Maynooth University
National University of Ireland Maynooth

Visualisation Techniques for Interpreting Machine Learning Models

A dissertation submitted for the degree of
Doctor of Philosophy

By:

Alan Inglis

Under the supervision of:

Prof. Andrew C. Parnell

Dr. Catherine Hurley

Department of Mathematics and Statistics
National University of Ireland Maynooth
Ollscoil na hÉireann, Má Nuad

October 2022

Declaration

I hereby declare that I have produced this manuscript without the prohibited assistance of any third parties and without making use of aids other than those specified.

The thesis work was conducted from October 2018 to October 2022 under the supervision of Professor Andrew C. Parnell and Dr. Catherine Hurley in the Hamilton Institute, National University of Ireland Maynooth.

Alan Inglis

Maynooth, Ireland,

October 2022

Sponsor

This work was supported by a Science Foundation Ireland Career Development Award grant number 17/CDA/4695.

Collaborations

Andrew C. Parnell: As my supervisor, Professor Parnell (Maynooth University) supervised and collaborated on the work of all chapters. This includes reviewing and editing all chapters.

Catherine Hurley: As my supervisor, Dr. Hurley (Maynooth University) supervised and collaborated on the work of all chapters. This includes reviewing and editing all chapters as well as collaborating on coding work.

Publications

Of the three chapters in this thesis which cover novel contributions, one has been published and two have been submitted to a pier-reviewed journal. Chapter 2 has been published in the journal *Journal of Computational and Graphical Statistics*. Chapter 3 is under review by the *Journal of Data Science, Statistics and Visualisation*. Chapter 4 is under review by the *The R Journal*. The lead author of the three papers that comprise Chapters 2-4 is Alan Inglis (author of thesis). Andrew Parnell contributed to each chapter by providing reviews and edits to the text. Catherine Hurley contributed to each chapter by providing reviews and edits to the text and collaborated with the coding of the associated R-packages.

Peer-reviewed journal article:

- Inglis, A., Parnell, A. C., & Hurley, C., Visualizing Variable Importance and Variable Interaction Effects in Machine Learning Models. *Journal of Computational and Graphical Statistics* 1-13, (2022). https://doi.org/10.1080/10618600.2021.2007935.

Submitted articles (under review):

- Inglis, A., Parnell, A. C., & Hurley, C., Visualisations for Bayesian Additive Regression Trees. Under review in the *Journal of Data Science, Statistics and Visualisation* (2022). *arXiv* pre-print: https://arxiv.org/abs/2208.08966.

- Inglis, A., Parnell, A. C., & Hurley, C., vivid: An R package for Variable Importance and Variable Interactions Displays for Machine Learning Models. Under review in the *R-Journal* (2022). *arXiv* pre-print: https://arxiv.org/abs/2210.11391.

Contents

Abstract

With the increase of complex Machine Learning (ML) models making decisions in everyday life in a wide range of fields from economics to healthcare, the demand for Interpretable Machine Learning (IML) techniques has grown. One method to broaden the understanding of the behaviour of a fitted ML model is through the use of informative visualisations. Visualisations can aid in interpretation and can provide a more thorough examination into the nature of the predictions generated from an ML model. This is of particular importance when using so-called *black-box* models, such as random forests or Bayesian Additive Regression Trees (BART) models.

In this thesis, various IML approaches are proposed through the use of novel visualisations for displaying different metrics and model summaries which can be used for examining the behaviour of a fitted ML model. First, we present flexible methods for investigating variable importance, interactions, and variable effects by presenting a suite of visualisations that can aid in the interpretation of statistical and ML models through the use of model-specific and agnostic methods. Following from this, motivated in part by the lack of existing visualisation methods and by the rise in popularity of this particular model, we develop novel visualisations for examining BART models that include examining the tree structures and, through the posterior distribution, the uncertainty surrounding predictions. Lastly, we demonstrate and discuss our implementation of the R package software `vivid` (Variable Importance and Variable Interaction Displays) which is used to explore the behaviour of fitted ML models. Here, we focus on key package features and general architectural principles used in `vivid` when designing informative IML visualisations and provide a practical illustration of the package in use.

Acknowledgements

First and foremost, I want to express my gratitude to my parents, family, friends, and partner for their unwavering support and patience during my many years of education. Without their tremendous understanding and encouragement in the past few years, it would have been impossible for me to complete my study.

I want to say thank you to Andrew and Catherine, who served as my supervisors, for their invaluable advice, continuous support, and patience during my PhD study. Their vast expertise and wealth of experience have inspired me throughout my academic career and daily life.

I would also like to thank the wider community of staff and students in both the Hamilton Institute and the Department of Mathematics & Statistics who create an approachable, productive, and enjoyable learning environment.

List of Figures

List of Tables

CHAPTER 1

Introduction

This thesis is concerned with visualising fitted Machine Learning (ML) model behaviour and is motivated by the lack of development of visualisation tools to aid in the interpretation of ML models. The aim is to facilitate the interpretation of complex ML models by constructing a general model-agnostic framework, for visualising different aspects of a model fit, that can be applied to any supervised learning algorithm. Additionally, in response to the growing popularity of this specific model and the lack of available interpretation tools, we develop a framework for evaluating and visualising Bayesian additive regression tree (BART; Chipman et al., 2010) model fits and their associated posterior distribution.

1.1 Visualisations and Interpretable Machine Learning Techniques

In this chapter we provide a non-exhaustive review of some of the more popular Interpretable Machine Learning (IML) approaches used for explaining model behaviour and the most common visualisations associated with them. Having foreknowledge of these techniques will aid the reader in understanding our visualisations in later chapters. As IML is a field that is still expanding and spans a wide range of disciplines, we focus our study on post-hoc interpretability. That

is, the use of interpretation techniques after the model has been fitted. Numerous software packages already exist to examine and visualise model behaviour, such as the `IML` (Molnar et al., 2018), `mlr3viz` (Lang et al., 2023), or `DALEX` (Biecek, 2018) packages for the `R`-language framework, or the `captium` library (Kokhlikyan et al., 2020) for `python`. For a listing of different packages available in the `R`-language, see Chapter 4 of this work, and for a more detailed review and discussion on interpretable machine learning and an examination of predictive models see Molnar et al. (2020a), Molnar (2022), and Biecek and Burzykowski (2021).

To begin, we provide a brief overview of the model development procedure and where in this process post-hoc interpretability techniques can be applied. Figure 1.1 shows a typical life cycle of a model, broken up into six steps and is a variation of common approach called cross-industry standard process for data mining (CRISP DM; Chapman et al., 2000). The process shown in Figure 1.1 can be summarised in the following; once the data has been prepared and cleaned (as described in the first step), it can be explored via simple data analysis to aid in understanding trends or patterns in the data (step two). Outliers are normally detected at this stage, though further outliers may be identified by the modelling process. Correlation between predictors and between predictors and response will also be assessed in stage two. The next step in the cycle is to select, fit, and validate an appropriate model. Step number four is when the model can be evaluated for both model performance and can be visualised using post-hoc interpretability techniques. The methods proposed in this thesis can be applied here to gain a deeper understanding of the relationship between variables and the predicted response and general model behaviour. The final two steps are concerned with deploying the model into a production environment and maintaining/updating the model to ensure it continues to perform as expected.

Numerous tools that automate model development exist in the `R`-language environment, such as `mlr3` (Lang et al., 2019), `H2O` (LeDell et al., 2020), and `tidymodels` (Wickham et al., 2019). As the visualisation methods presented in Chapters 2 and 4 are model agnostic, they can be used in conjunction with any of the aforementioned `R`-packages. For a more detailed discussion on the model development process see Biecek (2019).

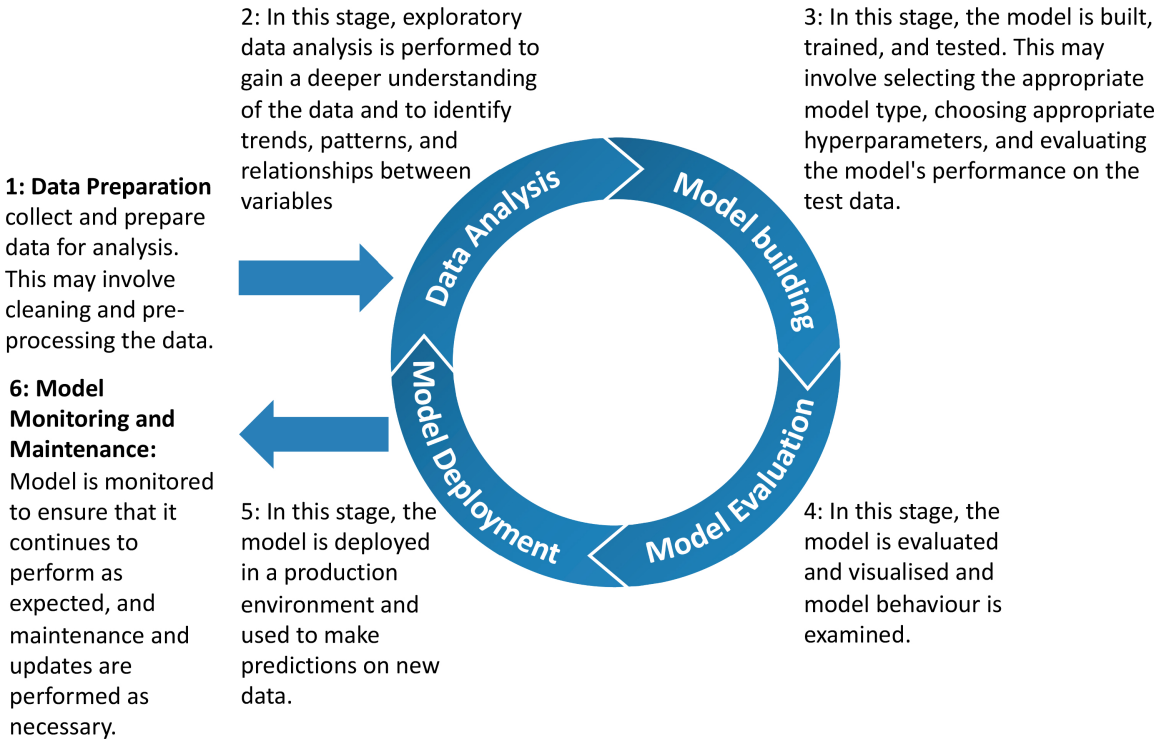


Figure 1.1: Typical model life cycle.

In general, interpretability methods can be divided into two distinct approaches. That is, either model-agnostic methods or model-specific methods. Model-agnostic interpretation methods are post-hoc techniques which can be applied to any ML algorithm. Model-specific methods are those which are tied to a specific ML algorithm and rely on the structure of the model to produce interpretation methods. A similar categorisation of these approaches has been described in Molnar (2022).

In the following examples, we visualise several common model-agnostic and model-specific methods used to explain model behaviour. To illustrate each method, we fit a random forest (Breiman, 2001) to the "air quality" data (Chambers et al., 2018) with Ozone (mean ozone in parts per billion from 1 p.m. to 3 p.m. at Roosevelt Island, New York, USA) as the response.

1.1.1 Model-Agnostic Methods

An advantage of a model-agnostic approach is model choice flexibility, as they can be applied regardless of the model that is used. Another benefit is that model-agnostic approaches can be used to compare multiple models using the same methodology, simplifying interpretation. A brief overview of popular model-agnostic methods is provided in the ensuing subsections where we provide an intuitive description of each approach. However, in Chapter 2, a mathematical definition is supplied.

Permutation Variable Importance: Permutation variable importance is a model-agnostic interpretation method which computes the change in the model's prediction accuracy (based on some error metric) after permuting a variable. Following from the permutation importance introduced by Breiman (2001) for random forests, Fisher et al. (2019) created a model-agnostic permutation method (for an in-depth discussion see Chapter 2). In Figure 1.2 we show a permutation variable importance plots produced from the `iml` package. Here, the root mean square error is used the error metric. The variable Temp is the most important variable for predicting the response with an importance score of 2.25. This means when the variable Temp is permuted, on average, 2.25 is added to the out of sample RMSE for the model.

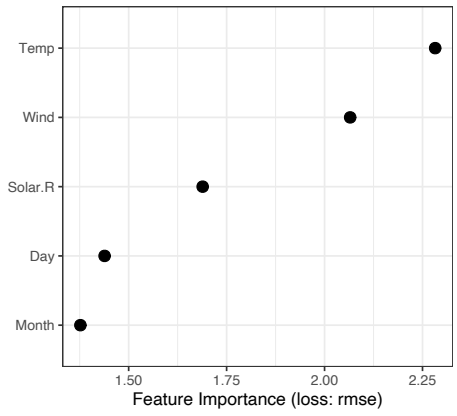


Figure 1.2: Permutation variable importance plot of a random forest fit on the air quality data.

Shapley Values. A popular local method, which has its origins in game theory, is Shapley values (Shapley, 1997). These values offer a solution for how to equally distribute a prize among the participants in a collaborative game. In ML terms, a prediction can be explained by measuring each variable's (or "participant's") contribution to the final prediction by breaking down the output into positive and negative effects. Shapley values can also be used to calculate the importance of a variable by comparing the model predictions with and without the variable included. Commonly Shapley values are visualised as a barplot.

LIME. Local Interpretable Model-agnostic Explanations (LIME; Ribeiro et al., 2016) is a method proposed using local-surrogate models, which are interpretable models, to explain individual predictions of black-box models. Although a local-surrogate model might not completely capture a black-box model's behaviour globally, it can approximate the predictions from a black-box model. The LIME approach works by creating a local linear approximation of the model's decision boundary in the vicinity of the prediction being explained. To get this approximation, LIME generates a set of perturbations around the original input data point and uses these perturbations to generate a new dataset. The model is then trained on this new dataset, and the resulting local decision boundary is used to approximate the original decision boundary. LIME then creates an explanation by identifying the features that have the highest impact on the prediction made by the model. This is done by examining the coefficients assigned to each feature in the local linear approximation of the decision boundary. These coefficients are then used to generate a set of weighted features that can be used to explain the model's decision.

Friedman's H-statistic: Friedman's H-statistic (Friedman and Popescu, 2008) is a model-agnostic method used to measure interactions between variables in a fitted model. The H-statistic is calculated by contrasting the partial dependence for a pair of variables to their marginal effects and is normalised in the range of zero to one. A disadvantage of Friedman's H-statistic is that is requires $O(n^2)$ predicts for each pair of variables, and so can be slow to evaluate. Sampling from the training set will speed up the process, but at the expense of increasing the variance of the partial dependence estimates and the H-statistic. For a more in-

depth discussion on Friedman's H-statistic, see Chapter 2 of this work. In Figure 1.3 we show all pairwise interactions for the variable Temp only. A drawback to this type of visualisation is that to view every pairwise interaction in this way would require a large plot, which can become quickly overcrowded if the number of variables used to train the model is large. A solution to this is presented in Chapter 2 of this work. In Figure 1.3, the variable pair Day:Temp has the greatest interaction, however the interaction measure only has a value of around 0.15. It is challenging to determine when the H-statistic is high enough for us to call an interaction "strong". Friedman and Popescu (2008) propose a test statistic to determine whether the H-statistic differs significantly from zero, however a model-agnostic version has not yet been implemented.

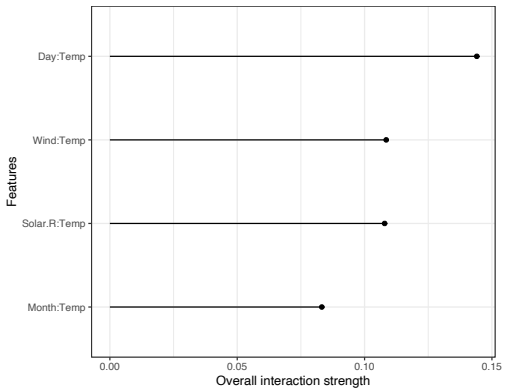


Figure 1.3: Two-way interactions, using Friedman's H-statistic, of all predictor variables with the variable Temp from a random forest fit on the air quality data.

Individual Conditional Expectation (ICE): ICE curves were first introduced by Goldstein et al. (2015) as a model-agnostic method to visualise variable effects. ICE plots show the dependence of the prediction on each individual observation of a variable (and draws a line for each observation). A drawback of ICE plots is that the variable effects can be poorly estimated in the presence of correlated predictors (Apley and Zhu, 2020). Additionally, ICE curves require extrapolation and results that are generated from extrapolating a curve in an area with few or no observations can be deceptive. Another disadvantage is if the data has many observations, many curves will be drawn and can make the plot overcrowded. A

solution to this is to only draw a sample of curves and practical example of this is shown in Chapter 2.

ICE curves can help visualise any linear/non-linear effects and potential interactions among the variables by observing the direction of each curve. An example of ICE curves for the variable Temp (maximum daily temperature in degrees Fahrenheit), obtained using the `iml` package, can be seen in Figure 1.4. Here, we can see the non-linear behaviour that Temp has on the response and as the temperature increases, the predicted ozone level also increases. As the curves are mostly parallel, this indicates no obvious interaction.

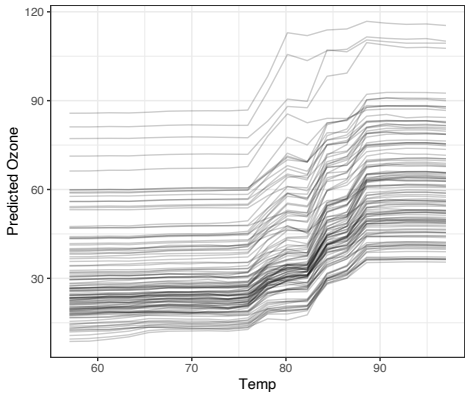
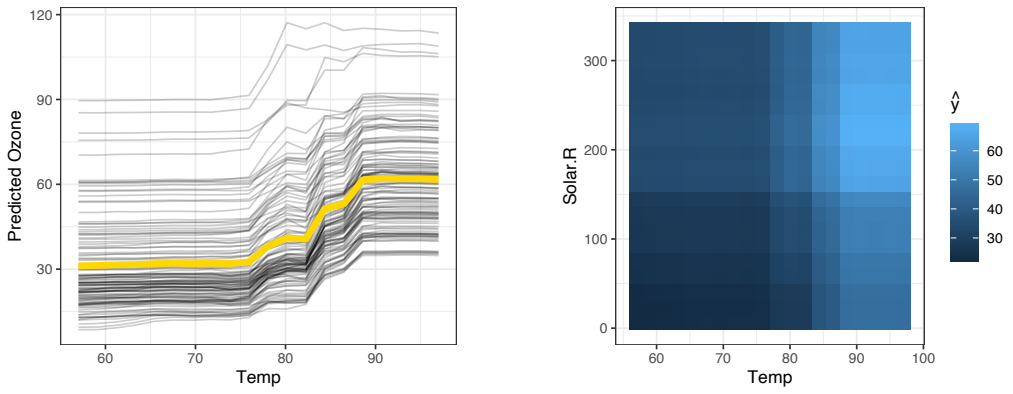


Figure 1.4: ICE curves of a random forest fit on the air quality data for the variable Temp.

Partial Dependence Plots (PDP): PDPs were first introduced by Friedman (2000) as a model-agnostic method to examine the model effects. A PDP visualises the change in the average predicted value when specified feature(s) vary over their marginal distribution. The partial dependence is the average of the aforementioned ICE curves and also suffers the same issues as ICE curves in relation to extrapolation and correlated predictors. For more on PDPs and correlated predictors see Chapter 2. Figure 1.5 (a) shows the partial dependence curve (in yellow) for the variable Temp coupled with the ICE curves (seen in Figure 1.4). Observing the partial dependence curve in (a) we can see that as the temperature increases, the predicted ozone level also increases, with a rise at around 76°Fahrenheit. Bivariate PDPs can also be visualised, as in Figure 1.5 (b) which show the partial

dependence for two variables at once. Here, we show a bivariate PDP for the variables Solar.R (measured solar radiation) and Temp. The $\hat{y}$ value corresponds to the predicted ozone level. In (b), we can see distinct rectangular patches which may be evidence of non-linear behaviour or an interaction between these variables (which could be further investigated using additional IML tools). In general, we observe that for high levels of solar radiation and high temperatures, the predicted ozone level value increases.



(a) (b)

Figure 1.5: Partial dependence plots of a random forest fit on the air quality data. In (a) the partial dependence and ICE curves are shown for the variable Temp. In (b) the bivariate PDP is shown for the variables Solar.R and Temp.

Conditional Visualisations: A local interpretable tool that is useful for exploring and visualising predictions is discussed in Hurley et al. (2022) and comes with an accompanying `R` package called `condvis2`. In Hurley et al. (2022), the authors demonstrate conditional visualisation methods for producing low-dimensional visualisations of models in high-dimensional space through the use of an interactive Shiny app (Chang et al., 2022). `condvis2` can visualise the effect of a predictor conditional on other predictors by taking a section or slice of the surface of the fitted model and results in a curve in two dimensions. Additionally, in `condvis2`, observed data points which are deemed to be near the slice are displayed (see Hurley et al., 2022, for more details).

Figure 1.6 shows a screenshot of the `condvis2` Shiny application in use. In the

top right is a scatterplot of Wind (average wind speed in miles per hour) and Solar.R, with the selected value of 9.7 for Wind and 157 for Solar.R, being chosen by clicking on the plot. Similarly, in the bottom right a scatterplot of the variables Month and Day is shown, with the chosen values of 8 and 14, respectively. The main plot window on the left shows the predicted values of Ozone, while varying Temp and fixing the values of the other variables. The observations shown in this plot are those near the selected values of the other variables and the proximity to this value can be changed via the similarity threshold slider. The main plot of the predicted ozone while varying Temp is in agreement with the previous ICE plot and PDP and shows an increase in the predicted ozone level as the temperature increases.

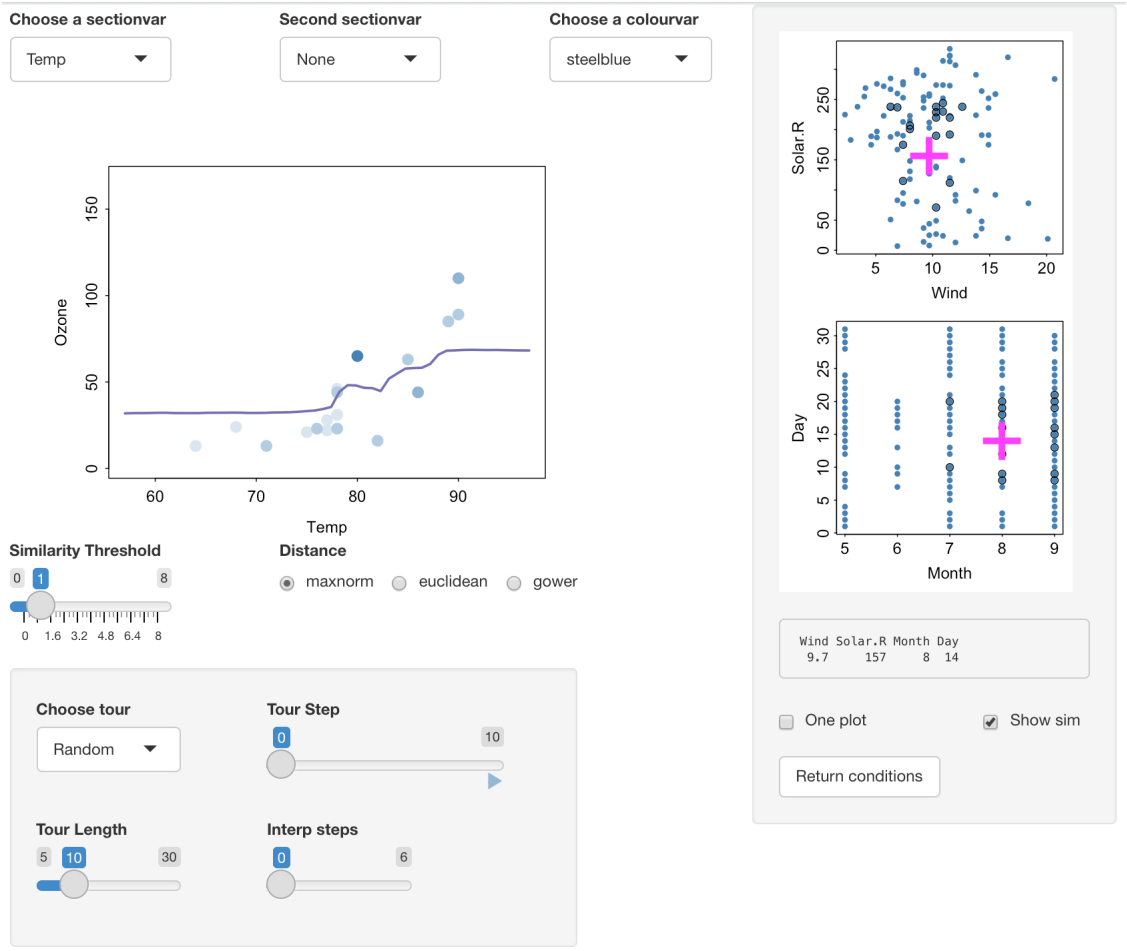


Figure 1.6: A screenshot of `condvis2`'s interactive Shiny application in use.

1.1.2 Model-Specific Methods

Model-specific approaches are those which rely on the inner-workings of the ML algorithm to produce interpretation methods. In the following subsections, we briefly discuss some of the common model-specific methods and the visualisations associated with them.

Model-Specific Variable Importance: Depending on the kind of model, a number of variable importance methods have been proposed. For example, in regression models, utilising specific aspects of a model's structure, such as summary statistics like the standardised regression coefficient or R^2, are commonly used to obtain a variable importance measure. For more complex ML models, like the tree-based algorithms of random forests or Gradient Boosting Machines (GBM) (Friedman, 2000), several model-specific variable importance metrics have been created. For a single decision tree, importance can be determined by how much each split point enhances some performance measure. This can then be measured across all trees to produce a single metric. The increase in node purity and Gini coefficient are commonly used as such a metric. Figure 1.7 below, shows a variable importance plot produced from the `randomForest` package (Liaw and Wiener, 2002) using the increase in node purity as an importance measure. In Figure 1.7 we can see that the variable Temp is the most important and that the results are comparable to those obtained using the model-agnostic approach seen in Figure 1.2.

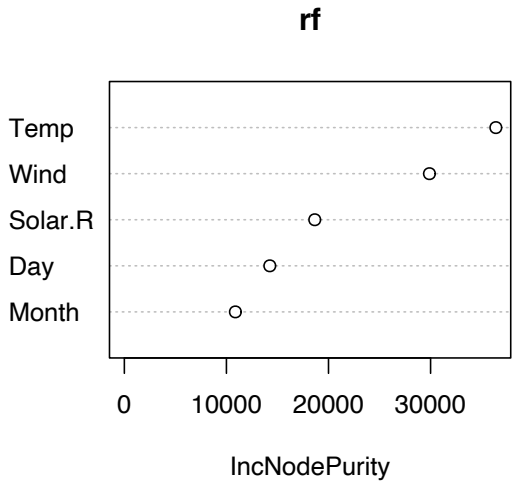


Figure 1.7: Variable importance plot of a random forest fit on the air quality data.

Additional variable importance methods have been produced for random forests, which we mention here but omit visualisations. For example, Strobl et al. (2008) developed conditional variations of the permutation variable importance in a random forest. Furthermore, Loyal et al. (2022) describe dimension reduction forests, which can obtain a local variable importance by using methods from sufficient dimension reduction (Li, 2018).

Variable Interactions: In tree-based ensembles, such as random forest or Bayesian Additive Regression Trees (BART; Chipman et al., 2010), one can use the structure of the trees from the fitted model to assess model-specific interactions. For BART models, the proportion of successive or concurrent splitting rules for each variable pair can be used to determine interactions. For an example see Chapter 3 of this work. For random forest models, the authors of the `randomForestExplainer` package (Paluszynska et al., 2020) propose using a concept known as the mean minimal depth (Ishwaran et al., 2010) to determine interactions, with an example shown in Figure 1.8. The depth of the node that splits on a variable and is the closest to the tree's root is the minimal depth for that variable in the tree. Figure 1.8 shows the conditional mean minimal depth, where one variable is used as the root node and the mean minimal depth for the other variable is calculated.

Consequently, the conditional minimal depth is an non-symmetric measure.

In Figure 1.8 the interactions are shown in descending order, from the most frequent interactions, which are on the left side of the plot and are in a lighter blue, to the least frequent interactions, which are on the right side of the plot and are in a deeper blue. The height of the bar represents the mean conditional minimal depth of each interaction. The minimum of this statistic across all interactions is represented by the horizontal red line. The unconditional mean minimal depth, which is just the mean minimal depth of the second variable in the interaction (as opposed to the conditioning variable), is displayed as a lollipop extended from each bar. From Figure 1.8 the variable pair Month:Wind has the lowest mean minimal depth whilst also occurring the most. However, as previously mentioned, this measure is asymmetric, and we note that the variable pair Wind:Month has a higher mean minimal depth and occurred less frequently.

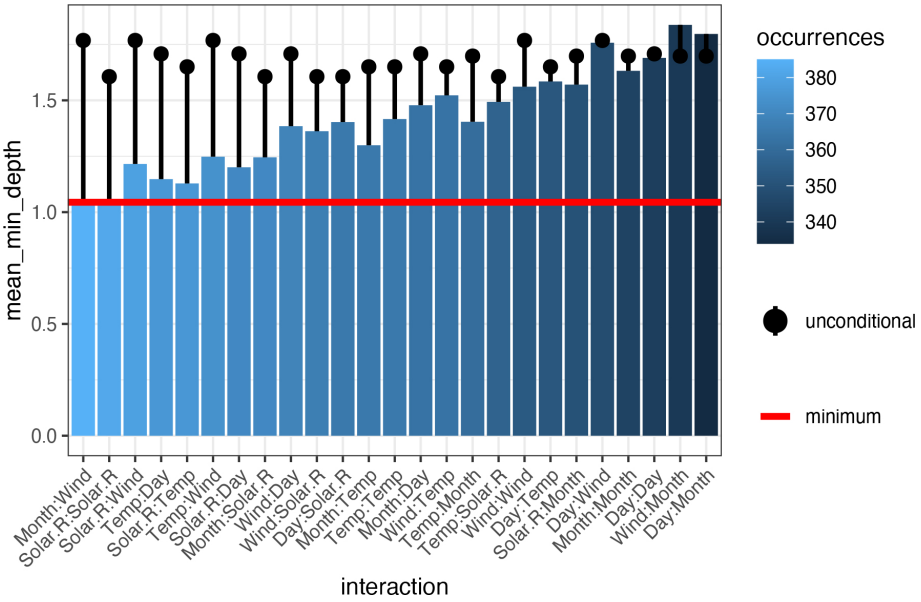


Figure 1.8: Visualisation of variable interactions using the mean minimal depth of a random forest fit on the air quality data.

