Variational oblique predictive clustering trees

Viktor Andonovikj[a,b,*], Sašo Džeroski[a], Biljana Mileva Boshkoska[a,c], Pavle Boškoski[a]

[a] *Jožef Stefan Institute, Jamova cesta 39, 1000 Ljubljana, Slovenia*
[b] *Jožef Stefan International Postgraduate School, Jamova cesta 39, 1000 Ljubljana, Slovenia*
[c] *Faculty of Information Studies in Novo mesto, Ljubljanska cesta 31b, 8000 Novo mesto, Slovenia*

Abstract

Oblique predictive clustering trees (SPYCTs) are semi-supervised multi-target prediction models mainly used for structured output prediction (SOP) problems. They are computationally efficient and when combined in ensembles they achieve state-of-the-art results. However, one major issue is that it's challenging to interpret an ensemble of SPYCTs without the use of a model-agnostic method. We propose variational oblique predictive clustering trees, which address this challenge. The parameters of each split node are treated as random variables, described with a probability distribution, and they are learned through the Variational Bayes method. We evaluate the model on several benchmark datasets of different sizes. The experimental analyses show that a single variational oblique predictive clustering tree (VSPYCT) achieves competitive, and sometimes better predictive performance than the ensemble of standard SPYCTs. We also present a method for extracting feature importance scores from the model. Finally, we present a method to visually interpret the model's decision making process through analysis of the relative feature importance in each split node.

1. Introduction

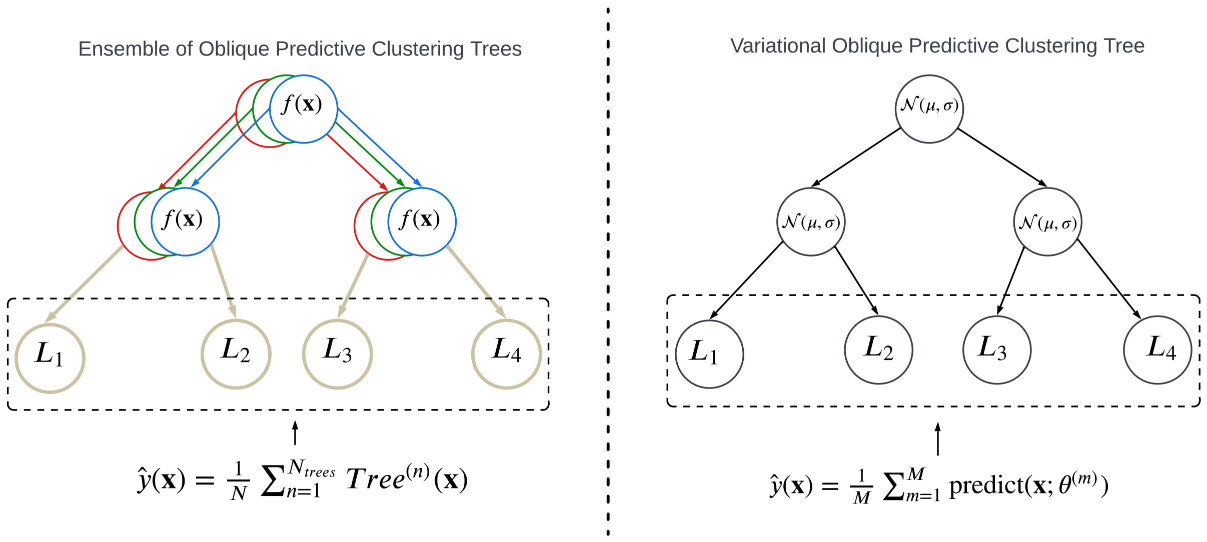
In the evolving field of machine learning, the fusion of semi-supervised learning and decision tree models has led to the development of models like predictive clustering trees (PCTs) (Kocev et al., 2013) and their advanced variant, oblique predictive clustering trees (SPYCTs) (Stepišnik and Kocev, 2021). These models leverage both labeled and unlabelled data to enhance prediction accuracy and efficiency. SPYCTs, in particular, have pioneered the use of oblique splits for structured output prediction (SOP), allowing for more intricate decision boundaries beyond the capabilities of traditional axis-parallel splits. Despite their advantages, a significant challenge with SPYCTs is their reliance on ensemble methods for optimal performance. While effective in improving accuracy, ensembles tend to obscure the model's interpretability—a fundamen-

*Corresponding author.

Email addresses: `viktor.andonovikj@ijs.si` (Viktor Andonovikj), `saso.dzeroski@ijs.si` (Sašo Džeroski), `biljana.mileva@ijs.si` (Biljana Mileva Boshkoska), `pavle.boskoski@ijs.si` (Pavle Boškoski)

tal attribute of decision trees—and do not inherently provide a means to quantify prediction uncertainty, crucial for informed decision-making across various applications.

To address these limitations, we introduce the variational oblique predictive clustering tree (VSPYCT) model. VSPYCT integrates the variational Bayes method (Blei et al., 2017) to facilitate complex decision-making within a singular model framework, thus maintaining the interpretability inherent to decision trees without necessitating ensembles. Furthermore, by embedding model-specific uncertainty quantification directly into the decision tree structure through Bayesian inference, VSPYCT enhances the depth of insight into the model's decision process and confidence levels. By combining the robustness of SPYCT ensembles with the clarity and interpretability of a single tree model, alongside introducing uncertainty quantification, our research provides a comprehensive tool aimed to overcome current limitations of predictive clustering methodologies.



Ensemble of Oblique Predictive Clustering Trees
$f(\mathbf{x})$
$f(\mathbf{x})$
$f(\mathbf{x})$
L_1
L_2
L_3
L_4
$\hat{y}(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N_{trees}} Tree^{(n)}(\mathbf{x})$
Variational Oblique Predictive Clustering Tree
$\mathcal{N}(\mu, \sigma)$
$\mathcal{N}(\mu, \sigma)$
$\mathcal{N}(\mu, \sigma)$
L_1
L_2
L_3
L_4
$\hat{y}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} \text{predict}(\mathbf{x}; \theta^{(m)})$

Figure 1: Ensemble of trees vs. Variational tree

Our proposed VSPYCT framework diverges from traditional ensemble approaches by directly incorporating Bayesian principles into oblique decision trees' architecture. This model avoids conventional ensemble strategies, which aggregate multiple trees' outputs, for a probabilistic treatment of model parameters, offering refined mechanisms for uncertainty quantification and predictive performance. The core of VSPYCT's innovation lies in applying the variational Bayes method for optimising parameters defining the oblique splits, modelling these parameters as random variables with probability distributions. This represents a paradigm shift toward a probabilistic understanding of decision-making processes within tree-based models, significantly enhancing interpretability, especially in uncertainty estimation domains. It sets a new benchmark for interpretable, efficient, and reliable machine learning models.

2. Related work

PCTs have been an essential development in extending decision tree capabilities to a variety of predictive modelling tasks, including structured output prediction. PCTs are versatile and can be combined into ensembles to achieve state-of-the-art performance (Kocev et al., 2013). However, as the dimensionality of the output space increases, the learning time of PCTs scales poorly, posing challenges in tasks like hierarchical multi-label classification where outputs can consist of hundreds of potential labels.

