5, simply because no similar training data exists for that expert. However, unlike machine learning models, human experts possess the ability to draw on information beyond the scope of the data used for training. They can reason causally, adapt to new environments, and continue learning in real-time. Given these capabilities, a more realistic estimate of expert correctness in situations of high epistemic uncertainty should reflect the expert's proficiency in some way. Specifically, a heuristic approach is to make the model incorporate the expert's average probability of correctness, calculated from the expert's predictions on the training data.

To address this, we adjust the model's logits to adjust the output probability of expert correctness when faced with high epistemic uncertainty. This adjustment is done progressively by leveraging the density score, $p(f(x; \alpha))$, which lies in the interval $[0, 1]$. The density score blends the model's predicted logits with logits that would yield the expert's historical average correctness when passed through the sigmoid function. Specifically, the adjusted logits are given by

$$p(f(x; \alpha))\hat{g}_{\perp,j} + (1 - p(f(x; \alpha)))\sigma^{-1}(\hat{p}_{j,\text{avg}}), \quad (3.17)$$

where, $\hat{g}_{\perp,j}$ represents the logit score of the binary classifier for expert j , while $\hat{p}_{j,\text{avg}}$ denotes the expert's average correctness, computed as

$$\hat{p}_{j,\text{avg}} = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \mathbb{I}[m_j(x_i) = y_i], \quad (3.18)$$

where, N_{train} is the total number of training instances, $m_j(x_i)$ is the prediction made by expert j for instance x_i , and y_i is the label. The adjusted logits are then passed through the sigmoid function (see Equation 3.9) to obtain the expert's final probability of correctness.

By using this approach, the model smoothly balances its own prediction of expert correctness with the expert's historical performance. In cases where the density score is low, indicating high epistemic uncertainty, the model's output will converge toward the expert's average correctness. Conversely, for in-distribution instances with high density scores, the model relies more heavily on its own prediction, ensuring a more calibrated estimate of expert correctness.

Chapter 4

Conformal Prediction for Human-AI Collaboration

In this chapter, we propose a different approach in order to address the same challenge discussed in the previous chapter: improving uncertainty estimation in L2D systems and enabling them to handle distribution shifts effectively. While the previous chapter focuses on combining classifiers with density estimation models to reflect uncertainty in OOD data, this chapter proposes an approach that separates uncertainty estimation from the classifiers themselves.

We hypothesize that L2D, while effective in handling familiar cases, should avoid managing instances that are unseen during training, as the probability estimates in such cases may be poorly calibrated, leading to suboptimal decision-making. The performance of the ML model tends to degrade in unfamiliar instances, making it more appropriate to defer them to human experts. Given their ability to generalize and access broader contextual information, experts are better suited to handle such novel situations. Furthermore, in many real-world systems, it is preferable—or even mandatory—that OOD instances be reviewed by experts to ensure accountability or explainability. To address this, we propose a system where OOD instances are automatically deferred to human experts via rejection learning, while in-distribution instances are assigned through L2D.

We propose a mechanism for assessing epistemic uncertainty on a per-instance basis, which will guide the decision of whether an instance should be processed through L2D or directly deferred to a human expert via rejection learning. In models with a vast hypothesis space \mathcal{H} , the uncertainty related to choosing the “best” model (model uncertainty) is typically small, making approximation uncertainty the primary factor contributing to epistemic uncertainty [43]. Aleatoric uncertainty, which stems from inherent data noise, is significant in regions where classes overlap, whereas epistemic uncertainty dominates in regions where the model has not observed sufficient data. Therefore, by focusing on epistemic

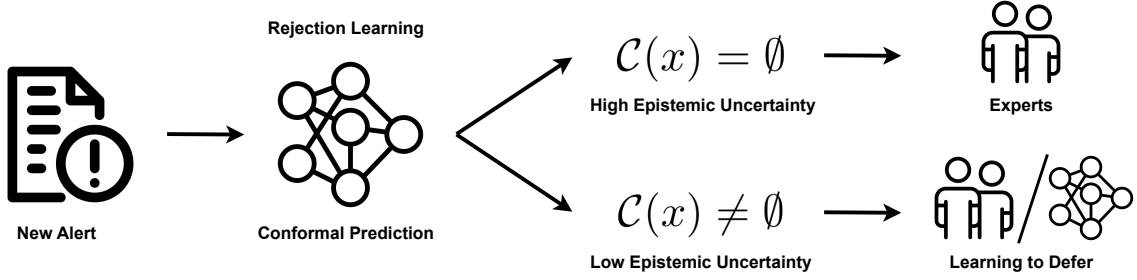


Figure 4.1: Assignment system that balances learning to defer and rejection learning.

uncertainty, we can make informed decisions about when to defer to an expert.

To quantify this uncertainty, we employ density-based conformal prediction [37, 62], which, as previously discussed, allows measuring how well a new instance conforms to the distribution of the training data. If an instance is predicted to belong to a null-set (i.e., it is significantly different from any previously seen examples), this suggests that the instance is OOD and should be assigned to a human expert. Conversely, if the instance is deemed in-distribution (i.e. a non-null set is predicted), L2D can be safely used to select the most likely correct decision-maker based on the model’s predictions (Figure 4.1).

The data and setting assumptions for this chapter remain the same as those described in Section 3.1. In the following sections, we describe the architecture and training process for this hybrid system, illustrating how it effectively balances the strengths of both L2D and rejection learning to handle dynamic environments and distribution shifts with improved robustness.

4.1 Density-Based Conformal Prediction

Density-based conformal prediction, as introduced by Hechtlinger et al. [37] and further explored by Messoudi et al. [62], offers a robust approach to managing uncertainty in classification tasks, particularly in the context of high-dimensional data and dynamic environments. This method divides the labeled data into two subsets: the proper training set $D^{tr} = (X^{tr}, Y^{tr})$, used to build a density estimate $\hat{p}(x|y)$ for each label y , and the calibration set $D^{cal} = (X^{cal}, Y^{cal})$, used to evaluate these density estimates and establish a threshold \hat{q}_y .

The procedure begins by using the proper training set to build $\{\hat{p}(x|y)\}_{y \in \mathcal{Y}}$, an estimate of the class-conditional density $p(x|y)$ for each class $y \in \mathcal{Y}$. The calibration set is then used to determine the empirical $1 - \alpha$ quantile \hat{q}_y of the density values for each class,

$$\hat{q}_y = \sup \left\{ t : \frac{1}{n_y} \sum_{z_i \in D_y^{cal}} \mathbb{I}(\hat{p}(x_i|y) \geq t) \geq 1 - \alpha \right\}, \quad (4.1)$$

where n_y is the number of elements belonging to class y in D^{cal} , $z_i = (x_i, y_i)$, and $D_y^{cal} = \{z_i \in D^{cal} :$

$y_i = y\}$ is the subset of calibration examples in class y . This quantile effectively acts as a threshold, allowing the model to define, for any new observation x_{n+1} , a prediction set

$$\mathcal{C}_\alpha(x_{n+1}) = \{y \in \mathcal{Y} : \hat{p}(x_{n+1}|y) \geq \hat{q}_y\}, \quad (4.2)$$

which includes all classes y for which the observed density is above the threshold.

This method is particularly advantageous in scenarios where the model encounters instances that significantly deviate from the training data, indicating high epistemic uncertainty. When the density $\hat{p}(x_{n+1}|y)$ for a given instance is low across all classes, the method returns an empty set \emptyset , signaling that the model is uncertain and that the instance does not resemble any previously seen examples. This cautious approach is particularly beneficial in high-stakes environments where incorrect predictions can have serious consequences, as it allows the model to effectively express uncertainty and defer to human experts when deemed necessary.

Notably, \mathcal{C}_α uses a conformity score based on density estimation, which measures how well an example fits with others, instead of the more common non-conformity score. Messoudi et al. [62] demonstrate that this conformity score can be transformed into a non-conformity score, achieving an equivalent formulation.

The class-conditional densities $p(x|y)$ must be estimated from the data. Standard kernel density estimation is a viable option, as Lei, Robins, and Wasserman [46] have shown it to be optimal in the conformal setting under weak conditions. Empirically, using the distance from the k nearest neighbors is faster. Despite the challenges of density estimation in high dimensions, this approach has proven effective in previous work [37, 62]. The intuitive reason for this success is that we do not need the density estimates $\hat{p}(x|y)$ to be close to the actual density values $p(x|y)$, we only need the ordering imposed by $\hat{p}(x|y)$ to approximate the ordering defined by $p(x|y)$.

Hechtlinger et al. [37] show that $|P(y \in \mathcal{C}_\alpha(x_{n+1})) - (1 - \alpha)| \rightarrow 0$ as $\min_y n_y \rightarrow \infty$, ensuring the asymptotic validity of the model. The training and prediction algorithms are defined in Algorithms 4.1 and 4.2.

4.2 Learning to Defer and Rejection Learning

As explained above, whether to apply L2D or rejection learning depends on the output of the conformal prediction process. When the conformal prediction outputs a null set for a given instance, it signifies high epistemic uncertainty. In this case, models that estimate expert correctness also become unreliable, as their predictions may be highly uncalibrated for OOD data. Therefore, assigning this instance via L2D is not advisable: rejection learning is preferred, thus deferring the instance to human experts who are better equipped to handle novel situations, possibly by leveraging their broader experience and

Algorithm 4.1: Training algorithm

Input: Training data $Z = (x_i, y_i), i = 1 \dots n$, Class list \mathcal{Y} , Confidence level α , Ratio p

Output: $\hat{p}_{\text{list}}, \hat{q}_{\text{list}}$

Initialize: $\hat{p}_{\text{list}} = \text{list}, \hat{q}_{\text{list}} = \text{list}$

for $y \in \mathcal{Y}$ **do**

$X_y^{tr}, X_y^{cal} \leftarrow \text{SubsetData}(Z, \mathcal{Y}, p)$

$\hat{p}_y \leftarrow \text{LearnDensityEstimator}(X_y^{tr})$

$\hat{q}_y \leftarrow \text{Quantile}(\hat{p}_y(X_y^{cal}), \alpha)$

$\hat{p}_{\text{list}}.\text{append}(\hat{p}_y)$

$\hat{q}_{\text{list}}.\text{append}(\hat{q}_y)$

end for

return $\hat{p}_{\text{list}}, \hat{q}_{\text{list}}$

Algorithm 4.2: Prediction algorithm

Input: Input to be predicted x , Trained $\hat{p}_{\text{list}}, \hat{q}_{\text{list}}$, Class list \mathcal{Y}

Output: \mathcal{C}

Initialize: $\mathcal{C} = \text{list}$

for $y \in \mathcal{Y}$ **do**

if $\hat{p}_y(x) \geq \hat{q}_y$ **then**
 $\mathcal{C}.\text{append}(y)$

end if

end for

return \mathcal{C}

access to contextual information not available to the ML model.

On the other hand, a non empty prediction set indicates that the instance conforms to the data distribution. In these cases, L2D is valuable because the system can assign the decision to the most likely correct decision-maker, whether the ML model or a human expert. Human experts, in particular, may reduce aleatoric uncertainty by incorporating additional information or reasoning at a higher dimensional level, as seen in cases where uncertainty is resolved by embedding data into a higher dimensional space (Figure 2.3). Therefore, when the prediction set is non-null, L2D is the optimal strategy for leveraging both the ML model's predictions and the expert's contextual knowledge.

If rejection learning is applied (i.e., if the conformal prediction outputs a null set), we must assign the instance to a human expert. One effective strategy is to assign it to the available expert with the highest average probability of correctness, calculated based on past expert performance on the training data. This approach ensures that even under uncertainty, the instance is assigned to a reliable decision-maker. Other strategies, such as randomly selecting an expert or considering the performance on the nearest neighbors of the instance, are also possible. All strategies must respect the capacity constraints of human experts, as discussed in Chapter 5.

The L2D framework remains largely the same as described in Chapter 3, where a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ and a rejector $r : \mathcal{X} \rightarrow \{0, 1, \dots, J\}$ are trained to assign decisions to either the classifier or a human expert. However, unlike in the previous chapter, the models are no longer combined with a density

function to account for uncertainty.

It is important to note that our method is versatile and could be adapted to work with any L2D approach that provides an estimate of correctness for each expert. However, we have chosen to employ the OvA method to have a fair comparison between both our methods. Additionally, the OvA approach allows working in scenarios where only one expert prediction is available per instance, addressing the data availability limitations that many L2D systems do not adequately consider.

Chapter 5

Cost-Sensitive Learning and Capacity Constraints

5.1 Cost-Sensitive Learning

All previously discussed methods focus on optimizing models to minimize the error rates. We now turn our attention to scenarios where each instance has its own cost, meaning that it is more costly to make errors in some instances than others (e.g. false positive (FP) and false negative (FN) errors may have different costs). In these cases, our objective shifts from minimizing the error rate to minimizing the overall misclassification cost.

To adapt our method to handle varying cost structures, we employ an instance re-weighting technique, as proposed by Zadrozny et al. [94] and Elkan [23], and further applied by Alves et al. [5] in the context of cost-sensitive L2D. This approach involves scaling each loss in the training process by the misclassification cost c_i associated with each instance. By doing so, the surrogate losses used for training both the classifier function g and the expert-related functions $g_{\perp,j}$ are optimized to reduce the total misclassification cost, instead of minimizing the error rate. We will now provide a detailed explanation of how this re-weighting process is applied to the training of classifiers and scoring functions.

When training a classifier h through a scoring function g , the surrogate loss function ℓ is typically approximated by averaging the individual losses across the entire training set

$$\mathbb{E}_{(x,y)\sim D}[\ell(x, g(x), y)] \approx \frac{1}{N} \sum_{i=1}^N \ell(x_i, g(x_i), y_i), \quad (5.1)$$

where D represents the data distribution. However, in situations where different instances incur varying misclassification costs, using a standard surrogate for the 0-1 loss is insufficient. Each instance x_i in

our data has an associated misclassification cost c_i , and our goal is to train a model that minimizes the expected misclassification cost,

$$\mathbb{E}_{x,y,c \sim D}[c \cdot \mathbb{I}(h(x) \neq y)], \quad (5.2)$$

where each example is now a triplet (x, y, c) of feature vector x , label y , and cost c . Simply minimizing the expected error rate $\mathbb{E}_{x,y \sim D}[\mathbb{I}(h(x) \neq y)]$, as we would with standard surrogate losses, does not align with our cost-sensitive objective. Zadrozny et al. [94] suggest a solution by redefining the data distribution \tilde{D} as

$$\tilde{D}(x, y) = \frac{c}{\mathbb{E}_{c \sim D}[c]} D(x, y, c). \quad (5.3)$$

Under this redefined distribution, minimizing the expected error rate becomes equivalent to minimizing the expected misclassification cost under the original distribution D (see Equation 5.2),

$$\mathbb{E}_{x,y \sim \tilde{D}}[\mathbb{I}(h(x) \neq y)] = \frac{1}{\mathbb{E}_{c \sim D}[c]} \mathbb{E}_{x,y,c \sim D}[c \cdot \mathbb{I}(h(x) \neq y)] \propto \mathbb{E}_{x,y,c \sim D}[c \cdot \mathbb{I}(h(x) \neq y)]. \quad (5.4)$$

Therefore, to achieve a classifier that optimally reduces misclassification costs, we can train using this adjusted distribution \tilde{D} and a conventional surrogate loss (e.g. log-loss). In practical terms, this entails adjusting the weight of each instance during training according to its associated cost, allowing the model to prioritize instances based on their relative importance, thus minimizing the empirical estimate of the misclassification cost

$$\sum_{i=1}^N c_i \ell(x_i, g(x_i), y_i). \quad (5.5)$$

To implement this approach in our framework, we modify the standard log-loss functions, used for training both the classifier g and the expert-related functions $g_{\perp,j}$, by incorporating the misclassification costs as instance weights. The adjusted empirical losses are defined as:

$$\mathcal{L}'_{\text{classifier}}(\{x_i, y_i, c_i\}_{i=1}^N, g) = \frac{1}{N} \sum_{i=1}^N c_i \left[-y_i \log(\psi^{-1}(g(x_i))) - (1 - y_i) \log(1 - \psi^{-1}(g(x_i))) \right] \quad (5.6)$$

$$\begin{aligned} \mathcal{L}'_j(\{x_i, y_i, c_i, m_{i,j}\}_{i=1}^N, g_{\perp,j}) &= \frac{1}{N} \sum_{i=1}^N c_i \left[-\mathbb{I}[m_{i,j} = y] \log(\psi^{-1}(g_{\perp,j}(x_i))) \right. \\ &\quad \left. - \mathbb{I}[m_{i,j} \neq y] \log(1 - \psi^{-1}(g_{\perp,j}(x_i))) \right] \end{aligned} \quad (5.7)$$

5.2 Human Work Capacity Constraints

Up until now, we have addressed a significant limitation found in many previous human-AI collaboration systems: their inability to effectively handle dynamic environments. Another critical limitation that must not be overlooked is the work capacity limitation faced by human experts. In practical settings, experts cannot process an unlimited number of instances, especially in real-time or high-volume scenarios. It is therefore essential to incorporate these capacity constraints into our system to ensure it can produce feasible solutions in practical applications.

To address this, we propose a solution for incorporating human work capacity constraints into both of our assignment systems. For the density-softmax approach outlined in Chapter 3, we adopt the strategy outlined by Alves et al. [5], as it is applicable to any purely L2D framework. In contrast, for the conformal prediction approach proposed in Chapter 4, we extend this framework by developing a method that optimizes the balance between L2D and rejection learning in the presence of capacity constraints.

Instead of considering the entire dataset at once, we define these constraints over batches of instances. This approach aligns with practical scenarios where data is processed in stages and also ensures that the system can dynamically adjust to varying workloads. Specifically, in a dataset composed of N instances, we represent capacity constraints using two key components: a batch vector \mathbf{b} and a human capacity matrix \mathbf{H} . The batch vector \mathbf{b} is used to assign each instance i to a specific batch, where b_i denotes which batch the instance $i \in \{1, \dots, N\}$ belongs to. Formally, the vector \mathbf{b} can be expressed as

$$\mathbf{b} = [\underbrace{1, 1, \dots, 1}_{n_1 \text{ times}}, \underbrace{2, 2, \dots, 2}_{n_2 \text{ times}}, \dots, \underbrace{B, B, \dots, B}_{n_B \text{ times}}]$$

where n_1, n_2, \dots, n_B represent the number of instances in each respective batch, and B is the total number of batches.

The human capacity matrix \mathbf{H} is then defined to specify the maximum number of instances each expert can handle within each batch. The element $H_{b,j}$ of the matrix represents the maximum number of instances from batch b that expert j can process. For example, if we have a scenario where each expert can handle 10 instances per batch, matrix \mathbf{H} is given by

$$\mathbf{H} = \begin{bmatrix} 10 & 10 & \cdots & 10 \\ 10 & 10 & \cdots & 10 \\ \vdots & \vdots & \ddots & \vdots \\ 10 & 10 & \cdots & 10 \end{bmatrix}_{B \times J}$$

To define the possible assignment decisions, consider a binary classification scenario where $\mathcal{Y} = \{0, 1\}$. The assignment decision a_i for each instance i can take one of several values from the set $\{1, \dots, J + 2\}$. Specifically, $a_i = y + 1$ (where $y \in \mathcal{Y} = \{0, 1\}$) indicates an automatic prediction of class y for instance i . Alternatively, if $a_i = j + 2$ (where $j \in \{1, \dots, J\}$), the instance i is deferred to the j th

expert. This framework can be easily extended to handle multiclass classification scenarios.

When adapting a L2D system under capacity constraints, the objective is to maximize the expected probability of correctness across all instances in a batch. The estimated probability of correctness for all possible assignments is represented as

$$\hat{\mathbb{P}}(\text{correct}|x) = \begin{cases} 1 - \sigma(\hat{g}(x)) & \text{if } a(x) = 1 \\ \sigma(\hat{g}(x)) & \text{if } a(x) = 2 \\ \sigma(\hat{g}_{\perp,j}(x)) & \text{if } a(x) = j + 2, \end{cases} \quad (5.8)$$

where the sigmoid function σ is the inverse link function for the surrogate loss used in training, $\hat{g}(x)$ is the output from the main ML classifier, and $\hat{g}_{\perp,j}(x)$ is the output from the model that predicts the probability of correctness for expert j .

