Explaining Predictive Uncertainty in Bayesian Neural Networks with Layer-Wise Relevance Propagation

- mi106f21-

Project Report

Casper Krogh Frydkjær
Lasse Østergaard
Mads Nørgaard Stenkær

Aalborg University
Department of Computer Science

Department of Computer Science
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Explaining Predictive Uncertainty in Bayesian Neural Networks with Layer-Wise Relevance Propagation

Theme:
Bayesian Deep Learning: Modeling Uncertainty in Medical Data

Project Period:
10. Semester 2020

Project Group:
mi106f21

Participants:
Casper Krogh Frydkjær
Lasse Østergaard
Mads Nørgaard Stenkær

Supervisor:
Thomas Dyhre Nielsen

Copies: 1

Page Numbers: 73 (181 with appendix)

Date of Completion: June 18, 2021

Abstract:

In this project, we investigate the use of BNNs for bridging the gap between expressing uncertainty in AI models and explaining the predictions of the models. The BNN used in this project is build with TensorFlow and TensorFlow Probability, and is trained through Variational Inference. Here, we propose the possibility of sampling models from a BNN and explaining these with LRP. By using the relevance scores calculated with LRP from multiple sampled models, we can consider the variance in relevance scores for each individual feature. We argue that this variance reflects the uncertainty in the predictions, and gives an insight into which features affect the uncertainty the most.
The LRP approach is evaluated by setting the values of features to 0, where we find that when setting features to 0, for features with low relevance scores and low variance, the predictions and uncertainty in these remain similar. On the other hand, when doing the same for features with high relevance scores and high variance in these, the predictions differ from the original predictions, often with a lower uncertainty. We conclude that there is a clear correlation between the variance in relevance scores and the uncertainty in predictions, and that our method is able to give an insight into which features contribute most to the uncertainty in predictions.

Summary

AI has proven itself in many different domains, and is now slowly finding its way into healthcare. Here, AI offers precise diagnoses on complex problems [1, 2], and the ability to express uncertainty and abstain from predictions by saying "i do not know", increases the trust in an AI model [3]. In addition to this, if an AI model is to be deployed in the EU, it is required by law that black box models such as neural networks can explain their predictions [4], when making decisions that affect people. In order to increase trustworthiness, we see great potential in combining explainability and uncertainty quantification. Also, to our knowledge, little research has been made about combining these two fields, with the goal of explaining the uncertainty in outputs of an AI model. Therefore, we set an initial problem statement with the goal of acquiring more knowledge about uncertainty quantification and explainability of AI models.

During research, we find and evaluate state of the art methods for quantifying uncertainty. Here, we choose to focus on explaining the uncertainty produced by Variational Inference, as this method uses probability distributions over the parameters of the model, which provides information about the uncertainty in the network. For explainability, we choose to focus on the LRP method, as it works on individual predictions and shows promise for the possibility of explaining uncertainty. Throughout this project, we use two different datasets for predicting sepsis. Here, we choose to use the data framing "Fixed Time To Onset", as this data framing yielded the best calibrated models. We also choose a time representation that use delta values, where the change in a feature is part of the dataset. This allowed for simpler models in terms of model architectures, which we deem fitting when testing the novel method we propose. Having these choices in mind, we present an idea of how to explain the features that affect the predictions of the model, in terms of model uncertainty.

Here, we get inspiration from Bykov et al. [5], who use LRP to explain predictions from a BNN, with the goal of determining the uncertainty in the explanation. We propose a method that follows this approach of sampling multiple different models from a BNN and explaining these models with LRP. However, we consider the variance in relevance scores for each sampled model, where we argue that the variance reflects the uncertainty of the predictions.

In order to ensure that the explanations of the uncertainty can be trusted, we see great importance in the model being well calibrated, and therefore conduct an experiment to investigate this. Here, we see that the BNN is generally well calibrated for both datasets, compared to its non stochastic counterpart. We also see that the model is better calibrated on the dataset with fewest missing values.

After this experiment, we experiment with different methods of using the uncertainty in the predictions to abstain from making a prediction, by saying "i do not know". Here, we see that the uncertainty of a model can be used to enhance the performance by abstaining. Following this, we show the results of the explanation method for uncertainties that we have proposed. Here, we see different patterns for different datasets, and that some features seem more important than others. Additionally, we see that when the variance in the predictions of the sampled models are high, this variance is also reflected in the relevance scores for the features.

After displaying the results, we evaluate the method. Here, we choose to set feature values to 0 for features with high relevance scores, and high variance in these. Based on the results, we see that setting a feature to 0, which has a high variance in its relevance scores, causes the variance in the predictions to decrease. However, this is not always the case for one of the datasets, which is significantly more sparse than the other. We suspect the different results are due to this sparsity, and that setting a feature to 0 reduces the provided information too much. Based on our results, we conclude that there is a clear correlation between variance in relevance scores and uncertainty in predictions, for the datasets and models used, and that our proposed model can give an insight into how the uncertainty in predictions is affected by the input features.

Preface

This report is written by three software students at Aalborg University, for the final semester project in a master's degree in software engineering.

During the project, experiments are conducted with two datasets containing Electronic Health Records from patients in risk of contracting sepsis. One dataset is publicly available from PhysioNet [6], and one is from the Cross-Tracks research project [7], which we have available through our task and employment at Enversion A/S. Before handling the dataset from the Cross-Tracks cohort, we have received the necessary training in data protection, and have signed a non-disclosure agreement. Additionally, all figures in the report are anonymized.

We want to thank Thomas Dyhre Nielsen for the supervision and sparring during the project. Also, we want to thank Enversion A/S for providing access to the Cross-Tracks cohort, and especially thanks to Bo Thiesson and Simon Lauritsen for further help and sparring during the project.

Aalborg University, June 18, 2021

Readers Guide

This report is targeted at students having experience with machine learning, who are studying on the last semester of their master's degree in software engineering.

Citations follow the IEEE citation style, where a cite is represented with a number within square brackets. The number refers to the numbers used in the bibliography.

All illustrations within the report are made by us.

When referencing other parts of the report, either 'Chapter' or 'Section' is used together with a number, which states which chapter or section within a chapter is referenced. For example, Section 4.3 refers to section 3 in chapter 4. When referring to parts in the appendix, the prefix is 'Appendix', followed by a letter and then by a number, which likewise refers to a chapter and a section.

For some experiments conducted during the project, many results are calculated. Therefore, to make the Appendix less cluttered, some results are moved into a Supplementary Appendix, which is included as an external document.

Contents

Part I

Problem Analysis

Chapter 1

Introduction

Artificial Intelligence (AI) is a rapidly evolving technology, and is starting to find its way into healthcare, where it shows great potential in improving healthcare by offering precise diagnoses [1]. The United Nations (UN) has set 17 sustainable development goals, which aim to improve the sustainability of the world, where one of these goals regards "Good health and well-being" [8]. Here, machine intelligence can be of great value in supporting the medical professionals in making the correct decisions [2]. As machine intelligence is moving into the field of healthcare, the trust in machine intelligence systems becomes increasingly vital for their viability [9].

The European Commission has released a report regarding the use of artificial intelligence in medicine and healthcare, stating that the trust of AI in healthcare is particularly important [1]. One way of establishing trust in a machine intelligence model for medical professionals, is to have the model reason about its uncertainty, with the possibility of providing the answer "i do not know" [3]. This helps medical professionals determine when to trust a machine intelligence model, which is especially important when regarding critical decisions [2].

In our pre-master thesis, *"Analyzing Calibration of State of the Art Deep Learning Architectures for Electronic Health Records"* [10], we explored calibration as a measure of uncertainty. Here, the notion of calibration relates to how well a model's predictions fit the overall frequency of an event. If we have 10 predictions, all with a confidence of 0.7 that patients contract sepsis, we expect that 7 out of 10 patients contract sepsis during their admission. Through experiments, we found that the architectures TCN and LSTM provided the overall lowest calibration scores on datasets regarding prediction of sepsis onset. While calibration is insufficient for capturing model uncertainty [2], Guo et al. discuss that a well calibrated model enhances the trustworthiness for its users [11], which means that it is still relevant to consider.

In order to establish more trustworthiness in the uncertainty of a prediction, it is also relevant to make a distinction between model uncertainty (epistemic uncertainty) and data uncertainty (aleatoric uncertainty) [12]. Here, epistemic uncertainty refers to the uncertainty inherent in the parameters of the model, and is caused by insufficient data. Epistemic uncertainty is pivotal in expressing the uncertainty of a model on out-of-distribution data, which might be the case with rare illnesses that has limited occurrences within a dataset [12]. Aleatoric on the other hand, refers to the uncertainty or randomness inherent in the data. Unlike epistemic uncertainty, aleatoric uncertainty cannot be reduced by gathering more data. A more in-depth coverage of aleatoric and epistemic uncertainty is found in Section 3.3. Making a distinction between these types of uncertainty makes it transparent whether the uncertainty is caused by inherent noise in the data or insufficient data, which in turn makes the predictor more trustworthy [13]. Based on these points, we want to explore other methods for capturing uncertainty, but also consider the calibration of the resulting model.

Even with uncertainty estimates from a neural network, it might be difficult for medical professionals to understand the meaning of these. Therefore it is also important to consider how the uncertainty of a machine intelligence system is communicated [3]. Expressing the uncertainty in a way that makes it clear why the model is uncertain, gives the medical professional a better foundation for decision making.

In 2018, the European Union updated its laws regarding General Data Protection Regulation (GDPR), to also include "right to explanation" [4]. This law requires, that if a decision is made by an algorithm, the user has the right to get an explanation for the decision. This law directly requires that black box models, such as neural networks, must be explainable when deployed in fields where a decision affects people. For example, if a person is rejected a loan by a machine intelligence system, the person has the right to know why.

We only find one paper working in the area of combining explanation and uncertainty [14], which itself states that little work has been done in this area. Explaining the uncertainty of a prediction provides new information in the context of what caused the uncertainty, in which we see great potential. For example, if there is high uncertainty in a specific prediction, this may have a correlation to some part of the input data. Explaining which features from the input data that cause this uncertainty, opens up for new possibilities of communicating the uncertainty. For example, uncertainty tied to a specific lab measurement might imply, for the medical professionals, that a new measurement is needed.

Based on the importance of expressing uncertainty in a medical setting, and the advantages from explainability of uncertainty, we construct the following initial problem statement:

How can we express uncertainty in a Neural Network to increase the trustworthiness of a prediction?

- How can we modify a neural network to make it possible to express the uncertainty of its predictions?
- How can the uncertainty of a prediction be explained, such that it is possible to determine which features in the input data cause the uncertainty?
- How can we make a distinction between epistemic and aleatoric uncertainty in the predictions of a model?

Chapter 2

Quantifying Uncertainty in Deep Learning

In this chapter, we present state of the art methods for quantifying uncertainty in deep learning. The motivation behind this chapter is to evaluate these methods, and determine which method is best suited for explaining the uncertainty contribution of features, as discussed in Chapter 1.

Our previous work consists of evaluating calibration in different state of the art deep learning architectures, as a measure of uncertainty [10]. Calibration is a way of determining whether the predicted probabilities of a network correlate to the real frequencies, which can be seen as a frequentist notion of uncertainty [15]. However, the calibration metrics alone do not capture the model uncertainty [2]. For traditional neural networks, the goal is to predict a single estimate of θ (the parameters of the network) [16]. Contrary to this, the Bayesian approach is to represent θ as random variables, which allows the possibility of expressing uncertainty in the parameters. If a model is able to express its uncertainty, it brings more insight into how certain the model is about the predicted probabilities, which would be of great value in the medical domain [17].
The remainder of this section is based on information from [16] and [18].

In Bayesian statistics, the principle is to express a level of belief in different outcomes. In Bayesian Neural Networks (BNNs), compared to neural networks, stochastic elements are introduced. For example, it is possible to express a BNN as a neural network where, instead of point estimate weights (shown on Figure 2.1), probability distributions over the weights are used (shown on Figure 2.2). These probability distributions express a belief in the different values of the weights.

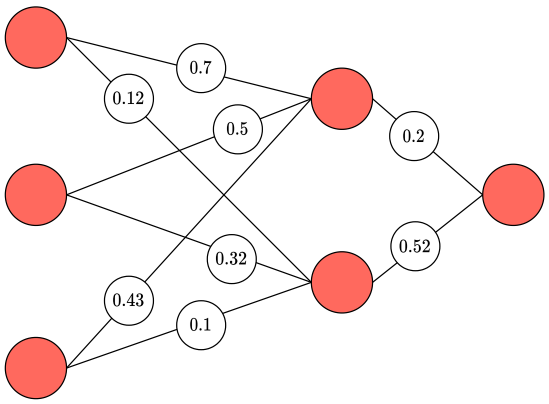


Figure 2.1: Neural Network with point estimate weights.

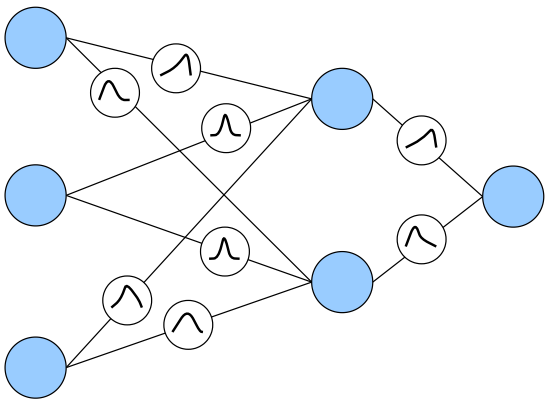


Figure 2.2: Bayesian Neural Network where stochastic elements express belief in different values of the weights.

One way to obtain these probability distributions, is to compute the posterior distribution $p(\theta|D)$ using Bayes Rule (see Equation 2.1a). Here, $p(D|\theta)$ is known as the likelihood, which represents the similarity between the parameters and data. $p(\theta)$ is the prior and represents knowledge about outcomes before evidence is observed about the outcomes. The prior can for example be expert knowledge from a doctor, that some disease occurs more often than others, or common knowledge that a coin flip has a probability around 0.5 for either outcome. $p(D)$ is the probability of the data, which can be calculated as the marginalization of θ from $p(D, \theta)$, which can be seen in Equation 2.1b.

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{2.1a}$$

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{\int_\theta p(D|\theta)p(\theta)d\theta} \tag{2.1b}$$

Ideally, the true posterior distribution can be computed as seen in Equation 2.1a. However, the marginalization of $\int_\theta p(D|\theta)p(\theta)d\theta$ is often intractable to compute, as it scales with the number of parameters in the network [19]. As a solution to this intractability, different methods exist for approximating the posterior distribution. After obtaining an approximated distribution over the parameters, it is possible to use this approximation to compute the probability distribution of a model's output, which can be seen in Equation 2.2. Here, y is a prediction given a data point x and a dataset D. This distribution also expresses the model's uncertainty, which will be covered more in-depth in Section 3.3.

$$p(y|x, D) = \int_\theta p(y|x, \theta)p(\theta|D)d\theta \tag{2.2}$$

In the following sections, we explore the predominant state of the art methods for computing the approximated posterior.

2.1 Deep Ensembles

Deep Ensembles is a method for approximating the intractable posterior by training an ensemble of N models [15]. By giving these models different random initializations, they follow different trajectories while training, which results in the models being able to explore different minima of the function space [20]. This creates diversity in the predictions, which can be used to express the uncertainty of the ensemble. Lakshminarayanan et al. [15] suggest that for classification, the predictions of an ensemble can be averaged by $p(y|x) = \frac{1}{M}\sum_{m=1}^{M} p_{\theta_m}(y|x, \theta_m)$. The diversity in predictions can then be found, by computing how much the predictions from the models in the ensemble disagree with the average prediction, where this disagreement represents the uncertainty in the predictions. More precisely, this disagreement can be computed by: $\sum_{m=1}^{M} KL(p_{\theta_m}(y|x)||p_E(y|x))$, where $p_E(y|x)$ is the average prediction of the ensemble and KL is the KL-divergence. Comparing Deep Ensembles to Bayesian methods, Dusenberry et al. find that their Bayesian models achieved similar performance metrics, while having significantly less parameters than the Deep Ensemble. This makes the Deep Ensemble method more computationally heavy, which also makes it a less attractive choice for clinical deployment [2].

2.2 Markov Chain Monte Carlo

As described in [17], Markov Chain Monte Carlo (MCMC) is a family of algorithms used for approximating the true posterior $p(\theta|D)$ by using a Markov Chain. The Metropolis-Hasting algorithm is an algorithm within the family of MCMC algorithms. In the Metropolis-Hasting algorithm, the Markov Chain is built by using an initial sample and a candidate distribution to draw a new subsequent sample x_n. If the sample x_n complies with the acceptance rules, a new sample x_{n+1} is sampled from the candidate distribution. By generating enough of these samples, the MCMC algorithm is guaranteed to produce an asymptotically exact distribution with respect to the true posterior distribution, if doing enough iterations of the MCMC algorithm [21]. While MCMC provides a good approximation, the computation time is heavily dependent on the size of the dataset, which can lead to problems on bigger datasets. In addition to this, the method is also more computationally expensive, compared to Variational Inference [18].

2.3 Variational Inference

Variational Inference is another method for approximating the true posterior. In Variational Inference, the parameters θ of a network consist of probability distributions, rather than point estimate values [18]. The parameters θ are trained to be as close as possible to the true distribution $p(\theta|D)$. When generating a prediction with the network, the weights are sampled from the probability distributions θ. Several predictions can be generated this way, where the diversity in the predictions represents the uncertainty, similar to the Deep Ensemble approach. An in-depth description of the method follows in Section 3.1. While Variational Inference is less computationally heavy than MCMC, Variational Inference does not offer the guarantee of an asymptotically correct approximation [18].

2.4 Monte Carlo Dropout

Monte Carlo Dropout (MC Dropout) is another method for approximating the true posterior $p(\theta|D)$. MC Dropout uses the regularization method of dropout, where random neurons are dropped during training, meaning that only a subset of the neurons in the network are propagating information. For MC Dropout, neurons are also dropped during inference, which makes it possible to simulate an ensemble of networks, which functions as a Bayesian approximater. Neurons in the network are dropped by drawing a probability from a Bernoulli distribution, deciding whether a neuron should be dropped or not [22]. MC Dropout can be seen as a more simple alternative compared to MCMC and Variational Inference, while being less computationally heavy than Deep Ensembles, as it only requires the training of a single model.

2.5 Evaluation

As mentioned in Chapter 1, we want to explore how to explain the uncertainty and which features cause it. In the previous sections, we have examined established methods for quantifying the

uncertainty of a model. In this section, we want to discuss the viability of these methods for explaining which features affect the uncertainty of a prediction.

When using Deep Ensembles, the uncertainty quantification is only based on the output of the models. Therefore, limited information is available to analyze what causes the uncertainty, making it difficult to connect the uncertainty to specific features.

For MC Dropout, we have a similar problem. Here, predictions change based on which neurons are dropped, but uncertainty is still based on the output, which means that limited information is available.

Based on this, we assert that explaining the uncertainty relative to specific features, for both Deep Ensembles and MC Dropout, is infeasible. Therefore, we do not choose Deep Ensembles or MC Dropout as the method for quantifying uncertainty with the objective of explaining it.

This leaves Variational Inference and MCMC as the remaining candidates. Both of these methods make use of probability distributions over the parameters of the model. This provides information about uncertainties on the entire network, as opposed to Deep Ensembles or MC Dropout. Therefore, we believe these two methods to be more viable and consistent methods to base explanations on. When comparing Variational Inference and MCMC, the big difference lies in computation time and the potential of the approximation. MCMC produces an asymptotically exact distribution, but scales poorly with the number of features and dataset size. Variational Inference does not provide this guarantee of an asymptotically exact distribution, but it scales better with features and dataset size. Based on the sizes of the datasets at our disposal, which consist of $36,232$ data points with 50 features and $39,789$ data points with 75 features, covered in Chapter 4, we believe that the best suited method for quantifying and explaining uncertainty in this project is Variational Inference.

Chapter 3

Theory

In this chapter, we present theory related to Variational Inference and uncertainty quantification, as this is the method we choose for quantifying uncertainty, described in Chapter 2. This chapter builds on top of Chapter 4 *"Theory"* from our pre-master thesis [10], which can be found in Appendix F.

3.1 Variational Inference

This section is based on information from [16] and [18].
In Variational Inference, the principle is to find an approximate probability distribution $q(\theta)$ that is sufficiently similar to the true intractable posterior probability $p(\theta|D)$, to be used as a substitute for it. The similarity of the two probability distributions can be calculated using the Kullback-Leibler (KL) divergence:

$$\mathrm{KL}(q(\theta)||p(\theta|D)) = \int_\theta q(\theta) \log \frac{q(\theta)}{p(\theta|D)} d\theta = \mathbb{E}_q \left[\log \frac{q(\theta)}{p(\theta|D)} \right] \tag{3.1}$$

The KL divergence is always non-negative, and is 0 iff $p(\theta|D) = q(\theta)$. $q(\theta)$, the candidate probability distribution, belongs to a family of probability distributions, $\mathbb{Q}$, that can approximate the true posterior probability. Therefore, finding the distribution $q \in \mathbb{Q}$ that minimizes KL divergence, can be seen as an optimization problem:

$$q^*(\theta) = \arg\min_{q(\theta)\in\mathbb{Q}} \mathrm{KL}(q(\theta)||p(\theta|D)) \tag{3.2}$$

However, when using KL divergence to measure the similarity between $p(\theta|D)$ and $q(\theta)$, the intractable true posterior is needed. Moreover, if the true posterior is already known, there is no need for finding the approximate posterior probability $q(\theta)$. Therefore, the evidence lower bound (ELBO) is used in the optimization problem instead of KL divergence.

To derive ELBO, we start by multiplying both the numerator and denominator of the KL divergence by $p(D)$, as seen on Equation 3.3a. This results in the joint probability $p(\theta, D)$ in the denominator, while the $p(D)$ in the numerator can be moved outside the expectation term, as it is constant wrt. q (Equation 3.3b).

$$\mathrm{KL}(q(\theta)||p(\theta|D)) = \mathbb{E}_q \left[\log \frac{q(\theta) \cdot p(D)}{p(\theta|D) \cdot p(D)} \right] \tag{3.3a}$$

$$= \mathbb{E}_q \left[\log \frac{q(\theta)}{p(\theta, D)} \right] + \log p(D) \tag{3.3b}$$

Isolating the constant $\log p(D)$, we get Equation 3.4a with the constant term, KL divergence term and the ELBO term, where the ELBO term covers the expectation term shown in Equation 3.4b.

$$\log p(D) = \mathrm{KL}(q(\theta)||p(\theta|D)) + \mathrm{ELBO}(q) \tag{3.4a}$$

$$\mathrm{ELBO}(q) = -\mathbb{E}_q\left[\log \frac{q(\theta)}{p(\theta, D)}\right] \tag{3.4b}$$

As the name implies, ELBO is a lower bound on the logarithm of the evidence. This is apparent in Equation 3.4a, where the constant equals the sum of the non-negative KL divergence and the ELBO.

With the knowledge that the KL divergence is non-negative and $\log p(D)$ is constant, it is apparent that the KL divergence decreases as ELBO increases. Therefore, the optimization problem where ELBO is maximized has the effect of minimizing KL divergence, and can be calculated without the intractable true posterior. However, because it is intractable to calculate the constant, we have the problem that it is intractable to say to which degree an approximate posterior is similar to the true posterior. However, it can still be used to express whether one approximate posterior is more similar to the true posterior than others, which means that it can be used to find out which $q \in \mathbb{Q}$ fits the true posterior the best.

3.2 Mean-Field Family

One possible family to use for the approximate posterior is the mean-field family. In the mean-field family, the approximate posterior is the product of the probability distributions of the latent variables, as shown in Equation 3.5. This means that it is assumed that the latent variables are not conditioned on the other latent variables. Additionally, each latent variable has its own set of variational parameters, which are updated independently of other latent variables [18].

$$q(\theta) = \prod_{j=1}^{|\theta|} q_j(\theta_j) \tag{3.5}$$

3.3 Aleatoric & Epistemic Uncertainty

In Section 5.1 *"Aleatoric & Epistemic Uncertainty"* in our pre-master thesis [10], we make a general description of aleatoric and epistemic uncertainty. In this section, we build on top of this with a more in-depth description.

In traditional neural networks, the output is often in the form of a predictive probability, that given an input $\boldsymbol{x}$, the output is $\boldsymbol{y}$. This predictive probability, $p(\boldsymbol{y}|\boldsymbol{x}, \theta)$, is a measure of the aleatoric uncertainty. This is the uncertainty stemming from irreducible noise in the data collection process. Epistemic uncertainty on the other hand, is sometimes called reducible uncertainty, as it originates in missing knowledge that can be reduced by introducing more data. This type of uncertainty is a result of uncertainty about the model, and is expressed as $p(\theta|D)$, which corresponds to the posterior distribution, representing a level of belief in the parameters of the model [12, 17]. Therefore, one way of quantifying epistemic uncertainty is to sample N models from the posterior

of a BNN, and use the disagreement of the sampled models as a measure of epistemic uncertainty. A high level of disagreement is caused by a high variance in the sampled parameters, indicating a high epistemic uncertainty [2].

In Equation 3.6, we show the marginalized predictive probability distribution, $p(y|x, D)$. $p(y|x, D)$ is an overall predictive distribution that considers both aleatoric and epistemic uncertainty [2]. This means that without additional effort, $p(y|x, D)$ cannot show aleatoric and epistemic uncertainty separately.

$$p(y|x, D) = \int_{\theta} p(y|x, \theta)p(\theta|D)d\theta \tag{3.6}$$

To get a more visual insight into what aleatoric and epistemic uncertainty is, a representation of a linear regression example, visualizing both types of uncertainty, can be seen on Figure 3.1. On the figure, it can be seen that the left group of data points is scattered with some lying close to the underlying function (shown with the dotted line) and some further away from it. In this group of data points, the aleatoric uncertainty is high. In the group of data points on the right side, the aleatoric uncertainty is lower, as the data points are closer to the underlying function. The epistemic uncertainty is high when it is difficult to decide whether the parameters of a model are correct, and thereby whether the function of the model follows the underlying function of the data. This is the case for areas where no or few data points are available.

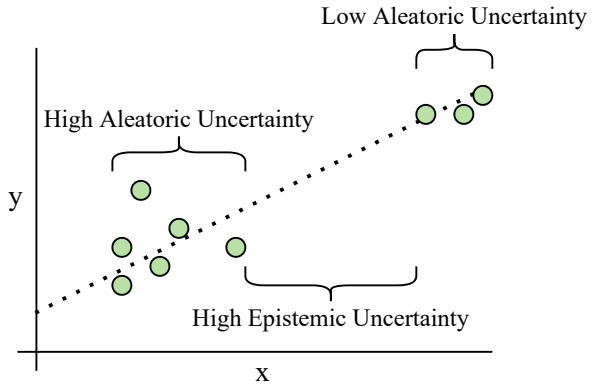


Figure 3.1: Linear regression example with labels showing the aleatoric and epistemic uncertainty.

3.4 Binary Cross Entropy

In this section, we cover the loss function we use when training networks throughout this project. Since we are working with a binary classification problem, we have chosen to use the Binary Cross-Entropy (BCE) loss function. BCE uses the logarithm of a model's confidence for the ground truth label as the loss, given by:

$$BCE = -\frac{1}{|D|} \sum_{(x,y)\in D} (y \cdot log(f(x)) + (1 - y) \cdot log(1 - f(x))) \tag{3.7}$$

Here, the logarithm used is the natural logarithm [23], and $f(x)$ is the predicted probability from the model, given input x from dataset D. Since the logarithm of probabilities results in negative values, the total sum is negated when used as a loss value for the network, which

is then minimized during training. When using BCE, the loss follows a negative logaritmic curve, meaning that the loss grows rapidly as the predicted probability approaches the wrong label value. This means that if $y = 1$, the loss is very high for a prediction near 0, and vice versa. Additionally, note that the negative logarithm of probabilities used by BCE is equivalent to negative log likelihood, which is particularly useful when implementing our BNN in Section 7.1 [16].

Chapter 4

Description of Datasets

This chapter builds on top of Chapter 2 *"Description of Datasets"* from our pre-master thesis [10], which can be found in Appendix C.

For our pre-master thesis [10], we used two different datasets, which we also use in this project. The first is a publicly available dataset from PhysioNet (DP) [6] for predicting sepsis in patients admitted to an intensive care unit (ICU). The second dataset consists of preprocessed data from the research project Cross-Tracks (DCT) [7], which is accessible to us through our employment and tasks at Enversion A/S (Enversion). Before handling the data, we have received appropriate data protection training, and have signed a non-disclosure agreement [10]. We denote the PhysioNet and Cross-Tracks datasets as DP and DCT, respectively.

Both DP and DCT contains Electronic Health Records (EHRs) and are therefore structured as time series data, which is described in Appendix C. At Enversion, they have previously been working with time series data [24, 25], but in their later work, they have experimented with different methods for preprocessing and framing the data, which is discussed in [26]. In the preprocessing method discussed in the paper, the temporal aspect of the data is encoded as delta values in the dataset of DCT, which is described in Section 4.2. This allows non sequential models to have a notion of time incorporated in the data. Because of the novelty of explaining uncertainty, we find this time representation fitting, as it allows for a simpler BNN in terms of model architecture. We therefore choose to use the delta representation covered in Section 4.2, on both DP and DCT. The preprocessing involves the calculation of delta values between two time periods, and we therefore denote it with a Δ prefix, followed by the name of the dataset (Eg. ΔDP or ΔDCT). In the remainder of this chapter, we want to provide additional information about DP and DCT, discuss how delta preprocessing is applied to DP and DCT, and which data framing method is used.

4.1 Expanding On Dataset Description

In this section, we provide additional information about the two datasets, which was not included in the description in Appendix C.

4.1.1 PhysioNet

The PhysioNet dataset is collected from two different hospitals, namely the Beth Israel Deaconess Medical Center (DP-A) and the Emory University Hospital (DP-B) [6]. As the two datasets are from two different hospitals, there are inherent differences in the datasets, as described in Appendix C. This means that when training and evaluating a model on DP, this is done on the two datasets separately. Something that was not mentioned in Appendix C, is that if an admission is

less than eight hours, the admission is not included in the dataset. In addition to this, if sepsis onset happens in the first four hours of an admission, this admission will not be included.

4.1.2 Cross-Tracks Cohort

Through our employment at Enversion we have access to DCT, which is a subset of the Cross-Tracks Cohort consisting of data from sepsis patients, which has been preprocessed by Enversion. The data regards inpatients of 18 years or older at the Horsens Regional Hospital, collected in the time period 1st of September 2012 to the 31st of December 2018.

4.2 Delta Preprocessing

In this section, we describe the delta preprocessing method, which follows the method described by Lauritsen et al. [26]. Note that all admissions shorter than the observation window of 12 hours (the period of data being considered when making a prediction for sepsis) are excluded. Delta preprocessing is done by applying the following steps to laboratory values and vital sign parameters:

1. The observation window of 12 hours is split into two periods of six hours, where values for each feature are averaged, meaning that there is only one value for each feature in each of the two periods.
2. If no value is available for a specific feature in one of the periods, forward and backward imputation is applied. This means that if there is no value for example for 'FiO2' in the first period, but the second period has one, the value from the second period is copied into the first, and vice versa.
3. If no value is available for a feature in either of the two periods, 0 is inserted in both.
4. Delta values are calculated as the differences between the averaged values in the first and the second period.
5. The averaged values from the second period are used together with the delta values as a data point in the Δdataset.
6. The sepsis label is set to 1 if the patient gets sepsis within the prediction window of 12 hours, and 0 otherwise.

A representation of a single data point can be seen in Table 4.1.

Vital Signs Δ	Laboratory Values Δ	Vital Signs 2. Period	Laboratory Values 2. Period	Demo-graphics	Sepsis Label
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Table 4.1: Representation of ΔDP. ΔDCT is equivalent, but without demographics. $\cdots$ represents the values of each feature under each category.

For DP, demographics are available together with the laboratory values and vital sign parameters, whereas only laboratory values and vital sign parameters are available in DCT. For the demographics in DP, delta value preprocessing is not applied, as most of them stay the same

throughout the admission. These features are age, gender, Unit1 and Unit2, which represents at which ICU the patient is admitted, and HospAdmTime, representing the number of hours the patient is admitted at a hospital before being admitted at an ICU. Instead of delta values, values for the features in demographics are extracted from the last time step in the observation window, which means that the last feature in demographics, ICU length-of-stay (ICULOS), represents the admission time of the patient when a prediction is made.

Gender from demographics in DP is originally encoded as a value of 0 for female and 1 for male. We see this as being problematic, especially if gender plays a significant role in the uncertainty of a prediction. This means that the gender feature can be asserted as the contributor to the uncertainty, but a specific gender cannot. Therefore, we encode it using a one-hot encoding, as shown in Table 4.2, such that no prior assumptions are made about the priority and order of the genders.

Male	Female
1	0
0	1

Table 4.2: One-hot encoding of gender for male and female.

4.3 Dataset Framing

In [26], Lauritsen et al. discuss multiple dataset framing methods: "Fixed Time To Onset", "Sliding Window", "Sliding Window With Dynamic Inclusion", and "On Clinical Demand". The framing methods are represented on Figure 4.1.

"On Clinical Demand" extracts samples when an Early Warning Score (EWS) assessment is made by a clinical staff. If the patient is sepsis positive within the 12 hour prediction window after the EWS assessment, the sample is positive. This data framing is only available for DCT, as there are no EWS assessments in DP.

With "Fixed Time To Onset", a model is trained to predict sepsis onset at a fixed time before it occurs for sepsis positive patients. If the patient is sepsis negative, a random point in time from the admission is used.

The last two framing methods, "Sliding Window" and "Sliding Window With Dynamic Inclusion", extract consecutive observation windows from the original health records. For "Sliding Window With Dynamic Inclusion", observation windows where the SOFA score is 0 are excluded, while for "Sliding Window" samples are extracted from the entire admission. Both methods result in models that can be used periodically during the admission of a patient.

4.3.1 Choice of Data Framing

Lauritsen et al. [26] mentions that "Model framing must reflect the expected clinical environment", and that this is important when considering risk prediction models for clinical settings.

All the data framing methods mentioned above are applicable on the DCT dataset. Here, we argue that the data framings "Sliding Window", "Sliding Window With Dynamic Inclusion" and

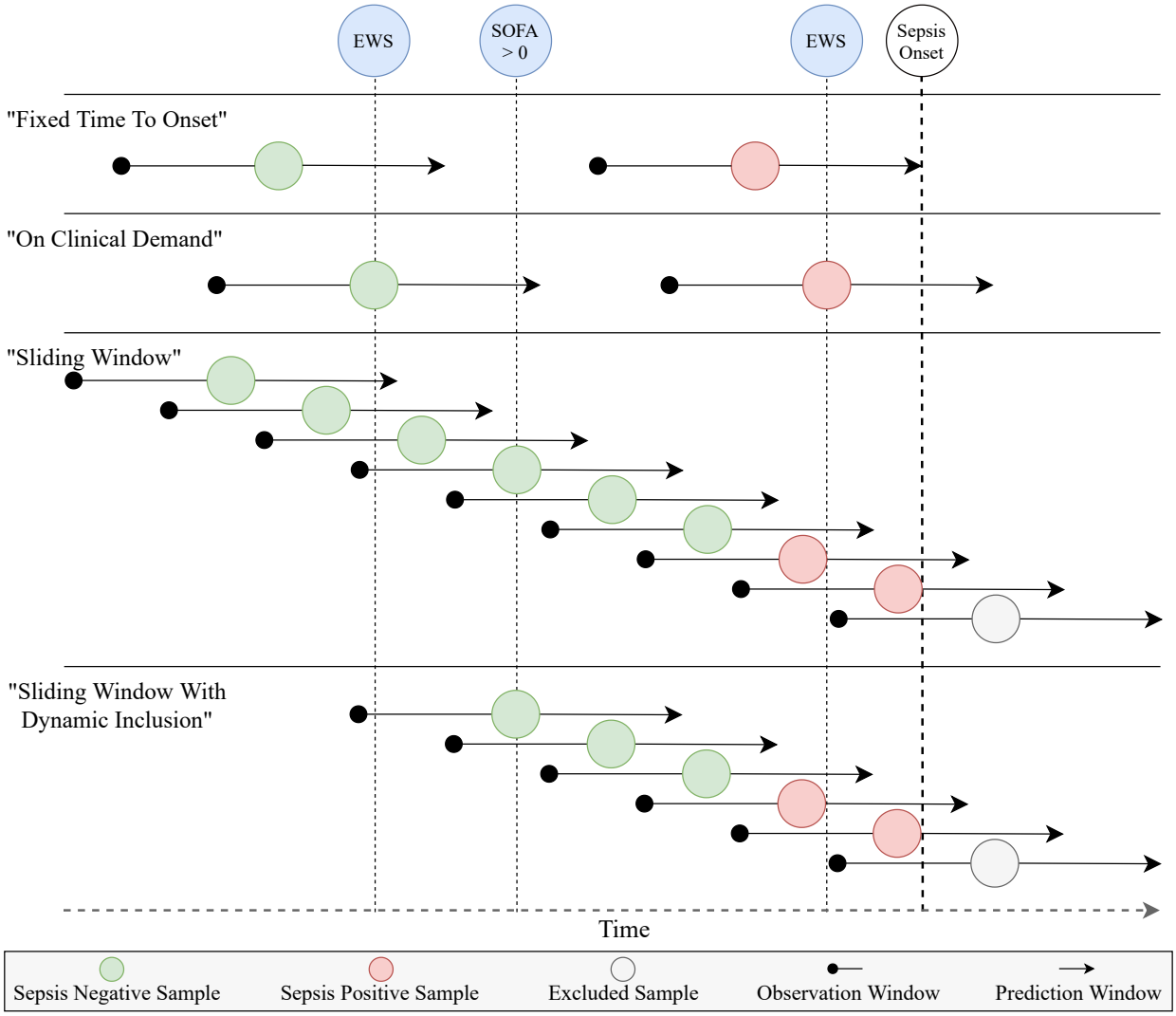


Figure 4.1: Representation of data framing methods, based on Figure 2 from [26].

"On Clinical Demand" best fits the real world scenario of a model for predicting sepsis. A model trained on "Sliding Window" and "Sliding Window With Dynamic Inclusion" should be able to do a sepsis prediction at any time throughout a patient's admission. For "On Clinical Demand", the model is trained to do predictions when a medical professional suspects an increase in SOFA score (see Appendix E). This would be an ideal time to predict sepsis, making "On Clinical Demand" fit a real world scenario as well. For "Fixed Time To Onset", the model is trained to predict positive sepsis at a specific time before onset. This arguably makes it less suited for a real world scenario, as the time to onset is unknown and might not fit with the actual prediction times.

Based on this, we would like to use "On Clinical Demand" for DCT and one of the "Sliding Window" framings for DP, since "On Clinical Demand" is not available for DP. However, through initial experiments, we find that we are unable to get satisfactory results with these data framings. This is described further in Section 9.2. Because of this, we try experimenting with the "Fixed Time To Onset" framing. While not being the best framing for reflecting the expected environment of the model, it is still realistic, and we achieve much better performance and calibration when using this framing. Additionally, this improvement gives a better foundation for experimenting with

uncertainty explanation, as we get more trustworthy predictions. Because of this, we choose to use "Fixed Time To Onset" as the data framing for both DP and DCT.

4.4 Sample Count

When using the data framing "Fixed Time To Onset" on ΔDP, the sample count is almost unchanged from the original dataset. For some patients, the admission length is too short for the 12 hour observation and prediction windows, which results in some admissions being excluded. This results in a positive percentage of 8.75% for ΔDP-A and 5.73% for ΔDP-B, as seen in Table 4.3. This is close to the original dataset, which contains 8.8% sepsis positve cases for DP-A and 5.8% sepsis positive cases for DP-B (Appendix C).

For the "Fixed Time To Onset" data framing on ΔDCT, the positive to negative sample ratio is very similar, with 5.75% positive and 94.25% negative samples. With the applied data framings, the positive and negative ratios are 1 : 16.4 for ΔDCT, and 1 : 10.4 and 1 : 16.4 for ΔDP-A and ΔDP-B respectively.

	Set A	Set B	Combination
Total	20,114	19,675	39,789
Negative	18,354	18,548	36,902
Positive	1,760	1,127	2,887
Negative percentage	91.25%	94.27%	92.74%
Positive percentage	8.75%	5.73%	7.26%

Table 4.3: Count of samples for the two datasets in ΔDP-A and ΔDP-B, and the combination of the two, using the "Fixed Time To Onset" dataset framing.

Total	36,232
Negative	34,149
Positive	2,083
Negative percentage	94.25%
Positive percentage	5.75%

Table 4.4: Count of samples for the ΔDCT dataset, using the "Fixed Time To Onset" dataset framing.

Chapter 5

Explanation Method

In this chapter, we explore state of the art methods for explanation of neural network models. The idea behind explanation methods is to explain the reasoning behind the predictions of a model, more specifically which features in the data were used to arrive at a given prediction [27]. This is very similar to our idea, described in the Introduction, Chapter 1, where we want to explain which features are the cause of the uncertainty in a given prediction. Therefore, we explore explanation methods to draw inspiration for the design of our own method.

5.1 Overview

In this section, we give a brief overview over different categories of explanation methods, mainly based on the paper by Samek et al. [27], and choose the method we find most promising for explaining the uncertainty of predictions relative to the input features. Many different methods exist, but for brevity we group them into the following four categories of methods:

- **Perturbation-Based Methods** work by adjusting the input to the model and observing how it affects the prediction. The intuition is that if adjusting a specific feature changes the prediction, that feature likely has a high relevance for the prediction. An example of this is the occlusion method used in [28], where part of the input image is covered by a grey square, to see how it affects classification.
- **Surrogate Methods** work by approximating the complex model by using a simpler surrogate function, that is easily explainable. This is done by training the surrogate function on some or all of the same data as the complex model, using the corresponding predictions from the complex model as target values. By fitting the surrogate function to the complex model, it can be used to explain the predictions of the model. An example of a surrogate method is LIME [29].
- **Propagation-Based Methods** work directly on the structure of the model, by backpropagating the predicted output, or some relevance score of the output, back to the input features. This gives a value for each feature, indicating its relevance for the prediction. An example of a propagation-based approach is Layer-wise Relevance Propagation (LRP) [30].
- **Meta Explanation Methods** work by aggregating individual explanations for a model. This means that a separate explanation method is used to get the explanations for individual predictions. These explanations are then analyzed to find patterns in the predictions of the model, in an attempt to better understand the general behavior of the model. An example of a meta explanation method is SpRAy [31], which clusters individual heatmap explanation of images.

Since meta explanation methods focus on aggregating explanations and finding general patterns, they do not fit our goals, as we want to focus on individual predictions.

For surrogate methods, the idea is to approximate the actual model with a simple, interpretable model. Since this model is only an approximation, it introduces some level of uncertainty. We see this as a big disadvantage, since we want to base our explanation on the uncertainty in predictions, and using a surrogate method adds a new layer of uncertainty.

The method we find most promising is a propagation-based method, namely the LRP method. LRP works by propagating a relevance score from the output back to the features of the input, by following one or more relevance propagation rules. We want to consider this concept for our method, and explore whether LRP can be used to propagate the uncertainty of a prediction back to the features causing it.

Perturbation-based methods are also interesting, as it is rather simple to adjust the input and observe how it affects the uncertainty of the prediction. However, perturbation methods are not computationally efficient, as the model must evaluate each perturbation of the input, which can grow to a large number of runs based on the number of features. In contrast, LRP requires one forward pass of the model to get the prediction, followed by one backwards pass to propagate the relevance scores. Therefore, we choose to focus on LRP when designing our method.

5.2 Layer-wise Relevance Propagation

This section is based on information from [30].
As mentioned, LRP works by propagating a relevance score from the model output to the input features. This gives each feature a score based on how large an influence it had on the output value. A positive relevance score means that a feature had a positive influence on the output, increasing its value, whereas a negative relevance score means that a feature had negative influence, decreasing the value of the output. The propagation of relevance is done in a layer-wise manner and follows a conservation property, such that the sum of relevance scores for each layer is identical, and is equal to the model output. However, this is not always true in practice, as bias neurons and some of the LRP rules can absorb part of the relevance from a layer. The process is visualized on Figure 5.1, where it can be seen that some neurons receive a greater part of the relevance from the following layer than other neurons. Here, the second input feature had the highest relevance score, and thus the highest impact on the prediction.

The amount of relevance each neuron receives is based on its contribution to the outputs of the neurons in the following layer, which is computed using one of the LRP rules. The general idea can be shown with the generic rule:

$$R_j = \sum_k \frac{z_{jk}}{\sum_j z_{jk}} R_k, \tag{5.1}$$

where z_{jk} denotes the contribution from neuron j to neuron k in the following layer. R_j denotes the relevance score for neuron j, and is computed by taking the sum of relevance scores R_k from neurons k in the following layer, that are connected to neuron j. Each relevance score R_k is scaled by the contribution from neuron j, divided by the total contribution received by neuron k. For

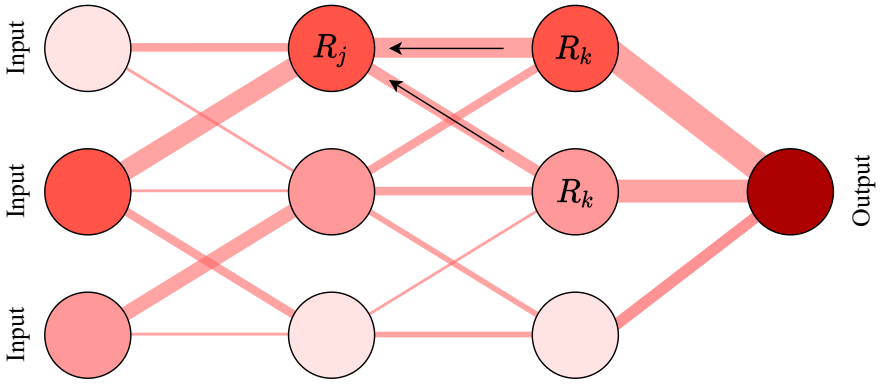


Figure 5.1: Propagation of relevance scores through a neural network.

example, if neuron j contributes 30% of the total value of neuron k, it receives 30% of the relevance score of neuron k.
The contribution z_{jk} is computed differently based on the LRP rule used for the specific layer, but is usually based on the activation of neuron j and the weight between neuron j and neuron k.
To get an insight into the variety of LRP rules, consider the three rules shown below. The first is the basic rule, LRP-0:

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k \tag{5.2}$$

This shows how the contribution of neurons are calculated based on activations and weights. Note that the 0 in the sum of the denominator refers to an additional neuron representing the bias, with an activation $a_0 = 1$. LRP-0 works well for the upper layers close to the output, as these contain lot of condensed latent information, and LRP-0 does not filter out information.
The second rule is LRP-ϵ:

$$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k \tag{5.3}$$

Here, a small positive constant ϵ is added to the total contribution, to prevent the denominator from getting too close to 0. This also has the effect of absorbing a small amount of the relevance from the individual neurons, to reduce the impact of weak or noisy contributions. This makes LRP-ϵ suited for the many middle layers of a network, as these can include more variations, which can then be filtered by LRP-ϵ.
The third rule is LRP-γ:

$$R_j = \sum_k \frac{a_j (w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j (w_{jk} + \gamma w_{jk}^+)} R_k \tag{5.4}$$

This rule increases the impact of positive contributions by adding the term γw_{jk}^+, where γ is a constant and $w_{jk}^+ = max(0, w_{jk})$. The larger γ becomes, the smaller the impact of negative contributions becomes. The LRP-γ rule distributes relevance more uniformly than other rules, which for example makes it good for layers close to the input when considering image classification, as

this makes more pixels light up on the resulting heatmap, making it easier to visualize relevant parts of the input.
The different LRP rules can be used in combination to get the best possible explanation, based on the structure of the network and the type of explanation that best fits the problem.

Chapter 6

Problem Statement

In this chapter, we highlight the most important points of the problem analysis. Based on these points, we present a problem statement, which shapes the remainder of this project.

From our pre-master thesis [10], we found that a well calibrated model enhances its trustworthiness, as is also discussed by Guo et al. in the paper *"On Calibration of Modern Neural Networks"* [11]. In Chapter 1, we find that trustworthiness is an important subject when machine learning systems are used in safety critical areas, which makes it relevant to consider the calibration of our model.

We find that machine intelligence systems show great potential in improving healthcare around the world, and thus directly impacting the UN's goal of "Good health and well-being" [8]. However, in order for machine intelligence systems to be fully viable in the medical field, they need to be explainable, which is required by laws in the EU [4]. Besides the need to be explainable, having the machine intelligence system express its uncertainty also increases the medical professionals' trust in the system [3]. As mentioned in Chapter 1, little research has been done on combining uncertainty and explanation methods, which we believe could be of great value.

In order to get a better understanding of how to express uncertainty in deep learning, we examine the most predominant methods for quantifying this uncertainty in Chapter 2. Here, we evaluate four different ways for expressing uncertainty in deep learning, namely: Deep Ensembles, MC Dropout, Markov Chain Monte Carlo and Variational Inference. Due to the limited span of this project, we limit our focus to explaining the uncertainty captured by Variational Inference. In Chapter 5, we research already established methods for explaining deep neural networks, and find that a propagation-based method, specifically LRP, may be suitable for explaining uncertainty.

The datasets we base our models and explanations on regards the prediction of sepsis onset from EHRs, described in Chapter 4. We have two datasets at our disposal, one from the CrossTracks cohort [7] and another from the PhysioNet sepsis challenge [6].

Based on the problem analysis, we present the following problem statement:

> **How can the uncertainty of a neural network for predicting sepsis be explained using LRP, such that is possible to determine the uncertainty contribution of each feature?**

- How can a BNN be implemented with Variational Inference to express the uncertainty of the network?
- How well calibrated is the resulting BNN?
- How can LRP be used to explain which features affect the uncertainty for a given prediction?

Part II

Implementation

Chapter 7

Implementing a BNN in TensorFlow

In this chapter, we discuss how our BNN is implemented using TensorFlow, including how DenseVariational layers are used in relation to Variational Inference, and how the reparameterization trick is used during optimization of the model.

7.1 DenseVariational

In this chapter, we cover how we implement a BNN trained with Variational Inference, answering the first subquestion of the problem statetment in Chapter 6. The BNN is implemented through the use of DenseVariational layers in TensorFlow (`tensorflow.org`). TensorFlow Probability (`tensorflow.org/probability`) is a library built on top of TensorFlow, which implements additional features that are useful for creating probabilistic models. The construction of a feed forward neural network with stochastic layers can be done by using the DenseVariational layer in the TensorFlow Probability library. A DenseVariational layer learns the parameters for the approximation of the posterior probability distributions over θ by using Variational Inference, described in Section 3.1.

The function signature for creating a DenseVariational layer in TensorFlow Probability can be seen in Code snippet 7.1. Here, the function parameters `units`, `make_posterior_fn` and `make_prior_fn` are required by the user. This means that when using the DenseVariational layer, we have to specify the prior $p(\theta)$ and approximated posterior $q(\theta)$ distributions. It is important to note that the posterior we define has to be learned, in order for it to be an approximation of the true posterior $p(\theta|D)$.

```
tfp.layers.DenseVariational(
    units, make_posterior_fn, make_prior_fn, kl_weight=None, kl_use_exact=False,
    activation=None, use_bias=True, activity_regularizer=None, **kwargs
)
```

Code snippet 7.1: The DenseVariational layer, with its corresponding inputs. Please note that this is a direct copy of the implementation in TensorFlow Probability [32].

The DenseVariational layer approximates the true posterior distribution $p(\theta|D)$ by using the ELBO term derived in Equation 3.4b from Section 3.1. This can be rewritten by dividing both the enumerator and denominator in the ELBO equation with $p(\theta)$:

$$ELBO = -\mathbb{E}_q\left[\log \frac{q(\theta)/p(\theta)}{p(\theta, D)/p(\theta)}\right] \tag{7.1}$$

By splitting up the logarithmic equation, we can further rewrite Equation 7.1 to:

$$ELBO = -\mathbb{E}_q\left[\log \frac{q(\theta)}{p(\theta)}\right] + \mathbb{E}_q\left[\log \frac{p(\theta, D)}{p(\theta)}\right] \tag{7.2}$$

Realizing that the first expectation in Equation 7.2 is equivalent to the KL divergence in Equation 3.1 from Section 3.1, and that the division in the second expectation term results in the conditional probability $p(D|\theta)$, we rewrite this:

$$ELBO = -KL(q(\theta)||p(\theta)) + \mathbb{E}_q\Big[\log p(D|\theta)\Big] \tag{7.3}$$

The first term of Equation 7.3 is also called the penalty term, which penalizes the approximated posterior if it deviates too far from the prior [33].

As mentioned in Section 3.1, we maximize the ELBO term in order to approximate the true posterior $p(\theta|D)$. However, when using the DenseVariational layer, this is done by minimizing the negative ELBO:

$$-ELBO = KL(q(\theta)||p(\theta)) - \mathbb{E}_q\Big[\log p(D|\theta)\Big] \tag{7.4}$$

In our case, a data input x has a corresponding data output y, which can be specified in the equation:

$$-ELBO = KL(q(\theta)||p(\theta)) - \mathbb{E}_q\Big[\log p(Y|X,\theta)\Big] \tag{7.5}$$

Here, X and Y represent the inputs and outputs of our dataset, respectively. By observing Equation 7.5, we can see that the last term is the negated expected value of the log probability for a prediction of the model. This is equivalent to the BCE loss function from Section 3.4 [34]. This means that when using the DenseVariational layer, the loss from BCE is added together with the penalty from the KL divergence between the posterior and prior, acting as a total loss. This total loss is then minimized through the training of the network.

When implementing the posterior and prior distributions, we follow a TensorFlow Probability code example [34]. The prior distribution is implemented as seen in Code snippet 7.2. The prior has `n` parameters, which is based on the number of parameters in the kernel and bias, where the kernel is a weight matrix for the layer [35]. Here the priors are initialized with a loc (mean) of 0, and a scale (standard deviation) of 1. The value 1 for standard deviation is determined through grid search.

```
def prior(kernel_size, bias_size=0, dtype=None):
    n = kernel_size + bias_size
    return tf.keras.Sequential([
        tfp.layers.DistributionLambda(lambda _: tfd.Independent(
            tfd.Normal(loc=tf.fill([1, n], (0.0)),
                       scale=tf.fill([1, n], (1.0))),
            reinterpreted_batch_ndims=1)),
    ])
```

Code snippet 7.2: The function that defines the prior distribution.

The posterior distribution, defined in Code snippet 7.3, is a mean field distribution, which is explained in Section 3.2. This is because of the VariableLayer, which uses independent variables that are not conditioned on the input [36]. The definition of the posterior distribution in Code snippet 7.3 shares some similarities with the prior from Code snippet 7.2. The main difference here

is that the posterior distribution uses a VariableLayer in order to have learnable parameters, whereas the prior is not trained. In order to avoid negative values for the standard deviation, the parameters are passed through a softplus activation function:

$$softplus(x) = \log(1 + \exp(x)) \tag{7.6}$$

This implementation of the mean field posterior (Code snippet 7.3) deviates from [34], since we have chosen to scale the output of the softplus function. With an unscaled softplus, we find that a small network is unable to learn, even after 2000 epochs. We suspect this being due to a high variance in the standard deviation in the distributions of the network. We therefore try scaling the softplus activation of the posterior, `hparams["posterior_softplus_weight"]` on line 7 in Code snippet 7.3, which results in much faster learning and better performance of the network. The increase in performance can be seen in Appendix A.

```
1 def posterior_mean_field(hparams, kernel_size, bias_size=0, dtype=None):
2     n = kernel_size + bias_size
3     return tf.keras.Sequential([
4         tfp.layers.VariableLayer(2 * n, dtype=dtype),
5         tfp.layers.DistributionLambda(lambda t: tfd.Independent(
6             tfd.Normal(loc=t[..., :n],
7                        scale=1e-5 + hparams["posterior_softplus_weight"] *
                               tf.nn.softplus(t[..., n:])),
8             reinterpreted_batch_ndims=1)),
9     ])
```

Code snippet 7.3: The function that defines the posterior distribution, with regularizing softplus scale.

However, we find that the network being unable to learn could also be due to poorly initialized posteriors, as found by Rossi et. al [37]. By using grid search, we find fitting initializations for the means and standard deviations of the parameters in the posterior. This is done by sampling means and standard deviations from two seperate normal distributions, each with a standard deviation of 0.1 and a mean found through grid search. Here, we find the best mean value to be 0 for the distribution for initializing mean values, and -6 for the distribution for initializing standard deviation values. This results in comparable learning and performance rates to that of scaling the softplus. The manual initializations of the VariableLayer is shown in Code snippet 7.4.

```
1 def posterior_mean_field(hparams, kernel_size, bias_size=0, dtype=None):
2     n = kernel_size + bias_size
3     return tf.keras.Sequential([
4         tfp.layers.VariableLayer(2 * n, dtype=dtype, initializer=lambda shape, dtype:
              random_gaussian_initializer(hparams["posterior_init_loc"],
              hparams["posterior_init_scale"], shape, dtype), trainable=True),
5         tfp.layers.DistributionLambda(lambda t: tfd.Independent(
6             tfd.Normal(loc=t[..., :n],
7                        scale=1e-5 + tf.nn.softplus(t[..., n:])),
8             reinterpreted_batch_ndims=1)),
9     ])
```

Code snippet 7.4: The function that defines the posterior distribution, using initialized weights.

7.2 Optimization

For optimization of parameters in the BNN, we use the Adam algorithm, described in Appendix G. The gradients used for optimizations in Adam are calculated in TensorFlow, following the principle of backpropagation, which is described in Appendix F.1.1.

The following description of the reparameterization trick is based on information from [38].

Here, it is important to notice that, as weights are sampled from distributions (in our case normal distributions), and as this sampling cannot be differentiated, the gradients with relation to the parameters of the distributions cannot be determined. As a solution to this, the reparameterization trick can be used.

The principle is to convert the random variable $\theta \sim q_\phi(\theta)$ into a deterministic variable (ϕ is the parameters of the posterior distributions, μ and σ). This is done using a deterministic mapping such that $\theta = g_\phi(\epsilon)$ where $\epsilon \sim \mathcal{N}(0,1)$, such that the stochasticity is not an inherent part of the sampling of θ. The function g is defined in relation to the distributions used, which for our case is: $g_\phi(\epsilon) = \epsilon * \sigma + \mu$. This simulates the sampling from the distribution, while allowing gradients to be computed for the parameters μ and σ.

In TensorFlow's implementation of the normal distributions, this reparameterization trick is used when sampling values, as shown in Code snippet 7.5. This means that operations using the parameters `loc` and `scale` (μ and σ) can be recorded using `GradientTape`, such that gradients of the loss function with respect to the parameters can be calculated during training.

```
def _sample_n(self, n, seed=None):
    loc = tf.convert_to_tensor(self.loc)
    scale = tf.convert_to_tensor(self.scale)
    shape = ps.concat([[n], self._batch_shape_tensor(loc=loc, scale=scale, axis=0)
    sampled = samplers.normal(
        shape=shape, mean=0., stddev=1., dtype=self.dtype, seed=seed)
    return sampled * scale + loc
```

Code snippet 7.5: The implementation of the method for sampling values from a normal distribution. Please note that this is a direct copy of the implementation in TensorFlow Probability [39].

Chapter 8

Uncertainty in Features

In this chapter, we give an overview of our method for evaluating the uncertainty of a given prediction relative to the input features, by using LRP on a BNN, as well as a description of the implementation of the method. This covers the third subquestion of the problem statement in Chapter 6. The idea takes inspiration from the B-LRP method from Bykov et al. [5], where they perform LRP on several sampled models from a BNN to consider the uncertainty in the explanations. Note that in the paper, they specifically consider the uncertainty in explanations, not the uncertainty of predictions or how these are affected by the input features.

8.1 Method Overview

The basis of our approach, that allows us to consider the uncertainty in our predictions, is our BNN. We train our BNN based on Variational Inference, described in Section 3.1, and use it to get predictions by sampling parameters from the approximated posterior $q(\theta)$. To consider the uncertainty relative to each feature, we look at the variance in LRP relevance scores for each feature. The general approach is as follows:

- Sample N models from the approximated posterior $q(\theta)$.
- Run LRP on each of the N models to get N relevance scores for each feature.
- Consider the distributions over relevance scores for each feature, and evaluate the variation in the distributions.

This builds on the intuition that a feature with high relevance has a high impact on the prediction. If a feature with high relevance has high variance, this indicates uncertainty in the prediction, which is tied to the specific feature.

As described in Section 5.2, relevance scores from LRP are based on the weights and activations of the network for the given prediction. As we sample N different sets of parameters, we get N different relevance scores for each feature. This means that the change in relevance scores across samples are caused by the change in parameters. From this, it follows that a high variance in relevance scores must be caused by a high variance in parameters, which stems from epistemic uncertainty in the model parameters of the BNN. However, since the relevance value we propagate back from the output is the probability of sepsis, the total relevance of each prediction is affected by the aleatoric uncertainty inherent in the prediction. Whereas the parameters of the sampled model determine how relevance is distributed among features, the prediction determines how much total relevance is distributed, which also affects the size of relevance scores. Therefore, by evaluating the variation in relevance scores, we get a feature-specific measure of total uncertainty for model predictions, including both aleatoric and epistemic uncertainty.

8.2 Method Implementation

In this section, we describe our implementation of the method, including how we compute the LRP relevance scores for a given model and data point, and how we sample the parameters for the models based on our BNN.

8.2.1 Computing LRP

As an overview, the computation of LRP relevance for a single model consists of a forward pass and a backward pass. In the forward pass, parameters are sampled, and activations and total contributions (the denominator in Equation 5.1 from Chapter 5) are computed, up to and including the output prediction. In the backward pass, the output probability is propagated back through the network as relevance, using the activations and total contributions computed during the forward pass. This approach is based on the description of how to efficiently implement LRP, described by Montavon et al. [30]. Here, they split the computation into four steps, as follows:

$$\forall_k : z_k = \epsilon + \sum_{0,j} a_j \cdot w_{jk} \tag{8.1a}$$

$$\forall_k : s_k = R_k / z_k \tag{8.1b}$$

$$\forall_j : c_j = \sum_k w_{jk} \cdot s_k \tag{8.1c}$$

$$\forall_j : R_j = a_j \cdot c_j \tag{8.1d}$$

Here, j and k denote neurons in the current layer and the following layer respectively, following the notation from the LRP rules in Section 5.2. The four computation steps change the order of computations in the LRP rules, such that computation steps are done for a full layer at a time.