PCTs are celebrated for their interpretability and efficiency, yet they often fall short in predictive performance due to their myopic, greedy nature of selecting splits. The introduction of option predictive clustering trees (OPCTs) (Stepisnik et al., 2020) aimed to address this limitation by incorporating multiple alternative splits within option nodes. This approach mitigates the inherent myopia and achieves competitive performance with ensemble methods like bagging and random forests, while still maintaining a degree of interpretability. OPCTs, despite mitigating myopia and enhancing performance, still face challenges in maintaining interpretability when extended to complex tasks.

On the other hand, SPYCTs (Stepišnik and Kocev, 2021) utilise oblique splits that incorporate linear combinations of features, allowing splits to correspond to arbitrary hyperplanes in the input space. This makes SPYCTs highly efficient for high-dimensional data and capable of leveraging data sparsity. Experimental evaluations on numerous benchmark datasets have shown that SPYCTs achieve performance on par with state-of-the-art methods while being significantly faster than standard PCTs (Andonovikj et al., 2024).

Despite these advancements, ensemble models such as random forests and gradient boosting machines remain popular due to their superior predictive accuracy. However, their complexity often obscures interpretability, making it challenging for users to understand the model's decision-making process. Efforts to bridge this gap include the iForest visual analytics system (Zhao et al., 2019), which summarises decision paths and tree structures within random forests, thereby making the ensemble's decisions more transparent. Additionally, the Tree Space Prototypes approach (Tan et al., 2020) uses representative points (prototypes) for each class, offering a more intuitive understanding of ensemble classifiers compared to traditional feature-based explanations. Visual analytics systems like iForest and prototype-based approaches offer some clarity but do not fully resolve the complexity issue.

In critical applications, such as medical diagnosis, the need for transparent and trustworthy predictions is paramount. Methods for explaining classifier predictions and estimating the reliability of regression predictions, as demonstrated in breast cancer recurrence prediction, provide users with additional insights and build trust by clarifying the decision-making process. Similarly, the MAPLE (Plumb et al., 2018) model combines local linear modelling with a dual interpretation of random forests, offering faithful self-explanations and maintaining high predictive accuracy. MAPLE addresses the accuracy-interpretability trade-off and provides both example-based and local explanations, making it a comprehensive tool for understanding model behaviour. Methods like MAPLE and other interpretability frameworks strike a balance between accuracy and transparency, but they often fall short in handling high-dimensional, sparse data efficiently.

While significant progress has been made in improving decision tree models and their ensembles, several weaknesses remain. Many of these approaches lack inherent mechanisms for uncertainty quantification, which is crucial for applications requiring high reliability and decision certainty.

Our proposed VSPYCT model addresses these critical challenges by integrating Bayesian principles directly into the predictive clustering tree framework. This approach maintains the clarity and simplicity of a single tree model while achieving state-of-the-art performance, thus offering a significant improvement over existing methods. By introducing uncertainty quantification directly into the decision-making process, VSPYCT enhances the reliability and applicability of the model across various domains where decision certainty is crucial.

3. The variational Bayes method

In conducting a stochastic analysis, the process of inferring model parameters hinges on applying Bayes' theorem. For observations x produced by a system defined by parameters θ, Bayes' theorem is given as:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \tag{1}$$

Here, the model-prescribed likelihood $p(x|\theta)$ and the selected prior $p(\theta)$ are generally well-defined. The primary computational challenge arises in determining the posterior distribution $p(\theta|x)$ due to the denominator $p(x)$, which is derived as:

$$p(x) = \int_{\theta} p(x|\theta)p(\theta)\, dx, \tag{2}$$

This integral rarely yields a closed-form solution. For cases with multiple dimensions, the computational burden often renders Monte Carlo methods infeasible. The VB approach offers an approximation to this issue by closely mimicking the true posterior $p(\theta|x)$ with an approximate distribution $q_{\omega^*}(\theta)$. This variational distribution is part of a family of distributions $q_\omega(\theta) \in \mathcal{Q}$, parameterised by $\omega \in \Omega$, where Ω represents the potential latent parameter values. Typically, the mean-field variational family is utilised, assuming independence among the parameters ω (Blei et al., 2017). To find the optimal ω^*, one minimises the Kullback–Leibler (KL) divergence $KL(q_\omega(\theta) \parallel p(\theta|x))$:

$$\omega^* = \underset{\omega \in \Omega}{\arg\min}\, KL(q_\omega(\theta) \parallel p(\theta|x)) \tag{3}$$

Given the unknown nature of the true posterior, a rearrangement is needed to compute the KL divergence, simplifying to (Murphy, 2023, Chapter 10):

$$KL(q_\omega(\theta) \parallel p(\theta|x)) = -\underbrace{\mathbb{E}_q\left[\log p(x,\theta) - \log q_\omega(\theta)\right]}_{\text{ELBO}} + \log p(x)$$

Maximising the Evidence Lower Bound (ELBO) serves to inversely minimise the KL divergence between the approximate posterior $q_\omega(\theta)$ and the true posterior $p_\omega(\theta)$. In the decomposition of the KL divergence, the term $\log p(x)$, known as the marginal likelihood or evidence, remains constant with respect to the variational parameters θ. Since this term does not involve θ, it does not influence the optimisation of the variational distribution and is thus excluded from the ELBO maximisation process. By maximising the ELBO, we aim to tighten the bound provided by the KL divergence, effectively making the variational posterior a better approximation of the true posterior. Typically, the ELBO maximisation criterion is non-convex, with no assurance of global extremum convergence by optimisation algorithms (Murphy, 2023, Chapter 10). Various algorithms have been proposed for this challenge, including variational EM (Bernardo et al., 2003), stochastic variational inference (Hoffman et al., 2013; Sashank et al., 2018), amortised variational inference (Gershman and Goodman, 2014; Le et al., 2017), and semi-amortised inference (Kim et al., 2018).

The adoption of an approximate variational distribution introduces bias contingent on the chosen variational family $\mathcal{Q}$, necessitating a decision grounded in empirical evidence or expert knowledge. This bias arises because the family $\mathcal{Q}$ may not be capable of capturing the true complexity of the posterior distribution $p_\omega(\theta)$. As a result, the accuracy of the approximation depends significantly on how well $\mathcal{Q}$ aligns with the true underlying distribution. The choice of $\mathcal{Q}$ thus not only affects the efficiency of the inference but also the quality and reliability of the model. Despite this bias, VB's computational efficiency significantly surpasses traditional methods like Markov Chain Monte Carlo (MCMC). In our case, the ADAM optimiser (Kingma and Ba, 2014) is used to facilitate the ELBO optimisation.

4. Methodology

As the foundation of the VSPYCT model, the underlying architecture closely follows that of the SPYCT model, particularly in its tree construction process. Similar to SPYCT, VSPYCT constructs a decision tree with an oblique structure, where each split is defined by a linear combination of input features, rather than relying on single-feature thresholds as in axis-aligned trees.