Neural Networks: Providing transparency to the intricate black-box nature of non-tree-based models, like Neural Networks (NN), can reveal important insights about the behaviour of the model. There are many tools specifically designed for

model-explanation for NN. For example, a common method used for visualising the features that the model is using to make predictions are saliency maps (for example, see Simonyan et al. (2013)). A saliency map is a heatmap that highlights the regions of an input image that have the greatest impact on the model's predictions. Saliency maps can be useful for interpreting the decisions made by a NN, as they allow us to see which regions of an image are most important for making a prediction.

Another technique used to explain deep-learning model behaviour is layer-wise relevance propagation (LRP; Bach et al., 2015). This method is used to understand the importance of individual input features or neurons in the output of a neural network and can be used for a variety of tasks, such as feature visualisation, identifying important features for decision-making, and detecting biases in the neural network. For a summary of recent developments in the field of visualising and interpreting deep-learning models, see Samek et al. (2017).

1.2 Outline of the thesis

The remainder of this thesis is organised as follows. The following chapters are presented in the style of three journal articles, with Chapter 2 being published in a peer-reviewed journal and Chapters 3 and 4 being under review at a peer-reviewed journals. As some of these journals are based in the United States, we switch between American English and British English spelling throughout Chapters 2, 3, and 4.

In Chapter 2 we propose novel visualisations using both model-agnostic and model-specific methods to view various aspects of a model's fit. We present heatmaps and network graphs to view variable importance and variable interactions jointly in an effort to give a more full description of the impact the variables have on the predictions. We create a generalised pairs plot style PDP, called a GPDP, that contains the bivariate, univariate, and scatterplot of the raw variable values to aid in understand the exact nature of variable effects and interactions. Additionally, we present a space-saving PDP utilising Eulerian paths to focus on variable subsets that have the most impact on the response. Through the use of practical examples

and a case study concerning cervical cancer we demonstrate the flexibility and usefulness of our approach.

In Chapter 3 we examine BART model fits using our R package `bartMan`. We present visualisations to assess the behaviour of a BART model such as new tree-based plots that can provide an insight into the tree structure. We construct heatmaps that incorporate value suppressing uncertainty palettes to display the importance and interaction jointly where the posterior uncertainty is visualised using colour scale. To evaluate the performance and stability of a model, we create conventional plots (such as convergence and residual plots) and to investigate potential outliers, we apply multidimensional scaling plots to BART. In this Chapter, we provide practical examples and apply our methods on a case study concerning bike rentals in Seoul, South Korea.

In Chapter 4 we provide a detailed discussion of the implementation of our R package `vivid` (Variable Importance and Variable Interaction Displays). We explain design decisions relating to the package architecture and visualisation interpretability by focusing on package functions and key features. The `vivid` package was developed with the aim of investigating ML models in a way that is intuitive while simultaneously providing useful insights into how variables affect the response. We additionally show a practical example of the package in use as well as providing the code necessary to produce our visualisations.

Finally, in Chapter 5 we conclude the thesis by providing a summary and outlining areas for additional research.

All proposed methods in this thesis were implemented using the R (R Core Team, 2019) software. The R package `vivid` (Variable importance and Variable Interaction Displays) is a companion to Chapters 2 and 4 of this thesis and is available from the Comprehensive R Archive Network[1]. For Chapter 3, the R package `bartMan` (Bayesian additive regression tree Model ANalysis) was created and is available on the author's Github[2]. Additionally, R scripts are also made accessible in order to make the studies and visualisations presented throughout reproducible.

[1] https://cran.r-project.org/web/packages/vivid

[2] https://github.com/AlanInglis/bartMan

These can be found on the author's GitHub. All datasets are freely available, either as files in the aforementioned repositories or within R packages.

CHAPTER 2

Visualizing Variable Importance and Variable Interaction Effects in Machine Learning Models

Variable importance, interaction measures, and partial dependence plots are important summaries in the interpretation of statistical and machine learning models. In this paper we describe new visualization techniques for exploring these model summaries. We construct heatmap and graph-based displays showing variable importance and interaction jointly, which are carefully designed to highlight important aspects of the fit. We describe a new matrix-type layout showing all single and bivariate partial dependence plots, and an alternative layout based on graph Eulerians focusing on key subsets. Our new visualizations are model-agnostic and are applicable to regression and classification supervised learning settings. They enhance interpretation even in situations where the number of variables is large. Our R package `vivid` *(variable importance and variable interaction displays) provides an implementation.*

2.1 Introduction

Visualization is a key tool in understanding statistical and machine learning models. In this paper we present new visualizations to serve two main goals, namely improved model understanding and interpretation. Our new visualizations are based on variable[3] importance and interaction measures, and partial dependence plots. A variable importance value is used to express (in a scalar quantity) the degree to which a variable affects the response value through the chosen model. A variable interaction is a scalar quantity that measures the degree to which two (or more) variables combine to affect the response variable. Variable importance and variable interaction (henceforth VImp and VInt; together VIVI) are widely used in many fields to understand and explain the behaviour of a model. In biology they are used to examine gene-gene interactions (e.g. Wang et al., 2012). In high-energy physics VImp can be an important tool in high dimensional feature selection processes (e.g. Gleyzer and Prosper, 2008). In econometrics they are common tools to evaluate interaction behaviour (e.g. Balli and Sorensen, 2010).

Traditional methods of displaying VImp or VInt use variants of line or bar plots, see for example Molnar (2019). However, in variable importance plots there is relatively little emphasis on displaying how pairs of interacting variables may be important in a model. This can be a hindrance to model interpretation, especially if a variable has low importance but a high interaction strength. The inclusion of interacting terms in a model has been shown to affect the prediction performance (Oh, 2019). However, as shown in Wei et al. (2015b), for high-dimensional models that are governed mainly by interaction effects, the performance of certain types of permutation-based variable importance measures will decrease and thereby produce low values of importance. Consequently, viewing the VInt and VImp together provides a more complete picture of the behaviour of a model fit.

Our new displays present VInt and VImp jointly in a single plot. We allow for seriation so that variables are reordered with those exhibiting high VIVI grouped together. This assists in interpretation and is particularly useful as the number of

[3]We use the term 'variable' throughout to denote the input to a statistical and machine learning model as this seems to be the most common parlance. Other terms commonly used include: feature, predictor, explanatory variable, independent variable, etc

variables becomes large. Furthermore, we make use of filtering, so less influential variables can be removed. For our network displays we use graph clustering to group together interacting variables.

Partial dependence plots (PDP) were introduced by Friedman (2000) to show how the model's predictions are affected by one or two predictors. In addition to the above we propose a new display which shows all pairwise partial dependence plots in a matrix-type layout, with a univariate partial dependence plot on the diagonal, similar to a scatterplot matrix. With this display the analyst can explore, at a glance, how important pairs of variables impact the fit. Once again, careful reordering of the variables facilitates interpretation.

Our final display takes the filtering of all pairwise partial dependence plots a step further. We select only those pairwise partial dependence plots with high VInt, and display an Eulerian path visiting these plots by extending the zigzag display algorithm of Hofert and Oldford (2020). We call this a zen-partial dependence plot (ZPDP).

These new visualizations can be used to explore machine learning models more thoroughly in an easily interpretable way, providing useful insights into variable impact on the fit. This is demonstrated by practical examples. In each plot careful consideration is given to various aspects of the design, including color choices, optimising layouts via seriation, graph clustering, and Euler paths for the ZPDP. Filtering options limit the plots to variables deemed relevant from VImp or VInt scores. Our new displays are appropriate for supervised regression and classification fits, and are model and metric agnostic in that no particular model fit nor importance method is prescribed. The methods described here are implemented in our R package `vivid` (Inglis et al., 2021).

The organisation of the paper is as follows. In Section 2.2 we discuss the concepts of VImp and VInt. Then we describe our new heatmap and network displays of joint variable importance and interaction and demonstrate these on an example. In Section 2.3 we discuss our new layouts for collections of partial dependence plots, either in a matrix format or zig-zag layout and show their application. In Section 2.4 we use our new methodology to explore a machine learning fit from a

larger dataset. Finally in Section 2.5, we offer some concluding discussion.

2.2 Visualizing variable importance and interaction

We begin with a non-exhaustive review of the concepts of VImp and VInt. Though the visualizations we present are agnostic to the measures used to determine these scalar quantities, some degree of understanding is helpful in interpreting the later plots. We then describe our new visualizations and their design principles and provide illustrations.

2.2.1 Measuring variable importance

A VImp is a scalar measure of a variable's influence on the response. Many techniques have been proposed to calculate variable importance, depending on the type of model. The term 'influence' here may encompass changes in the mean response or that of higher order uncertainty. In our work we focus exclusively on changes in the mean. For a wider review of variable importance techniques, and the different goals that a variety of approaches may achieve see Wei et al. (2015a).

Much of the initial work in VImp focused on estimating the partial derivative of the response with respect to one or two input variables (Frey and Patil, 2002). This is a global VImp measure when the model is linear, but perhaps less useful (though still potentially interesting) in non-linear models where it is often defined as a local importance measure. In high dimensional settings these methods can be discretized across a hyper-cube to allow for the identification of, e.g., linearity in a non-linear model (Helton and Davis, 2002). Due to their local behaviour, we do not incorporate them into our visualizations below.

Some VImp measures arise naturally out of a model structure. The most familiar would be those based on summary statistics created from regression models, such as standardized coefficient values, (partial) correlation coefficients, and R^2. Many of these can be extended to non-linear models such as generalized additive models (Wood, 2000), or projection pursuit regression (Friedman and Stuetzle, 1981). R^2 in particular seems useful as a VImp measure, as it can be defined for a wide

variety of statistical models and can be decomposed into main and potentially high order interaction effects, yielding a VInt measure in addition.

Similarly, other model-structure based methods arise out of now standard machine learning techniques. Random forests, for example, involves the use of the Gini coefficient, and the reduction in mean square error, to catalog a variable's influence on the 'purity' of a model output (Breiman, 2001). This can naturally be seen as a VImp measure. Others have extended these approaches to introduce conditional and permutation VImp statistics which aim to reduce the bias that may occur due to variable collinearity (for example, see Hothorn et al., 2006).

Conditional variants of permutation variable importance were proposed by Strobl et al. (2008) for a random forest. This method examines splits of the trees in a random forest and permutes the variables within these subgroups (see Section 2.2.2 for more details). Whereas Strobl et al. (2008) relied on the splitting of trees to determine the subgroups, a model-agnostic approach was introduced by Molnar et al. (2020b) that builds the subgroups explicitly from the conditional distribution of the variables. In tree-based models such as CART and random forests, Ishwaran et al. (2010) proposed a VImp called minimal depth, which is the proximity of a variable to the root node, averaged across all trees.

Permutation importance was introduced by Breiman (2001) and is measured by calculating the change in the model's predictive performance after a variable has been permuted. The algorithm works by initially recording the model's predictive performance, then, for each variable, randomly permuting a variable and re-calculating the predictive performance on the new dataset. The variable importance score is taken to be the difference between the baseline model's performance and the permuted model's performance when a single feature value is randomly shuffled. A similar agnostic permutation concept was developed by Fisher et al. (2019). This method permutes inputs to the overall model instead of permuting the inputs to each individual ensemble member. In situations where no embedded variable importance is available, a model-agnostic approach such as permutation importance is a useful tool.

In theory any of the above global importance measures could be used in our visu-

alizations. However, providing code for each would be a daunting task. Instead we take a pragmatic approach and use the associated VImp measure with the model that we are fitting. In cases where there is no such obvious method, we use the Fisher et al. (2019) agnostic permutation approach discussed above to measure VImp.[4]

2.2.2 Measuring variable interaction

Measuring variable interaction in a machine learning model can be considerably harder than estimating marginal importance. Even the definition of the term 'interaction' is disputed (Boulesteix et al., 2015). We focus here on bivariate interaction only, though higher order interactions may certainly be present in many situations. Friedman and Popescu (2008) state that a function $f(\boldsymbol{x})$ exhibits an interaction between two of its variables x_k and x_l if the difference in the value of a function $f(\boldsymbol{x})$ as a result of changing the value of x_k depends on the value of x_l. That is, the effect of one independent variable on the response depends on the values of a second independent variable. Often, an interaction is taken to mean a simple multiplication of two (continuous) variables (e.g. Berrington de González and Cox, 2007), though in machine learning models much more complex relationships can exist. We follow the definition of Friedman and Popescu (2008) by considering an interaction to be estimated from the difference between joint and marginal partial dependence; a full mathematical definition is given below. Even this definition should not be used without care, as in the case of highly correlated or potentially confounding variables.

In tree-based models such as CART and random forests, much focus has been on measuring interactions via the structure of trees (e.g. Ishwaran et al., 2010; Deng, 2019). If two variables are used as splits on the same branch, this might initially appear like a measure of interaction. However, this does not separate out the interaction from potential marginal effects. The problem is partially overcome by permuting the variables (individually for a VImp, jointly for VInt), to assess the effect on prediction performance. The resulting VInt measure is known as pairwise prediction permutation importance (Wright et al., 2016).

[4] In our implementation, any available VImp may be used.

For models that are not tree-based, or when a model-agnostic measure is required, a variety of other methods can be used. Many of these are based on the idea of partial dependence (Friedman, 2000). The partial dependence measures the change in the average predicted value as specified feature(s) vary over their marginal distribution.

The partial dependence of the model fit function g on predictor variables S (where S is a subset of the p predictor variables) is estimated as:

$$f_S(\boldsymbol{x}_S) = \frac{1}{n}\sum_{i=1}^{n} g(\boldsymbol{x}_S, \boldsymbol{x}_{C_i}) \tag{2.1}$$

where C denotes predictors other than those in S, $\{\boldsymbol{x}_{C_1}, \boldsymbol{x}_{C_2}, ..., \boldsymbol{x}_{C_n}\}$ are the values of $\boldsymbol{x}_C$ occurring in the training set of n observations, and $g()$ gives the predictions from the machine learning model. For one or two variables, the partial dependence functions $f_S(\boldsymbol{x}_S)$ are plotted (a so-called PDP) to display the marginal fits.

Friedman's H-statistic or H-index (Friedman and Popescu, 2008) is a VInt measure created from the partial dependence by comparing the partial dependence for a pair of variables to their marginal effects. Squaring and scaling gives a value in the range $(0, 1)$:

$$H_{jk}^2 = \frac{\sum_{i=1}^{n}[f_{jk}(x_{ij}, x_{ik}) - f_j(x_{ij}) - f_k(x_{ik})]^2}{\sum_{i=1}^{n} f_{jk}^2(x_{ij}, x_{ik})} \tag{2.2}$$

where $f_j(x_j)$ and $f_k(x_k)$ are the partial dependence functions of the single variables and $f_{jk}(x_j, x_k)$ is the two-way partial dependence function of both variables, where all partial dependence functions are mean-centered.

The H-statistic requires $O(n^2)$ predicts for each pair of variables, and so can be slow to evaluate. Sampling from the training set will reduce the time, though at a cost of increasing the variance of the partial dependence estimates and the H-statistic.

When the denominator in Equation 2.2 is small, the partial dependence function for variables j and k is flat, and small fluctuations in the numerator can yield spuriously high H-values. Biased partial dependence curves will also lead to inflated H. This occurs in some machine learning approaches which exhibit regression to the mean in their one-way partial dependencies. Furthermore biased partial dependence curves are a particular problem in the presence of correlated predictors.

These issues with the H-statistic seem to be not widely known by practitioners (though see Apley and Zhu, 2020), and we provide a short illustration of these problems in the appendix.

In our visualizations throughout this paper, we use the square-root of the average un-normalized (numerator only) version of Friedman's H^2 for calculating pairwise interactions:

$$H_{jk} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}[f_{jk}(x_{ij}, x_{ik}) - f_j(x_{ij}) - f_k(x_{ik})]^2} \tag{2.3}$$

This reduces the identification of spurious interactions and provides results that are on the same scale as the response (for regression). It does not, however, remove the possibility that some large H-values arise from correlated predictor variables.

We follow the convention of Hastie et al. (2009) by using the logit scale for both the partial dependence and in calculation of the H-statistic when fitting a classification model with a binary response. If the response is multi-categorical a near-logit is used, defined as:

$$g_k(x) = \log[p_k(x)] - \frac{1}{K}\sum_{k=1}^{K}\log[p_k(x)] \tag{2.4}$$

where $k = 1, 2, ..., K$ and $p_k(x)$ is the predicted probability of the k-th class. PDPs of $g_k(x)$ from Equation 2.4 can reveal the dependence of the log-odds for the k-th class on different subsets of the input variables.

Alternatives to the H-statistic have been suggested, which could be used in place of the the H-statistic in our visualizations. Hooker (2004) uses a functional ANOVA construction to decompose the prediction function into variable interactions and main effects. Greenwell and Boehmke (2020) suggested a partial dependence-based feature interaction which uses the variance of the partial dependence function as a measure of importance of one variable conditional on different fixed points of another.

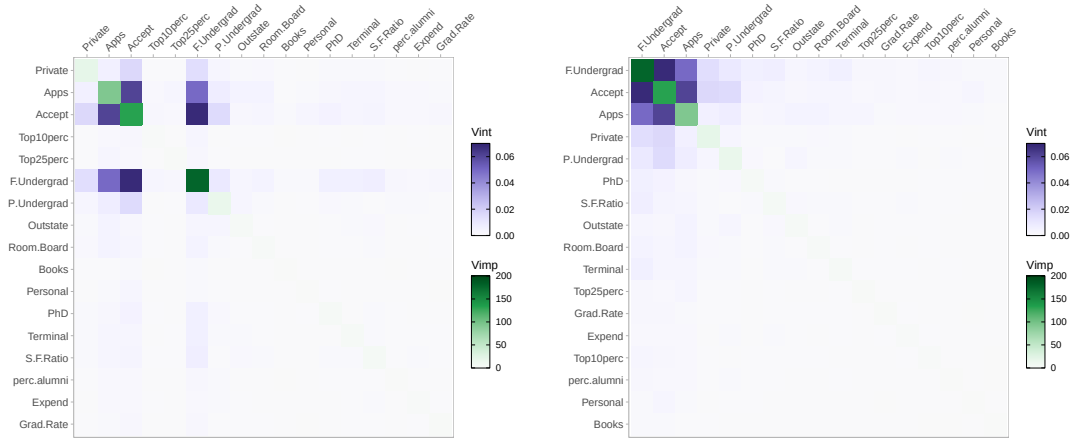
2.2.3 Heatmap visualization with seriation

Traditionally, variable importance and interaction are displayed separately, with variable interaction itself spread over multiple plots, one for each variable. We

direct the reader to Chapter 8 of Molnar (2019) for examples. We propose a new heatmap display showing VImp on the diagonal and VInt on the upper and lower diagonals. The benefit of such a display is that one can see which variables are important as individual predictors and at the same time see which pairs of variables jointly impact on the response. It also facilitates easy comparison of multiple model fits, which is far less straightforward with separate VImp and VInt displays.

We illustrate the heatmap using a random forest fit to a college applications data set (American Statistical Association, 1995), with Enroll (i.e., the number of new students enrolled) as the response. The data was gathered from 777 colleges across the U.S. and contains 18 variables ranging from economic factors (such as room and board and book costs) to the number of applications received and accepted. As some of the variables are skewed they are log-transformed prior to building the model. The data was split 70-30 into training and test sets. A value of $R^2 = 0.96$ was obtained for the test set. All plots were made from the training set. See the supplementary materials for a description of the data and transformations.

Figure 2.1 shows our heatmap with two different orderings. Figure 2.1(a) has the



(a) Unsorted ordering

(b) Leaf sort algorithm

Figure 2.1: Heatmap from random forest of college application data. In (a) variables are in original order. In (b), the heatmap is re-ordered using leaf sort. In (b) we can see three important and mutually interacting variables, F.Undergrad, Accept and Apps.

variables in their original order, while Figure 2.1(b) uses the leaf-sorting algorithm (described below). The purple color scale used on the off-diagonal shows the Friedman's H-statistic values (un-normalized) with deeper purple indicating a higher VInt. Similarly the green color scale on the diagonal represents the level of VImp, here measured using an embedded approach supplied by the random forest (in this case, the increase in node purity). We use colorblind-friendly, single-hued sequential color palettes from Zeileis et al. (2020) going from low to high luminance in both cases, designed to draw attention to high VInt/VImp variables. From the improved ordering in Figure 2.1(b), there are three clearly important and potentially interacting variables, F.Undergrad (the number of full-time undergraduate students), Accept (the number of applicants accepted), and Apps (the number of applications received), with F.Undergrad having the largest VImp when predicting Enroll.

Many authors have investigated the benefits of re-ordering (also known as seriation) for graphical displays, see for example Hurley (2004), Hahsler et al. (2008) and Earle and Hurley (2015). The benefits of reordering the variables in Figure 2.1(b) are clear. The right-hand plot lends itself to easy interpretation whereas the left-hand plot does not.

Most seriation algorithms start with a matrix of dissimilarities or similarities between objects and produce an ordering where similar objects are nearby in the sequence. Our goal here is a little different. As well as placing mutually interacting variables nearby in the sequence, we would like to bring important variables or pairs of variables to the start of the sequence so that the most relevant portion of the heatmap will be in the top-left corner.

We use the leaf sort seriation algorithm from Earle and Hurley (2015). This uses hierarchical clustering followed by a sorting step. Let v_i be a measure of variable importance and s_{ij} be the interaction measure between variables i and j. Treating the matrix of interactions as a similarity matrix, we first construct a hierarchical clustering. This produces a dendrogram, resulting in a variable ordering where high-interacting variables are nearby. Using this ordering in a heatmap generally brings high interactions close to the diagonal, but ignores our

goal of placing important variables early in the sequence. For the sorting step we calculate for each variable a combined measure of its importance and contribution to the interactions, defining these scores as:

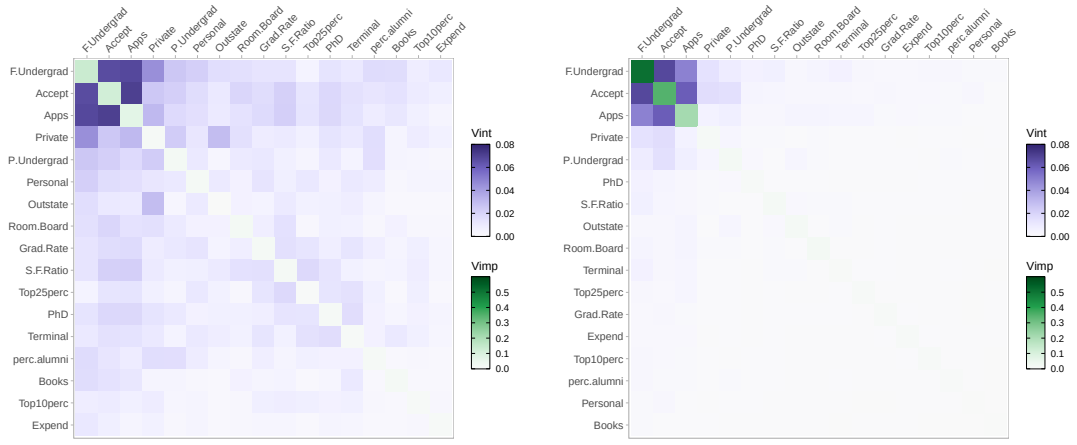
$$w_i = \lambda_1 v_i + \lambda_2 \max_{j \neq i} s_{ij}.$$

Here λ_1 and λ_2 are scaling parameters to account for the fact that variable importance and interaction are not measured in the same units. Reasonable choices of λ_1 and λ_2 rescale importance and interaction to, say, unit range or unit standard deviation. We use unit range by default. As there are many possible dendrogram orderings consistent with a hierarchical clustering of the matrix of interactions, the sorting step re-orders the dendrogram leaves so that the weights w_i are generally decreasing.

Sorting the variables in this way will achieve our goals of placing high-interacting pairs of variables nearby in the sequence, while simultaneously pulling predictors with high importance and interaction to the top-left of the heatmap, leaving less relevant predictors to the bottom-right. Setting $\lambda_2 = 0$ or $\lambda_1 = 0$ produces plots which sort by descending VImp or max VInt respectively. For all future heatmap plots, we use the sorting strategy discussed above to optimize the arrangement of variables. After using seriation to re-order the heatmap variables, filtering can be applied to limit the display to the most important or interacting variables; this strategy is especially useful when there are large numbers of predictors.

The heatmap display can be further used to compare different model fits. In Figure 2.2 we compare the random forest to a k-nearest neighbours (kNN) fit. In the left panel of Figure 2.2 we have a heatmap of a kNN fit (with $k = 7$ neighbours considered), while the right panel shows the random forest heatmap. To make a direct comparison of the heatmaps, we swap the embedded VImp measures that are available from a random forest fit and instead measure importance with an agnostic permutation approach that allows direct comparison of both the kNN and random forest models. Furthermore, we set both heatmaps to use the same color scale for the VImp and VInt values.

We see in Figure 2.2 that both the random forest and kNN fit identify F.Undergrad as the most important variable for predicting the number of students enrolled. The



(a) kNN fit

(b) Random forest fit

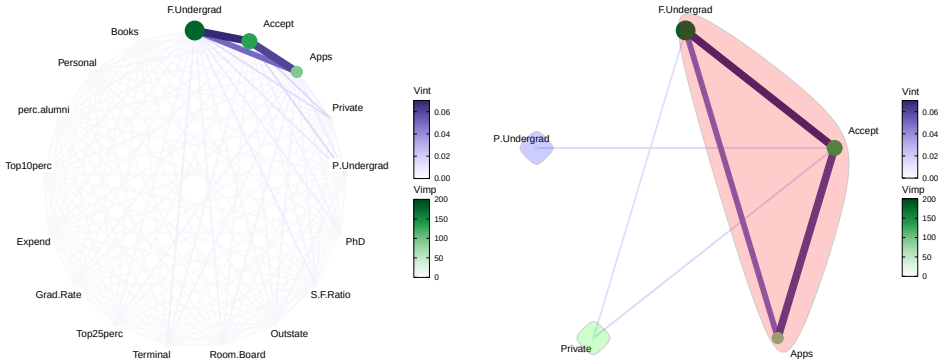
Figure 2.2: A comparison of a kNN and random forest fit on the college application data. Both fits identify F.Undergrad as the most important variable as well as having similar mutual interactions between F.Undergrad, Accept and Apps. The kNN fit identifies many more moderate interactions between variables, especially concerning the variable Private

top three variables are identical in both models, though the VImp values are much smaller in general across the kNN fit (e.g. the measured VImp for F.Undergrad for the kNN and random forest fits are 0.16 and 0.6 respectively). Both fits show mutual interactions between F.Undergrad, Accept and Apps. However, the kNN fit also suggests a moderate interaction between Private (i.e., whether the university was public or private) and F.Undergrad, which appears somewhat lower in the random forest fit. As Private has a relatively low VImp in both model fits, a simple VImp screening could miss its relevance to the fit. We note though, that this kNN-random forest comparison is for the sake of illustration only, as in this instance the kNN fits poorly by comparison with the random forest, having a test mean square error (MSE) over three times bigger.

2.2.4 Network visualization

As our second offering for displaying VIVI, we propose a network plot that shares similar benefits to the heatmap display but differs from it by giving a visual representation of the magnitude of the importance and interaction values not only via color but also by the size of the nodes and edges in a graph. In this plot, each

variable is represented by a node and each pairwise interaction is represented by a connecting edge. See Figure 2.3(a) for an example. The color scales were chosen



(a) (b)

Figure 2.3: Network plot of a random forest fit on the college application data. Three mutually interacting and important variables can be seen, namely F.Undergrad, Accept and Apps. In (a) all of the variables are displayed. In (b) the network plot has been filtered to display pairs of variables with high VInt and clustered to highlight variables with mutually high VInt.

to match that used in the heatmap, with node size and color luminance increasing with variable importance. Similarly, edge width and color reflects the strength of the VInt. By default we choose a radial layout to display the variables (although this can be changed according to preference) and use the same seriation of variables as the heatmap, with the variables of high importance and high interaction strength placed in a clock-wise arrangement starting at the top. The benefit of such a display is that one can quickly decipher the magnitude of the importance and interactions of the variables as well as seeing which variables both individually and jointly impact on the response.

In Figure 2.3(a) we again use the random forest fit of the college application data, using the same VImp and VInt measures as in Figure 2.1. In the network plot the strong mutual interactions between F.Undergrad, Accept and Apps and are represented by thick, intensely purple lines. F.Undergrad is identified as the most important single predictor and is represented by a large, intensely green node. For settings with large number of predictors, it will be useful to filter the display to focus on high VIVI variables. An additional step groups or clusters the

variables according to VImp or VInt values. For example, Figure 2.3(b) shows a network plot, filtered to display pairs of variables with high VInt and clustered to show groups with mutually similar VInt. Here it is clear that the cluster colored pink contains the variables with the largest VInt scores. In this example we use hierarchical clustering, but in our implementation, the graph clustering methods provided by the package `igraph` (Csardi and Nepusz, 2006) are directly available.

2.3 Visualizing partial dependence and individual conditional expectation

We introduce new variants of partial dependence and individual conditional expectation plots in two different layouts. With these plots, we can further investigate predictor effects singly and pairwise, especially for those predictors deemed important in our VIVI plots. Additionally, our new plots combine displays of variable pairs, thus highlighting the presence of strong correlations where VInt measures may mislead. Conventionally, partial dependence plots are shown singly or in linear layouts, see Section 8.1 of Molnar (2019) for examples. By comparison, our new displays are more compact, richer, and benefit from seriation.

2.3.1 Individual conditional expectation curves

Goldstein et al. (2015) described individual conditional expectation (ICE) curves, which are closely related to partial dependence plots (PDPs). While a PDP shows the average partial relationship between the response and one or two features S, ICE plots display a collection of curves, each showing the estimated relationship between the response and the feature S, at an observed value of other features. Recalling Equation (1), the ICE curves consist of $g(\boldsymbol{x}_S, \boldsymbol{x}_{C_i})$ *versus* $\boldsymbol{x}_S$, $i = 1, 2, \ldots, n$, while the PDP curve is their average $f_S(\boldsymbol{x}_S)$. If the ICE curves follow a similar pattern then the PDP is a useful overall summary, but if the pattern varies, then the feature effect is not homogeneous.