In our implementation, this is done by recursively going through the forward pass in the first step (Equation 8.1a) until reaching the output layer, after which we backtrace through the callstack while computing the backward pass in the last three steps (Equation 8.1b - Equation 8.1d). The block of code performing these four steps can be seen on Code snippet 8.1:

```
def lrp_step(self, activation_prev, layer_number, ...):
    ...

    with tf.GradientTape() as g:
        g.watch(activation_prev)

        #Adds a 1 as activation for bias neurons.
        input_with_bias = _add_bias_activation(activation_prev)

        #Forward step #1
        if "lrp0" in layer_name:
            z = tf.matmul(input_with_bias, weights)
        else:
            z = tf.matmul(input_with_bias, weights) + self.epsilon

        #Element-wise division #2
```

```
17          R, prediction = self.lrp_step(activation_next, layer_number + 1)
18          s = R / z
19
20          #Backward step #3
21          #Has to stop the tape from recording gradients or else the tape will consider
22          #how activation_prev is used in following calculations.
23          l = tf.reduce_sum(z * tf.stop_gradient(s))
24          c = g.gradient(l, activation_prev)
25
26      #Element-wise multiplication #4
27      return activation_prev * c, prediction
```

Code snippet 8.1: Code Snippet from the recursive `lrp_step` function, showing the implementation of the four steps for efficient LRP.

Note that the activation of the bias neuron is added on line 8, as it is used when calculating the contribution values in step 1.

The first step, Equation 8.1a, can be seen on lines $8 - 11$. Note that we use two different LRP rules, LRP-0 and LRP-ϵ. We use LRP-0 for the last layer before the output layer, as it does not filter out information, and LRP-ϵ for the other layers, as this can remove some noise in the relevance, as described in Chapter 5.

The second step, Equation 8.1b, can be seen on lines 14 and 15. On line 14 we get the relevance from the next layer through the recursive call to `lrp_step`. Note that the final prediction is also propagated back, to be returned alongside the relevance scores, but it is not used in the computations. The computation for the `s` values can be seen on line 15.

The third step, Equation 8.1c, can be seen on lines 20 and 21. Here, we use a trick from Montavon et al. [30], where we compute the `c` values as gradients, which allows them to be computed efficiently through TensorFlow's automatic differentiation. More specifically, TensorFlow uses reverse order auto differentiations with eager execution. When doing reverse order auto differentiation, a list of operations is required, which is why `GradientTape` is used. `GradientTape` logs specified variables and the operations done on these variables, and in this case we specify this variable to be the previous activations with `g.watch(activation_prev)` on line 2. The list of operations is then traversed in reverse order using the chain rule for each step [40, 41]. For more details on back propagation, see Section F.1 in Appendix F.

The last step, Equation 8.1d, can be seen on line 24, where we simply return the relevance as the product of activations and `c` values, along with the output prediction.

8.2.2 Sampling Parameters

As mentioned, the parameters of the model are sampled during the forward pass of the method, which means that parameters are sampled in a layerwise manner. In our implementation, we sample parameters manually and use them to create point estimate dense layers. This is done to ensure that the same parameters are used in both the forward and backward pass of our LRP computations. The sampling of parameters can be seen on Code snippet 8.2:

```
1   #Generates normaldistributions from the parameters and samples a weight or bias from
        each.
```

```
weights = []
for loc_weight, scale_weight in zip(loc_weights, scale_weights):
    weights.append(tfd.Normal(loc=loc_weight, scale=1e-5 +
        tf.nn.softplus(scale_weight)).sample())

biases = []
for loc_bias, scale_bias in zip(loc_biases, scale_biases):
    biases.append(tfd.Normal(loc=loc_bias, scale=1e-5 +
        tf.nn.softplus(scale_bias)).sample())
```

Code snippet 8.2: Implementation of the sampling of parameters from the BNN.

Here, weights and biases are sampled from the posterior of the BNN by recreating the posterior distribution from the `loc` (mean) and `scale` (standard deviation) from the corresponding layer of the BNN. The `loc` and `scale` parameters used in the loops are found by indexing into the parameters of the BNN. After the weights and biases have been sampled, they are used to create a Tensorflow `Dense` layer, which is then returned. The full function, including indexing and layer creation, can be seen on Code snippet I.1 in the Appendix (Part V).

Part III

Experiments

Chapter 9

Initial Experiments

In this section, we cover the setup of our experiments and how hyperparameters of our models are tuned and logged. Following this, we introduce a set of initial experiments, which is used to determine the model structure and hyperparameters of the BNN model to be used in further experiments. The models used throughout this chapter are implemented according to the description in Section 7.1.

9.1 Configuration Files

In order to ensure reproducibility and documentation of the experiments, we choose to use a configuration file for setting up and executing experiments. This configuration file is a JSON file with two main parts, one for overall `"setup"` and one specific to `"experiments"`, as shown in Code snippet 9.1.

```
{
    "setup": {
        "data_path": "", #Path for dataset.
        "log_path": "", #Path for where to put logs for the runs. The logs can be
            used when reproducing the experiments.
        "dataset": "", #Used for specifying either DP or DCT.
        "train_split": 0.7, #Specifies how much of the dataset should be used for
            training.
        "val_split": 0.15, #Specifies how much of the dataset should be used for
            validation.
        "shuffle_seed": 12345, #Used as seed for how the elements in the dataset are
            shuffled.
        "tensorflow_seed": 12345, #Used for specifying a seed to be used by
            TensorFlow for generating random values, for example for initial weights
            in a neural network.
        "log": true, #Specifies whether to create a log file for the run.
        "save_path": "", #Path for where to save diagrams, experiment results and
            logs used in TensorBoard for visualization of hyperparameters and
            training of models.
        "saved_models_dir": "" #Additional path used for saving a model, instead of
            retraining the same model for every run.
    },
    "experiments": [
        {
            "model": "bayesian_linear_model", #The identifier used to specify a
                specific model.
```

```
            "load_model_from": "", #Path used to load an existing model, rather than
                training a new model. If specified, Hyperparameters should be left
                empty.
            "hyperparameters": [ #A set of hyperparameters used in a model. Grid
                search is applied to the parameters specified in the hyperparameters.
                {
                    "units_layer1": [50, 100],
                    ...
                    "metric2": ["AUC"]
                }
            ]
        }
    ]
}
```

Code snippet 9.1: Representation of a config file used for running experiments.

In the `"experiments"` part, parameters specific to each experiment is specified. The `"experiments"` part can contain a list of experiments, which will be run in succession. In the `"setup"` part, parameters used across all experiments are specified.

The parameters specified in the scope of `"experiments"` are the hyperparameters used for the experiments. When multiple values are specified for a hyperparameter in an `"experiment"`, for example as shown with `"units_layer1"` in Code snippet 9.1, a grid search is applied. This means that the total number of runs for that experiment is the product of the number of values for each hyperparameter. These hyperparameters are saved as Hparams. Hparams is a feature within TensorBoard, that automatically logs all specified parameters in `"experiments"`. These logs can be visualized by TensorBoard, which shows in-depth information about a model's training process, and the results for different combinations of hyperparameters [42]. In addition to this, the entire configuration file and the commit ID is saved as a text file within a log file directory, which allows replication of the exact experiments. By using configuration files, we have created a pipeline for running experiments and saving relevant information from the runs.

9.2 Initial Experiments on ΔDP

This section covers a set of initial experiments on ΔDP, with the goal of obtaining an optimized model for this dataset. The two PhysioNet datasets, ΔDP-A and ΔDP-B, are from two different hospitals, which may be why we see different performances from models trained on ΔDP-A, compared to ΔDP-B. During the initial phases of the optimization process, we chose to focus on optimizing towards ΔDP-B, as this dataset resulted in the best performing models. Whereas ΔDCT must be accessed through a remote server, ΔDP-B is locally available, which provided the benefit of a fast experimentation process. The data framing we based the initial hyperparameter optimization on was "Sliding Window", as this best reflects the expected clinical environment, as mentioned in Chapter 4. All the BNNs in this section are implemented by using TensorFlow DenseVariational layers, which is described in Section 7.1.

9.2.1 Early Tests

At the start of the optimization process, we tested many different parameters, namely:

- Model Width and Model Length.
- Learning Rate.
- KL Weight Scale.
- Epochs.
- Batch Size.

For model width and length, we found that generally, smaller BNNs performed the best, as seen in Appendix B.1. We observe that a learning rate of 0.0001 and 0.001 yielded the best results, with varying numbers of epochs depending on model size and the learning rate.

We also tested changing the KL weight scale for scaling the penalty term described in Section 7.1, consisting of the KL divergence between the posterior and prior used by the DenseVariational layers. Through the tests, we found that changing the scaling of the penalty term gave worse results, and we therefore chose to keep using the suggested scaling value of $1/N$, where N is the number of data points in the dataset [34], during any following hyperparameter tuning.

At the time of the early tests, we were using the data framing "Sliding Window" on ΔDP-B. This provided us with great performance metrics and low ECE and ACE values. However, as seen on the reliability diagram Figure 9.1, this is due to the low probabilities of the predictions, resulting in a few well calibrated bins containing most of the predictions. Since ECE is weighted based on number of data points in the bin, as described in Appendix H, this resulted in a good ECE. For ACE, these predictions are dispersed into 10 ranges with very similar probabilities, in turn resulting in a low ACE.

Due to this, we decided to change the data framing from "Sliding Window" to "Fixed Time To Onset". We based a new grid search on parameters that showed the best performance on the "Fixed Time To Onset" data framing, which can be seen on Figure B.5 in Appendix B.4. Changing data framing provided better AUROC and AUPRC, as well as higher confidence predictions, as seen on Figure 9.2. Here, we also found that ΔDP-A now provided the best results. This means that through the remainder of the report, when referring to models trained on the PhysioNet dataset, this refers to ΔDP-A.

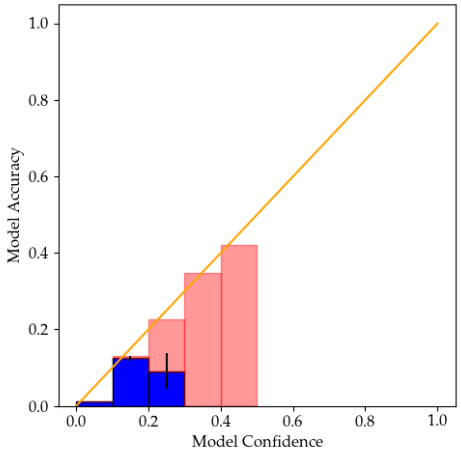


Figure 9.1: Reliability diagram based on results from 5 sampled models on the ΔDP-B test set, using the BNN from Appendix B.2.

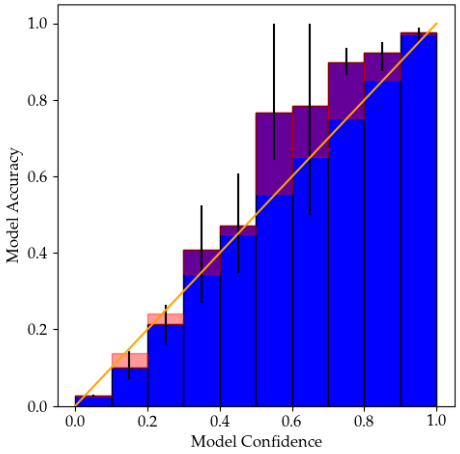
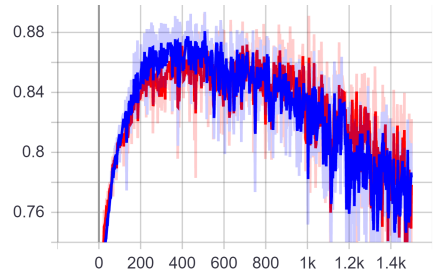
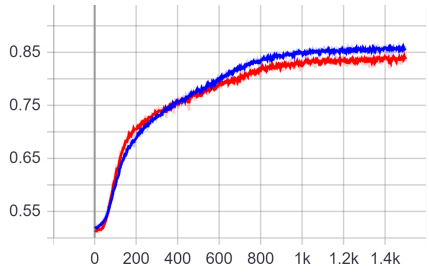


Figure 9.2: Reliability diagram based on results from 5 sampled models on the ΔDP-A test set, using the BNN from Appendix B.2.

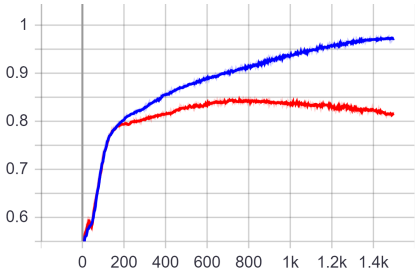
During the hyperparameter tuning, we find that training a BNN is more complicated, compared to training its point estimate counterpart. This might be due to the additional parameters that are present in a BNN, in terms of parameters for the prior and posterior distributions, which are very dependent on their initial values when working with complex problems [37]. The effect of a bad or good initialization can be seen on Figure 9.3a and Figure 9.3b respectively. As a result, searching for the right initializations of the prior and posterior distributions is tedious, as many parameters need to be searched. We observe that, when training an FFNN on the same data, with the same model structure, it is less affected by the hyperparameters. However, when good hyperparameters are found for the BNN, we often see better performance, better calibration and less tendency to overfitting, as shown on Figure 9.3b, whereas the FFNN is more prone to overfitting, as shown on Figure 9.3c. This is also discussed by Hernandez-Lobato et al. in [43].



(a) Training a BNN with bad hyperparameters.



(b) Training a BNN with good hyperparameters.



(c) Training an FFNN with good hyperparameters.

Figure 9.3: The figures show the AUROC during training for different model types. The x-axis shows the number of epochs, the y-axis shows the AUROC. The blue line represents AUROC for the training set, and the red line for the validation set.

9.2.2 Two Outputs for LRP

When doing LRP, we use the confidence in a given label as relevance for the last layer and propagate it back through the network, as described in Section 5.2. Since the BNN we use has a single output neuron, which expresses its confidence in the sepsis positive class, a prediction of 0.99 is a confident sepsis positive prediction. However, this also means that a prediction of 0.01 is a confident sepsis negative prediction. This means that when a prediction of 0.01 is propagated with LRP, this small value is distributed among the features that contributed to the prediction, resulting in low relevance scores, as seen on Figure 9.4.

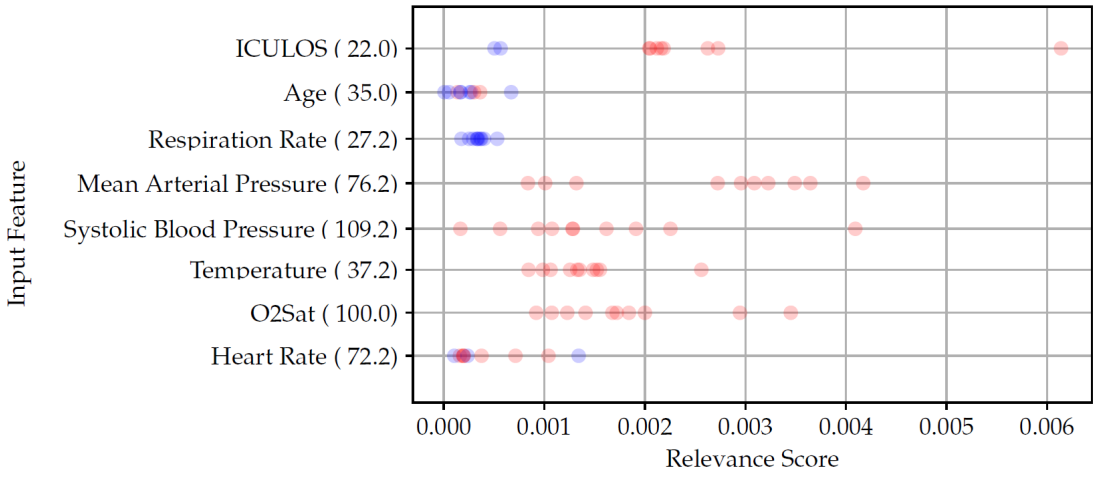


Figure 9.4: Relevance diagram showing low relevance scores (see x-axis) for a prediction on ΔDP-A, from a BNN with a single output neuron.

This makes it difficult to evaluate which features the model base its prediction on when predicting sepsis negative. In order to avoid this issue, we choose to use a network with two output neurons. Here, one neuron expresses its belief in the sepsis positive class, and the other expresses its belief in the sepsis negative class. We use a softmax activation function to ensure the two outputs sum to 1, such that if the positive class predicts 0.7, the negative class predicts 0.3, representing the probability of each outcome. Now, instead of having a prediction of 0.01 in the sepsis negative class, this would instead be a prediction of 0.99. After this change we saw similar performance for a model with the same hyperparameters.

9.2.3 Final Configuration

Through the different experiments, we end up with the following final configuration for the BNN used on ΔDP-A with the "Fixed Time to Onset" data framing:

- 1,500 epochs.
- 5,000 batch size.
- 0.0001 learning rate.
- 0.0 mean for the distribution used for initializing mean values in the posterior.
- -6.0 mean for the distribution used for initializing standard deviation values in the posterior.

The model consists of three stochastic layers, with 60, 40, and 20 neurons, followed by two output neurons, as seen on Figure 9.5.

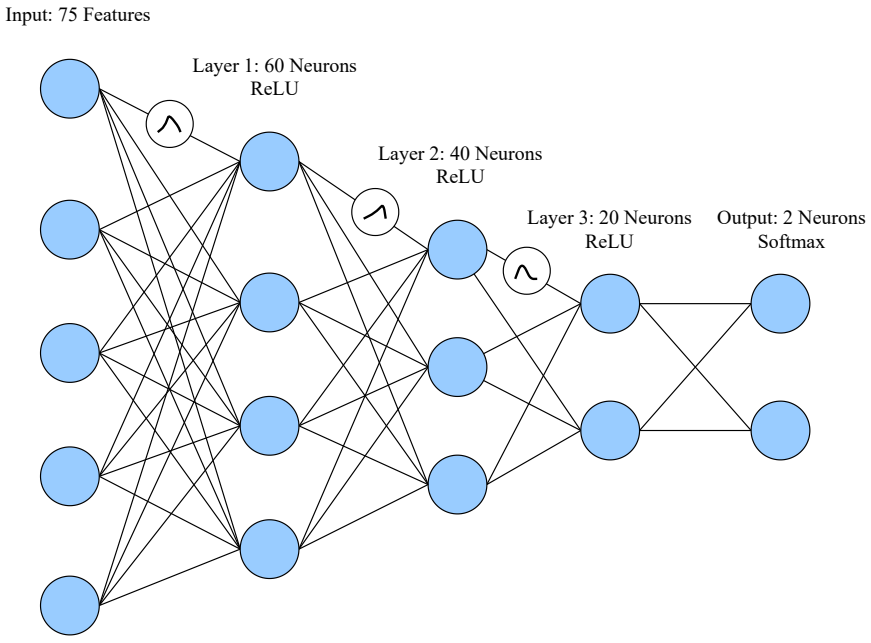


Figure 9.5: Model architecture of the final BNN on ΔDP-A, based on the grid search in Appendix B.3.

The final model structure and hyperparameters yield the following performance and calibration scores:

BNN	AUROC	AUPRC	ECE	ACE
Mean	0.8726	0.7295	0.0264	0.0239
Max	0.8785	0.7399	0.0273	0.0251
Min	0.8664	0.7255	0.0258	0.0232

Table 9.1: AUROC, AUPRC, ECE and ACE based on results from 5 sampled models, using the BNN with the final configuration on ΔDP-A with the data framing "Fixed Time To Onset".

9.3 Initial Experiments on ΔDCT

In this section, we cover the search for an optimized model on ΔDCT. Since we also observe low probabilities in the predictions on ΔDCT when using the data framing "Sliding Window", we choose to use the "Fixed Time To Onset" data framing for ΔDCT as well. We also modify the network to have two output neurons, to accommodate the problems encountered when doing LRP, mentioned in the previous section.

As for PhysioNet, we also observe through early experiments that smaller models give better results on ΔDCT. Therefore, minimal testing is done on model size, where the best performing model is identical to the model used for ΔDP-A in Figure 9.5. Instead, we focus on testing different hyperparameters for learning rate, batch size, number of epochs and initialization values for the distributions. A grid search of the hyperparameters on ΔDCT can be found in Figure B.7 in Appendix B.6. Here, we observe that the best hyperparameters for our model trained on DCT are:

- $3,000$ epochs.
- $5,000$ batch size.
- 0.0001 learning rate.
- 0.0 mean for the distribution used for initializing mean values in the posterior.
- -6.0 mean for the distribution used for initializing standard deviation values in the posterior.

The final configuration for the BNN trained on ΔDCT results in the following performance:

BNN	AUROC	AUPRC	ECE	ACE
Mean	0.7675	0.2882	0.0074	0.0073
Max	0.7787	0.2984	0.0095	0.0087
Min	0.7609	0.2744	0.0075	0.0065

Table 9.2: AUROC, AUPRC, ECE and ACE based on results from 5 sampled models, using the BNN with the final configuration.

We observe that the performance on Table 9.2 is significantly lower for AUROC and AUPRC, compared to the performance on Table 9.1. We believe the main reason for this, is that ΔDCT contains substantially more missing values than ΔDP-A, as it is not based on information from ICUs, which is also observed in Appendix C. However, we see that the calibration scores for

ΔDCT are better than for ΔDP-A. This is explored further in the calibration experiment, described in Chapter 10.

Chapter 10

Calibration

In Chapter 1, we mention that calibration is important when evaluating whether a model's prediction confidence can be trusted [11]. In addition to this, Jospin et al. [17] mentions that BNNs are often better calibrated, compared to non-Bayesian Feed Forward Neural Networks (FFNN). Therefore, we want to run an experiment, where we evaluate the expected calibration error (ECE) and adaptive calibration error (ACE) of a BNN, and compare these to the ECE and ACE of an equivalent FFNN. This answers the second subquestion of the problem statement in Chapter 6. More information about ECE and ACE can be found in Appendix H, which is a copy of parts from Chapter 5 *"Reasoning about Uncertainty"* from our pre-master thesis [10]. Guo et al. [11] observes that multiple factors impacts the calibration of a model, for example model size. Therefore, when selecting the equivalent FFNN, we do not grid search for different model sizes, and only replace the stochastic layers with corresponding deterministic layers. However, for epochs, learning rate and batch size, we conduct a small grid search, seen in Section J.1, as we find that these parameters have an impact on whether the FFNN starts to overfit or not. By observing Section J.1 and Section J.2, we can see that the most optimal models on either dataset use the same values for these parameters, namely:

- Identical model architecture to that found in Chapter 9.
- $1,500$ epochs.
- 0.0001 learning rate.
- $5,000$ batch size.

In this experiment, the results regarding the FFNN is based upon five models trained with different parameter initializations, whereas the results for the BNN is based upon five sampled models. This means that the error bars in the reliability diagrams on Figure 10.1 and Figure 10.2 represent the variations in model accuracy for each bin, with the lower part of the error bar being the minimum accuracy and the upper part being the highest accuracy. The tables throughout this section include the minimum, maximum and mean values for ECE and ACE over five runs.

10.1 PhysioNet

In this section, we compare calibration errors of the BNN found in Chapter 9 and the FFNN discussed above, on the PhysioNet dataset ΔDP-A. In Table 10.1 and Table 10.2, it can be seen that in general, the BNN is better calibrated. The BNN has a mean ECE of 0.0264, and a mean ACE of 0.0239, whereas the calibration errors of the FFNN are 0.0397, and 0.0350 respectively. In addition to this, there is also a significant difference in the performance. Here, the BNN has a mean AUROC of 0.8726 and a mean AUPRC of 0.7295, compared to the FFNN's mean AUROC of 0.8333 and a mean AUPRC of 0.6380.

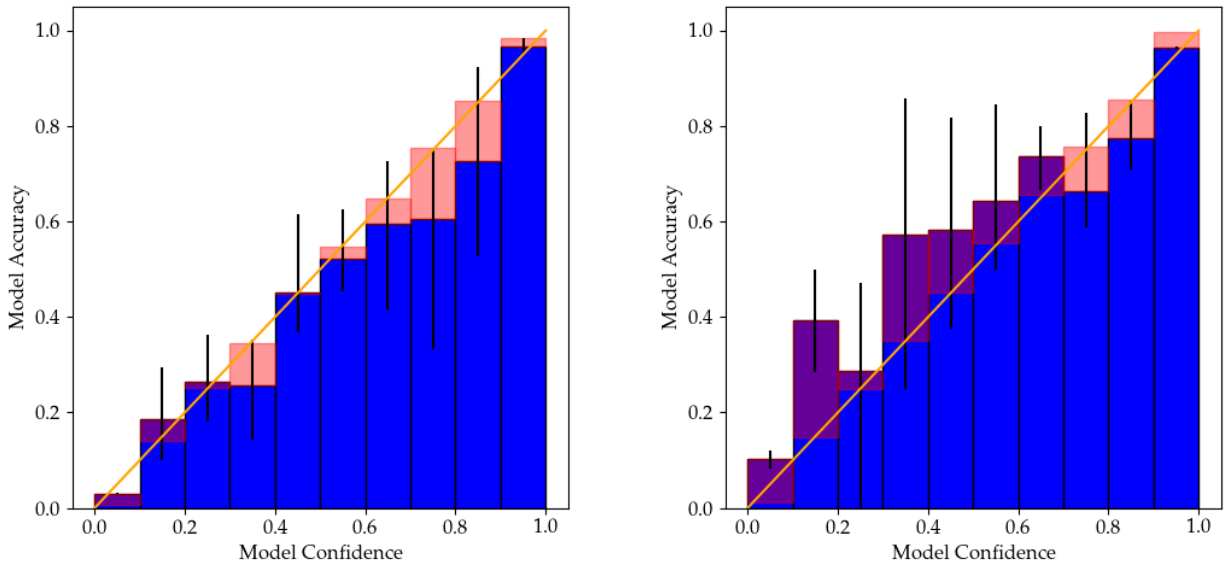
BNN	AUROC	AUPRC	ECE	ACE
Mean	0.8726	0.7295	0.0264	0.0239
Max	0.8785	0.7399	0.0273	0.0251
Min	0.8664	0.7255	0.0258	0.0232

Table 10.1: AUROC, AUPRC, ECE and ACE based on five models sampled from the BNN on ΔDP-A.

FFNN	AUROC	AUPRC	ECE	ACE
Mean	0.8333	0.6380	0.0397	0.0350
Max	0.8415	0.6557	0.0449	0.0372
Min	0.8198	0.6103	0.0373	0.0320

Table 10.2: AUROC, AUPRC, ECE and ACE based on five FFNN models with different initialized weights on ΔDP-A.

By examining the reliability diagrams in Figure 10.1, we observe that the BNN is better calibrated for most of the confidence ranges, and has lower variation in the accuracy in most of the bins.



(a) Reliability diagram for five sampled BNN models.

(b) Reliability diagram for five FFNN models with different initialized weights.

Figure 10.1: Reliability diagram for the five sampled models from the BNN and the five FFNN models.

10.2 Cross-Tracks

In this section, we compare calibration errors of the BNN found in Chapter 9 and the FFNN discussed in the start of this chapter on the ΔDCT dataset. By inspecting Table 10.3 and Table 10.4, we can see that the BNN, as also observed in Section 10.1, has significantly lower calibration errors. Where the BNN has a mean ECE of 0.0074, and a mean ACE of 0.0073, the FFNN has a mean ECE of 0.0417 and a mean ACE of 0.0316. Comparing the FFNN and BNN, the BNN has

higher AUROC and lower calibration errors, but the FFNN has the best AUPRC. Here, the FFNN has a mean AUPRC of 0.3380, compared to a mean AUPRC of 0.2882 for the BNN.

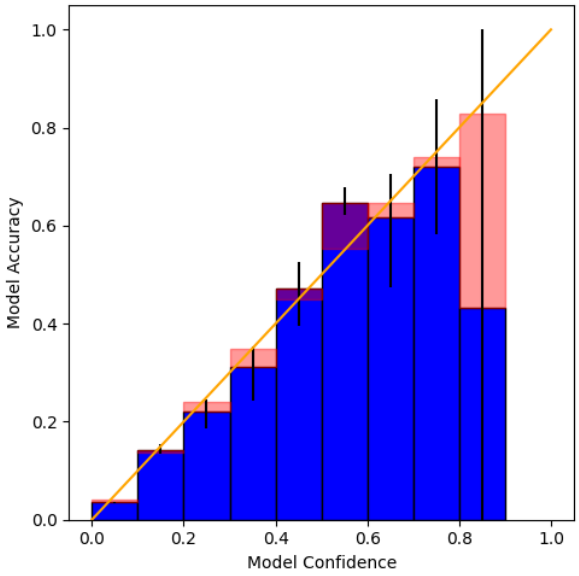
BNN	AUROC	AUPRC	ECE	ACE
Mean	0.7675	0.2882	0.0074	0.0073
Max	0.7787	0.2984	0.0095	0.0087
Min	0.7609	0.2744	0.0063	0.0065

Table 10.3: AUROC, AUPRC, ECE and ACE based on five models sampled from the BNN on ΔDCT.

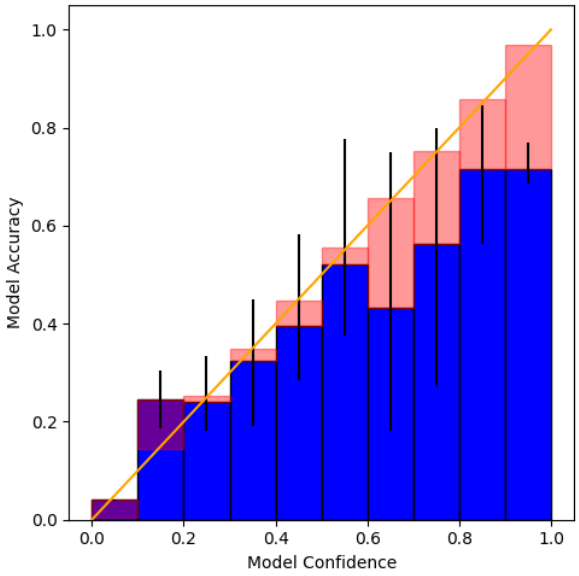
FFNN	AUROC	AUPRC	ECE	ACE
Mean	0.7300	0.3380	0.0417	0.0316
Max	0.7383	0.3535	0.0423	0.0329
Min	0.7167	0.3211	0.0408	0.0303

Table 10.4: AUROC, AUPRC, ECE and ACE based on five trained FFNN models with different initialized weights on ΔDCT.

The reliability diagrams in Figure 10.2 shows that the BNN lacks predictions in the last confidence bin, yet the BNN has a lower ECE and ACE. Also, the bin containing confidences within $0.8 - 0.9$ in Figure 10.2 has significantly larger error bars, compared to the other bins, which can indicate a low number of predictions for the bin. By observing the first two bins, $0.0 - 0.1$ and $0.1 - 0.2$, in Figure 10.2b, we can see that the reliability diagram for the FFNN has significantly larger errors, compared to the diagram for the BNN, shown in Figure 10.2a. As ΔDCT contains more negative cases (94.25%), being miscalibrated in these bins acts as a major contribution to the overall ECE of the model, as ECE is based on the volume of predictions within each bin. Here, one could assume that the bins containing the largest number of predictions would be these exact bins. Also, in relation to this, it is therefore expected that the model is better calibrated for these bins. This is not the case for the FFNN (Figure 10.2b), as it has large calibration errors relative to the $0.0 - 0.1$ and $0.1 - 0.2$ bins. Because of this miscalibration, we expect that the FFNN makes less FN and TN predictions, and that it predicts more TP and FP predictions as well, such that it gets a higher mean AUPRC of 0.3380, but also a lower mean AUROC of 0.7300, compared to the BNN, with 0.2882 and 0.7675, respectively. We also observe that the average error bars for confidences less than 0.8, on the reliability diagram of the FFNN in Figure 10.2b, are notably larger, compared to the BNN in Figure 10.2a. In addition to this, the BNN in Figure 10.2a is better calibrated on almost every bin with confidences less than 0.8. Keeping this in mind, we see the BNN as being the better calibrated model.



(a) Reliability diagram for five sampled BNN models.



(b) Reliability diagram for five FFNN models with different initialized weights.

Figure 10.2: Reliability diagram for the five sampled models from the BNN and the five FFNN models.

On both ΔDP-A and ΔDCT, when comparing the BNN and FFNN, the BNN has better calibration scores, while generally maintaining better performance. This means that the observations made by Jospin et al. [17], are also present in the models we have inspected.

Chapter 11

I Do Not Know

In Chapter 1, we discuss that a model being capable of answering "i do not know", is one way of enhancing trust in its predictions. Therefore, as we have implemented a BNN, which makes it possible to express uncertainty in predictions, we want to run some experiments where we apply this concept.

In the medical article *"Second opinion needed: communicating uncertainty in medical machine learning"* by Kompa et al. [3], the ability for neural networks to express "i do not know" (IDK), and abstain from making predictions, is discussed as being important when using AI in a medical context. They also discuss an approach for deciding when to abstain, where two functions, f and g, are used. f is the prediction function for a neural network, and g is a selection function that decides if the neural network can make a prediction or if it abstains. We want to follow this approach by having f be the mean of predictions from multiple sampled models, and g be a function that uses the uncertainty expressed by the BNN for deciding when to abstain and when to predict.

We see multiple ways of defining the function g. Either it selects whether to abstain or not based on disagreements in multiple predictions, together with a limit for how many predictions can disagree, or based on the standard deviation of multiple predictions, together with a threshold. For both of these definitions, we want to conduct an experiment, to evaluate whether the uncertainty of the BNN can be used to improve predictions.

In both experiments, we want to use the number of true positives (TPs), false negatives (FNs), false positives (FPs), and true negatives (TNs), together with AUPRC and AUROC, to analyze whether the model performs better or worse, when abstaining from predicting.

11.1 Abstain with Disagreement

In the first experiment, the procedure is as follows:

1. Sample N models from the BNN and make a prediction for each data point with each model.
2. Calculate the number of TP, FN, FP, and TN predictions together with AUPRC and AUROC, at different positive thresholds and different disagreement limits.

The number of models sampled will be $[5, 10, 50, 100, 150]$, to gain a deep insight into how the number of predictions affects the results.

The positive thresholds are defined as the limit for when a prediction is said to be sepsis positive or sepsis negative. For example, with the threshold 0.5, all confidence levels for the positive class over or equal to 0.5 are positive predictions, whereas all confidence levels below the threshold are negative predictions. In order to evaluate the entire range of confidence levels,

we want to experiment with varying the thresholds between 0 and 1, and to limit the scope of results, we use the following thresholds: $[0.1, 0.2, ..., 0.8, 0.9]$.

The disagreement limit is used as a cutoff for when the model makes a prediction and when it abstains. The limit defines the percentage of the N predictions from the sampled models, which can disagree before abstaining from a prediction. For example, with a disagreement limit of 10% and with 10 sampled models, 9 out of 10 predictions need to be on the same class, for the model to make a prediction, otherwise the model abstains from making predictions. For the disagreement limit, we want to experiment with the following thresholds: $[0\%, 10\%, 20\%, 30\%, 40\%]$.

If the model does not abstain from making a prediction, the confidence of the prediction is calculated as the mean of the predictions from the N sampled models.

For the different values of the disagreement limit we also show a baseline model, which cannot make IDK predictions, and where the prediction is the class with the highest mean confidence.

11.1.1 Discussion about Results

The most interesting results from the experiment are included in this section, and the full set of results can be found in Appendix K.1 for ΔDP-A and in Appendix K.3 for ΔDCT.

When observing the results, we see a very interesting outcome, that by allowing the model to express IDK, there is nearly always a metric (either TP, FN, FP, TN, AUPRC, or AUROC) that is improved compared to not expressing IDK.

If we first have a look at Table 11.1, we find that by having a low positive threshold of 0.1, the model is more likely to make positive predictions and less likely to make negative predictions. This means that TP and FN are improved, but at the cost of more FP and less TN predictions.

When decreasing the disagreement limit, the model makes more IDK predictions, and interestingly, a large part are FPs (around 30% of the 74 IDK predictions, between the disagreement limits 40% and 20%, and around 25% of the 117 additional IDK predictions, between the disagreement limits 20% and 0%), and only a small part is TPs, which results in a higher AUPRC and AUROC when the model abstains more from making predictions.
Note that the following tables show results from different numbers of sampled models, however the same tendencies are found.

Positive Threshold	DL	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	175	2842	-	161	103	14	2739	0.7472	0.8820
0.1	0 ($\lfloor 0\% \rfloor$)	204	2622	191	171	70	33	2552	0.7703	0.8847
0.1	1 ($\lfloor 20\% \rfloor$)	237	2706	74	177	76	60	2630	0.7583	0.8826
0.1	2 ($\lfloor 40\% \rfloor$)	267	2750	0	182	82	85	2668	0.7472	0.8820

Table 11.1: PhysioNet: Disagreement results where 5 models are sampled. DL is the disagreement limit specifying how many predictions on the same data point can disagree before the model expresses "i do not know" (IDK). IDK is the number of data points for which the sampled models disagree. Full table in Table K.1.

In connection to the previous result, we also find that, either with a positive threshold of 0.5 or a higher positive threshold of 0.9 (shown in Table 11.2 and Table 11.3 respectively), a large

part of the IDK predictions are TP. We find that this makes sense, as the datasets have a large percentage of negative samples, which means that the model might be more uncertain about the positive predictions and output lower confidence, resulting in more disagreement between sampled models at these thresholds.

Positive Threshold	DL	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	175	2842	-	159	105	16	2737	0.7471	0.8827
0.5	0 ($\lfloor 0\% \rfloor$)	142	2776	99	140	90	2	2686	0.7375	0.8726
0.5	2 ($\lfloor 20\% \rfloor$)	159	2823	35	152	98	7	2725	0.7432	0.8786
0.5	4 ($\lfloor 40\% \rfloor$)	171	2837	9	157	103	14	2734	0.7459	0.8815

Table 11.2: PhysioNet: Disagreement results where 10 models are sampled. Full table in Table K.2.

Positive Threshold	DL	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	177	2840	-	161	103	16	2737	0.7487	0.8845
0.9	5 ($\lfloor 10\% \rfloor$)	106	2860	51	105	118	1	2742	0.6953	0.8647
0.9	10 ($\lfloor 20\% \rfloor$)	117	2870	30	116	125	1	2745	0.7222	0.8744
0.9	15 ($\lfloor 30\% \rfloor$)	125	2877	15	124	128	1	2749	0.7354	0.8794
0.9	20 ($\lfloor 40\% \rfloor$)	128	2883	6	127	132	1	2751	0.7434	0.8824

Table 11.3: PhysioNet: Disagreement results where 50 models are sampled. Full table in Table K.3.

For ΔDP-A, we find the best results for AUPRC and AUROC when sampling 150 models and using a disagreement limit of 0% (See Table 11.4). With these settings, the AUPRC increases to 0.8054 and AUROC increases to 0.8937. We also find that the number of FN predictions are nearly halved compared to the baseline, and that TP only decrease slightly. We see this as a great improvement, as we argue that telling a person that he/she does not contract sepsis, when he/she does, is the most dangerous case. We therefore find that using the uncertainty, in order to determine when to abstain from making a prediction, is useful when the data has limited information about a class, resulting in uncertain predictions.

Positive Threshold	DL	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	176	2841	-	161	103	15	2738	0.7478	0.8833
0.1	0 ($\lfloor 0\% \rfloor$)	165	2334	518	155	53	10	2281	0.8054	0.8937
0.2	0 ($\lfloor 0\% \rfloor$)	144	2508	365	141	67	3	2441	0.7664	0.8797
0.3	0 ($\lfloor 0\% \rfloor$)	138	2580	299	136	71	2	2509	0.7487	0.8740

Table 11.4: PhysioNet: Disagreement results where 150 models are sampled. Full table in Table K.4.

For ΔDCT we find similar patterns and results, which can be found in Appendix K.3. Here, we find the highest AUPRC to be 0.3521, up from 0.3112, when using a positive threshold of 0.1, a

disagreement limit of 0%, and when 150 models are sampled. For AUROC, the highest result is 0.7947, up from 0.7888, when using the same positive threshold, a disagreement limit of 20%, and when 50 models are sampled.

11.2 Abstain with Standard Deviation

This experiment follows nearly the same procedure as the previous experiment. However, instead of using a disagreement limit, we now use the standard deviation of multiple predictions when deciding whether or not to abstain from making a prediction.

Here, we sample the same numbers of models as for the experiment with disagreement limit. Unlike the previous experiment, the positive threshold does not affect the number of IDK predictions, as this is only affected by the standard deviation, and that the IDK predictions are the same even with different thresholds. Therefore, we want to limit the positive thresholds to $[0.25, 0.5, 0.75]$ instead.

To get a deeper insight into how different numbers of IDK predictions affect the performance of the model, we want to use a large range of values for the standard deviation cutoff $[0.01, 0.02, ..., 0.08, 0.09, 0.1, 0.2, ..., 0.5, 0.6]$.

11.2.1 Discussion about Results

The most interesting results from the experiment are included in this section, and the full set of results can be found in Appendix K.2 for ΔDP-A and in Appendix K.4 for ΔDCT.

Compared to the previous experiment, the performance increase is less noticeable, and we only see slight improvements in AUPRC and AUROC with few of the settings.

With the lowest positive threshold of 0.25, shown in Table 11.5, we see similar results as for the previous experiment, where a large percentage of the IDK predictions come from positive predictions. When increasing the standard deviation, the number of IDK predictions decreases from 123 to 14 with the positive predictions increasing from 146 to 205, which means that more than 50% of the IDK predictions are from the positive predictions. This again supports the idea that the model is more uncertain about positive predictions, as the data is unbalanced.
Note that the following tables show results from different numbers of sampled models, however the same tendencies are found.

Positive Threshold	Std Threshold	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	176	2841	-	160	104	16	2737	0.7424	0.8801
0.25	0.09	146	2748	123	143	87	3	2661	0.7360	0.8717
0.25	0.1	149	2757	111	145	87	4	2670	0.7358	0.8719
0.25	0.2	186	2789	42	161	90	25	2699	0.7396	0.8769
0.25	0.3	205	2798	14	170	90	35	2708	0.7431	0.8793

Table 11.5: PhysioNet: Standard deviation results where 5 models are sampled. Std Threshold is an upper limit on standard deviation in predictions on the same data point, before the model expresses "i do not know" (IDK). IDK is the number of data points for which the model express "i do not know". Full table in Table K.5.

When using a positive threshold of 0.5, as shown in Table 11.6, we see that the number of IDK predictions that are from positive predictions falls, compared to a positive threshold of 0.25. When we increase the standard deviation threshold from 0.1 to 0.3, the number of IDK predictions decreases from 133 to 11. Of these predictions, 34 were positive, which means that positive predictions made up slightly less than a third of the IDK predictions. For a higher positive threshold, shown in Table 11.7, it seems that the positive IDK predictions are likewise saturated, which means that when the model abstains from more predictions, it is often for negative predictions. This makes sense, as the model has few positive predictions with high confidence, as shown in Chapter 10. Unfortunately, a large part of the negative predictions are from TNs, without a large improvement in FNs, which therefore generally results in lower AUPRC and AUROC.

Positive Threshold	Std Threshold	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	173	2844	-	160	104	13	2740	0.7487	0.8844
0.5	0.1	133	2751	133	132	87	1	2664	0.7334	0.8706
0.5	0.2	157	2815	45	150	99	7	2716	0.7458	0.8807
0.5	0.3	167	2839	11	159	104	8	2735	0.7530	0.8850

Table 11.6: PhysioNet: Standard deviation results where 10 models are sampled. Full table in Table K.6.

Positive Threshold	Std Threshold	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	176	2841	-	161	103	15	2738	0.7464	0.8805
0.75	0.08	123	2699	195	122	88	1	2611	0.7273	0.8645
0.75	0.09	123	2719	175	122	90	1	2629	0.7265	0.8643
0.75	0.1	125	2729	163	124	92	1	2637	0.7299	0.8661
0.75	0.2	141	2825	51	137	110	4	2715	0.7392	0.8757

Table 11.7: PhysioNet: Standard deviation results where 50 models are sampled. Full table in Table K.7.

In Table 11.8 we show the best settings for the experiment on ΔDP-A, when AUPRC and AURPC are considered. Compared to using disagreement for deciding when to abstain from making a prediction, the performance increase is less noticeable. When comparing the baseline to the best settings with a positive threshold of 0.5, only 11 IDK predictions are made, of which 6 result in fewer TPs and TNs with the rest being FNs. This, we see as a drawback for using standard deviation for deciding when to abstain, as nearly half of the IDK predictions result in a worse performance.

Positive Threshold	Std Threshold	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	173	2844	-	160	104	13	2740	0.7487	0.8844
0.25	0.3	206	2800	11	172	91	34	2709	0.7530	0.8850
0.5	0.3	167	2839	11	159	104	8	2735	0.7530	0.8850

0.75	0.3	144	2862	11	140	123	4	2739	0.7530	0.8850

Table 11.8: PhysioNet: Standard deviation results where 10 models are sampled. Full table in Table K.6

As for the previous experiment, we also see similar tendencies and patterns when comparing the results to ΔDCT, where AUPRC and AUROC are only slightly improved for a few settings. The highest AUPRC is 0.3181, up from 0.3173, and the highest AUROC is 0.7851, up from 0.7850. These results can be found in Appendix K.4.

11.3 Comparison

When comparing the two approaches for when to abstain from making a prediction, we argue that abstaining with disagreement is more useful. The reason is that for almost all settings, we see an improvement in multiple of the metrics used, though with different trade-offs. This means that, if for example FNs are very expensive to predict, nearly half can be excluded with the right settings, and oppositely nearly all FPs can be excluded with another setting. Also, when considering AUPRC and AUROC, we see a large boost in performance when sampling 150 models and with a disagreement limit of 0, but at the cost of making over 500 IDK predictions. For the standard deviation approach we see similar results, but often at a greater expense in true predictions. When considering both approaches, we also think that these results make sense, as the disagreement approach might result in situations where the predictions from the sampled models have a large variance, but all with confidence levels well within the range of either the positive or negative class, set by the positive threshold. This might be opposite for the standard deviation approach, as even though all predictions are well within one of the ranges, they might still be considered IDK predictions, because of too high standard deviation.

Chapter 12

Layer-wise Relevance Propagation

In this chapter, we consider the relevance scores from our method for evaluating the uncertainty of predictions relative to the input features, described in Chapter 8. This gives an insight into which features are generally used by the model to predict sepsis positive and sepsis negative, respectively. It also shows the variance in relevance scores, and to which degree it is present in different predictions. Here, we consider results from both the ΔDP-A and ΔDCT datasets.

12.1 Experiment Approach

Through the experiments, we use two different approaches to observe relevance scores. The first approach is to consider the relevance scores for individual data points based on 20 sampled models. Here, we consider the 20 different relevance scores and the mean relevance score for each feature, excluding features with 0 relevance. Additionally, we consider the minimum, maximum and mean prediction values for the 20 model samples, shown in the bottom of the diagrams. This is done separately for both the sepsis positive and sepsis negative class. An example of the resulting diagrams from this approach for the sepsis positive class can be seen on Figure 12.1, though note that the diagrams shown here only show the most relevant features. Full diagrams can be seen in Appendix L.

The second approach is to take a broader view at the distribution of relevance scores across data points. Here, we consider the relevance scores from 3,000 different data points from the testset of each dataset. The resulting relevance scores are then grouped together to create a diagram for each feature, showing the different relevance scores it received, based on its value. This is also done separately for the sepsis positive and sepsis negative class. To get a clearer insight into how the features affect predictions for the two classes, we filter the results to only show true positives for the respective class, with a probability threshold of 0.9. This filters out the low relevance scores from true negative predictions, as well as outliers with unusually high relevance scores from uncertain predictions. For ΔDCT, this threshold is lowered to 0.5, since positive predictions are generally lower for ΔDCT, as mentioned in Chapter 10. An example of this type of diagram can be seen on Figure 12.4.

12.2 Most Relevant Features

In this section, we review the most relevant features used by the BNN to predict sepsis positive and sepsis negative, respectively. Note that the features used by the ΔDP-A and ΔDCT datasets are not identical (see Appendix D), and that ΔDCT contains more missing values (see Appendix C), which can lead to differences in relevant features.

12.2.1 PhysioNet

For ΔDP-A, the most dominant feature for predicting sepsis positive is 'ICULOS' (ICU Length of Stay), where a high 'ICULOS' value receives positive relevance, and a low 'ICULOS' value receives negative relevance. This can be seen on Figure 12.1 and Figure 12.2 respectively.

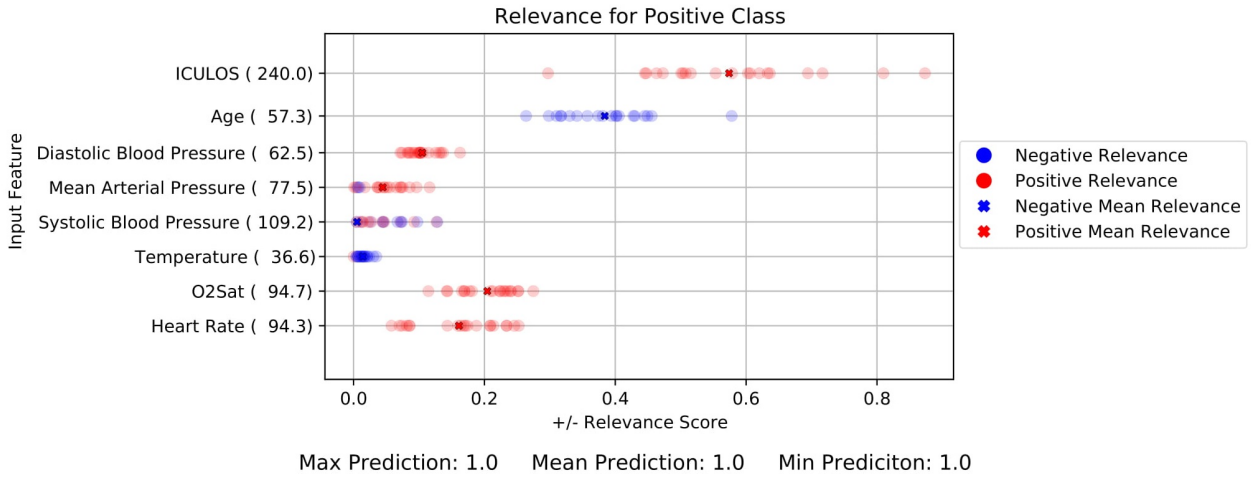


Figure 12.1: Sepsis positive data point from ΔDP-A with positive 'ICULOS' relevance for the positive class.

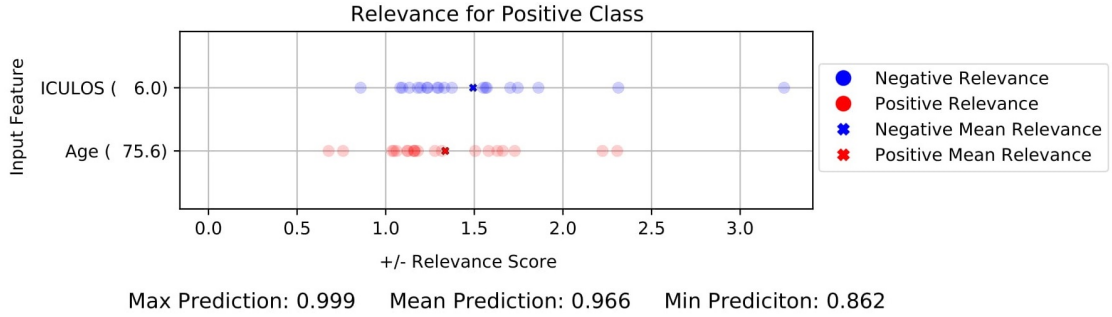


Figure 12.2: Sepsis positive data point from ΔDP-A with negative 'ICULOS' relevance for the positive class.

This indicates a pattern in the data, that patients with long admissions at an ICU are more likely to have contracted sepsis, which can be seen clearly on Figure 12.4. For sepsis negative predictions, we see the same pattern for 'ICULOS', but where a low 'ICULOS' value receives positive relevance, and vice versa (see Figure 12.3). This fits our expectations, as negative relevance for the sepsis positive prediction becomes positive relevance for the sepsis negative prediction.

Interestingly, the 'Age' feature seems to be inversely correlated to 'ICULOS', where predictions with positive relevance for 'ICULOS' has negative relevance for 'Age' and vice versa, as seen on Figure 12.1 and Figure 12.2. The correlation between 'ICULOS' and 'Age' results in mixed relevance scores for different 'Age' values, with little correlation between 'Age' value and relevance score, as seen on Figure 12.5 for sepsis negative. However, we do see that low 'Age' values tend to have low relevance, and that high 'Age' values have more data points with high negative relevance, suggesting a higher tendency towards sepsis positive.

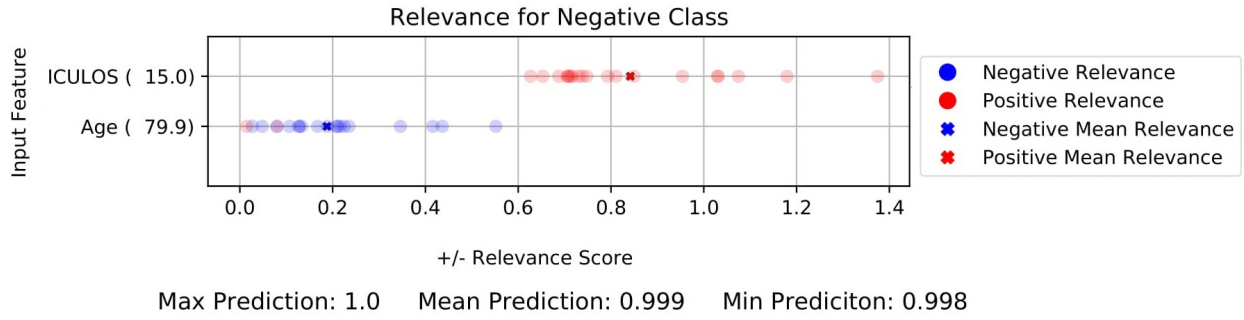


Figure 12.3: Sepsis negative data point from ΔDP-A with positive 'ICULOS' relevance for the negative class.

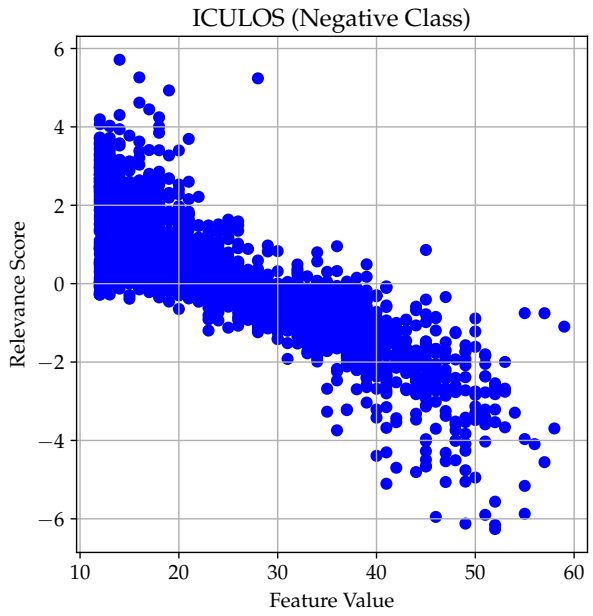


Figure 12.4: Overview of relevance scores for 'ICULOS' when considering sepsis negative predictions in ΔDP-A.

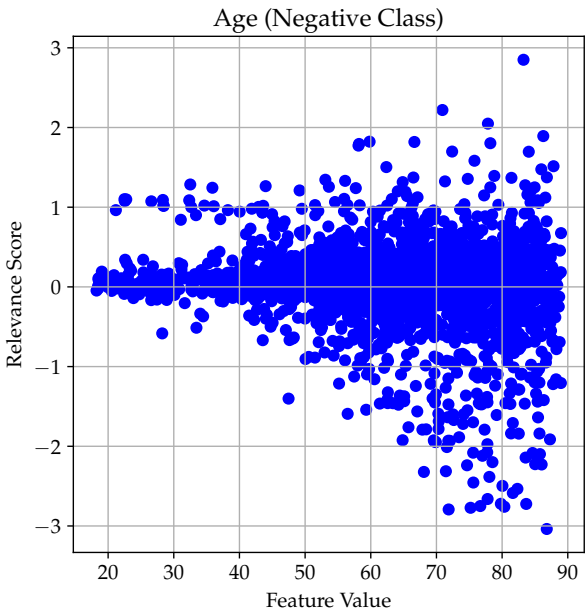


Figure 12.5: Overview of relevance scores for 'Age' when considering sepsis negative predictions in ΔDP-A.

Aside from 'ICULOS' and 'Age', we also see high relevance in the bottom six features on Figure 12.1: 'Diastolic Blood Pressure', 'Mean Arterial Pressure', 'Systolic Blood Pressure', 'Temperature', 'O2Sat' and 'Heart Rate', which are all vital signs. Of these features, 'O2Sat' and 'Temperature' show the clearest patterns, though note that they are still quite noisy. For 'O2Sat', we see increasing relevance for sepsis negative as the value rises towards 100 (Figure 12.6). This fits well with the fact that a normal 'O2Sat', the amount of oxygen in the blood, falls around $95 - 100\%$ [44]. For 'Temperature', we see a triangular shape for sepsis negative, where relevance increases towards 37 and decreases as the value moves above or below 37 (Figure 12.7). This also makes sense, as a normal body temperature is around 37°celsius.

Aside from the features mentioned here, 'Chloride', 'Glucose', 'Platelets', 'Hematocrit', 'SaO2' and 'PaCO2' also have notable relevance scores, but are much less prevalent in the data. It is somewhat disappointing that the most used feature is 'ICULOS', rather than one of the vital signs or lab values, as 'ICULOS' is not directly connected to the patient's health or the definition

of sepsis (see Appendix E). However, several other relevant features used by the model are more closely related to the patients health and sepsis.

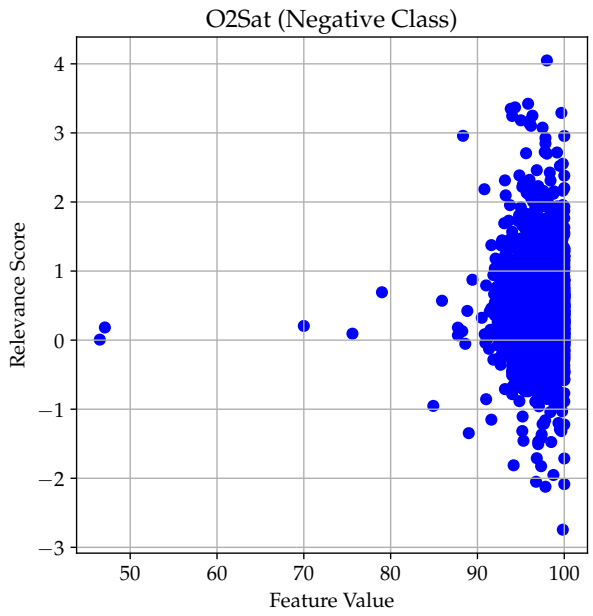


Figure 12.6: Overview of relevance scores for 'O2Sat' when considering sepsis negative predictions in ΔDP-A.

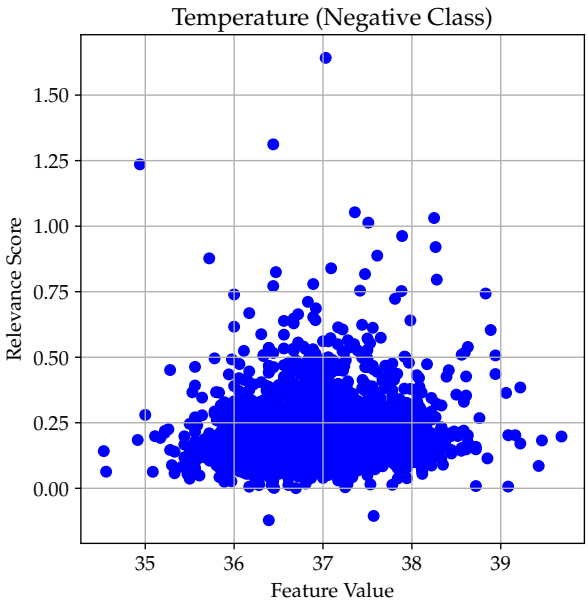


Figure 12.7: Overview of relevance scores for 'Temperature' when considering sepsis negative predictions in ΔDP-A.

We note that for many features, we see irregular relevance scores, that do not correspond to the pattern we otherwise see or expect. An example of this can be seen on Figure 12.6, where negative relevance also increases as 'O2Sat' increases, though at a smaller rate than the positive relevance. It is unclear what exactly causes this behavior, but the filtering of false positives and false negatives, along with the probability threshold of 0.9 to focus on certain predictions, helped reduce this behavior. We believe the most likely explanation is either that the model has not properly learned the patterns in the data, or that the patterns in the data are inherently noisy.

12.2.2 Cross-Tracks

For ΔDCT, we generally see that less features are present and that positive predictions are not as high as for ΔDP-A, with more uncertainty in the predictions. This is likely due to ΔDCT having more sparse data, as it is not from ICUs. We see a tendency that positive cases more often have more features, whereas some negative cases only have information for the six vital sign parameters, indicating that more measurements are done close to sepsis onset.

For the most certain sepsis positive predictions, the most relevant features are 'P(aB)-Sodium' and 'P(aB)-Chloride', as seen on Figure 12.8. 'P(aB)-Sodium' and 'P(aB)-Chloride' measure the levels of sodium and chloride in the blood, respectively, and have both been shown to be connected to sepsis [45, 46]. We do not see a clear pattern between the values for these features and the relevance scores they receive, but this may be due to the notable uncertainty present in the positive predictions.