The key distinction between VSPYCT and SPYCT lies in the optimisation step used to determine the parameters of these oblique splits. While SPYCT utilises a deterministic approach to optimise split parameters, VSPYCT introduces a probabilistic framework through variational Bayes (VB) optimisation. In VSPYCT, the parameters of the linear combination at each split—specifically, the weight vector $\mathbf{w}$ and the bias term b—are modelled as random variables with Gaussian distributions. This probabilistic representation of split parameters not only enhances the model's ability to handle uncertainty and noise in the data but also provides a measure of predictive uncertainty.

The construction of the VSPYCT begins with a root node and proceeds iteratively, expanding the tree by adding internal nodes or leaf nodes in a manner identical to SPYCT. Each node in the tree represents a binary decision point, where instances are split based on the oblique hyperplane defined by the linear model:

$$f(\mathbf{x}) = \sigma\left(\mathbf{w}^\top \mathbf{x} + b\right) \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^N$ is the feature vector, $\mathbf{w} \in \mathbb{R}^N$ represents the weight vector, $b \in \mathbb{R}$ is the bias term, and $\sigma(\cdot)$ denotes the sigmoid function, which ensures the output is a probability, determining whether an instance is sent to the right or left child node.

While the tree structure and the decision-making process follow the same principles as in SPYCT, the optimisation of $\mathbf{w}$ and b in VSPYCT differs fundamentally. In VSPYCT, these parameters are not fixed but are instead inferred through variational Bayes optimisation. The introduction of this probabilistic element distinguishes VSPYCT from SPYCT, enabling it to maintain the interpretability of a single decision tree while incorporating the robustness typically associated with ensemble methods.

4.1. Learning splits using Variational Bayes

The goal is to update these prior distributions to approximate the posterior distributions given the observed data, thereby capturing the uncertainty in the model parameters. The weights $\mathbf{w}$ and bias b in the linear model, as in (4), are initially modelled with Gaussian prior distributions:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad b \sim \mathcal{N}(0, 1) \tag{5}$$

The VB method aims to approximate the true posterior distributions of these parameters with a variational distribution $q(\mathbf{w}, b|\mathcal{D})$. We assume a simpler distributional form - a diagonal Gaussian:

$$q(\mathbf{w}, b|\mathcal{D}) = \mathcal{N}(\mathbf{w}|\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)\mathcal{N}(b|\mu_b, \sigma_b^2) \tag{6}$$

where $\boldsymbol{\mu}_w$, $\boldsymbol{\Sigma}_w$, μ_b, and σ_b^2 are the parameters of the variational distribution, learned through the optimisation process.

The optimisation of the variational parameters is performed by maximising the ELBO, which serves as a lower bound to the log marginal likelihood $\log p(\mathcal{D})$. The ELBO is given by:

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathbb{E}_{q(\mathbf{w},b|\mathcal{D})}\left[\log p(\mathcal{D}|\mathbf{w}, b)\right] - KL\left(q(\mathbf{w}, b|\mathcal{D}) \parallel p(\mathbf{w}, b)\right) \tag{7}$$

The first term represents the expected log-likelihood of the data under the variational distribution, while the second term is the Kullback-Leibler (KL) divergence between the variational distribution and the prior.

The ELBO is maximised using the ADAM optimiser, with the goal of refining the variational parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ to closely approximate the true posterior.

The quality of a split is assessed using an impurity function, which is defined as:

$$\text{Impurity} = \sum_{i=1}^{N} \left(\rho_i \cdot \text{Var}_\rho(\mathbf{Y}) + \lambda_i \cdot \text{Var}_\lambda(\mathbf{Y})\right) \tag{8}$$

where $\rho_i = \sigma(\mathbf{w}^\top \mathbf{x}_i + b)$ is the probability of an instance being routed to the right child node, and $\lambda_i = 1 - \rho_i$ is the probability of routing to the left child node. The variances $\text{Var}_\rho(\mathbf{Y})$ and $\text{Var}_\lambda(\mathbf{Y})$ represent the variance of the target variables in the right and left child nodes, respectively, weighted by their selection probabilities.

The objective is to minimise the impurity by optimising the parameters $\mathbf{w}$ and b using the VB method, thereby ensuring that the split at each node effectively separates the data into homogeneous subsets.

The overall procedure for learning a split in VSPYCT is detailed in Algorithm 1. The algorithm involves initialising the variational parameters, performing Monte Carlo sampling to estimate the ELBO, and iteratively updating the parameters until convergence.

4.2. Making a prediction

The procedure for making a prediction for a new instance using the VSPYCT model is detailed in Algorithm 2. The process involves traversing the tree from the root node to a leaf node, with decisions at each internal node being made probabilistically based on the parameters sampled from the learned variational posterior distributions. The final prediction is determined by the prototype value stored in the reached leaf node.

Mathematically, at each internal node, the oblique split is determined by evaluating the linear combination of input features, with the parameters $\Theta = (\mathbf{w}, b)$ drawn from their respective variational posterior distributions. The decision to branch left or right is based on the probability computed using the sigmoid function applied to this linear combination. This method accounts for the uncertainty in the model parameters, providing a probabilistic framework for decision-making.

The prediction $\hat{y}$ returned by the model is the prototype of the reached leaf node, where the prototype is defined as the mean of the target values associated with the training instances that were routed to that leaf during training.

$$\hat{y} = \frac{1}{|\mathbf{Y}|} \sum_{i=1}^{|\mathbf{Y}|} y_i \tag{9}$$

where y_i represents the target value of the i-th instance in $\mathbf{Y}$. This value $\hat{y}$ serves as the prediction for any new instance that reaches the particular leaf node.

Algorithm 1 Variational Learning of Split Parameters in VSPYCT with Impurity Minimization

```
1: Input: $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ (dataset), $\theta = \{\mathbf{w}_0, b_0\}$ (initial parameters), $E$ (epochs), $\lambda$ (learning rate), $\beta$ (batch size), $\sigma$ (selection probability)
2: Output: $\Theta = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \mu_b, \sigma_b^2\}$ (variational parameters)
3: procedure LearnSplit($\mathcal{D}, \theta, E, \lambda, \beta, \sigma$)
4:     Initialise $\Theta = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w, \mu_b, \sigma_b^2\}$          ▷ Initialise variational parameters
5:     Initialise optimiser: $\alpha \leftarrow \lambda$
6:     for $e \in \{1, \ldots, E\}$ do          ▷ Iterate over epochs
7:         for each mini-batch $\mathcal{B} \subseteq \mathcal{D}$ of size $\beta$ do
8:             Sample $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, $b \sim \mathcal{N}(\mu_b, \sigma_b^2)$
9:             Compute impurity $\Omega(\mathbf{X}, \mathbf{Y}; \mathbf{w}, b)$ as:
                   $$\Omega(\mathbf{X}, \mathbf{Y}; \mathbf{w}, b) = \sum_{\text{right}} \rho(\mathbf{w}^\top \mathbf{x} + b) \cdot \mathrm{Var}_\rho(\mathbf{Y}) + \sum_{\text{left}} (1 - \rho(\mathbf{w}^\top \mathbf{x} + b)) \cdot \mathrm{Var}_\lambda(\mathbf{Y})$$
10:            Set the observed impurity value: $\Omega_{\text{obs}} = \frac{\Omega(\mathbf{X}, \mathbf{Y}; \mathbf{w}, b)}{2}$
11:            Condition on the observed impurity using:
                   $$\log p(\Omega_{\text{obs}} | \Omega(\mathbf{X}, \mathbf{Y}; \mathbf{w}, b))$$
12:            Compute ELBO $\mathcal{L}(\mathcal{B}; \Theta)$ as per (7)
13:            Compute gradient $\nabla_\Theta$ELBO
14:            Update $\Theta \leftarrow \Theta + \alpha \nabla_\Theta$ELBO          ▷ Gradient descent update
15:        end for
16:        if early stopping criterion satisfied then
17:            break
18:        end if
19:    end for
20:    return $\Theta$
21: end procedure
```