2.3.2 Generalized partial dependence pairs plot with ICE curves

We propose a generalized pairs partial dependence plot (GPDP) with one-way partial dependence and ICE curves with a superimposed partial dependence curve on the diagonal, the bivariate partial dependence on the upper diagonal and scatter plots of raw variable values on the lower diagonal, all of which are colored by the predicted values $\hat{y}$. Figure 2.4 provides an example. With the generalized pairs

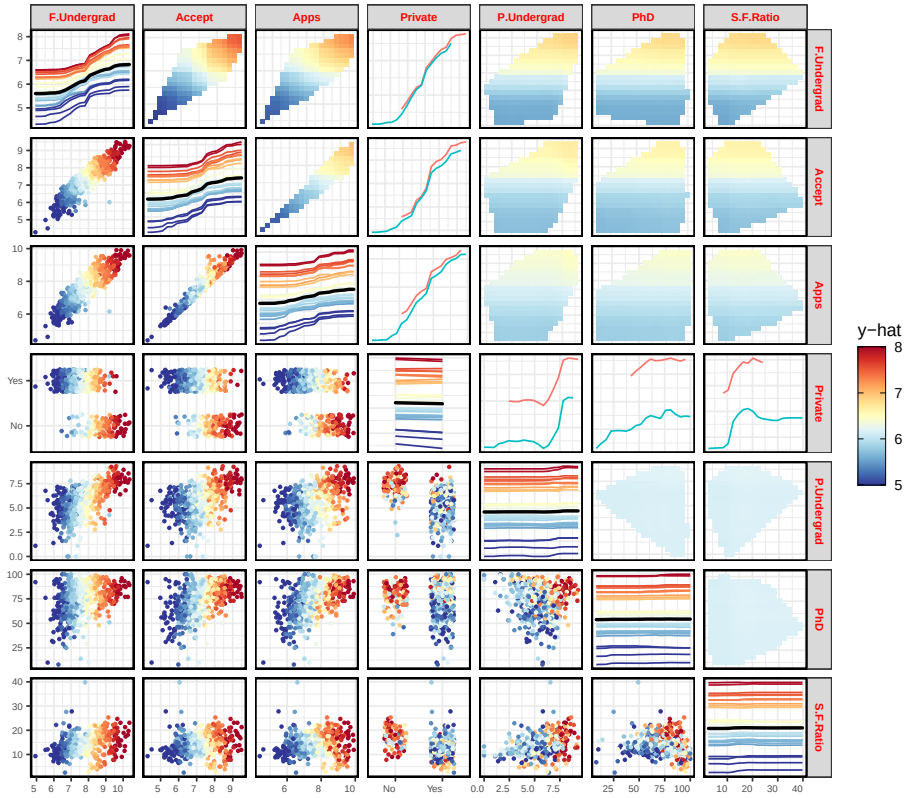


Figure 2.4: GPDP of a random forest fit on the college data showing the seven most influential variables. From the changing one and two-way partial dependence, we can see that F.Undergrad, Accept and Apps have some impact on the response. However, as they are highly correlated and have similar increasing marginal effects, the potential interactions identified by the H-statistic are likely to be spurious.

plot, an analyst can quickly identify which variables singly or jointly impact on the fit. We use a diverging palette so deviations from the average response are emphasized. Here, high values of $\hat{y}$ are shown in dark red and low values are

shown in dark blue. Mid-range values are shown in yellow. To avoid interpreting the PDPs where there are no data (and hence potentially spurious H-statistics), we mask out extrapolated areas by plotting the convex hull. For maximum resolution of the bivariate PDPs, the range of the collection of PDP surfaces dictates the limits of the color map. As predictions for individual observations and ice curves are likely to fall beyond these limits, colors are assigned using the closest value in the color map limits.

The ordering of the variables matches that of our heatmap and network plots. The GPDP differs from the previous plots by showing us the distribution of the explanatory variables (lower-diagonal), the exact nature of any linear/non-linear effects through the use of ICE curves (diagonal), and the average behaviour of the interactions through the use of two-way partial dependence (upper-diagonals). For the ICE curves we have limited the graphic to display a maximum of 30 randomly sampled curves by default, to allow individual ICE curves to be seen. As with the other visualizations, our GPDP can handle both categorical responses and predictors.

Figure 2.4 shows an example of a GPDP of the college applications data. In the interest of space, we pre-filter this plot to show the seven most influential variables. The bivariate PDPs show the response surface over the convex hull of each variable pair. The lower diagonal plots indicate that F.Undergrad:Accept, F.Undergrad:Apps and especially Accept:Apps are highly correlated with similar increasing marginal effects on the diagonal, suggesting that the the high H-values between these variables are likely to be spurious. This is verified by the bivariate linear PDPs for these variables. As Private is a factor with two levels (i.e., yes or no), the partial dependence for each factor level is shown in the upper-diagonal (with yes in red and no in blue). The remaining variables would appear to have little effect either singly or jointly on the response. This can be seen from the flat one-way PDP and ICE curves on the diagonal and the flat two-way PDPs.

2.3.3 Partial dependence zenplot

Our final display uses the methods of Hofert and Oldford (2020) to show selected panels of the all-pairs PDP in a space-saving layout, which we call a zen-partial

dependence plot (ZPDP). Zenplots (zigzag expanded navigation plots) were designed for showing pairwise plots of high-dimensional data in a zigzag layout. The motivation for zenplots is that they focus on interesting 2D displays, and they permit examination of high-dimensional data. Indeed, Hofert and Oldford (2018) present an example where they successfully explore pairwise dependence of 465 variables via 164 zenplots. Here we propose to adapt zenplots for bivariate partial dependence plots.

To describe the construction, consider a network plot showing VImp/VInt such as that in Figure 2.3(a). Then delete edges with VInt below a threshold, leaving a graph such as that in Figure 2.3(b). We wish to build partial dependence plots showing pairs of variables with high VInt, that is, visiting each of the edges in our thresholded graph. For a connected graph, the greedy Eulerian path algorithm of Hurley and Oldford (2011) visits each edge at least once, starting from the highest weighted edge and moving through edges giving preference to the highest-weight available edge. If the graph is not even, some edges may be visited more than once, or additional edges are visited. If the graph is not connected, we form sequences for the connected sub-graphs, which are optionally joined into a single sequence.

Zenplots use the zigzag display algorithm of Hofert and Oldford (2020) and allow for the display of high-dimensional data by alternating plot axes in a zigzag-like pattern where adjacent axes share the same variable. We adapt this concept replacing bivariate data plots with bivariate partial dependence plots. As interpretation issues may arise when the distribution of some of the variables is highly skewed, we display a rug plot on each axis to show the distribution of the data. For ease of viewing, the rug plots are a single color and use alpha blending to highlight the distribution. As with our GPDP, there is an option to mask areas where the partial dependence has been extrapolated. The resulting plot displays the most important interacting variables in as small a space as is possible, vastly reducing the number of plots that would be required for interpretation compared with a default matrix scatter plot of PDPs.

In Figure 2.5, we show a ZPDP for the random forest fit to the college applications data. The ZPDP shows the bivariate PDPs corresponding to each of the edges

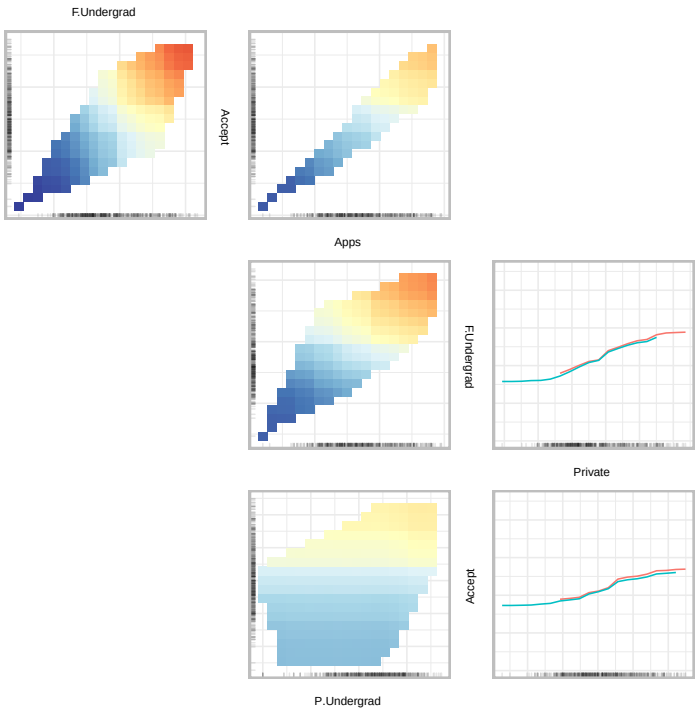


Figure 2.5: ZPDP of a random forest fit on the college data. We can see that the predicted value for the number of students enrolled increases with each of the variables.

of Figure 2.3b. The sequence of plots is obtained from an Eulerian visiting the edges starting with the highest-weight edge, here that is between F.Undergrad and Accept, and following available edges in order of preference by weight thereafter. The resulting Eulerian is F.Undergrad, Accept, Apps, F.Undergrad, Private, Accept, P.Undergrad. The plots shown correspond to a subset of those in Figure 2.4, limited to the more interesting high-interaction pairs. This more compact display helps focus the reader's attention where it is needed, especially as the plots are approximately ordered by decreasing H-index. The variables Private and P.Undergrad show little evidence of marginal importance, and notwithstanding the relatively large H-values, there is not much evidence of interaction with other predictors.

2.4 Case study: cervical cancer risk classification

Cervical cancer remains one of the most prevalent forms of cancer in women globally, ranking fourth in the global cancer incidence in women (Bray et al., 2018). The link between cervical cancer and sexually transmitted diseases (STDs) has been well established. The long-term use of hormonal oral contraceptives is associated with increased risk (Smith et al., 2003). Furthermore, having multiple children has been shown to increase risk (Lukac et al., 2018), particularly in women previously infected with HPV.

Here we examine and create visualizations for data concerning cervical cancer risk factors (Fernandes et al., 2017). Based on the previous studies, we would expect our visualizations to align with prior identification of important variables, with the addition of gaining new information about how the variables interact. The data is comprised of historical medical records (such as a patient's STD history, oral contraceptive or intrauterine device [IUD] use) and personal information (such as age and sexual activity). Due to the personal nature of the questions asked for the collection of the data, several patients decided not to answer some of the questions, particularly those concerning STDs. The data has been previously studied (for example see Alsmariy et al., 2020). The full dataset contains 36 variables with 858 observations and uses Biopsy (Healthy or Cancer) as the response.

For this case study, we use a subset of the variables (see supplementary materials for a listing). Preliminary exploration of the data shows that many variables are highly skewed and contain zero values; in this case we use a $\log(x+1)$ transformation. The data is split 70-30 into training and test sets. We fit a classification gradient boosting machine (GBM) model (Friedman, 2000) to the training data, with Biopsy as the response. The accuracy on the test set was measured to be 0.93, and the area under the curve (AUC) was 0.73. All plots were made using the training data, with all PDPs and the H-statistic measured on the logit scale.

Figure 2.6 displays a heatmap of the GBM fit on the cervical cancer risk data, using a permutation VImp method. Reading from the top-left, the first seven

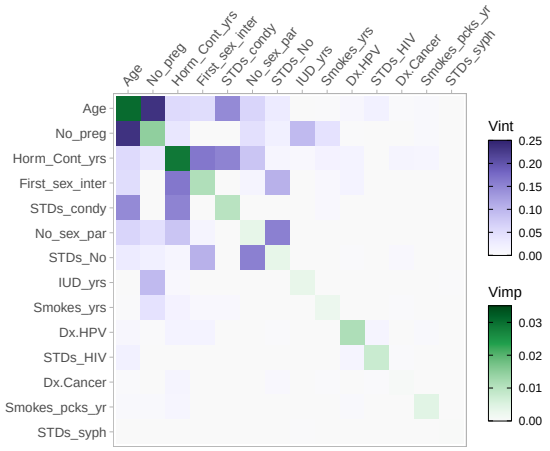


Figure 2.6: Heatmap of a GBM fit on the cervical cancer data. The first seven variables have the highest VIVI scores. Age and No_preg have the strongest interaction.

variables have the highest VIVI scores. Overall, Age has the highest importance followed closely by Horm_Cont_yrs (the number of years a patient has taken hormonal contraceptives). This is in agreement with the studies mentioned above. Age also shares the strongest interaction with No_preg (number of pregnancies), which has a medium Vimp but is highly important in terms of its interaction. We can see multiple interactions throughout the top seven variables. Of note is the interaction between STDs_No (number of STDs a patient has previously had) and No_sex_par (number of sexual partners). Both of these variables share a strong interaction but have low VImps and they may have been mistakenly eliminated from a model were VImp scores to be used as the sole variable selection metric.

We further explore the impact of the top five variables from Figure 2.6 on cancer classification in the GPDP plot of Figure 2.7. To compare response groups, the ICE plots on the diagonal show 25 instances sampled from each of the Cancer and Health groups. The ICE curves are colored according to the predicted log-odds of cancer for that instance. As there is only one red curve, the predicted model accords most observations low cancer probabilities, even for those known to have cancer. The solid black lines on the diagonal of Figure 2.7 show single variable PDPs. The PDP curve for Age, the single most important predictor, has a mostly decreasing log-odds trend up to an age of 43 (≈ 3.75 on the log scale), with a steep incline thereafter. But we can see from the Age scatterplots there are few cases

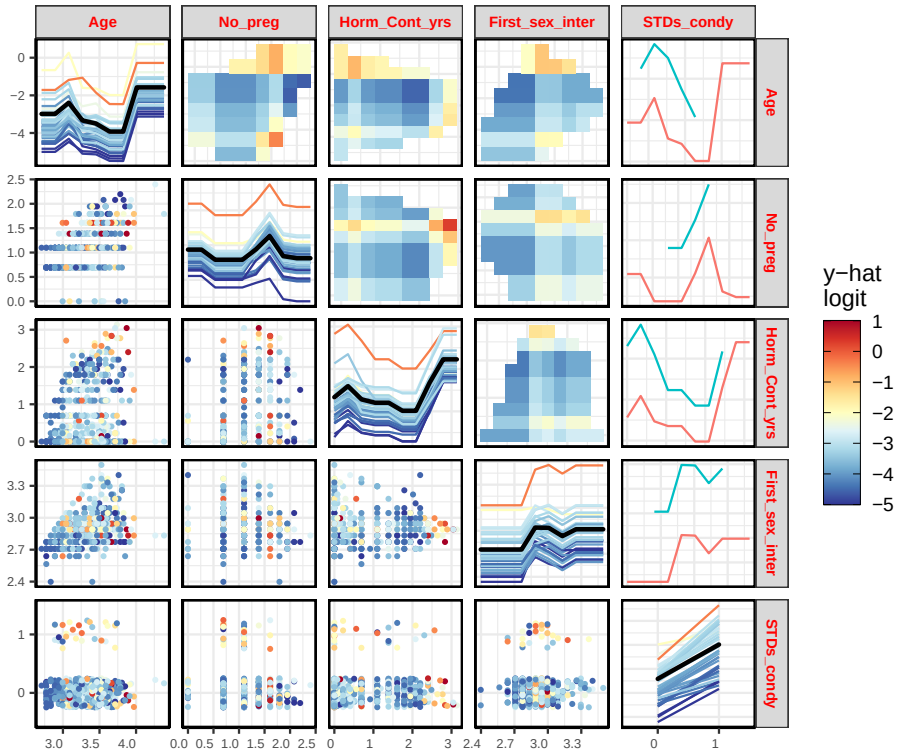


Figure 2.7: GPDP of a GBM fit on the cervical cancer data. The presence of the STD condylomatosis (STDS_condy) increases the risk of cervical cancer. Risk increases substantially at higher ages and with prolonged use of hormonal contraceptives.

with ages beyond 43, so the pattern in this area is not supported by much data. The pattern for the Horm_Cont_yrs PDP is similar to that for Age, where log-odds of cervical cancer increases rapidly beyond eight years. In this case though, there are quite a few observations in this region supporting this finding.

According to Figure 2.6, the predictors No_preg and Age have the strongest interaction. The bivariate PDP plot for No_preg:Age indicates the form of this interaction. A high number of pregnancies is associated with low cancer probability for middle age groups, but is associated with a higher cancer probability for older and, interestingly, younger patients. Note that in the plots with one numeric and one categorical variable, such as the plot for STD_condy (STDs: condylomatosis) and Age, the numeric variable is always drawn on the x-axis, notwithstanding the label is on the y-axis. This is to allow the plot to be more easily read. In this plot, the bivariate PDP is the same as two PDPs for each level

of STDs_condy (where the green curve is for STDs_condy = 1). Although this pair has a relatively high VInt score (as seen in Figure 2.6), there does not appear to be an interaction present in the bivariate PDP, as the difference between the two curves does not vary with age.

To focus just on predictors with high pairwise interaction scores, we turn to a network plot. Figure 2.8 displays a network plot of the GBM fit to the cervical

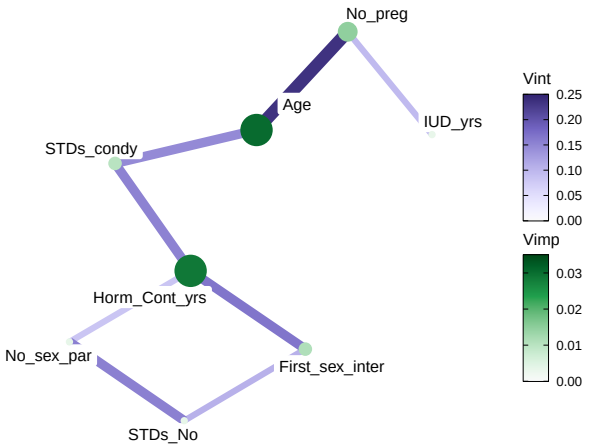


Figure 2.8: Network graph of a GBM fit on the cervical cancer data, filtered to show pairs of variables with H-index greater than 0.08.

cancer risk data, filtered to show pairs of variables with a H-index greater than 0.08 (with the cutoff chosen after inspection of the histogram of H values). The selected variables include the five variables appearing in Figure 2.7, and three additional variables, namely No_sex_par, STDs_No, and IUD_yrs (number of years with an intrauterine device), with eight relevant interactions between them. This display has some benefits over the heatmap display of Figure 2.6. Firstly, it focuses directly on pairs of variables with high interaction, particularly with the choice of network layout. Secondly, in the heatmap plot, even with seriation, some high-interaction pairs of variables may not be positioned nearby which detracts from readability. For example in Figure 2.6, associating the relevant variables with the strong interaction for (First_sex_inter, STDS_No) requires considerable effort from the reader. However, this strong interaction is immediately obvious in Figure 2.8.

To explore these interacting variables further, we use a ZPDP in Figure 2.9 to show

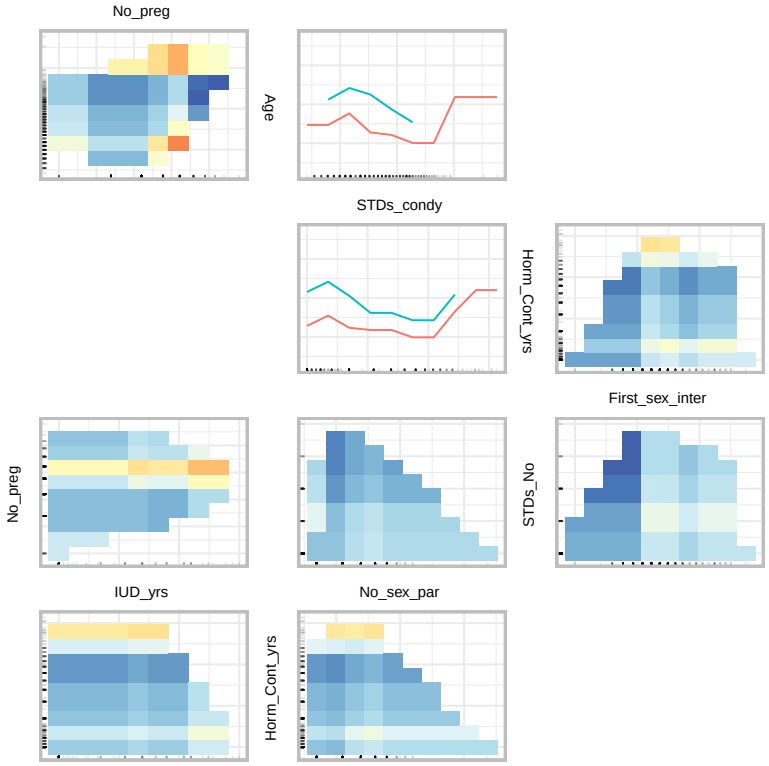


Figure 2.9: ZPDP of a GBM fit on the cervical cancer risk data. High cancer probability occurs with high number of pregnancies and high age. The color scale is the same as that of Figure 2.7.

the bivariate PDPS for the eight interactions. The Eulerian path starts with the pair of variables with the highest H-index (here No_preg:Age), and from there to Age:STDs_cond_y, ending up at No_preg:IUD_yrs. An additional plot is added corresponding to an edge between Horm_Cont_yrs and IUD_yrs to complete the Eulerian. (In this example, it would be possible to construct a ZPDP based on an Eulerian visiting each edge of the graph in Figure 2.8 exactly once, but this Eulerian ignores edge weights.) The STDs_No:No_sex_par plot (third row, second column) is a flat surface with no evidence of interaction, despite these variables having a moderate H-index. Interestingly, in the No_preg:IUD_yrs plot (third row, first column), the probability of developing cervical cancer is increasing with IUD_yrs, with a steeper gradient for moderately high No_preg. Further investigation is needed to determine the nature of this effect.

To summarize, we have used our visualizations to identify and examine some clear

risk factors associated with developing cervical cancer. Our novel approach allowed us to examine specific pairs of variables that interact and through our use of graphs and PDPs, we can examine how each variable affects the model's predictions. Specifically, the age of a patient and the number of years of hormonal contraceptive use seem to be important risk factors, agreeing with previous studies. From Figure 2.7, the women who took hormonal contraceptives for eight or more years appear to have a higher risk cervical cancer, which is in agreement with the findings of Smith et al. (2003). Surprisingly, as seen in Figure 2.6, Dx.HPV (i.e., whether the patient has had a previous diagnosis of HPV) was ranked to have middling importance, despite the known link between HPV and cervical cancer. Neither did we see evidence of an interaction between No_preg and Dx.HPV, which contrasts with Lukac et al. (2018). These differences may be due to the low frequency of positive cases in the data.

2.5 Discussion

We have presented innovative and informative methods to visualize the importance and interactions of variables simultaneously from a model. The seriated heatmap of Section 2.2.3 and the network plot of Section 2.2.4 are effective in determining which variables have the most impact on the response in a model fit. We view VIVI measures displayed in heatmap and network plots as a starting point for further detailed exploration of the nature of variable effects and interactions in the GPDP and ZPDP. The ZPDP construction is a novel application of the recently proposed zenplots of Hofert and Oldford (2020), which should prove particularly useful to focus exploration on high-VIVI subsets of variables.

Our methods are intuitive, flexible and easily customisable. Built-in or model-agnostic variable importance measures may be used in our heatmap and network displays. In our work to date, we use the model-agnostic H-statistic. Model-agnostic measures are particularly useful when comparing two or more fits. The heatmap and network displays will also be useful for comparing different VIVI measures for the same fit.

As calculation of the VIVI matrix and our visualizations are available for any

subset of the data, stratified versions or facetted displays will give insight into higher-order predictor interactions. A drawback to the H-statistic is calculation speed, which is highly model dependent, though sampling and parallel calculation offer useful speed-ups. For example, the 14×14 H-matrix for the GBM fit in Figure 2.6 computed on 30 randomly selected observations took approximately 16 seconds on a MacBook Pro 2.3 GHz Dual-Core Intel Core i5 with 8GB of RAM. Calculation for the 17-predictor random forest fit in Figure 2.1 is much slower, taking approximately 79 seconds, even though here we used just 20 randomly selected observations. A second drawback we have identified is that high H values can occur in settings where there is no feature interaction, especially in the presence of high variable correlation. The presence and nature of interactions can be further verified in the bivariate partial dependence plot, thus avoiding misleading conclusions.

A bivariate importance measure, perhaps obtained by permuting pairs of variables, could be used in place of the H-statistic in the heatmap and network visualizations. It would also be interesting to explore the interaction measures of Hooker (2004) and Greenwell and Boehmke (2020) in our visualizations, and whether these measures avoid the issues identified with the use of H.

A number of variants of the GPDP and ZPDP could be investigated in future work. One possibility for the bivariate PDP, is to subtract the two marginals plotting $f_{jk} - f_j - f_k$, which corresponds directly to the H-statistic. Alternatively, accumulated local effects (ALE) functions (Apley and Zhu, 2020) could be used in place of PDPs in our matrix layouts. ALE functions were constructed with the goal of counteracting the bias issues of partial dependence functions. Another option might be to replace the partial dependence f in Equation 2.3 with the corresponding ALE function, giving a new interaction measure.

2.6 Appendix

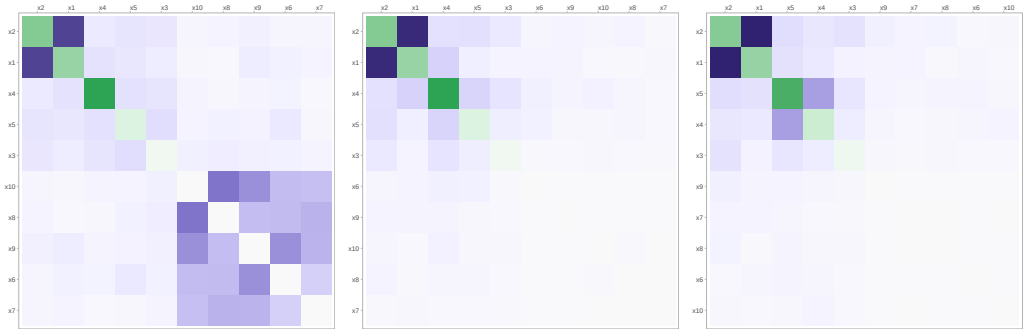
We explore some limitations of the H-statistic using a simulated dataset. We demonstrate the benefits of the un-normalized version of H, and show how correlated variables can result in spuriously high interaction measures.

Using the Friedman benchmark equation (Friedman, 1991),

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon \tag{2.5}$$
$$where \;\; x_j \sim U(0,1), j = 1, 2, \ldots, 10; \;\; \epsilon \sim N(0,1)$$

we simulate 1,000 observations and fit a random forest. There are five important variables with an interaction between x_1 and x_2, and five additional predictors $x_6, x_7, \ldots x_{10}$ unrelated to the response.

In Figure 2.10(a) and (b) we compare the normalized and un-normalized versions



(a) Normalized H (b) Un-normalized H (c) Correlated variables.

Figure 2.10: Comparison of the normalized and un-normalized H-statistic and the effect of including correlated variables for a random forest model. In (a) multiple spurious interactions are detected when using the normalized H-statistic. In (b) the spurious interactions have mostly disappeared when using the un-normalized version. In (c) (un-normalized H) a moderate spurious interaction between the correlated variables x_4 and x_5 is detected .

of the H-statistic for the simulated data. Colour legends are not useful here and are omitted. In all cases, the x_1:x_2 interaction is correctly identified. However, in (a) there are numerous spurious strong interactions among the noise variables. In (b) using un-normalized H these spurious strong interactions disappear. The culprit here is the denominator in Equation 2.2, which for variables $x_6, x_7, \ldots x_{10}$ will be close to zero, thus artificially inflating H. This is the rationale behind our use of the un-normalized H-statistic in our examples throughout.

A more subtle cause of spuriously inflated H is due to bias in the partial dependence curve. This is a particular problem in the presence of correlated predictor

variables (for example, see Apley and Zhu, 2020). To demonstrate this, we replace x_5 with $0.3x_5 + 0.7x_4$ in Equation 2.5 thus introducing a strong correlation (≈ 0.92) between x_4 and x_5. The resulting VIVI heatmap of the random forest fit in Figure 2.10(c) shows a moderate x_4:x_5 interaction which is spurious. Even in the absence of correlation, bias can occur if the fit exhibits regression to the mean. For example, this occurs with tree-based fits such as a random forest, where predictions cannot lie outside the range of training set responses. This bias is evident in Figure 2.10(b) and (c) as the light purple squares in the top-left section of the heatmaps.

CHAPTER 3

Visualisations for Bayesian Additive Regression Trees

Tree-based regression and classification has become a standard tool in modern data science. Bayesian Additive Regression Trees (BART) has in particular gained wide popularity due its flexibility in dealing with interactions and non-linear effects. BART is a Bayesian tree-based machine learning method that can be applied to both regression and classification problems and yields competitive or superior results when compared to other predictive models. As a Bayesian model, BART allows the practitioner to explore the uncertainty around predictions through the posterior distribution. In this paper, we present new Visualisation techniques for exploring BART models. We construct conventional plots to analyse a model's performance and stability as well as create new tree-based plots to analyse variable importance, interaction, and tree structure. We employ Value Suppressing Uncertainty Palettes (VSUP) to construct heatmaps that display variable importance and interactions jointly using colour scale to represent posterior uncertainty. Our new Visualisations are designed to work with the most popular BART R packages available, namely `BART`, `dbarts`, *and* `bartMachine`. *Our approach is implemented in the R package* `bartMan` *(BART Model ANalysis).*

3.1 Introduction

Bayesian Additive Regression Trees (BART; Chipman et al., 2010) is a non-parametric sum-of-trees-based ensemble method. BART has been shown to be a useful predictive tool and has been applied in diverse areas such as risk management (Liu et al., 2015), proteomics (Hernández et al., 2015), and avalanche forecasting (Blattenberger and Fowles, 2014). The BART method has also been extended into many areas, such as survival analysis (Sparapani et al., 2016) and causal inference (Hill, 2011; Hahn et al., 2020). Its excellent empirical performance has motivated works on its theoretical foundations (Linero and Yang, 2018; Prado et al., 2021). BART now enjoys widespread use due to its competitive performance against other tree-based predictive models, such as Random Forest (Breiman, 2001) and Gradient Boosted Trees (Friedman, 2000).

BART models are used for making predictions for both binary and continuous response variables and are fit using the `R` packages `dbarts` (Dorie, 2022), `bartMachine` (Kapelner and Bleich, 2016), and `BART` (Sparapani et al., 2021), among others. These packages offer limited Visualisations and in some cases leave it to the user to manually create their own Visualisations by extracting information from the fitted model. Our goal is to create novel Visualisations and to streamline this process for the aforementioned BART packages by creating a suite of plots for and evaluating both the BART fit and the posterior distribution. In our work, various aspects of a BART model can be assessed (e.g., variable importance and variable interaction) by analysing the structure of the trees used in the model. However, our approach goes beyond many standard machine learning Visualisation techniques by allowing for uncertainty in the posterior to propagate into the diagrams.

One of the more challenging aspects of model Visualisation is the depiction of uncertainty. The predictions from the BART models we create exhibit uncertainty associated with the posterior distribution, and the way we choose to represent this uncertainty may have an impact on how the model is analysed and how our audience interprets the findings. This issue has been well studied in areas that regularly deal with uncertainties in data (e.g. Pang et al., 1997; Brodlie et al., 2012). For

example, error bars, confidence intervals, or quantile intervals are common tools used to display uncertainty. However, these tools cannot be universally applied to all situations where displaying the uncertainty is necessary, such as in heatmaps or point clouds. When using point clouds to map data over many iterations, 95% confidence ellipses can be used to encircle points. An example of this can be seen in Section 3.3.3.

Methods for producing Visualisations of importance and interaction for standard machine learning models can be found in Inglis et al. (2022). However, in Bayesian models it is important to include the uncertainties that arise as part of the calculation of a full joint posterior distribution. Our new displays use a method called Value Suppressing Uncertainty Palettes (Correll et al., 2018), which allows for both the value and the uncertainty to be displayed in a single plot. Traditional methods for displaying a value and uncertainty simultaneously require a 2D bivariate map, conventionally displayed as a square (for example, see Robertson and O'Callaghan, 1986; Teuling et al., 2011). However, due to the large colour-space of 2D bivariate maps, the ability to distinguish between two different visual aspects can become challenging. VSUPs improve on this method by using an arc to assign colours and blend together data values with high uncertainty so that values become more distinguishable as the uncertainty decreases. This reduction of the visual colour-space helps to both distinguish between low and high uncertainty and promotes caution when the uncertainty is high (Correll et al., 2018).