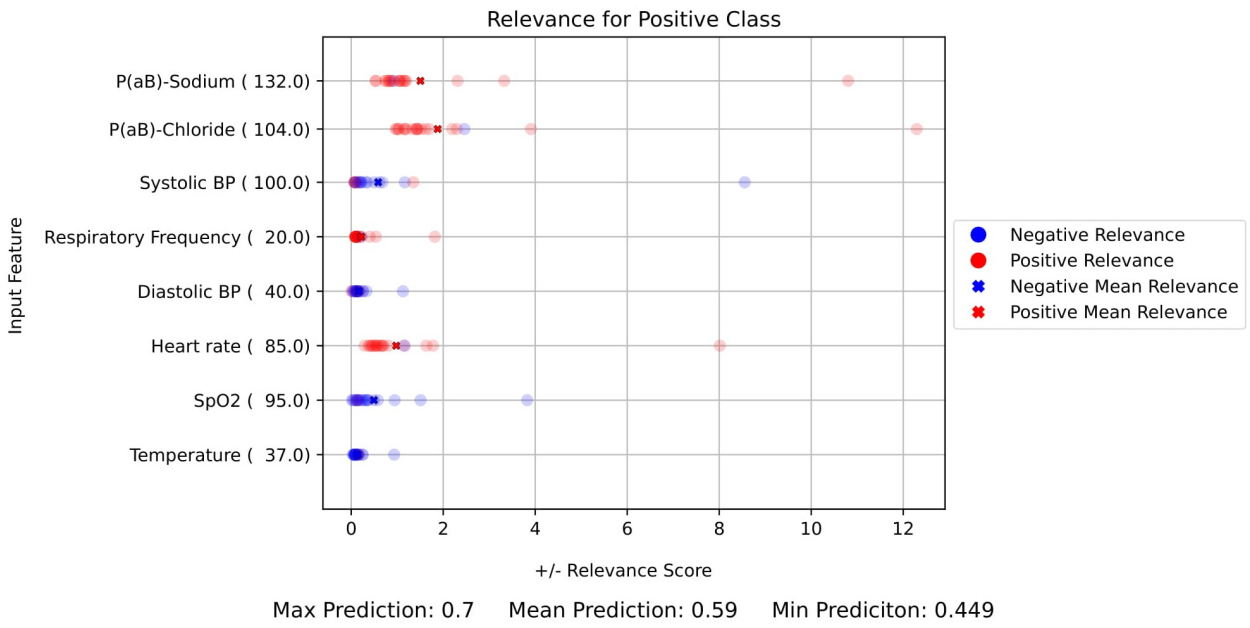


Figure 12.8: Sepsis positive data point from ΔDCT with positive relevance for 'P(aB)-Sodium' and 'P(aB)-Chloride' for the positive class.

As for ΔDP-A, we also see high relevance for the six vital sign parameters in our model trained on ΔDCT. This can be seen more clearly on Figure 12.9, which shows a prediction for sepsis negative. Here, we also see the clearest patterns for 'Temperature' and 'SpO2' (a measure for 'O2Sat'), shown on Figure 12.10 and Figure 12.11. The patterns are very similar to those from ΔDP-A, where sepsis negative relevance for 'SpO2' increases as the value increases towards 100, and sepsis negative relevance for 'Temperature' peaks around 37.

Note that for 'Temperature' and 'SpO2', relevance is almost always positive for the sepsis negative class. We see this pattern in several features in ΔDCT, where relevance is almost always either positive or negative. This may indicate that the model has found a pattern in some features, where a measurement simply being present for the feature has a certain effect. However, we also see that the actual value of the feature has a clear effect on the amount of relevance received.

It is interesting that we see similar patterns across datasets, with the main differences being the relevance for 'Age' and 'ICULOS' in ΔDP-A, which are both unavailable in ΔDCT, and the relevance for 'P(aB)-Sodium' and 'P(aB)-Chloride' in ΔDCT, where no sodium measure is available in ΔDP-A.

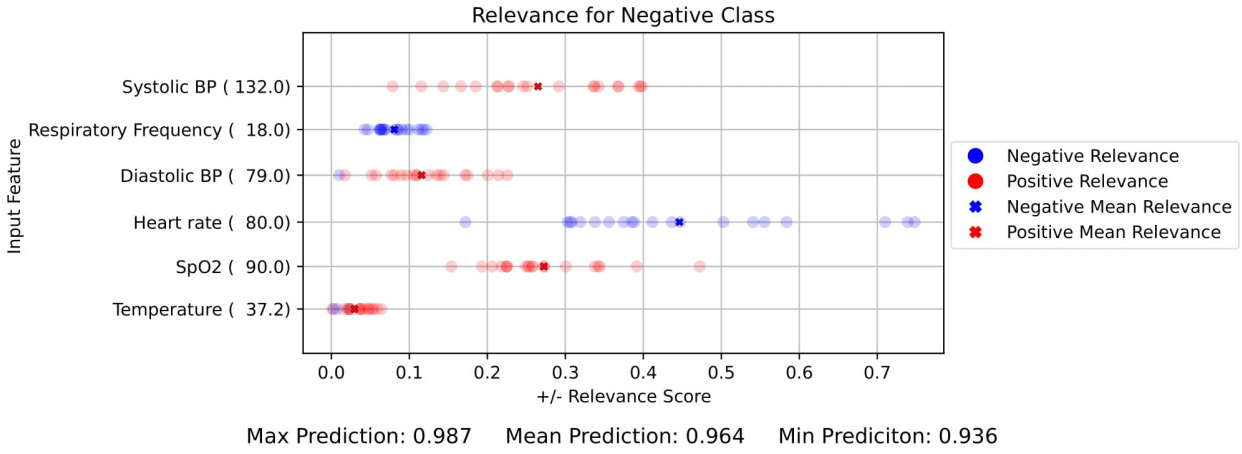


Figure 12.9: Sepsis negative data point from ΔDCT with relevance scores for vital sign features for the negative class.

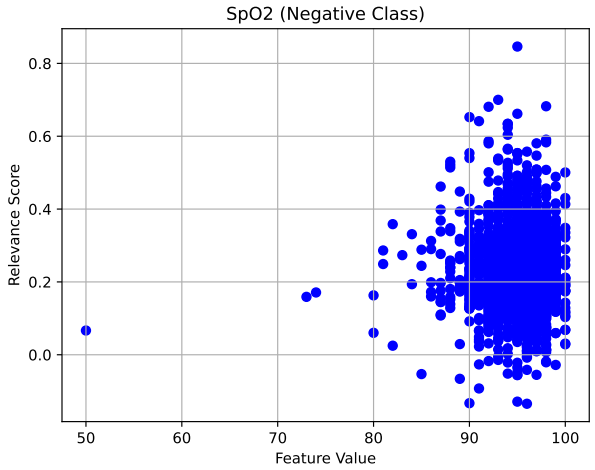


Figure 12.10: Overview of relevance scores for 'SpO2' when considering sepsis negative predictions in ΔDCT.

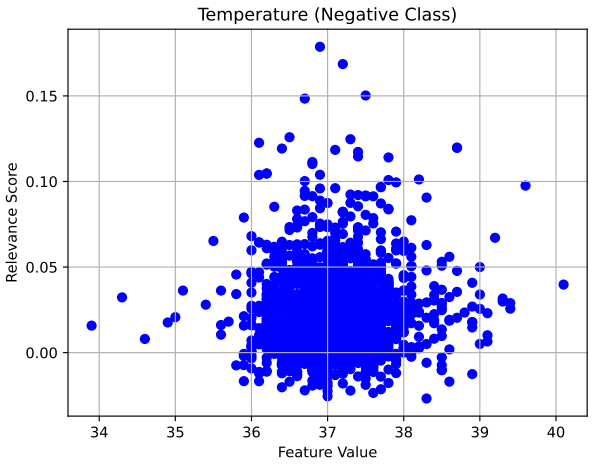


Figure 12.11: Overview of relevance scores for 'Temperature' when considering sepsis negative predictions in ΔDCT.

12.3 General Observations

In this section, we state some general observations we see during the LRP experiments.

The first observation occurs when one class dominates the prediction, e.g. when sepsis positive has a probability of around 1 and sepsis negative has a probability of around 0, or vice versa. Here, we see that the distribution of relevance from the dominating class is reflected in the distribution of relevance in the other class, where we see very similar results, but of course with much lower relevance scores. We believe this is due to the fact that the two classes share most of their parameters throughout the network, having only 20 unique weights each from the last layer, as seen on Figure 9.5. However, we still see a clear difference between relevance in high positive

predictions and relevance in high negative predictions, showing that the model is still able to use different features and patterns for sepsis positive and sepsis negative respectively.

Additionally, note that the relevance scores shown in this chapter are based on relatively certain predictions. However, when predictions are very uncertain, ranging between both high positive and high negative predictions, we see a more mixed result in relevance scores. Here, many features receive both positive and negative relevance for the same data point, as shown on Figure 12.12. We believe this shows how different relevance is assigned to the same feature value, based on whether the model tries to predict positive or negative. This can explain the mixed relevance scores, considering the shared weights in the network as explained above.

We also observe that the features with highest relevance are often features with high values. For example, 'ICULOS' can have values above 100, whereas the delta values are always comparatively low, typically with values below 10. We believe this might affect the patterns found by the model through training, which can explain the way relevance is distributed to high valued features. Therefore, we perform a short experiment where we normalize ΔDP-A such that all features have values between -1 and 1, and train a BNN using this dataset. However, this results in lower performance, and does not show significant changes in the distribution of relevance (e.g. 'ICULOS' still receives high relevance), other than the gender and unit features receiving higher relevance. An example can be seen on Figure 12.13. Based on these results, we choose to not use the normalized dataset.

Lastly, we see a general tendency that predictions with high uncertainty also show high variance in relevance scores, as well as having outliers with high scores, especially for some of the most relevant features. This shows that the variance in relevance scores are affected by the uncertainty of the predictions. We therefore expect that the uncertainty in the predictions can be explained using the variance in relevance scores, which we evaluate further in the experiment in Chapter 14.

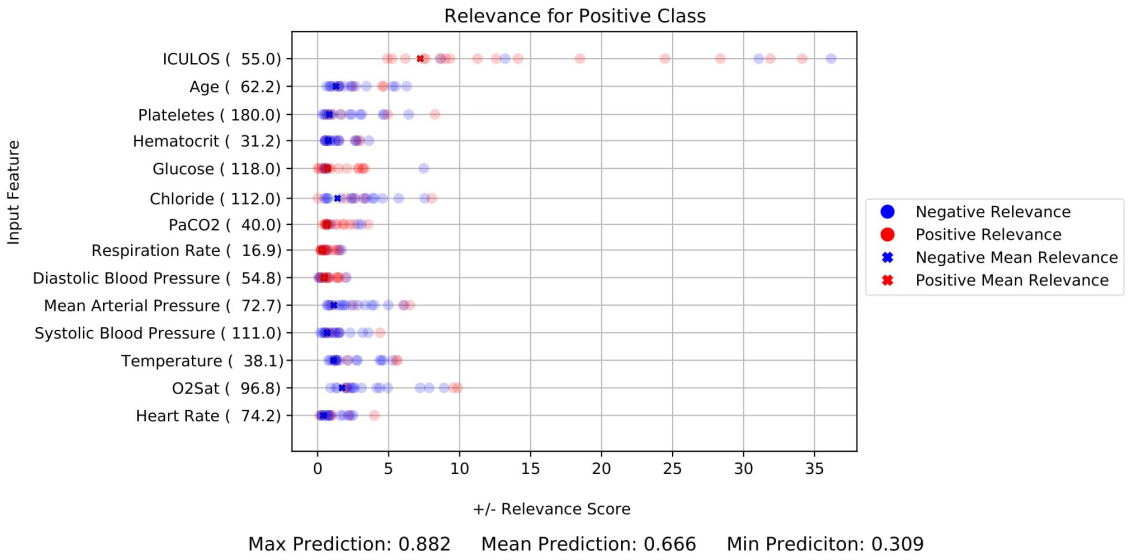


Figure 12.12: Sepsis positive data point from ΔDP-A with high uncertainty and mixed relevance scores.

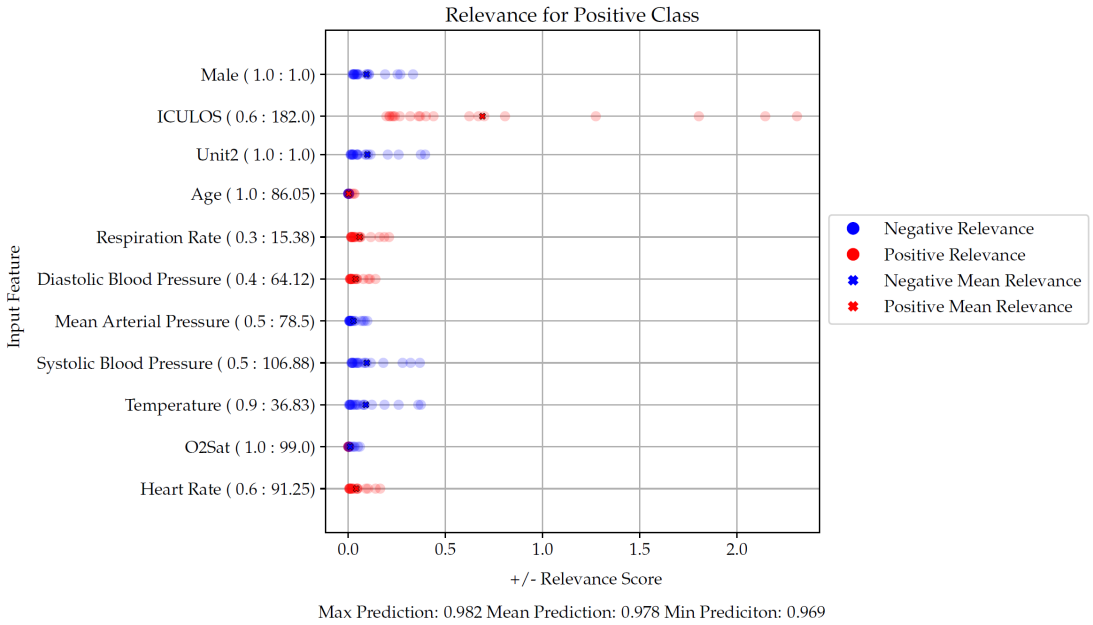


Figure 12.13: Sepsis positive data point from normalized ΔDP-A.

Chapter 13

Weight Uncertainty in First Layer

In this section, we cover an experiment of inspecting the first layer of either a fully Bayesian NN, or a BNN where only the first layer is stochastic. From the chapter discussing the results of experimenting with LRP (Chapter 12), we find that when conducting LRP on multiple sampled models, some features have larger variance compared to others. Based on these findings, we want to explore whether weight uncertainty in the first layer has a correlation with features that provide the most uncertainty to a prediction. Here, we represent the probability distributions over the weights, by plotting the mean and the standard deviation of the weights connected to a specific feature.

The chapter is split into two subsections, based on what is plotted on the x-axes, namely standard deviation and mean. Throughout this chapter, we only include the most interesting parts of the plots (full plots can be found in Appendix M).

13.1 Standard Deviation

In this section, we inspect the plots where the standard deviation of the first layer is plotted on the x-axis, for both the fully Bayesian NN and the BNN with one stochastic layer. When observing Figure 13.1, we can see that the standard deviation is generally very similar across features. Here, no clear patterns are visible, and the randomness of the standard deviation is a reoccurring pattern across datasets and the two different models featured in this experiment. This is a little surprising, as the idea to experiment with a BNN, in which only the first layer is stochastic, came from the assumption that, when limiting the layers in which uncertainty can be expressed, the standard deviation of the remaining layer has to compensate, which would then result in more noticeable changes.

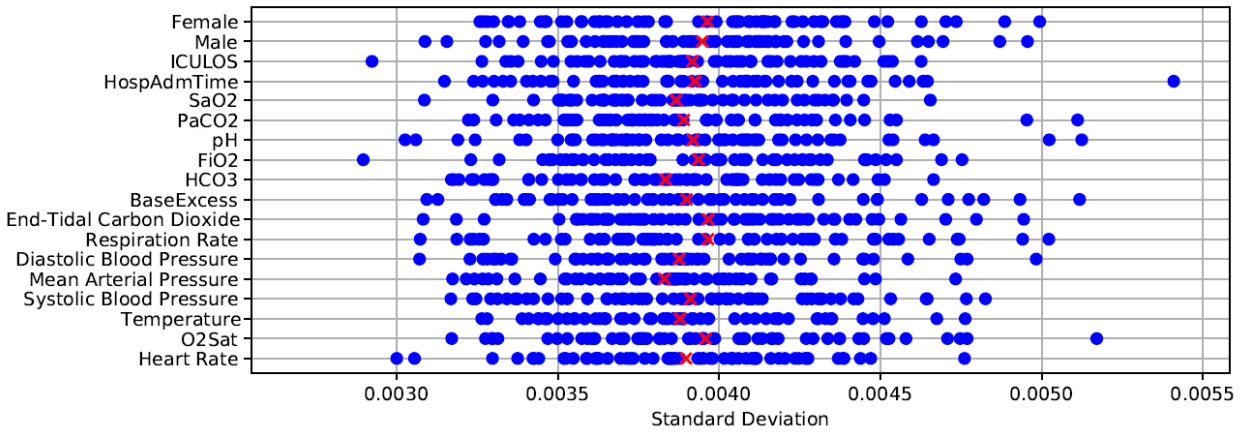


Figure 13.1: Standard deviation plot for probability distributions over weights in the first layer of a fully Bayesian NN on ΔDP-A. The full figure can be viewed on Figure M.1 in Appendix M.

13.2 Mean

In this section, we inspect the plots where the mean of the distributions over weights in the first layer is plotted on the x-axis, for both the fully Bayesian NN and the BNN with one stochastic layer. By inspecting Figure 13.2, we can see that some features have fluctuating mean values, whereas others have mean values that are very close to 0. For example, when looking at 'ICULOS' and 'Age' in Figure 13.2, we see that the absolute values of the means are high, and that the mean values have a high variance. When comparing to the results from Chapter 12, we can see that these features also often have large relevance scores. The same pattern can be seen for the features: 'Diastolic Blood Pressure', 'Mean Arterial Pressure', 'Systolic Blood Pressure', 'Temperature', 'O2Sat' and 'Heart Rate'. This means that if a feature has higher weights, because of higher mean values, these also receive higher relevance scores. This is probably due to how LRP propagates most of the relevance through high weights. This also means that these features have a greater impact on the output of the model, compared to features which mean values are centered around 0, for example 'End-Tidal Carbon Dioxide'.

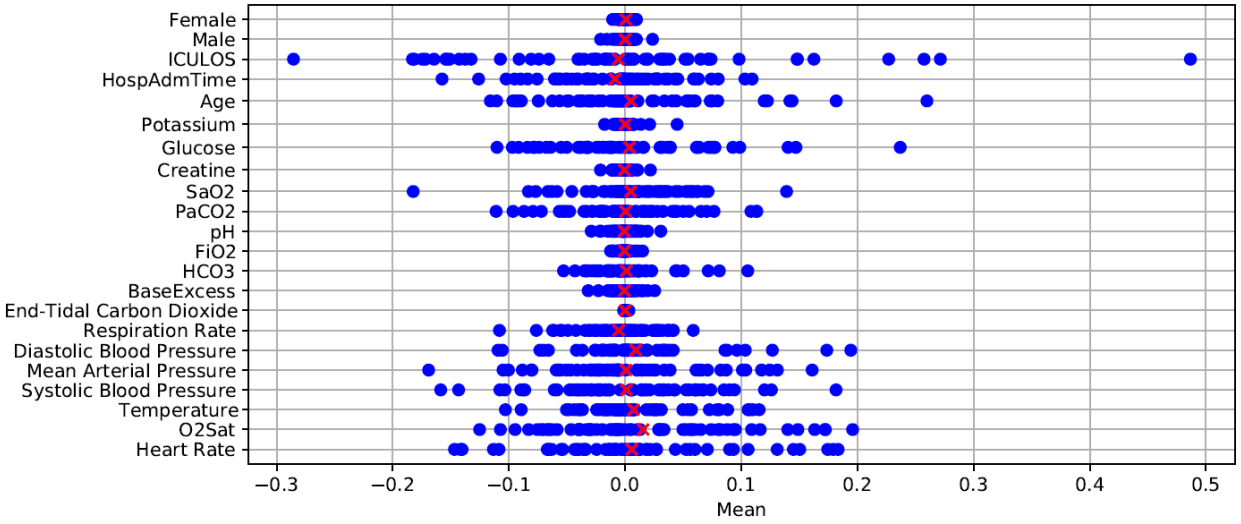


Figure 13.2: Mean plot for probability distributions over weights in the first layer of a fully Bayesian NN on ΔDP-A.

Figure 13.3 shows the BNN where only the first layer is stochastic. Here, we see the same patterns as we see in the fully Bayesian model on Figure 13.2.

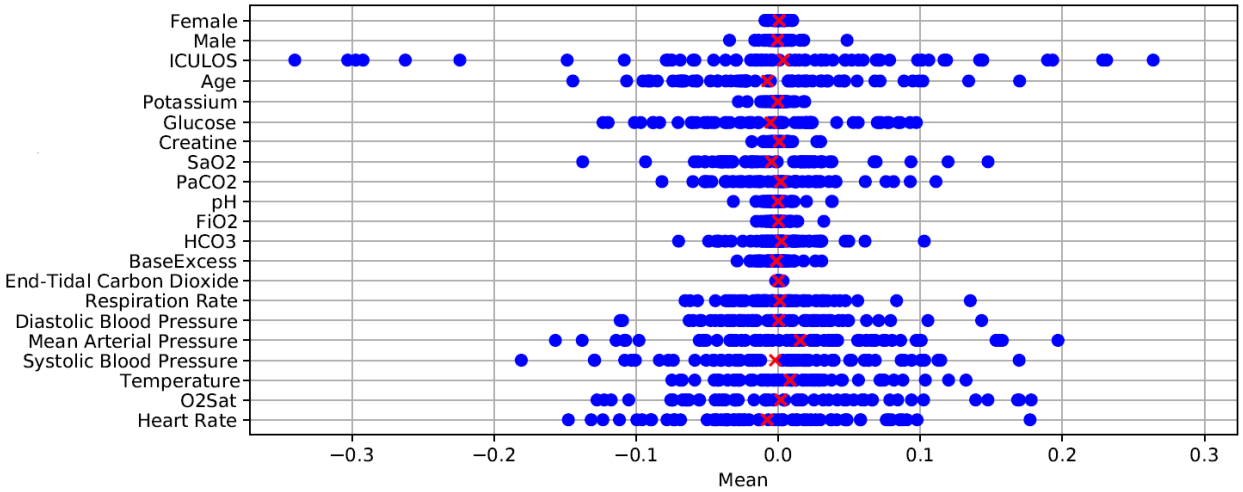


Figure 13.3: Mean plot for probability distributions over weights in the first layer of a BNN where only the first layer is stochastic, on ΔDP-A.

On Figure 13.4, the mean values for distributions in the first layer are plotted for ΔDCT. Here, we see similar patterns to Figure 13.3 and Figure 13.2, where some features have fluctuating mean values, whereas others are centered around 0. Interestingly, many of the features shared between the two datasets also share a correlation in the magnitude of the mean values. For example, 'HeartRate' and 'Systolic Blood Pressure' both have high mean values, whereas 'Potassium' has low values. However, we also see disagreement in some features, for example, 'Creatine' and 'Glucose'.

As ΔDCT is not composed of ICU patients, we see a lot more missing values, as observed in Section C.3, with the vital signs being the most observed features. This likely contributes to why these features are more important to the model.

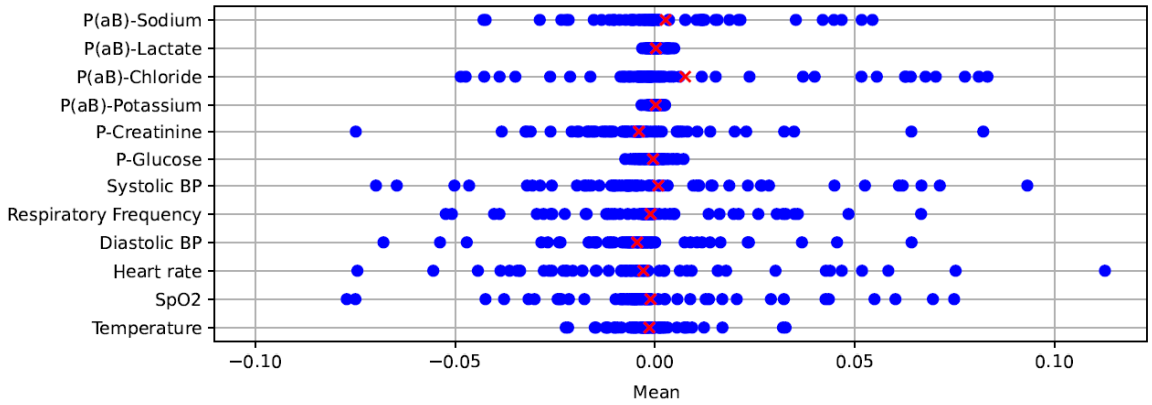


Figure 13.4: Mean plot of the first layer on a fully Bayesian network on ΔDCT.

13.3 Summary

Generally, we see that the absolute mean values of the first layer both in a fully Bayesian NN and a BNN with one stochastic layer, indicate which features the model base its predictions on. We find it surprising that such a clear pattern can be seen by only considering the first layer.

Regarding the standard deviation, we expect that there is a connection between the variance in the relevance scores of a feature, and the standard deviation of the weights connecting to that feature. However, we see no apparent correlation between the plots for standard deviation and the relevance scores of multiple sampled model. Therefore, without investigating other ways of representing the standard deviation in the first layer, we do not see much potential in using the first layer as an indicator of which features affect the uncertainty in predictions.

Chapter 14

Evaluating Uncertainty in Relevance Scores

In Chapter 12, we see a correlation between the uncertainty in predictions and the variance in the relevance scores of input features. Therefore, in this chapter we evaluate the method of using LRP to explain which features have a large impact on the uncertainty in predictions, as described in Chapter 8.

In [47], Binder et al. discuss a method for evaluating the relevance scores computed using LRP. This method is called pixel flipping, and is related to image classification, where a pixel's value is set to the negated value (flipped pixel $=$ pixel $\cdot\,(-1)$). Here, Binder et al. observe, that when flipping a pixel that has high relevance for the prediction, the confidence is reduced. However, since we do not use pixel data, we adapt it in the form of feature flipping, such that flipped feature $=$ feature $\cdot\,(-1)$. Even though there are inherent differences in flipping a single pixel in a picture with 400 pixels, and flipping a feature in a dataset consisting of 75 features, we still see this as an interesting experiment. Therefore, we want to evaluate how the variations in relevance scores are affected when using feature flipping.

To gain a wide insight into how feature flipping affects different outcomes from LRP using multiple sampled models, we want to experiment with three different types of features. These are features with:

- High positive relevance together with high variance.
- High negative relevance together with high variance.
- Low relevance and low variance.

These three types of outcomes should make it possible to evaluate feature flipping, both for features which have positive impact, negative impact, and little impact on the predictions and the uncertainty in them.

14.1 Feature Flipping

When comparing the outcomes of feature flipping for both ΔDP-A and ΔDCT, we see very mixed results. For example, one outcome is that when flipping a feature, the predictions remain somewhat similar, with similar uncertainty in the predictions, as shown in Figure 14.1 and Figure 14.2. For other outcomes, either the uncertainty decreases or increases, or the prediction confidences decrease or increase, without following a clear pattern (The full set of results can be found in Appendix N.3 for ΔDP-A and in Appendix N.4 for ΔDCT). Therefore, we do not see feature flipping as a reliable procedure, when evaluating the method.

An additional argument against using feature flipping is, that often when flipping values, the features assume an out of distribution value, which the model is not trained on. This means that for many cases, the feature is assigned an impossible value (e.g. negative 'Heart Rate'), which we think results in unexpected predictions. This might also be the reason why the results from feature flipping show no clear patterns, when considering the predictions of the model and the uncertainty in these.

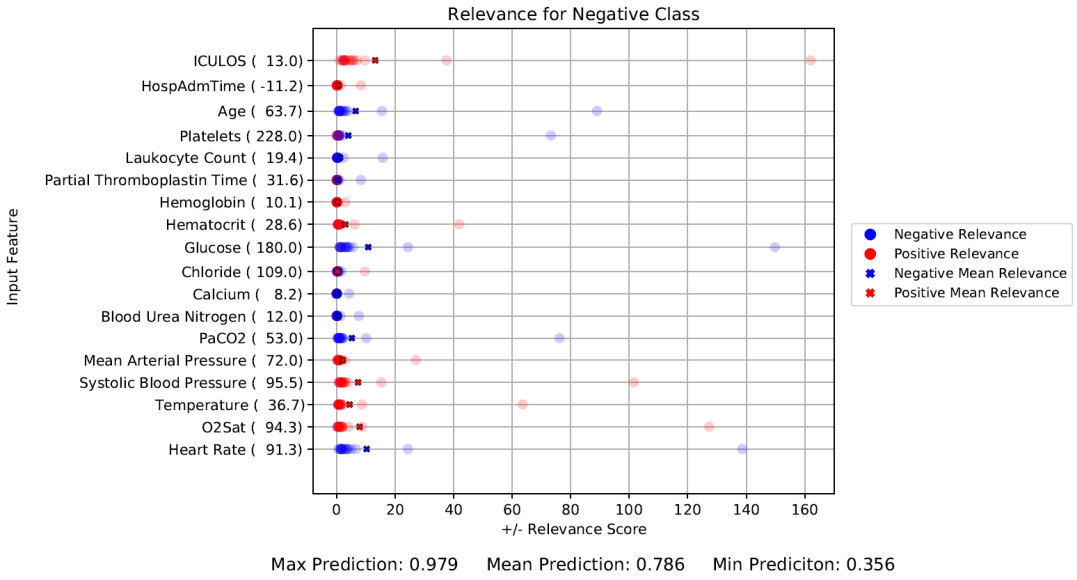


Figure 14.1: Negative prediction on negative data point from ΔDP-A. Features with low relevance and variance are excluded on the figure, full version can be seen on Figure N.1 in Appendix N.1.

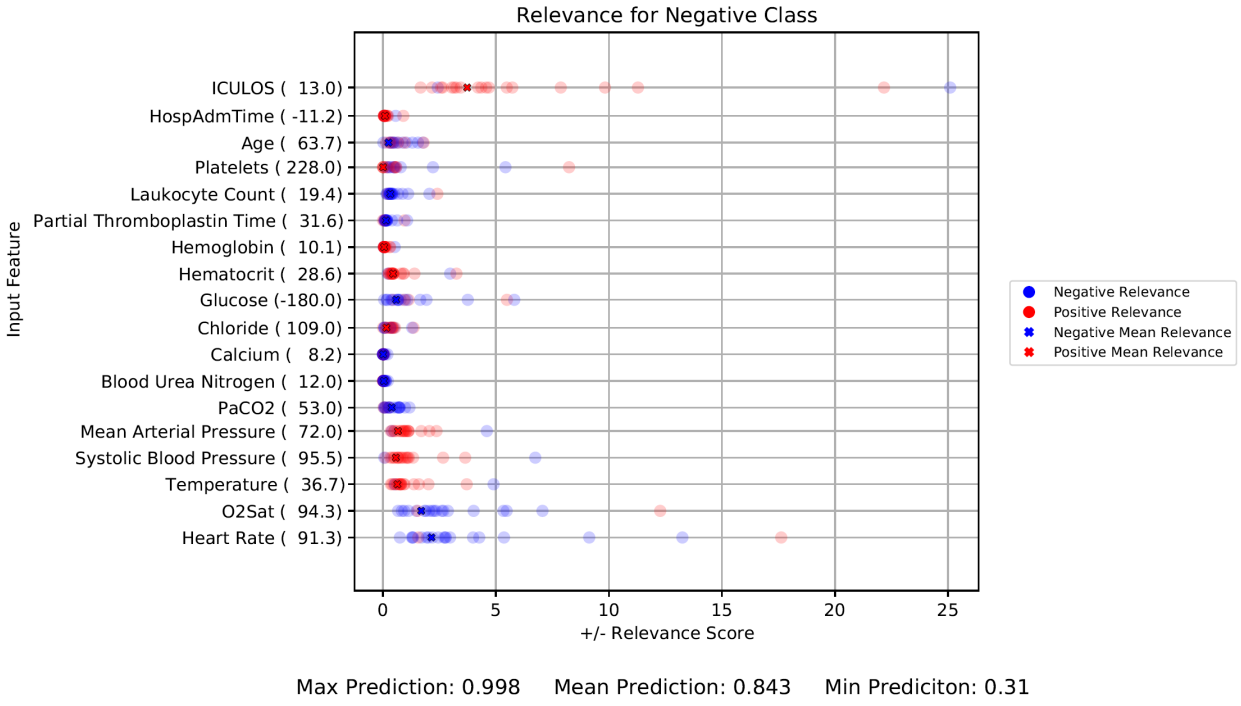


Figure 14.2: Negative prediction on negative data point from ΔDP-A, where 'Glucose' is feature flipped. Features with low relevance and variance are excluded on the figure, full version can be seen on Figure N.8 in Appendix N.3.

14.2 Setting Features To 0

As no useful results were found when using feature flipping for evaluating our method for explaining uncertainty in predictions using LRP, we want to try another approach. Here, we want to try to set different features to 0, such that those features have no relevance for the predictions. With this method, our expectation is that:

- When a feature has high relevance, the result of setting the feature value to 0 either pulls the predictions down, if the relevance is positive, or up, when the relevance is negative.
- When features with high variance in the relevance scores are set to 0, the uncertainty in the predictions decreases.

We acknowledge that when setting a feature to 0, the feature is not completely discarded from the data point, and is still given to the model as an input feature, which conveys information to the model.

14.2.1 Discussion of the Results

Contrary to feature flipping, we see more clear patterns in the outcomes, when features are set to 0.

Figure 14.4 shows the relevance scores for the same data point used in Figure 14.3, but where 'ICULOS' is set to 0. Here, we see that the relevance scores for 'ICULOS' in the original data point varies a lot, with both positive and negative relevance. The result of setting it to 0 is an increase in confidences in the predictions, and that the predictions has less variance. This fits our expectations, that setting a feature with high variance in relevance scores to 0 decreases the uncertainty in predictions. In addition to this, some features now provide relevance for the opposite class, where the biggest contributors for a positive prediction are the features 'Age', 'O2Sat', 'Heart Rate', and 'Chloride'. Also, note on the x-axes of the figures that the relevance scores are significantly lower for the more certain predictions, after setting 'ICULOS' to 0, meaning that the variance in the resulting relevance scores are significantly lower as well.

Another outcome of setting a feature to 0 is shown in Figure 14.5, where we set 'ICULOS' to 0 for the data point shown on Figure 14.1. Here, a short 'ICULOS' of 13 has positive relevance for the sepsis negative class, which means that 'ICULOS' pulls the prediction toward sepsis negative. When 'ICULOS' is set to 0, the model changes its predictions from mostly having a high confidence in a sepsis negative prediction, to being certain that the patient contracts sepsis. This means that without 'ICULOS', the model needs to use other features for the predictions, where it now finds high relevance in 'Platelets', but with negative relevance scores. This confirms that setting a feature with high relevance to 0 can push predictions towards the opposite class.

We also test the outcome of setting a feature with low relevance and low variance to 0. Here, we choose the feature 'BaseExcess' from the data point seen on Figure 14.3. As expected, setting 'BaseExcess' to 0 shows minimal changes in the other relevance scores and the uncertainty of the predictions, shown on Figure N.22.

The full set of results can be seen in Appendix N.5 for ΔDP-A and in Appendix N.6 for ΔDCT. Generally, the results confirm our expectations. For all data points, we see that when setting a

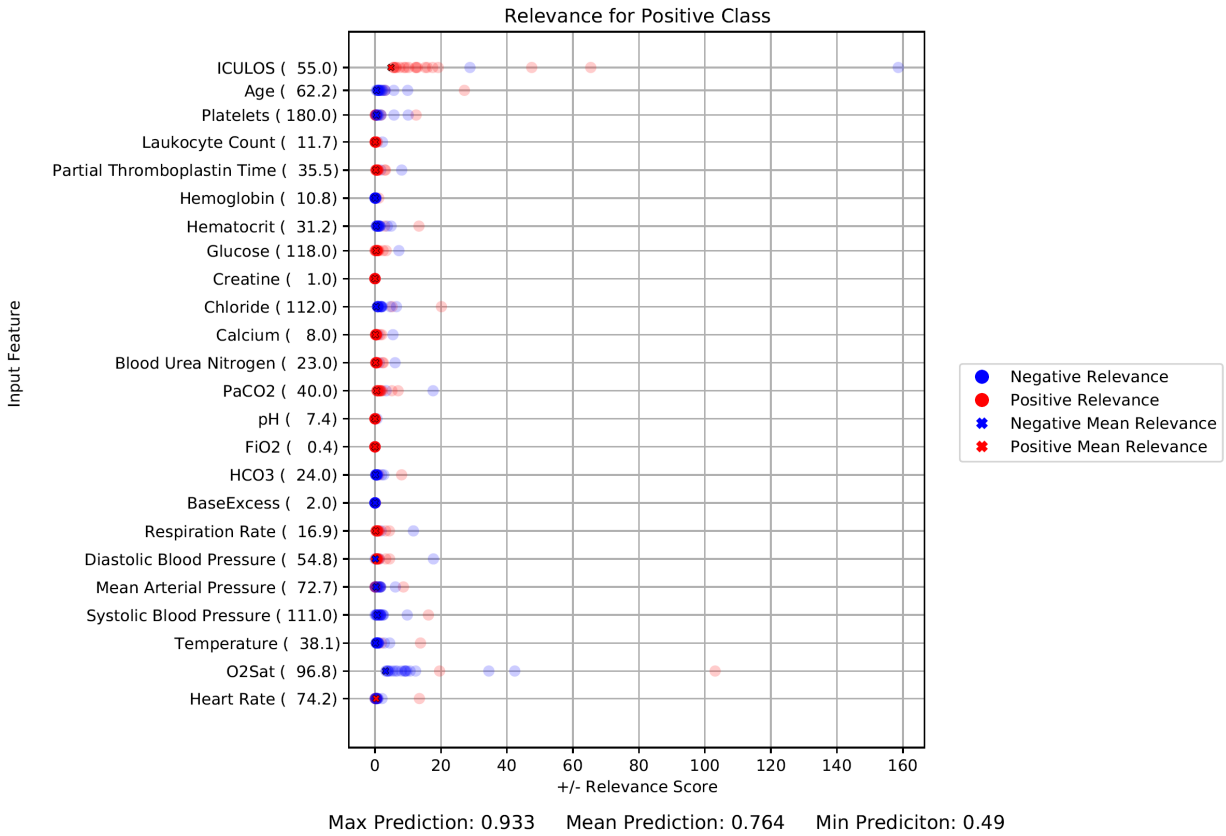


Figure 14.3: Positive prediction on positive data point from ΔDP-A. Features with low relevance and variance are excluded on the figure, full version can be seen on Figure N.2 in Appendix N.1.

feature with low relevance and low variance to 0, the predictions and the uncertainty in the predictions only change minimally. When setting features with high relevance to 0, we also see results that follow our expectations, however with few situations where it seems like too little information is available, resulting in the model having lower confidences in its predictions. For most of the cases, the confidences increase when setting a feature with negative relevance scores to 0, and oppositely it decreases for a feature with positive relevance.

We find that, when setting the value of a feature to 0, this affects the uncertainty in the prediction. For example, if a feature has a lot of variance in its relevance scores, compared to the other features in the prediction, setting this feature to 0 yields a decrease in the uncertainty of the prediction. We find this most evident for ΔDP-A. For example, for the negative data point in Figure 14.1, when setting 'Glucose' or 'ICULOS' to 0, it results in less variation in the predictions (shown in Figure N.20 and Figure 14.5 respectively). Note that for 'ICULOS', the prediction becomes certain for the opposite class in this case. Here, 'Glucose' and 'ICULOS' are chosen, as these have high negative and positive relevance, respectively, as well as high variance in relevance scores. Also, when setting 'O2Sat' or 'ICULOS' to 0 for the positive data point in Figure 14.3, the uncertainty in the predictions decreases, shown in Figure N.24 and Figure 14.4 respectively. Again, 'O2Sat' and 'ICULOS' are chosen as these have a high variance in relevance scores.

For data points from ΔDCT, we often find that when setting features with high positive relevance to 0, the confidence in the predictions decreases, sometimes resulting in more uncertainty.

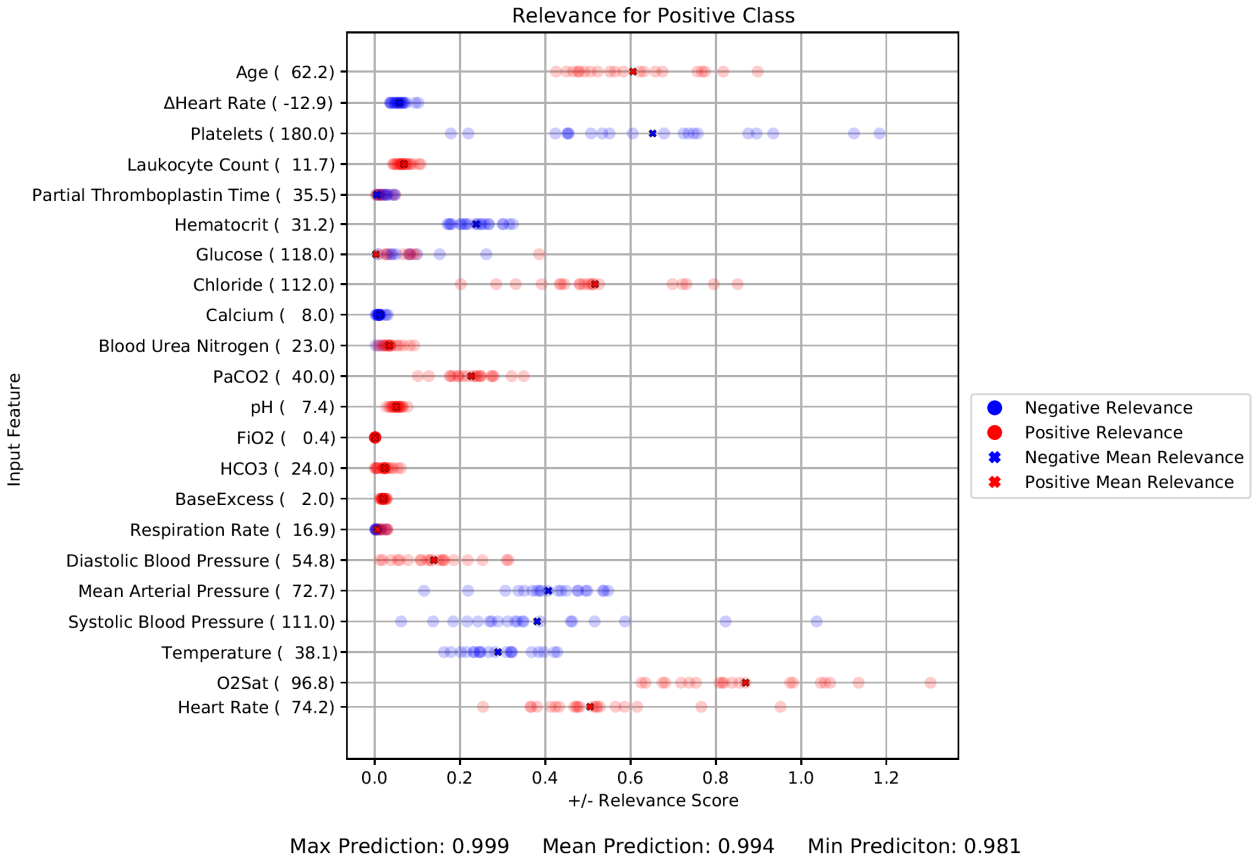


Figure 14.4: Positive prediction on positive data point from ΔDP-A where 'ICULOS' is set to 0. Features with low relevance and variance are excluded on the figure, full version can be seen on Figure N.23 in Appendix N.3.

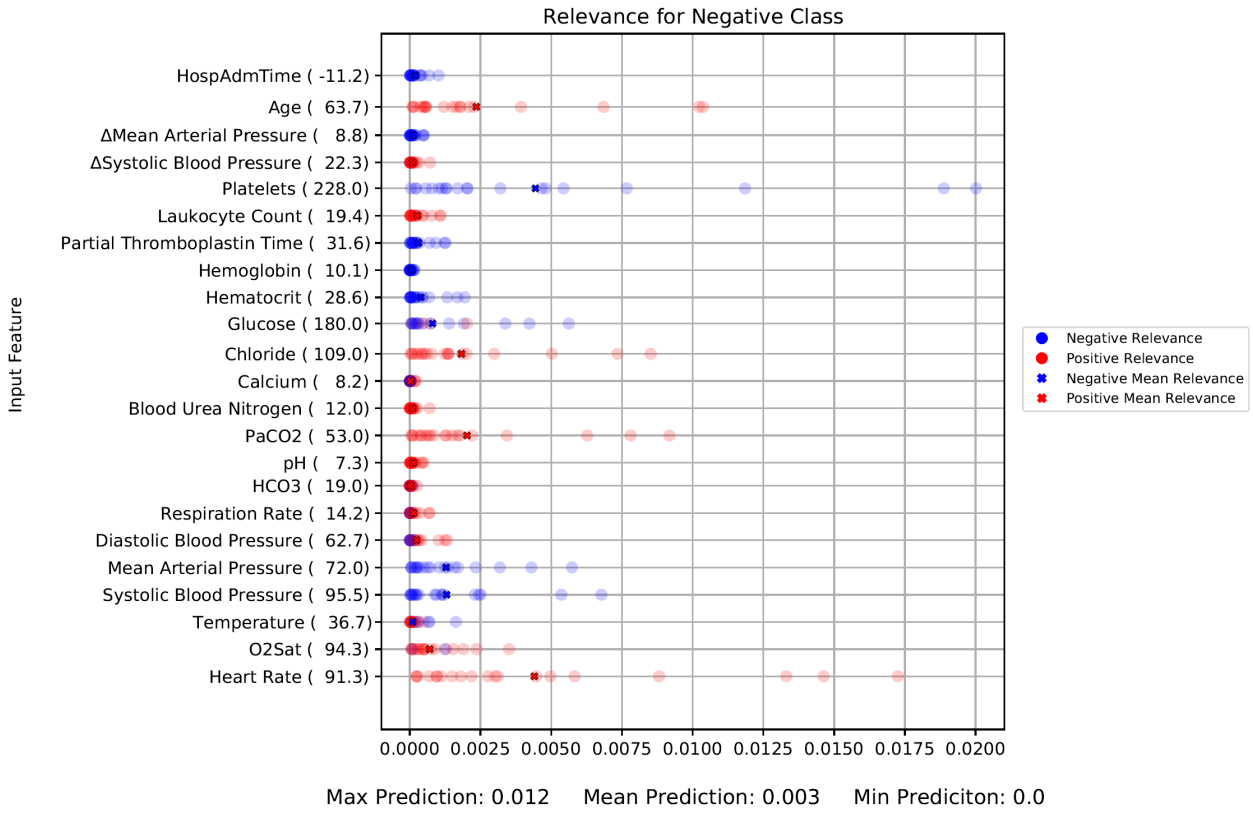


Figure 14.5: Negative prediction on negative data point from ΔDP-A where ICULOS is set to 0, features with low relevance and variance are excluded on the figure, full version can be seen on Figure N.21 in Appendix N.3.

Oppositely, we find that setting features with high negative relevance to 0 results in more certain predictions, generally with higher confidence. We think that this might be a result of the data points from ΔDCT generally containing fewer measurements, and therefore, that the uncertainty in the predictions are more affected by changes to the data points, than for ΔDP-A.

To conclude on our expectations from this experiment, we mostly find results that confirm these. For ΔDP-A, it is clear that the variations in the relevance scores is a result of uncertainty in the predictions. Setting a feature with high variance in its relevance scores to 0 clearly affects the uncertainty in the predictions, whereas setting a feature with low variance to 0 has a much smaller impact on the uncertainty. On ΔDCT, the results are less clear regarding uncertainty, likely due to the smaller amount of information in data points. However, it is clear that, when setting features with high relevance scores to 0, it has a direct relation to the confidence in the predictions.

Part IV

Evaluation

Chapter 15

Discussion

In this chapter, we discuss interesting findings and results found during this project.

15.1 Uncertainty in Deep Learning

In this project, we have chosen to use variational inference as our method for quantifying uncertainty. We have created a BNN that is comprised of elements from TensorFlow and TensorFlow Probability, such as DenseVariational layers and distributions. We argue that by using variational inference, we have been able to make a more in-depth analysis of uncertainty, that would otherwise not have been possible using either of the other methods discussed in Chapter 2. For example, the experiment discussed in Chapter 13 about analyzing mean and standard deviation of the distributions over the weights, would have been troublesome with MCMC, as distributions over the weights have to be constructed manually. We also see the use of MC Dropout and Deep Ensembles as being implausible for this, since either no distribution can be found for the weights, or the relation between weights from multiple trained models is unknown.

For the experiments with LRP, we sample models by sampling each layer from the DenseVariational layers of the BNN, and propagate relevance through each resulting model. This can also be done by training multiple models or by doing inference in an MC Dropout model multiple times. This means that these experiments can be conducted with either of the methods for quantifying uncertainty from Chapter 2, and that our method could be applied for either of the methods.

We think the choice of using variational inference provided us with a problem of finding the correct hyperparameters. During development, a lot of hyperparameter tuning was required to find a BNN with good performance. However, we also see that with good parameters, the BNN often gets better performance, is better calibrated, and does not overfit as easily, compared to a corresponding FFNN.

As mentioned in Chapter 8, the variance in relevance scores represents the total uncertainty of the predictions, where the aleatoric uncertainty comes from the predictions, and the epistemic uncertainty comes from the variance in sampled parameters. Instead, we can specifically consider the epistemic uncertainty, by propagating a value of 1 as the total relevance, rather than the predicted probability of sepsis for each sampled model. This eleminates the aleatoric uncertainty from the relevance scores, and since the distribution of relevance among features is exclusively based on the parameters and activations, propagating the same value through all sampled models means that the resulting variance in relevance scores will specifically represent the epistemic uncertainty.

15.2 Dataset Framing

In Chapter 4, we discuss that "Sliding Window" and "On Clinical Demand" best fit the clinical environment for ΔDP and ΔDCT respectively. However, through our conducted grid search experiments, we were not able to find a model with sufficient calibration on either of these datasets, so we opted to use the "Fixed Time To Onset" data framing instead. Here, we acknowledge that limited search has been done in terms of model architecture, as we have prioritized keeping the architecture simple, due to the novelty of our method. However, as the method we propose is based on LRP, other architectures can be used (if they allow for stochastic elements, and sampling of multiple models), though handling of special layers will have to be considered.

We do not think the "Fixed Time To Onset" data framing is ideal, as we argue it does not properly reflect the clinical environments from which the two datasets are gathered. Therefore, when using a model trained on the "Fixed Time To Onset" framing, the model will inherently be best at predicting a TP when the patient contracts sepsis 12 hours later. In addition to this, Lauritsen et al., in [26], find that when using "Fixed Time To Onset", they see an erroneous correlation between relevance scores for 'SpO2', whereas the other framing methods seem to find the correct correlation. Also, during the experiments, we find some results that we likewise speculate is affected by the framing. For example, when experimenting with LRP using ΔDP-A, 'ICULOS' often has a large relevance score compared to the other input features, which seems wrong as 'ICULOS' is not directly related to the health of the patient. We believe that using another data framing than "Fixed Time To Onset", which better fits the clinical context of the datasets, might lead to the model learning better patterns from the data, for example where 'ICULOS' receives less relevance. Another way of mitigating the problem, is to use the knowledge gained from the experiment in Chapter 12 to do feature engineering. Here, Ribeiro et al. [29] show, that information from explanations can be used for feature engineering on the dataset, in order to make a classifier use the "correct" features in the dataset.

Besides the unexpected patterns for 'ICULOS', when considering 'Age' from ΔDP-A, we see an unclear tendency that higher age results in higher negative relevance scores for the negative class, which means that it pulls towards a positive sepsis prediction. However, we also see an unclear tendency, that a higher age results in higher positive relevance scores for the negative class, though generally with lower scores. This seems counterintuitive, as the hospitalization rate for sepsis patients 65 years and above is much higher than for those under the age of 65 [48]. In Chapter 14, we also observe that when setting 'ICULOS' to a value of 0, we often see that this flips the relevance of the age feature. Based on this finding, we think that it would be interesting to train a new BNN on a dataset where 'ICULOS' is omitted, or maybe even all demographic features. Additionally, we find it interesting to apply other framing methods than "Fixed Time to Onset", though this means that additional investigation is needed, in order to find a more well calibrated model.

15.3 Experiments

In Section 14.2, we discuss the experiment about setting feature values to 0. Even though feature values are set to 0, information is still fed into the network. However, this information can be interpreted as the feature being either missing or having a value of 0. For the experiment about LRP, we argue that it still works as a way for evaluating which features uncertainty in the predictions stems from, as features with a value of 0 have no relevance. Considering this method, a feature with high variance in relevance should have a high effect on the uncertainty in predictions. Setting the feature to 0 means that the feature receives no relevance, and can therefore not be the contributing factor to the uncertainty of the prediction, according to our method. Arguably, changing the value of the feature to 0 causes a notable change in the data sample, which in theory can lead to more uncertainty. However, through our results we see a consistent pattern on ΔDP-A that setting the feature with highest variance to 0 makes the predictions more certain, which supports our expectations. For ΔDCT, this pattern is not as clear, which may be due to a lower amount of information being present in the data samples, meaning that setting a feature with high relevance to 0 removes too much information. However, this does not rule out the possibility that the feature being set to 0 is a highly contributing factor to the uncertainty in the predictions.

Another approach is to introduce two input values for each feature, one with the original value and one stating if the value is missing. This allows for another procedure, where the model might distinguish between when a feature is missing and when it has a value of 0. This can allow the model to find patterns in the presence of measurements, and allows missing values to receive relevance from LRP through the value stating whether the feature is missing. This might result in a more in-depth analysis of how missing values affect the uncertainty in predictions. However, for now we see the current evaluation as being sufficient for concluding that features with high variance in the relevance scores contribute more to the uncertainty in the predictions.

In the experiment about allowing the model to abstain from making a prediction, Chapter 11, we find that the number of FN and FP can be reduced by abstaining with either disagreement or standard deviation, with disagreement yielding the best results. However, we also discuss that for different limits and thresholds, the boost in performance is accompanied by tradeoffs in other metrics. Therefore, more investigation is needed for evaluating this approach in a specific context, for example by using net benefit analysis as discussed by Lauritsen et al. [24]. For example, a higher cost can be given to FNs, as these might result in situations where patients are discharged from the hospital, but contract sepsis within the next few hours, or higher cost for FPs, if treatment of healthy patients are expensive. So, even though we cannot conclude that the approach results in lower costs in specific situations, we still get better AUPRC and AUROC for some limits and thresholds, and we therefore argue that it will be useful in some situations.

Chapter 16

Conclusion

In the Introduction in Chapter 1, we emphasize that trustworthiness is important when deploying AI for clinical use. Here, we mention that trustworthiness can be achieved if a model is explainable, and if it can reason about its uncertainty. We discuss that calibration is important for trustworthiness, but is insufficient for capturing model uncertainty. Therefore, based on an evaluation of state of the art methods for quantifying uncertainty and explanation, we create the following problem statement in Chapter 6:

> **How can the uncertainty of a neural network for predicting sepsis be explained using LRP, such that is possible to determine the uncertainty contribution of each feature?**

To answer the first subquestion in the problem statement, we create a BNN trained with Variational Inference in Section 7.1, which is able to express uncertainty in its predictions. To answer the second subquestion, we investigate the ECE and ACE of said model. Here, we compare the calibration scores to an equivalent non stochastic FFNN, and find that in comparison, the BNN is well calibrated, which is also found by Jospin et al. in [17]. To answer the third subquestion, we propose a novel method in Chapter 8 for explaining which features the uncertainty in predictions stems from. Here, we use the BNN to sample multiple models, where each of these models are explained by using LRP. These different explanations are then used to consider the variance in relevance scores, which represents the uncertainty contribution for each feature in the prediction.

In Chapter 12, we show the results of using the proposed method, where we see that when there is high variance in the predictions, there is also high variance in the relevance scores for the features. We evaluate these observations in Chapter 14, where we set certain input features to 0 and observe its effect on the variance in the predictions. When setting features with high variance in relevance scores to 0, we often see a decrease in the variance of the predictions. However, this is not always the case on ΔDCT, which we speculate might be due to the sparsity of the data, which results in fewer features the model can base its prediction on. Based on this, we conclude that there is a clear correlation between variance in relevance scores and uncertainty in predictions, and that the proposed method can be used to explain which features contribute to uncertainty in predictions, on the datasets described in Chapter 4, with the model described in Section 7.1 and Chapter 9.

Chapter 17

Future Work

In this chapter, we present some additional experiments that we find interesting, but which we were not able to fit into the scope of this project.

Firstly, we would like to experiment with different architectures, in order to resolve the problem with badly calibrated models on the data framing "Sliding Window" and "On Clinical Demand". This would make the models more compelling for use in their respective clinical environments. Additionally, we would also like to experiment with other uncertainty quantification methods, and compare these to the BNN developed in this project. In theory, methods where multiple models can be sampled, will work with the proposed method. Therefore, we find it interesting to evaluate if there is any noticeable differences in how the uncertainty is quantified and how this influences the relevance scores and variance in these.

Additionally, we would like to try isolating the epistemic uncertainty, by propagating a relevance score of 1 through LRP, rather than the predicted output, as explained in Chapter 15. We can then compare the results from this with the results from Chapter 14, where we consider the total uncertainty. This way, we can consider how big an impact the epistemic uncertainty has on the predictions, compared to the total uncertainty.

In addition to this, we want to experiment with the method mentioned in Chapter 15, where an additional value is introduced for each feature, which denotes whether a value for the feature is present or not. We expect that this will introduce the possibility of evaluating whether relevance scores vary because of missing features, or because of features having a value of 0. This allows for distinguishing between these two cases, when explaining how the features affect uncertainty in the prediction. The results from this can then be compared to the results found in Section 14.2.

Bibliography

[1] European Comission, "On artificial intelligence - a european approach to excellence and trust," 2020, [Accessed 11-05-2020]. [Online]. Available: https://ec.europa.eu/info/sites/default/files/commission-white-paper-artificial-intelligence-feb2020_en.pdf

[2] M. W. Dusenberry, D. Tran, E. Choi, J. Kemp, J. Nixon, G. Jerfel, K. A. Heller, and A. M. Dai, "Analyzing the role of model uncertainty for electronic health records," *CoRR*, vol. abs/1906.03842, 2019. [Online]. Available: http://arxiv.org/abs/1906.03842

[3] B. Kompa, J. Snoek, and A. L. Beam, "Second opinion needed: communicating uncertainty in medical machine learning," *NPJ digital medicine*, vol. 4, no. 1, p. 4, January 2021. [Online]. Available: https://europepmc.org/articles/PMC7785732

[4] B. Goodman and S. Flaxman, "European union regulations on algorithmic decision-making and a "right to explanation"," *AI Magazine*, vol. 38, no. 3, p. 50–57, Oct 2017. [Online]. Available: http://dx.doi.org/10.1609/aimag.v38i3.2741

[5] K. Bykov, M. M. Höhne, K. Müller, S. Nakajima, and M. Kloft, "How much can I trust you? - quantifying uncertainties in explaining neural networks," *CoRR*, vol. abs/2006.09000, 2020. [Online]. Available: https://arxiv.org/abs/2006.09000

[6] M. Reyna, C. Josef, R. Jeter, S. Shashikumar, B. Moody, M. B. Westover, A. Sharma, S. Nemati, and G. Clifford, "Early prediction of sepsis from clinical data – the physionet computing in cardiology challenge 2019," August 2019. [Online]. Available: https://doi.org/10.13026/v64v-d857

[7] Tværspor - et tværsektorielt projekt i patientens spor. (2020, September). [Online]. Available: https://www.tvaerspor.dk/

[8] United Nations, "UN sustainable development goals," 2020, [Accessed 24-09-2020]. [Online]. Available: https://www.un.org/sustainabledevelopment/health/

[9] E. LaRosa and D. Danks, "Impacts on trust of healthcare ai," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 210–215. [Online]. Available: https://doi.org/10.1145/3278721.3278771

[10] C. K. Frydkjær, L. Østergaard, M. N. Stenkær, M. Simonsen, M. H. Svendsen, and S. D. Nielsen, "Analyzing calibration of state of the art deep learning architectures for electronic health records," December 2020, pre-Master Thesis.

[11] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1321–1330.

[12] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: A tutorial introduction," *CoRR*, vol. abs/1910.09457, 2019. [Online]. Available: http://arxiv.org/abs/1910.09457

[13] R. Senge, S. Bösner, K. Dembczyński, J. Haasenritter, O. Hirsch, N. Donner-Banzhoff, and E. Hüllermeier, "Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty," *Information Sciences*, vol. 255, pp. 16–29, 2014.

[14] J. Antorán, U. Bhatt, T. Adel, A. Weller, and J. M. Hernández-Lobato, "Getting a clue: A method for explaining uncertainty estimates," 2021.

[15] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6405–6416.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[17] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, "Hands-on bayesian neural networks–a tutorial for deep learning users," *arXiv preprint arXiv:2007.06823*, 2020.

[18] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, p. 859–877, Apr 2017. [Online]. Available: http://dx.doi.org/10.1080/01621459.2017.1285773

[19] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt, "Advances in variational inference," *CoRR*, vol. abs/1711.05597, 2017. [Online]. Available: http://arxiv.org/abs/1711.05597

[20] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," *arXiv preprint arXiv:1912.02757*, 2019.

[21] P. R. Conrad, Y. M. Marzouk, N. S. Pillai, and A. Smith, "Accelerating asymptotically exact mcmc for computationally intensive models via local approximations," *Journal of the American Statistical Association*, vol. 111, no. 516, p. 1591–1607, Oct 2016. [Online]. Available: http://dx.doi.org/10.1080/01621459.2015.1096787

[22] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[23] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[24] S. Lauritsen, M. Kalør, E. Kongsgaard, K. Lauritsen, M. Jørgensen, J. Lange, and B. Thiesson, "Early detection of sepsis utilizing deep learning on electronic health record event sequences," *Artificial Intelligence in Medicine*, vol. 104, 2020.

[25] S. M. Lauritsen, M. Kristensen, M. V. Olsen, M. S. Larsen, K. M. Lauritsen, M. J. Jørgensen, J. Lange, and B. Thiesson, "Explainable artificial intelligence model to predict acute critical illness from electronic health records," *Nature communications*, vol. 11, no. 1, pp. 1–11, 2020.

[26] S. M. Lauritsen, B. Thiesson, M. J. Jørgensen, A. H. Riis, U. S. Espelund, J. B. Weile, and J. Lange, "The consequences of the framing of machine learning risk prediction models: Evaluation of sepsis in general wards," *arXiv preprint arXiv:2101.10790*, 2021.

