Explainable Machine Learning for Science and Medicine

Scott M. Lundberg

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Su-In Lee, Chair

Larry Ruzzo

Ali Shojaie

Program Authorized to Offer Degree:

Paul G. Allen School of Computer Science and Engineering

University of Washington

Abstract

Explainable Machine Learning for Science and Medicine

Scott M. Lundberg

Chair of the Supervisory Committee:
Su-In Lee
Paul G. Allen School of Computer Science and Engineering

Understanding why a machine learning model made a certain prediction can be as crucial as the prediction's accuracy in many scientific and medical applications. However, the highest accuracy for large modern datasets is often achieved by complex models that even experts struggle to interpret, such as tree-based ensembles or deep learning models. In this dissertation I present several solutions that improve our ability to explain traditional (often complex) machine learning models. Each of these solutions were developed in response to specific challenges that we faced in our application of machine learning to biology and medicine. I present solutions that enable the better interpretation of very large graphical models, and show how that can enhance our understanding of human genome regulation. I then present a unified model agnostic approach to explain the output of any machine learning model that connects game theory with local explanations, uniting many previous methods. By applying this approach to early-warning medical decision support we are able to use a complex, high accuracy model, and also provide explanations of the clinical risk factors that impacted the model's prediction. I then focus specifically on tree-based models, such as random forests and gradient boosted trees, where we have developed the first polynomial time algorithm to exactly compute classic attribution values from game theory. Based on these methods we have created a new set of tools for understanding both global model structure and individual model predictions. The associated open source software supports many modern machine learning frameworks and is widely used across many industries.

I want to specifically acknowledge my kind wife who made all this possible through her willingness to go on this adventure together. Thank you!

There are so many at Paul Allen G. School to thank that I will inevitably leave many out. But I want to notably thank my advisor Su-In Lee for guiding me through this process and working tirelessly to create an excellent research environment.

Contents

Chapter 1

Overview

Machine learning models are now used almost ubiquitously in business, science, biology, medicine, and consumer products. The computational resources for building these machine learning models is much greater than even just a few years ago, and the size of the datasets we have available to train these models is growing even faster. This combination of big datasets and cheap computation has enabled the development and deployment of increasingly sophisticated and complex machine learning models. However, while the accuracy of these new models has continued to increase, their interpretability has actually decreased. This is because more data and more computational resources have enabled us to train increasingly complicated models that are no longer amenable to direct inspection by a data scientist. This means that the state of the art models used across industry and science today are often black boxes, where we do not have the ability to explain why a model has made a specific prediction. This dissertation seeks to help address this increasingly pressing problem, and so allow us to continue to use complicated models on large datasets without sacrificing interpretability and transparency.

The work presented in this dissertation was done in response to specific explainability challenges that we faced while attempting to apply machine learning to problems in basic science and medicine. Yet the methods I have developed (sometimes in partnership with others in our lab) are not restricted to the problem domains to which we have applied them. They are general approaches that are used by thousands of other researchers and data scientists across many different domains.

This dissertation is broken into four main chapters, where each chapter represents a first author paper written during the course of my graduate studies [103, 101, 105, 104]. You are free to review all the work as a single unit in this document, but you can also obtain the same information by reading each paper in isolation. There is not extra information hidden in this dissertation that I have not already released in each of the constituent papers. Note that each of the papers was published in the order they are presented in this dissertation. To help give a broad understanding of the work I have highlighted several of the key contributions in this overview (Figure 1.1).

Chapter 2 examines how we can use large databases of epigenetic experimental data to better understand how proteins work together inside human cells to manage and regulate our DNA. By combining information from thousands of genome-wide experimental studies (each of which has multiple gigabytes of data), we can learn a comprehensive network of protein interactions inferred using a standard (but very large scale) graphical model. Typically graphical models are considered interpretable, but when combining thousands of variables into a single model we found that redundancies between specific experiments tended to obscure the true structure of the graph. To mitigate this, we designed a new method to improve the interpretability of graphical models in the presence of redundant variables. This *group graphical model* method combines hierarchical agglomerative clustering with a standard graphical model, and helps mitigate the problems that arise when closely related variables are included in the same graphical model. This led to an improved ability to recover protein-protein interactions occurring inside human cells. The quality of this recovery was demonstrated by a large scale comparison with previous knowledge, and by the experimental validation of specific predicted relationships.

Chapter 3 unifies a many previous methods designed to explain individual predictions made by a supervised learning model. In contrast to the unsupervised graphical model learning interpretation in Chapter 2, Chapter 3

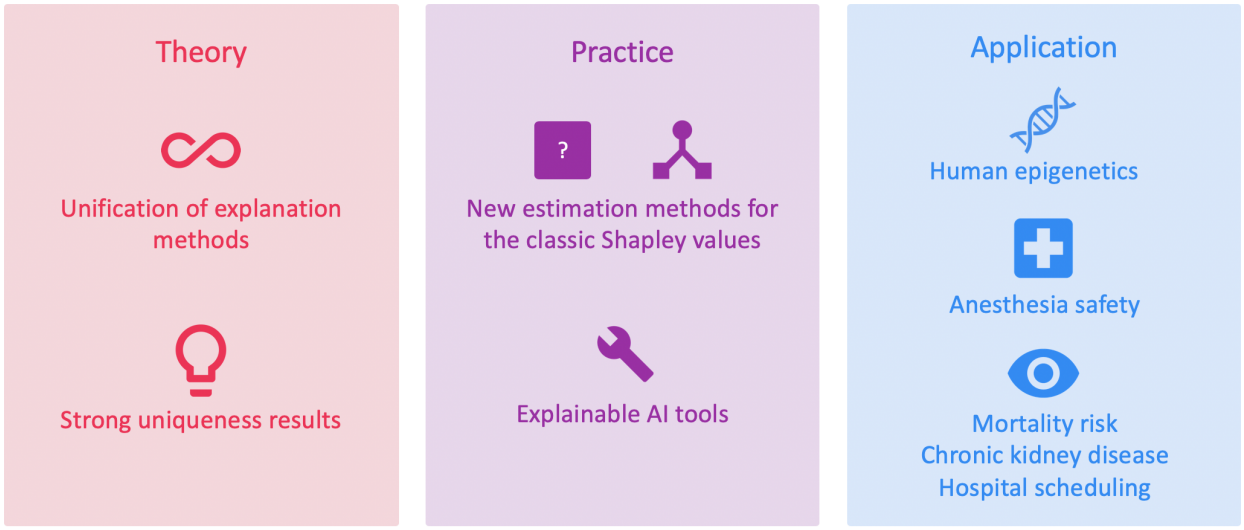


Figure 1.1: **Overview of the contributions contained in this dissertation.** The contributions contained in this dissertation can be broadly categorized into three areas: theory, practice, and application. Two important theoretical contributions include a unification of many previous machine learning explanation methods into a single class, and the demonstration that compelling classic uniqueness results from game theory apply to this entire class. Practical contributions include two fundamentally new methods to compute classic Shapley credit allocation values from game theory more quickly and efficiently, and a broad set of new explainability tools that are enabled by our new estimation methods. Application contributions include our work on learning how the human genome is regulated, improving anesthesia decision support systems, and better modeling mortality risks, chronic kidney disease, and hospital scheduling.

focuses on standard supervised machine learning models. Supervised machine learning is by far the most common type of learning used today, and so improvements that enhance our ability to explain these models have immediate benefits across many different areas. I show in Chapter 3 how classic uniqueness results from game theory apply to an entire class of explanation methods. This has important consequences for almost every method in the class, and led to the development of a new method for approximating the classic Shapley values [151] from cooperative game theory. This new method can use linear regression to estimate the Shapley values for (and so explain) the predictions from any machine learning model.

Chapter 4 uses the model agnostic explanation method described in Chapter 3 to build a medical decision support system for anesthesia care that has the high accuracy of a complex model, but yet retains interpretability. This is important since accuracy and interpretability are both extremely important for medical early warning systems. Poor accuracy can lead to an unusable system with excessive false alarms, and a lack of interpretabiliy can prevent the alarms that are raised from being actionable. It is important for medical decision support to know not only *that* a patient is risk, but also *why* that patient is at risk. Our *Prescience* system demonstrates how to do this without sacrificing model accuracy. This accuracy was validated against both previous machine learning approaches and expert anesthesiologists. When anesthesiologists were assisted by Prescience their ability to anticipate adverse events increased significantly.

Chapter 5 extends the ideas first presented in Chapter 3 in two important directions: 1) It presents the first polynomial time algorithm to compute the classic Shapley values for machine learning models based on trees. In contrast to the sampling based approximations used by previous methods, the new algorithm enables fast and exact computation for any machine learning model based on trees (such as gradient boosted decision trees, or random forests). This has significant practical implications since these types of machine learning models are widely used in industry and research. 2) This chapter also explores how explanations of a model's predictions across an entire dataset can be combined and used as building blocks for many downstream tasks. From improving our ability to monitor deployed machine learning models, to improving our ability to cluster samples in a dataset, the wide variety of tools that we propose suggests that these explanations may prove to be a fundamental enabler of many new ways to understand datasets and models. We demonstrate the value

of these new explanation-based tools in several application domains: understanding mortality risk in the general U.S. population, understanding the risk of progression for chronic kidney disease, and monitoring a hospital machine learning model over a simulated deployment period.

Each of these chapters addresses a different set of explainability challenges and opportunities. This research was motivated by specific applications, but the methods that were developed have broad applicability to almost every domain where machine learning models are used (and hence need to be explained). We have released open source implementations of all these methods. Those from Chapter 3 and Chapter 5 are now used by hundreds of data scientists every day. I am grateful that these methods have been helpful to so many, and I look forward to how these tools grow and mature in the future.

Chapter 2

ChromNet: Learning the human chromatin network from all ENCODE ChIP-seq data

A cell's epigenome arises from interactions among regulatory factors – transcription factors and histone modifications – co-localized at particular genomic regions. We developed a novel statistical method, ChromNet, to infer a network of these interactions, the *chromatin network*, by inferring conditional dependence relationships among a large number of ChIP-seq datasets. We applied ChromNet to all available 1,451 ChIP-seq datasets from the ENCODE Project, and showed that ChromNet revealed previously known physical interactions better than alternative approaches. We experimentally validated one of the previously unreported interactions, MYC – HCFC1. An interactive visualization tool is available at: `http://chromnet.cs.washington.edu`.

2.1 Introduction

Regulatory factors – such as transcription factors, histone modifications, and other DNA-associated proteins – co-localize in the genome and interact with each other to regulate gene expression [37], the physical structure of the genome [27], cell differentiation [12], and other cellular processes. Identifying the genomic co-localization in this network among regulatory factors, which we termed the *chromatin network*, is important for understanding genome regulation and the function of each regulatory factor [159, 11]. To identify the chromatin network, we can use chromatin immunoprecipitation-sequencing (ChIP-seq) to measure genome-wide localization of regulatory factors, and then compare ChIP-seq datasets to find regulatory factors that co-localize [122, 29]. Co-localization may indicate that two factors interact physically, by forming a complex, or functionally, by regulating similar DNA targets.

However, identifying pairwise co-localization alone fails to distinguish direct interactions from indirect interactions. A direct interaction represents physical contact or close functional coupling that requires spatial proximity. An indirect interaction is not from physical contact or direct functional coupling, but instead reflects the transitive effect of other direct interactions. Consider a simulated chromatin network among four factors, where factor C recruits A and B, and A in turn recruits D (Figure 2.1A, top). Because all pairs of ChIP-seq datasets are correlated to each other (Figure 2.1A, middle), a simple co-localization method would incorrectly infer interactions among all the factors (Figure 2.1A, bottom left). In a *conditional dependence network* (Figure 2.1A, bottom right), if two variables (here, factors) are *conditionally dependent*, then there is an edge between them. The *conditional dependence* between two factors measures their co-localization after accounting for information provided by other factors. If we infer a conditional dependence network, we eliminate indirect edges from the network, such as between factors A and B, because their co-localization at peaks 3 and 5 can be *explained away* by another factor C (C recruits A and B). Hence, incorporating more ChIP-seq datasets allows more indirect edges to be removed, resulting in a higher quality network.

Here we present ChromNet, an approach that estimates the human chromatin network using a conditional dependence network among regulatory factors from 1,451 human ENCODE ChIP-seq datasets (Table S1).

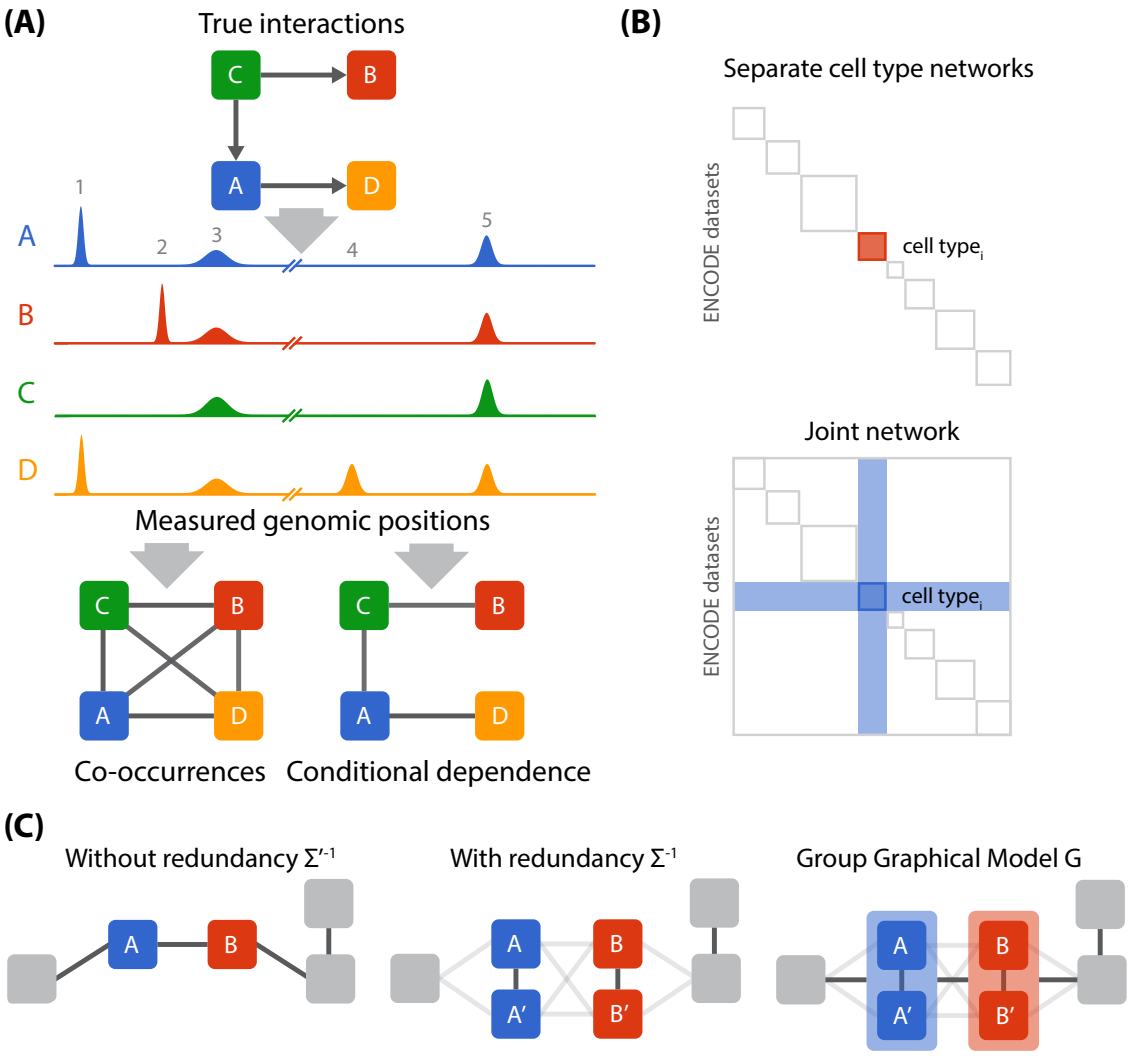


Figure 2.1: **(A)** *Top:* Interaction network among four simulated regulatory factors. *Middle:* Binding activity from simulated ChIP-seq datasets, where each peak represents a putative binding position of a protein. *Bottom:* Networks inferred from ChIP-seq datasets based on co-occurrence (left) or conditional dependence (right). **(B)** Comparison of separate cell type networks (top) with a single joint network (bottom). In a joint model, factors in each cell type have opportunities to be connected with new regulatory factors in other cell types, as highlighted by the blue shaded region (bottom). **(C)** Redundant information obscures conditional dependence connections. *Right:* Without redundancy, standard methods robustly infer a conditional dependence network. *Middle:* Highly correlated variables (such as A and A') are strongly connected with each other and lose their connections with other variables. *Left:* A Group Graphical Model (GroupGM) represents the conditional dependence between groups of correlated variables, which restores the connection between A and B.

Integrating all ENCODE datasets from many cell types into a single network provides several advantages. First, it enables the extraction of global patterns in the conditional dependence relationships among regulatory factors in all cell types. Second, it provides a flexible model that allows direct comparison of cell-type specific sub-networks because factors are conditioned on the same global set of ChIP-seq datasets across all cell types. Finally, it greatly increases the number of edges to consider by allowing edges connected to factors outside a single cell type (Figure 2.1B). We show that this leads to a substantially increased fold enrichment for known protein interactions.

Learning a joint network among all available ENCODE ChIP-seq datasets involves three key challenges. First, learning a network among thousands of ChIP-seq datasets based on millions of genomic regions is highly computationally intensive. To solve this challenge, we utilized an efficient approach that involves the computation of an *inverse correlation matrix*, which does not require an expensive iterative learning procedure. This is in contrast to some other methods, such as Bayesian networks [162, 10] and Markov random fields [184], which face difficulties scaling up and make it infeasible to run on all 1,451 ChIP-seq datasets (Supplementary Note 1). Second, some regulatory factors are in the same complexes, and factors are often measured in different labs, conditions, or cell types, which creates significant correlations in the data. When some variables are highly correlated with each other, standard methods often learn edges only among these variables and disconnect them from the rest of the network (Figure 2.1C, middle) [9]. Incorporating more ChIP-seq datasets exacerbates this problem. To solve this challenge, we present the *Group Graphical Model* (GroupGM) representation of a conditional dependence network that expresses conditional dependence relationships among groups of regulatory factors as well as individual factors (Figure 2.1C, right). We show that GroupGM improves the interpretation of a conditional dependence network by allowing edges to connect groups of variables, which makes the edges robust against data redundancy. Third, network edges can be driven by interactions in specific genomic contexts. To help understand these contexts we present an efficient method to estimate for each genomic position its impact on an inferred GroupGM edge.

Previous work on learning interactions among regulatory factors from ChIP-seq data used much smaller data collections. ENCODE identified conditional dependence relationships among groups of up to approximately 100 datasets in specific genomic contexts [56]. Other authors used partial correlation on 21 datasets [90], Bayesian networks for 38 datasets [93], and partial correlation combined with penalized regression for 27 human datasets [135] and for 139 mouse embryonic stem cell datasets [70]. Still other authors used a Markov random field with 73 datasets in *D. melanogaster* [184], a Boltzmann machine with 116 human transcription factors [117], and bootstrapped Bayesian networks in 112 regulatory factors in *D. melanogaster* [162, 10]. Only other approaches also based on linear dependence models, such as partial correlation used by Lasserre et al. [90], scale to all ENCODE datasets ('Partial correlation' and 'rank(Raw read pileup)' in Figure S1). The ChromNet approach extends these methods in four distinct ways: 1) We show that linear dependence models can directly be applied to the genome-wide untransformed read count data (Figure S1); 2) ChromNet addresses a fundamental challenge in network estimation when some of the variables are highly correlated with each other (collinearity) through a novel statistical method, the group graphical model; 3) ChromNet uses a novel method to identify genomic positions and genomic contexts that drive specific network edges; and 4) Jointly modeling multiple cell types leads to a more informative network with a substantially higher enrichment for known protein interactions. Network inference has also been applied to gene expression data, but the number of available samples in expression data is much lower than that in ChIP-seq datasets, which leads to different challenges.

ChromNet departs from previous approaches by enabling the inclusion of all 1,451 ENCODE ChIP-seq datasets into a single joint conditional dependence network. GroupGM and an efficient learning algorithm allow seamless integration of all datasets comprising 223 transcription factors and 14 histone marks from 105 cell types without requiring manual removal of potential redundancies (Table S1). We show that this approach significantly increases the proportion of network relationships among ChIP-seq datasets supported by previously known protein-protein interactions as compared to other scalable methods (Results). We also demonstrate the potential of ChromNet to aid new discoveries by experimentally validating a novel interaction.

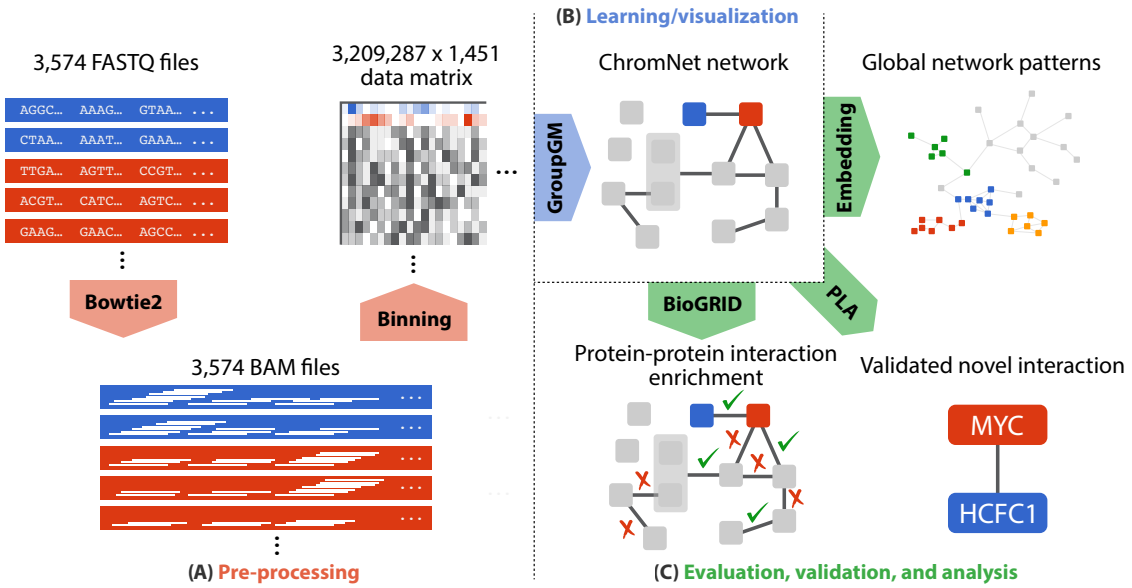


Figure 2.2: **(A)** The uniform processing pipeline includes aligning sequences using Bowtie2, and then binning them into 1,000–base-pair regions. **(B)** We inferred the GroupGM from all 1,451 ChIP-seq datasets and integrated the learned model into a web interface to facilitate broad use. **(C)** We evaluated the learned model against known physical protein interactions (BioGRID), mapped global patterns through network embedding, and validated a novel predicted MYC-HCFC1 interaction with the proximity ligation assay (PLA).

2.2 Results

2.2.1 Uniformly processed data reduces noise when learning conditional dependence

To ensure comparable signals across all ChIP-seq datasets, we re-processed raw ENCODE sequence data with a uniform pipeline (Figure 2.2A). We downloaded raw FASTQ files from the ENCODE Data Coordination Center [29, 144, 42] and mapped them using Bowtie2 [88] to the human genome reference assembly (build GRCh38/hg38) [54]. We binned mapped read start sites into 1,000–base-pair bins across the entire genome, which results in a $3{,}209{,}287 \times 1{,}451$ data matrix X where genomic positions are viewed as samples (Figure 2.2A). We compared several different data pre-processing methods and chose binned read counts for three reasons: 1) they allow easy integration of external ChIP-seq experiments, 2) they do not require the determination of various cut-offs in a peak calling algorithm, and 3) a network inferred from read counts performs well revealing previously known protein-protein interactions (Figure S1).

2.2.2 A conditional dependence network can be efficiently learned from binned read count data

Learning a conditional dependence network among thousands of ChIP-seq datasets each containing millions of samples (genomic positions) requires an efficient algorithm (Figure 2.2B). It is well known that the nonzero pattern of the *inverse covariance matrix* of Gaussian random variables represents the conditional dependence network [92, 108]. The inverse correlation matrix, Σ^{-1}, is a normalized version of the inverse covariance matrix and also represents conditional dependence. A zero element ($\{\Sigma^{-1}\}_{ij} = 0$) means that the ith and jth variables are conditionally independent of each other given all other variables—they are not connected by an edge.

While it is common practice to learn the conditional dependence network among continuous-valued variables based on the estimation of Σ^{-1} [61], count data requires more care. Distributions of counts in binned ChIP-seq reads are often clearly truncated at zero, and also increase in variance for high read counts.

Multivariate distributions with count-valued marginal distributions are often very restrictive (for example only allowing positive correlations) or are infeasible to estimate for thousands of dimensions [178]. An often employed alternative is to use a multivariate Gaussian distribution after appropriately transforming the count data, such as with the *sqrt* or *asinh* function [19]. However, interestingly, our results show that applying a linear Gaussian model directly to the binned read counts of ENCODE ChIP-seq data better recovers known protein-protein interactions than when using standard normalizing data transforms (Methods; Figure S1; Figure S2). This leads to an efficient and simple model formulation for ChromNet applied directly to the mapped read counts, which is relatively easier to obtain compared to other ChIP-seq data pre-processing methods and does not require any threshold.

ChromNet first computes the *inverse sample correlation matrix* $\hat{\Sigma}^{-1}$ from the data matrix X of 1,451 variables and 3,209,287 samples, and then uses a GroupGM approach to interpret elements of $\hat{\Sigma}^{-1}$ as weights of network edges (Figure 2.2B).

2.2.3 Group modeling mitigates the effects of redundancy

Many ENCODE ChIP-seq datasets contain redundant positional information. Conventional conditional dependence methods have a key limitation in modeling redundant data. If datasets A and A' are highly correlated, a conventional method would connect A with A' but connect A to the rest of the network only weakly (Figure 2.1C). Arbitrarily removing or merging redundant datasets can hide or eliminate important information in the data.

GroupGM overcomes challenges with redundant data in conditional dependence models by allowing edges that connect groups of datasets (such as $[A,A']$ and $[B,B']$). A group edge weight represents the total dependence between the variables in the two groups that the edge connects, and is computed from Σ^{-1} as (Methods):

$$G_{[A,A'][B,B']} = \Sigma^{-1}_{AB} + \Sigma^{-1}_{AB'} + \Sigma^{-1}_{A'B} + \Sigma^{-1}_{A'B'}.$$

An edge in a GroupGM model implies conditional dependence between the linked groups, but does not specify the involvement of individual factors in each group. We prove that GroupGM correctly reveals conditional dependencies in the presence of redundancy (Supplementary Note 2).

A group is defined as a set of highly correlated variables whose individual conditional dependence relationships with other variables are not likely to be captured, as illustrated in Figure 2.1C. To obtain groups, we used complete linkage hierarchical clustering, and restricted groups to have minimum pairwise correlation of ρ (= 0.8) within each group. The choice of complete-linkage clustering allows us to obtain groups where all the factors are highly correlated. Because the complete-linkage distance metric merges two clusters based on the minimum correlation between any two variables in the groups, we can stop merging when the minimum correlation becomes less than or equal to ρ before creating all $2p-1$ groups, where p = 1,451.

Each variable (a ChIP-seq dataset) can be in multiple groups as long as it is highly correlated with at least one other dataset. This multi-scale nature of groups is a unique feature of the group graphical model. It allows us to capture multiple ways each factor can be connected with other factors. Say that a dataset for factor A forms a group with another dataset for factor B. In the group graphical model, A can have connections specific to itself and connections shared with B, and their edge weight values would indicate which connections are statistically robust. This allows us to reveal multiple kinds of interactions A can have – specifically to itself and to A and B as a complex. The later may not be captured by a conventional conditional dependence network, such as inverse correlation or partial correlation, if A and B are highly correlated with one another.

The purpose of having a threshold for minimum pairwise correlation ρ is to identify sets of variables whose high within-group correlation is likely to prevent them from being connected to other variables in the network. The threshold used in this paper $\rho = 0.8$ captures 53% of all the multi-factor groups formed by hierarchical clustering, and was chosen so as to include strong groups while still keeping the size of groups small enough to interpret (Figure S3).

2.2.4 Conditional dependence and joint group modeling improve the recovery of known protein-protein interactions

To evaluate how conditional dependence and group modeling both contribute to the performance of ChromNet, we estimated three networks among ChIP-seq datasets using the following three methods, where each method produces a set of weighted edges:

1. **Correlation:** We learned a naive co-occurrence network, using pairwise Pearson's correlation between all pairs of datasets.
2. **Inverse correlation:** We learned a conditional dependence network, by computing the matrix inverse of the correlation matrix.
3. **GroupGM:** We learned a group conditional dependence network, which addresses tight correlation among datasets by allowing edges between groups of variables.

Partial correlation is similar to inverse correlation and performs nearly as well (Figure S4). We did not include other previously described methods because they do not scale to the large data collection we used (Supplementary Note 1; Figure S5).

To assess the quality of the estimated networks, we identified the edges corresponding to published protein-protein interactions. As ground truth, we used the BioGRID database's assessment of physical interactions between human proteins from experiments deemed low throughput [161]. For evaluation, we excluded edges connecting the same regulatory factor even when measured in different labs, cell types, or treatment conditions. These edges were excluded from evaluation to prevent them from artificially inflating the accuracy of the methods. We also excluded edges involving a histone mark because they do not exist in BioGRID. For these edges we ran a separate evaluation using the HIstome database [77] and showed that the group graphical model shows higher enrichment than the alternative methods (Figure S6). When we measured the conditional dependence between a pair of ChIP-seq datasets in GroupGM, to avoid the inclusion of many redundant edges, for each pair of datasets, we picked the maximum edge weight out of all network edges connecting groups, each of which contains one of the corresponding datasets. This way, we consider exactly the same number of dataset pairs for evaluation across all three methods. We only scored edges from groups containing a single type of factor (about half of the groups; see Figure S3), because if a group contains more than one factor, there is no clear way to characterize such an edge as true or false from BioGRID, or match it with an edge from competing methods for comparison.

Group modeling improves the recovery of interactions within and between cell types

We compared performance of the three methods described above across a range of prediction thresholds. For each network, we varied a number N of evaluated edges from 1 to the total number of edges. For each value of N, we identified the set of N edges with the largest weights. We also randomly picked N edges without regard to weight rank as a background set. We then calculated how many edges in each set matched known protein-protein interactions from BioGRID. We computed fold enrichment by dividing the number of matched edges in the prediction set by the expectation of the number in the background set. Since 8.4% of dataset pairs in the same cell type are supported by a BioGRID physical interaction, an enrichment fold of 1 corresponds to 8.4% of recovered edges matching prior knowledge. Enrichment fold captures the effect of both Type I and Type II error rates (see Methods).

We first measured performance within all cell types, excluding edges between datasets in different cell types (Figure 2.3A top). Since the limited number of annotations in BioGRID imperfectly represent the human chromatin network, one cannot draw strong conclusions about absolute performance from this benchmark. Relative performance of the methods, however, is clear; inverse correlation performs better than correlation, and GroupGM outperforms inverse correlation. This supports the idea that better resolution of direct versus indirect interactions contributes to improved performance of inverse correlation over correlation, while greater robustness against redundancy likely contributes to improved performance of GroupGM over inverse correlation. The value of conditional dependence and group modeling is also further supported by specific examples in the network (Figure 2.4; Figure S7; Figure S8), and by the fact that GroupGM still outperforms

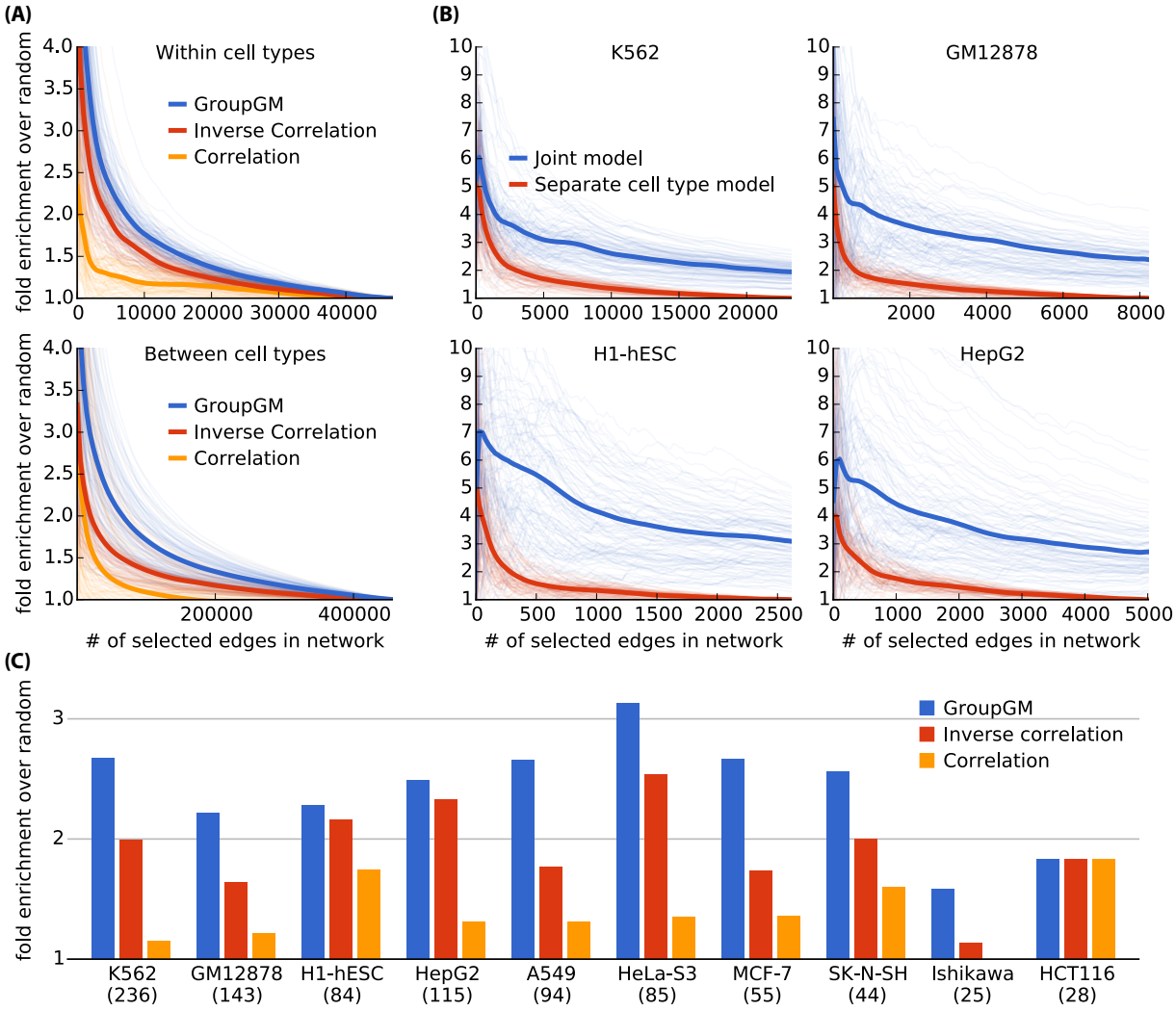


Figure 2.3: Enrichment of BioGRID-supported edges between transcription factors in networks estimated by correlation (yellow), inverse correlation (red), and GroupGM (blue). In (A) and (B), light lines represent bootstrap resampling variability, and dark lines represent average performance over all re-sampled networks. **(A)** Fold enrichment for BioGRID-supported edges against a varying number of evaluated network edges. *Top:* excluding edges between different cell types. *Bottom:* only including edges between different cell types. **(B)** Enrichment against a varying number of evaluated network edges. Here, we compared between a joint model and a cell-type specific model in four different cell types. **(C)** Enrichment within cell types that have 25 supported edges or more, where the network density was set to match the number of BioGRID-supported edges in each cell type. Beneath each cell type name is the number of datasets in that cell type.

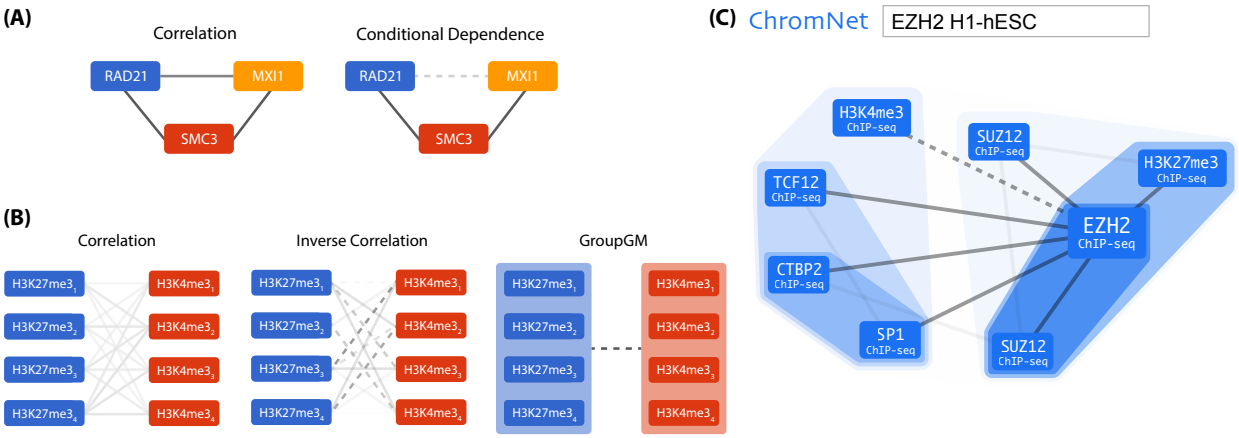


Figure 2.4: **(A)** *Left:* RAD21, MXI1, and SMC3 co-localize with one another, suggesting that they may all interact with each other. *Right:* ChromNet reveals that the co-localization of RAD21 and MXI1 was largely mediated by the presence of SMC3. **(B)** GroupGM overcomes edge instability between tight clusters of H3K27me3 (blue) and H3K4me3 (red) datasets in H7-hESC measured at different differentiation time points. Edge darkness indicates the strength of the connection; dashed lines indicate negative interactions. We have removed within-group edges for clarity. *Left*: Correlation. *Middle*: Inverse correlation. *Right*: GroupGM. **(C)** The part of the ChromNet network that interacts with EZH2 in H1-hESC embryonic stem cells. This is a screen capture from our web interface with a search for "EZH2 H1-hESC". Shaded regions represent the GroupGM groups learned by hierarchical clustering; darker regions represent tighter clusters. We set the edge threshold to capture the six strongest edges connected to EZH2.

inverse correlation even after attempting to remove the strongest redundancies by merging datasets from different labs targeting the same factor in the same cell type/condition (Figure S9).

To assess the variability of the enrichment estimate, we performed bootstrap re-sampling of regulatory factor targets (Figure 2.3A and B, light curves). All datasets with the same factor are sampled together, leading to a conservative (high) estimated variability (Methods). GroupGM showed a statistically significant improvement over both correlation ($P = 0.0004$) and inverse correlation ($P = 0.0036$) for edges within cell types (Figure S10).

To assess variability over cell types, we estimated enrichment separately for each cell type with 25 or more BioGRID-supported edges. In each cell type, we identified the number N of BioGRID-supported edges in that cell type. Then, we calculated the enrichment for BioGRID-supported edges among the top N edges in that cell type (Figure 2.3C). GroupGM performed consistently better than correlation or inverse correlation in individual cell types (Figure S11).

We also generated a simulated data set meant to mirror the characteristics of real ChIP-seq datasets (Figure S12). Using this simulated data, we found a similar relative performance of various methods, with GroupGM recovering the most true network edges (Figure S13; Methods).

To assess how well a joint model can recover relationships between factors measured in different cell types, we checked edges between different cell types for enrichment in known protein-protein interactions (Figure 3A bottom). The GroupGM network showed a clear enrichment for known interactions above random ($P = 0.0095$), and also outperformed inverse correlation ($P = 0.0174$) and correlation ($P = 0.0282$) (Figure S14; Methods). This implies that information about many physical protein interactions can be recovered even from datasets in different cell types.

Comparison between a joint model of all cell types and cell-type specific models

Integrating ChIP-seq datasets from multiple cell types into a single network model provides the following three advantages. First, we can capture high-level patterns in the joint chromatin network that would not otherwise be visible. Second, a joint model allows the direct comparison of cell-type specific sub-networks because factors are conditioned on the same global set of ChIP-seq datasets across all cell types. Finally, a

dataset for a regulatory factor in one cell type can serve as a proxy for a missing dataset for that factor in another cell type, if the factor's localization in the genome is conserved between the cell types (Figure S15). This greatly expands potential chromatin network edges to include the union of regulatory factors measured in any cell type. This global network contains both conserved and cell-type specific sub-networks, and proves useful in analyzing data from ENCODE, which only measures a few factors in some cell types.

To directly compare a joint model across all cell types with cell-type specific models for each cell type separately, we focused on the four best characterized ENCODE cell types and compared enrichment of BioGRID-supported edges (Figure 3B). By varying the number of edges in the networks we find that the joint model consistently identifies interactions with higher fold enrichment for known interactions. In addition, a joint model also identifies more unique BioGRID supported protein-protein interactions than cell-type specific models (Figure S16).

