

# Mixture of Linear Models Co-supervised by Deep Neural Networks

Beomseok Seo

Department of Statistics, The Pennsylvania State University

Lin Lin

Department of Biostatistics and Bioinformatics, Duke University

Jia Li

Department of Statistics, The Pennsylvania State University

## Abstract

Deep neural networks (DNN) have been demonstrated to achieve unparalleled prediction accuracy in a wide range of applications. Despite its strong performance, in certain areas, the usage of DNN has met resistance because of its black-box nature. In this paper, we propose a new method to estimate a mixture of linear models (MLM) for regression or classification that is relatively easy to interpret. We use DNN as a proxy of the optimal prediction function such that MLM can be effectively estimated. We propose visualization methods and quantitative approaches to interpret the predictor by MLM. Experiments show that the new method allows us to trade-off interpretability and accuracy. The MLM estimated under the guidance of a trained DNN fills the gap between a highly explainable linear statistical model and a highly accurate but difficult to interpret predictor.

*Keywords:* Explainable machine learning, DNN co-supervision, Explainable dimensions, Explainable conditions, Interpretation of DNN

## 1 Introduction

Deep neural network (DNN) models have achieved phenomenal success for applications in many domains, ranging from academic research in science and engineering to industry and business. The modeling power of DNN is believed to have come from the complexity and

over-parameterization of the model, which on the other hand has been criticized for the lack of interpretation. Although certainly not true for every application, in some applications, especially in economics, social science, healthcare industry, and administrative decision making, scientists or practitioners are resistant to use predictions made by a black-box system for multiple reasons. One reason is that a major purpose of a study can be to make discoveries based upon the prediction function, *e.g.*, to reveal the relationships between measurements. Another reason can be that the training dataset is not large enough to make researchers feel completely sure about a purely data-driven result. Being able to examine and interpret the prediction function will enable researchers to connect the result with existing knowledge or gain insights about new directions to explore. Although classic statistical models are much more explainable, their accuracy often falls considerably below DNN. In this paper, we propose an approach to fill the gap between relatively simple explainable models and DNN such that we can more flexibly tune the trade-off between interpretability and accuracy. Our main idea is a mixture of discriminative models that is trained with the guidance from a DNN. Although mixtures of discriminative models have been studied before, our way of generating the mixture is quite different.

## 1.1 Related Work

Despite the fact that many attempts have been made to make DNN models more interpretable, a formal definition of “human interpretability” has remained elusive. The concept of interpretability is multifaceted and inevitably subjective. An in-depth discussion about the meaning of being interpretable is given by the review article of Doshi-Velez and Kim (2017), which confirms the richness of this concept and provides philosophical viewpoints. In the literature, different approaches have been proposed to define “interpretation”. A complete unified taxonomy for all the existing approaches does not exist. Nevertheless, one way is to categorize recent works into two types: one that builds more interpretable models by reducing model complexity, and the other that attempts to interpret a complex model by examining certain aspects of it, *e.g.*, how decisions are made locally. For the first type of approaches, interpretability is aimed at during the model’s training phase,

and what constitutes good interpretability has been addressed in diverse ways. For example, attention-based methods (Bahdanau et al., 2014; Vaswani et al., 2017) propose an attention score for neural machine translation; generalized additive models (GAM) (Lou et al., 2013; Agarwal et al., 2020; Guo et al., 2020) interpret pair-wise interaction effects by imposing an additive structure; and feature selection methods impose cross-entropy error function (Verikas and Bacauskiene, 2002) or  $L_0$  penalization (Tsang et al., 2018). In contrast, the second type of approaches, the so-called model-agnostic methods (Ribeiro et al., 2016b), are post-hoc in the sense that they analyze an already trained model using interpretable measurements. For example, sensitivity analysis (Lundberg and Lee, 2017), local linear surrogate models (Ribeiro et al., 2016a) and Bayesian non-parametric mixture model (Guo et al., 2018) search for important features for specific samples according to a given prediction model. The same aspect of being interpretable can be tackled by either type of the approaches. For example, Tsang et al. (2017) searches for interaction effects of original features as captured in a trained neural network by computing a strength score based on the learned weights. In Tsang et al. (2018), similar results can be obtained by training the neural network model with a penalty to learn the interaction effects.

Various kinds of interpretation have been suggested, including but not limited to visualizing the result, generating human explainable rules, selecting features, or constructing prototype cases. The pros and cons of existing methods are discussed in Molnar (2020). What kind of interpretation is useful also depends on the application field. For instance, feature selection for image analysis is usually not as meaningful as that for tabular data. Many proposed methods target particular applications, *e.g.*, image analysis (Zhang et al., 2018; Chen et al., 2019), text mining (Vaswani et al., 2017), time series (Guo et al., 2019).

Ribeiro et al. (2016a) interprets neural networks by explaining how the decision is made in the vicinity of every instance. In particular, the trained neural network is used to generate pairs of input and output quantities, based on which a linear regression model is estimated. This linear model is used to explain the decision in the neighborhood of that instance. Although the local linear models help understand the prediction around every single point, they are far from providing global perspectives. Our new method is inspired

by Ribeiro et al. (2016a), but we tackle two additional problems. First, our method will produce a stand-alone prediction model that is relatively easy to interpret. In another word, our method is not the model-agnostic type to explain an actual operating DNN. Second, our method aims at globally interpreting the decision.

Although motivated from rather different aspects, in terms of the formulation of the model, our method is related to locally weighted regression (Cleveland and Devlin, 1988). In other words, our method splits the space of the original independent variables by exploiting a trained DNN and conducts linear regression inside each local region. In non-parametric regression, many other flexible regression models have been studied, for instance, kernel regression (Nadaraya, 1964; Watson, 1964), generalized additive models (Hastie and Tibshirani, 1990), and classification and regression trees (CART) (Breiman et al., 1984). Some of the methods such as CART are by construction highly interpretable, but some are not. More recently, Guo et al. (2020) have used neural networks to efficiently learn the classic GAM (Hastie and Tibshirani, 1990) models.

Our method is in spirit similar to the Mixture of Experts (MOE) model or a fuzzy system (Jacobs et al., 1991; Takagi and Sugeno, 1985) although profound differences exist in practically important aspects. MOE divides the feature space into regions and applies localized experts in each region. The study on MOE has focused on finding better ways to create the individual experts and to combine them into a single predictor. The approaches for creating the individual experts include stochastic partitioning (Ebrahimpour et al., 2008), negatively correlated experts (Masoudnia et al., 2012), and clustering-based experts (*e.g.*, self-organizing maps (SOM)) (Tang et al., 2002). Weights to combine the experts have been computed by methods such as gating network (Kuncheva, 2014) and boosting (Kégl, 2013). However, the MOE method partitions the feature space using only the features. Our method is different because it uses a pre-trained DNN to partition the samples, in a supervised manner based on the prediction of the DNN. Our method is motivated by finding an easier interpretation while preserving the accuracy of a DNN. Experimental results are provided to compare our method with the mixture of experts approach.

## 1.2 Overview of Our Approach

One natural idea to explain a complex model is to view the decision as a composite of decisions made by different functions in different regions of the input space. Explanation of the overall model consists of two parts: to explain how the partition of the input space is formed, and to explain the prediction in each region. This idea points to the construction of a mixture model in which every component is a linear model, which is called the *mixture of linear models* (MLM). Although a linear model is not necessarily always easy to explain, relatively speaking, it is explainable, and via LASSO-type sparsity penalty (Tibshirani, 1996) it can be made increasingly more explainable.

The main technical hurdle is to find a suitable partition of the instances such that component-wise linear models can be estimated. The challenge faced here is in stark contrast to that for building generative mixture models, *e.g.*, Gaussian mixture models (GMM), where the proximity of the independent variables themselves roughly determines the grouping of instances into components. The same approach of forming components is a poor choice for building a mixture of discriminative models. The similarity between the instances is no longer measured by the proximity of the independent variables but by the proximity of the relationships between the dependent variable ( $Y$ ) and the independent variables ( $X$ ). As a related approach, cluster-weighted modeling (CWM) (Gershenfeld, 1997; Ingrassia et al., 2012) takes into account the joint distribution of  $X$  and  $Y$  when creating a generative mixture model. However, in CWM, the grouping of  $X$  and the estimation of local models are treated separately. We will compare the performance of CWM and MLM in the experiments.

The dependence between  $Y$  and  $X$  cannot be adequately captured by the observation of one instance. In the case of moderate to high dimensions, it is also impractical to estimate the discriminative function,  $\hat{Y} = f(X)$ , based on nearby points. Our key idea is to use the DNN model as an approximation of the theoretically optimal prediction function, which is employed to guide the partition of the instances. We also propose to use visualization based on GMM and decision tree to interpret the partition of the instances, which is crucial for obtaining interpretation in a global sense.

It is debatable whether DNN serves well as an approximation of the optimal prediction function. In many applications, we observe that DNN achieves the best accuracy among the state-of-the-art methods, *e.g.*, support vector machine (SVM) (Cortes and Vapnik, 1995) and random forest (RF) (Breiman, 2001). It is thus reasonable to exploit DNN in this manner. However, we acknowledge that a weakness of our approach is that the accuracy is capped by DNN, and our approach is likely to suffer when DNN itself performs poorly.

The rest of the paper is organized as follows. In Section 2, we introduce notations and summarize existing methods most relevant to our proposed work. We present MLM in Section 3.1 and describe the tools developed for interpretation based upon MLM in Section 3.2. In Section 4, experimental results are reported for four real-world datasets including two clinical datasets for classification and two for regression. For prediction accuracy, comparisons have been made with multiple approaches including DNN. We also illustrate how the interpretation tools are used. Finally, conclusions are drawn in Section 5.

## 2 Preliminaries

Let  $X \in \mathbb{R}^p$  be the independent variables (or covariates) and  $Y \in \mathbb{R}$  be the dependent variable. Denote the sample space of  $X$  by  $\mathcal{X}$ . Let  $\{y_i\}_{i=1}^n$  and  $\{\mathbf{x}_i\}_{i=1}^n = \{(x_{i,1}, \dots, x_{i,p})^T\}_{i=1}^n$  be the  $n$  observations of  $Y$  and  $X$ . Denote the input data matrix by  $\mathbf{X} \in \mathbb{R}^{n \times p}$ . In regression analysis, we are interested in estimating the following regression function for any  $\mathbf{x} \in \mathcal{X}$  (for classification, we simply substitute  $Y$  by  $g(Y)$  via a link function  $g(\cdot)$ ).

$$m(\mathbf{x}) = \mathbb{E}(Y|X = \mathbf{x}) . \quad (1)$$

In linear regression, it is assumed that  $m(\mathbf{x}) = \alpha + \mathbf{x}^T\beta$ . In non-linear regression, a linear expansion of basis functions is often used to estimate  $m(\mathbf{x})$ . For example, Nadaraya-Watson kernel regression (Nadaraya, 1964; Watson, 1964) assumes  $m(\mathbf{x})$  to be an additive form of kernel functions with a given bandwidth  $h$ :

$$m(\mathbf{x}) = \sum_{i=1}^n \frac{G(\frac{\mathbf{x}-\mathbf{x}_i}{h})y_i}{\sum_{i'=1}^n G(\frac{\mathbf{x}-\mathbf{x}_{i'}}{h})} , \quad (2)$$

where  $G(\mathbf{x}) = (2\pi)^{-k/2}e^{-\mathbf{x}^T\mathbf{x}/2}$ . The locally weighted regression (Cleveland and Devlin, 1988) extends Nadaraya-Watson’s  $m(\mathbf{x})$  by changing the constant prediction in each band to a linear function. Regression splines use piecewise polynomial basis functions between fixed points, known as knots, and ensure smoothness at the knots (Schoenberg, 1973). Friedman (1991) extended the spline method to handle higher dimensional data. These kernel and spline-based approaches use the basic binning strategy to separate the covariate space into regions, suffering from curse of dimensionality even at moderate dimensions.