[27] W. Samek and K. Müller, "Towards explainable artificial intelligence," *CoRR*, vol. abs/1909.12072, 2019. [Online]. Available: http://arxiv.org/abs/1909.12072

[28] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[29] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," *CoRR*, vol. abs/1602.04938, 2016. [Online]. Available: http://arxiv.org/abs/1602.04938

[30] W. Samek and K.-R. Müller, "Layer-wise relevance propagation: An overview," *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, vol. 11700, p. 193, 2019.

[31] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K. Müller, "Unmasking clever hans predictors and assessing what machines really learn," *CoRR*, vol. abs/1902.10178, 2019. [Online]. Available: http://arxiv.org/abs/1902.10178

[32] TensorFlow, "tfp.layers.densevariational," 05 2021. [Online]. Available: https://www.tensorflow.org/probability/api_docs/python/tfp/layers/DenseVariational

[33] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[34] TensorFlow Probability, "TFP Probabilistic Layers: Regression," 03 2019. [Online]. Available: https://blog.tensorflow.org/2019/03/regression-with-probabilistic-layers-in.html

[35] TensorFlow, "tf.keras.layers.dense," 04 2021. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

[36] ——, "tfp.layers.variablelayer," 04 2021. [Online]. Available: https://www.tensorflow.org/probability/api_docs/python/tfp/layers/VariableLayer

[37] S. Rossi, P. Michiardi, and M. Filippone, "Good initializations of variational bayes for deep models," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5487–5497.

[38] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[39] TensorFlow, "tfp.distributions.normal," 06 2021. [Online]. Available: https://www.tensorflow.org/probability/api_docs/python/tfp/distributions/Normal

[40] ——, "Introduction to gradients and automatic differentiation," 05 2021. [Online]. Available: https://www.tensorflow.org/guide/autodiff

[41] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[42] TensorFlow, "Hyperparameter tuning with the hparams dashboard," 04 2021. [Online]. Available: https://www.tensorflow.org/tensorboard/hyperparameter_tuning_with_hparams

[43] J. M. Hernandez-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1861–1869. [Online]. Available: http://proceedings.mlr.press/v37/hernandez-lobatoc15.html

[44] K. Holland. (2019) Is my blood oxygen level normal? [Accessed 07-06-2020]. [Online]. Available: https://www.healthline.com/health/normal-blood-oxygen-level

[45] G. D. Freitas, A. Gudur, M. Vela-Ortiz, J. Jodelka, D. Livert, and M. Krishnamurthy, "Where there is sodium there may be sepsis," *Journal of Community Hospital Internal Medicine Perspectives*, vol. 9, no. 4, pp. 296–299, 2019, pMID: 31528275. [Online]. Available: https://doi.org/10.1080/20009666.2019.1634407

[46] B. Suetrong, C. Pisitsak, J. H. Boyd, J. A. Russell, and K. R. Walley, "Hyperchloremia and moderate increase in serum chloride are associated with acute kidney injury in severe sepsis and septic shock patients," Oct 2016. [Online]. Available: https://open.library.ubc.ca/collections/facultyresearchandpublications/52383/items/1.0362230

[47] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[48] M. J. Hall, S. N. Williams, C. J. DeFrances, and A. Golosinskiy, "Inpatient care for septicemia or sepsis: a challenge for patients and hospitals," 2011.

[49] C. Subbe, M. Kruger, P. Rutherford, and L. Gemmel, "Validation of a modified Early Warning Score in medical admissions," *QJM: An International Journal of Medicine*, vol. 94, no. 10, pp. 521–526, 10 2001. [Online]. Available: https://doi.org/10.1093/qjmed/94.10.521

[50] Fethi Gül, Mustafa Kemal Arslantaş, İsmail Cinel, and Anand Kumar, "Changing definitions of sepsis," June 2017, [Accessed 20-10-2020]. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5512390/

[51] C. Nickson, "Pao2/fio2 ratio (p/f ratio)," April 2020, [Accessed 20-10-2020]. [Online]. Available: https://litfl.com/pao2-fio2-ratio/

[52] U. I. Services, "Basic laboratory tests complete blood counts (cbc)," 2015, [Accessed 20-10-2020]. [Online]. Available: https://www.uiservices.com/wp-content/uploads/2015/01/CompleteBloodCounts.pdf

[53] WebMD, "Understanding low blood pressure – the basics," 2019, [Accessed 20-10-2020]. [Online]. Available: https://www.webmd.com/heart/understanding-low-blood-pressure-basics#1

[54] W. M. Reference, "What is a bilirubin test?" 2019, [Accessed 20-10-2020]. [Online]. Available: https://www.webmd.com/a-to-z-guides/bilirubin-test#1

[55] N. K. Foundation, "Creatinine: What is it?" 2017, [Accessed 20-10-2020]. [Online]. Available: https://www.kidney.org/atoz/content/what-creatinine

[56] BrainLine, "What is the glasgow coma scale?" 2018, [Accessed 20-10-2020]. [Online]. Available: https://www.brainline.org/article/what-glasgow-coma-scale

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

[58] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016. [Online]. Available: http://arxiv.org/abs/1609.04747

[59] V. Kuleshov and S. Ermon, "Reliable confidence estimation via online learning," *CoRR*, vol. abs/1607.03594, 2016. [Online]. Available: http://arxiv.org/abs/1607.03594

[60] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[61] J. Nixon, M. Dusenberry, L. Zhang, G. Jerfel, and D. Tran, "Measuring calibration in deep learning," *CoRR*, vol. abs/1904.01685, 2019. [Online]. Available: http://arxiv.org/abs/1904.01685

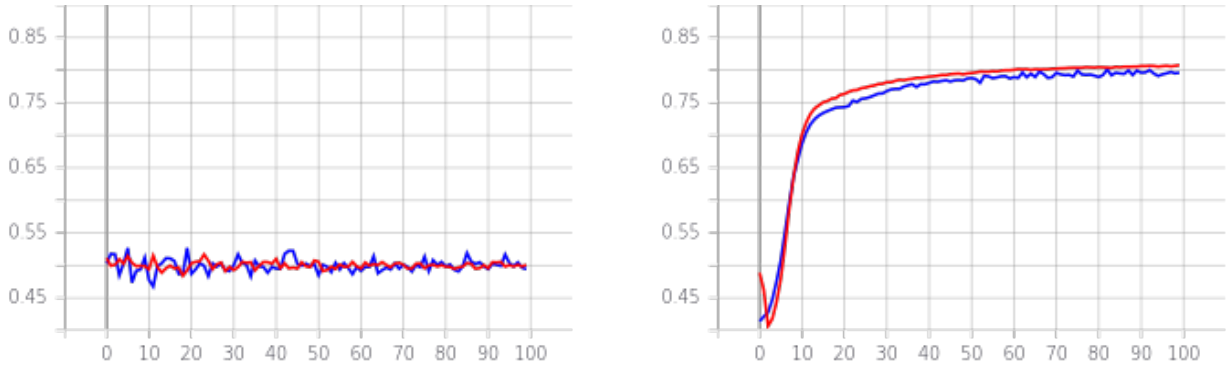
Part V

Appendix

Appendix A

PhysioNet Softplus Scaling

A.1 During Training



(a) `posterior_softplus_weight` $= 1$.

(b) `posterior_softplus_weight` $= 0.0001$.

Figure A.1: AUROC plotted for each epoch during training with different `posterior_softplus_weight`. Red plot is for training set and blue is for validation set. The y-axis shows the AUROC and the x-axis shows the epoch number.

A.2 Performance

posterior_softplus_weight	AUC_train	AUC_val	auprc_train	auprc_val
1	0.5	0.5	0.0155	0.0155
0.1	0.48	0.475	0.0146	0.0144
0.01	0.767	0.771	0.0665	0.0669
0.001	0.809	0.797	0.0862	0.0799
0.0001	0.808	0.796	0.0868	0.0786

Table A.1: Performance of the model in relation to different `posterior_softplus_weight` values, on the ΔDP-B dataset.

Appendix B

Hyperparameter Tuning

B.1 PhysioNet Config & Hyperparameters 1

```
"experiments": [
    {
        "model": "bayesian_long_model",
        "plot": false,
        "hyperparameters": [
            {
                "units_layer1": [100],
                "units_layer2": [150],
                "units_layer3": [200],
                "units_layer4": [100],
                "units_layer5": [50],
                "batch_size": [10000],
                "kl_weight_scale": [0.5, 1, 1.5],
                "posterior_softplus_weight":[0.0002, 0.0005, 0.001],
                "prior_scale":[5, 10],
                "learning_rate": [0.0001, 0.00005, 0.00001, 0.000001],
                "epochs": [500],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["auprc"],
                "metric2": ["AUC"]
            }
        ]
    },
    {
        "model": "bayesian_long_model",
        "plot": false,
        "hyperparameters": [
            {
                "units_layer1": [200],
                "units_layer2": [300],
                "units_layer3": [400],
                "units_layer4": [200],
                "units_layer5": [100],
                "batch_size": [10000],
                "kl_weight_scale": [0.5, 1, 1.5],
                "posterior_softplus_weight":[0.0002, 0.0005, 0.001],
```

```
                "prior_scale":[5, 10],
                "learning_rate": [0.0001, 0.00005, 0.00001, 0.000001],
                "epochs": [500],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["auprc"],
                "metric2": ["AUC"]
            }
        ]
    },
    {
        "model": "bayesian_linear_model",
        "plot": false,
        "hyperparameters": [
            {
                "point_estimate": [false],
                "units_layer1": [100],
                "units_layer2": [50],
                "units_layer3": [20],
                "dropout": [0],
                "batch_size": [10000],
                "kl_weight_scale": [0.5, 1, 1.5],
                "posterior_softplus_weight":[0.0002, 0.0005, 0.001],
                "prior_scale":[5, 10],
                "learning_rate": [0.0001, 0.00005, 0.00001, 0.000001],
                "epochs": [200],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["auprc"],
                "metric2": ["AUC"]
            }
        ]
    },
    {
        "model": "bayesian_linear_model",
        "plot": false,
        "hyperparameters": [
            {
                "point_estimate": [false],
                "units_layer1": [200],
                "units_layer2": [100],
                "units_layer3": [40],
                "dropout": [0],
                "batch_size": [10000],
                "kl_weight_scale": [0.5, 1, 1.5],
                "posterior_softplus_weight":[0.0002, 0.0005, 0.001],
                "prior_scale":[5, 10],
```

```
                "learning_rate": [0.0001, 0.00005, 0.00001, 0.000001],
                "epochs": [200],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["auprc"],
                "metric2": ["AUC"]
            }
        ]
    }
]
```

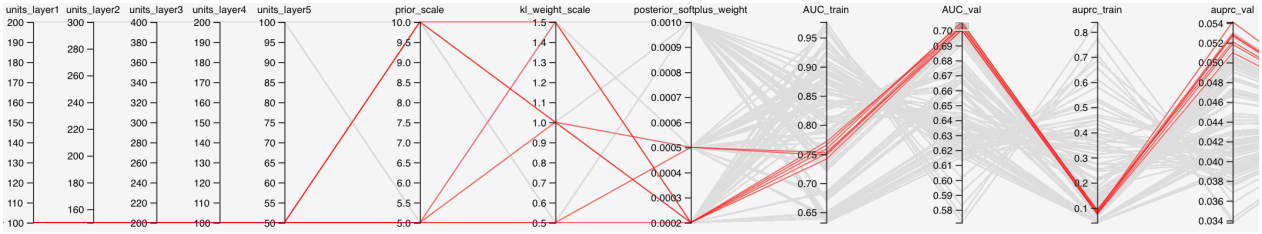


Figure B.1: bayesian_long_model on ΔDP-B.

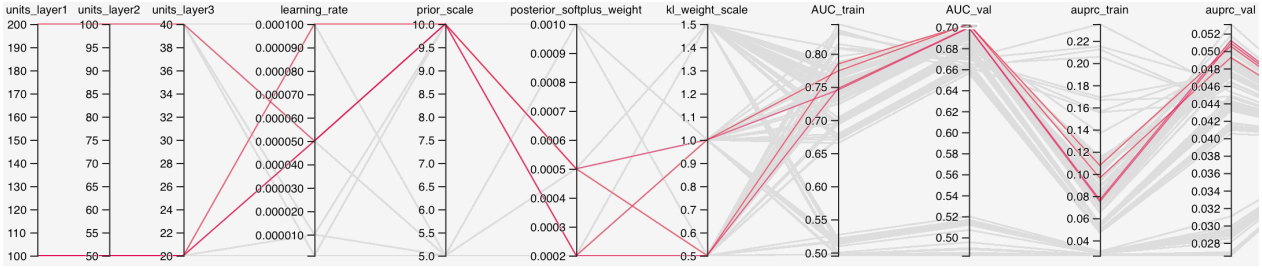


Figure B.2: bayesian_linear_model on ΔDP-B.

B.2 PhysioNet Config & Hyperparameters 2

```

"experiments": [
    {
        "model": "bayesian_manual_posterior_model",
        "plot": false,
        "calculate_uncertainty_metric": false,
        "load_model_from": [],
        "ece_error_bars" : true,
        "hyperparameters": [
            {
                "point_estimate": [false],
                "units_layer1": [60],
                "units_layer2": [40],
                "units_layer3": [20],
                "batch_size": [10000],
```

```
                "posterior_init_loc":[0],
                "posterior_init_scale":[-6],
                "learning_rate": [0.0001],
                "pos_weight": [1],
                "epochs": [300],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["auprc"],
                "metric2": ["AUC"]
            }
        ]
    }
]
```

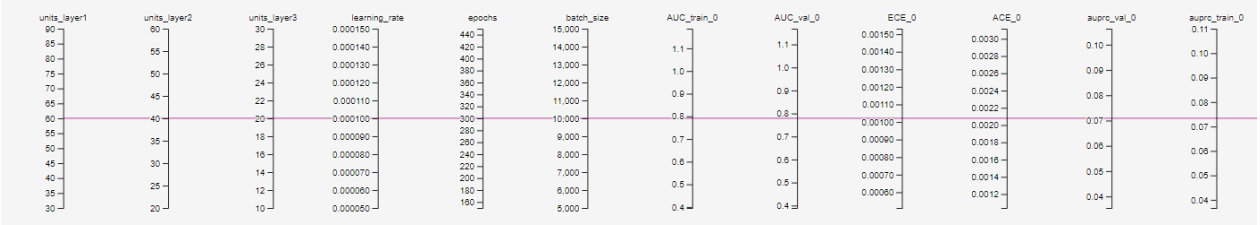


Figure B.3: bayesian_manual_posterior on ΔDP-B, with −6 scale initialization value.

B.3 PhysioNet Config & Hyperparameters 3 - Fixed Time To Onset

```

"experiments": [
    {
        "model": "bayesian_manual_posterior_model",
        "plot": false,
        "calculate_uncertainty_metric": false,
        "load_model_from": [],
        "ece_error_bars" : true,
        "hyperparameters": [
            {
                "point_estimate": [false, true],
                "units_layer1": [60],
                "units_layer2": [40],
                "units_layer3": [20],
                "batch_size": [5000, 10000],
                "posterior_init_loc":[0, 0.5],
                "posterior_init_scale":[-4, -5, -6],
                "learning_rate": [0.0001, 0.001],
                "pos_weight": [1],
                "epochs": [1500, 3000],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
```

```
                "metric1": ["auprc"],
                "metric2": ["AUC"]
            }
        ]
    }
]
```

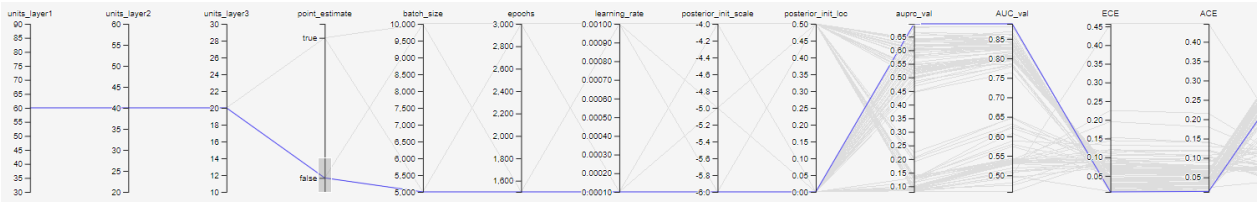


Figure B.4: Expanded grid on ΔDP-A, values based on earlier best performing grids.

B.4 PhysioNet Config & Hyperparameters 4 - Fixed Time To Onset, Best Model

```

"experiments": [
    {
        "model": "bayesian_manual_posterior_model",
        "plot": false,
        "calculate_uncertainty_metric": false,
        "load_model_from": [],
        "ece_error_bars" : true,
        "hyperparameters": [
            {
                "point_estimate": [false],
                "units_layer1": [60],
                "units_layer2": [40],
                "units_layer3": [20],
                "batch_size": [5000, 10000],
                "posterior_init_loc":[0, 0.5],
                "posterior_init_scale":[-4, -5, -6],
                "learning_rate": [0.001, 0.0001],
                "pos_weight": [1],
                "epochs": [1500, 3000],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["auprc"],
                "metric2": ["AUC"]
            }
        ]
    }
]
```

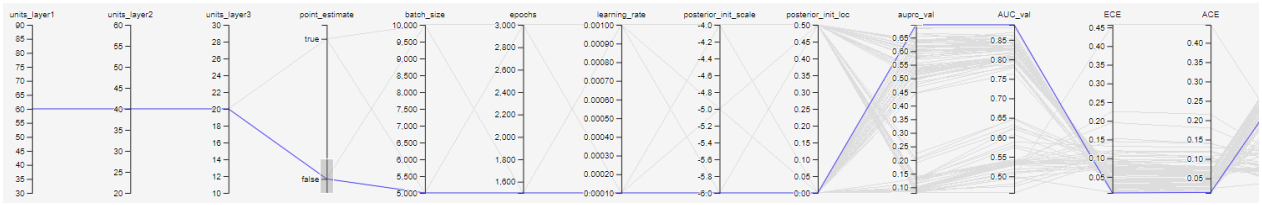


Figure B.5: Expanded grid on ΔDP-A, values based on earlier best performing grids.

The best performing parameters:

- 5,000 Batch Size.
- 0.0001 Learning Rate.
- 1,500 Epochs.
- 0 Posterior Init Loc
- −6 Posterior Init Scale

B.5 Tværspor Config & Hyperparameters 1

```
"experiments": [
    {
        "model": "bayesian_linear_model",
        "plot": false,
        "hyperparameters": [
            {
                "point_estimate": [false],
                "units_layer1": [50],
                "units_layer2": [30],
                "units_layer3": [10],
                "dropout": [0.02],
                "batch_size": [10000],
                "kl_weight_scale": [0.5, 1],
                "posterior_softplus_weight":[0.0001, 0.00001, 0.000001],
                "prior_scale":[1, 1.5, 2],
                "learning_rate": [0.001, 0.0001],
                "epochs": [1000],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["auprc"],
                "metric2": ["AUC"]
            }
        ]
    },
    {
        "model": "bayesian_linear_model",
        "plot": false,
        "hyperparameters": [
            {
```

```
                "point_estimate": [false],
                "units_layer1": [60],
                "units_layer2": [40],
                "units_layer3": [20],
                "dropout": [0.02],
                "batch_size": [10000],
                "kl_weight_scale": [0.5, 1],
                "posterior_softplus_weight":[0.0001, 0.00001, 0.000001],
                "prior_scale":[1, 1.5, 2],
                "learning_rate": [0.001, 0.0001],
                "epochs": [1000],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["auprc"],
                "metric2": ["AUC"]
            }
        ]
    }
]
```

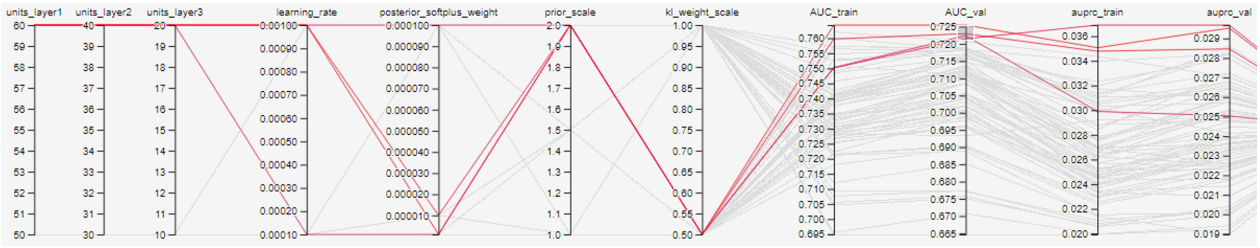


Figure B.6: bayesian_linear_model.

B.6 Tværspor Config & Hyperparameters 2

```
"experiments": [
    {
        "model": "bayesian_manual_posterior_model",
        "plot": false,
        "calculate_uncertainty_metric": true,
        "hyperparameters": [
            {
                "point_estimate": [false],
                "units_layer1": [60],
                "units_layer2": [40],
                "units_layer3": [20],
                "dropout": [0],
                "batch_size": [5000, 10000],
                "kl_weight_scale": [1],
                "posterior_init_loc":[0],
                "posterior_init_scale":[-4, -5, -6],
```

```
                "prior_scale":[1],
                "learning_rate": [0.0001, 0.001],
                "epochs": [1500, 3000],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["m_auprc"],
                "metric2": ["m_auroc"]
            }
        ]
    }
]
```

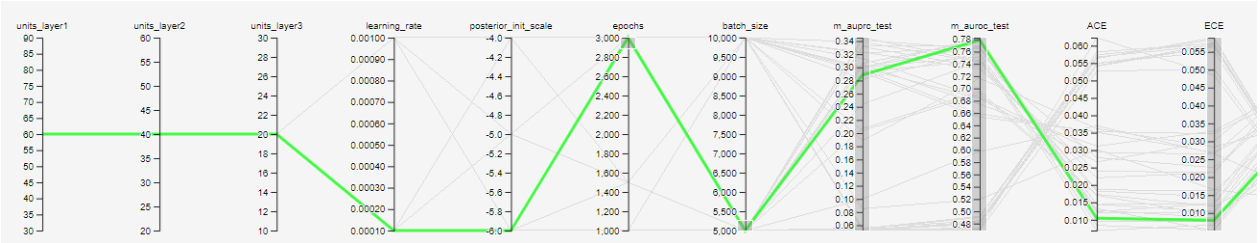


Figure B.7: Grid search on ΔDCT.

Appendix C

This appendix is a copy of Chapter 2 "Description of Datasets" from our pre-master thesis [10].

Description of Datasets

Enversion has earlier worked with two projects about early detection of sepsis, which both work on EHR data [24, 25]. We therefore choose to narrow our scope of EHR data to be within the area of sepsis detection, with the goal of reasoning about the uncertainty of models that are designed for early detection of sepsis. A description of sepsis can be found in Appendix A.

A dataset preprossed by Enversion, which is based on the CROSS-TRACKS cohort, has been made accessible to us through our employment and tasks at Enversion. As part of our terms of employment we have signed a non-disclosure agreement as well as received the appropriate data protection training. All data presented in the report is anonymous.

Since our supervisor has not signed a non-disclosure agreement, he cannot view the dataset made accessible to us by Enversion. Therefore, we choose to find another dataset with no accessibility restrictions, which also resembles this dataset. For this we find a dataset from PhysioNet [6], which also consists of EHR data, which can be used to predict sepsis. In this chapter, we describe the two datasets and a comparison between them.

C.1 Dataset Introduction

In this section, we give a general description of the datasets and what we investigate about them in the following sections. The first dataset is prepared by Enversion using data from the research project Cross-Track [7] and is referred to as DCT. The second dataset is from PhysioNet and is referred to as DP. Both datasets consist of different measurements taken from multiple patients admitted to hospitals, and can be divided into two kinds of values:

- Vital Signs - Essential values from measurements for monitoring the health of patients, such as heart rate, temperature, and blood pressure.
- Laboratory Values - Values from laboratory experiments, such as measuring the content of calcium or magnesium in the patient's blood.

For both datasets, the vital signs and laboratory values are aggregated over an hour and stored in a matrix, where each row corresponds to a time span of one hour, which we call timesteps, and each column consists of measurement values. We refer to what the columns describe as features. Since some measurements are not conducted for every timestep, some cells do not contain any value and is therefore assigned NaN. These are referred to as missing values.

To better understand the two datasets, we investigate the following for DCT and DP:

- How much of the data is missing values?
  - Does this change over time?
  - Which features contain missing values more frequently than others?

  - How often are measurements made?
- What is the distribution of sepsis-positive and -negative patients?
- Is there a correlation between the admission time and whether the patients are sepsis-positive or -negative?

The following two sections answer these questions for DP and DCT respectively, followed by a comparison of the two, to evaluate whether DP is similar enough to DCT.

C.2 PhysioNet

PhysioNet has a publicly available dataset with medical records of more than $40,000$ patients distributed over two hospitals, referred to as dataset A and B [6]. The framing used in DP is sequential data with a fixed time to sepsis onset. The data from the two hospitals can be downloaded as separate ZIP files through PhysioNet's archive and contains EHRs for patients, stored as Pipe Separated Values (PSV) files. As described in Section C.1, each row in the files represents a timestep and each column a feature. The columns describe a total of 40 features and a sepsis label, and are sorted according to the following categories:

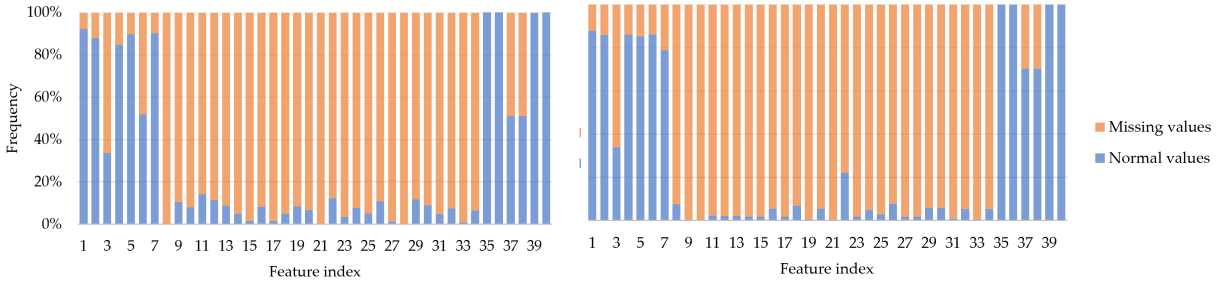
- $1 - 8$: Vital Signs
- $9 - 34$: Laboratory Values
- $35 - 40$: Demographics - Information about the patient, such as age and gender, which are constant for all timesteps.
- 41: Sepsis Label - A boolean label indicating whether the patient has been diagnosed with sepsis. The label is offset six hours before sepsis onset, which means that the label changes from 0 to 1, six hours before the patient is diagnosed with sepsis.

C.2.1 Data Analysis

In this section, we analyze DP to answer the questions from Section C.1.

Data Sparsity

We count the number of missing values and actual values for each feature in the timesteps, for each dataset, which is displayed in Figure C.1. Both datasets in Figure C.1 show that some features include missing values more frequently than others. The frequency is often related to its category, described in Section C.2.



(a) Missing values in dataset A.

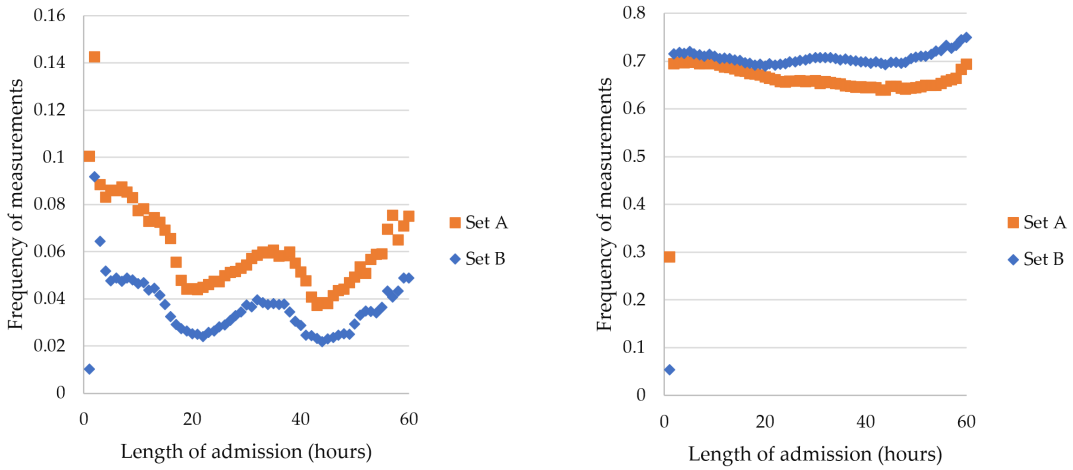
(b) Missing values in dataset B.

Figure C.1: Illustration of frequency of missing values in DP.

6 out of 8 features in the Vital Signs category include measurements for over 80% of the data points. The third feature, which is temperature, includes over 60% missing values in both dataset A and B, despite being a vital sign. The eighth feature, which is end-tidal carbon dioxide, only has missing values in dataset A and around 95% missing values in dataset B. Features within the laboratory values category are very sparse, and close to all features include 90% missing values. Missing values in the demographic features are only present in the features indicating which ICU the patient is located at. The only demographic feature that changes value over time is length of admission feature, while age, gender, ICU, and admission time remain the same.

To find out how the frequency of measurements changes over time, we find the percentage of patients with measurements for each lab value at every timestep. To simplify the graph, shown in Figure C.2a, we average the percentage over all the lab values to get one point for each timestep. Figure C.2a shows that DP has a few more measurements in the beginning, but is almost the same over time.

Figure C.2b shows the frequency of vital sign measurements. The frequency is high and stays consistent across the admission, since the data is from ICUs, where vitals are checked very often.



(a) Lab values.

(b) Vital signs.

Figure C.2: Frequency of measurements over time in DP.

Time Between Measurements

Concerning the sparse data, we examine how frequent each of the measurements occur by taking the average time between the measurements of each patient's features, and plot the median value for each feature, as seen in Figure C.3.

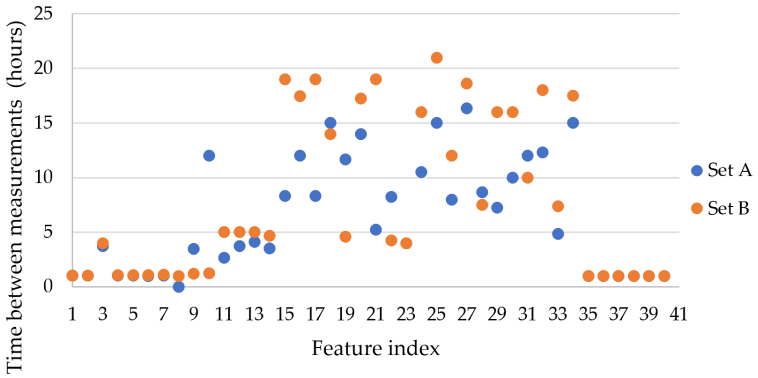


Figure C.3: Average time between measurements for each feature.

Patient Count

To get an overview of the number of patients with sepsis, we count the number of sepsis-negative and sepsis-positive patients, which are shown in Table C.1.

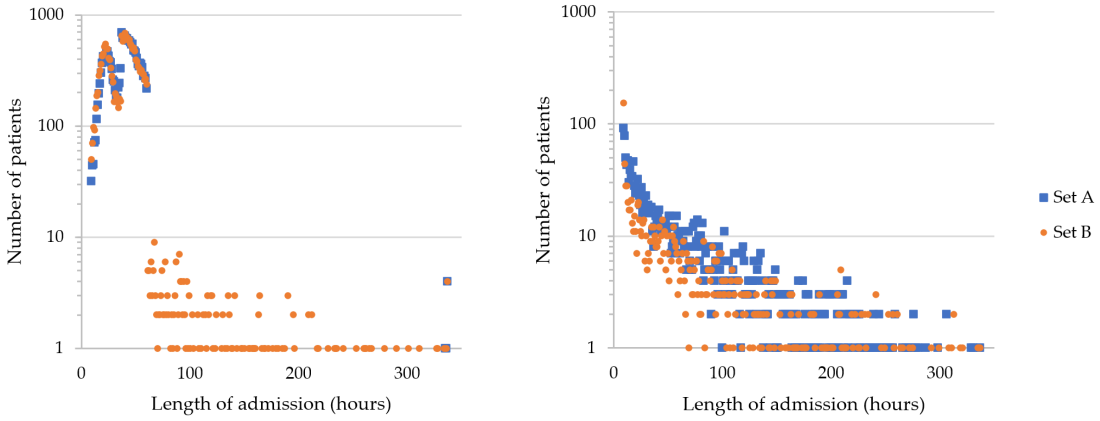
	Set A	Set B	Combination
Total	20,336	20,000	40,336
Negative	18,546	18,858	37,404
Positive	1,790	1,142	2,932
Negative percentage	91.1%	94.3%	92.7%
Positive percentage	8.8%	5.8%	7.3%

Table C.1: Count of patients for the two datasets, A and B, and the combination of the two.

We observe that the datasets include more sepsis-negative patients, with a percentage of over 90%. This shows a general imbalance in the distribution of postitive and negative samples in the data.

Length of Admission

We observe that in several cases, patients diagnosed with sepsis have a prolonged admission to the ICU, compared to sepsis-negative patients. Figure C.4 shows the admission length for both sepsis-positive and sepsis-negative patients in dataset A and B. Sepsis-positive patients have a prolonged admission in dataset A, with 6 out of 18,546 sepsis-negative patients and 651 out of 1,799 sepsis-positive patients being admitted for more than 60 hours. However, this is not the case for dataset B, where 223 out of 18,858 sepsis-negative patients and 393 out of 1,142 sepsis-positive patients in dataset B are admitted to the ICU for more than 60 hours.



(a) Length of admission for sepsis-negative.

(b) Length of admission for sepsis-positive.

Figure C.4: Length of admission.

Note that the scale of the y-axis for the two graphs in Figure C.4 is logarithmic to make it easier to distinguish the values below 10. The curve of sepsis-positive patients in Figure C.4b shows that a large part of the patients are diagnosed with sepsis within their first days of admission in the ICU.

C.3 Cross-Tracks

As DCT has been prepared by Enversion using data from the research project Cross-Tracks [7], it is already structured as a NumPy array (`numpy.org`). The NumPy array has been serialized and stored to disk using pickle (`docs.python.org/3/library/pickle.html`), so it can easily be loaded into memory. DCT has 25 features, where the first six are vital signs and the rest are laboratory values.

These are stored in a three-dimensional array with the following indexes: [anonymized patient ID][timestep][feature]. The data is prepared using the on-clinical demand framing, which means that the end time of sequences is the time where clinical staff performs an assesment of early warning score (EWS) [49]. The start time of the sequences is 24 hours before that. These assesments are conducted multiple times during an admission, with no indication in the data of when in the admission they are performed. Due to this, we cannot say anything about the patients' length of admission for DCT. The sepsis label indicates if a patient gets sepsis within 12 hours after a test is requested. These sepsis labels are stored in a separate array with one boolean entry per patient.

C.3.1 Data Analysis

In this section, we analyze DCT to answer the questions from Section C.1. The graphs and table in this section are made in the same way as the graphs in Section C.2.1.

Data Sparsity

Figure C.5 shows that DCT only has values for around 10% of the timesteps for the vital signs, and less than 3% for laboratory values.

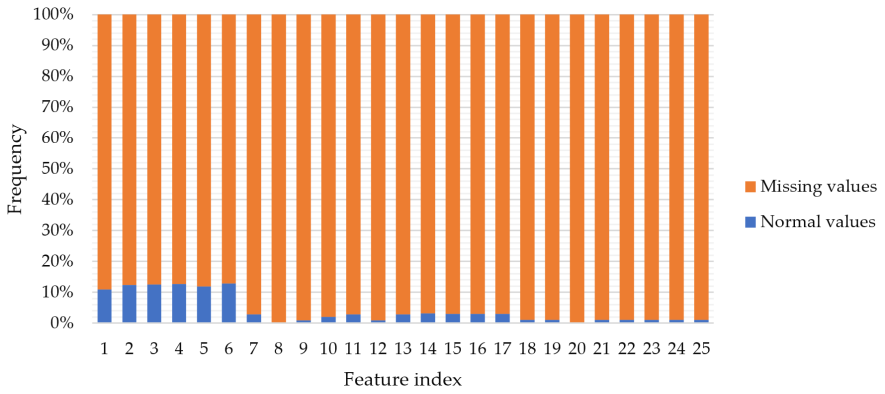
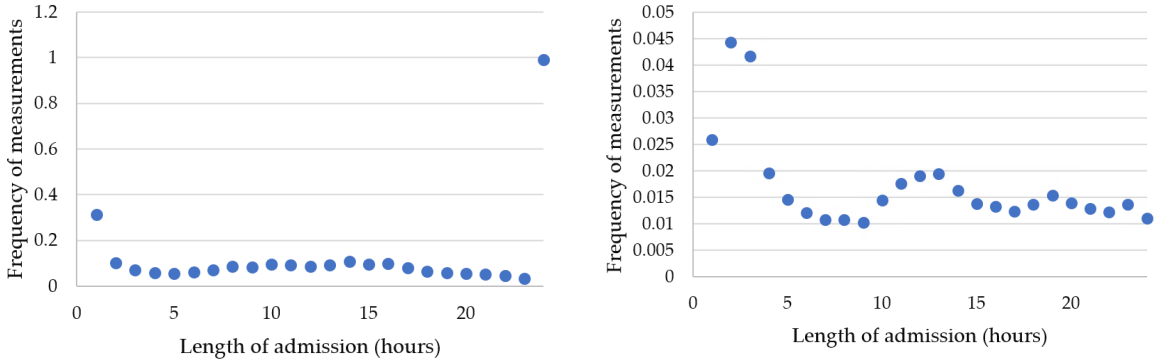


Figure C.5: Missing values in DCT.

If we look at how the frequency of measurements change over time, as shown in Figure C.6a and Figure C.6b, we see that there are more measurements in the beginning, and for vital signs there is a spike in the last timestep. We think the reason there are more measurements in the last hour, is because it is the timestep where the clinical staff performs the EWS assessment. We can see that the frequency of measurements are much lower than for DP, since the data from DCT is not from ICUs.



(a) Vital signs.

(b) Lab values.

Figure C.6: Frequency of measurements over time in DCT.

Time Between Measurements

Figure C.7 shows that most measurements are taken once every 11 hours and the rest of the measurements are taken around every four hours.

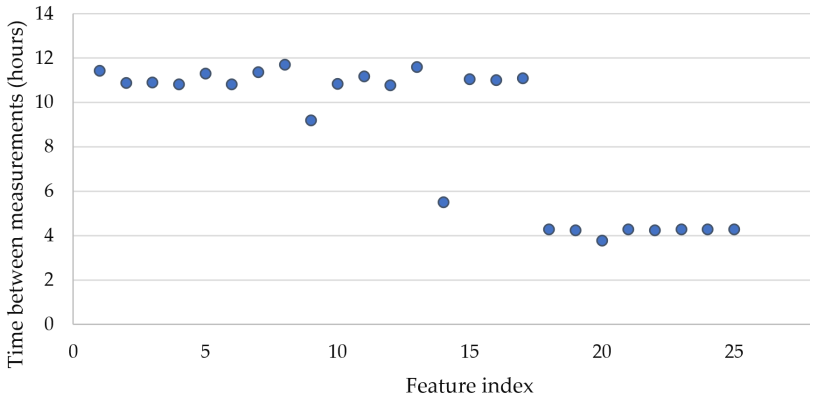


Figure C.7: Median time between measurements for each feature for DCT.

Sample Count

DCT has a lot of samples, but most of them are sepsis-negative as shown in Table C.2. Overall, DCT has more sepsis-negative samples compared to sepsis-positive than DP.

	DCT
Total	340,450
Negative	337,814
Positive	2,636
Negative percentage	99.2%
Positive percentage	0.8%

Table C.2: Count of samples for DCT.

Length of Admission

Because DCT only has data for the last 24 hours, we cannot say anything about the length of admission.

C.4 Dataset Comparison

After having analyzed and described DP and DCT, we show a comparison between the two datasets and discuss their similarities and differences, to evaluate whether DP is sufficiently similar to DCT.

Table C.3 shows the similarities and differences between the datasets. In general, the datasets have a lot of similarities. The raw data format of the datasets is different, but the content of the sepsis admissions contains the same kind of data. One of the main differences between them is that DP has demographic data, while DCT does not, and that the framing is different. Another difference is that DP has more features than DCT, with 40 and 25 features respectively. As the data from DP is taken from an ICU, the measurements are taken more frequently than other departments at the hospital. As the number of timesteps, i.e. the length of admission, in DCT is limited to 24, it differs from DP, which contains timesteps for all hours of admission. Therefore,

we need to consider the size of the observation window of timesteps when doing experiments with the data. Additionally, DCT has a higher frequency of missing values than DP.

We deem DP and DCT to be sufficiently similar to be used in the same models, as the differences mostly are that DP has more data about each patient and take measurements more often. However, because the framing is different we cannot directly compare performance metrics between the two datasets.

Attribute	PhysioNet	Enversion
Raw data format	Pipe-separated values	Python NumPy array
Framing	Fixed time to onset	On-clinical-demand
Vital signs	8	6
Laboratory-categories (e.g. Blood tests)	25	19
Average time between vital signs (hours)	1.5	11
Average time between Laboratory measurements (hours)	12	8
Age	Yes	No
Gender	Yes	No
Admission time	Yes	No
Aggregation time	One hour	One hour
Positive patient count	2,932	2,636
Country	USA	Denmark
Sepsis definition	Sepsis-3 definition	Sepsis-3 definition

Table C.3: Comparison between the datasets from PhysioNet and Enversion.

Appendix D

This appendix is a copy of Appendix C "Feature Description of Datasets" from our pre-master thesis [10].

Feature Description of Datasets

Feature number	Measurement
1	Temperature
2	SpO2
3	Heart rate
4	Diastolic BP
5	Respiratory Frequency
6	Systolic BP
7	B-Leukocytes
8	B-Neutrophils
9	B-Platelets
10	eGFR
11	P-Albumin
12	P-Bilirubine
13	P-C-reactive protein
14	P-Glucose
15	P-Potassium
16	P-Creatinine
17	P-Sodium
18	P(aB)-Hydrogen carbonate
19	P(aB)-Potassium
20	P(aB)-Chloride
21	P(aB)-Lactate
22	P(aB)-Sodium
23	P(aB)-pCO2
24	P(aB)-pH
25	P(aB)-pO2

Table D.1: Features of dataset prepared by Enversion from Cross-Track's dataset [7].

Feature number	Measurement	Description
Vital signs (Feature 1-8)		
1	HR	Heart rate (beats per minute)
2	O2Sat	Pulse oximetry (%)
3	Temp	Temperature (Deg C)

4	SBP	Systolic BP (mm Hg)
5	MAP	Mean arterial pressure (mm Hg)
6	DBP	Diastolic BP (mm Hg)
7	Resp	Respiration rate (breaths per minute)
8	EtCO2	End-tidal carbon dioxide (mm Hg)
Laboratory values (Feature 9-34)		
9	BaseExcess	Measure of excess bicarbonate (mmol/L)
10	HCO3	Bicarbonate (mmol/L)
11	FiO2	Fraction of inspired oxygen (%)
12	pH	N/A
13	PaCO2	Partial pressure of carbon dioxide from arterial blood (mm Hg)
14	SaO2	Oxygen saturation from arterial blood (%)
15	AST	Aspartate transaminase (IU/L)
16	BUN	Blood urea nitrogen (mg/dL)
17	Alkalinephos	Alkaline phosphatase (IU/L)
18	Calcium	(mg/dL)
19	Chloride	(mmol/L)
20	Creatinine	(mg/dL)
21	Bilirubin_direct	Bilirubin direct (mg/dL)
22	Glucose	Serum glucose (mg/dL)
23	Lactate	Lactic acid (mg/dL)
24	Magnesium	(mmol/dL)
25	Phosphate	(mg/dL)
26	Potassium	(mmol/L)
27	Bilirubin_total	Total bilirubin (mg/dL)
28	TroponinI	Troponin I (ng/mL)
29	Hct	Hematocrit (%)
30	Hgb	Hemoglobin (g/dL)
31	PTT	partial thromboplastin time (seconds)
32	WBC	Leukocyte count ($count \cdot 10^3/\mu L$)
33	Fibrinogen	(mg/dL)
34	Platelets	($count \cdot 10^3/\mu L$)
Demographics (Feature 35-40)		
35	Age	Years (100 for patients 90 or above)
36	Gender	Female (0) or Male (1)
37	Unit1	Administrative identifier for ICU unit (MICU)
38	Unit2	Administrative identifier for ICU unit (SICU)
39	HospAdmTime	Hours between hospital admit and ICU admit
40	ICULOS	ICU length-of-admission (hours since ICU admit)

Table D.2: Features of dataset from PhysioNet [6].

Appendix E

This appendix is a copy of Appendix A "Sepsis" from our pre-master thesis [10].

Sepsis

In this section, we describe the disease sepsis, and how it is diagnosed, based on the article [50]. Sepsis is a severe medical condition, where damage to tissue and organs is caused by the immune system's response to an infection. If bacteria from an infection spreads to the bloodstreams, it might spread to other organs, which can be fatal. The immune system reacts to the infection by lowering the blood pressure and thereby slowing down the blood flow, making it harder for the bacteria to reach the organs. However, this reduces the oxygen flow to the organs, which can cause them to fail, resulting in organ and tissue damage.

Sepsis is described as a life-threatening disease, as the mortality rate ranges between $\sim$ 20% to $\sim$ 40%, and is one of the most common causes of death in intensive care units (ICUs). The number of sepsis cases has doubled over the last 10 years, however, the overall mortality rate has decreased due to advancements within health care. Early detection of sepsis can be difficult, as the signs of the disease can be divergent.

As of the time of writing, there have been a total of three sepsis definitions. With the sepsis-3 definition, the old criteria were replaced with a new system, Sequential Organ Failure Assessment (SOFA). Now, sepsis is diagnosed with the SOFA scoring system, which evaluates the condition of six organ systems. This score can be used to determine whether any of the organ systems have reduced functionality, which may indicate that the patient has sepsis. The six organ systems that the SOFA scoring system evaluate are:

- **Respiratory System**: By measuring the partial pressure of oxygen in the arterial blood [51].
- **Coagulation**: By measuring the number of platelets in the blood. A higher number of platelets results in a higher chance of blood clotting [52].
- **Cardiovascular**: By measuring hypotension, which is a low systolic blood pressure [53].
- **Liver**: By measuring the amount of bilirubin present in the liver. A high amount of bilirubin is a sign of diseases [54].
- **Renal**: By measuring the amount of creatinine in the renal or urine output of the patient. Creatinine is a waste product in the blood, and is the result of muscle attrition [55].
- **Central Nervous System**: By measuring disruptions in brain function with the Glasgow Coma Scale (GCS). GCS analyses the mental status of the patient through a set of criteria and assign points to the patient according to their brain functions. A lower GCS score signifies less consciousness in the patient [56].

A score of 0 to 4 can be assigned to each of the six evaluated organ systems, meaning that the total score can be between 0 and 24. If a patient experiences symptoms that result in a SOFA score with an increase of two, the patient is diagnosed with sepsis. The number of points can reflect the mortality of the patient, as more points can indicate a more severe case of sepsis. For

example, with a score ranging between 0 and 6, the mortality of the patient can be expected to be less than 10%, and with a score above 15, the expectation is 90%.

Appendix F

This appendix is a copy of Chapter 4 "Theory", Section 4.1 "Artificial Neural Networks" from our pre-master thesis [10].

Artifical Neural Networks Theory

F.1 Artificial Neural Networks

In this section, we give an introduction to artificial neural networks, based on information from [16] and [23]. Artificial neural networks are a group of algorithms inspired by the biological neural network, like the human brain. Throughout this report, we refer to artificial neural networks as neural networks.

A neural network consists of an input layer, a number of hidden layers, and an output layer, each containing a number of neurons. Between two consecutive layers, a number of weights connect the neurons in layer $l-1$ to neurons in layer l. The weights are denoted w^l_{jk}, where j is the index of the neuron in layer l and k is the index of the neuron in layer $l-1$. Each neuron contains an internal value, that we denote z^l_j, and an output, denoted a^l_j. a^l_j is the output of an activation function, σ, with input z^l_j, as shown in Equation F.1b. z^l_j is calculated as the sum of products of all weights connected to the neuron and the output of the activation function for the corresponding neuron from the previous layer, a^{l-1}_k, shown in Equation F.1a. An activation function is used to make a neural network approximate nonlinear functions. Without an activation function, a neural network is only able to approximate linear functions, which is undesirable in many cases. An example of a nonlinear activation function is sigmoid $\left(\frac{1}{1+exp(-x)}\right)$, which produces outputs between 0 and 1. Additionally, neural networks use biases, which are used to shift the function of the neural network.

$$z^l_j = \sum_k w^l_{jk} \cdot a^{l-1}_k \tag{F.1a}$$

$$a^l_j = \sigma(z^l_j) \tag{F.1b}$$

In the following description, the bias is augmented as an extra activation from the previous layer, always having a value of 1, which is analogous to having a bias neuron with an output of 1, connected with weights to all neurons in a layer. Additionally, we will only be using σ to symbolize an activation function, although activation functions can be different between layers.

Instead of representing the activations from a layer as individual scalars, the activations can also be represented in the form of a vector, $\boldsymbol{a}^l$, where the elements are each activation from the layer. The weights for a layer l can be represented as a matrix $\boldsymbol{W}^l$, where index $\boldsymbol{W}^l[2,3]$ refers to w^l_{23}. With this representation, we can calculate the activations of a layer $\boldsymbol{a}^l$ using matrix multiplication as shown in Equation F.2:

$$\boldsymbol{z}^l = \boldsymbol{W}^l \cdot \boldsymbol{a}^{l-1} \tag{F.2a}$$

$$\boldsymbol{a}^l = \sigma(\boldsymbol{z}^l) \tag{F.2b}$$

These equations encapsulate the forward propagation of the neural network, from input to output.

F.1.1 Training a Neural Network

In this section, we explain how a neural network is trained. The basic principle of training a neural network is to update the parameters of the network based on the gradient of a cost function, C, given a set of input-output pairs. C is a function that measures the correctness of the prediction in relation to the ground truth labels. This measure is also known as the cost or loss. One example of a cost function is cross-entropy, which can be used when doing classification. In our case, we do binary classification when classifying whether a person gets sepsis or not.

$$C = -\frac{1}{|D|} \sum_{(x,y) \in D} (y \cdot \log(\hat{y}(x)) + (1 - y) \cdot \log(1 - \hat{y}(x))) \tag{F.3}$$

Equation F.3 calculates the cost for a neural network, in relation to a dataset D, where D contains the input-output pairs (x, y). $\hat{y}(x)$ is the output of the activation function of the output layer in the network, a^L.

Backpropagation

The purpose of backpropagation is to calculate the gradients of the cost function with respect to the parameters of the neural network.

The gradients of the cost function are calculated as the partial derivatives of C with respect to each weight in $\boldsymbol{W}$, denoted as $\frac{\partial C}{\partial w_{jk}^l}$. Using the chain rule, this can be rewritten as shown in Equation F.4, where the partial derivatives on the right-hand side each represent a backward step in the backpropagation.

$$\frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial a_j^l} \cdot \frac{\partial a_j^l}{\partial z_j^l} \cdot \frac{\partial z_j^l}{\partial w_{jk}^l} \tag{F.4}$$

As $\frac{\partial C}{\partial a_j^l} \cdot \frac{\partial a_j^l}{\partial z_j^l}$ is used for calculating the gradients with respect to all weights to neuron j in layer l, we introduce δ_j^l as an error term such that $\delta_j^l = \frac{\partial C}{\partial a_j^l} \cdot \frac{\partial a_j^l}{\partial z_j^l}$.

If we introduce $\boldsymbol{\nabla_a C}$ as a vector of partial derivatives of C with respect to each activation in the output layer, shown in Equation F.5a, then the error term of the output layer can be calculated

as shown in Equation F.9a. σ' is the derivative of the activation function with respect to the inputs to that activation function, z_j^L, such that the output of σ' is the vector shown in Equation F.5b.

$$\boldsymbol{\nabla}_{\boldsymbol{a}}\boldsymbol{C} = \begin{pmatrix} \frac{\partial C}{\partial a_1^L} \\ \vdots \\ \frac{\partial C}{\partial a_{|a^L|}^L} \end{pmatrix} \tag{F.5a}$$

$$\sigma'(\boldsymbol{z}^L) = \begin{pmatrix} \frac{\partial a_1^L}{\partial z_1^L} \\ \vdots \\ \frac{\partial a_{|a^L|}^L}{\partial z_{|a^L|}^L} \end{pmatrix} \tag{F.5b}$$

The error term of the output layer is used to calculate the error term of the preceding layer, using Equation F.9b, which is in turn used to further propagate the error term backwards through the network. This means that the error term has to include these extra backward steps, which can be seen in the expansion of the chain rule (Equation F.4), shown in Equation F.6.

$$\frac{\partial C}{\partial w_{jk}^l} = \sum_i \left(\frac{\partial C}{\partial a_i^{l+1}} \cdot \frac{\partial a_i^{l+1}}{\partial z_i^{l+1}} \cdot \frac{\partial z_i^{l+1}}{\partial z_j^l} \right) \cdot \frac{\partial z_j^l}{\partial w_{jk}^l} \tag{F.6}$$

The expansion introduces the term $\frac{\partial z_i^{l+1}}{\partial z_j^l}$ shown in Equation F.7, where the right hand side is derived from Equation F.1.

$$\frac{\partial z_i^{l+1}}{\partial z_j^l} = w_{ij}^{l+1} \cdot \sigma'(z_j^l) \tag{F.7}$$

In Equation F.8, the partial derivatives inside the summation in Equation F.6 is substituted by the error term from the following layer and Equation F.7.

$$\frac{\partial C}{\partial w_{jk}^l} = \sum_i \left(\delta_i^{l+1} \cdot w_{ij}^{l+1} \cdot \sigma'(z_j^l) \right) \cdot \frac{\partial z_j^l}{\partial w_{jk}^l} \tag{F.8}$$

The summation in Equation F.6 can be expressed in the form of matrix-vector multiplication, calculating all error terms for a layer, as shown in Equation F.9b.

$$\boldsymbol{\delta}^L = \boldsymbol{\nabla}_{\boldsymbol{a}}\boldsymbol{C} \odot \sigma'(\boldsymbol{z}^L) = \begin{pmatrix} \frac{\partial C}{\partial z_1^L} \\ \vdots \\ \frac{\partial C}{\partial z_{|a^L|}^L} \end{pmatrix} = \begin{pmatrix} \delta_1^L \\ \vdots \\ \delta_{|a^L|}^L \end{pmatrix} \tag{F.9a}$$

$$\boldsymbol{\delta}^l = ((\boldsymbol{W}^{l+1})^T \cdot \boldsymbol{\delta}^{l+1}) \odot \sigma'(\boldsymbol{z}^l) = \begin{pmatrix} \frac{\partial C}{\partial z_1^l} \\ \vdots \\ \frac{\partial C}{\partial z_{|a^l|}^l} \end{pmatrix} = \begin{pmatrix} \delta_1^l \\ \vdots \\ \delta_{|a^l|}^l \end{pmatrix} \tag{F.9b}$$

Now that the error terms are introduced, $\frac{\partial z_j^l}{\partial w_{jk}^l}$ from Equation F.4 and Equation F.6 is the only additional term that needs to be considered. $\frac{\partial z_j^l}{\partial w_{jk}^l}$ is calculated as the derivative of Equation F.1a

with respect to w^l_{jk}, which has the result: a^{l-1}_k. Therefore, the gradients of C with respect to each parameter in the neural network are calculated as shown in Equation F.10.

$$\frac{\partial C}{\partial w^l_{jk}} = \delta^l_j \cdot a^{l-1}_k \tag{F.10}$$

Optimization

When the gradients have been computed, they can be used to update the parameters of the neural network. One way of doing this is through gradient descent, which updates the parameters along their gradients towards a local optimum for the cost function. Equation F.11 calculates the delta values for each weight, which is how much the weights are increased or decreased in the optimization step.

$$\Delta w^l_{jk} = -\mu \cdot \frac{1}{m} \cdot \sum_i^m \frac{\partial C_{x_i}}{\partial w^l_{jk}} \tag{F.11}$$

Here, μ is the learning rate, which is a small positive real number that adjusts how much the parameters of the neural network are updated in a single training step. The delta values are calculated as the learning rate multiplied by the average of the gradients for data samples x_i. If we consider *simple gradient descend*, the gradients are calculated for the entire dataset, meaning that m denotes the total number of data samples in Equation F.11. This can be time consuming for large datasets, but can be improved by using *stochastic gradient descend*. In stochastic gradient descend, a randomly selected batch of training samples are considered at a time, and the average gradient for the training samples are calculated. In this case, m in Equation F.11 denotes the number of samples in the batch.

Choosing the value of the learning rate is important for the training of the network. A high learning rate makes larger changes to the parameters and thus converges faster, whereas a low learning rate makes smaller changes and is better at fine-tuning parameters. Therefore, it is advantageous to have a high learning rate early in the training process, to converge faster, and then change to a low learning rate later in the training process, to fine-tune parameters closer to the optimum. *Adam* is a method that uses this concept of adaptive learning rate, which finds individual learning rates for updating different parameters in the network.

Appendix G

Adam Optimizer

This section is based on information from [16], [57] and [58].
In this section, we cover the optimizer we use when training models throughout this project. The Adaptive Moment Estimation (Adam) is an adaptive optimizer where each parameter in the model has its own adaptive learning rate. Here, the gradients of a single parameter $g_{t,i}$ for the tth timestep for parameter i, with regards to loss function L, is defined as:

$$g_{t,i} = \nabla_{\theta_t} L(\theta)_{t,i} \tag{G.1}$$

Using this definition, we can define v_t as the exponentially moving average of the squared gradient from previous t timesteps. v_t is used to scale the global learning rate in relation to individual parameters, such that each parameter can have an individual learning rate. Here, if a parameter sees small updates, v_t will slowly decay, in turn resulting in an increase in learning rate for these parameters. v_t uses a decay factor β_2, where Kingma et al. propose a default value of 0.999 for β_2.

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{G.2}$$

In addition to this, we also define m_t, which is used for scaling the learning rate in a momentum like manner. This means that parameters with high gradients get higher m_t, which results in larger step sizes. m_t uses a different decay factor, namely β_1. Here, Kingma et al. propose a default value of 0.9 for β_1. The exponentially moving average for m_t is also based on previous gradients, but unlike v_t they are not squared:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{G.3}$$

As v_0 and m_0 are initialized to both be 0, the following v_t and m_t are biased towards 0. Due to this, Adam uses a regularization step for v_t and m_t:

$$\hat{m}_t = \frac{m_t}{(1 - \beta_1^t)} \tag{G.4a}$$

$$\hat{v}_t = \frac{v_t}{(1 - \beta_2^t)} \tag{G.4b}$$

Equation G.5 shows the update rule for Adam, which computes the updates for the parameters in the network. The rule makes use of the regularized $\hat{v}_t$ and $\hat{m}_t$, as well as a global learning rate α and an ϵ to avoid division by zero, where Kingma et al. propose a default of 10^{-8} for ϵ.

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t \tag{G.5}$$

Appendix H

This appendix is a copy of parts of Chapter 5 "Reasoning about Uncertainty" from our pre-master thesis [10].

Calibration

H.1 Calibration of a Neural Network

The calibration of a network indicates how well the confidence output of a network fits with the actual probability of an event. The confidence output corresponds to the network's predicted probability of the event, as a value between 0 and 1. The higher the confidence is, the more likely the event is, and vise versa.

If a neural network outputs ten predictions for the next ten days, where each prediction has a confidence of 0.3 for rain, we expect it to rain on three out of ten days. This means that the confidence output should be equal to the actual probability. However, this is not always the case, as proposed by the paper by Guo et al. [11], where they discover that modern neural networks are often miscalibrated. This is problematic if such a network is used for high risk decision making, such as whether to brake in self driving cars, or as a second opinion system for clinical use. For example, if a network has a low confidence of 0.1 that a patient gets sepsis, and $\frac{9}{10}$ patients with similar symptoms get sepsis, we have a bad calibration that might lead to a missed diagnosis. Perfect calibration can be formally defined as [11]:

$$\boldsymbol{P}(\hat{Y} = Y|\hat{P} = p) = p, \forall p \in [0,1] \tag{H.1}$$

Given $\hat{P}$, the predicted confidence, we assess the probability of the predicted class label $\hat{Y}$ being equal to the actual ground truth class label Y. For a perfectly calibrated network, this expression should be equal to the confidence for a specific prediction p. The left hand side of Equation H.1 corresponds to the accuracy of a network, and the right hand side corresponds to the confidence of a network. Since perfect calibration cannot be measured with a finite dataset, the accuracy and confidence are approximated by splitting the predictions into M bins, B_m, and then calculated for each bin using Equation H.2 and Equation H.3.

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i) \tag{H.2}$$

Equation H.2 shows how to compute accuracy for a binary classifier. Here, $\hat{y}_i$ is the predicted class label for sample i, y_i is the ground truth class label for sample i, and $\mathbf{1}$ is an indicator function. The indicator function is used to determine whether the predicted label is equal to the ground truth label, where 1 is returned if they are equal, and 0 is returned otherwise. Note that each prediction is considered positive, such that x predictions with a confidence of 0.2 should achieve an accuracy of 0.2 to be perfectly calibrated.

$$conf(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i \tag{H.3}$$

Equation H.3 shows how to compute the confidence for a binary classifier. Here, $\hat{p}$ is the predicted probability that sample i is positive, such that the confidence of a bin equals the average predicted probability for that bin.

By partitioning the predictions into bins, we can compute the difference between accuracy and confidence for each bin as a measure for calibration error. Calibration can be viewed as a way of telling to which degree a model actually reasons about its uncertainty [59]. If a network is well calibrated, its output confidence expresses the actual frequency, whereas a large difference between confidence and accuracy means that there is a lot of uncertainty tied to the prediction.

Additionally, Guo et al. [11] find that three general concepts used in machine learning affect the calibration of a network, namely model size (width and depth), weight decay and batch normalization. The descriptions for these, based on [16], and their effect on calibration, according to [11], is as follows:

- Increasing model width (adding more parameters per layer) and depth (adding more layers) both yield a progressively worse calibration.
- Weight decay concerns the regularization of a weight, where a factor between 0 and 1 is used to downscale the weight, which in turn reduces the chances of the network overfitting on training data. Using less weight decay has a negative impact on calibration.
- Batch normalization is a method for speeding up training in a network, by using batches of input [60]. By normalizing the output of each activation function in a layer, all outputs have a standard deviation of one and a mean of zero. This limits fluctuation of the output, which speeds up training of the network. Training models with batch normalization shows that batch normalization results in better accuracy, but worse calibration.