Finally, to represent the assignment decision over a batch b of n_b instances, consider the $n_b \times (2 + J)$ matrix of assignments \mathbf{A} , where each element A_{i,a_i} is a binary variable that denotes if the assignment decision a_i is taken for instance i . Alves et al. [5] determine the optimal set of assignments under capacity constraints by solving the following optimization problem:

$$\begin{aligned} \mathbf{A}^* &= \arg \max_{\mathbf{A} \in \{0,1\}^{n_b \times (2+J)}} \sum_{i=1}^{n_b} \sum_{a_i=1}^{J+2} \hat{\mathbb{P}}(\text{correct}|x_i, a_i) A_{i,a_i}, \\ \text{s.t.} \quad &\sum_{i=1}^{n_b} A_{i,a_i} = H_{b,a_i}, \quad \text{for } a_i \in \{3, \dots, J+2\}, \\ \text{and} \quad &\sum_{a_i=1}^{J+2} A_{i,a_i} = 1, \quad \text{for } i \in \{1, 2, \dots, n_b\}. \end{aligned} \quad (5.9)$$

The first constraint reflects the capacity limits of human experts: it ensures that the number of instances assigned to each expert is equal to the predefined capacity. The second constraint ensures that each instance is assigned to exactly one decision-maker. For instance, consider a scenario with $J = 3$ experts, each capable of handling 10 instances per batch. An example assignment matrix \mathbf{A} could be

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix},$$

$$\sum_{i=1}^{n_b} A_{i,a_i} = 10, \forall a_i \in \{3, 4, 5\},$$

where each of the last 3 columns sums to 10, indicating that each expert is assigned 10 instances and each row sums to 1, indicating that each instance is assigned to exactly one expert.

This method can be applied to any L2D system that provides estimates for the probability of correctness for each decision-maker. In our case, the density-softmax approach operates as a purely L2D system, where uncertainty estimation is incorporated directly into the binary classifiers. Thus, we employ the previously described method. However, in our conformal prediction approach, the system no longer functions purely as an L2D framework. Instead, uncertainty is first estimated to determine whether an instance should be handled through L2D or deferred as in rejection learning. To formalize the process of selecting between L2D and rejection learning, we define the binary function

$$f_\alpha(x_i) = \begin{cases} 0 & \text{if } \mathcal{C}_\alpha(x_i) = \emptyset \\ 1 & \text{if } \mathcal{C}_\alpha(x_i) \neq \emptyset, \end{cases} \quad (5.10)$$

where $f_\alpha(x_i)$ is a function that depends on the coverage level α and the instance x_i . If the conformal set $\mathcal{C}_\alpha(x_i)$ is empty, $f_\alpha(x_i)$ outputs 0, indicating that rejection learning should be applied and the instance should be deferred to an expert. Conversely, if $\mathcal{C}_\alpha(x_i)$ is not empty, the function outputs 1, signaling that L2D should be employed and the instance assigned to the decision-maker—either the ML model or an expert—with the highest probability of correctness.

In this system, the coverage level α defines how rejection learning and L2D are balanced. Lower values of α result in a more conservative detection of OOD instances, leading to more instances being processed through L2D. Conversely, higher values of α classify more instances as OOD, immediately deferring them to human experts. Since data distributions can vary between batches, it is important to allow the coverage level α to adapt dynamically. Therefore, we optimize α alongside the assignment matrix \mathbf{A} , ensuring the most appropriate decisions are made based on the batch's characteristics.

The optimal assignment matrix \mathbf{A}^* and coverage level α^* are determined by solving the following optimization problem, which maximizes the expected correctness of assignments under capacity constraints:

$$\begin{aligned} (\mathbf{A}^*, \alpha^*) &= \arg \max_{\mathbf{A} \in \{0,1\}^{n_b \times (2+J)}, \alpha} \sum_{i=1}^{n_b} \sum_{a_i=1}^{J+2} \left(f_\alpha(x_i) \frac{1}{w_i} \hat{\mathbb{P}}(\text{corr.} \mid x_i, a_i) + (1 - f_\alpha(x_i)) \hat{\mathbb{P}}(\text{corr.} \mid X_{\text{train}}, a_i) \right) A_{i,a_i}, \\ \text{s.t. } \sum_{i=1}^{n_b} A_{i,a_i} &= H_{b,a_i} \text{ for } a_i \in \{3, \dots, J+2\}, \\ \sum_{a_i=1}^{J+2} A_{i,a_i} &= 1 \text{ for } i \in \{1, \dots, n_b\} \text{ and } a_i > 2 \text{ if } f_\alpha(x_i) = 0. \end{aligned} \quad (5.11)$$

This optimization problem introduces several new components compared to the previous one (Equation (5.9)). First, the third constraint ensures that when rejection learning is applied ($f_\alpha(x_i) = 0$), the instance is deferred to a human expert rather than the ML model, preventing OOD instances from being handled by the model. In the objective function (Equation (5.11)), the estimated probability of correctness

$\hat{\mathbb{P}}(\text{correct} \mid x_i, a_i)$ remains, but for OOD instances, the expert’s average correctness $\hat{\mathbb{P}}(\text{correct} \mid X_{\text{train}}, a_i)$, which is derived from the training data, is used instead. Finally, a new downweighting factor $\frac{1}{w_i}$ adjusts for uncertainty or overconfidence in predicted probabilities.

The downweighting factor $\frac{1}{w_i}$ is determined by the conformal coverage level α and the proportion of null set predictions in the batch. For each new instance, we compute two density scores $\hat{p}(x_i|y)$ (one for each class in a binary setting). If neither score exceeds the quantile \hat{q}_y defined during calibration, the instance is classified into the null set. These quantiles are selected to meet the desired coverage level α . Lower α values represent a more conservative approach to detecting OOD data, prioritizing accuracy and reducing null set predictions, while higher α allows for more null set predictions with a higher error rate (Figure 5.1). For each instance, α is gradually increased until α_\emptyset , the smallest value at which the instance falls into the null set, is identified. This process requires minimal additional computation since the density scores are already available; we simply need to find the $1 - \alpha$ quantile. The smaller α_\emptyset , the more OOD the instance is considered, and one approach is to set $w_i = \alpha_\emptyset$.

It’s important to differentiate between instances classified as null sets due to the desired coverage level and those classified as null sets due to data drift. In the absence of data drift, the proportion of null set predictions should approximate α but should not exceed it. If the proportion of null set predictions does exceed α , this suggests that accuracy has fallen below $1 - \alpha$, indicating a lack of data exchangeability and confirming the presence of drift (as illustrated in Figure 5.1). Consequently, we should only downweight instances classified as null sets due to data drift, rather than those classified as null sets to meet the coverage level.

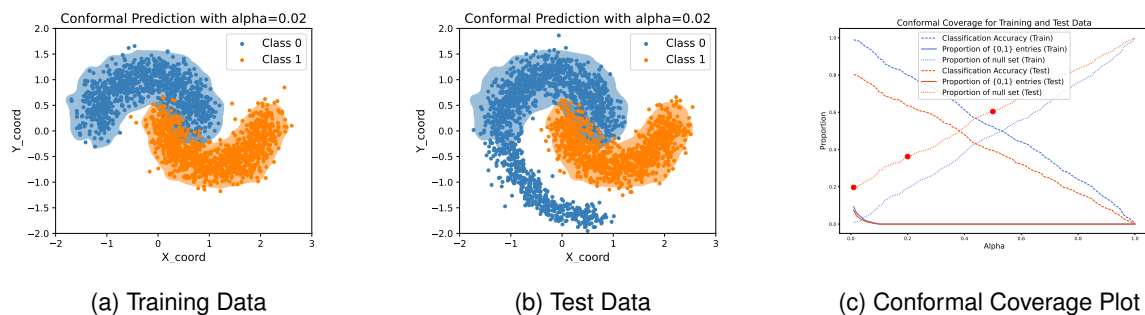


Figure 5.1: Example of the density-based conformal method trained and calibrated on the training data (a), and evaluated on the test data (b). The conformal coverage plot (c) indicates the presence of distribution drift, as evidenced by the proportion of null sets exceeding alpha, which in turn decreases the accuracy below the defined $1 - \alpha$ coverage level. This implies that the data is no longer exchangeable, confirming the presence of drift.

To build on this reasoning, we can detect OOD data by monitoring the ratio $\rho_{\emptyset, \alpha_\emptyset} / \alpha_\emptyset$, where $\rho_{\emptyset, \alpha_\emptyset}$ represents the proportion of null set predictions at a specific coverage level α_\emptyset . This ratio compares the actual proportion of null set predictions with the expected proportion based on α_\emptyset . A high ratio suggests that the proportion of null set predictions is higher than what is anticipated under normal circumstances,

indicating potential data drift or OOD instances. By calculating this ratio for each instance, we can distinguish null set predictions due to OOD data from those due to the desired coverage level α_\emptyset .

For example, in Figure 5.1 (c), consider instances predicted as null sets at $\alpha_\emptyset = 0$. Since $\rho_{\emptyset, \alpha_\emptyset} = 0.2$, the ratio becomes $\frac{0.2}{0} = \infty$, leading to these instances being fully downweighted with $\frac{1}{\infty} = 0$. For instances with $\alpha_\emptyset = 0.2$, the ratio is $\frac{0.4}{0.2} = 2$, resulting in a downweight factor of $\frac{1}{2}$.

One challenge here is that an instance predicted as a null set at a particular α_\emptyset will also remain a null set for all higher α values. This inflates the null set proportion $\rho_{\emptyset, \alpha_\emptyset}$, not specifically due to the amount of null sets at α_\emptyset , but due to previously identified null sets at lower α levels. For instance, in the previous example, all OOD data is detected for α as low as 0.02; however, the instance that is predicted as the null set for $\alpha = 0.2$ is still downweighted by 0.5.

To address this, rather than directly comparing $\rho_{\emptyset, \alpha_\emptyset}$ with α_\emptyset , we fit a line between the points $(\alpha_\emptyset - s, \rho_{\emptyset, \alpha_\emptyset - s})$ and $(1, 1)$, where s is the step size between successive values of α . This approach is based on the assumption that at $\alpha = 1$, we expect 100% of instances to be classified as null sets. The value of this line provides a heuristic for the expected proportion of null sets, assuming no additional OOD data is identified as α increases. Therefore, this adjustment allows determining if additional OOD data is identified at α_\emptyset . The resulting weight w_i for an instance x_i is then calculated as:

$$w(x_i, \mathcal{X}_{\text{batch}}, \mathcal{C}_{\{\alpha_i\}_{i=1}^n}) = \frac{\rho_{\emptyset, \alpha_\emptyset}}{\rho_{\emptyset, \alpha_\emptyset - s} + \frac{1 - \rho_{\emptyset, \alpha_\emptyset - s}}{1 - (\alpha_\emptyset - s)} s}, \quad (5.12)$$

where $\rho_{\emptyset, \alpha_\emptyset} = \frac{|\mathcal{C}_{\alpha_\emptyset}(\mathcal{X}_{\text{batch}}) = \emptyset|}{|\mathcal{X}_{\text{batch}}|}$ represents the proportion of null set predictions at α_\emptyset , based on the instance x_i and the batch $\mathcal{X}_{\text{batch}}$. The denominator provides the expected proportion of null sets, derived from fitting the line as described, where s is the step size between successive values of α . If $\alpha_\emptyset = 0$, the denominator is set to 0, ensuring that fully OOD instances are appropriately downweighted.

By applying this approach to the example in Figure 5.1 (c), the first point (corresponding to $\alpha_\emptyset = 0$) would still be significantly downweighted, as the high proportion of null set predictions indicates a clear detection of OOD data. However, the second point, which corresponds to a higher α_\emptyset , would not be downweighted at all, since all OOD data had already been identified at lower α levels. This behavior is exactly what we want, as it prevents the system from penalizing in-distribution data that comes after OOD data has been detected at lower α_\emptyset .

Both assignment problems 5.9 and 5.11 are solved using the constraint programming solver CP-SAT from Google Research's OR-Tools [71]. We selected CP-SAT as it has been repeatedly shown to be the best performing publicly available solver in a wide array of constraint programming problems in the MiniZinc challenge [82], where solvers are tasked with finding the optimal or a near-optimal solution within a time-limit.

Chapter 6

Experimental Setup for Financial Fraud Detection

In this chapter, we outline the experimental setup for training and evaluating a human-AI collaboration system within the context of financial fraud detection. In a real-world fraud detection scenario, the volume of instances far exceeds human capacity, making it impractical to assume that human experts will provide predictions for the same number of instances as the ML model. To address this, we explain how a separate ML model (denoted as alert model) is trained to flag potential fraud cases (Section 6.3), which are subsequently dealt with through our proposed assignment systems. This ensures humans review the most costly instances - the ones most likely to be fraud. For this setup, we use a synthetic tabular dataset replicating real-world bank account opening applications, detailed in Section 6.1. To simulate different scenarios of data drift, we introduce noise in the test set, as described in Section 6.4. Since this dataset does not provide human expert predictions, Section 6.5 presents the methodology used to generate realistic expert predictions. Additionally, we discuss the cost-sensitive optimization objective (Section 6.2), as well as a wide variety of configurations for data availability and capacity constraints (Section 6.7). Finally, we outline the baselines used to evaluate our approach (Section 6.8) and detail the training methods implemented for both uncertainty estimation techniques (Section 6.9).

6.1 Dataset

The bank-account-fraud (BAF) tabular dataset, introduced by Jesus et al. [45], is a synthetic replication of a real-world fraud detection dataset, generated using a CTGAN [92], consisting of one million rows spread over eight months. Each row represents a bank account opening application, with detailed information about the application and the applicant. Labels indicate whether each application was fraud-

ulent (1) or legitimate (0). Fraudsters may impersonate legitimate individuals or create fictitious identities to gain unauthorized access to banking services, resulting in financial losses for the bank. A significant challenge in training ML models on this dataset is its high class imbalance, with only about 1% of the applications being fraudulent. Additionally, the dataset captures distributional changes over time. The dataset includes 30 features: 19 numeric, 6 binary, and 5 categorical.

In this dataset, a positive prediction indicates fraud and leads to application rejection, while a negative prediction indicates legitimacy and results in account opening. The costs associated with false positives (rejecting a legitimate application) and false negatives (accepting a fraudulent application) are different, with false positives leading to the loss of potential customers and false negatives resulting in financial losses to the bank.

6.2 Misclassification Costs

Fraud detection is a cost-sensitive task, requiring a balance between the cost of false positive errors c_{FP} and false negative errors c_{FN} . Standard metrics like accuracy are not adequate in such scenarios, making it essential to adopt metrics that better capture the cost errors. Jesus et al. [45] employ a Neyman-Pearson criterion, which, in this case, involves maximizing recall (true positive rate - TPR) at a fixed 5% FPR. This criterion is commonly imposed by clients in the fraud detection domain as it balances fraud detection with minimizing customer attrition, accounting for the trade-off between c_{FP} and c_{FN} and the low prevalence of fraud. However, as our optimization objective is defined as a Neyman-Pearson Criterion, we do not have direct access to the values of c_{FP} and c_{FN} , which are necessary to apply the cost-sensitive learning method described in Section 5.1.

In a cost-sensitive task, assuming no cost for correct classifications, the objective is to generate a set of predictions \hat{y} that minimize the expected misclassification cost

$$\frac{1}{N} \sum_{i=1}^N [\lambda \mathbb{I}[y_i = 0 \wedge \hat{y}_i = 1] + \mathbb{I}[y_i = 1 \wedge \hat{y}_i = 0]], \quad (6.1)$$

where $\lambda = c_{FP}/c_{FN}$ represents the relative cost of false positives to false negatives. Given this, we derive a cost trade-off parameter λ based on the Neyman-Pearson criterion, which enables setting $c_{FP} = \lambda$ and $c_{FN} = 1$ for our misclassification cost re-weighting approach. This approach is valid because scaling the optimization objective by a constant factor does not alter the optimal solution.

According to Elkan [23], a relationship between a binary classifier's ideal threshold t and the misclassification costs c_{FP} and c_{FN} can be established. Given that, for any instance x_i , the optimal classification minimizes the expected cost, the positive class is predicted if and only if the expected cost of predicting

1 is less than or equal to the expected cost of predicting 0, that is

$$(1 - p)c_{FP} \leq p c_{FN}, \quad (6.2)$$

where $p = P(y = 1|x_i)$ is the posterior probability that x_i belongs to the positive class. The alert model (described in detail in section Section 6.3) provides an estimate of p through its score output for each instance. When the inequality becomes an equality, we obtain $p = t$, leading to

$$(1 - t)c_{FP} = tc_{FN} \implies \lambda_t = \frac{t}{1 - t} = 0.056. \quad (6.3)$$

Given that the optimal threshold t for our alert model is chosen to satisfy the Neyman-Pearson criterion, we can derive the misclassification costs as $c_{FP} = \lambda_t$ and $c_{FN} = 1$ by using this theoretical value λ_t .

6.3 Alert Data Setup

In prior L2D research, it is typically assumed that new instances can be assigned either to the ML model or human experts. However, in many real-world scenarios like fraud detection, it is common to use an alert model to screen instances and raise alerts that are then reviewed by humans [18]. This way humans end up only predicting on a critical subset of the feature space, making the most of their work-capacity. Our assignment systems will function alongside an alert model, which screens instances and raises alerts; our systems will then be trained and tested on these flagged instances.

The alert model is trained using the first three months of the dataset and is validated on the fourth month (Figure 6.1). We employ the LightGBM algorithm [48] due to its high performance on tabular data [8, 80] and its efficiency in large datasets compared to other gradient boosting methods, providing similar performance with significantly shorter computation times [48]. The alert model is trained by minimizing the binary cross-entropy loss. Details on the training process are provided in Appendix A.1.

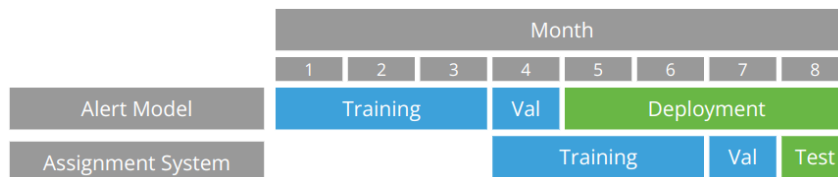


Figure 6.1: Training, validation, and deployment setup for the alert model and the assignment system. The alert model is trained on data from the first three months and validated on the fourth month, before being deployed in months 4 to 8. The assignment system is then trained using alerts from months 4 to 6, validated on month 7, and tested on the 8th month to evaluate its performance.

The alert model achieved a recall of 58.48% at 5% false positive rate (FPR) on the validation set,

using a decision threshold of $t = 0.0526$. When deployed on data from months 4 to 8, where the alerts raised are used to train and test our system, the model’s recall was 50.93% with 4.2% FPR, using the same threshold. The receiver operating characteristic (ROC) curve for the alert model on the validation and deployment split is presented in Figure 6.2.

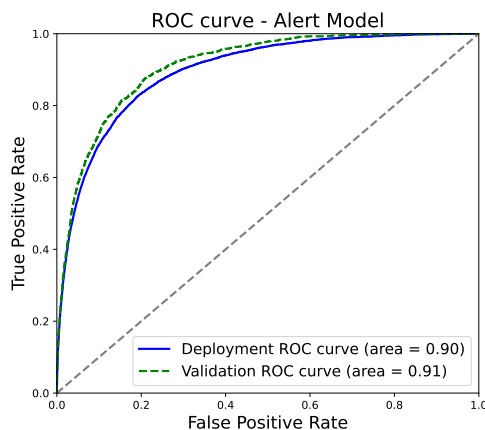


Figure 6.2: ROC curve for the alert model on the validation and deployment split.

In the deployment phase, spanning months 4 to 8, the alert model flagged approximately 29,000 instances, with a fraud prevalence of 12%. This flagged data will serve as the basis for training and testing our assignment systems. By focusing on this smaller subset, we simulate a more realistic human-AI collaboration scenario where experts are only involved in reviewing flagged cases.

6.4 Noise injection

Our training and validation data consists of alerts generated by the alert model from the fourth to the seventh month, while the eighth month is reserved exclusively for testing (Figure 6.1). To evaluate our system’s performance under data drift and assess its ability to detect OOD instances, noise was introduced into the test set. A primary motivation for this is to create a known subset of altered data, allowing for targeted evaluation of our systems on these modified instances. Additionally, the results from the alert model indicate similar performance in both validation and deployment splits, suggesting minimal data drift. To simulate different levels and types of noise, five distinct variants of the test set were created by introducing different amounts of noise on different subsets of the data. These variants include:

- **Low, medium, and high noise settings:** noise is introduced in all features, with magnitude varying between settings.
- **Categorical noise setting:** noise is introduced exclusively in the categorical features.

- **Numerical noise setting:** noise is introduced only in the numerical features.

For numerical features, the data was standardized to zero mean and unit variance before adding Gaussian zero-mean noise and variable standard deviation: 1.0 (low noise), 1.5 (medium and numerical noise), and 2.0 (high noise). After introducing noise, the data was transformed back to its original representation. For discrete-valued features, values were rounded to the nearest integer value. For features with a defined range, the values were clipped to remain within their bounds. For categorical and flag features, noise was introduced by randomly switching to another categorical/binary value with probabilities of 0.3 (low noise), 0.4 (medium and categorical noise), and 0.5 (high noise).

As a result, each test set consists of 80% original data and 20% noisy data. This noise injection process was repeated five times for each of the five settings, using different random seeds to introduce variability in the noise addition. This allows accounting for statistical fluctuations in the noise injection process, by reporting the average performance over these 5 random seeds.