We show as well that the large increase in potential edges from a joint model does not introduce spurious associations among edges within a cell type. When we excluded all cross-cell-type edges from the joint model, the joint model still marginally out-performs cell-type specific models ($P = 0.0672$; Figure S17).

2.2.5 An example of the importance of conditional dependence: SMC3 separates RAD21 and MXI1

A specific example illustrates how conditional dependence reveals experimentally-supported direct interactions better than pairwise correlation (Figure 2.4A). In the correlation network among RAD21, SMC3, and MXI1, the three factors were tightly connected with one another in HeLa-S3 cervical carcinoma cells. The conditional dependence network, however, separated RAD21 and MXI1. This separation arose from the ability of SMC3 to explain away the correlation between RAD21 and MXI1. The factor pairs left connected in the conditional dependence network, RAD21–SMC3 and SMC3–MXI1, have physical interactions described in BioGRID [98, 59]. BioGRID lacks any direct connection between RAD21 and MXI1. Panigrahi et al. discovered more than 200 RAD21 interactors using yeast two-hybrid screening, immunoprecipitation–coupled mass spectrometry, and affinity pull-down assays [128]. They did not identify a RAD21–MXI1 interaction, which implies that RAD21 may not directly interact with MXI1.

In order to focus on the comparison between conditional dependence and correlation, we have not displayed the group that contains SMC3 and RAD21. This grouping reflects their common role in the cohesion complex and is present in many cell types. We also note that Figure 2.4A is only a small part of the full ChromNet network and considering more factors reveals additional relationships that involve CTCF and ZNF143, which is consistent with prior knowledge [187] (Figure S18).

2.2.6 An example of the importance of group dependency: recovering a connection between H3K27me3 and H3K4me3

Another specific example shows how GroupGM mitigates the effect of redundancy on conventional conditional dependence models. We examined edges between multiple H3K27me3 and H3K4me3 datasets from H7-hESC embryonic stem cells, collected at different time points in differentiation [126]. H3K27me3 is a repressive mark and H3K4me3 is an activating mark. Since the datasets represent different portions of the differentiation process, one should not average them or pick a reference dataset arbitrarily. However, the H3K27me3 datasets are correlated highly enough with one another to form a group, and so are the four H3K4me3 datasets. This implies that conventional conditional dependence methods would identify edges between the two histone marks incorrectly.

Edges estimated using correlation indicate that the ChIP-seq datasets targeting H3K27me3 and those targeting H3K4me3 are positively correlated. However, H3K27me3 is associated with repressed genomic regions while H3K4me3 is associated with actively transcribed regions [185]. Since a minority of promoters in embryonic stem cells are "bivalently" marked, these two marks should not have an overall positive association [12, 185]. In fact, most ChIP-seq datasets are positively correlated with each other (Figure S15), which is induced by mappability and many regions that are transcriptionally silent or active. Resolving this problem by removing some of these regions is unlikely to be successful, because it is not clear based on what criteria we need to exclude regions. In conditional dependence models, such as inverse correlation and the group graphical model, by conditioning on many other variables, these global confounding effects are naturally

removed. Edges estimated by inverse correlation account for these confounders but become weak and unstable showing a mixture of positive and negative associations (Figure 2.4B, middle). By allowing group edges, GroupGM has power to recover the negative association between H3K27me3 and H3K4me3 (Figure 2.4B, right), which is consistent with prior knowledge [12, 185].

2.2.7 An example of learning genomic context: ZNF143 mediates the conditional dependence relationship between CTCF and SIX5

Many relationships between regulatory factors only occur in a particular genomic context. This raises the question of how, or whether, this context specificity is encoded in ChromNet. We can gain insight into this by considering what it means for one factor to mediate the relationship between two other factors, such as A mediating the relationship between C and D in Figure 1A. When this occurs it means that the connection between C and D can be explained by their co-occurrence with A. In other words, A is the context in which the relationship between C and D occurs.

A practical example of this is found in the relationships between SIX5, ZNF143, and CTCF in the K562 cell type. Simple correlation connects all three factors together with positive edges, but GroupGM shows that ZNF143 actually mediates the relationship between SIX5 and CTCF (Figure S7; Figure S8). This means that the association of SIX5 with CTCF primarily occurs in the presence of ZNF143, the CTCF-SIX5 relationship is context-specific and ZNF143 is the context. More generally when an association between two factors, C and D, is specific to a certain genomic context and that context is well represented by a third factor, A, then A would mediate C and D. This gives the connections C – A – D in the conditional dependence network; thus context-specific relationships, such as the relationship between CTCF and SIX5 in the presence of ZNF143, are captured in a GroupGM network, if all three factors are present.

It is important to understand the genomic context in which any given edge occurs regardless of whether that context is well represented by another factor in the network. Even if A is not observed we want to be able to infer the genomic context of the interaction between C and D. To address this need, we designed an efficient method to label every genomic position with its influence on a group network edge (Methods). Using CTCF–ZNF143–SIX5 as an example, we removed all ZNF143 experiments from ChromNet and then computed the genomic context of the edge between CTCF and SIX5. To validate this genomic context, we took the top 1,000 bins (1,000,000 bp) and intersected them with the top 1,000 bins from all other experiments in K562, including ZNF143. Even though ZNF143 was not present in the model and ZNF143 datasets were not used when inferring the genomic context, it had the highest overlap of any experiment with the context driving the CTCF–SIX5 edge, even higher than the CTCF and SIX5 experiments themselves (Figure S19).

2.2.8 An example of network accuracy: recovered interactions with EZH2 in H1-hESC recapitulate known functions

As an example illustrating the utility of ChromNet in revealing the potential interactors of a specific regulatory factor we examined a small portion of the network associated with the well-characterized protein EZH2 (Figure 2.4C). We focused on the H1-hESC cell type because it had many strong EZH2 connections in ChromNet. Examining connections to EZH2 in H1-hESC highlighted several known interactions, which we discuss in decreasing order of edge strength. The strongest connection is from H3K27me3, and EZH2 is a methyltransferase involved in H3K27me3 maintenance [3]. The next strongest connections are with SUZ12, which is an essential part of the Polycomb repressive complex 2 (PRC2), and is required for EZH2's methyltransferase activity [18, 36]. The next connection to CTBP2 is supported by this co-repressor's possible role in deacetylation of H3K27 in preparation for PRC2-mediated methylation [80]. H3K4me3 is well known to be present in active regions of the genome, so a negative relationship with EZH2 (represented by a dashed line) that deposits the repressive H3K27me3 mark is expected. SP1 is a potentially novel interactor of EZH2, while TCF12 is known to co-immunoprecipitate with EZH2, which suggests that TCF12 interacts with PRC2 [94]. In summary, most of the strongest interactions with EZH2 have support in the literature. We found this mixture of interactions supported by the literature and potential novel connections in many parts of the network.

2.2.9 An example of cross-cell-type comparison: enhancer associated regulatory factors

Learning a conditional dependence network for all ENCODE cell types allows the comparison of within cell type connections across different cell types. Active enhancers are known to be flanked by a combination of the histone marks, H3K27ac and H3K4me1 [152]. To quantify how strongly different transcription factors associate with active enhancers in different cell types we calculated the sum of the group edges between each regulatory factor (except histone marks) and H3K27ac and H3K4me1 measured in the that cell type. This provides a score for each factor in each cell type. Seven ENCODE cell types with 20 or more datasets contain both H3K27ac and H3K4me1, while also containing EP300, which is known to bind active enhancers [152]. We focused on these seven cell types and ranked the factors in each cell type by their association with H3K27ac and H3K4me1. Table S2 lists the top 10 factors in each cell type most associated with active enhancers. EP300 can be considered a validation for the list and is highly ranked in all seven cell types ($P < 10^{-5}$). Interestingly, even more highly ranked than EP300 is POLR2A. This association is likely because active enhancers are in close proximity to active transcription start sites in promoters in 3D space, due to the looping mechanisms for enhancer-promoter communication. The influence that 3D conformation can have on measures of co-localization in the genome is important to bear in mind when analyzing the edges in ChromNet. Other factors that are consistently associated with enhancers across cell types are shown in red, while cell-type specific associations are in black (Table S2).

2.2.10 An example of a novel protein interaction: experimental validation of an interaction between MYC and HCFC1

The c-MYC (MYC) transcription factor is frequently deregulated in a large number and wide variety of cancers [115, 132]. It heterodimerizes with its partner protein MAX to bind an estimated 10-15% of the genome to regulate the gene expression programs of many biological processes, including cell growth, cell cycle progression, and oncogenesis [115, 132, 13]. The mechanisms by which MYC regulates these specific biological and oncogenic outcomes are not well understood. Interactions with additional co-regulators are thought to modulate MYC's binding specificity and transcriptional activity [60, 172]; however, only a few MYC interactors have been evaluated on a genome-wide level. Analysis of the large number of ENCODE ChIP-seq datasets can therefore further elucidate MYC interactions at the chromatin level.

ChromNet showed that MAX is the strongest interactor of MYC across multiple cell types (Table S3), highlighting the ubiquitous nature of this interaction. Top-scoring ChromNet connections also included other known MYC interactors, for example, components of the RNA polymerase II complex such as POLR2A and chromatin-modifying proteins such as EP300 (Table S3). This shows how ChromNet can help identify protein complexes and interactions.

In addition to the known interactors described above, ChromNet also revealed previously uncharacterized, high-scoring interactions, including the transcriptional regulator Host Cell Factor C1 (HCFC1) (Table S3). HCFC1 binds largely to active promoters [116] and is involved in biological processes, such as cell cycle progression [131, 137] and oncogenesis [130, 35, 134]. This further supports its possible role as an interactor of MYC in regulating these activities. To validate the novel MYC-HCFC1 interaction, we performed a proximity ligation assay (PLA) in MCF10A mammary epithelial cells. This technique detects endogenous protein-protein interactions in intact cells [157] and has been used to validate novel interactors of MYC [51]. When two proteins that are probed with specific antibodies are within close proximity of each other, fluorescence signals are produced that are measured and quantified using fluorescence microscopy. We saw only background fluorescence when incubating with antibody against MYC (Figure 2.5A, top) or HCFC1 (Figure 2.5A, middle) alone. Incubation with both MYC and HCFC1 antibodies yielded a significant increase in fluorescence signal in the nuclear compartment (Figure 2.5A, bottom; Figure 2.5B; Figure S20). This suggests that MYC and HCFC1 interact in the nucleus, and HCFC1 may be a novel co-regulator of MYC. Future investigation will reveal the importance of HCFC1 in regulating the biological functions of MYC, such as cell cycle progression and oncogenesis. This discovery illustrates how ChromNet can suggest novel protein-protein interactions within chromatin complexes.

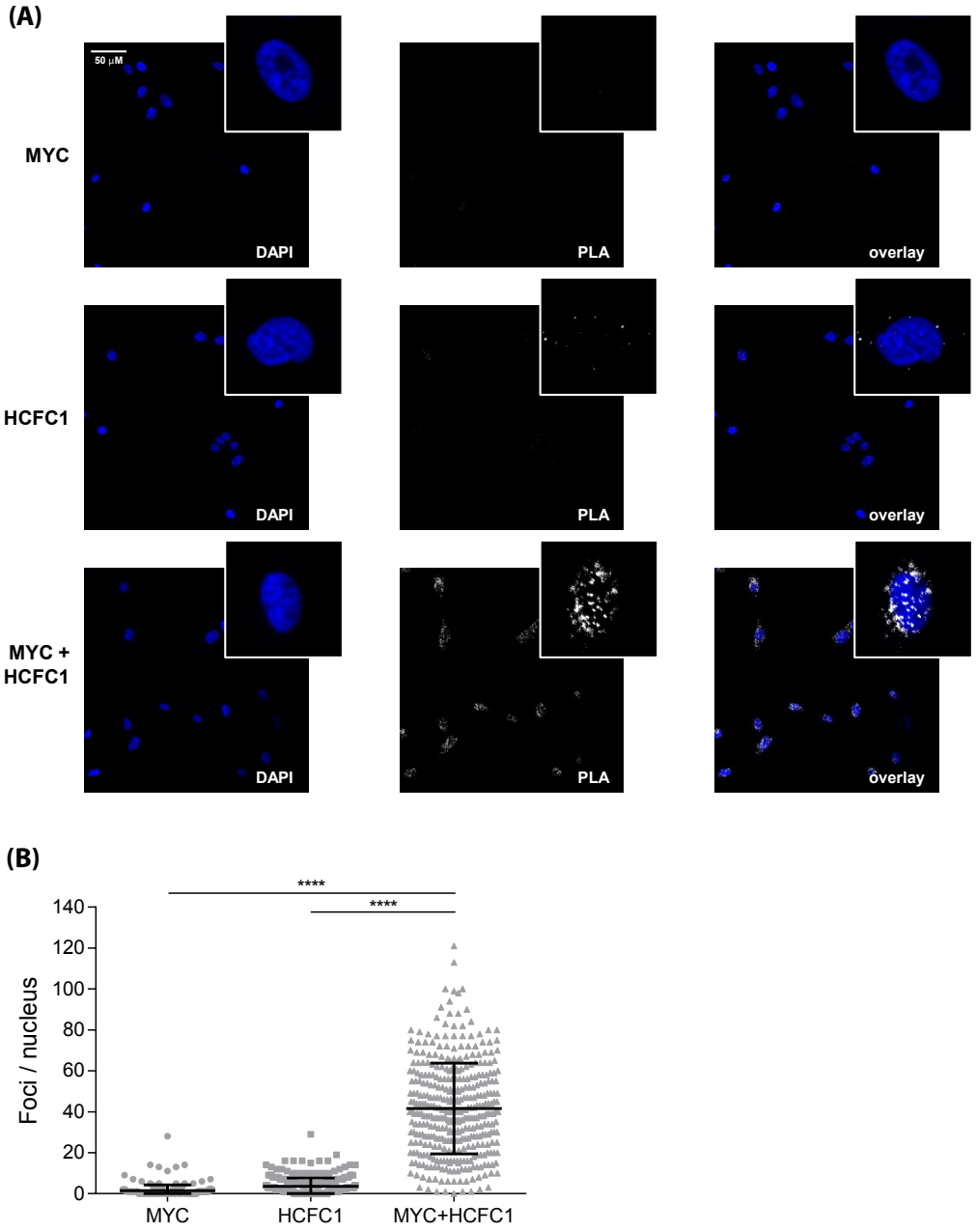


Figure 2.5: **(A)** Proximity ligation assay showing MYC and HCFC1 interaction in the nucleus. Representative micrographs show DAPI nuclear staining (left), proximity ligation signal (middle), and overlay (right) at 20× magnification, with insets at 100× magnification. *Top:* Cells probed with MYC antibody alone. *Middle:* Cells probed with HCFC1 antibody alone. *Bottom:* Cells probed with both antibodies. **(B)** Proximity ligation assay signal quantified as number of foci per nucleus, with 254, 293, and 381 nuclei quantified for the MYC antibody alone, the HCFC1 antibody alone, and both antibodies together conditions, respectively. Individual values (grey dots) and mean ± standard deviation black bars from three biological replicates are shown; **** $p < 0.0001$, one-way analysis of variance with Bonferroni post test. Quantifications for each independent replicate are shown in Figure S20.

2.2.11 Spatial embedding reveals global patterns in the human chromatin network

By integrating all ENCODE datasets from many cell types into a single network, ChromNet enables extraction of global patterns in the relationships among regulatory factors. We used multidimensional scaling [14] to embed the entire network into a 2D layout (Figure 2.6; Methods). In this embedding, the spatial proximity of two nodes is designed to reflect their distance in the network, where positive edges pull nodes closer together and negative edges push them father apart. Nodes for the same regulatory factor in different cell types form a cluster when that factor's genomic position is conserved across cell types. For example, CTCF forms a clear cluster in this manner (Figure 2.6A). Relationships between regulatory factors are represented by their proximity in the embedding. For example, MYC and MAX nodes are located in the same region; so are CTCF and RAD21. In contrast to the joint network, relationships in individual cell type-specific networks (Figure 1B top) are much less distinct (Figure S21).

Relative positions of regulatory factors in the embedded graph highlight important aspects of biology. This is especially apparent among histone marks, where there is a clear separation between activating marks such as H3K4me3 and H3K27ac on the lower right and repressive marks such as H3K27me3 and H3K9me3 on the upper left (Figure 2.6A). H3K27me3 and H3K9me3 are both repressive marks, but form distinct clusters because they target distinct regions of the genome. H3K27me3 marks facultative heterochromatin, thought to regulate temporary repression of gene-rich regions [79]. H3K9me3 marks constitutive heterochromatin, and acts as a more permanent repressor [87]. Between the active and repressive marks we find H3K36me3 and H3K79me2. H3K36me3 is closer to the inactive marks and is implicated in restricting the spread of H3K27me3 [180]. H3K79me2 varies with the cell cycle and is associated with replication initiation sites [49]. The relative position of histones and protein factors is also interesting. ZNF274 has been implicated in the recruitment of methyltranferases for H3K9me3 and is found nearby in the network [48]. EZH2 is involved in the deposition of H3K27me3 and is found between the H3K27me3 cluster and the rest of the network [175].

Positions of regulatory factor datasets reflect both their cell type identities and association with chromatin states. Highlighting the three Tier 1 ENCODE cell types shows a weak clustering of regulatory factor datasets by cell type (Figure 2.6B). K562 and GM12878 are both derived from blood cell lines and overlap spatially with one another in the network more than with H1-hESC, human embryonic stem cells. Coloring the network by correlation with chromatin state also reveals spatial patterns. We chose five (out of seven) Segway [65, 182] annotation labels that highlight distinct areas of the network (Figure 2.6C), illustrating a clear separation between active and inactive regions of the genome, and that chromatin domains are reflected in the interactions of the chromatin network. Spatially embedding regulatory factor datasets using the ChromNet network simultaneously captures many important aspects of their function, such as chromatin state, cell lineage, and known factor-factor interactions.

2.3 Discussion

Characterizing the chromatin network, the network of interactions among regulatory factors, is a key part of understanding gene regulation. ChromNet provides a new way to learn the chromatin network from ChIP-seq data. ChromNet addresses key problems encountered when learning a joint conditional dependence network from a large number of ChIP-seq datasets, such as the need to distinguish direct from indirect regulatory factor interactions while remaining robust to data redundancy. ChromNet also provides an efficient method to learn the genomic context driving an edge, which allows a more comprehensive understanding of the inferred interactions. We demonstrated that ChromNet's GroupGM network infers known protein-protein interactions in the joint chromatin network more accurately than other methods. Unlike many previous methods, ChromNet is also efficient enough to integrate thousands of genome-wide ChIP-seq datasets into a single joint network. To our knowledge, this study represents the first construction of an interaction network from all 1,451 ENCODE ChIP-seq datasets. ChromNet already scales to the number of datasets necessary to represent all 1,400–1,900 human transcription factors [174], once such data is available.

ChromNet provides a general computational framework to identify a joint dependence network from many ChIP-seq datasets. It can build a custom joint dependence network by incorporating user-provided ChIP-seq datasets or a combination of the ENCODE ChIP-seq datasets and user-provided datasets. To allow easier exploration of regulatory factor interactions and to facilitate generation of novel hypotheses, we have

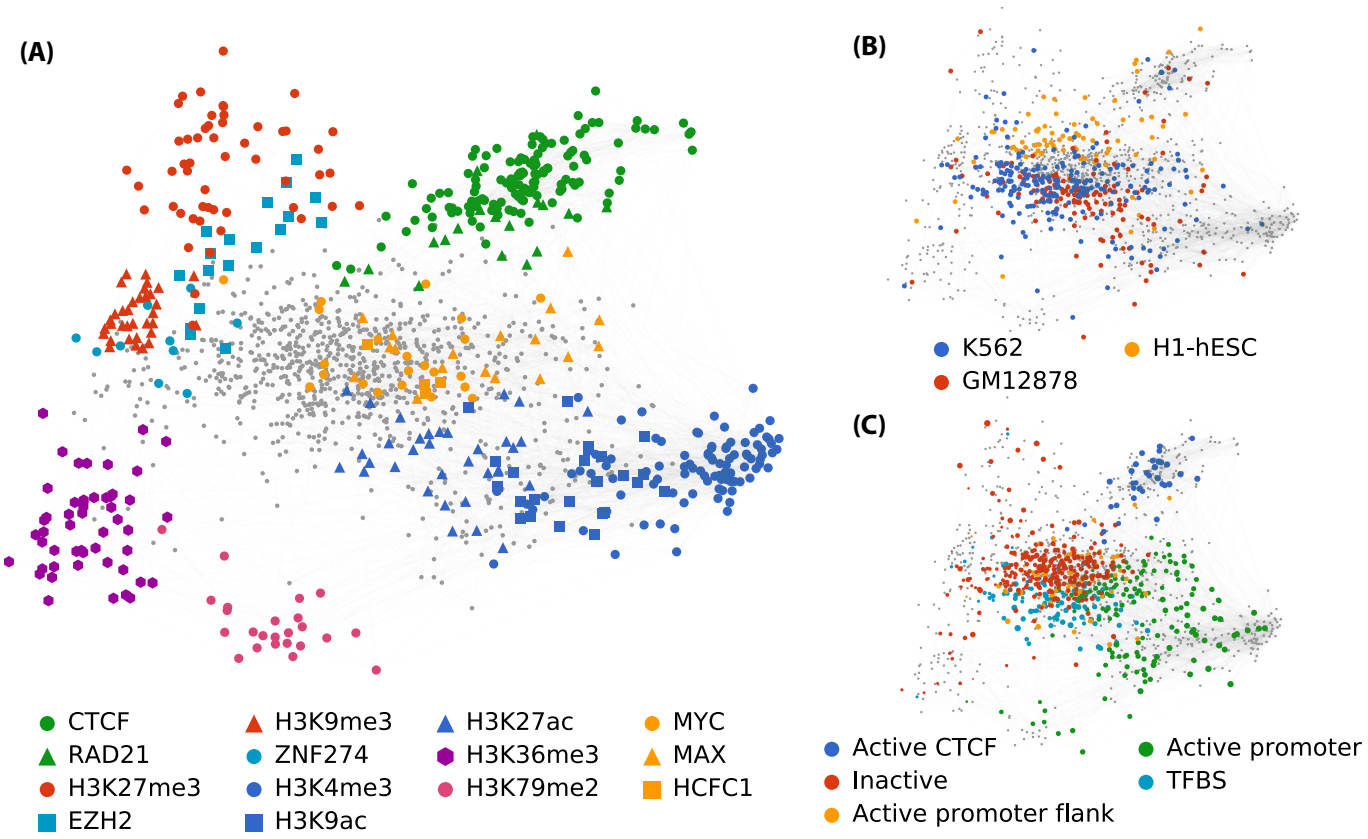


Figure 2.6: 2D embedding of the entire human chromatin network estimated by ChromNet. Spatial proximity of nodes and node groups reflects the strength of their inferred edge weights. In three views of this embedding, we have highlighted three different aspects: **(A)** Specific regulatory factors discussed in this article. **(B)** Datasets from the three ENCODE Tier 1 cell types, showing a separation of datasets by cell type. **(C)** Correlation with five Segway genome annotation labels. We only colored the datasets from cell types where the Ensembl Regulatory Build had a corresponding Segway annotation. Node size represents correlation with a label, in comparison to all other nodes assigned to that label.

created a dynamic search and visualization web interface for both the ENCODE network and networks built from custom datasets (`http://chromnet.cs.washington.edu`). By building a large model and allowing easy inspection of small sub-networks, ChromNet combines a large-scale conditional dependence model with practical accessibility.

To demonstrate ChromNet's ability to reveal novel regulatory factor interactions, we experimentally validated the interaction between the MYC and HCFC1 proteins. The biological functions of the MYC oncoprotein are complex and dependent on its protein-protein interactions. Uncovering these interactions will provide insights into MYC transcriptional complexes involved in the oncogenic process and may also reveal potential targets for anti-cancer therapies. While this manuscript was under review, the MYC-HCFC1 interaction was independently described by Thomas et al. (2015), further strengthening our validation of the interaction discovered through ChromNet and establishing HCFC1 as a bona fide interactor of MYC. Through ChromNet, we identified HCFC1 as a novel interactor of MYC that may be involved in regulating biological and oncogenic functions of MYC.

We envision several future extensions to the approach described in this article. First, while we have demonstrated the utility of applying ChromNet to ChIP-seq data alone, we plan to incorporate other data types into the network. RNA-seq expression datasets could resolve regulatory factor relationships that occur as a consequence of mutual involvement in gene expression. Incorporating feature annotations such as gene models could highlight direct interactions between factors and genomic regions of interest. The human genome's billions of base pairs provide a large sample size that allows joint comparisons of many genome-wide signals in a single model. Robust conditional dependence networks provide a benefit that is likely not limited to ChIP-seq data. Second, we plan to consider relationships between regulatory factors at genomic position offsets. Here, we considered only co-occurrence relationships within the same 1,000 bp region. To model positional ordering constraints, we can also consider relationships between a factor in one region and another factor in an adjacent or nearby region. This would allow us to learn phenomena such as promoter-associated factors preceding gene-body–associated factors. Third, just as the co-occurrence of different regulatory factors has been used to automatically annotate the genome, variations in the chromatin network at different positions may also prove useful to annotate functional genomic regions. This would also provide insight into the biological mechanisms behind specific regulatory factor interactions and the chromatin states in which they occur.

2.4 Methods

2.4.1 Data processing

ENCODE has the largest collection of high-quality ChIP-seq datasets [29], and continues depositing new datasets. ENCODE has processed many ChIP-seq datasets through a uniform pipeline. However, we reprocessed all the datasets from raw ChIP-seq reads (Figure 2.2) for two reasons. First, this allowed us to incorporate datasets not available yet through ENCODE's uniform pipeline. Second, specifying our own pipeline makes it easier to process external users' data in an identical way. This facilitates adding ChIP-seq datasets that are not from the ENCODE project to the ChromNet network.

We aligned reads from 3,574 FASTQ files to GRCh38/hg38 [54] using Bowtie2 [88]. We grouped BAM files by dataset using metadata from the ENCODE web site [42]. Then, we pooled and processed BAM files using a custom binning method that counts the number read starts in each of 3,209,287 1,000 bp bins covering all contigs in GRCh38/hg38. Binning all count datasets yielded a $X \in \mathbb{Z}_*^{3,209,287 \times 1,451}$ count-valued *data matrix*. Each bin has a corresponding row in the matrix. We interpreted each of the 3,209,287 rows as a sample from a set $\mathcal{X} = \{X_1 \dots X_p\}$ of $p = 1{,}451$ count-valued random variables representing occupancy of each regulatory factor at a given position. Using this interpretation, we computed a sample correlation matrix $\hat{\Sigma} \in \mathbb{R}^{p \times p}$ among the standardized variables in $\mathcal{X}$. To create the *correlation network*, we set the weight of every edge between two datasets i and j equal to the corresponding entry $\hat{\Sigma}_{i,j}$ in the sample correlation matrix. This captures the pairwise linear dependence between two datasets (Figure 2.1A; bottom left).

2.4.2 Generation of simulated data

A large-scale simulated dataset was generated to validate the ability of ChromNet to recover interactions from raw count data. Representing the conditional dependence among large numbers of count variables for the purpose of simulation is not trivial. It is important that the model is not overly simplistic, but also still interpretable. Here we use a multivariate Gaussian distribution to represent the means of marginal Poisson distributions, threshold the values when they fall below zero, and add additional negative binomial distributed noise to represent random reads unrelated to regulatory factor localization.

In this model the count for a ChIP-seq dataset j, c_j, at a given position is described by

$$c_j = rate_j + \epsilon_j, \tag{2.1}$$

where $rate_j \sim \text{Poisson}(\max(0, s_j))$ and $\epsilon_j \sim \text{NegativeBinomial}(r, p)$. The signal follows a thresholded normal distribution $s_j \sim \max(N(\mu, \Sigma), 0)$. The background noise ($r = 25, p = 0.9$) and the parameters of the normal distribution are all fixed during the course of the simulation. Σ^{-1} represents the structure of the underlying conditional dependence network.

The inverse covariance matrix of the simulated data Σ^{-1} was randomly generated with a sparsity of 10%. In addition, datasets were grouped into complexes of size one (60%), two (20%), or three (20%) to represent the type of close coupling observed in real data among some factors. The correlation within complexes was set to be between 0.8 to 0.9 to match the magnitudes of high correlations observed in real data (Figure S3). A total of 80 complexes were simulated across 200,000 positional samples. This results in 126 experiments and 200,000 samples. To better model the complexities found in real data sets we added dependency between nearby samples by replacing each $rate_j$ with $\frac{1}{20}rate_{j-1} + \frac{9}{10}rate_j + \frac{1}{20}rate_{j+1}$. This caused nearby bins to be more similar to each other and thus the samples are not independently and identically distributed. Since larger correlations between regions of the genome are also present due to batch effects or other confounding factors we added one of 8 different random genome-wide batch effects to each of the 126 datasets.

The resulting marginal count distributions from this model are visually similar to those observed in real data (Figure S12). Because we based the correlations between datasets on a (largely transformed) multivariate normal distribution we can treat datasets connected in the underlying generative model as true connections and seek to recover them using a variety of methods. The results of this analysis are shown in Figure S13, which is consistent with Figure 3, where the group graphical model performs better than alternative approaches including correlation, inverse correlation and partial correlation.

2.4.3 Efficient estimation of conditional dependence from count data

Given datasets drawn from a set $\mathcal{X}$ of count-valued random variables, learning an exact joint model that captures the dependency structure of these datasets could be challenging. Although there are a variety of multivariate count distributions, all are either overly restrictive or challenging to estimate for large numbers of variables [178]. A common alternative is to use a multivariate Gaussian distribution and some type of transform on the marginals to make them more Gaussian such as *sqrt* or *asinh*. Since count data is often heteroscedastic, where variance increases with higher counts, these transforms squash higher values, making the distribution more symmetric. This causes the least squares error term to focus less on high valued samples and proportionately more on lower values. Interestingly, for ChIP-seq datasets this is not desirable because higher values are more likely to represent strong signal while lower values are more likely driven by noise.

Because of its efficiency and interpretability we used a multivariate Gaussian approximation to the count data for ChromNet. We also chose to use untransformed raw read counts in the model. This choice was based on observing a clear decrease in performance when using transforms designed to mitigate heteroscedasticity (Figure S1). An additional benefit of using a multivariate Gaussian is that it can also serve as a reasonable approximation to a Markov random field distribution. This allows for the comparison with other methods designed to work strictly with binary data (Figure S22; Figure S23; Supplementary Note 3).

To create the *inverse correlation network* (Figure 2.3), we began by inverting the sample correlation matrix $\hat{\Sigma}$ to get an inverse sample correlation matrix $\hat{\Sigma}^{-1}$ [108, 92]. We then set the weight of every edge between two datasets i and j equal to the corresponding entry $\{\hat{\Sigma}^{-1}\}_{i,j}$. This inverse correlation network captures the pairwise linear dependence between two datasets when conditioned on all other variables in the network.

It should be noted that partial correlation is very similar to inverse correlation and has been used before by Lasserre et al. to effectively model connections between histone marks from human ChIP-seq data (using rank-transformed data from gene start sites) [90]. The matrix of partial correlations, P, is a renormalization of inverse correlation $P = -D^{-1/2} * \Sigma^{-1} * D^{-1/2}$ where D is the diagonal matrix of Σ^{-1}. A direct application of partial correlation to all ENCODE data suffers from the same issues as inverse correlation, performing slightly worse in the recovery of known protein-protein interactions (Figure S4). We chose to use inverse correlation as the foundation of the group graphical model (GroupGM) because the proof that GroupGM recovers the correct edge weights in the presence of near perfect redundancy does not hold when applied the to partial correlation matrix (Supplementary Note 2).

One additional concern when applying a Gaussian graphical model to ChIP-seq data is that the values at each 1,000bp bin in the genome are not independent of each other. Fortunately, while this may reduce the power of the model (i.e. it will need more samples), it does not bias the model. This is because the edges of a Gaussian graphical model can be interpreted in terms of linear regression coefficients. Standard linear regression coefficients are unbiased even when samples are not statistically independent when the data follows a linear relationship. To validate this on ChIP-seq data, and to confirm that any loss of power is unimportant we evenly subsampled the data at progressively larger intervals. We found that performance when recovering known protein-protein interactions does not degrade until we subsample 100-fold (Figure S24).

2.4.4 Group graphical model

To create the *Group Graphical Model (GroupGM) network*, we began with the inverse correlation matrix created above. We extended the idea of pairwise relationships to groups of datasets by considering a set $\mathcal{G}$ of q groups chosen by hierarchical clustering (see below). This effectively allows edges to express relationships between groups of variables. We let $\hat{G} \in \mathbb{R}^{q \times q}$ represent pairwise interaction strengths between all groups in the model. For any two groups i and j in the model their weight is given by the sum of entries between them in the inverse correlation matrix (Figure 2.1C).

$$\hat{G}_{i,j} = \sum_{k \in \mathcal{G}_i, l \in \mathcal{G}_j} \hat{\Sigma}^{-1}_{k,l} \tag{2.2}$$

We prove that Equation 2.2 correctly maintains the original edge magnitude in the case of redundancy (Supplementary Note 2).

To select the set $\mathcal{G}$ of groups, we used complete-linkage hierarchical agglomerative clustering of the correlation matrix [61]. This clustering method starts by merging the two groups with the smallest maximum correlation distance between their datasets, then continues recursively until all groups have been merged. The use of hierarchical clustering eliminates the need to chose a fixed arbitrary number of clusters in advance. From the clustering results we chose all the leaf and internal nodes from the clustering algorithm as groups $\mathcal{G}$. Then, G became a $q \times q$ matrix filled according to Equation 2.2, where $q = 2p - 1$ (the total number of internal and leaf nodes). This method avoids comparing all possible subsets of datasets, which would make calculating G prohibitively expensive. Since groups with low correlation are less likely to cause the collinearity problem (Figure 1C), we only consider groups with a correlation greater than 0.8 which captures 53% of all the multi-factor groups formed by the hierarchical clustering (Figure S3).

Since GroupGM uses the cluster assignments to mitigate strong redundancy, clustering accuracy is most important for tightly correlated datasets. When two datasets are highly correlated, it is important to group them together to mitigate the outcome of correlated datasets in network inference. When two datasets are only mildly correlated, the effects of their redundancy will also be mild, so it is less important to group them together. Hierarchical clustering is an attractive choice because it starts by creating groups among the most correlated datasets.

2.4.5 Computing the genomic context that drives a network edge

The conditional dependence relationships represented by an edge in ChromNet can occur primarily in certain genomic regions. Here we seek to identify what parts of the genome (i.e. samples) drove the creation of an edge in ChromNet. Understanding what positions in the genome caused ChromNet to estimate a network edge provides insight into the genomic regions driving the relationship.

The most natural way to define the influence of a genomic position (i.e. sample) on an edge is as the difference in edge value between when we observe a position and when we do not observe a position in the genome. If implemented directly this could easily become computationally intractable since it involves relearning the entire model for every position in the genome. For a highly optimized implementation on 16 cores, computing the correlation matrix takes approximately two minutes, which would lead to a run-time of over 12 years for 3,209,287 binned genomic positions. This can be sped up dramatically by using rank-1 matrix updates to avoid recalculating most of the correlation matrix. This results in a much faster method, where the slowest step is the inversion of the correlation matrix. However, computing this inversion for each genomic sample still leads to over four days of computation on recent high performance servers. Pre-computing this information is also undesirable since it would create 54TB of largely incompressible data for all group edges. Below we show that for the ChromNet model, the calculation of a genomic position's impact on an edge can be made extremely efficient. The ideas are similar to those used in efficient leave-one-out cross validation implementations for linear models.

Removing a genomic position and computing the new inverse correlation matrix can be written in terms of a rank-1 update and the inverse correlation matrix before the position (sample) is removed. This equation holds under the assumption that removing the sample does not change the mean of the data. Let Σ be the correlation matrix of all the data, and $\bar{\Sigma}$ be the correlation matrix with the sample removed. Let u be the column vector representing the sample to be removed (already mean centered). Letting D be a normalizing diagonal matrix $D_{i,i} = \sqrt{1 - u_i^2}$ we get:

$$\bar{\Sigma} = (D^{-1}(\Sigma - uu^T)D^{-1})^{-1} \tag{2.3}$$

$$= D(\Sigma - uu^T)^{-1}D \tag{2.4}$$

$$= D(\Sigma + uBu^T)^{-1}D \qquad B = -1 \tag{2.5}$$

$$= D(\Sigma^{-1} - \Sigma^{-1}u(B^{-1} + u^T\Sigma^{-1}u)^{-1}u^T\Sigma^{-1})D \qquad \text{Woodbury formula} \tag{2.6}$$

$$= D(\Sigma^{-1} - \Sigma^{-1}u(-1 + u^T\Sigma^{-1}u)^{-1}u^T\Sigma^{-1})D \tag{2.7}$$

$$= D(\Sigma^{-1} - v(-1 + u^Tv)^{-1}v^T)D \qquad \Sigma^{-1}u = v \tag{2.8}$$

$$= D(\Sigma^{-1} - \frac{vv^T}{u^Tv - 1})D \tag{2.9}$$

$$\tag{2.10}$$

Included in the ChromNet software release is an optimized implementation utilizing the above inverse rank-1 update formulation. It can solve 40,000 model updates to the full joint chromatin network per second, which leads to a run-time of just over one minute for a single group edge over the human genome. The output is the effect each genomic position has on an edge when that position is added to the dataset. This information can be used to examine the highest impact positions and determine the genomic context driving an edge (Figure S19).

2.4.6 Visualization of the hierarchical chromatin network

To enable exploration of the chromatin network, we built an interactive visualization tool (`http://chromnet.cs.washington.edu`). This tool displays the nodes and edges of the chromatin network using a real time force model (Figure 2.4C). The tool's responsive interface lets users control which nodes and edges it displays. It immediately changes its display after a user types a search term to restrict displayed nodes. It also immediately changes its display when a user moves a slider that controls the minimum strength of a displayed

edge. Our visualization tool facilitates exploring the chromatin network without excessive visual distraction.

The ChromNet visualization tool displays hierarchical groups from GroupGM by shading areas that enclose a group's members. It shades these areas with some amount of transparency. It displays the strongest groups with the highest opacity. The parents of two connected groups in the GroupGM hierarchy are themselves very likely connected. Therefore, for clarity we hide redundant parental edges.

To find a reasonable lower bound for the user-defined strength threshold, we examined the relationship between edge magnitude and known physical interactions. Within cell type edges from all cell types were sorted by magnitude and then binned. For each bin we computed the number of edges matching low throughput physical interactions in BioGRID and plotted how this varied over the bins. This enrichment curve suggested a lower bound of 0.2 to capture only edges enriched for known interactions (Figure S25).

2.4.7 Fold enrichment reflects both type I and type II error rates

The fold enrichment is a single quantity that captures the effects of both type I and type II error rates. This can be seen from the definition of fold enrichment:

$$fold\ enrichment = \frac{\#\ of\ correct\ edges}{\#\ of\ correct\ edges\ by\ random} \tag{2.11}$$

$$= \frac{TP}{(\#\ network\ edge\ predictions) \times (\#\ BioGRID\ interactions)/N} \tag{2.12}$$

$$= \frac{TP \times N}{(TP + FP) \times (TP + FN)} \tag{2.13}$$

where N is the total number of possible edges, and TP, FP, and FN refer to the number of true positives, false positives, and false negatives, respectively. The fold enrichment is inversely proportional to the number of false positives (type I error) and number of false negatives (type II error). The type I error rate is equal to (type I error) / (total number of BioGRID interactions), and type II error rate is equal to (type II error) / (total number of interactions – number of BioGRID interactions). Since the denominators of the type I and type II error rates are fixed numbers, we can say that the fold enrichment is inversely proportional to the type I and type II error rates.

2.4.8 A conservative bootstrap estimate of protein-protein interaction enrichment variability

We estimated the variability of enrichment for known protein-protein interactions in the chromatin network (Figure 2.3) using bootstrap re-sampling over regulatory factors. We performed re-sampling over regulatory factors, and not over edges or individual datasets, because valid bootstrap re-sampling assumes independent and identically distributed samples. If we had re-sampled over the edges, we would have estimated a much smaller variability. This is because edges do not vary independently, and changes in a single dataset can affect all edges connected to that dataset. Variation specific to a single regulatory factor would affect all datasets measuring that factor. Those individual datasets, therefore, lack the independence assumed by the bootstrap sampling.

Under a regulatory factor bootstrap, we might sample a widely-measured regulatory factor a number of times. For example, ChromNet contains 130 CTCF datasets. Every time we sample CTCF, we add all 130 of these columns (where a column represents a variable in the data matrix X) to the bootstrap data matrix. Adding many datasets in unison greatly increases variability in the re-sampled data matrix. This yields conservative high variability estimates, ensuring that enrichment performance is not solely due to a few commonly measured factors. Using these bootstrap samples, we compared the area under the enrichment rank curves (Figure 2.3A,B) between methods. The statistical significance of GroupGM's improvement was quantified as the fraction of bootstrap samples where GroupGM outperformed the other methods (Figure S10; Figure S14).