These findings make it interesting to consider these concepts when further working with calibration.

H.1.1 Expected Calibration Error

The calibration measure presented by Guo et al. [11] is called Expected Calibration Error (ECE), and is defined as:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)| \tag{H.4}$$

Here, the predictions are partitioned into M bins, for a total number of n data samples across all bins. Each bin includes the predictions from a specific prediction confidence interval, e.g. all predictions with confidence between 0 and 0.2. A perfectly calibrated network has an ECE value of zero, and the higher the value the worse calibrated the network is.

The comparison between confidence and accuracy can be visualized in a reliability diagram, as seen in Figure H.1, with confidence on the x-axis and accuracy on the y-axis. As mentioned earlier, perfect calibration is when the confidence is equal to the accuracy, which can be seen as the diagonal line. The blue bars represent the average accuracy for a given bin and the transparent red bars represent the difference between the calibration of the model and perfect calibration. When

the bar is above the diagonal, as with the sixth bin, it indicates that the model is underconfident for that bin, since the accuracy is higher than the confidence. Likewise, if the bar is below the diagonal, as with the last bin, it indicates that the model is overconfident, since the confidence is higher than the accuracy.

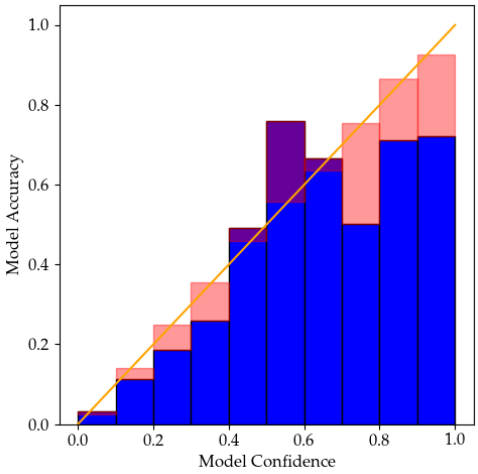


Figure H.1: Example reliability diagram for the ECE measure.

H.1.2 Adaptive Calibration Error

Nixon et al. [61] presents another calibration measure, called Adaptive Calibration Error (ACE). ACE is adapted to consider all predictions in a multi-class classification setting, and it uses a different binning scheme. The equation for ACE is defined as:

$$ACE = \frac{1}{KR} \sum_{k=1}^{K} \sum_{r=1}^{R} |acc(r,k) - conf(r,k)| \tag{H.5}$$

Here, K is the number of classes, R is the number of ranges, and $acc(r,k)$ is the accuracy for class k in range r. The ranges work similarly to bins, but rather than including an interval for confidence, each range includes a specific number of the total predictions, sorted by confidence value, such that all predictions are spread evenly across the R ranges. This prevents the cases where the bins are imbalanced, meaning that some bins include the majority of predictions, and other bins include only a small number of predictions.

Appendix I

BNN Implementation Code Snippets

```
def _get_dense_from_variational(layer, input_shape):
    weights = layer.weights[0]
    len_parameters = weights.shape[0]
    half_len = int(len_parameters/2)

    #Extracts the first half of the parameters for mean, and the second half for loc.
    loc_parameters = weights[0:half_len]
    scale_parameters = weights[-half_len:]

    output_shape = layer.output_shape[1]

    bias_size = output_shape
    weight_size = half_len - bias_size

    #Splits params for loc into params for weights and for biases.
    loc_weights = loc_parameters[0:weight_size]
    loc_biases = loc_parameters[-bias_size:]

    #Splits params for scale into params for weights and for biases.
    scale_weights = scale_parameters[0:weight_size]
    scale_biases = scale_parameters[-bias_size:]

    #Generates normaldistributions from the parameters and samples a weight or bias
        from each.
    weights = []
    for loc_weight, scale_weight in zip(loc_weights, scale_weights):
        weights.append(tfd.Normal(loc=loc_weight, scale=1e-5 +
            tf.nn.softplus(scale_weight)).sample())

    biases = []
    for loc_bias, scale_bias in zip(loc_biases, scale_biases):
        biases.append(tfd.Normal(loc=loc_bias, scale=1e-5 +
            tf.nn.softplus(scale_bias)).sample())

    #Constructs a initializer used for building the corresponding dense layer.
    weight_initializer = tf.keras.initializers.constant(np.array(weights))
    bias_initializer = tf.keras.initializers.constant(np.array(biases))

```

```
    dense = tf.keras.layers.Dense(units=output_shape,
        kernel_initializer=weight_initializer, bias_initializer=bias_initializer,
        activation='relu')
    dense.build(input_shape)
    return dense
```

Code snippet I.1: Function for sampling parameters from a specific layer in the BNN posterior.

Appendix J

FFNN Grid Search

J.1 FFNN Grid Search On ΔDCT

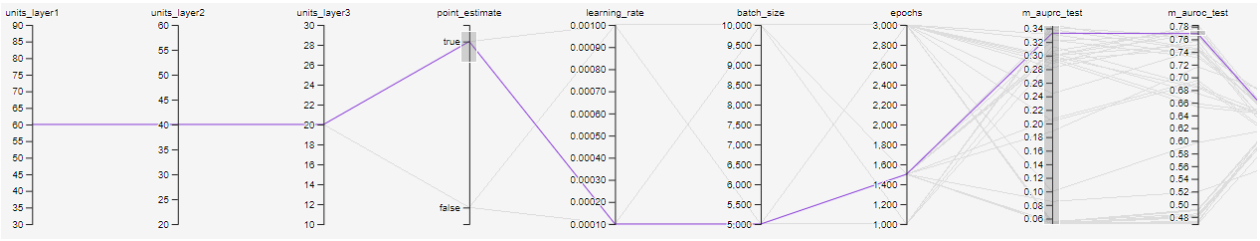


Figure J.1: FFNN grid search on ΔDCT.

```
"experiments": [
    {
        "model": "bayesian_manual_posterior_model",
        "plot": false,
        "calculate_uncertainty_metric": true,
        "hyperparameters": [
            {
                "point_estimate": [true],
                "units_layer1": [60],
                "units_layer2": [40],
                "units_layer3": [20],
                "batch_size": [5000, 10000],
                "learning_rate": [0.0001, 0.001],
                "epochs": [1500, 3000, 1000],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["auprc"],
                "metric2": ["auroc"]
            }
        ]
    }
]
```

Code snippet J.1: Configuration of the best FFNN, found in Figure J.1.

The best performing parameters:

- 5,000 Batch Size.
- 0.0001 Learning Rate.
- 1,500 Epochs.

J.2 FFNN Grid Search On ΔDP-A

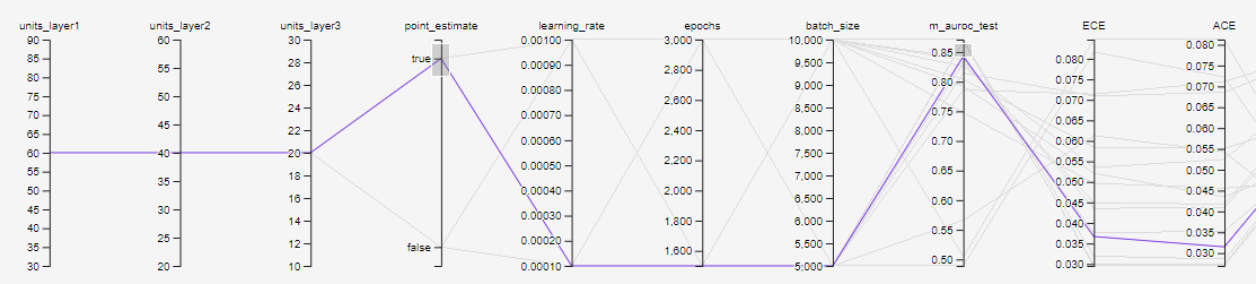


Figure J.2: FFNN grid search on ΔDP-A.

```
"experiments": [
    {
        "model": "bayesian_manual_posterior_model",
        "plot": false,
        "calculate_uncertainty_metric": true,
        "hyperparameters": [
            {
                "point_estimate": [true],
                "units_layer1": [60],
                "units_layer2": [40],
                "units_layer3": [20],
                "batch_size": [5000, 10000],
                "learning_rate": [0.0001, 0.001],
                "epochs": [1500, 3000, 1000],
                "optimizer": ["adam"],
                "loss_function": ["binary_crossentropy"],
                "metric1": ["auprc"],
                "metric2": ["auroc"]
            }
        ]
    }
]
```

Code snippet J.2: Configuration of the best FFNN, found in Figure J.2.

The best performing parameters:

- 5,000 Batch Size.
- 0.0001 Learning Rate.
- 1,500 Epochs.

Appendix K

Uncertainty Experiments

The most relevant results can be found in this chapter, and the full set of results can be found in the Supplementary Appendix.

K.1 Disagreement Results (PhysioNet)

Positive Threshold	DL	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	175	2842	-	161	103	14	2739	0.7472	0.8820
0.1	0 (⌊0%⌋)	204	2622	191	171	70	33	2552	0.7703	0.8847
0.2	0 (⌊0%⌋)	180	2715	122	163	79	17	2636	0.7599	0.8806
0.3	0 (⌊0%⌋)	171	2751	95	159	84	12	2667	0.7539	0.8790
0.4	0 (⌊0%⌋)	161	2770	86	153	90	8	2680	0.7519	0.8784
0.5	0 (⌊0%⌋)	150	2789	78	145	94	5	2695	0.7429	0.8753
0.6	0 (⌊0%⌋)	140	2807	70	136	99	4	2708	0.7308	0.8719
0.7	0 (⌊0%⌋)	137	2826	54	133	106	4	2720	0.7301	0.8729
0.8	0 (⌊0%⌋)	127	2836	54	124	108	3	2728	0.7173	0.8684
0.9	0 (⌊0%⌋)	111	2854	52	110	118	1	2736	0.7072	0.8654
0.1	1 (⌊20%⌋)	237	2706	74	177	76	60	2630	0.7583	0.8826
0.2	1 (⌊20%⌋)	202	2767	48	171	87	31	2680	0.7561	0.8831
0.3	1 (⌊20%⌋)	184	2792	41	164	89	20	2703	0.7491	0.8799
0.4	1 (⌊20%⌋)	173	2810	34	160	95	13	2715	0.7487	0.8804
0.5	1 (⌊20%⌋)	162	2825	30	153	99	9	2726	0.7423	0.8783
0.6	1 (⌊20%⌋)	155	2841	21	147	105	8	2736	0.7385	0.8774
0.7	1 (⌊20%⌋)	146	2850	21	141	112	5	2738	0.7403	0.8780
0.8	1 (⌊20%⌋)	139	2861	17	135	118	4	2743	0.7384	0.8776
0.9	1 (⌊20%⌋)	121	2875	21	120	127	1	2748	0.7293	0.8745
0.1	2 (⌊40%⌋)	267	2750	0	182	82	85	2668	0.7472	0.8820
0.2	2 (⌊40%⌋)	225	2792	0	175	89	50	2703	0.7472	0.8820
0.3	2 (⌊40%⌋)	201	2816	0	171	93	30	2723	0.7472	0.8820
0.4	2 (⌊40%⌋)	185	2832	0	165	99	20	2733	0.7472	0.8820
0.5	2 (⌊40%⌋)	172	2845	0	161	103	11	2742	0.7472	0.8820
0.6	2 (⌊40%⌋)	166	2851	0	155	109	11	2742	0.7472	0.8820
0.7	2 (⌊40%⌋)	157	2860	0	148	116	9	2744	0.7472	0.8820
0.8	2 (⌊40%⌋)	150	2867	0	144	120	6	2747	0.7472	0.8820

0.9	2 (⌊40%⌋)	131	2886	0	128	136	3	2750	0.7472	0.8820

Table K.1: PhysioNet: Disagreement results where 5 models are sampled. DL is the disagreement limit specifying how many predictions on the same data point can disagree before the model expresses "i do not know" (IDK). IDK is the number of data points for which the sampled models disagree.

Positive Threshold	DL	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	175	2842	-	159	105	16	2737	0.7471	0.8827
0.1	0 (⌊0%⌋)	195	2570	252	172	69	23	2501	0.7829	0.8897
0.2	0 (⌊0%⌋)	170	2669	178	161	75	9	2594	0.7682	0.8825
0.3	0 (⌊0%⌋)	160	2719	138	154	81	6	2638	0.7562	0.8788
0.4	0 (⌊0%⌋)	150	2750	117	146	88	4	2662	0.7488	0.8765
0.5	0 (⌊0%⌋)	142	2776	99	140	90	2	2686	0.7375	0.8726
0.6	0 (⌊0%⌋)	135	2794	88	133	95	2	2699	0.7294	0.8702
0.7	0 (⌊0%⌋)	128	2814	75	126	103	2	2711	0.7234	0.8695
0.8	0 (⌊0%⌋)	120	2830	67	119	107	1	2723	0.7141	0.8667
0.9	0 (⌊0%⌋)	103	2854	60	102	116	1	2738	0.6901	0.8601
0.1	1 (⌊10%⌋)	212	2660	145	173	71	39	2589	0.7658	0.8841
0.2	1 (⌊10%⌋)	188	2737	92	167	82	21	2655	0.7607	0.8829
0.3	1 (⌊10%⌋)	170	2762	85	159	85	11	2677	0.7539	0.8797
0.4	1 (⌊10%⌋)	161	2788	68	155	91	6	2697	0.7508	0.8794
0.5	1 (⌊10%⌋)	153	2804	60	148	94	5	2710	0.7407	0.8763
0.6	1 (⌊10%⌋)	145	2827	45	141	104	4	2723	0.7395	0.8766
0.7	1 (⌊10%⌋)	138	2838	41	135	108	3	2730	0.7343	0.8750
0.8	1 (⌊10%⌋)	130	2852	35	128	114	2	2738	0.7288	0.8738
0.9	1 (⌊10%⌋)	111	2869	37	110	123	1	2746	0.7088	0.8680
0.1	2 (⌊20%⌋)	226	2688	103	175	74	51	2614	0.7607	0.8835
0.2	2 (⌊20%⌋)	198	2764	55	173	88	25	2676	0.7625	0.8860
0.3	2 (⌊20%⌋)	179	2794	44	162	93	17	2701	0.7526	0.8821
0.4	2 (⌊20%⌋)	168	2810	39	157	94	11	2716	0.7460	0.8795
0.5	2 (⌊20%⌋)	159	2823	35	152	98	7	2725	0.7432	0.8786
0.6	2 (⌊20%⌋)	150	2838	29	145	108	5	2730	0.7456	0.8797
0.7	2 (⌊20%⌋)	143	2850	24	138	113	5	2737	0.7388	0.8780
0.8	2 (⌊20%⌋)	134	2861	22	131	119	3	2742	0.7358	0.8772
0.9	2 (⌊20%⌋)	118	2875	24	116	129	2	2746	0.7256	0.8743
0.1	3 (⌊30%⌋)	239	2718	60	176	78	63	2640	0.7554	0.8828
0.2	3 (⌊30%⌋)	205	2783	29	173	91	32	2692	0.7564	0.8852
0.3	3 (⌊30%⌋)	192	2804	21	168	94	24	2710	0.7522	0.8836
0.4	3 (⌊30%⌋)	176	2819	22	160	96	16	2723	0.7458	0.8806
0.5	3 (⌊30%⌋)	164	2828	25	154	101	10	2727	0.7462	0.8804

0.6	3 (⌊30%⌋)	154	2843	20	148	111	6	2732	0.7511	0.8821
0.7	3 (⌊30%⌋)	144	2856	17	139	115	5	2741	0.7401	0.8790
0.8	3 (⌊30%⌋)	138	2867	12	134	122	4	2745	0.7403	0.8795
0.9	3 (⌊30%⌋)	125	2879	13	122	131	3	2748	0.7343	0.8779
0.1	4 (⌊40%⌋)	254	2746	17	178	81	76	2665	0.7491	0.8819
0.2	4 (⌊40%⌋)	213	2791	13	173	91	40	2700	0.7509	0.8838
0.3	4 (⌊40%⌋)	199	2815	3	170	94	29	2721	0.7481	0.8830
0.4	4 (⌊40%⌋)	184	2826	7	163	97	21	2729	0.7450	0.8813
0.5	4 (⌊40%⌋)	171	2837	9	157	103	14	2734	0.7459	0.8815
0.6	4 (⌊40%⌋)	162	2849	6	150	113	12	2736	0.7492	0.8828
0.7	4 (⌊40%⌋)	149	2860	8	143	116	6	2744	0.7434	0.8808
0.8	4 (⌊40%⌋)	142	2873	2	137	125	5	2748	0.7446	0.8818
0.9	4 (⌊40%⌋)	131	2883	3	128	134	3	2749	0.7456	0.8819

Table K.2: PhysioNet: Disagreement results where 10 models are sampled. DL is the disagreement limit specifying how many predictions on the same data point can disagree before the model expresses "i do not know" (IDK). IDK is the number of data points for which the sampled models disagree.

Positive Threshold	DL	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	177	2840	-	161	103	16	2737	0.7487	0.8845
0.1	0 (⌊0%⌋)	172	2423	422	157	56	15	2367	0.7930	0.8900
0.2	0 (⌊0%⌋)	153	2580	284	147	71	6	2509	0.7600	0.8813
0.3	0 (⌊0%⌋)	145	2649	223	141	75	4	2574	0.7462	0.8757
0.4	0 (⌊0%⌋)	137	2688	192	135	78	2	2610	0.7353	0.8713
0.5	0 (⌊0%⌋)	126	2724	167	125	86	1	2638	0.7237	0.8678
0.6	0 (⌊0%⌋)	119	2756	142	118	88	1	2668	0.7052	0.8618
0.7	0 (⌊0%⌋)	115	2775	127	114	97	1	2678	0.7079	0.8640
0.8	0 (⌊0%⌋)	107	2803	107	106	104	1	2699	0.6923	0.8609
0.9	0 (⌊0%⌋)	92	2828	97	91	111	1	2717	0.6677	0.8535
0.1	5 (⌊10%⌋)	202	2627	188	173	70	29	2557	0.7726	0.8885
0.2	5 (⌊10%⌋)	176	2717	124	161	80	15	2637	0.7604	0.8831
0.3	5 (⌊10%⌋)	166	2760	91	155	88	11	2672	0.7536	0.8815
0.4	5 (⌊10%⌋)	153	2785	79	148	91	5	2694	0.7440	0.8782
0.5	5 (⌊10%⌋)	147	2795	75	144	93	3	2702	0.7384	0.8766
0.6	5 (⌊10%⌋)	139	2816	62	137	101	2	2715	0.7345	0.8759
0.7	5 (⌊10%⌋)	132	2829	56	131	104	1	2725	0.7263	0.8735
0.8	5 (⌊10%⌋)	123	2844	50	122	109	1	2735	0.7143	0.8702
0.9	5 (⌊10%⌋)	106	2860	51	105	118	1	2742	0.6953	0.8647
0.1	10 (⌊20%⌋)	218	2677	122	175	73	43	2604	0.7658	0.8865
0.2	10 (⌊20%⌋)	190	2751	76	169	85	21	2666	0.7619	0.8861

0.3	10 (⌊20%⌋)	173	2782	62	161	89	12	2693	0.7533	0.8829
0.4	10 (⌊20%⌋)	162	2804	51	154	94	8	2710	0.7468	0.8808
0.5	10 (⌊20%⌋)	153	2818	46	148	98	5	2720	0.7419	0.8792
0.6	10 (⌊20%⌋)	146	2833	38	143	103	3	2730	0.7383	0.8784
0.7	10 (⌊20%⌋)	139	2841	37	137	106	2	2735	0.7324	0.8765
0.8	10 (⌊20%⌋)	132	2854	31	131	115	1	2739	0.7345	0.8777
0.9	10 (⌊20%⌋)	117	2870	30	116	125	1	2745	0.7222	0.8744
0.1	15 (⌊30%⌋)	231	2710	76	177	78	54	2632	0.7612	0.8864
0.2	15 (⌊30%⌋)	199	2777	41	172	89	27	2688	0.7582	0.8866
0.3	15 (⌊30%⌋)	183	2799	35	164	92	19	2707	0.7514	0.8836
0.4	15 (⌊30%⌋)	168	2816	33	158	95	10	2721	0.7473	0.8819
0.5	15 (⌊30%⌋)	159	2827	31	152	99	7	2728	0.7435	0.8806
0.6	15 (⌊30%⌋)	150	2838	29	146	104	4	2734	0.7411	0.8798
0.7	15 (⌊30%⌋)	144	2849	24	141	110	3	2739	0.7399	0.8799
0.8	15 (⌊30%⌋)	138	2864	15	136	119	2	2745	0.7420	0.8812
0.9	15 (⌊30%⌋)	125	2877	15	124	128	1	2749	0.7354	0.8794
0.1	20 (⌊40%⌋)	247	2734	36	179	79	68	2655	0.7543	0.8848
0.2	20 (⌊40%⌋)	206	2790	21	174	90	32	2700	0.7548	0.8863
0.3	20 (⌊40%⌋)	194	2810	13	169	92	25	2718	0.7499	0.8842
0.4	20 (⌊40%⌋)	179	2824	14	163	96	16	2728	0.7481	0.8833
0.5	20 (⌊40%⌋)	168	2838	11	157	102	11	2736	0.7467	0.8830
0.6	20 (⌊40%⌋)	156	2845	16	150	108	6	2737	0.7479	0.8830
0.7	20 (⌊40%⌋)	147	2860	10	144	115	3	2745	0.7465	0.8829
0.8	20 (⌊40%⌋)	142	2870	5	140	121	2	2749	0.7468	0.8834
0.9	20 (⌊40%⌋)	128	2883	6	127	132	1	2751	0.7434	0.8824

Table K.3: PhysioNet: Disagreement results where 50 models are sampled. DL is the disagreement limit specifying how many predictions on the same data point can disagree before the model expresses "i do not know" (IDK). IDK is the number of data points for which the sampled models disagree.

Positive Threshold	DL	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	176	2841	-	161	103	15	2738	0.7478	0.8833
0.1	0 (⌊0%⌋)	165	2334	518	155	53	10	2281	0.8054	0.8937
0.2	0 (⌊0%⌋)	144	2508	365	141	67	3	2441	0.7664	0.8797
0.3	0 (⌊0%⌋)	138	2580	299	136	71	2	2509	0.7487	0.8740
0.4	0 (⌊0%⌋)	125	2637	255	123	76	2	2561	0.7227	0.8646
0.5	0 (⌊0%⌋)	118	2677	222	117	81	1	2596	0.7136	0.8612
0.6	0 (⌊0%⌋)	116	2716	185	115	86	1	2630	0.7088	0.8604
0.7	0 (⌊0%⌋)	107	2749	161	106	90	1	2659	0.6901	0.8542
0.8	0 (⌊0%⌋)	99	2781	137	98	99	1	2682	0.6792	0.8525

0.9	0 (⌊0%⌋)	88	2816	113	87	108	1	2708	0.6565	0.8478
0.1	15 (⌊10%⌋)	199	2626	192	172	70	27	2556	0.7738	0.8871
0.2	15 (⌊10%⌋)	175	2715	127	162	78	13	2637	0.7620	0.8817
0.3	15 (⌊10%⌋)	162	2760	95	153	89	9	2671	0.7525	0.8800
0.4	15 (⌊10%⌋)	151	2781	85	147	91	4	2690	0.7435	0.8769
0.5	15 (⌊10%⌋)	147	2796	74	144	95	3	2701	0.7408	0.8765
0.6	15 (⌊10%⌋)	140	2814	63	137	100	3	2714	0.7325	0.8742
0.7	15 (⌊10%⌋)	133	2827	57	132	104	1	2723	0.7280	0.8729
0.8	15 (⌊10%⌋)	120	2841	56	119	108	1	2733	0.7082	0.8668
0.9	15 (⌊10%⌋)	104	2860	53	103	120	1	2740	0.6959	0.8636
0.1	30 (⌊20%⌋)	214	2682	121	174	72	40	2610	0.7649	0.8843
0.2	30 (⌊20%⌋)	188	2756	73	169	86	19	2670	0.7623	0.8852
0.3	30 (⌊20%⌋)	175	2785	57	162	91	13	2694	0.7548	0.8828
0.4	30 (⌊20%⌋)	163	2802	52	154	94	9	2708	0.7462	0.8796
0.5	30 (⌊20%⌋)	152	2816	49	147	98	5	2718	0.7406	0.8776
0.6	30 (⌊20%⌋)	148	2831	38	144	103	4	2728	0.7394	0.8778
0.7	30 (⌊20%⌋)	137	2840	40	135	106	2	2734	0.7292	0.8743
0.8	30 (⌊20%⌋)	131	2856	30	130	116	1	2740	0.7330	0.8763
0.9	30 (⌊20%⌋)	117	2869	31	116	124	1	2745	0.7206	0.8727
0.1	45 (⌊30%⌋)	230	2713	74	176	79	54	2634	0.7603	0.8851
0.2	45 (⌊30%⌋)	197	2775	45	172	89	25	2686	0.7589	0.8858
0.3	45 (⌊30%⌋)	183	2799	35	165	92	18	2707	0.7519	0.8829
0.4	45 (⌊30%⌋)	170	2817	30	159	97	11	2720	0.7494	0.8820
0.5	45 (⌊30%⌋)	159	2830	28	151	100	8	2730	0.7412	0.8791
0.6	45 (⌊30%⌋)	150	2840	27	145	105	5	2735	0.7392	0.8784
0.7	45 (⌊30%⌋)	146	2848	23	142	109	4	2739	0.7388	0.8786
0.8	45 (⌊30%⌋)	137	2861	19	135	117	2	2744	0.7382	0.8787
0.9	45 (⌊30%⌋)	121	2877	19	120	128	1	2749	0.7288	0.8762
0.1	60 (⌊40%⌋)	241	2746	30	178	82	63	2664	0.7533	0.8840
0.2	60 (⌊40%⌋)	207	2791	19	173	90	34	2701	0.7524	0.8845
0.3	60 (⌊40%⌋)	192	2809	16	167	94	25	2715	0.7499	0.8833
0.4	60 (⌊40%⌋)	181	2825	11	164	97	17	2728	0.7483	0.8828
0.5	60 (⌊40%⌋)	167	2836	14	157	102	10	2734	0.7473	0.8821
0.6	60 (⌊40%⌋)	153	2845	19	148	106	5	2739	0.7421	0.8799
0.7	60 (⌊40%⌋)	147	2859	11	142	115	5	2744	0.7424	0.8807
0.8	60 (⌊40%⌋)	139	2869	9	137	122	2	2747	0.7449	0.8816
0.9	60 (⌊40%⌋)	127	2884	6	126	133	1	2751	0.7424	0.8812

Table K.4: PhysioNet: Disagreement results where 150 models are sampled. DL is the disagreement limit specifying how many predictions on the same data point can disagree before the model expresses "i do not know" (IDK). IDK is the number of data points for which the sampled models disagree.

K.2 Standard Deviation Results (PhysioNet)

Positive Threshold	Std Threshold	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	176	2841	-	160	104	16	2737	0.7424	0.8801
0.25	0.01	97	2404	516	96	62	1	2342	0.6979	0.8473
0.5	0.01	97	2404	516	96	62	1	2342	0.6979	0.8473
0.75	0.01	97	2404	516	96	62	1	2342	0.6979	0.8473
0.25	0.02	106	2546	365	105	67	1	2479	0.6948	0.8481
0.5	0.02	106	2546	365	105	67	1	2479	0.6948	0.8481
0.75	0.02	106	2546	365	105	67	1	2479	0.6948	0.8481
0.25	0.03	109	2618	290	108	72	1	2546	0.6938	0.8488
0.5	0.03	109	2618	290	108	72	1	2546	0.6938	0.8488
0.75	0.03	109	2618	290	108	72	1	2546	0.6938	0.8488
0.25	0.04	118	2664	235	116	77	2	2587	0.7019	0.8548
0.5	0.04	117	2665	235	116	77	1	2588	0.7019	0.8548
0.75	0.04	116	2666	235	115	78	1	2588	0.7019	0.8548
0.25	0.05	122	2684	211	120	78	2	2606	0.7069	0.8568
0.5	0.05	120	2686	211	119	79	1	2607	0.7069	0.8568
0.75	0.05	119	2687	211	118	80	1	2607	0.7069	0.8568
0.25	0.06	127	2714	176	125	83	2	2631	0.7144	0.8613
0.5	0.06	123	2718	176	122	86	1	2632	0.7144	0.8613
0.75	0.06	122	2719	176	121	87	1	2632	0.7144	0.8613
0.25	0.07	132	2725	160	130	85	2	2640	0.7212	0.8649
0.5	0.07	127	2730	160	126	89	1	2641	0.7212	0.8649
0.75	0.07	125	2732	160	124	91	1	2641	0.7212	0.8649
0.25	0.08	138	2740	139	135	86	3	2654	0.7259	0.8672
0.5	0.08	133	2745	139	131	90	2	2655	0.7259	0.8672
0.75	0.08	127	2751	139	126	95	1	2656	0.7259	0.8672
0.25	0.09	146	2748	123	143	87	3	2661	0.7360	0.8717
0.5	0.09	139	2755	123	137	93	2	2662	0.7360	0.8717
0.75	0.09	132	2762	123	131	99	1	2663	0.7360	0.8717
0.25	0.1	149	2757	111	145	87	4	2670	0.7358	0.8719
0.5	0.1	140	2766	111	138	94	2	2672	0.7358	0.8719
0.75	0.1	132	2774	111	131	101	1	2673	0.7358	0.8719
0.25	0.2	186	2789	42	161	90	25	2699	0.7396	0.8769
0.5	0.2	159	2816	42	150	101	9	2715	0.7396	0.8769
0.75	0.2	136	2839	42	134	117	2	2722	0.7396	0.8769
0.25	0.3	205	2798	14	170	90	35	2708	0.7431	0.8793

0.5	0.3	169	2834	14	157	103	12	2731	0.7431	0.8793
0.75	0.3	140	2863	14	137	123	3	2740	0.7431	0.8793
0.25	0.4	217	2799	1	174	90	43	2709	0.7428	0.8802
0.5	0.4	176	2840	1	160	104	16	2736	0.7428	0.8802
0.75	0.4	140	2876	1	137	127	3	2749	0.7428	0.8802
0.25	0.5	218	2799	0	174	90	44	2709	0.7424	0.8801
0.5	0.5	176	2841	0	160	104	16	2737	0.7424	0.8801
0.75	0.5	140	2877	0	137	127	3	2750	0.7424	0.8801
0.25	0.6	218	2799	0	174	90	44	2709	0.7424	0.8801
0.5	0.6	176	2841	0	160	104	16	2737	0.7424	0.8801
0.75	0.6	140	2877	0	137	127	3	2750	0.7424	0.8801

Table K.5: PhysioNet: Standard deviation results where 5 models are sampled. Std Threshold is an upper limit on standard deviation in predictions on the same data point, before the model expresses "i do not know" (IDK). IDK is the number of data points for which the model express "i do not know".

Positive Threshold	Std Threshold	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	173	2844	-	160	104	13	2740	0.7487	0.8844
0.25	0.01	92	2373	552	91	57	1	2316	0.6951	0.8465
0.5	0.01	92	2373	552	91	57	1	2316	0.6951	0.8465
0.75	0.01	92	2373	552	91	57	1	2316	0.6951	0.8465
0.25	0.02	103	2517	397	102	68	1	2449	0.6989	0.8540
0.5	0.02	103	2517	397	102	68	1	2449	0.6989	0.8540
0.75	0.02	103	2517	397	102	68	1	2449	0.6989	0.8540
0.25	0.03	110	2587	320	109	71	1	2516	0.7063	0.8566
0.5	0.03	110	2587	320	109	71	1	2516	0.7063	0.8566
0.75	0.03	110	2587	320	109	71	1	2516	0.7063	0.8566
0.25	0.04	113	2627	277	112	73	1	2554	0.7092	0.8574
0.5	0.04	113	2627	277	112	73	1	2554	0.7092	0.8574
0.75	0.04	111	2629	277	110	75	1	2554	0.7092	0.8574
0.25	0.05	116	2671	230	115	77	1	2594	0.7083	0.8588
0.5	0.05	116	2671	230	115	77	1	2594	0.7083	0.8588
0.75	0.05	114	2673	230	113	79	1	2594	0.7083	0.8588
0.25	0.06	121	2696	200	120	79	1	2617	0.7154	0.8619
0.5	0.06	119	2698	200	118	81	1	2617	0.7154	0.8619
0.75	0.06	117	2700	200	116	83	1	2617	0.7154	0.8619
0.25	0.07	125	2711	181	124	81	1	2630	0.7216	0.8648
0.5	0.07	122	2714	181	121	84	1	2630	0.7216	0.8648
0.75	0.07	119	2717	181	118	87	1	2630	0.7216	0.8648

0.25	0.08	128	2726	163	127	81	1	2645	0.7228	0.8655
0.5	0.08	125	2729	163	124	84	1	2645	0.7228	0.8655
0.75	0.08	122	2732	163	121	87	1	2645	0.7228	0.8655
0.25	0.09	134	2740	143	133	82	1	2658	0.7293	0.8687
0.5	0.09	130	2744	143	129	86	1	2658	0.7293	0.8687
0.75	0.09	125	2749	143	124	91	1	2658	0.7293	0.8687
0.25	0.1	139	2745	133	137	82	2	2663	0.7334	0.8706
0.5	0.1	133	2751	133	132	87	1	2664	0.7334	0.8706
0.75	0.1	128	2756	133	127	92	1	2664	0.7334	0.8706
0.25	0.2	182	2790	45	160	89	22	2701	0.7458	0.8807
0.5	0.2	157	2815	45	150	99	7	2716	0.7458	0.8807
0.75	0.2	141	2831	45	137	112	4	2719	0.7458	0.8807
0.25	0.3	206	2800	11	172	91	34	2709	0.7530	0.8850
0.5	0.3	167	2839	11	159	104	8	2735	0.7530	0.8850
0.75	0.3	144	2862	11	140	123	4	2739	0.7530	0.8850
0.25	0.4	216	2801	0	173	91	43	2710	0.7487	0.8844
0.5	0.4	173	2844	0	160	104	13	2740	0.7487	0.8844
0.75	0.4	144	2873	0	140	124	4	2749	0.7487	0.8844
0.25	0.5	216	2801	0	173	91	43	2710	0.7487	0.8844
0.5	0.5	173	2844	0	160	104	13	2740	0.7487	0.8844
0.75	0.5	144	2873	0	140	124	4	2749	0.7487	0.8844
0.25	0.6	216	2801	0	173	91	43	2710	0.7487	0.8844
0.5	0.6	173	2844	0	160	104	13	2740	0.7487	0.8844
0.75	0.6	144	2873	0	140	124	4	2749	0.7487	0.8844

Table K.6: PhysioNet: Standard deviation results where 10 models are sampled. Std Threshold is an upper limit on standard deviation in predictions on the same data point, before the model expresses "i do not know" (IDK). IDK is the number of data points for which the model express "i do not know".

Positive Threshold	Std Threshold	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	176	2841	-	161	103	15	2738	0.7464	0.8805
0.25	0.01	87	2270	660	86	54	1	2216	0.6960	0.8374
0.5	0.01	87	2270	660	86	54	1	2216	0.6960	0.8374
0.75	0.01	87	2270	660	86	54	1	2216	0.6960	0.8374
0.25	0.02	95	2452	470	94	61	1	2391	0.6891	0.8395
0.5	0.02	95	2452	470	94	61	1	2391	0.6891	0.8395
0.75	0.02	95	2452	470	94	61	1	2391	0.6891	0.8395
0.25	0.03	100	2539	378	99	66	1	2473	0.6905	0.8425
0.5	0.03	100	2539	378	99	66	1	2473	0.6905	0.8425

0.75	0.03	100	2539	378	99	66	1	2473	0.6905	0.8425
0.25	0.04	105	2597	315	104	72	1	2525	0.6923	0.8468
0.5	0.04	105	2597	315	104	72	1	2525	0.6923	0.8468
0.75	0.04	105	2597	315	104	72	1	2525	0.6923	0.8468
0.25	0.05	110	2631	276	109	74	1	2557	0.6970	0.8498
0.5	0.05	110	2631	276	109	74	1	2557	0.6970	0.8498
0.75	0.05	109	2632	276	108	75	1	2557	0.6970	0.8498
0.25	0.06	116	2653	248	115	77	1	2576	0.7078	0.8551
0.5	0.06	116	2653	248	115	77	1	2576	0.7078	0.8551
0.75	0.06	114	2655	248	113	79	1	2576	0.7078	0.8551
0.25	0.07	124	2674	219	123	78	1	2596	0.7197	0.8600
0.5	0.07	122	2676	219	121	80	1	2596	0.7197	0.8600
0.75	0.07	118	2680	219	117	84	1	2596	0.7197	0.8600
0.25	0.08	130	2692	195	129	81	1	2611	0.7273	0.8645
0.5	0.08	127	2695	195	126	84	1	2611	0.7273	0.8645
0.75	0.08	123	2699	195	122	88	1	2611	0.7273	0.8645
0.25	0.09	132	2710	175	131	81	1	2629	0.7265	0.8643
0.5	0.09	127	2715	175	126	86	1	2629	0.7265	0.8643
0.75	0.09	123	2719	175	122	90	1	2629	0.7265	0.8643
0.25	0.1	135	2719	163	134	82	1	2637	0.7299	0.8661
0.5	0.1	130	2724	163	129	87	1	2637	0.7299	0.8661
0.75	0.1	125	2729	163	124	92	1	2637	0.7299	0.8661
0.25	0.2	179	2787	51	157	90	22	2697	0.7392	0.8757
0.5	0.2	155	2811	51	147	100	8	2711	0.7392	0.8757
0.75	0.2	141	2825	51	137	110	4	2715	0.7392	0.8757
0.25	0.3	207	2802	8	173	90	34	2712	0.7485	0.8808
0.5	0.3	172	2837	8	160	103	12	2734	0.7485	0.8808
0.75	0.3	146	2863	8	142	121	4	2742	0.7485	0.8808
0.25	0.4	214	2803	0	174	90	40	2713	0.7464	0.8805
0.5	0.4	176	2841	0	161	103	15	2738	0.7464	0.8805
0.75	0.4	146	2871	0	142	122	4	2749	0.7464	0.8805
0.25	0.5	214	2803	0	174	90	40	2713	0.7464	0.8805
0.5	0.5	176	2841	0	161	103	15	2738	0.7464	0.8805
0.75	0.5	146	2871	0	142	122	4	2749	0.7464	0.8805
0.25	0.6	214	2803	0	174	90	40	2713	0.7464	0.8805
0.5	0.6	176	2841	0	161	103	15	2738	0.7464	0.8805
0.75	0.6	146	2871	0	142	122	4	2749	0.7464	0.8805

Table K.7: PhysioNet: Standard deviation results where 50 models are sampled. Std Threshold is an upper limit on standard deviation in predictions on the same data point, before the model expresses "i do not know" (IDK). IDK is the number of data points for which the model express "i do not know".

K.3 Disagreement Results (Cross-Tracks)

Positive Threshold	DL	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	58	5378	-	37	276	21	5102	0.3025	0.7858
0.1	0 ($\lfloor 0\% \rfloor$)	183	4246	1007	86	119	97	4127	0.3287	0.7792
0.2	0 ($\lfloor 0\% \rfloor$)	99	4918	419	60	179	39	4739	0.2770	0.7626
0.3	0 ($\lfloor 0\% \rfloor$)	69	5165	202	46	214	23	4951	0.2629	0.7617
0.4	0 ($\lfloor 0\% \rfloor$)	36	5269	131	24	237	12	5032	0.2260	0.7545
0.5	0 ($\lfloor 0\% \rfloor$)	7	5327	102	6	253	1	5074	0.1974	0.7486
0.6	0 ($\lfloor 0\% \rfloor$)	0	5351	85	0	262	0	5089	0	0
0.7	0 ($\lfloor 0\% \rfloor$)	0	5387	49	0	281	0	5106	0	0
0.8	0 ($\lfloor 0\% \rfloor$)	0	5419	17	0	304	0	5115	0	0
0.9	0 ($\lfloor 0\% \rfloor$)	0	5434	2	0	312	0	5122	0	0
0.1	1 ($\lfloor 10\% \rfloor$)	262	4543	631	107	139	155	4404	0.3292	0.7887
0.2	1 ($\lfloor 10\% \rfloor$)	123	5078	235	68	196	55	4882	0.2850	0.7693
0.3	1 ($\lfloor 10\% \rfloor$)	85	5254	97	54	228	31	5026	0.2781	0.7711
0.4	1 ($\lfloor 10\% \rfloor$)	54	5309	73	34	246	20	5063	0.2538	0.7663
0.5	1 ($\lfloor 10\% \rfloor$)	14	5345	77	10	257	4	5088	0.2046	0.7539
0.6	1 ($\lfloor 10\% \rfloor$)	2	5371	63	1	274	1	5097	0.2260	0.7602
0.7	1 ($\lfloor 10\% \rfloor$)	0	5411	25	0	295	0	5116	0	0
0.8	1 ($\lfloor 10\% \rfloor$)	0	5432	4	0	310	0	5122	0	0
0.9	1 ($\lfloor 10\% \rfloor$)	0	5436	0	0	313	0	5123	0	0
0.1	2 ($\lfloor 20\% \rfloor$)	335	4675	426	118	153	217	4522	0.3228	0.7914
0.2	2 ($\lfloor 20\% \rfloor$)	140	5154	142	71	207	69	4947	0.2842	0.7723
0.3	2 ($\lfloor 20\% \rfloor$)	96	5280	60	55	240	41	5040	0.2914	0.7782
0.4	2 ($\lfloor 20\% \rfloor$)	65	5330	41	42	251	23	5079	0.2734	0.7742
0.5	2 ($\lfloor 20\% \rfloor$)	27	5353	56	17	261	10	5092	0.2322	0.7622
0.6	2 ($\lfloor 20\% \rfloor$)	4	5390	42	2	286	2	5104	0.2554	0.7698
0.7	2 ($\lfloor 20\% \rfloor$)	1	5422	13	0	303	1	5119	0.2696	0.7792
0.8	2 ($\lfloor 20\% \rfloor$)	0	5435	1	0	313	0	5122	0	0
0.9	2 ($\lfloor 20\% \rfloor$)	0	5436	0	0	313	0	5123	0	0
0.1	3 ($\lfloor 30\% \rfloor$)	414	4795	227	126	163	288	4632	0.3117	0.7882
0.2	3 ($\lfloor 30\% \rfloor$)	155	5196	85	76	216	79	4980	0.2927	0.7782
0.3	3 ($\lfloor 30\% \rfloor$)	103	5298	35	58	245	45	5053	0.2974	0.7819
0.4	3 ($\lfloor 30\% \rfloor$)	75	5337	24	48	253	27	5084	0.2857	0.7790

0.5	3 (⌊30%⌋)	41	5360	35	25	265	16	5095	0.2578	0.7708
0.6	3 (⌊30%⌋)	10	5406	20	8	296	2	5110	0.2985	0.7811
0.7	3 (⌊30%⌋)	1	5425	10	0	304	1	5121	0.2679	0.7796
0.8	3 (⌊30%⌋)	0	5435	1	0	313	0	5122	0	0
0.9	3 (⌊30%⌋)	0	5436	0	0	313	0	5123	0	0
0.1	4 (⌊40%⌋)	476	4885	75	136	168	340	4717	0.3052	0.7861
0.2	4 (⌊40%⌋)	174	5228	34	79	222	95	5006	0.2933	0.7802
0.3	4 (⌊40%⌋)	109	5309	18	61	248	48	5061	0.3024	0.7847
0.4	4 (⌊40%⌋)	82	5346	8	52	257	30	5089	0.2972	0.7836
0.5	4 (⌊40%⌋)	59	5365	12	34	270	25	5095	0.2847	0.7800
0.6	4 (⌊40%⌋)	13	5415	8	10	301	3	5114	0.3090	0.7853
0.7	4 (⌊40%⌋)	3	5432	1	2	311	1	5121	0.3090	0.7859
0.8	4 (⌊40%⌋)	0	5436	0	0	313	0	5123	0	0
0.9	4 (⌊40%⌋)	0	5436	0	0	313	0	5123	0	0

Table K.8: Cross-Tracks: Disagreement results where 10 models are sampled. DL is the disagreement limit specifying how many predictions on the same data point can disagree before the model expresses "i do not know" (IDK). IDK is the number of data points for which the sampled models disagree.

Positive Threshold	DL	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	67	5369	-	44	269	23	5100	0.3092	0.7879
0.1	0 (⌊0%⌋)	113	3505	1818	62	85	51	3420	0.3388	0.7741
0.2	0 (⌊0%⌋)	66	4562	808	45	150	21	4412	0.2750	0.7515
0.3	0 (⌊0%⌋)	34	4957	445	25	184	9	4773	0.2105	0.7319
0.4	0 (⌊0%⌋)	3	5171	262	3	217	0	4954	0.1536	0.7244
0.5	0 (⌊0%⌋)	0	5269	167	0	235	0	5034	0	0
0.6	0 (⌊0%⌋)	0	5315	121	0	248	0	5067	0	0
0.7	0 (⌊0%⌋)	0	5351	85	0	260	0	5091	0	0
0.8	0 (⌊0%⌋)	0	5395	41	0	290	0	5105	0	0
0.9	0 (⌊0%⌋)	0	5428	8	0	310	0	5118	0	0
0.1	10 (⌊10%⌋)	217	4417	802	96	129	121	4288	0.3416	0.7875
0.2	10 (⌊10%⌋)	107	5025	304	63	189	44	4836	0.2852	0.7672
0.3	10 (⌊10%⌋)	76	5237	123	50	224	26	5013	0.2765	0.7690
0.4	10 (⌊10%⌋)	53	5304	79	35	244	18	5060	0.2603	0.7684
0.5	10 (⌊10%⌋)	2	5336	98	1	251	1	5085	0.1524	0.7427
0.6	10 (⌊10%⌋)	0	5354	82	0	260	0	5094	0	0
0.7	10 (⌊10%⌋)	0	5402	34	0	293	0	5109	0	0
0.8	10 (⌊10%⌋)	0	5434	2	0	312	0	5122	0	0
0.9	10 (⌊10%⌋)	0	5436	0	0	313	0	5123	0	0
0.1	20 (⌊20%⌋)	291	4620	525	111	147	180	4473	0.3333	0.7923

0.2	20 (⌊20%⌋)	127	5125	184	67	201	60	4924	0.2858	0.7701
0.3	20 (⌊20%⌋)	87	5273	76	56	232	31	5041	0.2897	0.7763
0.4	20 (⌊20%⌋)	66	5320	50	44	248	22	5072	0.2821	0.7766
0.5	20 (⌊20%⌋)	21	5349	66	14	256	7	5093	0.2125	0.7579
0.6	20 (⌊20%⌋)	0	5377	59	0	275	0	5102	0	0
0.7	20 (⌊20%⌋)	0	5421	15	0	303	0	5118	0	0
0.8	20 (⌊20%⌋)	0	5436	0	0	313	0	5123	0	0
0.9	20 (⌊20%⌋)	0	5436	0	0	313	0	5123	0	0
0.1	30 (⌊30%⌋)	364	4742	330	120	158	244	4584	0.3231	0.7911
0.2	30 (⌊30%⌋)	148	5174	114	74	210	74	4964	0.2952	0.7769
0.3	30 (⌊30%⌋)	97	5293	46	60	242	37	5051	0.3086	0.7845
0.4	30 (⌊30%⌋)	72	5333	31	48	250	24	5083	0.2873	0.7795
0.5	30 (⌊30%⌋)	38	5355	43	27	262	11	5093	0.2686	0.7731
0.6	30 (⌊30%⌋)	0	5396	40	0	288	0	5108	0	0
0.7	30 (⌊30%⌋)	0	5433	3	0	311	0	5122	0	0
0.8	30 (⌊30%⌋)	0	5436	0	0	313	0	5123	0	0
0.9	30 (⌊30%⌋)	0	5436	0	0	313	0	5123	0	0
0.1	40 (⌊40%⌋)	446	4844	146	132	166	314	4678	0.3154	0.7898
0.2	40 (⌊40%⌋)	167	5219	50	82	219	85	5000	0.3047	0.7840
0.3	40 (⌊40%⌋)	107	5312	17	63	246	44	5066	0.3090	0.7867
0.4	40 (⌊40%⌋)	83	5343	10	53	254	30	5089	0.2993	0.7843
0.5	40 (⌊40%⌋)	54	5362	20	37	265	17	5097	0.2916	0.7815
0.6	40 (⌊40%⌋)	5	5409	22	3	297	2	5112	0.2921	0.7801
0.7	40 (⌊40%⌋)	0	5435	1	0	313	0	5122	0	0
0.8	40 (⌊40%⌋)	0	5436	0	0	313	0	5123	0	0
0.9	40 (⌊40%⌋)	0	5436	0	0	313	0	5123	0	0

Table K.9: Cross-Tracks: Disagreement results where 100 models are sampled. DL is the disagreement limit specifying how many predictions on the same data point can disagree before the model expresses "i do not know" (IDK). IDK is the number of data points for which the sampled models disagree.

K.4 Standard Deviation Results (Cross-Tracks)

Positive Threshold	Std Threshold	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	60	5376	-	42	271	18	5105	0.3041	0.7850
0.25	0.01	0	2772	2664	0	64	0	2708	0	0
0.5	0.01	0	2772	2664	0	64	0	2708	0	0
0.75	0.01	0	2772	2664	0	64	0	2708	0	0
0.25	0.02	0	4188	1248	0	124	0	4064	0	0
0.5	0.02	0	4188	1248	0	124	0	4064	0	0

0.75	0.02	0	4188	1248	0	124	0	4064	0	0
0.25	0.03	0	4622	814	0	157	0	4465	0	0
0.5	0.03	0	4622	814	0	157	0	4465	0	0
0.75	0.03	0	4622	814	0	157	0	4465	0	0
0.25	0.04	1	4827	608	1	174	0	4653	0.0909	0.6936
0.5	0.04	1	4827	608	1	174	0	4653	0.0909	0.6936
0.75	0.04	0	4828	608	0	175	0	4653	0	0
0.25	0.05	5	4995	436	3	190	2	4805	0.1126	0.7024
0.5	0.05	3	4997	436	3	190	0	4807	0.1126	0.7024
0.75	0.05	0	5000	436	0	193	0	4807	0	0
0.25	0.06	14	5116	306	8	213	6	4903	0.1649	0.7272
0.5	0.06	8	5122	306	7	214	1	4908	0.1649	0.7272
0.75	0.06	0	5130	306	0	221	0	4909	0	0
0.25	0.07	30	5182	224	18	220	12	4962	0.1939	0.7387
0.5	0.07	18	5194	224	14	224	4	4970	0.1939	0.7387
0.75	0.07	0	5212	224	0	238	0	4974	0	0
0.25	0.08	49	5235	152	27	227	22	5008	0.2220	0.7483
0.5	0.08	24	5260	152	19	235	5	5025	0.2220	0.7483
0.75	0.08	0	5284	152	0	254	0	5030	0	0
0.25	0.09	71	5262	103	35	230	36	5032	0.2381	0.7543
0.5	0.09	29	5304	103	23	242	6	5062	0.2381	0.7543
0.75	0.09	0	5333	103	0	265	0	5068	0	0
0.25	0.1	87	5273	76	43	233	44	5040	0.2591	0.7619
0.5	0.1	36	5324	76	28	248	8	5076	0.2591	0.7619
0.75	0.1	0	5360	76	0	276	0	5084	0	0
0.25	0.2	148	5287	1	75	238	73	5049	0.3047	0.7851
0.5	0.2	60	5375	1	42	271	18	5104	0.3047	0.7851
0.75	0.2	0	5435	1	0	313	0	5122	0	0
0.25	0.3	149	5287	0	75	238	74	5049	0.3041	0.7850
0.5	0.3	60	5376	0	42	271	18	5105	0.3041	0.7850
0.75	0.3	0	5436	0	0	313	0	5123	0	0
0.25	0.4	149	5287	0	75	238	74	5049	0.3041	0.7850
0.5	0.4	60	5376	0	42	271	18	5105	0.3041	0.7850
0.75	0.4	0	5436	0	0	313	0	5123	0	0
0.25	0.5	149	5287	0	75	238	74	5049	0.3041	0.7850
0.5	0.5	60	5376	0	42	271	18	5105	0.3041	0.7850
0.75	0.5	0	5436	0	0	313	0	5123	0	0
0.25	0.6	149	5287	0	75	238	74	5049	0.3041	0.7850

0.5	0.6	60	5376	0	42	271	18	5105	0.3041	0.7850
0.75	0.6	0	5436	0	0	313	0	5123	0	0

Table K.10: Cross-Tracks: Standard deviation results where 10 models are sampled. Std Threshold is a upper limit on standard deviation in predictions on the same data point, before the model expresses "i do not know" (IDK). IDK is the number of data points for which the model express "i do not know".

Positive Threshold	Std Threshold	Pos	Neg	IDK	TP	FN	FP	TN	AUPRC	AUROC
0.5 (Baseline)	-	62	5374	-	41	272	21	5102	0.3100	0.7880
0.25	0.01	0	2319	3117	0	58	0	2261	0	0
0.5	0.01	0	2319	3117	0	58	0	2261	0	0
0.75	0.01	0	2319	3117	0	58	0	2261	0	0
0.25	0.02	0	4021	1415	0	105	0	3916	0	0
0.5	0.02	0	4021	1415	0	105	0	3916	0	0
0.75	0.02	0	4021	1415	0	105	0	3916	0	0
0.25	0.03	0	4497	939	0	141	0	4356	0	0
0.5	0.03	0	4497	939	0	141	0	4356	0	0
0.75	0.03	0	4497	939	0	141	0	4356	0	0
0.25	0.04	0	4736	700	0	162	0	4574	0	0
0.5	0.04	0	4736	700	0	162	0	4574	0	0
0.75	0.04	0	4736	700	0	162	0	4574	0	0
0.25	0.05	0	4923	513	0	177	0	4746	0	0
0.5	0.05	0	4923	513	0	177	0	4746	0	0
0.75	0.05	0	4923	513	0	177	0	4746	0	0
0.25	0.06	0	5050	386	0	193	0	4857	0	0
0.5	0.06	0	5050	386	0	193	0	4857	0	0
0.75	0.06	0	5050	386	0	193	0	4857	0	0
0.25	0.07	3	5157	276	2	212	1	4945	0.1427	0.7185
0.5	0.07	3	5157	276	2	212	1	4945	0.1427	0.7185
0.75	0.07	0	5160	276	0	214	0	4946	0	0
0.25	0.08	15	5223	198	10	228	5	4995	0.1876	0.7397
0.5	0.08	12	5226	198	9	229	3	4997	0.1876	0.7397
0.75	0.08	0	5238	198	0	238	0	5000	0	0
0.25	0.09	29	5264	143	19	233	10	5031	0.2165	0.7489
0.5	0.09	18	5275	143	14	238	4	5037	0.2165	0.7489
0.75	0.09	0	5293	143	0	252	0	5041	0	0
0.25	0.1	56	5282	98	29	236	27	5046	0.2387	0.7572
0.5	0.1	24	5314	98	19	246	5	5068	0.2387	0.7572
0.75	0.1	0	5338	98	0	265	0	5073	0	0

0.25	0.2	147	5289	0	74	239	73	5050	0.3100	0.7880
0.5	0.2	62	5374	0	41	272	21	5102	0.3100	0.7880
0.75	0.2	0	5436	0	0	313	0	5123	0	0
0.25	0.3	147	5289	0	74	239	73	5050	0.3100	0.7880
0.5	0.3	62	5374	0	41	272	21	5102	0.3100	0.7880
0.75	0.3	0	5436	0	0	313	0	5123	0	0
0.25	0.4	147	5289	0	74	239	73	5050	0.3100	0.7880
0.5	0.4	62	5374	0	41	272	21	5102	0.3100	0.7880
0.75	0.4	0	5436	0	0	313	0	5123	0	0
0.25	0.5	147	5289	0	74	239	73	5050	0.3100	0.7880
0.5	0.5	62	5374	0	41	272	21	5102	0.3100	0.7880
0.75	0.5	0	5436	0	0	313	0	5123	0	0
0.25	0.6	147	5289	0	74	239	73	5050	0.3100	0.7880
0.5	0.6	62	5374	0	41	272	21	5102	0.3100	0.7880
0.75	0.6	0	5436	0	0	313	0	5123	0	0

Table K.11: Cross-Tracks: Standard deviation results where 100 models are sampled. Std Threshold is a upper limit on standard deviation in predictions on the same data point, before the model expresses "i do not know" (IDK). IDK is the number of data points for which the model express "i do not know".

Appendix L

LRP Relevance Diagrams

The most relevant results can be found in this chapter, and the full set of results can be found in the Supplementary Appendix.

L.1 PhysioNet

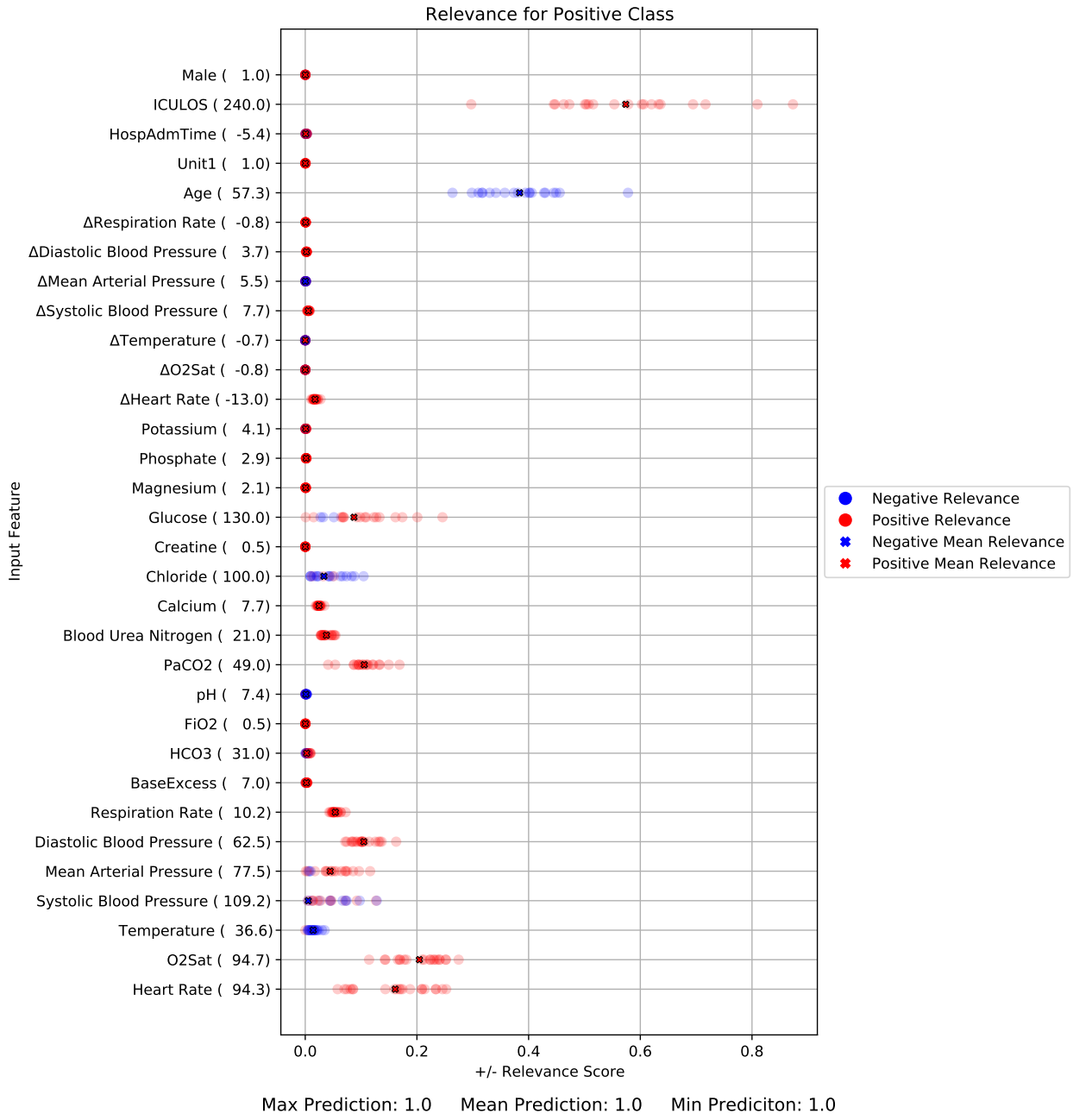


Figure L.1: ΔDP-A POS8True.