6.5 Synthetic Expert Decision Generation

Datasets with human predictions are scarce and obtaining expert predictions is often prohibitively expensive. To address this difficulty, prior research on generating synthetic human decision-makers typically involves introducing label noise to simulate human errors. For example, Verma and Nalisnick [90] use the CIFAR-10 dataset [52], and simulate experts who predict with 70% accuracy on a subset of the classes and make random predictions on the remaining classes. For the same dataset, Mozannar and Sontag [64] and Charusaie et al. [11] simulate experts by defining perfect accuracy on a fraction of the classes and random predictions on the rest. On the Hate Speech and Offensive Language Detection dataset, Keswani et al. [49] simulate racially biased experts by setting varying error probabilities based on whether a tweet is labeled as African-American English or not, i.e., experts are more likely to make an error if the tweet contains African-American English.

However, these approaches have a major limitation: they define expert accuracy based solely on the label of each instance or on a single feature. This overlooks the influence of the full set of input features on the probability of error and restricts the ability to simulate a diverse team of human experts with varied expertise. Alves et al. [5] address this limitation by introducing instance-dependent label noise, making the probability of error a function of instance features and any other information available to human experts at the time of decision-making (i.e., the alert model's score, which is often shown to decision makers [18]). This creates a more complex simulation of human decision-making, capturing how different input characteristics influence expert behavior. We follow this approach in our experimental setup, generating expert decisions that better reflect the complexities of real-world expertise, where error probabilities vary dynamically based on the properties of each instance.

Each expert's synthetic predictions are generated by flipping the label y_i with probability $P(m_{j,i} \neq y_i | x_i, y_i)$. In our human-AI collaboration systems, the expert's decision is also influenced by the ML model's score $M(x_i)$, as this is common in other human-AI collaboration systems [18, 44]. To account for this, we set the expert's probability of error for each instance as a function of the pre-processed features \bar{x}_i and the alert model's score $M(x_i)$. The probabilities of error are given by

$$\begin{cases} P(m_{j,i} = 1 | y_i = 0, x_i) = \sigma \left(\beta_0 - \alpha \frac{w \cdot \bar{x}_i + w_M M(x_i)}{\sqrt{\|w\|^2 + w_M^2}} \right) \\ P(m_{j,i} = 0 | y_i = 1, x_i) = \sigma \left(\beta_1 + \alpha \frac{w \cdot \bar{x}_i + w_M M(x_i)}{\sqrt{\|w\|^2 + w_M^2}} \right), \end{cases} \quad (6.4)$$

where σ is the sigmoid function. Each expert's error probabilities are controlled by five parameters: β_0 , β_1 , α , w , and w_M . The weight vector w and w_M define the relative influence of each feature and the ML model score on the error likelihood. The features are preprocessed by normalizing numeric features to the $[-0.5, 0.5]$ range and target-encoding categorical features to the $[0, 1]$ interval and shifted to have zero mean. Normalizing the feature weights enables α to control the impact of the features on the instance-wise probability of error, while β_0 and β_1 serve the purpose of controlling the base error rates.

The expected cost resulting from an expert's decisions is given by

$$\begin{aligned} \mathbb{E}[C]_j &= \mathbb{E}_y [\lambda \mathbb{P}(m_j = 1 \wedge y = 0) + \mathbb{P}(m_j = 0 \wedge y = 1)] \\ &\approx \frac{1}{N} \sum_{i=1}^N [\lambda \mathbb{P}(m_{j,i} = 1 | y_i = 0) \mathbb{P}(y_i = 0) + \mathbb{P}(m_{j,i} = 0 | y_i = 1) \mathbb{P}(y_i = 1)]. \end{aligned} \quad (6.5)$$

We assume that, on average, human experts perform no worse than the ML classifier. If this were not the case, randomly assigning instances to human experts would degrade the system performance, thus making the alert review system harmful. Since we can train the ML classifier before generating synthetic expert predictions, we know its average misclassification cost on the training data, $\mathbb{E}[C]_h = 0.04$. We use this value to sample the target misclassification cost for each expert, $\mathbb{E}[C]_j$, from the normal distribution $T_{\mathbb{E}[C]_j} \sim \mathcal{N}(\mathbb{E}[C]_h, 0.2\mathbb{E}[C]_h)$, aligning the expert's performance with that of the classifier. Note that we are using the classifier's training misclassification cost, but at test time, the classifier is more affected by drift and noise than the experts, giving the experts a performance advantage, as expected. Based on the target misclassification cost, we first sample each expert's FPR randomly and then calculate the corresponding false negative rate (FNR) required to achieve $T_{\mathbb{E}[C]_j}$ (or vice versa) using a partition of the dataset (month 7). With the FPR and FNR values, we determine the parameters β_0 and β_1 by following the methodology outlined by Alves et al. [5]. In our experiments, we consider a range of 1 to 5 experts. To ensure our results are not influenced by randomness in the selection of expert properties, we generate 15 synthetic experts and randomly sample experts using a different random seed for each experimental run (details on the testing scenarios are provided in Section 6.7).

6.6 Expert Properties

To illustrate how different experts are influenced by input features and the model score, we display a heatmap of the normalized weight vector w for each expert in Figure 6.3. The heatmap shows the varying degrees to which different features affect the experts' probabilities of error. Notably, all experts are also influenced by the model score, reflecting the human-AI collaboration setup where the model's output may affect expert decisions [18, 44].

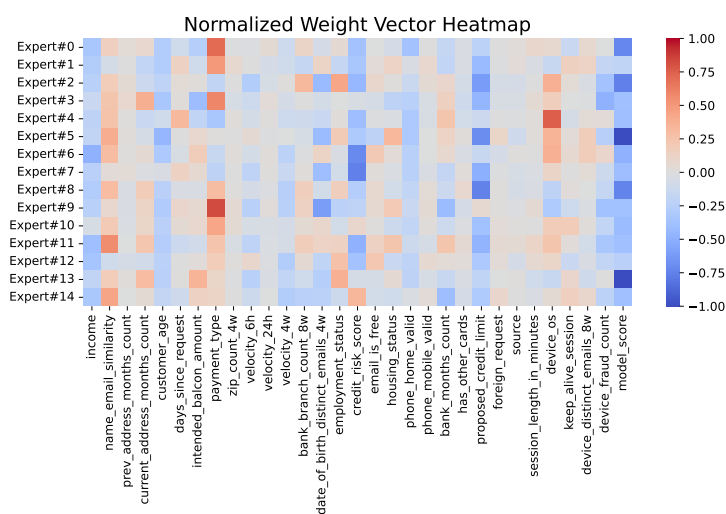


Figure 6.3: Weight feature vector w heatmap for each expert.

In our work, we aim to explore the benefits of integrating decisions from both human experts and an ML classifier by taking advantage of each human's and the ML classifier's complementary expertise. To verify this assumption, Figure 6.4 presents a heatmap showing the proportion of cases where the decision-maker in a given row correctly predicted the fraud label while the one in each column did not. These values, derived from the medium noise test setting, are representative of similar patterns across other noise levels, making this a comprehensive single representation. These results show that each decision-maker excels over another in a significant number of instances, as their strengths vary across different regions of the feature space. This demonstrates that this experimental setup is appropriate to test our human-AI collaboration methods, as there is performance gain to be had by optimizing assignments.

6.7 Data Availability and Capacity Constraints

As mentioned previously in Section 2.4, the need for human predictions for every training instance is not feasible for large datasets due to the high cost involved in obtaining human labels. In most decision-

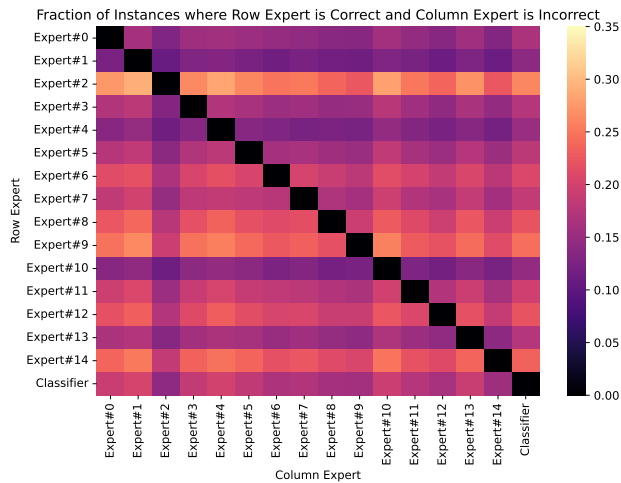


Figure 6.4: Fraction of instances in which the row expert is correct and column expert is incorrect.

making tasks where a team of humans is involved, due to the limited human work capacity, each instance is reviewed by a single human worker [18], which would result in having a single expert label per training instance.

To simulate realistic scenarios where expert labels may not be available for all training instances, we experiment with four different data availability conditions. The first condition assumes full availability of expert labels for all instances, allowing us to train each expert model on the entire training set. This is commonly done in previous L2D work [39, 89]. In the remaining three conditions, expert labels are available only for a fraction of the training data, with each expert labeling either 1/5 (i.e., up to one expert per instance), 1/20, or 1/40 of the total instances. For the density-softmax approach, the density models are trained on the same subset of data used for training the expert models. To introduce variability, subsets of training data are randomly distributed among experts across five different random seeds for each different data availability scenario.

Additionally, we impose uniform work capacity constraints on the experts, assuming that humans have similar working capacities. This entails that for each deferral rate (the percentage of alerted instances assigned to experts), the capacity constraints are evenly distributed across all experts. We conduct experiments across multiple testing scenarios with deferral rates ranging from 10% to 50%, in order to evaluate whether our system can accurately detect OOD data and effectively defer those instances to human decision-makers, even in settings with low deferral rates.

Given the various data availability and capacity constraint scenarios, along with the different types and magnitudes of noise in the test set (Section 6.4) and the number of experts considered (Section 6.5), our testing scenarios are defined by the following four variables:

- **Noise:** five different settings that vary in type and magnitude, as described in detail in Section 6.4.

- **Number of experts (NE):** ranging from 1 to 5, in increments of 1.
- **Deferral rate (DR):** the percentage of instances deferred to human experts, with values from 10% to 50%, in 10% steps.
- **Data availability (DA):** the fraction of human-labeled training data available for training expert models, defined by four scenarios, as outlined in this section.

Together, these configurations create 500 distinct testing scenarios, far too many to be fully presented in this thesis. Therefore, results will often focus on a representative subset of these settings.

6.8 Baselines

Learning to defer: as a baseline for L2D, we adopt the OvA L2D algorithm proposed by Verma et al. [89]. While the method was described in detail in Section 3.3, for this baseline we use a standard LightGBM model for each binary classifier, instead of the distance-aware models discussed previously. In other words, this is exactly our density-softmax approach without distance-awareness, so any change in results is due to the contributions made in this thesis. It is also the same L2D method used in the conformal prediction approach when an instance is not deemed OOD (i.e. when it is not predicted as the null set) so it also offers a relevant baseline in this case.

Rejection learning: in this approach, we must first define an uncertainty measure to decide whether to defer an instance to a human expert or assign it to the classifier. We employ density-based conformal prediction as the uncertainty metric, randomly deferring instances that fall into the null set and assigning the remaining instances to the ML model. This approach aligns with the separated rejector framework illustrated in Figure 2.1. The coverage level, α , is selected so that the number of null set predictions matches the expert’s capacity.

Random assignment: in this baseline, experts handle a randomly selected subset of the test instances that matches their capacity.

6.9 System Training

6.9.1 Density-Based Conformal Prediction

For density-based conformal prediction, we divide our data into a *proper training* set and a *calibration* set. The proper training set consists of alert data from months 4 to 6, while the calibration set comprises alert data from month 7.

To obtain a class-conditional density estimate $p(x|y)$ for each label $y \in \mathcal{Y}$, we need to encode our tabular data, which includes numerical, binary, and categorical features, into a format suitable for density estimation. Following the approach by Messoudi et al. [62], we train a multi-layer perceptron (MLP) and use its penultimate dense layer as a feature extractor. These feature vectors are subsequently used for the conformal prediction step, in order to estimate the density. Details on the training of the MLP are provided in Appendix A.2.

Feature vectors are extracted from the penultimate dense layer, which has a size of 50, for all alert instances from months 4 to 8. We then use these feature vectors to perform Gaussian kernel density estimation (KDE) with a bandwidth of 0.5. As described in Algorithm 4.1, KDE is fit on the proper training set independently for each class. The empirical $1 - \alpha$ quantile \hat{q}_y is then determined using the calibration data. Prediction sets are subsequently created as outlined in Algorithm 4.2.

6.9.2 Density-Softmax

The feature vectors used for the RealNVP models are extracted from the penultimate layer of the MLP described in the previous section. These feature vectors serve as the input to the RealNVP models, which estimate the likelihood of each instance. A unique density model is trained for each expert and each data availability scenario, using solely the subset of training and validation data labeled by that expert. Details on the training of the RealNVP models are provided in Appendix A. These expert-specific density models are then integrated with the binary classifiers that model each expert’s correctness, as described in Chapter 3.

6.9.3 Learning to Defer

To train the main ML classifier h and the models that predict each expert’s correctness, we again employ LightGBM models [48]. These models are trained on the alerts raised from months 4 to 6 and validated on the alert data from month 7. For the ML classifier, we use the true labels as the target variable. In contrast, for the expert models, we use a label of 1 if the expert provides a correct prediction on the training instance and a label of 0 if the expert’s prediction is incorrect. Both models are optimized to minimize the weighted log-loss, as detailed in Section 5.1. Training details for the ML classifier h and for the expert models are provided in Appendix A.

Chapter 7

Results

7.1 L2D System Performance and Calibration

7.1.1 Classifier

We begin by evaluating the classifier h regarding predictive performance and calibration. Predictive performance is measured using the area under the receiver operating characteristic curve (ROC-AUC), and calibration is assessed via the ECE [36, 67], with both metrics computed under the re-weighted data distribution as the model was trained under said distribution (Section 5.1). The ECE quantifies the discrepancy between predicted probabilities and actual outcomes by partitioning predictions into M bins and computing the weighted average of the absolute difference between accuracy and confidence in each bin,

$$\text{ECE} = \sum_{m=1}^M \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|, \quad (7.1)$$

where B_m is the set of predictions falling into the m -th bin, $|B_m|$ is the number of predictions in that bin, n is the total number of predictions, $\text{acc}(B_m)$ is the accuracy of predictions in bin B_m , and $\text{conf}(B_m)$ is the average predicted confidence in that bin. In our evaluation, we use quantile bins to partition the predictions. Unlike fixed-width bins, quantile bins are adaptive and ensure that each bin contains approximately the same number of predictions.

Figure 7.1 shows the ROC and the calibration curves, along with ROC-AUC and ECE values. The classifier achieves a ROC-AUC of 0.68 and an ECE of 5.6% on the original test data. As expected, performance declines significantly, both in predictive capability and calibration, as the noise level rises. This highlights the classifier's reduced ability to generate calibrated estimates under noisy conditions and underscores the need for incorporating uncertainty measures in these dynamic environments.

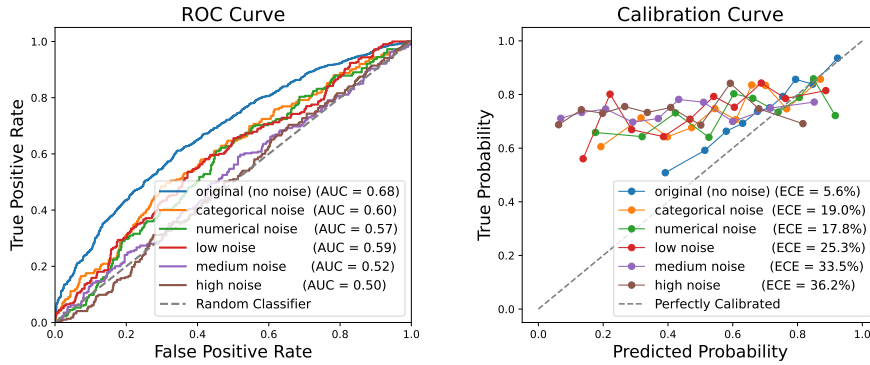


Figure 7.1: ROC and calibration curves for classifier h . Values are shown for the original (non-noisy) test data and the noisy subset of the test set in the five different scenarios. The classifier h is no better than a random predictor in the high noise setting.

7.1.2 Expert Decision Modeling

We now assess the predictive performance and calibration of the models that predict expert correctness by calculating their ROC-AUC and ECE values. We test these models' calibration and predictive performance under several different amounts of human-labeled data, thus simulating real-world challenges in obtaining expert annotated data. As in the previous section, these metrics are computed under the re-weighted data distribution, as detailed in Section 5.1.

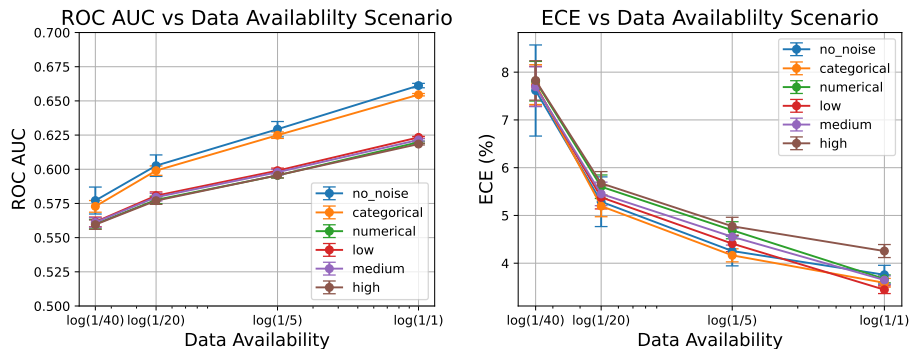


Figure 7.2: Mean ROC-AUC and ECE for estimates of $\mathbb{P}(y_i = m_{j,i})$, varying the amount of data the models are trained on. Values are calculated for each model $g_{\perp,j}$ and averaged, with error bars representing 95% confidence intervals for the mean, accounting for the randomness in the injection of noise and selection of training data. The data availability axis is shown on a logarithmic scale to improve visualization.

Figure 7.2 illustrates that training these models on smaller subsets of the training data negatively impacts both performance and calibration. It is also evident that noise on categorical features has little effect on performance, whereas noise on numerical features significantly degrades performance, regardless of its magnitude. While calibration remains relatively consistent across settings, the high noise scenario consistently exhibits higher ECE across all data availability scenarios.

7.2 MLP and Feature Extraction

The goal of the MLP model is to encode the BAF dataset's features into a numerical vector so that we can apply density estimation. While the MLP's performance is important to obtain better embeddings from the last layer, we also evaluate it for comparison with the classifier h . Since both models are trained and validated on the same data, the MLP offers a relevant benchmark. When tested on alert data from month 8, the MLP achieves a ROC-AUC of 0.63 and an ECE of 6.16%, achieving a slightly worse performance and calibration than classifier h . This is most likely due to model architecture, further validating our choice to use a LightGBM model.

To visualize the encoded features and evaluate how noisy and non-noisy instances, as well as fraud and non-fraud instances, differentiate from each other in the test set, we use the popular t-SNE (t-distributed stochastic neighbor embedding) method [87]. The results are shown in Figure 7.3. Panel (a) shows the data without added noise, where fraud and non-fraud instances overlap considerably, reflecting the fact that all of these cases have been flagged by the alert model as potential fraud. Subsequent panels demonstrate how the noisy data seems to become more concentrated in certain regions of the feature space as the noise level rises, thus we expect these to be more easily identifiable by conformal prediction and density methods in our experiments.