In recent years, DNN has become increasingly popular in high-dimensional applications because of its remarkable prediction accuracy. Consider a feed-forward neural network with  $L$  hidden layers and one output layer. Its regression function, denoted by  $\tilde{m}(\mathbf{x})$ , is defined by the composition of the affine transform and a non-linear activation function at each layer. Let  $p_l$  be the number of hidden units at the  $l$ -th hidden layer,  $l = 0, \dots, L$ , and  $p_0 = p$ . Let  $\mathbf{z}^{(l)} \in \mathbb{R}^{p_l}$  be the outputs of the  $l$ -th hidden layer. Set  $\mathbf{z}^{(0)} = \mathbf{x}$ . The mapping at the  $l$ -th hidden layer,  $\mathbf{z}^{(l)} = h_l(\mathbf{z}^{(l-1)})$ , is defined by

$$\mathbf{z}^{(l)} = h_l(\mathbf{z}^{(l-1)}) = \sigma_l(\mathbf{W}^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1, \dots, L, \quad (3)$$

where  $\sigma_l(\cdot)$  is a non-linear element-wise activation function such as ReLU (Nair and Hinton, 2010), sigmoid or hyperbolic tangent, and  $\mathbf{W}^{(l)} \in \mathbb{R}^{p_l \times p_{l-1}}$  and  $\mathbf{b}^{(l)} \in \mathbb{R}^{p_l}$  are the model parameters. At the output layer,  $g(\mathbf{z}^{(L)})$ , is defined as either a linear or softmax function depending on whether the purpose is regression or classification. In summary, the neural network model,  $\tilde{m}(\mathbf{x})$ , is given by

$$\tilde{m}(\mathbf{x}) = (g \circ h_L \circ h_{L-1} \circ \dots \circ h_1)(\mathbf{x}), \quad (4)$$

and its parameters are trained by gradient descent algorithm so that the mean squared error loss for regression or the cross-entropy loss for classification is minimized. Specifically, the mean squared error loss is defined by  $\sum_{i=1}^n [y_i - \tilde{m}(\mathbf{x}_i)]^2 / n$ , and the cross-entropy loss is  $-\sum_{i=1}^n [y_i \log(\tilde{m}(\mathbf{x}_i)) + (1 - y_i) \log(1 - \tilde{m}(\mathbf{x}_i))] / n$ . Because of the multiple layers and the large number of hidden units at every layer, it is difficult to interpret a DNN model, for example, to explain the effects of the input variables on the predicted variable.

### 3 Methods

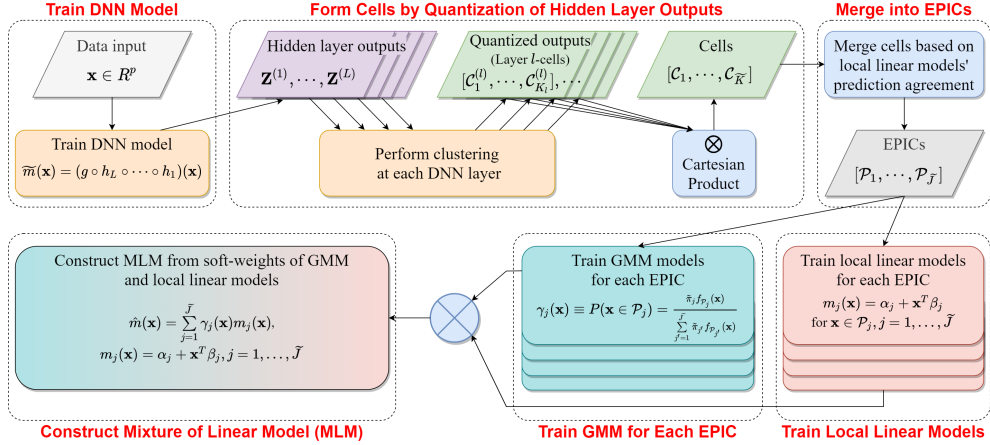


Figure 1: A schematic plot showing the steps of creating MLM.

Our core idea is to approximate the prediction function  $\tilde{m}(\mathbf{x})$  of a DNN model by a piecewise linear function  $\hat{m}(\mathbf{x})$  called *Mixture of Linear Models (MLM)*. We will present in the context of regression. Adaption to classification will be remarked upon later. Suppose the input sample space  $\mathcal{X}$  is divided into  $K$  mutually exclusive and collectively exhaustive sets,  $\{\mathcal{P}_1, \dots, \mathcal{P}_K\}$ , which form a *partition* of  $\mathcal{X}$ . Denote the partition by  $\mathcal{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_K\}$ . The partition  $\mathcal{P}$  induces a row-wise separation of the input data matrix  $\mathbf{X}$  into  $K$  submatrices:  $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(K)}$ . Within each  $\mathcal{P}_k$ ,  $\tilde{m}(\mathbf{x})$  is approximated by  $m_k(\mathbf{x})$ , which is referred to as the *local linear model*. To create  $\mathcal{P}$ , our criterion is not based on the proximity of  $\mathbf{x}_i$ 's but on the similarity between  $\tilde{m}(\mathbf{x})$  in the neighborhood of each  $\mathbf{x}_i$ . Because of this difference from the conventional mixture model, we are motivated to cluster  $\mathbf{x}_i$ 's by simulating from the prediction function of an estimated DNN. Without the guidance of the DNN, there will not be enough training points in the neighborhood of any  $\mathbf{x}_i$  to permit a reasonable estimation of the prediction function unless the dimension is very low.

Let  $I_{\mathcal{P}}(\mathbf{x})$  be the indicator function that equals 1 when  $\mathbf{x} \in \mathcal{P}$  and zero otherwise. Given the partition  $\mathcal{P}$ , we express  $\hat{m}(\mathbf{x})$  by

$$\begin{aligned} \hat{m}(\mathbf{x}) &= I_{\mathcal{P}_1}(\mathbf{x})m_1(\mathbf{x}) + \dots + I_{\mathcal{P}_K}(\mathbf{x})m_K(\mathbf{x}), \\ m_k(\mathbf{x}) &= \alpha_k + \mathbf{x}^T \beta_k, \quad k = 1, \dots, K. \end{aligned} \tag{5}$$



A schematic plot is provided in Figure 1 to show the major steps in generating an MLM. The partition  $\mathcal{P}$  is obtained by a two-stage process. First, clustering is performed based on the outputs at each layer of the DNN. This stage provides refined clusters, referred to as *cells*, of the data points and helps reduce computation at the next stage. Secondly, the cells are merged into larger clusters based on the similarity of  $\tilde{m}(\mathbf{x})$  computed via simulations. Clusters obtained at this stage become  $\mathcal{P}_k$ 's in  $\mathcal{P}$ . We refer to  $\mathcal{P}_k$  as an *EPIC* (*Explainable Prediction-induced Input Cluster*). A linear model is then fitted based on the data points in each EPIC as well as simulated data generated using the DNN. GMM models are then formed for each EPIC so that test data can be classified into the EPICs. Finally, a soft-weighted MLM modified from Eq. (5) is used as  $\hat{m}(\mathbf{x})$ .

The interpretability of MLM relies on both the local linear models  $m_k(\mathbf{x})$  and the characterization of EPICs. We develop a visualization method and a decision-tree based method to help users explain EPICs and to open avenues for potential discovery. These methods are presented in Section 3.2. Next, we describe the steps to build an MLM.

### 3.1 Construction of a Mixture of Linear Models

In Subsection 3.1.1, an approach is presented to approximate a neural network by a piecewise linear model and obtain cells. The approach to merge cells and obtain EPICs is described in Subsection 3.1.2. Finally, we describe the soft-weighted form of MLM in Subsection 3.1.3.

#### 3.1.1 Piecewise Linear Approximation of DNN

The neural network model  $\tilde{m}(\mathbf{x})$  in Eq. (4) is non-linear due to the non-linear activation functions  $\sigma_l(\cdot)$  at each hidden layer. If the outputs at each layer are partitioned such that the non-linearity can be neglected within a cluster, the mapping of DNN from the original input to the prediction is approximately linear for data points that belong to the same cluster across all the layers. It is difficult to find such a partition directly from the original feature space  $\mathcal{X}$  because points yielding similar outputs through the DNN layers are not necessarily close in  $\mathcal{X}$ . On the other hand, because the  $l$ -th hidden layer output  $\mathbf{z}^{(l)}$  in

DNN is formed by the composite of monotone functions and linear functions of the input, clusters generated based on output  $\mathbf{z}^{(l)}$  correspond to polytopes in the input space. Within each polytope, the mapping from the input to the output is approximately linear. After we conduct clustering of  $\mathbf{z}^{(l)}$  at each layer, the final cluster is determined by the sequence of clusters through the layers. Within each final cluster, the DNN mapping from the input to the output at every layer is approximately linear, and thus the overall mapping is linear.

Let  $\{\mathcal{C}_1^{(l)}, \dots, \mathcal{C}_{K_l}^{(l)}\}$  be a partition of the  $l$ -th layer outputs of the DNN, where  $K_l$  is the number of clusters. We call this partition *layer  $l$ -cells*. Then, with layer  $l$ -cells, we can approximate the  $l$ -th hidden layer map  $h_l(\mathbf{z}^{(l-1)})$  as a piecewise linear function:

$$\hat{h}_l(\mathbf{z}^{(l-1)}) = I_{\mathcal{C}_1^{(l)}}(\mathbf{z}^{(l-1)})(\mathbf{W}_1^{(1)}\mathbf{z}^{(l-1)} + \mathbf{b}_1^{(1)}) + \dots + I_{\mathcal{C}_{K_l}^{(l)}}(\mathbf{z}^{(l-1)})(\mathbf{W}_{K_l}^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}_{K_l}^{(l)}), \quad (6)$$

where  $\mathbf{W}_k^{(l)} \in \mathbb{R}^{p_l \times p_{l-1}}$  and  $\mathbf{b}_k^{(l)} \in \mathbb{R}^{p_l}$  are model parameters, and  $I_{\mathcal{C}_k^{(l)}}(\mathbf{z}^{(l-1)})$  is the indicator function.

The layer  $l$ -cells are obtained by clustering the observed  $l$ -th hidden layer outputs. Denote by  $\{\mathbf{z}_i^{(l)}\}_{i=1}^n$  the observed  $l$ -th hidden layer outputs corresponding to  $\{\mathbf{x}_i\}_{i=1}^n$  respectively, that is,  $\mathbf{x}_i \rightarrow \mathbf{z}_i^{(l)}$ . We apply GMM-based clustering on  $\{\mathbf{z}_i^{(l)}\}_{i=1}^n$  with  $K_l$  clusters, and obtain the following GMM model. Denote the density function by  $f(\cdot)$ , and the prior, mean, and covariance matrix of each Gaussian component by  $\pi_k^{(l)}$ ,  $\mu_k^{(l)}$ , and  $\Sigma_k^{(l)}$ . Then

$$f(\mathbf{z}^{(l)}) = \sum_{k=1}^{K_l} \pi_k^{(l)} \phi(\mathbf{z}^{(l)} | \mu_k^{(l)}, \Sigma_k^{(l)}). \quad (7)$$

As usual, the maximum a posteriori (MAP) criterion is applied to cluster  $\mathbf{z}_i^{(l)}$ 's based on the GMM. We apply GMM clustering on each hidden layer separately to compute layer  $l$ -cells. Denote the set of cluster labels for the layer  $l$ -cells by  $\mathcal{K}_l = \{1, \dots, K_l\}$ . The set of all possible sequences of the cluster labels across the  $L$  layers is the Cartesian product  $\tilde{\mathcal{K}} = \mathcal{K}_1 \times \dots \times \mathcal{K}_L$ . Let the cardinality of  $\tilde{\mathcal{K}}$  be  $\tilde{K}_0 = \prod_{l=1}^L K_l$ . For each sequence in  $\tilde{\mathcal{K}}$ ,  $(k_1, \dots, k_L)$ ,  $k_l \in \mathcal{K}_l$ , assign it a label  $k$ ,  $1 \leq k \leq \tilde{K}_0$ , according to the lexicographic order. Denote the mapping by  $(k_1, \dots, k_L) \rightarrow k$ . Then, the original input  $\mathbf{x}_i$  belongs to cluster  $\mathcal{C}_k$  if  $(k_1, \dots, k_L) \rightarrow k$  and  $\mathbf{z}_i^{(l)} \in \mathcal{C}_{k_l}^{(l)}$  for  $l = 1, \dots, L$ . It can often occur that no  $\mathbf{x}_i$  corresponds to a particular sequence in  $\tilde{\mathcal{K}}$ . Thus the actual number of clusters formed, denoted by  $\tilde{K}$ , is smaller than  $\tilde{K}_0$ .