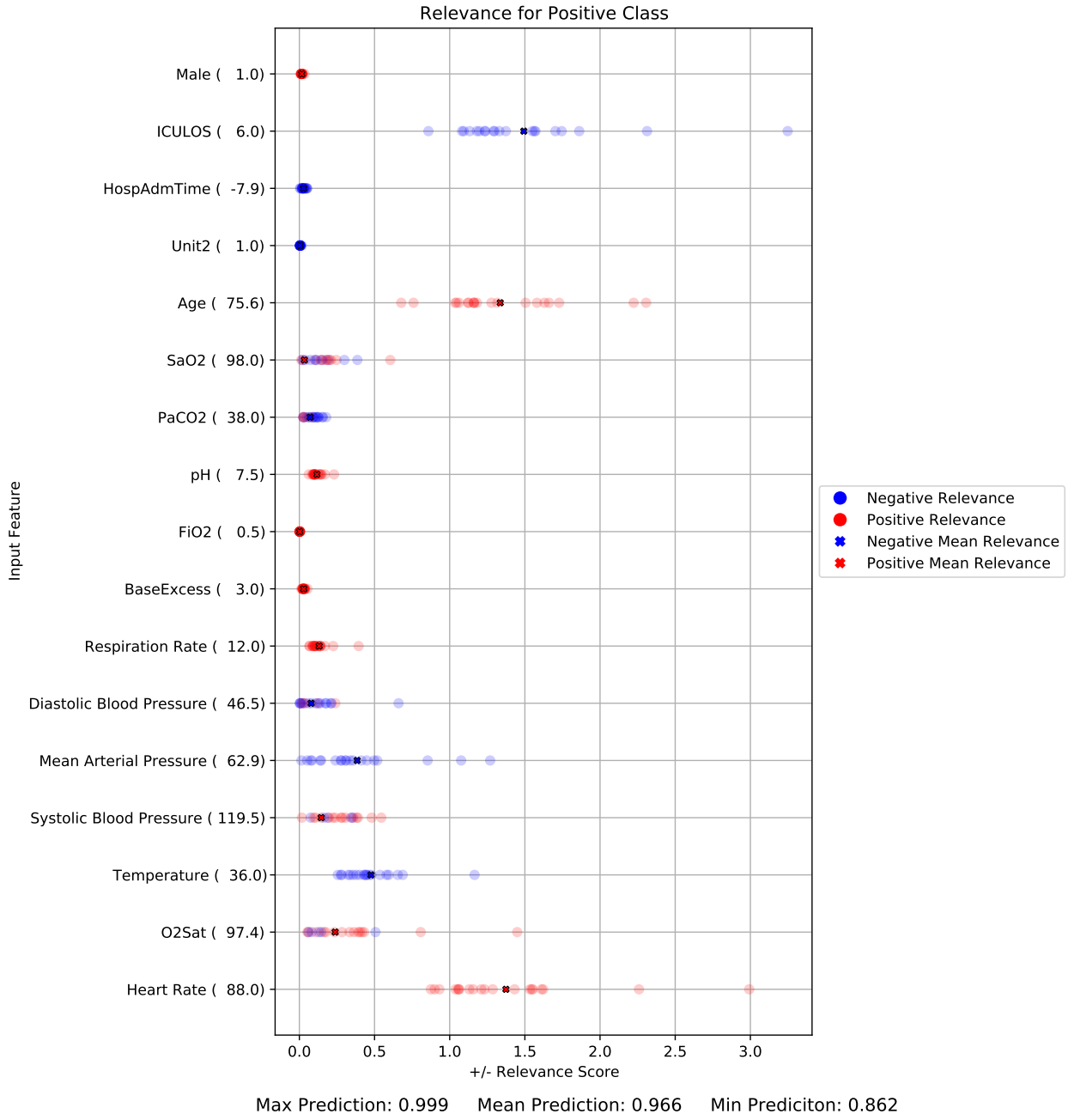


Figure L.2: ΔDP-A POS3True.

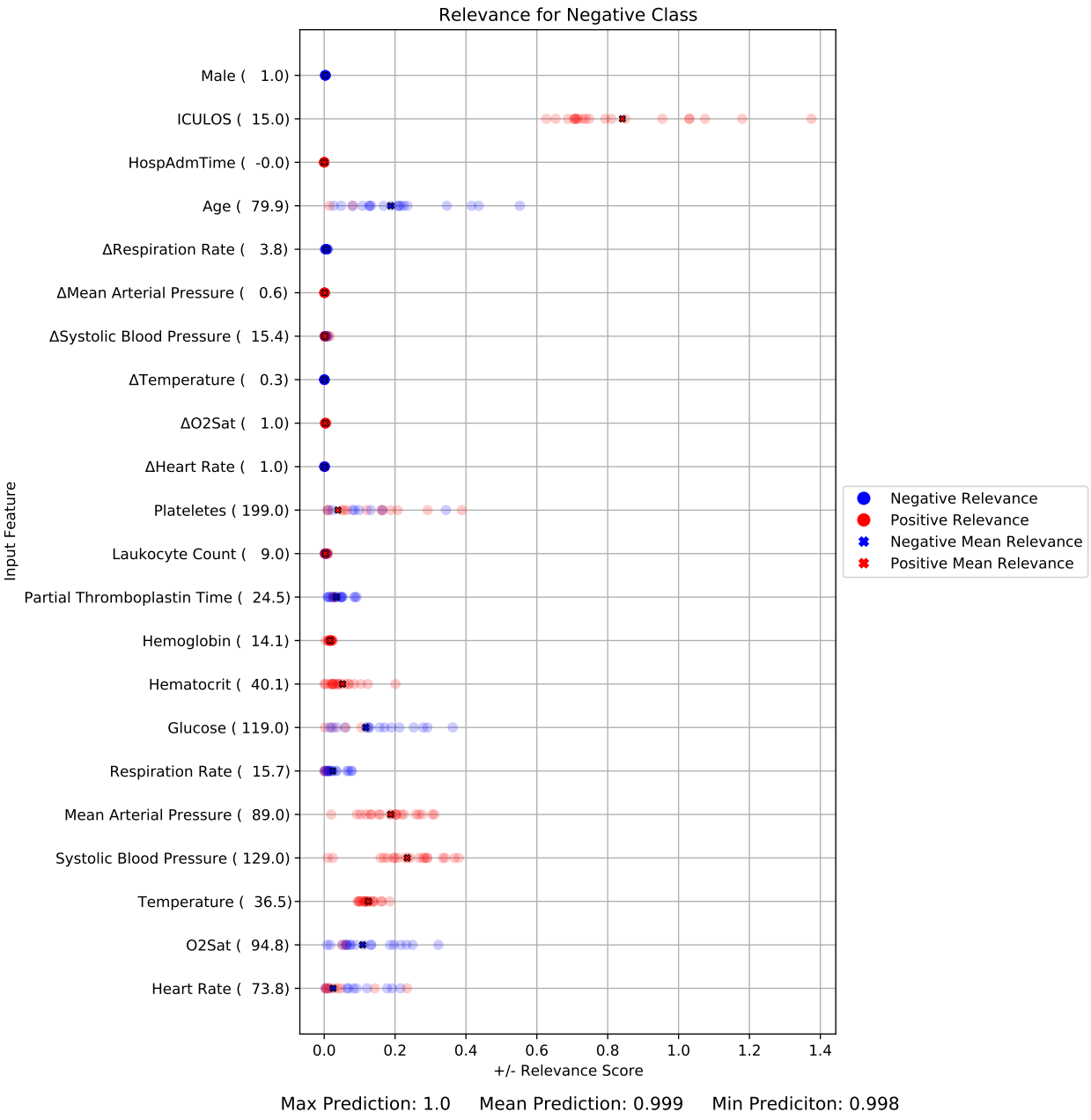


Figure L.3: ΔDP-A NEG2False.

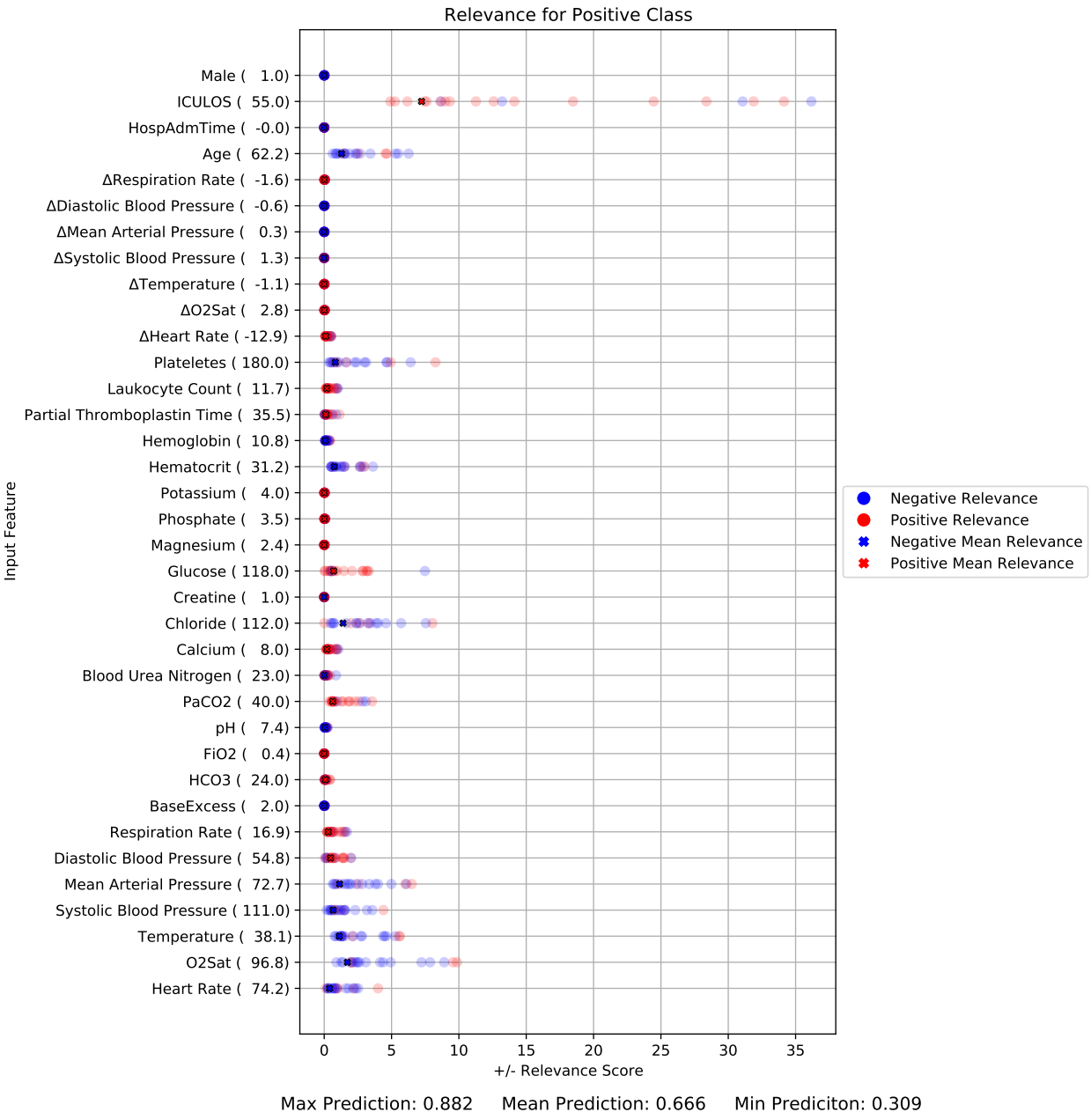


Figure L.4: ΔDP-A POS9True.

L.2 Cross-Tracks

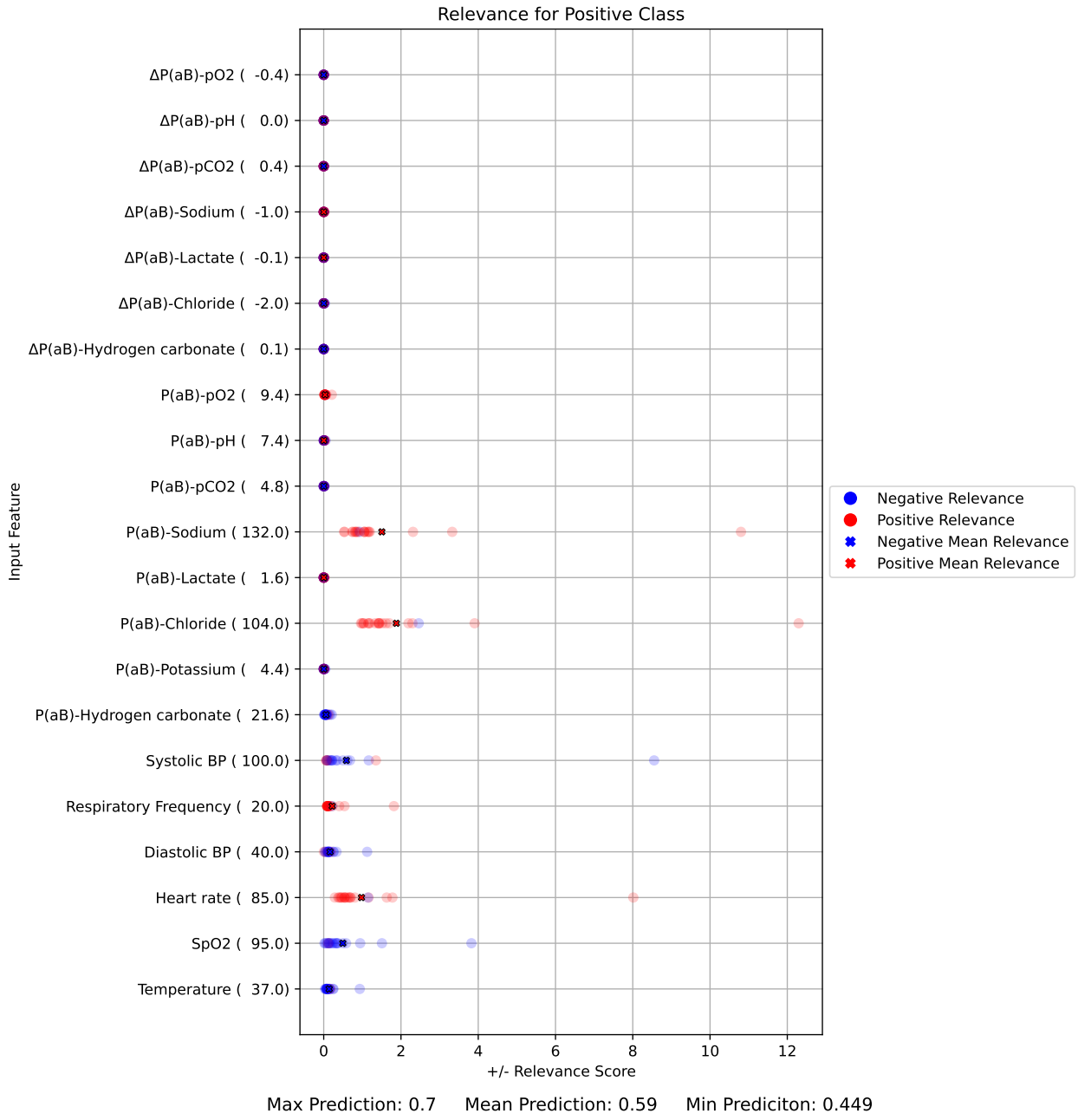


Figure L.5: ΔDCT POS11True.

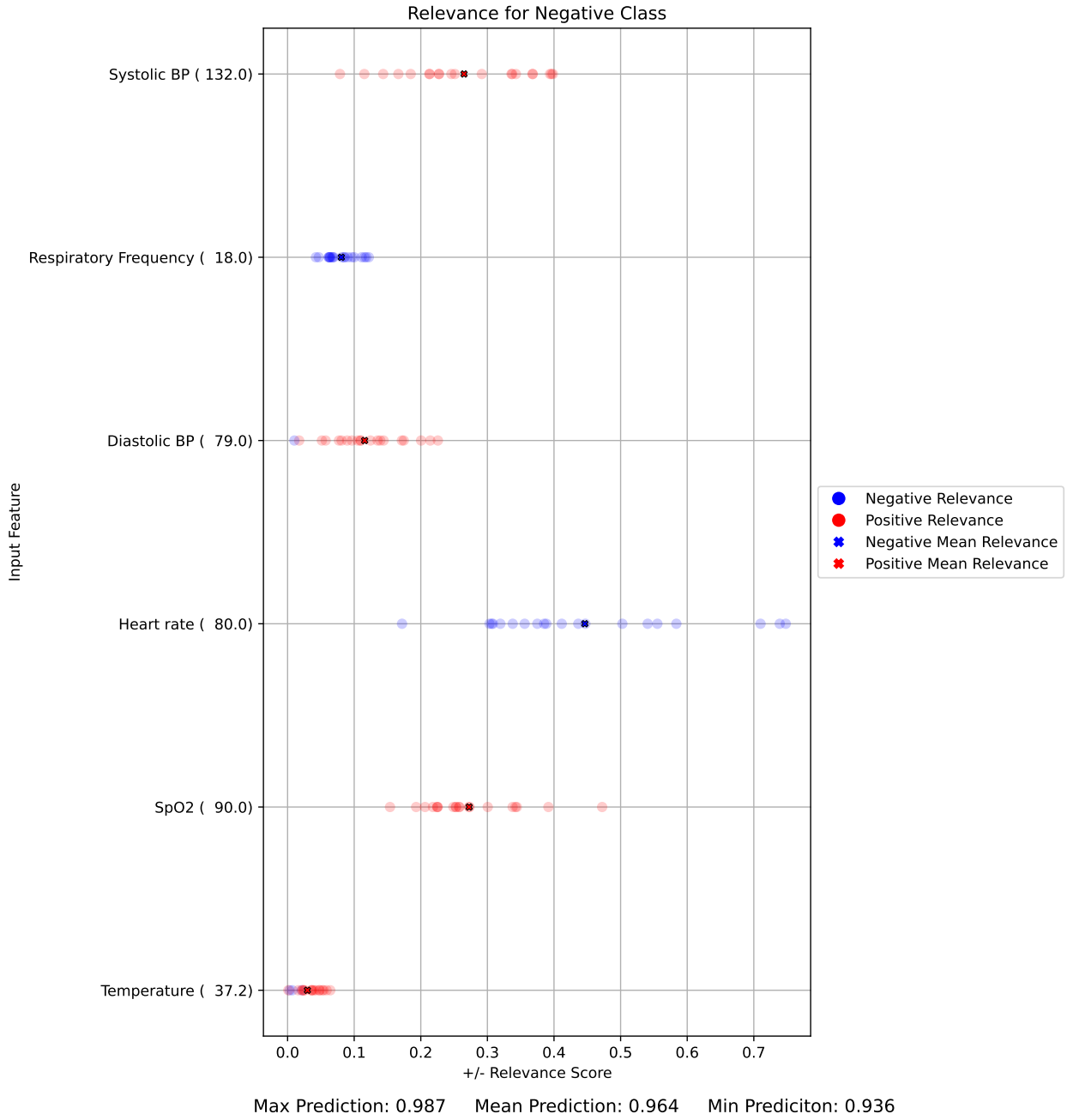


Figure L.6: ΔDCT NEG3False.

Appendix M

Weight Uncertainty in First Layer

M.1 Uncertainty In Fully a Bayesian Network

This section covers the results of inspecting the mean and standard deviation of the probability distributions for weights in the first layer, in a fully Bayesian NN.

M.1.1 Mean (ΔDP-A)

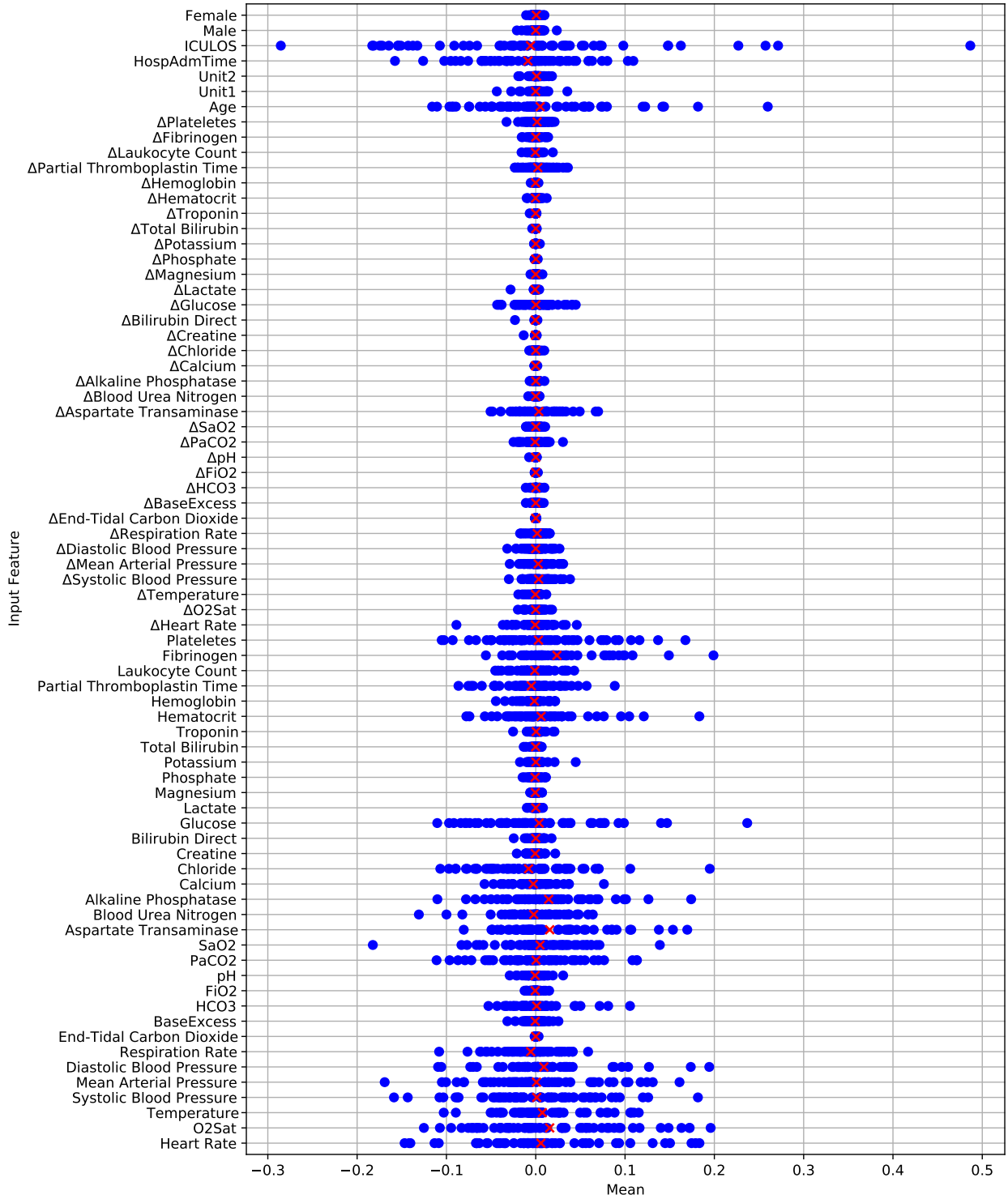


Figure M.1

M.1.2 Standard Deviation (ΔDP-A)

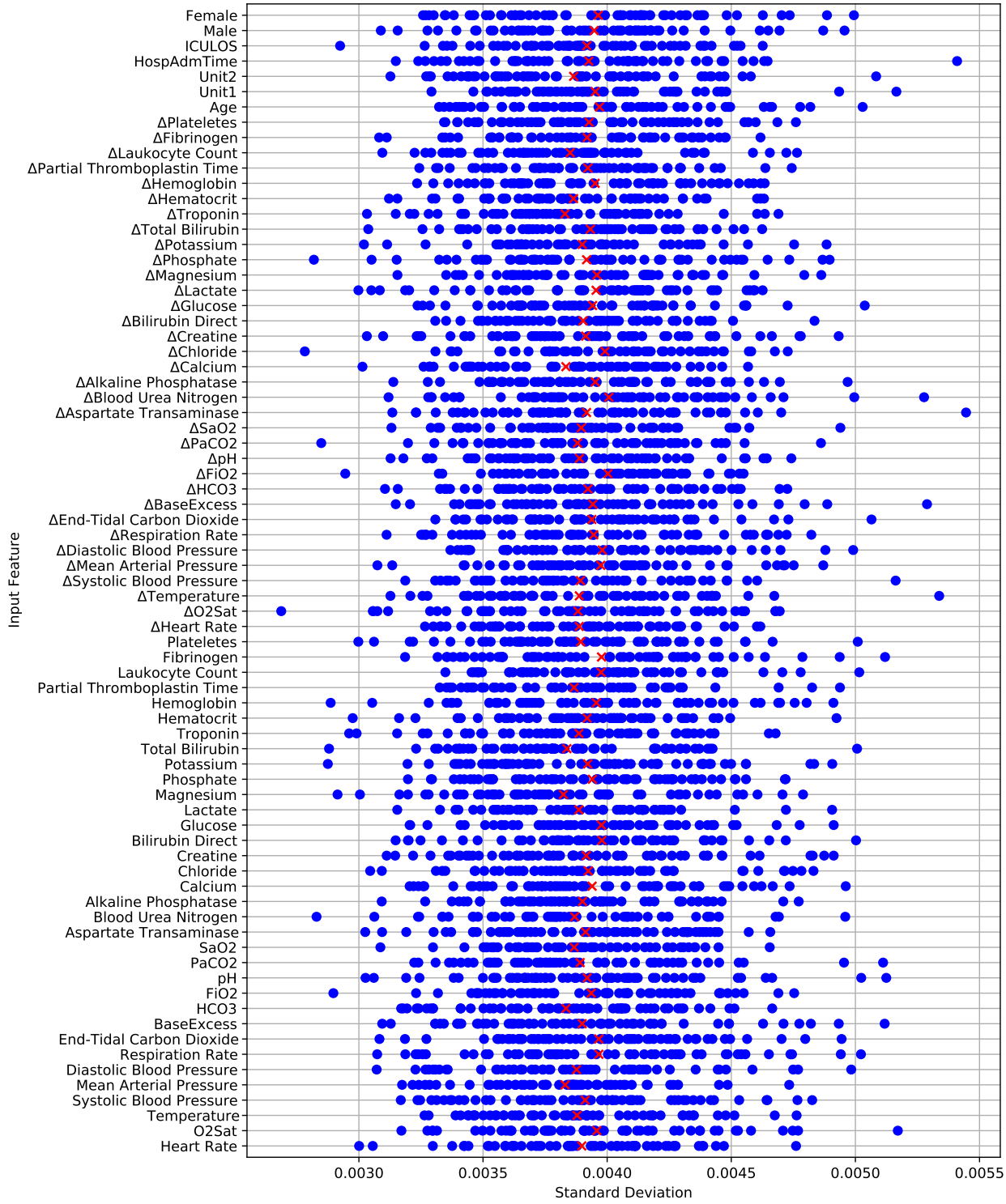


Figure M.2

M.1.3 Mean (ΔDCT)

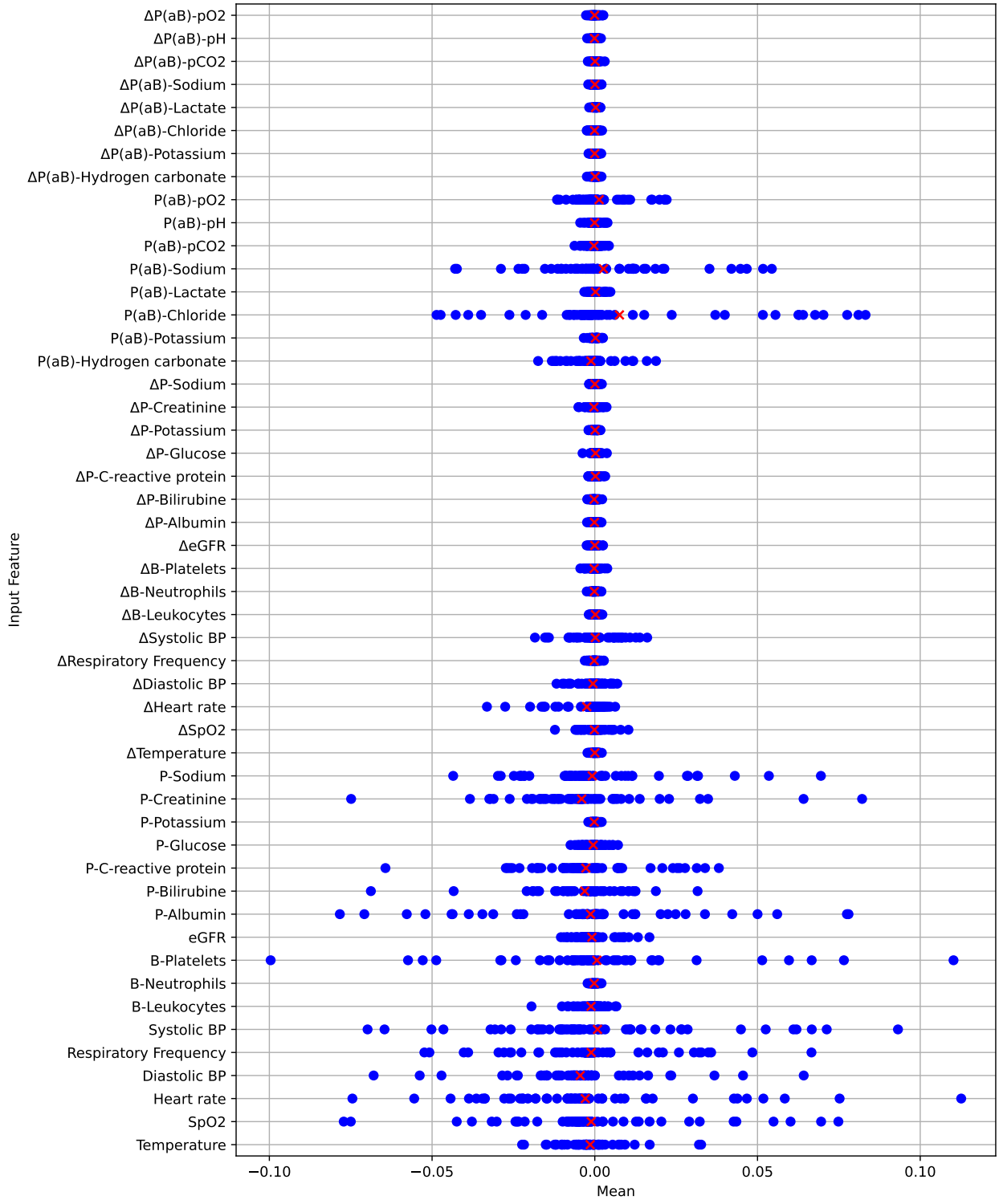


Figure M.3

M.1.4 Standard Deviation (ΔDCT)

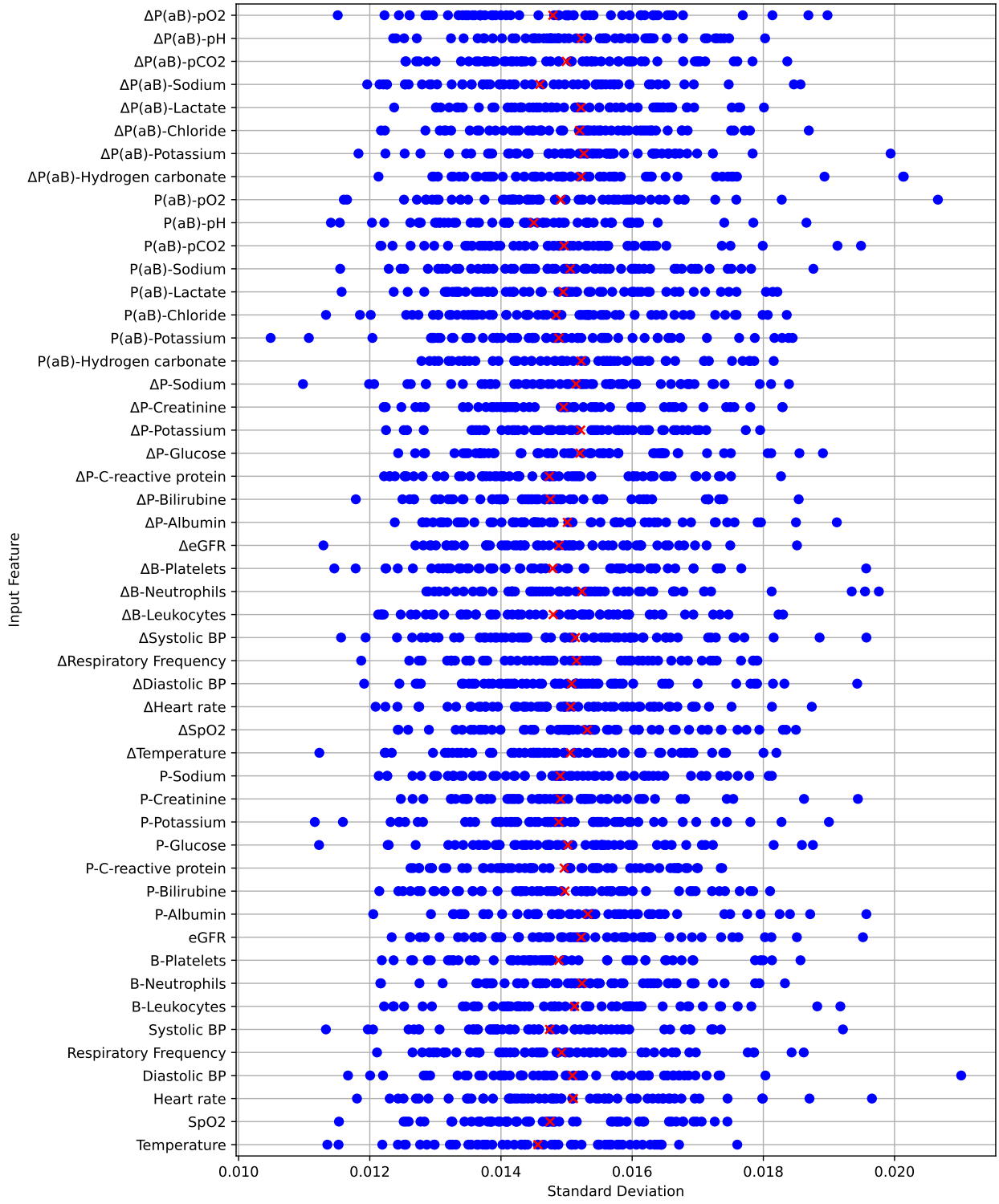


Figure M.4

M.2 Limiting Uncertainty To First Layer

This section covers the results of inspecting the mean and standard deviation of the probability distributions for weights in the first layer, in a BNN where only the first layer is stochastic.

M.2.1 Mean On ΔDP-A

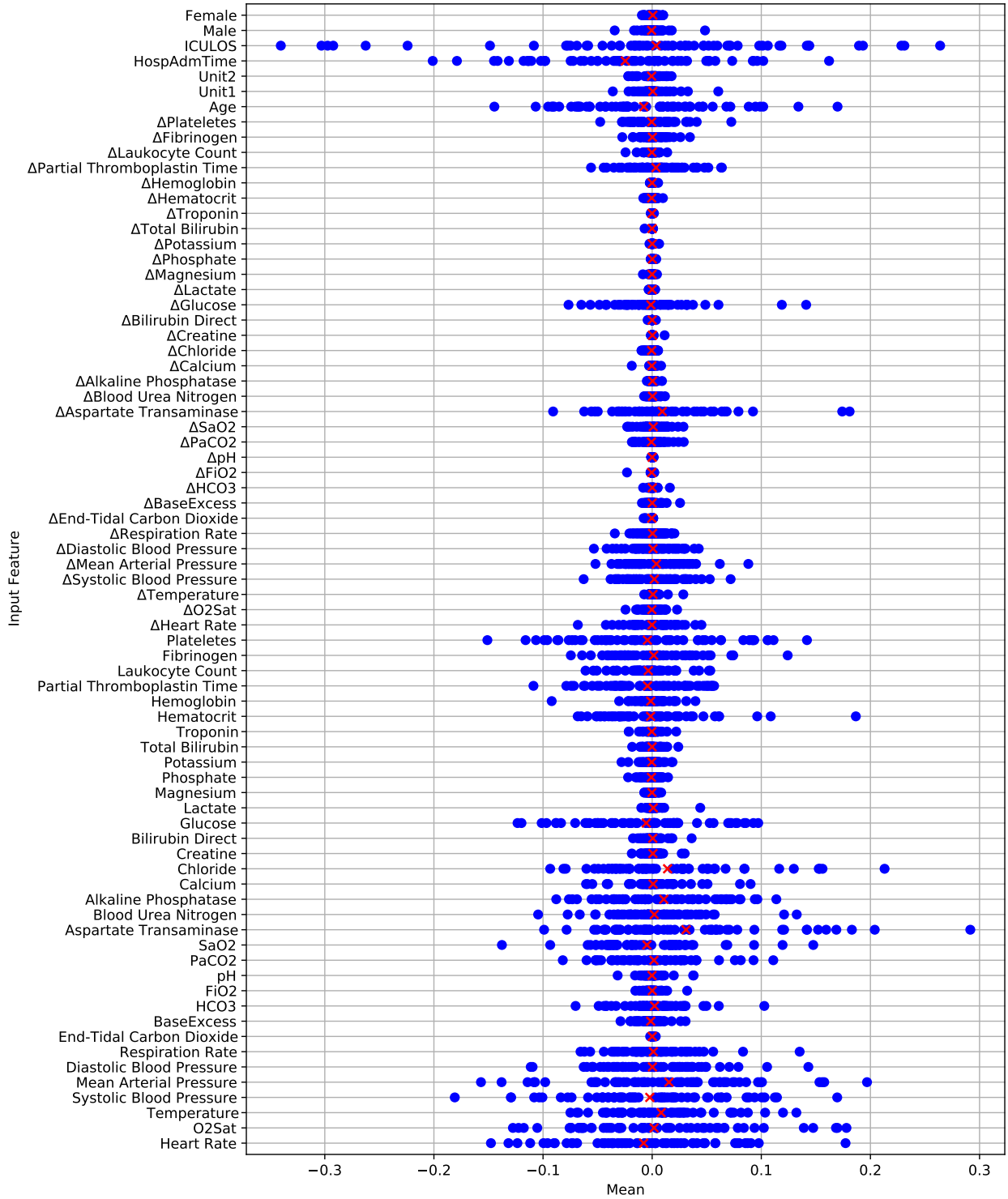


Figure M.5

M.2.2 Standard Deviation On ΔDP-A

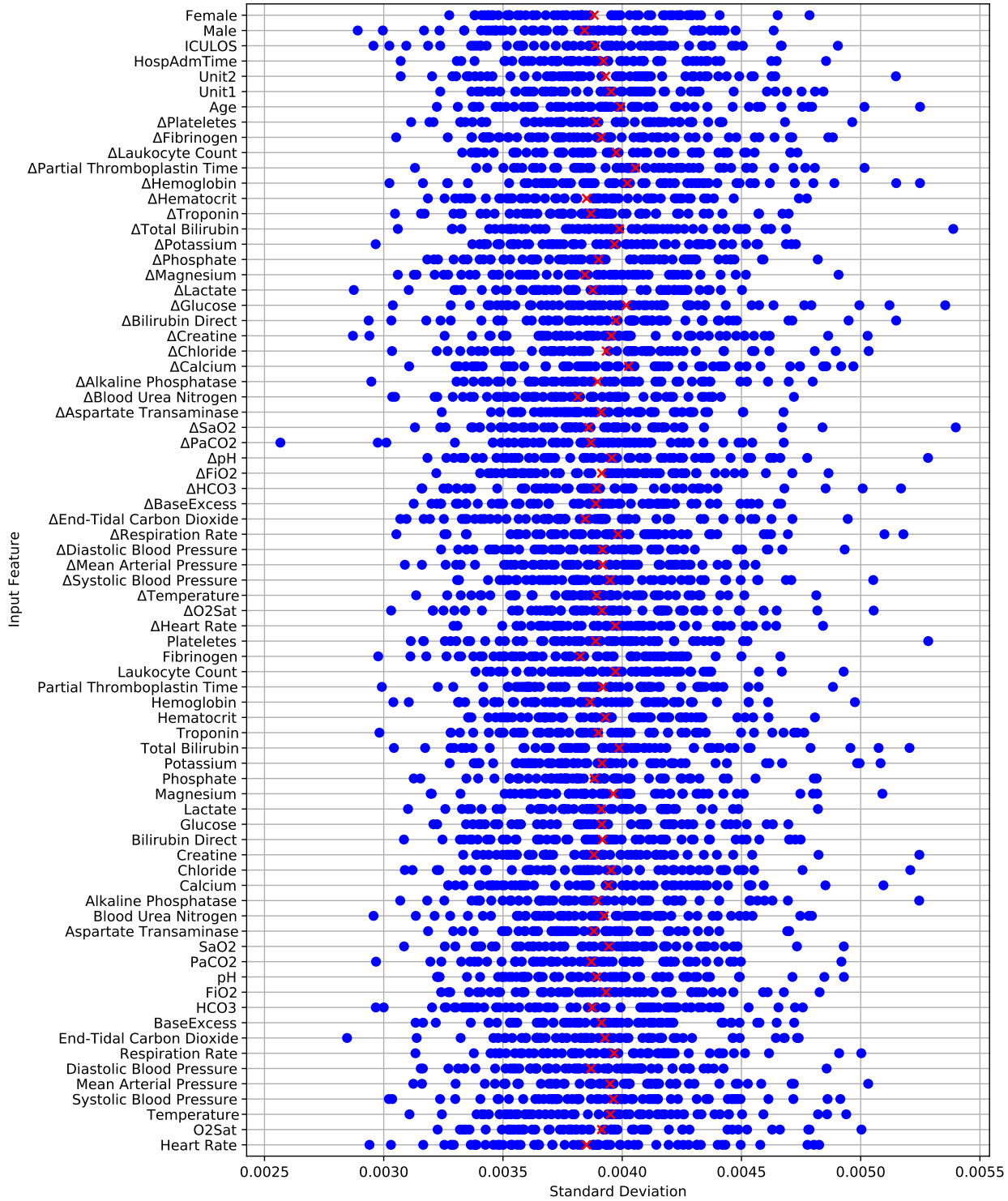


Figure M.6

M.2.3 Mean On ΔDCT

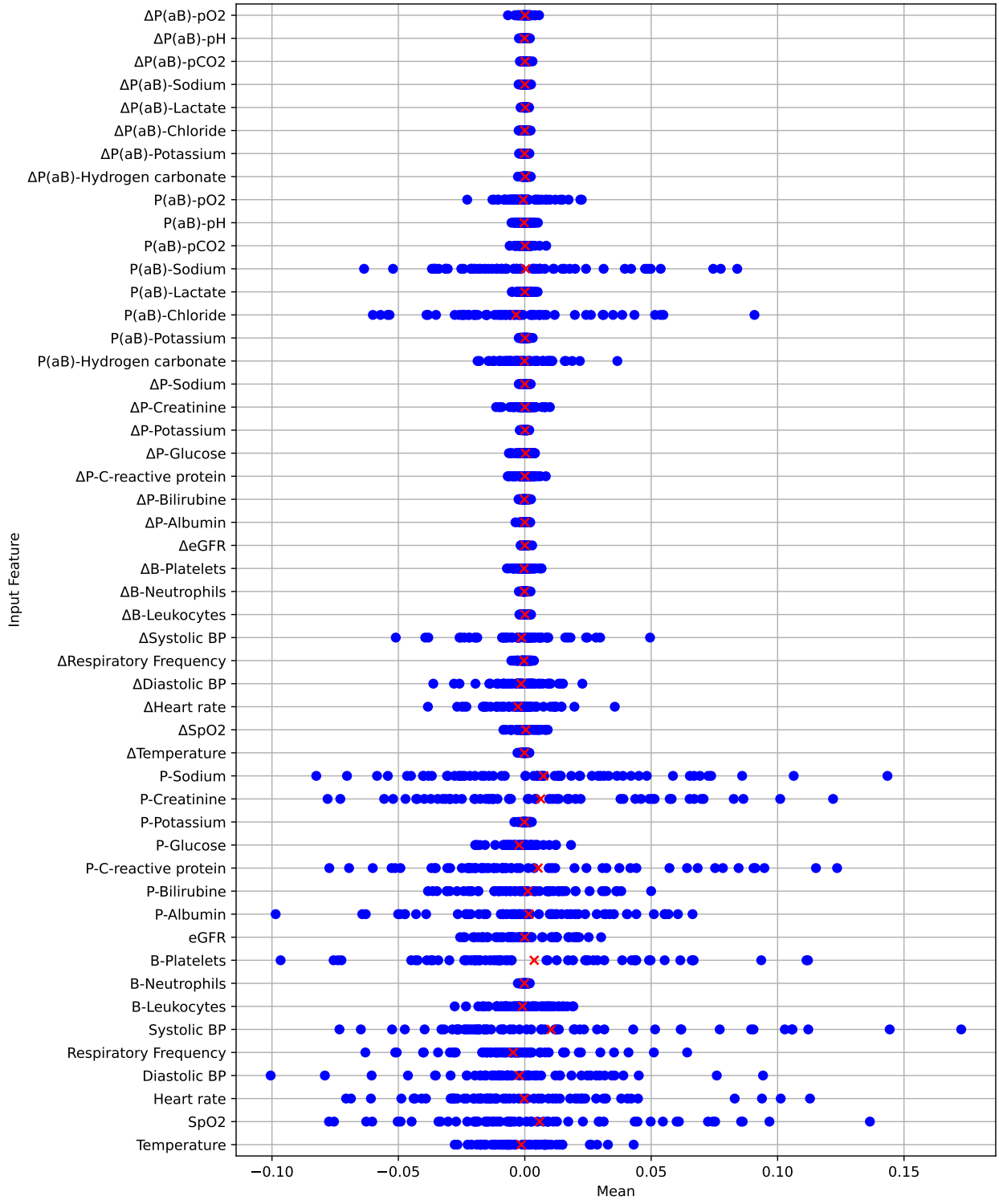


Figure M.7

M.2.4 Standard Deviation On ΔDCT

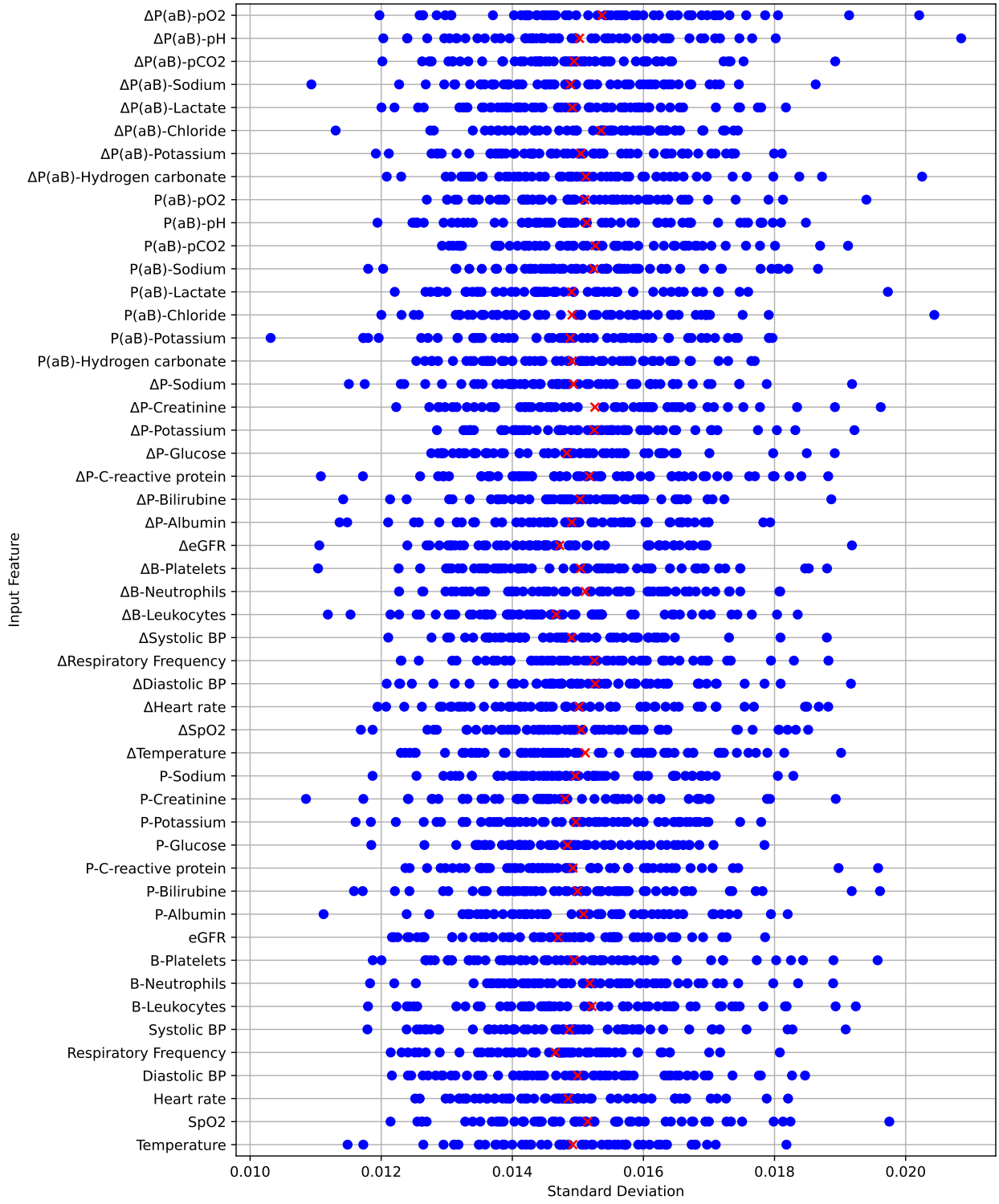


Figure M.8

Appendix N

Evaluating LRP

The naming scheme for captions in the following figures is as follows: Figures starting with Neg is for a data point with a sepsis negative label, and Pos is for a sepsis positive label. The following integer is an index for the data point in the test set. Following is a parenthesis, containing either "Original", if neither feature flipping nor feature exclusion is applied, "FF" if feature flipping is applied for that feature, or "0" if the feature is excluded. Lastly, False means that relevance is propagated from the output for the negative class, and True for the positive class.

N.1 Originals (ΔDP-A)

N.1.1 Neg0(Original)False

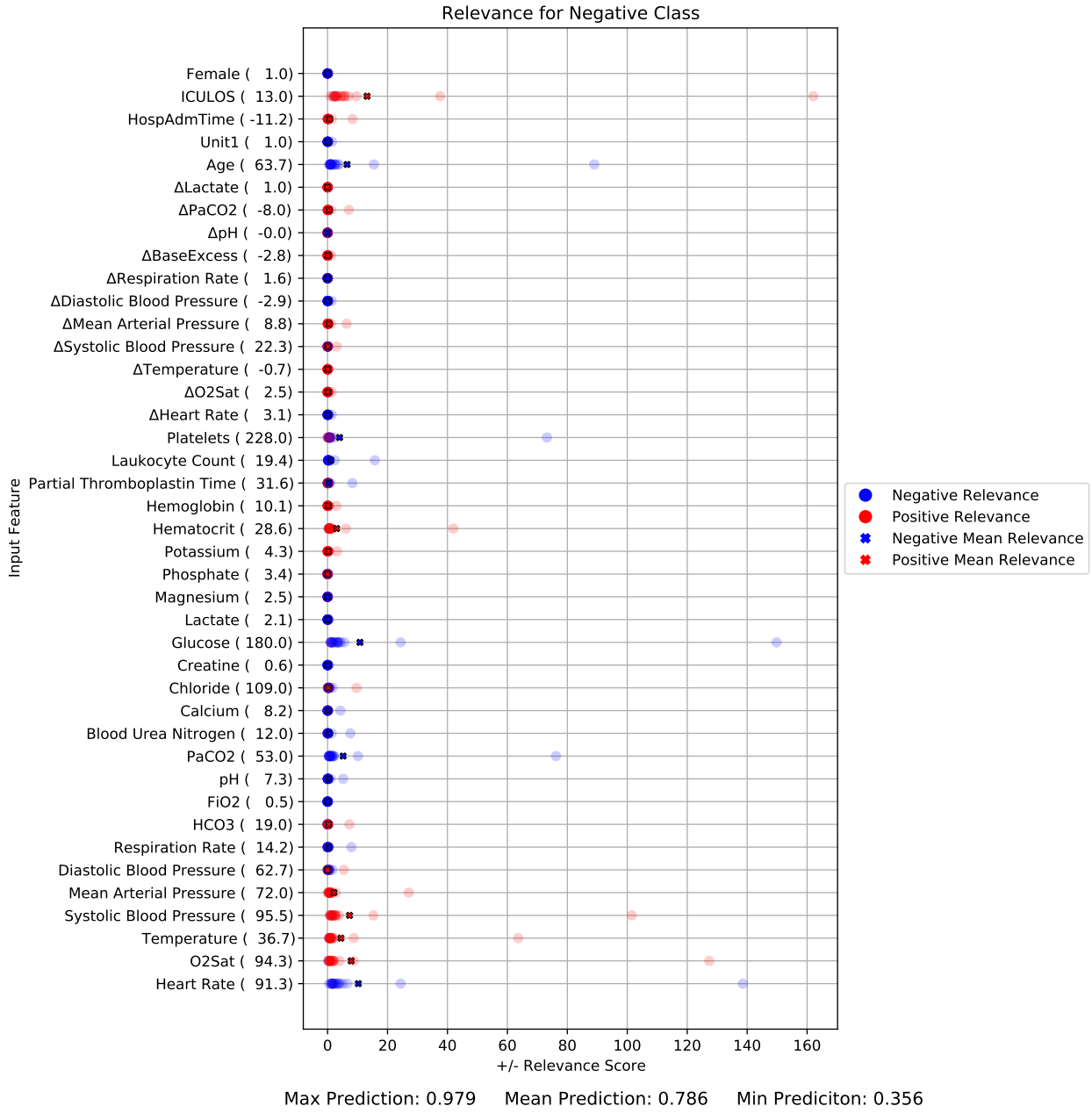


Figure N.1: Neg0(Original)False

N.1.2 POS8(Original)True

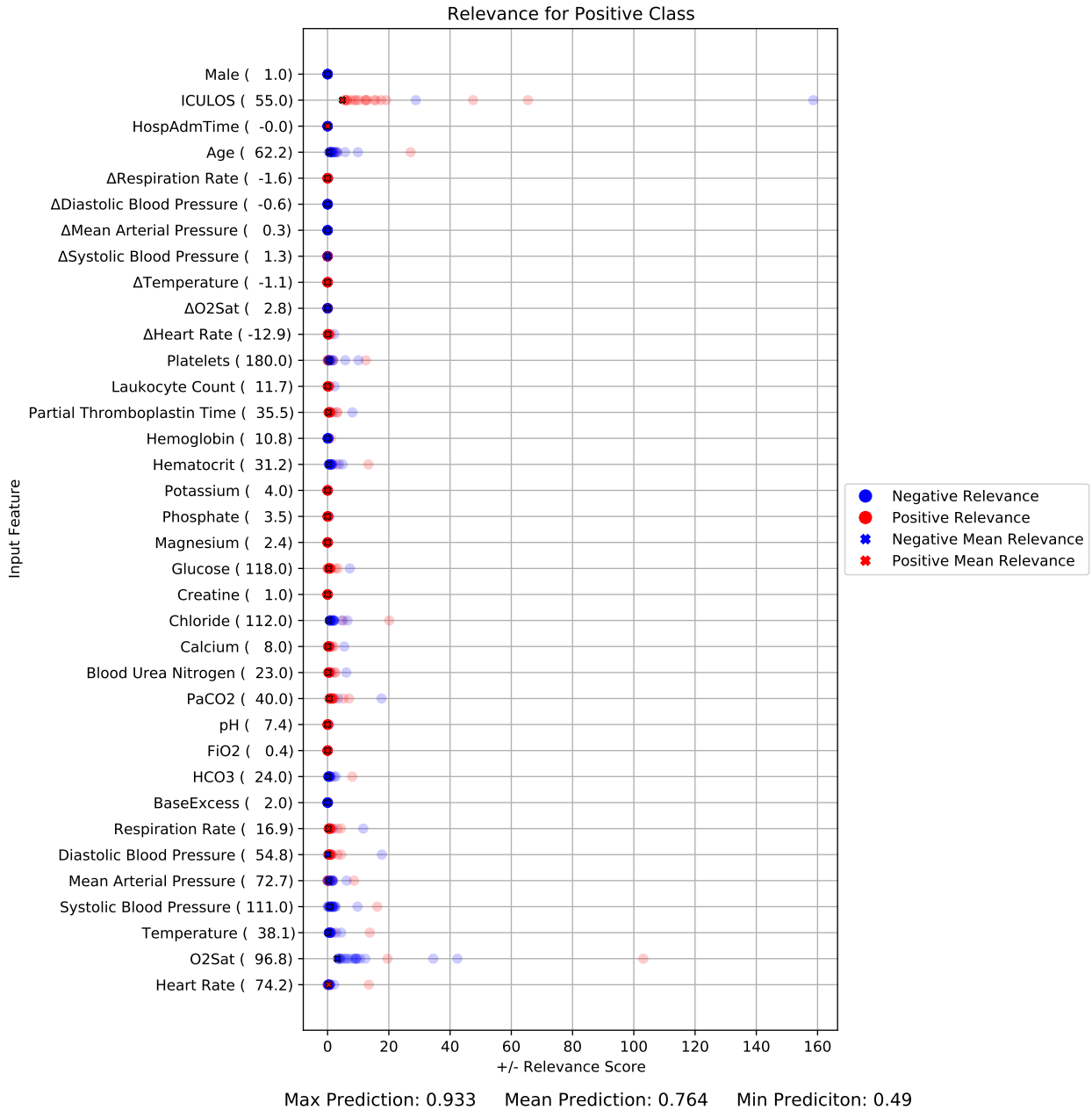


Figure N.2: POS8(Original)True

N.2 Originals (ΔDCT)