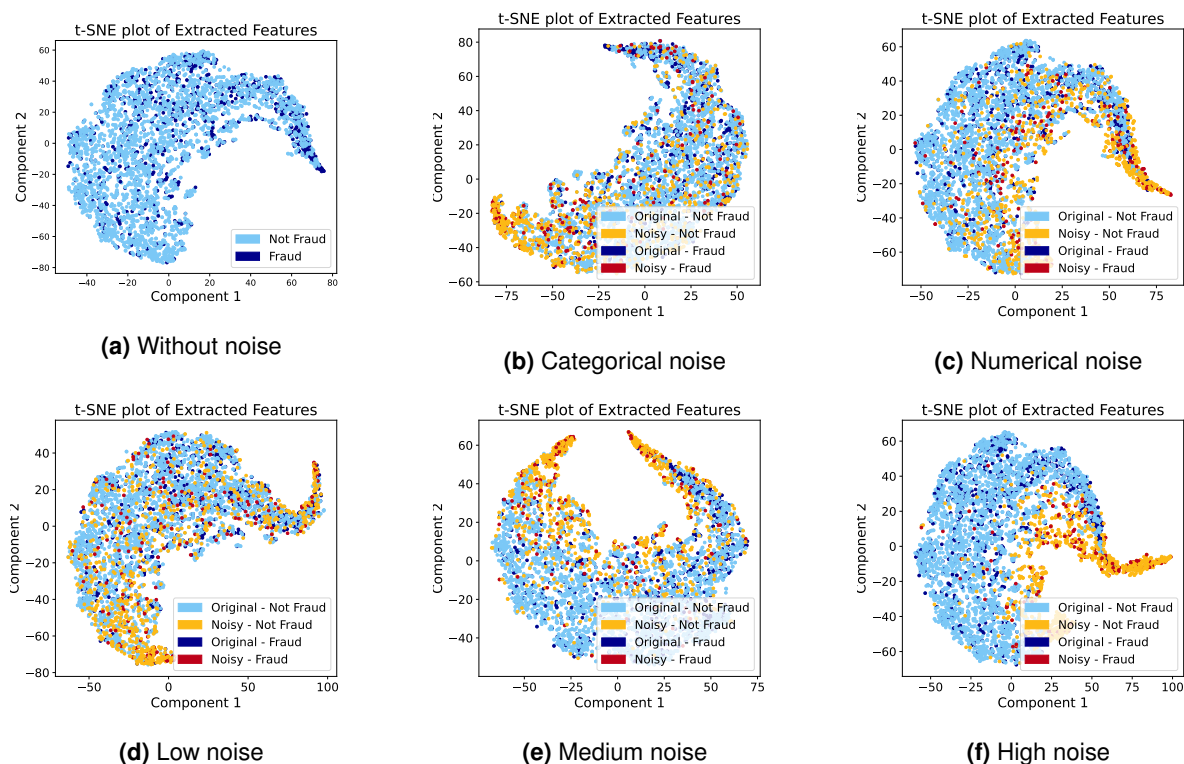


Figure 7.3: t-SNE visualization of the MLP encoded features of the test set under different noise conditions.

7.3 Density-Softmax

Figure 7.4 displays the density scores obtained from the RealNVP models, visualized in the same low-dimensional space as in Figure 7.3 using t-distributed Stochastic Neighbor Embedding (t-SNE). Comparing these two figures, we observe clear patterns regarding the relationship between noise levels and density scores. In cases with low levels of noise, the lower density scores are scattered throughout the space, reflecting how the noisy instances are similarly distributed. As the noise magnitude increases, we see a sharper contrast: regions without noise tend to exhibit density scores close to 1, while areas corresponding to OOD data display density scores much closer to 0. In contrast with the low noise scenarios, we also see a much more defined border between noisy and in-distribution samples.

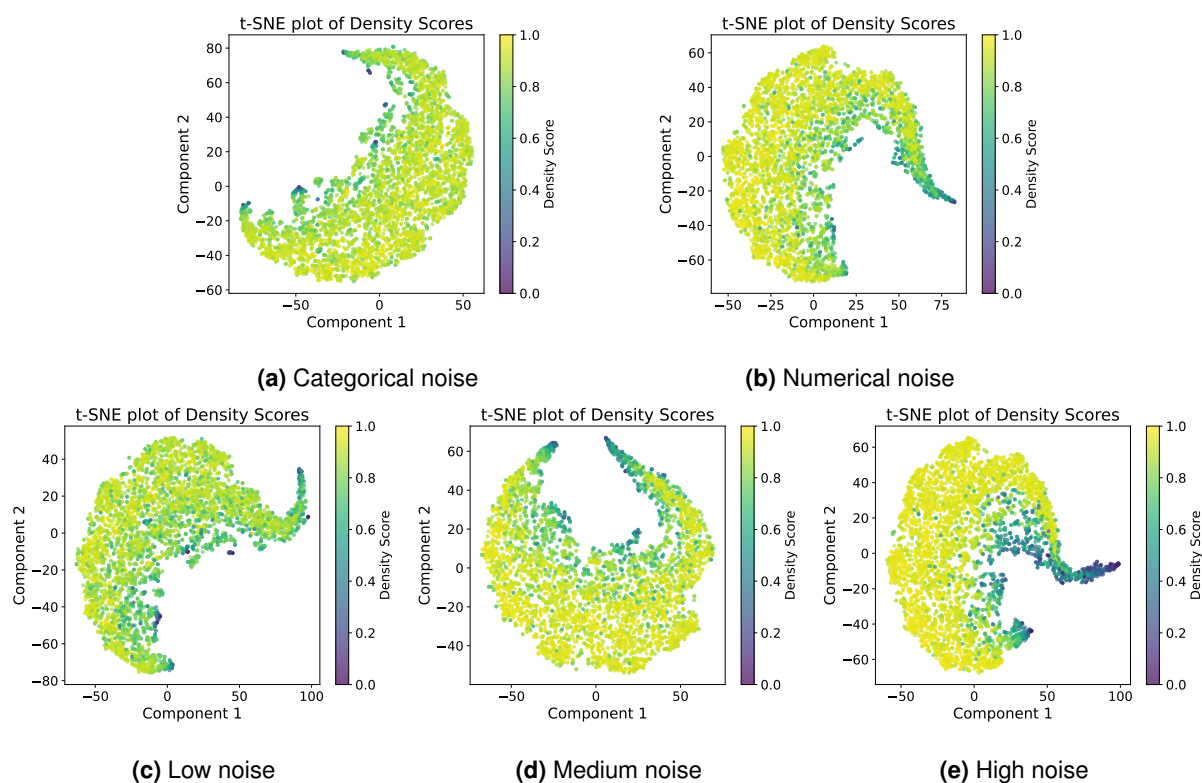


Figure 7.4: t-SNE visualization of the density scores on the test set under different noise conditions. The RealNVP model used to obtain these scores was trained on the whole training set (months 4 to 7).

As explained in Chapter 3, these density scores are used to downweigh the logits of the classifier h and the binary classifiers that predict the correctness of the experts. Figure 7.5 illustrates the effectiveness of this approach in the high noise setting. In the left image, the probabilities given by the classifier h can be arbitrarily high or low for noisy instances. However, on the right, with our density-softmax approach, the probabilities for noisy instances are more uniformly distributed and closer to random, reflecting uncertainty awareness.

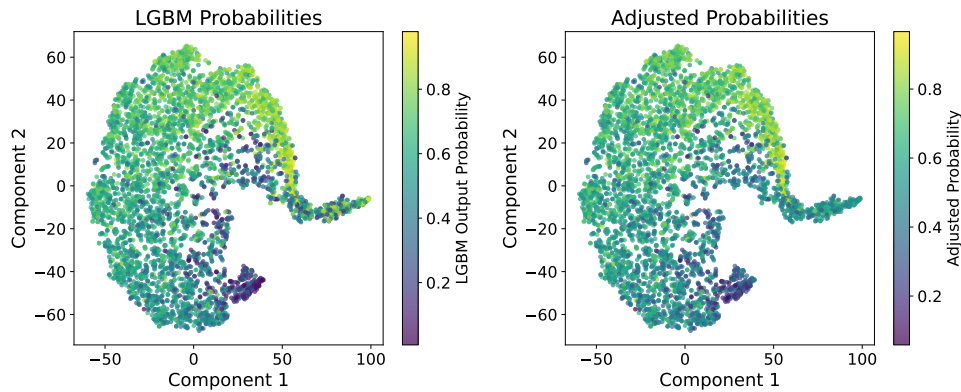


Figure 7.5: Comparison of the probabilities given by the classifier h . On the left, the LGBM classifier h 's probabilities are shown without adjustment from the density-softmax method, while on the right, the probabilities are adjusted using the density-softmax approach to downweigh the logits in regions of low density.

For the classifier h , improvements in performance are subtle, as the ROC-AUC values, originally shown in Figure 7.1, change by at most 0.01 with the introduction of distance-awareness (Figure 7.6). In noisy subsets, ROC-AUC consistently increases, while in the subset without noise, it drops slightly to 0.67. Regarding calibration, ECE decreases for noisy subsets across all scenarios. This reduction is modest in the categorical, numerical, and low noise scenarios but more pronounced with higher noise magnitudes, showing a 6% decrease in the high noise setting. Conversely, ECE increases by 2% for the subset of test data without noise. The density model naturally gives lower scores to instances in the fringe of the feature space, even though that does not mean that said region constitutes an unseen region where the logits must be downweighted, thus leading to this slight deterioration in calibration and performance.

For the binary classifiers predicting expert correctness, the impact on performance and calibration is tremendous. Across the entire test set for all noisy scenarios, ROC-AUC values rise to the same as the no noise setting across different data availability settings, a huge gain in performance. This contrasts with results observed without distance-aware models (Figure 7.2). Additionally, ECE improves across all scenarios, especially in cases with 1/5 and 1/20 of data availability. Notably, there is no significant difference in ECE between using the full training set versus 1/5 of the training data, unlike the disparities seen in Figure 7.2.

7.4 Density-Based Conformal Prediction

7.4.1 Kernel Density Estimation

As with the RealNVP models, KDE is applied to the features extracted from the MLP described in Section 7.2. To evaluate the performance of KDE, we use Bayes' theorem to compute the posteriors and

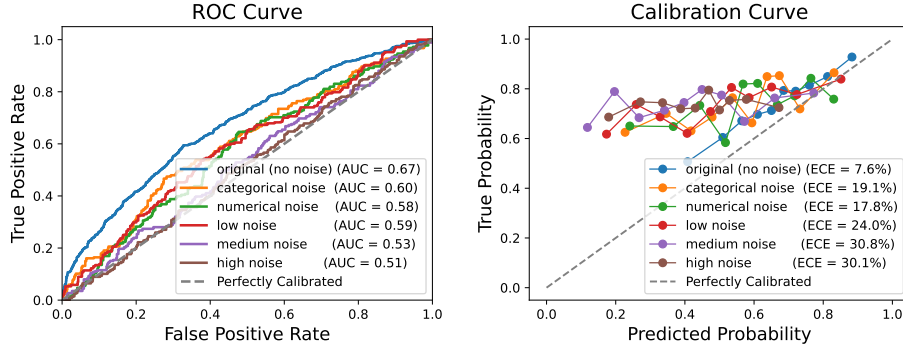


Figure 7.6: ROC and calibration curves for classifier h in the density-softmax approach. Values are shown for the original (non-noisy) test data and the noisy subset of the test set in the five different scenarios.

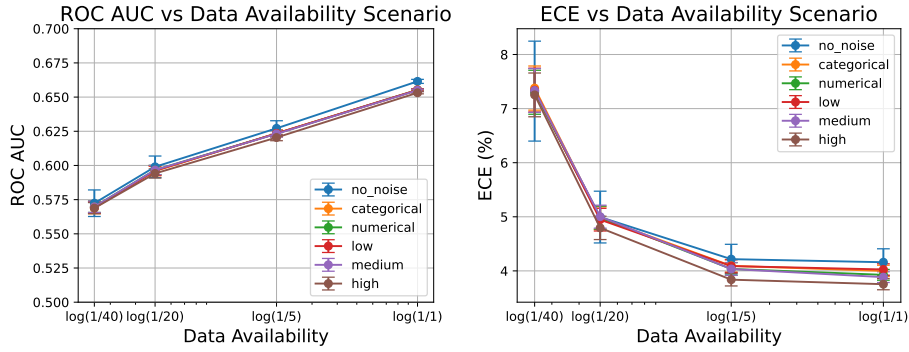


Figure 7.7: Mean ROC-AUC and ECE for estimates of $\mathbb{P}(y_i = m_{j,i})$ in the density-softmax approach, varying the amount of data the models are trained on. Values are calculated for each distance-aware model and averaged, with error bars representing 95% confidence intervals for the mean, accounting for the randomness in the injection of noise and selection of training data. The data availability axis is shown on a logarithmic scale to improve visualization.

assess the ROC-AUC on the test set. While this evaluation does not directly impact the performance of our system, it provides insight into the quality of our density estimation and serves as another baseline to compare the classifier h to, allowing us to assess whether the LightGBM algorithm provides the best performance.

The priors $\hat{p}(y)$ for each class are obtained from the reweighted distribution described in Section 5.1 as

$$\tilde{D}(x, y) = \frac{c}{\mathbb{E}_{c \sim D}[c]} D(x, y, c). \quad (7.2)$$

Using Bayes' theorem, we compute the posterior probabilities as

$$\hat{p}(y = 0|x) = \frac{\hat{p}(x|y = 0)\hat{p}(y = 0)}{\hat{p}(x|y = 0)\hat{p}(y = 0) + \hat{p}(x|y = 1)\hat{p}(y = 1)}, \quad (7.3)$$

$$\hat{p}(y = 1|x) = 1 - \hat{p}(y = 0|x) = \frac{\hat{p}(x|y = 1)\hat{p}(y = 1)}{\hat{p}(x|y = 0)\hat{p}(y = 0) + \hat{p}(x|y = 1)\hat{p}(y = 1)}, \quad (7.4)$$

and use them to calculate the ROC-AUC and assess performance. The ROC-AUC for this density model is 0.63, aligning with the performance of the MLP. This is expected, as we don't expect KDE to outperform the MLP from which the features are extracted. Nevertheless, it is another baseline the classifier h outperforms and an indicator of good performance.

7.4.2 Conformal Coverage

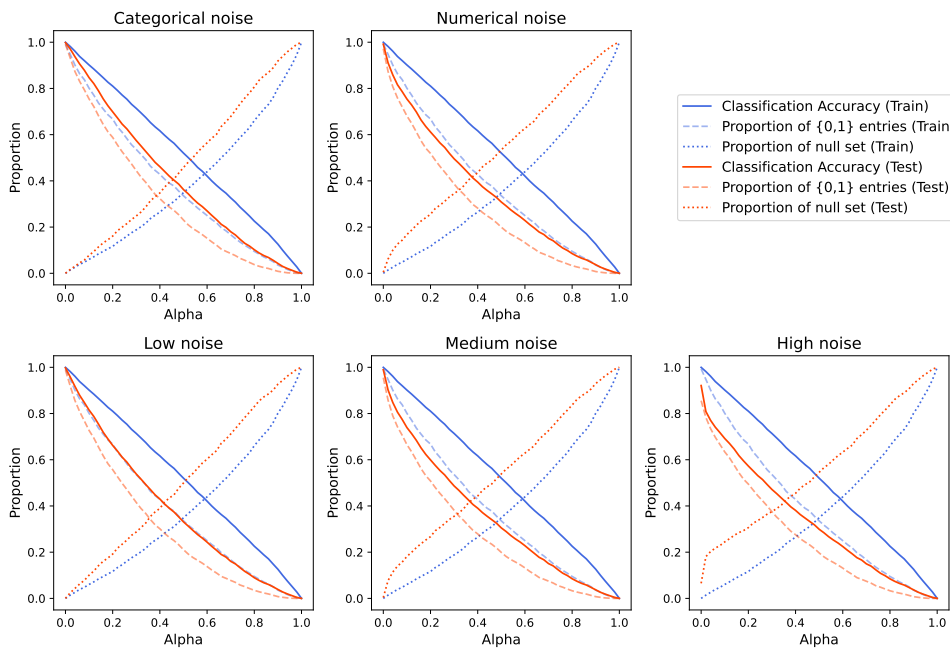


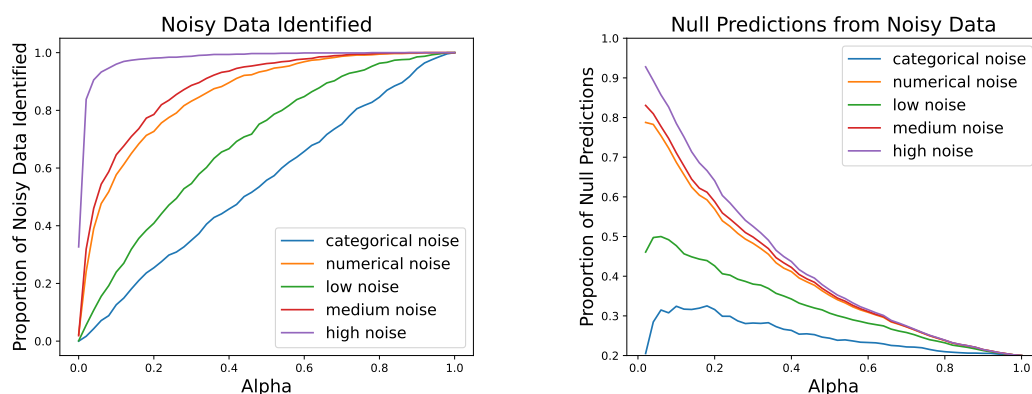
Figure 7.8: Conformal coverage for training and test data under different noise conditions. As the noise magnitude increases, more test instances are predicted as the empty set for the same value of α . This not only indicates that we are in the presence of distribution shift but also showcases the method's ability to identify OOD instances.

As detailed in Algorithm 4.1 and Algorithm 4.2, density scores obtained from the KDE model are used as conformity scores for our conformal prediction approach. In this method, the confidence level α is predefined, and the number of classes in the prediction sets is adjusted to meet the desired accuracy level. Smaller values of α result in higher accuracy but may predict both labels for a larger proportion of observations. Conversely, higher values of α result in more null set predictions, where the model refrains from making any prediction.

Figure 7.8 demonstrates how varying coverage levels α affect prediction sets across different test scenarios. As expected, in the training data, classification accuracy is roughly $1 - \alpha$, consistent with the design of conformal prediction, while the proportion of null predictions is close to α . This suggests that

for all values of α , errors are mainly attributed to a high number of null set predictions, with very few non-empty sets that do not contain the true label.

However, at test time, the accuracy drops below $1 - \alpha$, signaling a violation of the exchangeability assumption in conformal prediction due to distribution shift. Additionally, the higher number of null set predictions at test time suggests that the method is able to identify instances that deviate from the training distribution. As the noise level increases, more instances are classified into the null set for the same level α , reinforcing this observation. This trend is evidenced further in Figure 7.9, which demonstrates the model’s ability to detect OOD data as the noise increases.



(a) Proportion of noisy data predicted as the null set for each coverage level, across various noise conditions.

(b) Proportion of null predictions originating from noisy data for each coverage level, across various noise conditions.

Figure 7.9: Demonstrating the effectiveness of the conformal prediction method in detecting OOD data by outputting null set predictions. In (b), note that the y-axis starts at 0.2 as the test sets are designed to have 20% noisy data. As α approaches 1, all instances are predicted as the null set, causing the proportion to converge to 0.2.

In Figure 7.9 (a), the proportion of noisy data predicted as the null set increases as α rises. For lower noise magnitudes, this identification process is more gradual, meaning that the risk of incorrectly classifying in distribution samples as OOD samples is higher for lower noise settings. For higher noise magnitudes, the detection becomes more pronounced, as nearly all noisy data is identified even for lower α values in the higher noise settings. Figure 7.9 (b) shows that as noise levels increase, a larger proportion of null set predictions come from noisy data, particularly for lower values of α . This confirms the model’s ability to correctly identify uncertain predictions.

7.5 Assignment System Performance and Calibration

Having trained the classifier h , the OvA binary classifiers that model expert correctness, and the uncertainty estimation methods required for our system implementations, we now evaluate the perfor-

mance of our assignment systems, i.e. conformal prediction based and density-softmax based. In this section, we compare our proposed assignment systems against the baselines defined in Section 6.8 across various testing scenarios.

Each testing scenario is determined by four variables, as described in Section 6.7. These variables are noise type/level, number of experts, deferral rate, and data availability. Due to the large number of testing scenarios (500 to be precise), we showcase a summarized view of the results that represent the general pattern seen in our experiments, while in Appendix B, we present another 240 scenarios that showcase a larger and representative subset of the results, covering different combinations of experts, deferral rates, and data availability.

The results summarized in Table 7.1 are derived by fixing the number of experts at five, the data availability at one-fifth, and the deferral rate at 40%. Each setting is run five times with different random seeds. For each run, five experts are randomly sampled from a pool of 15, and different random seeds are used for both the training data selection and the noise addition. The table presents the expected misclassification cost (Equation 6.1) per 100 instances across five runs for each noise setting, comparing five deferral strategies: the density-softmax approach (DS), the conformal prediction approach (CP), and the baseline methods—L2D, rejection learning (RL), and random assignment. The results are averaged over the five runs, with 95% confidence intervals for the mean.

Table 7.1: Comparison of deferral strategies under different noise settings. The reported values are the expected misclassification cost per 100 instances with 95% confidence intervals across five runs. For each noise setting, the best-performing strategy is in bold, and the second-best is underlined.

Setting	Deferral Strategy				
	Noise	CP	DS	L2D	RL
numerical	<u>5.06</u> ± 0.28	4.93 ± 0.39	5.07 ± 0.46	5.36 ± 0.32	5.25 ± 0.06
categoryal	<u>4.81</u> ± 0.13	4.80 ± 0.07	5.07 ± 0.21	5.23 ± 0.10	5.31 ± 0.23
low	<u>5.18</u> ± 0.13	5.17 ± 0.10	5.55 ± 0.27	5.39 ± 0.22	5.70 ± 0.19
medium	5.25 ± 0.14	<u>5.41</u> ± 0.17	5.84 ± 0.15	5.71 ± 0.22	6.04 ± 0.18
high	4.83 ± 0.21	<u>5.48</u> ± 0.09	5.89 ± 0.18	5.75 ± 0.12	6.08 ± 0.07

Both proposed assignment methods consistently outperform the baselines across all noise settings. In low-noise scenarios (numerical, categoryal, and low noise), DS and CP perform similarly, with DS slightly outperforming CP. However, as noise magnitude increases (medium and high noise), CP becomes the superior method. This is due to the fact that as the noise levels get higher, having a hard cutoff and just sending OOD samples to humans makes results better than having a sort of model score fade caused by the density model, where OOD instances may sometimes still be assigned to the classifier. We show this in Figure 7.10 and discuss it in more detail below.