We call the clusters  $\mathcal{C}_k$  *cells*. After obtaining the cells, we estimate a linear model for each cell. As in Eq.(5), the linear model for the  $k$ th cell is  $m_k(\mathbf{x})$ . To train each local linear model  $m_k(\mathbf{x})$ , we use both the original data points that belong to a cell  $k$  and simulated data points perturbed from the original points with responses generated by the DNN model  $\tilde{m}(\mathbf{x})$ . A major reason for adding the simulated sample is that a cell usually does not contain sufficiently many points for estimating  $m_k(\mathbf{x})$ , the very difficulty of high dimensions. By the same rationale, we attempt to mimic the decision of the DNN, which is an empirically strong prediction model trained using the entire data. If the DNN can be well approximated, we expect MLM to perform strongly as well. Estimating a linear model using simulated data generated by the DNN is a data-driven approach to approximate the DNN locally. We thus aptly call this practice *co-supervision* by DNN.

Let  $n_k$  be the number of observations in  $\{\mathbf{x}_i\}_{i=1}^n$  that belong to cell  $\mathcal{C}_k$ . We have  $\sum_{k=1}^{\tilde{K}} n_k = n$ . Without loss of generality, let  $\{\mathbf{x}'_{k,i}\}_{i=1}^{n_k}$  be the set of points that belong to cell  $\mathcal{C}_k$ :  $\{\mathbf{x}'_{k,i}\}_{i=1}^{n_k} = \{\mathbf{x}_j | \mathbf{x}_j \in \mathcal{C}_k \text{ for } j = 1, \dots, n\}$  with an arbitrary ordering of  $\mathbf{x}'_{k,1}, \dots, \mathbf{x}'_{k,n_k}$ , and let  $y'_{k,i}$  for  $i = 1, \dots, n_k$  be the corresponding dependent variable of  $\mathbf{x}'_{k,i}$  for  $i = 1, \dots, n_k$ . We denote by  $\bar{\mathbf{x}}'_k$  the sample mean of  $\mathbf{x}'_{k,1}, \dots, \mathbf{x}'_{k,n_k}$ :  $\bar{\mathbf{x}}'_k = \frac{\sum_{i=1}^{n_k} \mathbf{x}'_{k,i}}{n_k}$ . Then we generate  $m$  perturbed sample points  $\mathbf{v}_{k,1}, \dots, \mathbf{v}_{k,m}$  by adding Gaussian noise to the mean  $\bar{\mathbf{x}}'_k$  with a pre-specified variance parameter  $\epsilon$  for each cell  $k = 1, \dots, \tilde{K}$ . That is,

$$\mathbf{v}_{k,i} \sim \mathcal{N}(\bar{\mathbf{x}}'_k, \epsilon I_p), \text{ for } i = 1, \dots, n_k. \quad (8)$$

For the perturbed sample points  $\{\mathbf{v}_{k,i}\}_{i=1}^m$ , the predicted dependent variable  $\{w_{k,i}\}_{i=1}^m$  is computed using the DNN model  $\tilde{m}(\cdot)$ :

$$w_{k,i} = \tilde{m}(\mathbf{v}_{k,i}), \quad i = 1, \dots, m. \quad (9)$$

In the case of classification, the prediction of DNN is the probabilities of the classes. We then generate a class label by taking the maximum. We combine the original sample points and the simulated ones,

$$\{(\mathbf{x}'_{k,i}, y'_{k,i})\}_{i=1}^{n_k} \cup \{(\mathbf{v}_{k,i}, w_{k,i})\}_{i=1}^m$$

to train  $m_k(\mathbf{x})$  for  $k = 1, \dots, \tilde{K}$ . For each local linear model, if the dimension  $p$  of  $X$  is large, we can apply a penalized method such as LASSO to select the variables. For the brevity of

presentation below, we introduce unified notations for the original data and the simulated data within each cell:  $\mathbf{v}'_{k,i} = \mathbf{x}'_{k,i}$ ,  $i = 1, \dots, n_k$ ,  $\mathbf{v}'_{k,n_k+i} = \mathbf{v}_{k,i}$ ,  $i = 1, \dots, m$ ,  $w'_{k,i} = y'_{k,i}$ ,  $i = 1, \dots, n_k$ ,  $w'_{k,n_k+i} = w_{k,i}$ ,  $i = 1, \dots, m$ .

We need to pre-choose the hyperparameters  $K_l$ ,  $l = 1, \dots, L$ ,  $m$ , and  $\epsilon$ . In our experiments, we set  $m = 100$  and  $\epsilon \leq 0.1$ , the particular value of which is selected to minimize the cross-validation error for the training data. In general, when the data dimension is high, we tend to use a larger  $m$  to prevent multicollinearity when estimating the local linear models. For simplicity, we set  $K_1 = K_2 = \dots = K_L$  and choose  $K_1$  using cross-validation.

This method of constructing a piecewise linear function from DNN can be viewed as model regularization. The DNN prediction function is piecewise linear when ReLU is used, usually the number of pieces being enormous. Roughly speaking, the clustering algorithm combines these linear pieces into a smaller number of regions, and consequently, the model based on the combined regions reduces the complexity of the original DNN.

### 3.1.2 MLM based on EPICs

The complexity of MLM in Eq. (5) depends primarily on the number of local linear models. If we fit a local linear model for each cell, that is, to treat a cell directly as a EPIC, we usually end up with too many local models. To interpret the model  $\hat{m}(\mathbf{x})$ , we must interpret the EPICs. The task is easier if there are a smaller number of EPICs. Therefore, we further merge the cells by hierarchical clustering to generate the EPICs. The similarity between two cells is defined by the similarity of their corresponding local linear models.

For classification, we define  $d_{s,t}$  using the inverse of F1-score (Rijsbergen, 1979):  $d_{s,t} = \frac{fp + fn}{2 \cdot tp}$ , where

$$\begin{aligned} tp &= |\{\mathbf{v}'_{k,i} | m_s(\mathbf{v}'_{k,i}) = 1 \text{ and } m_t(\mathbf{v}'_{k,i}) = 1, \text{ for } i = 1, \dots, n_k + m, k = s, t\}|, \\ fp &= |\{\mathbf{v}'_{k,i} | m_s(\mathbf{v}'_{k,i}) = 1 \text{ and } m_t(\mathbf{v}'_{k,i}) = 0, \text{ for } i = 1, \dots, n_k + m, k = s, t\}|, \\ fn &= |\{\mathbf{v}'_{k,i} | m_s(\mathbf{v}'_{k,i}) = 0 \text{ and } m_t(\mathbf{v}'_{k,i}) = 1, \text{ for } i = 1, \dots, n_k + m, k = s, t\}|, \end{aligned}$$

and  $|\cdot|$  denotes the cardinality of a set.

Treating  $d_{s,t}$  as a distance between the  $s$ th and  $t$ th cells,  $s, t \in \{1, \dots, \tilde{K}\}$ , we apply hierarchical clustering, specifically, Ward's linkage (Ward Jr, 1963), to merge the cells

$\mathcal{C}_1, \dots, \mathcal{C}_{\tilde{K}}$  into  $\tilde{\mathcal{J}}$  clusters. Ward’s linkage generates a dendrogram by recursively computing the distance between a newly merged cluster  $\nu$  of  $s$  and  $t$ , and another cluster  $v$  using the following equation:

$$d_{\nu,v} = \sqrt{\frac{|v| + |s|}{T} d_{v,s}^2 + \frac{|v| + |t|}{T} d_{v,t}^2 - \frac{|v|}{T} d_{s,t}^2},$$

where  $T = |v| + |s| + |t|$ . Ward’s linkage is known to perform well for clusters with spherical multivariate normal distributions (Kaufman and Rousseeuw, 2009). We provide users the option of different linkage schemes in our python package. We assume  $\tilde{\mathcal{J}}$  is user specified. Let  $\mathcal{J}_j$  be the set of the indices of cells that are merged into the  $j$ th cluster,  $j = 1, \dots, \tilde{\mathcal{J}}$ . For example, if cell  $\mathcal{C}_1$  and  $\mathcal{C}_2$  are merged into the first cluster, then  $\mathcal{J}_1 = \{1, 2\}$ . Clearly  $\bigcup_{j=1}^{\tilde{\mathcal{J}}} \mathcal{J}_j = \{1, \dots, \tilde{K}\}$ . We define the merged cells as an EPIC. For  $j = 1, \dots, \tilde{\mathcal{J}}$ ,

$$\mathcal{P}_j = \bigcup_{k \in \mathcal{J}_j} \mathcal{C}_k. \quad (10)$$

The original data and the simulated data contained in  $\mathcal{P}_j$  form the set  $\bigcup_{k \in \mathcal{J}_j} \{(\mathbf{v}'_{k,i}, w'_{k,i})\}_{i=1}^{n_k+m}$ , based on which we refit a local linear model for the EPIC. With a slight abuse of notation, we still denote each local linear model by  $m(\mathbf{x})$ :

$$\begin{aligned} \hat{m}(\mathbf{x}) &= I_{\mathcal{P}_1}(\mathbf{x})m_1(\mathbf{x}) + \dots + I_{\mathcal{P}_{\tilde{\mathcal{J}}}}(\mathbf{x})m_{\tilde{\mathcal{J}}}(\mathbf{x}), \\ m_j(\mathbf{x}) &= \alpha_j + \mathbf{x}^T \beta_j, \text{ for } j = 1, \dots, \tilde{\mathcal{J}}. \end{aligned} \quad (11)$$

### 3.1.3 Soft-weighted MLM based on EPICs

Although Eq. (11) is used to fit the training data, it is not directly applicable to new test data because which  $\mathcal{P}_j$  a test point belongs to is unknown. We need a classifier for the EPICs:  $\mathcal{P}_1, \dots, \mathcal{P}_{\tilde{\mathcal{J}}}$ . Given how the EPICs are generated in training, one seemingly obvious choice is to compute the DNN inner layer outputs for the test data and associate these outputs to the trained cells and subsequently EPICs. However, this approach hinders us from interpreting the overall MLM because categorization into the EPICs requires complicated mappings of a DNN model, even though the local linear models in MLM are relatively easy to interpret. We thus opt for an easy to interpret classifier for EPICs. Instead, we will

build a classifier for the EPICs using the original independent variables. In Subsection 3.2, we will also develop ways for visualization and rule-based descriptions of EPICs.

From now on, we treat the partition of the training data  $\mathbf{x}_i$ ,  $i = 1, \dots, n$ , into the EPICs  $\mathcal{P}_1, \dots, \mathcal{P}_{\tilde{J}}$  as the “ground truth” labels when discussing classification of EPICs based on the original variables. Denote the labels by  $\zeta_i$ . We have  $\mathbf{x}_i \in \mathcal{P}_{\zeta_i}$ . We first construct a GMM directly from the cells  $\mathcal{C}_1, \dots, \mathcal{C}_{\tilde{K}}$  by fitting a single Gaussian density for each cell. Denote the estimated prior, mean, and covariance matrix of  $\mathcal{C}_k$  by  $\hat{\pi}_k$ ,  $\hat{\mu}_k$ , and  $\hat{\Sigma}_k$ ,  $k = 1, \dots, \tilde{K}$ . The estimated prior  $\hat{\pi}_k$  is simply the empirical frequency, and  $\hat{\mu}_k$  is given by the sample mean. The covariance  $\hat{\Sigma}_k$  can be estimated with different types of structural constraints, *e.g.*, diagonal, spherical, or pooled covariance across components. Recall that  $\mathcal{P}_j$  contains cells  $\mathcal{C}_k$  with  $k \in \mathcal{J}_j$ . Let  $\phi(\cdot)$  be the Gaussian density. Let the estimated prior of  $\mathcal{P}_j$  be  $\tilde{\pi}_j = \sum_{k \in \mathcal{J}_j} \hat{\pi}_k$ . The density of  $X$  given that  $X \in \mathcal{P}_j$  is

$$f_{\mathcal{P}_j}(\mathbf{x}) = \sum_{k \in \mathcal{J}_j} \frac{\hat{\pi}_k}{\tilde{\pi}_j} \cdot \phi(\mathbf{x} \mid \mu_k, \Sigma_k), \quad j = 1, \dots, \tilde{J}. \quad (12)$$

The posterior  $P(X \in \mathcal{P}_j \mid X = \mathbf{x}) \propto \tilde{\pi}_j f_{\mathcal{P}_j}(\mathbf{x})$  is used as the weight for the local linear models in MLM. Let the posterior for  $\mathcal{P}_j$  be  $\gamma_j(\mathbf{x}) = \frac{\tilde{\pi}_j f_{\mathcal{P}_j}(\mathbf{x})}{\sum_{j'=1}^{\tilde{J}} \tilde{\pi}_{j'} f_{\mathcal{P}_{j'}}(\mathbf{x})}$ . Then the soft-weighted MLM is

$$\begin{aligned} \hat{m}(\mathbf{x}) &= \sum_{j=1}^{\tilde{J}} \gamma_j(\mathbf{x}) m_j(\mathbf{x}) \\ m_j(\mathbf{x}) &= \alpha_j + \mathbf{x}^T \beta_j, \quad j = 1, \dots, \tilde{J}. \end{aligned} \quad (13)$$

The above soft-weighted MLM yields smoother transitions between the EPICs. As explained in Subsection 3.1.2, the local linear models  $m_j(\mathbf{x})$  are fitted by least square regression using the original and simulated data in  $\mathcal{P}_j$ . We note that the weights  $\gamma_j(\mathbf{x})$  play a similar role as the kernel functions in Eq.(2). Instead of using Gaussian density functions centered at each data point, we use  $f_{\mathcal{P}_j}(\mathbf{x})$ , the densities of the EPICs. Another important difference is that we use local linear models trained under the co-supervision of a DNN.