N.2.1 Neg4(Original)False

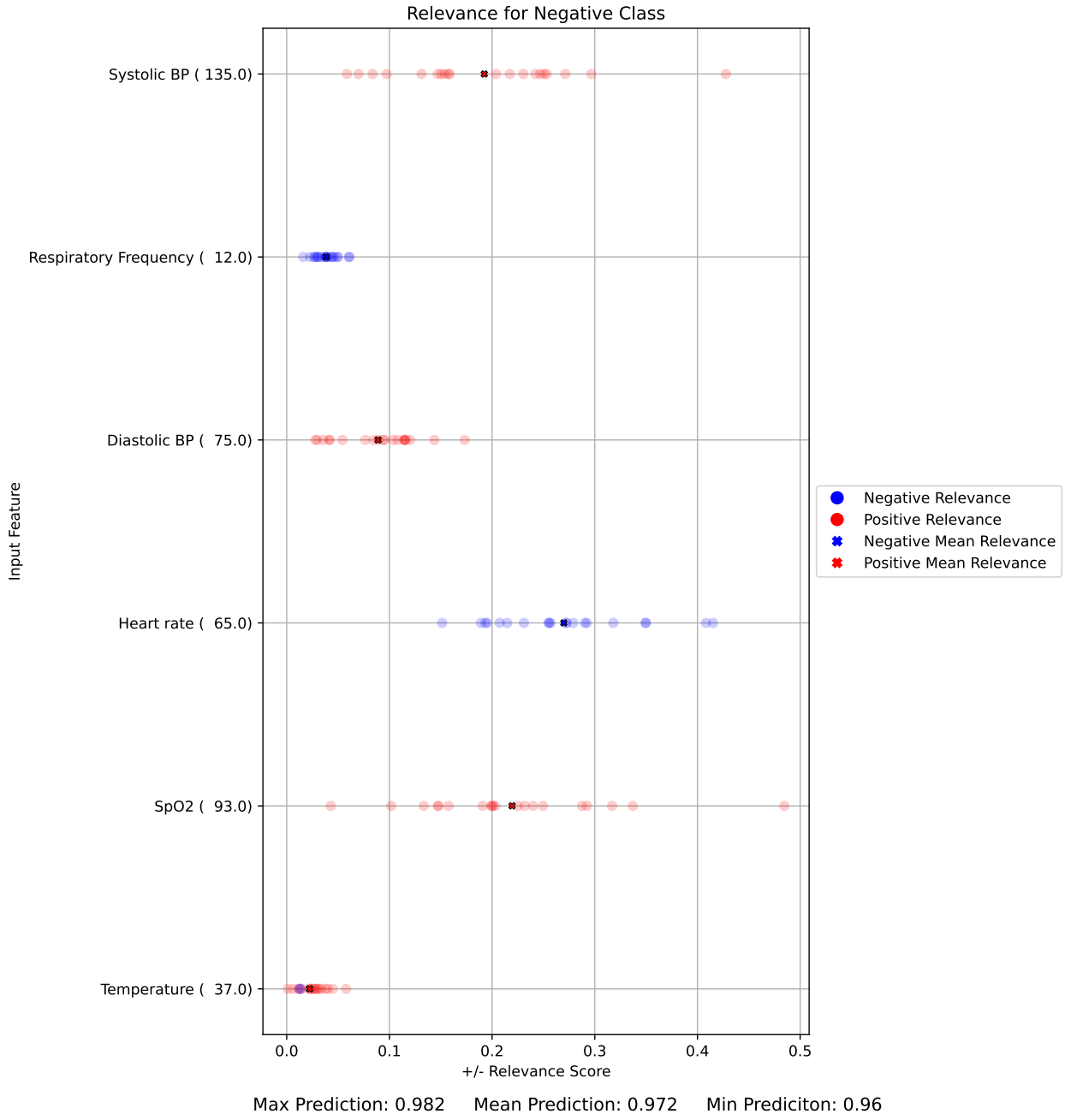


Figure N.3: Neg4(Original)False

N.2.2 Neg6(Original)False

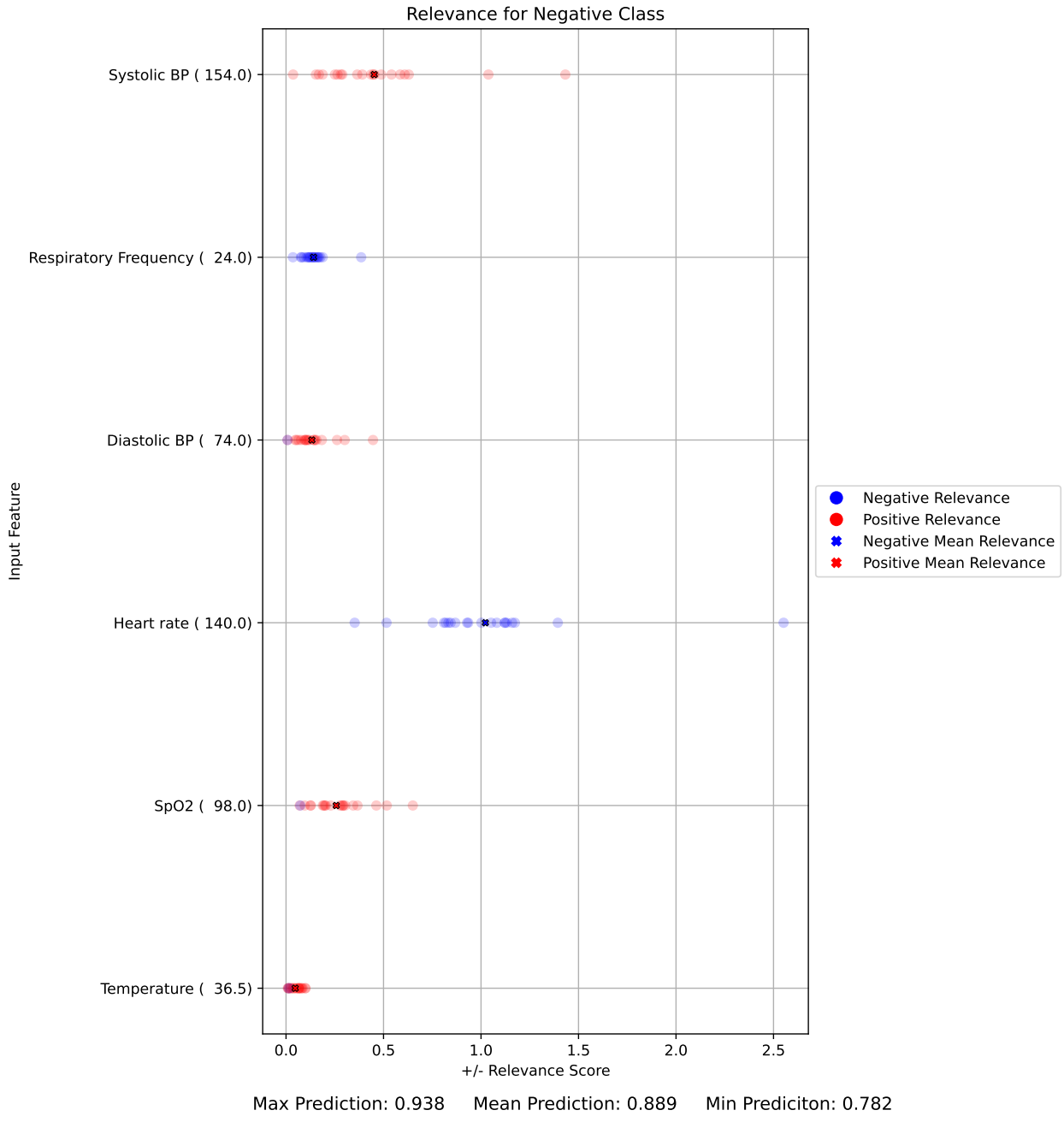


Figure N.4: Neg6(Original)False

N.2.3 Pos38(Original)True

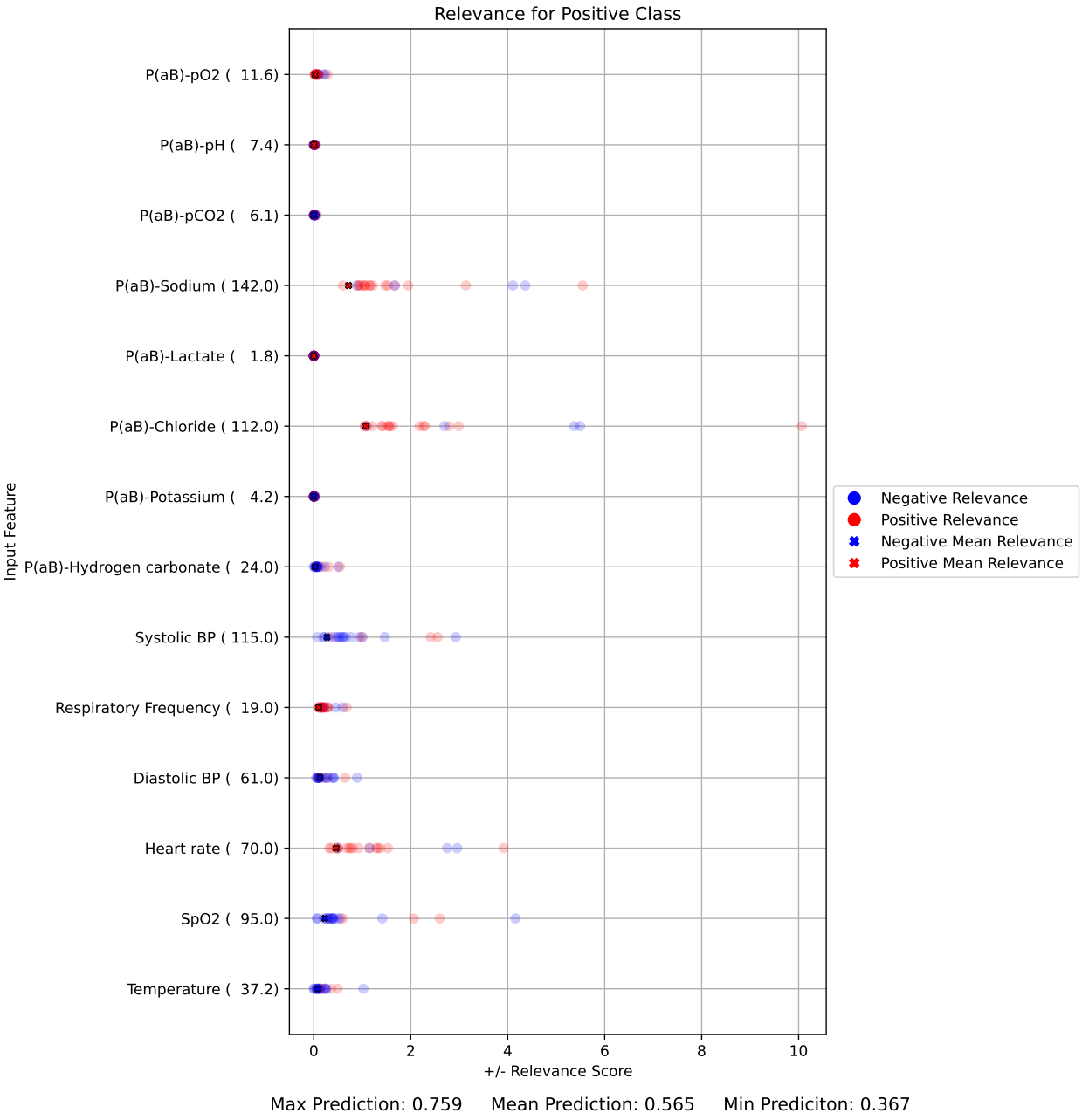


Figure N.5: Pos38(Original)True

N.2.4 Pos48(Original)True

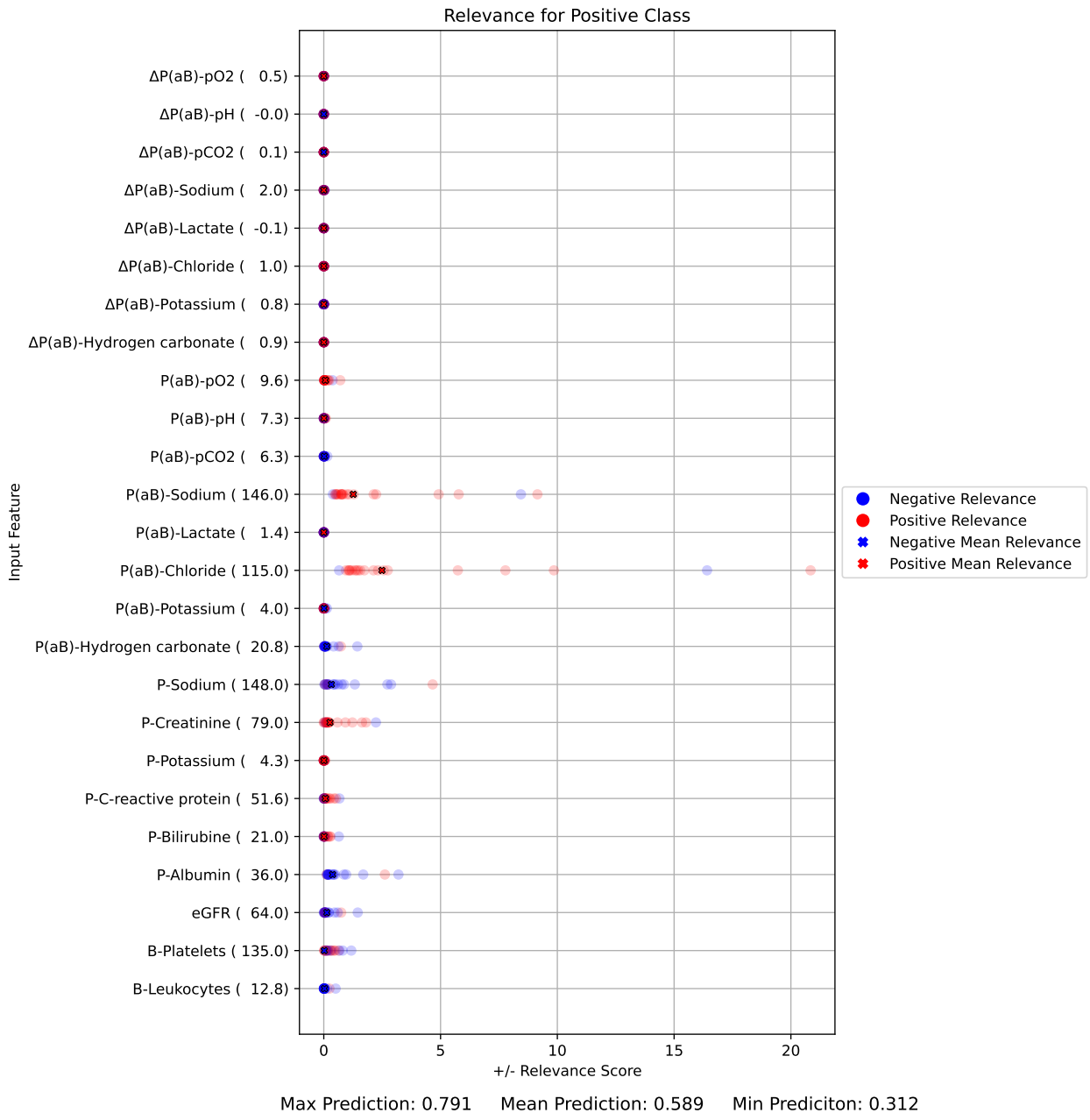


Figure N.6: Pos48(Original)True

N.3 Feature Flipping (ΔDP-A)

N.3.1 Neg0(deltaRespirationRateFF)False

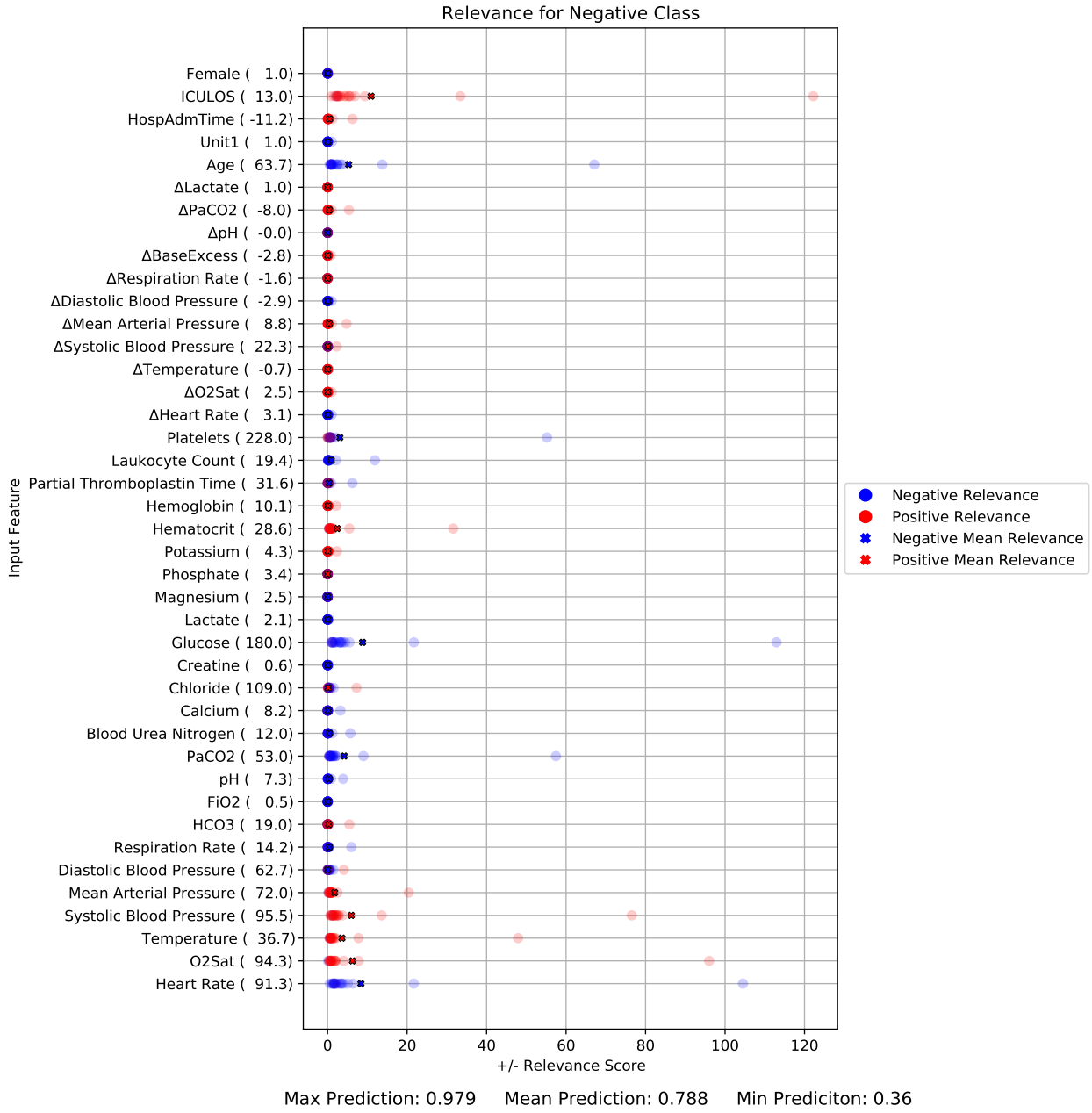


Figure N.7: Neg0(deltaRespirationRateFF)False

N.3.2 Neg0(glucoseFF)False

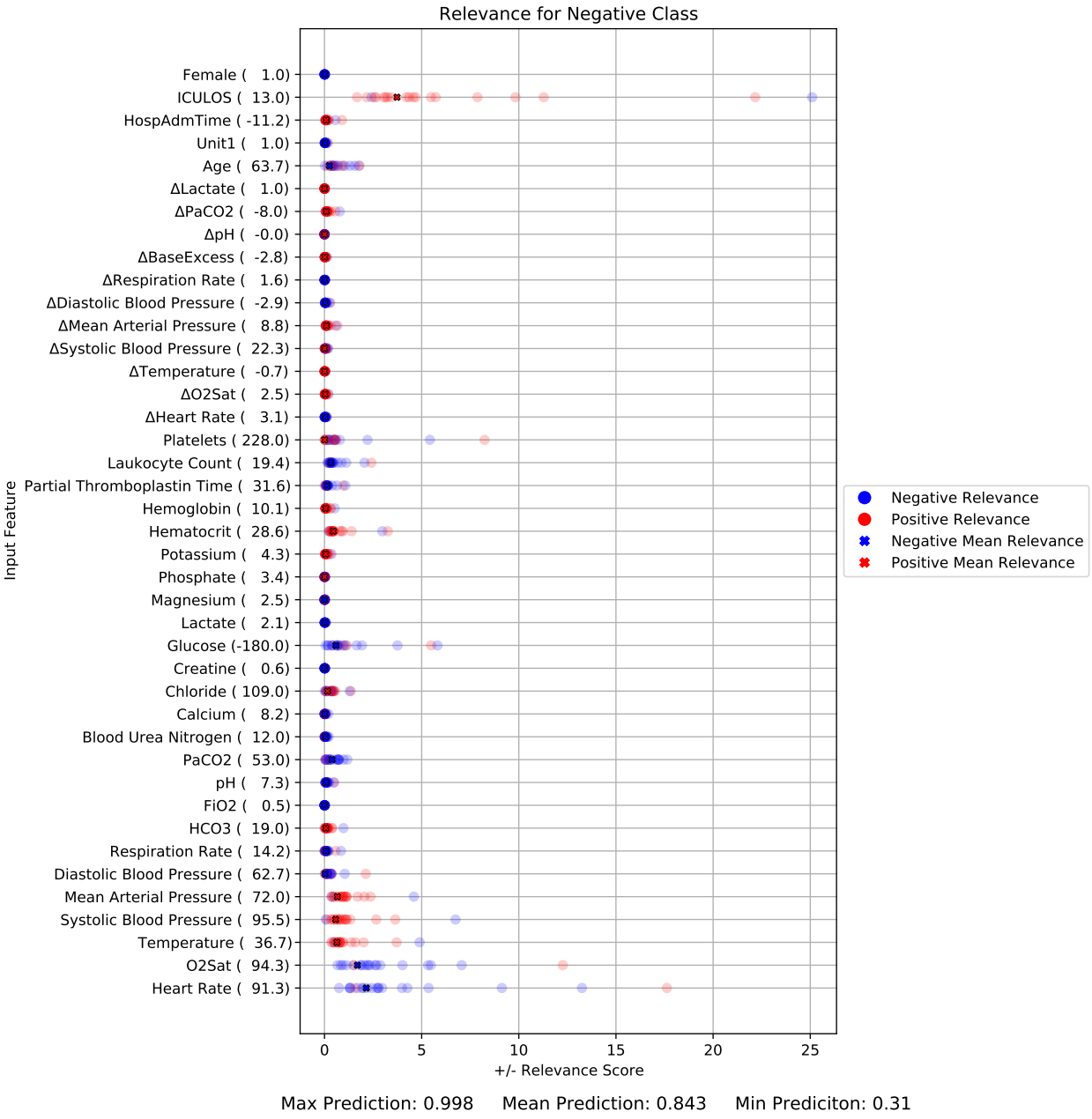


Figure N.8: Neg0(glucoseFF)False

N.3.3 Neg0(iculosFF)False

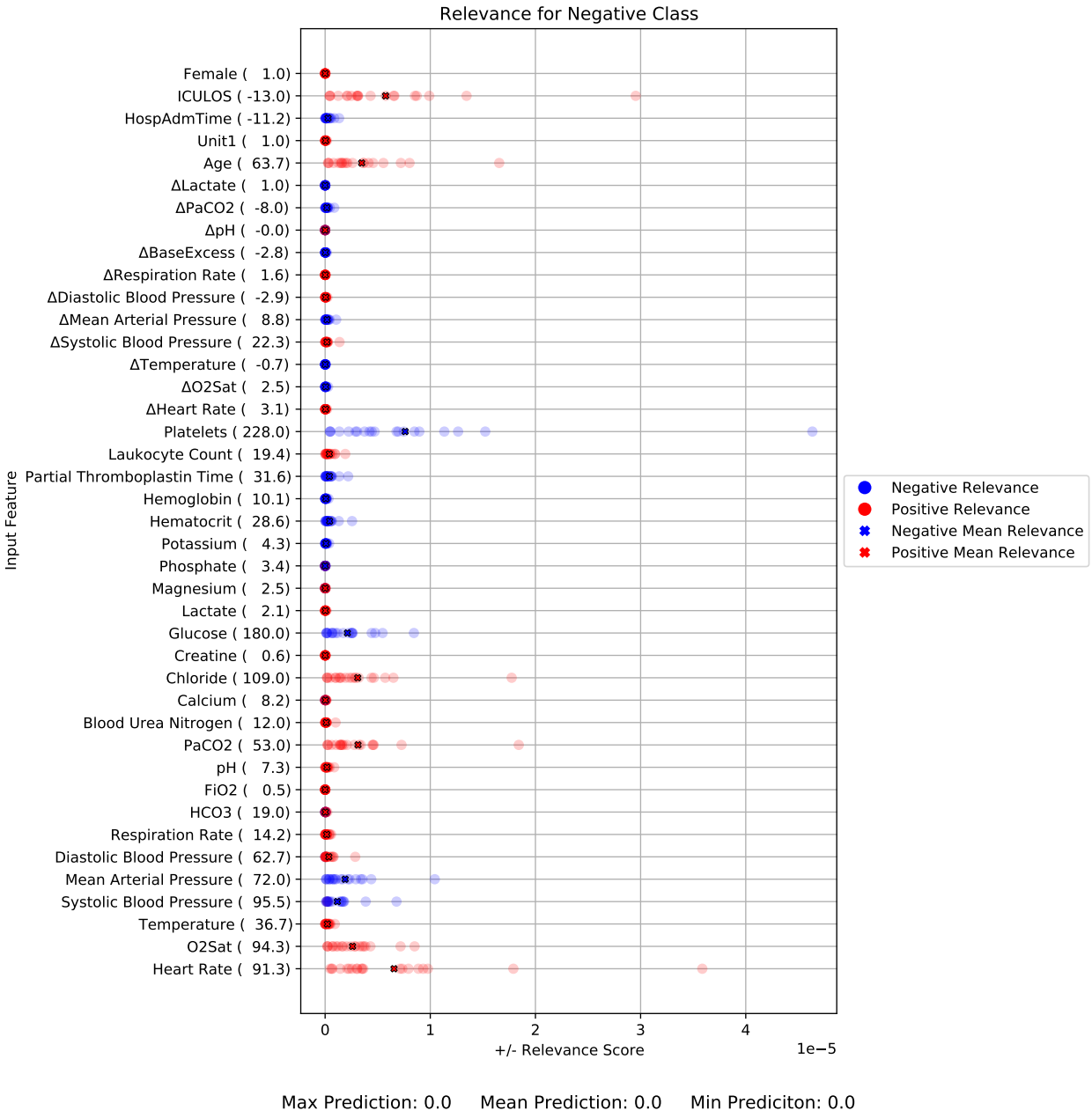


Figure N.9: Neg0(iculosFF)False

N.3.4 POS8(baseExcessFF)True

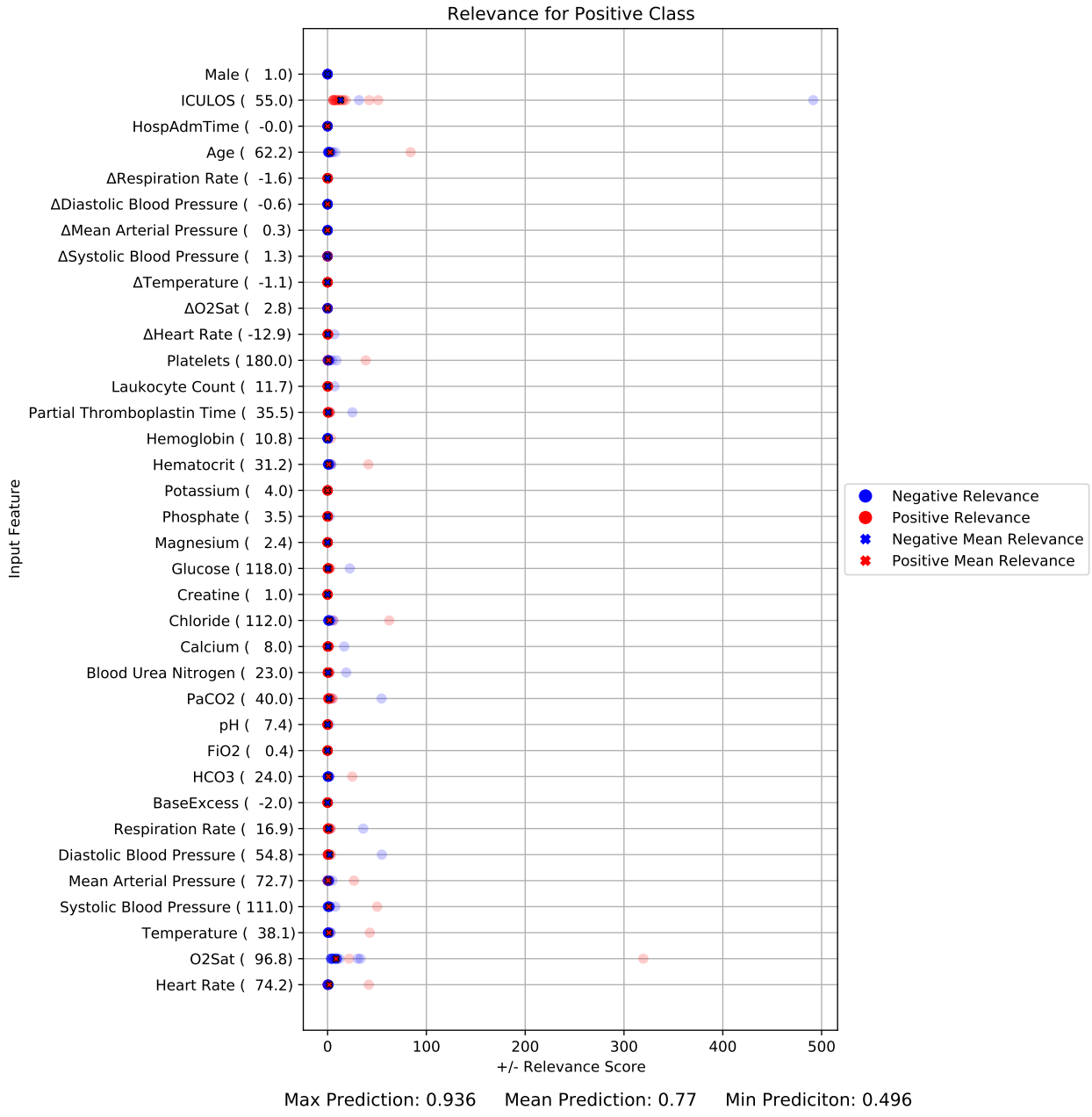


Figure N.10: POS8(baseExcessFF)True

N.3.5 POS8(iculosFF)True

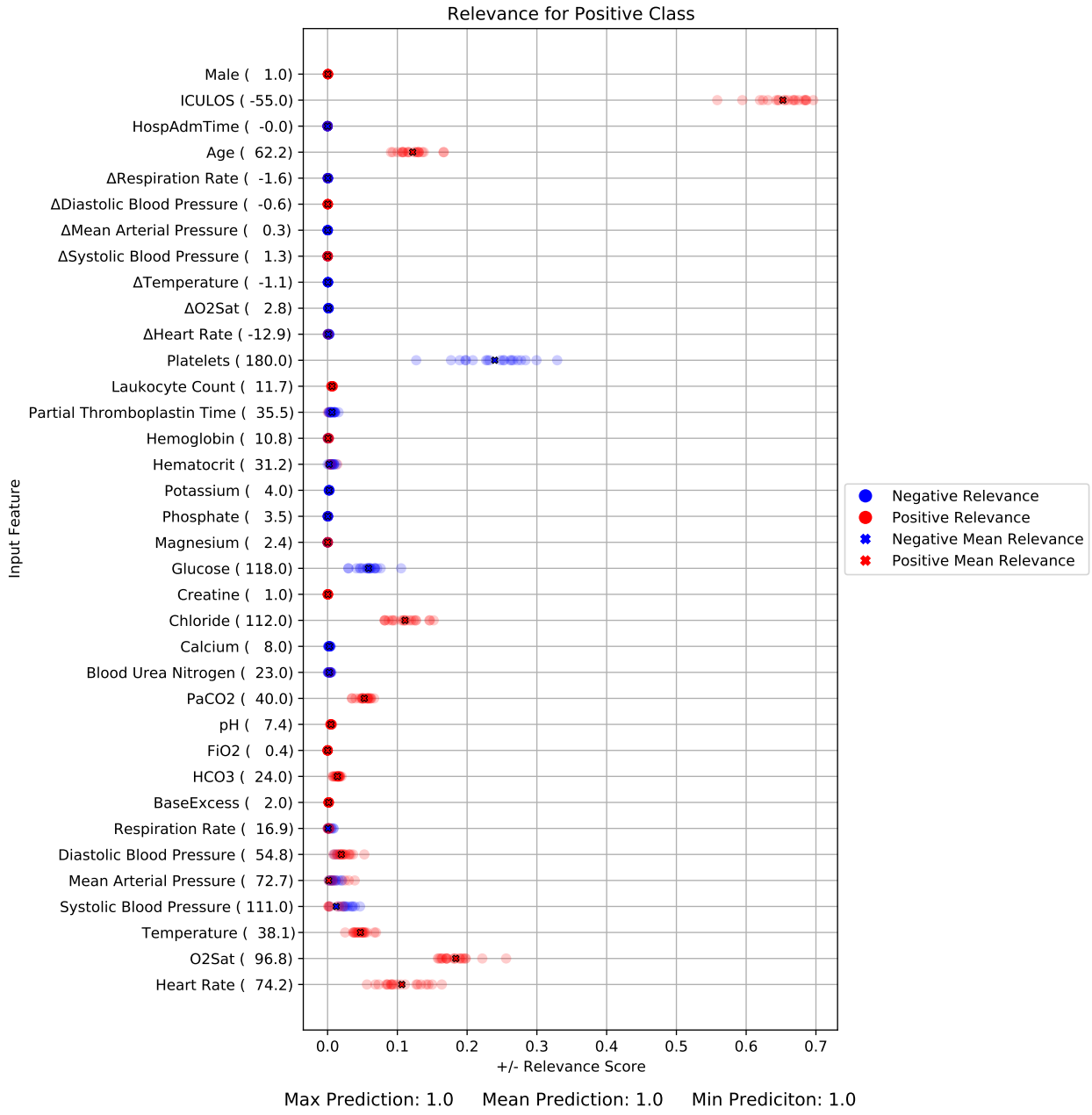


Figure N.11: POS8(iculosFF)True

N.3.6 POS8(o2satFF)True

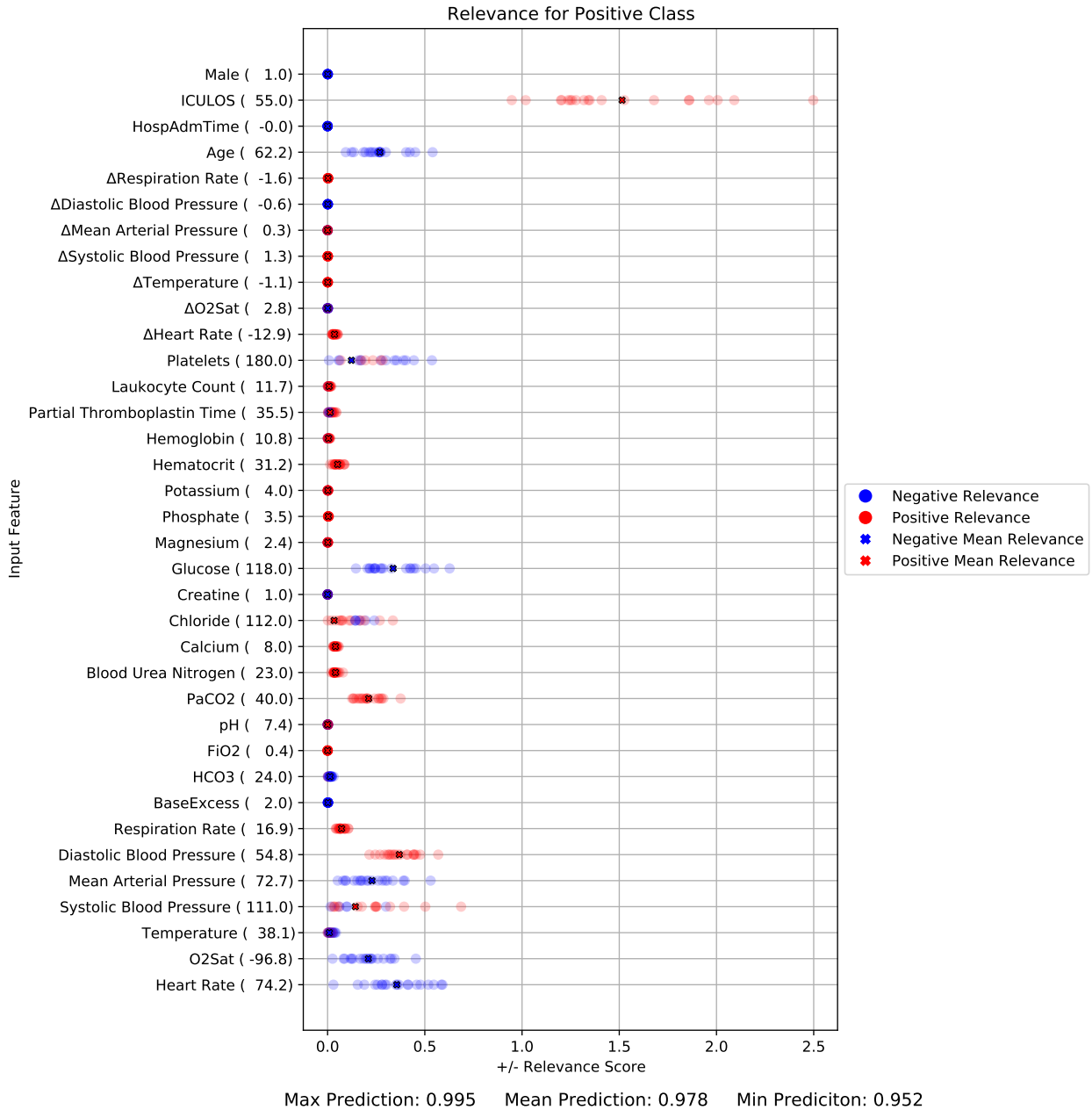


Figure N.12: POS8(o2satFF)True

N.4 Feature Flipping (ΔDCT)

N.4.1 Neg6(heartRateFF)False

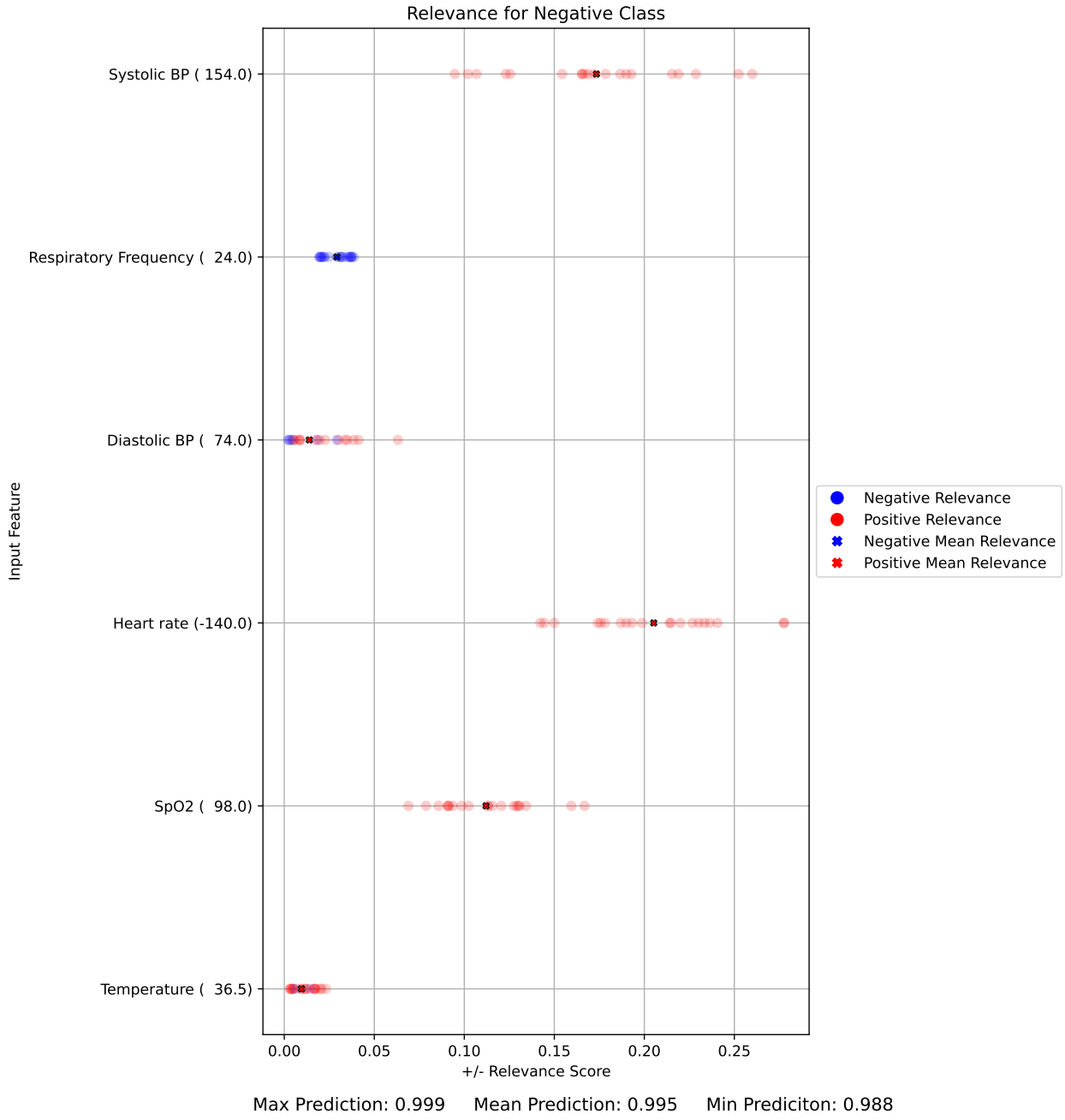


Figure N.13: Neg6(heartRateFF)False

N.4.2 Neg6(systolicBPFF)False

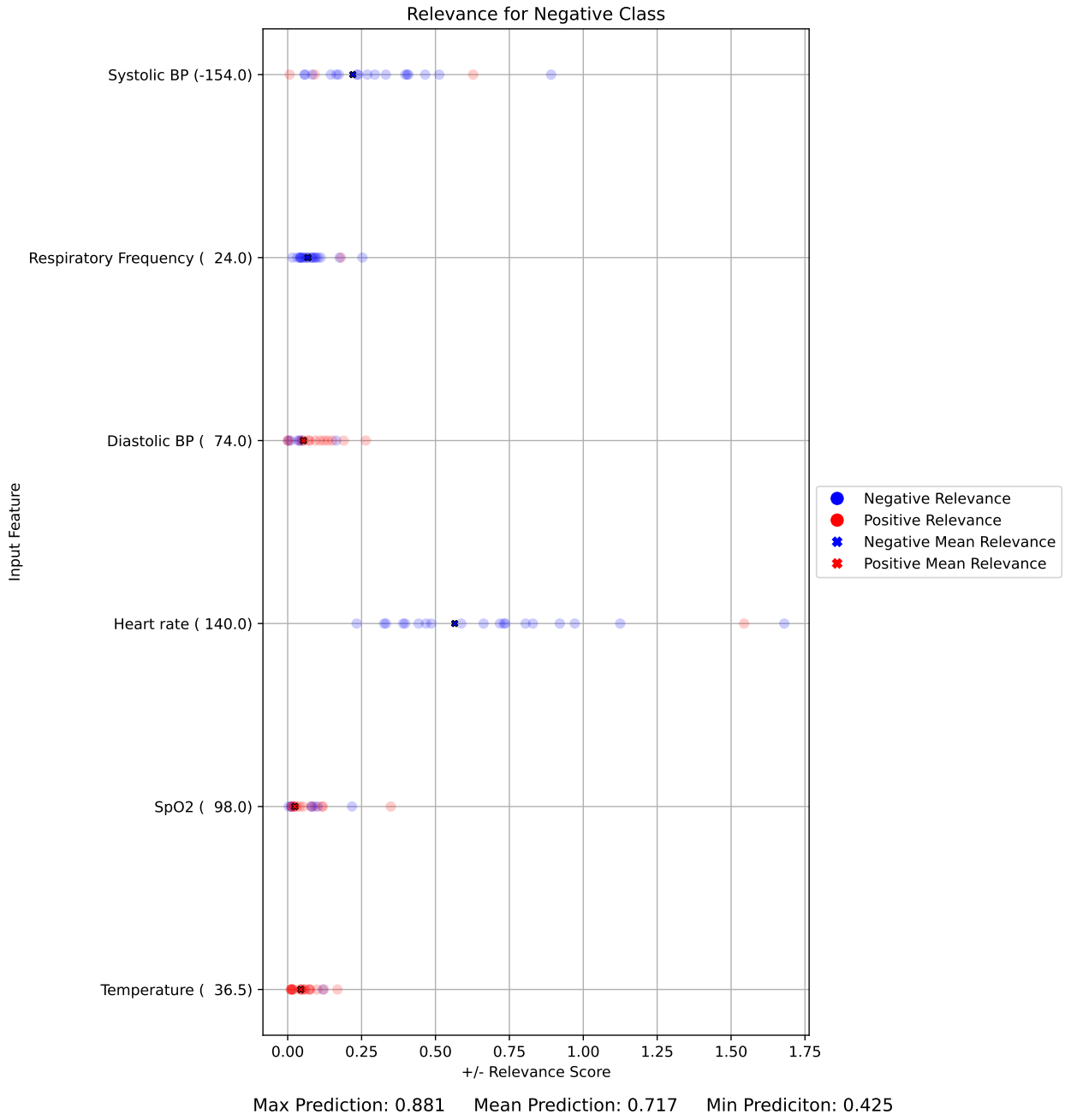


Figure N.14: Neg6(systolicBPFF)False

N.4.3 Neg6(temperatureFF)False

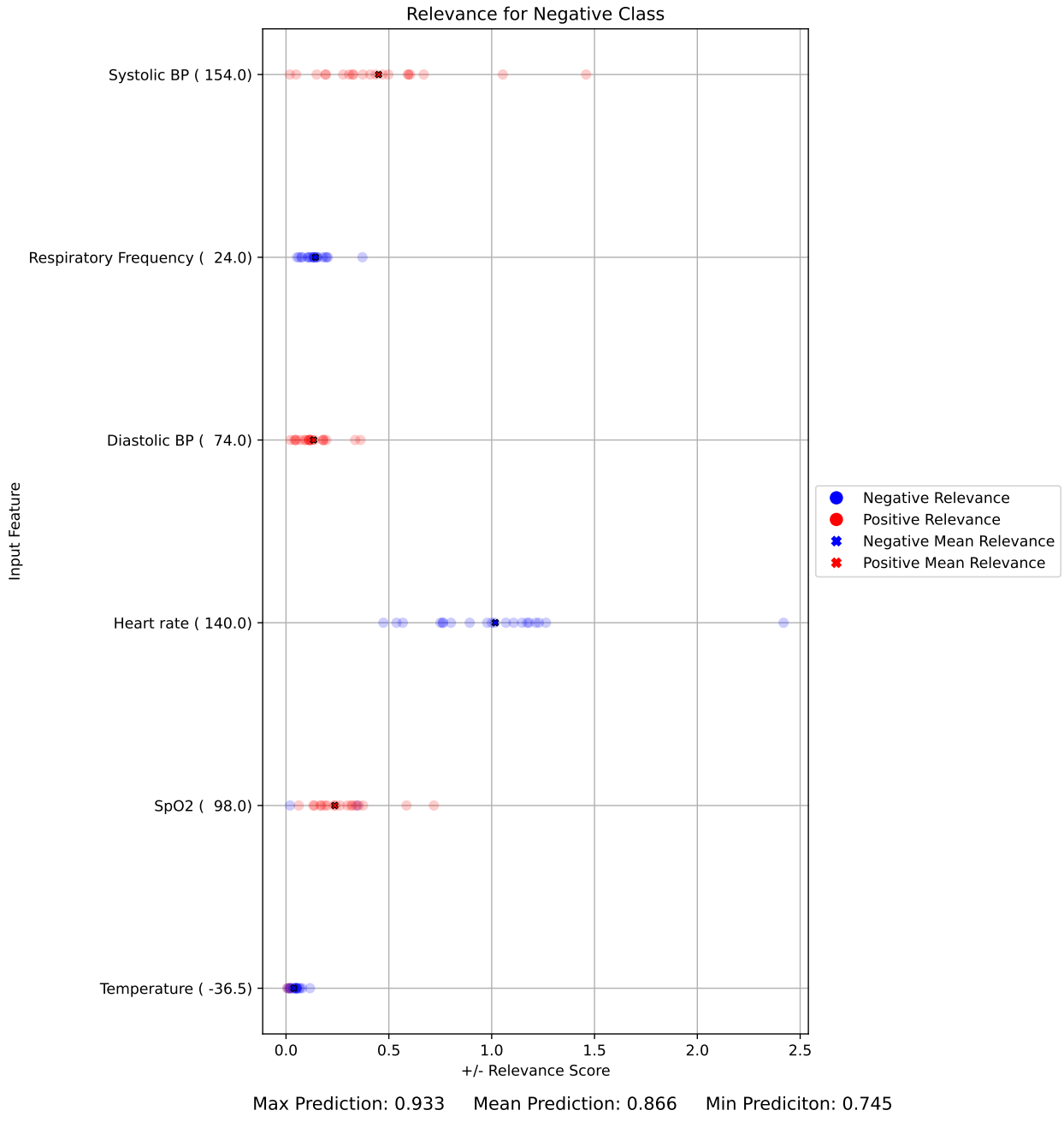


Figure N.15: Neg6(temperatureFF)False

N.4.4 Pos38(pabChlorideFF)True

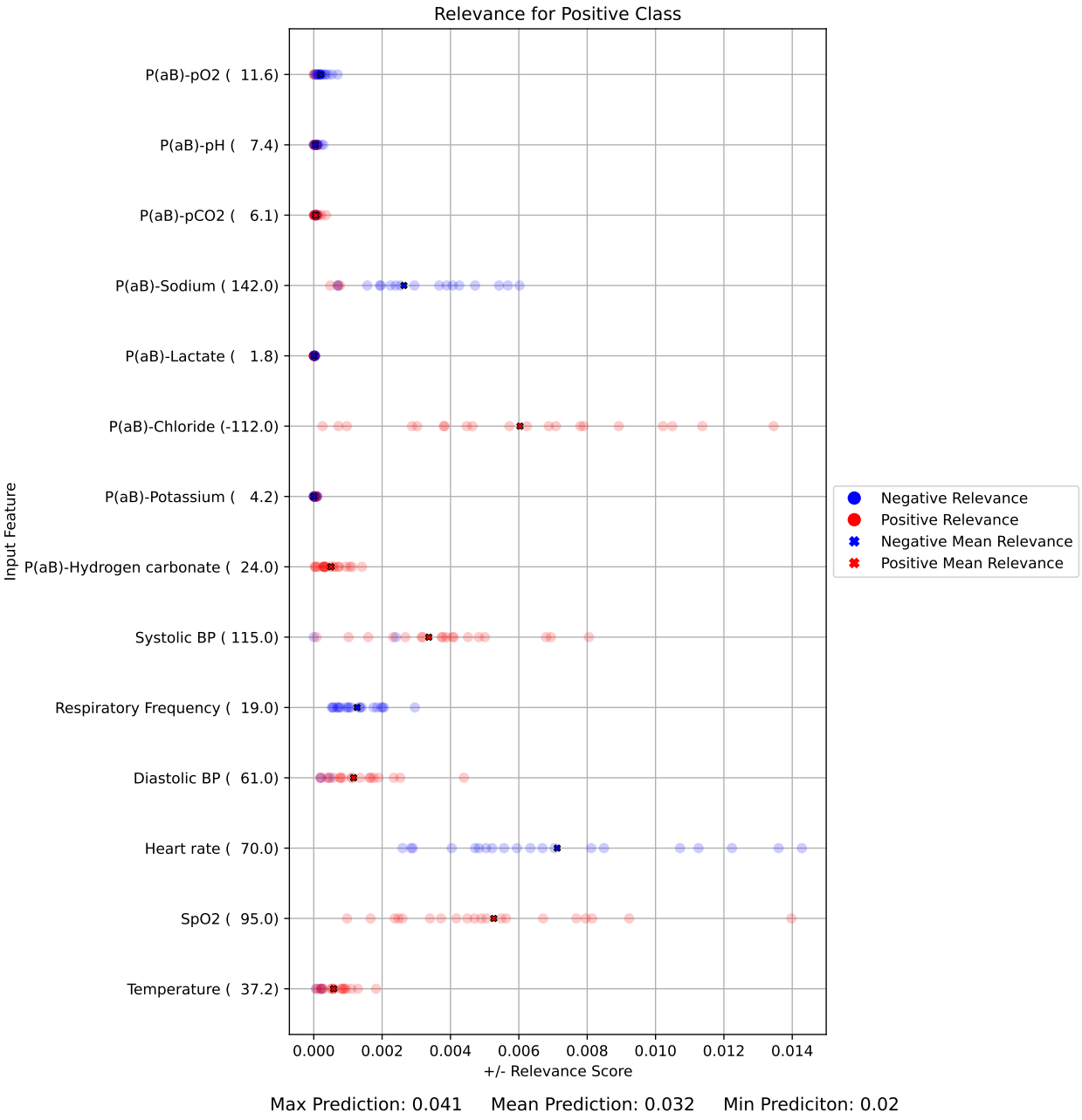


Figure N.16: Pos38(pabChlorideFF)True

N.4.5 Pos38(pabPotassiumFF)True

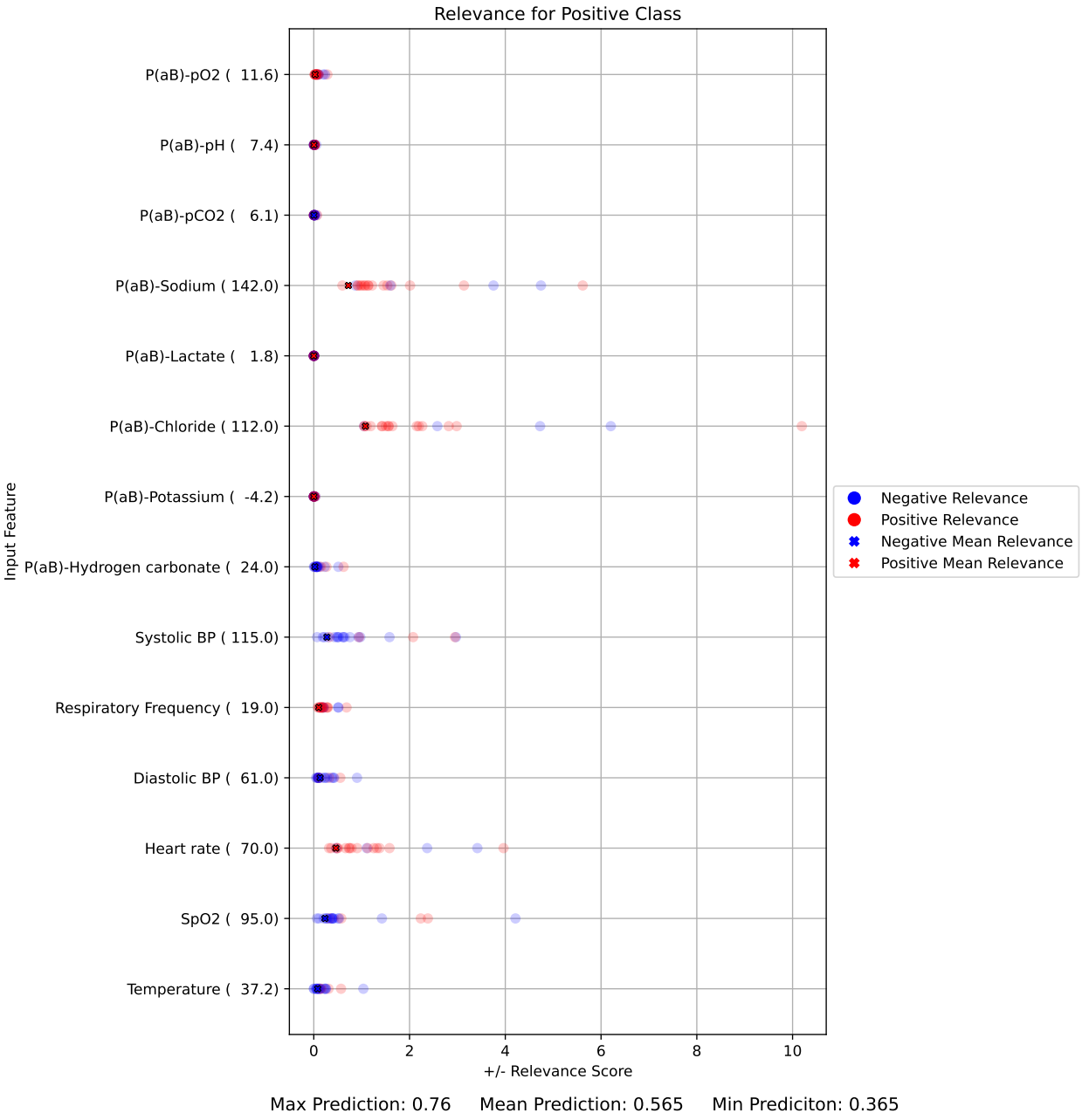


Figure N.17: Pos38(pabPotassiumFF)True

N.4.6 Pos38(systolicBPFF)True

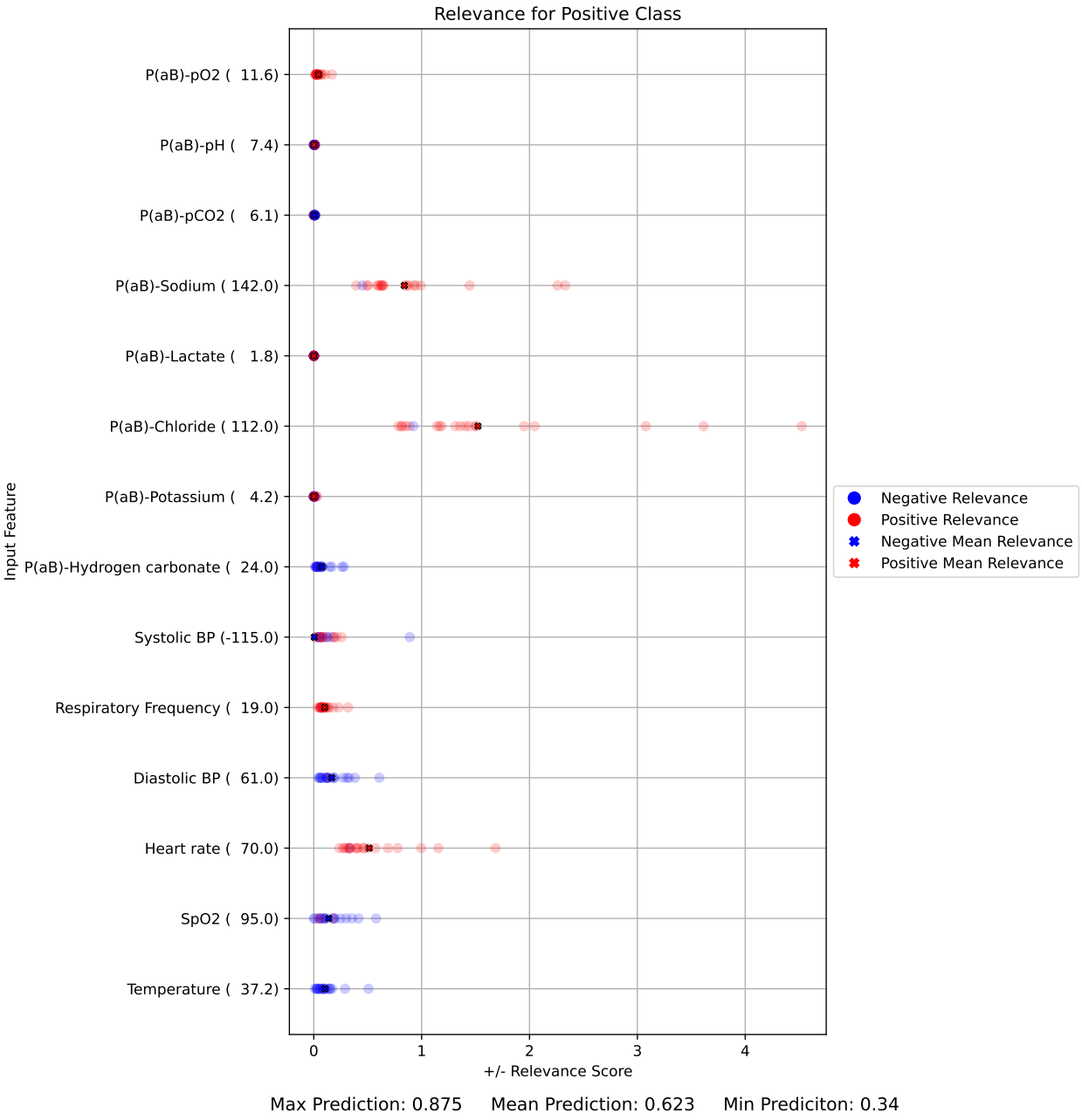


Figure N.18: Pos38(systolicBPFF)True

N.5 Setting Value to 0 (ΔDP-A)