The baseline strategies—L2D, rejection learning, and random assignment—show poorer performance overall. The random strategy consistently yields the highest misclassification costs. Interestingly,

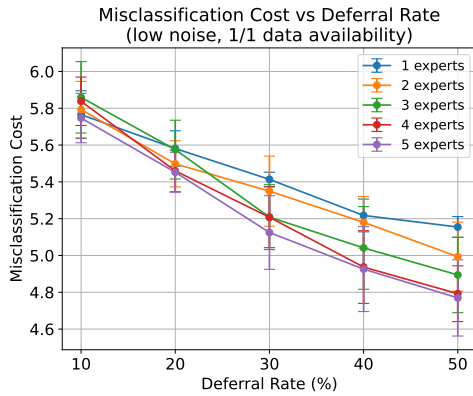
in the extended results provided in Appendix B, L2D performs better than rejection learning in low-noise settings (categorical, numerical, and low noise), while rejection learning outperforms L2D in higher-noise conditions (medium and high noise). Among the 72 scenarios shown with lower noise magnitude (low, categorical and numerical settings) and a high deferral rate (40% or 50%), L2D outperforms RL 94% of the time. Conversely, RL outperforms L2D, 78% of the time in the medium and high noise setting, regardless of deferral rate. These results highlight L2D’s effectiveness in scenarios where it can reliably model decision-maker correctness and where data drift at test time is minimal. In contrast, rejection learning proves advantageous when data that deviates from the training distribution is readily identifiable and can be deferred to human experts. This also justifies the conformal prediction method’s advantage in high noise settings.

In the following analysis, we evaluate how the misclassification cost changes as we vary one variable while keeping the others fixed. Figure 7.10 presents the expected misclassification cost per 100 instances plotted against the deferral rate, with each line representing a different number of experts.

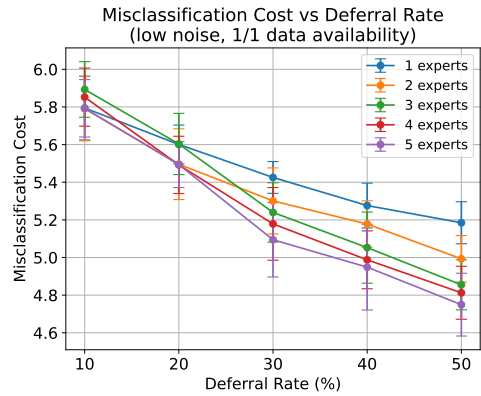
There are three main takeaways from these plots. First, as expected, misclassification cost decreases as experts are tasked with predicting a larger fraction of the test set. This result stems from how our experiments are set up: on the test set, the synthetic experts perform better than the main classifier h on average (Section 6.5).

Second, we observe that as the deferral rate increases, the performance gap between settings with different numbers of experts becomes more pronounced. At lower deferral rates, the systems prioritize assigning noisy and OOD instances to human experts, as these are the instances where the most significant performance improvements can be achieved given the experts’ limited capacity. This assignment is primarily based on the experts’ past performance, as described in Chapters 3 and 4, and does not account for the varying strengths of experts across different regions of the feature space. However, when the deferral rate exceeds 20% (recall that our test sets contain 20% noisy data), experts are also tasked with handling in-distribution instances. In this case, having a larger team of experts becomes a clear advantage, as it enables better assignment of instances through L2D, matching each instance to the expert most likely to make a correct prediction based on their expertise in specific regions of the feature space.

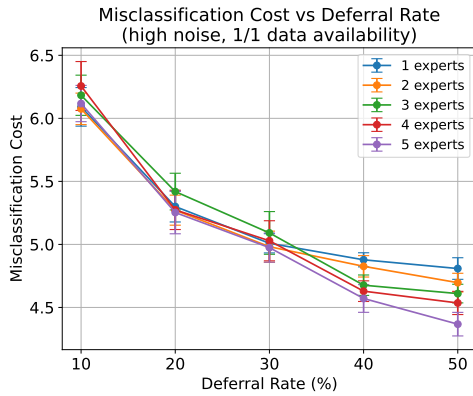
Finally, the plots for the high-noise setting illustrate why the conformal prediction method outperforms the density-softmax approach in settings with greater noise magnitude. In Figure 7.10 (c), we see a significant reduction in misclassification cost up to a deferral rate of 20%, followed by a slower rate of improvement beyond that. In contrast, Figure 7.10 (d) shows a slower but steady reduction in misclassification cost across all deferral rates in the density-softmax approach. This difference stems from how each method handles OOD instances: conformal prediction assigns instances labeled as the empty set to human experts, assuming the classifier h has a zero probability of correctness on these instances.



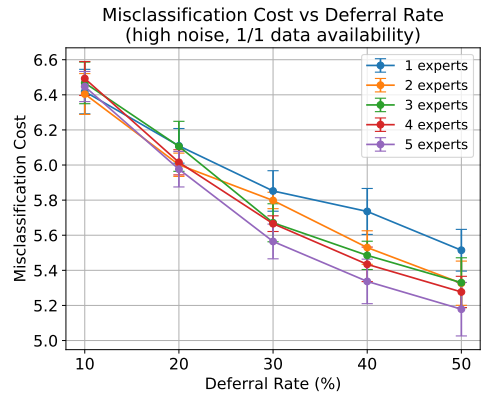
(a) Conformal Prediction - Low Noise



(b) Density-Softmax - Low Noise



(c) Conformal Prediction - High Noise



(d) Density-Softmax - High Noise

Figure 7.10: Misclassification Cost vs Deferral Rate. Each line represents a different scenario where we change the number of experts in the system. These results were obtained for the full data availability scenario. Each plot represents the results for one of our proposed assignment systems for either the high or the low noise setting.

In contrast, the density-softmax approach assigns a minimum probability of correctness of 0.5 to the classifier h , meaning that some OOD instances may still be deferred to the classifier rather than experts. This occurs if there are in-distribution instances in which the experts are significantly more accurate than the classifier. In such cases, the assignment system from the density-softmax approach prioritizes these instances for expert prediction over OOD instances, leading to a less pronounced improvement in misclassification cost for low deferral rates.

To address this limitation of the density-softmax approach, a potential improvement would be to allow the classifier h 's probability of correctness to approach zero when high epistemic uncertainty is detected. This adjustment would make the approach more comparable to a smoothed version of the conformal prediction method, as it would exclude the classifier from decision-making on OOD instances, thereby applying a form of rejection learning. Exploring this enhancement presents a promising direction for future work.

We previously assessed deferral quality using the expected misclassification cost per 100 instances. Now, we turn our focus to evaluating calibration by measuring the ECE of our proposed assignment systems compared to the L2D baseline. This comparison is feasible only with the L2D baseline, as deferral in the random and rejection learning baselines is not driven by any ML model.

For the density-softmax approach, the ECE comparison with the L2D baseline is straightforward. Both are L2D systems, with the primary distinction being that density-softmax incorporates distance-aware models that provide less confident (closer to random) probabilities on OOD data (Figure 7.5). Thus, we calculate the ECE by comparing the true label of each instance with the probability of correctness output for the assigned decision-maker in each method. The results, shown in Table 7.2, were derived in a similar manner to the performance results in Table 7.1; specifically, we conducted five runs for each setting, sampling five experts from a pool of 15, and applying random seeds for noise addition and training data selection. For consistency, we maintained the data availability scenario at 1/5 (i.e., one expert prediction per training instance). However, here we varied the deferral rate between 20% and 40%, as we observed relevant differences across this range.

Table 7.2: Comparison of the Density-Softmax and Learning to Defer strategies under different noise and deferral rate settings. The reported values are the expected calibration error (%) with 95% confidence intervals across five runs.

Setting		Deferral Strategy	
Noise	DR	DS	L2D
categorical	20	2.78 \pm 0.31	4.56 \pm 1.35
categorical	40	3.68 \pm 0.79	3.73 \pm 0.81
numerical	20	3.24 \pm 0.74	4.49 \pm 1.29
numerical	40	4.19 \pm 0.62	4.24 \pm 0.66
low	20	3.77 \pm 0.65	5.10 \pm 1.25
low	40	4.41 \pm 0.84	3.79 \pm 0.49
medium	20	3.64 \pm 0.43	5.04 \pm 1.29
medium	40	4.75 \pm 0.56	4.16 \pm 0.61
high	20	3.93 \pm 0.57	5.39 \pm 1.14
high	40	4.70 \pm 1.18	4.49 \pm 0.53

At lower deferral rates, the density-softmax approach consistently achieves better-calibrated probabilities than the L2D baseline, as reflected by lower ECE values across all settings. However, at higher deferral rates, the calibration advantage is less pronounced. Results indicate that as the deferral rate increases, calibration for the density-softmax method slightly worsens, while it improves for the L2D baseline. This trend can be attributed to a slight calibration deterioration for classifier h on the non-noisy subset of the test data when using distance-aware models, where the ECE increases by approximately 2%. However, as detailed in Section 7.3, employing distance-aware models enhances calibration on the noisy subset of the test data and significantly improves calibration for the expert models in the entire test set, consistently across all scenarios.

For the conformal prediction approach, we cannot evaluate calibration in the same straightforward way as with the density-softmax approach. In our conformal method, instances predicted as the null set are handled via a rejection learning strategy, so we focus on calibration for instances that are not flagged as null sets. These instances are assigned through the same L2D approach used in the baseline, as described in Section 6.8. Therefore, our comparison is not directly between different methods, but rather an evaluation of whether restricting L2D assignment to a smaller subset of in-distribution data (non-null set predictions) enhances calibration. By excluding null set predictions, we aim to improve calibration by concentrating the L2D assignment on instances where the models provide more reliable estimates of correctness.

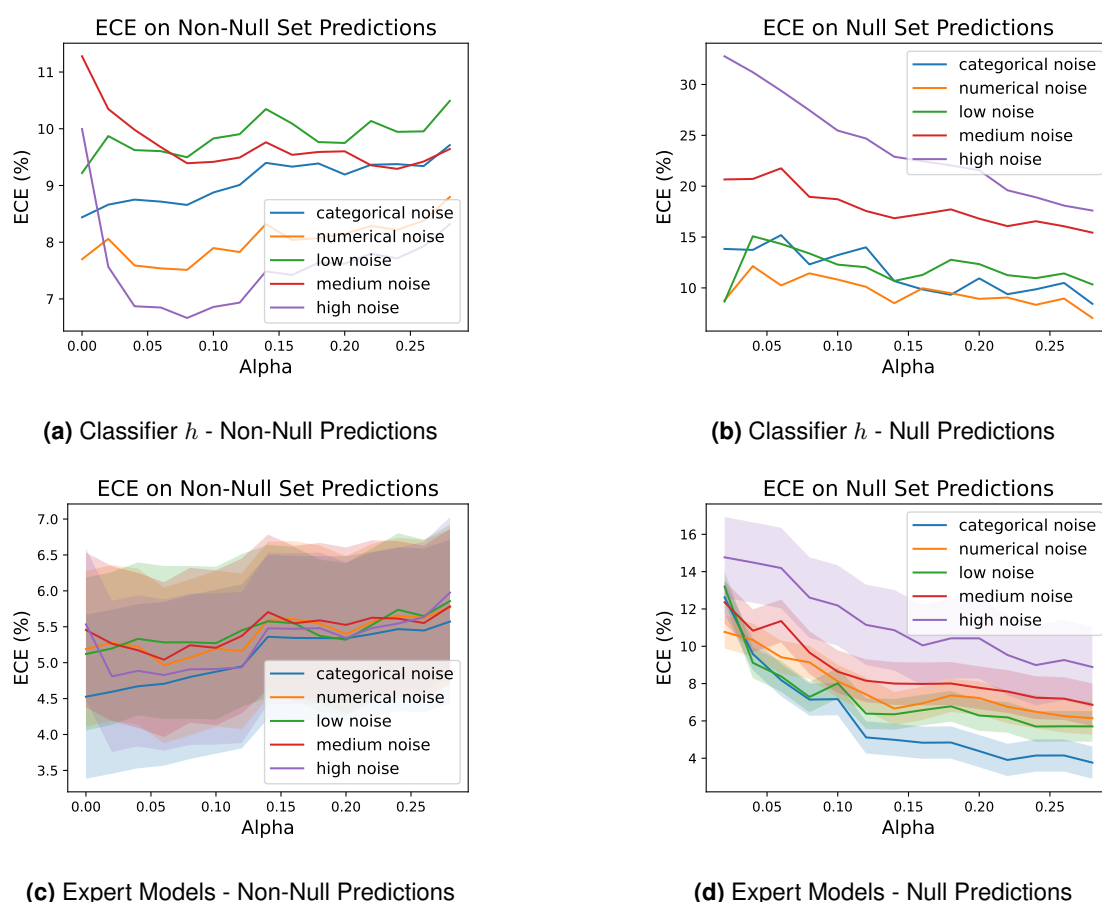


Figure 7.11: Expected Calibration Error vs Coverage Level α . Each line represents a different noise scenario. As we increase the coverage level the number of non-null set predictions decreases and the number of null set predictions increases, affecting the ECE values. For the expert models, the ECE is averaged across all models, shown in the plots with 95% confidence intervals.

In Figure 7.11, we plot the ECE for instances not predicted as the null set (left column) and for those predicted as the null set (right column) across various coverage levels α . This comparison is shown for both classifier h and the expert models, allowing us to observe how calibration changes as α varies.

As α increases, more instances are predicted as null sets, reducing the number of non-null predictions, as seen in Section 5.2. In settings with medium and high noise, calibration on the non-null predictions improves considerably for classifier h and moderately for the expert models at low α values, indicating better-calibrated estimates as we start to detect and exclude more OOD data. Conversely, the ECE on the null set predictions is initially high but decreases across all noise settings as α rises. This is expected: when α is close to 0, only the most extreme OOD data is excluded, resulting in few null set predictions, all of which have poor calibration. As α continues to rise, the threshold for detecting OOD data becomes less conservative, leading to more instances being categorized as null sets and improving calibration as these instances increasingly resemble the training data distribution. However, at higher α levels, fewer instances are flagged as non-null sets, shifting the distribution of the data remaining for assignment through L2D. This distribution change results in poorer calibration on the non-null predictions.

These observations motivate the need for an optimal coverage level α that effectively detects OOD data while improving calibration. The plots in Figure 7.11 demonstrate why jointly optimizing the coverage level α and the assignment matrix is necessary, as described in Section 5.1. Selecting an appropriate α allows the system to detect as much OOD data as possible, while avoiding excessive null set predictions that would shift the distribution of the remaining non-null predictions and lead to poorer calibration. As shown in Table B.6 from Appendix B, the number of null set predictions varies considerably with different noise settings, aligning with earlier observations in Section 7.4.2. Furthermore, the optimization process consistently selects α values between 0 and 0.05—ranges that balance OOD detection with calibration stability, as confirmed by the ECE results discussed in this section. For the medium and high noise settings, these values of α allow for significant improvement in calibration, while for lower noise magnitudes they do not shift the distribution on which L2D occurs, allowing calibration to not deteriorate.

Chapter 8

Conclusions and Future Work

Machine learning (ML) models are increasingly employed in high-stakes decision-making settings due to their scalability, speed, and impressive performance. However, these models often lack transparency and adaptability in evolving environments, limiting their applicability in high-risk dynamic contexts. Human decision-makers complement artificial intelligence (AI) by offering strengths that counteract these weaknesses, including the ability to adapt to novel situations, access to additional information, and an ability to reason through causal relationships. A significant area of research within human-AI collaboration has thus focused on determining which decision-maker should handle a given instance. The learning to defer (L2D) framework enhances traditional rejection learning by modeling human decision-making behavior, allowing for a more effective allocation of instances. However, L2D still faces several limitations that challenge its implementation in real-world settings, particularly concerning work capacity constraints, limited data availability, and adapting to distribution shifts in dynamic environments. To address these challenges, we proposed two novel methods that improve the robustness of human-AI collaboration in these complex scenarios.

The first approach, based on distance-aware models, leverages density estimation to create a L2D system more attuned to distribution shifts in the data. In this approach, each classifier employed by the L2D system is combined with a density function and thus yields lower confidence scores on instances far away from the training distribution. Our results showed that employing distance-aware models improved the misclassification cost and the expected calibration error in several scenarios where data drift occurred at test time. Additionally, this method achieved more consistent performance and calibration results in low data availability scenarios when compared with the L2D baseline. This indicates that density-aware adjustments help the model to better detect instances that differ from the training data, resulting in improved calibration and reliability in the model's estimates.

The second approach employs density-based conformal prediction to balance rejection learning with L2D. This method identifies instances that do not conform to the training data distribution and defers

them directly to human experts. By restricting L2D to in-distribution instances, where the model can make more reliable predictions, and deferring the remaining instances to human experts, we saw a further reduction in misclassification cost compared to both our density-softmax method and traditional L2D systems. Additionally, our optimization of the coverage level led to an effective balance of rejection learning and L2D: in high-noise settings, around 17% of instances were identified as out-of-distribution and deferred to human experts through a rejection learning strategy, aligning closely with our introduction of roughly 20% noisy instances at test time. Importantly, we demonstrated, in this approach, that the majority of instances assigned through rejection learning originated from noisy data, demonstrating the system’s ability to detect and appropriately defer these challenging instances. Overall, the conformal prediction system achieved the best performance in the higher noise settings, with similar performance to the density-softmax approach in the remaining testing scenarios.

8.1 Future Work

A promising direction for future work is to modify the density-softmax approach by allowing the main classifier’s probability of correctness to approach zero under high epistemic uncertainty. This adjustment would align it more closely with the conformal prediction method by excluding the classifier from decision-making on OOD instances, thereby incorporating a form of rejection learning. This would allow the system to defer more OOD instances to human experts, especially in high-noise settings, potentially achieving reductions in the misclassification cost across all deferral rates. Moreover, prioritizing human predictions on OOD instances may be required in high-stakes settings where accountability and explainability are critical.

Another area to explore is the system’s performance across various cost ratios for FP and FN errors, as this work focuses on a single cost structure. Extending the approach to multiple cost structures would provide insights into its flexibility and robustness in diverse cost-sensitive environments. Additionally, refining the selection of the λ parameter used to express the Neyman-Pearson criterion could improve adaptability. Currently, this parameter is defined at training time to ensure that the training process is cost-aware. However, this may limit the system’s responsiveness to evolving cost structures, such as shifts in class prevalence. Adjusting λ dynamically as the system gathers more information about its environment could make the model more responsive to changes in prevalence or other relevant cost changes.

Finally, we aim to test the integration of fairness incentives within both assignment systems. Given that human decision-makers and ML models may exhibit biases, it is essential to ensure that decisions in high-stakes scenarios are made equitably, fostering a system that prioritizes fairness alongside accuracy and accountability.

Bibliography

- [1] Heritage health prize.
- [2] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) (text with eea relevance), May 2016.
- [3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2623–2631. ACM, 2019.
- [4] Saar Alon-Barkat and Madalina Busuioc. Human-ai interactions in public sector decision making: “automation bias” and “selective adherence” to algorithmic advice. *Journal of Public Administration Research and Theory*, 33(1):153–169, 2023.
- [5] Jean V. Alves, Diogo Leitão, Sérgio M. Jesus, Marco O. P. Sampaio, Javier Liébana, Pedro Saleiro, Mário A. T. Figueiredo, and Pedro Bizarro. Cost-sensitive learning to defer to multiple experts with workload constraints. *Trans. Mach. Learn. Res.*, 2024, 2024.
- [6] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *CoRR*, abs/2107.07511, 2021.
- [7] John O. Awoyemi, Adebayo O. Adetunmbi, and Samuel A. Oluwadare. Credit card fraud detection using machine learning techniques: A comparative analysis. In *2017 International Conference on Computing Networking and Informatics (ICCNi)*, pages 1–9, 2017.
- [8] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7499–7519, June 2024.