## 3.2 Interpretation

To interpret MLM, we assume that the local linear model within each EPIC can be interpreted, or useful insight can be gained for each EPIC based on its local linear model. Although this assumption may not always hold depending on the dataset, our focus here is to interpret the EPICs. If we can understand the EPICs, we can better understand the heterogeneity across the sample space in terms of the relationship between the dependent and independent variables. Our experiments show that the heterogeneity across EPICs can be large, and MLM can achieve considerably higher accuracy than a single linear model.

To help understand EPICs, we develop two approaches, one based on visualization and the other based on descriptive rules. For the first approach, we aim at selecting a small number of variables based on which a EPIC can be well separated from the others. If such a small subset of variables exist, we can visualize the EPIC in low dimensions. The second approach aims at identifying easy-to-describe regions in the sample space that are dominated by one EPIC. We call the first approach the *Low Dimensional Subspace (LDS)* method and the second the *Prominent Region (PR)* method.

### 3.2.1 Interpretation of EPIC by LDS

Recall that we model the density of  $X$  in each EPIC, denoted by  $f_{\mathcal{P}_j}(\mathbf{x})$ ,  $j = 1, \dots, \tilde{J}$ , by a GMM in Eq.(12). The marginal density of  $f_{\mathcal{P}_j}(\mathbf{x})$  on any subset of variables of  $X$  can be readily obtained because the marginal density of any Gaussian component in the mixture is Gaussian. Denote a subset of variable indices by  $\mathbf{s}$ ,  $\mathbf{s} \subseteq \{1, \dots, p\}$ . Denote the subvector of  $X$  specified by  $\mathbf{s}$  by  $X_{[\mathbf{s}]}$  and correspondingly the subvector of a realization  $\mathbf{x}$  by  $\mathbf{x}_{[\mathbf{s}]}$ . Denote the marginal density of  $X_{[\mathbf{s}]}$  by  $f_{\mathcal{P}_j, \mathbf{s}}(\mathbf{x}_{[\mathbf{s}]})$ . Using the marginal densities  $f_{\mathcal{P}_{j'}, \mathbf{s}}(\mathbf{x}_{[\mathbf{s}]})$ ,  $j' = 1, \dots, \tilde{J}$ , and applying MAP, we can classify whether a sample point  $\mathbf{x}_i$ ,  $i = 1, \dots, n$ , belongs to  $\mathcal{P}_j$ . Let  $\hat{\mathbf{q}}_j = (\hat{q}_{j,1}, \dots, \hat{q}_{j,n})$  be the indicator vector for  $\mathbf{x}_i$  being classified to  $\mathcal{P}_j$  based on the marginal densities of  $X_{[\mathbf{s}]}$ :

$$\hat{q}_{j,1} = \begin{cases} 1 & \tilde{\pi}_j f_{\mathcal{P}_j, \mathbf{s}}(\mathbf{x}_{[\mathbf{s}]}) \geq \sum_{j': j' \neq j} \tilde{\pi}_{j'} f_{\mathcal{P}_{j'}, \mathbf{s}}(\mathbf{x}_{[\mathbf{s}]}) \\ 0 & \text{otherwise.} \end{cases}$$

Let  $\mathbf{q}_j = (q_{j,1}, \dots, q_{j,n})$  be the indicator for EPIC  $\mathcal{P}_j$  based on the EPIC labels  $\zeta_i$ , that is,  $q_{j,i} = 1$  if  $\zeta_i = j$ , zero otherwise.

---

Algorithm to find explainable dimensions  $\mathbf{s}_j^*$  for EPIC  $\mathcal{P}_j$

---

- 1 Set hyper-parameter  $0 < \xi < 1$
  - 2  $\mathbf{s}_j^\dagger = \emptyset$ ;  $r_j^\dagger = 0$
  - 3 **Do while**  $r_j^\dagger \leq \xi$  and  $|\mathbf{s}_j^\dagger| < p$
  - 4 Form a collection of sets  $\mathcal{S} = \{\mathbf{s} | \mathbf{s} = \mathbf{s}_j^\dagger \cup s \text{ for any } s \in \{1, \dots, p\} \setminus \mathbf{s}_j^\dagger\}$
  - 5 Compute  $r_{\mathbf{s}} = F_1(\hat{\mathbf{q}}_j, \mathbf{q}_j)$  for all  $\mathbf{s} \in \mathcal{S}$
  - 6 Update  $\mathbf{s}_j^\dagger$ :  $\arg \max_{\mathbf{s} \in \mathcal{S}} r_{\mathbf{s}} \rightarrow \mathbf{s}_j^\dagger$
  - 7 Update  $r_j^\dagger$ :  $r_{\mathbf{s}_j^\dagger} \rightarrow r_j^\dagger$
  - 8 Return  $\mathbf{s}_j^\dagger$  and  $r_j^\dagger$  as  $(\mathbf{s}_j^*, r_{\mathbf{s}_j^*})$  if  $r_j^\dagger > \xi$ . Otherwise, declare failure to find a valid  $\mathbf{s}_j^*$ .
- 

Table 1: The algorithm to find explainable dimensions and the corresponding explainable rate for each EPIC.

We seek for a subset  $\mathbf{s}_j^*$  such that the cardinality  $|\mathbf{s}_j^*|$  is small and  $\hat{\mathbf{q}}_j$  is close to  $\mathbf{q}_j$ , the disparity between them measured by the F1-score between binary classification results. Denote the F1-score by  $F_1(\hat{\mathbf{q}}_j, \mathbf{q}_j)$ , which is larger for better agreement between the binary vectors. In the algorithm presented in Table 1, we find  $\mathbf{s}_j^*$  by step-wise greedy search. In a nutshell, the algorithm adds variables one by one to a set until  $F_1(\hat{\mathbf{q}}_j, \mathbf{q}_j) > \xi$ , where  $0 < \xi < 1$  is a pre-chosen hyper-parameter. It is possible that the search does not yield any valid  $\mathbf{s}_j^*$ , which indicates that the EPICs cannot be accurately classified and thus easily interpreted. We call variables in  $\mathbf{s}_j^*$  *explainable dimensions* for EPIC  $\mathcal{P}_j$  and the F1-score obtained by  $\mathbf{s}_j^*$  *explainable rate*, which is denoted by  $r_{\mathbf{s}_j^*}$ .

### 3.2.2 Interpretation of EPIC by PR

In this subsection, we explore a more explicit way to characterize EPICs. Using a decision tree, we seek prominent regions that are dominated by points from a single EPIC and can be characterized by a few conditions on individual variables. Such descriptions of EPICs are more direct interpretation than visualization in low dimensions. However, the drawback is



that prominent regions are not guaranteed to exist. Given a prominent region, researchers can pose a hypothesis specific to this region rather than for the whole population.

Let  $\mathcal{D}$  be a decision tree trained for binary classification by CART (Breiman et al., 1984). Let the training class labels be  $\mathbf{q} = \{q_i | q_i \in \{0, 1\}, i = 1, \dots, n\}$  and input data matrix  $\mathbf{X} \in \mathbb{R}^{n \times p}$ . CART recursively divides the data by thresholding one variable at each split. A *leaf node*, also called *terminal node*, is a node that is not split; a *pure node* is a node that contains sample points from a single class; and a *fully grown tree* is a tree whose leaf nodes either are pure nodes or contain a single point or multiple identical points. It is assumed that when a node becomes pure, it is not further split. Consider a node denoted by  $e$ . Define the *depth* of  $e$ ,  $d(e)$ , as the number of splits to traverse from the root node to  $e$ . Define the *size* of  $e$ ,  $s(e)$ , as the number of sample points contained in the node. Define the *sample index set*  $u(e)$  as the set of indices of sample points contained in  $e$ , and the *decision path* of  $e$ , denoted by  $\mathcal{H}$ , as the sequence of split conditions to traverse from the root node to  $e$ .  $\mathcal{H}$  consists of  $d(e)$  number of conditions each expressed by thresholding one variable, e.g.,  $x_{.j} > a$  or  $x_{.j} \leq a$ .

Again let  $\mathbf{q}_j = (q_{j,1}, \dots, q_{j,n})$  be the indicator vector for any sample point belonging to EPIC  $\mathcal{P}_j$ . We fit a fully grown decision tree  $\mathcal{D}_j$  based on  $\mathbf{X}$  and the class labels  $\mathbf{q}_j$  (class 1 means belonging to  $\mathcal{P}_j$ ). Let  $\psi$  be a pre-chosen threshold,  $0 < \psi \leq 1$ . We prune off the descendant nodes of  $e$  if its proportion of points in class 1 reaches  $\psi$ :

$$\frac{\sum_{i=1}^n I_{u(e)}(i) q_{j,i}}{s(e)} \geq \psi. \quad (14)$$

We collect all the leaf nodes satisfying Eq.(14) with sizes above a pre-chosen minimum size threshold  $\eta$ . Since leaf nodes are not further split, they are ensured to contain non-overlapping points. Suppose there are  $\kappa_j$  leaf nodes  $\hat{e}_\tau$ ,  $\tau = 1, \dots, \kappa_j$ , such that  $s(\hat{e}_\tau) > \eta$  and  $\hat{e}_\tau$  satisfies Eq.(14). Denote the decision path of  $\hat{e}_\tau$  by  $\mathcal{H}_\tau$ , which contains  $d(\hat{e}_\tau)$  conditions. In  $\mathcal{H}_\tau$ , one variable may be subject to multiple conditions (that is, this variable is chosen multiple times to split the data). We will identify the intersection region of multiple conditions applied to the same variable, which is in general a finite union of intervals. At the end, a decision path  $\mathcal{H}_\tau$  specifies a region of the sample space  $\mathcal{X}$  by the

Cartesian product  $\mathcal{R}_1 \times \mathcal{R}_2 \times \dots \times \mathcal{R}_p$ . If  $\mathcal{R}_{j'} = \mathbb{R}$ , then variable  $X_{j'}$  does not appear in any condition of  $\mathcal{H}_\tau$ . Suppose  $\mathcal{R}_{j'_1}, \dots, \mathcal{R}_{j'_p}$  are proper subsets of  $\mathbb{R}$ . The region given by  $\mathcal{H}_\tau$  can be described by  $\{X_{j'_1} \in \mathcal{R}_{j'_1}\} \cap \{X_{j'_2} \in \mathcal{R}_{j'_2}\} \cap \dots \cap \{X_{j'_p} \in \mathcal{R}_{j'_p}\}$ . We call this form of the region decided by  $\mathcal{H}_\tau$  an *explainable condition*.