N.5.1 Neg0(deltaRespirationRate0)False

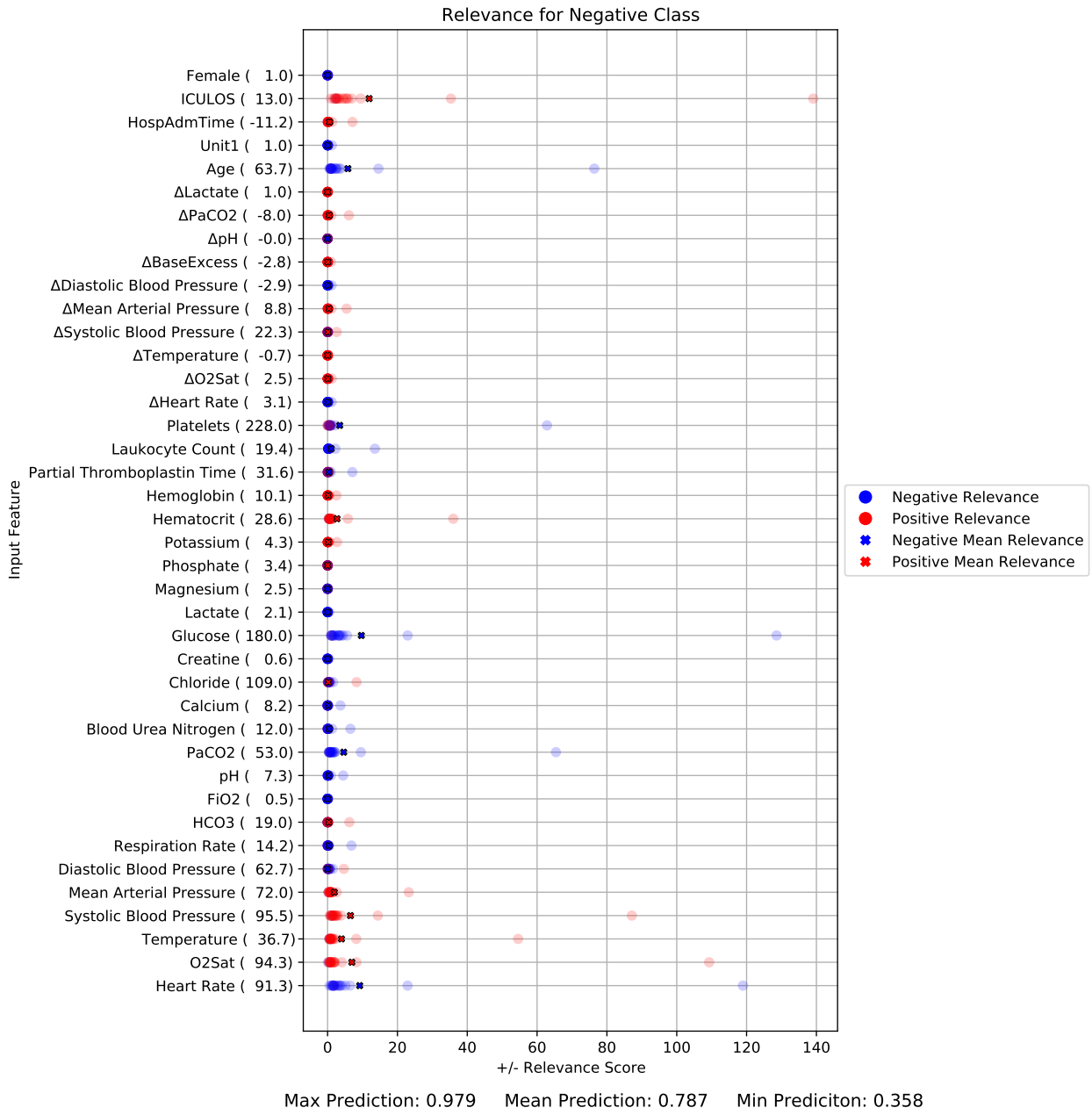


Figure N.19: Neg0(deltaRespirationRate0)False

N.5.2 Neg0(glucose0)False

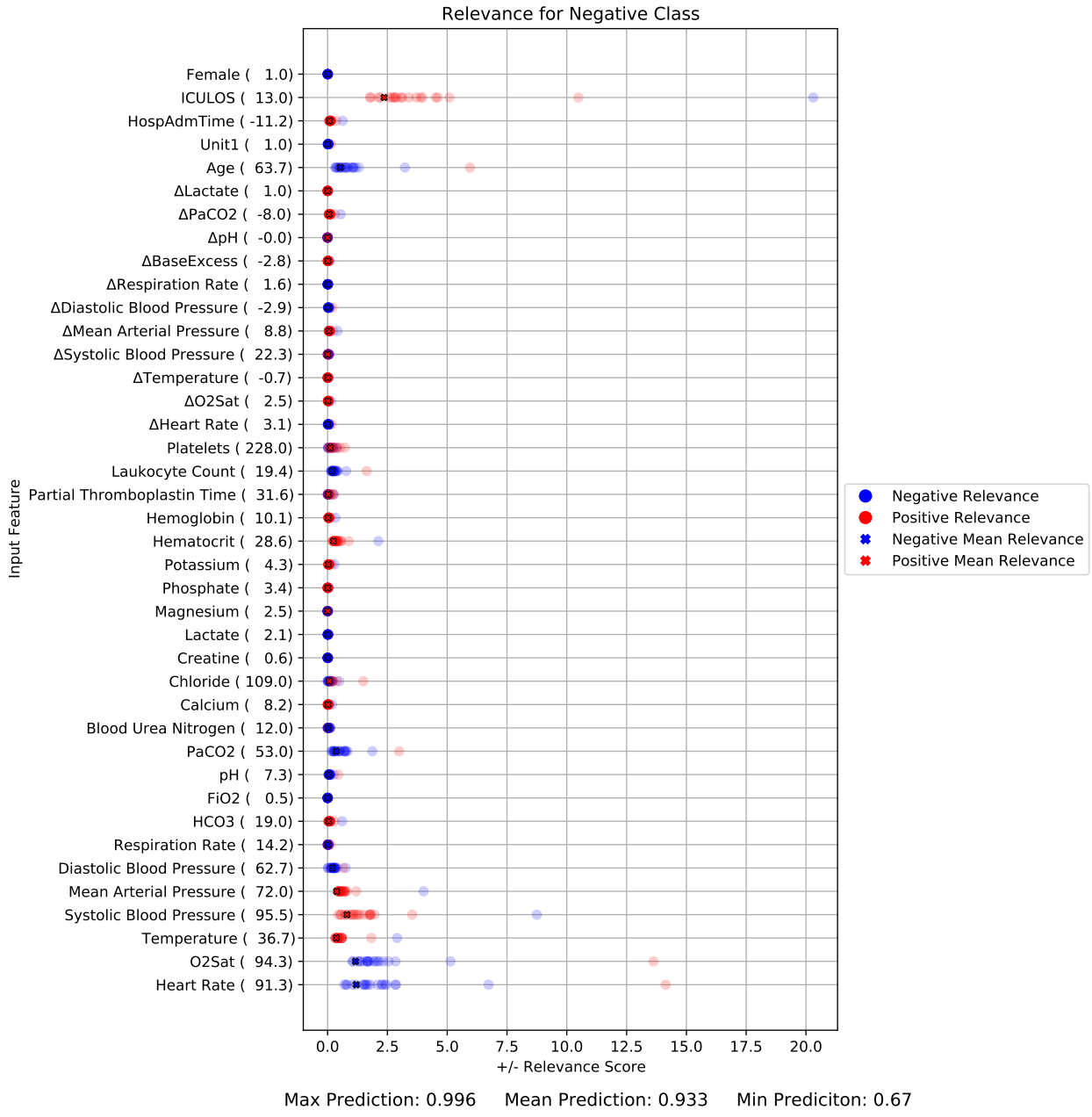


Figure N.20: Neg0(glucose0)False

N.5.3 Neg0(iculos0)False

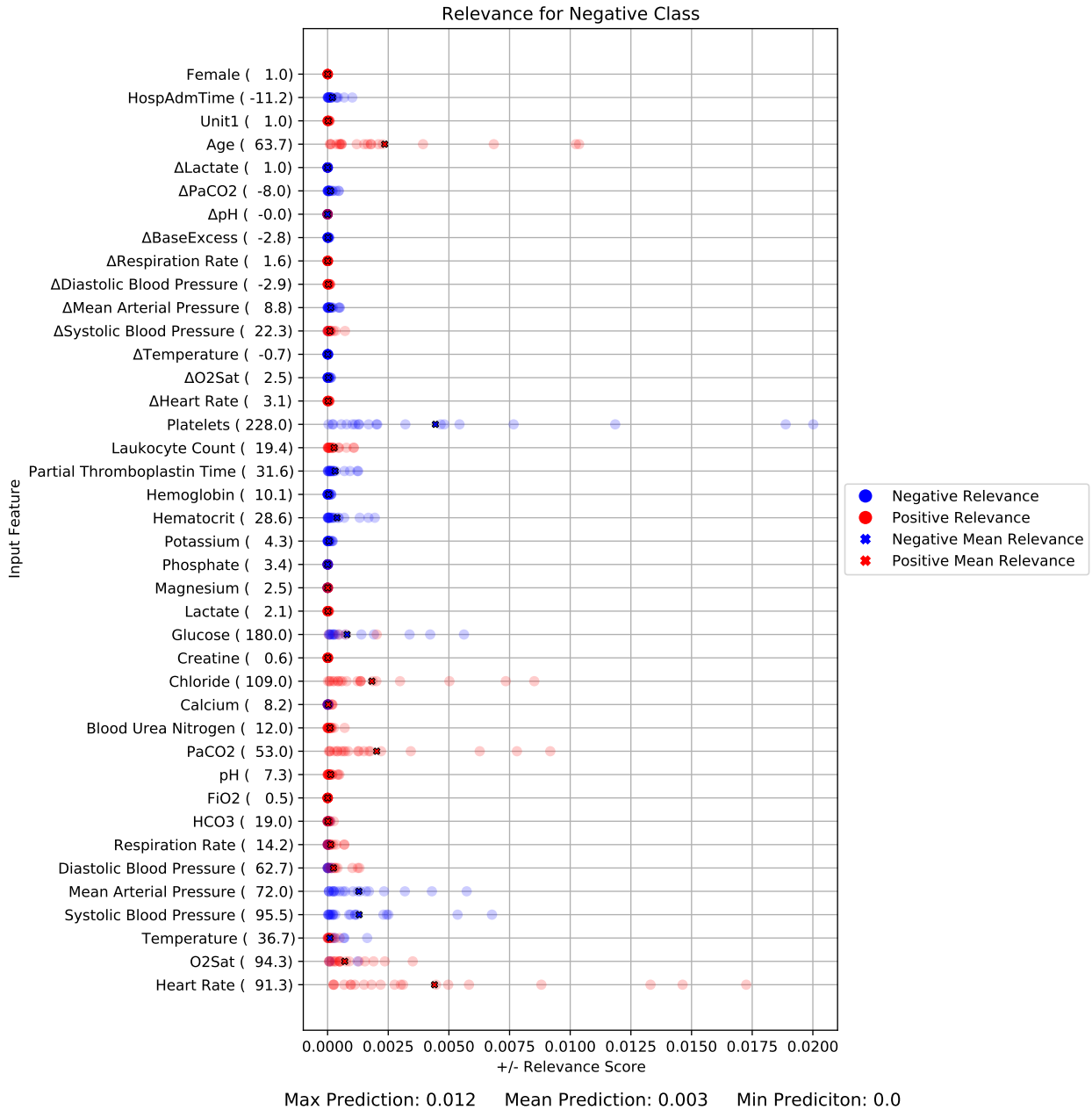


Figure N.21: Neg0(iculos0)False

N.5.4 POS8(baseExcess0)True

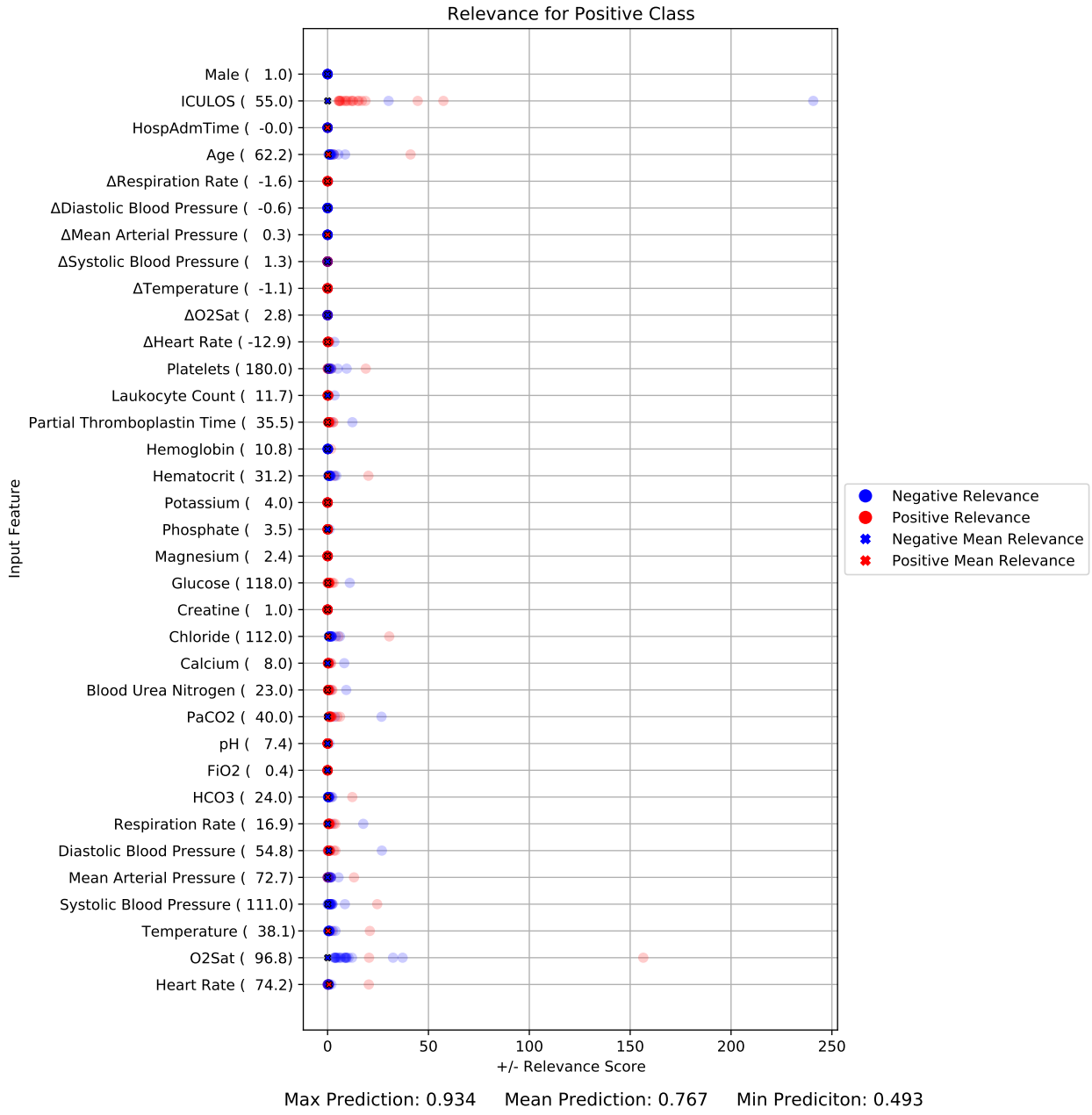


Figure N.22: POS8(baseExcess0)True

N.5.5 POS8(iculos0)True

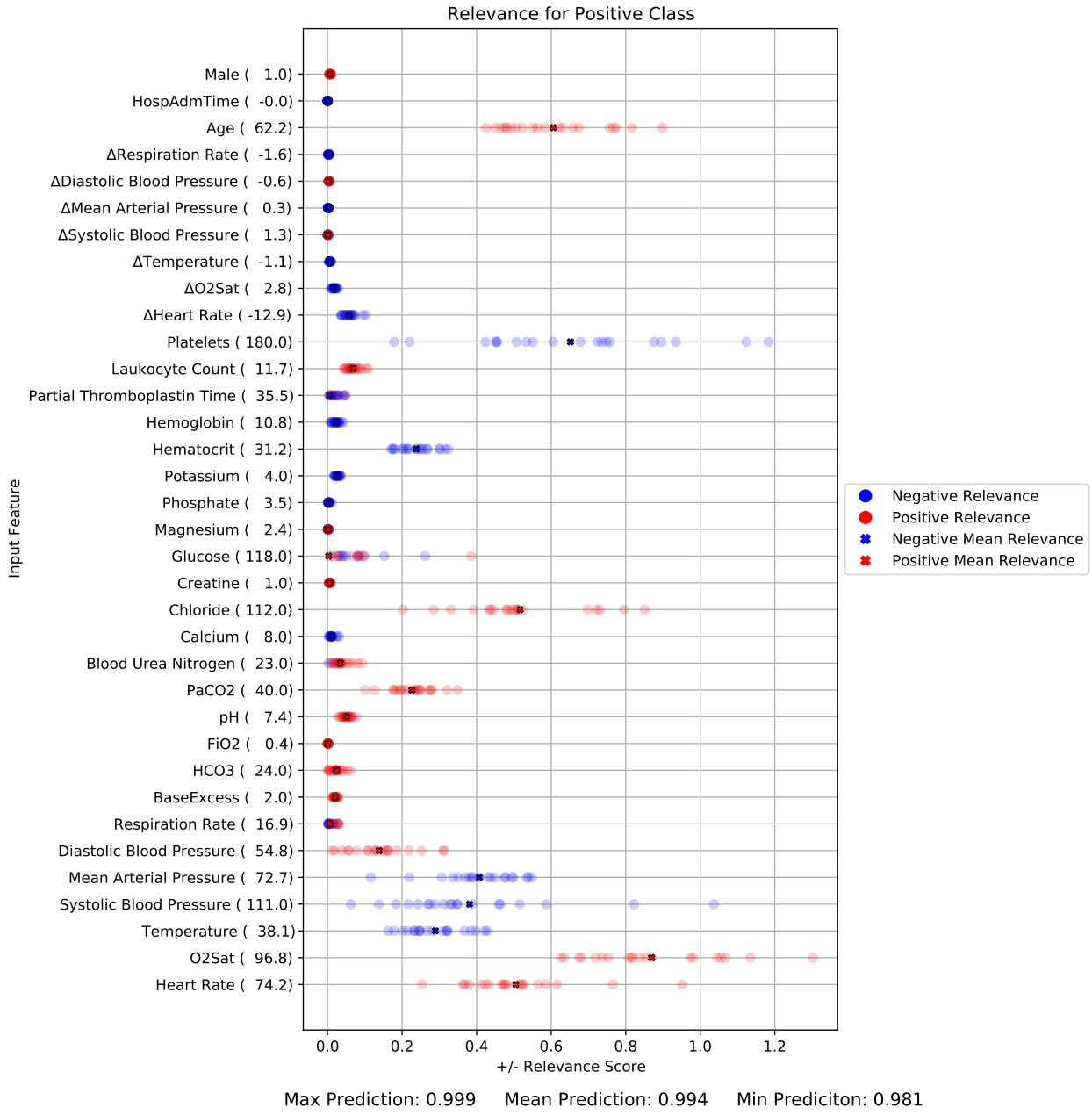


Figure N.23: POS8(iculos0)True

N.5.6 POS8(o2sat0)True

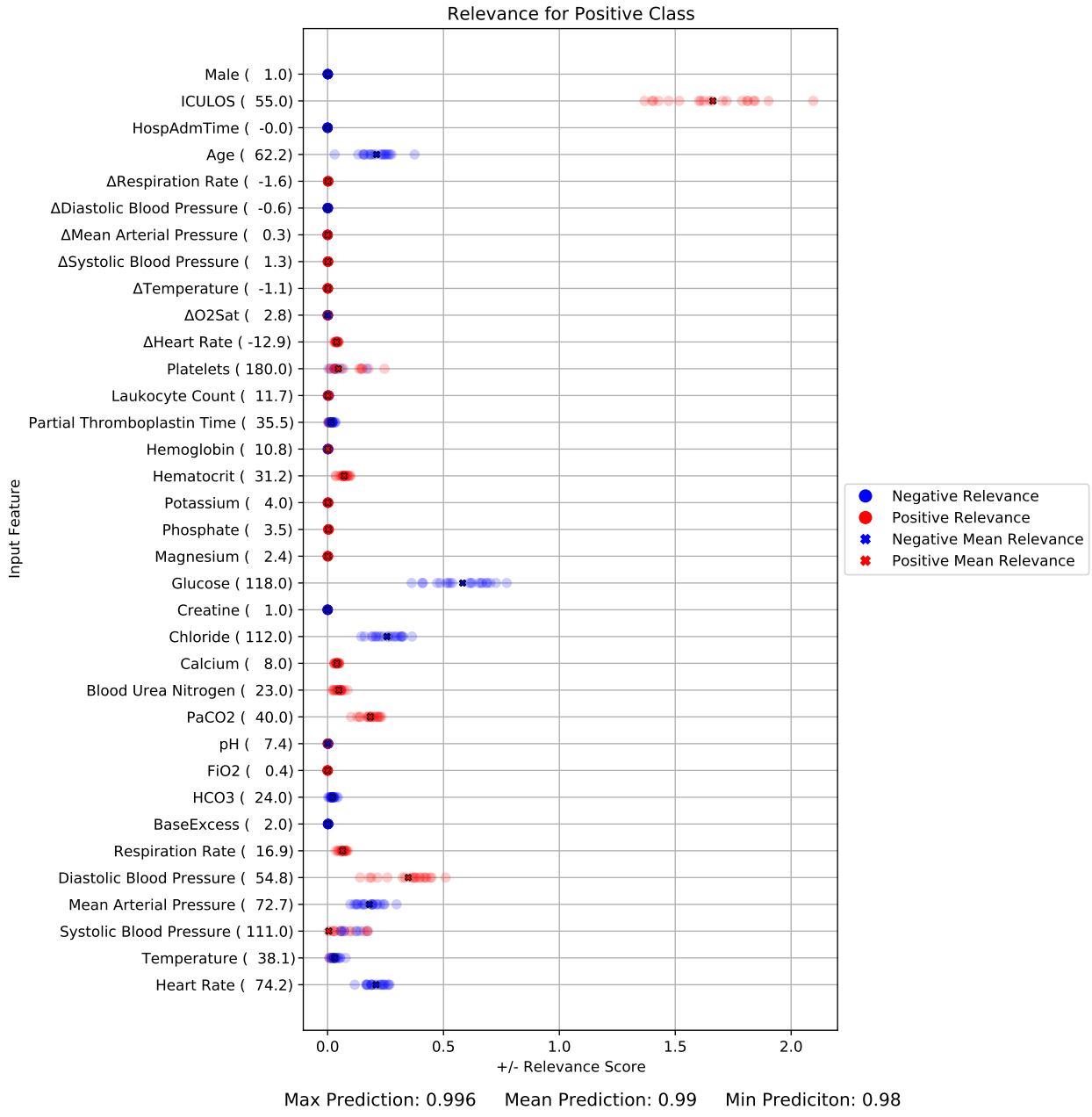


Figure N.24: POS8(o2sat0)True

N.6 Setting Value to 0 (ΔDCT)

N.6.1 Neg4(heartRate0)False

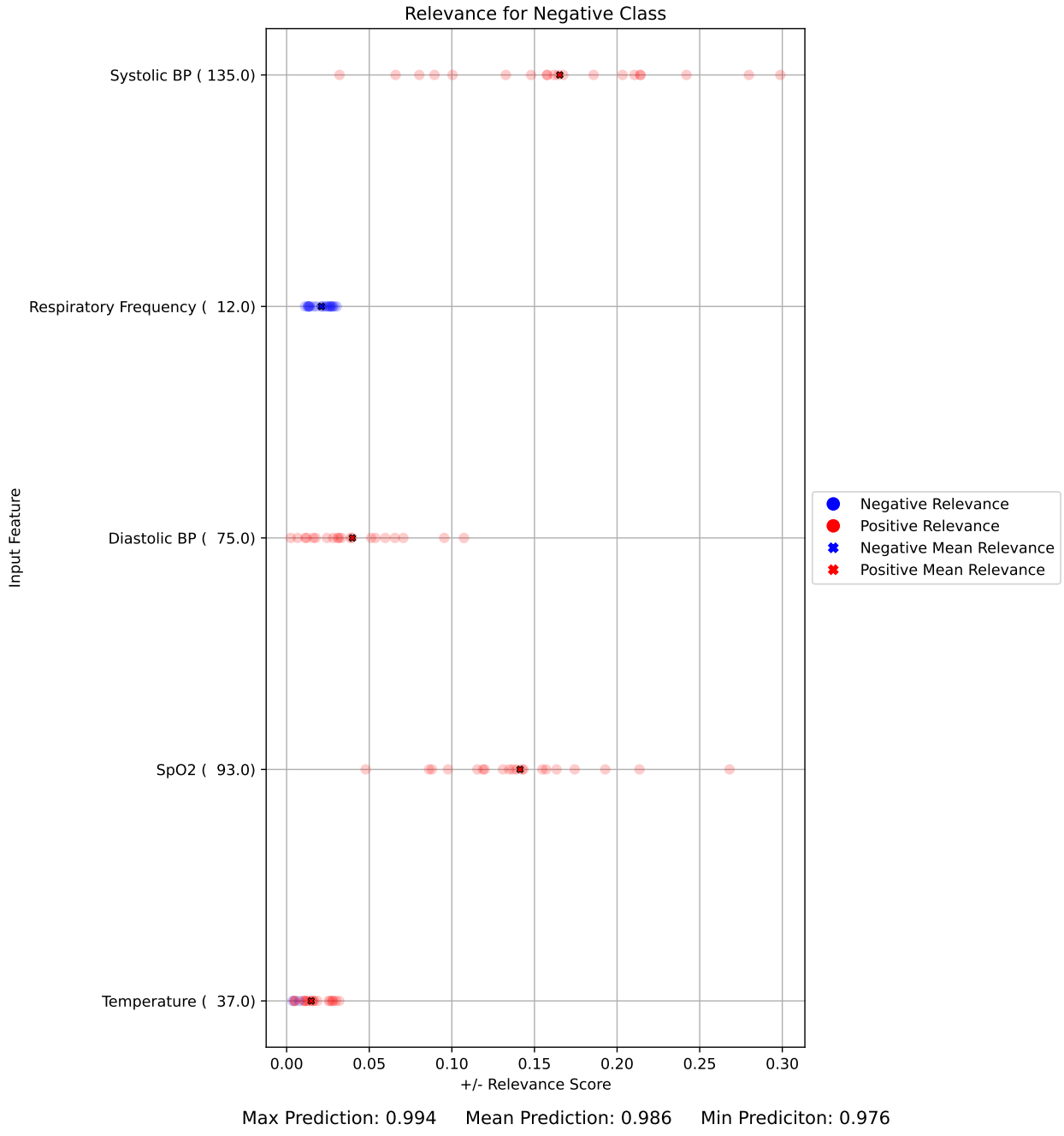


Figure N.25: Neg4(heartRate0)False

N.6.2 Neg4(spo20)False

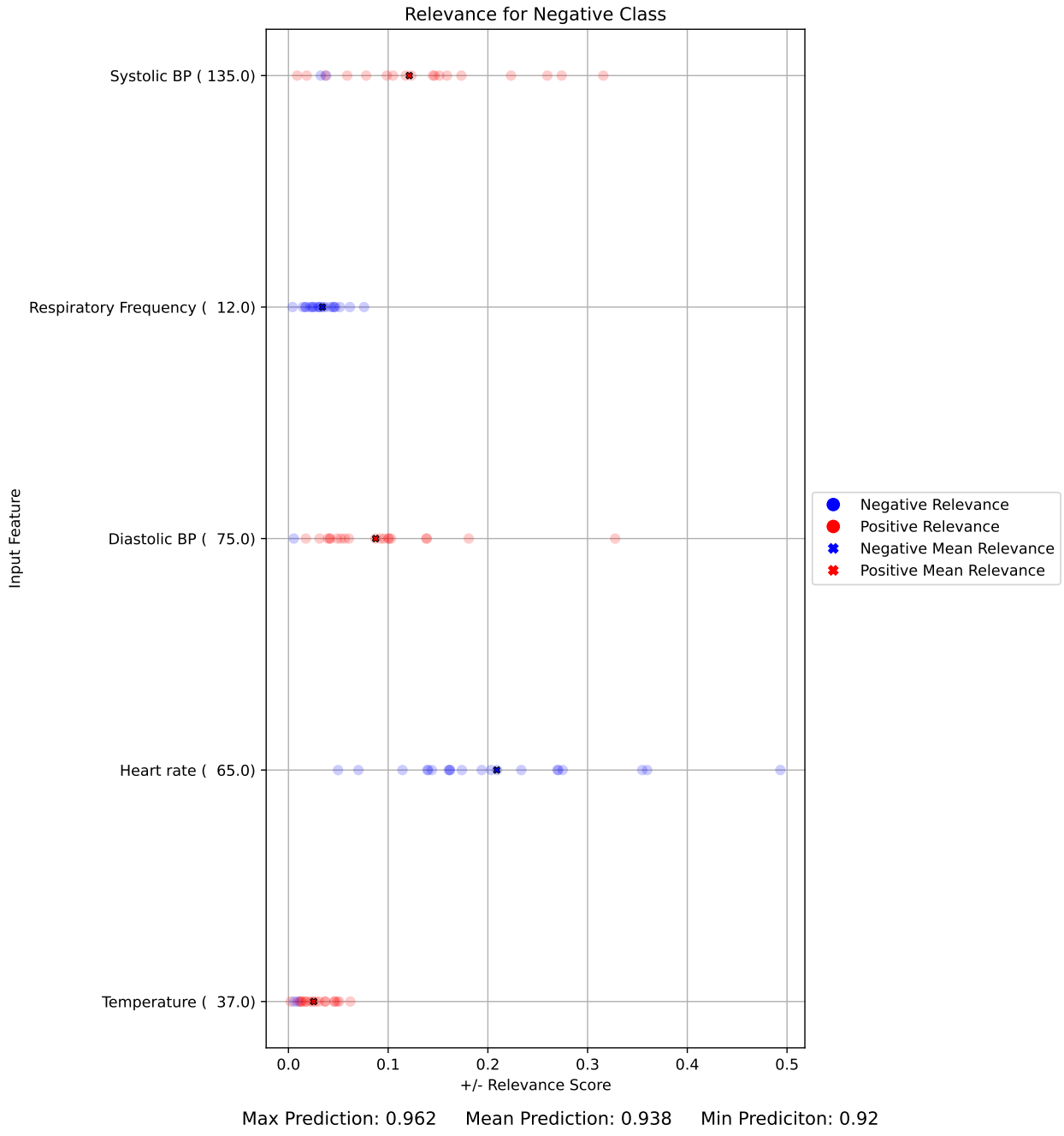


Figure N.26: Neg4(spo20)False

N.6.3 Neg4(temperature0)False

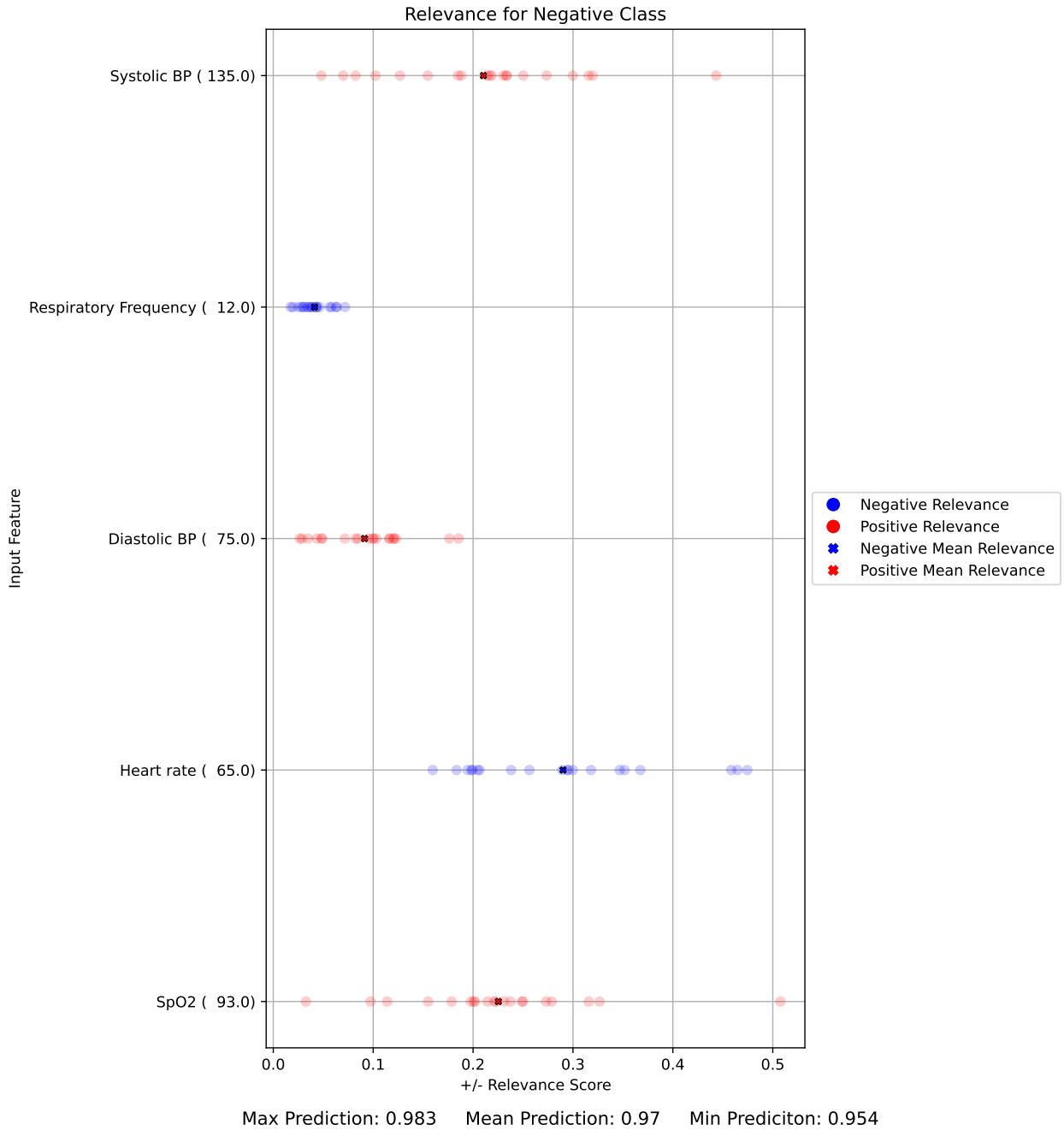


Figure N.27: Neg4(temperature0)False

N.6.4 Neg6(heartRate0)False

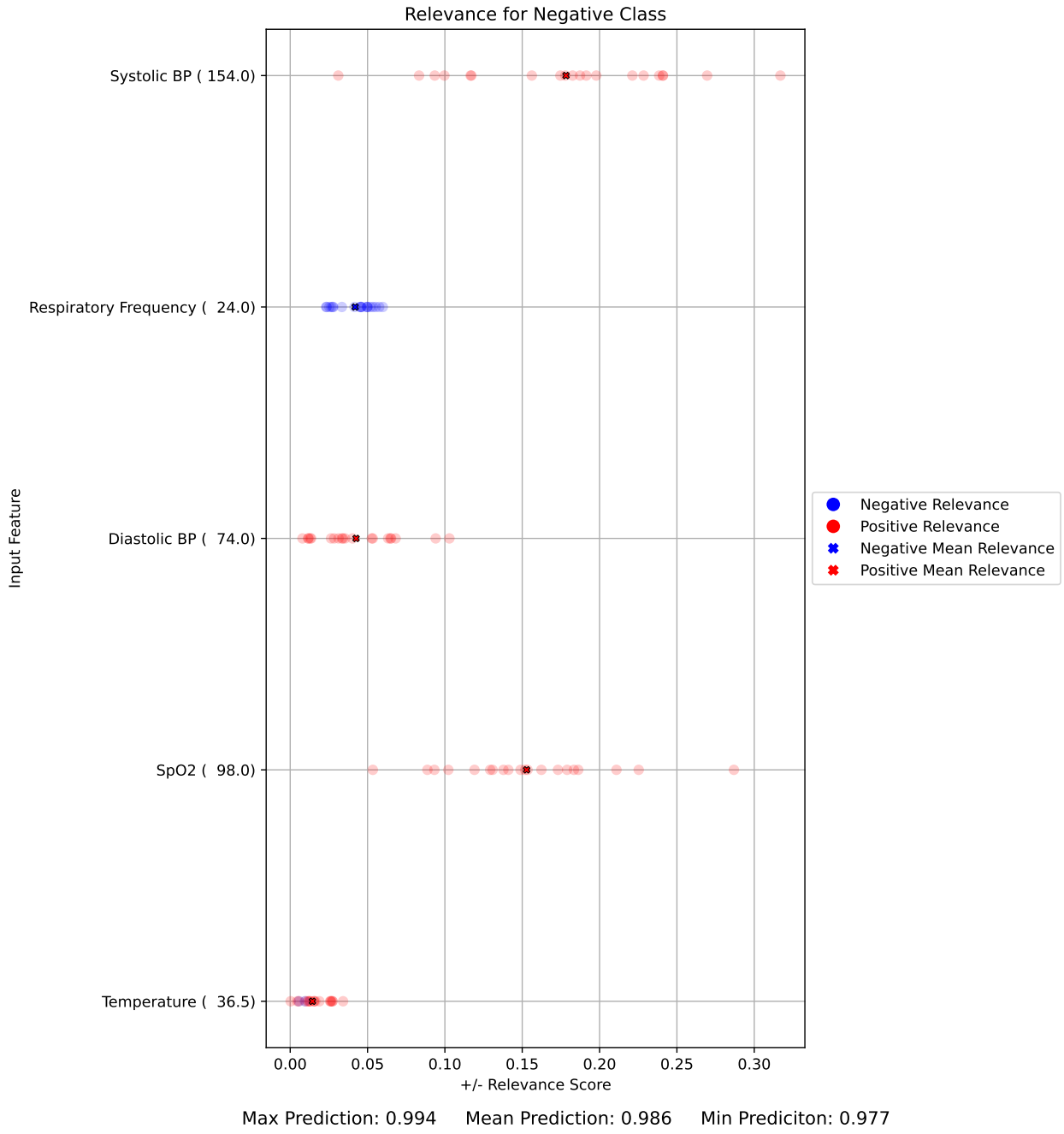


Figure N.28: Neg6(heartRate0)False

N.6.5 Neg6(systolicBP0)False

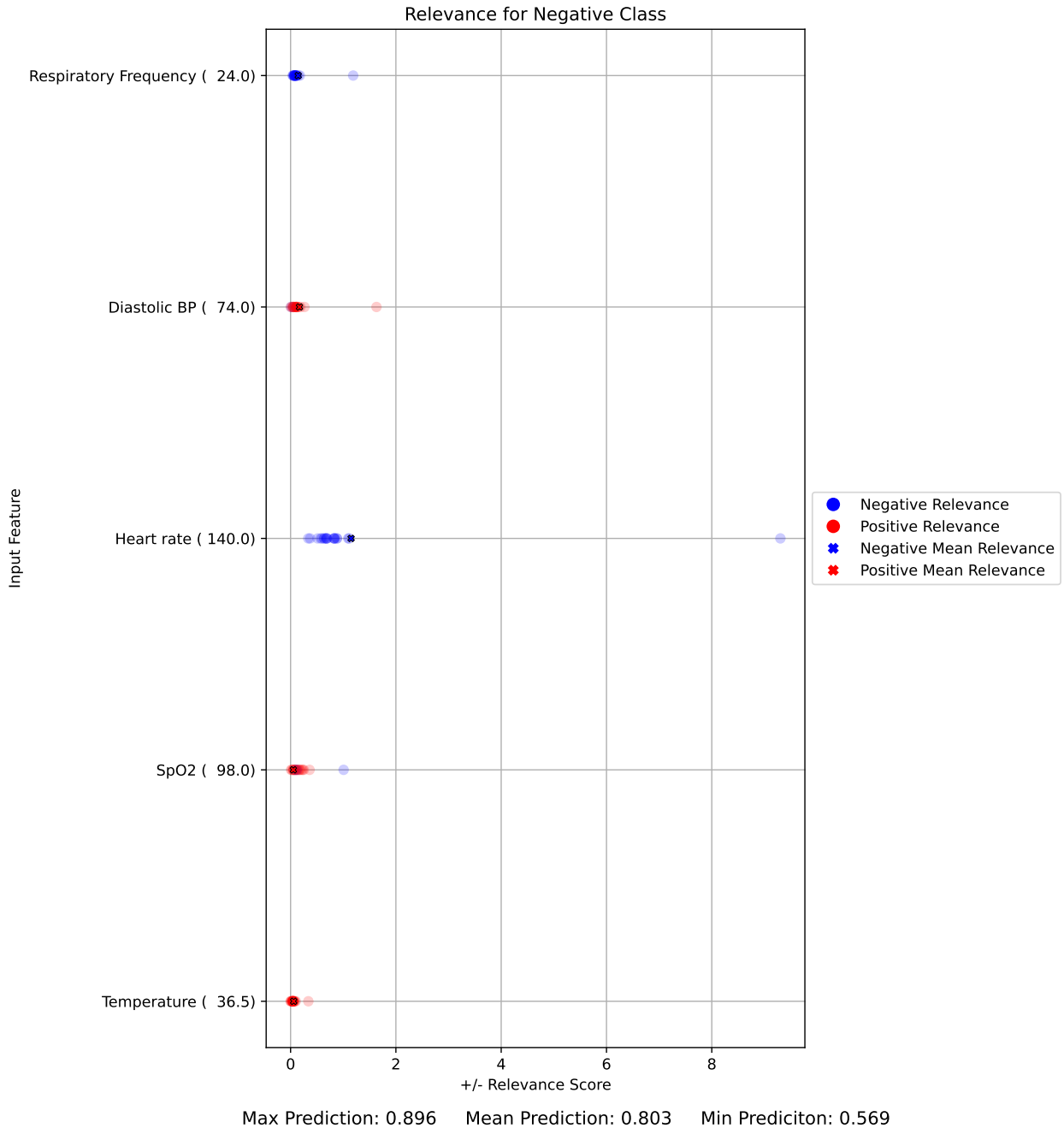


Figure N.29: Neg6(systolicBP0)False

N.6.6 Neg6(temperature0)False

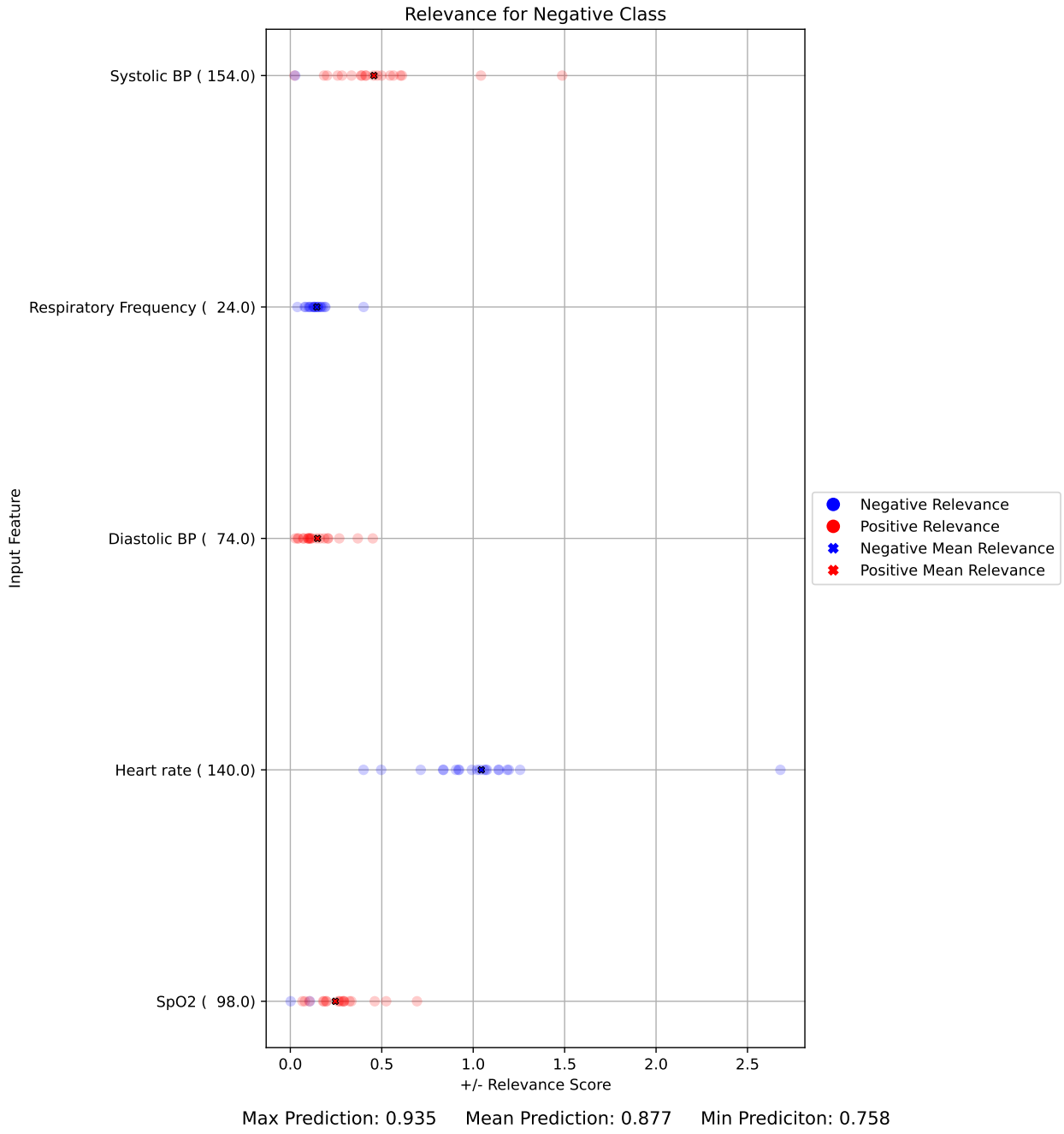


Figure N.30: Neg6(temperature0)False

N.6.7 Pos38(pabChloride0)True

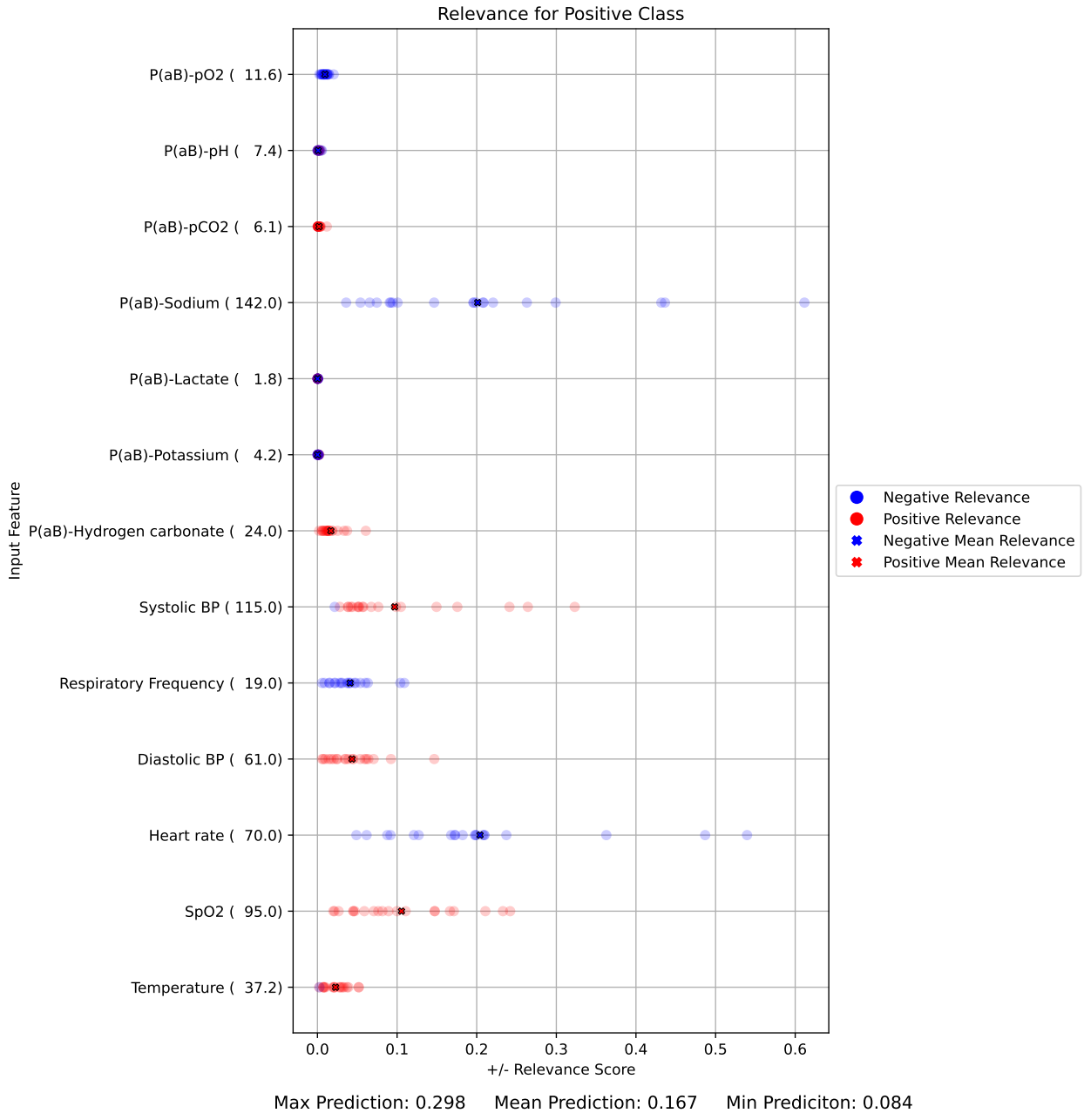


Figure N.31: Pos38(pabChloride0)True

N.6.8 Pos38(pabPotassium0)True

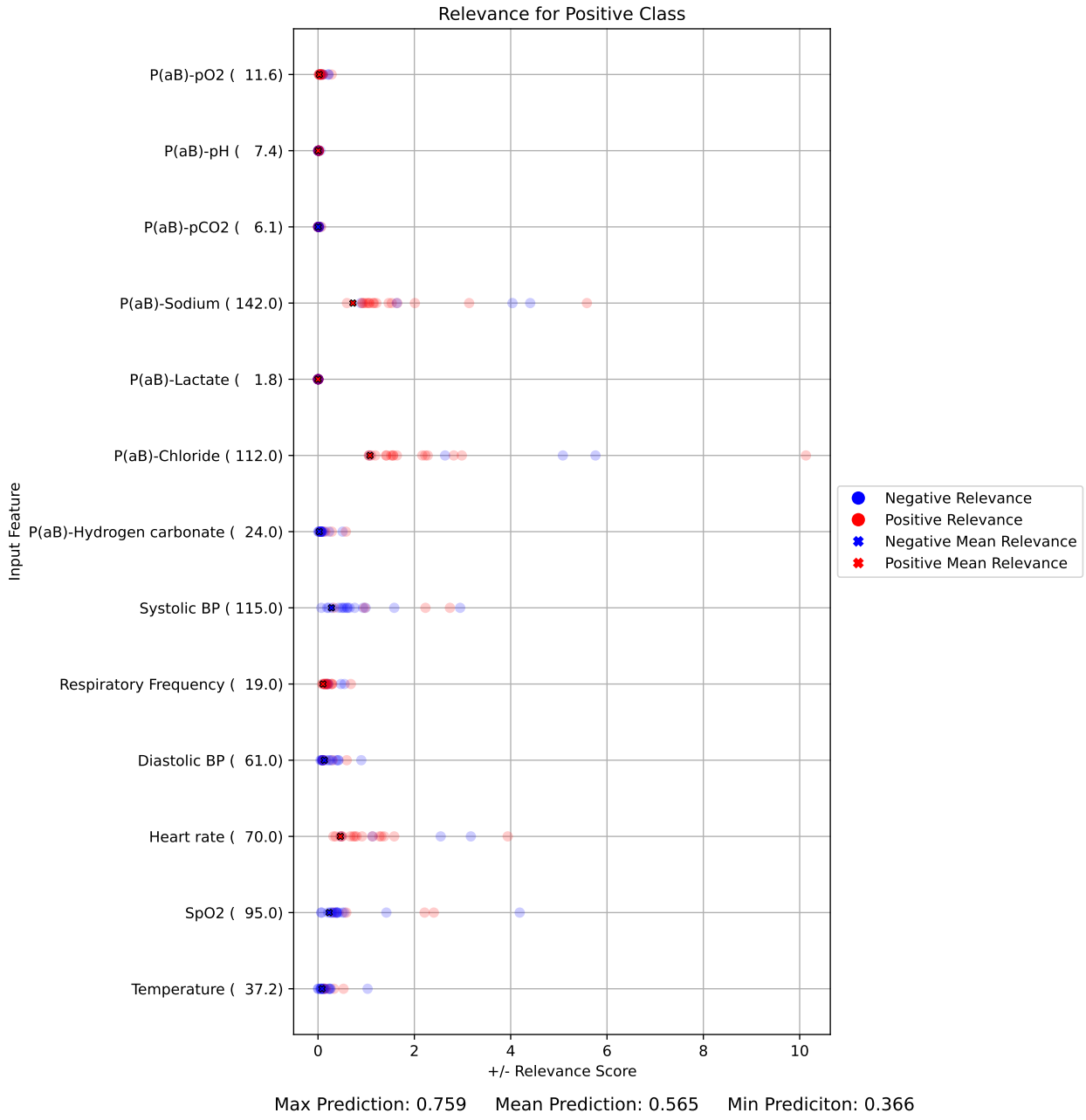


Figure N.32: Pos38(pabPotassium0)True

N.6.9 Pos38(systolicBP0)True

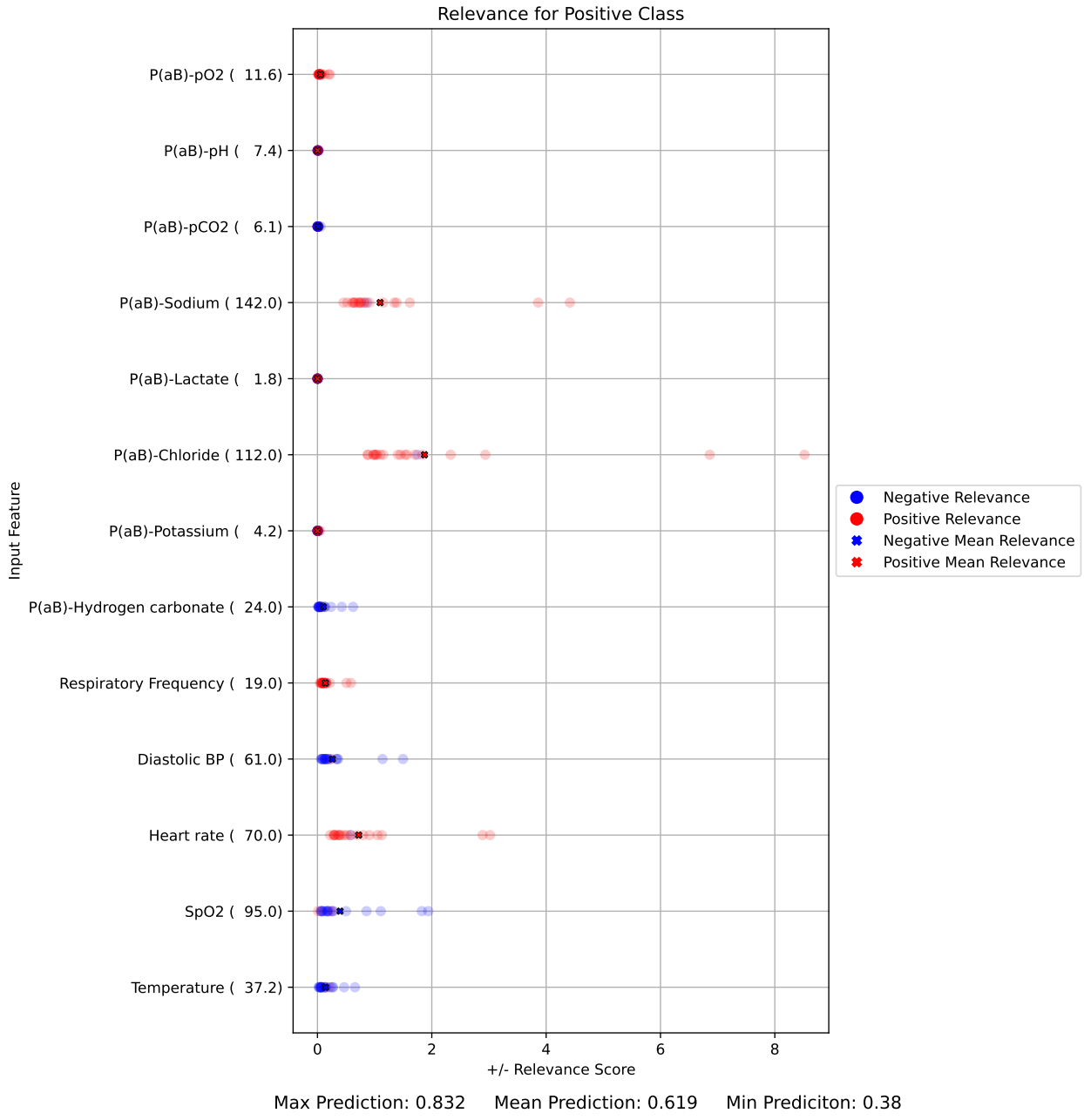


Figure N.33: Pos38(systolicBP0)True

N.6.10 Pos48(bLeukocytes0)True

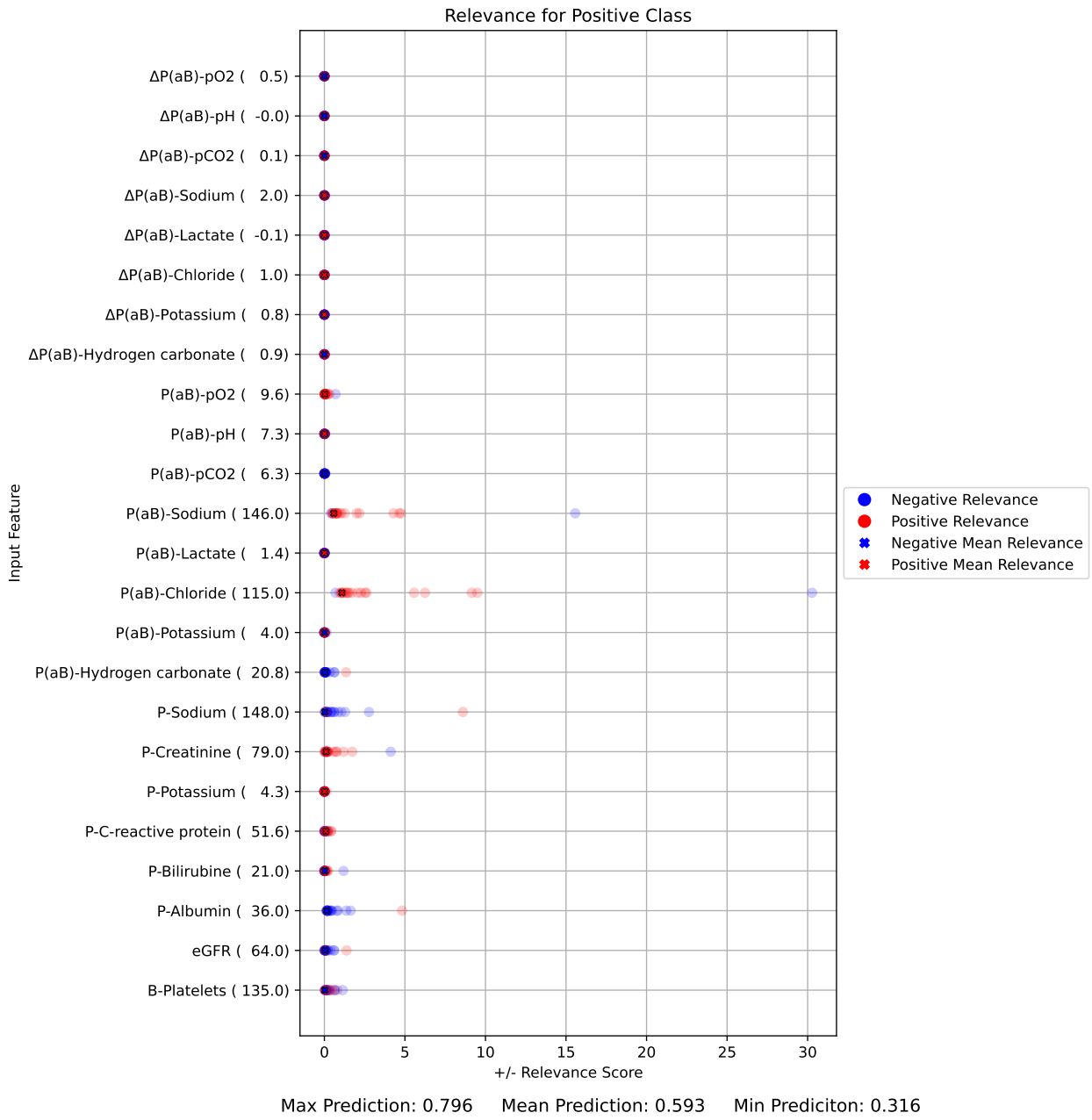


Figure N.34: Pos48(bLeukocytes0)True

N.6.11 Pos48(pabChloride0)True

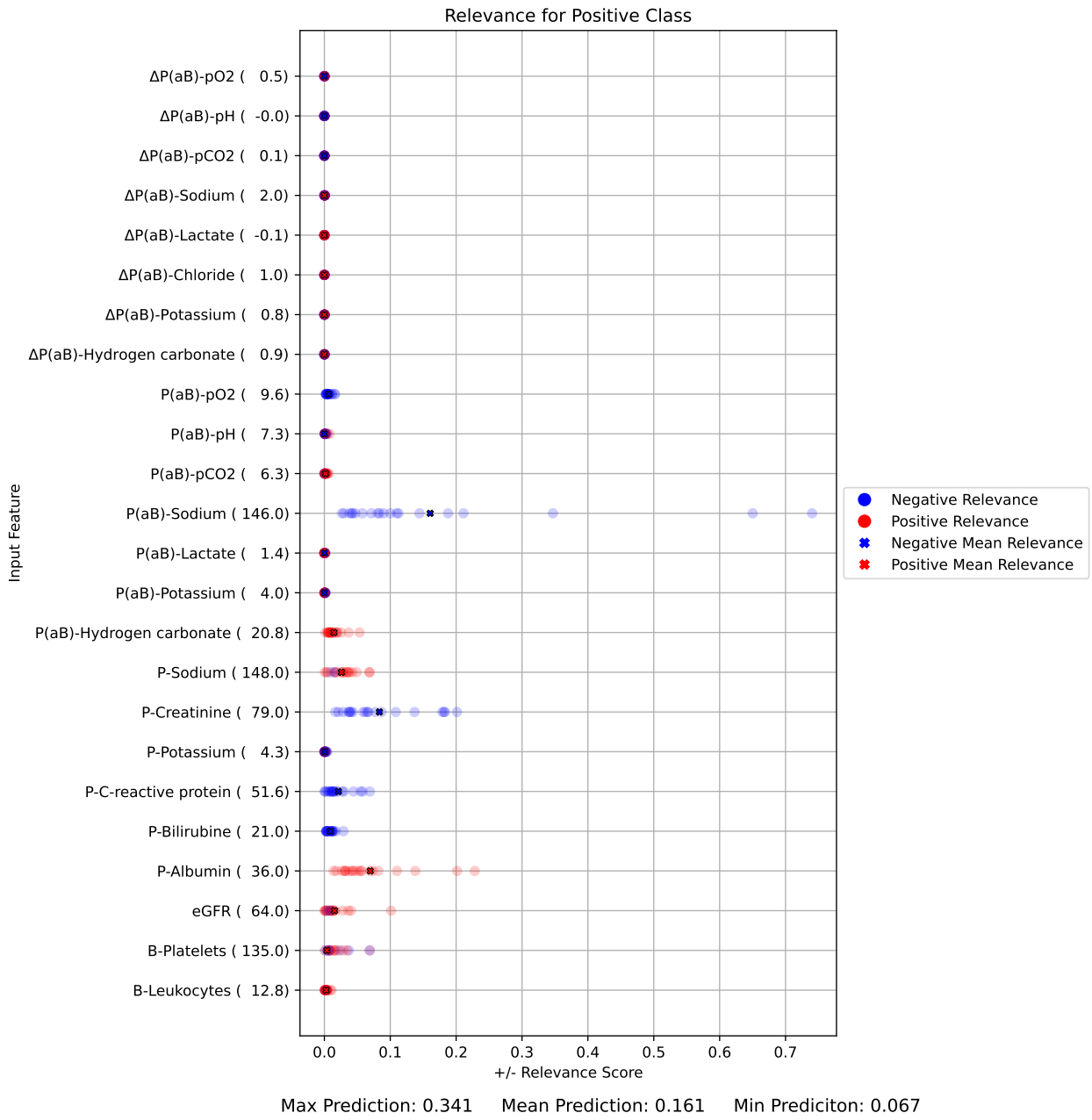


Figure N.35: Pos48(pabChloride0)True

N.6.12 Pos48(pSodium0)True

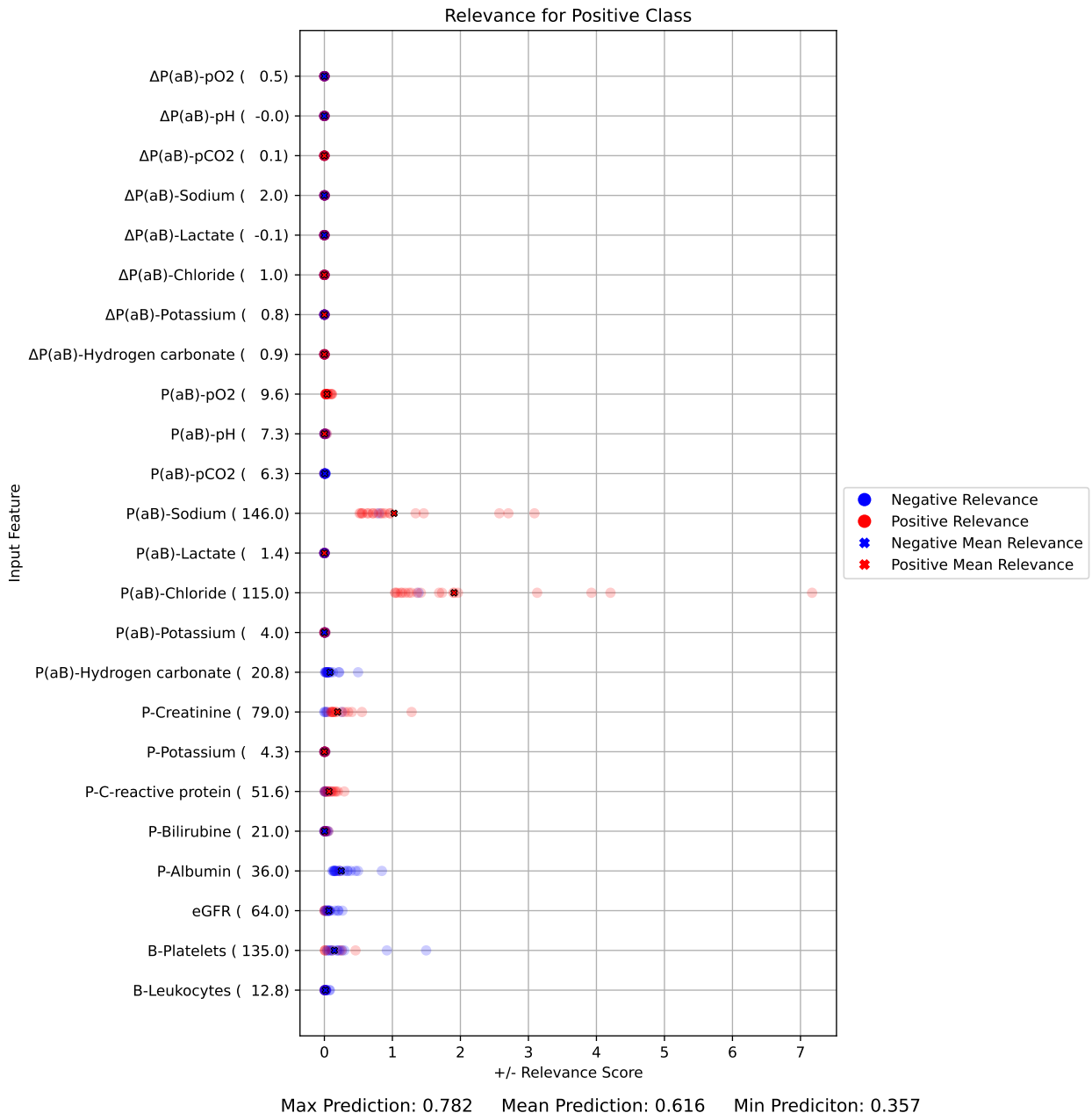


Figure N.36: Pos48(pSodium0)True