Towards a Unified Framework for Uncertainty-aware Nonlinear Variable Selection with Theoretical Guarantees (with Supplementary Material)

Wenying Deng
Harvard University
wdeng@g.harvard.edu

Beau Coker
Harvard University
beaucoker@g.harvard.edu

Rajarshi Mukherjee
Harvard University
ram521@mail.harvard.edu

Jeremiah Zhe Liu*
Harvard University & Google Research
jereliu@google.com

Brent A. Coull*
Harvard University
bcoull@hsph.harvard.edu

Abstract

We develop a simple and unified framework for nonlinear variable importance estimation that incorporates uncertainty in the prediction function and is compatible with a wide range of machine learning models (e.g., tree ensembles, kernel methods, neural networks, etc). In particular, for a learned nonlinear model $f(\mathbf{x})$, we consider quantifying the importance of an input variable $\mathbf{x}^j$ using the integrated partial derivative $\Psi_j = \|\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{x})\|^2_{P_X}$. We then (1) provide a principled approach for quantifying uncertainty in variable importance by deriving its posterior distribution, and (2) show that the approach is generalizable even to non-differentiable models such as tree ensembles. Rigorous Bayesian nonparametric theorems are derived to guarantee the posterior consistency and asymptotic uncertainty of the proposed approach. Extensive simulations and experiments on healthcare benchmark datasets confirm that the proposed algorithm outperforms existing classical and recent variable selection methods. Supplementary material is at the end of this document.

1 Introduction

Variable selection is often of fundamental interest in many data science applications, providing benefits in prediction error, interpretability, and computation by excluding unnecessary variables. As datasets grow in complexity and size, it is crucial that variable importance estimation methods can account for complex dependencies among variables while remaining computationally feasible. Furthermore, as the number of approaches to model such datasets has increased, it is crucial that the importance of each variable can be compared across model classes and extended to new ones as they are developed.

While there are established approaches for quantifying variable importance in linear models (e.g., LASSO regression Hastie et al. [2015]), there is little consensus as to the preferred methodology or theory for variable importance in nonlinear models. Generalized additive models Hastie and Tibshirani [1990] use similar methods as their linear counterparts Wang et al. [2014], but the additivity assumption for nonlinear functions of the variables is too restrictive in many applications. Random Forests (RF) Breiman [2001] measure variable importance using an impurity measure, which is based on the average reduction of the loss function were a given variable removed from the model. Friedman [2001] extended this method to boosting, where the definition of variable importance is generalized by considering the average over all of the decision trees. Deep neural networks (DNNs) are widely-used for many artificial intelligence applications, and a substantial effort has been invested

*Co-senior author. Work done at Harvard University.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

into developing DNNs with variable selection capabilities. Typically, this class of models involves manipulating the input layer, for example by imposing an L_1 penalty Castellano and Fanelli [2000], Feng and Simon [2019], using backward selection Castellano and Fanelli [2000], or knockoffs Lu et al. [2018]. Unfortunately, each model class based on DNNs requires a tailored procedure, which limits comparability across different model formulations.

Bayesian variable selection methods provide principled uncertainty quantification in variable importance estimates as well as a complete characterization of their dependency structure. These methods allow the variable importance estimation procedure to tailor its decision rule with respect to the correlation structure Liu [2021]. Yet, as in frequentist models, each method has a different definition of a variable's importance. For example, in Bayesian additive regression trees (BART), a variable's importance can be measured by the proportion of trees that use it Chipman et al. [2010], while in Gaussian process (GP) models, a variable's importance can be measured by the frequency of the fluctuations of the estimated outcome-predictor function (e.g., the length-scale parameter as controlled by the automatic relevance determination) in the direction of the variable Neal [1996], Wipf and Nagarajan [2007]. Recently, a closely-related line of work uses the norm of the kernel gradient to quantify variable importance under classical GP models [He et al., 2021] or deep Bayesian neural networks [Liu, 2021]. However, these work either do not incorporate uncertainty, or are restricted to a particular model class (see Appendix J). Furthermore, the traditional Bayesian modeling procedures tend to be computationally burdensome, making them less feasible for large-scale applications [Andrieu et al., 2003].

Our work starts with the observation that many machine learning models can be written as kernel methods by constructing a corresponding feature map. For example, random forests can be written as kernel methods by partitions Davies and Ghahramani [2014], and deep neural networks can be written as kernel methods by using the last hidden layer as the feature map Snoek et al. [2015], Hinton and Salakhutdinov [2007], Calandra et al. [2016]. Each of these feature maps can be constructed before Bayesian learning of the GP (e.g., by pre-training on the same or a separate dataset), providing additional modeling expressiveness and representational capacity. Then, the GP learning is equivalent to performing Bayesian inference with respect to the (linear) weighting parameters of the feature-map basis functions and the posterior inference proceeds analogously to that of a Bayesian linear regression (see Section 2.1 for details). The ability of a GP model to incorporate these adaptive feature maps becomes especially important in high-dimensional applications, where effective dimension reduction is necessary to circumvent the curse of dimensionality and ensure good finite-sample performance [Bach, 2016].

Contributions. We propose a unified variable importance estimation framework that is compatible with a wide range of machine learning models and can be defined by, or be closely approximated by, a differentiable feature map. Notable members include neural networks and random forests (Appendix B). Our approach defines variable importance as the norm of the function's partial derivative, as was previously studied in the context of frequentist nonparametric regression Rosasco et al. [2013]. We extend it to a much wider class of models than previously considered (Section 2), propose a principled Bayesian approach to quantify the variable importance uncertainty in finite data (Section 3.1), and derive rigorous Bayesian nonparametric theorems to guarantee the method's consistency and asymptotic optimality (Section 3.2). To incorporate powerful non-differentiable models into our framework, we also show how to apply this approach to partition-based methods (e.g., decision trees) by leveraging their (soft) feature representation (Appendix F.1). This leads to the first derivative-based Bayesian variable importance estimation approach for tree-type models that is both theoretically grounded and empirically powerful. This method strongly outperforms other variable importance estimation approaches tailor-designed for random forests (e.g., impurity or random-forest knockoff [Breiman et al., 1984, Candes et al., 2017]). We conduct extensive empirical validation of our approach and compare its performance to that of many existing methods across a wide range of data generation scenarios. The results show a clear advantage of the proposed approach, especially in complex scenarios or when the input is a mixture of discrete and continuous features (Section 4).

2 Preliminaries

Problem Setup. We consider the classical nonparametric regression setting with d-dimensional features $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^d) \in \mathcal{X} = \mathbb{R}^d$ and a continuous response $y \in \mathbb{R}$. The features $\mathbf{x}$ are allowed to have a flexible nonlinear effect on y, such that:

$$y = f_0(\mathbf{x}) + e_i, \quad \text{where } e_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2), \tag{1}$$

with homoscedastic noise level σ^2. The data dimension d is allowed to be large but assumed to be constant and does not grow with the sample size n. Here the data-generating function f_0 is a flexible nonlinear function that resides in an reproducing kernel Hilbert space (RKHS) $\mathcal{H}_0$ induced by a certain positive definite kernel function k_0, and the input space $\mathcal{X}_0$ of the true function spans only a small subset of the input features $(\mathbf{x}^1, \ldots, \mathbf{x}^d)$, i.e., $\mathcal{X}_0 \subset \mathcal{X}$.

To this end, the goal of *global* variable importance estimation is to produce a variable importance score ψ_j for each of the input features $(\mathbf{x}^1, \ldots, \mathbf{x}^d)$ such that it can be used as a classification signal for whether $\mathbf{x}^j \in \mathcal{X}_0$. As a result, the variable selection decision can be made by threhsolding $\psi_j > s$ with a pre-defined threshold s. The quality of a variable selection signal ψ_j can be evaluated comprehensively using a standard metric such as the *area under the receiver operating characteristic* (AUROC), which measures the Type-I and Type-II errors of variable selection decision $I(\psi_j > s)$ over a range of thresholds s.

2.1 Quantifying Model Uncertainty via Featurized GP

In the nonlinear regression scenario given by Equation (1), a classical approach to uncertainty-aware model learning is the Gaussian process (GP). Specifically, assuming that f_0 can be described by a flexible RKHS $\mathcal{H}_k$ governed by the kernel function k, the GP model imposes a Gaussian process prior $f \sim \mathcal{GP}(0, k)$, such that the function evaluated at any collection of examples follows a multivariate normal ($\mathcal{MVN}$) distribution

$$\mathbf{f} \equiv (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^\top \sim \mathcal{MVN}(\mathbf{m}_{n\times 1}, \mathbf{K}_{n\times n}),$$

with mean $\mathbf{m}_i = m(\mathbf{x}_i)$ and covariance matrix $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. The choice of the prior mean m and kernel k enables prior specification directly in the function space. For example, the Matérn kernel with parameter ν places a prior over $\lceil \nu \rceil - 1$ times differentiable functions, with length-scale l^2 and amplitude variance σ^2. As $\nu \to \infty$, this reduces to the common radial basis function (RBF) kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp(\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / l^2)$.

Under the above construction, the posterior predictive distribution of f evaluated at new observations $\mathbf{x}_1^*, \ldots, \mathbf{x}_{n^*}^*$ is also a multivariate normal,

$$\mathbf{f}^* | \{\mathbf{x}_i\, y_i\}_{i=1}^n \sim \mathcal{MVN}(\mathbb{E}[\mathbf{f}^*], \mathrm{Cov}[\mathbf{f}^*]), \quad \text{where} \tag{2}$$
$$\mathbb{E}[\mathbf{f}^*] = \mathbf{m}^* + \mathbf{K}^*(\mathbf{K} + \sigma^2 \mathbf{I}_n)^{-1}(\mathbf{y} - \mathbf{m}); \quad \mathrm{Cov}[\mathbf{f}^*] = \mathbf{K}^{**} - \mathbf{K}^*(\mathbf{K} + \sigma^2 \mathbf{I}_n)^{-1}\mathbf{K}^{*\top},$$

with $\mathbf{m}_i^* = m(\mathbf{x}_i^*)$, $\mathbf{K}_{ij}^* = k(\mathbf{x}_i^*, \mathbf{x}_j)$, and $\mathbf{K}_{ij}^{**} = k(\mathbf{x}_i^*, \mathbf{x}_j^*)$. Equation (2) is known as the kernel-based representation (or dual representation) of a GP Rasmussen and Williams [2005]. Although mathematically elegant, the posterior (2) is expensive to compute due to the need to invert the $n \times n$ matrix $(\mathbf{K} + \sigma^2 \mathbf{I})$.

Feature-based Representation of A GP. Alternatively, Mercer's theorem Cristianini and Shawe-Taylor [2000] states that as long as the kernel function $k(\cdot, \cdot)$ can be written as the inner product of a set of basis functions $\phi(\mathbf{x}) = \{\phi_k(\mathbf{x})\}_{k=1}^D$, such that $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$, then elements of the RKHS $f \in \mathcal{H}_k$ can be written in terms of a linear expansion of basis functions Rasmussen and Williams [2005]:

$$f(\mathbf{x}) = \sum_{k=1}^D \boldsymbol{\beta}_k \phi_k(\mathbf{x}) = \phi(\mathbf{x})^\top \boldsymbol{\beta}, \text{ where } \boldsymbol{\beta} \sim \mathcal{MVN}(\boldsymbol{\mu}, \mathbf{I}_D). \tag{3}$$

This is known as the feature-based representation (or primal representation) of a GP. Notice that (3) is not an approximation method but an *exact* reparametrization of the GP model whose kernel function is induced by feature representation $\phi(\mathbf{x})$. Also note that under this featurized representation (3), the predictive model f is linear in terms of the model parameters $\boldsymbol{\beta} = \{\boldsymbol{\beta}_k\}_{k=1}^D$. However, this "linearity" in the model parameters does not restrict the expressiveness of f, since the GP model is essentially learning to use the weights $\{\boldsymbol{\beta}_k\}_{k=1}^D$ to flexibly combine the nonlinear basis functions $\{\phi_k\}_{k=1}^D$ to best fit the outcome. Furthermore, the basis functions $\{\phi_k(\mathbf{x})\}_{k=1}^D$ can be updated as part of the learning process, which we discuss in the sequel.

Scalable Posterior Computation via Minibatch Updates. The above feature-based representation is powerful in that it reduces the GP posterior inference into a Bayesian linear regression problem for $\boldsymbol{\beta}$. This brings two concrete benefits. First, the posterior of $\boldsymbol{\beta}$ in Equation (3) adopts a closed form:

$$\boldsymbol{\beta} \sim \mathcal{MVN}(\mathbb{E}[\boldsymbol{\beta}], \mathrm{Cov}[\boldsymbol{\beta}]), \quad \text{where} \tag{4}$$
$$\mathbb{E}[\boldsymbol{\beta}] = \boldsymbol{\mu} + \Sigma_{\boldsymbol{\beta}} \Phi^\top (\mathbf{y} - \Phi\boldsymbol{\mu})/\sigma^2; \quad \mathrm{Cov}[\boldsymbol{\beta}] = \Sigma_{\boldsymbol{\beta}} = (\Phi^\top \Phi/\sigma^2 + \mathbf{I})^{-1},$$

where $\Phi = (\phi(\mathbf{x}_1)^\top, \ldots, \phi(\mathbf{x}_n)^\top)^\top \in \mathbb{R}^{n \times D}$ is the feature matrix evaluated on the training data Rasmussen and Williams [2005]. For large-scale applications, Equation (4) enables us to compute the exact posterior of $\boldsymbol{\beta}$ in a mini-batch fashion. For example, the posterior matrix $\text{Cov}[\boldsymbol{\beta}] = \Sigma_{\boldsymbol{\beta}}$ can be updated using the Woodbury identity:

$$\Sigma_{\boldsymbol{\beta},t+1} = \Sigma_{\boldsymbol{\beta},t} - \Sigma_{\boldsymbol{\beta},t}\Phi_m^\top(\sigma^2\mathbf{I} + \Phi_m\Sigma_{\boldsymbol{\beta},t}\Phi_m^\top)^{-1}\Phi_m\Sigma_{\boldsymbol{\beta},t}, \tag{5}$$

where Φ_m is the D-dimension batch-specific feature matrix evaluated on the mini-batch. Similarly, the posterior mean $\mathbb{E}[\boldsymbol{\beta}]$ can be computed by accumulating the $D \times 1$ vector $\Phi^\top(\mathbf{y} - \Phi\boldsymbol{\mu}) = \sum_m \Phi_m^\top(\mathbf{y}_m - \Phi_m\boldsymbol{\mu})$, and computing the posterior mean according to Equation (4) at the end.

The posterior distribution of $\boldsymbol{\beta}$ induces a GP posterior for the prediction function $\mathbf{f}^* = \Phi^*\boldsymbol{\beta}$, where Φ^* is the feature map evaluated on the test data, with mean $\mathbb{E}[\mathbf{f}^*] = \Phi^*\boldsymbol{\mu} + \Phi^*\Sigma_{\boldsymbol{\beta}}\Phi^\top(\mathbf{y} - \Phi\boldsymbol{\mu})/\sigma^2$ and covariance $\text{Cov}[\mathbf{f}^*] = \Phi^*\Sigma_{\boldsymbol{\beta}}\Phi^{*\top}$. This distribution is equivalent to the kernel-based representation (2) but reduces the computational complexity from cubic time $O(n^3)$ to linear time $O(n)$ and is minibatch compatible (i.e., Equation (5)). Algorithm 1 and 3 provides a summary of the learning algorithm. Finally, we note that the basis functions $\phi = \{\phi_k\}_{k=1}^D$ can also be updated as part of the learning procedure (e.g., via *maximum a posteriori* (MAP) inference), which we discuss in Appendix A.4.

Incorporating Modern ML Model Classes. The second key advantage of the feature-based representation (3) is its generality: a wide range of machine learning models can be written in the feature-based form $f(\mathbf{x}) = \phi(\mathbf{x})^\top\boldsymbol{\beta}$ Rahimi and Recht [2007], Davies and Ghahramani [2014], Lee et al. [2017], making the GP a unified framework for quantifying model uncertainty for a wide array of modern ML models. Appendix B summarizes important examples including GAMs, decision trees, random-feature models, deep neural networks and their ensembles. Appendix B.1 summarizes a list of general conditions the model should satisfy for it to be compatible with the proposed framework (i.e., weak differentiability, Lipschitz condition, and growth rate of model complexity). Furthermore, when a deterministically-trained $\hat{\boldsymbol{\beta}}$ is available (e.g., via a sophisticated adaptive shrinkage procedure that is not available in a Bayesian context), we can incorporate this as prior knowledge into GP modeling by setting $\boldsymbol{\mu} = \hat{\boldsymbol{\beta}}$ (Equation (3)).

2.2 Bayesian Nonparametric Guarantees for Probabilistic Learning

The quality of a Bayesian learning procedure is commonly measured by the learning rate of its posterior distribution $\Pi_n = \Pi(\cdot \mid \{\mathbf{x}_i, y_i\}_{i=1}^n)$. Intuitively, the rate of this convergence is measured by the size of the smallest shrinking balls around f_0 that contains most of the posterior probability. Specifically, we consider the size of the set $A_n = \{g \mid \|g - f_0\|_n^2 \leq M\epsilon_n\}$ such that $\Pi_n(A_n) \to 1$ [Ghosal and Vaart, 2007, Polson and Rockova, 2018]. The concentration rate ϵ_n here indicates how fast the small ball A_n concentrates towards f_0 as the sample size increases. Below we state the formal definition of posterior convergence Ghosal and Vaart [2007].

Definition 1 (Posterior Convergence)**.** For $f_0 : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} = \mathbb{R}^d$, let $\mathcal{H}_0$ denote the true RKHS induced by a kernel function k_0, and let $\mathcal{H}_\phi$ denote the RKHS induced by the feature function $\phi : \mathcal{X} \to \mathbb{R}^D$. Let $f_0 \in \mathcal{H}_0$ be the true function, and let $\mathbb{E}_0$ denote the expectation with respect to the true data-generation distribution. Assuming $\mathcal{H}_\phi$ is dense in $\mathcal{H}_0$, then, the posterior distribution $\Pi_n(f)$ concentrates around f_0 at the rate ϵ_n if there exists an $\epsilon_n \to 0$ such that, for any $M_n \to \infty$,

$$\mathbb{E}_0\Pi_n(f : \|f - f_0\|_n^2 \geq M_n\epsilon_n) \to 0. \tag{6}$$

Notice that we allow the model space $\mathcal{H}_\phi$ and the true function space $\mathcal{H}_0$ to be different, but $\mathcal{H}_\phi$ must be *dense* in $\mathcal{H}_0$ for the convergence to happen. Fortunately, this condition is shown to hold for a wide variety of ML models, including random features, random forests, and neural networks [Biau, 2012, Hornik et al., 1989, Rahimi and Recht, 2008, Schmidt-Hieber, 2020, Ročková and van der Pas, 2020]. The notion of posterior convergence can also be used to discuss the learning quality of other probabilistic estimates (e.g., variable importance ψ_j). In that case, we can simply replace (f, f_0) in (6) by their variable importance counterparts. This is the focus of Section 3.2.

3 Methods

3.1 Quantifying Variable Importance under Uncertainty

In this work, we consider quantifying the *global* importance of a variable based on the norm of the corresponding partial derivative. This is motivated by the observation that, if a function f is

differentiable, the relative importance of a variable $\mathbf{x}^j$ at a point $\mathbf{x}$ can be captured by the magnitude of the partial derivative function, $|\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{x})|$ Rosasco et al. [2013]. This quantity requires the consideration of two issues. First, instead of quantifying the relevance of a variable on a single input point, we need to define a proper global notion of variable importance. Therefore, it is natural to integrate this partial derivative over the input space $\mathbf{x} \in \mathcal{X} : \Psi_j(f) = \|\frac{\partial}{\partial \mathbf{x}^j} f\|^2_{P_\mathcal{X}} = \int_{\mathbf{x}\in\mathcal{X}} |\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{x})|^2 \, dP_\mathcal{X}(\mathbf{x})$. Second, since $P_\mathcal{X}(\mathbf{x})$ is not known from the training observations, $\Psi_j(f)$ can be approximated by its empirical counterpart,

$$\psi_j(f) = \|\frac{\partial}{\partial \mathbf{x}^j} f\|^2_n = \frac{1}{n}\sum_{i=1}^{n} |\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{x}_i)|^2. \tag{7}$$

Notice that $\psi_j(f)$ is an estimator that is derived from the prediction function f estimated using finite data. Consequently, to make a proper decision regarding the importance of an input variable $\mathbf{x}^j$, it is important to take into account uncertainty in f. To this end, by leveraging the featurized GP representation introduced in Section 3.1, we show that this can be done easily for a wide range of ML models $f(\mathbf{x}) = \phi(\mathbf{x})^\top \boldsymbol{\beta}$ by studying the posterior distribution of ψ_j.

Posterior Distribution of Variable Importance. After we obtain the posterior distribution of $\boldsymbol{\beta}$ (4), the posterior distribution of variable importance can be derived according to Equation (7):

$$\psi_j(f) = \frac{1}{n}|\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{X})|^\top |\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{X})| = \frac{1}{n}\boldsymbol{\beta}^\top \frac{\partial \Phi}{\partial \mathbf{x}^j}\frac{\partial \Phi^\top}{\partial \mathbf{x}^j}\boldsymbol{\beta}, \tag{8}$$

where $\frac{\partial \Phi}{\partial \mathbf{x}^j} \in \mathbb{R}^{D\times n}$ is the derivative of the feature map with respect to $\mathbf{x}^j$, across n training samples. The posterior distribution of $\psi_j(f)$ adopts a closed form as a generalized chi-squared distribution (see Appendix A.2 for derivation). In practice, we can sample ψ_j conveniently from its posterior distribution by computing $\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{X}) = \left(\frac{\partial \Phi}{\partial \mathbf{x}^j}\right)^\top \boldsymbol{\beta}^{(s)}$, where $\boldsymbol{\beta}^{(s)}$ are Monte Carlo samples from the closed-form posterior (4).

There are two ways in which uncertainty aids the variable importance estimation process. First, the posterior survival function $P(\psi_j(f) > s)$ of the variable importance utilizes the full posterior distribution of $\psi_j(f)$ to identify the probability that the variable $\mathbf{x}^j$ exceeds a given threshold s. By increasing $s \in (0, \infty)$, $P(\psi_j > s)$ provides an intuitive sense of how a model's belief about the importance of variable $\mathbf{x}^j$ changes as the criteria s becomes more stringent, similar to the regularization path used by LASSO methods [Friedman et al., 2010] but with the incorporation of posterior uncertainty about the variable importance. See Appendix I for an application to a Bangladesh birth cohort study. Second, by integrating the survival function over the threshold, i.e., $\int_{s>0} P(\psi_j(f) > s)\, ds$, we obtain the posterior mean of $\psi_j(f)$, and this too incorporates uncertainty in f. To see this, notice that by using the "trace trick" we can write

$$\mathbb{E}[\psi_j(f)] = \mathbb{E}\left[\text{tr}\left(\boldsymbol{\beta}^\top \frac{\partial \Phi}{\partial \mathbf{x}^j}\frac{\partial \Phi^\top}{\partial \mathbf{x}^j}\boldsymbol{\beta}\right)\right] = \mathbb{E}[\boldsymbol{\beta}]^T \frac{\partial \Phi}{\partial \mathbf{x}^j}\frac{\partial \Phi^\top}{\partial \mathbf{x}^j}\mathbb{E}[\boldsymbol{\beta}] + \text{tr}\left(\frac{\partial \Phi}{\partial \mathbf{x}^j}\frac{\partial \Phi^\top}{\partial \mathbf{x}^j}\text{Cov}[\boldsymbol{\beta}]\right), \tag{9}$$

where all expectations are taken with respect to the posterior. Therefore, the posterior mean of $\psi_j(f)$ depends on the covariance structure of $\boldsymbol{\beta}$, and how it interacts with the eigenspace of the partial derivative functions (encoded by $\frac{\partial \Phi}{\partial \mathbf{x}^j}\frac{\partial \Phi^\top}{\partial \mathbf{x}^j}$). In Section 4 we provide an extensive investigation of AUROC scores using the posterior mean of $\psi_j(f)$ for quantifying variable importance.

In Appendix A.3, we summarize the algorithms for computing the posterior distributions of the featurized Gaussian process (Equation (4)) and for the posterior distributions of variable importance (Equation (8)), and discuss their space and time complexity.

3.2 Theoretical Guarantees

From a theoretical perspective, the variable importance measure ψ_j introduced in (7) can be understood as a quadratic functional of the GP model f Efromovich and Low [1996]. To this end, rigorous Bayesian nonparametric guarantees can be obtained for ψ_j's ability in learning the true variable importance in finite samples (i.e., posterior convergence, Theorem 1) and its statistical optimality from a frequentist perspective, in providing a low-variance estimator that attains the Cramér-Rao bound (i.e., Bernstein von-Mises phenomenon, Theorem 2). Note that for a given general model $f(\mathbf{x}) = \phi(\mathbf{x})^\top \boldsymbol{\beta}$, it only need to satisfy three mild regularity conditions to be fully compatible with the proposed framework (i.e., weak differentiability, Lipschitz condition, and growth rate of model

complexity). We summarize these conditions in Appendix B.1 and explain them in detail in the sequel.

Posterior Convergence. We first show that, for an ML model f that can learn the true function f_0 with rate ϵ_n (in the sense of Definition 1), the entire posterior distribution of the variable importance measure $\psi_j(f)$ converges consistently to a point mass at the true $\Psi_j(f_0)$ at a speed that is equal or faster than ϵ_n.

Theorem 1 (Posterior Convergence of Variable Importance ψ_j)**.** *Suppose* $y_i = f_0(\mathbf{x}_i) + e_i,\ e_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)$*, and denote as* $\mathbb{E}_0$ *the expectation with respect to the true data-generation distribution centered around* f_0*. For the RKHS* $\mathcal{H}_\phi$ *induced by the feature function* $\phi : \mathcal{X} \to \mathbb{R}^D$ *and* $f \in \mathcal{H}_\phi$*, if:*

(1) The posterior distribution $\Pi_n(f)$ *converges toward* f_0 *at a rate of* ϵ_n*;*

(2) The differentiation operator $D_j : f \to \frac{\partial}{\partial \mathbf{x}^j} f$ *is bounded:* $\|D_j\|_{op}^2 = \inf\{C \geq 0 : \|D_j f\|_2^2 \leq C\|f\|_2^2$*, for all* $f \in \mathcal{H}_\phi\}$*;*

Then the posterior distribution for $\psi_j(f) = \|\frac{\partial}{\partial \mathbf{x}^j} f\|_n^2$ *contracts toward* $\Psi_j(f_0) = \|\frac{\partial}{\partial \mathbf{x}^j} f_0\|_{P_\mathcal{X}}^2$ *at a rate not slower than* ϵ_n*. That is, for any* $M_n \to \infty$*,*

$$\mathbb{E}_0 \Pi_n \left[\sup_{j \in \{1,\ldots,d\}} |\psi_j(f) - \Psi_j(f_0)| \geq M_n \epsilon_n \right] \to 0.$$

The proof is in Appendix C. Theorem 1 is a generalization of the classical result of quadratic functional convergence under linear models and sparse neural networks to a much wider range of ML models in the context of Bayesian variable importance estimation [Efromovich and Low, 1996, Liu, 2021, Wang and Rocková, 2020]. It confirms the important fact that, for an ML model f that can accurately learn the true function f_0 under finite data, we can consistently recover the true variable importance at a fast rate by using the proposed variable importance estimate $\psi_j(f)$, despite the potential lack of identifiablity in the model parameters (e.g., weights in a neural network). Importantly, although our main setting assumes fixed data dimension d (see Problem Setup), the posterior concentration result Theorem 1 does not rely on this assumption in its proof, and is in fact compatible with the high-dimensional setting where d is allowed to grow with sample size at a rate of $o(n)$. See Appendix C (in particular, Remark 4) for further discussion.