4.3. Time complexity analysis

We analyse the time complexity of learning a split in VSPYCT and compare it with SPYCT. Our analysis focuses on the computational expense associated with determining a split, as this represents the principal distinction between the two approaches. In the predictive clustering framework, the set of clustering attributes K, which contributes to the variance calculation, encompasses both the features and the target variables. As a result, K equals $D + T$, with D representing the feature count and T denoting the target

Algorithm 2 Prediction Process in VSPYCT using Monte Carlo Sampling

```
1: Input: Feature vector $\mathbf{x}$, decision tree $T$ rooted at node $\Omega$, number of samples $M$
2: Output: Prediction $\hat{y}$
3: function MC_Predict($\Omega$, $\mathbf{x}$, $M$)
4:     Initialise $\hat{y}_{\text{sum}} = 0$
5:     for $m = 1$ to $M$ do
6:         $\hat{y}_{\text{sum}} \leftarrow \hat{y}_{\text{sum}} + \text{Predict}(\Omega, \mathbf{x}, m)$
7:     end for
8:     return $\hat{y} = \frac{\hat{y}_{\text{sum}}}{M}$
9: end function
10: function Predict($\Omega$, $\mathbf{x}$, $m$)
11:     if $\Omega$ is a leaf node then
12:         return $\hat{y}_\Omega = \frac{1}{|\mathcal{D}_\Omega|} \sum_{i \in \mathcal{D}_\Omega} y_i$
13:     else
14:         Sample $\Theta^{(m)} = (\mathbf{w}^{(m)}, b^{(m)})$ from $q(\mathbf{w}, b|\mathcal{D})$
15:         Compute $\rho(\Theta^{(m)}, \mathbf{x}) = \sigma(\mathbf{w}^{(m)\top}\mathbf{x} + b^{(m)})$
16:         if $\rho(\Theta^{(m)}, \mathbf{x}) \leq 0.5$ then
17:             return Predict($\Omega_{\text{left}}$, $\mathbf{x}$, $m$)
18:         else
19:             return Predict($\Omega_{\text{right}}$, $\mathbf{x}$, $m$)
20:         end if
21:     end if
22: end function
23: $\hat{y} \leftarrow$ MC_Predict($\Omega$, $\mathbf{x}$, $M$)
24: Return $\hat{y}$
```

count. The time complexity for learning in SPYCTs can be expressed as $\mathcal{O}(NI_o(D+K))$, wherein N is the number of training instances, and I_o is the iteration count needed for optimisation (Stepišnik and Kocev, 2021). Notably, the computational demand for the SPYCTs's gradient-based version increases linearly with the attribute count.

The introduction of variational Bayes method in the architecture of the model adds a new layer of computational complexity. Specifically, the complexity of learning a split in VSPYCT is influenced by the following factors: the need to sample parameter sets from the approximative posterior distributions, the iterative optimisation required to refine these estimates towards minimising the ELBO, and the dimensionality imposed by the number of features. The combined computational complexity for learning a split in

VSPYCT can thus be expressed as $\mathcal{O}(MNI_{vb}(D+K))$, where:

- M represents the number of Monte Carlo samples needed to approximate the posterior distributions adequately.
- N is the number of data points evaluated during each iteration.
- D indicates the number of features, each contributing to the parameter space that must be sampled and optimised.
- K indicates the number of clustering attributes, used to calculate the variance in each of the node.
- I_{vb} indicates the number of iterations required for the variational inference process to achieve convergence.

This complexity reflects the multiplicative impact of sampling, feature dimensionality, and iterative optimisation, highlighting how the variational approach scales with the size and feature richness of the dataset.

Comparing VSPYCTs to SPYCTs, the VB method increases the computational load due to the necessity of Monte Carlo sampling and iterative ELBO optimisation. However, this additional complexity is counterbalanced by the enhanced model interpretability, and uncertainty quantification offered by VSPYCTs.

4.4. Feature Importance

The procedure for computing feature importance scores in VSPYCT is inspired by the methodology introduced in the original SPYCT model (Stepišnik and Kocev, 2021), while extending it to incorporate the model's probabilistic nature. Let $\mathbb{E}[\mathbf{w}_s]$ and $\mathrm{Var}(\mathbf{w}_s)$ denote the element-wise mean and variance of the weights at node s. We define:

$$\mathbf{u}_s = \frac{\mathbb{E}[\mathbf{w}_s]}{\mathrm{Var}(\mathbf{w}_s) + \epsilon}$$

where ϵ is a small constant added for numerical stability.

The feature importance vector $\mathrm{Imp}(T)$ is then computed as:

$$\mathrm{Imp}(T) = \sum_{s \in T} \left(\frac{s_n}{N}\right) \cdot \frac{|\mathbf{u}_s|}{\|\mathbf{u}_s\|_2} \tag{10}$$

Here, $\frac{s_n}{N}$ weights the contribution of node s by the fraction of training samples passing through it, and $\|\mathbf{u}_s\|_2$ is the L2 norm of $\mathbf{u}_s$. This formulation integrates both the magnitude and the uncertainty of the weight parameters, potentially yielding a more robust measure of feature importance under a probabilistic model. The vector $\mathbf{w}_s$ specifies the weights that define the oblique split at node s. The expected value of the weights, $\mathbb{E}[\mathbf{w}_s]$, is computed through Monte Carlo simulation, where weights are sampled M times from their approximation of the posterior distribution, and the mean of these samples is taken as the representative value for each weight. This integrates the stochastic nature of the model parameters into the calculation of feature importance. The repeated sampling and averaging process ensures that the estimated importance reflects both the central tendency and the variability of the weight parameters, crucial for interpreting the model's decision-making process under uncertainty.

Toy Example. Consider a simple dataset where two features, *age* and *salary*, predict the likelihood of purchasing a car. In a VSPYCT model, the influence of each feature at a node depends not only on the magnitude of its mean weight but also on how this mean compares to the feature's weight variance.