By examining the trees in a BART model we can learn about the stability and variability of tree structures as the algorithm iterates to build the posterior. We offer new tree-based plots that focus attention on certain aspects of the model fit in an intuitive way. We provide space-saving layouts as well as providing various sorting/filtering methods and colouring options. When combined with ordination techniques, we provide easy to use tools which aid in highlighting interesting aspects of the model fit, such as variable importance or common interactions.

Multidimensional scaling (MDS) plots are a common method for graphically displaying relationships between objects in multidimensional space (Torgerson, 1952). Objects that are similar appear closer on the graph, whereas objects objects that

are less similar are farther away. MDS can be used to reduce the number of dimensions in high-dimensional data as well as interpret dissimilarities as graph distances. We construct an MDS display of a BART fit and extend it to display the uncertainty. For each iteration of the BART fit, we perform MDS on proximities and rotate each plot to match a particular target iteration. From this we get a point cloud, where a confidence ellipse is used to encircle each observation. With this display the analyst can explore, for example, outliers that may require further investigation.

Aside from our three main novel Visualisations, we include a selection of standard diagnostic plots, such as trace, residual, and overall model fit plots, that will quickly assess aspects such as convergence and model behavior. Each of our plots can be run on any of the aforementioned `R`-packages, despite their differing formats and function arguments.

While we make what we believe to be good default choices for the plots we produce, we provide the option to adjust many of the settings. Each aspect of the design of our plots is given careful consideration; we focus on efficient layouts, which includes both clustering and filtering, colour choice, and effectively displaying uncertainty. Our new displays are appropriate for regression and classification fits and are designed to work with the three aforementioned BART packages but could readily be extended to incorporate other BART packages. Our implementation is available as the `R` package `bartMan` (BART Model ANalysis) which is found at https://github.com/AlanInglis/bartMan.

The outline of this paper is as follows: in Section 3.2 we describe the formulation of a BART model and provide a brief discussion on how to access variable importance and variable interactions. In Section 3.3 we describe our new Visualisations for assessing variable importance and variable interactions with uncertainty, tree-based analysis, outlier identification with multidimensional scaling, and a selection of enhanced model diagnostic plots on a simple example. In Section 3.4 we study BART's variable importance and variable interaction methods compared to a model agnostic approach. In Section 3.5 we demonstrate our new methods on a case study. Finally, in Section 3.6 we conclude by discussing potential advan-

tages and disadvantages of our approach, as well as potential avenues for further research.

3.2 Bayesian Additive Regression Trees and Variable Importance

We begin with by reviewing Bayesian additive regression trees and follow with a review of both variable importance and variable interactions in a BART model.

3.2.1 A Short Introduction to Bayesian Additive Regression Trees

In this section we provide a brief overview of the BART model to aid the reader in understanding our later Visualisations. Those looking for a more complete description should see Chipman et al. (2010). BART is a Bayesian non-parametric model based on an ensemble of trees that can be used for predicting continuous and multi-class responses. Unlike regression models where a linear structure is pre-specified, BART does not assume any functional form for the model, and so automatically uncovers main and interaction effects. Given a continuous response variable y_i with associated predictors $\mathbf{x}_i$, the BART model, with m trees is expressed as:

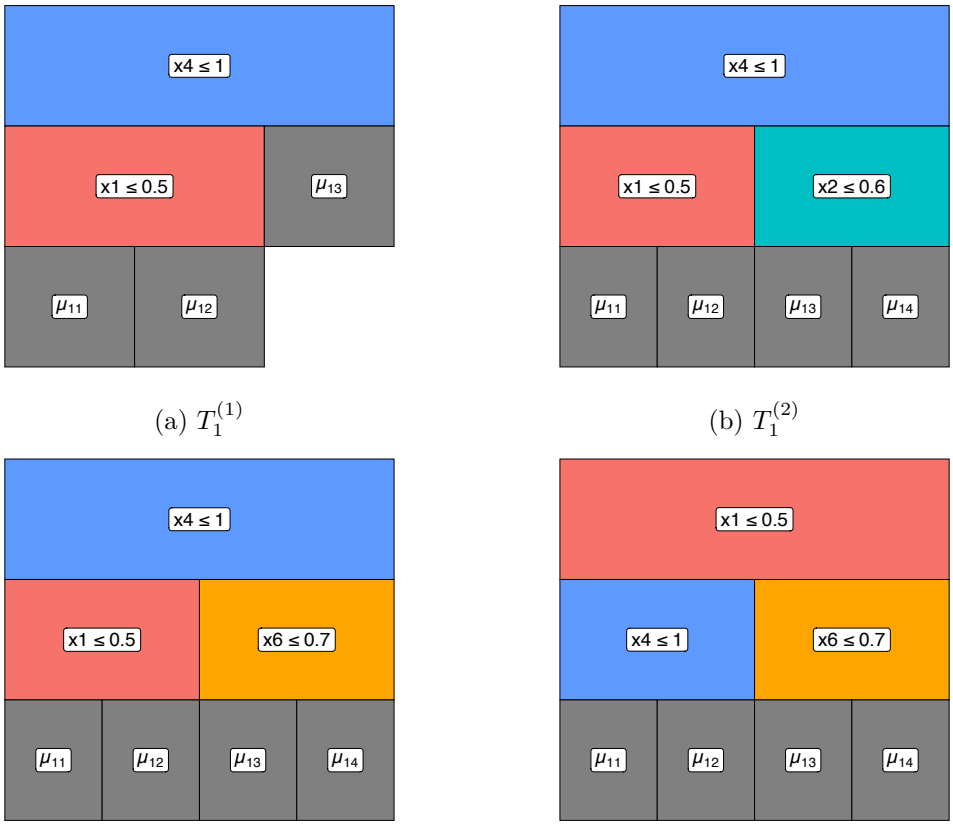
$$y_i = \sum_{j=1}^{m} g(\mathbf{x}_i, T_j, M_j) + \epsilon_i, \tag{3.1}$$

where $\epsilon_i \sim \mathrm{N}(0, \sigma^2)$ and $g(\mathbf{x}_i, T_j, M_j) = \mu_{j\ell}$ is a function that assigns a predicted value for the observations falling into terminal node ℓ of tree j. T_j represents the structure/topology of tree j including the split variables and the values associated with the splits $M_j = (\mu_{j1}, \cdots, \mu_{jb_j})$ represent the set of predicted values at the b_j terminal nodes of the trees.

The tree structure T is composed of binary splitting rules of the form $[x_j \leq c]$, where observations which satisfy the condition go to the left and the remainder to the right. The trees are updated at each iteration in a Markov chain Monte Carlo approach where each tree structure is modified by either growing, pruning,

changing, or swapping nodes. Growing a tree means that a terminal node is randomly chosen and two new terminal nodes are created, while pruning collapses a pair of terminal nodes to their parent. A splitting rule can also be changed to a different rule, or swapped for another splitting rule in the same tree. In the grow and change moves a new splitting rule is required and is proposed by uniformly sampling a splitting variable and a split value though the exact generation of these rules is implementation dependent.

Figure 3.1 shows an example of the tree structure modifications in action. In Figure 3.1, a tree, T_1^k, is generated from BART in 4 different instances, where $k = 1, 2, 3, 4$ indicates the iteration number in which the tree is updated. In the full BART model multiple trees are estimated and the predictions are created from the sum of the μ values across the trees. The tree is displayed as an icicle plot (Kruskal and Landwehr, 1983) with the splitting rules (that is, covariates and split points) shown as coloured rectangles and the terminal nodes $\mu_{j\ell}$ are shown as grey rectangles. Icicle plots were first introduced by Kruskal and Landwehr (1983) as a way to display hierarchical data in a space efficient manner. We use icicle plots to display our tree plots in later sections.



(a) $T_1^{(1)}$ (b) $T_1^{(2)}$

(c) $T_1^{(3)}$ (d) $T_1^{(4)}$

Figure 3.1: An example of a tree, T_1^k, generated from BART over $k = 1, 2, 3, 4$ iterations. displayed as an icicle plot with the splitting rules (that is, covariates and split points) shown as coloured rectangles and the terminal nodes $\mu_{j\ell}$ are shown as grey rectangles. In panel (a), $k = 1$, observations that satisfy the splitting criterion go left and tree $T_1^{(1)}$ has two internal nodes and three terminal nodes. Moving from panel (a) to (b) shows the grow move for the tree. Reverting from (b) back to (a) corresponds to a prune move. Panel (c) shows the change move as the splitting rule that defines μ_{13} and μ_{14} in $T_1^{(2)}$ is changed. Finally, in (d) the swap move can be seen when comparing the internal nodes of $T_1^{(3)}$ and $T_1^{(4)}$.

In panel (a) of Figure 3.1 at iteration 1, observations that satisfy the splitting criterion go left and tree $T_1^{(1)}$ has two internal nodes and three terminal nodes. The grow move is shown going from panel (a) to panel (b), that is $T_1^{(1)}$ to $T_1^{(2)}$. An example prune move would correspond to $T_1^{(2)}$ reverting to $T_1^{(1)}$. In panel (c)

we can see the change move as the splitting rule that defines μ_{13} and μ_{14} in $T_1^{(2)}$ is changed. Finally, in (d) the swap move can be seen when comparing the internal nodes of $T_1^{(3)}$ and $T_1^{(4)}$.

As a Bayesian model, BART adopts a set of prior distributions for the tree structure, terminal node parameters, and residual variance. To control the depth and shape of the tree structure, a branching process prior is considered where the probability of a node being non-terminal at depth d is proportional to $\alpha(1+d)^{-\beta}$, where $\alpha \in (0,1)$ and $\beta \geq 0$. Chipman et al. (2010) recommend $\alpha = 2$ and $\beta = 0.95$ as default, which favours shallow and balanced trees. A side effect of this choice is that noise (i.e. uninformative) variables are often chosen in shallower tree structures as the tree prior can outweigh the likelihood when a large number of trees is used. The terminal node parameters $\mu_{j\ell}$ are assumed to be independent and identically distributed, that is, $\mu_{j\ell} \sim \mathrm{N}(0, \sigma_\mu^2)$, where σ_μ^2 is the residual variance of the terminal node parameters, which is usually fixed. The value of σ_μ^2 is usually set with the aim of forcing the trees to be shallow and shrink their predictions towards zero so that each tree only contributes a small amount to the overall prediction. Finally, the prior on the residual variance σ^2 is an Inverse Gamma.

Posterior sampling is based on a Metropolis-within-Gibbs MCMC structure where the trees are sequentially updated through partial residuals. For one MCMC iteration, each tree in the ensemble is modified and then compared to its previous version via a Metropolis-Hastings update. The update involves a marginalised likelihood and the tree prior. The marginalised likelihood is an essential element to avoid trans-dimensional MCMC, and simplifies computation. Given the tree structure, all terminal node parameters $\mu_{j\ell}$ are updated based on a closed-form posterior conditional distribution. After updating all trees, the variance σ^2 is updated; a more complete description can be found in Chipman et al. (2010) and Tan and Roy (2019).

In the above, we have described the BART model for a univariate and continuous response variable. However BART has been extended into many different areas, such as survival analysis (Sparapani et al., 2016; Linero et al., 2021), time series analysis (Starling et al., 2020), multivariate skewed response (Um, 2021), and high-

dimensional data (Hernández et al., 2018; Linero and Yang, 2018; He et al., 2019). While in this work we do not apply our methods to these extensions of BART, there is in principle no reason why our methodology could not be extended to incorporate the above BART extensions.

3.2.2 Variable Importance and Variable Interaction Calculations with BART

Variable importance is a measure of a single variable's impact on the response. Multiple methods exist for evaluating variable importance, depending on the model; for a comprehensive review of different variable importance techniques see Wei et al. (2015a). Chipman et al. (2010) propose a method called the inclusion proportion to evaluate the variable importance in a BART model from the posterior samples of the tree structures. Their measure of variable importance first calculates for each iteration the proportion of times a variable is used to split nodes considering all m trees, and then averages these proportions across all iterations.

More formally, let K be the number of posterior samples obtained from a BART model. Let c_{rk} be the number of splitting rules using the rth predictor as a split variable in the kth posterior sample of the trees' structure across m trees. Additionally, let $c_{\cdot k} = \sum_{r=1}^{p} c_{rk}$ represent the total number of splitting rules found in the kth posterior sample across the total p variables. Therefore, $z_{rk} = c_{rk}/c_{\cdot k}$ is the proportion of splitting rules for the rth variable, and the average use per splitting rule is given by:

$$\text{VImp}_r = \frac{1}{K} \sum_{k=1}^{K} z_{rk} \tag{3.2}$$

However Chipman et al. (2010) noted that this method of evaluating importance is less effective when the number of trees, m, is large because weakly influential predictor variables can be added to the tree structure and so may provide spurious importance values for the non-important variables. As m decreases this effect is diminished because the less important variables get swapped out of the trees for more informative variables.

Variable interaction is generally considered as when a pair (or more) of variables jointly impact on the response. In our work we focus on bivariate interactions only. Kapelner and Bleich (2016) suggested a measure of interaction obtained by observing successive splitting rules in each tree. Let c_{rqk} be the number of splitting rules using predictors r and q successively (in either order) in the kth posterior sample. Additionally, let $c_{\cdot\cdot k} = \sum_{r=1}^{p} \sum_{q=1}^{p} c_{rqk}$ represent the total number of successive splitting rules found in the kth posterior sample. We follow the convention of Kapelner and Bleich (2016) and we treat the order of successive splits as not important and we sum the r, q counts with the q, r counts. Therefore, the proportion $z_{rqk} = c_{rqk}/c_{\cdot\cdot k}$, provides an estimate of the interaction between variables r and q:

$$\text{VInt}_{rq} = \frac{1}{K} \sum_{k=1}^{K} z_{rqk} \tag{3.3}$$

As this method follows a similar technique to evaluating the inclusion proportion, the same pitfalls noted by Chipman et al. (2010) apply, namely that the prior distribution may favour trees containing successive predictor variables where there is no true interaction present if the number of trees is large. For a comparison of both the variable importance and variable interaction methods against a model agnostic approach for evaluating these metrics, see Section 3.4.

It should be noted that if any of the variables used to build the BART model are categorical, the aforementioned BART packages replace the categorical variables with d dummy variables, where d is the number of factor levels. For some of our plots, the inclusion proportions for variable importance and interaction are then adjusted by aggregating over factor levels. This provides a complete picture of the importance of a factor, rather than that associated with individual factor levels.

Since both the VImp and VInt values are calculated from the full posterior, it is trivial to compute an uncertainty associated with their measurement, simply by storing the importance and interaction calculations per iteration. These can be summarised by the usual means by which posterior distributions are analysed. We will use uncertainty metrics obtained from these distributions in our variable importance and interaction displays of Section 3.3.

3.3 New visualisations for BART

To illustrate our new Visualisations we use a subset of the iris data (Fisher, 1936) where the response is binary and made up of two species (that is, setosa and versicolor). We then fit a BART model to the data using `bartMachine`, using the default setting of 1000 iterations with a burn-in of 250. For simplicity of exposition we set the number of trees to be 20.

We introduce the following Visualisations: improved plots of variable importance and interaction which include the uncertainty induced by the posterior distribution of trees; plots of the tree structures which show the splitting variables, the split distribution, and the terminal node values; the ability to identify outlying and influential observations through the terminal node proximity matrix and multi-dimensional scaling; and a set of enhanced model diagnostics for identifying convergence and performance issues.

3.3.1 Variable Importance and Interaction with Uncertainty

In this section we present visualisations of the variable importance methods described in Section 3.2.2. In Figure 3.2 we show the median of the inclusion proportion as a black point, with the variables ordered from the largest median importance measure (at the top) and descending. In this case the 25% to 75% quantile interval extending from each point is displayed as a grey bar. We can see that Petal.Length is the most important variable and that Sepal.Width and Petal.Width have similar inclusion proportions. Sepal.Width importance has a lower degree of uncertainty, as indicated by the relatively small quantile interval, whereas the Petal.Width importance has a large quantile interval associated with it, and therefore its importance measure should be viewed with a level of caution.

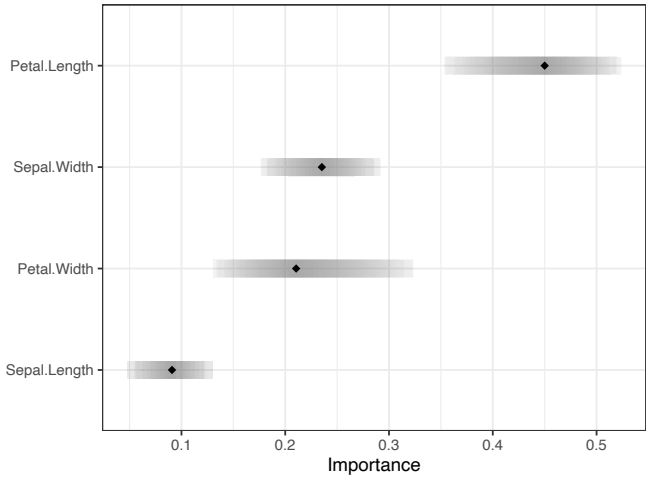
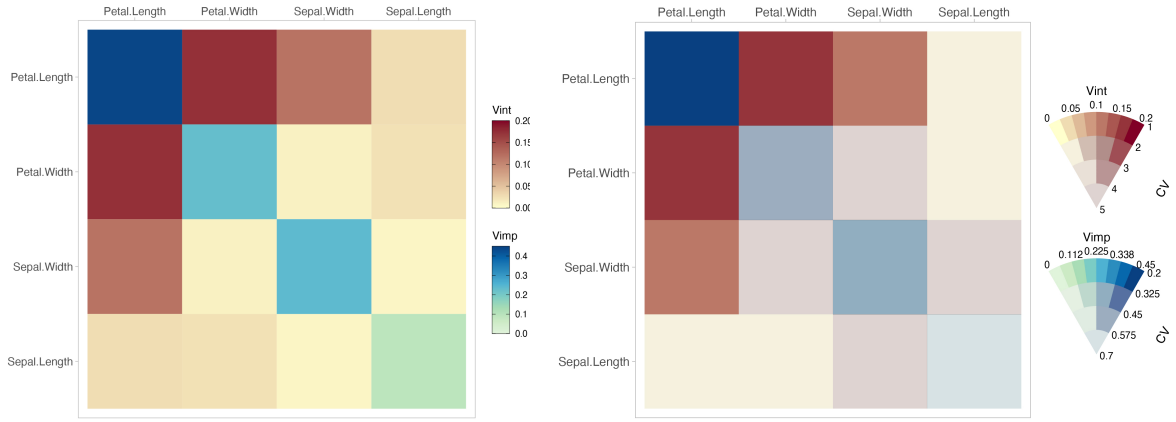


Figure 3.2: Inclusion proportions for the iris data are shown with the 25% to 75% quantile interval extending from the points. Here Petal.Length is ranked as the most important variable.

In Inglis et al. (2022), the authors propose using a heatmap to display both importance and interactions simultaneously, where the importance values are on the diagonal and interaction values on the off-diagonal. The advantage of such a display is that it allows one to easily identify which variables are relevant as separate predictors while also seeing which variable pairs have high interaction. This method, coupled with the seriation technique described by Inglis et al. (2022), brings predictors with high importance and interaction to the top-left of the heatmap and less relevant predictors to the bottom-right.

Here we adapt the heatmap displays of importance and interactions to include the uncertainty using a VSUP. The colours for the VSUP heatmap were carefully chosen to be distinguishable, colour-blind friendly, and to aid in highlighting high values, while still making the uncertainty prominent. To achieve this, we follow the advice of Strode et al. (2019), who build upon the work of Trumbo (1981), and aim to highlight and focus the reader's attention on the interesting data.



(a) (b)

Figure 3.3: In (a) the importance values are on the diagonal and interaction values on the off-diagonal. Petal.Length is the most important variable and there is a strong interaction between Petal.Length and Petal.Width. In (b) the same values values are shown but with the coefficient of variation included by use of a VSUP. Both the importance measure of Petal.Length, and the interaction measure between Petal.Length and Petal.Width have low coefficient of variation.

Figure 3.3 presents a comparison of heatmaps showing the importance and interactions jointly with and without uncertainty. In both heatmaps the variable importance is displayed on the diagonal and the interactions on the off-diagonal. In (a), we can see that Petal.Length is the most important variable when predicting Species. There also appears to be a strong interaction between Petal.Length and Petal.Width. In (b) the same values are shown but with a measure of uncertainty included, in this case the coefficient of variation (CV). Other error metrics such as standard deviation can be applied, though in the case of using proportions larger values tend to have greater uncertainty and so our preference is for the CV. In both (a) and (b) the same method is used to obtain the importance and interaction scores, resulting in comparable scales. Comparing the two plots we observe that in (b) the most important variable, Petal.Length, has a small variation relative to its mean. The Petal.Length and Petal.Width interaction value has a low coefficient of variation and is consequently highlighted in (b), whereas Petal.Length and Sepal.Length have a low interaction score with relatively high variation.

3.3.2 Tree-Based Plots

In this section we examine more closely the structure of the decision trees created when building a BART model. Examining the tree structure may yield information on the stability and variability of the tree structures as the algorithm iterates to create the posterior. By sorting and colouring the trees appropriately we can identify important variables and common interactions between variables for a given iteration. Alternatively we can look at how a single tree evolves through the iteration to explore the fitting algorithm's stability.

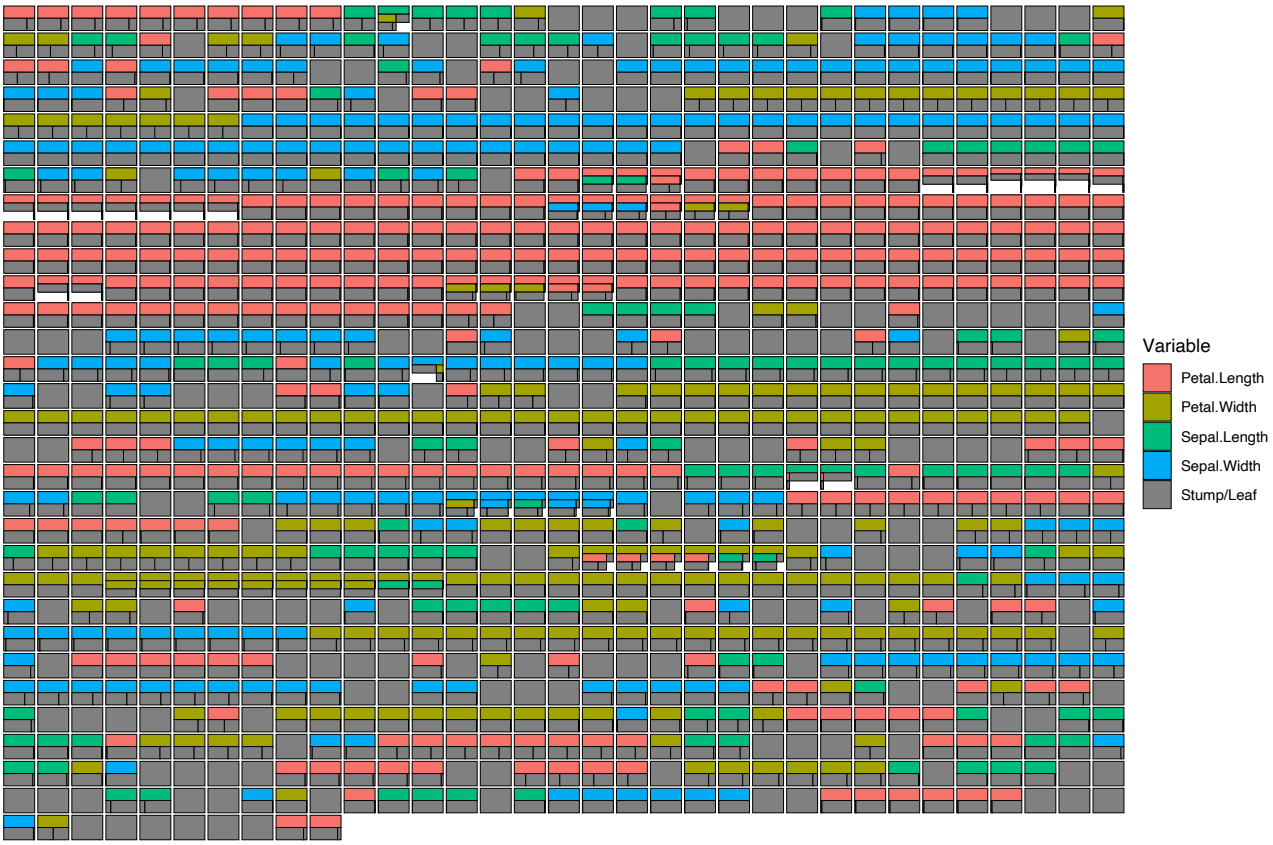


Figure 3.4: A single tree over 1000 iterations. The coloured bars indicate which variable is used for the split at that point. Grey boxes indicate stumps or terminal nodes. The vertical black lines in the terminal nodes indicate the proportion of the data being split into the left or right terminal node.

In Figure 3.4, we show how a single selected tree changes over all 1000 post burn-in iterations. We use an icicle plot to display the trees. As noted by Barlow and

Neville (2001), icicle plots are preferred by users when compared to other methods to display decision trees and use space more efficiently. Additionally, the number of observations within each decision tree node is represented in icicle plots by scaling the node size accordingly. In Figure 3.4, each parent node is coloured according to the variable with the terminal nodes all coloured a dark grey. A stump is represented by a solid grey square (although stumps can be removed from the plots if desired). (More options to colour the nodes by certain parameters are shown in later plots in this section.) With this display we see how a tree evolves over iterations. Here we see the prevalence of Petal.Length as a splitting variable (red rectangles) once again indicating the importance of this predictor. Additionally, most iterations have a single split on the root node, with very few trees showing an interaction. As the nodes are sized according to the number of observations, we observe that in the seventh and eighth rows some trees have an empty, white space. In this case most of the observations fall into the single terminal node on the left. The remaining observations go right and split again.

In our tree displays, it is also useful to view different aspects or metrics. In Figure 3.5 we explore some of these aspects by displaying all the trees in a selected iteration (in this case, we chose the iteration with lowest residual standard deviation). We consider variations which colour terminal nodes and stumps by the mean response (panel (a)), colour them by the terminal node parameter value (panel (b)), sort the trees by structure starting with the most common tree and descending to the least common tree found for easy identification of the most important splits (panel (c)), or sort the trees by depth (panel (d)). As the μ values in (b) are centred around zero, we use a single-hue, colourblind friendly, diverging colour palette to display the values. For comparison, we use the same palette to represent the mean response values in (a).

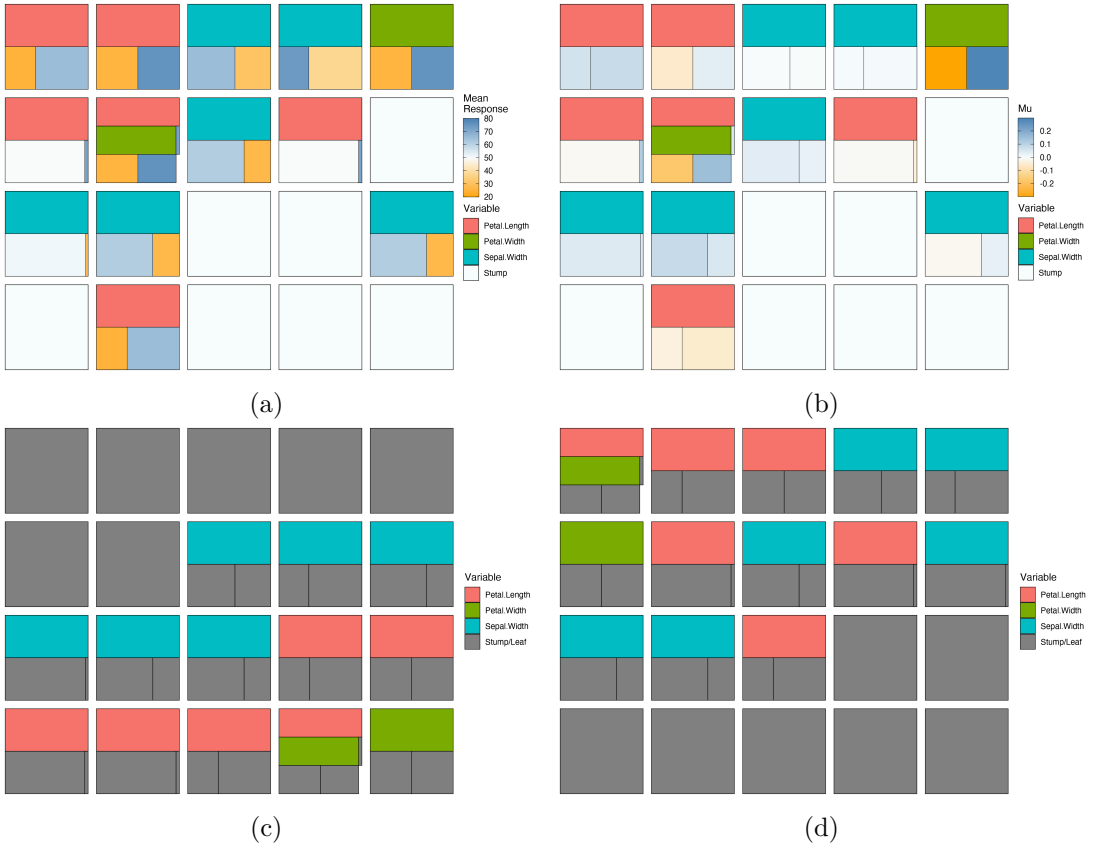


Figure 3.5: All trees in a selected iteration. In (a) the terminal nodes and stumps are coloured by the mean response. In (b) the terminal nodes and stumps are coloured by the predicted value μ. In (c) we sort the trees by structure starting with the most common tree and descending to the least common tree shape and in (d) we sort the trees by tree depth.

Different interesting findings are seen in the four panels. Panel (b) indicates that tree 5 (the top right tree displayed) has a much greater influence on the overall predictions than the others, which seems surprising given the nature of the shrinkage prior used in BART which aims to shrink the terminal node parameters towards zero. From (c) we observe that the most common tree structure in this iteration is actually a stump. The most common non-stump tree type has Sepal.Width as the root with a single binary split. Furthermore, in this iteration Petal.Length and Sepal.Width are both used as a splitting variable an equal number of times. In (d) it is quickly identified that the vast majority of trees in this iteration have

one or zero splits.

When the number of variables or trees is large it can become harder to identify interesting features. We provide a plot that can be used to highlight interesting features by accentuating selected variables by colouring them brightly while uniformly colouring the remaining variables a light grey. When coupled with the sorting shown previously in Figure 3.5 we have found that this more clearly identifies relationships of interest. As the iris data has very few predictors, we omit this plot here but an example of it can be seen the larger case study example of Figure 3.14 in Section 3.5.