From a practical point of view, Theorem 1 reveals that the finite-sample performance of variable importance $\psi_j(f)$ depends on two factors: (1) the finite-sample generalization performance of the prediction function f, and (2) the mathematical property of f in terms of its Lipschitz condition. Therefore, to ensure effective variable importance estimation in practice, the practitioner should take care to select a model class f that has a theoretical guarantee in capturing the target function f_0, empirically delivers strong generalization performance under finite data, and is well-conditioned in terms of the behavior of its partial derivatives. To this end, we note that, under the featurized Gaussian process $f = \phi(\mathbf{x})^\top \boldsymbol{\beta}$ discussed in this work, users are free to choose a performant model class (e.g., random forest, random-feature or DNN) whose feature representation spans an RKHS $\mathcal{H}_\phi$ that is *dense* in the infinite-dimensional function space (therefore f enjoys a convergence guanrantee, see Remark 3 in Appendix C for further discussion) [Biau, 2012, Hornik et al., 1989, Rahimi and Recht, 2008, Schmidt-Hieber, 2020, Ročková and van der Pas, 2020], and is empirically more effective than the GP methods based on classical kernels such as RBF. We discuss the Lipschitz condition of these models in Appendix E.1. Indeed, as we will verify in experiments (Section 4), there does not exist an "optimal" model class that performs universally well across all data settings (i.e., no free lunch theorem [Wolpert and Macready, 1997]). This highlights the importance of having a general-purpose framework for variable importance estimation that can flexibly incorporate the most effective model for the task at hand. Finally, we notice that although Theorem 1 is stated as an asymptotic result, when a finite-sample error bound ϵ_n for the model class is available (i.e., Condition (1) in Theorem 1), it is trivial to obtain a finite-sample error bound for variable importance $\psi_j(f)$ by extending the proof of Theorem 1. Appendix C.1 provides an example of such a bound based on the Bernstein inequality.

Statistical Efficiency & **Uncertainty Quantification.** Next, we verify the uncertainty quantification ability of the variable importance measure $\psi_j(f)$ under a featurized GP by showing that it exhibits the *Bernstein-von Mises* (BvM) phenomenon. That is, its posterior measure $\Pi_n(\psi_j(f))$ converges towards a Gaussian distribution that is centered around the truth $\Psi_j(f_0)$, so that its $(1-\alpha)\%$ level

credible intervals achieve the nominal coverage probability for the true variable importance. More importantly, the BvM theorem verifies that the posterior distribution of $\psi_j(f)$ is *statistically optimal*, in the sense that its asymptotic variance attains the Cramér-Rao bound (CRB) that cannot be improved upon [Bickel and Kleijn, 2012].

Theorem 2 (Bernstein-von Mises Theorem for Variable Importance ψ_j)**.** *Suppose* $y_i = f_0(\mathbf{x}_i) + e_i,\ e_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2),\ i = 1, \ldots, n$. *Denote* $D_j : f \to \frac{\partial}{\partial \mathbf{x}^j} f$ *the differentiation operator and* $H_j = D_j^\top D_j$ *the inner product of* D_j*, such that:*

$$\psi_j(f) = \|D_j(f)\|_n^2 = \frac{1}{n}\langle D_j f, D_j f\rangle = \frac{1}{n} f^\top H_j f. \tag{10}$$

Assuming conditions (1)-(2) in Theorem 1 hold, and additionally:

(3) f_0 *is square-integrable over the support* $\mathcal{X}$ *and* $\|f_0\|_2 = 1$*;*

(4) $\texttt{rank}(H_j) = o_p(\sqrt{n})$*;*

Then

$$\sqrt{n}(\psi_j(f) - \psi_j(f_0)) \xrightarrow{d} \mathcal{N}(0, 4\sigma^2 \|H_j f_0\|_n^2).$$

The proof is in Appendix D. Theorem 2 provides a rigorous theoretical justification for $\psi_j(f)$'s ability to quantify its uncertainty about the variable importance. More importantly, it verifies that $\psi_j(f)$ has the good frequentist property that it quickly converges to a minimum-variance estimator at a fast speed, which is important for obtaining good variable importance estimation performance in practice. Compared to the previous BvM results that tend to focus on a specific Bayesian ML model, Theorem 2 is considerably more general (i.e., applicable to a much wider range of models) and comes with a simpler set of conditions [Rockova, 2020, Wang and Rocková, 2020, Liu, 2021]. Specifically, (3) is a standard assumption in nonparametric analysis. It ensures the true function f_0 does not diverge towards infinity and makes learning possible [Castillo and Rousseau, 2015]. The unit norm assumption $\|f_0\|_2 = 1$ is only needed to simplify the exposition of the proof, and the theorem can be trivially extended to $\|f_0\|_2 = C$ for any $C > 0$. The most interesting condition is (4). Let us denote $\mathcal{H}_j$ as the space of partial derivatives functions $\frac{\partial}{\partial \mathbf{x}^j} f$ of the model functions $f \in \mathcal{H}_\phi$. Then, intuitively, (4) says that to attain the BvM phenomenon, the effective dimension of the derivative function space $\mathcal{H}_j$ (as measured by $\texttt{rank}(H_j) = \texttt{rank}(D_j)$) cannot be too large. Since the effective dimension of the derivative space is bounded above by that of the original RKHS $f \in \mathcal{H}_\phi$, (4) essentially states that the effective dimension of the model space $\mathcal{H}_\phi$ cannot grow too fast with data size (i.e., $o_p(\sqrt{n})$). Fortunately, this condition is satisfied by a wide range of ML models including trees and deep networks [Rockova, 2020, Wang and Rocková, 2020]. See Appendix E.2 for further discussion.

4 Experiment Analysis

In this section, we investigate the finite-sample performance of the derivative norm metric ψ_j for variable importance estimation (7) under a wide variety of ML methods. We illustrate the breath of our framework by applying it to tree ensembles (Appendix F.1), where a principled and gradient-based uncertainty-aware variable importance estimation approach has been previously unavailable. We also apply it to linear models and (approximate) kernel machines, which are standard approaches to variable selection in data science practice [Tibshirani, 1996, Bobb et al., 2015]. Over a wide range of complex and realistic data scenarios (e.g., discrete features, interactions, between-feature correlations) derived from socioeconomic and healthcare datasets, we investigate the method's statistical performance in accurately recovering the ground-truth features (in terms of the Type I and Type II errors), and compare it to other well-established approaches in each of the model classes (Table 1). Our main observations are:

O1: **Importance of generality**. There does not exist a model class that performs universally well across all data scenarios (i.e., no free lunch theorem [Wolpert and Macready, 1997], Figures 1, 6-15). This highlights the importance of an unified framework for variable importance that incorporates a wide range of models, so that practitioners have the freedom of choosing the most suitable model class for the task at hand.

O2: **Good prediction translates to effective variable importance estimation**. Comparing between different model classes, the ranking of models' predictive accuracy is generally consistent with the

ranking of their variable importance estimation performance under ψ_j (i.e., better prediction translates to better variable importance estimation, as suggested in Theorem 1).

O3: **Statistical efficiency of** ψ_j. Comparing within each model class, the derivative norm metric ψ_j generally outperforms other measures of variable importance. The advantage is especially pronounced in small samples and for correlated features. This empirically verifies that ψ_j has good finite-sample statistical efficiency even under complex data scenarios (as suggested in Theorem 2).

Model Class	(Ours)	Baselines
Tree Ensembles	RF-FDT	RF-Impurity, RF-Knockoff, BART
Kernel Methods & NNs	RFF, NN	BKMR, BAKR
Linear Models	GAM	BRR, BL

Table 1: Summary of methods considered in the experiments.

Models & **Methods**. We consider three main classes of models (Table 1): (I) **Random Forests (RF)**. Given a trained forest, we quantify variable importance using ψ_j by translating it to an ensemble of featurized decision trees (**FDT**) (Appendix F.1), and compare it to three baselines: *impurity* (**RF-impurity**) [Breiman et al., 1984], *RF-based kernel knockoff* (**RF-knockoff**) [Candes et al., 2017], and *Bayesian Additive Regression Trees* (**BART**). (II) **(Approximate) Kernel Methods** & **Neural Networks**. We apply ψ_j to a random-feature model that approximates a GP with an RBF kernel Rahimi and Recht [2007], and set the number of features to $\sqrt{n}\,log(n)$ to ensure proper approximation of the exact RBF-GP Rudi and Rosasco [2018], which is termed *Random Fourier Feature model* (**RFF**). We also apply ψ_j to *Neural Networks* (**NN**) based on wide ReLU neural network with 512 hidden units and LASSO regularization in the hidden layer weights [Lemhadri et al., 2021]. We compare them to *Bayesian Kernel Machine Regression* (**BKMR**) Bobb et al. [2015] based on a GP with an exact RBF kernel and a spike-and-slab prior, and *Bayesian Approximate Kernel Regression* (**BAKR**) based on random-feature model with a projection-based feature importance measure and an adaptive shrinkage prior [Crawford et al., 2018]. (III) **Linear Models**. We apply ψ_j to a featurized GP representation of the *Generalized Additive Model* (**GAM**), with the prior center $\boldsymbol{\mu}$ set at the frequentist estimate of the original GAM model obtained from a sophisticated REML procedure [Wood, 2006]. We compare it to two baselines: *Bayesian Ridge Regression* (**BRR**) Hoerl and Kennard [1970] and *Bayesian LASSO* (**BL**) Park and Casella [2008]. Appendix G provides further detail. Previously, [Liu, 2021] studied the specialization of our framework to the deep neural networks (DNNs), so we do not repeat that work here as DNN is not yet a standard data science model for tabular data.

To quantify variable importance while accounting for posterior uncertainty of the variable importance $\psi_j(f)$, we examine its posterior survival function $\int_{s>0} P(\psi_j(f) > s)\, ds$ (i.e., the posterior likelihood of $\psi_j(f)$ being greater than the threshold s integrated over the full range of thresholds s). For other methods, we use their default metrics to quantify variable importance (e.g., variable inclusion probabilities in **BART** and **BKMR**. See Appendix G).

Datasets and Tasks. We consider two synthetic benchmark datasets and three real-world socio-economic and healthcare datasets, encapsulating challenging phenomena such as between-feature correlations and interaction effects. For the synthetic benchmark datasets, we generate data under the Gaussian noise model $y \sim \mathcal{N}(f_0, 0.01)$ for four types of outcome-generation functions f_0 (`linear`, `rbf`, `matern32` and `complex`, see Appendix G.2 for a full description) with the number of causal variables set at $d^\star = 5$. We consider two types of feature distribution: (1) **synthetic-continuous**: all features follow $\mathbf{x}^j \sim Unif(-2, 2)$; (2) **synthetic-mixture**: two of the causal features and two of the non-causal features are distributed as $Bern(0.5)$ and the rest are distributed as $Unif(-2, 2)$. Features in both distributions are independent. We vary sample size $n \in \{100, 200, 500, 1000\}$ and data dimension $d \in \{25, 50, 100, 200\}$, leading to 128 total scenarios.

For real-world data, we consider (1) **adult**: 1994 U.S. census data of 48842 adults with eight categorical and six continuous features Kohavi; (2) **heart**: a coronary artery disease dataset of 303 patients from Cleveland clinic database with seven categorical and six continuous features Detrano et al. [1989]; and (3) **mi**: disease records of myocardial infarction (MI) of 1700 patients from Krasnoyarsk interdistrict clinical hospital during 1992-1995, with 113 categorical and 11 continuous features Golovenkin et al. [2020]. All datasets exhibit non-trival correlation structure among features (Appendix Figures 3-5). Since the ground-truth causal features on these datasets are not known, in order to rigorously evaluate variable importance estimation performance, we follow the standard practice in causal ML to simulate the outcome based on causal features selected from data [Yao et al., 2021]. We use the four outcome-generating functions as described previously and evaluate

over the same data size $\times$ dimension combinations, leading to 192 total scenarios[2]. We repeat the simulation 20 times for each scenario, and use AUROC to measure the variable importance estimation performance (in terms of Type I and Type II errors) of each method.

In Appendix I, we further evaluate the method on a well-studied environmental health dataset (Bangladesh birth cohort study [Kile et al., 2014]) with respect to the real outcome (infant development scores). We visualize the "Bayesian" regularization path as introduced earlier. The selected variables correspond well with the established toxicology pathways in the literature [Gleason et al., 2014].

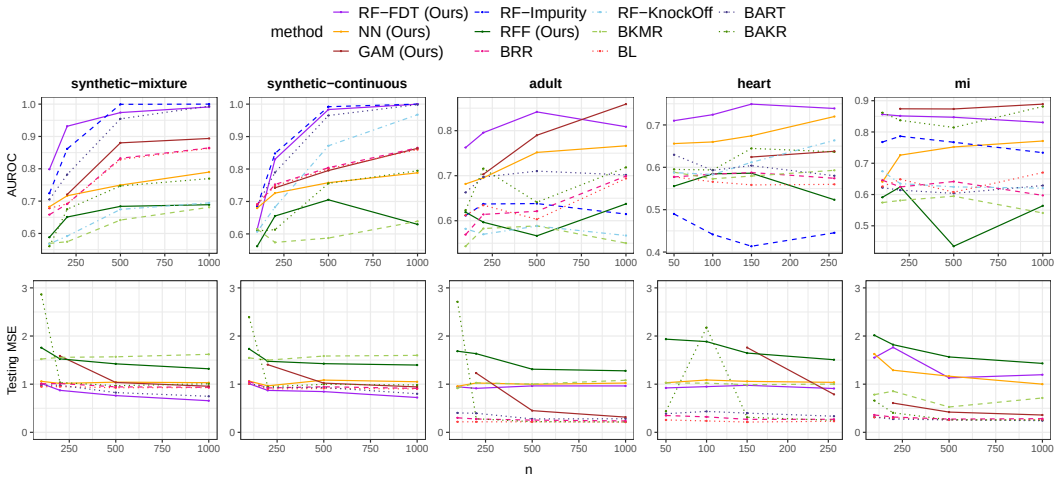


Figure 1: Method performance in variable importance estimation (measured by AUROC, row 1) and prediction (measured by test MSE, row 2) under `matern32` data-generation function and with input dimension 100 (five causal features). Therefore, the variable importance scores of the five causal features are expected to be higher than the other 95 variable importance estimations. The x-axis represents the training sample sizes $n \in \{100, 200, 500, 1000\}$. **GAM** does not produce valid results for case of $n \leq d$ so the results from this model in these cases are not shown. The ranking of **FDT** (solid purple) outperforms other methods in most of the data settings, and **GAM** outperforms in the setting of large data size and high percentage of categorical features (**adult** and **mi**). The rankings of performance are roughly consistent between prediction and variable importance estimation.

4.1 Results

Figure 1 shows the methods' performance in variable importance estimation **(Row 1)** and prediction **(Row 2)**[3] in an exemplary setting, where the true function f_0 is `matern32` with an input dimension $d = 100$. It represents the tabular data setting that we are the most interested in: nonlinear feature-response relationship with interaction effects and high input dimension. This is because f_0 is sampled from an RKHS induced by Mátern $\frac{3}{2}$ kernel, which contains a large space of continuous and at least once differentiable functions [Rasmussen and Williams, 2005]. We delay complete visualizations for all 320 scenarios to Appendix H. Recalling the three observations introduced earlier:

O1 ("No free lunch"): No method performs universally well. For example, **BAKR** performs robustly in correlated datasets (**heart** and **mi**), but poorly otherwise. Kernel approaches (**RFF** and **BKMR**) perform competitively in low dimension, but their performances deteriorate quickly as dimension d increases (Figure 1 and Figure 6-7). This is likely due to the classical kernel method's well-known inability to learn an adaptive feature representation, which consequently leads to suffering from the curse of dimensionality and unstable and suboptimal variable importance performance in high dimensions [Bach, 2017]. **FDT** is generally the strongest method in small samples and high dimensions, but can be outperformed by **GAM** in large samples and data with a high percentage of categorical features (**adult** and **mi**). Notably, they often outperform **NN**, which is traditionally regarded as the go-to model for high-dimensional nonlinear settings. This highlights the importance of a unified framework that allows users to select the most appropriate model for variable importance estimation depending on the data setting.

[2]In the setting where required data dimension is higher than that of the real data, we generate additional synthetic features from $Unif(-2, 2)$. We use $n \in \{50, 100, 150, 257\}$ for **heart** due to data size restrictions.

[3]For the prediction plots, a method will not be visualized if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if it does not produce valid results due to small sample size (**GAM**).

O2 ("Good prediction implies effective variable importance estimation"): Fixing the variable importance ψ_j and comparing the variable importance estimation performance of each model class (i.e., **FDT**, **GAM**, **NN** and **RFF**, which are solid lines in Figure 1), we see that their rankings in prediction (row 2) are largely consistent with the corresponding rankings in variable importance estimation. It is worth noting that this pattern is occasionally violated (e.g., **GAM** in **adult**, $n = 500$ and **heart**, $n = 250$), but that does not contradict our conclusion (Theorem 1) since the convergence rate of the prediction function only forms an *upper bound* for the convergence rate of ψ_j. Finally, when models have comparable generalization performance, we observe that the Lipschitz condition plays a role in variance importance performance (which is consistent with our theoretical observations in Theorem 1). For example, in Figure 1, **NN** and **FDT** are largely comparable in predictive performance among the real datasets (**adult**, **heart** and **mi**). However, tree-based **FDT** are known to have well-conditioned Lipschitz behavior when compared to **NN** (Appendix E.1), which is consequently translated to improved finite-sample performance in variable importance estimation.

O3 (Statistical efficiency of ψ_j): When comparing among variable importance estimation methods from the same class (especially for tree models, i.e., **FDT** v.s. **RF-impurity / RF-knockoff / BART**), we see that **FDT** is competitive or strongly outperforms its baselines in variable importance estimation, despite being based on exactly the same fitted model (**RF-impurity / RF-knockoff**), or not accounting for the uncertainty in the tree growing process (**BART**). This pattern is consistent in most data settings, and the advantage is especially pronounced in high dimensions, small data sizes, and correlated datasets (Appendix H, Figure 6-10). This provides strong empirical evidence for the fact that ψ_j is a statistically efficient estimator for variable importance with good finite-sample behavior (as suggested in Theorem 2), and can deliver strong performance for tabular data when combined with a performant ML model like random forests. Appendix H contains further discussion.

5 Discussion and Future Directions

The modern data analysis pipeline typically involves fitting multiple models, comparing their performance, and iterating as necessary. When variable selection is involved, the practitioner may ask *are the variable importance scores across models measuring the same behavior?* And, *what if the most suitable model does not have a satisfactory variable importance estimation procedure?* By framing model choice as the specification of a kernel — which includes kernels corresponding machine learning methods like neural networks and random forests in addition to the long list of traditional kernels — we propose a unified variable importance estimation procedure that is compatible across models and prove strong guarantees for this procedure.

Limitations. We do not consider uncertainty in the feature map itself. For example, the kernel induced by the featurized decision tree studied here does not consider uncertainty in the tree's partitioning process. Meanwhile, the fact that the full posterior inference is performed only with respect to $\boldsymbol{\beta}$ indeed places a limitation on the model's ability in uncertainty quantification, as the uncertainty in the model hyperparameters is not accounted for. Yet, this does not seem to be a **significant limitation in the method's empirical performance** (e.g., **FDT** outperforms **BART** in our experiments), although this point still merits further investigation in the future. On the other hand, in the future, it would be worth expanding this framework to other model classes (e.g., MARS Friedman [1991]) and estimating the importance of interaction effects and higher-order terms (see Appendix K for details).

Societal Impacts. We expect the method proposed to provide a set of powerful tools for practitioners to understand the importance of input variables in their ML models with limited data, which is especially important for scientific investigations in the fields of epidemiology and computational biology. However, we recognize that this approach can potentially be utilized by bad actors to probe the input-variable uncertainty of an existing ML system, and use it to engineer more targeted white-box adversarial attacks. To this end, we recommend system developers to incorporate this approach into the formal verification procedure of an ML system, so as to monitor and understand the model uncertainty with respect to input variables, and devise proper improvement and prevention strategies (e.g., data augmentation or randomized smoothing targeted at specific variables) accordingly.

Acknowledgement

This research was supported by NIH grants ES000002, ES028800, ES028811, ES028688, and ES030990. We would also like to thank Dr. Andrew Beam for the helpful discussions.

References

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

F. Bach. Breaking the Curse of Dimensionality with Convex Neural Networks. *arXiv:1412.8690 [cs, math, stat]*, Oct. 2016. URL http://arxiv.org/abs/1412.8690.

F. Bach. Breaking the Curse of Dimensionality with Convex Neural Networks. *Journal of Machine Learning Research*, 18(19):1–53, 2017. ISSN 1533-7928.

G. Biau. Analysis of a random forests model. *The Journal of Machine Learning Research*, 13(1): 1063–1095, 2012.

G. Biau, E. Scornet, and J. Welbl. Neural Random Forests. *Sankhya A*, 81(2):347–386, Dec. 2019. ISSN 0976-8378. doi: 10.1007/s13171-018-0133-y. URL https://doi.org/10.1007/s13171-018-0133-y.

P. J. Bickel and B. J. Kleijn. The semiparametric bernstein–von mises theorem. *The Annals of Statistics*, 40(1):206–237, 2012.

J. F. Bobb, L. Valeri, B. Claus Henn, D. C. Christiani, R. O. Wright, M. Mazumdar, J. J. Godleski, and B. A. Coull. Bayesian kernel machine regression for estimating the health effects of multi-pollutant mixtures. *Biostatistics*, 16(3):493–508, July 2015. ISSN 1465-4644. doi: 10.1093/biostatistics/kxu058.

D. Bontemps. Bernstein–von mises theorems for gaussian regression with increasing number of regressors. *The Annals of Statistics*, 39(5):2557–2584, 2011.

L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Oct. 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Taylor & Francis, Jan. 1984. ISBN 978-0-412-04841-8.

E. Burnaev, A. Zaytsev, and V. Spokoiny. The bernstein-von mises theorem for regression based on gaussian processes. *Russ. Math. Surv*, 68(5):954–956, 2013.

R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3338–3345. IEEE, 2016.

E. Candes, Y. Fan, L. Janson, and J. Lv. Panning for Gold: Model-X Knockoffs for High-dimensional Controlled Variable Selection. *arXiv:1610.02351 [math, stat]*, Dec. 2017. URL http://arxiv.org/abs/1610.02351. arXiv: 1610.02351.

G. Castellano and A. M. Fanelli. *Variable Selection Using Neural-Network Models*. 2000.

I. Castillo and J. Rousseau. A bernstein–von mises theorem for smooth functionals in semiparametric models. *The Annals of Statistics*, 43(6):2353–2383, 2015.

H. A. Chipman, E. I. George, and R. E. McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, Mar. 2010. ISSN 1932-6157, 1941-7330. doi: 10.1214/09-AOAS285. Publisher: Institute of Mathematical Statistics.

K. Choromanski, M. Rowland, T. Sarlos, V. Sindhwani, R. Turner, and A. Weller. The geometry of random features. In A. Storkey and F. Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1–9. PMLR, 09–11 Apr 2018. URL https://proceedings.mlr.press/v84/choromanski18a.html.

L. Crawford, K. C. Wood, X. Zhou, and S. Mukherjee. Bayesian Approximate Kernel Regression With Variable Selection. *Journal of the American Statistical Association*, 113(524):1710–1721, Oct. 2018. ISSN 0162-1459. doi: 10.1080/01621459.2017.1361830. URL https://doi.org/10.1080/01621459.2017.1361830.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, 2000. ISBN 978-0-521-78019-3. doi: 10.1017/CBO9780511801389.

S. d'Ascoli, L. Sagun, and G. Biroli. Triple descent and the two kinds of overfitting: Where & why do they appear? *Advances in Neural Information Processing Systems*, 33:3058–3069, 2020.

A. Davies and Z. Ghahramani. The random forest kernel and other kernels for big data from random partitions. *arXiv preprint arXiv:1402.4293*, 2014.

R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. J. Schmid, S. Sandhu, K. H. Guppy, S. Lee, and V. Froelicher. International application of a new probability algorithm for the diagnosis of coronary artery disease. *Am J Cardiol*, 64(5):304–310, Aug. 1989. ISSN 0002-9149. doi: 10.1016/0002-9149(89)90524-9.

S. Efromovich and M. Low. On optimal adaptive estimation of a quadratic functional. *The Annals of Statistics*, 24(3):1106–1125, 1996.

J. Feng and N. Simon. Sparse-Input Neural Networks for High-dimensional Nonparametric Regression and Classification. *arXiv:1711.07592 [stat]*, June 2019. URL `http://arxiv.org/abs/1711.07592`.

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

J. H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1 – 67, 1991. doi: 10.1214/aos/1176347963. URL `https://doi.org/10.1214/aos/1176347963`.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, Oct. 2001. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1013203451. URL `https://projecteuclid.org/euclid.aos/1013203451`.

N. Frosst and G. Hinton. Distilling a Neural Network Into a Soft Decision Tree. *arXiv:1711.09784 [cs, stat]*, Nov. 2017. URL `http://arxiv.org/abs/1711.09784`. arXiv: 1711.09784.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Mach Learn*, 63(1):3–42, Apr. 2006. ISSN 1573-0565. doi: 10.1007/s10994-006-6226-1. URL `https://doi.org/10.1007/s10994-006-6226-1`.

S. Ghosal and A. v. d. Vaart. Convergence rates of posterior distributions for noniid observations. *The Annals of Statistics*, 35(1):192–223, Feb. 2007. ISSN 0090-5364, 2168-8966. doi: 10.1214/009053606000001172.

K. Gleason, J. P. Shine, N. Shobnam, L. B. Rokoff, H. S. Suchanda, M. O. S. Ibne Hasan, G. Mostofa, C. Amarasiriwardena, Q. Quamruzzaman, M. Rahman, et al. Contaminated turmeric is a potential source of lead exposure for children in rural bangladesh. *Journal of Environmental and Public Health*, 2014, 2014.

S. E. Golovenkin, J. Bac, A. Chervov, E. M. Mirkes, Y. V. Orlova, E. Barillot, A. N. Gorban, and A. Zinovyev. Trajectories, bifurcations, and pseudo-time in large clinical datasets: applications to myocardial infarction and diabetes data. *GigaScience*, 9(11):giaa128, Nov. 2020. ISSN 2047-217X. doi: 10.1093/gigascience/giaa128. URL `https://doi.org/10.1093/gigascience/giaa128`.