When the depth of  $\mathcal{H}_\tau$  is small, the number of conditions  $p'$  in  $\mathcal{H}_\tau$  will be small, and thus the explainable condition is simpler. For the sake of interpretation, simpler explainable conditions are preferred. It is also desirable to have large  $s(\hat{e}_\tau)$ , which means that many data points from  $\mathcal{P}_j$  are covered by this explainable condition. Whether we can find simple explainable conditions that also have high coverage of points depends strongly on the data being analyzed. Hence, the search for explainable conditions can only be viewed as a tool to assist interpretation, but it cannot guarantee that simple interpretations will be generated. In practice, it may suffice to require that class 1 accounts for a sufficiently high percentage of points in  $\hat{e}_\tau$  by setting the purity parameter  $\psi < 1$ . With this relaxation, we can find  $\hat{e}_\tau$ 's with smaller depth.

## 4 Experiments

In this section, we present experimental results of the proposed methods. We demonstrate prediction accuracy of MLM based on cells (MLM-cell) and EPICs (MLM-EPIC), and interpret the formed EPICs by the LDS and PR methods. We use one synthesized dataset and five real datasets of different fields: two The-Cancer-Genome-Atlas (TCGA) datasets (Section 4.2.1), one Parkinson's disease (PD) clinical dataset (Section 4.3) for classification, the bike sharing demand data (Section 4.2.2), and California housing price data (Section 4.2.3) for regression.

We compare the prediction accuracy of MLM-cell and MLM-EPIC with the following methods: nearest class mean (NCM) (k-means is applied to the feature vectors to generate 100 classes) (Webb, 2003), random forest (RF) (Breiman, 2001), support vector machine (SVM) (Cortes and Vapnik, 1995), multilayer perceptron (MLP), linear regression (LR) (specifically, logistic regression for KIRC, SKCM and PD data, and Poisson regression for Bike Sharing data), generalized additive model (GAM) (Hastie and Tibshirani, 1990), Spa-

tial Autoregressive (SAR) (Rey and Anselin, 2007) (for California housing price data that include spatial variables), multivariate adaptive regression splines (MARS) (only for regression tasks, i.e., bike sharing and California housing datasets) (Friedman, 1991), mixture of expert neural networks (MOE) (Jacobs et al., 1991), cluster-weighted modeling(CWM) (Gershenfeld, 1997) (for KIRC, SKCM, and PD data, CWM is not applied because the estimated covariance matrix is singular). The following python packages are used: **pygam** for GAM, **pyearth** for MARS, **pysal** for SAR, and **scikit-learn** for LR, RF, SVM. CWM is provided by the **flexCWM** R package.

### 4.1 EPIC Clusters with Synthesized Data

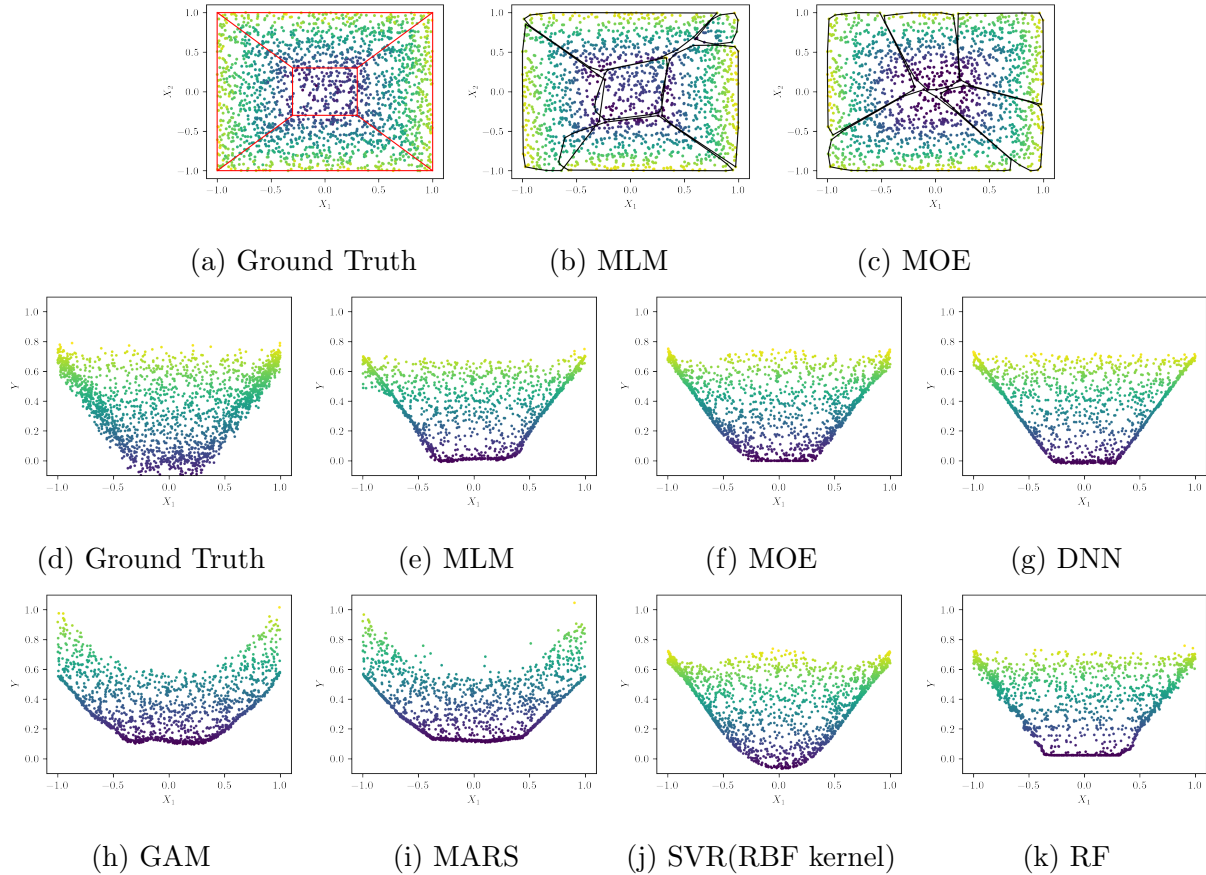


Figure 2: (a-c) Scatter plots on the two explaining variables,  $X_1$  and  $X_2$ , with ground truth groups or convex hulls of estimated clusters for each model. (d-k) Scatter plots on the predicted values,  $\hat{Y}$ , and an explaining variable,  $X_1$ , for each model.

We study a synthesized dataset with given clusters and examine whether MLM can identify those clusters via EPICs. This dataset is generated by a uniform distribution on the square  $[-1, 1] \times [-1, 1]$  (two independent variables  $X_1$  and  $X_2$ ). Two thousand sample points are generated. The dependent variable  $Y$  is computed from the following formula, bearing the shape of a trapezoid bowl plus random noise.

$$y = \begin{cases} X_2 - 0.3 + e & \text{if } X_2 - X_1 > 0 \text{ and } X_1 + X_2 > 0 \\ X_1 - 0.3 + e & \text{if } X_2 - X_1 < 0 \text{ and } X_1 + X_2 > 0 \\ -X_2 - 0.3 + e & \text{if } X_2 - X_1 < 0 \text{ and } X_1 + X_2 < 0 \\ -X_1 - 0.3 + e & \text{if } X_2 - X_1 > 0 \text{ and } X_1 + X_2 < 0 \\ 0 + e & \text{otherwise,} \end{cases} \quad (15)$$

where  $e$  is a Gaussian noise with standard deviation 0.05.

Clearly the best partition of splitting the sample points into local regions, which of each is modeled by a linear regression, is to follow the edges of the trapezoid bowl. Figure 2(b) shows that MLM generates EPICs in better agreement with the geometry of the trapezoid bowl than the local regions of MOE shown in (c). Figures 2(d-k) show how the nonlinear methods predict  $Y$ . Only the neural-network-based predictors, i.e., MLM, MOE, DNN, and the tree-based predictor RF can fit the non-smooth edges of the regression function well, while the other methods generate over-smoothed prediction functions.

## 4.2 MLM for Real-world Problems

For the real-world problems, we use five datasets described below. For all the datasets, we use dummy encoding for nominal variables. That is, we generate  $c - 1$  binary vectors for a nominal variable with  $c$ -classes. The numbers of samples and features in all the datasets are shown in Table 2.

- *KIRC* and *SKCM* are respectively the Cancer Genome Atlas Kidney Renal Clear Cell Carcinoma (TCGA-KIRC) (Akin et al., 2016) and The Cancer Genome Atlas Skin Cutaneous Melanoma (TCGA-SKCM) clinical data. Both are part of a large collection made for studying the connection between cancer phenotypes and genotypes.

Data	KIRC	SKCM	PD	Bike Sharing	Cal Housing
Number of samples	430	388	756	17379	20640
Number of original features	24	30	753	12	8
Continuous / Ordinal	20	25	753	10	8
Nominal	4	5	0	2	0
After dummy encoding	45	73	753	16	8

Table 2: Data descriptions.

The original datasets include tissue images, clinical, biomedical, and genomics data. We only use clinical data in this experiment. For both datasets, the target variable is overall survival (OS) status coded in binary (1 indicating living and 0 deceased). The covariates consist of demographic and clinical variables (24 for KIRC and 30 for SKCM) such as age, gender, race, tumor status, dimension of specimen, and so on. After dummy encoding of categorical variables, KIRC has 45 features and SKCM has 73 features. Both datasets are available in R package `TCGAretriever`.

- *Bike Sharing* (Fanaee-T and Gama, 2014) is an hourly time series data for bike rentals in the Capital Bikeshare system between years 2011 and 2012. The count of total rental bikes is the target variable, and 12 covariates consist of  $\text{Year} \in \{2011, 2012\}$ ,  $\text{Month} \in \{1, \dots, 12\}$ ,  $\text{Hour} \in \{0, \dots, 23\}$ ,  $\text{Holiday} \in \{0, 1\}$ ,  $\text{Weekday} \in \{0, \dots, 6\}$ ,  $\text{Working day} \in \{0, 1\}$ ,  $\text{Temperature} \in (0, 1)$  that is normalized from the original range of  $(-8, 39)$ ,  $\text{Feeling temperature} \in (0, 1)$  that is normalized from  $(-16, 50)$ ,  $\text{Humidity} \in (0, 1)$ ,  $\text{Wind speed} \in (0, 1)$ ,  $\text{Season} \in \{0(\text{winter}), 1(\text{spring}), 2(\text{summer}), 3(\text{fall})\}$ ,  $\text{Weather} \in \{0(\text{clear}), 1(\text{cloudy}), 2(\text{light rain or snow}), 3(\text{heavy rain or snow})\}$ . After dummy encoding of Season and Weather, we get 16 covariates.
- *Cal Housing* (Pace and Barry, 1997) is a dataset for the median house values in California districts, which are published by the 1990 U.S. Census. In this dataset, a geographical block group is an instance/unit. The original covariates consist of 8 features including latitude and longitude which are spatial variables, median income,

median house age (house age), average number of rooms, average number of bedrooms (bedrooms), block population (population), average house occupancy (occupancy), and latitude and longitude. Given the average number of rooms and the average number of bedrooms, we compute the average number of non-bedroom rooms as a new covariate to replace the former.

- *Parkinson’s Disease (PD)* dataset (Sakar et al., 2019) is collected from 188 patients with PD and 64 healthy individuals to study PD detection from the vocal impairments in sustained vowel phonations of patients. The target variable is binary, 1 being PD patients and 0 otherwise. The covariates consist of various clinical information generated by processing the sustained phonation of the vowel ‘a’ from each subject. A total of 753 features are generated by various speech signal processing algorithms, *e.g.*, Time Frequency Features, Mel Frequency Cepstral Coefficients (MFCCs).

#### 4.2.1 TCGA Clinical Data

For both KIRC and SKCM data, MLP is constructed using 2 hidden layers that have respectively 128 and 32 units for KIRC and 256 and 32 units for SKCM, and trained for 10 epochs. Random forest is trained with maximum depth equal to 10. For both MLM-cell and MLM-EPIC, hyper-parameters  $K_l$  is set based on 10-fold cross validation (CV) within training data, and assumed to be equal across the 2 hidden layers. We choose another hyper-parameter  $\tilde{J}$  by comparing the training accuracy obtained at several values. For KIRC, the number of layer  $l$ -cells is set to  $K_l = 4$  according to CV for  $l = 1, 2$ , and 12 cells are formed at the end. These 12 cells are merged into  $\tilde{J} = 10$  EPICs. For SKCM, the number of layer  $l$ -cells is set to  $K_l = 13$  according to CV for  $l = 1, 2$ , and 77 cells are formed at the end. The cells are merged into  $\tilde{J} = 10$  EPICs.