- [9] Titus J. Brinker, Achim Hekler, Alexander H. Enk, Carola Berking, Sebastian Haferkamp, Axel Hauschild, Michael Weichenthal, Joachim Klode, Dirk Schadendorf, Tim Holland-Letz, Christof von Kalle, Stefan Fröhling, Bastian Schilling, and Jochen S. Utikal. Deep neural networks are superior to dermatologists in melanoma image classification. *European Journal of Cancer*, 119:11–17, 2019.
- [10] Ha Manh Bui and Anqi Liu. Density-softmax: Efficient test-time model for uncertainty estimation and robustness under distribution shifts. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.
- [11] Mohammad-Amin Charusaie, Hussein Mozannar, David A. Sontag, and Samira Samadi. Sample efficient learning of predictors that complement humans. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2972–3005. PMLR, 2022.
- [12] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1800–1807. IEEE Computer Society, 2017.
- [13] C. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970.
- [14] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2898–2909, 2019.
- [15] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In Ronald Ortner, Hans Ulrich Simon, and Sandra Zilles, editors, *Algorithmic Learning Theory*, pages 67–82, Cham, 2016. Springer International Publishing.
- [16] Andreas C. Damianou and Neil D. Lawrence. Deep gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, volume 31 of *JMLR Workshop and Conference Proceedings*, pages 207–215. JMLR.org, 2013.

- [17] Shai Danziger, Jonathan Levav, and Liora Avnaim-Pesso. Extraneous factors in judicial decisions. *Proceedings of the National Academy of Sciences of the United States of America*, 108:6889–92, 04 2011.
- [18] Maria De-Arteaga, Riccardo Fogliato, and Alexandra Chouldechova. A case for humans-in-the-loop: Decisions in the presence of erroneous algorithmic scores. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20. ACM, April 2020.
- [19] Hans de Bruijn, Martijn Warnier, and Marijn Janssen. The perils and pitfalls of explainable ai: Strategies for explaining algorithmic decision-making. *Government Information Quarterly*, 39(2):101666, 2022.
- [20] Dominik Dellermann, Philipp Ebel, Matthias Söllner, and Jan Marco Leimeister. Hybrid intelligence. *Business amp; Information Systems Engineering*, 61(5):637–643, March 2019.
- [21] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1192–1201. PMLR, 2018.
- [22] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [23] Charles Elkan. The foundations of cost-sensitive learning. *Proceedings of the Seventeenth International Conference on Artificial Intelligence: 4-10 August 2001; Seattle*, 1, 05 2001.
- [24] Andre Esteva, Brett Kuprel, Roberto Novoa, Justin Ko, Susan Swetter, Helen Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542, 01 2017.
- [25] Jeffrey Fauw, Joseph Ledsam, Bernardino Romera-Paredes, Stanislav Nikolov, Nenad Tomasev, Sam Blackwell, Harry Askham, Xavier Glorot, Brendan O’Donoghue, Daniel Visentin, George Driessche, Balaji Lakshminarayanan, Clemens Meyer, Faith Mackinder, Simon Bouton, Kareem Ayoub, Reena Chopra, Dominic King, Alan Karthikesalingam, and Olaf Ronneberger. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature Medicine*, 24, 09 2018.
- [26] Jerome Friedman. Stochastic gradient boosting. *Computational Statistics Data Analysis*, 38:367–378, 02 2002.

- [27] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059. JMLR.org, 2016.
- [28] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Hamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46, 04 2014.
- [29] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4878–4887, 2017.
- [30] Tilmann Gneiting and Adrian Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102:359–378, 03 2007.
- [31] Sharad Goel, Ravi Shroff, Jennifer Skeem, and Christopher Slobogin. *The accuracy, equity, and jurisprudence of criminal risk assessment*. 05 2021.
- [32] Alison Gopnik and Henry M Wellman. Reconstructing constructivism: causal models, bayesian learning mechanisms, and the theory theory. *Psychological bulletin*, 138(6):1085, 2012.
- [33] Ben Green and Yiling Chen. The principles and limits of algorithm-in-the-loop decision making. *Proceedings of the ACM on Human-Computer Interaction*, 3:1 – 24, 2019.
- [34] Stein Grimstad and Magne Jørgensen. Inconsistency of expert judgment-based estimates of software development effort. *Journal of Systems and Software*, 80:1770–1777, 11 2007.
- [35] Varun Gulshan, Lily Peng, Marc Coram, Martin Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Ramasamy Kim, Rajiv Raman, Philip Nelson, Jessica Mega, and Dale Webster. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316, 11 2016.
- [36] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017.

- [37] Yotam Hechtlinger, Barnabás Póczos, and Larry A. Wasserman. Cautious deep learning. *CoRR*, abs/1805.09460, 2018.
- [38] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 41–50. Computer Vision Foundation / IEEE, 2019.
- [39] Patrick Hemmer, Sebastian Schellhammer, Michael Vössing, Johannes Jakubik, and Gerhard Satzger. Forming effective human-ai teams: Building machine learning models that complement the capabilities of multiple experts. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 2478–2484. ijcai.org, 2022.
- [40] Patrick Hemmer, Lukas Thede, Michael Vössing, Johannes Jakubik, and Niklas Kühl. Learning to defer with limited expert predictions. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, pages 6002–6011. AAAI Press, 2023.
- [41] Kilian Hendrickx, Lorenzo Perini, Dries Van der Plas, Wannes Meert, and Jesse Davis. Machine learning with a reject option: a survey. *Mach. Learn.*, 113(5):3073–3110, 2024.
- [42] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
- [43] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 110(3):457–506, March 2021.
- [44] Sérgio Jesus, Catarina Belém, Vladimir Balayan, João Bento, Pedro Saleiro, Pedro Bizarro, and João Gama. How can i choose an explainer? an application-grounded evaluation of post-hoc explanations. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21*, page 805–815, New York, NY, USA, 2021. Association for Computing Machinery.
- [45] Sérgio M. Jesus, José Pombal, Duarte M. Alves, André Ferreira Cruz, Pedro Saleiro, Rita P. Ribeiro, João Gama, and Pedro Bizarro. Turning the tables: Biased, imbalanced, dynamic tabular datasets for ML evaluation. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and

- A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, 2022*.
- [46] James Robins Jing Lei and Larry Wasserman. Distribution-free prediction sets. *Journal of the American Statistical Association*, 108(501):278–287, 2013. PMID: 25237208.
- [47] Wittawat Jitkrittum, Neha Gupta, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, and Sanjiv Kumar. When does confidence-based cascade deferral suffice? In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023*.
- [48] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Neural Information Processing Systems*, 2017.
- [49] Vijay Keswani, Matthew Lease, and Krishnaram Kenthapadi. Towards unbiased and accurate deferral to multiple experts. In Marion Fourcade, Benjamin Kuipers, Seth Lazar, and Deirdre K. Mulligan, editors, *AIES '21: AAAI/ACM Conference on AI, Ethics, and Society, Virtual Event, USA, May 19-21, 2021*, pages 154–165. ACM, 2021.
- [50] Amir E. Khandani, Adlar J. Kim, and Andrew W. Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking Finance*, 34(11):2767–2787, 2010.
- [51] Lauren Kirchner and Jeff Larson. How we analyzed the compas recidivism algorithm. 2016.
- [52] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.
- [53] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6402–6413, 2017.
- [54] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David J. Crandall, and Dhruv Batra. Why M heads are better than one: Training a diverse ensemble of deep networks. *CoRR*, abs/1511.06314, 2015.
- [55] Diogo Leitão, Pedro Saleiro, Mário A. T. Figueiredo, and Pedro Bizarro. Human-ai collaboration in decision-making: Beyond learning to defer. *CoRR*, abs/2206.13202, 2022.

- [56] Jeremiah Z. Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [57] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4765–4774, 2017.
- [58] David John Cameron MacKay. A practical bayesian framework for backpropagation networks. *Neural Computation*, 4:448–472, 1992.
- [59] David Madras, Toniann Pitassi, and Richard S. Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6150–6160, 2018.
- [60] Andrey Malinin, Liudmila Prokhorenkova, and Aleksei Ustimenko. Uncertainty in gradient boosting via ensembles. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [61] Natalia Martínez, Martín Bertrán, and Guillermo Sapiro. Minimax pareto fairness: A multi objective perspective. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6755–6764. PMLR, 2020.
- [62] Soundouss Messoudi, Sylvain Rousseau, and Sébastien Destercke. Deep conformal prediction for robust models. In Marie-Jeanne Lesot, Susana M. Vieira, Marek Z. Reformat, João Paulo Carvalho, Anna Wilbik, Bernadette Bouchon-Meunier, and Ronald R. Yager, editors, *Information Processing and Management of Uncertainty in Knowledge-Based Systems - 18th International Conference, IPMU 2020, Lisbon, Portugal, June 15-19, 2020, Proceedings, Part I*, volume 1237 of *Communications in Computer and Information Science*, pages 528–540. Springer, 2020.
- [63] Hussein Mozannar, Hunter Lang, Dennis Wei, Prasanna Sattigeri, Subhro Das, and David A. Sonntag. Who should predict? exact algorithms for learning to defer to humans. In Francisco J. R.

- Ruiz, Jennifer G. Dy, and Jan-Willem van de Meent, editors, *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*, volume 206 of *Proceedings of Machine Learning Research*, pages 10520–10545. PMLR, 2023.
- [64] Hussein Mozannar and David A. Sontag. Consistent estimators for learning to defer to an expert. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7076–7087. PMLR, 2020.
- [65] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015.
- [66] Vu-Linh Nguyen, Sébastien Destercke, and Eyke Hüllermeier. Epistemic uncertainty sampling. In Petra Kralj Novak, Tomislav Smuc, and Saso Dzeroski, editors, *Discovery Science - 22nd International Conference, DS 2019, Split, Croatia, October 28-30, 2019, Proceedings*, volume 11828 of *Lecture Notes in Computer Science*, pages 72–86. Springer, 2019.
- [67] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Feb. 2015.
- [68] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Machine Learning: ECML 2002*, pages 345–356, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
- [69] Juan C. Perdomo, Tijana Zrnica, Celestine Mendler-Dünner, and Moritz Hardt. Performative prediction. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 7599–7609. PMLR, 2020.
- [70] Miquel Perello-Nieto, Telmo M Silva Filho, Meelis Kull, and Peter Flach. Background check: A general technique to build more reliable and versatile classifiers. In *2016 IEEE 16th International Conference on Data Mining (ICDM 2016)*, Proceedings of the IEEE International Conference on Data Mining (ICDM), pages 1143–1148, United States, March 2017. Institute of Electrical and Electronics Engineers (IEEE). 16th IEEE International Conference on Data Mining, ICDM 2016 ; Conference date: 12-12-2016 Through 15-12-2016.
- [71] Laurent Perron and Frédéric Didier. Cp-sat.

- [72] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features, 2019.
- [73] Mark D. Reid and Robert C. Williamson. Composite binary losses, 2009.
- [74] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1530–1538. JMLR.org, 2015.
- [75] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.
- [76] Waddah Saeed and Christian Omlin. Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems*, 263:110273, 2023.
- [77] Robin Senge, Stefan Bösner, Krzysztof Dembczyński, Jörg Haasenritter, Oliver Hirsch, Norbert Donner-Banzhoff, and Eyke Hüllermeier. Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Information Sciences*, 255:16–29, 2014.
- [78] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *J. Mach. Learn. Res.*, 9:371–421, 2008.
- [79] Mohammad Hossein Shaker and Eyke Hüllermeier. Aleatoric and epistemic uncertainty with random forests, 2020.
- [80] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need, 2021.
- [81] Sriram Somanchi, Samrachana Adhikari, Allen Lin, Elena Eneva, and Rayid Ghani. Early prediction of cardiac arrest (code blue) using electronic medical records. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 2119–2126, New York, NY, USA, 2015. Association for Computing Machinery.
- [82] Peter J. Stuckey, Thibaut Feydy, Andreas Schutt, Guido Tack, and Julien Fischer. The minizinc challenge 2008–2013. *AI Magazine*, 35(2):55–60, Jun. 2014.
- [83] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.
- [84] Dharmesh Tailor, Aditya Patra, Rajeev Verma, Putra Manggala, and Eric Nalisnick. Learning to defer to a population: A meta-learning approach, 2024.

- [85] Guido Travaini, Federico Pacchioni, Silvia Bellumore, Marta Bosia, and Francesco De Micco. Machine learning and criminal justice: A systematic review of advanced methodology for recidivism risk prediction. *International Journal of Environmental Research and Public Health*, 19:10594, 08 2022.
- [86] Aleksei Ustimenko and Liudmila Prokhorenkova. Sglb: Stochastic gradient langevin boosting. In *International Conference on Machine Learning*, pages 10487–10496. PMLR, 2021.
- [87] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.
- [88] Kush R. Varshney and Homa Alemzadeh. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products, 2017.
- [89] Rajeev Verma, Daniel Barrejón, and Eric T. Nalisnick. Learning to defer to multiple experts: Consistent surrogate losses, confidence calibration, and conformal ensembles. In Francisco J. R. Ruiz, Jennifer G. Dy, and Jan-Willem van de Meent, editors, *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain*, volume 206 of *Proceedings of Machine Learning Research*, pages 11415–11434. PMLR, 2023.
- [90] Rajeev Verma and Eric T. Nalisnick. Calibrated learning to defer with one-vs-all classifiers. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 22184–22202. PMLR, 2022.
- [91] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. 01 2005.
- [92] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7333–7343, 2019.
- [93] Fan Yang, Hua-zhen Wang, Hong Mi, Lin Chengde, and Wei-wen Cai. Using random forest for reliable classification and cost-sensitive learning for medical diagnosis. *BMC bioinformatics*, 10 Suppl 1:S22, 02 2009.
- [94] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Third IEEE International Conference on Data Mining*, pages 435–442, 2003.

Appendix A

Hyperparameter Selection

A.1 Alert Model

Hyperparameter optimization for the alert model is conducted using tree-structured parzen estimators (TPE), a Bayesian optimization technique, to explore the hyperparameter space [3]. This optimization process runs for 100 trials, aiming to maximize recall at a 5% FPR on the validation set. The hyperparameter search space and the values selected for the final model are shown in Table A.1.

Table A.1: Alert Model: LightGBM hyperparameter search space

Hyperparameter	Values or Interval	Distribution	Selected
boosting_type	“goss”		“goss”
enable_bundle	False		False
n_estimators	[50, 5000]	Log	163
max_depth	[2, 20]	Uniform	2
num_leaves	[10, 1000]	Log	208
min_child_samples	[5, 500]	Log	20
learning_rate	[0.01, 0.5]	Log	0.3506776
reg_alpha	[0.0001, 0.1]	Log	0.0036325
reg_lambda	[0.0001, 0.1]	Log	0.0043664

A.2 MLP

The MLP is trained on alerts from months 4 to 6 and validated on alerts from month 7. It is optimized to minimize the weighted log-loss described in Equation (5.6). Hyperparameter optimization is performed using TPE [3], running for 100 trials with 10 startup trials. The hyperparameter search space and the selected parameters are shown in Table A.2.

Table A.2: Feature Extraction MLP: Hyperparameter Search Space

Hyperparameter	Values or Interval	Distribution	Selected
optimizer	Adam		Adam
hidden_layer_sizes	{[128/200, 20/50/128, 0/20/50]}		[128, 50]
learning_rate	[1e-5, 1e-3]	Log	3.096e-5
weight_decay	[1e-5, 1e-3]	Log	3.84e-4
dropout_rate	[0.0, 0.5]	Uniform	0.056
num_epochs	[20, 200]	Uniform	73
batch_size	[10, 128]	Log	30

A.3 RealNVP models

We conduct hyperparameter optimization for the RealNVP model using TPE [3]. We use alert data from months 4 to 6 as the training set and validate the model on data from month 7. The optimization process involves 100 trials, with 10 startup trials, where the goal is to find a configuration that minimizes the negative log-likelihood of the validation data under the model’s distribution. The final model parameters were selected based on their performance on the validation set. Table A.3 summarizes the hyperparameter search space.

Table A.3: RealNVP: Hyperparameter Search Space

Hyperparameter	Values or Interval	Distribution
num_coupling_layers	[2, 10]	Uniform
hidden_dim	[64, 256]	Uniform (step 32)
learning_rate	[1e-5, 1e-3]	Log
num_epochs	[50, 300]	Uniform

This procedure is repeated for each expert and each data availability scenario, resulting in a unique density model for each expert, trained solely on the subset of training and validation data labeled by that expert (we don’t include the selected parameters due to the large number of models this results in).

A.4 Learning to Defer models

For the ML classifier h , we perform hyperparameter optimization using TPE [3], running the optimization process for 1700 trials with 1500 start-up trials, using the hyperparameter search space in Table A.4. The goal of this search is to minimize the weighted log-loss (see Equation 5.6).

Since false negatives carry a higher cost, we explore whether introducing a bias towards predicting fraud is beneficial for our model. This is done by testing different initial probability estimates for the base predictor in the boosting model. During model training, we perform an independent hyperparameter search over a range of values for the initial prediction value $g_{initial}$, from g_d to $g_d + 2$, in increments of

Table A.4: ML classifier: LightGBM hyperparameter search space

Hyperparameter	Values or Interval	Distribution
boosting_type	“dart”	
enable_bundle	[False, True]]	
n_estimators	[50, 250]	Uniform
max_depth	[2, 5]	Uniform
num_leaves	[100, 1000]	Uniform
min_child_samples	[5, 200]	Uniform
learning_rate	[0.001, 1]	Uniform
reg_alpha	[0.0001, 2]	Uniform
reg_lambda	[0.0001, 2]	Uniform

0.2, where g_d is the default initial prediction value

$$g_d = \text{logit} \left(\frac{\sum_i c_i y_i}{\sum_i c_i} \right). \quad (\text{A.1})$$

For the expert models, hyperparameter optimization is also done using TPE [3], running the optimization for 120 trials with 100 start-up trials, on the search space described in Table A.5. The goal is also to minimize the weighted log-loss (see Equation 5.7).

Table A.5: Expert Models: LightGBM hyperparameter search space

Hyperparameter	Values or Interval	Distribution
boosting_type	“dart”	
enable_bundle	[False, True]]	
n_estimators	[50, 250]	Uniform
max_depth	[2, 20]	Uniform
num_leaves	[100, 1000]	Log
min_child_samples	[5, 100]	Log
learning_rate	[0.005, 0.5]	Log
reg_alpha	[0.0001, 0.1]	Log
reg_lambda	[0.0001, 0.1]	Log

Appendix B

Extended Results

B.1 Assignment system extended results

This appendix provides extended details on the results of the deferral systems in the various testing scenarios tested. In Table B.1, B.2, B.3, B.4 and B.5, we present the expected misclassification cost per 100 instances, as in chapter 7, but for a larger variety of settings. Each setting is the result of selecting one of the five possible test sets with noise introduced, selecting 1, 3, or 5 as the number of experts (NE) to integrate the expert team, selecting between 20% or 50% deferral rate (DR), and selecting a data availability (DA) scenario of either the full dataset, 1/5, 1/20 or 1/40 of the dataset. The results shown are averaged over 5 runs with 95% confidence intervals for the mean. For each run, we change the random seeds used for the addition of noise and the training data selection.

B.2 Null set predictions

The number of null set predictions outputted by our density based conformal prediction depends highly on the noise setting and on the deferral rate. The noise setting determines the magnitude of drift in the test set. In cases with more drift, the conformal method outputs more null set predictions for lower coverage levels α . The deferral rate affects the optimization process described in Equation 5.11. The optimization problem will result in a coverage level α that respects the human experts' capacity constraints.

In Table B.6 we present the percentage of null set predictions outputted for each of these settings, with 95% confidence intervals. The results are averaged over 5 runs, changing the random seed used for the introduction of noise in each run. Table B.6 was obtained for 5 experts and the full training data availability scenario, however, these results are representative of what is obtained in other scenarios.