J. D. Hamadani, F. Tofail, B. Nermell, R. Gardner, S. Shiraji, M. Bottai, S. Arifeen, S. N. Huda, and M. Vahter. Critical windows of exposure for arsenic-associated impairment of cognitive function in pre-school girls and boys: a population-based cohort study. *International journal of epidemiology*, 40(6):1593–1604, 2011.

D. A. Harville. On the distribution of linear combinations of non-central chi-squares. *The Annals of Mathematical Statistics*, 42(2):809–811, 1971.

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

T. Hastie, R. Tibshirani, and M. Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015. ISBN 978-1-4987-1216-3.

T. J. Hastie and R. J. Tibshirani. *Generalized Additive Models*. Chapman and Hall/CRC, Boca Raton, Fla, 1st edition edition, June 1990. ISBN 978-0-412-34390-2.

X. He, J. Wang, and S. Lv. Efficient kernel-based variable selection with sparsistency. *Statistica Sinica*, 31(4):2123–2151, 2021.

G. E. Hinton and R. R. Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. *Advances in neural information processing systems*, 20, 2007.

A. E. Hoerl and R. W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, Feb. 1970. ISSN 0040-1706. doi: 10.1080/00401706.1970.10488634.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

O. Irsoy, O. T. Yildiz, and E. Alpaydin. Soft decision trees. *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, 2012.

A. Jacot, B. Simsek, F. Spadaro, C. Hongler, and F. Gabriel. Implicit regularization of random feature models. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4631–4640. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/jacot20a.html`.

A. Karthikeyan, N. Jain, N. Natarajan, and P. Jain. Learning accurate decision trees with bandit feedback via quantized gradient descent. *arXiv preprint arXiv:2102.07567*, 2021.

M. L. Kile, E. G. Rodrigues, M. Mazumdar, C. B. Dobson, N. Diao, M. Golam, Q. Quamruzzaman, M. Rahman, and D. C. Christiani. A prospective cohort study of the association between drinking water arsenic exposure and self-reported maternal health symptoms during pregnancy in bangladesh. *Environmental Health*, 13(1):1–13, 2014.

R. Kohavi. Scaling Up theaADceccuisriaocny-TorfeNeaHivyeb-Bridayes Classi ers:. page 6.

J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

I. Lemhadri, F. Ruan, and R. Tibshirani. Lassonet: Neural networks with feature sparsity. In *International Conference on Artificial Intelligence and Statistics*, pages 10–18. PMLR, 2021.

F. Liu, X. Huang, Y. Chen, and J. A. Suykens. Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (01):1–1, 2021.

J. Liu. Variable selection with rigorous uncertainty quantification using deep bayesian neural networks: Posterior concentration and bernstein-von mises phenomenon. In *International Conference on Artificial Intelligence and Statistics*, pages 3124–3132. PMLR, 2021.

J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss, and B. Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in Neural Information Processing Systems*, 33:7498–7512, 2020.

J. Z. Liu. Gaussian Process Regression and Classification under Mathematical Constraints with Learning Guarantees. *arXiv:1904.09632 [cs, math, stat]*, Apr. 2019. URL `http://arxiv.org/abs/1904.09632`.

Y. Lu. On the bernstein-von mises theorem for high dimensional nonlinear bayesian inverse problems. *arXiv preprint arXiv:1706.00289*, 2017.

Y. Y. Lu, Y. Fan, J. Lv, and W. S. Noble. DeepPINK: reproducible feature selection in deep neural networks. *arXiv:1809.01185 [cs, stat]*, Sept. 2018. URL `http://arxiv.org/abs/1809.01185`.

J. Mairal and B. Yu. Complexity analysis of the lasso regularization path. *arXiv preprint arXiv:1205.0079*, 2012.

S. Mei and A. Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *arXiv: Statistics Theory*, 2019.

C. A. Micchelli, Y. Xu, and H. Zhang. Universal kernels. *Journal of Machine Learning Research*, 7 (12), 2006.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996. ISBN 978-0-387-94724-2.

S. W. Ober and C. E. Rasmussen. Benchmarking the neural linear model for regression. In *2nd Symposium on Advances in Approximate Bayesian Inference*, 2019.

T. Park and G. Casella. The Bayesian Lasso. *Journal of the American Statistical Association*, 103(482):681–686, June 2008. ISSN 0162-1459. doi: 10.1198/016214508000000337. URL `https://doi.org/10.1198/016214508000000337`.

N. Polson and V. Rockova. Posterior Concentration for Sparse Deep Learning. *arXiv:1803.09138 [cs, stat]*, Mar. 2018. URL `http://arxiv.org/abs/1803.09138`.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1177–1184, Red Hook, NY, USA, Dec. 2007. Curran Associates Inc. ISBN 978-1-60560-352-0.

A. Rahimi and B. Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 555–561. IEEE, 2008.

A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009. URL `https://proceedings.neurips.cc/paper/2008/file/0efe32849d230d7f53049ddc4a4b0c60-Paper.pdf`.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, hardcover edition, 11 2005. ISBN 978-0262182539. URL `https://lead.to/amazon/com/?op=bt&la=en&cu=usd&key=026218253X`.

V. Rockova. On semi-parametric inference for bart. In *International Conference on Machine Learning*, pages 8137–8146. PMLR, 2020.

V. Ročková and S. van der Pas. Posterior concentration for bayesian regression trees and forests. *The Annals of Statistics*, 48(4):2108–2131, 2020.

L. Rosasco, S. Villa, S. Mosci, M. Santoro, and A. Verri. Nonparametric sparsity and regularization. *The Journal of Machine Learning Research*, 14(1):1665–1714, Jan. 2013. ISSN 1532-4435.

A. Rudi and L. Rosasco. Generalization Properties of Learning with Random Features. *arXiv:1602.04474 [cs, stat]*, Jan. 2018. URL `http://arxiv.org/abs/1602.04474`.

J. Schmidt-Hieber. Nonparametric regression using deep neural networks with ReLU activation function. *The Annals of Statistics*, 48(4):1875–1897, Aug. 2020. ISSN 0090-5364, 2168-8966. doi: 10.1214/19-AOS1875.

F. Sigrist. Gaussian process boosting. *arXiv preprint arXiv:2004.02653*, 2020.

J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, Prabhat, and R. P. Adams. Scalable bayesian optimization using deep neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

R. Tanno, K. Arulkumaran, D. Alexander, A. Criminisi, and A. Nori. Adaptive Neural Trees. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6166–6175. PMLR, May 2019. URL `https://proceedings.mlr.press/v97/tanno19a.html`. ISSN: 2640-3498.

S. Thakur, C. Lorsung, Y. Yacoby, F. Doshi-Velez, and W. Pan. Uncertainty-aware (una) bases for deep bayesian regression using multi-headed auxiliary networks. *arXiv preprint arXiv:2006.11695v4*, 2021.

R. Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. ISSN 2517-6161. doi: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

L. Valeri, M. M. Mazumdar, J. F. Bobb, B. Claus Henn, E. Rodrigues, O. I. Sharif, M. L. Kile, Q. Quamruzzaman, S. Afroz, M. Golam, et al. The joint effect of prenatal exposure to metal mixtures on neurodevelopmental outcomes at 20–40 months of age: evidence from rural bangladesh. *Environmental health perspectives*, 125(6):067015, 2017.

A. van der Vaart and H. Zanten. Information Rates of Nonparametric Gaussian Process Methods. *Journal of Machine Learning Research*, 12:2095–2119, June 2011.

A. Virmaux and K. Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

L. Wang, L. Xue, A. Qu, and H. Liang. Estimation and model selection in generalized additive partial linear models for correlated data with diverging number of covariates. *Annals of Statistics*, 42(2): 592—624, 2014.

Y. Wang and V. Rocková. Uncertainty quantification for sparse deep learning. In *International Conference on Artificial Intelligence and Statistics*, pages 298–308. PMLR, 2020.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016a.

A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing. Stochastic variational deep kernel learning. *Advances in Neural Information Processing Systems*, 29, 2016b.

D. Wipf and S. Nagarajan. A new view of automatic relevance determination. *Advances in neural information processing systems*, 20, 2007.

D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

S. N. Wood. *Generalized additive models: an introduction with R*. chapman and hall/CRC, 2006.

Y. Yang, G. Cheng, and D. B. Dunson. Semiparametric bernstein-von mises theorem: Second order studies. *arXiv preprint arXiv:1503.04493*, 2015.

Y. Yang, I. G. Morillo, and T. M. Hospedales. Deep Neural Decision Trees. *arXiv:1806.06988 [cs, stat]*, June 2018. URL `http://arxiv.org/abs/1806.06988`. arXiv: 1806.06988.

L. Yao, Z. Chu, S. Li, Y. Li, J. Gao, and A. Zhang. A survey on causal inference. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(5):1–46, 2021.

A Additional Background and Technical Derivations

A.1 Neural Network Representation of Decision Tree

For each node in a learned decision tree, we know the feature the node is splitting on and its corresponding threshold. Karthikeyan et al. [2021] provides a neural network representation of a decision tree:

$$f(\mathbf{x}|\mathbf{W},\mathbf{b},\boldsymbol{\beta}) = \sum_{l=0}^{D} \phi_l(\mathbf{x}|\mathbf{W},\mathbf{b})\boldsymbol{\beta}_l, \text{ where}$$
$$\phi_l(\mathbf{x}|\mathbf{W},\mathbf{b}) = \sigma_{\texttt{step}}\big(\sum_{i=0}^{h-1} \sigma_{\texttt{step}}\big((\mathbf{x}^\top \mathbf{w}_{i,I(i,l)} + b_{i,I(i,l)})S(i,l)\big) - h\big). \tag{11}$$

In the above equations, $\boldsymbol{\beta}_l \in \mathbb{R}$ is the prediction given by the l^{th} leaf node, h is the height of the tree and D is the number of leaf nodes. $I(i,l)$ denotes the index of the l^{th} leaf's predecessor in the i^{th} level of the tree. $\mathbf{w}_{ij} \in \mathbb{R}^d$ indicates the feature the node is splitting on using one hot encoding, with only one element being 1 or -1 and the rest being 0. $b_{ij} \in \mathbb{R}$ is the corresponding threshold (or the threshold multiplied by -1). The -1 is to guarantee that $\mathbf{x}^\top \mathbf{w}_{i,j} + b_{i,j} > 0$ so that when multiplied by

$$S(i,l) = \begin{cases} -1 & \text{if } l^{th} \text{ leaf } \in \text{ left subtree of node } I(i,l), \\ +1 & \text{otherwise,} \end{cases}$$

the direction of $(\mathbf{x}^\top \mathbf{w}_{i,I(i,l)} + b_{i,I(i,l)})S(i,l)$ can be kept. $\sigma_{\texttt{step}}(\cdot)$ is the step function,

$$\sigma_{\texttt{step}}(a) = 1, \text{ if } a \geq 0, \text{ and } \sigma_{\texttt{step}}(a) = 0, \text{ if } a < 0.$$

Therefore, the model space can be regarded as a three-layer neural network with $\sigma_{\texttt{step}}$ as activation function, with $\mathbf{W}$ as hidden weights and $\mathbf{b}$ as hidden bias.

A.2 Derivation of Posterior Distribution of Variable Importance

Recall from Equation 4 that the posterior distribution of $\boldsymbol{\beta}$ is $\mathcal{MVN}(\mathbb{E}[\boldsymbol{\beta}], \text{Cov}[\boldsymbol{\beta}])$, which can be computed in closed form. This induces a distribution over the variable importance $\psi_j(f)$:

$$\begin{aligned}
\psi_j(f) &= \frac{1}{n}|\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{X})|^\top |\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{X})| \\
&= \frac{1}{n}\boldsymbol{\beta}^\top \big(\frac{\partial}{\partial \mathbf{x}^j}\phi(\mathbf{X})\big)\big(\frac{\partial}{\partial \mathbf{x}^j}\phi(\mathbf{X})\big)^\top \boldsymbol{\beta} \\
&= \frac{1}{n}\boldsymbol{\beta}^\top \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top \boldsymbol{\beta} && \text{(Eigen-decomposition on } \big(\tfrac{\partial}{\partial \mathbf{x}^j}\phi(\mathbf{X})\big)\big(\tfrac{\partial}{\partial \mathbf{x}^j}\phi(\mathbf{X})\big)^\top\text{)} \\
&= \frac{1}{n}\sum_{i=1}^{D} \lambda_i (\mathbf{q}_i^\top \boldsymbol{\beta})^2 && (\lambda_i \text{ is eigenvalue, } \mathbf{q}_i \text{ is eigenvector}) \\
&= \frac{1}{n}\sum_{i=1}^{D} (\lambda_i V_i)\cdot Z_i, && (V_i = \mathbf{q}_i^\top \text{Cov}(\boldsymbol{\beta})\mathbf{q}_i)
\end{aligned}$$

where $Z_i := (\mathbf{q}_i^\top \boldsymbol{\beta})^2/V_i \sim \chi_1^2(\mu_i)$ are independent random variables that follows a noncentral χ^2 distribution with 1 degree of freedom and parameter $\mu_i = (\mathbf{q}_i^\top \mathbb{E}[\boldsymbol{\beta}])^2$. The values $\{\lambda_i \cdot V_i\}_{i=1}^D$ are scalar constants weighting each noncentral χ^2 random variable Z_i. As a result, the full distribution is a well-known distribution of a linear combination of non-central χ^2 distributions [Harville, 1971]. This distribution has mean $\sum_{i=1}^D (\lambda_i V_i)\cdot(1+\mu_i)$, variance $\frac{2}{n}\sum_{i=1}^D (\lambda_i V_i)^2 \cdot (1+2\mu_i)$, and it can be sampled efficiently from by using the linear combination representation as introduced above.

A.3 Algorithm Summary

Given a fixed[4] feature function $\phi : \mathcal{X} \to \mathbb{R}^D$, we present algorithm summaries for (1) Computing the posterior distribution of $\boldsymbol{\beta}$ in the feature-based representation of a Gaussian process, and (2) Computing the posterior distribution of the integrated partial derivative metric.

First consider (1), it involves computing two closed-form updates (for posterior mean and variance) over the training data in mini-batches for one epoch. The algorithm has a linear complexity with respect to data size.

Algorithm 1 Posterior Computation, Feature-based Representation of Gaussian Process

Input: Training data mini-batches $\{(\mathbf{X}_m, \mathbf{y}_m)\}_{m=1}^M$. Fixed feature function $\phi : \mathcal{X} \to \mathbb{R}^D$.
Output: Posterior mean and variance $\mathbb{E}[\boldsymbol{\beta}]_{D\times 1}$, $\text{Cov}[\boldsymbol{\beta}]_{D\times D}$.
Initialize: Feature-label product matrix $\mathbf{P} = \mathbf{0}_{D\times 1}$, covariance matrix $\boldsymbol{\Sigma} = \mathbf{0}_{D\times D}$
for $m = 1$ **to** M **do**
    Compute minibatch feature representation $\boldsymbol{\Phi}_m = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_{n_m})]_{n_m \times D}$
    Update $\mathbf{P} = \mathbf{P} + \Phi_m^\top (\mathbf{y}_m - \Phi_m \boldsymbol{\mu})/\sigma^2$
    Update $\boldsymbol{\Sigma} = \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\Phi_m^\top (\sigma^2 \mathbf{I} + \boldsymbol{\Phi}_m \boldsymbol{\Sigma} \boldsymbol{\Phi}_m^\top)^{-1} \Phi_m \boldsymbol{\Sigma}$ ▷ Equation (5)
end for
Compute $\text{Cov}[\boldsymbol{\beta}] = \boldsymbol{\Sigma}_{\boldsymbol{\beta}} = \boldsymbol{\Sigma}$ ▷ Equation (4)
Compute $\mathbb{E}[\boldsymbol{\beta}] = \boldsymbol{\mu} + \boldsymbol{\Sigma}_{\boldsymbol{\beta}} \mathbf{P}$ ▷ Equation (4)

As shown, during mini-batch computation, the algorithm computes the posterior mean and precision matrix by linearly accumulating the statistic $\Phi_m^\top (\mathbf{y}_m - \Phi_m \boldsymbol{\mu})$, and performs one computation in the end to obtain the $\mathbb{E}[\boldsymbol{\beta}]$. As a result, the space complexity of the algorithm is $O(D^2)$ (for the covariance matrix) and time complexity of the algorithm is $O(nD^3)$ for the matrix inversion. In large-scale applications, the model dimension D is usually fixed and is significantly smaller than the data size n, leading to a linear-time algorithm. Notice that in actual implementation, this algorithm can be made much more efficient (i.e., $O(nD^2)$) by changing how covariance matrix is computed. We introduce this improved algorithm at the end of this section in Algorithm 3.

Now consider (2). Given the posterior of $\boldsymbol{\beta}$ from Algorithm 1, the posterior distribution of the integrated partial derivative metric $\psi_j(f) = \|\frac{\partial}{\partial \mathbf{x}^j} f\|_n^2 = \frac{1}{n} \boldsymbol{\beta}^\top \frac{\partial \Phi}{\partial \mathbf{x}_i^j} \frac{\partial \Phi^\top}{\partial \mathbf{x}_i^j} \boldsymbol{\beta}$ can be computed conveniently by sampling $\boldsymbol{\beta}$ from its posterior.

Algorithm 2 Posterior Computation, Integrated Partial Derivative Metric

Input: Data $\mathbf{X}^*$ with size n^*. Posterior distribution $\mathcal{MVN}(\mathbb{E}[\boldsymbol{\beta}]_{D\times 1}, \text{Cov}[\boldsymbol{\beta}]_{D\times D})$.
Output: Posterior samples of $\psi_j(f)$ of size K: $\{\psi_j(f)_k\}_{k=1}^K$
Sample $\{\boldsymbol{\beta}_k\}_{k=1}^K \sim \mathcal{MVN}(\mathbb{E}[\boldsymbol{\beta}], \text{Cov}[\boldsymbol{\beta}])$
Compute partial derivative feature matrix $[\frac{\partial \Phi}{\partial \mathbf{x}^j}]_{D\times N^*} = [\partial\phi(\mathbf{x}_1)^\top, \ldots, \partial\phi(\mathbf{x}_{N^*})^\top]^\top$
Compute $\mathbf{G}_{j,D\times D} = \frac{\partial \Phi}{\partial \mathbf{x}^j} \frac{\partial \Phi^\top}{\partial \mathbf{x}^j}$
Compute $\psi_j(f)_k = \frac{1}{N^*} \boldsymbol{\beta}_k^\top \mathbf{G}_j \boldsymbol{\beta}_k$ for $k = 1, \ldots, K$ ▷ Equation (8)

When the data size is large, the $\mathbf{G}_j$ matrices can usually be computed as part of Algorithm 1 by accumulating gradient partial derivative matrices $\mathbf{G}_j = \mathbf{G}_j + \frac{\partial \Phi_m}{\partial \mathbf{x}^j} \frac{\partial \Phi_m^\top}{\partial \mathbf{x}^j}$. The time complexity of the algorithm is $O(D^2 n^*)$ which is again a linear-time algorithm with respect to data size n^*. When the data size is extremely large, one can consider reduce computational burden by subsampling from $\mathbf{X}^*$, which is equivalent to performing a Monte Carlo approximation to the integration over the empirical measure (Equation (7)).

Finally, we present a more efficient implementation of Algorithm 1, which improved the run time from $O(nD^3)$ to $O(nD^2)$ by changing how covariance matrix is computed during minibatch accumulation:

[4]Namely, the feature function $\phi(\mathbf{x})$ is either fixed by construction like random feature models or kernel machine using classical kernels (RBF, Matérn, etc). Or $\phi(\mathbf{x})$ is already learned elsewhere (i.e., pre-trained on the same or a separate dataset) like random forests or neural networks.

Algorithm 3 Posterior Computation, Feature-based Representation of Gaussian Process (Version 2)

Input: Training data mini-batches $\{(\mathbf{X}_m, \mathbf{y}_m)\}_{m=1}^M$. Fixed feature function $\phi : \mathcal{X} \to \mathbb{R}^D$.
Output: Posterior mean and variance $\mathbb{E}[\boldsymbol{\beta}]_{D\times 1}$, $\text{Cov}[\boldsymbol{\beta}]_{D\times D}$.
Initialize: Feature-label product matrix $\mathbf{P} = \mathbf{0}_{D\times 1}$, precision matrix $\mathbf{S} = \mathbf{I}_{D\times D}$
for $m = 1$ **to** M **do**
    Compute minibatch feature representation $\boldsymbol{\Phi}_m = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_{n_m})]_{n_m\times D}$
    Update $\mathbf{P} = \mathbf{P} + \Phi_m^\top(\mathbf{y}_m - \Phi_m\boldsymbol{\mu})/\sigma^2$
    Update $\mathbf{S} = \mathbf{S} + \Phi_m^\top\Phi_m/\sigma^2$
end for
Compute $\text{Cov}[\boldsymbol{\beta}] = \boldsymbol{\Sigma}_{\boldsymbol{\beta}} = \mathbf{S}^{-1}$ ▷ Equation (4)
Compute $\mathbb{E}[\boldsymbol{\beta}] = \boldsymbol{\mu} + \boldsymbol{\Sigma}_{\boldsymbol{\beta}}\mathbf{P}$ ▷ Equation (4)

As shown, during mini-batch computation, the algorithm computes the posterior mean and precision matrix by linearly accumulating two statistics $\Phi_m^\top(\mathbf{y}_m - \Phi_m\boldsymbol{\mu})$ and $\Phi_m^\top\Phi_m/\sigma^2$, and performs one matrix inversion in the end to obtain the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\beta}}$ (hence even more efficient than the Woodbury update formula introduced in Algorithm 1, which requires an inversion for every single update step). As a result, the space complexity of the algorithm is $O(D^2)$ (for the covariance matrix) and time complexity of the algorithm is $O(nD^2 + D^3)$. Since in practice, the model dimension D is usually fixed and much smaller than n, the time complexity is in fact $O(nD^2)$.

A.4 Additional Algorithm Summary: Joint Learning of Basis Functions and Featurized GP

Notice that given a fixed set of feature functions $\phi = \{\phi_k\}_{k=1}^D$, the GP posterior can be learned scalably and in closed-form via Algorithm 1 (or Algorithm 3). Therefore, when the feature functions $\phi_{\boldsymbol{\theta}}$ are indexed by hyper-parameters $\boldsymbol{\theta}$ (e.g., the bandwidth parameter of an RBF kernel, or the hidden weights of a deep neural network kernel), we can combine Algorithm 1 (or Algorithm 3) with a pre-existing learning method for the hyperparamters to form a coherent procedure that jointly learns the feature functions and the GP posterior.

Concretely, for example, given a hyperparameter learning procedure `update_hyper` that relies on model prediction $(\mathbb{E}[\boldsymbol{\beta}], \text{Cov}[\boldsymbol{\beta}])$, we can consider the below meta-algorithm for alternative inference:

Algorithm 4 Joint learning of feature functions and the GP posterior. Alternative Inference.

Input: Training data mini-batches $\{(\mathbf{X}_m, \mathbf{y}_m)\}_{m=1}^M$. Fixed feature function $\phi_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}^D$.
Output: Learned hyperparameter $\hat{\boldsymbol{\theta}}$. Posterior mean and variance $\mathbb{E}[\boldsymbol{\beta}|\hat{\boldsymbol{\theta}}]_{D\times 1}$, $\text{Cov}[\boldsymbol{\beta}|\hat{\boldsymbol{\theta}}]_{D\times D}$.
Initialize: Hyperparameter $\boldsymbol{\theta}_0$.
for iterations $t = 1$ **to** T **do**
    Update GP posterior based on $\boldsymbol{\theta}_{t-1}$: $\mathbb{E}[\boldsymbol{\beta}|\boldsymbol{\theta}_{t-1}]$, $\text{Cov}[\boldsymbol{\beta}|\boldsymbol{\theta}_{t-1}]$. (using Algorithm 1 or 3)
    Update hyperparameter $\boldsymbol{\theta}_{t-1} \to \boldsymbol{\theta}_t$ based on GP estimate. (using `update_hyper`)
end for
Set $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_T$.
Compute $\mathbb{E}[\boldsymbol{\beta}|\hat{\boldsymbol{\theta}}]$, $\text{Cov}[\boldsymbol{\beta}|\hat{\boldsymbol{\theta}}]$ (using Algorithm 1 or 3).

This is essentially the idea behind many classical GP kernel learning algorithm. For example, for RBF kernel, $\boldsymbol{\theta}$ can be the kernel bandwidth parameter, and `update_hyper` is the gradient-based update procedure with respect to marginalized likelihood or leave-one-out cross-validation error (see [Rasmussen and Williams, 2005], Chapter 5.4, where both of these quantities are generally computed by integrating over the model's predictive posterior). Or, Algorithm 4 can be a MCMC procedure for the joint inference of $\boldsymbol{\theta}$ and GP posterior, where `update_hyper` is essentially a Metropolis-Hasting or Hamiltonian Monte Carlo step with respect to the model likelihood[5].

Alternatively, the hyperparameters can be learned using a procedure that does not rely on model posterior $(\mathbb{E}[\boldsymbol{\beta}], \text{Cov}[\boldsymbol{\beta}])$. In this case, we can simply first learn the hyperparameter $\hat{\boldsymbol{\theta}}$ (e.g., for neural network kernel, we can learn the hidden weights of the neural networks using SGD with respect to a

[5] In this latter case, the algorithm can be modified to return the full samples of $\{\boldsymbol{\theta}_t\}_{t=1}^T$ and their corresponding conditional posterior samples $\{\mathbb{E}[\boldsymbol{\beta}|\boldsymbol{\theta}_t], \text{Cov}[\boldsymbol{\beta}|\boldsymbol{\theta}_t]\}_{t=1}^T$.

target loss) and then perform posterior inference using Algorithm 1 or 3. Denoting such procedure as `pretrain_hyper`, this leads to the below "pre-training"-style meta-algorithm for joint inference:

Algorithm 5 Joint learning of feature functions and the GP posterior. Pretraining.

1: **Input:** Training data mini-batches $\{(\mathbf{X}_m, \mathbf{y}_m)\}_{m=1}^M$. Fixed feature function $\phi_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}^D$.
2: **Output:** Learned hyperparameter $\hat{\boldsymbol{\theta}}$. Posterior mean and variance $\mathbb{E}[\boldsymbol{\beta}|\hat{\boldsymbol{\theta}}]_{D\times 1}$, $\text{Cov}[\boldsymbol{\beta}|\hat{\boldsymbol{\theta}}]_{D\times D}$.
3: Compute hyperparameter $\hat{\boldsymbol{\theta}}$ using `pretrain_hyper`.
4: Compute $\mathbb{E}[\boldsymbol{\beta}|\hat{\boldsymbol{\theta}}]$, $\text{Cov}[\boldsymbol{\beta}|\hat{\boldsymbol{\theta}}]$ (using Algorithm 1 or 3).