Suppose that at the root node, the mean weights associated with *age* and *salary* are $\mathbb{E}[w_{\text{age}}] = 0.2$ and $\mathbb{E}[w_{\text{salary}}] = 0.8$, and their respective variances are $\text{Var}(w_{\text{age}})$ and $\text{Var}(w_{\text{salary}})$. The importance at this node will now reflect the ratio $\frac{\mathbb{E}[w]}{\text{Var}(w)}$, such that a feature with a relatively high mean weight but also a high variance will not necessarily dominate one with a slightly lower mean but substantially more stable (lower variance) weights. Thus, if *salary* maintains a higher $\frac{\mathbb{E}[w]}{\text{Var}(w)}$ ratio than *age*, it indicates that *salary* is not only influential but also more reliably so at this particular node.

However, the overall feature importance is derived by aggregating these normalised ratios across all splits within the tree. Even if *salary* appears strongly influential at the root node, the final importance calculation considers both the magnitude and the stability of the weights across all nodes. A feature consistently demonstrating a favourable $\frac{\mathbb{E}[w]}{\text{Var}(w)}$ ratio throughout multiple splits will be deemed more important globally. In this way, the final importance assessment accounts for both the strength and certainty of each feature's contribution to the model's decision-making process.

5. Experimental setting

The experimental evaluation was conducted to compare the VSPYCT model with several benchmarks, including the SPYCT model, option predictive clustering trees (OPCTs) (Stepisnik et al., 2020), and ensemble of PCTs. Specifically, we considered both a single SPYCT and an ensemble of SPYCTs, as well as a single OPCT and an ensemble of PCTs. We chose these methods as primary comparisons because they have been extensively evaluated in the literature, demonstrating strong performance across various SOP tasks (Andonovikj et al., 2024; Stepišnik and Kocev, 2021).

The comparison encompassed a range of predictive modelling tasks, including single-target regression, multi-target regression, binary classification, and multi-class classification. By evaluating not only a single SPYCT tree, but also an ensemble of SPYCTs, a single OPCT tree, and an ensemble of PCTs, we aimed to assess how VSPYCT's predictive performance compares against a spectrum of models, from single decision trees to more complex ensembles and option-based trees. This comprehensive approach allowed us to rigorously evaluate VSPYCT's effectiveness across different modelling scenarios and relative to both well-established baseline models and state-of-the-art tree-based methods.

5.1. Data

The experimental evaluation was performed on a diverse collection of datasets, each reflecting a distinct predictive modelling task: single-target regression (STR), multi-target regression (MTR), binary classification (BC), and multi-class classification (MCC). The properties of these benchmark datasets are detailed in Tables 3 through 4, respectively. Each table lists the number of examples (N) and the number of features (D) for all datasets, along with the number of targets (T) for MTR datasets and the number of classes (C) for MCC datasets.

Table 1: Properties of the benchmark STR datasets

Dataset	**N**	**D**
era (OpenML)	1000	5
cpmp (OpenML)	2108	27
qsar234 (OpenML)	2145	1026
concrete_compressive_strength (OpenML)	1030	9
yprop (OpenML)	8885	252
puma8NH (OpenML)	8192	8
spacega (OpenML)	3107	6
bike_sharing (OpenML)	17379	6
quake (OpenML)	2178	3
ailerons (OpenML)	13750	40

In STR tasks, the output is a single continuous variable, while in MTR tasks, the output comprises multiple continuous target variables. BC tasks involve two possible classes, and MCC tasks extend to multiple classes, with the exact number of classes depending on the specific dataset. This comprehensive range of datasets ensures a robust evaluation of the models across varying complexity and output structures.

5.2. Evaluation

The performance of the models was evaluated using appropriate metrics for each predictive modelling task. For single-target regression, the Mean Absolute Error (MAE) was employed as the evaluation metric:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

where y_i denotes the true value, $\hat{y}_i$ is the predicted value, and N is the total number of instances. A lower MAE indicates better predictive accuracy.

Table 2: Properties of the benchmark MTR datasets

Dataset	**N**	**D**	**T**
wq (OpenML)	1060	16	14
atp1d (OpenML)	337	411	6
atp7d (OpenML)	296	411	6
scpf (OpenML)	1137	23	3
osales (OpenML)	639	411	12
sf1 (OpenML)	323	10	3
sf2 (OpenML)	1066	10	3
rf1 (OpenML)	9125	64	8
rf2 (OpenML)	9125	576	8
slump (OpenML)	103	7	3

Table 3: Properties of the benchmark BC datasets

Dataset	**N**	**D**
scene (Mulan)	2407	299
ova_breast (OpenML)	1545	10935
ova_lung (OpenML)	1545	10935
bio_response (OpenML)	3751	1777
chronic_kidney_disease (OpenML)	400	25
compas_two_years (OpenML)	5278	13
spambase (OpenML)	4601	57
gina_agnostic (OpenML)	3468	970
credit-g (OpenML)	1000	20
diabetes(OpenML)	768	8

For multi-target regression, a normalised Mean Absolute Error ($NMAE$) was used to ensure comparability across targets with different value ranges. Letting y_{ij} and $\hat{y}_{ij}$ be the true and predicted values for the j-th target of the i-th instance, and T be the total number of target variables, the $NMAE$ is defined as:

$$NMAE = \frac{1}{T}\sum_{j=1}^{T}\left(\frac{\frac{1}{N}\sum_{i=1}^{N}|y_{ij}-\hat{y}_{ij}|}{\max_i(y_{ij})-\min_i(y_{ij})}\right)$$

If $\max_i(y_{ij}) - \min_i(y_{ij}) = 0$ for a given target j, this denominator is replaced by 1, effectively leaving that target unscaled.

Table 4: Properties of the benchmark MCC datasets

Dataset	**N**	**D**	**C**
amazon_reviews (OpenML)	1500	10000	50
dermatology (OpenML)	366	34	6
micro_mass (OpenML)	360	1300	10
balance (OpenML)	625	4	3
yeast (Mulan)	1484	8	10
wine (OpenML)	178	13	3
mfeat_zernike (OpenML)	2000	47	10
har (OpenML)	10299	561	6
gas_drift (OpenML)	13910	128	6
semeion (OpenML)	1593	257	10

A lower *NMAE* indicates better predictive performance, as it accounts for the range of each target variable to produce a comparable error measure across multiple targets.

For binary and multi-class classification tasks, the *F1* score was used to measure model performance. In the case of multi-class classification, the *F1* score was macro-averaged across all classes. A higher *F1* score indicates better classification performance, as it reflects a balance between precision and recall.

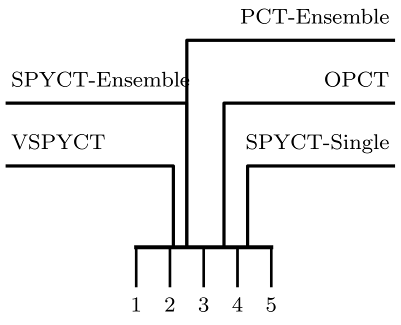
All evaluations were conducted using 5-fold cross-validation, to ensure a comprehensive assessment of the model's performance across different subsets of the data, reducing the potential for bias and overfitting.