2.4.9 Proximity ligation assay

We seeded 2.5×10^4 MCF10A cells (a kind gift from S. Muthuswamy, Princess Margaret Cancer Centre) onto glass cover slips. After one day, we fixed cells in 2% paraformaldehyde, permeabilized the cells, and blocked them with bovine serum albumin. We then incubated the cells overnight with a mouse monoclonal antibody against MYC (1:25; C-33, Santa Cruz Biotechnology, Dallas, TX) and a rabbit polyclonal antibody against HCFC1 (1:50; A301-400, Bethyl Laboratories, Montgomery, TX) overnight. Then, we incubated cells with Duolink In Situ PLA anti-mouse MINUS and anti-rabbit PLUS probes (Sigma-Aldrich, St. Louis, MO). We processed cells using Duolink In Situ Detection Reagents Red following manufacturer's instructions (Sigma-Aldrich, St. Louis, MO). We imaged six fields of view per slide with a LSM700 confocal fluorescence microscope (Zeiss, Oberkochen, Germany). We unbiasedly quantified proximity ligation assay signal per nucleus (as defined by DAPI staining) using the software ImageJ [150].

2.4.10 Embedding the full chromatin network into a single plot

Embedding a graph into a space involves defining distances between all nodes in the graph. Because the GroupGM is inherently multi-scale, we sought a distance metric that accurately represented forces between individual nodes, and between all possible node groupings. In GroupGM, the edge weight between two groups is the sum of the conditional dependence weights between all the individual datasets of those groups.

A common method of computing graph distances that accounts for the total effect of all edges between two groups is the resistance distance [83]. The name is derived from an interpretation of the distance as the electrical resistance between two nodes in the graph where edges are viewed as wires. This can be computed as:

$\Omega_{i,j} = \Gamma_{i,i} + \Gamma_{j,j} - \Gamma_{i,j} - \Gamma_{j,i}$,

where Γ is the inverse of the graph Laplacian. While at first glance the resistance distance may seem like an arbitrary metric to use for node distances, upon closer inspection we find striking parallels between it and Gaussian graphical models. First note that the weighted graph laplacian [121], L, is defined as:

$L = W \otimes (D - A)$,

where D is a diagonal matrix of edge degrees, A is the binary adjacency matrix of the graph, W is a matrix of positive edge weights, and $\otimes$ represents element-wise multiplication. A general Gaussian graphical model has a complete graph so A will be all ones, and D will be constant on the diagonal, the edge weights will be symmetric and can be positive or negative. Positive edge weights will lead to negative off diagonal entries in L, just as positive connections in the GGM will lead to negative off-diagonal entries in $\Theta = \Sigma^{-1}$. So by allowing W to contain negative entries we can view Θ as a type of graph laplacian.

Viewing Θ as a type of graph Laplacian allows us to compute the resistance distance by setting $\Gamma = \Theta^{-1}$. Simplifying gives the following: $\Omega_{i,j} = 1 - \Theta^{-1}_{i,j}$.

So the resistance distance is just a constant offset of the correlation matrix of the network. This means that if we are trying to compute distances between nodes in a graph represented by the inverse correlation matrix, correlation is a very natural distance measure. We note however that unlike the original data correlation matrix this matrix is computed from the inverse of the edge weights matrix. This causes a difference because we threshold small edge values that are likely to only represent noise. We chose this threshold to maximize the visual clarity of the network, which lead to a threshold of 0.01.

We overlaid chromatin state annotation on the graph embedding by computing the correlation between each dataset and each Segway [65] region from the Ensembl Regulatory Build for GRCh38/hg38 [182]. We drew a separate network labeling for each region by sizing each dataset node by its correlation with that Segway region. We normalized the size of the largest node in each network to a constant value and overlaid three of these network colorings (Figure 2.6C).

2.4.11 Availability of supporting data

ChromNet is freely available as a ready-to-use package under an Apache license at: `https://github.com/slundberg/ChromNet`
Supporting data including a preprocessed data matrix from all human ENCODE ChIP-seq data is linked from the code repository and at `http://dx.doi.org/10.5281/zenodo.45900`. The microscopy data which we used for validation of the MYC – HCFC1 interaction is available at `http://dx.doi.org/10.5281/zenodo.45768`.

2.4.12 Acknowledgements

The results presented in this chapter have been released as a paper in collaboration with my co-authors [103]: William B. Tu, Brian Raught, Linda Z. Penn, Michael M. Hoffman, and Su-In Lee.

We would like to acknowledge William S. Noble, Zhiping Weng, R. David Hawkins, W. Larry Ruzzo, and Maxwell W. Libbrecht for their helpful feedback during the development of ChromNet.

This work was supported by a National Science Foundation (NSF) Graduate Research Fellowship (DGE-1256082) to SML; NSF (DBI-1355899) to SIL; Natural Sciences and Engineering Research Council of Canada (RGPIN-2015-03948) to MMH; Canada Research Chair in Molecular Oncology to LZP, Canadian Institute for Health Research (MOP-275788) to LZP and BR; and a Canadian Breast Cancer Foundation Ontario Region Doctoral Fellowship to WBT. Cloud computing resources for this research were generously provided by Google.

2.5 Supplementary information

2.5.1 Supplementary figures

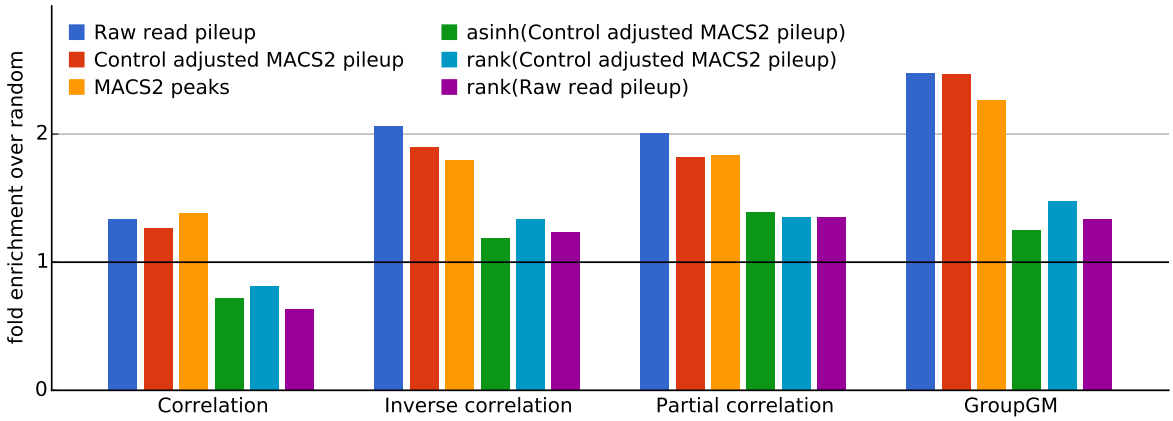


Figure S1: Enrichment of BioGRID-supported edges within all cell lines across four different modeling approaches and six different pre-processing methods. For raw pileup (blue) we binned raw Hg38 mapped read start sites. For control-adjusted pileup (red), we took MACS2 pileup output and normalized by a paired control. For MACS2 peaks (yellow), we used MACS2 with paired controls and a lenient peak threshold (varying the threshold produced similar results, see Figure S2). For each of the data transforms we applied the given function to the value in each 1,000 bp bin.

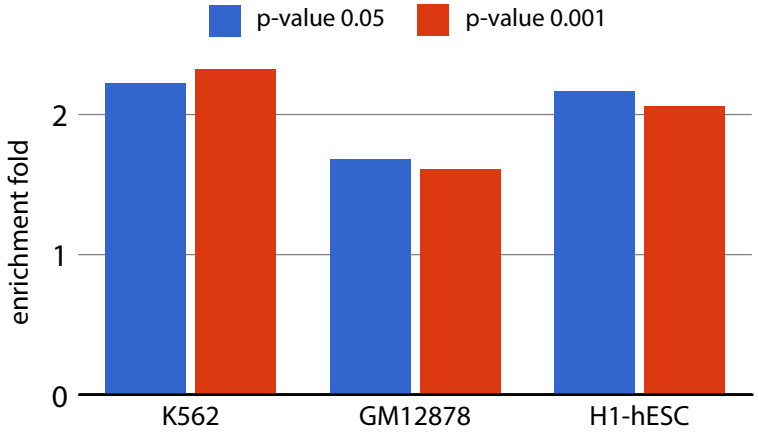


Figure S2: Enrichment of BioGRID-supported edges in a GroupGM created from a binary data matrix of MACS peaks called at two different thresholds ($P < 0.05$, blue; $P < 0.001$, red). Within the larger network we examined BioGRID enrichment among ENCODE tier 1 cell lines: K562 myeloid leukemia cells, GM12878 lymphoblastoid cells, and H1-hESC embryonic stem cells.

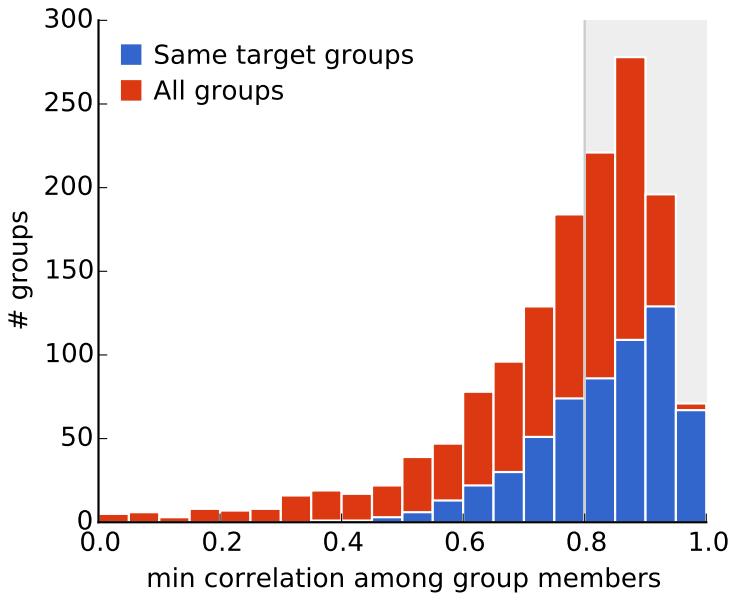


Figure S3: Distribution of group widths within ChromNet. Groups containing only a single regulatory factor type tend to have a stronger correlation, but many heterogeneous groups also show tight correlations. The gray region highlights the groups we allowed in the ChromNet network used for analysis in this paper.

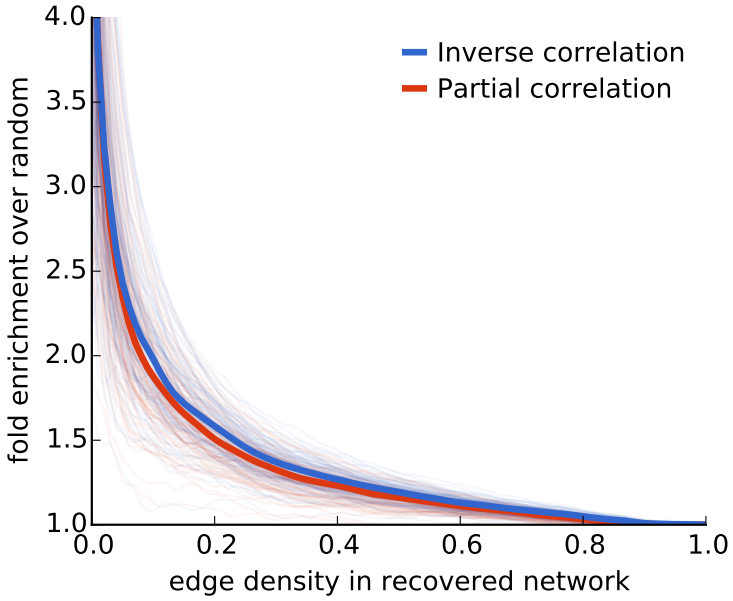


Figure S4: Comparison of BioGRID enrichment performance between inverse correlation and partial correlation. Partial correlation can be viewed as a re-normalized version of the inverse correlation matrix, but is not used in ChromNet since the group graphical model proof is specific to the inverse correlation matrix.

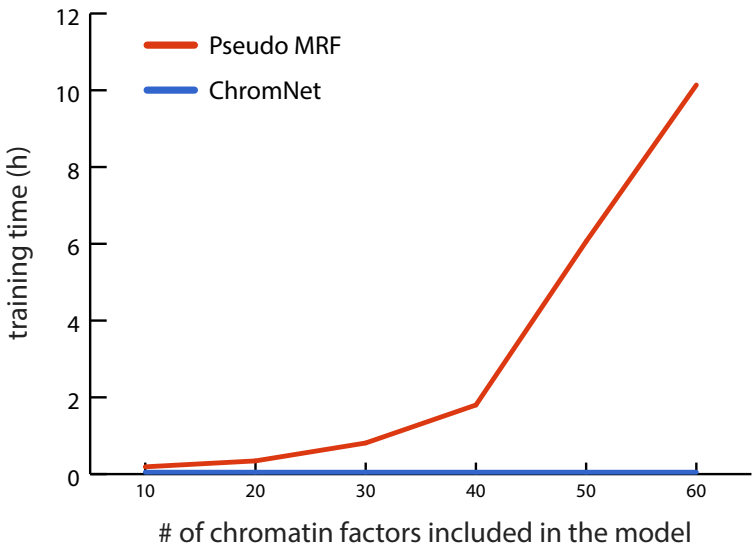


Figure S5: Wall clock training time to fit the pairwise pseudo-likelihood Markov random field model from Zhou et al. [184] on ENCODE data. As the number of variables in the model increases, the method's running time becomes infeasible. We tuned regularization parameters using the same 61 warm-started optimizations used in [184]. We ran this test on a 12-core Intel Xeon CPU E5645 2.40GHz computer with 24 GB of random access memory.

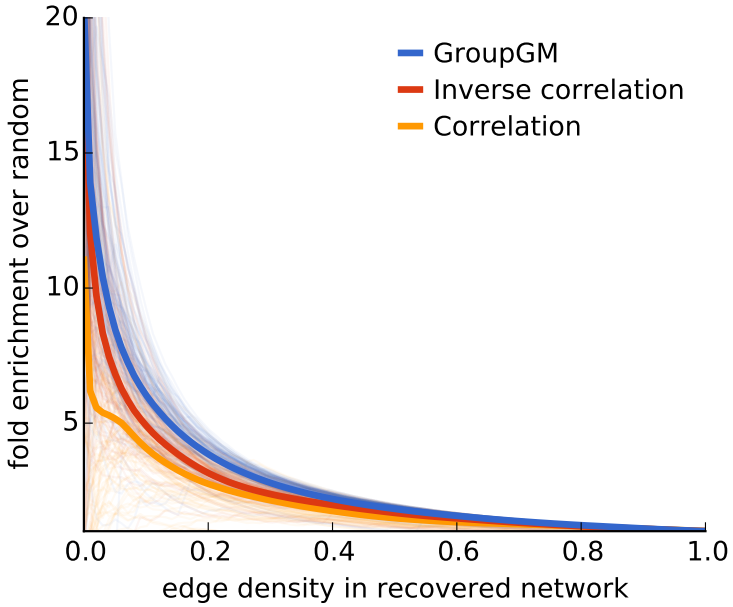


Figure S6: Just as BioGRID was used to validate the performance of protein-protein connections in ChromNet (Figure 3A), histone mark writers can be used to validate protein/histone-mark connections. All histone-mark/writer combinations were taken from the HIstome database [77] and enrichment for these edges among all protein/histone-mark connections was calculated.

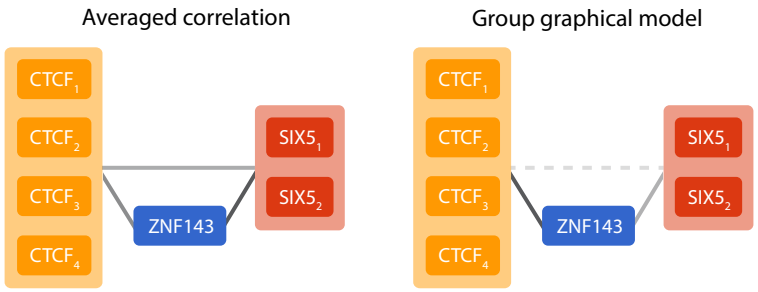


Figure S7: The factors CTCF and SIX5 closely associate specifically when ZNF143 is also present. This causes ZNF143 to mediate the interaction between CTCF and SIX5. The presence of ZNF143 can be also be viewed as the “context" in which CTCF and SIX5 co-localize (Figure S19).

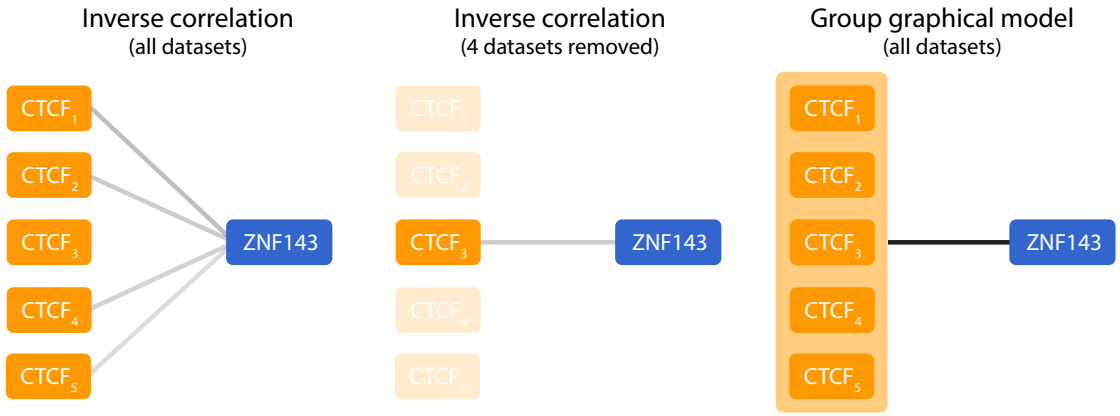


Figure S8: Illustration of how GroupGM helps with problems in inverse correlation caused by collinearity. When using all 1,451 datasets inverse correlation recovers edges between ZNF143 and four of the five CTCF datasets in K562 *(left)*. When the four datasets with an edge to ZNF143 are removed the other dataset gets an edge to ZNF143 stronger than any of the original four datasets *(middle)*. This means that redundancy with the other datasets caused the other dataset to be ignored, even though it was strongly related to ZNF143. In contrast the group graphical model recovers a much stronger edge between CTCF and ZNF143 *(right)*.

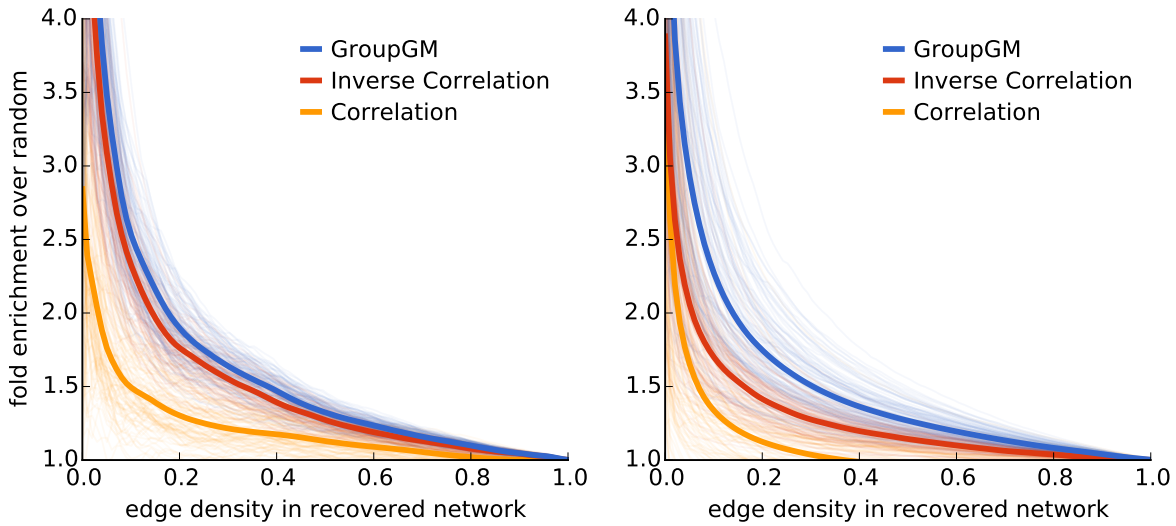


Figure S9: Some datasets target the same factor in the same cell type/condition. Here we average those datasets under the assumption that the distinction between them is not important. GroupGM still provides an improvement even in the absence of these potentially redundant datasets both within cell types (P-value = 0.002) and between cell types (P-value = 0.021). The left figure is enrichment for BioGRID supported edges within all cell types, while the right figure is enrichment between all cell types.

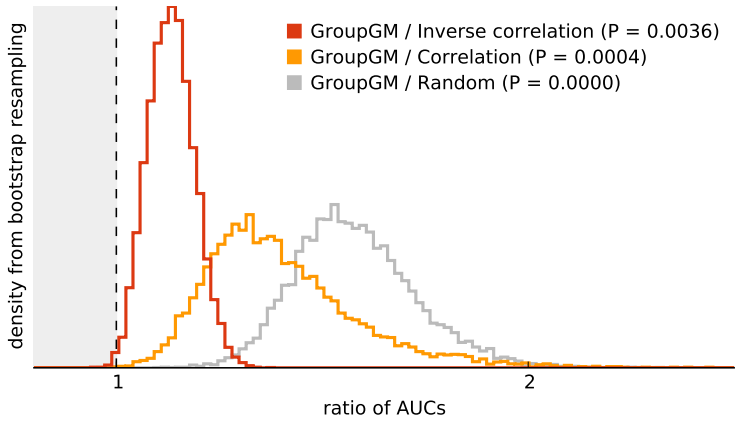


Figure S10: Histogram of area under the curve (AUC) ratios comparing enrichment of BioGRID-supported edges in a GroupGM network versus networks created by inverse correlation (red), correlation (yellow), and random edge score assignment (grey). Specifically, we compared the area under enrichment–edge density curves from 10,000 bootstrap samples from regulatory factors, excluding edges between different cell types (Figure 2.3A top). *P*-values represent the fraction of bootstrap samples with a ratio of AUC's less than 1. Being less than 1 means that GroupGM performed worse than the alternative method.

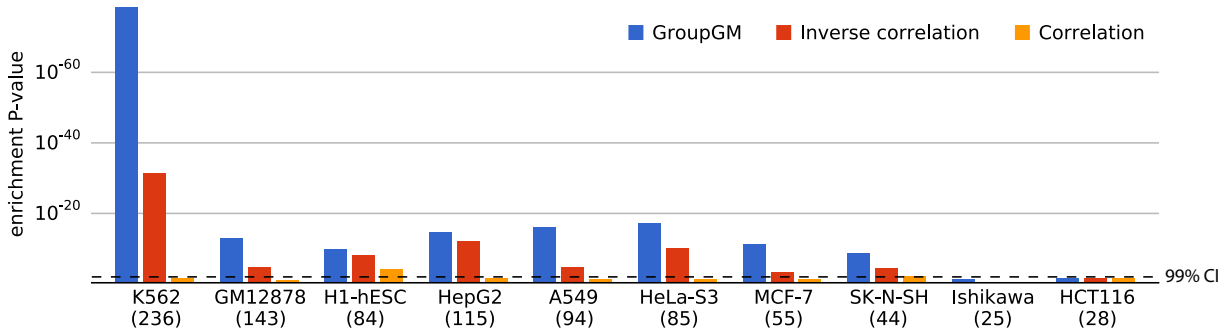


Figure S11: One-sided hypergeometric test negative $\log_{10} P$-values for enrichment of BioGRID-supported edges within cell types that have 25 supported edges or more (Figure 2.3C). The hypergeometric test is less conservative than the bootstrap approach used in Figure S10 and Figure S14. Cell types with more datasets will likely have more significant P-values, since they have more edges to compare. Dashed line indicates 99% confidence level ($P = 0.01$). Beneath each cell type name is the number of datasets in that cell type.

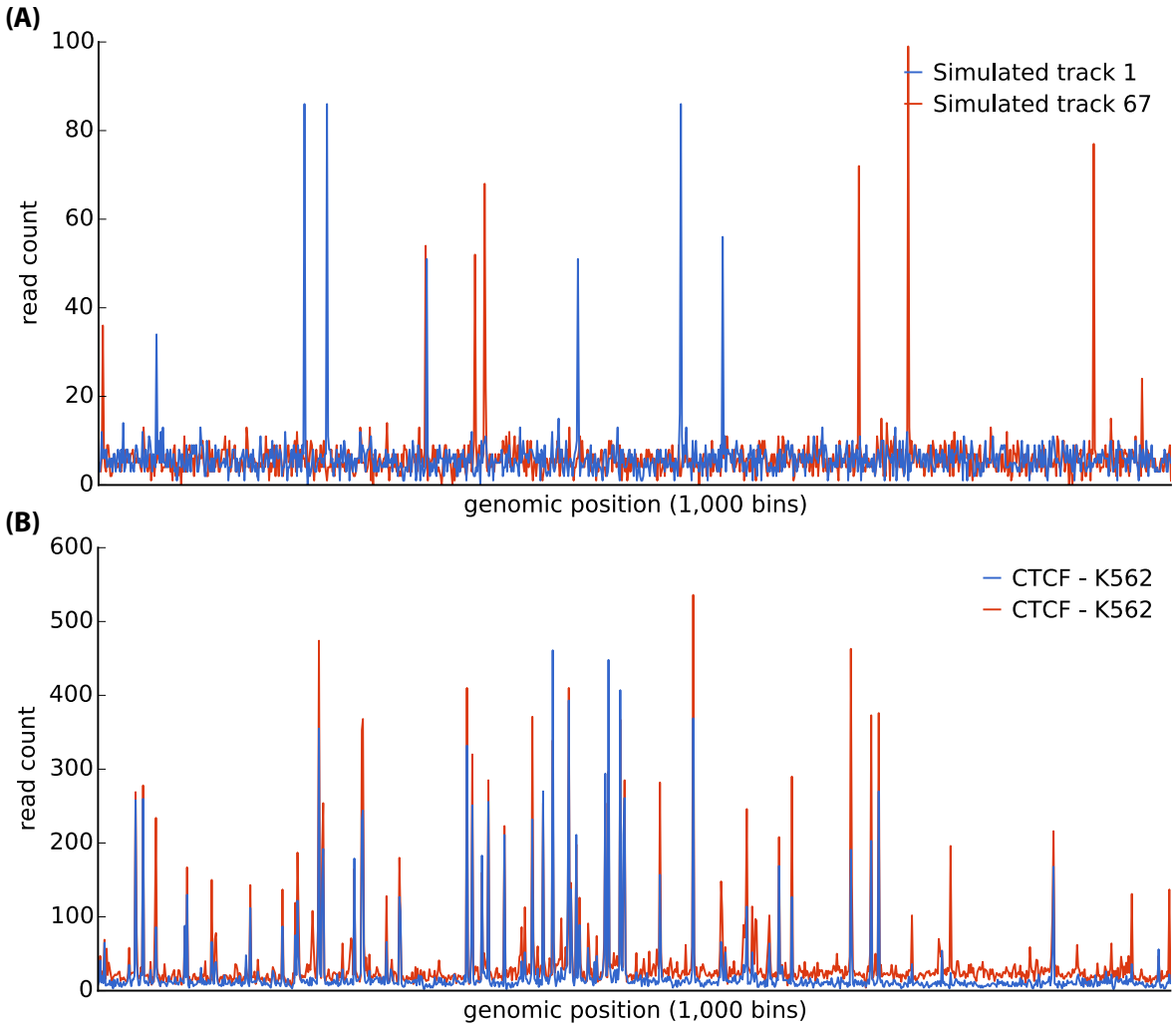


Figure S12: Visual comparison of simulated data and read data from two CTCF ChIP-seq tracks. While not identical, the simulated data is designed to be qualitatively similar to the distribution of real data tracks.

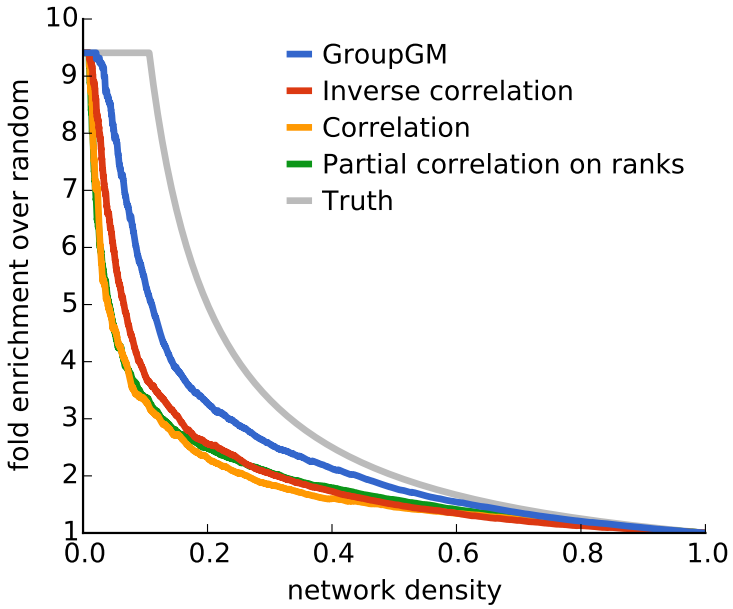


Figure S13: Results from a simulated data study with 126 datasets and 200,000 samples. Complexes of one, two and three simulated proteins were created where within complex correlations matched correlations observed in real data. Each method was then run and compared to known simulated interactions between complexes.

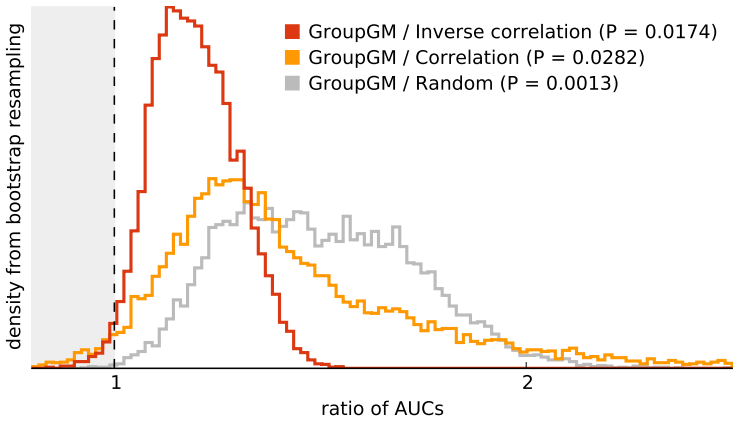


Figure S14: Histogram of area under the curve (AUC) ratios comparing enrichment of BioGRID-supported edges in a GroupGM network versus networks created by inverse correlation (red), correlation (yellow), and random assignment (grey). Specifically, we compared the area under enrichment–edge density curves from 10,000 bootstrap samples from regulatory factors, including edges between different cell types (Figure 2.3A bottom). Variability was higher than in an examination of edges within cell types (Figure S10). This is because resampling regulatory factors measured in many cell types alters many edges across cell types.

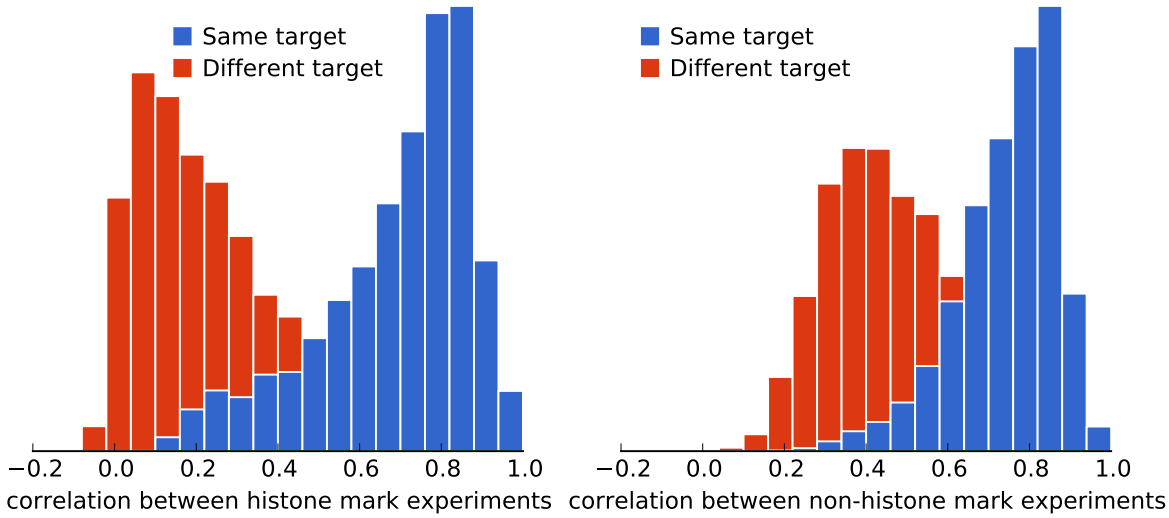


Figure S15: Correlations of the same factor between different cell types. Correlations between the same histone marks in different cell types is shown on the left, while correlations between the same non-histone factors is shown on the right. The clear bias towards positive correlation is likely the result of regions of consistent chromatin accessibility and mappability between all ChIP-seq datasets.

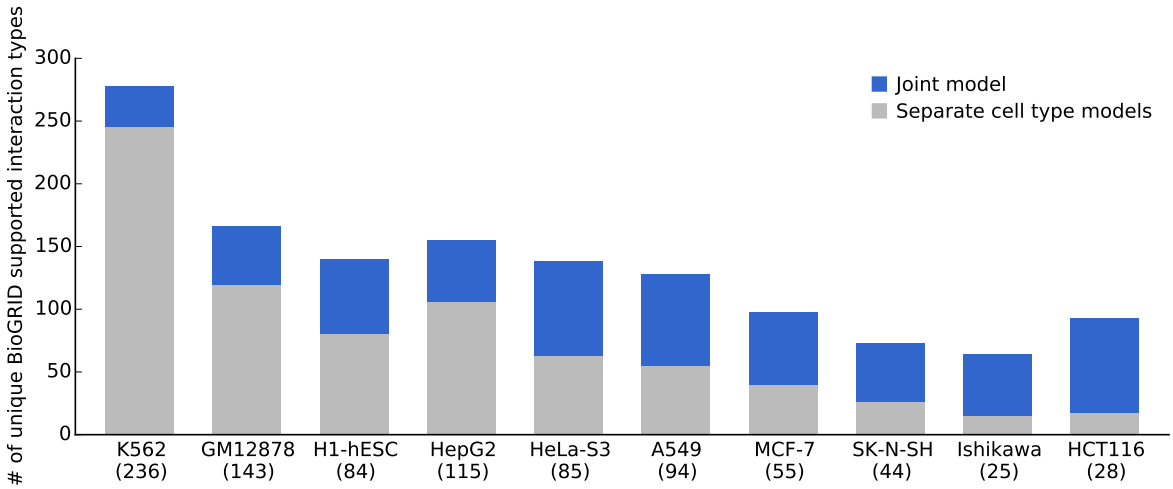


Figure S16: A joint model allows comparison with datasets not only within a single cell type but also across cell types. Here the increased number of BioGRID supported unique factor-factor interaction types detected at a threshold of 0.2 by a joint model is shown for each cell type.

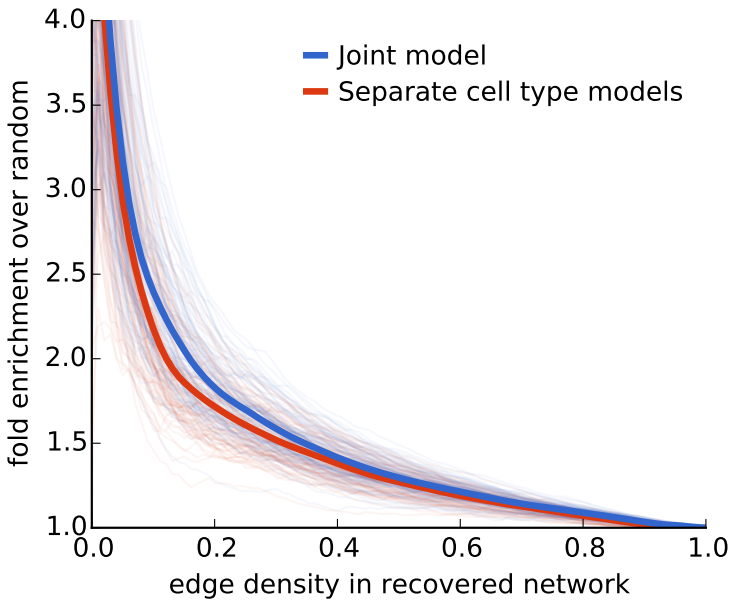


Figure S17: Comparison of a joint GroupGM of all cell types vs. individually learned GroupGM networks for each cell type. cross-cell-type edges from the joint model are ignored and only edges common to both networks are compared for enrichment of BioGRID supported edges. The joint model is marginally better than individual cell type models (P-value = 0.0672).

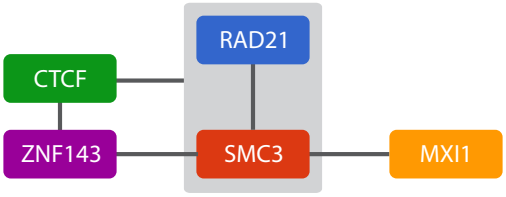


Figure S18: Similar to Figure 4A but with a CTCF experiment and ZNF143 experiment added to the figure. The network edges are from a GroupGM model with an edge threshold of 0.3. Both new experiments tend to associate with the cohesion complex proteins RAD21 and SMC3. The CTCF association is consistent with its combined role with cohesion in mediating chromosomal structure [129].

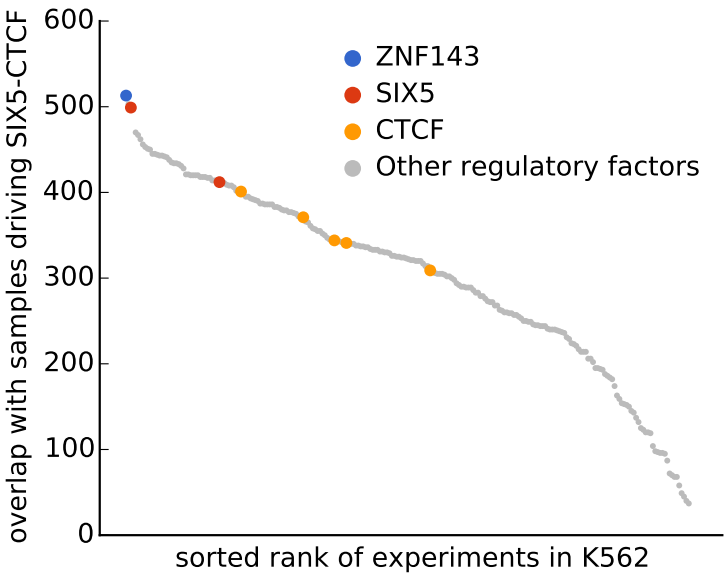


Figure S19: If all ZNF143 datasets from ChromNet are removed and then samples that drive a connection between CTCF and SIX5 in K562 are estimated we find that those samples strongly overlap with positions where ZNF143 is present. The top 1,000 positions driving the edge between SIX5 and CTCF overlap more strongly with the highest 1,000 ZNF143 positions than with any other dataset in K562, including the CTCF and SIX5 datasets the edge actually connects.

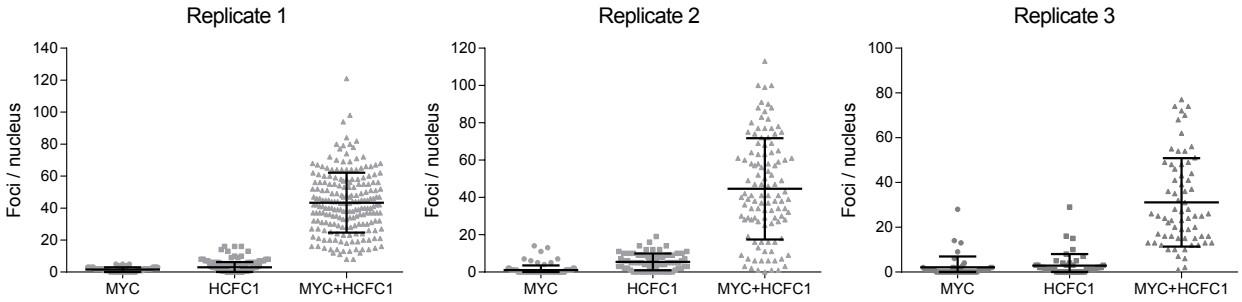


Figure S20: Quantifications for each independent replicate for the MYC–HCFC1 proximity ligation assay. Signal is quantified as the number of foci per nucleus. Individual values (grey dots) and mean ± standard deviation black bars are shown for each replicate.