This is essentially the core idea behind some of the classical or state-of-the-art neural Gaussian process algorithms (e.g., [Hinton and Salakhutdinov, 2007] or [Liu et al., 2020]), where a Gaussian process model is fit on top of the basis functions (i.e., the hidden features) of a neural network. There, the hidden weights of the neural network can be first learned via regular SGD-based inference with respect to the MAP objective. Then, in the final epoch, the GP posterior is computed conditional on the learned parameters using Algorithm 1 (see, e.g., Algorithm 1 of [Liu et al., 2020]). The similiar idea can be applied to the tree-based models. For example, the partition rules in a decision-tree kernel can be first learned via a conventional tree-learning procedure [Geurts et al., 2006]. Then, a Gaussian process posterior can be fitted to the decision tree by leveraging its featurized representation (Equation (11)).

B Featurized Representation of ML Models

The second key advantage of the feature-based representation (3) is its generality: a wide range of machine learning models can be written in term of the feature-based form $f(\mathbf{x}) = \phi(\mathbf{x})^\top \boldsymbol{\beta}$ Rahimi and Recht [2007], Davies and Ghahramani [2014], Lee et al. [2017], making the Gaussian process a unified framework for quantifying model uncertainty with a wide array of modern machine learning models. This section enumerates a few important examples:

Generalized Additive Models (GAM). For a regression task with d input features, a generalized additive model (GAM) has the form $f(\mathbf{x}) = \beta_0 + \sum_{j=1}^{d} \beta_j h_j(\mathbf{x}^j)$, where $h_j' s$ are flexible functions (e.g., splines) with bounded norm Hastie et al. [2009]. GAM induces a d-dimensional feature representation ([Hastie et al., 2009], Chapter 9):

$$\phi(\mathbf{x})_{d\times 1} = [1, h_1(\mathbf{x}^1), \ldots, h_d(\mathbf{x}^d)],$$

where $h_j' s$ are usually spline functions that are differentiable. In the special case where all $h_j' s$ are identity functions, GAM reduces to a linear model, and the corresponding $f = \phi(\mathbf{x})^\top \boldsymbol{\beta}$ becomes a GP with linear kernel.

Decision Trees. By partitioning the whole feature space into D cells $\mathcal{X} = \cup_{j=1}^{D} \mathcal{X}_j$, a decision tree model essentially induces a one-hot feature map, e.g.,

$$\phi(\mathbf{x})_{D\times 1} = [0, \ldots, 1, \ldots, 0],$$

where each element is a indicator function $\mathbb{1}(\mathbf{x} \in \mathcal{X}_j)$ for whether the data point $\mathbf{x}$ falls into the j^{th} cell (Figure 2). This connection is crucial for extending Gaussian process treatment to tree models. Appendix F.1 introduce this formulation in more detail. Following the same construction, the features learned by the majority of partition-based learning methods (e.g., CART, PRIM, etc.) can be used to construct Gaussian process kernels.

Random Feature Models. The random-feature model takes the form:

$$\phi(\mathbf{x})_{D\times 1} = \sqrt{2}\sigma(\mathbf{W}^\top \mathbf{x} + \mathbf{b}),$$

where $\mathbf{W}_{d\times D}$ and $\mathbf{b}_{D\times 1}$ are frozen weights initialized from i.i.d. samples from certain fixed distributions, and σ is an activation function. For example, in the case of classical random Fourier features whose inner product approximates the RBF kernel, we have $\sigma(\cdot) = \cos(\cdot), \mathbf{W} \overset{iid}{\sim} N(0,1), \mathbf{b} \overset{iid}{\sim} Unif(0, 2\pi)$ Liu et al. [2021]. Although first introduced as a scalable approximation to GP models equipped with certain kernels (e.g., radial basis function (RBF)), the modern literature treats it as a standalone class of models with its own unique set of theoretic guarantees [Mei and Montanari, 2019, Jacot et al., 2020, Rahimi and Recht, 2009].

(Deep) Neural Networks. For a trained L-layer neural network of the from $f(\mathbf{x}) = \boldsymbol{\beta}^\top g_L \cdot g_{L-1} \cdots \cdot g_0(\mathbf{x})$ with $g_l(\mathbf{x}) = \sigma_l(\mathbf{W}_l^\top \mathbf{x} + \mathbf{b}_l)$, the last-layer representation function

$$\phi(\mathbf{x}) = g_L \cdot g_{L-1} \cdots \cdot g_1(\mathbf{x})$$

can be understood as the feature map. Then, the feature map can be used to construct the Gaussian process kernel $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$. This approach was studied extensively in prior literature, due to a neural network's appealing ability in learning an effective representation for the task at hand [Hinton and Salakhutdinov, 2007, Calandra et al., 2016]. Works like [Wilson et al., 2016a,b, Liu et al., 2020] further extended this in the context of modern deep learning.

Ensembles. An ensemble model of linear models, trees, or neural networks can be written as a mixture of Gaussian processes. Specifically, an ensemble model can be written as $f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})$, where $h_m' s$ are weak learners such as linear models, trees, or neural networks, and α_m are model weights that are either learned or set to uniform $\frac{1}{M}$. This formulation covers well-known examples such as AdaBoost, boosted trees, and random forests Hastie et al. [2009]. As introduced above, since many classical weak learners $h_m = \phi_m(\mathbf{x})^\top \boldsymbol{\beta}_m$ induces a Gaussian process with kernel k_m via their feature representation $k_m(\mathbf{x}, \mathbf{x}') = \phi_m(\mathbf{x}')^\top \phi_m(\mathbf{x})$, the full ensemble model induces a mixture of Gaussian processes with fixed mixing weights dictated by the ensemble weights $\{\alpha\}_{m=1}^{M}$. That is, the ensemble induces a Bayesian model $f'(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h'(\mathbf{x})$ where α_m's are fixed constants and $h_m'(\mathbf{x})$'s are Gaussian process models with prior $\mathcal{GP}(0, k_m)$. In the actual implementation, we

fit each of the individual GP model $h'_m(\mathbf{x})$ following exactly how it is done in the original ensemble model. For example, for random forest models, we fit each $h'_m(\mathbf{x})$ models independently with respect to the original label y. While not a focus of this work, for gradient boosting models, we fit h'_m's recursively with respect to the residual $y - \sum_{l<m} \alpha_l h'_l(\mathbf{x})$ [Sigrist, 2020].

B.1 Summary of model assumptions

This section introduces the theoretical conditions on the featurized model $f(\mathbf{x}) = \phi(\mathbf{x})^\top \boldsymbol{\beta}$ that is required by the proposed framework (i.e., for satisfying Theorems 1 and 2). The intention of this section is to provide a centralized place for readers to verify whether a general model would fit into the framework, and provide pointers to more detailed discussions (e.g., the interpretation of these conditions for important model classes).

First recall that our measure of variable importance $\psi_j(f) = ||D_j f||_n^2$ relies on the differentiation operator D_j. For continuous features $x^j \in \mathbb{R}$, the differentiation operator is defined as the conventional partial derivative: $D_j : f \to \frac{\partial}{\partial \mathbf{x}^j} f$, while for discrete features $x^j \in \{0, 1\}$, we overload the derivative operator as the discrete difference $\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{x}) = f(\mathbf{x}^j = 1, \mathbf{x}^{-j}) - f(\mathbf{x}^j = 0, \mathbf{x}^{-j})$. In such way, the differentiation operator is well-defined for both types of features.

Under the above-defined notion of differentiation operator D_j, the proposed framework imposes below three assumptions on the fitted model $f(\mathbf{x}) = \phi(\mathbf{x})^\top \hat{\boldsymbol{\beta}}$ with $\hat{\boldsymbol{\beta}} \in \mathbb{R}^D$:

1. Weak differentiability. For the set of causal features $j^* \in \mathcal{A}^*$, the partial derivative $D_{j^*} f(\mathbf{x})$'s exist almost everywhere for $\mathbf{x} \in \mathcal{X}$, so that $||D_{j^*} f(\mathbf{x})||_2^2 > \delta$ for a small positive $\delta > 0$ with non-zero probability $\forall j^* \in \mathcal{A}^*$.

This is a basic condition to ensures the model f's partial derivative is well-defined and can capture the true variable importance $\Phi_{j^*}(f_0) = ||\frac{\partial}{\partial \mathbf{x}^{j^*}} f_0||_2^2$. This condition is weaker than full differentiability, and allows model to have non-differentiable or discontinuous points at discrete locations (e.g., a ReLU network whose gradient is not well-defined at 0). However, we do note that the fully non-differentiable models (e.g., decision trees) can still be incorporated into our framework by applying differentiable approximations (see Appendix F). Remark 2 in Appendix C) provides additional relevant discussion of this condition.

2. Lipschitz condition. The model f should be Lipschitz with a bounded Lipschitz constant $C < \infty$, so that $\frac{|f(\mathbf{x}_1) - f(\mathbf{x}_2)|}{||\mathbf{x}_1 - \mathbf{x}_2||_2} \leq C$ for all pairs $(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{X} \times \mathcal{X}$. Here $||\mathbf{x}_1 - \mathbf{x}_2||_2$ is the L_2 metric in $\mathcal{X}$.

The Lipschitz condition is a mild requirement indicating that the model gradient is well-behaved (i.e., bounded away from infinity). It is essential for the posterior concentration of the variable importance metric, and is used to satisfy condition (2) of Theorem 1 (Section 3.2). Appendix E.1 discusses the Lipschitz condition of important model classes, including Generalized additive models, deep neural networks, and decision trees under differentiable approximation.

3. Growth condition on model rank ($o_p(\sqrt{n})$). Given the feature map $\phi : \mathcal{X} \in \mathbb{R}^D$ and the data-generating distribution $P(\mathbf{x})$, the rank of the model space $\mathcal{H}$ can be empirically measured as $\lim_{n\to\infty} rank(\Phi)$, where $\Phi_{n\times D} = [\phi^\top(\mathbf{x}_1), \ldots, \phi^\top(\mathbf{x}_n)]^\top$ is the feature matrix evaluated at $\mathbf{x}_i \sim P(\mathbf{x})$.

To this end, we require the growth condition of the model rank to not increase faster than $o_p(\sqrt{n})$, so that the model complexity is well-controlled. This condition is essential for the BvM phenomenon, and is used to satisfy condition (4) of Theorem 2 (see Section 3.2 for related discussion). This condition is easily satisfied by ML models in practice. For example, since the model rank is upper bounded by the dimension of the feature map D, models with fixed feature dimension trivially satisfies this condition. This growth condition can also be satisfied by adaptive-rank models such random forest or sparse neural networks, and are in fact much less stringent than what is required by the BvM theorems in the existing literature. See Appendix E.2 for detailed discussion.

C Proof for Posterior Convergence

Proof for Theorem 1
Recall the list of technical conditions:

i) **(Convergence of Prediction Function f)** The posterior distribution $\Pi_n(f)$ converges toward f_0 at a rate of ϵ_n. (Note that in nonparametric learning setting, this rate is not faster than $O_p(n^{-\frac{1}{2}})$ which is the optimal parametric rate);

ii) **(Well-conditioned Derivative Functions)** $D_j : f \to \frac{\partial}{\partial \mathbf{x}^j} f$ the differentiation operator is bounded: $\|D_j\|_{op}^2 = \inf\{C \geq 0 : \|D_j f\|_2^2 \leq C\|f\|_2^2, \text{ for all } f \in \mathcal{H}_\phi\}$;

Proof. Denote $A_n = \{f : \|f - f_0\|_n^2 > M_n\epsilon_n\}$ and $B_n = \{f : |\psi_j(f) - \Psi_j(f_0)| > M_n\epsilon_n\}$, then showing the statement in Theorem 1 is equivalent to showing $\Pi_n(B_n) \to 0$.

Specifically, we assume below two facts hold:

Fact 1. $|\psi_j(f) - \psi_j(f_0)| \leq \|D_j f - D_j f_0\|_n^2$

Fact 2. $\sup_{j\in\{1,\ldots,d\}} |\psi_j(f_0) - \Psi_j(f_0)| \lesssim \|f - f_0\|_n^2$

Because if the above facts hold, we then have

$$\begin{aligned}
\sup_{j\in\{1,\ldots,d\}} |\psi_j(f) - \Psi_j(f_0)| &\leq \sup_{j\in\{1,\ldots,d\}} |\psi_j(f) - \psi_j(f_0)| + \sup_{j\in\{1,\ldots,d\}} |\psi_j(f_0) - \Psi_j(f_0)| \\
&\leq \sup_{j\in\{1,\ldots,d\}} \|D_j f - D_j f_0\|_n^2 + \sup_{j\in\{1,\ldots,d\}} |\psi_j(f_0) - \Psi_j(f_0)| \\
&\leq \sup_{j\in\{1,\ldots,d\}} \|D_j f - D_j f_0\|_2^2 + O_p(n^{-\frac{1}{2}}) + \sup_{j\in\{1,\ldots,d\}} |\psi_j(f_0) - \Psi_j(f_0)| \\
&\leq C\|f - f_0\|_2^2 + O_p(n^{-\frac{1}{2}}) + \sup_{j\in\{1,\ldots,d\}} |\psi_j(f_0) - \Psi_j(f_0)| \\
&\qquad\qquad (D_j \text{ is bounded}) \\
&\leq C\|f - f_0\|_n^2 + O_p(n^{-\frac{1}{2}}) + \sup_{j\in\{1,\ldots,d\}} |\psi_j(f_0) - \Psi_j(f_0)| \\
&\lesssim \|f - f_0\|_n^2.
\end{aligned}$$

It then follows that:

$$\mathbb{E}_0\Pi_n\Big(\sup_{j\in\{1,\ldots,d\}} |\psi_j(f) - \Psi_j(f_0)| \geq M_n\epsilon_n\Big) \lesssim \mathbb{E}_0\Pi_n\Big(\|f - f_0\|_n^2 \geq M_n'\epsilon_n\Big) \to 0.$$

We now show Facts 1 and 2 are true.

- **Fact 1** follows simply from the triangular inequality:

$$\begin{aligned}
|\psi_j(f) - \psi_j(f_0)| &= \Big|\|D_j f\|_n^2 - \|D_j f_0\|_n^2\Big| \\
&= \max\Big\{\|D_j f\|_n^2 - \|D_j f_0\|_n^2,\ \|D_j f_0\|_n^2 - \|D_j f\|_n^2\Big\} \leq \|D_j f - D_j f_0\|_n^2.
\end{aligned}$$

- **Fact 2** follows from standard Bernstein-type concentration inequality (see, e.g., Lemma 18 of Rosasco et al. [2013]). Specifically, for $|D_j f_0(\mathbf{x})|^2$ a random variable with respect to probability measure $P(\mathbf{x})$ that is bounded by L. Given n iid samples $\{|D_j f_0(\mathbf{x}_i)|^2\}_{i=1}^n$, recall that $\psi_j(f_0) = \frac{1}{n}\sum_{i=1}^n |D_j f_0(\mathbf{x}_i)|^2$ and $\Psi(f_0) = \mathbb{E}(|D_j f_0|^2)$, then with probability $1 - \eta$:

$$|\psi_j(f_0) - \Psi(f_0)| \leq n^{-\frac{1}{2}} * \big(2\sqrt{2} * L * \log(2/\eta)\big),$$

that is, $|\psi_j(f_0) - \Psi(f_0)| \to 0$ at the rate of $O(n^{-\frac{1}{2}})$. Notice that $O(n^{-\frac{1}{2}})$ is the optimal parametric rate that cannot be surpassed by the convergence speed of the ReLU networks

(recall the typical convergence rate is $\epsilon_n \asymp n^{-\frac{\beta}{2\beta+\delta}} * \log(n)^\gamma$ for some $\delta > 0$ and $\gamma > 1$). Therefore we have:

$$\sup_{j\in\{1,\ldots,d\}} |\psi_j(f_0) - \Psi_j(f_0)| \lesssim \|f - f_0\|_n^2.$$

□

Remark 1 (**Convergence of L_2 norm**). The sample L_2 norm and the expected L_2 norm are closed to each other at the rate of $O(n^{-\frac{1}{2}})$. This will happen when $\mathbf{x}$'s are sampled randomly from a probability measure $P(\mathbf{x})$.

Remark 2 (**A condition on weak differentiabilty**). Although not listed explicitly in the main theorem, we also impose a weak technical condition (i.e., Non-trivial Gradient Function) on model function f and true function f_0 to avoid certain pathological situations:

iii) **(Non-trivial Derivative Functions)** Denote $j^* \in \{1, \ldots, d^*\}$ the index of the causal variables, and recall $P_{\mathcal{X}}(\mathbf{x})$ the distribution of the input features $\mathbf{x}$. Then there exists $\delta > 0$ such that for all $j^* \in \{1, \ldots, d^*\}$, $||D_{j^*} f_0(\mathbf{x})||_2^2 > \delta$ and $||D_{j^*} f(\mathbf{x})||_2^2 > \delta$ with non-zero probability.

Note that this condition is weak in that it only requires the partial derivative under model function f and f_0 are not zero almost everywhere. For differentiable functions under continuous features, this should be satisfied by definition. This basic technical condition is intended to remove two pathological situations. The first is non-differentiable models (e.g., tree models), whose gradient is zero almost everywhere in the feature space. The second case are the discrete features, where the traditional sense of partial derivative is not well defined. In Appendix F, we discuss how to incorporate non-differentiable models and discrete features into our framework. Briefly, a non-differentiable model (e.g., partition-based models) can be made differentiable by employing a differentable approximation. For discrete features, we can compute the discrete version of the differentiable operator, e.g., $D_j f(\mathbf{x}) = f(\mathbf{x}^j = 1, \mathbf{x}^{-j}) - f(\mathbf{x}^j = 0, \mathbf{x}^{-j})$ for binary feature where $\mathbf{x}_{d\times 1} = [\mathbf{x}^j, [\mathbf{x}^{-j}]^\top_{(d-1)\times 1}]^\top$ (known as *contrast* in statistics). Notice that this discrete differentiation operator $D_j f(\mathbf{x})$ is a linear function of the original prediction function f. As a result, the posterior convergence of ψ_j with respect to this operator is again guaranteed by the convergence of the prediction f.

Remark 3 (**Convergence guanrantee of f**). Note that our result focuses on posterior concentration of variable importance ψ_j, not of prediction function f. In fact, the convergence of ψ_j depends on the convergence of the prediction function f, as introduced in the assumptions of 1. In practice, it is up to the practitioners to select a proper prediction model f that has a convergence guarantee for the task at hand. Specifically, we showed that for any model, if its prediction function has a posterior concentration guarantee, its variable importance has a convergence guarantee as well. To this end, we notice that majority of popular machine learning methods (e.g., random features, neural networks, tree ensembles) has a posterior concentration guarantee for target functions in certain general function space (e.g., the space of α-Hölder space), given the recent advances in the approximation and convergence guarantees of parametric (finite-dimensional) ML models in both frequentist and Bayesian settings Ročková and van der Pas [2020], Wang and Rocková [2020], Liu [2021], Schmidt-Hieber [2020].

Furthermore, we note that although the ML models covered in our work are not traditional universal kernels Micchelli et al. [2006], most of them (e.g., random features, neural networks, tree ensembles) do come with a universal approximation guarantee for an appropriately defined function class Rahimi and Recht [2008], Biau [2012], Schmidt-Hieber [2020]. As a result, the kernel functions defined by these models provide basis functions that span function spaces that are often dense in an infinite-dimensional RKHS, implying that the resulting model can approximate f_0 to arbitrary precision Rahimi and Recht [2008]. Please see Rahimi and Recht [2008], Hornik et al. [1989], Biau [2012] for specific results for random features, neural networks and random forests.

Remark 4 (**Compatibility with high-dimensional settings**). Although in the main text, we assumed the data dimension d is fixed with respect to n (see Section 2, Problem Setup). Theorem 1 in fact does not rely on a fixed d, and the proof can go through whenever $d = o(n)$. That is to say, the posterior concentration of variable importance measure $\psi_j(f)$ can occur even in the high-dimensional settings of d, which is allowed to grow with sample size n. In comparison, the ***Bernstein-von Mises***

(BvM) result in Theorem 2 implies a stronger sense of convergence (i.e., convergence in distribution) and requires more stringent conditions (i.e., fixed data dimension d). This is in fact consistent with modern BvM analysis of Bayesian ML models, where a fixed data dimension is commonly assumed [Wang and Rocková, 2020, Rockova, 2020, Yang et al., 2015, Burnaev et al., 2013, Castillo and Rousseau, 2015].

C.1 A finite-sample error bound

Although Theorem 1 is stated as an asymptotic result, we note that if the error bound for the prediction function ϵ_n (Condition 1) is a finite-sample error bound, then a finite-sample error bound for the variable importance $\psi_j(f)$ can be trivially derived by extending the proof of Theorem 1. For example, we can have the below finite-sample concentration result:

Proposition 3 (Finite-sample Error Bound for $\psi_j(f)$)**.** *Assume*

- *i)* ***(Finite-sample Error Bound for Prediction Function** f**)** *The posterior distribution of f on average converges toward f_0 at a finite-sample rate of ϵ_n, such that $\mathbb{E}_0 E_n \|f - f_0\|_n^2 \leq \epsilon_n$ and ϵ_n is finite-sample error bound that is an explicit function of n, and E_n is the expectation with respect to the posterior distribution Π_n.*
- *ii)* ***(Well-conditioned Derivative Functions)*** *$D_j : f \to \frac{\partial}{\partial \mathbf{x}^j} f$ the differentiation operator is bounded: $\|D_j\|_{op}^2 = \inf\{C \geq 0 : \|D_j f\|_2^2 \leq C\|f\|_2^2$, for all $f \in \mathcal{H}_\phi\}$;*

Also assumes the data $\mathbf{x} \sim P(\mathbf{x})$ is generated with respect to probability measure $P(\mathbf{x})$ that is bounded by L. Then, with probability $1 - \eta$, the posterior distribution for $\psi_j(f) = \|\frac{\partial}{\partial \mathbf{x}^j} f\|_n^2$ contracts toward $\Psi_j(f_0) = \|\frac{\partial}{\partial \mathbf{x}^j} f_0\|_{P_\mathcal{X}}^2$ at a finite-sample rate ϵ_n' such that:

$$\mathbb{E}_0 E_n\big(\sup_{j \in \{1,\ldots,d\}} |\psi_j(f) - \Psi_j(f_0)|\big) \leq \epsilon_n'$$

where

$$\epsilon_n' = C * \epsilon_n + n^{-\frac{1}{2}} * \big(2\sqrt{2} * L * \log(2/\eta)\big).$$

Proof. Recall in **Fact 2** of the proof of Theorem 1. By Bernstein inequality, we have, with probability $1 - \eta$:

$$|\psi_j(f_0) - \Psi(f_0)| \leq n^{-\frac{1}{2}} * \big(2\sqrt{2} * L * \log(2/\eta)\big).$$

Then, following the same line of argument as the proof of Theorem 1, we have:

$$\begin{aligned}
\sup_{j \in \{1,\ldots,d\}} |\psi_j(f) - \Psi_j(f_0)| &\leq \sup_{j \in \{1,\ldots,d\}} |\psi_j(f) - \psi_j(f_0)| + \sup_{j \in \{1,\ldots,d\}} |\psi_j(f_0) - \Psi_j(f_0)| \\
&\leq \sup_{j \in \{1,\ldots,d\}} \|D_j f - D_j f_0\|_n^2 + \sup_{j \in \{1,\ldots,d\}} |\psi_j(f_0) - \Psi_j(f_0)| \\
&\leq C\|f - f_0\|_n^2 + \sup_{j \in \{1,\ldots,d\}} |\psi_j(f_0) - \Psi_j(f_0)|. \\
&\leq C\|f - f_0\|_n^2 + n^{-\frac{1}{2}} * \big(2\sqrt{2} * L * \log(2/\eta)\big).
\end{aligned}$$

Now, take expectation $\mathbb{E}_0 E_n$ for both sides, we arrive at:

$$\begin{aligned}
\mathbb{E}_0 E_n\Big(\sup_{j \in \{1,\ldots,d\}} |\psi_j(f) - \Psi_j(f_0)|\Big) &\leq C\mathbb{E}_0 E_n\|f - f_0\|_n^2 + n^{-\frac{1}{2}} * \big(2\sqrt{2} * L * \log(2/\eta)\big) \\
&= C\epsilon_n + n^{-\frac{1}{2}} * \big(2\sqrt{2} * L * \log(2/\eta)\big).
\end{aligned}$$

□

D Proofs for Asymptotic Normality

Lemma 4. *Functional Delta Method (univariate) Suppose $\mathcal{P}_n$ is the empirical distribution of a random sample $X_1, \ldots, X_n$ from a distribution P, and ϕ is a function that maps the distribution of interest into some space. Define the Gateaux derivative*

$$\phi'_P(\delta_x - P) = \frac{\mathrm{d}}{\mathrm{d}t} \mid_{t=0} \phi((1-t)P + t\delta_x) = IF_{\phi,P}(x),$$

which is also the Influence Function, and $\gamma^2 = \int IF_{\phi,P}(x)^2 dP$. If integration and differentiation can be exchanged, then

$$\int \phi'_P(\delta_x - P) dP = 0.$$

Further, if $\sqrt{n}R_n \xrightarrow{P} 0$, where

$$R_n = \phi(\mathcal{P}_n) - \phi(P) - \frac{1}{n}\sum_i \phi'_P(\delta_{x_i} - P),$$

then from the Central Limit Theory that

$$\sqrt{n}(\phi(\mathcal{P}_n) - \phi(P)) \xrightarrow{d} \mathcal{N}(0, \gamma^2).$$

Lemma 5. *Functional Delta Method (multivariate) Suppose $\mathcal{P}_n$ is the empirical distribution of a random sample $X_1, \ldots, X_n$ from a distribution P, and $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^k$. Define the Gateaux derivative*

$$\boldsymbol{\phi}'_P(\delta_x - P) = \frac{\mathrm{d}}{\mathrm{d}t} \mid_{t=0} \boldsymbol{\phi}((1-t)P + t\delta_x) = IF_{\boldsymbol{\phi},P}(x),$$

which is also the Influence Function, and $[\mathbf{V}_0]_{i,j} = \int \langle [IF_{\boldsymbol{\phi},P}(x)]_i, [IF_{\boldsymbol{\phi},P}(x)]_j \rangle dP$. If integration and differentiation can be exchanged, then

$$\int \boldsymbol{\phi}'_P(\delta_x - P) dP = 0.$$

Further, if $\sqrt{n}\mathbf{R}_n \xrightarrow{P} 0$, where

$$\mathbf{R}_n = \boldsymbol{\phi}(\mathcal{P}_n) - \boldsymbol{\phi}(P) - \frac{1}{n}\sum_i \boldsymbol{\phi}'_P(\delta_{x_i} - P),$$

then from the Central Limit Theory that

$$\sqrt{n}(\boldsymbol{\phi}(\mathcal{P}_n) - \boldsymbol{\phi}(P)) \xrightarrow{d} \mathcal{MVN}(0, \mathbf{V}_0).$$

Proof for Theorem 2
To make our assumptions explicit, we list out a collection of easily-satisfied technical conditions.