6. Results

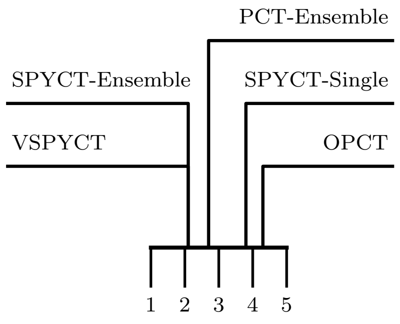
Figure 2 shows the average rankings of the evaluated methods—VSPYCT, OPCT, single SPYCT, ensemble of SPYCTs, and ensemble of PCTs—across four task types: binary classification (BC), multi-class classification (MCC), single-target regression (STR), and multi-target regression (MTR). For improved readability, the actual performance of the models on each dataset is given in the Appendix.

In BC tasks, VSPYCT ranks first, outperforming ensembles of SPYCTs and PCTs, which tie for second. This suggests that VSPYCT's probabilistic modeling effectively handles simple binary boundaries, surpassing both ensemble variance reduction and structural options (OPCT). In MCC tasks, VSPYCT and the ensemble of SPYCTs lead, followed by the ensemble of PCTs, single SPYCT, and OPCT, indicating that VSPYCT's uncertainty-aware splits can handle more complex class structures as well as ensembles can.

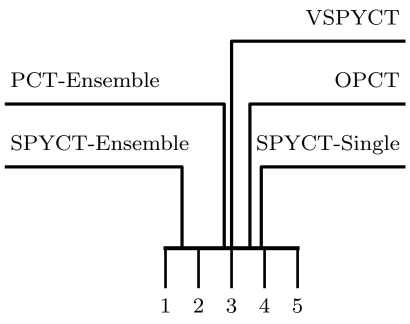
For STR tasks, both ensemble-based approaches—those combining multiple SPYCTs and those combining multiple PCTs—achieve the highest ranks, followed by VSPYCT. It can be seen that ensembles appear



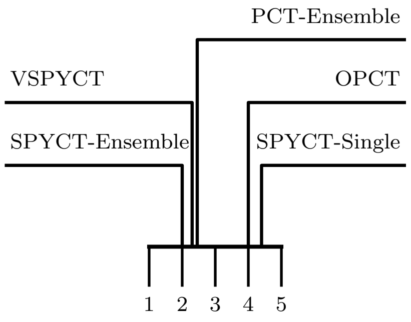
(a) BC



(b) MCC



(c) STR



(d) MTR

Figure 2: Average rank of predictive performance. Lower ranks indicate better performance.

particularly effective for precise single-target continuous predictions. However, in MTR scenarios, VSPYCT surpasses the ensemble of PCTs, indicating that its probabilistic representation of oblique splits can better capture complex inter-target relationships. By contrast, OPCT and single SPYCT lag behind, indicating that their individual enhancements—option nodes or a single-tree oblique structure—do not offer sufficient complexity or flexibility to match the performance gains achieved by either ensembles or the probabilistic modelling of VSPYCT.

6.1. Discussion

The results underscore the interplay between model complexity, uncertainty quantification, and variance reduction. Ensemble methods consistently perform well in scenarios demanding high precision (e.g., STR) or broad generalisation over multiple outputs (MTR). VSPYCT, capitalises on its Bayesian inference framework to excel in tasks where modeling uncertainty and complex decision boundaries are crucial (e.g., BC and MCC), and remains competitive even in more challenging regression contexts.

These findings underscore that VSPYCT is not a universal replacement for ensembles, but rather a complementary alternative whose benefits depend on the task at hand. When dealing with classification problems—binary or multi-class—where interpretability and uncertainty quantification are critical, VSPYCT offers a potent blend of predictive performance, clarity, and inherent uncertainty modeling. Its capability to handle intricate class boundaries and its flexibility in representing parameter uncertainties make it a valuable method for domains demanding transparency and robustness.

In continuous-output scenarios, ensembles still exhibit an edge, suggesting that if raw predictive precision is paramount and interpretability or uncertainty modeling are secondary considerations, ensemble models may remain the preferred choice. Nevertheless, VSPYCT provides a single-tree model that narrows the performance gap considerably and is particularly appealing when balancing predictive accuracy with the need for more direct insight into the model's decision process.

6.2. Feature importance scores

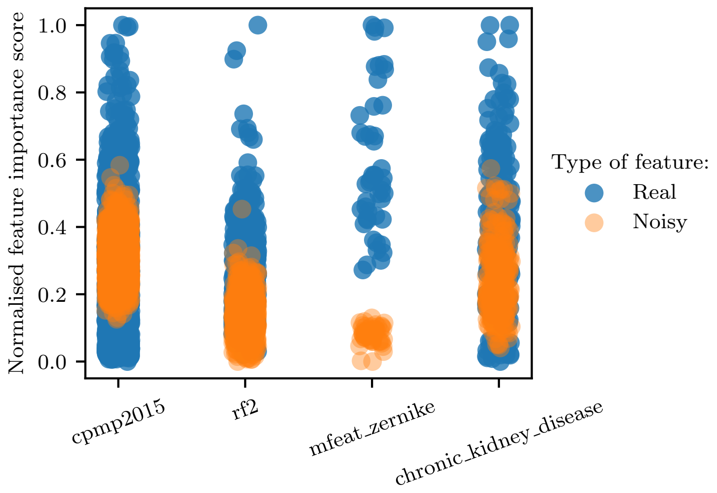


Figure 3: Feature importance scores obtained on a single dataset from per category. The results indicate that the model does not depend on irrelevant features.

As illustrated in Figure 3, the proposed VSPYCT model yields meaningful feature importance scores. To rigorously evaluate this property, we introduced synthetic noise features into the datasets before model training. Let the original dataset contain d features. We then augmented it by adding d additional features, each generated by random noise. Thus, the total number of features in the augmented dataset is $2d$.

The resulting feature importance scores reveal that the original (real) features exhibit a broad range of importance values. This heterogeneity is expected, as real-world datasets often comprise both highly informative variables and those with negligible predictive power. In contrast, the artificially introduced noise features consistently received low importance scores. Formally, if we let $\mathcal{F} = \{f_1, f_2, \ldots, f_d\}$ denote the set of original features, and $\mathcal{F}' = \{f_{d+1}, \ldots, f_{2d}\}$ the set of noise features, the estimated importances $I(f_j)$ for $f_j \in \mathcal{F}'$ are significantly lower than those for at least a subset of $\mathcal{F}$. This indicates that the

VSPYCT-derived importance measures are not only able to identify relevant features but also robust to the inclusion of spurious, non-informative features.

6.3. Interpretability

The dataset used for the interpretability analysis consists of $74,086$ anonymised instances of jobseekers from the Slovenian Public Employment Service (PES). We will refer to it as the Unemployment dataset. It includes a variety of personal and professional characteristics, such as age, gender, education, and work experience. The target variable is the time until the jobseeker either becomes employed or exits the study, which is measured in days. The dataset is challenging due to the different forms of its attributes (categorical, numerical, temporal) and the presence of censored data, where some jobseekers' outcomes are not fully observed.