Finally, as an alternative to the sorting of the tree structures, seen in Figure 3.5 (c), we provide a bar plot summarising the tree structures. Figure 3.6 shows a barplot of the frequency of the tree types over all iterations, filtered to show the top 10 most frequent trees, where the legend indicates the tree structure with the node sizes equally proportioned. To count the tree structures, we use the same sorting algorithm as Figure 3.5 (c). This seems most useful when summarising a large number of trees (though again these plots can also be created for a single tree across iterations or to display all trees in a single iteration). We can see that the most common tree type over all iterations is the tree that has a single binary split on Petal.Length, with the second most common being the tree that has a single binary split on Sepal.Width. Additionally, we can see that Petal.Length appears in several of the other top 10 most common tree structures. This is in agreement with the inclusion proportion variable importance plot of Figure 3.2 which tells us that Petal.Length is used as a splitting rule most often.

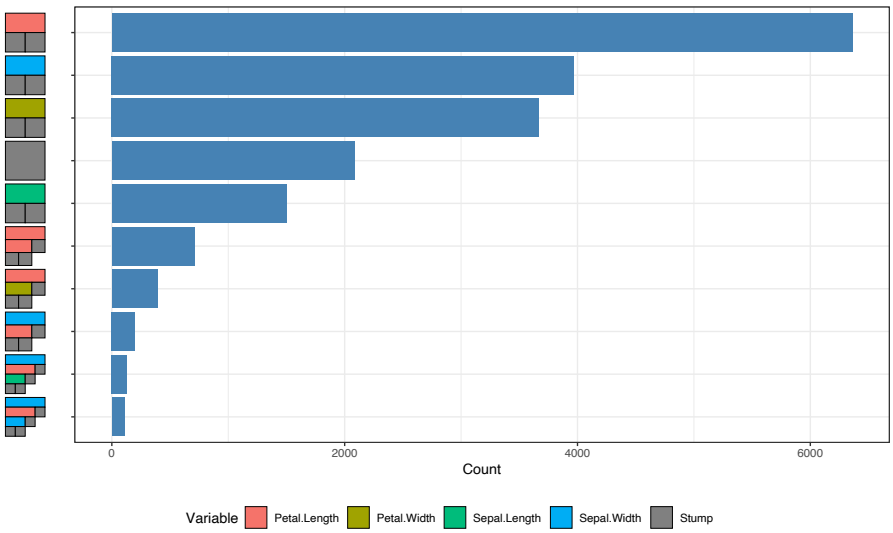


Figure 3.6: Bar plot of the top 10 most frequent tree types over all iterations. Trees with a single binary split on Petal.Length occur the most often.

3.3.3 Outlier Identification with Multidimensional Scaling

Proximity matrices combined with multidimensional scaling (MDS) are commonly used in random forests to identify outlying observations (Breiman, 2001). Both proximites and MDS have been shown to be useful tools and can be applied to wide range of data types, including genomic and ecological data (for example, see Englund and Verikas, 2012; Cutler et al., 2007). However, to our knowledge, these methods have not yet been implemented for a BART model. When two observations lie in the same terminal node repeatedly they can be said to be similar, and so an $N \times N$ proximity matrix is obtained by accumulating the number of times at which this occurs for each pair of observations, and subsequently divided by the total number of trees. A higher value indicates that two observations are more similar. The proximity matrix is then visualised using classical MDS (henceforth MDS) to plot their relationship in a lower dimensional projection.

In BART there is a proximity matrix for every iteration and thus a posterior distribution of proximity matrices. While trivial to then apply MDS to each matrix we introduce a rotational constraint so that we can similarly obtain a posterior distribution of each observation in the lower dimensional space. We first choose a

target iteration (we use the iteration with lowest residual standard deviation) and apply MDS. For each subsequent iteration we rotate the MDS solution matrix to match this target as closely as possible using Procrustes' method. We end up with a point for each observation per iteration per MDS dimension. We then group the observations by the mean of each group and produce a scatterplot, where each point represents the centroid of the location of each observation across all the MDS solutions. This allows for an easier to read estimate of potentially outlying data points. We extend this further by displaying the 95% confidence ellipses around each observation's posterior location in the reduced space. Since these are often overlapping we have created an interactive version that highlights an observation's ellipse when hovering the mouse pointer above the ellipse (Figure 3.7 shows a screenshot of this interaction in use). The observation number is also displayed during this action.

In Figure 3.7, each point represents the centroid of the location of each observation across all the MDS solutions and are coloured according to their class (in this case, either Species). We can see that most of the variability is, unsurprisingly, in the first-dimension, and while some points have quite different posterior distributions, the uncertainty on many of them is large. Observation 86 appears to have a large uncertainty and is separated by some distance from the other observations in that class. This is an interesting finding as previous outlier detection studies using the iris data (such as Acuna and Rodriguez (2004) and Liu et al. (2015)) have not identified this observation as an outlier. Further investigation, by examining the tree structure and a proximity matrix plot (which, in the interest of space, we omit here) show that this observation is commonly found in the same nodes as those observations from the other class.

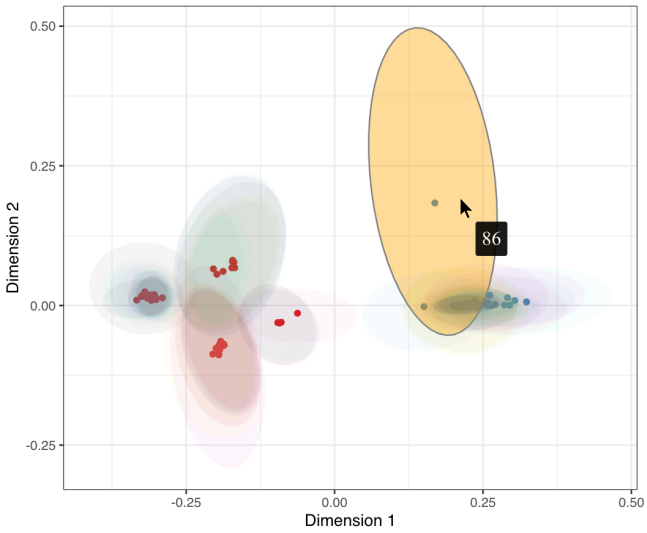


Figure 3.7: Interactive MDS plot of the iris data where the points are coloured by class (in this case, either Species). Each 95% confidence ellipse corresponds to each observation's posterior location. When hovering the mouse pointer over an ellipse, the ellipse is highlighted and the observation is displayed.

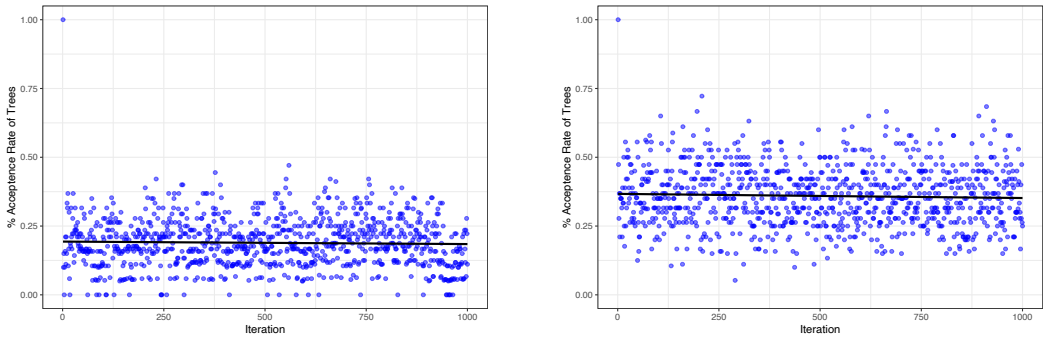
3.3.4 Enhanced BART model diagnostics

In this section, we examine some of the more common issues a researcher may face when running a BART model. These include checking for convergence, the stability of the trees, the efficiency of the algorithm, and the predictive performance of the model. In our experience, most popular BART `R` packages are limited in scope for creating informative model Visualisations (with the possible exception of `bartMachine` which features versions of Figures 3.8 and 3.9). Our goal in these plots is to provide a convenient and useful summary of the model's characteristics which is invariant to the choice of package. A useful side effect of these plots is the ability to compare BART fits from different BART `R` packages. In the following section we show a selection of diagnostic plots using both the `bartMachine` and `dbarts` packages to build our models. Both models have the same hyperparameters of 1000 iterations with a burn-in of 250 and 20 trees. We use the same two-species subset of the iris data as before.

3.3.4.1 Acceptance Rate of Trees

As discussed in Section 3.2, BART uses a Metropolis-Hastings algorithm to determine the type of tree structure accepted at each tree in each MCMC iteration. The trees are individually modified by either a grow, prune, change, or swap step and compared to its previous version by calculating the acceptance ratio. The acceptance rate is therefore measured as the percentage of accepted proposed trees across the iterations.

Figure 3.8 shows the post burn-in percentage acceptance rate across 1000 iterations for both BART models, where each point represents a single iteration. A regression line is shown to indicate the changes in acceptance rate across iterations and to identify the mean rate. Both plots are forced to display the same vertical axis range. Clearly there is a higher acceptance rate (approx 35%) in the `dbarts` fit. None of the iterations in `dbarts` have zero trees accepted, while this occurs commonly for `bartMachine`. This can also be seen in Figure 3.4 where there are runs of identical trees, indicating that no new trees were accepted during this period.



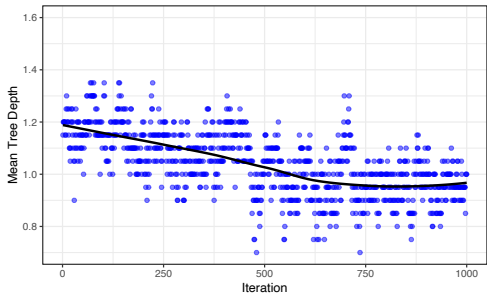
(a) Acceptance rate per iteration using `bartMachine`

(b) Acceptance rate per iteration using `dbarts`

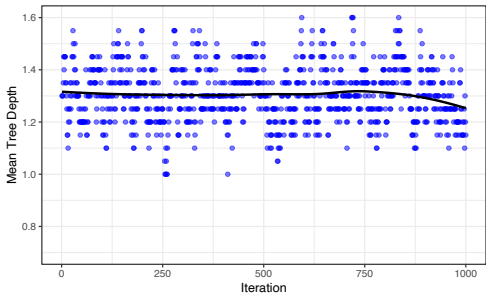
Figure 3.8: Post burn-in acceptance rate of trees per iteration for a `bartMachine` and `dbarts` fit in (a) and (b), respectively. A black regression line is shown to indicate the changes in acceptance rate across iterations and to identify the mean rate. We can see that the `dbarts` fit has a higher acceptance rate than the `bartMachine` fit.

3.3.4.2 Tree Depth, Node Number, and Split Distribution

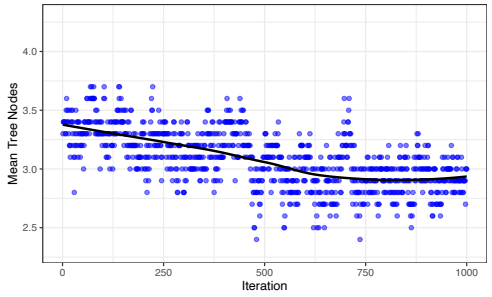
As with the acceptance rate, the average tree depth and average number of all nodes per iteration can give an insight into the fit's stability. Figure 3.9 displays these two metrics for both BART fits. A locally estimated scatterplot smoothing (LOESS) regression line is shown to indicate the changes in both the average tree depth and the average number of nodes across iterations. From Figure 3.9 (a) and (c), we can see that both the post burn-in average tree depth and the average number of nodes per iteration is much more stable in the `dbarts` fit. However, although we use the default number of iterations suggested by the `bartMachine` package, increasing this may improve stability.



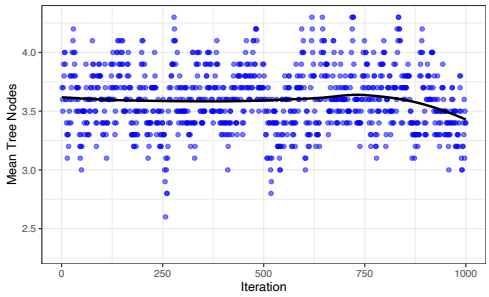
(a) Average tree depth from `bartMachine`.



(b) Average tree depth from `dbarts`.



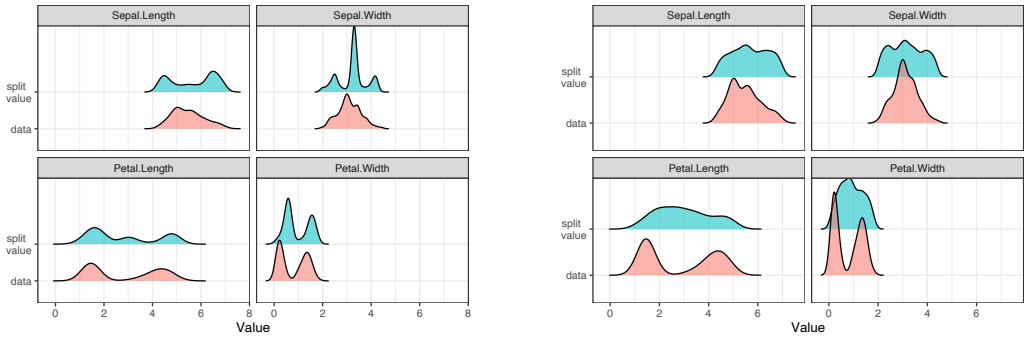
(c) Average number of nodes from `bartMachine`.



(d) Average number of nodes from `dbarts`.

Figure 3.9: In the top row we show the post burn-in average tree depth per iteration for a `bartMachine` and `dbarts` fit in (a) and (b), respectively. In the bottom row we show the post burn-in average number of nodes per iteration for a `bartMachine` and `dbarts` fit in (c) and (d), respectively. A black LOESS regression curve is shown to indicate the changes in both the average tree depth and number of nodes across iterations.

Figure 3.10 shows the densities of split values over all post burn-in iterations for each variable for both models (in green), combined with the densities of the predictor variables (labelled "data", in red). This plot appears to be new; we have not found anything similar in any of the existing packages. We can see that the split value density for Sepal.Width in the `bartMachine` fit, in (a), has large peak at around 3.2 and the `bartMachine` fit's split values have more modes.



(a) Split value distribution obtained from a `bartMachine` fit.

(b) Split value distribution obtained from a `dbarts` fit.

Figure 3.10: Split values densities (in green) over all iterations for each variable overlayed on the densities of the predictors (in red) for a `bartMachine` fit in (a) and a `dbarts` fit in (b).

In addition to the previous plots, we provide a panel of basic summary diagnostics of the model fit which can be used for both classification and regression models. For the former, we display metrics such as precision-recall and ROC (with uncertainties included), a confusion matrix, fitted value plots, and a histogram of predicted probabilities. For the latter, we show a trace plot of the model variance, a Q-Q plot, and an array of model performance plots and residual plots over all iterations. In the interest of space, we exclude the summary diagnostics for the classification model and display the summary diagnostic plots for the regression model only, as seen in Section 3.5.

3.4 Comparative analysis of variable Importance and Interactions in a BART model

In this section we provide an examination of the variable inclusion proportion methods for evaluating importance and interactions in a BART model (as outlined in Section 3.2.2) by comparing the raw inclusion proportions with and without uncertainty included against alternative methods used to assess the importance and interactions of variables. These alternative methods do not allow for the inclusion of uncertainty in the metrics they create.

As previously discussed, BART models obtain a measure of importance by observing the proportion of times a variable is used as a split variable across all trees, averaged over all iterations. The more times a variable is used as a split variable, the more important that variable is deemed to be. Similarly, a measure of interaction can be obtained in a BART model by observing the proportion of successive splits over all trees, averaged over all iterations. However, as noted by Chipman et al. (2010), this method of assessing importance (and interactions) comes with certain pitfalls. Namely, if the number of trees is large, then non-important predictor variables can be preferred as the likelihood is relatively flat and so the tree prior dominates. This can lead to to spurious importance and interactions scores for variables that, in reality, have little influence on the response. This effect can be mitigated somewhat by the inclusion of uncertainty to evaluate the reliability of the measured importance or interaction scores. Additionally, Chipman et al. (2010) state that decreasing the number of trees when building the model diminishes this effect as less important variables get swapped out of the trees for more informative variables.

To compare the usefulness of a BART model's importance and interactions, we compare the BART methodology, with and without uncertainty included, against a model agnostic approach to assess the importance and interactions. To measure the agnostic variable importance we use a permutation method. Permutation importance was first introduced by Breiman (2001) and works by calculating the

change in the model's predictive performance after a variable has been randomly permuted. That is, a model score is initially recorded, then a single variable is randomly permuted (this is repeated for each variable) and the model score is re-calculated on the new dataset. The difference between the baseline model's performance and the permuted model's performance is taken as the variable importance score. To measure the agnostic interactions we use Friedman's H-statistic (or H-index) (Friedman and Popescu, 2008). For this method the partial dependence for a pair of variables is compared to their marginal effects.

Friedman's H-statistic is defined as:

$$H_{jk}^2 = \frac{\sum_{i=1}^{n}[f_{jk}(x_{ij}, x_{ik}) - f_j(x_{ij}) - f_k(x_{ik})]^2}{\sum_{i=1}^{n} f_{jk}^2(x_{ij}, x_{ik})} \tag{3.4}$$

where $f_{jk}(x_j, x_k)$ represents the two-way partial dependence function of both variables, $f_j(x_j)$ and $f_k(x_k)$ represent the partial dependence functions of the single variables, and all partial dependence functions are mean-centered. The obtained measure is scaled in the range (0,1). Inglis et al. (2022) note, however, that variations in the numerator can lead to spuriously high H-values when the denominator in (3.4) is small because the partial dependence function for the variables j and k is flat in this case. To combat this, the square-root of the average un-normalized (numerator only) version of Friedman's H^2 for calculating pairwise interactions is suggested:

$$H_{jk} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}[f_{jk}(x_{ij}, x_{ik}) - f_j(x_{ij}) - f_k(x_{ik})]^2} \tag{3.5}$$

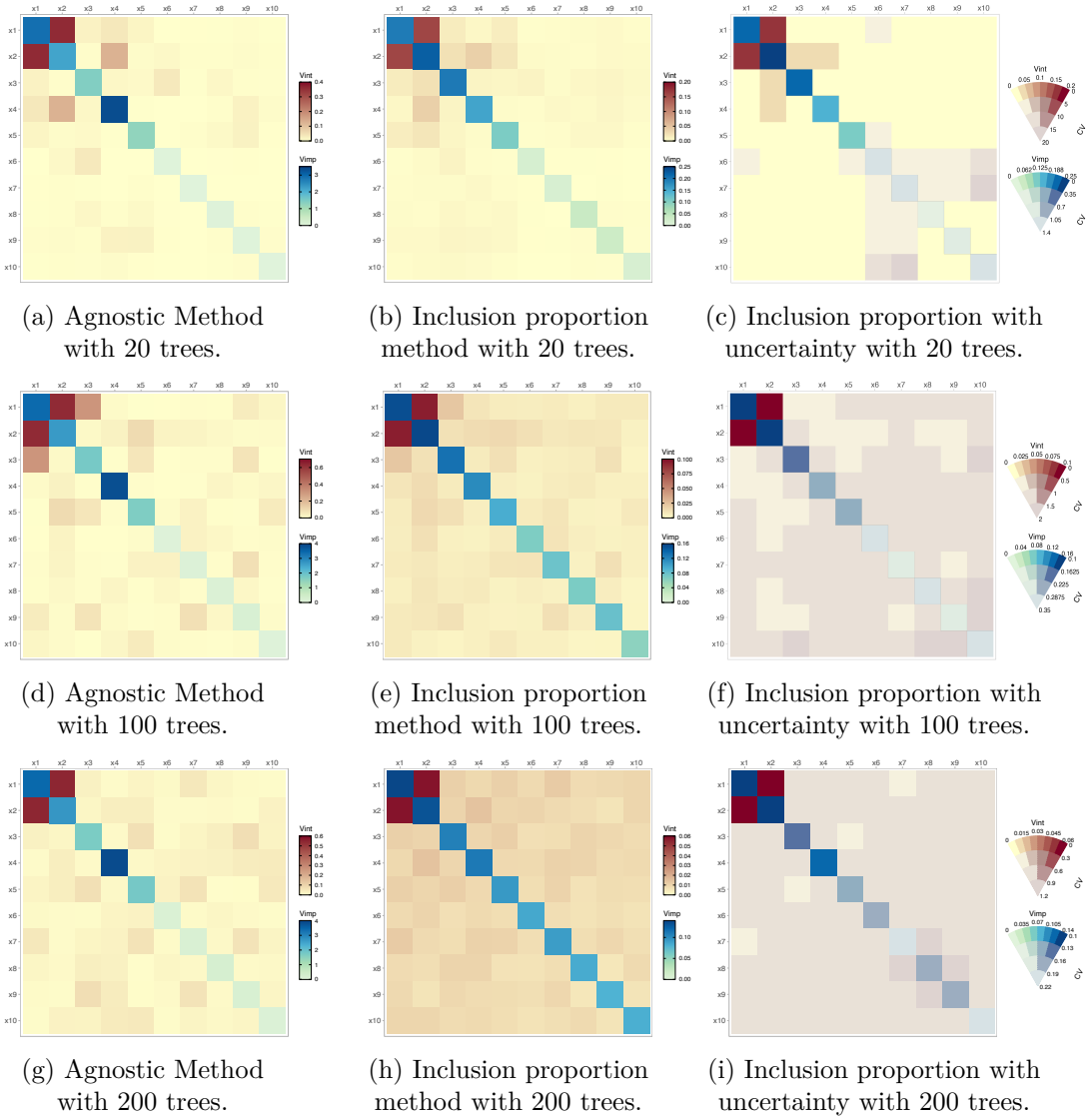
To explore and compare the variable importance and variable interactions, we generate data using the Friedman benchmark equation (Friedman, 1991):

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon$$
$$where \;\; x_j \sim U(0,1), j = 1, 2, \ldots, 10; \;\; \epsilon \sim N(0,1).$$

We simulate 250 observations and fit a BART model using the `dbarts` `R` package, using the default number of iterations (1000) and burn-in (100). We then set the number of trees to be 20, 100, and 200 to evaluate how well the BART model can capture the importance and interactions. There are five important variables and

an interaction between x_1 and x_2 in Equation 3.6, and five additional predictors $x_6, x_7, \ldots x_{10}$ unrelated to the response.

In the first column of Figure 3.11 (panels (a), (d), and (g)) we use the alternative agnostic permutation approach for measuring importance and Friedman's H-statistic to obtain the interaction measures. In the second column (panels (b), (e), and (h)) we calculate the standard BART model variable inclusion proportion for the importance and interactions. Finally, in the third column (panels (c), (f), and (i)), we display the same information as in the second column but with uncertainty included, in this case via the coefficient of variation. For each row of Figure 3.11 we set the number of trees to 20, 100, and 200.



(a) Agnostic Method with 20 trees.

(b) Inclusion proportion method with 20 trees.

(c) Inclusion proportion with uncertainty with 20 trees.

(d) Agnostic Method with 100 trees.

(e) Inclusion proportion method with 100 trees.

(f) Inclusion proportion with uncertainty with 100 trees.

(g) Agnostic Method with 200 trees.

(h) Inclusion proportion method with 200 trees.

(i) Inclusion proportion with uncertainty with 200 trees.

Figure 3.11: Comparison of different methods to determine importance and interactions in a BART model with 20, 100, and 200 trees in the first, second, and third rows respectively.

Using the alternative agnostic method (first column) the five important variables are identified with x_4 being ranked as the most important and the interaction between x_1 and x_2 is prominent. This remains consistent, regardless of the number of trees used when building the model. When using the inclusion proportions

(second column) the interaction between x_1 and x_2 is strong and individually x_1 and x_2 are the most important. In (b) the five important variables are identified. However, as the number of trees increases (see (e) and (h)) variables $x_6, \ldots, x_{10}$ are incorrectly designated as important. Spurious values are measured for both importance and interactions when increasing the number of trees. Examining the VSUPs (third column) the interaction between x_1 and x_2 is prominent and the five important variables are again evident. Increasing the number of trees has the effect of increasing the relative uncertainty for the spurious values and therefore, highlights the variables of interest. For example, if we compare panels (e) and (f) each based on 100 trees, we see that most of the spurious importance and interaction values in (e) have a moderate degree of relative uncertainty in (f).

It is worth noting that for 20, 100, or 200 trees, although the agnostic method had relatively consistent results, this method may not be computationally practical as it is a slow calculation which gets compounded by the increase in trees. Additionally, the agnostic approach would have be repeated multiple times to allow a measure of uncertainty to be obtained. Conversely, calculating the inclusion proportion is quick. For example, calculating the inclusion proportion for importance and interactions for when the number of trees is 20 (as in panels (b) and (c)) took approximately 1.5 seconds on a MacBook Pro 2.3 GHz Dual-Core Intel Core i5 with 8GB of RAM. Whereas, using the agnostic approach to measure the importance and uncertainty (as in panel (a)) took approximately 43 seconds on the same machine. When viewed with the uncertainty included, the inclusion proportion method performs well when compared to the agnostic method, particularly when the number of trees is low.

3.5 Case Study: Seoul Bike Sharing Data

In this section we apply our methods on a larger real-world data set. Here we examine and create Visualisations concerning bike sharing data from Seoul, South Korea (Sathishkumar, 2020). The data contains 14 features and includes weather data (for example, humidity, rainfall, snowfall, and several others), the time of the bike rental (in seasons, months, and days), and some local information (such as if the day of rental was a holiday), with the total number of bikes rented per day

as the response. The original data contained 8760 hourly observations which we summarise to obtain the daily counts. For a full description of the data see the Supplementary Materials. The data has been previously studied in Sathishkumar and Yongyun (2020a) and Sathishkumar et al. (2020) who found that the temperature of the day was an important factor for predicting the total number of rentals. Sathishkumar and Yongyun (2020b) also found that the individual month and season play a significant role in predicting bike rentals.

For our study we fit a BART model, using the `BART` package, with 1000 iterations, a burn-in of 100, and 100 trees, with the goal of investigating which of the predictor variables has a significant impact on the response. We apply a cube root transformation to the response as initially the residuals displayed some evidence of non-normality. As mentioned in Section 3.2.2 on factor dummy variables, we perform an aggregation of the dummy variables' inclusion proportions for both the importance and the interactions so these metrics can be assessed on the entire factor. The variables treated as factors in the data are Month (the month of the year a bike is rented), Season (season of the year a bike is rented), Wkend (if the day of bike rental is a weekend or not), and Holiday (if the day of bike rental is a public holiday or not).

To begin Figure 3.12 shows the model's diagnostics to assess the stability of the model fit. The top two rows indicate a reasonable performance of the residuals with a moderately stable convergence of the residual standard deviation. The black vertical line in the trace plot indicates the separation between the pre and post burn-in period. The bottom row shows that the model fits the training data well and that the Month is clearly the most important variable for predicting the count of bikes rented. This makes intuitive sense as more bikes are rented during the months with better weather, hence the variable Month should have a significant impact on the response. However, in the bottom right panel, we can see that Month has a large 25-75% quantile interval when compared to the other variables. The second most important variable is Season followed by Temp (average daily temperature in $^{\circ}C$).

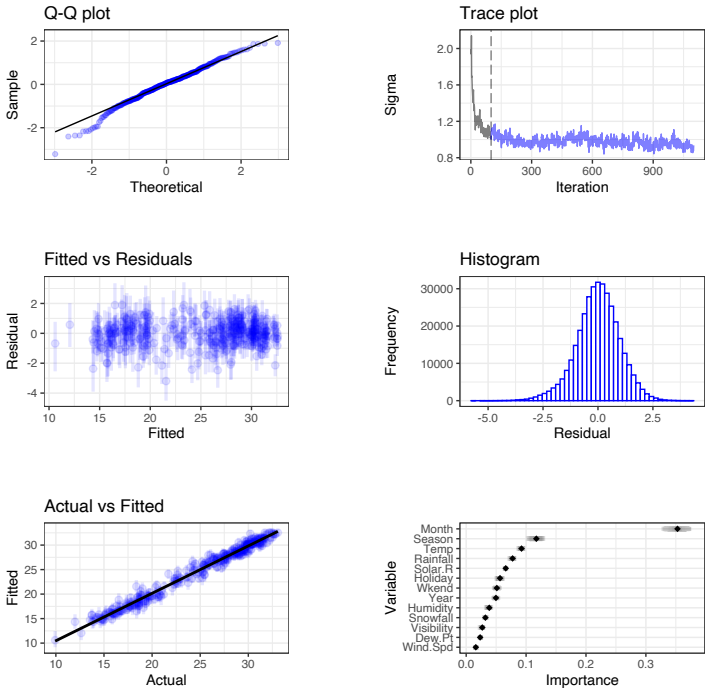
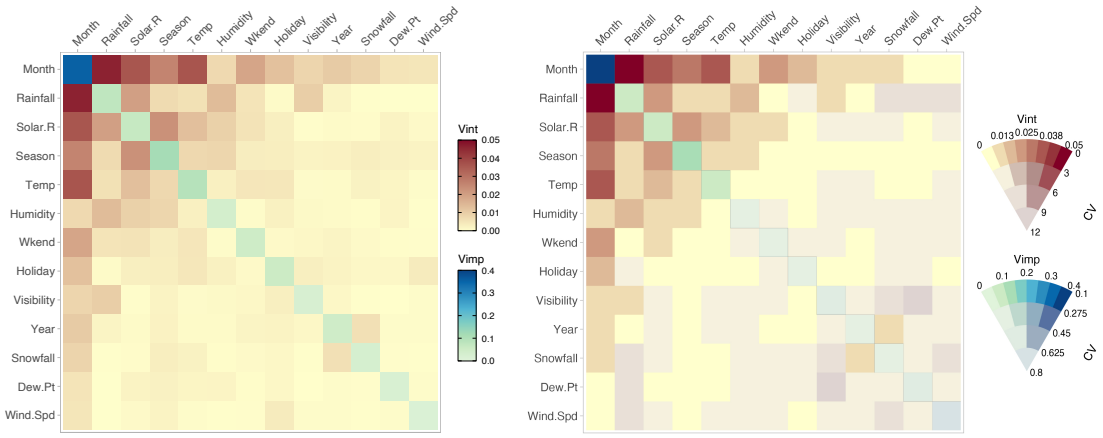


Figure 3.12: General diagnostic plots for a BART regression fit on bike sharing demand data. Top left: A QQ-plot of the residuals after fitting the model. Top right: σ by MCMC iteration. Middle left: Residuals versus fitted values with 95% credible intervals. Middle right: A histogram of the residuals. Bottom Left: Actual values versus fitted values with 95% credible intervals. Bottom right: Variable importance plot with 25 to 75% quantile interval shown. We can see in the bottom left panel that the model fits the training data reasonably well, with a good convergence seen in the top right panel.