(1) f is a consistent estimator of f_0;
(2) D_j is bounded: $\|D_j\|_{op}^2 = \inf\{C \geq 0 : \|D_j f\|_2^2 \leq C\|f\|_2^2, \text{ for all } f \in \mathcal{H}_\phi\}$;
(3) f_0 is square-integrable over the support of X and $\|f_0\|_2 = 1$;
(4) $\texttt{rank}(H_j) = o_p(\sqrt{n})$;

Proof. Since $H_j = D_j^\top D_j$, Condition (2) is equivalent to the largest eigenvalue of H_j being bounded, i.e., $\lambda_{max}(H_j) = O_p(1)$. From the definition in Equation (10), we have

$$\psi'_j(f) = \frac{\partial}{\partial f}\psi_j(f) = \frac{2}{n}H_j f.$$

Define a mean functional $m : F \to E(F)$, where F is the distribution. Then in our case, $f_0 = E(F) = m(F)$. According to Lemma 4, we have

$$\psi_j(f_0) = \psi_j(E(F)) = \psi_j(m(F)) = \phi(F),$$

i.e., $\phi(\cdot) = \psi_j(m(\cdot))$. Therefore,

$$\begin{aligned}
\phi'_F(\delta_y - F) &= \psi'_j(m(\delta_y - F)) \\
&= \frac{\mathrm{d}}{\mathrm{d}t} \mid_{t=0} \psi_j(m((1-t)F + t\delta_y)) \\
&= \frac{\mathrm{d}}{\mathrm{d}t} \mid_{t=0} \psi_j((1-t)f_0 + ty) \\
&= \frac{\mathrm{d}}{\mathrm{d}t} \mid_{t=0} \frac{1}{n}[(1-t)f_0 + ty]^\top H_j[(1-t)f_0 + ty] \\
&= \frac{2}{n}(y - f_0)^\top H_j f_0 \\
&= IF_{\phi,F}(y).
\end{aligned}$$

On the other hand,

$$\begin{aligned}
\gamma^2 &= \int IF_{\phi,F}(y)^2 dF \\
&= 4\int \frac{1}{n} \cdot f_0^\top H_j(y - f_0)(y - f_0)^\top H_j f_0 \cdot \frac{1}{n} dF \\
&= 4\sigma^2 \|H_j f_0\|_n^2.
\end{aligned}$$

Moreover, we have

$$\int \phi'_F(\delta_y - F)dF = \frac{2}{n}\int (y - f_0)^\top H_j f_0 dF = 0,$$

and

$$\begin{aligned}
\sqrt{n}R_n &= \sqrt{n}[\phi(\mathcal{F}_n) - \phi(F) - \frac{1}{n}\sum_i \phi'_F(\delta_{y_i} - F)] \\
&= \sqrt{n}[\psi_j(f) - \psi_j(f_0) - \frac{1}{n} \cdot \frac{2}{n}\sum_i (y_i - f_{0,i})^\top [H_j f_0]_i] \\
&= \sqrt{n}[\frac{1}{n} \cdot (f^\top H_j f - f_0^\top H_j f_0) - \frac{1}{n} \cdot \frac{2}{n}(y - f_0)^\top H_j f_0] \\
&= \frac{1}{\sqrt{n}}[f^\top H_j f - f^\top H_j f_0 + f^\top H_j f_0 - f_0^\top H_j f_0 - \frac{2}{n}(y - f_0)^\top H_j f_0] \\
&= \frac{1}{\sqrt{n}}[(f - f_0)^\top H_j(f + f_0) - \frac{2}{n}(y - f_0)^\top H_j f_0] \\
&= \frac{1}{\sqrt{n}}[(f - f_0)^\top H_j(f - f_0) + 2(f - f_0)^\top H_j f_0 - \frac{2}{n}(y - f_0)^\top H_j f_0] \\
&= \frac{1}{\sqrt{n}}[\hat{\epsilon}_n^\top H_j \hat{\epsilon}_n + 2\hat{\epsilon}_n^\top H_j f_0 - \frac{2}{n}(y - f_0)^\top H_j f_0] \qquad (12) \\
&= \frac{1}{\sqrt{n}} o_p(\sqrt{n}) \qquad (13) \\
&= o_p(1) \xrightarrow{P} 0,
\end{aligned}$$

where $\hat{\epsilon}_n = f - f_0$. We can prove the result from Equation (12) to Equation (13) as following: Denote $k = \mathtt{rank}(H_j)$, then the eigendecomposition of H_j is $H_j = U_j \Lambda U_j^\top$, with $U_j = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$ a $n \times k$ orthogonal matrix and Λ a $k \times k$ diagonal matrix with elements $\{\lambda_i\}_{i=1}^k$ being the eigenvalues of H_j, then define

$$\mathbf{v} = U_j^\top \hat{\epsilon}_n = \begin{bmatrix} \mathbf{u}_1^\top \hat{\epsilon}_n \\ \vdots \\ \mathbf{u}_k^\top \hat{\epsilon}_n \end{bmatrix}.$$

Therefore,

$$\begin{aligned}
\hat{\epsilon}_n^\top H_j \hat{\epsilon}_n = \mathbf{v}^\top \Lambda \mathbf{v} &= \sum_{i=1}^{k} \lambda_i v_i^2 \\
&\leq \lambda_{max}(H_j) \sum_{i=1}^{k} v_i^2 \\
&= \lambda_{max}(H_j) \sum_{i=1}^{k} \mathbf{u}_i^\top \hat{\Sigma}_n \mathbf{u}_i \\
&\leq \lambda_{max}(H_j) \sum_{i=1}^{k} \lambda_{max}(\hat{\Sigma}_n) \\
&= k \cdot \lambda_{max}(H_j) \cdot \lambda_{max}(\hat{\Sigma}_n) \\
&= o_p(\sqrt{n}) \cdot O_p(1) \cdot O_p(1) \\
&= o_p(\sqrt{n}),
\end{aligned}$$

where $E(\hat{\epsilon}_n) = \mathbf{0}$, $\text{cov}(\hat{\epsilon}_n) = \hat{\Sigma}_n$, and $\lambda_{max}(\hat{\Sigma}_n)$ is the largest eigenvalue of $\hat{\Sigma}_n$.

On the other hand, $2\hat{\epsilon}_n^\top H_j f_0 = o_p(\sqrt{n})$ because f is a consistent estimator of f_0. Moreover, since $y_i - f_{0,i} = O_p(1)$, we know $\frac{2}{n}(y - f_0)^\top H_j f_0 = o_p(\sqrt{n})$. So,

$$\hat{\epsilon}_n^\top H_j \hat{\epsilon}_n + 2\hat{\epsilon}_n^\top H_j f_0 - \frac{2}{n}(y - f_0)^\top H_j f_0 = o_p(\sqrt{n}) \tag{14}$$

Therefore, by Lemma 4, we have

$$\sqrt{n}(\psi_j(f) - \psi_j(f_0)) \xrightarrow{d} \mathcal{N}(0, 4\sigma^2 \|H_j f_0\|_n^2).$$

□

Theorem 6 (Asymptotic Distribution of Variable Importance (multivariate))**.** *Suppose* $y_i = f_0(\mathbf{x}_i) + e_i,\ e_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2),\ i = 1, \ldots, n$. *Denote* $\boldsymbol{\psi} = [\psi_1, \ldots, \psi_d]$ *for* ψ_j *as defined in Equation* (10)*. If the following conditions are satisfied:*

i) $\texttt{rank}(H_j) = o_p(\sqrt{n}), j = 1, \ldots, d$*;*

ii) f_0 *is square-integrable over the support of* X *and* $\|f_0\|_2 = 1$*;*

iii) f *is a consistent estimator of* f_0*;*

iv) D_j *is bounded:* $\|D_j\|_{op}^2 = \inf\{C \geq 0 : \|D_j f\|_2^2 \leq C\|f\|_2^2,$ *for all* $f \in \mathcal{H}_\phi\}$.

Then $\boldsymbol{\psi}(f)$ *asymptotically converges toward a multivariate normal distribution surrounding* $\boldsymbol{\psi}(f_0)$*, i.e.,*

$$\sqrt{n}(\boldsymbol{\psi}(f) - \boldsymbol{\psi}(f_0)) \xrightarrow{d} \mathcal{MVN}(\mathbf{0}, \mathbf{V}_0),$$

where $\mathbf{V}_0$ *is a* $d \times d$ *matrix such that* $[\mathbf{V}_0]_{j1,j2} = 4\sigma^2 \langle H_{j1} f_0, H_{j2} f_0 \rangle_n$.

Proof. Define a mean function $m : F \to E(F)$, where F is the distribution. Then in our case, $f_0 = E(F) = m(F)$. According to Lemma 5, we have

$$[\boldsymbol{\psi}(f_0)]_j = \psi_j(E(F)) = \psi_j(m(F)) = [\boldsymbol{\phi}(F)]_j,$$

i.e., $\boldsymbol{\phi}(\cdot) = \boldsymbol{\psi}(m(\cdot))$ and $[\boldsymbol{\phi}(\cdot)]_j = \psi_j(m(\cdot))$, where $\boldsymbol{\phi} : \mathcal{R} \to \mathcal{R}^P$. Therefore,

$$\begin{aligned}
[\boldsymbol{\phi}'_F(\delta_y - F)]_j &= \psi'_j(m(\delta_y - F)) \\
&= \frac{\mathrm{d}}{\mathrm{d}t} \mid_{t=0} \psi_j(m((1-t)F + t\delta_y)) \\
&= \frac{\mathrm{d}}{\mathrm{d}t} \mid_{t=0} \psi_j((1-t)f_0 + ty) \\
&= \frac{\mathrm{d}}{\mathrm{d}t} \mid_{t=0} \frac{1}{n}[(1-t)f_0 + ty]^\top H_j[(1-t)f_0 + ty] \\
&= \frac{2}{n}(y - f_0)^\top H_j f_0 \\
&= [IF_{\boldsymbol{\phi},F}(y)]_j.
\end{aligned}$$

On the other hand,

$$\begin{aligned}
[\mathbf{V}_0]_{j1,j2} &= \int \langle [IF_{\boldsymbol{\phi},F}(y)]_{j1}, [IF_{\boldsymbol{\phi},F}(y)]_{j2} \rangle dF \\
&= 4 \int \frac{1}{n} \cdot f_0^\top H_{j1}(y - f_0)(y - f_0)^\top H_{j2} f_0 \cdot \frac{1}{n} dF \\
&= 4\sigma^2 \langle H_{j1} f_0, H_{j2} f_0 \rangle_n.
\end{aligned}$$

Moreover, we have

$$[\int \boldsymbol{\phi}'_F(\delta_y - F)dF]_j = \frac{2}{n} \int (y - f_0)^\top H_j f_0 dF = 0,$$

and

$$\begin{aligned}
[\sqrt{n}\mathbf{R}_n]_j &= \sqrt{n}[[\boldsymbol{\phi}(\mathcal{F}_n)]_j - [\boldsymbol{\phi}(F)]_j - \frac{1}{n}\sum_i [\boldsymbol{\phi}'_F(\delta_{y_i} - F)]_j] \\
&= \frac{1}{\sqrt{n}}[f^\top H_j f - f_0^\top H_j f_0 - \frac{2}{n}(y - f_0)^\top H_j f_0] \\
&= \frac{1}{\sqrt{n}}[(f - f_0)^\top H_j (f - f_0) + 2(f - f_0)^\top H_j f_0 - \frac{2}{n}(y - f_0)^\top H_j f_0] \\
&= \frac{1}{\sqrt{n}}[\hat{\epsilon}_n^\top H_j \hat{\epsilon}_n + 2\hat{\epsilon}_n^\top H_j f_0 - \frac{2}{n}(y - f_0)^\top H_j f_0] \qquad (15) \\
&= \frac{1}{\sqrt{n}} o_p(\sqrt{n}) \qquad (16) \\
&= o_p(1) \xrightarrow{P} 0,
\end{aligned}$$

where $\hat{\epsilon}_n = f - f_0$ and the reason from Equation (15) to Equation (16) is because of Equation (14). Therefore, by Lemma 5, we have

$$\sqrt{n}(\boldsymbol{\psi}(f) - \boldsymbol{\psi}(f_0)) \xrightarrow{d} \mathcal{MVN}(\mathbf{0}, \mathbf{V}_0),$$

where $\mathbf{V}_0$ is a $d \times d$ matrix such that $[\mathbf{V}_0]_{j1,j2} = 4\sigma^2 \langle H_{j1} f_0, H_{j2} f_0 \rangle_n$. □

E Additional Theoretical Discussions

E.1 Lipschitz condition of ML models

The condition of the differentiation operator $D_j : f \to \frac{\partial}{\partial \mathbf{x}^j} f$ being bounded is guaranteed if f is differentiable and Lipschitz, so that $\frac{|f(\mathbf{x}_1)-f(\mathbf{x}_2)|}{||\mathbf{x}_1-\mathbf{x}_2||_2} \leq C$ where $||\mathbf{x}_1 - \mathbf{x}_2||_2$ is the L_2 metric in $\mathcal{X}$. Fortunately, a wide range of machine learning models (under proper regularity condition) satisfy the Lipschitz condition. Below we consider a few important examples:

Generalized Additive Models (GAM). The generalized additive models is often written as the sum of smooth functions,

$$f(\mathbf{x}) = \beta_0 + \sum_{j=1}^{d} \boldsymbol{\beta}_j h_j(\mathbf{x}^j).$$

As a result, f is Lipschitz if every individual smooth function h_j is Lipschitz. To this end, we notice that in the GAM algorithm, the h_j's are commonly estimated under a smoothness constraint in terms of its second derivatives [Wood, 2006] $\psi_{2,j} = \int_{\mathcal{X}} |\frac{\partial^2}{\partial (\mathbf{x}^j)^2} f(\mathbf{x}^j)|^2 d\mathbf{x}$, which essentially imposes an upper bound on the first-order partial derivatives $\frac{\partial}{\partial \mathbf{x}^j} f(\mathbf{x}^j)$ (assuming bounded support). As a result, the Lipschitz of GAM function is guanranteed by the virtue of its smoothing constraints.

Decision Trees. Interestingly, we can understand the Lipschitz condition of a tree-type model by investigating its model structure from a neural network lens. Specifically, for a depth-L tree model with D leaf nodes, Karthikeyan et al. [2021] shows that a it can be written in the form of a neural network layer:

$$f(\mathbf{x}) = \sum_{k=1}^{D} q_k(\mathbf{x}) \boldsymbol{\beta}_k, \quad \text{where} \quad q_k(\mathbf{x}) = \sigma_{\texttt{step}}\Big(\sum_{l=1}^{L} \sigma_{\texttt{step}}\big((\mathbf{x}^\top \mathbf{w}_{k,I(l,k)} + b_{k,I(l,k)}) S(l,k) \big) - h \Big).$$

Here $q_k(\mathbf{x})$ is a re-parametrization for the indicator function of whether $\mathbf{x}$ belongs to the k^{th} leaf node, i.e., $\prod_{l=1}^{L} \sigma_{\texttt{step}}\Big[(\mathbf{x}^\top \mathbf{w}_{k,I(l,k)} + b_{k,I(l,k)}) S(l,k) \Big]$. (See Appendix A.1 or Section 3 of Karthikeyan et al. [2021] for full detail.) Briefly, $\sigma_{\texttt{step}}(x) = I(x > 0)$ is the step function, $I(l,k)$ indicates the index for the ancestor node for the k^{th} leaf at depth l, and $S(l,k) \in \{-1, 1\}$ is a sign function for whether k^{th} leaf is the right subtree of node $I(l,k)$. As a result, $q_k(\mathbf{x})$ measures whether $\mathbf{x}$ satisfies every ancestry decision rules $I\big[S(l,k)(\mathbf{x}^\top \mathbf{w}_{k,I(l,k)} - b_{k,I(l,k)}) > 0\big]$ at every level $l \in \{1, \ldots, L\}$, where $\mathbf{w}_{k,I(l,k)}$ is a $d \times 1$ one-hot vector indicating the index of feature being selected by that node.

As a result, the tree model can be viewed as a wide 1-hidden layer neural network model with bounded activation function $\sigma_{\texttt{step}}$ and hidden weights bounded within $[-1, 1]$, which leads to a Lipschitz function. Furthermore, the function $f(\mathbf{x})$ remains Lipschitz if we replace the non-differentiable $\sigma_{\texttt{step}}$ with a differentiable activation function that is Lipschitz (e.g., Appendix F).

Random Feature Models. The random feature methods are also structured the same way as $f(\mathbf{x}) = \sigma(\mathbf{W}^\top \mathbf{x} + \mathbf{b})$, where $\mathbf{W}$ are frozen weights that are independently sampled from distribution with finite second moments (e.g., Gaussian distribution), and σ is a trignomitric function (sin and cos), or common activation functions that are used in the neural networks [Choromanski et al., 2018, Liu et al., 2021]. As a result, $f(\mathbf{x})$ is also Lipschitz with high probability. In practice, the Lipschitz condition can be guaranteed in absolute terms by truncating the individual terms in $\mathbf{W}$ to be within a range $[-C, C]$ (e.g., $C = 4.$ for $W \overset{iid}{\sim} N(0,1)$), which often leads to almost identical performance.

(Deep) Neural Networks. Both deep neural networks and random-feature models can be written as a composition of functions:

$$f(\mathbf{x}) = \boldsymbol{\beta}^\top g_L \cdot g_{L-1} \cdots \cdot g_1(\mathbf{x}), \quad \text{where} \quad g_l(\mathbf{x}) = \sigma(\mathbf{W}_l^\top \mathbf{x} + \mathbf{b}_l).$$

As a result, due to chain rule, f is Lipschitz if each of its individual layer g_l is Lipschitz [Virmaux and Scaman, 2018]. Similarly, since the layer function g_l is a composition of the linear function $\mathbf{W}_l^\top \mathbf{x} + \mathbf{b}_l$ and a non-linear activation σ, g_l is guanranteed to be Lipschitz if both the linear function is bounded with high probability, and the activation function σ is also Lipschitz. In the context of neural network learning, this is often satisfied by the common practice of imposing L_1 or L_2 regularization to neural network weights, and by using standard choices of activation functions such as ReLU, leaky ReLU, tanh, etc [Virmaux and Scaman, 2018, Liu, 2019].

E.2 Discussion on *Bernstein-von Mises* (BvM) phenomenon

Dimensionality of the Derivative Function Space. Denote $\mathcal{H}$ the space of model functions spanned by the basis functions $\{b_k(\mathbf{x})\}_{k=1}^D$, such that $f(\mathbf{x}) = \sum_{k=1}^D \alpha_k b_k(\mathbf{x})$. Then, the space of partial derivative function is $\mathcal{H}_j = \{\frac{\partial}{\partial \mathbf{x}^j} f | f \in \mathcal{H}\}$. Furthermore, for every element in $\mathcal{H}_j$, we have:

$$\frac{\partial}{\partial \mathbf{x}^j} f = \sum_{k=1}^{D} \alpha_k \cdot [\frac{\partial}{\partial \mathbf{x}^j} b_k(\mathbf{x})].$$

That is, the derivative function space $\mathcal{H}_j$ can be spanned by $\{\frac{\partial}{\partial \mathbf{x}^j} b_k(\mathbf{x})\}_{k=1}^D$, the partial derivatives of the basis functions for the original model space $\mathcal{H}$. Furthermore, since differentiation is a linear operator, the set of linearly independent functions in $\{\frac{\partial}{\partial \mathbf{x}^j} b_k(\mathbf{x})\}_{k=1}^D$ should be equivalent to that in $\{b_k(\mathbf{x})\}_{k=1}^D$. As a result, the effective dimensionality of the derivative function space $\mathcal{H}_j$ can be controlled by the effective dimensionality of the model space $\mathcal{H}$. As an aside, for a model space $\mathcal{H}_\phi$ induced by the feature representation $\phi : \mathcal{X} \to \mathbb{R}^D$, its effective dimensionality can be measured by the rank of the feature matrix $\texttt{rank}(\Phi)$ for $\Phi = [\phi(\mathbf{x}_1)^\top, \ldots, \phi(\mathbf{x}_n)^\top]^\top$. Alternatively, in the nonparametric literature, the effective dimensionality can also be measured by model-specific notions of "parameter count", such as the number of leaf partitions of a tree model, or the number of non-zero hidden weights of a deep neural network [Schmidt-Hieber, 2020].

Effective Dimensionality of Statistical ML Models. The BvM result (Theorem 2) contains a key condition (4) $H_j = o_p(\sqrt{n})$. As stated in the main text, this condition can be satisfied if the effective dimensionality of model space $\mathcal{H}_\phi$ does not grow faster than $o_p(\sqrt{n})$ with respect to the data.

Combined with the posterior convergence condition (i.e., (1)-(2) from Theorem 1), (4) provides a more precise characterization of the convergence behavior of the model $f \in \mathcal{H}_\phi$ for the BvM phenomenon to occur. Loosely, (1)-(2) states that the model f should balance its bias-variance tradeoff well enough so that the overall error rate is controlled at the rate ϵ_n. Then, (4) goes one step further and states that within this bias-variance tradeoff, the variance term must be well managed, which is guaranteed by bounding the model complexity at the rate of $o_p(\sqrt{n})$.

As a matter of fact, for a wide class of ML models, a $o_p(\sqrt{n})$ bound on model complexity is not a stringent requirement, as it only prescribes a growth rate of model complexity with respect to data size. For example, the effective data size can be $C * \sqrt{n}$ for an bounded but very large C). Interestingly, condition (4) is in fact equivalent or looser than some of the previous BvM results obtained for specific ML models. For example, the decision tree models (e.g., BART) obtains a optimal rate when its number of partitions grow at a rate of $O((n/logn)^{d/2\gamma+d})$ for learning the space of γ-Hölder continuous functions with $\gamma > d/2$ [Rockova, 2020], which leads to a more stringent $o(\sqrt{n/logn}) < o(\sqrt{n})$ bound on complexity. A similar result also holds for deep learning models, where the number of non-zero model weights is controlled at $O(n^{d/(2\gamma+d)})$ for $\gamma > \frac{d}{2}$ ([Wang and Rocková, 2020], Theorem 3.2), which also leads to a rate of $o(\sqrt{n})$.

F Incorporating Non-differentiability

F.1 Incorporating Non-differentiable Model: featurized decision trees (FDT)

Several techniques have been proposed to learn a (soft) tree-structured model using gradient-optimization methods. However, either their accuracies do not match the state-of-the-art tree learning methods Yang et al. [2018] or result in models that do not obey the tree structure Irsoy et al. [2012], Frosst and Hinton [2017], Biau et al. [2019], Tanno et al. [2019]. We propose to translate a learned tree into its exact feature representation, and leverage this representation to unlock a rigorous uncertainty-aware variable importance estimation method that was previously not available for this class of models.

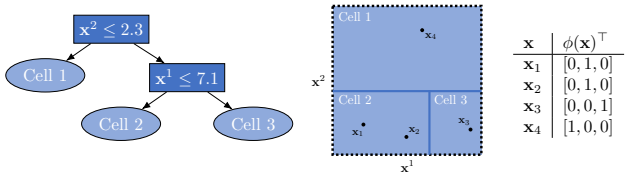
Feature-based Representation of a Decision Tree For a certain decision tree m in a learned random forest, consider the following feature map $\phi : \mathbb{R}^d \to \mathbb{R}^D$:

1. The decision tree partitions the whole feature space into D cells $\mathcal{X} = \cup_{k=1}^{D} \mathcal{X}_k$. Label the cells of the generated partition by $1, 2, \ldots, D$ in arbitrary order.
2. To encode a data point $\mathbf{x} \in \mathbb{R}^d$, look up the label y of the cell that $\mathbf{x}$ falls into and set $\phi(\mathbf{x})$ to be the (column) indicator vector of whether $\mathbf{x} \in \mathcal{X}_k$, i.e., $\phi(\mathbf{x}) = \{\mathbb{1}(\mathbf{x} \in \mathcal{X}_k)\}_{k=1}^{D}$.

The dimensionality D of ϕ equals the number of leaf nodes, and each feature mapping $\phi(\mathbf{x})$ takes the one-hot form. This feature map ϕ induces a kernel

$$k_{dt}(\mathbf{x}, \mathbf{x}') := \phi(\mathbf{x})^\top \phi(\mathbf{x}') = \begin{cases} 1 & \text{if } \mathbf{x}, \mathbf{x}' \text{ in the same partition cell} \\ 0 & \text{otherwise} \end{cases}$$

As a result, the feature mapping $\phi(\mathbf{x})$ defines a featurized decision tree.



$\mathbf{x}$	$\phi(\mathbf{x})^\top$
$\mathbf{x}_1$	$[0, 1, 0]$
$\mathbf{x}_2$	$[0, 1, 0]$
$\mathbf{x}_3$	$[0, 0, 1]$
$\mathbf{x}_4$	$[1, 0, 0]$

Figure 2: Feature expansion of a decision tree evaluated on 4 data points in $\mathbb{R}^2$. The middle panel shows the partition of $\mathbb{R}^2$ defined by the decision tree on the left. On the right is the associated feature map.

As introduced in Section 3.1, the solution for $\boldsymbol{\beta}$ is $(\Phi^\top \Phi + \sigma^2 \mathbf{I}_D)^{-1} \Phi^\top \mathbf{y}$. Note that under the decision tree kernel, $\Phi^\top \Phi = diag(n_1, \ldots, n_D)$ is a diagonal matrix of the number of training samples in each leaf cell. Therefore, the time complexity to invert the matrix $(\Phi^\top \Phi + \sigma^2 \mathbf{I}_D)$ is $O(D)$.

Differentiable Approximation The random features generated by Figure 2 can be written as

$$\begin{aligned} \phi(\mathbf{x}) &= (\mathbb{1}(\mathbf{x}^2 \leq 2.3), \mathbb{1}(\mathbf{x}^2 > 2.3, \mathbf{x}^1 \leq 7.1), \mathbb{1}(\mathbf{x}^2 > 2.3, \mathbf{x}^1 > 7.1)) \\ &= (\mathbb{1}(\mathbf{x}^2 \leq 2.3), \mathbb{1}(\mathbf{x}^2 > 2.3) \cdot \mathbb{1}(\mathbf{x}^1 \leq 7.1), \mathbb{1}(\mathbf{x}^2 > 2.3) \cdot \mathbb{1}(\mathbf{x}^1 > 7.1)). \end{aligned}$$

To calculate variable importance, the indicator function needs to be approximated by a smooth function, so that we can take the derivative with respect to each feature. In this work, we consider approximating the indicator function using the sigmoid function Irsoy et al. [2012]:

$$\mathbb{1}(x > a) \approx i_c(x > a) = \frac{1}{1 + \exp(-c \cdot (x - a))},$$

and analogously, $\mathbb{1}(x \leq a) = 1 - i_c(x > a)$. Here c is a hyperparameter that controls the smoothness of the approximation. A larger c leads to a better approximation to the random forest algorithm, but may result in a non-smooth prediction function which may be undesirable for approximating an continuous regression function f_0.