The first two columns of Table 3 show the prediction accuracy for KIRC and SKCM. The area under curve (AUC) score is used as the efficacy metric. For both datasets, MLP achieves the best prediction for the overall survival (OS) status. RF has a higher training AUC, but not for the test data. Among the interpretable models, GAM does not converge for SKCM. LR for KIRC and SKCM, and GAM for KIRC have higher test AUC scores

Model	Data	KIRC (AUC)		SKCM (AUC)		PD (AUC)		Bike Sharing (RMSE)		Cal Housing (RMSE)	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
NCM		.834	.723	.774	.650	.794	.750	122.1	127.7	.795	.815
MOE		.888	.873	.973	.756	.891	.815	47.6	52.8	.684	.687
CWM		-	-	-	-	-	-	38.5	109.2	.369	.623
RF		.983	.869	.997	.688	.997	.794	43.8	52.2	.421	.554
SVM		.843	.856	.764	.667	.756	.728	145.6	148.1	.671	.676
MARS		-	-	-	-	-	-	141.1	140.1	.629	.640
MLP		.974	<b>.907</b>	.987	<b>.777</b>	.975	<b>.833</b>	40.1	<b>47.6</b>	.503	<b>.517</b>
LR		.853	.888	.826	.702	.880	.789	141.1	140.3	.723	.727
SAR		-	-	-	-	-	-	-	-	.655	.652
GA <sup>1</sup> M		.838	.847	-	-	-	-	99.4	101.1	.613	.640
MLM-cell		.904	<b>.891</b>	.948	.728	1.000	<b>.860</b>	52.7	<b>60.9</b>	.560	<b>.570</b>
MLM-EPIC		.902	<b>.891</b>	.861	<b>.742</b>	.975	.851	62.8	66.7	.569	.584

Table 3: Prediction accuracy of regression and classification methods. We partition these methods into two categories: complex (top panel) and interpretable models (bottom panel). For each method, the model parameters are fine tuned. If a method is not applicable to a dataset, the corresponding entry in the table is not listed. The best test accuracy achieved for each dataset and by either category of methods is highlighted in bold font. Higher values of AUC or lower values of RMSE correspond to better performance.

compared to other complex models although their training AUC scores are not better than RF. On the other hand, MLM-cell and MLM-EPIC achieve relatively high AUC scores in both training and testing accuracy, and outperform LR.

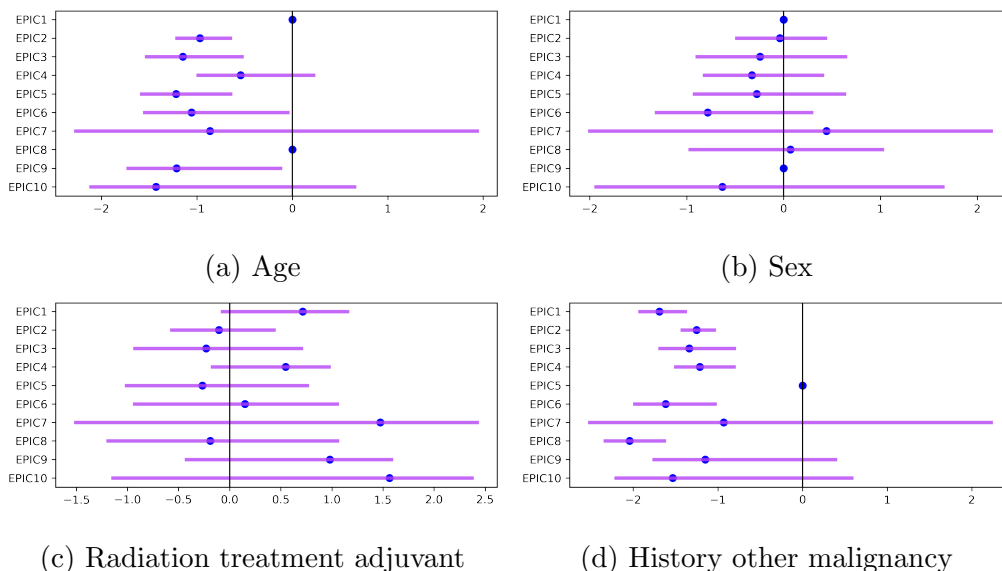


Figure 3: Estimated regression coefficients (blue dots) with "naive" confidence intervals (purple lines) for four selected variables in SKCM. The X-axis indicates the value of each regression coefficient. The linear effects of age, sex, whether the patient has radiation treatment adjuvant, and whether the patient has the history of other malignancy, vary by EPICs.

We interpret MLM-EPIC via local linear models for each EPIC. Figure 3 shows the estimated regression coefficients and the associated "naive" confidence intervals for four selected variables of SKCM. The EPICs are sorted in descending order according to their sizes. Note that the "naive" confidence intervals here are not confidence intervals in a rigorous sense. They are computed under the assumption that the model structure of MLM is correct, which involves data-driven model selection. We provide these intervals as nonstandard quantities to help us understand MLM. A more rigorous study on valid confidence intervals under the MLM setting is an interesting future work. The work of Leeb et al. (2015) provides useful insights into similar problems.

Figure 3(a) demonstrates that age is not associated with the OS status in EPICs 1 and



8, as the coefficients related to age are shrunk to zero by LASSO penalty. However, the irrelevance of age does not hold in other EPICs. Such EPIC-specific results enable us to gain understanding impossible with a single LR model or a DNN. In addition, EPICs 2, 3, 5, 6, and 9 have negative coefficients for age. This result may motivate researchers to hypothesize whether the impact of age differs between different groups of patients. Similar analyses can be made on the other three variables. Moreover, Figure 3(d) demonstrates that the prediction model for subjects belonging to EPIC 5 does not include the variable, “history of other malignancy”, suggesting a heterogeneous effect of this covariate on the OS status.

Now, we investigate how EPICs are formed. We compute explainable conditions for each EPIC to check if they provide any insight for distinguishing the EPICs. Here we interpret the largest EPIC that consists 59 samples.  $\psi$  and  $\eta$  are set to 1 and 4. Table 4 shows explainable conditions for the EPIC 1. More explainable conditions for other EPICs are provided in Supplementary Material. Specifically, around half of patients in EPIC 1 are relatively young patients who have new tumor events after the initial treatment and have less than 306 survival months. Combining this information with the information from Figure 3, an interesting hypothesis can be that for the patients in EPIC 1 under these conditions, age does not impact the OS status. Such insight is not available from MLP or other complex models which are hard to interpret.

#### 4.2.2 Bike Sharing Data

The bike sharing dataset is often used to demonstrate the efficacy of complex machine learning models. For this dataset, RF is set with its maximum depth equal to 10. MLP is constructed with 3 layers with 30 units per layer, trained by 20 epochs. MLM-cell is computed using MLP as its co-supervising neural network. The number of cells per layer is 100, which is chosen by CV. Finally, 1712 cells are formed. MLM-EPIC is constructed by merging 1712 cells into 150 EPICs. For the estimation of the MLM-EPIC parameters, we apply pooled covariance to estimate GMM of each EPIC. With pooled covariance, we essentially assume that the Gaussian components in every EPIC share the same volume

EPIC	Descriptions	Size
1	Age < 29.0, Days to collection $\leq$ 10345, Initial pathology DX year > 1996, New tumor event after initial treatment = 1, Overall survival months $\leq$ 306.2, Retrospective collection = 1, AJCC staging edition $\neq$ 5	11
(59)	Age < 29.0, 8701 < Days to collection > 10345, Initial pathology DX year > 1996, New tumor event after initial treatment = 1, Overall survival months $\leq$ 306.2, Retrospective collection = 1, Submitted tumor DX days > 9737, AJCC staging edition $\neq$ 5	8
	Age < 29.0, Days to collection > 10345, ICD-10 < 5, ICD-O-3 site > 23.5, Overall survival months $\leq$ 306.2, Retrospective collection = 1, AJCC staging edition $\neq$ 5	7

Table 4: Explainable conditions for the largest EPIC for SKCM data. Numbers within the bracket indicate the size of the EPIC in training data.

and shape in their covariance matrices. This restriction has been stated as a possibility to regularize a mixture model (Fraley and Raftery, 1998). In our software package, pooled covariance is provided as an option. Cross-validation on the training samples can be used to decide whether to use this option. For other methods, we apply the default settings of the methods from **pyearth** (for MARS), **pygam** (for GAM), and **scikit-learn** python packages.

First, we examine the prediction accuracy of the methods. Table 3 shows that RF and MLP achieve higher prediction accuracy in comparison with other simpler methods. The RMSEs of MLM-cell and MLM-EPIC are considerably lower than any other method except for RF and MLP. Figure 4 (a) and (b) show the trade-off between the prediction accuracy and the model complexity of MLM (indicated by  $K_l$  for MLM-cell in (a) and by  $\tilde{J}$  for MLM-EPIC in (b)). As the model complexity increases, both training and testing RMSE decreases. We do not observe overfitting as the model complexity of MLM is capped by the underlying neural network. The measure for agreement between MLP and MLM-cell is computed by the RMSE between the predicted values of MLP and MLM-cell respectively. As shown in the figure, the agreement with MLP decreases as the model complexity in-

creases. Figure 4 (c) is the histogram of EPIC sizes obtained from the 150 EPICs of the training data. The histogram shows that many data points belong to small EPICs. For the sample points in small EPICs, the predicted values based on MLM-EPIC are similar to using the nearest neighbor method. For example, in the extreme case, if all EPICs consist of a single point, MLM-EPIC is equivalent to using 1-nearest neighbor method.

Among the other methods, MOE achieves higher accuracy than MLM-cell and MLM-EPIC. However, although MOE exploits local models, the prediction at any point is rarely dominated by a single local model. Specifically, the prediction of MOE is a weighted sum over the predicted values given by its local experts. The weights assigned to the local experts vary with the input points. If these weights flatly spread over the local experts, interpretation based on MOE is still difficult. This scenario is what we have observed with this dataset. For each training sample point, we compute the maximum weight assigned to the local experts of MOE and find that the average of these maximum weights (across the sample points) is 0.65 and at half of the sample points, the maximum weight is below 0.31. In contrast, when MLM-EPIC is used, the average maximum weight assigned to a local model is 0.91, and 73% of the sample points have a maximum weight above 0.9. In Figure 4 (d), we show the kernel density estimate (KDE) for the maximum weights across the training data for MOE and MLM-EPIC respectively. For MLM-EPIC, the KDE has a high peak close to 1.0, while for MOE, the KDE is flat across a wide range.

Figure 5 shows the fitted regression coefficients of MLM-EPIC for the 5 largest EPICs. The values of the coefficients are indicated by color. Depending on which EPIC a sample point belongs to, the effect of a covariate is different when predicting the bike rental count. Specific interpretation of the fitted MLM-EPIC is provided in Supplementary Material.

### 4.2.3 California Housing Prices

California housing data was first introduced to demonstrate the efficacy of the spatial autoregressive (SAR) model (Pace and Barry, 1997), yet it is now often used to test neural network models. Previous works have shown that neural networks perform well with spatial data (Zhu, 2000; Özesmi and Özesmi, 1999), however they cannot be interpreted. We hereby

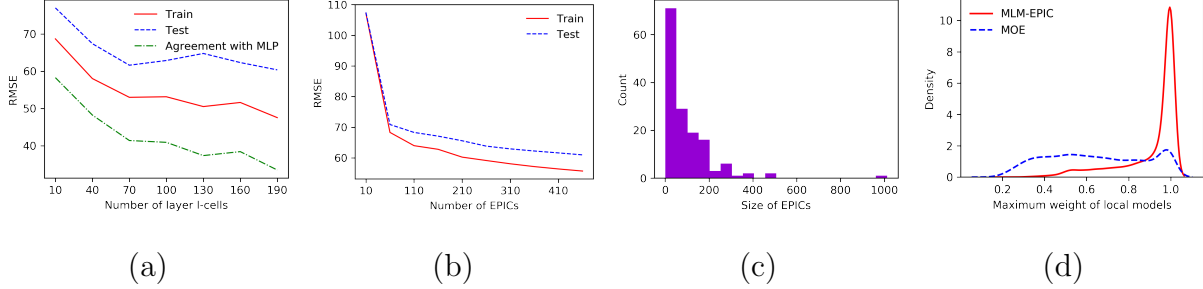


Figure 4: Impact of hyper-parameters,  $K_l$  and  $\tilde{J}$ , on MLM for bike sharing data. (a) RMSE of MLM-cell at different  $K_l$ 's (the number of cells at layer  $l$ ). (b) RMSE of MLM-EPIC at different  $\tilde{J}$  (the number of EPICs). (c) Histogram of the sizes of the 150 EPICs. (d) Kernel density estimate (KDE) plot of the maximum weights assigned to an MLM-EPIC or MOE local expert model, computed at all the training points.

analyze this dataset with MLM for both prediction and interpretation.