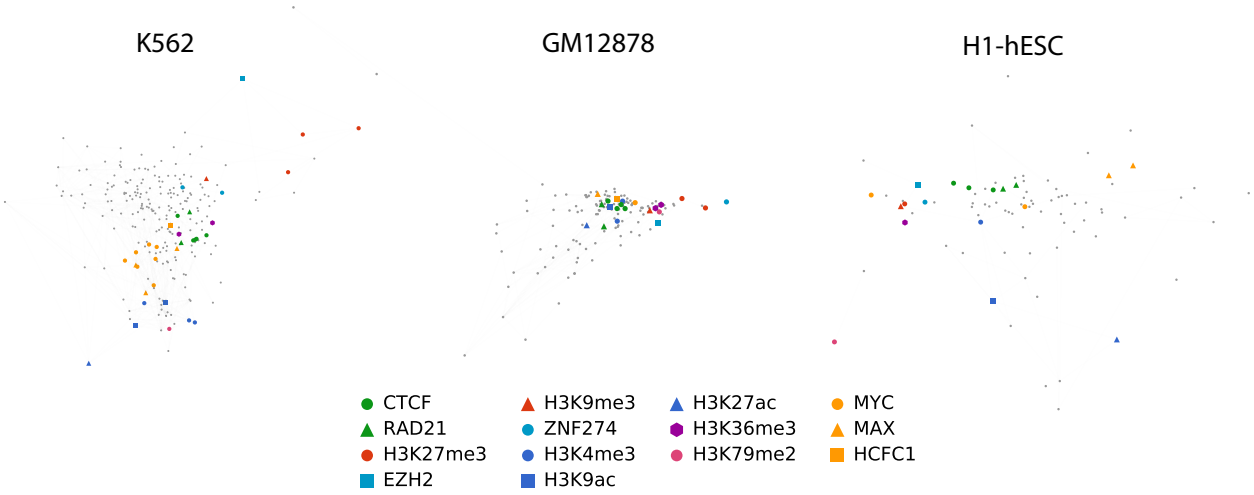


Figure S21: Embeddings of cell-type specific networks using the same approach as in Figure 2.6 (Methods). All three Tier 1 ENCODE cell types are highlighted with the same coloring used in Figure 2.6A.

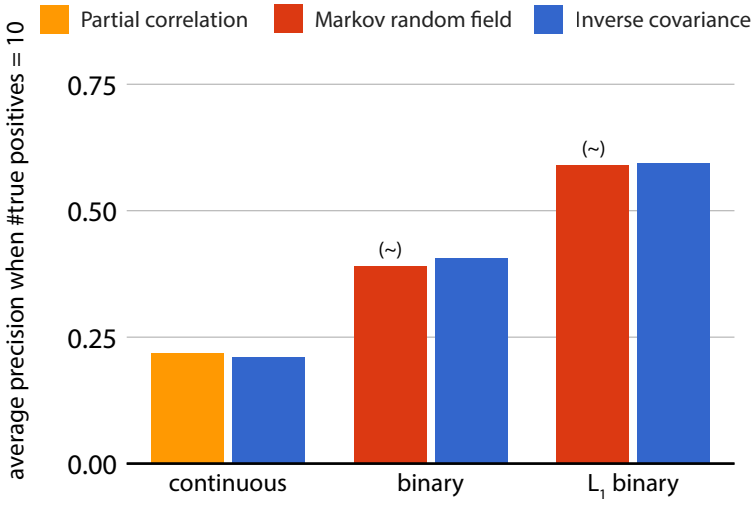


Figure S22: Precision when predicting BioGRID interactions using inverse covariance (blue), a binary Markov random field model from [184] (red), and partial correlation (yellow). A tilde (~) indicates we took Markov random field precision numbers directly from the published precision-recall plot in [184]. To generate inverse covariance and partial correlation results, we started with processed data from [184]. Then, we calculated bootstrap-averaged performance on BioGRID interactions as Zhou et al. did in their article. We compared methods under three different testing regimes. Continuous represents testing on the original control-adjusted, normalized, and binned data. Binary represents testing on binarized data, without regularization. L_1 binary represents testing on binarized data, with L_1 regularization of both models.

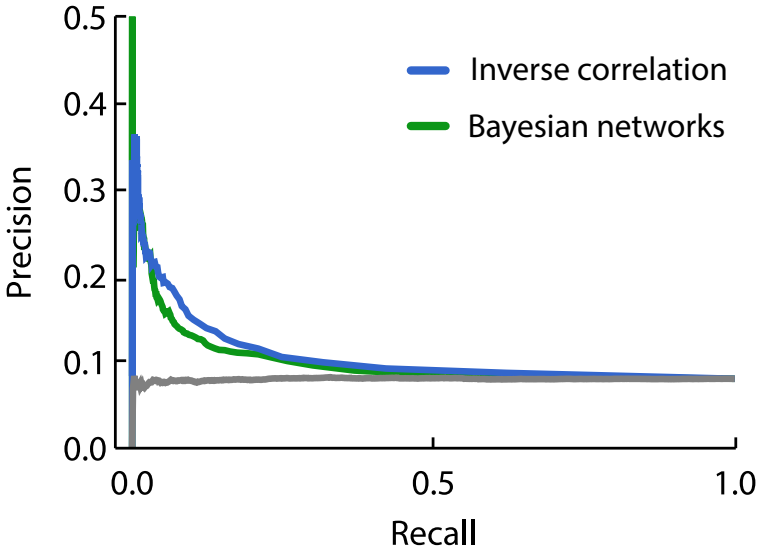


Figure S23: A precision-recall curve for known protein-protein interactions in BioGRID among experiments from the K562 cell type. Bootstrapped Bayesian network inference was performed as in previous work on *D. melanogaster* [162, 10]. We used networks from 400 bootstrap re-samples to estimate 400 Bayesian networks.

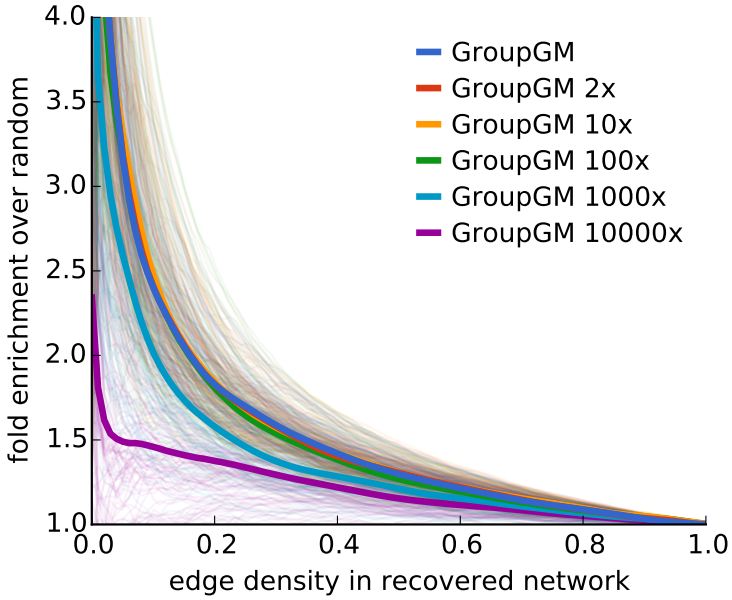


Figure S24: Enrichment of BioGRID supported edges within all cell types as the number of samples used to build the network is varied. Subsampling is done uniformly from the 1,000 bp bins across the genome and has the effect of both reducing the number of samples and also decreasing the correlation between neighboring samples. Up to 100-fold subsampling is possible before noticeable performance degradation.

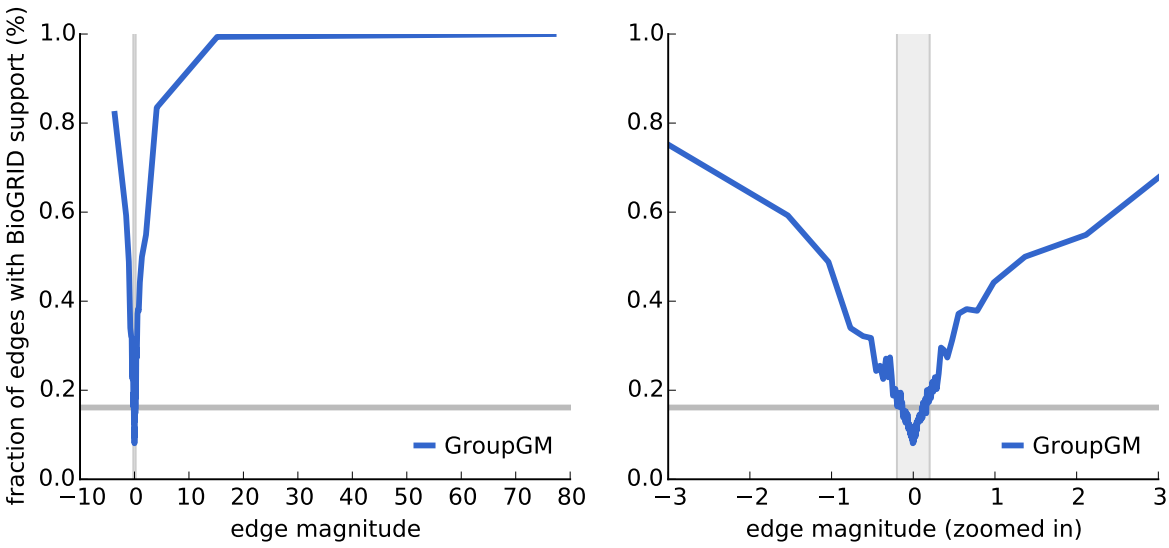


Figure S25: Enrichment of BioGRID support in edges with a given weight. Negative coefficients indicate negative correlatsion. Dark grey line indicates the fraction of BioGRID-supported edges in a randomly connected network (8.4%). Light grey shaded area represents those edges with coefficient magnitude less than the 0.2 minimum used in the ChromNet interface.

2.5.2 Supplementary tables

Table S1: Summary of all ENCODE datasets processed by ChromNet broken down by cell type. This summarizes the full listing of all 1,451 datasets with ENCODE experiment identifiers (Supplementary Data 1). The transcription factor and histone columns represent how many unique transcription factors or histone modifications were measured in that cell type. The treatments column lists the number of additional treatment conditions each cell type was measured under.

Cell type	Datasets	Transcription factors	Histone modifications	Treatments
K562	236	154	12	2
GM12878	143	107	11	1
HepG2	115	81	11	3
A549	94	51	11	2
HeLa-S3	85	62	11	1
H1-hESC	84	60	11	0
MCF-7	55	35	6	1
SK-N-SH	44	27	6	1
endothelial cell of umbilical vein	28	9	12	0
HCT116	28	22	5	0
Ishikawa	25	21	0	6
fibroblast of lung	22	2	11	0
keratinocyte	19	2	12	0
neural cell	17	9	8	0
mammary epithelial cell	16	2	11	0
SUDHL6	14	2	12	0
Karpas-422	14	2	12	0
CD14-positive monocyte	14	1	11	0
skeletal muscle myoblast	13	2	11	0
myotube	13	2	11	0
fibroblast of dermis	13	2	11	0
astrocyte	13	2	11	0
Panc1	13	4	6	0
DND-41	13	2	11	0
osteoblast	12	2	10	0
cardiac mesoderm	12	0	3	0
MCF 10A	12	5	0	1
HEK293	12	7	5	0
DOHH2	12	1	11	0
OCI-LY7	11	1	10	0
OCI-LY3	11	1	10	0
OCI-LY1	11	0	11	0
Loucy	11	1	10	0
IMR-90	10	10	0	0
GM12891	10	9	0	1
T47D	9	6	0	4
NT2/D1	9	3	6	0
GM12892	8	7	0	1
B cell	8	2	5	0
PFSK-1	6	5	0	0
HL-60	6	5	1	0
NB4	5	4	1	0
mononuclear cell	4	0	4	0

Cell type	Datasets	Transcription factors	Histone modifications	Treatments
kidney epithelial cell	4	1	3	0
foreskin fibroblast	4	1	1	0
bronchial epithelial cell	4	1	3	0
U2OS	4	2	2	0
GM06990	4	1	3	0
Caco-2	4	1	3	0
BJ	4	1	3	0
ACC112	4	0	4	0
erythroblast	3	2	0	0
cardiac fibroblast	3	1	1	0
WI38	3	1	1	1
SK-N-MC	3	2	1	0
LNCaP clone FGC	3	1	1	1
H7-hESC	3	0	3	0
retinal pigment epithelial cell	2	1	1	0
fibroblast of villous mesenchyme	2	1	1	0
fibroblast of upper leg skin	2	1	1	0
fibroblast of the aortic adventitia	2	1	1	0
fibroblast of skin of abdomen	2	1	1	0
fibroblast of pulmonary artery	2	1	1	0
fibroblast of pedal digit skin	2	1	1	0
fibroblast of mammary gland	2	1	1	0
fibroblast of gingiva	2	1	1	0
epithelial cell of proximal tubule	2	1	1	0
epithelial cell of esophagus	2	1	1	0
choroid plexus epithelial cell	2	1	1	0
cardiac muscle cell	2	1	1	0
brain microvascular endothelial cell	2	1	1	0
astrocyte of the spinal cord	2	1	1	0
astrocyte of the cerebellum	2	1	1	0
WERI-Rb-1	2	1	1	0
SH-SY5Y	2	2	0	0
HFF-Myc	2	1	1	0
H54	2	2	0	0
GM19193	2	2	0	1
GM19099	2	2	0	1
GM18951	2	2	0	1
GM18526	2	2	0	1
GM18505	2	2	0	1
GM15510	2	2	0	1
GM12875	2	1	1	0
GM12866	2	1	1	0
GM12865	2	1	1	0
GM12864	2	1	1	0
GM10847	2	2	0	1
GM08714	2	1	1	0
BE2C	2	1	1	0
spleen	1	1	0	0
skeletal muscle cell	1	0	1	0
pancreas	1	1	0	0
medulloblastoma	1	1	0	0

Cell type	Datasets	Transcription factors	Histone modifications	Treatments
lung	1	1	0	0
kidney	1	1	0	0
Raji	1	1	0	0
Jurkat	1	0	1	0
GM20000	1	1	0	0
GM19240	1	1	0	0
GM19239	1	1	0	0
GM19238	1	1	0	0
GM13977	1	1	0	0
GM13976	1	1	0	0
GM12874	1	1	0	0
GM12873	1	1	0	0
GM12872	1	1	0	0
GM12871	1	1	0	0
GM12870	1	1	0	0
GM12869	1	1	0	0
GM12868	1	1	0	0
GM12867	1	1	0	0
GM12801	1	1	0	0
GM10266	1	1	0	0
GM10248	1	1	0	0
Total	**1,451**	**812**	**376**	**33**

K562	GM12878	H1-hESC	HepG2	A549	HeLa-S3	SK-N-SH
POLR2A	POLR2A	POLR2A	POLR2A	POLR2A	POLR2A	REST
EP300	MTA3	TAF1	**EP300**	NR3C1	**EP300**	**EP300**
MAX	NFIC	**EP300**	NFIC	SP1	SMARCC1	POLR2A
MTA3	ATF2	GABPA	SP1	**EP300**	TBP	RAD21
WHSC1	YY1	HDAC2	MBD4	MAX	MAX	MXI1
eGFP-JUND	STAT5A	RBBP5	MAX	SIN3A	SREBF2	YY1
HDAC2	SP1	CHD1	FOXA2	FOSL2	CEBPB	RFX5
CBX3	IKZF1	CTBP2	TBP	REST	TAF1	JUND
YY1	RUNX3	ATF2	MYBL2	SIX5	MYC	SMC3
MYC	**EP300**	SP1	ZHX2	USF1	ELK4	CTCF

Table S2: H3K4me1 and H3K27ac combine to mark active enhancers [31]. Here group edges from these two histone marks are computed to all other non-histone regulatory factors and the top ten within each cell type are listed. The well known enhancer associated transcription factor EP300 [152] is found in each cell type and is a validation that we are finding enhancer associated transcription factors (P-value $< 1 \times 10^{-6}$). Using a false discovery threshold of 0.1 we highlighted in red those factors with a significant bias towards a high ranked associated with H3K4me1 and H3K27ac in these cell types.

2.5.3 Supplementary Note 1: Scalability of previous methods

Only correlation and inverse correlation are compared to ChromNet for the full human chromatin network in Figure 2.3. This is because the other previous methods we considered could not scale to the full 1,451 datasets. These are ARACNE (a well-known network learning method for gene expression data) [109], binary Markov random fields [184], and bootstrapped Bayesian networks [162, 10].

ARACNE is designed to handle gene expression which contains a large number of variables, but not necessarily a large number of samples. This was evident when we sought to apply it to chromatin network estimation. ARACNE exhausted all memory on a 24 gigabyte system with only 10 variables and 100,000 samples. This precludes it from even approaching the 3 million samples and 1,451 variables in the ENCODE dataset.

Binary Markov random fields were used successfully to recover regulatory factor interactions in *D. melanogaster*, with 73 variables and 100,000 samples [184]. Using the code kindly provided by Zhou et al., we attempted to apply the Markov random field to the human ENCODE data. Estimating the full joint distribution of a binary Markov random field model is very expensive. One approximation that is much more efficient involves the use of the psuedo-likelihood instead of the joint likelihood. This was one of the methods used by Zhou et al. [184], however even the pseudo-likelihood becomes intractable when we consider all ENCODE datasets, taking over 10 hours with just 60 variables in the model (Figure S5). Furthermore, when we compared inverse correlation to the Binary Markov random field recovery methods on the original *D. melanogaster* data we obtained equivalent performance (Figure S22).

Bootstrapped versions of Bayesian network inference have been used previously to infer networks among regulatory factors in *D. melanogaster* [162, 10]. These experiments were run on binary data among up to 112 factors, but scaling them to human data is much more challenging. Because of run-time constraints we restricted the model to only consider 238 datasets from the K562 cell type. We then used networks from 400 bootstrap re-samples to estimate 400 networks. Each network took about 1.2 hours of processing time to find good solutions, leading to over 500 CPU hours of compute time. Inverse correlation uses a normal approximation for the binary data, runs in less than 10 seconds, and out-performs the far less efficient Bayesian network inference method in terms of known agreement with physical protein-protein interactions labeled in BioGRID (Figure S23).

Note that to allow for a direct comparison with these methods we used binary data rather than raw read counts. The inverse correlation model still outperformed or matched these methods even on their own data types. GroupGM provides and additional level of improvement on top of inverse correlation as demonstrated in Figure 2.3.

Regulatory factor	Max total edge weight	Known in BioGRID
MAX	5.93	+
POLR2A	1.66	+
PHF8	1.26	−
NEUROD1	0.81	−
CREB3L1	0.79	−
CEBPB	0.78	+
HCFC1	0.73	−
ATF7	0.70	−
SUZ12	0.67	−
EP300	0.64	+

Table S3: Top 10 (out of 193) regulatory factors with a strong connection to MYC in ChromNet. Scores are the sum of within cell type group edges connecting MYC experiments to the listed factor. The maximum score is then taken over all ENCODE tier 1 cell types. For comparison we also ran the same experiment using standard correlation instead of group edges and HCFC1 was the 31st strongest interaction with MYC (as opposed to the 7th here).

2.5.4 Supplementary Note 2: Proof that the group graphical model preserves edge magnitudes in the presence of arbitrary collinearity

The inverse covariance matrix (a symmetric matrix) can be interpreted in terms of multiple regression [86, 163], where for simplicity of notation we assume infinite data samples so $\hat{\Sigma} = \Sigma$:

$$\Sigma^{-1} = \Omega = \begin{bmatrix} 1/[\Sigma_{11}(1-R_1^2)] & -\beta_{12}/[\Sigma_{11}(1-R_1^2)] & \cdots & -\beta_{1n}/[\Sigma_{11}(1-R_1^2)] \\ -\beta_{21}/[\Sigma_{22}(1-R_2^2)] & 1/[\Sigma_{22}(1-R_2^2)] & \cdots & -\beta_{2n}/[\Sigma_{22}(1-R_2^2)] \\ \vdots & \vdots & \ddots & \vdots \\ -\beta_{n1}/[\Sigma_{nn}(1-R_n^2)] & -\beta_{n2}/[\Sigma_{nn}(1-R_n^2)] & \cdots & 1/[\Sigma_{nn}(1-R_n^2)] \end{bmatrix}$$

where β_{ij} is a parameter of the ith regression that predicts the ith variable from all the others, and R_i^2 is the proportion of the variance in variable i explained by the ith regression. For correlation matrices the on-diagonal Σ_{ii} entries will be one:

$$\Omega = \begin{bmatrix} 1/(1-R_1^2) & -\beta_{12}/(1-R_1^2) & \cdots & -\beta_{1n}/(1-R_1^2) \\ -\beta_{21}/(1-R_2^2) & 1/(1-R_2^2) & \cdots & -\beta_{2n}/(1-R_2^2) \\ \vdots & \vdots & \ddots & \vdots \\ -\beta_{n1}/(1-R_n^2) & -\beta_{n2}/(1-R_n^2) & \cdots & 1/(1-R_n^2) \end{bmatrix}$$

To further simplify, we can define $S_i = \frac{-1}{1-R_i^2}$:

$$\Omega = \begin{bmatrix} -S_1 & S_1\beta_{12} & \cdots & S_1\beta_{1n} \\ S_2\beta_{21} & -S_2 & \cdots & S_2\beta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ S_n\beta_{n1} & S_n\beta_{n2} & \cdots & -S_n \end{bmatrix}$$

Consider an arbitrary edge between two nodes A and B with that correspond to rows A_1 and B_1 in Ω. The strength of the connection in the symmetric matrix Ω is $S_{A_1}\beta_{A_1B_1} = S_{B_1}\beta_{B_1A_1}$.

Now consider a new data set with a superset of the variables in the original network represented by Ω. This new dataset, represented by $\Omega^{(2)}$, has a second B variable with index B_2. These two B variables (B_1 and B_2) are arbitrarily similar to one another but not identical, and the second variable bears no relationship to other variables in the network beyond what it gains by being similar to B_1. The regression problem for A_1 would be unstable, because B_1 and B_2 are highly correlated to each other, which makes it unclear how the weights should be distributed to these two predictor variables. However, the sum of the coefficients for the B

group remains the same:

$$\beta^{(2)}_{A_1B_1} + \beta^{(2)}_{A_1B_2} = \beta_{A_1B_1},$$

In addition, no new information has been provided about A, so S_A remains unchanged (because the amount of variance explained remains the same):

$$S^{(2)}_A = S_A,$$

which means the following:

$$S^{(2)}_A \beta^{(2)}_{A_1B_1} + S^{(2)}_A \beta^{(2)}_{A_1B_2} = S_A \beta_{A_1B_1},$$

which is equivalent to:

$$\Omega^{(2)}_{A_1B_1} + \Omega^{(2)}_{A_1B_2} = \Omega_{A_1B_1}.$$

This means that the connection strength that was present in between A and B in Ω is now preserved as a sum of two entries in $\Omega^{(2)}$. This argument generalizes to any number of variables in the B group.

Now after adding a redundant B variable consider adding a redundant A variable to create a new data set $\Omega^{(3)}$. Since the B variables cannot choose between A_1 and A_2 their coefficients are unstable but still sum to their previous value:

$$\beta^{(3)}_{B_1A_1} + \beta^{(3)}_{B_1A_2} = \beta^{(2)}_{B_1A_1} \tag{2.14}$$
$$\beta^{(3)}_{B_2A_1} + \beta^{(3)}_{B_2A_2} = \beta^{(2)}_{B_2A_1} \tag{2.15}$$

adding A_2 provided no new explanatory power for the B variables so

$$S^{(3)}_{B_1} = S^{(2)}_{B_1} \tag{2.16}$$
$$S^{(3)}_{B_2} = S^{(2)}_{B_2}, \tag{2.17}$$

which means

$$S^{(3)}_{B_1} \beta^{(3)}_{B_1A_1} + S^{(3)}_{B_1} \beta^{(3)}_{B_1A_2} = S^{(2)}_{B_1} \beta^{(2)}_{B_1A_1} \tag{2.18}$$
$$S^{(3)}_{B_2} \beta^{(3)}_{B_2A_1} + S^{(3)}_{B_2} \beta^{(3)}_{B_2A_2} = S^{(2)}_{B_2} \beta^{(2)}_{B_2A_1}, \tag{2.19}$$

and

$$\Omega^{(3)}_{B_1A_1} + \Omega^{(3)}_{B_1A_2} = \Omega^{(2)}_{B_1A_1} \tag{2.20}$$
$$\Omega^{(3)}_{B_2A_1} + \Omega^{(3)}_{B_2A_2} = \Omega^{(2)}_{B_2A_1}. \tag{2.21}$$

Because Ω is symmetric we know that

$$\Omega^{(2)}_{A_1B_1} + \Omega^{(2)}_{A_1B_2} = \Omega^{(2)}_{B_1A_1} + \Omega^{(2)}_{B_2A_1}.$$

Using this we can now calculate the original connection strength $\Omega_{A_1B_1}$ as a sum of entries in $\Omega^{(3)}$. This can be directly generalized to any number of variables in each group, which means that the connection strength of an edge between two variables in a non-redundant data set can be recovered by summing edges in a data set where both variables are in groups of redundant variables.

$$\Omega_{A_1B_1} = \Omega^{(2)}_{A_1B_1} + \Omega^{(2)}_{A_1B_2} \tag{2.22}$$

$$\Omega_{A_1B_1} = \Omega^{(2)}_{B_1A_1} + \Omega^{(2)}_{B_2A_1} \tag{2.23}$$

$$\Omega_{A_1B_1} = \Omega^{(3)}_{B_1A_1} + \Omega^{(3)}_{B_1A_2} + \Omega^{(3)}_{B_2A_1} + \Omega^{(3)}_{B_2A_2} \tag{2.24}$$

$$\tag{2.25}$$

2.5.5 Supplementary Note 3: Estimation of conditional dependence from binary data

For the binary data tracks compared in Figure S1 we matched datasets with controls using metadata from the ENCODE web site [42], then ran MACS2 [183] without peak shift adjustments and with a P-value peak threshold of 0.05. Peak data from MACS2 was then binned into 1,000 bp windows by labeling a window 1 if any peak overlapped the window and 0 otherwise.

Here we briefly discuss why we considered binary data in the context of inverse correlation. Given datasets drawn from a set $\mathcal{X}_b$ of binary random variables, we can represent a joint pairwise model of these datasets without loss of generality as a pairwise Markov random field:

$$P(x) = \frac{1}{Z} \exp\left(- \sum_{X_i \in \mathcal{X}_b, X_j \in \mathcal{X}_b} \Phi_{i,j} X_i X_j \right) \tag{2.26}$$

where Φ is a matrix of pairwise interaction terms and Z is a normalizing constant. Previous work on estimating a smaller subset of the chromatin network from binary data used Markov random fields and higher-order extensions [184, 117]. These works employ iterative or approximate methods, as exact inference on their models with many variables is computationally intractable. For certain graph classes, however, the sparsity structure of Φ and of the inverse correlation matrix Σ^{-1} are equivalent [97].

To compare Σ^{-1} with one of these methods we compared with estimates of conditional dependence from a pairwise Markov random field of binary data [184] (Supplementary Note 2). The Markov random field implementation was based on unnormalized binary data, so for comparison we used the inverse covariance matrix which results from an unnormalized dataset (meaning the data was not mean centered or scaled to unit variance). We used the original processed data kindly provided by the authors of [184] (J. Zhou, personal communication). These data are from 73 modENCODE ChIP-chip datasets on *Drosophila melanogaster* S2-DRSC cells. We calculated precision for the inverse covariance of binary data using the same bootstrap procedure as the authors, and compared against Markov random field precision numbers from their published precision-recall plot [184].

On this smaller data set, the enrichment of known protein-protein interactions in $\hat{\Phi}$ and the inverse covariance matrix were similar (Figure S22). This near-equivalence between the methods supports the use of an inverse covariance (or correlation) matrix to estimate edge strength in a pairwise Markov random field. It is infeasible to estimate a Markov random field among all ENCODE datasets (Supplementary Note 1). Using a matrix inverse dramatically increases computational efficiency, while maintaining results similar to a full binary pairwise Markov random field.

Chapter 3

A Unified Approach to Interpreting Model Predictions

Understanding why a model makes a certain prediction can be as crucial as the prediction's accuracy in many applications. However, the highest accuracy for large modern datasets is often achieved by complex models that even experts struggle to interpret, such as ensemble or deep learning models, creating a tension between *accuracy* and *interpretability*. In response, various methods have recently been proposed to help users interpret the predictions of complex models, but it is often unclear how these methods are related and when one method is preferable over another. To address this problem, we present a unified framework for interpreting predictions, SHAP (SHapley Additive exPlanations). SHAP assigns each feature an importance value for a particular prediction. Its novel components include: (1) the identification of a new class of additive feature importance measures, and (2) theoretical results showing there is a unique solution in this class with a set of desirable properties. The new class unifies six existing methods, notable because several recent methods in the class lack the proposed desirable properties. Based on insights from this unification, we present new methods that show improved computational performance and/or better consistency with human intuition than previous approaches.

3.1 Introduction

The ability to correctly interpret a prediction model's output is extremely important. It engenders appropriate user trust, provides insight into how a model may be improved, and supports understanding of the process being modeled. In some applications, simple models (e.g., linear models) are often preferred for their ease of interpretation, even if they may be less accurate than complex ones. However, the growing availability of big data has increased the benefits of using complex models, so bringing to the forefront the trade-off between accuracy and interpretability of a model's output. A wide variety of different methods have been recently proposed to address this issue [141, 155, 167, 32, 95, 6]. But an understanding of how these methods relate and when one method is preferable to another is still lacking.

Here, we present a novel unified approach to interpreting model predictions.[1] Our approach leads to three potentially surprising results that bring clarity to the growing space of methods:

1. We introduce the perspective of viewing *any* explanation of a model's prediction as a model itself, which we term the *explanation model*. This lets us define the class of *additive feature attribution methods* (Section 3.2), which unifies six current methods.

2. We then show that game theory results guaranteeing a unique solution apply to the *entire class* of additive feature attribution methods (Section 3.3) and propose *SHAP values* as a unified measure of feature importance that various methods approximate (Section 3.4).

[1] `https://github.com/slundberg/shap`

3. We propose new SHAP value estimation methods and demonstrate that they are better aligned with human intuition as measured by user studies and more effectually discriminate among model output classes than several existing methods (Section 3.5).

3.2 Additive Feature Attribution Methods

The best explanation of a simple model is the model itself; it perfectly represents itself and is easy to understand. For complex models, such as ensemble methods or deep networks, we cannot use the original model as its own best explanation because it is not easy to understand. Instead, we must use a simpler *explanation model*, which we define as any interpretable approximation of the original model. We show below that six current explanation methods from the literature all use the same explanation model. This previously unappreciated unity has interesting implications, which we describe in later sections.

Let f be the original prediction model to be explained and g the explanation model. Here, we focus on *local methods* designed to explain a prediction $f(x)$ based on a single input x, as proposed in LIME [141]. Explanation models often use *simplified inputs* x' that map to the original inputs through a mapping function $x = h_x(x')$. Local methods try to ensure $g(z') \approx f(h_x(z'))$ whenever $z' \approx x'$. (Note that $h_x(x') = x$ even though x' may contain less information than x because h_x is specific to the current input x.)

Definition 1. Additive feature attribution methods *have an explanation model that is a linear function of binary variables:*

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i, \tag{3.1}$$

where $z' \in \{0,1\}^M$*,* M *is the number of simplified input features, and* $\phi_i \in \mathbb{R}$*.*

Methods with explanation models matching Definition 1 attribute an effect ϕ_i to each feature, and summing the effects of all feature attributions approximates the output $f(x)$ of the original model. Many current methods match Definition 1, several of which are discussed below.

3.2.1 LIME

The *LIME* method interprets individual model predictions based on locally approximating the model around a given prediction [141]. The local linear explanation model that LIME uses adheres to Equation 3.1 exactly and is thus an additive feature attribution method. LIME refers to simplified inputs x' as "interpretable inputs," and the mapping $x = h_x(x')$ converts a binary vector of interpretable inputs into the original input space. Different types of h_x mappings are used for different input spaces. For bag of words text features, h_x converts a vector of 1's or 0's (present or not) into the original word count if the simplified input is one, or zero if the simplified input is zero. For images, h_x treats the image as a set of super pixels; it then maps 1 to leaving the super pixel as its original value and 0 to replacing the super pixel with an average of neighboring pixels (this is meant to represent being missing).

To find ϕ, LIME minimizes the following objective function:

$$\xi = \underset{g \in \mathcal{G}}{\arg\min}\ L(f, g, \pi_{x'}) + \Omega(g). \tag{3.2}$$

Faithfulness of the explanation model $g(z')$ to the original model $f(h_x(z'))$ is enforced through the loss L over a set of samples in the simplified input space weighted by the local kernel $\pi_{x'}$. Ω penalizes the complexity of g. Since in LIME g follows Equation 3.1 and L is a squared loss, Equation 3.2 can be solved using penalized linear regression.

3.2.2 DeepLIFT

DeepLIFT was recently proposed as a recursive prediction explanation method for deep learning [155, 154]. It attributes to each input x_i a value $C_{\Delta x_i \Delta y}$ that represents the effect of that input being set to a reference value as opposed to its original value. This means that for DeepLIFT, the mapping $x = h_x(x')$ converts binary

values into the original inputs, where 1 indicates that an input takes its original value, and 0 indicates that it takes the reference value. The reference value, though chosen by the user, represents a typical uninformative background value for the feature.

DeepLIFT uses a "summation-to-delta" property that states:

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta o} = \Delta o, \tag{3.3}$$

where $o = f(x)$ is the model output, $\Delta o = f(x) - f(r)$, $\Delta x_i = x_i - r_i$, and r is the reference input. If we let $\phi_i = C_{\Delta x_i \Delta o}$ and $\phi_0 = f(r)$, then DeepLIFT's explanation model matches Equation 3.1 and is thus another additive feature attribution method.

3.2.3 Layer-Wise Relevance Propagation

The *layer-wise relevance propagation* method interprets the predictions of deep networks [6]. As noted by Shrikumar et al., this menthod is equivalent to DeepLIFT with the reference activations of all neurons fixed to zero. Thus, $x = h_x(x')$ converts binary values into the original input space, where 1 means that an input takes its original value, and 0 means an input takes the 0 value. Layer-wise relevance propagation's explanation model, like DeepLIFT's, matches Equation 3.1.

3.2.4 Classic Shapley Value Estimation

Three previous methods use classic equations from cooperative game theory to compute explanations of model predictions: Shapley regression values [95], Shapley sampling values [167], and Quantitative Input Influence [32].

Shapley regression values are feature importances for linear models in the presence of multicollinearity. This method requires retraining the model on all feature subsets $S \subseteq F$, where F is the set of all features. It assigns an importance value to each feature that represents the effect on the model prediction of including that feature. To compute this effect, a model $f_{S\cup\{i\}}$ is trained with that feature present, and another model f_S is trained with the feature withheld. Then, predictions from the two models are compared on the current input $f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)$, where x_S represents the values of the input features in the set S. Since the effect of withholding a feature depends on other features in the model, the preceding differences are computed for all possible subsets $S \subseteq F \setminus \{i\}$. The Shapley values are then computed and used as feature attributions. They are a weighted average of all possible differences:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S) \right]. \tag{3.4}$$

For Shapley regression values, h_x maps 1 or 0 to the original input space, where 1 indicates the input is included in the model, and 0 indicates exclusion from the model. If we let $\phi_0 = f_\varnothing(\varnothing)$, then the Shapley regression values match Equation 3.1 and are hence an additive feature attribution method.

Shapley sampling values are meant to explain any model by: (1) applying sampling approximations to Equation 3.4, and (2) approximating the effect of removing a variable from the model by integrating over samples from the training dataset. This eliminates the need to retrain the model and allows fewer than $2^{|F|}$ differences to be computed. Since the explanation model form of Shapley sampling values is the same as that for Shapley regression values, it is also an additive feature attribution method.

Quantitative input influence is a broader framework that addresses more than feature attributions. However, as part of its method it independently proposes a sampling approximation to Shapley values that is nearly identical to Shapley sampling values. It is thus another additive feature attribution method.

3.3 Simple Properties Uniquely Determine Additive Feature Attributions

A surprising attribute of the class of additive feature attribution methods is the presence of a single unique solution in this class with three desirable properties (described below). While these properties are familiar to the classical Shapley value estimation methods, they were previously unknown for other additive feature attribution methods.

The first desirable property is *local accuracy*. When approximating the original model f for a specific input x, local accuracy requires the explanation model to at least match the output of f for the simplified input x' (which corresponds to the original input x).

Property 1 (Local accuracy)**.**

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{3.5}$$

The explanation model $g(x')$ matches the original model $f(x)$ when $x = h_x(x')$, where $\phi_0 = f(h_x(\mathbf{0}))$ represents the model output with all simplified inputs toggled off (i.e. missing).

The second property is *missingness*. If the simplified inputs represent feature presence, then missingness requires features missing in the original input to have no impact. All of the methods described in Section 3.2 obey the missingness property.

Property 2 (Missingness)**.**

$$x'_i = 0 \implies \phi_i = 0 \tag{3.6}$$

Missingness constrains features where $x'_i = 0$ to have no attributed impact.

The third property is *consistency*. Consistency states that if a model changes so that some simplified input's contribution increases or stays the same regardless of the other inputs, that input's attribution should not decrease.

Property 3 (Consistency)**.** *Let $f_x(z') = f(h_x(z'))$ and $z' \setminus i$ denote setting $z'_i = 0$. For any two models f and f', if*

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i) \tag{3.7}$$

for all inputs $z' \in \{0,1\}^M$, then $\phi_i(f', x) \geq \phi_i(f, x)$.

Theorem 1. *Only one possible explanation model g follows Definition 1 and satisfies Properties 1, 2, and 3:*

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} \left[f_x(z') - f_x(z' \setminus i) \right] \tag{3.8}$$

where $|z'|$ is the number of non-zero entries in z', and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

Theorem 1 follows from combined cooperative game theory results, where the values ϕ_i are known as Shapley values [151]. Young (1985) demonstrated that Shapley values are the only set of values that satisfy three axioms similar to Property 1, Property 3, and a final property that we show to be redundant in this setting (see Supplementary Material). Property 2 is required to adapt the Shapley proofs to the class of additive feature attribution methods.

Under Properties 1-3, for a given simplified input mapping h_x, Theorem 1 shows that there is only one possible additive feature attribution method. This result implies that methods not based on Shapley values violate local accuracy and/or consistency (methods in Section 3.2 already respect missingness). The following section proposes a unified approach that improves previous methods, preventing them from unintentionally violating Properties 1 and 3.

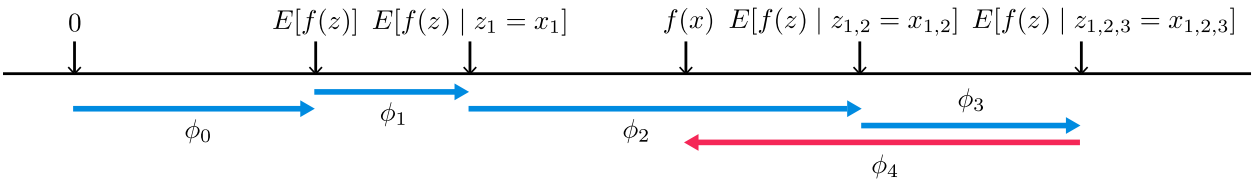


Figure S1: SHAP (SHapley Additive exPlanation) values attribute to each feature the change in the expected model prediction when conditioning on that feature. They explain how to get from the base value $E[f(z)]$ that would be predicted if we did not know any features to the current output $f(x)$. This diagram shows a single ordering. When the model is non-linear or the input features are not independent, however, the order in which features are added to the expectation matters, and the SHAP values arise from averaging the ϕ_i values across all possible orderings.

3.4 SHAP (SHapley Additive exPlanation) Values

We propose SHAP values as a unified measure of feature importance. These are the Shapley values of a conditional expectation function of the original model; thus, they are the solution to Equation 3.8, where $f_x(z') = f(h_x(z')) = E[f(z) \mid z_S]$, and S is the set of non-zero indexes in z' (Figure S1). Based on Sections 2 and 3, SHAP values provide the unique additive feature importance measure that adheres to Properties 1-3 and uses conditional expectations to define simplified inputs. Implicit in this definition of SHAP values is a simplified input mapping, $h_x(z') = z_S$, where z_S has missing values for features not in the set S. Since most models cannot handle arbitrary patterns of missing input values, we approximate $f(z_S)$ with $E[f(z) \mid z_S]$. This definition of SHAP values is designed to closely align with the Shapley regression, Shapley sampling, and quantitative input influence feature attributions, while also allowing for connections with LIME, DeepLIFT, and layer-wise relevance propagation.

The exact computation of SHAP values is challenging. However, by combining insights from current additive feature attribution methods, we can approximate them. We describe two model-agnostic approximation methods, one that is already known (Shapley sampling values) and another that is novel (Kernel SHAP). We also describe four model-type-specific approximation methods, two of which are novel (Max SHAP, Deep SHAP). When using these methods, feature independence and model linearity are two optional assumptions simplifying the computation of the expected values (note that $\bar{S}$ is the set of features not in S):

$$
\begin{aligned}
f(h_x(z')) &= E[f(z) \mid z_S] && \text{SHAP explanation model simplified input mapping} && (3.9)\\
&= E_{z_{\bar{S}} \mid z_S}[f(z)] && \text{expectation over } z_{\bar{S}} \mid z_S && (3.10)\\
&\approx E_{z_{\bar{S}}}[f(z)] && \text{assume feature independence (as in [167, 141, 154, 32])} && (3.11)\\
&\approx f([z_S, E[z_{\bar{S}}]]). && \text{assume model linearity} && (3.12)
\end{aligned}
$$

3.4.1 Model-Agnostic Approximations

If we assume feature independence when approximating conditional expectations (Equation 3.11), as in [167, 141, 154, 32], then SHAP values can be estimated directly using the Shapley sampling values method [167] or equivalently the Quantitative Input Influence method [32]. These methods use a sampling approximation of a permutation version of the classic Shapley value equations (Equation 3.8). Separate sampling estimates are performed for each feature attribution. While reasonable to compute for a small number of inputs, the Kernel SHAP method described next requires fewer evaluations of the original model to obtain similar approximation accuracy (Section 3.5).