We explore the impact of the variables on the response by examining the importance and interactions jointly in the variable importance and variable interaction plots of Figure 3.13. For illustration purposes only we show the plot without uncertainty in the left panel and with uncertainty on the right (as before we use the coefficient of variation).



(a) (b)

Figure 3.13: In (a) Variable importance and interaction plot without uncertainty. In (b) the same values are shown but with the uncertainty included by use of a VSUP. In (b) we can see that the interaction values between Month and several other variables have a low coefficient of variation associated with them.

In Figure 3.13 (a) we observe a strong interaction between the variable Month and several others, notably; Rainfall (in mm), Solar.R (Solar radiance in mJ/m^2), Season, and Temp. These interaction effects are somewhat intuitive, as we would expect the weather (that is, rainfall and temperature) to be vastly different for each month and season. For example, during Winter months, the number of bike rentals is significantly lower due to bad weather. The strongest interaction can be seen between Month and Rainfall. In Figure 3.13(b) many of the low importance and interaction scores have high relative uncertainty, so the viewer's attention is drawn to the interesting variables. The most important variable Month remains important relative to its uncertainty. Equally, the strong interactions observed in (a) between Month and several others have a low associated variation in (b). In (a) all variables except Month have similar importance scores, but relative to uncertainty, the importance of the last seven variables (Humidity to Wind.Spd) is reduced, represented by greeny-grey colours along the diagonal (b). The interactions between these variables are mostly low and/or with high relative uncertainty, the interaction between Snowfall and Year being an exception.

In Figure 3.14 we take a deeper look at the structure of the trees for a selected iteration. As before we choose the iteration with the lowest residual standard deviation. As with the importance and interactions, by default we recombine the categorical variables to display the entire factor. With such a large number of predictors, it can become challenging to effectively display a distinguishable hue for each when plotting the trees. To combat this we can select the most interesting variables observed in Figure 3.13 and highlight them by using bright discernible colours. To aid in efficient examination, we sort the trees by frequency of tree type and remove the stumps.

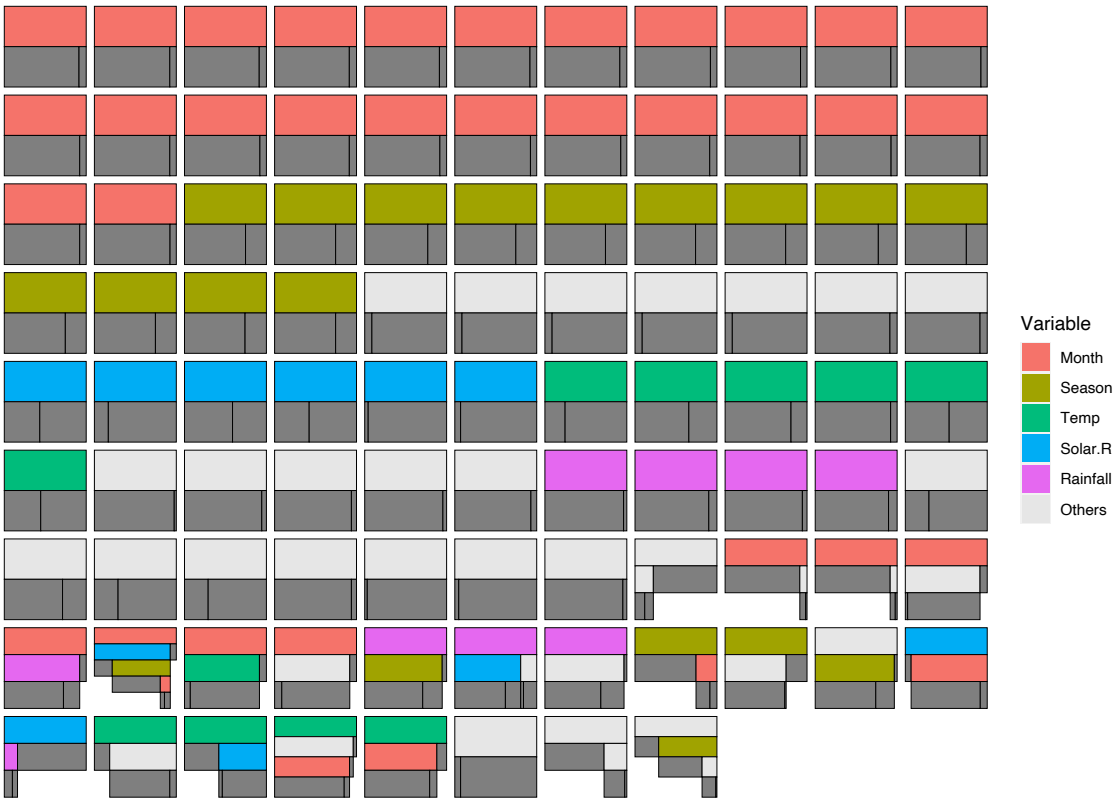


Figure 3.14: All trees from a selected iteration, highlighting the most interesting variables and sorted by the frequency of tree type. In this case, the terminal nodes are coloured dark grey and the stumps have been removed.

In this iteration we can see that the most common tree is a single binary split with Month as the parent. It should also be noted that Month is chosen as the root parent more frequently than any other variable and also appears deeper in several

other trees and is subsequently the most common variable found in this iteration. The previously noted interactions between Month and the other variables can be observed in the lower portion of the plot. We can also see in Figure 3.14 that for the variable Month, most of the observations fall into a single terminal node, making one terminal node much larger than the other. Upon further investigation, the ensemble is commonly splitting between the warmer and colder months for this variable, such that the observations corresponding to January and February (the coldest months with the fewest bike rentals) comprise the smaller terminal node.

We employ our MDS plot in Figure 3.15 to help find outliers. Here we can see that each observation has moderate uncertainty, represented by the surrounding 95% uncertainty ellipses. We have highlighted observation 347 which lies slightly farther away from the group. Inspecting this observation in the data tells us that this observation corresponds to bike rentals on December 24^{th}, which is a public holiday. Bike rentals were well below average for this day, particularly for a public holiday, which has usually high bike rentals. The temperature on this day was also well below average. This may indicate as to why this particular observation lies slightly farther from its group.

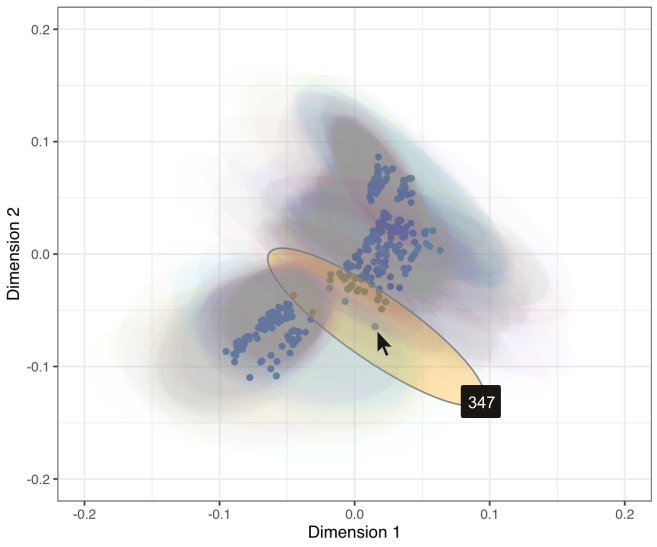


Figure 3.15: MDS plot of a BART fit on the Seoul bike sharing data. The observations appear to have a moderate degree of uncertainty. Observation 347 (highlighted) appears to be an outlier as it lies slightly farther away from its group.

To summarise, we have used our Visualisations to identify and examine variables associated with the prediction of bike rentals in Seoul, South Korea. Our approach allowed us to examine the overall model fit and how individual and pairs of variables impact on the fit. Through our tree-based plots we can examine the inner structure of our fit. Specifically, we found the month a bike was rented was ranked the most important variable. Our methods rated Season as an important predictor, agreeing with previous studies (Sathishkumar and Yongyun, 2020b). We also find Temperature to be important, again verifying the findings of Sathishkumar and Yongyun (2020a) and Sathishkumar et al. (2020). These findings indicate that bike rentals are significantly affected by weather conditions, as during periods of cold, wet, or snowy weather, bike rentals are reduced.

3.6 Discussion

We have presented new and informative Visualisations for posterior evaluation of BART models. We extend the traditional method of assessing variable importance and variable interactions by including the uncertainty that comes with Bayesian models in our point plots and heatmaps that feature the value suppressing uncertainty palettes methods of Correll et al. (2018). With our tree-based plots in Section 3.3.2 we can examine the structure of the decision trees that are created when building the model as well as providing useful summaries of tree types by way of grouping tree structures by different metrics. We display outlier detection methods by way of an interactive multidimensional scaling plot in Section 3.3.3 to provide an in-depth examination of a model's fit. Finally, we provide a selection of enhanced model diagnostic plots in Section 3.3.4, which are practical for assessing a model fit via a suite of plots that visualise aspects of a model such as stability, tree acceptance rate, average number of nodes, and average tree depth plots. These plots also provide a useful summary of the overall model fit via convergence, residual, and Q-Q plots (for regression), and ROC, precision-recall, and confusion matrices (for classification). Our approach is simple to use, adaptable, customisable, and can be useful for comparing different BART model fits.

Our importance and interaction plots can be useful in determining which variables have the greatest impact on the response and the inclusion of uncertainty can

help in deciding if a given variable's importance is worthwhile. A drawback to this method is that the use of inclusion proportions as an importance/interaction measure relies on the splitting rules in the model. Since BART chooses the splitting rule uniformly across all variables, non-important variables can be included. This effect can be mitigated by selecting a smaller number of trees, however this may limit the predictive performance of the model, as noted by Chipman et al. (2010). The examples of Section 3.4 show that using the proportions alone as an importance measure can be misleading, but that the use of a VSUPs with relative uncertainty provide a correction.

A current drawback occurs when the number of trees and/or MCMC iterations is large, so that the computational time to build the data frame of trees used for producing these visualisations can vary, depending on the `R` package used. For example, a model with 20 trees and 500 MCMC iterations took approximately 8.2, 9.2, and 90 seconds for a `BART`, `dbarts`, and `bartMachine` fit, respectively, on a MacBook Pro 2.3 GHz Dual-Core Intel Core i5 with 8GB of RAM. The disparity between `bartMachine` and the other packages is due to the way `bartMachine` uses a `Java` back-end to extract the raw node data from the model. Although some steps were taken to speed up this process, it remains largely outside of our control.

Our methods are flexible and can be easily extended to work with other BART packages, such as `bayesplot` (Gabry et al., 2019), which is an `R` package that provides a large library of plotting methods for use with Bayesian models fits. Similarly, our methods could be extended to incorporate different extensions of BART, such as the methods of Prado et al. (2021) for model trees BART (MOTR-BART). Rather than having a single value for the prediction at the node level, MOTR-BART estimates a linear predictor using the covariates that were used as split variables in the relevant tree. A different method for measuring the importance and interactions could also be investigated for future work, such as DART (Linero, 2018), which modifies a BART model by placing a Dirichlet hyper-prior on the splitting proportions of the regression tree prior. When using DART, Linero (2018) recommend selecting predictor variables from a so-called median probability model (Barbieri and Berger, 2004) to conduct variable selection, where the median probability model is defined as a model containing variables whose poste-

rior inclusion probability is at least 50%. Alternatively, Shapley values (Shapley, 1997) could be used to measure importance.

CHAPTER 4

vivid: An R package for Variable Importance and Variable Interactions Displays for Machine Learning Models

We present `vivid`*, an R package for visualizing variable importance and variable interactions in machine learning models. The package provides a range of displays including heatmap and graph-based displays for viewing variable importance and interaction jointly and partial dependence plots in both a matrix layout and an alternative layout emphasizing important variable subsets. With the intention of increasing a machine learning models' interpretability and making the work applicable to a wider readership, we discuss the design choices behind our implementation by focusing on the package structure and providing an in-depth look at the package functions and key features. We also provide a practical illustration of the software in use on a data set.*

4.1 Introduction

Our motivation behind the creation of the `vivid` package is to investigate ML models in a way that is simple to understand while also offering helpful insights

into how variables affect the fit. We do this through the use of heatmaps, network graphs, and both a generalized pairs plot style partial dependence plot (PDP) (Friedman, 2000) and a space saving PDP based on key variable subsets. While the techniques and fundamental goals of these visualizations have been discussed in Inglis et al. (2022), we focus here on the implementation details of the package by providing a complete listing of the functions and arguments included in the vivid package with further examples indicating advanced usage beyond that previously shown. In this work we examine the decisions made when designing the package and provide an in-depth look at the package functions and features with the intention of making the work applicable to a larger readership. This article outlines the general architectural principles implemented in `vivid`, such as the data structures we use and data formatting, function design, filtering techniques, and more. We illustrate each function by way of a practical example. Our package `vivid` is available on the Comprehensive R Archive Network at `https://cran.r-project.org/web/packages/vivid` or on GitHub at `https://github.com/AlanInglis/vivid`.

In recent years machine learning (ML) algorithms have emerged as a valuable tool for both industry and science. However, due to the black-box nature of many of these algorithms it can be challenging to communicate the reasoning behind the algorithm's decision-making processes. With the need for transparency in ML growing it is important to gain understanding and clarity about how these algorithms are making predictions (Antunes et al., 2018; Felzmann et al., 2019). Many R packages are now available that aid in creating interpretable machine learning (IML) models such as `iml` (Molnar et al., 2018), `DALEX` (Biecek, 2018), and `lime` (Hvitfeldt et al., 2022). For a comprehensive review of IML see Molnar (2022), and Biecek and Burzykowski (2021).

How we choose to visualize aspects of the model output is of vital importance in how a researcher can interpret and communicate their findings. Consequently model summaries such as variable importance and variable interactions (VImp and VInt; together we term these VIVI) are frequently used in various fields to comprehend and explain the hidden structure in an ML fit. In ecology they are employed to determine the causes of ecological phenomena (e.g. Murray and Con-

ner, 2009); in meteorology VImp measures and partial dependence plots are used to examine air quality (e.g. Grange et al., 2018); in bioinformatics, understanding gene-environment interactions have made these measures an important tool for genomic analysis (e.g. Chen and Ishwaran, 2012).

In Table 4.1 we summarize VIVI measures and visualizations provided by a selection of R packages. VIVI measures from fitted ML models fall into two categories; model specific (embedded) methods or model agnostic methods. In embedded methods the variable importance is incorporated into the ML algorithm. For example random forests (RF; Breiman, 2001) and gradient boosting machines (GBM; Friedman, 2000) use the tree structure to evaluate the performance of the model. Bayesian additive regression tree models (BART; Chipman et al., 2010) also use an embedded method to obtain VIVI measures by looking at the proportion of splitting rules used in the trees. Specifically for random forests, the `randomForestExplainer` package (Paluszynska et al., 2020) provides a set of tools to understand what is happening inside a random forest and uses the concept of minimal depth Ishwaran et al. (2010) to assess both importance and interaction strength by examining the position of a variable within the trees. For gradient boosted machines (GBMs) the `EIX` (Maksymiuk et al., 2021) package can be used to measure and identify VIVI and visualize the results.

Package	Package Description	Visualizations Description
`vivid`	Contains a suite of plots for viewing VIVI jointly and the partial dependence. Both model-agnostic and model-specific methods are catered for.	Heatmaps and network plots to view VIVI jointly. Univariate, bivariate PDPs, ICE curves with functionality to plot all pairs and a PDP in a "zenplots" style. Built with `ggplot2`.
`vip`	A general framework for constructing VImp plots from various types of ML models in R. Both model-agnostic and model-specific methods are catered for.	Has built-in `ggplot2` functionality to display VIVI measures. Also provides univariate PDPs and ICE curves and ability to plot Shapley values.
`iml`	A general framework for analysing the behavior of ML models. Includes model-agnostic VIVI measures.	Ability to plot VIVI measures using lollipop, dot, and barplots. Also includes univariate and bivariate PDPs, ICE curves, LIME, and Shapley visualizations. Built with `ggplot2`.
`flashlight`	A general framework for analyzing the behavior of ML models. Includes model-agnostic VIVI measures.	Ability to plot VIVI measures using barplots. Includes univariate and bivariate PDPs, ICE curves, Global surrogate, and SHAP visualizations. Built with `ggplot2`.
`DALEX`	A general framework for analyzing the behavior of ML models. Includes model-agnostic VImp measures.	Contains a suite of visualizations including Ceteris Paribus, Shapley, PDPs, model performance, and diagnostic plots. Built with `ggplot2`.
`lime`	A general framework for fitting a local interpretable model. Includes model-agnostic VImp measures.	Ability to create VImp and model visualizations using barplots and heat maps. Can also create interactive plots. Built with `ggplot2`.
`randomForestExplainer`	Contains a set of model-specific tools to determine which random forests variables are most important. Can assess VIVI	Ability to create VIVI plots displaying the mean minimal depth distribution and conditional minimal depth. Can also display multi-way importance, pairs plots containing different metrics, and Bivariate PDP. Built with `ggplot2`.
`EIX`	Contains a set of model-specific tools to determine which GBM variables are most important. Can assess VIVI	Ability to create VIVI plots using lollipops, barplots, and heatmaps. Can also display dot and radar plots. Built with `ggplot2`.
`varImp`	Computes model-specific random forest VImps for the conditional inference random forest (cforest) of the `party` package.	None available.
`randomForest`	Used to build random forest models. Can assess VImp.	Offers VImp, error rate, and univariate PDPs. Built using base `R`.
`bartMachine`	Used to build Bayesian additive regression tree models. Can assess VIVI.	Ability to plot VIVI measures with uncertainty included using barplots. Also includes a suite of model diagnostic plots and univariate PDP. Built using base `R`.
`pdp`	A general framework for constructing PDPs from various types machine learning models.	Can plot univariate, bivariate, and trivariate PDPs and ICE curves. Built with `ggplot2`.
`ICEbox`	Used to create Individual Conditional Expectation (ICE) plots.	Can plot univariate and bivariate PDPs and ICE curves. Built with `ggplot2`.

Table 4.1: Summary of a selection of R packages that can be used to assess the variable importance, variable interactions, or partial dependence and if these metrics are model-specific or model-agnostic. A brief description of available visualizations for evaluating model behavior is also provided. Our `vivid` package differentiates itself by allowing the VIVI measures to be viewed jointly and allowing all pairs of PDPs to displayed in a single plot. For more on the `lime` and `varImp` packages see Hvitfeldt et al. (2022) and Probst (2020) respectively.

Model-agnostic methods are techniques that can, in principle, be applied to any ML algorithm. Agnostic methods not only provide flexibility in relation to model selection but are also useful for comparing different fitted ML models. An example of a model agnostic approach for evaluating VImps is permutation importance (Breiman, 2001). This method calculates the difference in a model's predictive performance following a variable's permutation; implementations are available in `iml`, `flashlight`, `vip` (Greenwell and Boehmke, 2020), and `DALEX` packages. For VInts, Friedman's H-statistic (Friedman and Popescu, 2008) is an agnostic interaction measure derived from the partial dependence by comparing a pair of variables' partial dependency with their marginal effects. Packages `iml` and `flashlight` provide implementations.

Partial dependence plots (PDPs) were first introduced by Friedman (2000) as a model agnostic way to visualize the relationship between a specified predictor variable and the fit, averaging over other predictors' effects. Similar to PDPs, individual conditional expectation curves (ICE; Goldstein et al., 2015) show the relationship between a specified predictor and the fit, fixing the levels of other predictors at those of a particular observation. PDP curves are then the average of the ICE curves over all observations in the dataset. R packages offering PDPs include `pdp`, (Greenwell, 2017), `iml`, and `DALEX`; the package `ICEbox` (Goldstein et al., 2015) provides ICE curves and variations.

In `vivid` we provide a suite of functions (see Table 4.2) for calculating and visualizing variable importance, interactions and the partial dependence. Our displays conveniently show (both model specific and agnostic) VImp and VInt jointly using heatmaps and network graphs, thus providing a more informative picture identifying relevant features. Our generalized PDP (GPDP) displays partial dependence plots in a matrix layout combining univariate and bivariate partial dependence plots with variable scatterplots. We furthermore provide a more compact version of the GDPD, the so-called zen-partial dependence plot (ZPDP) consisting only of those bivariate partial dependence plots with high VInt. All of our displays are designed to quickly identify how variables, both singly and jointly, affect the fitted response and can be used for regression or classification fits. As the output of our displays are `ggplot2` objects (Wickham, 2016), they are easily customizable and

provide the flexibility to create custom VIVI visualizations.

Function	Description	Type
`vivi`	Create a VIVI matrix of class `vivid`	VIVI construction
`vividReorder`	Reorders a square matrix so high VIVI values are pushed to the top left of the matrix	VIVI construction
`CVpredictfun`	Predict function	VIVI construction
`viviHeatmap`	Heatmap plot of VIVI values	Visualization
`viviNetwork`	Network plot of VIVI values	Visualization
`pdpVars`	Univariate partial dependence plot with ICE curves displayed as a grid	Visualization
`pdpPairs`	Pairs plot showing bivariate PDP, ice/univariate PDP, and data	Visualization
`pdpZen`	A zigzag expanded navigation plot (zenplot) displaying partial dependence values	Visualization
`zPath`	Constructs a zenpath for connecting and displaying pairs to be used with `pdpZen`	Utility
`as.data.frame.vivid`	Takes a matrix of class `vivid` and turns it into a data frame	Utility
`vip2vivid`	Takes measured importance and interactions from the `vip` package and turns them into `vivid` matrix which can be used for plotting	Utility

Table 4.2: Summary of functions available in the `vivid` package. The main construction function is `vivi` which is used to calculate the VIVI values for subsequent use in the visualizations.

This paper is structured as follows. First we introduce a dataset and fits models that will be used as examples throughout this paper. Following this, we describe `vivid` functionality for calculating VIVI. We then move on to visualizations and focus on the functionality provided by the two functions `viviHeatmap` and `viviNetwork` for displaying VIVI, and two functions for displaying PDPs namely, `pdpPairs` and `pdpZen`. Finally we provide some concluding discussion.

4.2 Example: Data and Models

The well-known Boston housing data (Harrison Jr and Rubinfeld, 1978) from the R package `MASS` (Venables and Ripley, 2002) concerns prices of 506 houses and 14 predictor variables including property attributes such as number of rooms and

social attributes like crime rate and pollution levels. The response is the median value of owner-occupied homes in $1000s (medv).

We first fit a random forest (using the `randomForest` package). In order to avail of embedded variable importance scores for the random forest, the `importance` argument must be `TRUE`.

```
library("randomForest")
library("MASS")
set.seed(1701)
data("Boston")

rf <- randomForest(medv ~.,
                   data = Boston,
                   importance = TRUE)
```

Next we fit a gradient boosted machine (using the `xgboost` package). For the GBM we set the maximum number of boosting iterations, `nrounds`, to 100 as no default is provided in `xgboost`.

```
library("xgboost")
gbst <- xgboost(data = as.matrix(Boston[,1:13]),
                label =  as.matrix(Boston[,14]),
                nrounds = 100)
```

In the following sections we will explain how aspects of the two fits can be compared with `vivid` software. We will also explain aspects of our software design with reference to these fits.

4.3 Calculating VIVI

The first step in using `vivid` is to calculate variable importance and interactions for a model fit. The `vivi` function calculates both of these, creating a square, symmetric matrix containing variable importance on the diagonal and variable

interactions on the off-diagonal. Required inputs are a fitted ML model, a data frame on which the model was trained, and the name of the response variable for the fit. The returned matrix has importance and interaction values for all variables in the supplied data frame, other than the response. Variables that are not used by the supplied ML fit will have their importance and interaction values set to zero. Our visualizations functions `viviHeatmap` and `viviNetwork` are designed to show the results of a `vivi` calculation, but will work equally well for any square, symmetric matrix with identical row and column names. Note, the symmetry assumption is not required for `viviHeatmap`. `viviNetwork` uses interaction values from the lower-triangular part of the matrix only.

The code snippet below shows the creation of a `vivid` matrix for the random forest fit. For clarity, we include all of the `vivi` function arguments for the random forest fit, though only the first three arguments are required. Other arguments will be described in the Section on `vivid` matrix additional arguments.

```
library("vivid")

set.seed(1701)
viviRf <- vivi(fit = rf,
               data = Boston,
               response = "medv",
               reorder = FALSE,
               normalized = FALSE,
               importanceType = 'agnostic',
               gridSize = 50,
               nmax = 500,
               class = 1,
               predictFun = NULL)
```

In the absence of any model-specific importance measure we use an agnostic permutation method described by Fisher et al. (2019) to obtain the variable importance

scores. In this method a model error score (root mean square error) is calculated, then each feature is randomly permuted and the model error is re-calculated. The difference in performance is considered to be the variable importance score for that feature.

The `vivi` function calculates importance using an S3 method called `vividImportance`. We provide methods for `randomForest`, `ranger` (Wright and Ziegler, 2017), `mlr` (Bischl et al., 2016), `mlr3` (Lang et al., 2019), and `tidymodels` (Kuhn and Wickham, 2020) to access embedded model-specific measures. When `vivi` is provided with a model fitted using one of these packages, importance defaults to the embedded method, as set when the model was fit. By specifying `importanceType = "agnostic"` in the call to `vivi` as in the example above, agnostic importance is calculated instead. If the model fit offers more than one embedded importance measure, these may be selected by specifying suitable values to `importanceType`. `vivid` relies on the package `flashlight` package to calculate agnostic importance via `flashlight::light_importance` which currently works for numeric and numeric binary responses only.

For variable interactions, we use the model-agnostic Friedman's H-statistic to identify any pairwise interactions. As discussed in Inglis et al. (2022), we recommend the unnormalized version of the H-statistic which prevents detection of spurious interactions which can occur when the bivariate partial dependence function (used in the construction of the H-statistic) is flat. In the case of a binary response classification model, we follow Hastie et al. (2009) and compute the H-statistic and partial dependence using the logit scale.

The `vivi` function calculates interactions using an S3 method called `vividInteraction`, which again relies on the `flashlight` package to calculate Friedman's H-statistic via `flashlight::light_interaction`. Friedman's H-statistic is the only interaction measure currently available in `vivid`, though the method of Greenwell et al. (2018) could also be used for this purpose. Embedded interaction measures could easily be encorporated via S3 methods in future.

`flashlight` simplifies the calculation of VIVI values as it allows a custom predict function to be supplied for the calculation of agnostic importance and the

H-statistic; this flexibility means importance and the H-statistic can be calculated for any ML model. We supply an internal custom predict function called `CVpredictfun` to both `flashlight::light_importance` and `flashlight::light_interaction`. `CVpredictfun` is a wrapper around `CVpredict` from the `condvis2` package, which adds an option for the classification to select (via the `class` argument to `vivi`) the class to be used for prediction and calculates predictions on the logit scale by default. `CVpredict` accepts a broad range of fit classes thus streamlining the process of calculating VIVI.

In situations where the fit class is not covered by `CVpredict` (as is the case for the GBM model created from `xgboost`), supplying a custom predict function to the `vivi` function by way of the `predictFun` argument allows the agnostic VIVI values to be calculated. In the code snippet below, we build the `vivid` matrix for the GBM fit by providing a custom predict function. A custom predict function must be of the form given in the code snippet. For brevity we omit some of the optional `vivi` function arguments.

```
pFun <- function(fit, data, ...) predict(fit, as.matrix(data[,1:13]))

set.seed(1701)
viviGBst <- vivi(fit = gbst,
                 data = Boston,
                 response = "medv",
                 reorder = FALSE,
                 normalized = FALSE,
                 predictFun = pFun)
```

4.3.1 `vivid` matrix additional arguments

The `vivi` function takes 10 arguments. Some of these have been discussed above, including `fit`, `data`, `response`, `importanceType`, and `predictFun`. Here we provide a summary of the remaining arguments. First, the `normalized` argument determines if Friedman's H-statistic should be normalized or not (see Inglis et al., 2022, for the pros and cons of each version). The arguments `gridSize` and `nmax`

are used to set the size of the grid for evaluating the predictions and maximum number of data rows to consider, respectively. As the calculation of the H-statistic can be slow (as discussed in Section Calculating VIVI) lowering the grid size can provide a significant speed boost. However this increase in speed can come at the expense of predictive accuracy. Additionally, sampling the data via `nmax` can offer a practical speed boost. The default values for `gridSize` and `nmax` are 50 and 500, respectively. These values were chosen as to provide reasonable computational time while maintaining predictive accuracy. As `nmax` randomly samples rows of the data, to get reproducible results a seed must be set before running the `vivi` function.

When `reorder = TRUE` (the default setting) we apply a seriation technique to reorder the matrix so that both high values of importance and interactions are pushed to the top left of the matrix. This allows the user to quickly identify and highlight the variables with the greatest impact on the response in the model. We use the leaf sort algorithm of (Earle and Hurley, 2015) to generate the reordering; see Inglis et al. (2022) for more details. If `reorder = FALSE`, then the `vivid` matrix is returned in the same order as the variable names in the data set. However, as the output of `vivi` is a matrix, a custom ordering can be easily applied. This will be useful when there is a specific reason for ordering the matrix in a certain way, such as to compare two different fits side by side. `reorder = TRUE` has the same effect as the `vividReorder` function (seen in Table 4.2). Having `vividReorder` as a separate exported function allows the user to reorder any matrix using the same sorting methodology. An example of using the `vividReorder` can be seen in the Heatmap of Variable Importance and Variable Interactions Section.

4.3.2 Speed tests

A drawback of using Friedman's H-statistic as a measure of interaction is that it is a computationally expensive calculation for models that are slower to produce predictions. Consequently, the time taken to build the `vivid` matrix can vary. Figure 4.1 below shows the build time (rounded to the nearest second) averaged over five runs for the creation of a `vivid` matrix with default parameters for different ML algorithms using the Boston Housing data. As the Boston housing

data has 13 predictor variables, Friedman's H-statistic is computed for 91 predictor pairs. The ML algorithms are: GBM, random forest, support vector machine (SVM), neural network (NN), and k-nearest neighbors (KNN). The SVM, NN, and KNN were built using the `e1071` (Meyer et al., 2021), `nnet` (Venables and Ripley, 2002), and `kknn` (Schliep and Hechenbichler, 2016) packages, with the KNN being built through the `mlr3` (Lang et al., 2019) framework. Each of the models were built using their default settings and, for each model fit, the agnostic VImp was measured. The speed tests were performed on both a 2017 MacBook Pro 2.3 GHz Dual-Core Intel Core i5 with 8GB of RAM (MBP 2017 in Figure 4.1) and a 2021 32GB MacBook M1 Pro (MBP 2021 in Figure 4.1). In Figure 4.1 we can see that a NN model created using the `nnet` package was the fastest for both scenarios. Surprisingly, the time to build the `vivid` matrix for a random forest model created from the `ranger` package was over 1.5 times longer than the random forest fit from the `randomForest` package for MBP 2017. Interestingly, repeating these tests for MBP 2021, the `ranger` calculation is reduced by over 60% while the `randomForest` actually increases by 47%.