The goal is to predict the time-to-event (employment) for jobseekers, with some records being right-censored—meaning the exact time of the event is not observed. Each data instance is described by features $\mathbf{X}_i$, observed time t_i, and a censoring indicator δ_i, where $\delta_i = 1$ indicates the event was observed, and $\delta_i = 0$ indicates a right-censored record. This setup, with its inherent missing data due to censoring, makes it suitable for semi-supervised learning approaches. Detailed information on the dataset and the semi-supervised multi-target regression framework can be found in Andonovikj et al. (2024).

Figure 4 presents the survival curve predicted by the VSPYCT model for a specific jobseeker in the Unemployment dataset. The curve shows the estimated probability of remaining unemployed over time, with the blue line representing the mean prediction. The shaded area around the curve illustrates the uncertainty in the prediction, captured as a confidence interval between the 10th and 90th percentiles. This visualisation effectively communicates the model's predictions along with its confidence, providing valuable insights into the jobseeker's likelihood of finding employment over time. The dashed vertical line marks the actual observed time of employment, allowing for an immediate comparison between the predicted and actual outcomes.

Figure 5 highlights the feature importance scores obtained from the VSPYCT model, focusing on the most influential features that contribute to predicting the unemployment duration. The horizontal bars show the relative importance of each feature, with *Months of work experience* emerging as the most critical factor, followed by *Age* and *Entry month.*

Figure 6 presents the visual representation of the VSPYCT applied to the Unemployment dataset. Based on that, one can distill a concise interpretation enriched with practical insights.

At the heart of this VSPYCT model lies the ability to visually demarcate the influence of various feature categories on unemployment duration predictions. The tree's root node is predominantly influenced by features categorised under "other" (represented in green), encompassing age, months of work experience,

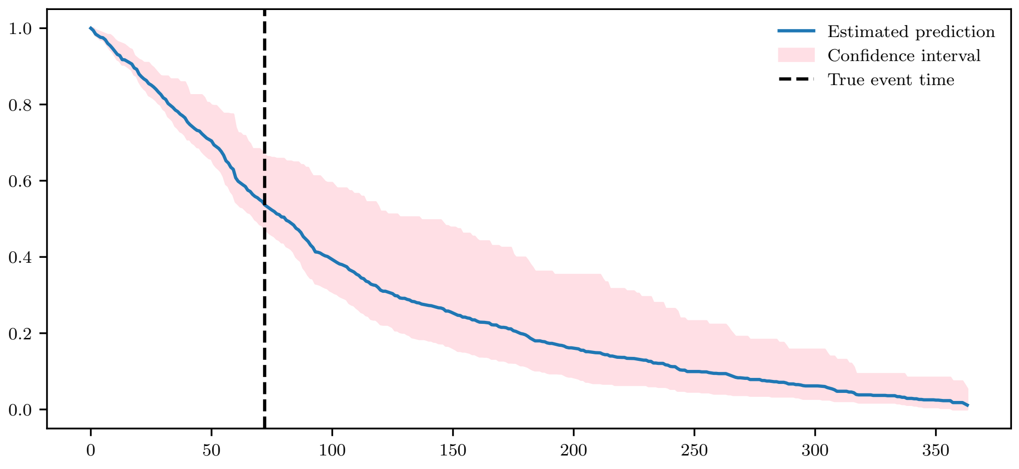


Figure 4: Survival curve predicted by the VSPYCT model on the Unemployment dataset. The blue line represents the estimated mean survival time, while the shaded area indicates the confidence interval (10th to 90th percentile) around the prediction. The vertical dashed line marks the actual time of the observed event.

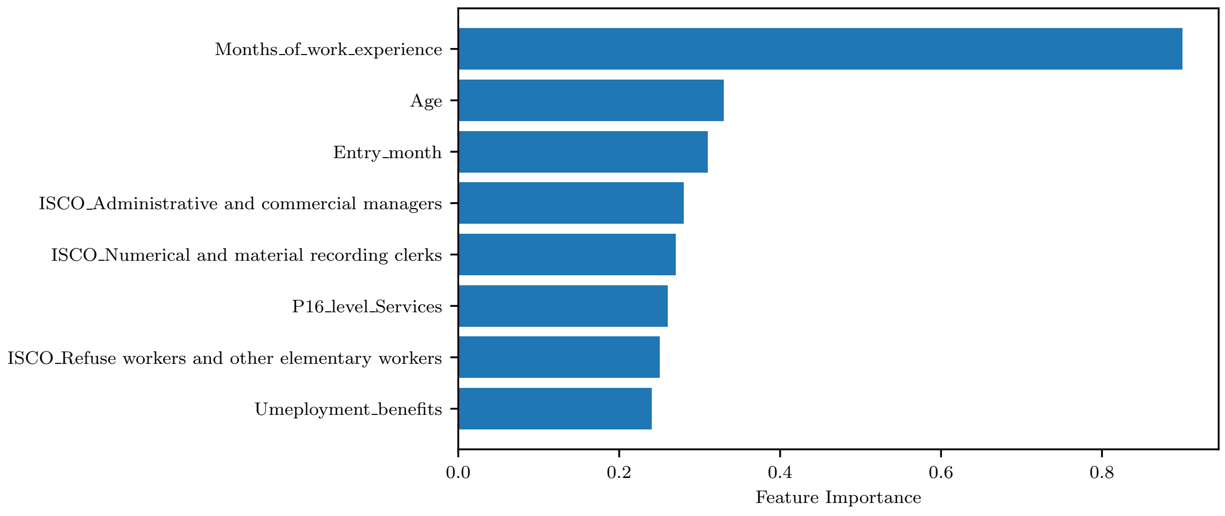


Figure 5: Feature importance scores extracted from the VSPYCT model applied to the Unemployment dataset. The horizontal bars represent the relative importance of each feature, with the most influential feature being "Months of work experience."

and various demographic characteristics. This indicates that such features play a pivotal role in initial split decisions, underscoring the broad impact of demographic and experiential factors on unemployment trends.