Kernel SHAP (Linear LIME + Shapley values)

Linear LIME uses a linear explanation model to locally approximate f, where local is measured in the simplified binary input space. At first glance, the regression formulation of LIME in Equation 3.2 seems very different from the classical Shapley value formulation of Equation 3.8. However, since linear LIME is an

additive feature attribution method, we know the Shapley values are the only possible solution to Equation 3.2 that satisfies Properties 1-3 – local accuracy, missingness and consistency. A natural question to pose is whether the solution to Equation 3.2 recovers these values. The answer depends on the choice of loss function L, weighting kernel $\pi_{x'}$ and regularization term Ω. The LIME choices for these parameters are made heuristically; using these choices, Equation 3.2 does not recover the Shapley values. One consequence is that local accuracy and/or consistency are violated, which in turn leads to unintuitive behavior in certain circumstances (see Section 3.5).

Below we show how to avoid heuristically choosing the parameters in Equation 3.2 and how to find the loss function L, weighting kernel $\pi_{x'}$, and regularization term Ω that recover the Shapley values.

Theorem 2 (Shapley kernel)**.** *Under Definition 1, the specific forms of $\pi_{x'}$, L, and Ω that make solutions of Equation 3.2 consistent with Properties 1 through 3 are:*

$$\begin{aligned}\Omega(g) &= 0,\\ \pi_{x'}(z') &= \frac{(M-1)}{(M \text{ choose } |z'|)|z'|(M-|z'|)},\\ L(f,g,\pi_{x'}) &= \sum_{z'\in Z}\left[f(h_x(z')) - g(z')\right]^2 \pi_{x'}(z'),\end{aligned}$$

where $|z'|$ is the number of non-zero elements in z'.

The proof of Theorem 2 is shown in the Supplementary Material.

It is important to note that $\pi_{x'}(z') = \infty$ when $|z'| \in \{0, M\}$, which enforces $\phi_0 = f_x(\varnothing)$ and $f(x) = \sum_{i=0}^{M}\phi_i$. In practice, these infinite weights can be avoided during optimization by analytically eliminating two variables using these constraints.

Since $g(z')$ in Theorem 2 is assumed to follow a linear form, and L is a squared loss, Equation 3.2 can still be solved using linear regression. As a consequence, the Shapley values from game theory can be computed using weighted linear regression.[2] Since LIME uses a simplified input mapping that is equivalent to the approximation of the SHAP mapping given in Equation 3.12, this enables regression-based, model-agnostic estimation of SHAP values. Jointly estimating all SHAP values using regression provides better sample efficiency than the direct use of classical Shapley equations (see Section 3.5).

The intuitive connection between linear regression and Shapley values is that Equation 3.8 is a difference of means. Since the mean is also the best least squares point estimate for a set of data points, it is natural to search for a weighting kernel that causes linear least squares regression to recapitulate the Shapley values. This leads to a kernel that distinctly differs from previous heuristically chosen kernels (Figure S2A).

3.4.2 Model-Specific Approximations

While Kernel SHAP improves the sample efficiency of model-agnostic estimations of SHAP values, by restricting our attention to specific model types, we can develop faster model-specific approximation methods.

Linear SHAP

For linear models, if we assume input feature independence (Equation 3.11), SHAP values can be approximated directly from the model's weight coefficients.

Corollary 1 (Linear SHAP)**.** *Given a linear model $f(x) = \sum_{j=1}^{M} w_j x_j + b$: $\phi_0(f,x) = b$ and*

$$\phi_i(f,x) = w_j(x_j - E[x_j])$$

This follows from Theorem 2 and Equation 3.11, and it has been previously noted by Štrumbelj and Kononenko [167].

[2]During the preparation of this manuscript we discovered this parallels an equivalent constrained quadratic minimization formulation of Shapley values proposed in econometrics [22].

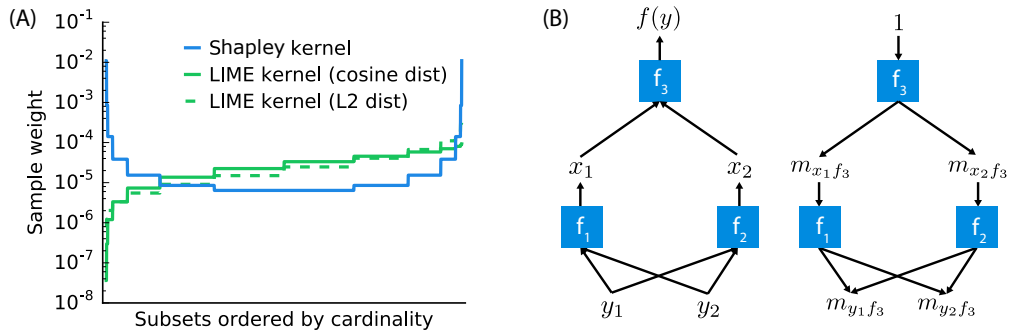


Figure S2: (A) The Shapley kernel weighting is symmetric when all possible z' vectors are ordered by cardinality there are 2^{15} vectors in this example. This is distinctly different from previous heuristically chosen kernels. (B) Compositional models such as deep neural networks are comprised of many simple components. Given analytic solutions for the Shapley values of the components, fast approximations for the full model can be made using DeepLIFT's style of back-propagation.

Low-Order SHAP

Since linear regression using Theorem 2 has complexity $O(2^M + M^3)$, it is efficient for small values of M if we choose an approximation of the conditional expectations (Equation 3.11 or 3.12).

Max SHAP

Using a permutation formulation of Shapley values, we can calculate the probability that each input will increase the maximum value over every other input. Doing this on a sorted order of input values lets us compute the Shapley values of a max function with M inputs in $O(M^2)$ time instead of $O(M2^M)$. See Supplementary Material for the full algorithm.

Deep SHAP (DeepLIFT + Shapley values)

While Kernel SHAP can be used on any model, including deep models, it is natural to ask whether there is a way to leverage extra knowledge about the compositional nature of deep networks to improve computational performance. We find an answer to this question through a previously unappreciated connection between Shapley values and DeepLIFT [155]. If we interpret the reference value in Equation 3.3 as representing $E[x]$ in Equation 3.12, then DeepLIFT approximates SHAP values assuming that the input features are independent of one another and the deep model is linear. DeepLIFT uses a linear composition rule, which is equivalent to linearizing the non-linear components of a neural network. Its back-propagation rules defining how each component is linearized are intuitive but were heuristically chosen. Since DeepLIFT is an additive feature attribution method that satisfies local accuracy and missingness, we know that Shapley values represent the only attribution values that satisfy consistency. This motivates our adapting DeepLIFT to become a compositional approximation of SHAP values, leading to Deep SHAP.

Deep SHAP combines SHAP values computed for smaller components of the network into SHAP values for the whole network. It does so by recursively passing DeepLIFT's multipliers, now defined in terms of

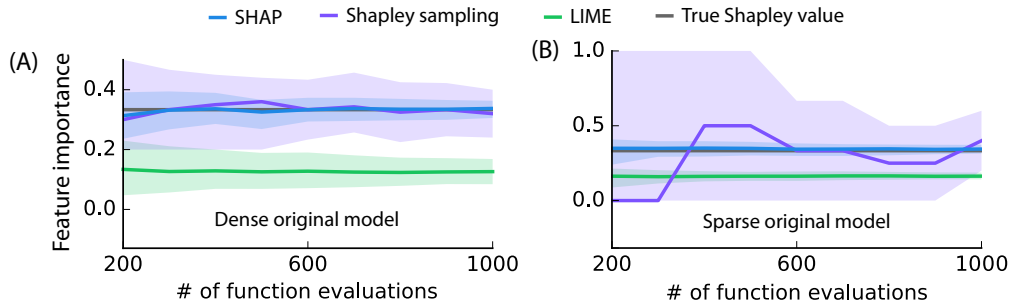


Figure S3: Comparison of three additive feature attribution methods: Kernel SHAP (using a debiased lasso), Shapley sampling values, and LIME (using the open source implementation). Feature importance estimates are shown for one feature in two models as the number of evaluations of the original model function increases. The 10th and 90th percentiles are shown for 200 replicate estimates at each sample size. (A) A decision tree model using all 10 input features is explained for a single input. (B) A decision tree using only 3 of 100 input features is explained for a single input.

SHAP values, backwards through the network as in Figure S2B:

$$m_{x_j f_3} = \frac{\phi_i(f_3, x)}{x_j - E[x_j]} \tag{3.13}$$

$$\forall_{j \in \{1,2\}} \quad m_{y_i f_j} = \frac{\phi_i(f_j, y)}{y_i - E[y_i]} \tag{3.14}$$

$$m_{y_i f_3} = \sum_{j=1}^{2} m_{y_i f_j} m_{x_j f_3} \qquad \text{chain rule} \tag{3.15}$$

$$\phi_i(f_3, y) \approx m_{y_i f_3}(y_i - E[y_i]) \qquad \text{linear approximation} \tag{3.16}$$

Since the SHAP values for the simple network components can be efficiently solved analytically if they are linear, max pooling, or an activation function with just one input, this composition rule enables a fast approximation of values for the whole model. Deep SHAP avoids the need to heuristically choose ways to linearize components. Instead, it derives an effective linearization from the SHAP values computed for each component. The *max* function offers one example where this leads to improved attributions (see Section 3.5).

3.5 Computational and User Study Experiments

We evaluated the benefits of SHAP values using the Kernel SHAP and Deep SHAP approximation methods. First, we compared the computational efficiency and accuracy of Kernel SHAP vs. LIME and Shapley sampling values. Second, we designed user studies to compare SHAP values with alternative feature importance allocations represented by DeepLIFT and LIME. As might be expected, SHAP values prove more consistent with human intuition than other methods that fail to meet Properties 1-3 (Section 3.2). Finally, we use MNIST digit image classification to compare SHAP with DeepLIFT and LIME.

3.5.1 Computational Efficiency

Theorem 2 connects Shapley values from game theory with weighted linear regression. Kernal SHAP uses this connection to compute feature importance. This leads to more accurate estimates with fewer evaluations of the original model than previous sampling-based estimates of Equation 3.8, particularly when regularization is added to the linear model (Figure S3). Comparing Shapley sampling, SHAP, and LIME on both dense and sparse decision tree models illustrates both the improved sample efficiency of Kernel SHAP and that values from LIME can differ significantly from SHAP values that satisfy local accuracy and consistency.

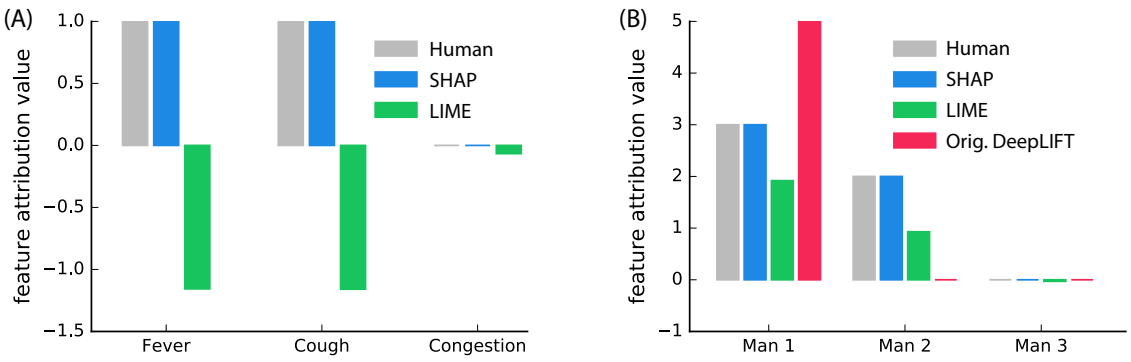


Figure S4: Human feature impact estimates are shown as the most common explanation given among 30 (A) and 52 (B) random individuals, respectively. (A) Feature attributions for a model output value (sickness score) of 2. The model output is 2 when fever and cough are both present, 5 when only one of fever or cough is present, and 0 otherwise. (B) Attributions of profit among three men, given according to the maximum number of questions any man got right. The first man got 5 questions right, the second 4 questions, and the third got none right, so the profit is $5.

3.5.2 Consistency with Human Intuition

Theorem 1 provides a strong incentive for all additive feature attribution methods to use SHAP values. Both LIME and DeepLIFT, as originally demonstrated, compute different feature importance values. To validate the importance of Theorem 1, we compared explanations from LIME, DeepLIFT, and SHAP with user explanations of simple models (using Amazon Mechanical Turk). Our testing assumes that good model explanations should be consistent with explanations from humans who understand that model.

We compared LIME, DeepLIFT, and SHAP with human explanations for two settings. The first setting used a sickness score that was higher when only one of two symptoms was present (Figure S4A). The second used a max allocation problem to which DeepLIFT can be applied. Participants were told a short story about how three men made money based on the maximum score any of them achieved (Figure S4B). In both cases, participants were asked to assign credit for the output (the sickness score or money won) among the inputs (i.e., symptoms or players). We found a much stronger agreement between human explanations and SHAP than with other methods. SHAP's improved performance for max functions addresses the open problem of max pooling functions in DeepLIFT [154].

3.5.3 Explaining Class Differences

As discussed in Section 3.4.2, DeepLIFT's compositional approach suggests a compositional approximation of SHAP values (Deep SHAP). These insights, in turn, improve DeepLIFT, and a new version includes updates to better match Shapley values [154]. Figure S5 extends DeepLIFT's convolutional network example to highlight the increased performance of estimates that are closer to SHAP values. The pre-trained model and Figure S5 example are the same as those used in [154], with inputs normalized between 0 and 1. Two convolution layers and 2 dense layers are followed by a 10-way softmax output layer. Both DeepLIFT versions explain a normalized version of the linear layer, while SHAP (computed using Kernel SHAP) and LIME explain the model's output. SHAP and LIME were both run with 50k samples (Supplementary Figure 1); to improve performance, LIME was modified to use single pixel segmentation over the digit pixels. To match [154], we masked 20% of the pixels chosen to switch the predicted class from 8 to 3 according to the feature attribution given by each method.

3.6 Conclusion

The growing tension between the accuracy and interpretability of model predictions has motivated the development of methods that help users interpret predictions. The SHAP framework identifies the class of

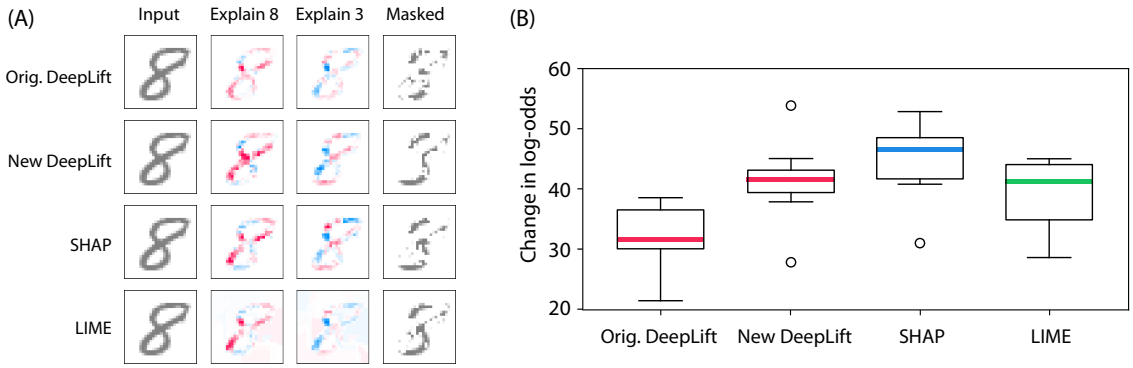


Figure S5: Explaining the output of a convolutional network trained on the MNIST digit dataset. Orig. DeepLIFT has no explicit Shapley approximations, while New DeepLIFT seeks to better approximate Shapley values. (A) Red areas increase the probability of that class, and blue areas decrease the probability. Masked removes pixels in order to go from 8 to 3. (B) The change in log odds when masking over 20 random images supports the use of better estimates of SHAP values.

additive feature importance methods (which includes six previous methods) and shows there is a unique solution in this class that adheres to desirable properties. The thread of unity that SHAP weaves through the literature is an encouraging sign that common principles about model interpretation can inform the development of future methods.

We presented several different estimation methods for SHAP values, along with proofs and experiments showing that these values are desirable. Promising next steps involve developing faster model-type-specific estimation methods that make fewer assumptions, integrating work on estimating interaction effects from game theory, and defining new explanation model classes.

Acknowledgements

The results presented in this chapter have been released as a paper in collaboration with my advisor Su-In Lee [101].

This work was supported by a National Science Foundation (NSF) DBI-135589, NSF CAREER DBI-155230, American Cancer Society 127332-RSG-15-097-01-TBG, National Institute of Health (NIH) AG049196, and NSF Graduate Research Fellowship. We would like to thank Marco Ribeiro, Erik Štrumbelj, Avanti Shrikumar, Yair Zick, the Lee Lab, and the NIPS reviewers for feedback that has significantly improved this work.

Chapter 4

Explainable machine learning predictions to help anesthesiologists prevent hypoxemia during surgery

Hypoxemia causes serious patient harm, and while anesthesiologists strive to avoid hypoxemia during surgery, reliably predicting future intraoperative hypoxemia is not currently possible. Using minute by minute EMR data from fifty thousand surgeries, we developed and tested a machine learning based system called *Prescience* that predicts real-time hypoxemia risk and presents *an explanation of factors* contributing to that risk during general anesthesia. Prescience improved anesthesiologists' performance when providing *interpretable hypoxemia risks with contributing factors*. The results suggest that if anesthesiologists currently anticipate 15% of events, then with Prescience assistance they could anticipate 30% of events or an estimated additional 2.4 million annually in the US, a large portion of which may benefit from early intervention because they are associated with modifiable factors. The prediction explanations are broadly consistent with the literature and anesthesiologists' prior knowledge. Prescience can also help improve clinical understanding of hypoxemia risk during anesthesia care by providing general insights into the exact changes in risk induced by certain patient or procedure characteristics. Making predictions of complex medical machine learning models (such as Prescience) interpretable has broad applicability to other data-driven prediction tasks in medicine.

4.1 Introduction

Over 200 million surgeries are performed worldwide every year, with 30 million in the United States alone [62]. Though an integral part of healthcare, surgery and anesthesia pose considerable risk of complications and death. Studies have shown a perioperative mortality rate of 0.4 to 0.8% and a complication rate of 3 to 17%, just in industrialized countries [53, 71]. Fortunately, half of these complications are preventable [53, 71]. With increasing adoption of electronic medical record systems, high fidelity heterogeneous data are being captured during surgery and anesthesia care, yet the utilization of this data to improve patient safety and quality of care remains poor [120]. There is untapped potential for data science to utilize perioperative data to positively impact surgical and anesthesia care [107]. To address this unmet need we leverage recent advances in perioperative informatics and present new machine learning methods to predict harmful physiological events and to inform anesthesiologists.

Hypoxemia or low arterial blood oxygen tension is an unwanted physiological condition known to cause serious patient harm during general anesthesia and surgery [38]. Hypoxemia is associated with cardiac arrest, cardiac arrhythmias, postoperative infections and wound healing impairments, decreased cognitive function and delirium, and cerebral ischemia through a number of metabolic pathways [164]. Despite the advent and use of pulse oximetry to continuously monitor blood oxygen saturation (SpO_2) during general and regional anesthesia, hypoxemia can neither be reliably predicted nor prevented at future time points [40]. Real-time blood oxygen monitoring through pulse oximetry only allows anesthesiologists to take reactive actions to minimize the duration of hypoxemic episodes after occurrence. Decision support systems that

process electronic medical record data have been shown to help increase adherence to guidelines, but remain primarily reactive rather than predictive in nature [84, 52]; see [120] for a full review. If hypoxemia can be predicted or anticipated before it occurs, then actions can be taken by anesthesiologists to proactively prevent hypoxemia and minimize patient harm.

Machine learning (ML) techniques use statistical methods to infer relationships between patient attributes and outcomes in large datasets, and have been successfully applied to predict adverse events in health care settings, such as sepsis, or patient deterioration in the intensive care unit [41, 96, 64, 148, 20]. Yet ML techniques to predict adverse events such as hypoxemia in a considerably more complex setting such as the operating room are currently lacking. Moreover, though previous complex ML approaches provide good prediction accuracy, their application in an actual clinical setting is limited because their predictions are difficult to interpret, and hence not actionable. Interpretable methods explain *why* a certain prediction was made for a patient, i.e., specific patient characteristics that led to the prediction. This lack of interpretability has thus far limited the use of powerful methods such as deep learning and ensemble models in medical decision support.

We present an ensemble model based machine learning method, *Prescience*, that predicts the near-term risk of hypoxemia during anesthesia care *and* explains the patient and surgery specific factors that led to that risk (Figure S1). We believe this is an important step forward for machine learning in medicine because while machine learning models have significantly improved the ability to predict a patient's future condition [34, 114], the inability to explain the predictions from accurate, complex models is a serious limitation. Understanding what drives a prediction is important for determining targeted interventions in a clinical setting. For this reason, machine learning methods employed in clinical applications avoid using complex, yet more accurate models and retreat to simpler interpretable (e.g., linear) models at the expense of lower accuracy. To address this problem, some approaches have achieved interpretability by carefully limiting the complexity of the machine learning model [20]. In contrast, we demonstrate how to retain interpretability, even when complex models such as nonparametric methods or deep learning are used, by developing a method to provide theoretically justified explanations of model predictions building on recent advances in model-agnostic prediction explanation methods [167, 141, 101, 102]. This allows these accurate, but traditionally hard to interpret, models to be used while still providing intuitive explanations of what led to a patient's predicted risk. Our ability to provide simple explanations of predictions from arbitrarily complex models helps eliminate the typical accuracy vs. interpretability tradeoff, thus allowing broader applicability of machine learning to medicine.

Prescience was trained to use standard operating room sensors to predict hypoxemic events in the near future and explain why an event is, or is not, likely to occur. It departs from the relatively few previous approaches to this problem in two important ways:

First, unlike ElMoaqet et al. who used a linear autoregressive support vector machine on arterial oxygen saturation times series [96], and Tarassenko et al. who used Parzen windows to find outliers from five input patient measurement types [171], Prescience integrates a comprehensive dataset from a hospital's Anesthesia Information Management System (AIMS) (see Methods for details). While some operating room forecasting approaches have relied on simulated physiology [168], the AIMS data consists of high fidelity *real-time data* – such as time series data from patient monitors and anesthesia machines, bolus and infusion medications, input and output fluid totals, laboratory results, templated and free text descriptions of anesthesia techniques and management, and *static data* – such as American Society for Anesthesiology (ASA) physical status, surgical procedure and diagnoses codes [2], as well as patient demographic information such as age, sex, smoking status, height and weight. Continuously integrating a broad set of patient and procedure *features* extracted from the AIMS data, Prescience surpasses human-level accuracy while maintaining consistent performance during every minute of a surgery.

Second, Prescience explains why a prediction was made, regardless of the complexity of the machine learning model used to make the prediction. Significant progress has been made recently integrating predictive machine learning solutions into medical care [41, 96, 64, 148]. However, accurately and intuitively conveying to doctors *why* a prediction was made remains a key challenge. For example, a numeric representation of risk is useful (e.g., the 2.4 odds ratio in Figure S1). However, a more detailed presentation that shows the risk is due to the patient's BMI, current tidal volume, and pulse rate is more clinically meaningful since some factors may be modifiable and result in clinical changes mitigating that risk (Figure S1). Typically, understanding why a prediction was made requires limiting the complexity of the model [20], but Prescience

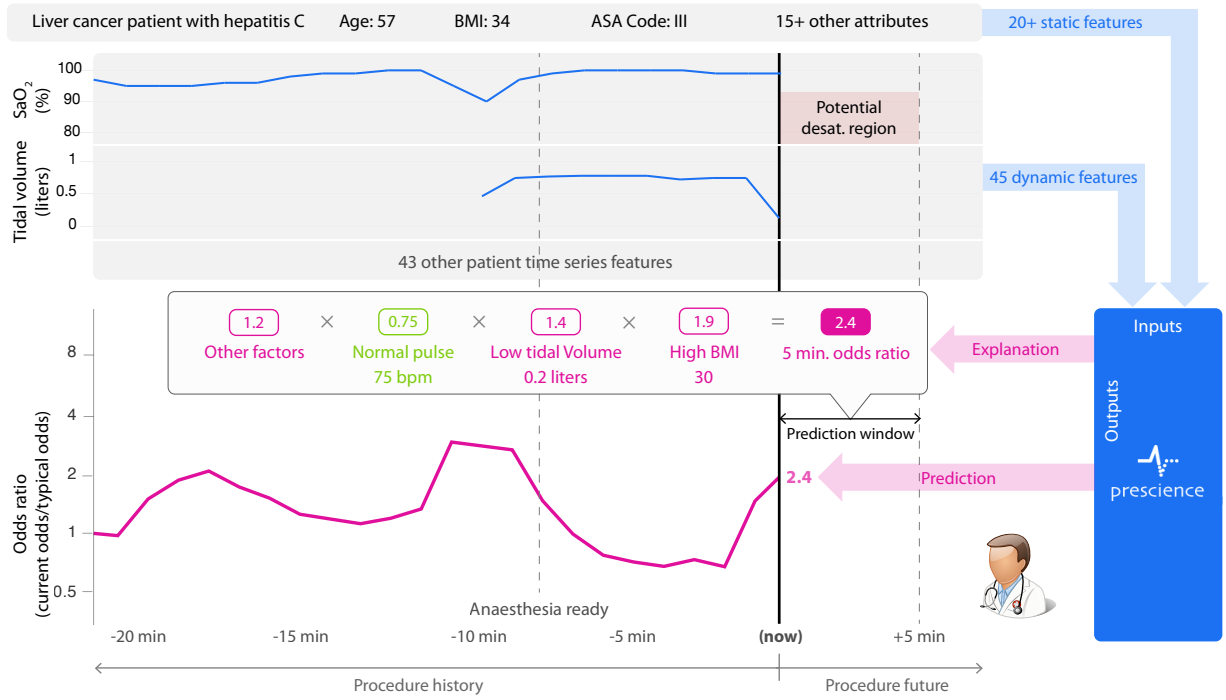


Figure S1: **Prescience integrates many data sources into a single risk, which is explained through a succinct visual summary.** A wide variety of data sources were used to build a predictive model of hypoxemia events. An explanation (shown above) is then built for each prediction. Purple features have values that increased risk, while green features decreased hypoxemia risk. The combination of impacts of all features is the predicted Prescience risk; in this case the odds are 2.4 times higher than normal. Each feature impact value represents the change in risk when that feature's value is known vs. unknown. Qualitative terms such as "low" or "high" are based on a feature value's distribution in our dataset.

enables explanations for models of arbitrary complexity. The feature impact values computed by Prescience essentially represent the change in the model's predicted risk when we observe a feature (such as a patient's weight) vs. when we do not observe the feature (such as not knowing a patient's weight). This change in a model's output prediction when a feature is observed indicates its importance for the prediction. Feature importances do not imply a causal relationship, and so do not represent a complete diagnosis of hypoxemia in a patient. However, they do enable an anesthesiologist to better formulate a diagnosis by knowing which attributes of the patient and procedure contributed to the current risk predicted by the machine learning model.

4.2 Results

To demonstrate the value of Prescience's explained predictions and gain insight into factors affecting intraoperative hypoxemia, we present the following results: 1) a comparison of Prescience hypoxemia predictions against anesthesiologists' predictions with and without the aid of Prescience, 2) an example of how Prescience explains hypoxemia risk at a specific time-point during a surgical procedure, 3) a comparative summary of relevant AIMS data features for hypoxemia prediction chosen by Prescience and by anesthesiologists, and 4) a detailed presentation of key risk factors for hypoxemia identified by Prescience.

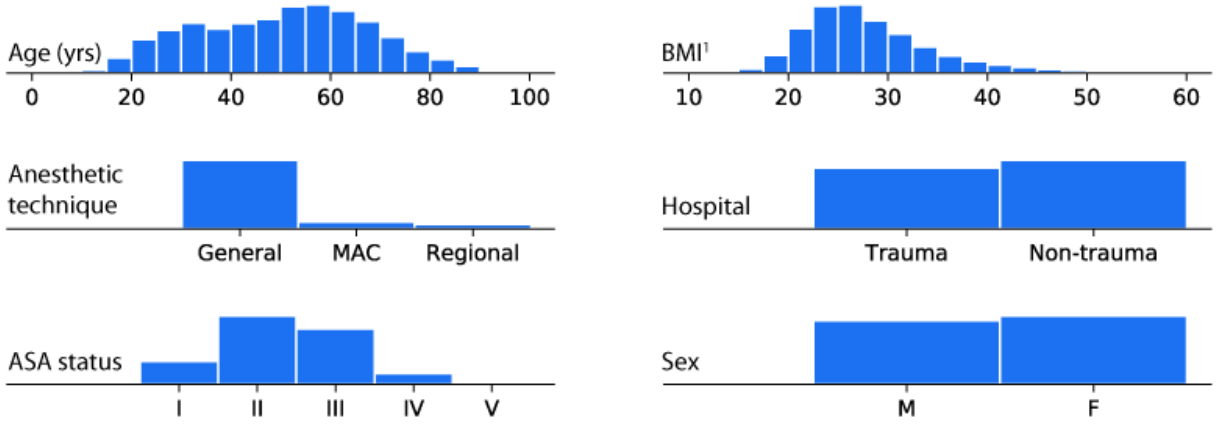


Figure S2: **Patient and procedure characteristics.** Histograms summarizing basic properties of the anesthesia procedures used for training. Prescience was trained and evaluated using data from 53,126 procedures recorded at two hospitals over two years. MAC = monitored anesthesia care; M = Male and F = Female, ASA = American Society of Anesthesiologists; BMI = Body Mass Index. (In our dataset 37.4% of adults aged 20 or over have a BMI of 30 or more, which is a close match to the U.S. obesity rate of 37.9% [123]. Note that the 53k procedures represent 36k unique patients.)

4.2.1 Prescience overview – data preparation, model learning and feature importance estimation

Based on World Health Organization recommendations and for the purposes of prediction, we defined hypoxemia as the decrease in SpO_2, i.e., arterial blood oxygen saturation as measured by pulse oximetry, to a threshold value of 92% or lower (see Methods; Supplementary Figure S7). From the AIMS data, we extracted 3,797 *static extracted features* for each patient from 20+ original static sources and an expanded superset of 3,905 *real-time* and *static extracted features* for each time point during anesthesia care from the 20+ original static sources as well as 45 different real-time data sources (see Methods; Supplementary Table S1). Features such as words from text data get directly mapped to the display in Figure S1, while sets of features from a single time series (such as tidal volume) are combined in Figure S1. We excluded select cases (heart transplant, lung transplant, tracheostomy, and coronary artery bypass surgeries) in which SpO_2 and other hemodynamic parameters can be significantly affected by non-physiological measurements such as during cardiopulmonary bypass. All the experiments were performed after appropriate Institutional Review Board (IRB) approval (see Methods), with clinical data summarized in Table S2.

We trained a gradient boosting machine model [46] to solve the following two types of prediction problems: A) *initial prediction*: predicting at the start of a procedure the risk of hypoxemia anytime during a procedure based on the static extracted features, and B) *real-time prediction*: predicting hypoxemia in the next 5 minutes at various points of the operative period based on real-time and static extracted features collected up to that time point. We chose this 5 minute window to be long enough that an anesthesiologist could have time to intervene, while also keeping the window short enough that it represents near term risks which would benefit from immediate attention. For task A) we used 42,420 procedures (a single surgical case) as *training samples* to train the gradient boosting machine, 5,649 procedures as *validation samples* to choose the tuning parameters for the gradient boosting machine (and other prediction models for comparison), and 5,057 as *test samples* for comparing across different prediction models (Supplementary Figure S10). For task B), we used 8,087,476 per-minute time points as training samples, 1,053,629 as validation samples, and 963,674 as test samples, where all time points from the same procedure were included in the same sample set and no missing data imputation was performed (Supplementary Figure S9). Dividing the time points by procedure is important since samples from the same procedure are not independently and identically distributed, but have some time dependence. To ensure that there was no bias towards the final test set, the test data was initially

compressed and left compressed until method development was completed.

As shown in Supplementary Figures S9 and S10, the gradient boosting machine outperforms alternative prediction models previously used for similar problems.

For tasks A) and B), we use 198 and 523 test samples respectively for evaluating anesthesiologists' performance for initial and real-time prediction tasks, respectively (see below). Prescience outputs the risk prediction and its explanations (Figure S1; Figure S4A) which show a set of features that increased (purple-colored) and decreased (green-colored) the risk.

We developed an efficient, theoretically justified machine learning technique to estimate the importance of each feature on a prediction made for a single patient, which drives real-time explanations (Figure S4) for the Prescience model. We verified the quality of the explanations given to the anesthesiologists (in the experiments described below) by comparing the explanations with the change in model output when a feature is perturbed (Supplementary Figure S11). We also developed effective visualizations of these explanations that encodes them in a compact visual form for anesthesiologists (Figure S1; Supplementary Figures S12-S14), and a more detailed visualization which highlights the relevant contributing features (Figure S4) (see Methods for details).

4.2.2 Prescience improves anesthesiologist's ability to predict hypoxemia

To test the potential of Prescience to aid hypoxemia prediction we replayed prerecorded intraoperative data from test sample procedures in a web-based visualization to five practicing anesthesiologists (Supplementary Figures S12-S14). Each anesthesiologist was given two types of prediction tasks: A) *initial prediction* (198 tasks), and B) *real-time prediction* (523 tasks). For each prediction task, anesthesiologists were asked to provide a *relative risk* of hypoxemia as compared to a normal acceptable risk, for example, 0.01 for $1/100^{\text{th}}$ the normal risk or 3.4 for 3.4 times the normal risk. These relative risks were then used to calculate standard receiver operating characteristic (ROC) curves averaged over five anesthesiologists as shown in Figure S3, which plots the true positive rate (i.e., % of desaturations correctly predicted) in the y-axis against the false positive rate (i.e., % of non-desaturations incorrectly predicted) in the x-axis. Note that ROC curves only depend on the order of the relative risk values among predictions from a single anesthesiologist. This eliminates the need to choose a threshold and the need to separately calibrate risk scores between anesthesiologists.

Figure S3A-B shows that for both types of prediction tasks, predictions made by Prescience (purple) are considerably more accurate than anesthesiologists' predictions (green). The prediction accuracy of anesthesiologists (green) markedly improved when the anesthesiologists were given Prescience's risk prediction and its explanations in addition to the original procedure data (blue) (Supplementary Figures S12-S14). A clear separation between the performance of anesthesiologists with (blue) and without (green) the aid of Prescience is observed for both initial prediction (Figure S3A, P-value < 0.0001) and real-time prediction (Figure S3B, P-value < 0.0001). This suggests that Prescience can enhance anesthesiologists' assessment of future risk and their ability to proactively anticipate hypoxemia events. Interestingly, the prediction performance of anesthesiologists with Prescience explanations (blue) was slightly lower than direct predictions from Prescience (purple). This means that when the anesthesiologists adjust their risk estimate for a patient away from what Prescience originally predicted they are more likely to be wrong than right.

To avoid the scenario in which an anesthesiologist is tested on the same prediction task twice – one with and the other without Prescience, we created replicate test sets by dividing the prediction tasks into two groups of similar size: (100, 98) tasks for initial prediction and (260, 263) tasks for real-time prediction. Each of the five recruited anesthesiologists was assigned to receive Prescience's assistance in one of these two replicate test sets (Methods). The procedures shown to anesthesiologists were chosen such that $\sim 50\%$ showed at least one incident of hypoxemia (for preoperative prediction), and time points were chosen such that $\sim 33\%$ had hypoxemia in the next 5 minutes (for intraoperative prediction). The anesthesiologist test time points for hypoxemia were chosen to be drops of SpO_2 (Supplementary Figure S7) with a preceding period of stable and normal SpO2. However, the entire dataset includes easier to predict hypoxemic events following prior SpO2 drops and decreasing SpO2 trends. For the entire dataset, Prescience achieves an even higher area under the ROC curve of 0.90 (Supplementary Figure S9).

If we extrapolate the real-time results to the 30 million annual surgeries in the US under the assumption that doctors anticipate 15% of hypoxemic events while SpO_2 is still ≥ 95, then with Prescience assistance they may be able to anticipate 30% of these events, or approximately 2.4 million additional episodes of hypoxemia

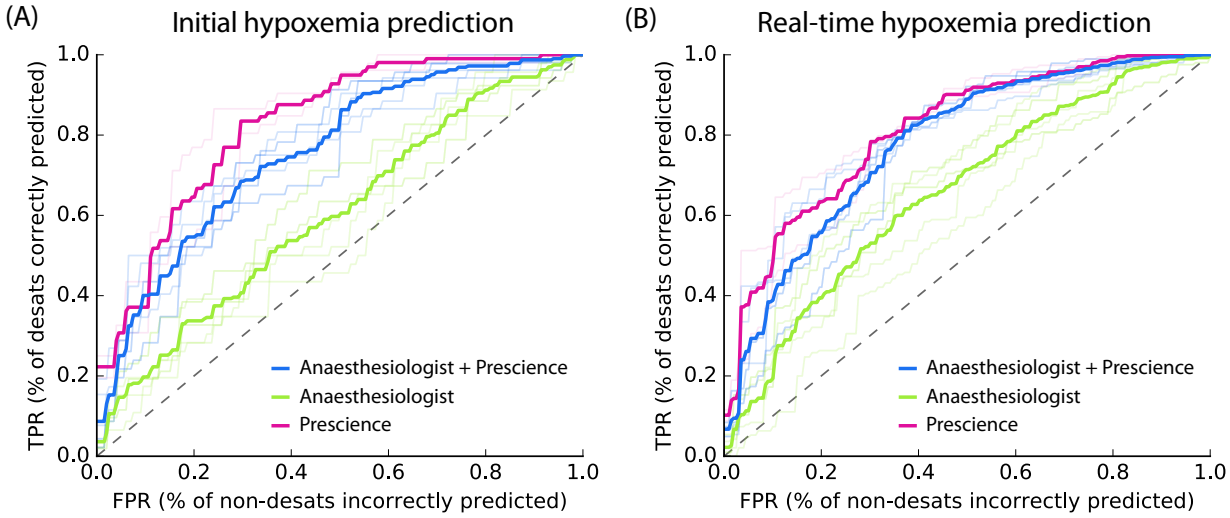


Figure S3: **Pooled comparison of five anesthesiologists' prediction performance with and without assistance by Prescience.** Receiver Operating Characteristic (ROC) plots comparing five anesthesiologists' predictions from recorded data with and without Prescience assistance. Light colored lines represent individual anesthesiologist's performances; dark lines represent their average performance. (**A**) For initial risk prediction, anesthesiologists (green, AUC = 0.60) performed significantly better with Prescience assistance (blue, AUC = 0.76; P-value < 0.0001) than without Prescience assistance, and Prescience performed better in a direct comparison with anesthesiologists (purple AUC = 0.83); P-value < 0.0001). (**B**) For intraoperative real-time (next 5 minute) risk prediction anesthesiologists (green, AUC = 0.66) again performed better with Prescience assistance (blue, AUC = 0.78; P-value < 0.0001), and Prescience alone outperformed anesthesiologists predictions (purple, AUC = 0.81); P-value < 0.0001). Note that the False Positive Rate (FPR) (x-axis) measures how many points without upcoming hypoxemia were incorrectly predicted to have upcoming hypoxemia. The True Positive Rate (TPR) (y-axis) measures what percentage of hypoxemic events was correctly predicted. P-values were computed using bootstrap resampling over the tested time points while measuring the difference in area between the curves. If we instead resample over anesthesiologists we observe bootstrap P-values of 0, and t-test P-values < 0.001 for Prescience improvements. See Supplementary Figure S16 for plots of the statistical separation between the mean ROC curves across all false positive rates.

annually (defined here as $SpO_2 \leq 92$). Since 20% percent of the Prescience risk prediction is based on drugs and settings under the control of the anesthesiologist, a large portion of these predicted events may benefit from early intervention.

The anesthesiologists consulted had experience after residency ranging from 3 to 26 years (median 7 years), and were all actively practicing at University of Washington Medical Center, VA Puget Sound Health System or Seattle Children's hospital. The extensive experience level of the anesthesiologists who participated in our study may not represent the typical experience level of anesthesia providers, especially when nurse anesthetists and residents in training provide anesthesia care.