Setting				Deferral Strategy				
Noise	NE	DR	DA	CP	DS	L2D	RL	Random
categorical	1	20	1	<u>5.24</u> ± 0.04	5.23 ± 0.08	5.54 ± 0.14	5.59 ± 0.19	5.47 ± 0.22
categorical	1	30	1	5.03 ± 0.07	<u>5.09</u> ± 0.03	5.18 ± 0.14	5.43 ± 0.12	5.50 ± 0.17
categorical	1	40	1	<u>4.93</u> ± 0.09	4.93 ± 0.08	5.18 ± 0.14	5.24 ± 0.13	5.29 ± 0.17
categorical	1	50	1	<u>4.83</u> ± 0.12	4.78 ± 0.11	5.18 ± 0.14	5.05 ± 0.20	4.95 ± 0.16
categorical	3	20	1	5.36 ± 0.16	<u>5.37</u> ± 0.20	5.78 ± 0.14	5.62 ± 0.16	5.75 ± 0.24
categorical	3	30	1	4.95 ± 0.10	<u>4.97</u> ± 0.10	5.25 ± 0.14	5.40 ± 0.13	5.51 ± 0.17
categorical	3	40	1	4.73 ± 0.06	<u>4.76</u> ± 0.08	5.10 ± 0.14	5.44 ± 0.14	5.34 ± 0.25
categorical	3	50	1	4.60 ± 0.04	<u>4.61</u> ± 0.05	5.09 ± 0.14	5.24 ± 0.13	5.30 ± 0.18
categorical	5	20	1	5.22 ± 0.15	<u>5.28</u> ± 0.12	5.72 ± 0.14	5.60 ± 0.21	5.63 ± 0.11
categorical	5	30	1	<u>4.93</u> ± 0.10	4.93 ± 0.14	5.13 ± 0.14	5.40 ± 0.15	5.29 ± 0.16
categorical	5	40	1	4.69 ± 0.11	<u>4.73</u> ± 0.13	4.99 ± 0.14	5.27 ± 0.16	5.17 ± 0.22
categorical	5	50	1	4.52 ± 0.09	<u>4.53</u> ± 0.04	4.92 ± 0.14	5.03 ± 0.22	5.22 ± 0.21
numerical	1	20	1	5.33 ± 0.23	5.29 ± 0.22	5.55 ± 0.13	<u>5.31</u> ± 0.11	5.35 ± 0.22
numerical	1	30	1	<u>5.19</u> ± 0.21	5.25 ± 0.22	5.12 ± 0.13	5.24 ± 0.08	5.30 ± 0.13
numerical	1	40	1	5.08 ± 0.23	<u>5.06</u> ± 0.22	5.05 ± 0.13	5.13 ± 0.12	5.27 ± 0.15
numerical	1	50	1	4.96 ± 0.21	<u>4.93</u> ± 0.21	5.04 ± 0.13	5.07 ± 0.11	4.88 ± 0.21
numerical	3	20	1	<u>5.46</u> ± 0.17	5.31 ± 0.25	5.56 ± 0.13	5.49 ± 0.10	5.66 ± 0.07
numerical	3	30	1	5.12 ± 0.19	4.99 ± 0.23	<u>5.02</u> ± 0.13	5.28 ± 0.12	5.29 ± 0.15
numerical	3	40	1	<u>4.91</u> ± 0.20	4.80 ± 0.18	5.01 ± 0.13	5.34 ± 0.11	5.20 ± 0.19
numerical	3	50	1	<u>4.72</u> ± 0.11	4.68 ± 0.16	5.04 ± 0.13	5.21 ± 0.12	5.15 ± 0.19
numerical	5	20	1	5.30 ± 0.17	5.30 ± 0.24	5.57 ± 0.13	<u>5.30</u> ± 0.16	5.54 ± 0.15
numerical	5	30	1	5.04 ± 0.23	<u>5.00</u> ± 0.19	4.92 ± 0.13	5.24 ± 0.09	5.12 ± 0.15
numerical	5	40	1	4.75 ± 0.18	4.76 ± 0.18	<u>4.75</u> ± 0.13	5.16 ± 0.11	5.03 ± 0.21
numerical	5	50	1	<u>4.58</u> ± 0.15	4.58 ± 0.18	4.76 ± 0.13	5.06 ± 0.12	5.06 ± 0.11
low	1	20	1	5.58 ± 0.10	<u>5.60</u> ± 0.10	5.80 ± 0.14	5.69 ± 0.10	5.78 ± 0.21
low	1	30	1	5.41 ± 0.04	<u>5.43</u> ± 0.08	5.47 ± 0.14	5.49 ± 0.13	5.64 ± 0.14
low	1	40	1	5.22 ± 0.09	<u>5.28</u> ± 0.12	5.47 ± 0.14	5.35 ± 0.14	5.50 ± 0.08
low	1	50	1	<u>5.15</u> ± 0.06	5.18 ± 0.11	5.45 ± 0.14	5.25 ± 0.12	5.14 ± 0.18
low	3	20	1	5.58 ± 0.16	<u>5.60</u> ± 0.16	6.01 ± 0.14	5.85 ± 0.10	5.95 ± 0.15
low	3	30	1	5.21 ± 0.18	<u>5.24</u> ± 0.16	5.44 ± 0.14	5.56 ± 0.13	5.64 ± 0.13
low	3	40	1	5.04 ± 0.22	<u>5.05</u> ± 0.19	5.43 ± 0.14	5.61 ± 0.24	5.53 ± 0.05
low	3	50	1	<u>4.89</u> ± 0.20	4.86 ± 0.13	5.47 ± 0.14	5.61 ± 0.19	5.55 ± 0.09
low	5	20	1	5.45 ± 0.11	<u>5.49</u> ± 0.12	5.98 ± 0.14	5.71 ± 0.10	5.90 ± 0.16
low	5	30	1	<u>5.12</u> ± 0.20	5.09 ± 0.20	5.36 ± 0.14	5.50 ± 0.08	5.49 ± 0.10
low	5	40	1	4.93 ± 0.23	<u>4.95</u> ± 0.23	5.20 ± 0.14	5.41 ± 0.15	5.47 ± 0.27
low	5	50	1	<u>4.77</u> ± 0.21	4.75 ± 0.17	5.12 ± 0.14	5.25 ± 0.15	5.32 ± 0.14
medium	1	20	1	5.72 ± 0.20	6.01 ± 0.09	6.39 ± 0.03	<u>5.72</u> ± 0.11	6.28 ± 0.10
medium	1	30	1	5.45 ± 0.24	5.82 ± 0.15	5.98 ± 0.03	<u>5.49</u> ± 0.11	6.18 ± 0.08
medium	1	40	1	5.29 ± 0.17	<u>5.65</u> ± 0.15	5.96 ± 0.03	5.76 ± 0.14	6.04 ± 0.27
medium	1	50	1	5.20 ± 0.17	5.54 ± 0.14	5.93 ± 0.04	5.63 ± 0.14	<u>5.38</u> ± 0.19
medium	3	20	1	5.81 ± 0.19	6.03 ± 0.11	6.57 ± 0.03	<u>6.02</u> ± 0.14	6.48 ± 0.08
medium	3	30	1	5.52 ± 0.15	5.70 ± 0.09	5.94 ± 0.03	<u>5.58</u> ± 0.08	6.12 ± 0.12
medium	3	40	1	5.26 ± 0.15	<u>5.45</u> ± 0.08	5.84 ± 0.03	5.90 ± 0.15	6.04 ± 0.09
medium	3	50	1	5.00 ± 0.20	<u>5.25</u> ± 0.09	5.81 ± 0.03	5.93 ± 0.11	6.12 ± 0.13
medium	5	20	1	5.65 ± 0.18	5.94 ± 0.10	6.52 ± 0.03	<u>5.70</u> ± 0.13	6.41 ± 0.08
medium	5	30	1	5.40 ± 0.20	5.62 ± 0.09	5.86 ± 0.03	<u>5.53</u> ± 0.15	6.07 ± 0.11
medium	5	40	1	5.12 ± 0.16	<u>5.36</u> ± 0.08	5.71 ± 0.03	5.74 ± 0.19	5.88 ± 0.26
medium	5	50	1	4.93 ± 0.15	<u>5.13</u> ± 0.13	5.56 ± 0.03	5.63 ± 0.17	5.97 ± 0.12

Table B.1: Expected Misclassification Cost per 100 instances for each setting - part 1

Setting				Deferral Strategy				
Noise	NE	DR	DA	CP	DS	L2D	RL	Random
high	1	20	1	5.30 ± 0.12	6.11 ± 0.10	6.40 ± 0.13	<u>5.69</u> ± 0.12	6.23 ± 0.18
high	1	30	1	5.01 ± 0.08	5.85 ± 0.12	6.02 ± 0.13	<u>5.58</u> ± 0.07	6.24 ± 0.20
high	1	40	1	4.88 ± 0.06	5.74 ± 0.13	6.00 ± 0.13	<u>5.70</u> ± 0.09	6.00 ± 0.25
high	1	50	1	4.81 ± 0.09	5.51 ± 0.12	5.97 ± 0.13	5.73 ± 0.06	<u>5.39</u> ± 0.17
high	3	20	1	5.42 ± 0.15	6.11 ± 0.14	6.44 ± 0.13	<u>6.08</u> ± 0.09	6.51 ± 0.17
high	3	30	1	5.09 ± 0.17	5.67 ± 0.11	5.95 ± 0.13	<u>5.65</u> ± 0.19	6.13 ± 0.17
high	3	40	1	4.68 ± 0.08	<u>5.49</u> ± 0.08	6.03 ± 0.13	5.91 ± 0.16	6.22 ± 0.14
high	3	50	1	4.61 ± 0.07	<u>5.33</u> ± 0.14	5.96 ± 0.13	6.07 ± 0.15	6.05 ± 0.15
high	5	20	1	5.25 ± 0.17	5.98 ± 0.10	6.37 ± 0.13	<u>5.68</u> ± 0.19	6.50 ± 0.14
high	5	30	1	4.97 ± 0.12	<u>5.56</u> ± 0.10	5.83 ± 0.13	5.59 ± 0.13	6.13 ± 0.15
high	5	40	1	4.57 ± 0.11	<u>5.34</u> ± 0.13	5.74 ± 0.13	5.73 ± 0.08	5.94 ± 0.07
high	5	50	1	4.37 ± 0.09	<u>5.18</u> ± 0.15	5.69 ± 0.13	5.69 ± 0.18	5.90 ± 0.18
categorical	1	20	1/5	5.28 ± 0.12	<u>5.29</u> ± 0.13	5.49 ± 0.17	5.59 ± 0.19	5.54 ± 0.11
categorical	1	30	1/5	5.07 ± 0.13	5.06 ± 0.13	<u>5.07</u> ± 0.14	5.43 ± 0.12	5.47 ± 0.30
categorical	1	40	1/5	<u>4.85</u> ± 0.06	4.85 ± 0.06	5.03 ± 0.20	5.24 ± 0.13	5.32 ± 0.20
categorical	1	50	1/5	4.67 ± 0.04	4.68 ± 0.09	5.05 ± 0.11	5.05 ± 0.20	4.99 ± 0.13
categorical	3	20	1/5	5.36 ± 0.16	<u>5.37</u> ± 0.11	5.69 ± 0.17	5.62 ± 0.16	5.72 ± 0.17
categorical	3	30	1/5	5.02 ± 0.12	<u>5.02</u> ± 0.15	5.16 ± 0.26	5.47 ± 0.16	5.39 ± 0.13
categorical	3	40	1/5	4.82 ± 0.14	<u>4.84</u> ± 0.14	5.07 ± 0.21	5.46 ± 0.14	5.35 ± 0.14
categorical	3	50	1/5	4.62 ± 0.12	<u>4.65</u> ± 0.10	5.05 ± 0.17	5.23 ± 0.20	5.14 ± 0.18
categorical	5	20	1/5	<u>5.29</u> ± 0.12	5.26 ± 0.15	5.56 ± 0.09	5.56 ± 0.20	5.67 ± 0.14
categorical	5	30	1/5	5.04 ± 0.15	<u>5.04</u> ± 0.22	5.06 ± 0.21	5.42 ± 0.12	5.28 ± 0.22
categorical	5	40	1/5	4.79 ± 0.08	<u>4.81</u> ± 0.05	4.95 ± 0.13	5.31 ± 0.17	5.21 ± 0.25
categorical	5	50	1/5	4.61 ± 0.13	<u>4.62</u> ± 0.13	5.00 ± 0.28	5.13 ± 0.16	5.15 ± 0.07
numerical	1	20	1/5	5.30 ± 0.23	5.34 ± 0.21	5.53 ± 0.19	<u>5.31</u> ± 0.11	5.41 ± 0.14
numerical	1	30	1/5	<u>5.14</u> ± 0.22	5.15 ± 0.19	4.98 ± 0.16	5.24 ± 0.08	5.30 ± 0.21
numerical	1	40	1/5	4.99 ± 0.19	<u>4.97</u> ± 0.22	4.90 ± 0.18	5.13 ± 0.12	5.24 ± 0.17
numerical	1	50	1/5	<u>4.81</u> ± 0.08	4.78 ± 0.08	4.91 ± 0.10	5.07 ± 0.11	4.99 ± 0.21
numerical	3	20	1/5	<u>5.41</u> ± 0.20	5.29 ± 0.21	5.68 ± 0.11	5.45 ± 0.14	5.63 ± 0.16
numerical	3	30	1/5	5.13 ± 0.15	5.03 ± 0.26	<u>5.11</u> ± 0.23	5.28 ± 0.15	5.24 ± 0.17
numerical	3	40	1/5	<u>4.96</u> ± 0.25	4.88 ± 0.30	4.97 ± 0.10	5.31 ± 0.17	5.25 ± 0.23
numerical	3	50	1/5	<u>4.86</u> ± 0.10	4.80 ± 0.23	4.92 ± 0.30	5.21 ± 0.10	5.13 ± 0.19
numerical	5	20	1/5	5.35 ± 0.18	5.26 ± 0.23	5.57 ± 0.18	<u>5.32</u> ± 0.09	5.56 ± 0.17
numerical	5	30	1/5	5.05 ± 0.15	5.05 ± 0.25	5.00 ± 0.17	5.24 ± 0.12	5.10 ± 0.22
numerical	5	40	1/5	4.92 ± 0.24	4.85 ± 0.21	<u>4.89</u> ± 0.24	5.15 ± 0.08	4.99 ± 0.14
numerical	5	50	1/5	4.74 ± 0.12	4.63 ± 0.16	4.86 ± 0.33	5.06 ± 0.16	5.03 ± 0.14
low	1	20	1/5	5.54 ± 0.12	5.56 ± 0.13	5.74 ± 0.13	5.69 ± 0.10	5.78 ± 0.15
low	1	30	1/5	5.36 ± 0.16	5.30 ± 0.22	<u>5.32</u> ± 0.20	5.49 ± 0.13	5.65 ± 0.17
low	1	40	1/5	5.12 ± 0.15	5.14 ± 0.18	5.32 ± 0.26	5.35 ± 0.14	5.57 ± 0.28
low	1	50	1/5	4.99 ± 0.15	5.00 ± 0.13	5.32 ± 0.05	5.25 ± 0.12	5.14 ± 0.12
low	3	20	1/5	5.57 ± 0.16	<u>5.61</u> ± 0.09	6.02 ± 0.17	5.86 ± 0.12	5.97 ± 0.16
low	3	30	1/5	<u>5.30</u> ± 0.14	5.28 ± 0.21	5.45 ± 0.18	5.57 ± 0.09	5.54 ± 0.26
low	3	40	1/5	5.16 ± 0.13	5.13 ± 0.10	5.36 ± 0.26	5.58 ± 0.11	5.50 ± 0.28
low	3	50	1/5	4.92 ± 0.11	4.93 ± 0.10	5.34 ± 0.24	5.59 ± 0.14	5.47 ± 0.08
low	5	20	1/5	5.44 ± 0.08	5.54 ± 0.10	5.87 ± 0.11	5.70 ± 0.18	5.95 ± 0.12
low	5	30	1/5	5.26 ± 0.19	<u>5.29</u> ± 0.17	5.37 ± 0.13	5.47 ± 0.13	5.51 ± 0.12
low	5	40	1/5	5.02 ± 0.18	<u>5.04</u> ± 0.19	5.26 ± 0.21	5.42 ± 0.17	5.42 ± 0.10
low	5	50	1/5	<u>4.86</u> ± 0.22	4.86 ± 0.17	5.28 ± 0.19	5.32 ± 0.18	5.47 ± 0.18

Table B.2: Expected Misclassification Cost per 100 instances for each setting - part 2

Setting				Deferral Strategy				
Noise	NE	DR	DA	CP	DS	L2D	RL	Random
medium	1	20	1/5	5.71 ± 0.19	6.03 ± 0.10	6.30 ± 0.08	<u>5.72</u> ± 0.11	6.20 ± 0.11
medium	1	30	1/5	<u>5.49</u> ± 0.15	5.76 ± 0.17	5.82 ± 0.12	5.49 ± 0.11	6.15 ± 0.05
medium	1	40	1/5	5.26 ± 0.21	<u>5.54</u> ± 0.14	5.81 ± 0.13	5.76 ± 0.14	6.05 ± 0.18
medium	1	50	1/5	<u>5.13</u> ± 0.17	5.32 ± 0.08	5.81 ± 0.11	5.63 ± 0.14	5.12 ± 0.08
medium	3	20	1/5	5.88 ± 0.15	6.10 ± 0.08	6.53 ± 0.09	<u>6.03</u> ± 0.11	6.47 ± 0.08
medium	3	30	1/5	5.51 ± 0.23	5.76 ± 0.26	5.93 ± 0.25	<u>5.53</u> ± 0.10	6.11 ± 0.14
medium	3	40	1/5	5.36 ± 0.22	<u>5.51</u> ± 0.15	5.86 ± 0.18	5.97 ± 0.13	6.05 ± 0.09
medium	3	50	1/5	5.17 ± 0.27	<u>5.36</u> ± 0.19	5.82 ± 0.19	5.85 ± 0.14	6.03 ± 0.05
medium	5	20	1/5	<u>5.81</u> ± 0.14	5.98 ± 0.08	6.36 ± 0.14	5.73 ± 0.14	6.44 ± 0.04
medium	5	30	1/5	<u>5.52</u> ± 0.17	5.72 ± 0.11	5.82 ± 0.14	5.51 ± 0.05	6.08 ± 0.06
medium	5	40	1/5	5.28 ± 0.26	<u>5.46</u> ± 0.06	5.80 ± 0.15	5.69 ± 0.22	5.87 ± 0.15
medium	5	50	1/5	5.04 ± 0.18	<u>5.20</u> ± 0.23	5.78 ± 0.18	5.64 ± 0.14	5.99 ± 0.05
high	1	20	1/5	5.36 ± 0.18	6.08 ± 0.05	6.35 ± 0.10	<u>5.69</u> ± 0.12	6.22 ± 0.13
high	1	30	1/5	5.02 ± 0.12	5.78 ± 0.11	5.86 ± 0.12	<u>5.58</u> ± 0.07	6.22 ± 0.15
high	1	40	1/5	4.85 ± 0.14	<u>5.56</u> ± 0.13	5.84 ± 0.16	5.70 ± 0.09	6.04 ± 0.12
high	1	50	1/5	4.64 ± 0.10	<u>5.23</u> ± 0.22	5.84 ± 0.24	5.73 ± 0.06	5.29 ± 0.24
high	3	20	1/5	5.50 ± 0.13	6.09 ± 0.10	6.47 ± 0.19	<u>6.08</u> ± 0.11	6.48 ± 0.14
high	3	30	1/5	5.04 ± 0.12	5.74 ± 0.11	5.98 ± 0.25	<u>5.68</u> ± 0.16	6.21 ± 0.10
high	3	40	1/5	4.87 ± 0.06	<u>5.53</u> ± 0.16	5.92 ± 0.27	5.98 ± 0.25	6.13 ± 0.16
high	3	50	1/5	4.81 ± 0.14	<u>5.37</u> ± 0.17	5.87 ± 0.11	6.05 ± 0.13	6.08 ± 0.24
high	5	20	1/5	5.37 ± 0.22	6.01 ± 0.13	6.33 ± 0.15	<u>5.66</u> ± 0.14	6.43 ± 0.16
high	5	30	1/5	5.03 ± 0.15	5.75 ± 0.12	5.85 ± 0.20	<u>5.58</u> ± 0.13	6.12 ± 0.21
high	5	40	1/5	4.82 ± 0.16	<u>5.47</u> ± 0.12	5.82 ± 0.12	5.75 ± 0.18	6.07 ± 0.25
high	5	50	1/5	4.69 ± 0.13	<u>5.23</u> ± 0.13	5.83 ± 0.19	5.70 ± 0.17	6.00 ± 0.15
categorical	1	20	1/20	5.34 ± 0.18	<u>5.34</u> ± 0.21	5.28 ± 0.29	5.59 ± 0.19	5.50 ± 0.14
categorical	1	30	1/20	<u>5.14</u> ± 0.10	5.17 ± 0.09	5.02 ± 0.16	5.43 ± 0.12	5.44 ± 0.22
categorical	1	40	1/20	4.90 ± 0.10	<u>4.92</u> ± 0.12	5.04 ± 0.19	5.24 ± 0.13	5.29 ± 0.23
categorical	1	50	1/20	4.67 ± 0.11	4.68 ± 0.10	4.84 ± 0.23	5.05 ± 0.20	4.83 ± 0.12
categorical	3	20	1/20	5.45 ± 0.13	<u>5.45</u> ± 0.16	5.52 ± 0.18	5.56 ± 0.17	5.66 ± 0.14
categorical	3	30	1/20	5.11 ± 0.14	<u>5.09</u> ± 0.08	5.08 ± 0.12	5.37 ± 0.19	5.42 ± 0.26
categorical	3	40	1/20	<u>4.90</u> ± 0.10	4.89 ± 0.11	5.02 ± 0.07	5.41 ± 0.24	5.49 ± 0.09
categorical	3	50	1/20	<u>4.65</u> ± 0.16	4.65 ± 0.11	5.00 ± 0.23	5.20 ± 0.19	5.33 ± 0.19
categorical	5	20	1/20	5.42 ± 0.19	<u>5.40</u> ± 0.24	5.39 ± 0.19	5.54 ± 0.17	5.74 ± 0.17
categorical	5	30	1/20	5.09 ± 0.13	<u>5.07</u> ± 0.11	5.05 ± 0.18	5.44 ± 0.15	5.26 ± 0.24
categorical	5	40	1/20	4.77 ± 0.16	<u>4.78</u> ± 0.13	5.03 ± 0.30	5.28 ± 0.18	5.16 ± 0.10
categorical	5	50	1/20	4.60 ± 0.09	4.57 ± 0.07	5.07 ± 0.29	5.08 ± 0.28	5.10 ± 0.21
numerical	1	20	1/20	5.36 ± 0.22	5.34 ± 0.25	5.30 ± 0.28	<u>5.31</u> ± 0.11	5.41 ± 0.23
numerical	1	30	1/20	<u>5.16</u> ± 0.18	5.17 ± 0.21	4.94 ± 0.17	5.24 ± 0.08	5.25 ± 0.27
numerical	1	40	1/20	5.03 ± 0.17	<u>4.92</u> ± 0.17	4.90 ± 0.14	5.13 ± 0.12	5.23 ± 0.09
numerical	1	50	1/20	<u>4.77</u> ± 0.13	4.72 ± 0.16	4.80 ± 0.13	5.07 ± 0.11	4.81 ± 0.15
numerical	3	20	1/20	<u>5.48</u> ± 0.28	5.42 ± 0.25	5.50 ± 0.19	5.49 ± 0.13	5.65 ± 0.14
numerical	3	30	1/20	5.24 ± 0.22	<u>5.17</u> ± 0.23	4.96 ± 0.18	5.23 ± 0.09	5.33 ± 0.22
numerical	3	40	1/20	5.08 ± 0.23	<u>4.92</u> ± 0.25	4.87 ± 0.18	5.31 ± 0.15	5.12 ± 0.12
numerical	3	50	1/20	4.84 ± 0.27	4.73 ± 0.21	4.87 ± 0.10	5.17 ± 0.12	5.13 ± 0.16
numerical	5	20	1/20	5.36 ± 0.15	<u>5.32</u> ± 0.13	5.36 ± 0.12	5.30 ± 0.14	5.55 ± 0.16
numerical	5	30	1/20	5.17 ± 0.19	<u>5.06</u> ± 0.19	4.96 ± 0.02	5.21 ± 0.16	5.15 ± 0.18
numerical	5	40	1/20	4.97 ± 0.13	4.84 ± 0.17	<u>4.89</u> ± 0.26	5.14 ± 0.12	5.05 ± 0.21
numerical	5	50	1/20	<u>4.73</u> ± 0.27	4.63 ± 0.19	4.91 ± 0.21	5.09 ± 0.05	5.09 ± 0.13