F.2 Incorporating Discrete Features

Compared to the empirical derivative norm, a more principled way to measure the variable importance of a discrete feature is *contrast*, which is the square of the difference in predictions when fixing the feature to a certain value versus fixing it to the other value, while keeping the other features the same. Specifically, we can consider defining a discrete version of the derivative:

$$D_j f = f(\mathbf{x}^j = 1, \mathbf{x}^{-j}) - f(\mathbf{x}^j = 0, \mathbf{x}^{-j}), \tag{17}$$

where $\mathbf{x}^{-j}$ denotes all features with $\mathbf{x}^j$ removed.

Then, in the case where the feature takes two values, we can set one of them as the reference group with value 0 and the other group with value 1,

$$\begin{aligned}\Psi_j(f) = \|D_j f\|_2^2 &= \int_{\mathbf{x}\in\mathcal{X}} |D_j f|^2 dP(\mathbf{x}) \\ &= \int_{\mathbf{x}\in\mathcal{X}} |f(\mathbf{x}^j = 1, \mathbf{x}^{-j}) - f(\mathbf{x}^j = 0, \mathbf{x}^{-j})|^2 dP(\mathbf{x}),\end{aligned}$$

Since $P(\mathbf{x})$ is not known from the training observations, $\Psi_j(f)$ can be approximated by its empirical counterpart:

$$\begin{aligned}\psi_j(f) = \|D_j f\|_n^2 &= \frac{1}{n}\sum_{i=1}^{n} |D_j f|^2. \\ &= \frac{1}{n}\sum_{i=1}^{n} |f(\mathbf{x}_i^j = 1, \mathbf{x}_i^{-j}) - f(\mathbf{x}_i^j = 0, \mathbf{x}_i^{-j})|^2.\end{aligned}$$

In the case where the feature takes multiple groups, we can calculate the pairwise contrasts and take the L_2 norm. Empirically, using contrast for discrete feature improves the performance of variable importance estimation. As contrast is a linear function of the original prediction function f, the posterior convergence of ψ_j with respect to this operator is guaranteed by the convergence of the prediction function f. Similarly, the BvM phenomenon is guaranteed when $D_j f$ is bounded and $H_j = D_j^\top D_j$ has rank $o_p(\sqrt{n})$ (i.e., the similar set of conditions in Theorem 2 but with the original D_j replaced by its discrete counterpart Equation (17)).

G Further Experiment Detail

G.1 Methods

We consider three main classes of models (Table 1).

I **Random Forests (RF)**

- **FDT**: Given a trained forest, we quantify variable importance using ψ_j by translating it to an ensemble of **FDT** (Appendix F.1). We use a variant of random forest here, extra trees Geurts et al. [2006] since it performs better. We use 50 trees to build the forest and maximum number of leaf nodes for each tree is $\sqrt{n}\,log(n)$. Throughout our experiment, we fix $c = 1$ for continuous features calculated using integrated partial derivatives and fix $c = 0.1$ for discrete features calculated using contrasts. We use `scikit-learn` package in Python to train the random forest.
- **RF-impurity** [Breiman et al., 1984]: It measures variable importance with their impurity based on the average reduction of the loss function were the variable to be removed. We also use extra trees here. We use 50 trees to build the forest and maximum number of leaf nodes for each tree is $\sqrt{n}\,log(n)$. We use `scikit-learn` package in Python to train the random forest.
- **RF-knockoff** [Candes et al., 2017]: It uses random forest statistics to assess variable importance in our case. We use `knockoff` package in R to calculate the statistic.
- Bayesian additive regression trees (**BART**) Chipman et al. [2010]: It produces a measure of variable importance by tracking variable inclusion proportions, enabling variable selection with a user-defined threshold. We use `bartMachine` package in R to train the model.

II **(Approximate) Kernel Methods** & **Neural Networks**

- Random Fourier Feature model (**RFF**): We apply ψ_j to a random-feature model that approximates a Gaussian process with an RBF kernel Rahimi and Recht [2007], and set the number of features to $\sqrt{n}\,log(n)$ to ensure proper approximation of the exact RBF-GP Rudi and Rosasco [2018]. We choose the lengthscale parameter of RBF-GP from a list of lengthscale candidates $\{5, 10, 16, 23\}$ based on the prediction performance on testing data.
- Bayesian kernel machine regression (**BKMR**) Bobb et al. [2015]: It is based on a GP with exact RBF kernel and spike-and-slab prior, using posterior inclusion probabilities to perform variable selection. We use `bkmr` package in R to train the model and the number of iterations of the MCMC sampler is set to be 4000.
- Bayesian Approximate Kernel Regression (**BAKR**) [Crawford et al., 2018]: It is based on random-feature model with a projection-based feature importance measure and an adaptive shrinkage prior, using squared estimates of the parameter coefficients to perform variable selection. We use `BAKR` repository from the author's GitHub to train the model and the number of iterations of the MCMC sampler is set to be 2000.
- Sparse Neural Networks (**NN**): We apply ψ_j to a 1-layer neural network with 512 hidden units and L_1 regularization (i.e., LASSO net) on the hidden layer, implemented in the `tensorflow.keras` framework. We train the model with Adam optimizer and early stopping with respect to validation RMSE. We sweep the regularization strength of L_1 penalty in exponential grids with exponents $\{-3, -2, -1, 0, 1., 2, 3\}$. We also experimented with deeper layers (up to 3 layers) and observed similar performance.

III **Linear Models**

- **GAM**: We apply ψ_j to a featurized GP representation of the **GAM**, with the prior center $\boldsymbol{\mu}$ set at the frequentist estimate of the original GAM model obtained from a sophisticated REML procedure [Wood, 2006]. We use `mgcv` package in R to train the model.
- Bayesian Ridge Regression (**BRR**) Hoerl and Kennard [1970]: It applies a fixed prior for each feature, using squared estimates of the parameter coefficients to perform variable selection. We use `BGLR` package in R to train the model and the number of iterations of the MCMC sampler is set to be 2000.

- Bayesian Lasso (**BL**) Park and Casella [2008]: It developed a Bayesian way to access the Lasso estimate which allows tractable full conditional distributions, using squared estimates of the parameter coefficients to perform variable selection. We use `BGLR` package in R to train the model and the number of iterations of the MCMC sampler is set to be 2000.

The results in this paper were obtained using R 4.1.0 or Python 3.7. All experiments were run on a Linux-based high performance computing cluster using SLURM-managed CPU resources.

G.2 Data

Outcome-generating function As discussed earlier, we generate data under the homoscedastic Gaussian noise model $y \sim \mathcal{N}(f_0(\mathbf{x}), 0.01)$ for different sparse functions f_0 and features $\mathbf{x}$. Given $n \in \{100, 200, 500, 1000\}$ observations in $d \in \{25, 50, 100, 200\}$ dimensions, the goal is to model f_0 while identifying the $d^* = 5$ features on which f_0 depends. To this end we report mean squared error (MSE) to quantify prediction performance and AUROC scores to quantify variable importance estimation performance.

We consider four settings of the data-generation function f_0:

1) `linear`: a simple linear function $f_0(\mathbf{x}) = \mathbf{x}^1 - \mathbf{x}^2 + \mathbf{x}^3 + 0.5\mathbf{x}^4 + 2\mathbf{x}^5$;
2) `rbf`: a Gaussian RBF kernel with length-scale 1. This kernel represents the space of functions that are smooth (i.e., infinitely differentiable) and have reasonable complexity (i.e., does not have fast-varying fluctuations that are difficult to model);
3) `matern32`: a matérn $\frac{3}{2}$ kernel with length-scale 1. Compared to RBF, it has the same degree of complexity but is less smooth, in the sense that it represents the space of once-differentiable functions, but is not necessarily infinitely differentiable;
4) `complex`: a complicated and non-smooth multivariate function that is outside the RKHS $\mathcal{H}$: $f_0(\mathbf{x}) = \frac{\sin(\max(\mathbf{x}^1, \mathbf{x}^2)) + \arctan(\mathbf{x}^2)}{1 + \mathbf{x}^1 + \mathbf{x}^5} + \sin(0.5\mathbf{x}^3)(1 + \exp(\mathbf{x}^4 - 0.5\mathbf{x}^3)) + \mathbf{x}^{3^2} + 2\sin(\mathbf{x}^4) + 4\mathbf{x}^5$, which is non-continuous in terms of $\mathbf{x}^1, \mathbf{x}^2$ but infinitely differentiable in terms of $\mathbf{x}^3, \mathbf{x}^4, \mathbf{x}^5$.

Synthetic Benchmarks We create synthetic benchmark datasets of varying number of observations n and number of features d. The **synthetic-continuous** dataset uses only continuous features, and the **synthetic-mixture** dataset uses a mixture of continuous and discrete features. The synthetic features are drawn either from $Bern(0.5)$ (if discrete) or $Unif(-2, 2)$ (if continuous). Additionally, each feature is either causal (i.e., used by f_0) or non-causal. For each simulation setting, there are always $d^* = 5$ causal features. Specifically, in the **synthetic-continuous** dataset, all features are continuous, while in the **synthetic-mixture** dataset, there are 2 discrete and 3 continuous causal features, while there are 2 discrete non-causal features (all the rest of non-causal features are continous).

For each sample size - data dimension scenario, we use the same set of generated features across the repeated simulation runs.

Socio-economic and Healthcare Data

- **adult**: 1994 U.S. census data of 48842 adults with 8 categorical and 6 continuous features Kohavi. The data is publicly available[6] and does not contain personally identifiable information or offensive content. We concatenated the training data (`adult.data`) and testing data (`adult.test`), and remove all observations with missing features. Additionally, we removed the redundant feature "`education`", and performed suitable re-categorization for discrete features: For "`race`", we encoded "White" as 0 and the rest as 1; for "`sex`", we encoded "Female" as 1 and "Male" as 0; for "`relationship`", we encoded "Husband" as 0, "Not-in-family" as 1 and the rest as 2; for "`workclass`", we encoded "Private" as 0, "Self-emp-not-inc" as 1 and the rest as 2; for "marital_status", we encoded "Married-civ-spouse" as 0, "Never-married" as 1 and the rest as 2; for "`occupation`", we encoded "Prof-specialty" as 0, "Craft-repair" as 1 and the rest as 2;

[6] `https://archive.ics.uci.edu/ml/machine-learning-databases/adult/`

for "native_country", we encoded "United-States" as 0, "Mexico" as 1 and the rest as 2. The final features in the dataset are: ("race", "sex", "education_num", "hours_per_week", "age", "relationship", "workclass", "fnlwgt", "capital_gain", "capital_loss", "marital_status", "occupation", "native_country"). If the data dimension is higher than 13, additional features will be generated from $Unif(-2, 2)$.

- **heart**: a coronary artery disease dataset of 303 patients from Cleveland clinic database with 7 categorical and 6 continuous features Detrano et al. [1989]. The data is publicly available[7] and does not contain personally identifiable information or offensive content. All observations with missing features are removed before analysis.

  The list of features used in the final datasets are ("sex", "exang", "thal", "oldpeak", "age", "ca", "cp", "chol", "trestbps", "thalach", "fbs", "restecg", "slope"). If the data dimension is higher than 13, additional features will be generated from $Unif(-2, 2)$.

- **mi**: disease records of myocardial infarction (MI) of 1700 patients from Krasnoyarsk interdistrict clinical hospital during 1992-1995, with 113 categorical and 11 continuous features Golovenkin et al. [2020]. The data is publicly available[8] and does not contain personally identifiable information or offensive content. We imputed missing values using the `IterativeImputer` method from `scikit-learn` package and with a `BayesianRidge` regressor. Specifically, it imputes each feature with missing values as a function of other features in a round-robin fashion: At each step, a feature column is designated as output y and the other feature columns are treated as inputs X. A regressor is fit on (X, y) for known y. Then, the regressor is used to predict the missing values of y. This is done for each feature in an iterative fashion, and then is repeated for 10 imputation rounds. The results of the final imputation round are returned.

  The listed of features used in the analysis are as below: ("sex", "ritm_ecg_p_01", "age", "s_ad_orit", "d_ad_orit", "ant_im", "ibs_post", "k_blood", "na_blood", "l_blood", "inf_anam", "stenok_an", "fk_stenok", "ibs_nasl", "gb", "sim_gipert", "dlit_ag", "zsn_a", "nr11", "nr01", "nr02", "nr03", "nr04", "nr07", "nr08", "np01", "np04", "np05", "np07", "np08", "np09", "np10", "endocr_01", "endocr_02", "endocr_03", "zab_leg_01", "zab_leg_02", "zab_leg_03", "zab_leg_04", "zab_leg_06", "s_ad_kbrig", "d_ad_kbrig", "o_l_post", "k_sh_post", "mp_tp_post", "svt_post", "gt_post", "fib_g_post", "lat_im", "inf_im", "post_im", "im_pg_p", "ritm_ecg_p_02", "ritm_ecg_p_04", "ritm_ecg_p_06", "ritm_ecg_p_07", "ritm_ecg_p_08", "n_r_ecg_p_01", "n_r_ecg_p_02", "n_r_ecg_p_03", "n_r_ecg_p_04", "n_r_ecg_p_05", "n_r_ecg_p_06", "n_r_ecg_p_08", "n_r_ecg_p_09", "n_r_ecg_p_10", "n_p_ecg_p_01", "n_p_ecg_p_03", "n_p_ecg_p_04", "n_p_ecg_p_05", "n_p_ecg_p_06", "n_p_ecg_p_07", "n_p_ecg_p_08", "n_p_ecg_p_09", "n_p_ecg_p_10", "n_p_ecg_p_11", "n_p_ecg_p_12", "fibr_ter_01", "fibr_ter_02", "fibr_ter_03", "fibr_ter_05", "fibr_ter_06", "fibr_ter_07", "fibr_ter_08", "gipo_k", "giper_na", "alt_blood", "ast_blood", "kfk_blood", "roe", "time_b_s", "r_ab_1_n", "r_ab_2_n", "r_ab_3_n", "na_kb", "not_na_kb", "lid_kb", "nitr_s", "na_r_1_n", "na_r_2_n", "na_r_3_n", "not_na_1_n", "not_na_2_n", "not_na_3_n", "lid_s_n", "b_block_s_n", "ant_ca_s_n", "gepar_s_n", "asp_s_n", "tikl_s_n", "trent_s_n").

We standardize (by subtracting from mean and dividing by standard deviation) all features except for 2 discrete causal features and 2 discrete non-causal features.

[7] https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data

[8] https://archive.ics.uci.edu/ml/machine-learning-databases/00579/

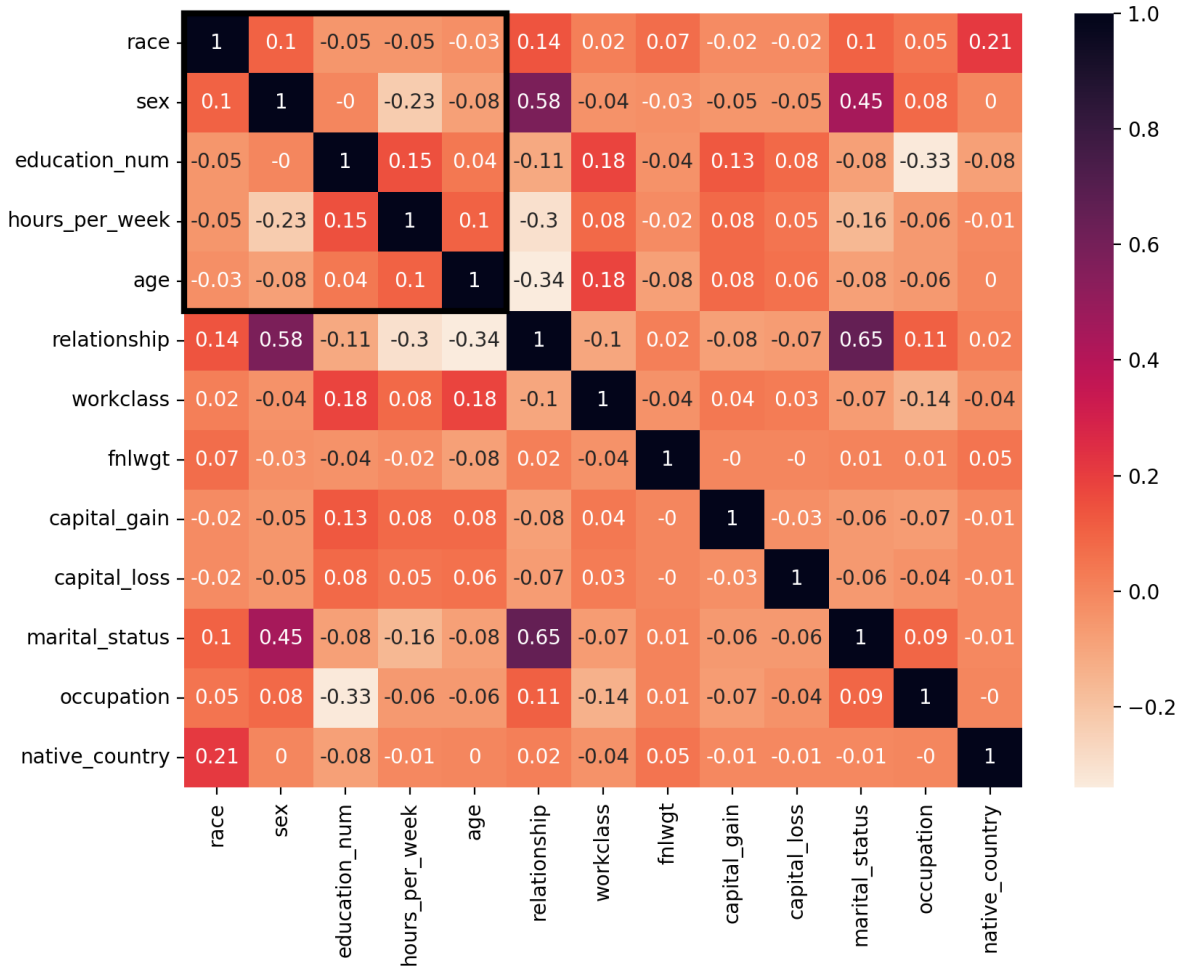


Figure 3: Correlation matrix for **adult** dataset, where the upper left black box indicates the five causal features.

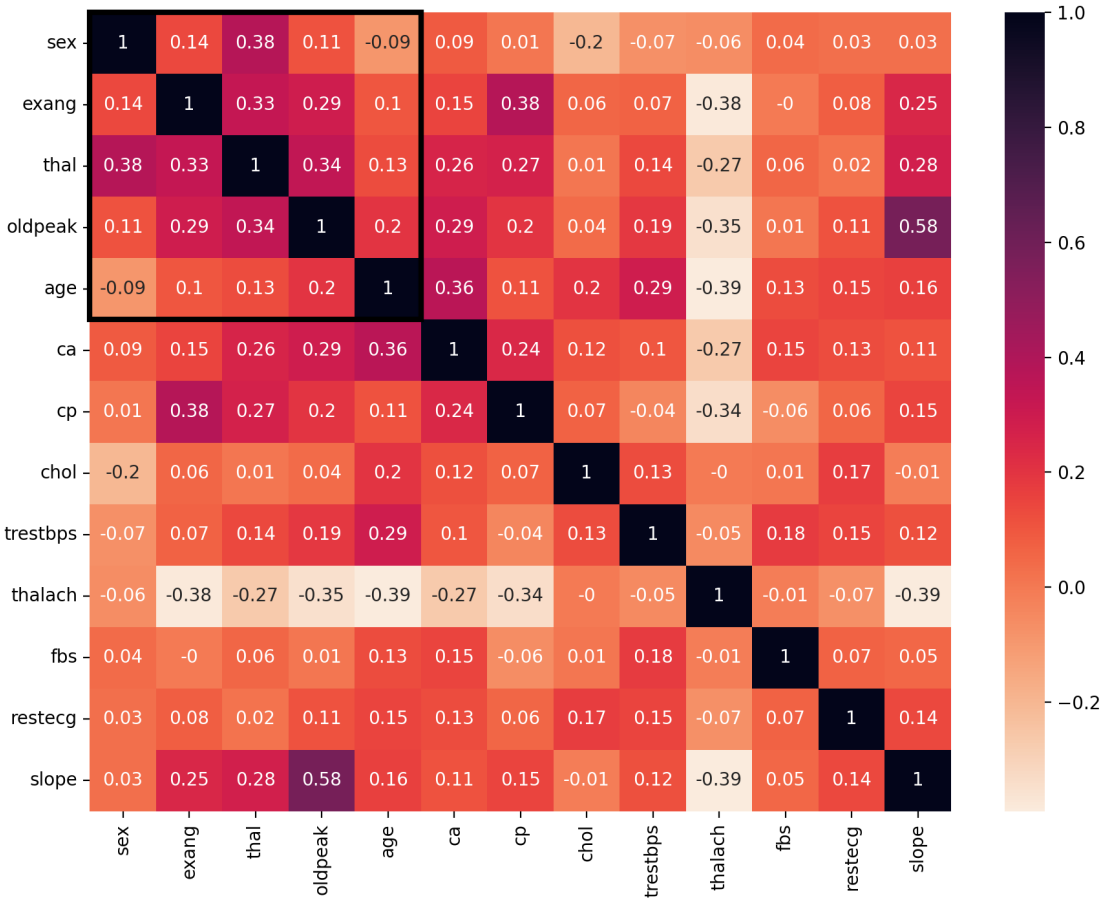


Figure 4: Correlation matrix for **heart** dataset, where the upper left black box indicates the five causal features.

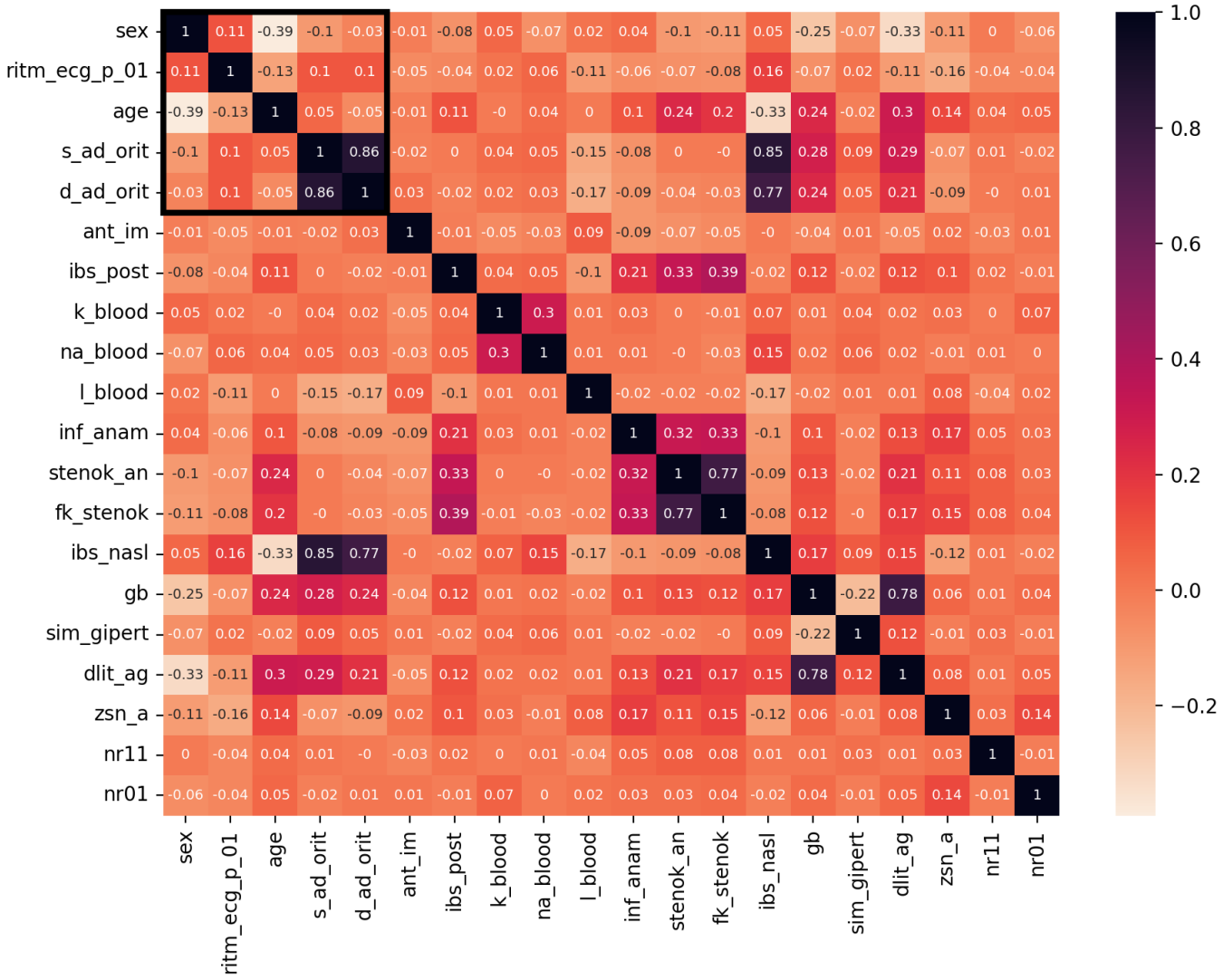


Figure 5: Correlation matrix for the first 20 features in **mi** dataset, where the upper left black box indicates the five causal features.

G.3 Error Bars

Tables 2-11 shows the AUROC scores or Testing MSE's for the result presented in the main text. For the Testing MSE tables, a method will not be shown if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if the method does not produce valid result due to small sample size (**GAM**).