When using Prescience predictions to generate early warning alarms in the operating room, it is important to minimize false alarm rates. This can be accomplished by adjusting the tradeoff between precision (the positive predictive value) and recall (the sensitivity). High precision means a low false alarm rate (1 - precision), however, it comes at the cost of low recall. Supplementary Figure S17 plots the precision and recall tradeoff for Prescience on the full set of test time points. Since the performance of the complex model in Prescience improves with larger datasets, we also included results from a model trained on an expanded 175 thousand procedure dataset to measure the benefit of using more data to train Prescience. The larger dataset resulted in notably better performance and could capture 9% of all minutes with upcoming hypoxemia at 70% precision, (or 44% of all minutes with upcoming hypoxemia at a precision of 30% if the threshold for precision-recall tradeoff is selected for higher recall). These are strikingly higher precisions than those we project anesthesiologists would achieve on the full test data set (Supplementary Figure S17). We also note that the predictive capacity can be further improved by shortening the predictive window to less than 5 minutes (Supplementary Figure S15).

Anesthesiologists must not only decide when to act to prevent hypoxemia, but also when not to act. To assist in this, Prescience can predict not just when hypoxemia will occur, but also when it will not occur. Prescience can predict when hypoxemia will not occur for 60% of all time points while maintaining a precision of 99.9% (Supplementary Figure S18).

4.2.3 Explained risks reveal both procedure and time specific effects

An explanation from Prescience represents the effects of interpretable groups of extracted patient features (see Figure S1 and Figure S4A), where each group corresponds to the set of extracted patient features from a single input feature in the AIMS dataset, such as the SpO2 monitor time series. These effects explain why the model predicted a specific risk, and thus allow an anesthesiologist to plan appropriate interventions. In Figure S1 only the most significant features contributing to hypoxemia risk are shown for quick reference, however in Figure S4 the relative contributions of all patient and case features (i.e. attributes) towards hypoxemia risk can be seen at every sample time point during a procedure (Figure S4B). Without a meaningful explanation, the sudden increase in risk shown at the time point marked 'Now' might be hard to interpret; however, by representing the predicted risk as a cumulative effect of contributing patient and procedure features, the reason for the increase becomes clear (Figure S4A).

The increase in the risk of hypoxemia in the next 5 minutes shown in Figure S4 is driven by a set of features capturing both static attributes, such as patient height and weight, and dynamic parametric values, such as tidal volume (i.e., volume of gas exhaled per breath) and administration of drugs. The risk explanation bar in Figure S4A has purple-colored features that push the risk higher (to the right) and green-colored features that push the risk lower (to the left). Each group of features is sorted by the magnitude of their impact, and the largest impact features are labeled. Through this representation we can see that many of the 3,905 real-time extracted features have only a small impact, and the risk for this time point is predominantly driven by a few features. The choice of features provided to the model was driven by the data recorded in the AIMS system and hence was available for training. Rather than only provide the model with features we believed important, we let the model use any feature it chose. This means that it may find features we would not initially expect are predictive of hypoxemia. For some of these features it is helpful before final deployment in an operating room to tag them with indicators of how they relate to hypoxemia risk. This can help anesthesiologists quickly see non-obvious connections with patient physiology, such as how the muscle relaxant succinylcholine in Figure S4 does not represent a direct causal impact on hypoxemia, but rather is a proxy that captures the risk from a potentially difficult airway or full stomach (in the hospital system we considered, succinylcholine is given to patients with a high risk for a difficult airway during intubation).

Figure S4B shows the trend in the Prescience risk predictions over the course of the procedure. The plot in Figure S4B is equivalent to rotating the feature explanation in Figure S4A by 90 degrees and then stacking the explanations for each time point horizontally. We can see from the risk trend in Figure S4B that the large increase in risk at the current time was driven by 'Tidal volume', meaning a drop in the patient's tidal volume. The future SpO_2 (blood oxygen concentration) measurements confirm that the patient did indeed progress to hypoxemia (i.e., $SpO_2 \leq 92$). Not only does Prescience alert anesthesiologists when a patient's risk for hypoxemia is high, but also provides information on the factors and their relative contributions driving the risk. This informed risk prediction enables anesthesiologists to plan an appropriate course of action to avoid hypoxemia.

4.2.4 Averaged feature importance estimates broadly align with a survey of prior expectations

To gain an understanding of the general impact of features across all procedures, we computed the average importance of each feature in the Prescience model. In contrast to the explanations shown in Figure S1 and Figure S4A which are specific to a single prediction at a particular time point, these average feature importance estimates are over many procedures and time points [24]. These averaged feature importance estimates are shown for both initial prediction (Figure S5A) and real-time prediction (Figure S5B).

To estimate which clinical features anesthesiologists use to estimate hypoxemia risk, we first performed a survey before using Prescience, which asked four anesthesiologists to list the most important factors they consider when assessing the risk of hypoxemia, both before (for initial prediction) and during a procedure (for real-time prediction). Their responses were then aggregated into a single ranked list of features (Supplementary Tables S2 and S3). Figure S5 shows the rankings chosen by anesthesiologists next to the feature importance estimates derived by Prescience for (A) initial and (B) real-time predictions. The ranking of features by anesthesiologists appears to correspond well with the ranking by Prescience.

As another way to measure which features anesthesiologists' think contribute to hypoxemia, we learned from anesthesiologists' behavior by training a separate gradient boosting machine model based on their predictions. This allows a direct comparison between the anesthesiologists and Prescience on the same set of features. We fit this model to all the anesthesiologist relative risk predictions using 10-fold cross validation. We then computed the feature importance estimates for this model that was trained to mimic the behavior of anesthesiologists. Given the relatively smaller set of training examples used to train the model (198 initial predictions, and 523 real-time predictions), we used bootstrapping to estimate the variability of the feature importance estimates (Figure S5 right).

In general, there is reasonable agreement between the Prescience feature importance estimates and those identified by the anesthesiologists. However, there are important differences that may stem from the comprehensive nature of the Prescience analysis, while anesthesiologists necessarily focus on what they consider the most likely causes for hypoxemia concern. One striking difference is the reduced role of current SpO_2 levels in anesthesiologists' predictions. While anesthesiologists are clearly influenced by the recent patterns of patient SpO_2 levels, Prescience strongly depends on these patterns, while anesthesiologists appear to be equally influenced by other factors, such as end tidal CO_2 and peak ventilation pressure. The second and fourth ranked features by anesthesiologists for initial prediction were lung disease and asthma respectively, and did not show up as important features for Prescience. This is potentially because they must be extracted from preoperative text notes and only about 1% of the procedures recorded the term "COPD" for example, and only 3% of case notes mention "asthma" .

Our study used data from two hospitals and initial hypoxemia predictions were driven by a bias between the two hospitals. This is perhaps unsurprising since one hospital is a level 1 trauma center and a significant proportion of its surgical cases involve trauma patients who are more susceptible to hypoxemia. However, it is interesting to note that the importance of hospital as a risk factor became insignificant for the intraoperative real-time predictions, presumably because the risk differences in each hospital were captured by the real-time features.

Among the static features, BMI (body mass index) and age were significant risk factors. These features are well understood in the medical literature as risk factors that can increase the chances of hypoxemia [100, 76]. The American Society of Anesthesiology (ASA) physical status feature represents the severity of a patient's medical condition and a higher ASA number indicates a higher comorbidity. Prescience determined

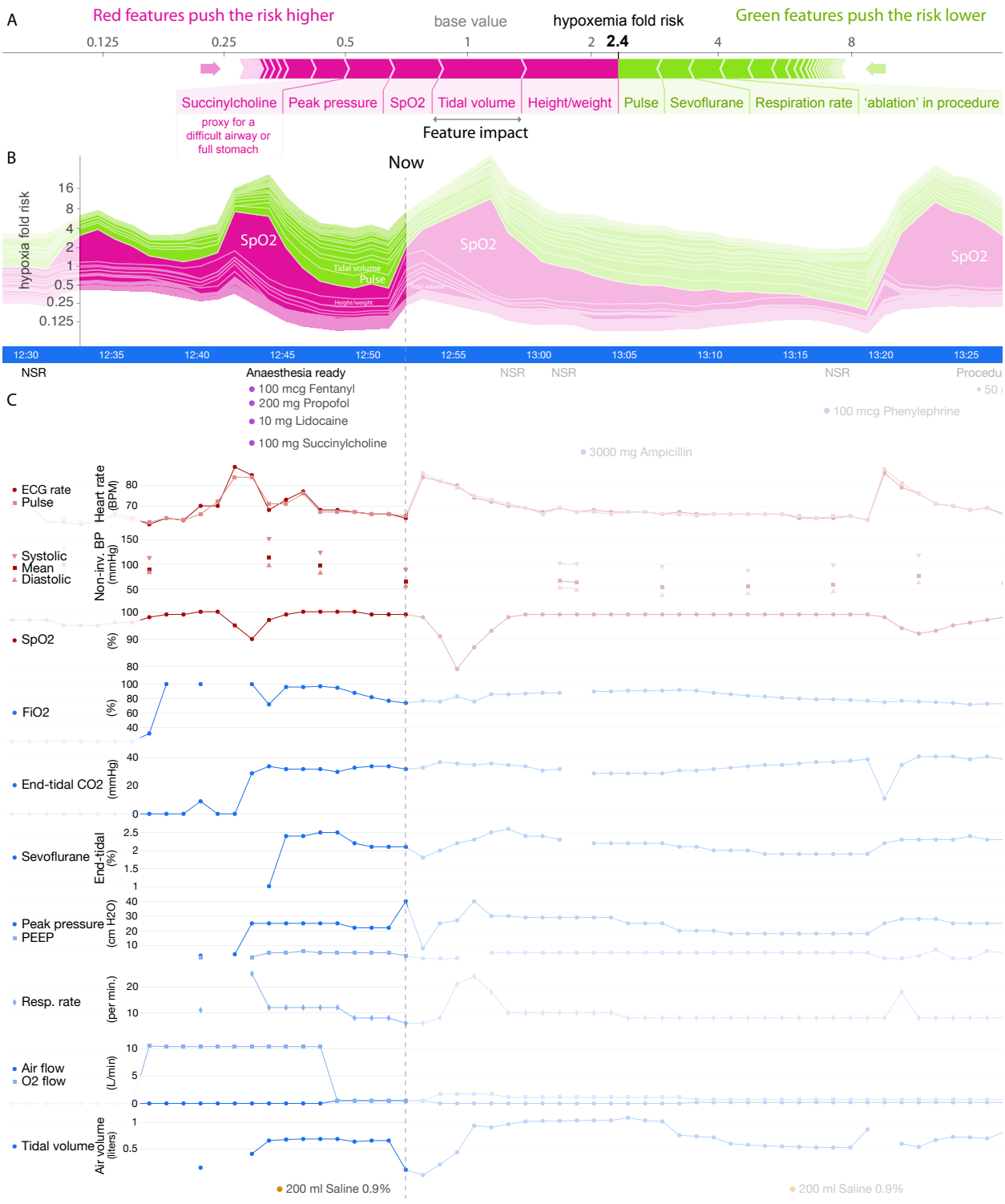


Figure S4: **Sample real-time prediction during a procedure.** One hour of data is shown from a procedure. (**A**) Explained risk of hypoxemia in the next five minutes. (**B**) Plot of the explained risks evolving over time. This plot is equivalent to rotating (A) 90 degrees and stacking the risk explanations for every time point horizontally. (**C**) A subset of the patient data for this procedure, plotted both before and after the current time point.

that higher ASA physical status values predisposes a patient to higher hypoxemia risk. While this finding may be clinically intuitive, anesthesiologists can now use this information in their preoperative evaluation as a pre-specified risk factor for intraoperative hypoxemia. Eye procedures were informative to the model and carried a reduced risk of hypoxemia, while surgeries for fractures had a slightly higher risk. These patterns may reflect the composite risk of hypoxemia to patients undergoing these particular procedures. In the case of eye surgeries, the risk was lower even though many are elderly and have accompanying co-morbid conditions. The low risk of eye procedures is in contrast to fractures which carry an increased risk for hypoxemia. Together these findings provide new data on the relative "risk" of these procedures which has implications for anesthesia staffing, need for equipment, and preparation for the ability to rescue patients from hypoxemia. Eye procedures and surgeries for fractures are two examples of text based features extracted from diagnosis and preoperative procedure notes. They demonstrate that unstructured text notes can be combined with structured patient data to improve patient risk prediction. Although many of the risk factors identified by Prescience reconfirmed those expected by the anesthesiologists, it is informative that Prescience independently identified these features with no prior knowledge.

Among real-time (intraoperative) features SpO_2 (arterial oxygen saturation) is, as expected, the strongest predictor of future potential decreases in SpO_2. End tidal CO_2 (amount of carbon dioxide exhaled by the patient) was also a significant intraoperative feature identified by Prescience as predictive of hypoxemia. Lower values may indicate inadequate ventilation or airway obstruction which can in-turn increase the risk of hypoxemia. Prescience also determined that hypotension (systolic blood pressure below 80mmHg) increases the risk of hypoxemia. On the other hand, higher FiO_2 (inspired O_2 concentration) and positive pressure ventilation can reduce the risk of hypoxemia, as expected by the anesthesiologists.

4.2.5 Prescience's estimated importance of individual features on hypoxemia risk highlight important clinical relationships

Three important features each for beginning of anesthesia care (initial) and during anesthesia care (real-time) predictions were chosen to illustrate how the Prescience model modifies hypoxemia risk based on changes to feature characteristics (Figure S5). While many such relationships are present for the various features, Figure S6 shows a representative selection demonstrating informative risk relationships that are captured in the Prescience model.

Among static features we find that patient BMI has a clear effect on the risk of hypoxemia. When the BMI is over 26, the risk of hypoxemia increases linearly until it has more than doubled when BMI is over 50. Though a qualitative association between hypoxemia and body weight is well established in the field of anesthesia [100, 76], Prescience quantifies this relative risk.

Prescience shows that patients with higher ASA physical status codes have higher risk of intraoperative hypoxemia. This is not surprising since higher ASA codes represent increased severity of a patent's physical condition such as preexisting pulmonary and cardiac conditions that can predispose a patient to develop hypoxemia. Prescience data support clinical observations that the risk of hypoxemia more than doubles when the ASA status increases from I to V. Advancing age also predicted intraoperative hypoxemia, likely representing the presence of comorbidities [100]. These data show that BMI > 30, which meets the clinical definition of obesity [21], is associated with intraoperative hypoxemia, suggesting impaired pulmonary mechanics. While we agree that these findings confirm clinical observations and suspicions of the relationship between these patient factors and adverse anesthesiology outcomes, for the first time, Prescience quantifies this association and the risks, giving a more clinically useful interpretation to anesthesiologists.

For real-time prediction, measurements from each time series are represented by a set of multiple features. For simplicity, we focus here only on the effect of the shortest time lag exponentially weighted moving average, which essentially represents the most recent reported value in the time series (see Methods for details).

Tidal volume represents the amount of gas exhaled per breath when the patient is either breathing spontaneously or mechanically ventilated during general anesthesia. As the tidal volume drops below 0.6 liters (keeping all other features the same), Prescience risk for hypoxemia increases. This increase could be due to hypoventilation, in which case anesthesiologists take preventative steps to avoid inadequate ventilation.

End tidal CO_2 represents the amount of carbon dioxide exhaled gas. Figure S6 shows the relationship between end tidal CO_2 and risk of hypoxemia under general anesthesia. End tidal CO_2 below 35 mmHg is associated with an increasing risk of intraoperative hypoxemia. While we cannot definitively attribute

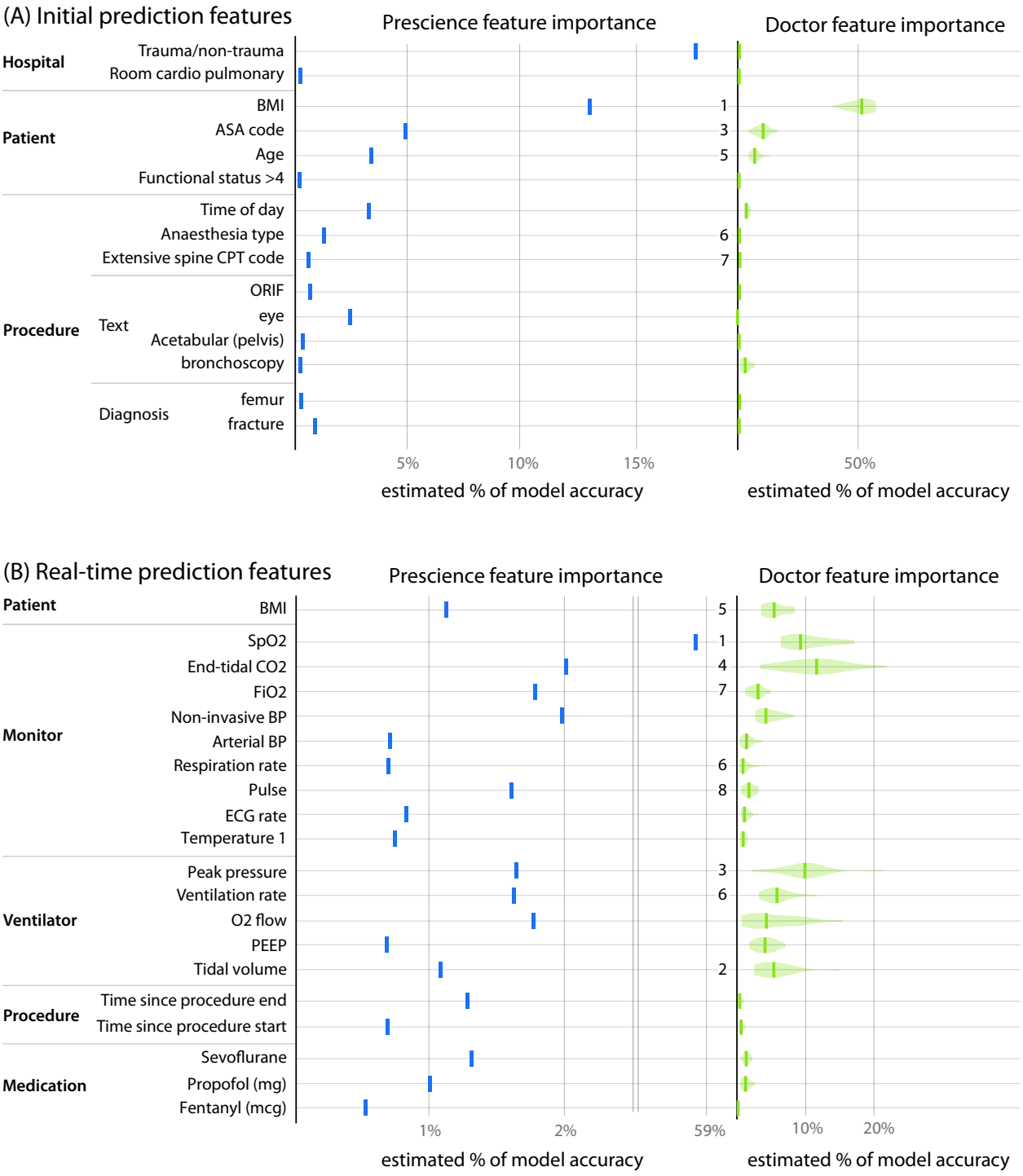


Figure S5: **Comparison of averaged feature importance estimates between Prescience and anesthesiologists for both initial and real-time prediction.** Importance estimates assigned by the Prescience model and anesthesiologists to the top features in both (**A**) initial and (**B**) real-time prediction. The importance of features is measured as the estimated percent of the model's prediction accuracy that is due to that feature. The numbers presented to the left of the imputed anesthesiologist importance estimates are feature rankings from a consensus of anesthesiologist responses about which features they believed would be important. The quantitative anesthesiologist feature importance estimates were estimated using 20 bootstrapped models trained to mimic the anesthesiologist's predictions when unassisted by Prescience.

hypocapnia with intraoperative hypoxemia, these associations may represent underlying patient conditions such as chronic obstructive pulmonary disease that affect both physiological conditions. Alternately, the low end tidal CO_2 and may result from either intentional or unwanted hyperventilation during anesthesia care.

Examining FiO_2 is important because anesthesiologists can control the amount of oxygen delivered to patients. Current practice is to not provide all patients with 100% FiO_2 since not all patients need it, prolonged ventilation with 100% FiO_2 is associated with pulmonary atelectasis, and because oxygen if delivered when not needed is costly and wasteful. These data show that FiO_2 below 40% is independently associated with intraoperative hypoxemia irrespective of other features. These findings provide important information regarding safe practice of FiO_2 in patients during general anesthesia. It is possible that the routine practice of maintaining FiO_2 30% or close to room air may be harmful to patients and not desirable. While these effects are adjusted for all other available features, it is important to note that as with any observational study some residual confounding with patient risk may still exist. This could explain the increase in hypoxemia risk we observed for high O_2 levels.

These representative features illustrate the ability of our machine learning-based prediction method, Prescience, to not only provide explained risk predictions for a complex model, but also quantitative insights into the exact change in risk induced by certain patient or procedure characteristics.

4.3 Discussion

To the best of our knowledge, Prescience is the first method designed to comprehensively integrate high fidelity operating room data to predict intraoperative hypoxemia events before they occur. Based on a comparison against practicing anesthesiologists and existing computational methods applied to other clinical problems, Prescience achieves superior performance when predicting hypoxemia risk from electronically recorded intraoperative data.

To our knowledge, Prescience is also the first intraoperative method to combine high accuracy complex models with interpretable explanations. This combination of accuracy and interpretability allows physicians to receive the best possible predictions while also gaining insight into why those predictions were made. To test how Prescience predictions with explanations would impact an anesthesiologist's ability to estimate hypoxemia risk we compared anesthesiologist predictions with and without Prescience assistance. We observed a clear increase in prediction accuracy when doctors were assisted by Prescience, demonstrating that anesthesiologists may make more accurate hypoxemia risk assessments in the operating room if they had access to Prescience. Prescience based technology may also be an important tool to account for the variation in knowledge and or practice among providers.

Empirically derived black box algorithms such as the bispectral index have been used to track brain states of patients undergoing general anesthesia by processing real-time EEG [119, 5]. These algorithms have been criticized because they do not utilize physiological models, do not identify factors associated with risk of events, and produce empirically derived metrics to represent neurophysiology of how the anesthetics affect the brain. The black box nature of the EEG algorithms has made it difficult to interpret their output and understand how physiological mechanisms and anesthetic states determine the algorithm output. A similar danger exists with the application of complex black-box machine learning models in the operating room, where predictions are difficult to interpret, and hence less actionable. Prescience demonstrates a solution that promises to avoid the obscurity traditionally associated with black-box models, and instead maintain interpretability even as increasingly complex machine learning models are applied to operating room decision support.

It should be clarified that our exercise at developing machine learning methods to predict intraoperative hypoxemia, though promising, should still be considered an initial attempt. In this first attempt, we did not categorize procedures to assess hypoxemia predictions in specific types of procedures. For this reason, clinical interpretation of the results had to be somewhat generic. For enhanced interpretation of risks, future attempts can focus on specific categories of cases and phases of anesthesia. Another future enhancement would be integrating additional preoperative data such as a patient's detailed medical history into the prediction models. Higher fidelity intraoperative data such as patient monitor waveform data could enrich machine learning, thus potentially leading to more accurate predictions. Prospective trials of Prescience during live procedures are also needed before deployment to verify the improvements in anesthesiologist's performance

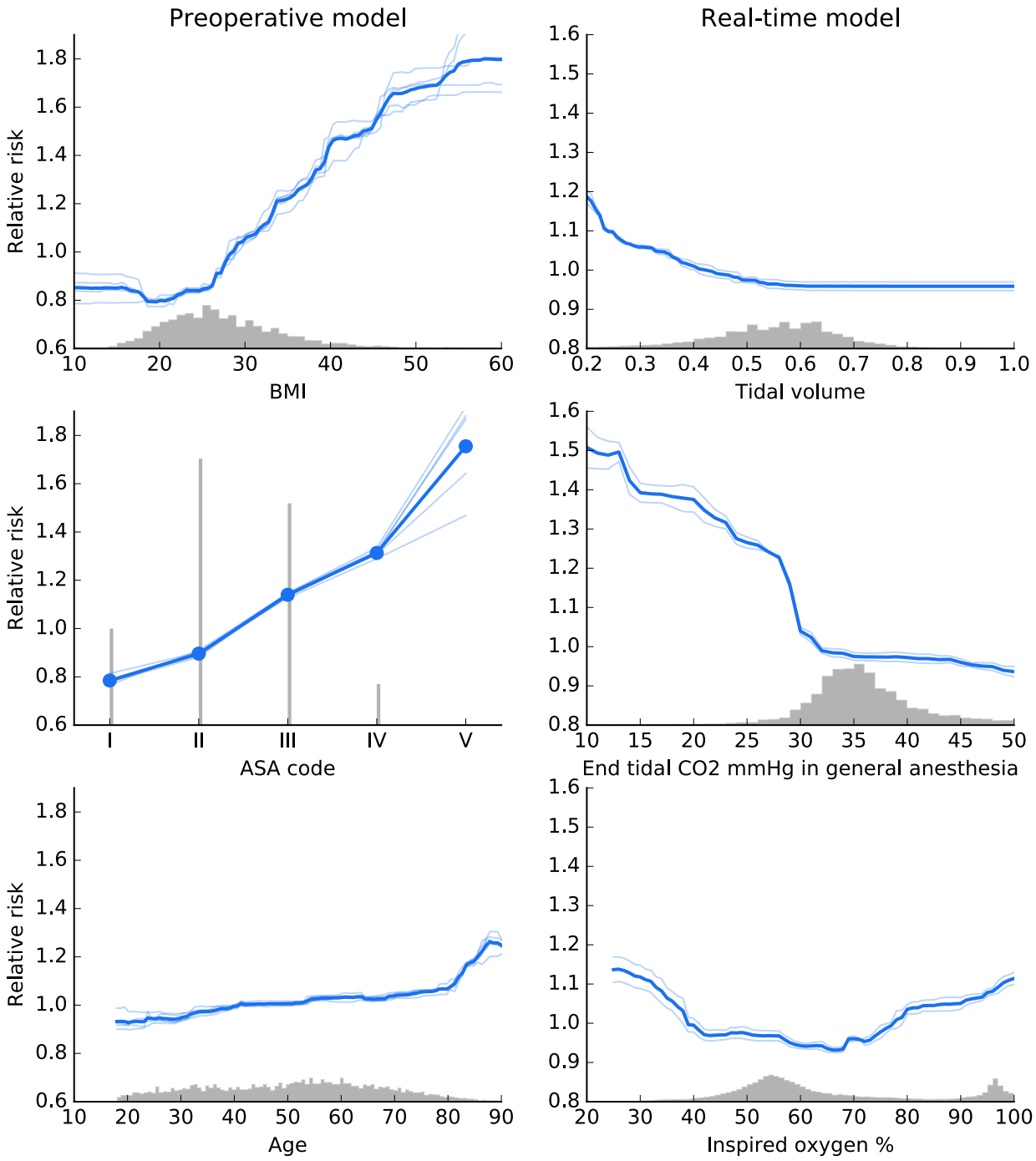


Figure S6: **Effect of varying individual feature values for both initial features (left) and real-time features (right).** These partial dependence plots show the change in hypoxemia risk for all values of a given feature. The gray histograms on each plot show the distribution of values for that feature in the validation dataset. Light colored lines represent model variability from bootstrap resampling of the training data.

we retrospectively observed in prerecorded procedures [43].

This paper focuses on hypoxemia risk during intraoperative anesthesia care. However, the importance of coupling accurate predictions from complex models with interpretable explanations of why a prediction was made, has broad applicability throughout medicine. Extending the approach taken by Prescience and providing these model explanation tools to the community is a clear next step that we have begun (http://github.com/slundberg/shap). Because Prescience effectively decouples the interpretable explanation from the prediction model, we are also free to continue to refine the core prediction model without changing the user experience for anesthesiologists.

The global risk profiles learned by Prescience (Figures S5-S6) are clinically relevant for a number of reasons. First, they show that in the health system examined, trauma hospital patients may be more critically ill as they have more intraoperative hypoxemia. In current times when harmonization of care and standardization are considered to reduce unwanted clinical variation, these data suggest that resources may need to be differentially deployed to address differential rates of adverse events. Second, anesthesiologists can now quantify risks of intraoperative hypoxemia adjusted for other factors to the very elderly, those who are overweight, and those with more comorbid conditions. The exact relationships described in Figure S6 clearly show the patterns and threshold points for the risk. Whereas low tidal volume is often suggested for patients with acute lung injury [58], these data suggest that overall, low lung tidal volumes are, in-fact, associated with intraoperative hypoxemia. The relationship between low end-tidal CO_2 levels and intraoperative hypoxemia may reflect underlying critical illness. Despite our inability to fully exclude residual confounding, these data shed new light on physiological relationships as well as provide a mechanism to facilitate provision of anesthesia care that can mitigate intraoperative hypoxemia.

As a limitation we acknowledge that there are several clinical diagnoses that are associated with hypoxemia, but not directly observable in Prescience. The main clinical diagnoses include mainstem intubation, mucus plug, low FiO2, low tidal volume, tracheal tube balloon leak, patient factors like COPD from smoking, and pulmonary embolus. Among these only low FiO2 and low tidal volume are directly observable in Prescience since the other data elements are not fully captured in the clinical databases. In these cases secondary risk indicators will show up in Prescience. The differential diagnosis of hypoxemia could also have been categorized using ACLS strategies. However, as opposed to a study where factors are *a priori* identified, Prescience considers all available factors, and renders an output with associated relative risks. Clinicians must then evaluate the feature relevance based on context and clinical relevance.

The field of medicine is full of data science challenges that have the potential to fundamentally impact the way medicine is practiced. More and more data driven predictions of patient outcomes are being proposed and used. However, black-box prediction models which provide simply predictions, without explanation, are difficult for physicians to trust and provide little insight about how they should respond. The interpretable explanations used by Prescience represent a powerful technique that can transform any current prediction method from one that provides *what* the prediction is, into one that also explains *why*.

4.4 Materials and Methods

4.4.1 IRB statement

The electronic data for this study was retrieved from institutional electronic medical record and data warehouse systems after receiving approval from the Institutional Review Board (University of Washington Human Subjects Division, Approval # 46889). Protected health information was excluded from the data set that was used for machine learning methods.

4.4.2 Data sources

Our hospital system has installed an Anesthesia Information Management System (AIMS) (Merge AIM, Merge Inc, Hartland, WI) that automatically captures minute by minute hemodynamic and ventilation parameters from the patient monitor and the anesthesia machine. The system also integrates with other hospital electronic medical record (EMR) systems to automatically acquire laboratory and patient registration information. The automatic capture of data is supplemented by manual documentation of medications and anesthesia interventions to complete the anesthesia record during a surgical episode. For the current project, we extracted the high-fidelity anesthesia data from the AIMS database for the period May 2012 through June 2014. Additionally, for each patient, medical history data were extracted from our EMR data warehouse (Caradigm, Bellevue, WA). The high-fidelity anesthesia record data and the corresponding medical history data from the hospital EMR formed the underlying data for machine learning. The various data elements used for machine learning are outlined in Supplementary Table S1.

Element	Description
Patient	
Age	Age of patient
Sex	Sex
ASA Physical Status	ASA physical status
Height (cm)	Height of patient
Weight (kg)	Weight of patient
Patient class	Inpatient or outpatient
Procedure	
Procedure	Procedure description
Procedure code	Surgical procedure
Billing codes	Anesthesia Crosswalk/ procedure codes
Diagnosis	Diagnosis description
Diagnosis codes	ICD-9/10 codes
Location	Operating room location
Facility	Hospital facility
Emergency status	Whether case is an emergency or not – Y/N
Anesthesia type	Type of anesthesia
Case Events	
Anesthesia Start	Time of Anesthesia Start
In Room	Time of patient in room
Induction	Time of Induction start
Anesthesia Ready	Time of Induction end or Anesthesia ready
Procedure Start	Time of Procedure start (incision)

continued on next page

continued from previous page

Closing	Time of Closing
Procedure End	Time of Procedure End
Emergence	Time of start of emergence
Leave OR	Time of leave OR/ Transport to recovery
Anesthesia End	Time of Anesthesia End
Patient monitor/Ventilator Data	(Value, Time & Unit of measurement)
Heart Rate	Heart rate from ECG signal (Patient monitor)
O_2 Sat	O_2 saturation from pulse oximetry (Patient Monitor)
Pulse rate	Pulse rate from pulse oximetry (Patient Monitor)
NIBP – Sys	Cuff BP – systolic (Patient monitor)
NIBP – Dia	Cuff BP – diastolic (Patient monitor)
NIBP - Mean	Cuff BP – mean (Patient monitor)
Art BP - Sys	Arterial BP – systolic (Patient monitor)
Art BP - Dia	Arterial BP – diastolic (Patient monitor)
Art BP - Mean	Arterial BP – mean (Patient monitor)
PA - Sys	Pulmonary artery pressure – systolic (Patient monitor)
PA - Dia	Pulmonary artery pressure – diastolic (Patient monitor)
CVP	Central venous line pressure – mean (Patient monitor)
$EtCO_2$	End tidal CO_2 (Capnography) (Patient monitor)
Resp Rate	Measured respiration rate – capnography (Patient monitor)
FiO_2	Inspired O_2 (Patient monitor)
ET Sevo	End tidal Sevoflurane anesthetic agent (Patient monitor)
ET Des	End tidal Desflurane anesthetic agent (Patient monitor)
ET ISO	End tidal Isoflurane anesthetic agent (Patient monitor)
ET N20	End tidal Nitrous oxide (Patient monitor)
BIS	Bispectral Index (Patient/BIS monitor)
SQI	Signal quality index of BIS (Patient/BIS monitor)
TEMP	Temperature (Patient monitor)
SvO_2	Mixed venous oxygenation (Patient monitor)
CCO	Continuous cardiac output (Patient monitor)
TV	Tidal volume (Patient monitor)
RATE	Ventilator rate setting (Ventilator)
PIP	Peak Inspiratory pressure (Ventilator)
PEEP	Positive End Expiration Pressure (Ventilator)
O_2 FLOW	O_2 flow rate (Ventilator)
Air FLOW	Air flow rate (Ventilator)
N_2O FLOW	N_2O Flow rate (Ventilator)
Cardiac Rhythm	Type of cardiac rhythm
Medications	Delivery information of medications
Time	Delivery time (start / end for infusion medications)
Drug Name	Drug Name

continued on next page

continued from previous page

Dose	Drug dose
Dose Unit	Drug dose unit
Route	Drug route
Fluid totals	Fluid totals
Time	Time of fluid input or output
Fluid Name	Fluid name
Volume	Fluid volume recorded
Laboratory results	Intraoperative laboratory results
Time	Time of taking sample or Lab result time
Lab results	Lab result description
Sample type	Sample type – arterial/venous
Lab result	Result value
Unit	Unit of measurement
Notes	Attestations, AIMS Clinical notes, Note option selections
Time	Time associated with the note
Context code	Context code associated with a note or its selections
Note content	Note description

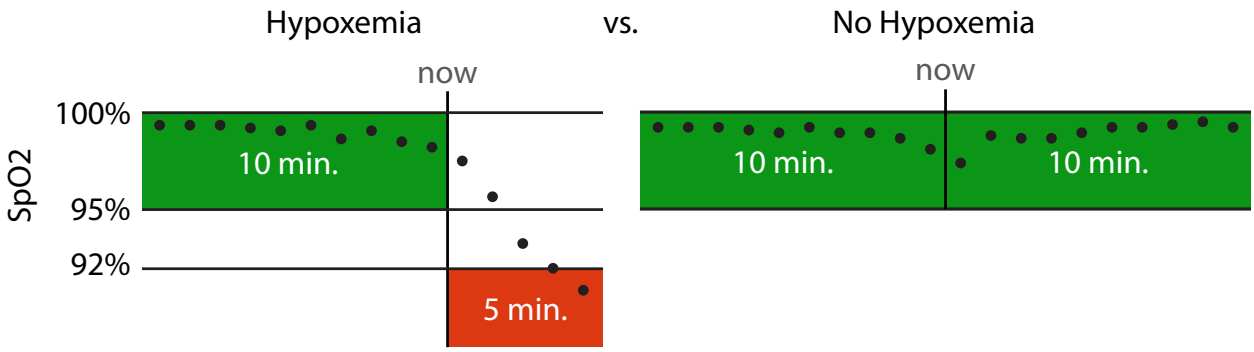
Supplementary Table S1: **Raw data sources used for machine learning.** A wide variety of data sources were used that included text values, time series, discrete values, and quantitative values. These were then processed into thousands of extracted features for machine learning.

4.4.3 SpO2 desaturation labels

We considered $SpO_2 \leq 92\%$ as hypoxemia, which falls between the World Health Organization's recommended intervention level ($< 94\%$) and emergency level ($< 90\%$) [124]. Predictions of hypoxemia were made for a window 5 minutes into the future. If the SpO_2 was $\leq 92\%$ at any point during those 5 minutes then it was considered a positive label, otherwise it was negative. The machine learning algorithm was trained using these *training labels* on all time points where SpO_2 was not already $\leq 92\%$ at that time point.

When evaluating the machine learning algorithm's performance by comparing with anesthesiologists (Figure S3) we deliberately chose to use hypoxemia events encountered after a period of stable and normal SpO2 (Supplementary Figure S7). This was done to maximize the separation observed between different prediction approaches and so minimize the number of time points anesthesiologists needed to label. For a more generalized prediction of all low SpO_2 values the performance reported on the test set using training labels should be used (Supplementary Figure S9). The more stringent testing definition used for Figure S3 excludes some time points, leading to a smaller set of *anesthesiologist testing labels.* Anesthesiologist testing labels were positive only if SpO_2 was $\geq 95\%$ for the past 10 minutes and then fell below 92% in the next five minutes (Supplementary Figure S7; left). Anesthesiologist testing labels were negative only if SpO_2 remained $\geq 95\%$ for the past ten minutes and the next ten minutes (Supplementary Figure S7; right). All the other cases do not have anesthesiologist testing labels. This more restrictive labeling scheme ensures that positive testing labels are clear drops in SpO_2 levels that would be hard to predict in advance, while negative testing labels are clearly not drops in SpO_2 (Supplementary Figure S7).

An important point to consider when building labels for health outcome prediction is that anesthesiologist interventions can affect outcomes. It was recently noted by Dyagilev et al. that models can learn when an anesthesiologist is likely to intervene and hence lower the risk of an otherwise high-risk patient [39]. This means that patients with low risk (from the model) may still need treatment. To address this, they proposed removing examples from the training set where anesthesiologists have intervened. This allows one to learn a model which predicts patient outcome without intervention. In our case, it is not possible to fully identify



Supplementary Figure S7: **Criteria for defining testing labels.** When comparing the performance with anesthesiologists, only time points that clearly either desaturate after little prior SpO_2 warning, or do not desaturate are used. Hypoxemia involves dropping from $\geq 95\%$ to $\leq 92\%$ in the next 5 minutes. Not desaturating means remaining $\geq 95\%$ for both the past 10 minutes and the next 10 minutes.

when or how an anesthesiologist is intervening (and if that intervention prevented hypoxemia), so we sought to address this issue in two ways:

1. It must be recognized that the model predicts hypoxemia when following standard procedures, *not* the occurrence of hypoxemia if the anesthesiologist takes no action to influence hypoxemia. This is a natural assumption in the operating room where interventions that may affect SpO_2 levels are performed frequently.

2. By focusing on clear explanations of why a certain risk was predicted we enable anesthesiologists to identify when the algorithm may be basing its risk on their actions vs. when the risk is based on other factors.

4.4.4 Extracted time series features

To make a prediction at an arbitrary point in time, a consistent set of extracted features should be computed that capture the information present in all previous time points. All the data provided about a procedure is associated with a specific date and time. Text data has the time it was provided, minute-by-minute data from the patient monitor has the time at which each measurement was taken, and single point measurements have the times they were recorded.

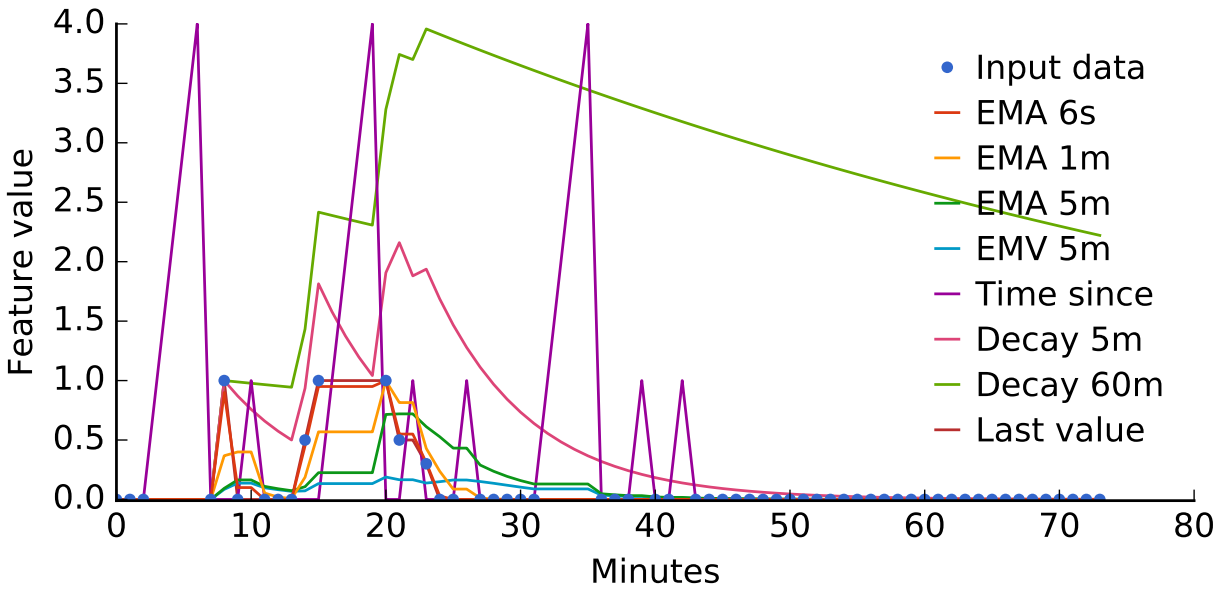
We summarized these unevenly sampled time registered data into a fixed length feature vector at any point in time using several complementary methods:

- Patient data, procedure information, and pre-operative notes are represented by a "last value" extracted feature, which is zero before any data is recorded and the data's value afterwards.