As we navigate through the tree, a dichotomy emerges based on the expected survival times—essentially, the projected duration an individual is likely to remain unemployed. On branches leading to higher expected survival times, features categorised under "p16 level of education" (coloured in blue) become increasingly

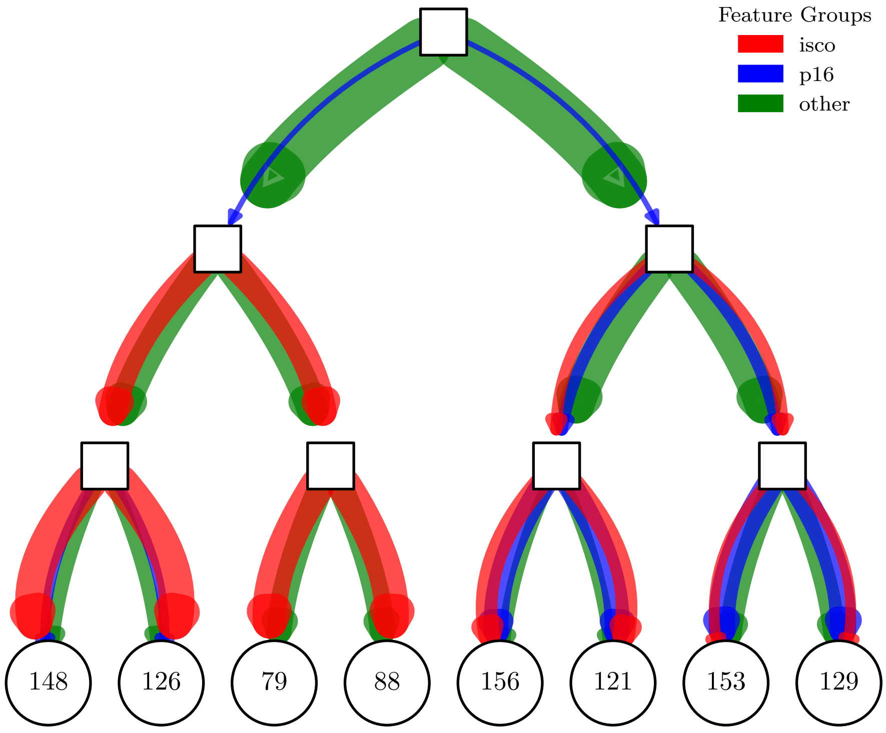


Figure 6: The VSPYCT model applied to the Unemployment dataset. The colouring distinguishes the different feature groups of the data set

dominant. This suggests that higher levels of education are associated with longer periods of unemployment, potentially reflecting challenges in finding employment that matches an individual's qualifications or aspirations.

Conversely, on branches associated with lower expected survival times, the dominance shifts towards features represented by the "ISCO classification of professional occupation" (coloured in red). This dominance implies that certain professional occupations, as defined by the ISCO classification, may expedite reemployment, possibly due to higher demand or greater flexibility in job roles within these sectors.

This visual breakdown not only highlights the model's interpretability but also offers tangible insights into how different factors—ranging from personal demographics and work experience to education and professional classifications—impact unemployment durations. Such insights are invaluable for policymakers and stakeholders aiming to devise targeted interventions that address specific barriers to employment faced by diverse demographic groups.

In essence, the VSPYCT model's tree structure, with its intuitive colour-coded feature categories and quantified predictions at the leaf nodes, offers a powerful tool for understanding the multifaceted nature of unemployment. It bridges the gap between complex machine learning models and practical socio-economic applications, providing a clear, actionable framework for addressing unemployment through informed, data-driven strategies.

7. Conclusion

We introduced the VSPYCT model, an approach that combines the benefits of oblique splits, variational Bayes, and Bayesian inference within a single, interpretable predictive framework. The VSPYCT model addresses the trade-off between the performance of ensemble models and the interpretability of single decision trees, and the absence of inherent uncertainty quantification in predictions. Our results demonstrate that the VSPYCT model is competitive to, and on some datasets exceeds the performance of ensemble of SPYCTs, without sacrificing the model's interpretability. By employing variational Bayes method, we introduce a mechanism for uncertainty quantification directly into the decision-making process, enhancing the reliability and applicability of the model across various domains where decision certainty is paramount. The model opens new possibilities for research in semi-supervised learning and Bayesian inference, promising to bridge the gap between theoretical advancements and practical applications. Future work will focus on further refining the model's performance, and extending its applications in diverse predictive tasks.

References

Andonovikj, V., Boškoski, P., Džeroski, S., Boshkoska, B.M., 2024. Survival analysis as semi-supervised multi-target regression for time-to-employment prediction using oblique predictive clustering trees. Expert Systems with Applications 235, 121246. URL: `http://dx.doi.org/10.1016/j.eswa.2023.121246`, doi:10.1016/j.eswa.2023.121246.

Bernardo, J., Bayarri, M., Berger, J., Dawid, A., Heckerman, D., Smith, A., West, M., et al., 2003. The variational bayesian em algorithm for incomplete data: with application to scoring graphical model structures. Bayesian statistics 7, 210.

Blei, D.M., Kucukelbir, A., McAuliffe, J.D., 2017. Variational inference: A review for statisticians. Journal of the American Statistical Association 112, 859–877. URL: `http://dx.doi.org/10.1080/01621459.2017.1285773`, doi:10.1080/01621459.2017.1285773.

Gershman, S., Goodman, N., 2014. Amortized inference in probabilistic reasoning, in: Proceedings of the annual meeting of the cognitive science society.

Hoffman, M.D., Blei, D.M., Wang, C., Paisley, J., 2013. Stochastic variational inference. Journal of Machine Learning Research .

Kim, Y., Wiseman, S., Miller, A., Sontag, D., Rush, A., 2018. Semi-amortized variational autoencoders, in: International Conference on Machine Learning, PMLR. pp. 2678–2687.

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

Kocev, D., Vens, C., Struyf, J., Džeroski, S., 2013. Tree ensembles for predicting structured outputs. Pattern Recognition 46, 817–833. URL: `http://dx.doi.org/10.1016/j.patcog.2012.09.023`, doi:10.1016/j.patcog.2012.09.023.

Le, T.A., Baydin, A.G., Wood, F., 2017. Inference compilation and universal probabilistic programming, in: Artificial Intelligence and Statistics, PMLR. pp. 1338–1348.

Mulan, . Mulan: A java library for multi-label learning. http://mulan.sourceforge.net/datasets.html. Accessed: 2020-04-15.

Murphy, K.P., 2023. Probabilistic machine learning: Advanced topics. MIT press.

OpenML, . Openml. https://www.openml.org. Accessed: 2020-04-15.

Plumb, G., Molitor, D., Talwalkar, A.S., 2018. Model agnostic supervised local explanations. Advances in neural information processing systems 31.

Sashank, J.R., Satyen, K., Sanjiv, K., 2018. On the convergence of adam and beyond, in: International conference on learning representations.

Stepisnik, T., Osojnik, A., Dzeroski, S., Kocev, D., 2020. Option predictive clustering trees for multi-target regression. Computer Science and Information Systems 17, 459–486. URL: http://dx.doi.org/10.2298/CSIS190928006S, doi:10.2298/csis190928006s.

Stepišnik, T., Kocev, D., 2021. Oblique predictive clustering trees. Knowledge-Based Systems 227, 107228. URL: http://dx.doi.org/10.1016/j.knosys.2021.107228, doi:10.1016/j.knosys.2021.107228.

Tan, S., Soloviev, M., Hooker, G., Wells, M.T., 2020. Tree space prototypes: Another look at making tree ensembles interpretable, in: Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference, ACM. URL: http://dx.doi.org/10.1145/3412815.3416893, doi:10.1145/3412815.3416893.

Zhao, X., Wu, Y., Lee, D.L., Cui, W., 2019. iforest: Interpreting random forests via visual analytics. IEEE Transactions on Visualization and Computer Graphics 25, 407–416. URL: http://dx.doi.org/10.1109/TVCG.2018.2864475, doi:10.1109/tvcg.2018.2864475.