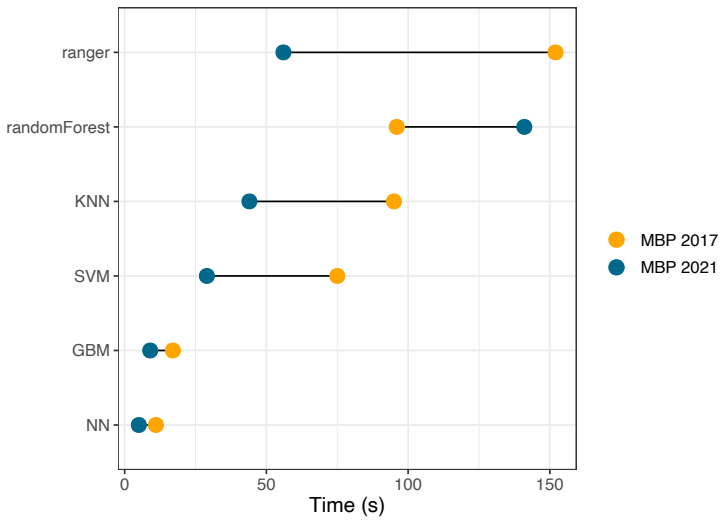


Figure 4.1: Mean time over five runs, on separate MacBook machines, for the creation of a `vivid` matrix for different models.

In all of the `vivid` functions making use of model predictions (namely `vivi`, and

the three variants of partial dependence plots described in the next section) we provide two arguments to reduce the number of predictions necessary. These are `gridSize`, which controls the number of points at which predictions are to be made, and `nmax` which specifies the maximum number of observations for which predictions are to be evaluated.

4.3.3 Alternative construction of a `vivid` matrix

A further way to create a valid `vivid` matrix is to construct one from variable importance and interaction values calculated elsewhere. The package `vip` offers these, for which we provide a convenient translation function. `vip` provides the ability to evaluate interactions using a method called the *feature importance ranking measure* (FIRM; see Greenwell et al., 2018, for more details). The `vip2vivid` function we provide in `vivid` takes VIVI values created in `vip` and turns them into a `vivid` matrix, that can be subsequently used with our plotting tools. For example, in the code below, model-specific VImp and FIRM VInt scores are calculated for the random forest fit, and subsequently arranged into a `vivid` matrix with the VImps on the diagonal and VInts on the off-diagonal. In the first two lines of the code block below, we calculate the model specific VImps and FIRM VInt using the `vip` package, before finally applying our `vip2vivid` function to transform the results into a `vivid` matrix.

```
library("vip")
vipVImp <- vi(rf, method = 'model')
vipVInt <- vint(rf, feature_names = names(Boston[-14]))
vipViviMat <- vip2vivid(importance = vipVImp, interaction = vipVInt)
```

4.4 Heatmap of Variable Importance and Variable Interactions

The `viviHeatmap` function constructs a heatmap displaying both importance and interactions, with importance on the diagonal and interactions on the off-diagonals. In the following examples we display the importance and interaction values obtained from the `vivid` matrix, however we are not limited to displaying these

measures. As the only required input for the `viviHeatmap` is a matrix (not necessarily symmetric), a user may choose to create a matrix containing different measures, such as correlation values in the upper and lower diagonals and the skewness values on the diagonal. Color palettes for the importance and interactions are optionally provided via `impPal` and `intPal` arguments. For the default color palette we choose single-hue, color-blind friendly sequential color palettes from Zeileis et al. (2020), where low VIVI values are represented by low luminance color values and and high VIVI by high luminance colors, which aids in highlighting values of interest.

The ordering of the heatmap is taken from the ordering of the input matrix. As `reorder` was set to `FALSE` when building both the random forest and GBM fit `vivid` matrix, the ordering of the heatmaps matches the variable order in the dataset. This is useful for directly comparing multiple heatmaps, however it does not necessarily lend itself for easy identification of the largest VIVI values. If we were to seriate both `vivi-matrices` separately, we would end up with different optimal orderings for each matrix. An alternative is to create a common ordering by averaging over the two `vivid` matrix objects and applying the `vividReorder` function to the result. Both VIVI matrices are then re-ordered using the newly obtained variable order. The code below shows such a strategy for our chosen model fits.

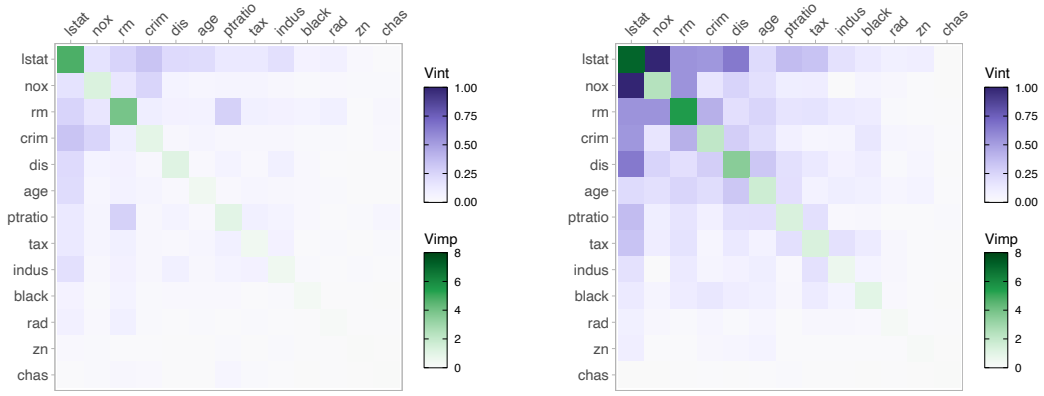
```
viviAvg <- (viviRf + viviGBst) / 2
viviAvgReorder <- vividReorder(viviAvg)
ord <- colnames(viviAvgReorder)
viviRf <- viviRf[ord,ord]
viviGBst <- viviGBst[ord,ord]
```

Arguments `impLims` and `intLims` specify the range of importance and interaction values to be mapped to colors. Default values are calculated from the maximum and minimum VIVI values in the `vivid` matrix. Importance and interaction values falling outside the supplied limits are squished to the closest limit. It can be useful to specify these limits in the situation where there is an extremely large VIVI value

that dominates the display, or where we wish two or more plots to have the same limits for comparison purposes, as in the example below.

```
viviHeatmap(viviRf, angle = 45, intLims = c(0,1), impLims = c(0,8))
viviHeatmap(viviGBst, angle = 45, intLims = c(0,1), impLims = c(0,8))
```

Figure 4.2 shows our improved ordering so that variables with high VIVI values are pushed to the top left of the plots. Filtering can also be applied to the input matrix to display a subset of variables. When compared to the GBM fit in (b), the random forest fit in (a) appears to create weaker interactions and lower importance scores. Both plots identify *lstat* as being the most important. Both fits also show that *lstat* interacts with several other variables. Notably the strongest interaction in both fits are different. Namely $lstat : crim$ (where *crim* is the per capita crime rate by town) for the random forest fit and $lstat : nox$ (where *nox* is parts per 10 million nitrogen oxides concentration) for the GBM fit.



(a) (b)

Figure 4.2: Agnostic variable importance and variable interaction scores for a random forest fit in (a) and GBM fit in (b) on the Boston housing data displayed as a heatmap. The random forest fit appears to produce weaker interactions and lower importance scores when compared to the GBM fit. Both fits identify *lstat* as the most important followed by *rm*. In both fits we can see that *lstat* has numerous interactions with other variables, notably *crim* in the random forest fit in (a) and *nox* in the GBM fit in (b).

To provide a broader context and application of the `vivid` package, in Figure 4.3, we show a comparison of the VIVI values obtained from `vivid` with those obtained from the `randomForesExplainer` package in a `viviHeatmap`. This package uses the concept of minimal depth to assess the importance and interaction values by examining the position of a variable within the trees (for more details see Ishwaran (2007)). In Figure 4.3, the conditional minimal depth (see 3.1 Section of this thesis for more details) is used for the interaction values.

As the `randomForesExplainer` package returns asymmetric interaction values (for example, the interaction between *lstat*:*rm* may have a different value than the interaction between *rm*:*lstat*), the upper and lower triangles of Figure 4.3 differ. A single interaction value could be obtained by averaging over the variable pairs. Additionally, in Figure 4.3 we exclude interactions with themselves, which are measured in the `randomForesExplainer` package (for example *rm*:*rm*). It should also be noted that as lower values are considered to have greater importance/interactions when using the minimal depth method, consequently, the color scale has been inverted in Figure 4.3 to be inline with the previously shown heatmaps. In Figure 4.3, we can see that both *lstat* and *rm* are the most important, which is in agreement with the permutation approach used in Figure 4.2. However, when compared to Friedman's H-statistic in Figure 4.2, we get quite a contrasting result for the interaction values. In Figure 4.3 we can see that there are many more detected interactions among the variables, with the variable pair *rm* and *lstat* having the greatest interaction in both the upper and lower portions of the heatmap. However, in 4.2 the interaction between *lstat* and *rm* is given far less significance. Another stark difference is the asymmetry of the interaction values. For example, *crim*:*ptratio* has a moderate interaction in the upper triangle of Figure 4.3 (having a minimal depth of around 3.5), whereas the interaction between *ptratio*:*crim* in the lower portion of the heatmap displays a lower interaction interaction value.

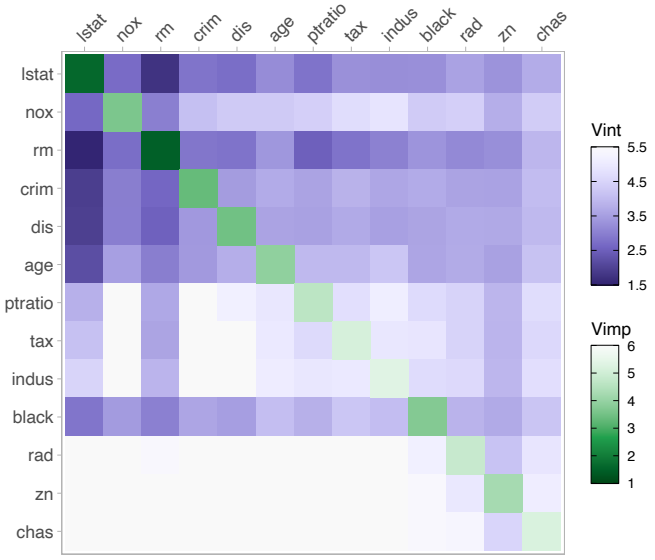


Figure 4.3: Variable importance and interaction values obtained from the `randomForestExplainer` package displayed as a heatmap from `vivid`. The most important variables are *lstat* and *rm*, with these variables displaying the greatest interaction.

4.5 Network of Variable Importance and Variable Interactions

The `viviNetwork` function constructs a network graph displaying both importance and interactions. Similar to the heatmap, this function takes a `vivid` matrix as the only required input and provides a visual representation of the magnitude of the importance and interaction values through the size of the nodes and edges in the graph, in addition to color. In the plot each variable is represented as a node, with its importance being represented through size and color such that larger, darker nodes indicate a higher importance. Each pairwise interaction is represented by a connecting edge, where larger interaction values get thicker, darker edges; Figure 4.4 provides an example. This type of plot benefits from being able to quickly identify the magnitude of the importance and interactions of the variables that have the most impact on the response in an efficient manner. The `viviNetwork` function follows the same convention as the heatmap and allows custom color palettes for the importance and interactions to be provided via the `impPal` and

`intPal` arguments and the range of VIVI values to be mapped to the colors can be specified via the `impLims` and `intLims` arguments.

By default, we choose a circular layout to display the graphs as when coupled with the seriation techniques described previously, variables with high VIVI are grouped in a clock-wise arrangement starting at the top. This arrangement allows for quick identification of variables with high VIVI. Custom layouts are possible by providing a numeric matrix with two columns and one row per node to the `layout` argument. Additionally, any of the layouts available in the `igraph` package (Csardi and Nepusz, 2006) can be passed to `layout`. This can be useful for for producing force-directed layouts, which try to produce a visually appealing graph with minimal cross-over of the edges and organizes edges so that they are of a similar length.

We provide options to filter the graph via the `intThreshold` and `removeNode` arguments. This helps to highlight variables with high VIVI scores, which can be useful in settings with many predictors. The `intThreshold` argument filters edges with weight (i.e., VInt value) below a specified value and `removeNode` removes nodes with no connecting edges after thresholding interaction values. We can optionally cluster similar variables together with respect to their VIVI scores via the `cluster` argument, thereby aiding in the process of highlighting variables of interest. The `cluster` argument can take either a vector of cluster memberships for nodes or an appropriate `igraph` clustering function. We allow for a multitude of clustering and filtering approaches, some of which are shown below.

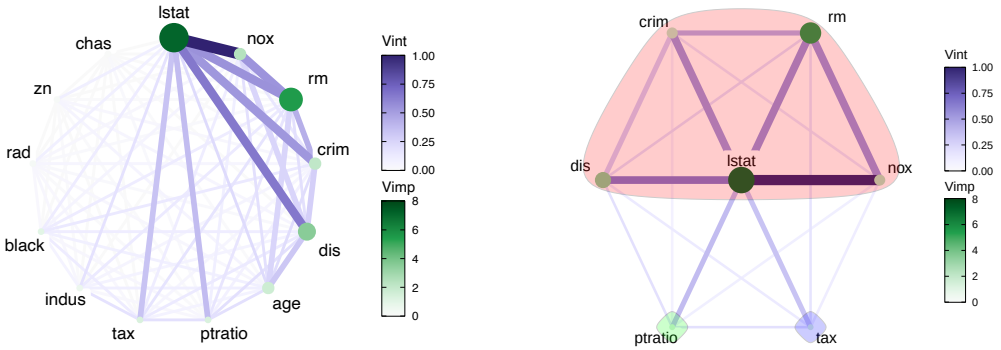
In the interest of space, we only include network plots displaying VIVI values for the GBM fit. In Figure 4.4 we show both a default network plot including all variables in (a) and a filtered and clustered network plot in (b). For the filtered plot we select VIVI values above the median. This selection allows us to focus only on the variables with the most impact on the response. The variables that remain are *lstat*, *nox*, *rm*, *crim*, *dis* (weighted mean of distances to five Boston employment centers), *age* (proportion of owner-occupied units built prior to 1940), and *ptratio* (pupil-teacher ratio by town). We then perform a hierarchical clustering treating variable interactions as similarities, with the goal of grouping together

high-interaction variables. Finally we rearrange the layout using `igraph`. Here, `igraph::layout_as_star` places the first variable (deemed most relevant using the VIVI seriation process above) at the center, which in Figure 4.4 (b) emphasizes its key role as the most important predictor which also has the strongest interactions. The first line in the following code block shows the default network plot for the GBM fit. The remaining code outlines the aforementioned process for the clustered and filtered network.

```
viviNetwork(viviGBst)

intVals <- viviGBst
diag(intVals) <- NA
impTresh <- quantile(diag(viviGBst),.9)
intThresh <- quantile(intVals,.9,na.rm=TRUE)
sv <- which(diag(viviGBst) > impTresh |
              apply(intVals, 1, max, na.rm=TRUE) > intThresh)
h <- hclust(-as.dist(viviGBst[sv,sv]), method="single")

viviNetwork(viviGBst[sv,sv],
            intLims = c(0,1),
            impLims = c(0,8),
            cluster = cutree(h, k = 3), # specify number of groups
            layout = igraph::layout_as_star)
```



(a) (b)

Figure 4.4: Network plots showing VIVI scores obtained from a GBM fit on the Boston housing data. In (a) we display the all values in a circle. In (b) we use a hierarchical clustering to group variable with high VIVI together and rearrange the layout using an `igraph` function.

In Figure 4.4 (a), when displaying all the variables, we can clearly identify which variables have the highest VIVI values. The large darker nodes of *lstat* and *rm* indicate their importance and the dark, thick connecting edge between *lstat* and *nox* tell us that these two variables strongly interact. In (b), after applying a hierarchical clustering on the variables with a VIVI value in the top 10%, we can see the strongest mutual interactions have been grouped together for the GBM fit. Namely, *lstat*, *nox*, *crim*, *rm*, and *dis* are all grouped together. The remaining variables are individually clustered.

We provide a conversion of `vivid` matrix objects to a data frame via an `as.data.frame` method, which facilitates plotting with base R and `ggplot2`. The following code snippet shows the structure of the first four rows of the created data frame for the random forest VIVI matrix, where we can see that each variable pair is represented with their corresponding VIVI value. This data frame can then be easily manipulated and plotted as, for example, a barplot of either VImp or VInt values.

```
class(viviRf)<- c("vivid", class(viviRf))
head(as.data.frame(viviRf), 4)
```

```
#>  Variable_1 Variable_2     Value Measure Row Col
#> 1      lstat      lstat 4.7720394    Vimp   1   1
#> 2        nox      lstat 0.1789794    Vint   2   1
#> 3         rm      lstat 0.2562817    Vint   3   1
#> 4       crim      lstat 0.3371364    Vint   4   1
```

4.6 Partial Dependence and Individual Conditional Expectation Curves

4.6.1 Univariate Partial Dependence Plot

The `pdpVars` function constructs a grid of univariate PDPs with ICE curves for selected variables. We use ICE curves to assist in the identification of linear or non-linear effects. The fit, data frame used to train the model, and the name of the response variable are required inputs. In the code below, we show an example of the partial dependence and ICE curves for each feature from the GBM fit, with output shown in Figure 4.5. We use the custom GBM predict function given previously.

```
pdpVars(data = Boston,
        fit = gbst,
        response = 'medv',
        vars = colnames(viviGBst),
        predictFun = pFun
)
```

All of our PDP variants handle categorical responses and predictors. The color palette is customized via the `pal` argument. In all of our PDPs, this defaults to a diverging palette which accentuates fitted values that differ from the average. Dark red and dark blue are used to indicate high and low values of $\hat{y}$ respectively. The middle values are displayed in yellow. The `nIce` argument specifies the number of ICE curves to be drawn. This is either a single number specifying the number

of observations to be sampled for the ICE curves, or a vector of row indices. The default value for `nIce` is 30, which allows individual curves to be seen.

The ordering of the PDPs is taken from the ordering of variables in the data set, however custom ordering or filtering is obtained via the `var` argument (seen in the code above). As with the construction of the `vivid` matrix, the `gridSize` and `nmax` arguments determine the number of predictions required.

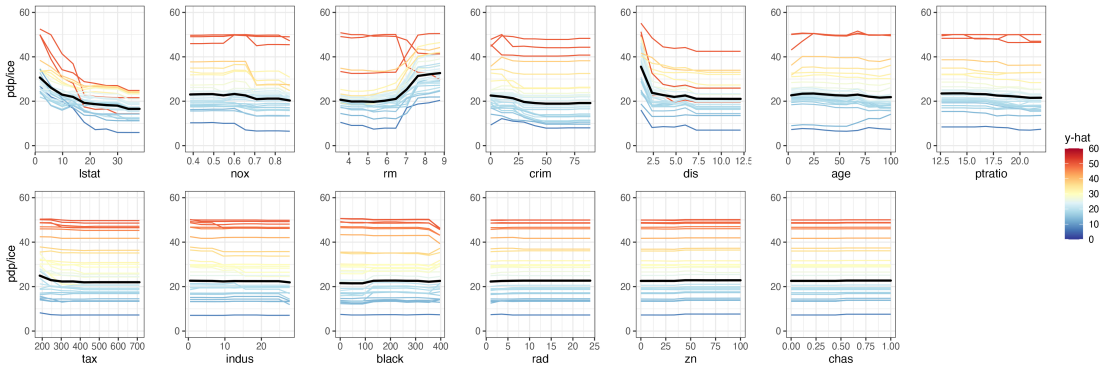


Figure 4.5: Partial dependence plot (black line) with individual conditional expectation curves (colored lines) of a GBM fit on the Boston housing data. The changing partial dependence and ICE curves of *lstat* and *rm* indicate that these variables have some impact on the response.

In Figure 4.5 we can see from the changing PDP and ICE curves that *lstat* and *rm* have the clearest impact on the response, with the predicted median house price being higher for low values of *lstat* and high values of *rm*. Additionally, the predicted median house price appears to be higher for low values of *dis* before leveling off at around 2.5. The remaining variables have generally flat partial dependence and ICE curves.

4.6.2 Generalized Pairs Partial Dependence Plot

The `pdpPairs` function creates a generalized pairs partial dependence plot (GPDP). In our GPDP, we use a matrix layout and plot the univariate partial dependence (with ICE curves) on the diagonal, bivariate partial dependence on the upper diagonal and a scatterplot of the data on the lower diagonal, where all colours are

assigned to points and ICE curves by the predicted $\hat{y}$ value. As with the univariate PDP, the fit, data frame used to train the model, and the name of the response variable are required inputs. However, in the code below we include additional arguments which are described in the following paragraph.

```
filteredVars <- colnames(viviGBst)[1:5]
rmHigh <- sample(which(Boston$rm > mean(Boston$rm)), 25)
lstatLow <- sample(which(Boston$lstat < mean(Boston$lstat)), 25)

set.seed(1701)
pdpPairs(data = Boston,
         fit = gbst,
         response = "medv",
         gridSize = 20,
         nIce = c(rmHigh, lstatLow),
         var = filteredVars,
         convexHull = TRUE,
         fitlims = "pdp",
         predictFun = pFun)
```

In the above code, we filter the plot to display only the interesting variables seen in previous plots by passing a vector of variable names via the `var` argument. In this case, we select the first five variables from our `vivid` matrix. We also chose to display 50 ICE curves, where 25 instances are sampled from rows above the mean value for *rm* and the other 25 are sampled from rows below the mean value for *lstat*. These values were chosen as it seems evident in Figure 4.5 that as the number of rooms increases and as the lower status value of the population decreases, the predicted median house price goes up. The previously mentioned arguments from `pdpVars` for controlling the color palette, grid size, the number of rows of data considered, and the number of ICE curves drawn can equally be applied to our GPDP.

For our GPDP, we follow the general design choices in `vivid` and so provide the

functionality to specify the range of predicted values to be mapped to the colors via the `fitlims` argument. As before the custom limits allow for a direct comparison of different model fits should they require. We set the default fit range for the color map for the GPDP to the range of the collection of PDP surfaces with `fitlims = 'pdp'`. The setting of this argument at its default value allows for maximum resolution of the bivariate PDPs. Since predictions for specific observations and ICE curves could exceed these bounds, the closest value within the color map's bounds is used to allocate colors. Alternatively to set the full range of the data as the limits we can use `fitlims = 'all'`.

In the upper diagonals we exclude extrapolated areas from the bivariate PDPs to prevent interpretation of the PDPs in areas where there are no data. The removal of extrapolated areas is a default setting but can be removed with the argument `convexHull = FALSE`. In this example, we set the grid size equal to 20 (with the default being 10). We increase the value from the default grid size here, despite this increasing processing time since the data set is reasonably small. As with the previous PDPs, we use the custom predict function to generate PDPs for the GBM fit.

In Figure 4.6, in addition to the univariate PDPs, we capture the effects of the variables on the response via the bivariate PDP on the upper-diagonal and the distribution of the data in the lower-diagonal. The scatterplots are useful for determining if any of the variables are highly correlated, as highly correlated variables may spuriously affect the partial dependence and give erroneous results (Apley and Zhu, 2020). Of note in Figure 4.6 are the variables $lstat$ and rm. We can clearly see that when the number of rooms (rm) is high and the percentage of lower status of the population ($lstat$) is low, the predicted $\hat{y}$ median house price value is high. This is exemplified in the changing bivariate PDP.

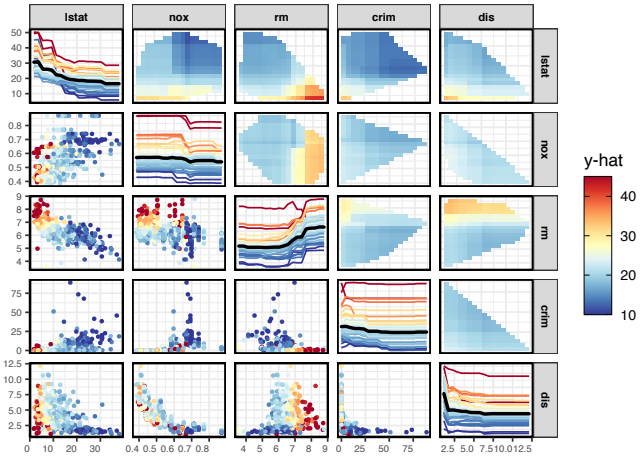


Figure 4.6: Filtered generalized pairs partial dependence plot for a GBM fit on the Boston housing data. From both the univariate and bivariate PDPs, we can see that *lstat* and *rm* have an impact on the response. As *lstat* decreases and *rm* increases, predicted median house price value goes up. The bivariate PDP of *lstat* : *nox* shows that as *nox* increases, the predicted value decreases.

In the case of categorical predictors, the partial dependence for each factor level is shown in the upper-diagonal (for an example of this, see Inglis et al. (2022)). Additionally, in classification scenarios, being able to select specific rows of data to display their corresponding ICE curves can be useful for displaying ICE curves from particular classes.

4.7 Zen Partial Dependence Plots

The `pdpZen` function creates a PDP that utilizes a space-saving method based on graph Eulerians to show the bivariate partial dependence, which we call zen-partial dependence plots (ZPDP). This plot is based on the zigzag expanded navigation plots, known as zenplots (Hofert and Oldford, 2020), available in the `zenplots` package. Zenplots were created to display paired graphs of high-dimensional data focusing on the most important 2D displays. In our adaptation we show bivariate PDPs that focus on the most important interacting variables in a compact zigzag layout, helpful when predictor space is high-dimensional.

The code below illustrates `pdpZen`, here displaying the first five variables from GBM's `vivid` matrix. Later we show an example focusing on high-interacting pairs of variables. We use the same convention as our previous PDPs with regard to color palette and limits, grid size, and the number of rows considered for evaluation. The ZPDP also has a variable rug plot on each axis to avoid interpretation problems that may occur in the presence of skewness.

```
pdpZen(data = Boston,
       fit = gbst,
       response = "medv",
       convexHull = TRUE,
       zpath = colnames(viviGBst)[1:5],
       predictFun = pFun)
```

The argument `zpath` specifies the variables to be plotted, defaulting to all dataset variables aside from the response. In the code above, `zpath` is the vector *lstat*, *nox*, *rm*, *crim* and *dis*. The resulting plot shows the bivariate PDP for every consecutive pair of variables in a zigzag layout. Figure 4.7 shows the resulting visualization.

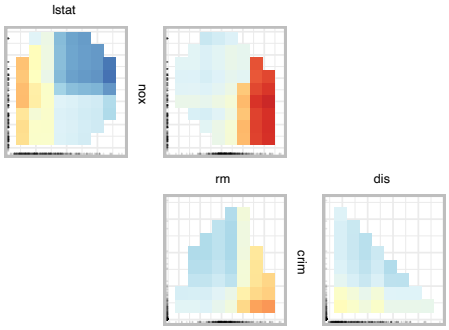


Figure 4.7: Zen partial dependence plot for the GBM fit on the Boston data. Here we display first five variables from the GBM's `vivid` matrix. Only plots for consecutive variables are shown.

4.7.1 Zen-paths

ZPDP are most useful when the bivariate PDPs plotted are selected to be an interesting subset of all pairwise plots. To obtain this subset, we consider a network graph displaying VIVI values, such as that in Figure 4.4 (a). We then filter the edges below a selected interaction value, leaving only highly interacting variable pairs, as in Figure 4.4 (b). Our goal is to then build a ZPDP consisting of the bivariate plots represented by each edge of the thresholded graph. The `zPath` function creates a sequence or sequences of variable paths for use in `pdpZen`.

The `zPath` function takes four arguments. These are: `viv` - a matrix of interaction values, `cutoff` - exclude interaction values below this threshold, `method` - a string indicating which method to use to create the path, and `connect` - a logical value indicating if separate Eulerians should be connected

Two methods are provided, either `"greedy.weighted"` or `"strictly.weighted"`. The first option uses the greedy Eulerian path algorithm of Hurley and Oldford (2011) (available in the `PairViz` package Hurley and Oldford, 2022) for connected graphs. This visits each edge at least once, beginning at the edge with the highest weight and traversing through the remaining edges, giving priority to the highest-weighted edge. Some edges may be visited more than once or additional edges may be visited if the number of nodes in the graph is not even. The second method `"strictly.weighted"` (provided by `zenplot`) visits edges strictly in decreasing order by weight (here the interaction values). If `connect` is `TRUE` the sequences obtained by the strictly weighted method are concatenated to form a single path.

In the code below, we provide two examples of creating zen-paths, from the top 10% of interaction scores in `viviGBst`.

```
intThresh <- quantile(intVals,.9,na.rm=TRUE)
zpGw  <- zPath(viv = viviGBst, cutoff = intThresh,
method = 'greedy.weighted')
zpGw
#>  [1] "nox"     "lstat"   "dis"     "ptratio" "lstat"   "rm"      "crim"
#>  [8] "lstat"   "tax"     "rm"      "nox"
```

```
zpSw <- zPath(viv = viviGBst, cutoff = intThresh, connect = FALSE,
method = 'strictly.weighted')
zpSw
#> [[1]]
#> [1] "nox"   "lstat" "dis"
#>
#> [[2]]
#> [1] "lstat" "rm"    "nox"
#>
#> [[3]]
#> [1] "lstat" "crim"  "rm"
#>
#> [[4]]
#> [1] "ptratio" "lstat"   "tax"
```

Our first created zen-path object, zpGw, uses the greedy.weighted method and visits each edge at exactly once. The second zen-path uses the strictly.weighted method with connect = FALSE. zpSw consists of four unconnected paths. The zenplots for two of these paths are constructed below.

```
pdpZen(data = Boston,
       fit = gbst,
       response = "medv",
       zpath = zpGw,
       convexHull = TRUE,
       predictFun = pFun)

pdpZen(data = Boston,
       fit = gbst,
       response = "medv",
       zpath = zpSw,
```

```
        convexHull = TRUE,
        predictFun = pFun)
```

Note that there are 7 different variables involved in high interactions, which could be displayed in a 7×7 GPDP, showing a total of 21 bivariate PDPs. But only 8 of these have VInt values above the 90% quantile, and Figure 4.8 (b) using the `strictly.weighted` path shows just these bivariate PDPs compact layout. Using the `greedy.weighted` sorting method in (a) produces a smaller, neater plot but at the expense of including some plots that are not particularly interesting (for example the pair *dis* : *ptratio*).