- Time series data are captured using exponentially decaying weighted average and variance estimates using multiple decay rates. These decay rates specify how much impact each past time point has on the computed mean or variance for the time series. We used 6 second, 1 minute, and 5-minute half-life times to capture both high and low frequency components of the signal in each time series (Supplementary Figure S8).

- Drug dose data are captured using both an exponentially decaying sum, and a time since the last measurement. Decay rates with half-lives of 5 minutes and 60 minutes were used to capture both near term and longer average drug dosing effects.

To ensure that there was enough training data for each extracted feature we removed extracted features that had less than 100 recorded data values for the real-time model, and less than 50 for the initial model.



Supplementary Figure S8: **Responses of the eight different time series feature extraction approaches used in Prescience to a sample set of unevenly reported data values.** The blue dots represent the original unevenly sampled data, while curves represent the value of a extracted feature over time. 'EMA' stands for exponential moving average and 'EMV' for exponential moving variance. Both EMA and EMV extracted features are computed over weighted samples, where the weights decay with a specific half-life (6 seconds, 1 minute, or 5 minutes).

4.4.5 Gradient boosting machines for prediction

The extracted features we compute from real-time operating room data have a variety of complex nonlinear interactions. Capturing these requires a model with significant flexibility, and we chose a non-parametric approach called *gradient boosting machines* [46].

We compared the performance of gradient boosting against three baseline methods: Lasso penalized linear logistic regression; a linear SVM autoregressive model previously proposed for predicting hypoxemia based only on the SpO_2 data stream [41]; and an unsupervised Parzen window method used previously to predict patient deterioration [171]. Gradient boosting machines significantly outperformed all baseline methods for our primary endpoint, real-time hypoxemia prediction (Supplementary Figure S9). For our secondary task of initial prediction gradient boosting machines were only slightly superior (Supplementary Figure S10). The large performance gain of gradient boosting for intraoperative prediction (Supplementary Figure S9) is likely because there are 8 million training samples, while for preoperative predictions (Supplementary Figure S10) there are only 42,000 samples and no time series data. Note that for initial prediction the autoregressive SVM and Parzen window methods were not applicable and hence not evaluated.

Gradient boosting machines are non-parametric models that draw a parallel between boosting and gradient descent in function space. They additively build up simpler models, like boosting, and these models are fit to the gradient of the loss at every data point. The most common type of basic model used is a regression tree because it is both robust to outliers and flexible. Taking some small fraction, η , of many trees fit to the gradient results in many small gradient descent steps in function space.

Fitting the trees is computationally challenging on large datasets so we used XGBoost, a recent high performance implementation of gradient boosting machines [24]. For the real-time model we used $\eta = 0.2$ and 1,242 trees, while for the initial model we chose $\eta = 0.1$ and 4,000 trees. Using a smaller η value means more trees are required for fitting, which requires more time to run, but results in a smoother (and generally better) model. For both initial and real-time models we used bagging, where trees were trained on a random 50% subsample of the training data. For the preoperative model the max tree depth was 4 and the minimum

child weight of any branch in the trees was 1. For the real-time model the max tree depth was 6 and the minimum child weight of any branch in the trees was 10.

All method parameters were tuned (and methods were chosen) using a validation set of operating room procedures separate from the final test set used for all final performance results. To ensure that there was no bias towards the final test set, the test data were initially compressed and left compressed until after method development was completed.

4.4.6 Computing feature importance estimates

Understanding why a statistical model has made a specific prediction is a key challenge in machine learning. It engenders appropriate trust in predictions and provides insight into how a model may be improved. However, many complex models with excellent accuracy, such as gradient boosting, make predictions even experts struggle to interpret. This forces a tradeoff between accuracy and interpretability. In response to this we chose to use a model agnostic representation of feature importance, where the impact of each feature on the model is represented using *Shapley values* [167, 151], which have been shown to be the only way to assign feature importance while maintaining two important properties *local accuracy*, and *consistency* (defined below)[101]. The application of these values in Prescience uses fast estimation methods we have developed to compute the Shapley values (i.e., the estimated importance of features for a particular prediction) in a real-time manner [101, 102].

Shapley values are from the game theory literature and provide a theoretically justified method for allocation of a coalition's output among the members of the coalition. In Prescience the coalition is a set of interpretable model input feature values, and the coalition's output is the value of the prediction made by the model when given those input feature values. Feature impact is defined as the change in the expected value of the model's output when a feature is observed vs. unknown. Some feature values have a large impact on the prediction, while others have a small impact. The Shapley values $\phi_i(f, x)$, explaining a prediction $f(x)$, are an allocation of credit among the various features in x (such as age, weight, time series features, etc.), and are the only such allocation that obeys a set of desirable properties. Note that $\phi_i(f, x)$ is a single numerical value representing the impact of feature i , on the prediction of the model f when given the input x . For Prescience f is a gradient boosting model, and x is the set of all input features from a time point. We provide a brief summary of these properties below, and refer the reader to Lundberg et al. [101] for a full discussion, and for connections with several other recent methods in complex model interpretability. In the properties below $f_x(S) = E[f(x)|x_S]$, where x_S is a subset of the input vector with only the features in the set S present.

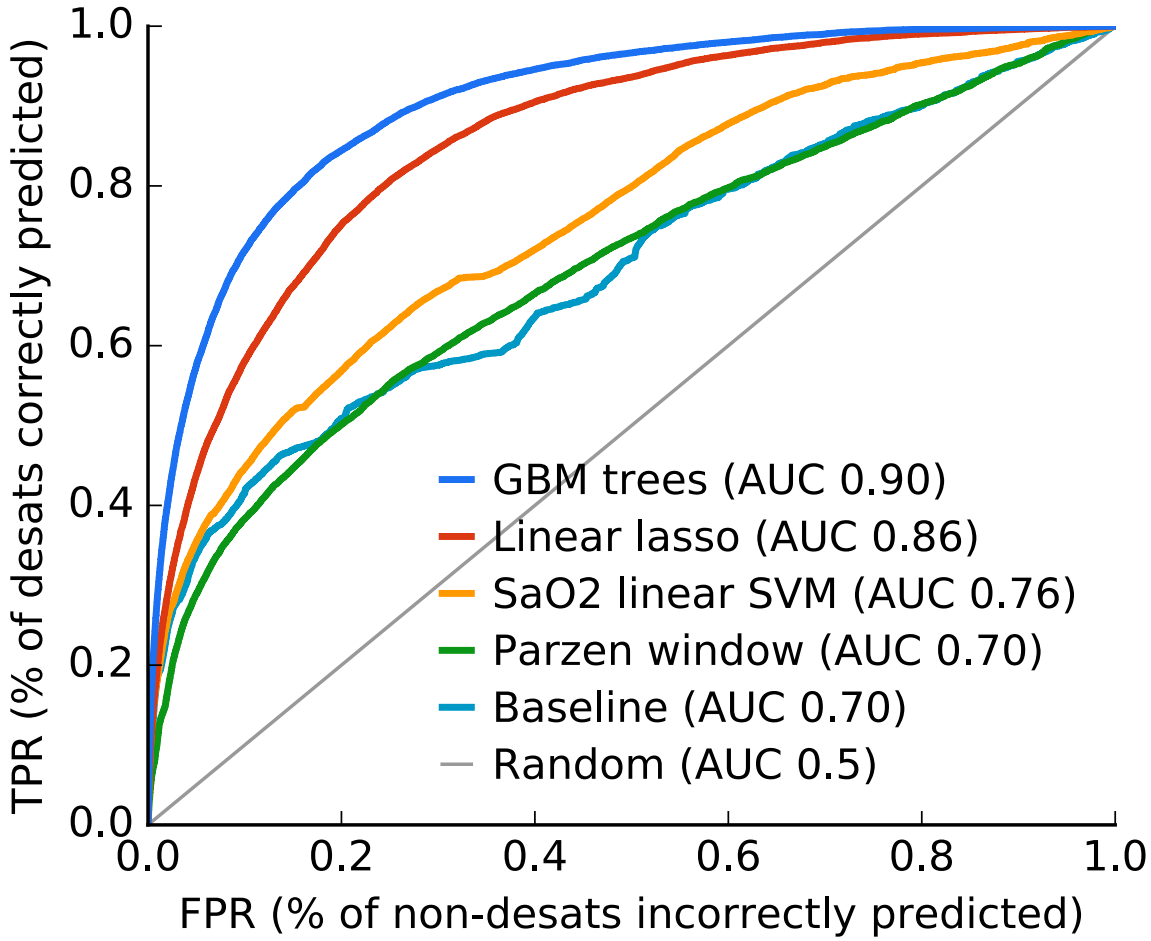
Local Accuracy.

$$f(x) = \phi_0(f, x) + \sum_{i=1}^{M} \phi_i(f, x),$$

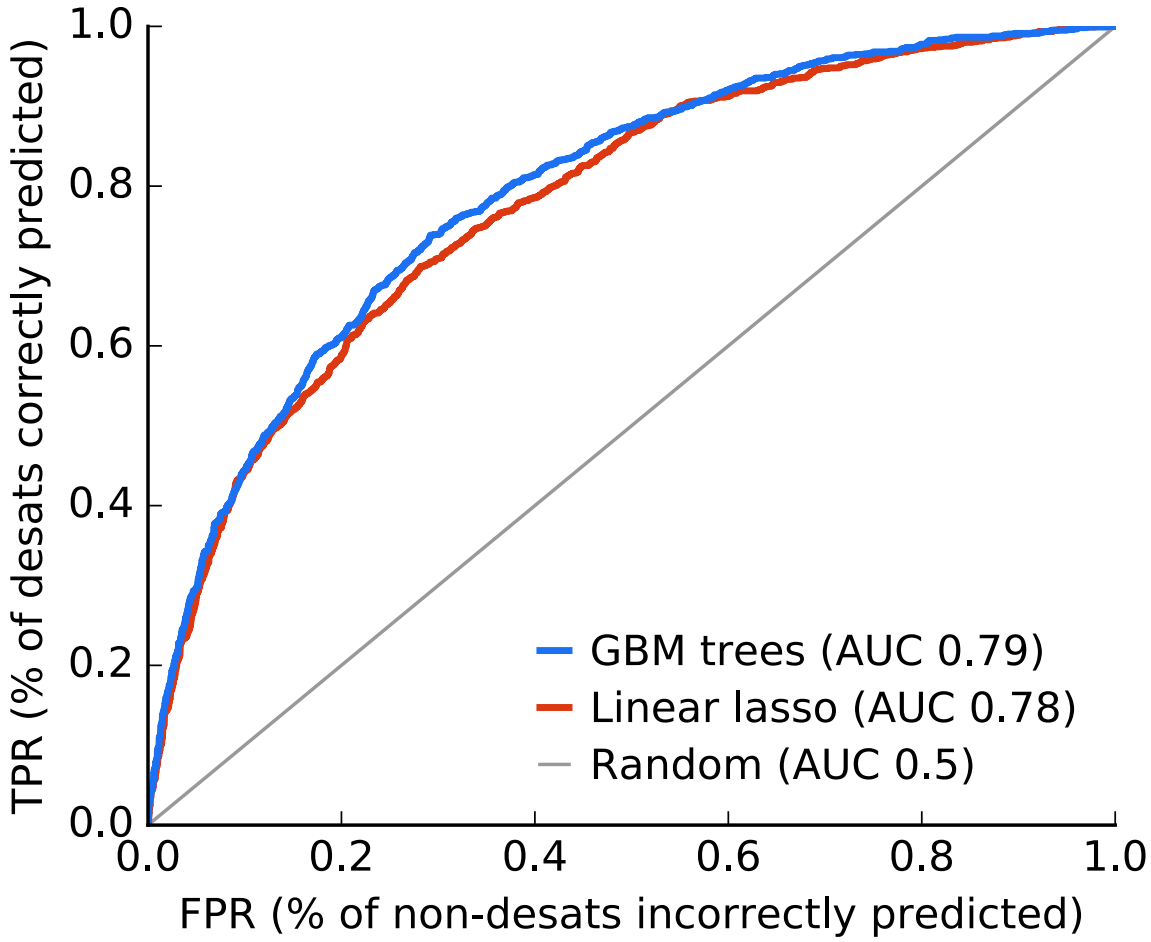
where $\phi_0(f, x) = E[f(x)]$

(the expected value of the model over the training data set), and M is the number of 'interpretable' inputs, which each correspond to a group of original input features (such as those shown in Figure S4). The local accuracy assumption forces the attribution values to correctly capture the difference between the expected model output and the output for the current prediction. For Prescience the input feature groups are the sets of extracted features associated with each time series. For instance, the 6 second, 1 minute, and 5 minute moving average extracted features, and the 5-minute moving variance extracted feature from the SpO_2 time series are all considered as a single group. This manual grouping process is not strictly necessary, but can help improve the interpretation of partially redundant features.

Consistency. For any two models f and f' , if
$f'_x(S \cup \{i\}) - f'_x(S) \geq f_x(S) - f_x(S \cup \{i\})$,
for all $S \in Z \backslash \{i\}$ where Z is the set of all M input features, then $\phi_i(f', x) \geq \phi_i(f, x)$. This states that if a feature is more important in one model than another, no matter what other features are also present, then the importance attributed to that feature should be also be higher.



Supplementary Figure S9: **Real-time performance of gradient boosting machines vs a linear lasso model, a SVM based on ElMoaqet et al. [41], and an unsupervised Parzen window method used by Tarassenko et al. [171].** There are $\sim$ 8 million training samples and the increased flexibility of gradient boosting trees clearly outperforms the more restrictive linear models, with a 32% reduction in test error over linear Lasso (bootstrap confidence interval 31.5-32.9% ; P-value < 0.0001). The linear Lasso had 244 non-zero coefficients chosen using validation error.



Supplementary Figure S10: **Preoperative performance of gradient boosting machines vs a linear lasso model.** Given a much smaller preoperative dataset with $\sim$ 42,000 training examples the difference between the complex gradient boosting trees model and a linear model becomes smaller, but still meaningful, with a 6% reduction in test error over Linear Lasso (bootstrap confidence interval 4.4-7.5% ; P-value $<$ 0.0001). The linear Lasso had 423 non-zero coefficients chosen using validation error.

Only one allocation of credit satisfies these two properties and that allocation is the one given by the Shapley values [101].

Given a specific prediction $f(x)$ we can compute the Shapley values using a weighted sum that represents the impact of each feature being added to the model averaged over all possible orders of features being introduced:

$$\phi_i(f,x) = \sum_{S \subseteq S_{all} \setminus \{i\}} \frac{|S|!\,(M-|S|-1)!}{M!} \left[f_x(S \cup \{i\}) - f_x(S)\right]$$

$$= \sum_{S \subseteq S_{all} \setminus \{i\}} \frac{1}{(M choose |S|)(M-|S|)} \left[f_x(S \cup \{i\}) - f_x(S)\right] . \qquad \text{Eq (1)}$$

In practice, there are far too many terms to evaluate this sum completely, so we can instead approximate it by a sampling procedure [167, 101]. We have released an open implementation of this explanation approach which also includes additional upcoming improvements for tree models at: http://github.com/slundberg/shap

To compute the Shapley values of each prediction we need to estimate the predictions of the model when specific input features are missing (those not in the set S). Since the model was not trained to support missing values we approximate what the model would predict (if retrained on that subset of input features) by sampling from the training data set and replacing the missing features with the values they would have had in that sample. By averaging many such samples, we can estimate the expected value of $f_x(S)$ only using evaluations of $f_x(S_{all})$ where no features are missing.

The approach above requires nested sampling, once to estimate the Shapley value and then from each sample we again sample to estimate $f_x(S)$ and $f_x(S \cup \{i\})$. To reduce the number of samples in the inner step, we used k-medians to generate 20 medians of the entire dataset, and then performed a weighted evaluation for only these 20 summary inputs as an approximation for the entire dataset. This removes the need for nested sampling.

In Prescience we also used a non-linear link function h such that:

$h(f(x)) = \sum_{i=0}^{M} \phi_i(f,x)$.

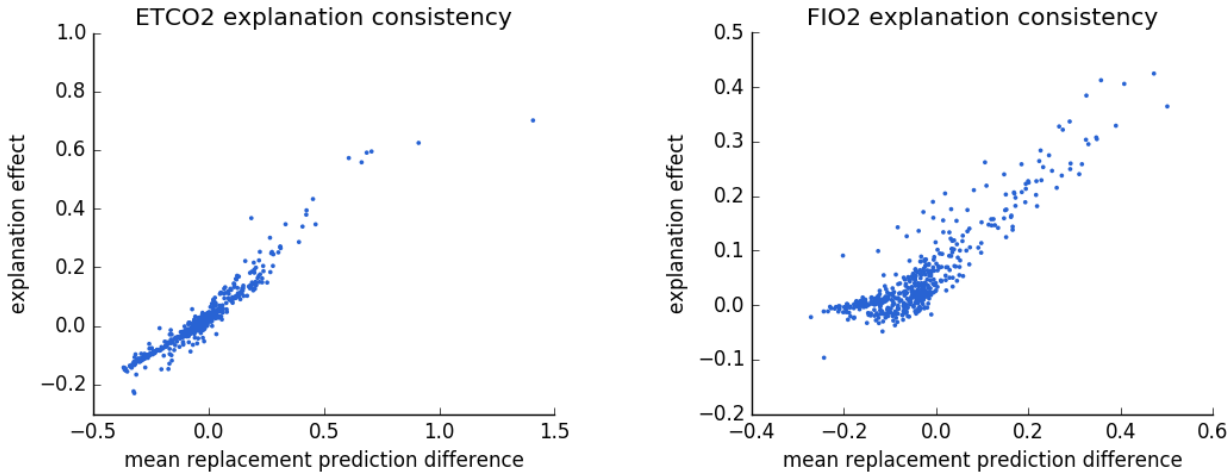
Since Prescience uses logistic regression the use of a $h = logit$ link function transforms the output space from probabilities to log odds. Assuming the importance of features is additive in the log odds space is much more natural than assuming they are additive in the space of probabilities (which must fall between 0 and 1). The same reasoning also drives the use of the logit link function during standard logistic regression.

We were able to get stable feature importance estimates for thousands of features in less than 5 seconds on our server (in large part because these inputs typically had less than 100 non-zero entries). We compared these theoretically grounded explanations with a simple estimate of feature importance to verify they showed reasonable consistency. The simple method we chose was to replace a single feature group with random values from other samples in the data set, and determine the average model output over different possible samplings. We then subtracted this mean value from the original model prediction to get a difference from a prediction with a typical value of that feature vs. the current value. This simple method is not very scalable and does not account for interactions with other features, yet is useful to compare with the Prescience explanations to ensure the Prescience estimates of feature effects are consistent with an intuition of how much a feature's change from its typical value effects the current risk of hypoxemia (Supplementary Figure S11).

4.4.7 Physician evaluation

The potential benefit Prescience provides to physicians was evaluated using previously recorded procedures. Both before a procedure begins, and at several time points during the operation all the available electronically recorded data were shown to the anesthesiologist and they were asked to predict if a desaturation (as defined above) will occur in the next 5 minutes (Supplementary Figures S12, S13, and S14). For half of the procedures anesthesiologists are given Prescience explained risks, and for the other half they are given the same data, but without any Prescience assistance. In both cases anesthesiologists are asked to provide a fold change in the risk that desaturation will occur.

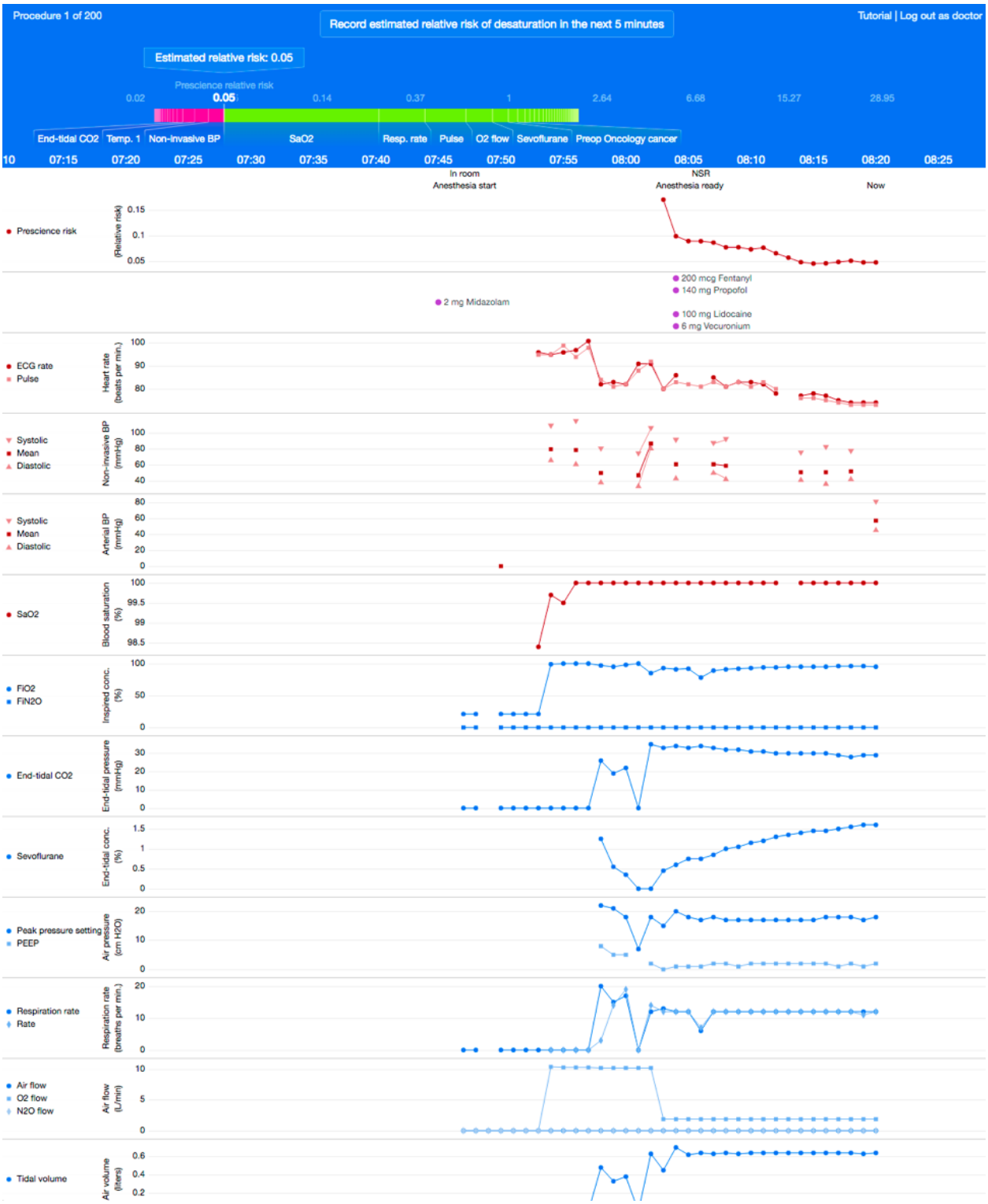
The test procedures were divided into two equal sized groups, replicate 1 and replicate 2. Anesthesiologists were also divided into two groups, A and B. Group A was given Prescience assistance on replicate 1 but not on replicate 2, while group B was given Prescience assistance on replicate 2 but not replicate 1. After



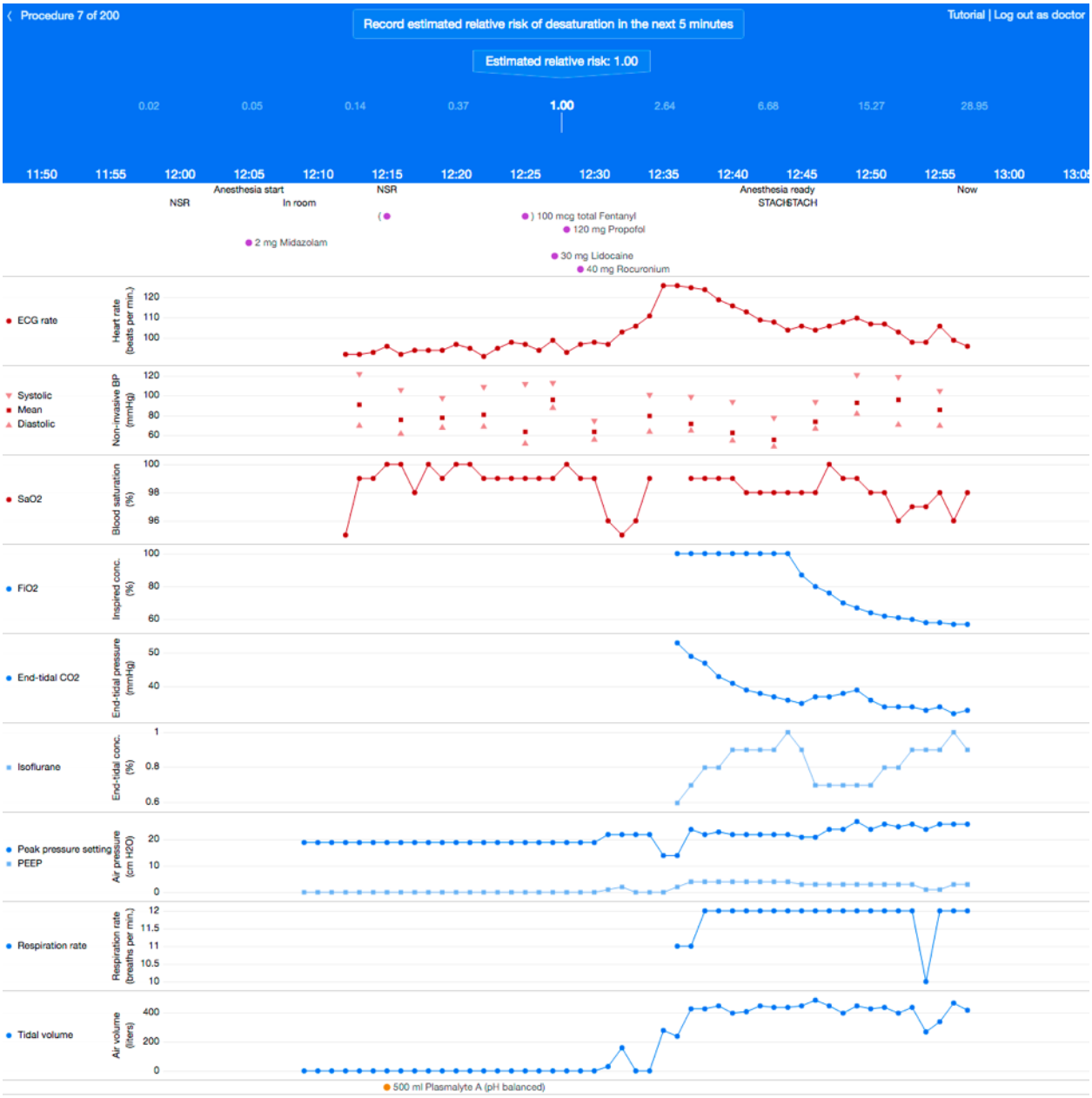
Supplementary Figure S11: **Consistency of Prescience explanation effects.** Comparing the Prescience explanation effects with the difference between the current model output and the outputs when a specific extracted feature is replaced with its typical value. The effects are from the cases shown to anesthesiologists during testing in Figure S3. The strong correlation ($R^2 = 0.92$ for $ETCO_2$ and $R^2 = 0.81$ for FIO_2) demonstrates the consistency of Prescience explanation effect sizes with the intuitive notion of the change in model out from replacing an extracted feature's value with a typical value. Note that both values are shown explaining the additive portion of the classification model (inside the logistic function).

randomly assigning anesthesiologists to groups, three anesthesiologists from group A completed the evaluation and two anesthesiologists from group B. We pooled the results within each group and between groups, and the results of this evaluation are shown in Figure S3. The order in which anesthesiologists were presented with cases was random across both replicate sets.

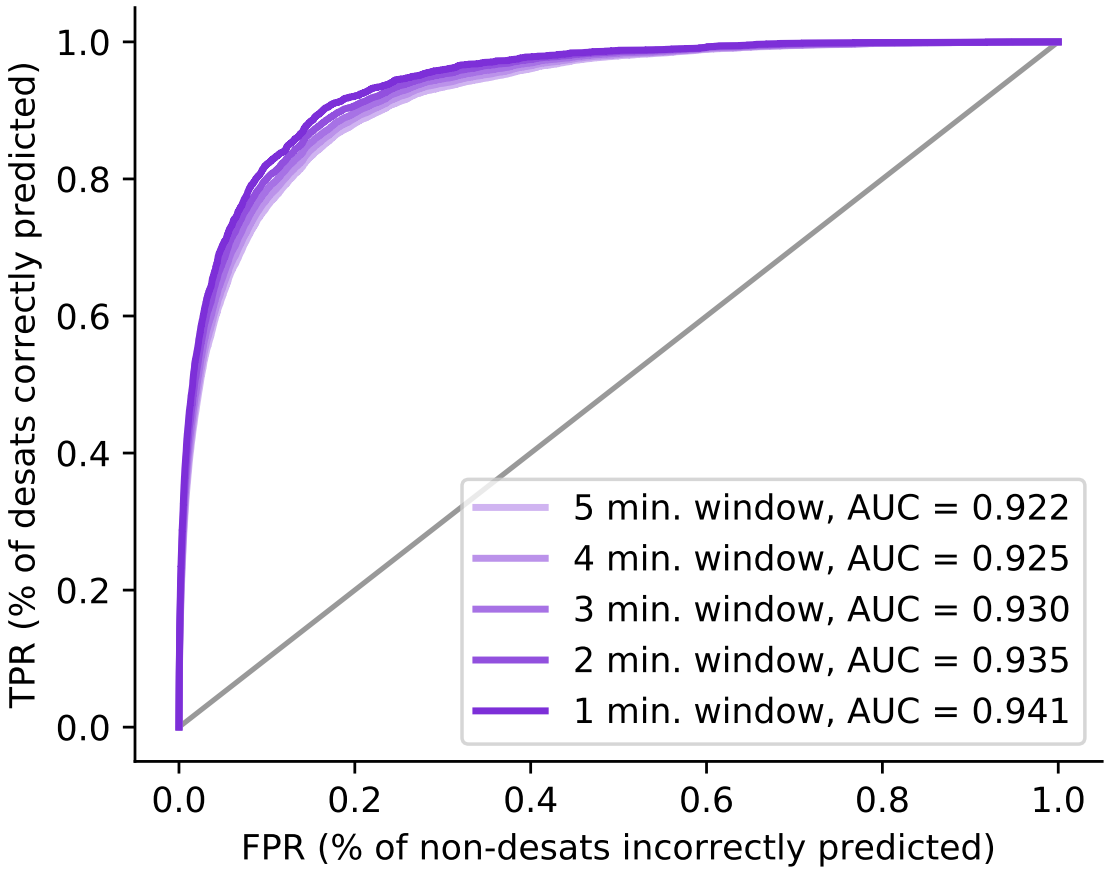
Supplementary Figure S12: **Sample of a physician's test interface for preoperative prediction of risk.** Prescience assistance is given for this preoperative prediction. Anesthesiologists choose an estimated relative risk by moving the given slider, then recording the score.



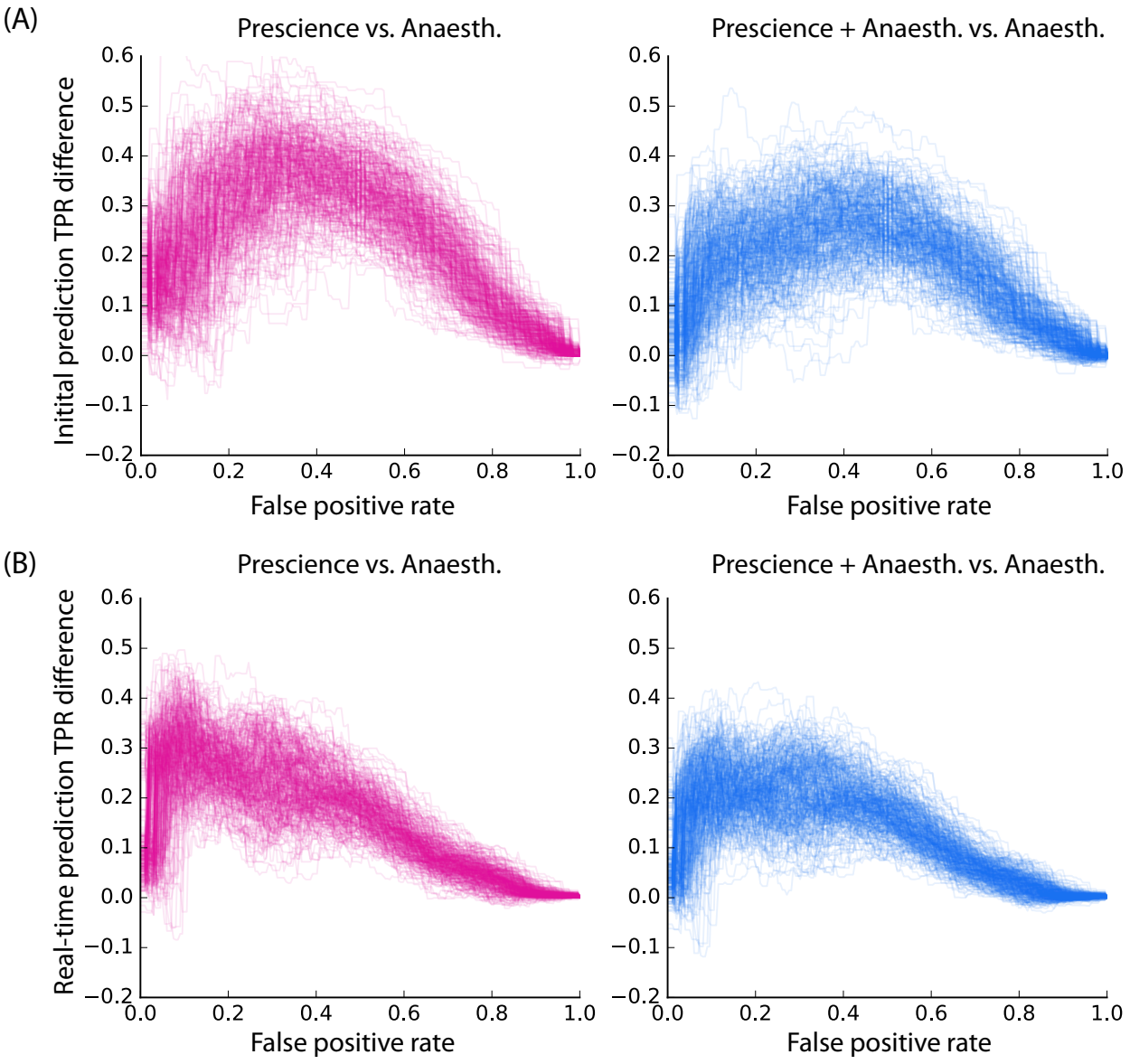
Supplementary Figure S13: **Sample of a physician's test interface for real-time prediction of risk.** Prescience assistance is given for this intraoperative prediction. Anesthesiologists choose an estimated relative risk by moving the given slider. The preoperative data are also shown lower down in the interface just as illustrated in Supplementary Figure S12.



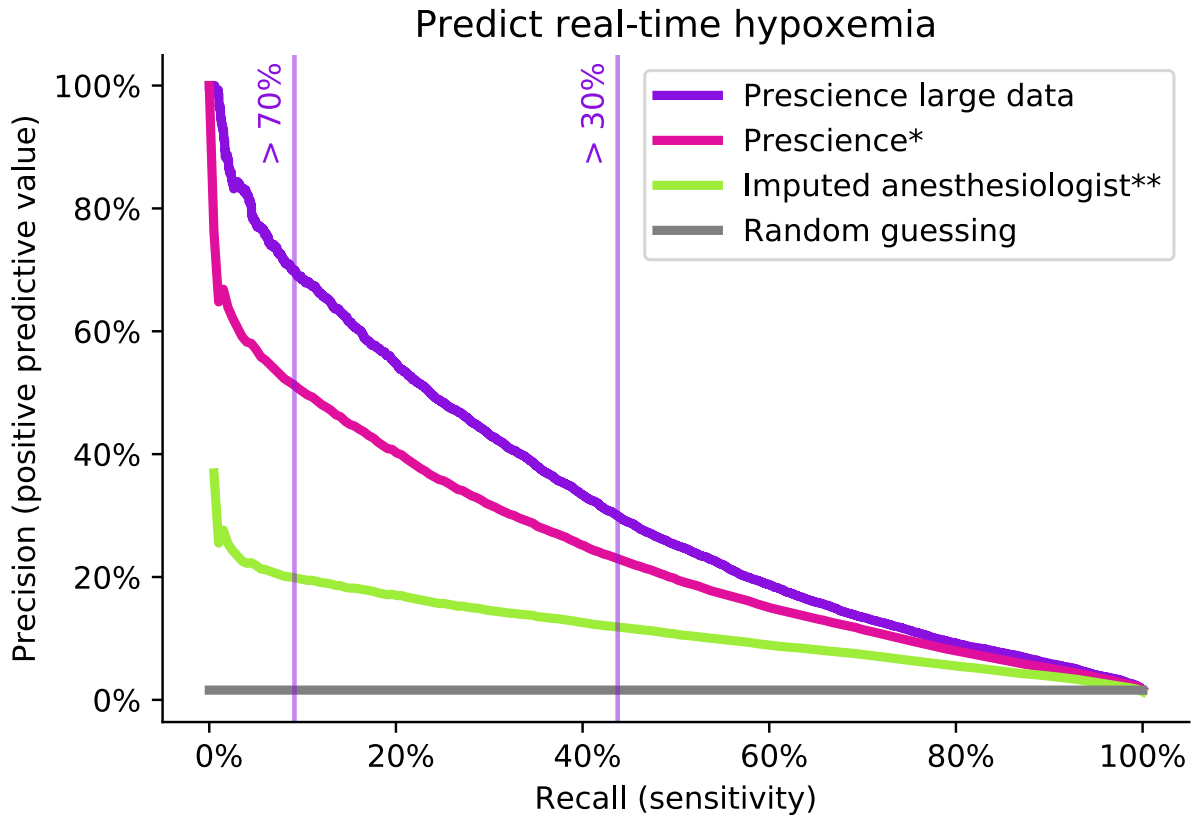
Supplementary Figure S14: **Sample of a physician's test interface for real-time prediction of risk.** This is similar to Supplementary Figure S13 but this time the anesthesiologist must choose a risk on their own using only the original data without help from Prescience.



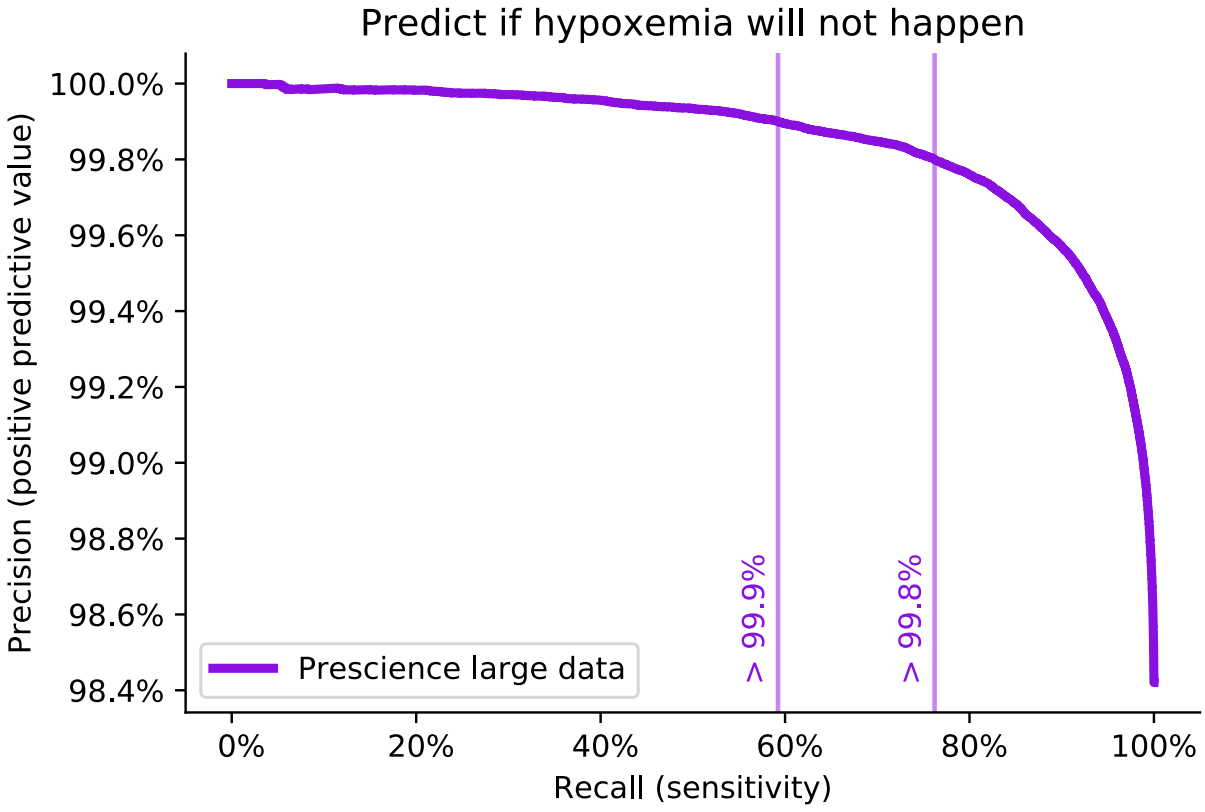
Supplementary Figure S15: **Prediction performance increases as the window size is reduced.** To evaluate the impact of window size on the prediction performance of Prescience we restricted the window size and measured the effect on the area under the receiver operating characteristic (ROC) curve. A steady increase in accuracy is observed as the window size shrinks, matching the intuition that it is easier to predict near-term events than more distant events.