RF is fitted with maximum depth 10. MLP contains 3 hidden layers with 30 hidden units per layer, trained by 50 epochs. SAR is fitted with the consideration of spatially lagged covariates except for the longitude and latitude. MLM-cell and MLM-EPICs are fitted based on the MLP model. MLM-cell is constructed with 6 cells per layer, and 64 final cells are formed (some combinations of cells over the 3 layers are empty). MLM-EPIC merged the 64 cells into 30 EPICs. As shown by Table 3, the prediction accuracy of MLM-cell and MLM-EPIC is slightly lower (i.e., slightly larger RMSE) than that of MLP or RF, but they achieve better accuracy than any other interpretable models.

MLM-EPIC provides more useful interpretation than the commonly-used autoregressive models for California housing price data. Autoregressive models are primarily used to describe a time-varying or space-varying process, and the interpretation of the model is based on the model's representation of time or spatial variables. MLM-EPIC can be utilized to describe both time and space-varying effects as it divides samples into EPICs in different time and space. Figure 6 shows the change of regression coefficients for each variable in the longitude and latitude space. The value of the coefficient for the variable in consideration is indicated by color. According to the fitted MLM-EPIC, the impact of the median income on the house values is relatively consistent across the California map. On the other hand,

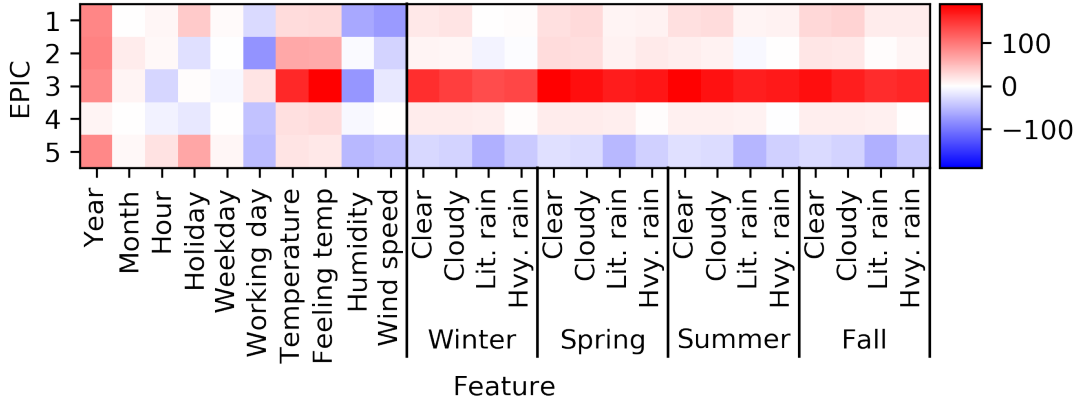


Figure 5: The linear mixture model regression coefficients  $\{\hat{\beta}_j | \mathbf{x} \in \mathcal{P}_j\}_{j=1}^{\tilde{J}}$  for the top 5 largest EPICs for bike sharing data.

house age affects the house values differently depending on which EPIC the sample points belong to. Near the coastal area, old houses tend to be appreciated whereas in the inland area, house age has a negative impact on the house value. This observed pattern may be explained by the fact that houses in the highly populated area were built earlier and are still in high demand. Another interesting pattern in Figure 6 is that the increase of the number of non-bedroom rooms has a negative impact on the house values in city center areas. The log-valued confidence intervals and estimates of the 6 covariates are shown in Figure 7 for the 7 biggest EPICs. These 7 EPICs cover about 70% of the training samples.

For this dataset, we cannot get concise explainable conditions for the EPICs. For example, an explainable condition for EPIC 1 is ‘Median income  $\leq 86.6k$  and House age  $\notin (3.5, 27.5]$  and Bedrooms  $\notin (33.1, 33.6]$  and Population  $\notin (32.1k, 35.7k]$  and Average occupancy  $\notin (1239.6, 1241.8]$  and Latitude  $\notin (40.1, 40.9]$  and Longitude  $\notin (-121.2, -119.5]$ ’. Instead, we use LDS to find explainable dimensions for the EPICs. The explainable dimensions for the 6 biggest EPICs are shown in Figure 8. For example, based on 3 features: Median income, House age, Longitude, EPIC 1 is distinguishable from the other EPICs with accuracy 0.84 as measured by F-1 score. EPIC 1 mostly contains points with income lower than the median, house age older than the median, and located in the middle east part of California. In EPIC 1, older houses tend to increase the house values slightly. Also,

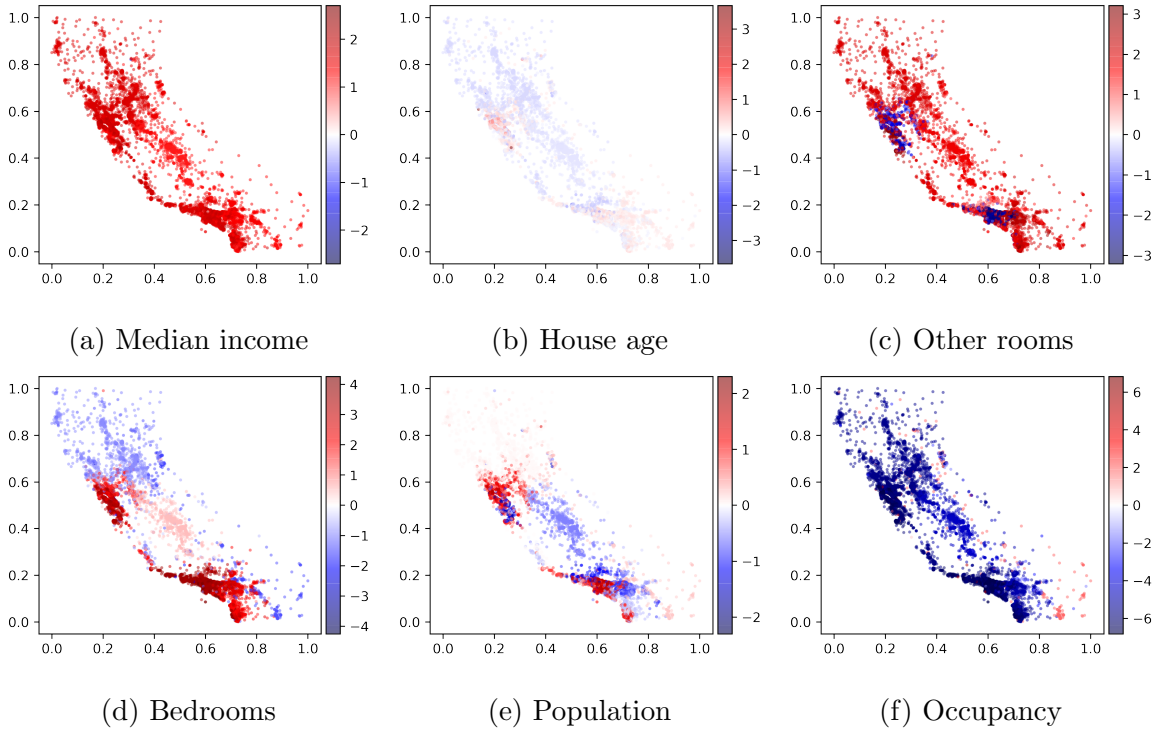


Figure 6: Log-valued regression coefficients of local linear models plotted on longitude (horizontal axis) and latitude (vertical axis) space. The coefficients are transformed by  $sign(x) \cdot \log(|x|)$  for better visualization.

having more bedrooms is appreciated in the house value, while the increase of rooms other than bedrooms tends to decrease the value. EPIC 2 contains mostly points with newer houses in the middle northern part of California (with the explainable rate equal 0.82). For points in EPIC 2, in contrast to EPIC 1, the house age has a negative impact on the house value. Also, the increase in the number of rooms other than bedrooms positively affects the house value, whereas the impact of the number of bedrooms is less clear.

### 4.3 Parkinson’s Disease Detection

Many works have studied Parkinson’s disease (PD) to detect PD from vocal impairments in sustained vowel phonations of PD patients. Due to the limited understanding of the mechanism to detect PD from the complex characteristics of patients’ recorded voices, a dataset often involves a huge number of covariates that are extracted from various speech signal

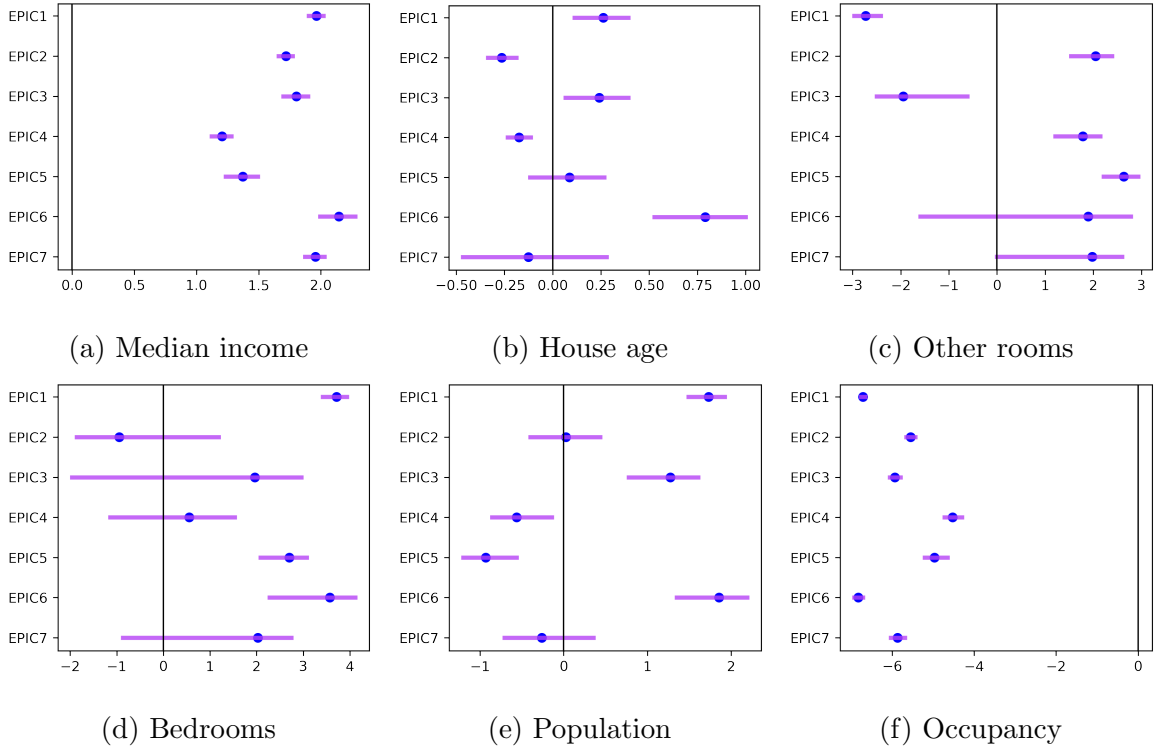


Figure 7: Log-valued “naive” confidence intervals of each variable in the 7 largest EPICs.

processing algorithms. In this subsection, we demonstrate that MLM-cell and MLM-EPIC work well with this high-dimensional dataset. At reduced complexity, the two methods achieved slightly better accuracy than the more complex models.

For this dataset, RF is fitted with maximum depth 10. MLP is constructed with 3 layers with 30 hidden units per layer. MLP is trained for 5 epochs. Other methods follow the default parameter setting from **sci-kit learn** python package. MLM-cell is constructed based on MLP with layer- $l$  cells equal to 5, which is chosen by CV, for  $l = 1, 2, 3$ . 52 cells are formed. We merge the cells into 3 EPICs to build MLM-EPIC. For the local linear models of MLM-EPIC, we applied LASSO penalty with its weight parameter  $\alpha = 0.2$ .