Table 2: AUROC scores and their standard deviations for **synthetic-mixture** dataset.

n	RF-FDT (Ours)	NN (Ours)	GAM (Ours)	RF-Impurity	RFF (Ours)	BRR	RF-KnockOff	BKMR	BL	BART	BAKR
100	0.8 (0.09)	0.68 (0.03)	NaN (NA)	0.72 (0.13)	0.59 (0.11)	0.66 (0.12)	0.57 (0.06)	0.57 (0.09)	0.68 (0.09)	0.71 (0.16)	0.56 (0.09)
200	0.93 (0.1)	0.72 (0.03)	0.72 (0.15)	0.86 (0.2)	0.65 (0.18)	0.69 (0.16)	0.59 (0.06)	0.57 (0.08)	0.69 (0.17)	0.78 (0.14)	0.68 (0.12)
500	0.97 (0.05)	0.75 (0.03)	0.88 (0.07)	1 (0)	0.68 (0.2)	0.83 (0.09)	0.67 (0.1)	0.64 (0.11)	0.83 (0.09)	0.95 (0.08)	0.75 (0.14)
1000	0.99 (0.03)	0.79 (0.03)	0.89 (0.1)	1 (0)	0.69 (0.25)	0.86 (0.1)	0.69 (0.1)	0.68 (0.16)	0.86 (0.1)	0.99 (0.02)	0.77 (0.1)

Table 3: AUROC scores and their standard deviations for **synthetic-continuous** dataset.

n	RF-FDT (Ours)	NN (Ours)	GAM (Ours)	RF-Impurity	RFF (Ours)	BRR	RF-KnockOff	BKMR	BL	BART	BAKR
100	0.61 (0.13)	0.68 (0.05)	NaN (NA)	0.68 (0.13)	0.56 (0.12)	0.69 (0.17)	0.61 (0.12)	0.62 (0.1)	0.69 (0.13)	0.69 (0.14)	0.61 (0.1)
200	0.83 (0.13)	0.73 (0.04)	0.74 (0.12)	0.85 (0.13)	0.66 (0.11)	0.75 (0.15)	0.68 (0.13)	0.57 (0.08)	0.75 (0.13)	0.79 (0.12)	0.61 (0.14)
500	0.98 (0.03)	0.76 (0.03)	0.8 (0.14)	0.99 (0.02)	0.71 (0.16)	0.8 (0.12)	0.87 (0.1)	0.59 (0.1)	0.8 (0.12)	0.96 (0.03)	0.76 (0.12)
1000	1 (0)	0.79 (0.03)	0.86 (0.11)	1 (0)	0.63 (0.24)	0.87 (0.14)	0.97 (0.08)	0.64 (0.15)	0.86 (0.14)	1 (0)	0.79 (0.14)

Table 4: AUROC scores and their standard deviations for **adult** dataset.

n	RF-FDT (Ours)	NN (Ours)	GAM (Ours)	RF-Impurity	RFF (Ours)	BRR	RF-KnockOff	BKMR	BL	BART	BAKR
100	0.76 (0.09)	0.68 (0.04)	NaN (NA)	0.61 (0.15)	0.62 (0.13)	0.57 (0.09)	0.58 (0.13)	0.54 (0.1)	0.61 (0.09)	0.66 (0.11)	0.62 (0.13)
200	0.8 (0.09)	0.7 (0.04)	0.7 (0.14)	0.64 (0.11)	0.6 (0.14)	0.61 (0.11)	0.57 (0.1)	0.58 (0.12)	0.63 (0.09)	0.7 (0.12)	0.72 (0.12)
500	0.84 (0.07)	0.75 (0.02)	0.79 (0.13)	0.64 (0.09)	0.57 (0.18)	0.62 (0.09)	0.59 (0.12)	0.59 (0.08)	0.6 (0.1)	0.71 (0.08)	0.64 (0.11)
1000	0.81 (0.1)	0.77 (0.02)	0.86 (0.1)	0.61 (0.08)	0.64 (0.18)	0.7 (0.12)	0.57 (0.09)	0.55 (0.08)	0.69 (0.1)	0.7 (0.11)	0.72 (0.14)

Table 5: AUROC scores and their standard deviations for **heart** dataset.

n	RF-FDT (Ours)	NN (Ours)	GAM (Ours)	RF-Impurity	RFF (Ours)	BRR	RF-KnockOff	BKMR	BL	BART	BAKR
50	0.71 (0.06)	0.66 (0.04)	NaN (NA)	0.49 (0.15)	0.56 (0.14)	0.58 (0.08)	0.59 (0.06)	0.59 (0.09)	0.58 (0.07)	0.63 (0.09)	0.6 (0.09)
100	0.72 (0.06)	0.66 (0.04)	NaN (NA)	0.44 (0.11)	0.58 (0.12)	0.58 (0.09)	0.58 (0.07)	0.57 (0.08)	0.57 (0.08)	0.59 (0.11)	0.59 (0.12)
150	0.75 (0.08)	0.67 (0.05)	0.62 (0.12)	0.41 (0.12)	0.59 (0.13)	0.59 (0.06)	0.61 (0.11)	0.58 (0.07)	0.56 (0.08)	0.6 (0.07)	0.64 (0.12)
257	0.74 (0.09)	0.72 (0.03)	0.64 (0.12)	0.45 (0.12)	0.52 (0.17)	0.57 (0.07)	0.66 (0.13)	0.59 (0.09)	0.56 (0.06)	0.58 (0.09)	0.64 (0.11)

Table 6: AUROC scores and their standard deviations for **mi** dataset.

n	RF-FDT (Ours)	NN (Ours)	GAM (Ours)	RF-Impurity	RFF (Ours)	BRR	RF-KnockOff	BKMR	BL	BART	BAKR
100	0.86 (0.05)	0.64 (0.04)	NaN (NA)	0.77 (0.08)	0.59 (0.13)	0.65 (0.1)	0.67 (0.12)	0.57 (0.09)	0.63 (0.11)	0.62 (0.14)	0.86 (0.05)
200	0.85 (0.04)	0.73 (0.05)	0.87 (0.05)	0.79 (0.07)	0.62 (0.08)	0.62 (0.09)	0.63 (0.12)	0.58 (0.1)	0.65 (0.1)	0.61 (0.11)	0.84 (0.07)
500	0.85 (0.05)	0.75 (0.03)	0.87 (0.07)	0.77 (0.06)	0.43 (0.15)	0.64 (0.09)	0.62 (0.08)	0.59 (0.1)	0.61 (0.09)	0.6 (0.08)	0.81 (0.1)
1000	0.83 (0.04)	0.77 (0.02)	0.89 (0.06)	0.73 (0.07)	0.56 (0.17)	0.6 (0.13)	0.62 (0.11)	0.54 (0.08)	0.67 (0.1)	0.63 (0.11)	0.88 (0.09)

Table 7: Testing MSE's and their standard deviations for **synthetic-mixture** dataset. A method will not be shown if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if the method does not produce valid result due to small sample size (**GAM**)

n	RF-FDT (Ours)	NN (Ours)	GAM (Ours)	RFF (Ours)	BRR	BKMR	BL	BART	BAKR
100	1.02 (0.25)	1.06 (0.22)	NaN (NA)	1.76 (0.28)	1.01 (0.23)	1.52 (0.14)	0.95 (0.21)	0.98 (0.23)	2.87 (1.06)
200	0.87 (0.16)	1.02 (0.12)	1.59 (0.29)	1.53 (0.22)	1.01 (0.13)	1.56 (0.09)	0.95 (0.12)	0.98 (0.16)	1.04 (0.14)
500	0.76 (0.13)	1.04 (0.12)	1.04 (0.2)	1.42 (0.15)	0.94 (0.13)	1.57 (0.08)	0.93 (0.13)	0.83 (0.14)	0.97 (0.11)
1000	0.66 (0.12)	1.03 (0.13)	0.96 (0.18)	1.32 (0.15)	0.94 (0.16)	1.62 (0.07)	0.93 (0.17)	0.75 (0.12)	1.01 (0.13)

Table 8: Testing MSE's and their standard deviations for **synthetic-continuous** dataset. A method will not be shown if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if the method does not produce valid result due to small sample size (**GAM**)

n	RF-FDT (Ours)	NN (Ours)	GAM (Ours)	RFF (Ours)	BRR	BKMR	BL	BART	BAKR
100	1.01 (0.2)	1.06 (0.16)	NaN (NA)	1.73 (0.26)	1.05 (0.15)	1.55 (0.11)	1.01 (0.16)	1.02 (0.15)	2.39 (0.59)
200	0.87 (0.15)	0.97 (0.19)	1.41 (0.32)	1.48 (0.23)	0.92 (0.18)	1.5 (0.15)	0.9 (0.16)	0.93 (0.19)	0.95 (0.19)
500	0.85 (0.19)	1.08 (0.12)	1.02 (0.25)	1.43 (0.13)	0.95 (0.19)	1.58 (0.09)	0.91 (0.19)	0.93 (0.2)	1 (0.15)
1000	0.72 (0.15)	1.05 (0.18)	0.94 (0.2)	1.4 (0.19)	0.91 (0.18)	1.6 (0.11)	0.9 (0.18)	0.8 (0.19)	0.98 (0.18)

Table 9: Testing MSE's and their standard deviations for **adult** dataset. A method will not be shown if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if the method does not produce valid result due to small sample size (**GAM**)

n	RF-FDT (Ours)	NN (Ours)	GAM (Ours)	RFF (Ours)	BRR	BKMR	BL	BART	BAKR
100	0.95 (0.4)	0.96 (0.13)	NaN (NA)	1.69 (0.44)	0.3 (0.11)	0.92 (0.15)	0.22 (0.07)	0.4 (0.11)	2.71 (0.92)
200	0.91 (0.24)	1.02 (0.2)	1.23 (0.32)	1.63 (0.31)	0.28 (0.07)	1.03 (0.07)	0.22 (0.05)	0.4 (0.12)	0.28 (0.07)
500	0.96 (0.18)	1 (0.11)	0.45 (0.08)	1.31 (0.14)	0.25 (0.06)	1 (0.08)	0.22 (0.07)	0.28 (0.08)	0.22 (0.07)
1000	0.96 (0.18)	1.02 (0.14)	0.32 (0.06)	1.28 (0.17)	0.24 (0.05)	1.08 (0.04)	0.21 (0.04)	0.28 (0.12)	0.21 (0.04)

Table 10: Testing MSE's and their standard deviations for **heart** dataset. A method will not be shown if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if the method does not produce valid result due to small sample size (**GAM**)

n	RF-FDT (Ours)	NN (Ours)	GAM (Ours)	RFF (Ours)	BRR	BKMR	BL	BART	BAKR
50	0.92 (0.21)	1.03 (0.16)	NaN (NA)	1.93 (0.5)	0.35 (0.12)	1.03 (0.13)	0.26 (0.1)	0.39 (0.12)	0.44 (0.18)
100	0.95 (0.27)	1.09 (0.23)	NaN (NA)	1.88 (0.29)	0.32 (0.08)	1.02 (0.1)	0.24 (0.06)	0.43 (0.13)	2.18 (0.68)
150	0.98 (0.22)	1.06 (0.19)	1.76 (0.37)	1.65 (0.32)	0.27 (0.08)	0.99 (0.12)	0.21 (0.08)	0.4 (0.14)	0.31 (0.1)
257	0.91 (0.26)	1.04 (0.15)	0.79 (0.2)	1.51 (0.2)	0.27 (0.06)	1 (0.1)	0.23 (0.06)	0.33 (0.12)	0.25 (0.06)

Table 11: Testing MSE's and their standard deviations for **mi** dataset. A method will not be shown if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if the method does not produce valid result due to small sample size (**GAM**)

n	RF-FDT (Ours)	NN (Ours)	GAM (Ours)	RFF (Ours)	BRR	BKMR	BL	BART	BAKR
100	1.55 (0.94)	1.63 (1.1)	NaN (NA)	2.02 (0.44)	0.36 (0.15)	0.78 (0.22)	0.31 (0.1)	0.32 (0.1)	0.66 (0.32)
200	1.76 (2.86)	1.29 (0.63)	0.61 (0.27)	1.82 (0.48)	0.32 (0.11)	0.85 (0.22)	0.3 (0.11)	0.28 (0.12)	0.41 (0.15)
500	1.13 (0.35)	1.17 (0.44)	0.42 (0.22)	1.57 (0.27)	0.27 (0.07)	0.53 (0.25)	0.26 (0.06)	0.26 (0.09)	0.26 (0.09)
1000	1.2 (1.05)	1 (0.16)	0.36 (0.08)	1.43 (0.3)	0.28 (0.07)	0.71 (0.34)	0.27 (0.07)	0.24 (0.07)	0.25 (0.07)

G.4 Code

For the code, data, instructions, the total amount of compute and the type of resources used to reproduce the experimental results, please visit `https://github.com/wdeng5120/featurized-decision-tree`.

H Experiment Results and Additional Figures

Figures 6-10 and 11-15 show the AUROC scores and MSE results, respectively, across all of the datasets. Here we also summarize additional observations that are not included in the main text. The figure captions contain further descriptions of the results.

Synthetic Benchmarks. In the synthetic datasets, where all features are independent, FDT, RF, BART, GAM, BRR and BL perform better and more stable than they do in the real datasets where there's feature correlation. The better performance of FDT compared to **RF-impurity** and **RF-knockoff** illustrates the advantage of the proposed integrated partial derivative metric for variable importance estimation. For the **synthetic-continuous** and **synthetic-mixture** cases, FDT has higher AUROC scores across most scenarios, especially when data are generated having high complexity with quickly-varying local fluctuations (rbf, matern32). Moreover, all 11 methods perform only moderately well in complex data settings. The two tree-based methods, RF and BART also have high AUROC scores across scenarios, since the tree-based methods naturally rank by how well the features improve the purity of the node. Note that under low dimension case ($d = 25$), BKMR is comparable to FDT when $f_0 \in \mathcal{H}$ (linear, rbf, matern32). However, when it comes to medium- or relatively high-dimension settings ($d = 50, 100$), BKMR produces low AUROC scores due to suffering from the issue of curse of dimensionality van der Vaart and Zanten [2011]. RFF, also a kernel-based method, has similar trend as BKMR. Finally, BAKR performs consistently poorly and has lowest AUROC scores in relatively low-dimension setting ($d = 25, 50$). Linear models (GAM, BRR and BL) achieve comparable or superior performance under the linear data setting. However, for more complicated data generation functions, BRR and BL consistently perform poorly with low AUROC scores.

Socio-economic and Healthcare Datasets In the **adult**, **heart** and **mi** cases, where the features are correlated, the performances of all 11 methods are worse than in the **synthetic-mixture** and **synthetic-continuous** cases (where the features are independent). Their performance tends to saturate earlier and are less stable with respect to the sample size. In relatively low-dimension settings ($d = 25, 50$), the standard methods such as BART has higher AUROC scores than FDT. However, when the dimension is higher ($d = 100, 200$), FDT consistently performs better.

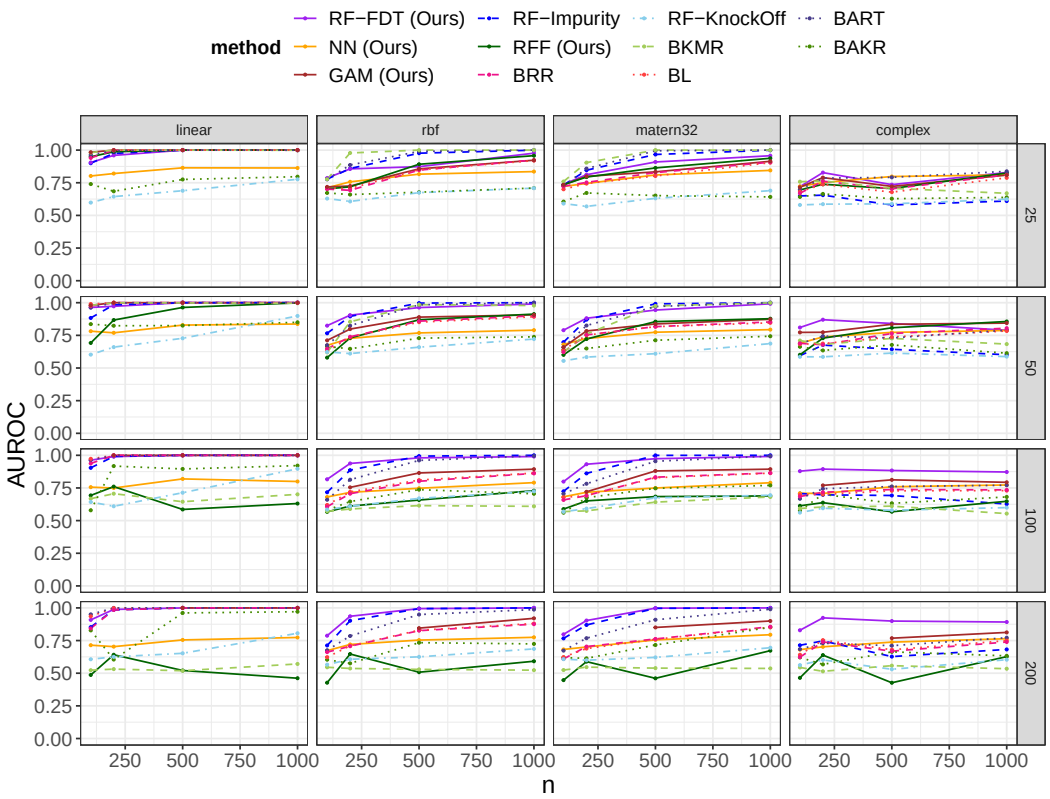


Figure 6: AUROC scores for **synthetic-mixture** data. FDT generally outperforms other methods in most of the data settings in relatively higher dimension ($d = 50, 100, 200$). Knockoff with random forest statistics produce lower AUROC scores than in **synthetic-continuous**, even in linear data settings. Additive models BRR, BL and GAM have mediocre scores under the nonlinear settings. Some model (e.g., GAM) reports missing result in $n > p$ setting due to the restrictions of their implementations.

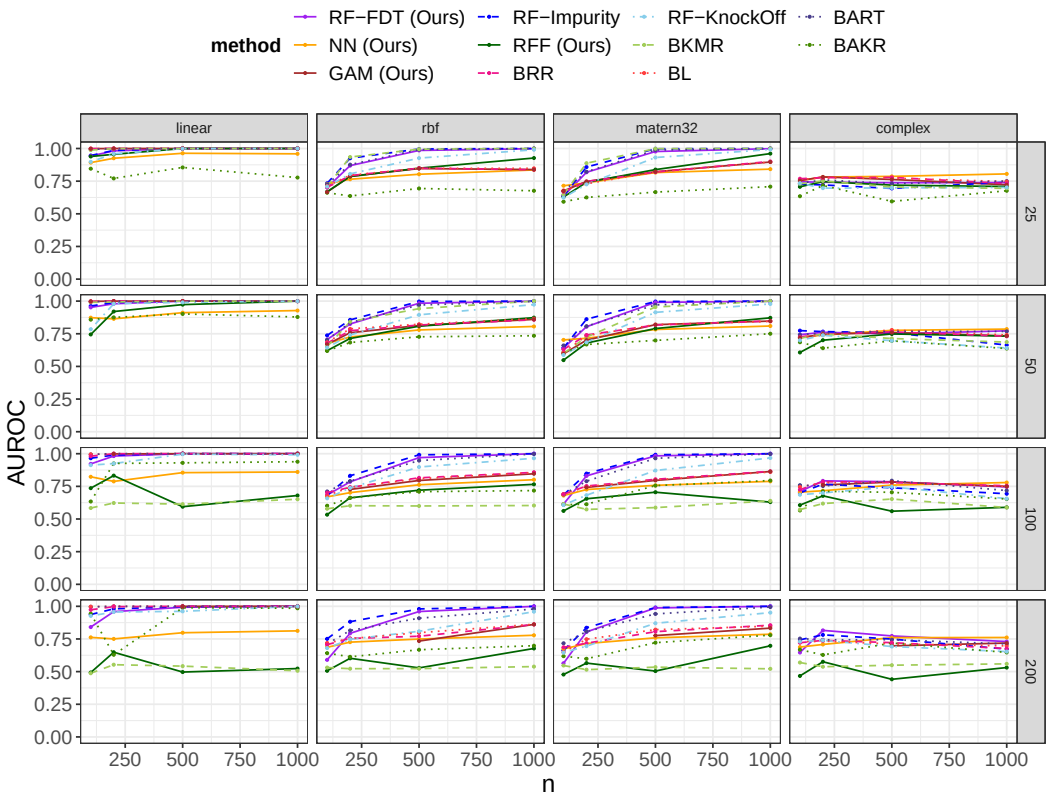


Figure 7: AUROC scores for **synthetic-continuous** data. FDT generally outperforms other methods in most of the data settings, with BKMR as the comparable one when $d = 25$. However, BKMR performs poorly in higher dimension. Tree-based methods RF, BART and Knockoff with random forest statistics have high AUROC scores. Additive models BRR, BL and GAM have mediocre scores under the nonlinear settings. Some model (e.g., GAM) reports missing result in $n > p$ setting due to the restrictions in their implementations.

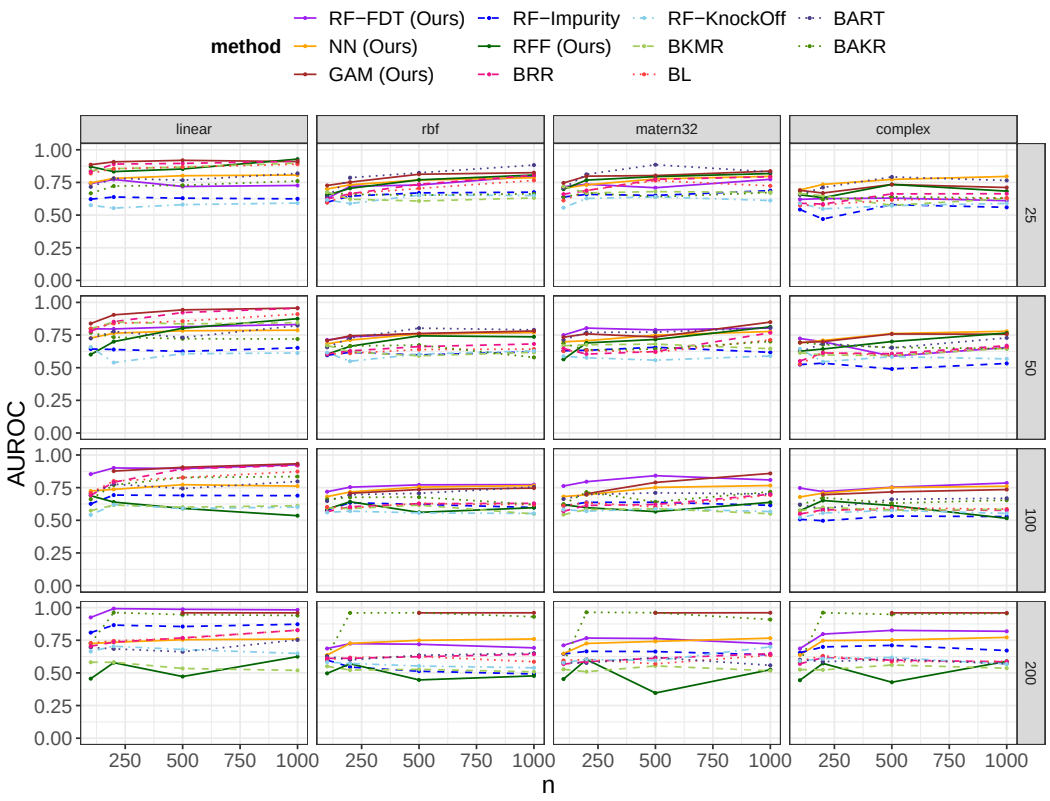


Figure 8: AUROC scores for **adult** data. In relatively low-dimension settings ($d = 25, 50$), the standard methods such as BART has higher AUROC scores than FDT. However, when the dimension is higher ($d = 100, 200$), FDT performs better consistently. Some model (e.g., GAM) reports missing result in $n > p$ setting due to the restrictions in their implementations.

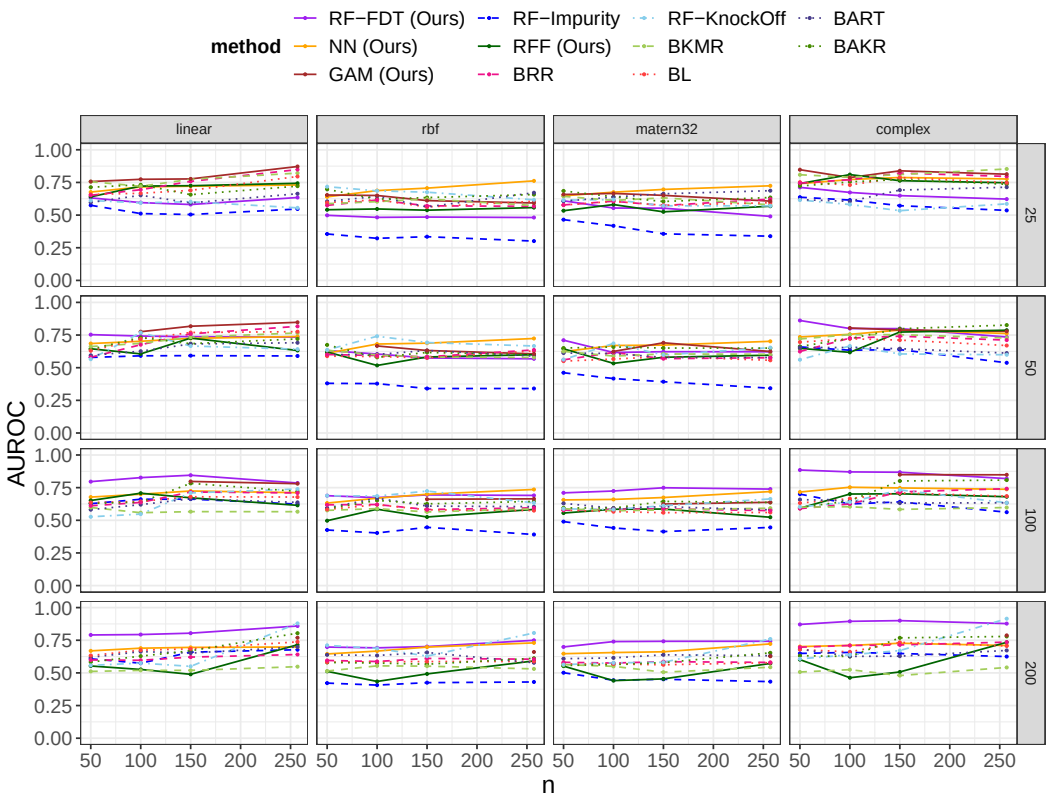


Figure 9: AUROC scores for **heart** data. In relatively low-dimension settings ($d = 25, 50$), the standard methods such as BART has higher AUROC scores than FDT. However, when the dimension is higher ($d = 100, 200$), FDT performs better consistently. Some model (e.g., GAM) reports missing result in $n > p$ setting due to the restrictions in their implementations.

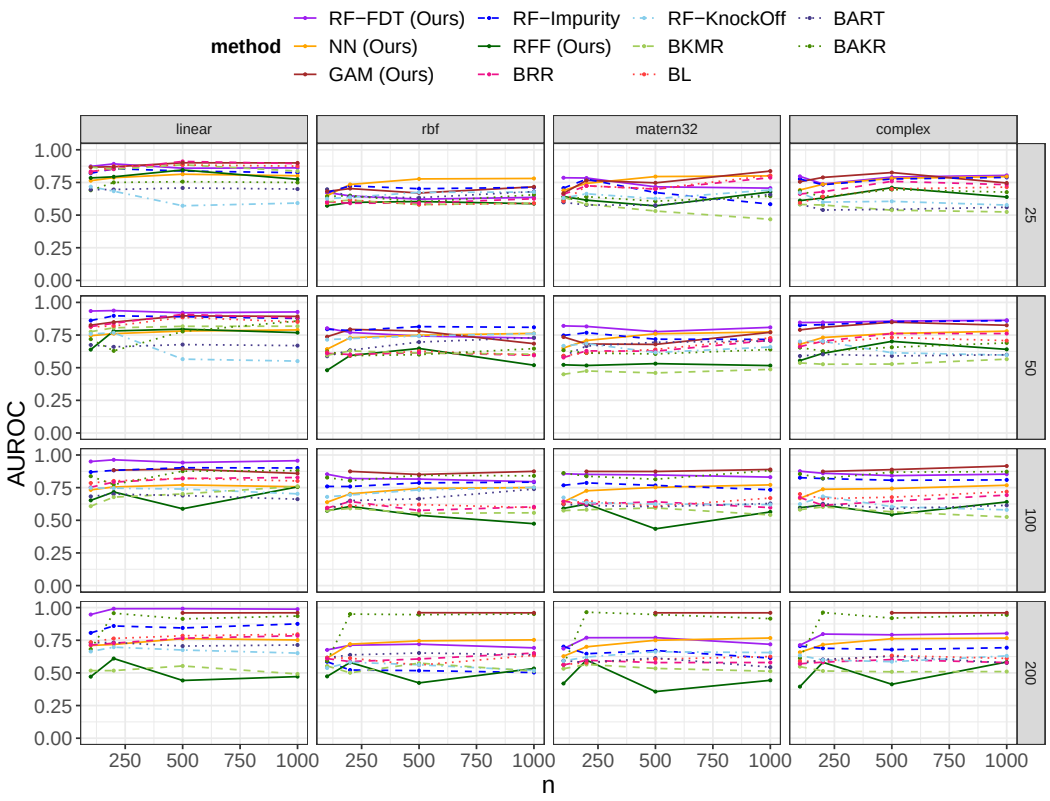


Figure 10: AUROC scores for **mi** data. In low-dimension setting ($d = 25$), the standard methods such as BART has higher AUROC scores than FDT. However, when the dimension is higher ($d = 50, 100$), FDT and GAM perform better consistently. Some model (e.g., GAM) reports missing result in $n > p$ setting due to the restrictions in their implementations.