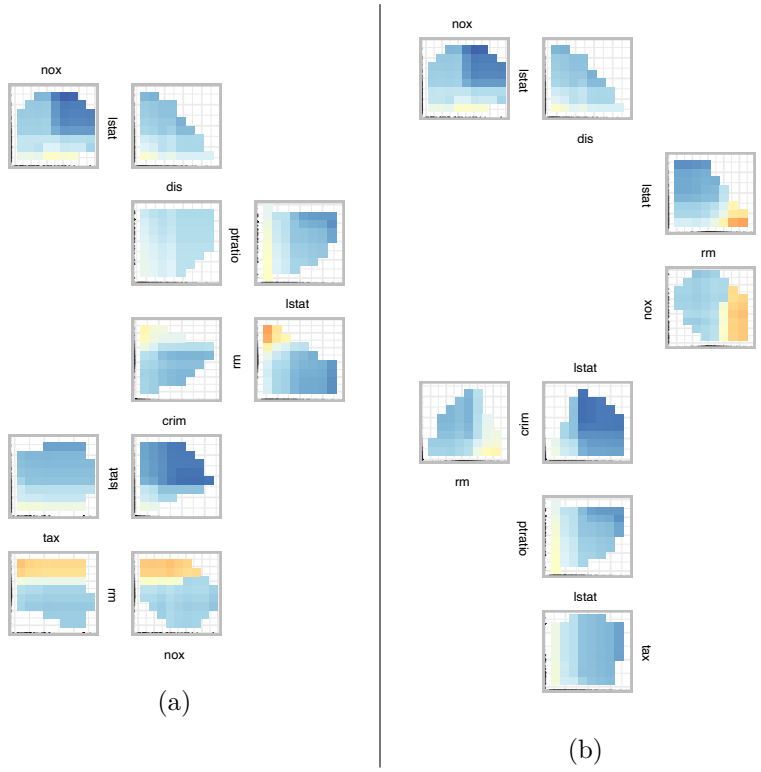


Figure 4.8: ZPDP for a GBM fit on the Boston data. In (a) the zpath is defined by the `greedy.weighted` sorting method. In (b), the sorting method is defined by the `strictly.weighted` method and is unconnected. For low values of *lstat* and and high values of *rm*, predicted median house price value increases.

4.8 Summary

We have presented a detailed exposition of our R package `vivid` which contains a suite of integrated functions implementing algorithms and novel visualizations for exploring variable importance and variable interactions in machine learning models. Our techniques are intuitive, adaptable, easy to customize and facilitate model comparison. When building the `vivid` matrix to use in our heatmap and network visualizations, VIVI metrics that are model specific or model-agnostic may be employed. For measuring interactions we currently only provide the option to use the agnostic Friedman's H-statistic. However, as outlined in the Calculating VIVI Section, the inclusion of different VIVI measures is easily possible.

Our `vivid` package is a useful addition to the other packages in the area of model visualization, such as those discussed in the Introduction Section. Our heatmap and network plots efficiently determine which variables have the greatest impact on the response. When coupled with the seriation, filtering, and clustering techniques, these visualizations enhance the interpretation of ML predictions. Our GPDP and ZPDP can be used to provide a thorough examination of the behavior of a fitted ML model by examining the individual variable effects and their pairwise interactions. These plots combine the bivariate PDP, ICE curves, and scatterplots of the raw variable values. They further allow focusing on subsets of variables with high VInt, and so allow us to efficiently explore a fitted ML model by focusing attention to only the most important aspects.

For future work, the inclusion of other model summaries could be incorporated into `vivid`, such as the interaction statistics described in Greenwell et al. (2018) or the use of Accumulated Local Effects (ALE; Apley and Zhu, 2020). This latter method was created to address bias problems with partial dependency functions and could be used in place of the bivariate PDPs seen in both the GPDP and ZPDP. However the calculation of an agnostic, easily interpretable variable interaction measure that accounts for correlated variables remains an ongoing research goal.

CHAPTER 5

Conclusion

In this thesis we have introduced novel methods to visualise different aspects of ML model fits with the goal of improving post-hoc interpretability. The methods presented are innovative and informative for visualising the importance and interactions of variables as well as various facets of model behaviour.

In Chapter 2 we proposed methods for visualising variable importance and variable interactions in ML models through the use of heatmap and network plots. Coupled with the seriation techniques discussed Chapter 2, a more complete picture of which variables have the most impact on the response in a model fit is provided. Using the VIVI values as a starting point of model investigation, we provided a more in-depth analysis of the variable effects and interactions in our GPDP and ZPDP. Our GPDP is useful for investigating both the bivariate and univariate variable effects by displaying the partial dependence in a generalised pairs plot matrix style. By incorporating ICE curves into the display, we demonstrate the precise nature of any linear or non-linear effects, along with displaying the distribution of the predictor variables. In our ZPDP we expand upon the work of Hofert and Oldford (2020) by presenting a novel method to visualise the partial dependence of variable subsets containing high VIVI measures. In doing so, we focus attention on which variables have the most impact on the predictions.

In Appendix 2.6 of Chapter 2, we examine the use of Friedman's H-statistic as a measure of interaction. It can produce spurious results in situations where there is no interaction present. To avoid any potential misinterpretation of the H-statistic values, our GPDP can be used to investigate the nature of any interactions that may be present.

In Chapter 3 we presented visualisations for posterior evaluation of BART models. We broaden the conventional approach to evaluating the importance and interactions of variables by incorporating the uncertainty that comes with Bayesian models. Using a technique called value suppressing uncertainty palettes, we were able to show the importance, the pairwise interactions, and uncertainty associated with these values in a single heatmap, using colour scale to represent posterior uncertainty. We explored the structure of the decision trees in a BART model by use of our tree plots. Through the use of a multidimensional scaling plot we provided outlier detection and we presented a suite of plots for enhanced model diagnostics, which can be used for gaining a deeper understanding of the behaviour of a model fit. These include visualisations for assessing stability, the acceptance rate of trees, average tree depth and nodes, as well as providing an overall model fit summary via convergence and residual plots.

For evaluating the importance and interactions in a BART model, we used the *inclusion proportion* of these values. This method counts splitting rules and converts these summaries into corresponding VIVI values. However, a drawback of this method is that non-important variables may be included as BART selects the splitting rule uniformly across all variables. This may lead to spurious VIVI values being obtained. As discussed by Chipman et al. (2010), using a small number of trees can offset any spurious values being measured, but this comes at the expense of predictive performance. However, when coupled with the use of a VSUP that displays the relative uncertainty, we provide a correction and has the advantage of being able to be used when the number of trees is large.

In Chapter 4 we presented a detailed discussion of the implementation of our R software package `vivid`, which contains a collection of integrated functions implementing algorithms and novel visualizations for investigating variable importance

and variable interactions in machine learning models. In `vivid` all aspects, from the coded-functions to visualisations, have been carefully designed for efficiency in use and interpretation. Our methods are intuitive, flexible, and easily customisable. Model-specific or model-agnostic methods can be used in our `vivid` matrix. However, custom VIVI values can be supplied to our `vivid` matrix which can be used in our heatmap and network plots.

Finally, the implementation of the proposed methods presented in this work are freely available at https://github.com/alaninglis in the repositories named `vivid` for Chapters 2 and 4 and `bartMan`, for Chapter 3. Thus, all analyses in this thesis are reproducible and methodologies are available to interested practitioners.

5.1 Limitations

In this section we discuss some limitations of our proposed methods.

Scalability

Scalability is an important consideration when it comes to statistical visualisations. A visualisation's capacity to handle large amounts of data without becoming difficult to understand and avoiding information overload is a challenging task. To mitigate information overload, our visualisations have been carefully designed to clearly and effectively communicate the relevant information they contain, while avoiding unnecessary complexity. For example, the heatmap and network graph described in Chapters 2 and 4 make use of seriation techniques that can help avoid any potential information overload. As shown in Chapter 4, these graphics are created by passing a seriated `vivid` matrix to the plotting functions. Consequently, these plots will display the most relevant variables in the top left of the graphic (as for the heatmap) or at the top of the plot (as for the network graph), thereby highlighting the most influential variables. Filtering can be further applied to the seriated `vivid` matrix, before being passed to the plotting functions, to limit the number of variables to be displayed in the visualisations. In the case of the network plot, we additionally provide a filtering option to eliminate variables with VIVI values below a selected threshold.

In Figure 3.4 of Chapter 3, we present a graphic for visualising how a single tree changes over each iteration. In this example we use the default value of 1000 iterations. However, if a larger number of iterations is chosen, this graphic can become quickly overcrowded and difficult to read. We recommend this graphic to be used when the number of iterations is ≤ 1000. In situations where more than 1000 iterations are used, using the summary plots (as shown in Figure 3.5) may be preferable.

Correlated Variables

Correlation between variables is an important element when explaining predictive models. In the case of multicollinearity, where two or more variables are highly correlated and provide similar information, it becomes difficult for a model to differentiate the contribution of each variable to the prediction, making it challenging to assess their importance and interaction values. In Chapter 2, Section 2.6, we show how correlated variables can result in spuriously high interaction measures, which in turn gets translated to a user via our visualisations. The lower portion of our GPDP can be used to visually check if two variables are correlated, thereby mitigating any misrepresentation of importance/interactions. However, we also recommend evaluating any potential correlation between variables in conjunction with our proposed visualisations. Several `R`-packages are available for assessing and visualising correlations, such as `corrplot` (Wei and Simko, 2021) and `corrgrapher` (Morgen and Biecek, 2020). The latter of which displays correlations in a network plot.

Higher Order Interactions

Visualising higher order interactions in machine learning models can be challenging. Currently our proposed methods can only visualise bivariate interactions. Although the H-statistic described in Chapter 2 can be extended to compute higher order interactions, visualising an interaction between three or more variables is challenging. The `pdp` package (Greenwell, 2017) can be used to create partial dependence plots with three dimensions and the `easyalluvial` package (Koneswarakantha, 2022) can be used to plot more than three dimensions side by side in an alluvial style plot.

5.2 Future Work

Our research approached the research question of how to visualise model behaviour from different perspectives, including model-agnostic explanations and model-specific explanations. However, there are numerous potential improvements to explore and new directions for research to go.

For future work we could consider different agnostic methods to be used in place of the partial dependence in our GPDP and ZPDP discussed in Chapters 2 and 4. For example, accumulated local effects (ALE) functions were created by Apley and Zhu (2020) as an alternative to PDPs and were designed as a method to compensate for the bias issues present in partial dependence. In our work to date, we only implement Friedman's H-statistic as a model-agnostic measure of interaction. As discussed in Chapter 4, the H-statistic is a computationally expensive calculation. Although we provide options to sample the data and limit the size of the grid used for evaluating predictions, the time taken to evaluate the H-statistic can still be slow and is highly model dependant. Consequently, a new measure of interaction could be produced to both decrease computational time and reduce any potential bias present in the H-statistic using ALE functions by replacing the the partial dependence functions used in the H-statistic equation (i.e, Equation 2.2) with the corresponding ALE functions. In a similar manner, additional agnostic methods to measure the variable importance could be investigated, such as conditional permutation importance or a partial dependence-based variable importance measure, such as the method described in Greenwell et al. (2018).

From a visualisation perspective, although we did not preform external validation, such as a large scale user study, other members of the machine learning research group provided useful feedback concerning design elements. This information proved to be especially useful and subsequently, as of this date, our `vivid` package has over 8,000 downloads from CRAN and multiple citations. However, a large scale validation approach is a useful tool for future projects.

In our heatmaps and network graphs we show bivariate interactions only. An interesting avenue of research would be to investigate visualisations that are both intuitive and informative for higher dimension interactions. In relation to visual-

ising BART model fits, we presented tree-based plots that can be used to examine the structure of the decision trees. This could be expanded upon by creating interactive plots that allow a user to select individual trees to investigate further. Additionally, developing new methods for effectively visualising the posterior uncertainty associated with BART models presents both challenges and promising areas for future study.

Bibliography

Acuna, E. and Rodriguez, C. (2004). A Meta Analysis Study of Outlier Detection Methods in Classification. *Technical paper, Department of Mathematics, University of Puerto Rico at Mayaguez*, 1:25. 61

Alsmariy, R., Healy, G., and Abdelhafez, H. (2020). Predicting cervical cancer using machine learning methods. *International Journal of Advanced Computer Science and Applications*, 11. 34

American Statistical Association (1995). College data contains statistics for a large number of US colleges from the 1995 issue of US News and World Report. The dataset was used in the American Statistical Association A.S.A statistical graphics section's 1995 Data Analysis Exposition. Accessed: 10-10-2020. 24

Antunes, N., Balby, L., Figueiredo, F., Lourenco, N., Meira, W., and Santos, W. (2018). Fairness and transparency of machine learning for trustworthy cloud services. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 188–193. IEEE. 80

Apley, D. W. and Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(4):1059–1086. 6, 23, 40, 42, 102, 108, 113

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140. 13

Balli, H. and Sorensen, B. (2010). Interaction effects in econometrics. *C.E.P.R. Discussion Papers*, 45. 17

Barbieri, M. M. and Berger, J. O. (2004). Optimal Predictive Model Selection. *The Annals of Statistics*, 32(3):870–897. 77

Barlow, T. and Neville, P. (2001). A Comparison of 2-D Visualizations of Hierarchies. In *IEEE Symposium on Information Visualization*, pages 131–131. INFOVIS. 56

Berrington de González, A. and Cox, D. (2007). Interpretation of interaction: A review. *The Annals of Applied Statistics*, 1. 21

Biecek, P. (2018). Dalex: Explainers for complex predictive models in r. *Journal of Machine Learning Research*, 19(84):1–5. 2, 80

Biecek, P. (2019). Model development process. 2

Biecek, P. and Burzykowski, T. (2021). *Explanatory model analysis: Explore, explain and examine predictive models*. Chapman and Hall/CRC. 2, 80

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G., and Jones, Z. M. (2016). mlr: Machine learning in R. *Journal of Machine Learning Research*, 17(170):1–5. 87

Blattenberger, G. and Fowles, R. (2014). Avalanche Forecasting: Using Bayesian Additive Regression Trees (BART). In *Demand for Communications Services–Insights and Perspectives*, pages 211–227. Springer. 44

Boulesteix, A. L., Janitza, S., Hapfelmeier, A., Steen, K. V., and Strobl, C. (2015). Letter to the editor: On the term 'interaction' and related phrases in the literature on random forests. *Briefings in Bioinformatics*, 16(2):338–345. 21

Bray, F., Ferlay, J., Soerjomataram, I., Siegel, R., Torre, L., and Jemal, A. (2018). Global cancer statistics 2018: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries: Global cancer statistics 2018. *CA: A Cancer Journal for Clinicians*, 68. 34

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32. 3, 4, 20, 44, 60, 66, 81, 83

Brodlie, K., Allendes Osorio, R., and Lopes, A. (2012). A Review of Uncertainty in Data Visualization. *Expanding the Frontiers of Visual Analytics and Visualization*, pages 81–109. 44

Chambers, J. M., Cleveland, W. S., Kleiner, B., and Tukey, P. A. (2018). *Graphical methods for data analysis*. Chapman and Hall/CRC. 3

Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., and Borges, B. (2022). *shiny: Web Application Framework for R*. R package version 1.7.2. 8

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R., et al. (2000). Crisp-dm 1.0: Step-by-step data mining guide. *SPSS inc*, 9(13):1–73. 2

Chen, X. and Ishwaran, H. (2012). Random forests for genomic data analysis. *Genomics*, 99(6):323–329. 81

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010). Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298. 1, 11, 44, 47, 50, 51, 52, 66, 77, 81, 110

Correll, M., Moritz, D., and Heer, J. (2018). Value-Suppressing Uncertainty Palettes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–11. 45, 76

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, Complex Systems:1695. 29, 96

Cutler, D. R., Edwards Jr, T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., and Lawler, J. J. (2007). Random Forests for Classification in Ecology. *Ecology*, 88(11):2783–2792. 60

Deng, H. (2019). Interpreting tree ensembles with intrees. *International Journal of Data Science and Analytics*, 7(4):277–287. 21

Dorie, V. (2022). dbarts: Discrete bayesian additive regression trees sampler. R package version 0.9-22. 44

Earle, D. and Hurley, C. (2015). Advances in Dendrogram Seriation for Application to Visualization. *Journal of Computational and Graphical Statistics*, 24. 25, 89

Englund, C. and Verikas, A. (2012). A Novel Approach to Estimate Proximity in a Random Forest: An Exploratory Study. *Expert Systems with Applications*, 39(17):13046–13050. 60

Felzmann, H., Villaronga, E. F., Lutz, C., and Tamò-Larrieux, A. (2019). Transparency you can trust: Transparency requirements for artificial intelligence between legal norms and contextual concerns. *Big Data & Society*, 6(1):2053951719860542. 80

Fernandes, K., Cardoso, J. S., and Fernandes, J. (2017). Transfer learning with partial observability applied to cervical cancer screening. In *Iberian conference on pattern recognition and image analysis*, pages 243–250. Springer. 34

Fisher, A. J., Rudin, C., and Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research*, 20:177:1–177:81. 4, 20, 21, 86

Fisher, R. A. (1936). The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2):179–188. 53

Frey, H. and Patil, S. (2002). Identification and review of sensitivity analysis methods. *Risk Analysis*, 22:553–578. 19

Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67. 41, 67

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29. 7, 10, 18, 22, 34, 44, 80, 81, 83

Friedman, J. H. and Popescu, B. E. (2008). Predictive Learning via Rule Ensembles. *The Annals of Applied Statistics.*, pages 916–954. 5, 6, 21, 22, 67, 83

Friedman, J. H. and Stuetzle, W. (1981). Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823. 19

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., and Gelman, A. (2019). Visualization in Bayesian Workflow. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 182:389–402. 77

Gleyzer, S. and Prosper, H. (2008). Paradigm, a decision making framework for variable selection and reduction in high energy physics. *PoS*. 17

Goldstein, A., Kapelner, A., Bleich, J., and Pitkin, E. (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65. 6, 29, 83

Grange, S. K., Carslaw, D. C., Lewis, A. C., Boleti, E., and Hueglin, C. (2018). Random forest meteorological normalisation models for swiss pm 10 trend analysis. *Atmospheric Chemistry and Physics*, 18(9):6223–6239. 81

Greenwell, B. M. (2017). pdp: An R package for constructing partial dependence plots. *The R Journal*, 9(1):421–436. 83, 112

Greenwell, B. M. and Boehmke, B. C. (2020). Variable importance plots—an introduction to the vip package. *The R Journal*, 12(1):343–366. 23, 40, 83

Greenwell, B. M., Boehmke, B. C., and McCarthy, A. J. (2018). A simple and effective model-based variable importance measure. *arXiv preprint arXiv:1805.04755*. 87, 91, 108, 113

Hahn, P. R., Murray, J. S., Carvalho, C. M., et al. (2020). Bayesian Regression Trees Models for Causal Inference: Regularization, Confounding, and Heterogeneous Effects. *Bayesian Analysis*. 44

Hahsler, M., Hornik, K., and Buchta, C. (2008). Getting things in order: An introduction to the R package seriation. *Journal of Statistical Software*, 25(3):1–34. 25

Harrison Jr, D. and Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102. 84

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer-Verlag., 2nd edition. 23, 87

He, J., Yalov, S., and Hahn, P. R. (2019). Xbart: Accelerated Bayesian Additive Regression Trees. *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, 89. 51

Helton, J. and Davis, F. (2002). Illustration of sampling-based methods for uncertainty and sensitivity analysis. *Risk analysis : an official publication of the Society for Risk Analysis*, 22(3):591—622. 19

Hernández, B., Pennington, S. R., and Parnell, A. C. (2015). Bayesian Methods for Proteomic Biomarker Development. *EuPA Open Proteomics*, 9:54–64. 44

Hernández, B., Raftery, A. E., Pennington, S. R., and Parnell, A. C. (2018). Bayesian Additive Regression Trees using Bayesian Model Averaging. *Statistics and Computing*, 28(4):869–890. 51

Hill, J. L. (2011). Bayesian Nonparametric Modeling for Causal Inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240. 44

Hofert, M. and Oldford, W. (2018). Visualizing dependence in high-dimensional data: An application to S&P 500 constituent data. *Econometrics and Statistics*, 8(C):161–183. 32

Hofert, M. and Oldford, W. (2020). Zigzag expanded navigation plots in R: The R package zenplots. *Journal of Statistical Software*, 95(4):1–44. 18, 31, 32, 39, 103, 109

Hooker, G. (2004). Discovering additive structure in black box functions. *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 575–580. 23, 40

Hothorn, T., Hornik, K., and Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674. 20

Hurley, C. (2004). Clustering visualizations of multidimensional data. *Journal of Computational and Graphical Statistics*, 13(4):788–806. 25

Hurley, C., OConnell, M., and Domijan, K. (2022). *condvis2: Interactive Conditional Visualization for Supervised and Unsupervised Models in Shiny*. https://cbhurley.github.io/condvis2,. 8

Hurley, C. and Oldford, R. (2011). Eulerian tour algorithms for data visualization and the pairviz package. *Computational Statistics*, 26:613–633. 32, 105

Hurley, C. and Oldford, R. (2022). *PairViz: Visualization using Graph Traversal*. R package version 1.3.6. 105

Hvitfeldt, E., Pedersen, T. L., and Benesty, M. (2022). *lime: Local Interpretable Model-Agnostic Explanations*. R package version 0.5.3. xvii, 80, 82

Inglis, A., Parnell, A., and Hurley, C. (2021). *vivid: Variable Importance and Variable Interaction Displays*. R package version 0.1.0. 18

Inglis, A., Parnell, A., and Hurley, C. B. (2022). Visualizing Variable Importance and Variable Interaction Effects in Machine Learning Models. *Journal of Computational and Graphical Statistics*, pages 1–13. 45, 54, 67, 80, 87, 88, 89, 103

Ishwaran, H. (2007). Variable importance in binary regression trees and forests. *Electronic Journal of Statistics*, 1:519–37. 94

Ishwaran, H., Kogalur, U., Gorodeski, E., Minn, A., and Lauer, M. (2010). High-dimensional variable selection for survival data. *Journal of the American Statistical Association*, 105:205–217. 11, 20, 21, 81

Kapelner, A. and Bleich, J. (2016). bartmachine: Machine Learning with Bayesian Additive Regression Trees. *Journal of Statistical Software*, 70(4):1–40. 44, 52

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., et al. (2020). Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*. 2

Koneswarakantha, B. (2022). *easyalluvial: Generate Alluvial Plots with a Single Line of Code*. R package version 0.3.1. 112

Kruskal, J. B. and Landwehr, J. M. (1983). Icicle Plots: Better Displays for Hierarchical Clustering. *The American Statistician*, 37(2):162–168. 48

Kuhn, M. and Wickham, H. (2020). *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles.* https://www.tidymodels.org. 87

Lang, M., Binder, M., Richter, J., Schratz, P., Pfisterer, F., Coors, S., Au, Q., Casalicchio, G., Kotthoff, L., and Bischl, B. (2019). mlr3: A modern object-oriented machine learning framework in R. *Journal of Open Source Software*. 2, 87, 90

Lang, M., Schratz, P., Sonabend, R., Becker, M., and Richter, J. (2023). *mlr3viz: Visualizations for 'mlr3'*. R package version 0.6.1. 2

LeDell, E., Gill, N., Aiello, S., Fu, A., Candel, A., Click, C., Kraljevic, T., Nykodym, T., Aboyoun, P., Kurka, M., and Malohlava, M. (2020). *h2o: R Interface for the 'H2O' Scalable Machine Learning Platform*. R package version 3.32.0.1. 2

Li, B. (2018). *Sufficient dimension reduction: Methods and applications with R*. Chapman and Hall/CRC. 11

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22. 10

Linero, A. R. (2018). Bayesian Regression Trees for High-Dimensional Prediction and Variable Selection. *Journal of the American Statistical Association*, 113(522):626–636. 77

Linero, A. R., Basak, P., Li, Y., and Sinha, D. (2021). Bayesian Survival Tree Ensembles with Submodel Shrinkage. *Bayesian Analysis*, 1(1):1–24. 50

Linero, A. R. and Yang, Y. (2018). Bayesian Regression Trees Ensembles that Adapt to Smoothness and Sparsity. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(5):1087–1110. 44, 51

Liu, Y., Traskin, M., Lorch, S. A., George, E. I., and Small, D. (2015). Ensemble of Trees Approaches to Risk Adjustment for Evaluating a Hospital's Performance. *Health Care Management Science*, 18(1):58–66. 44, 61

Loyal, J. D., Zhu, R., Cui, Y., and Zhang, X. (2022). Dimension reduction forests: Local variable importance using structured random forests. *Journal of Computational and Graphical Statistics*, pages 1–10. 11

Lukac, A., Sulovic, N., Smiljic, S., Ilic, A., and Saban, O. (2018). The prevalence of the most important risk factors associated with cervical cancer. *Materia Socio Medica*, 30:131. 34, 39

Maksymiuk, S., Karbowiak, E., and Biecek, P. (2021). *EIX: Explain Interactions in 'XGBoost'*. R package version 1.2.0. 81

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2021). *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.7-8. 90

Molnar, C. (2019). *Interpretable Machine Learning*. lulu.com. https://christophm.github.io/interpretable-ml-book/. 17, 24, 29

Molnar, C. (2022). *Interpretable Machine Learning*. lulu.com, 2 edition. 2, 3, 80

Molnar, C., Bischl, B., and Casalicchio, G. (2018). iml: An R package for interpretable machine learning. *JOSS*, 3(26):786. 2, 80

Molnar, C., Casalicchio, G., and Bischl, B. (2020a). Interpretable machine learning–a brief history, state-of-the-art and challenges. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 417–431. Springer. 2

Molnar, C., König, G., Bischl, B., and Casalicchio, G. (2020b). Model-agnostic feature importance and effects with dependent features–a conditional subgroup approach. *arXiv preprint arXiv:2006.04628*. 20

Morgen, P. and Biecek, P. (2020). *corrgrapher: Explore Correlations Between Variables in a Machine Learning Model*. R package version 1.0.4. 112

Murray, K. and Conner, M. M. (2009). Methods to quantify variable importance: implications for the analysis of noisy ecological data. *Ecology*, 90(2):348–355. 80

Oh, S. (2019). Feature interaction in terms of prediction performance. *Applied Sciences*, 9:5191. 17

Paluszynska, A., Biecek, P., and Jiang, Y. (2020). *randomForestExplainer: Explaining and Visualizing Random Forests in Terms of Variable Importance*. R package version 0.10.1. 11, 81

Pang, A. T., Wittenbrink, C. M., Lodha, S. K., et al. (1997). Approaches to Uncertainty Visualization. *The Visual Computer*, 13(8):370–390. 44

Prado, E. B., Moral, R. A., and Parnell, A. C. (2021). Bayesian Additive Regression Trees with Model Trees. *Statistics and Computing*, 31(3):1–13. 44, 77

Probst, P. (2020). *varImp: RF Variable Importance for Arbitrary Measures*. R package version 0.4. xvii, 82

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. 14

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "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. 5

Robertson, P. K. and O'Callaghan, J. F. (1986). The Generation of Color Sequences for Univariate and Bivariate Mapping. *IEEE Computer Graphics and Applications*, 6(2):24–32. 45

Samek, W., Wiegand, T., and Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*. 13

Sathishkumar, V. (2020). Seoul Bike Sharing Demand Prediction. 70

Sathishkumar, V., Park, J., and Cho, Y. (2020). Using Data Mining Techniques for Bike Sharing Demand Prediction in Metropolitan City. *Computer Communications*, 153:353–366. 71, 76

Sathishkumar, V. and Yongyun, C. (2020a). A Rule-Based Model for Seoul Bike Sharing Demand Prediction using Weather Data. *European Journal of Remote Sensing*, 53(sup1):166–183. 71, 76

Sathishkumar, V. and Yongyun, C. (2020b). Season Wise Bike Sharing Demand Analysis using Random Forest Algorithm. *Computational Intelligence*. 71, 76

Schliep, K. and Hechenbichler, K. (2016). *kknn: Weighted k-Nearest Neighbors*. R package version 1.3.1. 90

Shapley, L. S. (1997). A Value for n-Person Games. *Classics in Game Theory*, 69. 5, 78

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*. 13

Smith, J., Green, J., Berrington de González, A., Appleby, P., Peto, J., Plummer, M., Franceschi, S., and Beral, V. (2003). Cervical cancer and use of hormonal contraceptives: A systematic review. *Lancet*, 361:1159–67. 34, 39

Sparapani, R., Spanbauer, C., and McCulloch, R. (2021). Nonparametric machine learning and efficient computation with Bayesian additive regression trees: The BART R package. *Journal of Statistical Software*, 97(1):1–66. 44

Sparapani, R. A., Logan, B. R., McCulloch, R. E., and Laud, P. W. (2016). Nonparametric Survival Analysis using Bayesian Additive Regression Trees (BART). *Statistics in Mdicine*, 35(16):2741–2753. 44, 50

Starling, J. E., Murray, J. S., Carvalho, C. M., Bukowski, R. K., and Scott, J. G. (2020). BART with Targeted Smoothing: An Analysis of Patient-Specific Stillbirth Risk. *The Annals of Applied Statistics*, 14(1):28–50. 50

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. *BMC bioinformatics*, 9:307. 11, 20

Strode, G., Morgan, J. D., Thornton, B., Mesev, V., Rau, E., Shortes, S., and Johnson, N. (2019). Operationalizing Trumbo's Principles of Bivariate Choropleth Map Design. *Cartographic Perspectives*, 94:5–24. 54

Tan, Y. V. and Roy, J. (2019). Bayesian Additive Regression Trees and The General BART Model. *Statistics in Medicine*, 38(25):5048–5069. 50

Teuling, A., Stöckli, R., and Seneviratne, S. I. (2011). Bivariate Colour Maps for Visualizing Climate Data. *International Journal of Climatology*, 31(9):1408–1412. 45

Torgerson, W. S. (1952). Multidimensional Scaling: I. Theory and Method. *Psychometrika*, 17(4):401–419. 45

Trumbo, B. E. (1981). A Theory for Coloring Bivariate Statistical Maps. *The American Statistician*, 35(4):220–226. 54

Um, S. (2021). *Bayesian Additive Regression Trees for Multivariate Responses*. PhD thesis, The Florida State University. 50

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. 84, 90

Wang, M., Lo, S.-H., Zheng, T., and Hu, I. (2012). Interaction-based feature selection and classification for high-dimensional biological data. *Bioinformatics (Oxford, England)*, 28:2834–42. 17

Wei, P., Lu, Z., and Song, J. (2015a). Variable Importance Analysis: A Comprehensive Review. *Reliability Engineering & System Safety*, 142:399–432. 19, 51

Wei, P., Zhenzhou, L., and Song, J. (2015b). A comprehensive comparison of two variable importance analysis techniques in high dimensions: Application to an environmental multi-indicators system. *Environmental Modelling & Software*, 70. 17

Wei, T. and Simko, V. (2021). *R package 'corrplot': Visualization of a Correlation Matrix*. (Version 0.92). 112

Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. 83

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., et al. (2019). Welcome to the tidyverse. *Journal of open source software*, 4(43):1686. 2

Wood, S. (2000). Modelling and smoothing parameter estimation with multiple quadratic penalties. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, 62(2):413–428. 19

Wright, M. N. and Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17. 87

Wright, M. N., Ziegler, A., and König, I. (2016). Do little interactions get lost in dark random forests? *BMC Bioinformatics*, 17:145. 21

Zeileis, A., Fisher, J. C., Hornik, K., Ihaka, R., McWhite, C. D., Murrell, P., Stauffer, R., and Wilke, C. O. (2020). colorspace: A toolbox for manipulating and assessing colors and palettes. *Journal of Statistical Software, Articles*, 96(1):1–49. 25, 92