As shown by Table 3, MLM-cell achieves the highest training and testing accuracy. In addition, although MLM-EPIC is only a mixture of 3 local linear models, it has better testing accuracy (lower testing AUC) than MLP. Specifically, MLM-EPIC forms 3 EPICs with sizes 442, 133, and 29 for the training data. The local linear models respectively have 700, 14, 0 non-zero regression coefficients. The last local linear model is a constant function

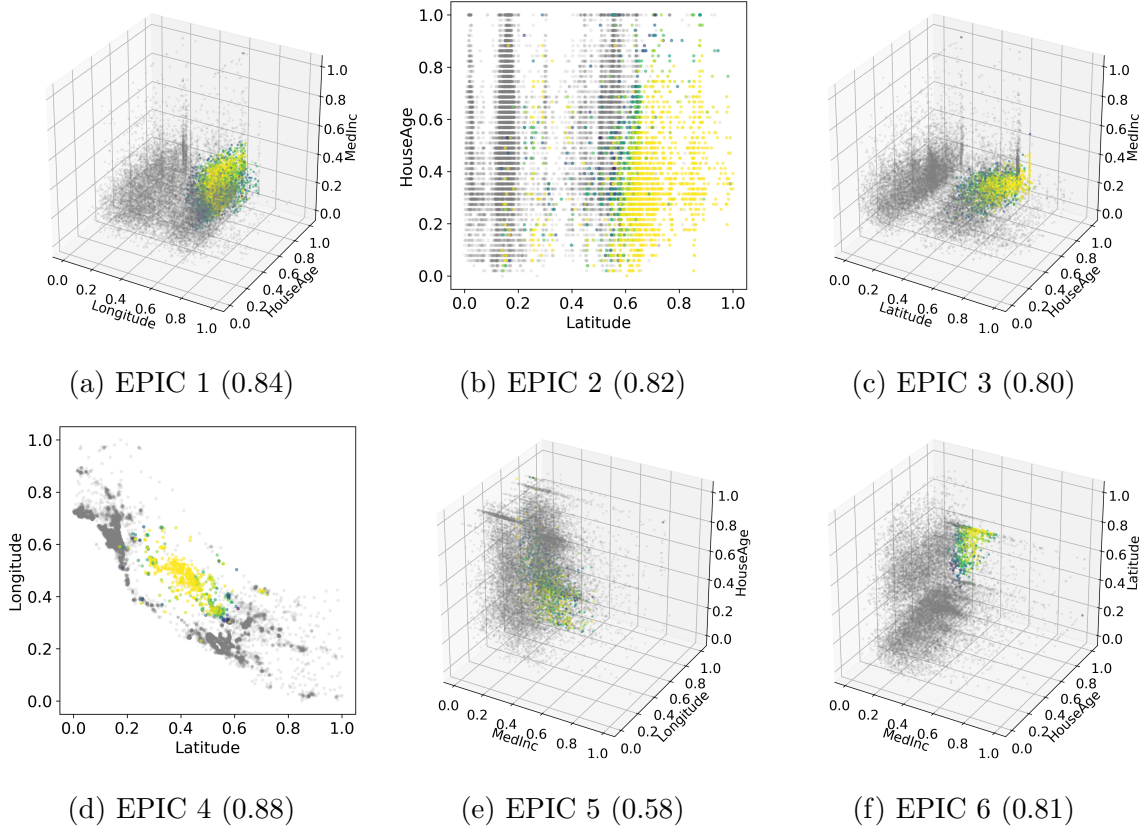


Figure 8: Explainable dimensions for the first 6 biggest EPICs. The values in the parentheses are explainable rates of the explainable dimensions for each EPIC.

which classifies all the points in EPIC 3 as class 0. In this case, the interpretation of EPIC 3 provides an explanation of a local region for class 0.

We compute the explainable dimensions for each EPIC. We set  $\xi = 0.8$ . EPIC 1 is distinguishable with only one variable. In fact, 694 single variables achieve an explainable rate higher than 0.8. Among them, the variable 'tqwt\_TKEO\_std\_dec\_12' achieves the highest explainable rate of 0.87. Whereas, EPIC 2 needs 7 variables to distinguish the EPIC with explainable rate higher than 0.8, and EPIC 3 needs 13 variables.

Explainable conditions provide a different perspective for interpreting the EPICs. With  $\psi = 0.9$  and  $\eta = 50$  for EPIC 1 and 2, and  $\eta = 5$  for EPIC 3, we find the explainable conditions listed in Table 5. In the explainable conditions, 'tqwt\_TKEO\_std\_dec\_12' appears again as an important feature to distinguish EPIC 1. EPIC 2 and 3 are relatively hard to



EPIC	Descriptions	Size
1 (442)	$\text{std\_delta\_delta\_log\_energy} > 0.057$ , $\text{tqwt\_energy\_dec\_18} \leq 0.562$ , $\text{tqwt\_TKEO\_std\_dec\_12} \leq 0.102$ , $\text{tqwt\_medianValue\_dec\_31} \leq 0.454$	381
2 (133)	$\text{mean\_MFCC\_2nd\_coef} \leq 0.397$ , $\text{tqwt\_entropy\_log\_dec\_27} \leq 0.696$ , $\text{tqwt\_entropy\_log\_dec\_35} > 0.368$ , $\text{tqwt\_TKEO\_std\_dec\_17} \leq 0.216$	75
3 (29)	$\text{tqwt\_entropy\_shannon\_dec\_8} \leq 0.007$ , $\text{tqwt\_stdValue\_dec\_19} > 0.373$	9

Table 5: Explainable conditions for the 3 EPICs for Parkinson’s disease data. Numbers in the bracket indicates the size of EPIC in training data.

be distinguished with explainable conditions as the explainable conditions only capture 75 out of 133 points for EPIC 2, and 9 out of 29 points for EPIC 3.

## 5 Conclusions

In this paper, we develop MLM under co-supervision of a trained DNN. Our goal is to estimate interpretable models without compromising performance. The experiments show that MLM achieves higher prediction accuracy than other explainable models. However, for some data sets, the gap between the performance of MLM and DNN is not negligible. Interpretation is intrinsically subjective. For different datasets, different approaches can be more suitable. We have developed a visualization method and a decision rule-based method to help understand the prediction function. In any case, these methods rely heavily on the local linear models constructed in MLM. To explore the aspect of interpretation at a greater depth, we plan to examine applications in more specific contexts. The idea of co-supervision is interesting in its own right, which can be taken as a principle for developing interpretable models. The technical components in MLM may be replaced by other approaches, for example, the regression model used within each EPIC and the method to generate EPICs. These variations can be explored in future works. Also, the statistical inference of MLM coefficients is an interesting future direction. For more rigorous interpretation of MLM, we should validate the model structure using statistical inference methods.

## Acknowledgments

Li and Lin’s research is supported by the National Science Foundation under grant DMS-2013905.

## References

- Agarwal, R., Frosst, N., Zhang, X., Caruana, R., and Hinton, G. E. (2020). Neural additive models: Interpretable machine learning with neural nets. *arXiv preprint arXiv:2004.13912*.
- Akin, O., Elnajjar, P., Heller, M., Jarosz, R., Erickson, B., Kirk, S., and Filippini, J. (2016). Radiology data from the cancer genome atlas kidney renal clear cell carcinoma [tcga-kirc] collection. *The Cancer Imaging Archive*.
- Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.
- Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.
- Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., and Su, J. K. (2019). This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32:8930–8941.
- Cleveland, W. S. and Devlin, S. J. (1988). Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American statistical association*, 83(403):596–610.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

- Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.
- Ebrahimpour, R., Kabir, E., Esteky, H., and Yousefi, M. R. (2008). A mixture of multi-layer perceptron experts network for modeling face/nonface recognition in cortical face processing regions. *Intelligent Automation & Soft Computing*, 14(2):151–162.
- Fanaee-T, H. and Gama, J. (2014). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2):113–127.
- Fraley, C. and Raftery, A. (1998). Mclust: Software for model-based cluster and discriminant analysis. *Department of Statistics, University of Washington: Technical Report*, 342.
- Friedman, J. H. (1991). Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67.
- Gershenfeld, N. (1997). Nonlinear inference and cluster-weighted modeling. *Annals of the New York Academy of Sciences*, 808(1):18–24.
- Guo, M., Zhang, Q., Liao, X., and Zeng, D. D. (2020). An interpretable neural network model through piecewise linear approximation. *arXiv preprint arXiv:2001.07119*.
- Guo, T., Lin, T., and Antulov-Fantulin, N. (2019). Exploring interpretable lstm neural networks over multi-variable data. *arXiv preprint arXiv:1905.12034*.
- Guo, W., Huang, S., Tao, Y., Xing, X., and Lin, L. (2018). Explaining deep learning models – a bayesian non-parametric approach. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.
- Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized additive models*, volume 43. CRC press.

- Ingrassia, S., Minotti, S. C., and Vittadini, G. (2012). Local statistical modeling via a cluster-weighted approach with elliptical distributions. *Journal of classification*, 29(3):363–401.
- Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.
- Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons.
- Kégl, B. (2013). The return of adaboost. mh: multi-class hamming trees. *arXiv preprint arXiv:1312.6086*.
- Kuncheva, L. I. (2014). *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons.
- Leeb, H., Pötscher, B. M., and Ewald, K. (2015). On various confidence intervals post-model-selection. *Statistical Science*, 30(2):216–227.
- Lou, Y., Caruana, R., Gehrke, J., and Hooker, G. (2013). Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631.
- Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774.
- Masoudnia, S., Ebrahimpour, R., and Arani, S. A. A. A. (2012). Combining features of negative correlation learning with mixture of experts in proposed ensemble methods. *Applied Soft Computing*, 12(11):3539–3551.
- Molnar, C. (2020). *Interpretable Machine Learning*. Lulu. com.
- Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142.

- Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml*.
- Özesmi, S. L. and Özesmi, U. (1999). An artificial neural network approach to spatial habitat modelling with interspecific interaction. *Ecological modelling*, 116(1):15–31.
- Pace, R. K. and Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297.
- Rey, S. J. and Anselin, L. (2007). PySAL: A Python Library of Spatial Analytical Methods. *The Review of Regional Studies*, 37(1):5–27.
- Ribeiro, M. T., Singh, S., and Guestrin, C. (2016a). ” why should i trust you?” explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.
- Ribeiro, M. T., Singh, S., and Guestrin, C. (2016b). Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.
- Rijsbergen, C. V. (1979). *Information Retrieval*, volume 2. Butterworths.
- Sakar, C. O., Serbes, G., Gunduz, A., Tunc, H. C., Nizam, H., Sakar, B. E., Tutuncu, M., Aydin, T., Isenkul, M. E., and Apaydin, H. (2019). A comparative analysis of speech signal processing algorithms for parkinson’s disease classification and the use of the tunable q-factor wavelet transform. *Applied Soft Computing*, 74:255–263.
- Schoenberg, I. J. (1973). *Cardinal spline interpolation*. SIAM.
- Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, (1):116–132.
- Tang, B., Heywood, M. I., and Shepherd, M. (2002). Input partitioning to mixture of experts. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN’02 (Cat. No. 02CH37290)*, volume 1, pages 227–232. IEEE.

- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.
- Tsang, M., Cheng, D., and Liu, Y. (2017). Detecting statistical interactions from neural network weights. *arXiv preprint arXiv:1705.04977*.
- Tsang, M., Liu, H., Purushotham, S., Murali, P., and Liu, Y. (2018). Neural interaction transparency (nit): Disentangling learned interactions for improved interpretability. In *Advances in Neural Information Processing Systems*, pages 5804–5813.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- Verikas, A. and Bacauskiene, M. (2002). Feature selection with neural networks. *Pattern recognition letters*, 23(11):1323–1335.
- Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.
- Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372.
- Webb, A. R. (2003). *Statistical pattern recognition*. John Wiley & Sons.
- Zhang, Q., Nian Wu, Y., and Zhu, S.-C. (2018). Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836.
- Zhu, A.-X. (2000). Mapping soil landscape as spatial continua: the neural network approach. *Water Resources Research*, 36(3):663–677.