Table B.3: Expected Misclassification Cost per 100 instances for each setting - part 3

Setting				Deferral Strategy				
Noise	NE	DR	DA	CP	DS	L2D	RL	Random
low	1	20	1/20	5.56 ± 0.17	5.61 ± 0.18	<u>5.60</u> ± 0.34	5.69 ± 0.10	5.69 ± 0.28
low	1	30	1/20	<u>5.40</u> ± 0.16	5.40 ± 0.18	5.31 ± 0.11	5.49 ± 0.13	5.71 ± 0.14
low	1	40	1/20	<u>5.17</u> ± 0.16	5.16 ± 0.19	5.31 ± 0.18	5.35 ± 0.14	5.53 ± 0.09
low	1	50	1/20	<u>4.92</u> ± 0.13	4.89 ± 0.11	5.16 ± 0.30	5.25 ± 0.12	5.10 ± 0.17
low	3	20	1/20	5.68 ± 0.19	<u>5.71</u> ± 0.17	5.86 ± 0.18	5.82 ± 0.13	6.00 ± 0.19
low	3	30	1/20	5.34 ± 0.12	<u>5.32</u> ± 0.16	5.31 ± 0.16	5.49 ± 0.17	5.72 ± 0.17
low	3	40	1/20	<u>5.15</u> ± 0.11	5.09 ± 0.15	5.31 ± 0.24	5.61 ± 0.21	5.63 ± 0.07
low	3	50	1/20	<u>4.96</u> ± 0.17	4.91 ± 0.12	5.25 ± 0.24	5.64 ± 0.18	5.53 ± 0.18
low	5	20	1/20	5.59 ± 0.22	<u>5.63</u> ± 0.23	5.72 ± 0.21	5.65 ± 0.14	5.89 ± 0.14
low	5	30	1/20	<u>5.30</u> ± 0.19	5.30 ± 0.20	5.31 ± 0.23	5.48 ± 0.16	5.52 ± 0.11
low	5	40	1/20	5.03 ± 0.27	<u>5.04</u> ± 0.24	5.34 ± 0.40	5.38 ± 0.12	5.41 ± 0.03
low	5	50	1/20	<u>4.77</u> ± 0.18	4.76 ± 0.13	5.32 ± 0.36	5.34 ± 0.17	5.37 ± 0.17
medium	1	20	1/20	<u>5.76</u> ± 0.22	6.05 ± 0.15	6.06 ± 0.22	5.72 ± 0.11	6.24 ± 0.07
medium	1	30	1/20	<u>5.54</u> ± 0.17	5.81 ± 0.14	5.81 ± 0.07	5.49 ± 0.11	6.23 ± 0.10
medium	1	40	1/20	5.30 ± 0.17	<u>5.56</u> ± 0.15	5.81 ± 0.02	5.76 ± 0.14	6.11 ± 0.20
medium	1	50	1/20	5.01 ± 0.14	<u>5.28</u> ± 0.04	5.59 ± 0.31	5.63 ± 0.14	5.36 ± 0.07
medium	3	20	1/20	5.96 ± 0.23	6.17 ± 0.12	6.34 ± 0.07	<u>6.03</u> ± 0.06	6.45 ± 0.10
medium	3	30	1/20	<u>5.65</u> ± 0.14	5.88 ± 0.10	5.82 ± 0.06	5.63 ± 0.14	6.08 ± 0.21
medium	3	40	1/20	5.37 ± 0.15	<u>5.59</u> ± 0.17	5.78 ± 0.10	5.95 ± 0.17	6.07 ± 0.15
medium	3	50	1/20	5.13 ± 0.08	<u>5.31</u> ± 0.12	5.75 ± 0.16	5.83 ± 0.13	5.95 ± 0.16
medium	5	20	1/20	<u>5.90</u> ± 0.17	6.09 ± 0.12	6.21 ± 0.18	5.72 ± 0.16	6.47 ± 0.10
medium	5	30	1/20	<u>5.55</u> ± 0.11	5.82 ± 0.16	5.77 ± 0.13	5.50 ± 0.10	6.03 ± 0.11
medium	5	40	1/20	5.27 ± 0.09	<u>5.49</u> ± 0.14	5.74 ± 0.19	5.76 ± 0.14	5.98 ± 0.14
medium	5	50	1/20	5.02 ± 0.19	<u>5.23</u> ± 0.11	5.78 ± 0.21	5.58 ± 0.16	5.89 ± 0.10
high	1	20	1/20	5.33 ± 0.09	6.05 ± 0.19	6.10 ± 0.31	<u>5.69</u> ± 0.12	6.27 ± 0.14
high	1	30	1/20	5.03 ± 0.10	5.85 ± 0.09	5.85 ± 0.13	<u>5.58</u> ± 0.07	6.15 ± 0.08
high	1	40	1/20	4.84 ± 0.13	<u>5.50</u> ± 0.20	5.82 ± 0.13	5.70 ± 0.09	6.06 ± 0.20
high	1	50	1/20	4.63 ± 0.11	<u>5.22</u> ± 0.20	5.65 ± 0.19	5.73 ± 0.06	<u>5.19</u> ± 0.23
high	3	20	1/20	5.58 ± 0.11	6.27 ± 0.13	6.31 ± 0.18	<u>6.09</u> ± 0.11	6.45 ± 0.11
high	3	30	1/20	5.12 ± 0.10	5.86 ± 0.16	5.86 ± 0.12	<u>5.66</u> ± 0.22	6.06 ± 0.25
high	3	40	1/20	4.95 ± 0.04	<u>5.59</u> ± 0.21	5.84 ± 0.10	5.92 ± 0.16	6.04 ± 0.24
high	3	50	1/20	4.73 ± 0.10	<u>5.30</u> ± 0.21	5.79 ± 0.18	5.98 ± 0.18	6.06 ± 0.16
high	5	20	1/20	5.40 ± 0.11	6.18 ± 0.20	6.15 ± 0.15	<u>5.66</u> ± 0.18	6.46 ± 0.21
high	5	30	1/20	5.10 ± 0.20	5.84 ± 0.23	5.83 ± 0.15	<u>5.58</u> ± 0.14	6.04 ± 0.16
high	5	40	1/20	4.81 ± 0.13	<u>5.50</u> ± 0.24	5.87 ± 0.25	5.73 ± 0.09	6.07 ± 0.25
high	5	50	1/20	4.62 ± 0.18	<u>5.17</u> ± 0.17	5.84 ± 0.14	5.74 ± 0.10	5.96 ± 0.15
categorical	1	20	1/40	<u>5.32</u> ± 0.06	5.34 ± 0.04	5.23 ± 0.16	5.59 ± 0.19	5.41 ± 0.25
categorical	1	30	1/40	5.15 ± 0.09	<u>5.14</u> ± 0.07	5.13 ± 0.19	5.43 ± 0.12	5.37 ± 0.16
categorical	1	40	1/40	<u>4.86</u> ± 0.12	4.85 ± 0.09	5.02 ± 0.19	5.24 ± 0.13	5.33 ± 0.23
categorical	1	50	1/40	4.67 ± 0.12	4.66 ± 0.11	<u>4.67</u> ± 0.27	5.05 ± 0.20	5.00 ± 0.17
categorical	3	20	1/40	5.39 ± 0.15	5.39 ± 0.14	<u>5.39</u> ± 0.28	5.59 ± 0.15	5.69 ± 0.18
categorical	3	30	1/40	5.14 ± 0.11	<u>5.10</u> ± 0.10	5.08 ± 0.20	5.45 ± 0.17	5.32 ± 0.21
categorical	3	40	1/40	4.88 ± 0.15	4.90 ± 0.22	5.10 ± 0.23	5.39 ± 0.15	5.27 ± 0.25
categorical	3	50	1/40	4.69 ± 0.19	<u>4.72</u> ± 0.15	5.00 ± 0.24	5.21 ± 0.15	5.31 ± 0.19
categorical	5	20	1/40	5.36 ± 0.18	5.39 ± 0.19	<u>5.38</u> ± 0.26	5.59 ± 0.21	5.63 ± 0.21
categorical	5	30	1/40	<u>5.11</u> ± 0.14	5.13 ± 0.14	5.09 ± 0.18	5.44 ± 0.17	5.35 ± 0.16
categorical	5	40	1/40	4.89 ± 0.13	4.91 ± 0.15	5.06 ± 0.23	5.26 ± 0.11	5.15 ± 0.28
categorical	5	50	1/40	<u>4.64</u> ± 0.15	4.63 ± 0.12	4.96 ± 0.18	5.04 ± 0.27	5.12 ± 0.29

Table B.4: Expected Misclassification Cost per 100 instances for each setting - part 4

Setting				Deferral Strategy				
Noise	NE	DR	DA	CP	DS	L2D	RL	Random
numerical	1	20	1/40	5.31 ± 0.25	<u>5.30</u> ± 0.24	5.16 ± 0.17	5.31 ± 0.11	5.34 ± 0.14
numerical	1	30	1/40	5.09 ± 0.22	<u>5.05</u> ± 0.24	5.00 ± 0.11	5.24 ± 0.08	5.28 ± 0.22
numerical	1	40	1/40	<u>4.92</u> ± 0.18	4.82 ± 0.22	4.94 ± 0.22	5.13 ± 0.12	5.17 ± 0.26
numerical	1	50	1/40	4.69 ± 0.10	<u>4.67</u> ± 0.09	4.66 ± 0.16	5.07 ± 0.11	4.86 ± 0.27
numerical	3	20	1/40	5.40 ± 0.28	5.34 ± 0.21	<u>5.38</u> ± 0.19	5.41 ± 0.14	5.68 ± 0.13
numerical	3	30	1/40	5.18 ± 0.09	<u>5.04</u> ± 0.14	4.97 ± 0.17	5.23 ± 0.09	5.24 ± 0.21
numerical	3	40	1/40	5.00 ± 0.19	4.87 ± 0.26	<u>4.94</u> ± 0.22	5.32 ± 0.10	5.15 ± 0.32
numerical	3	50	1/40	<u>4.88</u> ± 0.17	4.77 ± 0.11	4.93 ± 0.24	5.15 ± 0.21	5.18 ± 0.20
numerical	5	20	1/40	5.29 ± 0.26	5.34 ± 0.17	<u>5.29</u> ± 0.27	5.28 ± 0.07	5.62 ± 0.22
numerical	5	30	1/40	5.17 ± 0.24	<u>5.06</u> ± 0.25	5.00 ± 0.19	5.22 ± 0.13	5.26 ± 0.14
numerical	5	40	1/40	4.97 ± 0.16	4.85 ± 0.16	<u>4.95</u> ± 0.19	5.18 ± 0.18	5.04 ± 0.13
numerical	5	50	1/40	<u>4.78</u> ± 0.20	4.69 ± 0.18	4.84 ± 0.25	5.03 ± 0.10	4.97 ± 0.17
low	1	20	1/40	<u>5.60</u> ± 0.12	5.60 ± 0.17	5.52 ± 0.29	5.69 ± 0.10	5.71 ± 0.07
low	1	30	1/40	<u>5.37</u> ± 0.12	5.33 ± 0.10	5.42 ± 0.20	5.49 ± 0.13	5.62 ± 0.28
low	1	40	1/40	<u>5.14</u> ± 0.18	5.13 ± 0.21	5.29 ± 0.18	5.35 ± 0.14	5.58 ± 0.15
low	1	50	1/40	<u>4.87</u> ± 0.22	4.86 ± 0.21	4.89 ± 0.45	5.25 ± 0.12	5.01 ± 0.22
low	3	20	1/40	5.65 ± 0.15	<u>5.69</u> ± 0.15	5.74 ± 0.23	5.85 ± 0.07	6.05 ± 0.14
low	3	30	1/40	5.39 ± 0.14	5.36 ± 0.19	<u>5.38</u> ± 0.13	5.52 ± 0.15	5.58 ± 0.23
low	3	40	1/40	<u>5.15</u> ± 0.09	5.12 ± 0.14	5.38 ± 0.23	5.57 ± 0.19	5.56 ± 0.18
low	3	50	1/40	<u>4.95</u> ± 0.20	4.95 ± 0.23	5.30 ± 0.30	5.60 ± 0.12	5.54 ± 0.21
low	5	20	1/40	5.57 ± 0.20	<u>5.63</u> ± 0.17	5.64 ± 0.17	5.69 ± 0.15	5.87 ± 0.19
low	5	30	1/40	5.29 ± 0.16	<u>5.29</u> ± 0.16	5.37 ± 0.11	5.45 ± 0.15	5.50 ± 0.15
low	5	40	1/40	<u>5.10</u> ± 0.14	5.04 ± 0.13	5.37 ± 0.24	5.41 ± 0.21	5.49 ± 0.20
low	5	50	1/40	<u>4.88</u> ± 0.19	4.87 ± 0.15	5.29 ± 0.22	5.33 ± 0.11	5.46 ± 0.20
medium	1	20	1/40	<u>5.80</u> ± 0.18	6.07 ± 0.05	5.99 ± 0.18	5.72 ± 0.11	6.22 ± 0.09
medium	1	30	1/40	<u>5.55</u> ± 0.21	5.77 ± 0.13	5.88 ± 0.07	5.49 ± 0.11	6.17 ± 0.06
medium	1	40	1/40	5.23 ± 0.23	<u>5.50</u> ± 0.25	5.74 ± 0.27	5.76 ± 0.14	6.06 ± 0.18
medium	1	50	1/40	4.98 ± 0.16	5.28 ± 0.26	<u>5.22</u> ± 0.47	5.63 ± 0.14	5.32 ± 0.17
medium	3	20	1/40	5.96 ± 0.18	6.16 ± 0.08	6.24 ± 0.16	<u>6.03</u> ± 0.09	6.46 ± 0.09
medium	3	30	1/40	<u>5.59</u> ± 0.23	5.83 ± 0.10	5.82 ± 0.09	5.51 ± 0.09	6.13 ± 0.19
medium	3	40	1/40	5.37 ± 0.26	<u>5.60</u> ± 0.09	5.84 ± 0.14	5.96 ± 0.11	6.08 ± 0.11
medium	3	50	1/40	5.19 ± 0.13	<u>5.37</u> ± 0.09	5.70 ± 0.15	5.93 ± 0.18	6.06 ± 0.08
medium	5	20	1/40	<u>5.88</u> ± 0.13	6.11 ± 0.09	6.10 ± 0.14	5.73 ± 0.16	6.44 ± 0.12
medium	5	30	1/40	<u>5.60</u> ± 0.19	5.83 ± 0.13	5.80 ± 0.09	5.52 ± 0.18	6.01 ± 0.13
medium	5	40	1/40	5.34 ± 0.21	<u>5.56</u> ± 0.29	5.80 ± 0.18	5.74 ± 0.16	5.97 ± 0.19
medium	5	50	1/40	5.14 ± 0.16	<u>5.31</u> ± 0.12	5.74 ± 0.12	5.57 ± 0.14	5.84 ± 0.06
high	1	20	1/40	5.35 ± 0.12	6.09 ± 0.16	6.04 ± 0.26	5.69 ± 0.12	6.29 ± 0.15
high	1	30	40	5.01 ± 0.10	5.78 ± 0.28	5.93 ± 0.14	<u>5.58</u> ± 0.07	6.16 ± 0.21
high	1	40	1/40	4.80 ± 0.16	5.43 ± 0.30	5.78 ± 0.19	5.70 ± 0.09	6.04 ± 0.13
high	1	50	40	4.63 ± 0.13	<u>5.12</u> ± 0.21	5.23 ± 0.52	5.73 ± 0.06	5.42 ± 0.11
high	3	20	1/40	5.50 ± 0.19	6.26 ± 0.07	6.22 ± 0.25	<u>6.11</u> ± 0.07	6.50 ± 0.15
high	3	30	40	5.12 ± 0.15	5.92 ± 0.12	5.92 ± 0.14	<u>5.62</u> ± 0.12	6.17 ± 0.21
high	3	40	1/40	4.98 ± 0.12	<u>5.62</u> ± 0.21	5.89 ± 0.16	5.96 ± 0.14	6.04 ± 0.12
high	3	50	40	4.82 ± 0.13	5.40 ± 0.11	5.78 ± 0.18	6.06 ± 0.12	6.08 ± 0.16
high	5	20	1/40	5.40 ± 0.17	6.14 ± 0.18	6.13 ± 0.17	<u>5.70</u> ± 0.14	6.47 ± 0.13
high	5	30	40	5.08 ± 0.23	5.84 ± 0.20	5.89 ± 0.15	<u>5.56</u> ± 0.11	6.08 ± 0.11
high	5	40	1/40	4.86 ± 0.14	<u>5.58</u> ± 0.28	5.89 ± 0.17	5.74 ± 0.14	5.90 ± 0.06
high	5	50	40	4.74 ± 0.19	<u>5.32</u> ± 0.22	5.78 ± 0.35	5.71 ± 0.10	6.00 ± 0.14

Table B.5: Expected Misclassification Cost per 100 instances for each setting - part 5

Setting		Null set predictions	Alpha
Noise	DR		
categorical	10	0.25% ± 0.10	0.00 ± 0.0010
categorical	20	0.80% ± 0.16	0.01 ± 0.0023
categorical	30	1.03% ± 0.33	0.01 ± 0.0043
categorical	40	1.40% ± 0.27	0.01 ± 0.0040
categorical	50	2.07% ± 0.29	0.02 ± 0.0041
numerical	10	5.82% ± 0.46	0.02 ± 0.0010
numerical	20	7.22% ± 0.42	0.02 ± 0.0018
numerical	30	8.18% ± 0.40	0.03 ± 0.0021
numerical	40	9.23% ± 0.83	0.03 ± 0.0048
numerical	50	10.38% ± 0.85	0.04 ± 0.0061
low	10	1.24% ± 0.40	0.01 ± 0.0029
low	20	2.06% ± 0.33	0.01 ± 0.0032
low	30	2.66% ± 0.41	0.02 ± 0.0037
low	40	3.24% ± 0.53	0.02 ± 0.0047
low	50	4.01% ± 0.35	0.03 ± 0.0029
medium	10	6.26% ± 0.45	0.02 ± 0.0016
medium	20	7.81% ± 0.53	0.02 ± 0.0018
medium	30	8.79% ± 0.20	0.03 ± 0.0022
medium	40	9.67% ± 0.33	0.03 ± 0.0047
medium	50	10.56% ± 0.38	0.04 ± 0.0046
high	10	7.50% ± 0.59	0.00 ± 0.0009
high	20	15.64% ± 0.33	0.02 ± 0.0009
high	30	16.79% ± 0.34	0.02 ± 0.0007
high	40	16.94% ± 0.28	0.02 ± 0.0006
high	50	17.21% ± 0.35	0.02 ± 0.0013

Table B.6: Percentage of null set predictions