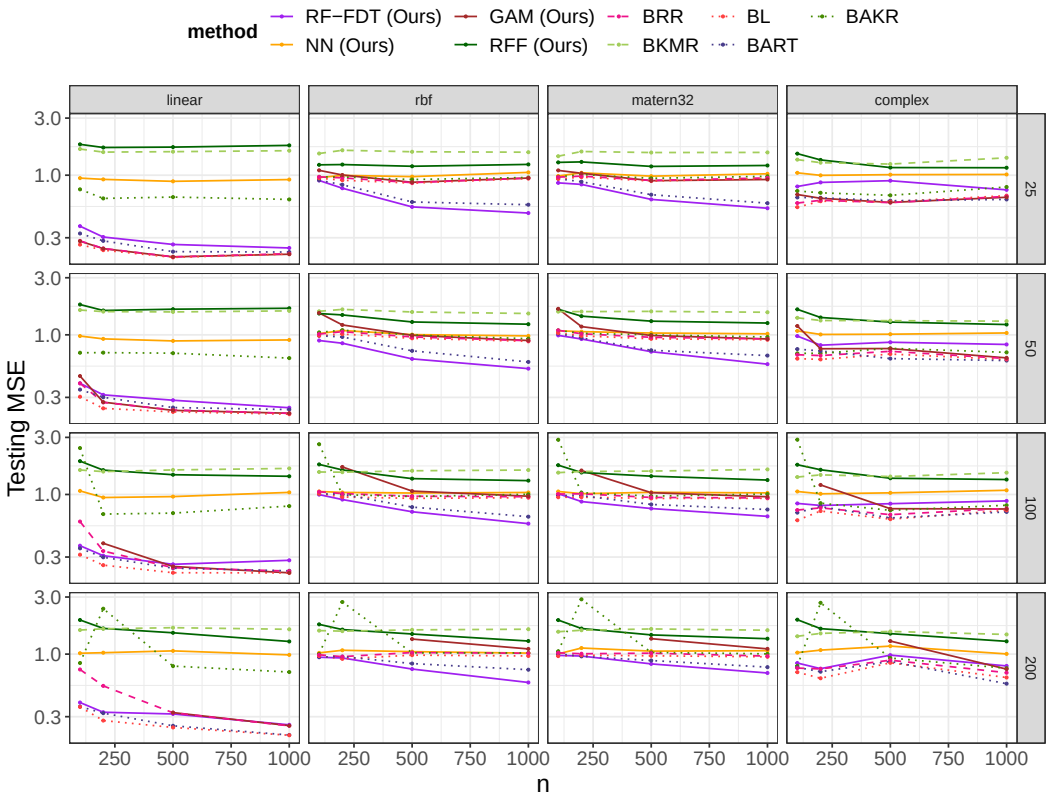


Figure 11: Testing MSE for **synthetic-mixture** data. FDT generally performs better or competitively with baselines, except in the `linear` case where **BL** unsurprisingly does best. **BKMR** consistently performs worse than other methods, except in the low data size, high dimension setting when **BAKR** performs worst. Some model (e.g., GAM) reports missing result in $n > p$ setting due to the restrictions in their implementations. Notice that this dataset contains a setting $n = p$, which can lead to the double descent phenomenon for some random-feature-based models [d'Ascoli et al., 2020].

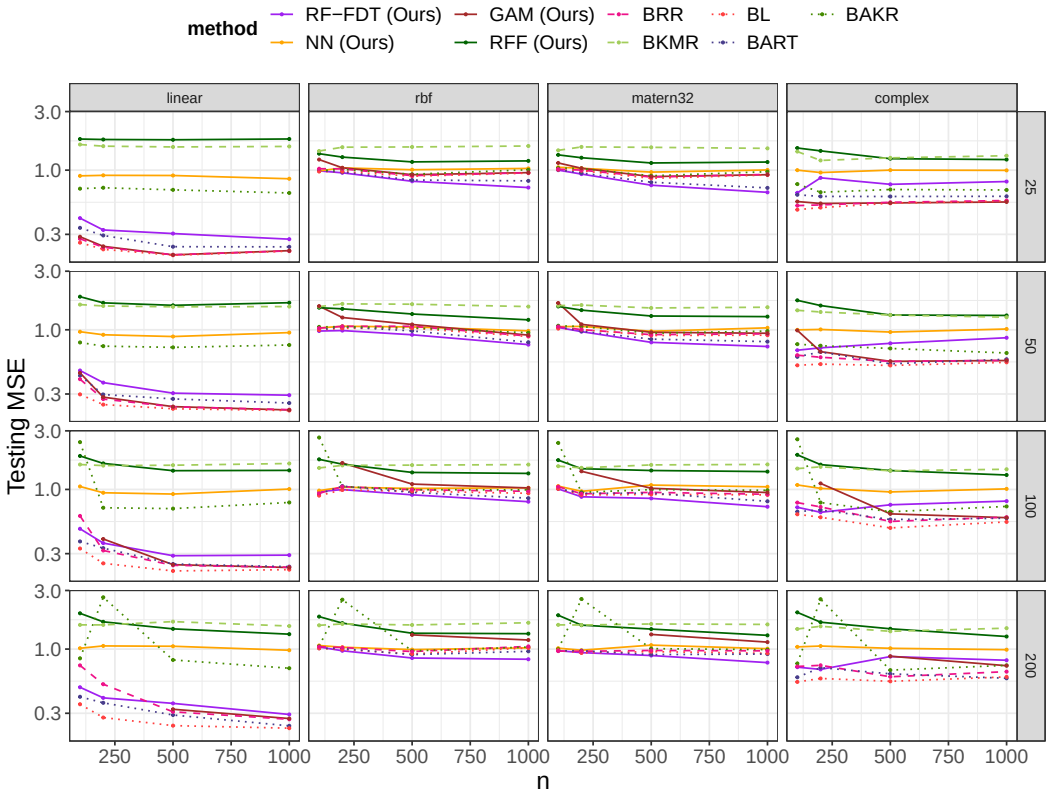


Figure 12: Testing MSE for **synthetic-continuous** data. A method will not be shown if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if the method does not produce valid result due to small sample size (**GAM**).

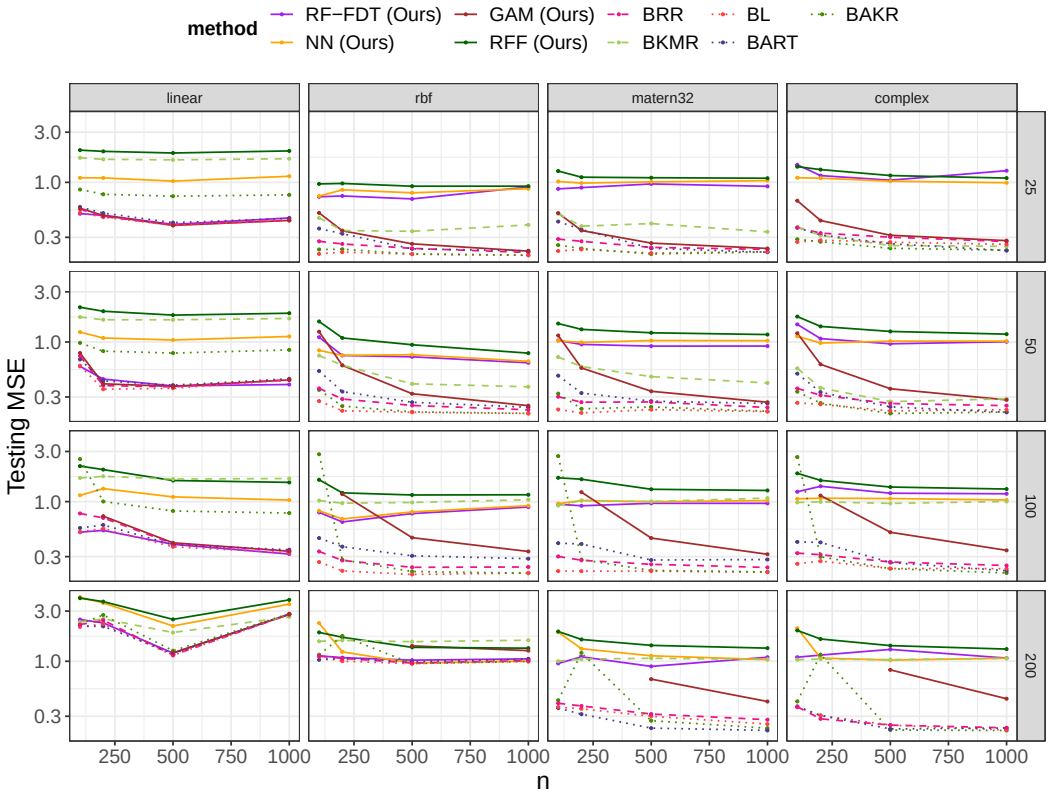


Figure 13: Testing MSE for **adult** data. A method will not be shown if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if the method does not produce valid result due to small sample size (**GAM**).

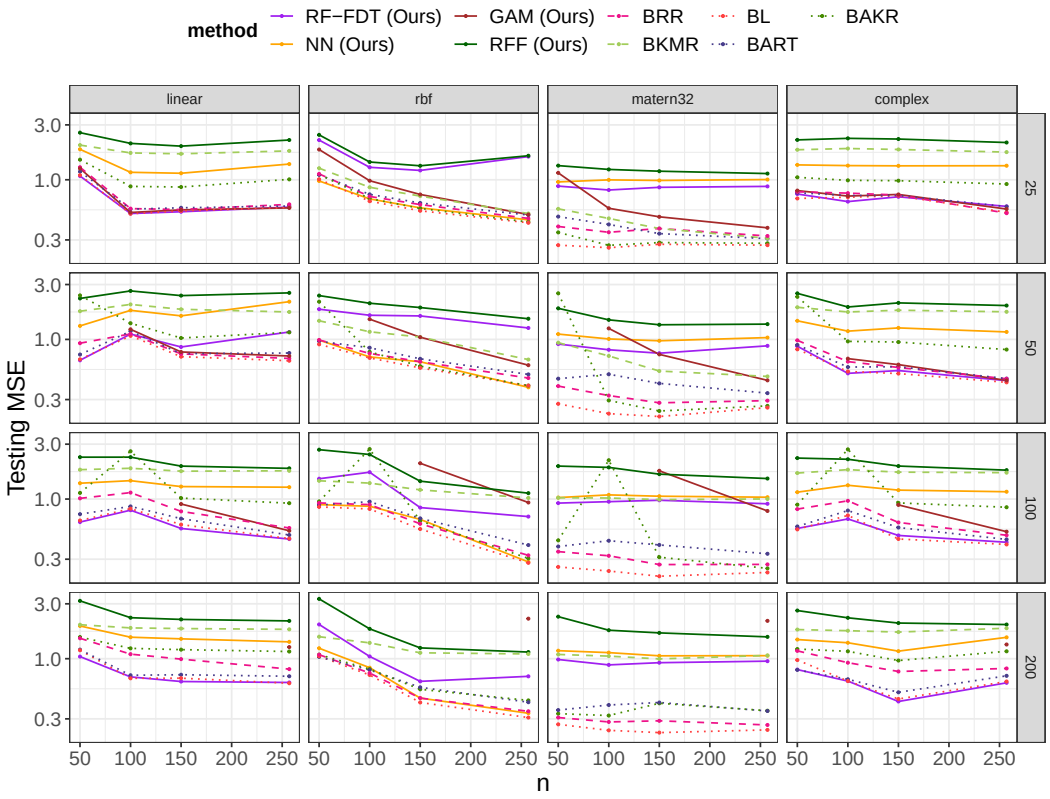


Figure 14: Testing MSE for **heart** data. A method will not be shown if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if the method does not produce valid result due to small sample size (**GAM**). Notice that this dataset contains a setting $n = p$, which can lead to the double descent phenomenon for some random-feature-based models [d'Ascoli et al., 2020].

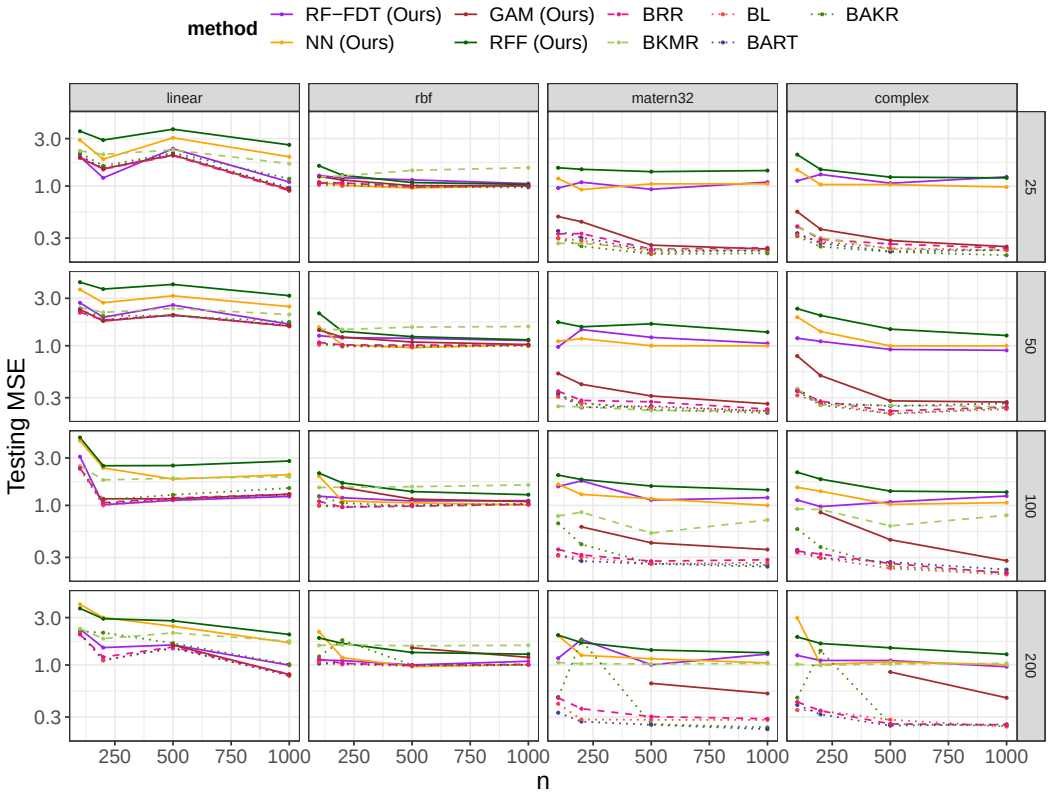


Figure 15: Testing MSE for **mi** data. A method will not be shown if they share the model fit with another method (**RF-impurity** and **RF-knockoff**), or if the method does not produce valid result due to small sample size (**GAM**).

I Additional Experiments: Regularization Path for Bangladesh birth cohort study

We propose a way to visualize the selection path that incorporates the uncertainty of variable importance scores. Specifically, we consider the posterior survival function $S(s) = P(\psi_j > s), j = 1, \ldots, d$ for increasing s starting from 0. Larger value of $S(s)$ indicates larger probability of that certain feature being relevant. This is analogous to the regularization path under the LASSO method. However, our approach incorporates posterior uncertainty, and does not require repeated model fitting at different levels of regularization strength Mairal and Yu [2012].

We apply this to Bangladesh birth cohort study [Kile et al., 2014] (a well-established dataset in the environmental health literature), where we fit models to learn the association between infant's neural development scores and key environmental factors such as hospital location (`clinic`), sex (`sex`), levels of macro nutrient intake (`prot`, `fat`, `carb`, `fib`, `ash`) and levels of measured concentration of environmental toxins in body fluids (`as_ln`, `mn_ln`, `pb_ln`), while controlling for other socio-economic and biological factors (family income, parent education levels, etc). In general, the level of macro-nutrient intake (in particular fiber and protein) indicates a child's general nutrition status (i.e., whether he/she is eating well), and is known to be positively associated with neural development. On the other hand, the existing studies in the Bangladesh population have established a neurotoxic effect between arsenic exposure (i.e., `as_ln`), through drink water) on the early-stage cognitive development [Hamadani et al., 2011], as well as weak but significant effect of the joint mixture of other environmental toxins (manganese (`mn_ln`) and lead (`pb_ln`))) [Gleason et al., 2014, Valeri et al., 2017]. Furthermore, due the fact that the model has already controlled for biological and socio-economic confounding factors, non-nutrient-related factors such as hospital location and sex should not have a significant effect on the children's neural development status.

The result of variable importance estimation is shown in Figure 16, where we plot the posterior survival function $P(\psi_j > s)$ for $s \in (0, 1)$, and compare it to the survival function under *Bayesian Approximate Kernel Regression* (**BAKR**), *Bayesian Ridge Regression* (**BRR**), *Bayesian LASSO* (**BL**), and also the frequentist LASSO regularization path under the GAM model. We normalized all variable importance scores within the range $(0, 1)$. As a result, the variable selection performance is indicated by the relative magnitudes of the area under the curve for each variable (and not by the absolute magnitude due to the normalization).

As shown in Figure 16, the top variables selected by our method (FDT) correspond well with existing conclusions in the literature: it correctly picked up the larger impact of macro-nutrients (in particular, fibre, fat and protein) and smaller but still significant effects of environmental toxins (arsenic, manganese and lead), also notice that it ranked known non-causal factors such as hospital location and sex to be the lowest. In comparison, the linear methods (**GAM**, **BRR** and **BL**) all incorrectly reported high effect from hospital location on children's neural developement outcome (likely due to their restrict model form), while the nonlinear model (BAKR, based on RBF kernel) did not properly pick up the effect of environmental toxins.

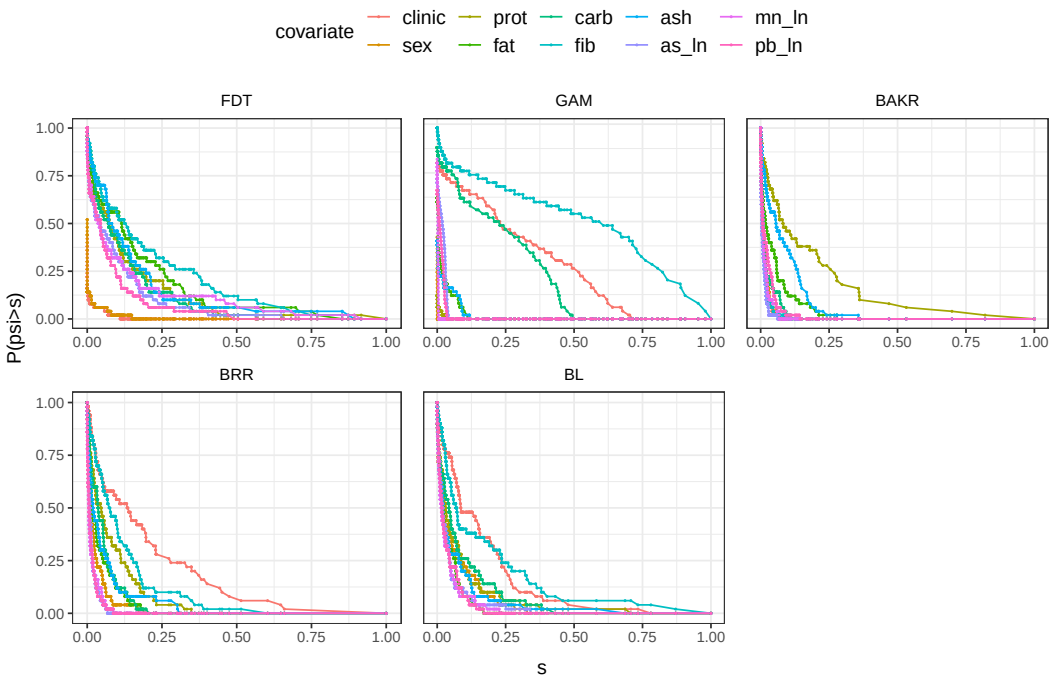


Figure 16: Regularization path for Bangladesh birth cohort study. The top variables selected by our method (FDT) correspond well with established toxicology pathways in the literature.

J Detailed Comparison with Related Work

Liu (2021) [Liu, 2021] is a closely-related work that derives posterior concentration and BvM result for the partial-derivative estimator of variable importance under Bayesian neural networks. Comparing to the current work, [Liu, 2021] focuses narrowly on the Deep Bayesian neural network model, while our work establishes a general set of conditions that is applicable to a much wider classes of machine learning models (GAMs, decision trees, random-feature models, DNNs, ensemble). More specifically, [Liu, 2021]'s result relies on specific assumptions on hidden weights and network width of a ReLU network (Section 2 and Assumption 1), and cannot be generalized straightforwardly to the wider class of models. In comparison, this work established much more generalized conditions in terms of the Lipschitz condition of predictive function, and the effective dimensionality of the model space (see Appendix B.1 for a summary), and we consider in detail the implication of these assumptions for different model classes (Appendix E). On the empirical side, [Liu, 2021] only studied model performance on simulated data sampled i.i.d. from fixed simple distributions, while our work investigated performance of the metric on a wide range of model classes and on non-simulated, realistic data distributions and on correlated, discrete variables (see Section 4 and Appendix G). Furthermore, some of our model variants (e.g., **FDT**) strongly outperforms neural network in almost all situations.

He et al. (2021) [He et al., 2021] is another closely-related work that employ partial derivative-based kernel method to realize variable selection. [He et al., 2021] shares similarity with this work in that we both consider gradient norm as the variable importance estimator, but with drastically different focus in theoretical results and empirical investigations. Specifically, [He et al., 2021] (1) studies a frequentist variable selection approach based on thresholding (2) does not address uncertainty in the variable importance estimators, and (3) focuses on classical, non-adaptive Gaussian process kernels (e.g., linear, quadratic, RBF kernels, see Sections 4 and 6 of [He et al., 2021]) and does not consider generalization across modern ML models. In comparison, our framework (1) focuses on Bayesian estimation of variable importance, (2) explicitly incorporates model uncertainty and derives its theoretical guarantee, (3) considers and empirically investigates the generalization of the approach across wide range of modern ML models, in particular tree-ensemble models whose compatibility to kernel-based variable importance method is not obvious.

K Further Discussion of Limitations

Our proposed framework provides principled uncertainty quantification by performing exact Bayesian inference on the weights $\boldsymbol{\beta}$ of a feature map $\phi(\mathbf{x})$. We do not consider uncertainty in the feature map itself. This means, for example, that if the feature map is given by the last hidden layer of a neural network trained by maximizing the posterior, then our model class corresponds to the *neural linear model*. This model is different from a fully Bayesian neural network, which performs posterior inference also on the kernel hyperparameters (i.e., the hidden weights) Ober and Rasmussen [2019], Snoek et al. [2015], Thakur et al. [2021]. Likewise, the kernel induced by the featurized decision tree studied here does not consider uncertainty in the tree's partitioning process. However, as discussed in the method section (Section 2.1), this "linearity" of the model parameter does not impact the expressiveness of the GP model, since the basis functions $\phi = \{\phi_k\}_{k=1}^D$ themselves are nonlinear and are allowed to be updated as part of the learning process. At the same time, the fact that the full posterior inference is performed only with respect to $\boldsymbol{\beta}$ indeed places a limitation on the model's ability in uncertainty quantification, as the uncertainty in the model hyperparameters is not accounted for. Yet, this does not seem to be a **significant limitation in the method's empirical performance** (e.g., **FDT** outperforms **BART** in our experiments), although this point still merits further investigation in the future.

The theoretical results of the current paper assume fixed data dimension $d = O(1)$. However, this does not restrict the significance and practicality of our theoretical results. On the theoretical side, even for fixed d, a general framework to nonparametric Bayesian inference of feature importance is currently missing in the field. On the application side, the majority of the machine learning applications fall into the fixed dimension setting. For example, in vision tasks, we usually handle images with fixed dimensions (i.e. height, width, and number of channels); in language tasks, we handle sentences with fixed vocabulary size and maximum sentence length; and in tabular tasks, we often work with tables with fixed number of columns. Furthermore, as commented in Section 3.2, the posterior concentration of variable importance (Theorem 1) does not rely on this assumption in its proof. Therefore, posterior concentration can occur even for high-dimensional settings with $d = o(n)$. In contrast, the BvM theorem usually represents a much stronger type of convergence (i.e., convergence in the predictive CDF of the entire Bayesian predictive posterior) and usually requires a stricter set of conditions. This is especially true in our setting, where our BvM results consider the quadratic functional of the nonlinear model, rather than the predictive function of the model itself. To this end, we highlight that most of the modern BvM results in ML models are derived by assuming fixed dimension. This includes [Wang and Rocková, 2020] for Bayesian neural networks, [Burnaev et al., 2013, Yang et al., 2015] for Gaussian process models, [Rockova, 2020] for BART (i.e., Bayesian tree models), and finally [Castillo and Rousseau, 2015] for general semi-parametric models. Therefore, our setting is consistent with the modern literature in Bernstein von-Mises results for ML models. At the same time, it should be acknowledged that in recent years, there have been a few "high-dimensional" Bernstein von Mises results for specialized settings. The more recent ones include [Bontemps, 2011] for linear regression, and [Lu, 2017] for Bayesian inverse problems with nonlinear forward dynamics. However, extending these results to the *quadratic functionals* of a *general* nonlinear model is non-trivial and out-of-scope for our current work, but is an important and interesting direction of future theoretical investigations.

In our experiments, we focused on kernels based on tree ensembles, kernel methods and linear models. In the future, it would be worth expanding this framework to other model classes (e.g., MARS Friedman [1991]) and estimating the importance of interaction effects and higher-order terms. We would also like to apply this method to large-scale scientific studies (e.g., epidemiology studies based on extremely large EHR datasets) where an uncertainty-aware nonlinear variable importance estimation method is typically impossible due to challenges with scalability.