Supplementary Figure S16: **Bootstrap analysis of the increase in the true positive rate (TPR) between anesthesiologists and Prescience.** Three hundred bootstrap replicates were run and the difference between the Prescience ROC curve and the anesthesiologists' ROC curve was computed for each bootstrap sample. This was repeated for the difference between the anesthesiologist + Prescience ROC curve and the anesthesiologists' ROC curve. Plotting these differences shows the variability of Prescience's true positive rate improvements across all false positive rates. (A) Initial prediction performance. (B) Real-time prediction performance.



Supplementary Figure S17: **Precision-recall curves for real-time hypoxemia prediction.** To evaluate possible hypoxemia warning alarm thresholds and also to evaluate the benefit of including additional data to train the model, we compare Prescience with Prescience trained with an expanded data set of 175k procedures (2-3 times the original dataset). The much larger dataset lead to a noticeable improvement in performance, and a significant fraction of hypoxemia minutes (9%) can be predicted with high precision (70%) and hence few false alarms (false alarm rate = 1 – precision). $*$ Note that 4% of test procedures in the original test set were not matched in the new larger dataset, hence some variability in the comparison of Prescience to Prescience large data should be expected. $**$ Since we didn't test doctors on all of the large number of test hypoxemia time points, we imputed the performance of anesthesiologists by assuming that the proportionality between the likelihood ratio of Prescience and the likelihood ratio of anesthesiologists remained the same between the doctor test set used in Figure S3 and the full test dataset (the likelihood ratio was computed from the sensitivity and specificity of the methods).



Supplementary Figure S18: **Precision-recall curve for predicting in real-time when hypoxemia will not happen.** Anesthesiologists must determine both when to intervene and when to not intervene for a predicted event. Because hypoxemia is rare and Prescience has strong predictive power, over half of all time points can be classified as not having hypoxemia within 5 minutes at a precision of 99.9% in the test data set.

Initial predictions					
	Anesthesiolog 1	Anesthesiolo 2	Anesthesiologi 3	Anesthesiologist 4	Doctor composite
1	surgical procedure	BMI	BMI (and by default height and weight)	Baseline saturation (patient on home O_2/room air)	**BMI**
2	BMI	Weight	History of OSA	Pulmonary disease (Fibrosis, asthma, COPD - Chronic Obstructive Pulmonary Disease)	**Lung disease**
3	age	Smoker	Type of surgery	Obesity or high BMI (Body mass index)	**ASA code**
4	pre-existing heart or lung disease	Presence of lung disease	History of Asthma	Certain planned procedures (Thoracic surgery)	**Asthma**
5	ASA status	History of difficult airway	Functional Status	Anesthesia method (sedation)	**Age**
6				Children	**Anesthesia type**
7				Pregnancy	**CPT code**
8				High metabolic status (sepsis)	

Supplementary Table S2: **Physician's responses listing the most important factors when assessing the risk for hypoxemia at any point during an upcoming procedure.** Four anesthesiologists were asked to list the most important factors to consider for hypoxemia. The responses of all 4 anesthesiologists were then compiled into a single composite list. For a comparison with features chosen by Prescience, see Figure S5 in the main text.

Real-time predictions					
	Anesthesiolo 1	Anesthesiologist 2	Anesthesiologis 3	Anesthesiologist 4	Anesthesiologist composite
1	surgical procedure	Trend of SPO_2 if downward trend more likely	Tidal Volume	Trends in SpO_2	**SpO_2**
2	BMI	Tidal volume - if trending down suggests a problem.	Respiratory Rate	Surgical interventions (1-lung ventilation, insufflation of the abdomen, pushing the lung)	**Tidal Volume (TV)**
3	FiO_2	Increasing Peak inspiratory pressure	SpO_2	High airway pressure	**Peak Inspiratory Pressure (PIP)**
4	minute ventilation/tidal volume	Increasing Co_2	End Tidal CO_2	Surgical and Anesthesia events (bronchospasm, pneumothorax, pulmonary embolism, mucus plug, airway trapping, fighting against vent etc.)	**End tidal CO_2**
5	recent anesthetic drug administration	Increasing heart rate	Peak Inspiratory Pressure (PIP)	Patient position (head down)	**BMI**
6			Tachycardia	Anesthesia machine malfunction (tube kinking)	**Resp rate**
7			BMI (and by default height and weight)		**FiO2**
8			History of OSA		**Heart rate**
9			Type of surgery		
10			History of Asthma		

Supplementary Table S3: **Physician's responses listing the most important factors when assessing the risk for hypoxemia in the next 5 minutes.** Four anesthesiologists were asked to list the most important factors to consider for hypoxemia. The responses of all 4 anesthesiologists were then compiled into a single composite list. For a comparison with features chosen by Prescience, see Figure S5 in the main text.

4.4.8 Supplementary Materials

Funding

This work was supported by a National Science Foundation (NSF) DBI-135589, NSF Graduate Research Fellowship, and a UW eScience/ITHS seed grant *Machine Learning in Operating Rooms (06-1019).*

4.4.9 Acknowledgements

The results presented in this chapter have been released as a paper in collaboration with my co-authors [105]: Bala Nair, Monica S. Vavilala, Mayumi Horibe, Michael J. Eisses, Trevor Adams, David E. Liston, Daniel King-Wai Low, Shu-Fang Newman, Jerry Kim, and Su-In Lee.

We thank G. Erion, M. T. Ribeiro, J. Schreiber and members of the Lee laboratory for feedback and suggestions that improved the manuscript and experiments. This work was supported by National Science Foundation grant nos. DBI-135589 and DBI-1552309, National Institutes of Health grant no. 1R35GM128638, NSF Graduate Research Fellowship grant no. DGE-1256082 and a UW eScience/ITHS seed grant Machine Learning in Operating Rooms.

Chapter 5

Explainable AI for Trees: From Local Explanations to Global Understanding

Tree-based machine learning models such as random forests, decision trees, and gradient boosted trees are the most popular non-linear predictive models used in practice today, yet comparatively little attention has been paid to explaining their predictions. Here we significantly improve the interpretability of tree-based models through three main contributions: 1) The first polynomial time algorithm to compute optimal explanations based on game theory. 2) A new type of explanation that directly measures local feature interaction effects. 3) A new set of tools for understanding global model structure based on combining many local explanations of each prediction. We apply these tools to three medical machine learning problems and show how combining many high-quality local explanations allows us to represent global structure while retaining local faithfulness to the original model. These tools enable us to i) identify high magnitude but low frequency non-linear mortality risk factors in the general US population, ii) highlight distinct population sub-groups with shared risk characteristics, iii) identify non-linear interaction effects among risk factors for chronic kidney disease, and iv) monitor a machine learning model deployed in a hospital by identifying which features are degrading the model's performance over time. Given the popularity of tree-based machine learning models, these improvements to their interpretability have implications across a broad set of domains.

5.1 Introduction

Machine learning models based on trees are the most popular non-linear models in use today [72, 47]. Random forests, gradient boosted trees, and other tree-based models are used in finance, medicine, biology, customer retention, advertising, supply chain management, manufacturing, public health, and many other areas to make predictions based on sets of input features (Figure S1A left). In these applications it is often important to have models that are *both* accurate and interpretable, where being interpretable means that we can understand how the model uses the input features to make predictions [101]. Yet while there is a rich history of *global* interpretation methods for trees that summarize the impact of input features on the model as a whole, much less attention has been paid to *local* explanations that explain the impact of input features on individual predictions (i.e. for a single sample) (Figure S1A).

There are three ways we are aware of to explain individual predictions from trees (i.e. local explanation methods): 1) reporting the decision path; 2) an unpublished heuristic approach that assigns credit to each input feature [146]; and 3) a variety of model-agnostic approaches that require executing the model many times for each explanation [141, 32, 167, 101, 7]. These methods have the following limitations: 1) Simply reporting the decision path of a prediction is unhelpful for most models, particularly those based on multiple trees. 2) The behavior of the heuristic credit allocation approach has not yet been carefully analyzed, and as we show in Section 5.2.3, it is strongly biased to alter the impact of features based on their tree depth. 3) Since model-agnostic methods rely on post-hoc modeling of an arbitrary function, they can be slow and suffer from sampling variability (Section 5.2.4).

Here we propose TreeExplainer, a new local explanation method for trees that enables the tractable

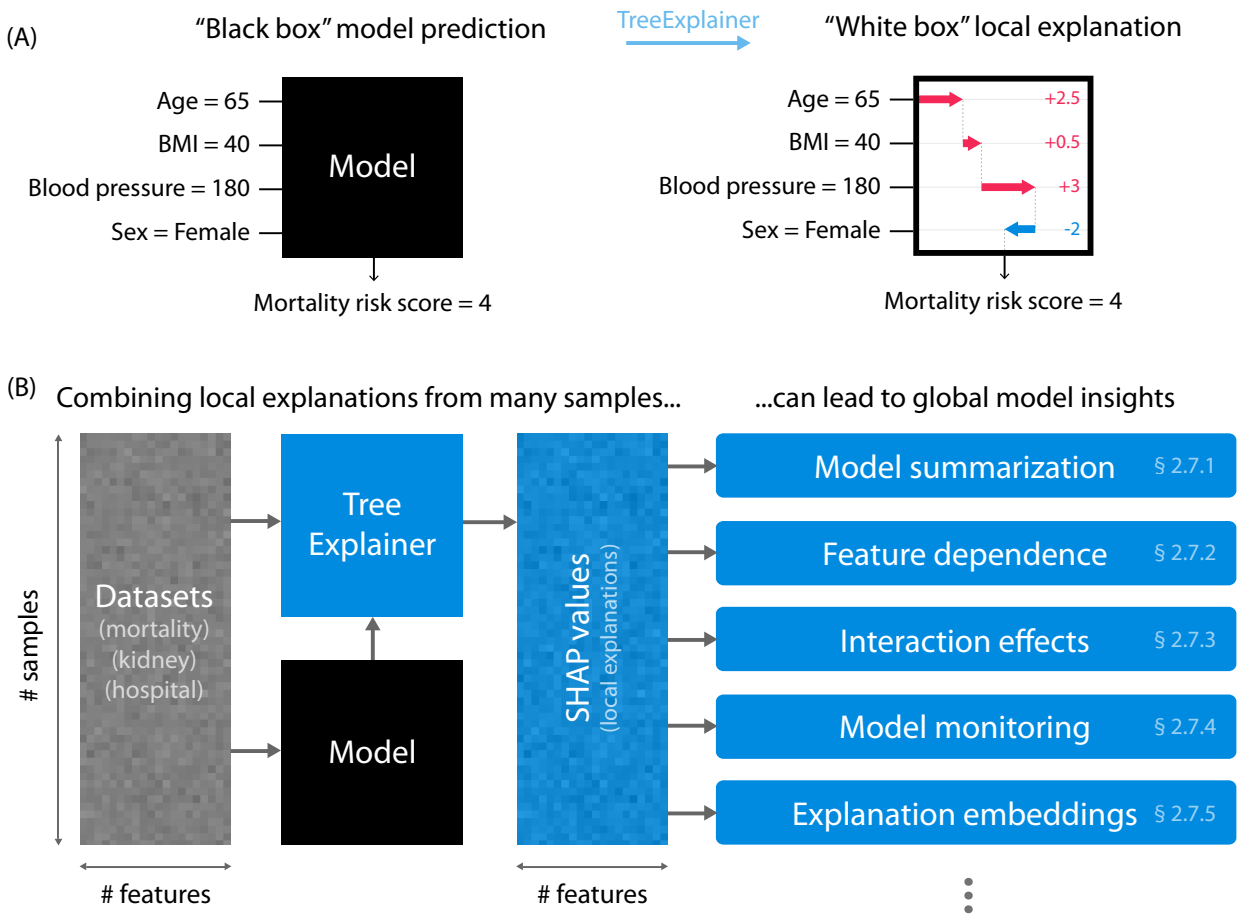


Figure S1: **Local explanations based on TreeExplainer enable a wide variety of new ways to understand global model structure.** (A) A local explanation based on assigning a numeric measure of credit to each input feature (Section 5.2.5). (B) By combining many local explanations we can represent global structure while retaining local faithfulness to the original model. To demonstrate this we use three illustrative medical datasets to train gradient boosted decision trees and then compute local explanations based on SHapley Additive exPlanation (SHAP) values (Section 5.2.5). Computing local explanations across all samples in a dataset enables many new tools for understanding global model structure (Section 5.2.7).

computation of *optimal* local explanations, as defined by desirable properties from game theory (Section 5.2.5). It bridges theory to practice by building on our previous model-agnostic work based on the classic game-theoretic Shapley values [101, 151] and leads to three notable improvements:

1. *TreeExplainer enables the exact computation of optimal local explanations for tree-based models.* The classic Shapley values can be considered "optimal" in the sense that within a large class of approaches they are the only way to measure feature importance while maintaining several natural properties from cooperative game theory [101]. Unfortunately, in general these values can only be approximated since computing them exactly is NP-hard [111], requiring a summation over all feature subsets. Sampling based approximations have been proposed [167, 101], but using these methods to compute low variance versions of the results in this paper for even our smallest dataset would take years of CPU time (Section 5.2.4). However, by focusing specifically on trees we were able to develop an algorithm that computes local explanations based on the exact Shapley values in polynomial time. This enables us to provide local explanations that come with *theoretical guarantees* of *local accuracy* and *consistency* [101] (defined in Section 5.2.5; Methods 5.4.9).

2. *TreeExplainer extends local explanations to directly capture feature interactions.* Local explanations

that assign a single number to each input feature are very intuitive, but they cannot directly represent *interaction* effects. We provide a theoretically grounded way of measuring local interaction effects based on a generalization of Shapley values proposed in game theory literature [50]. We show that this can provide valuable insights into a model's behavior (Section 5.2.7).

3. *TreeExplainer provides a new set of tools for understanding global model structure based on many local explanations.* The ability to efficiently and exactly compute local explanations using Shapley values across an entire dataset enables a whole range of new tools to understand the global behavior of the model (Figure S1B; Section 5.2.7). We show that combining many local explanations allows us to represent global structure while retaining *local faithfulness* [140] to the original model, which produces more detailed and accurate representations of model behavior.

The need to explain predictions from tree models is widespread. It is particularly important in medical applications, where the patterns uncovered by a model are often even more important than the model's prediction performance [153, 105]. We use three medical datasets to demonstrate the value of TreeExplainer (Methods 5.4.2); they represent three types of loss functions (Methods 5.4.3): 1) *Mortality* – a mortality dataset with 14,407 individuals and 79 features based on the NHANES I Epidemiologic Followup Study [30], where we model the risk of death over twenty years of followup. 2) *Chronic kidney disease* – a kidney dataset that follows 3,939 chronic kidney disease patients from the Chronic Renal Insufficiency Cohort study over 10,745 visits with the goal of using 333 features to classify if patients will progress to end-stage renal disease within 4 years. 3) *Hospital procedure duration* – a hospital electronic medical record dataset with 147,000 procedures and 2,185 features, where we predict the duration of an upcoming procedure.

We discuss why tree models are the most appropriate models in many situations, both because of their accuracy (Section 5.2.1), and their interpretability (Section 5.2.2). We discuss the need for better local explanations of tree-based models (Sections 5.2.3-5.2.4), and how we address that need with TreeExplainer (Section 5.2.5). We then extend local explanations to capture interaction effects (Section 5.2.6). Finally, we demonstrate the value of the new explainable AI tools enabled by combining many local explanations from TreeExplainer (Section 5.2.7). To enable the wide use of TreeExplainer, high-performance implementations have also been released and integrated with many major tree-based machine learning packages (`https://github.com/suinleelab/treeexplainer-study`).

5.2 Results

5.2.1 Tree-based models can be more accurate than neural networks

Tree-based ensemble methods such as random forests and gradient boosted trees achieve state-of-the-art performance in many domains. They have a long history of use in machine learning [47], and new high-performance implementations are an active area of research [24, 75, 139, 133]. While deep learning models are more appropriate in fields like image recognition, speech recognition, and natural language processing, tree-based models consistently outperform standard deep models on tabular-style datasets where features are individually meaningful and do not have strong multi-scale temporal or spatial structures [24]. A balance of computational efficiency, ease of use, and high accuracy have made tree-based models the most popular non-linear model type; four out of the five most popular machine learning models used by data scientists involve trees [72]. The three medical datasets we examine here all represent tabular-style data, and gradient boosted trees outperform both deep learning and linear regression across all three datasets (Figure S2A) (Methods 5.4.3).

5.2.2 Tree-based models can be more interpretable than linear models

While it is well-known that the bias/variance trade-off in machine learning has implications for model accuracy, it is less appreciated that the trade-off also affects interpretability. While simple high-bias models (such as linear models) are often easy to understand, they are also more sensitive to model mismatch – where the true relationships in the data do not match the form of the model.

To illustrate why low-bias models can be more interpretable than high-bias models we compare gradient boosted trees with lasso regularized linear logistic regression using the mortality dataset. We simulated

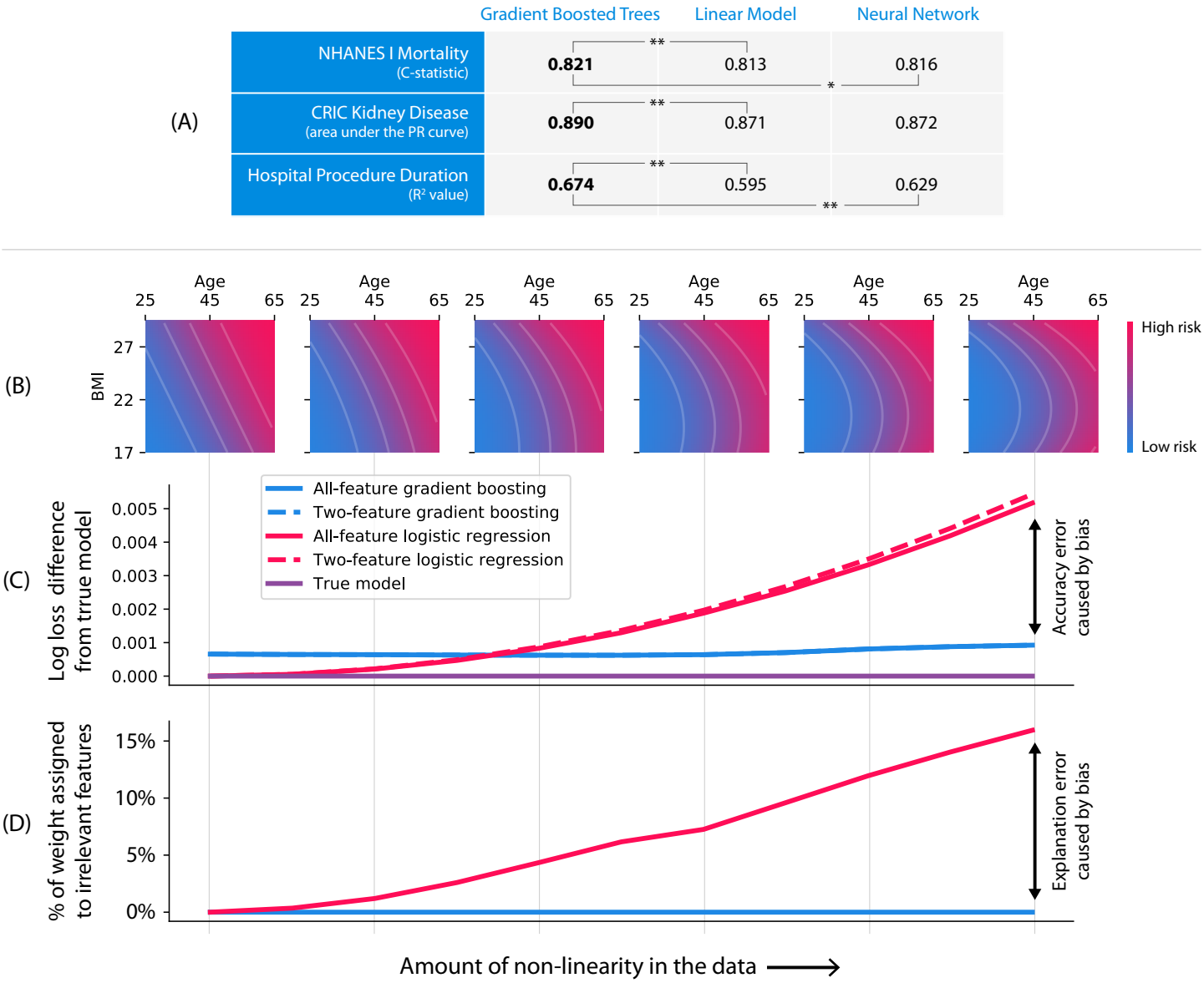


Figure S2: **Gradient boosted tree models can be both more accurate than neural networks and more interpretable than linear models.** (A) Gradient boosted tree models outperform both linear models and neural networks on all our medical datasets. (**) represents a P-value < 0.01, and (*) represents a P-value of 0.03 (Methods 5.4.3). (B-D) Linear models exhibit explanation error as well as accuracy error in the presence of non-linearity. (B) Data generating models used for the simulation, ranging from linear to quadratic along the body mass index (BMI) dimension. (C) The test performance of linear logistic regression (red) is better than gradient boosting (blue) up until a specific amount of non-linearity. Not surprisingly, the bias of the linear model is higher than the gradient boosting model as shown by the steeper slope as we increase the non-linearity. (D) As the true function becomes more non-linear the linear model assigns more credit (coefficient weight) to features that were not used by the data generating model.

a binary outcome based on a participant's age and body mass index (BMI) (Methods 5.4.4), and varied the amount of non-linearity in the simulated relationship (Figure S2B). As expected, when we increase the non-linearity, the bias of the linear model causes a drop in accuracy (Figure S2C). However, what is perhaps unexpected is that it also causes a drop in interpretability (Figure S2D). We know that the model should only depend on age and BMI, but even a moderate amount of non-linearity in the true relationship causes the linear model to start using other irrelevant features (Figure S2D). This means that even when a linear model can achieve the same test accuracy as a gradient boosted tree model, the gradient boosted tree model is preferable because its connection to the training data is more interpretable (Figure S2D; Methods 5.4.4).

5.2.3 Current local explanations for tree-based models are inconsistent

Despite the long history of approaches designed to compute global measures of feature importance in ensemble tree models (Methods 5.4.5), to our knowledge, there are only two approaches to quantify a feature's *local* importance for an individual prediction. The first is simply reporting the decision path, which is unhelpful for ensembles of many trees. The second is an unpublished heuristic approach (proposed by *Saabas*) that explains a prediction by following the decision path and attributing changes in the expected output of the model to each feature along the path (Methods 5.4.6). The Saabas method has not been well studied, and we demonstrate here that it is strongly biased to alter the impact of features based on their distance from the root of a tree (Supplementary Figure S7A). This causes Saabas values to be *inconsistent*, which means we can modify a model to make a feature clearly more important, and yet the Saabas value attributed to that feature will decrease (Supplementary Figure S8). The difference this makes can be seen by examining trees representing multi-way AND functions. No feature in an AND function should have any more credit than another, yet Saabas values give splits near the root much less credit than splits near the leaves (Supplementary Figure S7A). Consistency is critical for an explanation method because it makes comparisons among feature importance values meaningful.

5.2.4 Model-agnostic local explanations are slow and variable

Model-agnostic local explanation approaches can be used to explain tree models (Methods 5.4.7), but they rely on post-hoc modeling of an arbitrary function and thus can be slow and/or suffer from sampling variability when applied to models with many input features. To illustrate this we generated random datasets of increasing size and then explained (over)fit XGBoost models with 1,000 trees. This experiment shows a linear increase in complexity as the number of features increases; model-agnostic methods take a significant amount of time to run over these datasets, even though we allowed for non-trivial estimate variability (Supplementary Figure S7D; Methods 5.4.8) and only used a moderate numbers of features (Supplementary Figure S7C). Calculating low variance estimates of the Shapley values for the results in this paper would be intractable; just the chronic kidney disease dataset experiments would have taken almost 2 CPU days for basic explanations, and over 3 CPU years for interaction values (Section 5.2.6) (Supplementary Figure S7E-F; Methods 5.4.8). While often practical for individual explanations, model-agnostic methods can quickly become impractical for explaining entire datasets (Supplementary Figure S7C-F).

5.2.5 TreeExplainer provides fast local explanations with guaranteed consistency

Here we introduce a new local feature attribution method for trees, TreeExplainer, which can exactly compute the classic Shapley values from game theory [151]. TreeExplainer bridges theory to practice by reducing the complexity of exact Shapley value computation from exponential to polynomial time. This is important since within the class of *additive feature attribution methods*, a class that we have shown contains many previous approaches to local feature attribution [101], results from game theory imply the Shapley values are the only way to satisfy three important properties: *local accuracy*, *consistency*, and *missingness* (Methods 5.4.9). Local accuracy states that when approximating the original model f for a specific input x, the explanation's attribution values should sum up to the output $f(x)$. Consistency states that if a model changes so that some feature's contribution increases or stays the same regardless of the other inputs, that input's attribution should not decrease. Missingness is a trivial property satisfied by all previous explanation methods (Methods 5.4.9).

Shapley values, as applied here to feature importance, are defined as the sequential impact on the model's output of observing each input feature's value, averaged over all possible feature orderings (Supplementary Figure S9; Equation 5.5 in Methods). This means for each of *all possible* orderings, we introduce features one at a time into a conditional expectation of the model's output, then attribute the change in expectation to the feature that was introduced. Since Shapley values can be computed using any set function, not just conditional expectations, we will use the more specific term, *SHapley Additive exPlanation (SHAP) values* [101], to clarify that we are using conditional expectations to measure the impact of a set of features on the model.

Perhaps surprisingly, the independently developed Saabas values are computed the same way as SHAP values, but rather than averaging over all feature orderings, Saabas values only consider the single ordering defined by a tree's decision path. This connection leads to two new insights: 1) The bias and consistency problems of Saabas values (Section 5.2.3) result from a failure to average feature impacts over all orderings. 2) For an infinite ensemble of random fully-developed trees on binary features, Saabas values effectively consider all orderings and so converge to the SHAP values. In practice, however, tree ensemble models are not infinite, random, or fully developed; to guarantee consistency we need to compute SHAP values exactly. TreeExplainer makes this possible by *exactly* computing SHAP values in *low order polynomial time*. This represents an exponential complexity improvement over previous exact Shapley methods. By default, TreeExplainer computes conditional expectations using tree traversal, but it also provides an option that enforces feature independence and supports explaining a model's loss function (Methods 5.4.10).

Efficiently and exactly computing the Shapley values guarantees that explanations will always be consistent and locally accurate. This results in several improvements over previous local explanation methods:

- *TreeExplainer impartially assigns credit to input features regardless of their depth in the tree.* In contrast to Saabas values, TreeExplainer allocates credit uniformly among all features participating in multi-way AND operations (Supplementary Figures S7A-B) and avoids inconsistency problems (Supplementary Figure S8).
- *For moderate sized models, TreeExplainer is several orders of magnitude faster than model-agnostic alternatives, and has zero estimation variability* (Supplementary Figures S7C-F). Since solutions from model-agnostic sampling methods are approximate, there is always the additional burden of checking their convergence and accepting a certain amount of noise in their estimates. This burden is eliminated by TreeExplainer's exact explanations.
- *TreeExplainer consistently outperforms alternative methods across a benchmark of 21 different local explanation metrics* (Figure S3; Supplementary Figures S10-S11). We designed 21 metrics to comprehensively evaluate the performance of local explanation methods, and applied these metrics to eight different explanation methods across three different model types and three datasets (Methods 5.4.11). The results for the chronic kidney disease dataset are shown in Figure S3, and demonstrate consistent performance improvement for TreeExplainer.
- *TreeExplainer matches human intuition across a benchmark of 12 user study scenarios* (Supplementary Figure S12). We evaluated how well explanation methods match human intuition by comparing their outputs with human consensus explanations of 12 scenarios based on simple models. In contrast to the heuristic Saabas values, Shapley value based explanation methods agree with human intuition in all the scenarios we tested (Methods 5.4.12).

TreeExplainer simultaneously addresses the consistency issues faced by the heuristic Saabas values (Section 5.2.3), and the computational issues faced by model-agnostic methods (Section 5.2.4). This leads to fast practical explanations with strong theoretical guarantees that result in improved performance across many quantitative metrics.

5.2.6 TreeExplainer extends local explanations to measure interaction effects

Traditionally, local explanations based on feature attribution assign a single number to each input feature. The simplicity of this natural representation comes at the cost of combining main and interaction effects. While interaction effects between features can be reflected in the global patterns of many local explanations, their distinction from main effects is lost in each local explanation (Section 5.2.7; Figure S4B-G).

Figure S3: **Explanation method performance across 21 different evaluation metrics and three classification models in the chronic kidney disease dataset.** Each tile represents the performance of a local explanation method on a given metric for a given model. Within each model the columns of tiles are scaled between the minimum and maximum value, and methods are sorted by their overall performance. TreeExplainer outperforms previous approaches not only by having theoretical guarantees of consistency, but also across a large set of other metrics (Methods 5.4.11). When these experiments were repeated in two synthetic datasets, TreeExplainer remained the top performing method (Supplementary Figures S10 and S11). Note that, as expected, Saabas becomes a better approximation to the Shapley values (and so a better attribution method) as the number of trees increases (Methods 5.4.9).

Here we propose *SHAP interaction values* as a new richer type of local explanation (Methods 5.4.13). These values use the 'Shapley interaction index,' a relatively recent concept from game theory, to capture local interaction effects. They follow from generalizations of the original Shapley value properties [50] and allocate credit not just among each player of a game, but among all pairs of players. The SHAP interaction values consist of a matrix of feature attributions (the main effects on the diagonal and the interaction effects on the off-diagonal) and have uniqueness guarantees similar to SHAP values [50]. By enabling the separate consideration of main and interaction effects for individual model predictions, TreeExplainer can uncover important patterns that might otherwise be missed (Section 5.2.7).

5.2.7 Local explanations from TreeExplainer can be used as building blocks for global understanding

We present five new methods that combine many local explanations to provide global insight into a model's behavior. This allows us to retain local faithfulness to the model while still capturing global patterns, resulting in richer, more accurate representations of the model's behavior. Each application presented below illustrates how local explanations can be used as building blocks for explainable machine learning. For all experiments we use gradient boosted trees since they have high accuracy (Figure S2A), low bias (Figure S2B-D), and support fast exact local explanations through TreeExplainer (Sections 5.2.5 and 5.2.6).

Local model summarization reveals rare high-magnitude effects on mortality risk and increases feature selection power

Combining local explanations from TreeExplainer across an entire dataset enhances traditional global representations of feature importance by: 1) avoiding the inconsistency problems of current methods (Supplementary Figure S8), 2) increasing the power to detect true feature dependencies in a dataset (Supplementary Figure S13), and 3) enabling us to build *SHAP summary plots* that succinctly display the magnitude, prevalence, and direction of a feature's effect. SHAP summary plots avoid conflating the magnitude and prevalence of an effect into a single number, and so reveal rare high magnitude effects. Figure S4A illustrates its benefits using the mortality dataset: (left) a standard bar-chart based on the average magnitude of the SHAP values, and (right) a set of beeswarm plots where each dot corresponds to an individual person in the study. The position of the dot on the x-axis is the impact that feature has on the model's prediction for that person. When multiple dots land at the same x position they pile up to show density.

Figure S4A (right) reveals the direction of effects, such as men (blue) having a higher mortality risk than women (red); and the distribution of effect sizes, such as the long right tails of many medical test values. These long tails mean features with a low global importance can yet be extremely important for specific individuals. Interestingly, rare mortality effects always stretch to the right, which implies there are many ways to die abnormally early when medical measurements are out-of-range, but not many ways to live abnormally longer (Methods 5.4.14).

Local feature dependence reveals both global patterns and individual variability in mortality risk and chronic kidney disease

SHAP dependence plots show how a feature's value (x-axis) impacted the prediction (y-axis) of every sample (each dot) in a dataset (Figures S4B and E; Methods 5.4.15). This provides richer information than traditional partial dependence plots (Supplemental Figure S14). For the mortality model this reproduces the standard risk inflection point of systolic blood pressure [57], while also highlighting that the impact of blood pressure risk is different for people of different ages (Figure S4B). Many individuals have a recorded blood pressure of 180 mmHg in the mortality dataset, but the impact of that measurement on their mortality risk varies because early onset high blood pressure is more concerning to the model than late onset high blood pressure. These types of interaction effects show up as vertical dispersion in SHAP dependence plots.

For the chronic kidney disease model, a dependence plot again clearly reveals a risk inflection point for systolic blood pressure, but in this dataset the vertical dispersion from interaction effects appears to be partially driven by differences in blood urea nitrogen (Figure S4E). Correctly modeling blood pressure risk while retaining interpretabilty is important since blood pressure control in select chronic kidney disease

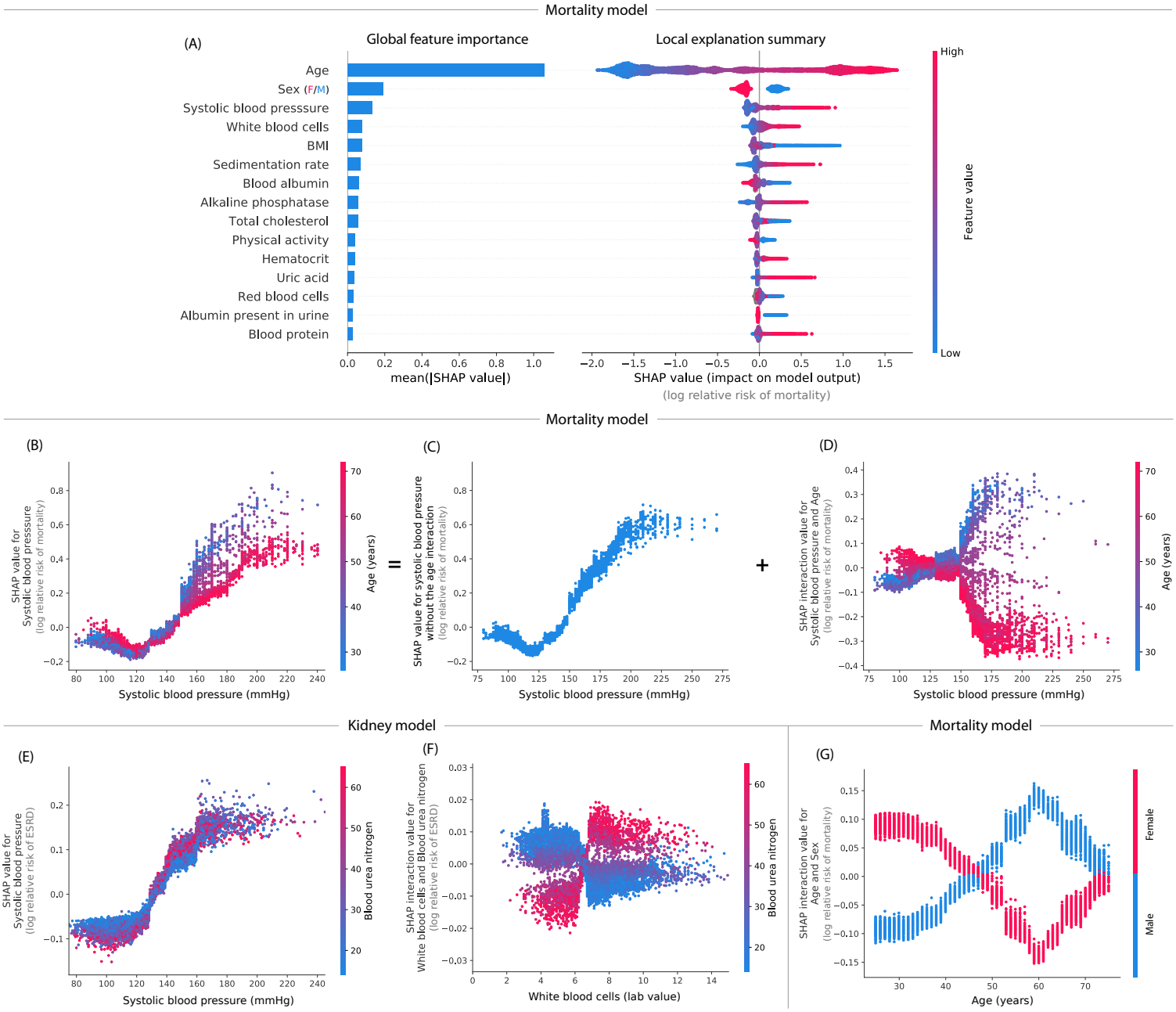


Figure S4: **By combining many local explanations we can provide rich summaries of both an entire model and individual features.** (A) Bar chart (left) and SHAP summary plot (right) for a gradient boosted decision tree model trained on the mortality dataset. The long right tails in the summary plot are from rare but high-magnitude risk factors. (B) SHAP dependence plot of systolic blood pressure vs. its SHAP value in the mortality model. A clear interaction effect with age is visible that increases the impact of early onset high blood pressure. (C) Using SHAP interaction values we can remove the interaction effect of age from the model. (D) Plotting just the interaction effect of systolic blood pressure with age shows how the effect of systolic blood pressure on mortality risk varies with age. Adding the y-values of C and D produces B. (E) A dependence plot of systolic blood pressure vs. its SHAP value in the kidney model shows an increase in kidney disease risk at a systolic blood pressure of 125 (which parallels the increase in mortality risk). (F) Plotting the SHAP interaction value of 'white blood cells' with 'blood urea nitrogen' shows that high white blood cell counts increase the negative risk conferred by high blood urea nitrogen. (G) Plotting the SHAP interaction value of sex vs. age in the mortality model shows how the differential risk of men and women changes over their lifetimes.

(CKD) populations may delay progression of kidney disease and reduce the risk of cardiovascular events (Methods 5.4.15).

Local interactions reveal sex-specific life expectancy changes during aging and inflammation effects in chronic kidney disease

Using SHAP interaction values, we can decompose the impact of a feature on a specific sample into a main effect and interaction effects with other features. This allows us to measure global interaction strength as well as decompose the SHAP dependence plots into main effects and interaction effects at a local (i.e., per sample) level (Figures S4B-D; Methods 5.4.16).

In the mortality dataset, plotting the SHAP interaction value between age and sex shows a clear change in the relative risk between men and women over a lifetime (Figure S4G). The largest difference in risk between men and women is at age 60. It is plausible that this increased risk is driven by increased cardiovascular mortality in men relative to women near that age [118]. This pattern is not clearly captured without SHAP interaction values because being male always confers greater risk of mortality than being female (Figure S4A).

In the chronic kidney disease model, an interesting interaction is observed between 'white blood cells' and 'blood urea nitrogen' (Figure S4F). High white blood cell counts are more concerning to the model when they are accompanied by high blood urea nitrogen. This supports the notion that inflammation may interact with high blood urea nitrogen to contribute to faster kidney function decline [15, 44].

Local model monitoring reveals previously invisible problems with deployed machine learning models

Here we show how using TreeExplainer to explain a model's *loss*, instead of a model's prediction, can improve our ability to monitor deployed models (Methods 5.4.17). Deploying machine learning models in practice is challenging because of the potential for input features to change after deployment. It is hard to detect when such changes occur, so many bugs in machine learning pipelines go undetected, even in core software at top tech companies [186]. We demonstrate that local model monitoring helps debug model deployments by decomposing the loss among the model's input features and so identifying problematic features (if any) directly. This is a significant improvement over simply speculating about the cause of global model performance fluctuations.

We simulated a model deployment with the hospital procedure duration dataset using the first year of data for training and the next three years for deployment. We present three examples: (1) is an intentional error, (2) and (3) are previously undiscovered problems. 1) We intentionally swapped the labels of operating rooms 6 and 13 two-thirds of the way through the dataset to mimic a typical feature pipeline bug. The overall loss of the model's predictions gives no indication that a problem has occurred (Figure S5A), whereas the *SHAP monitoring plot* for the room 6 feature clearly shows when the labeling error begins (Figure S5B). 2) Figure S5C shows a spike in error for the general anesthesia feature shortly after the deployment window begins. This spike corresponds to a subset of procedures affected by a previously undiscovered temporary electronic medical record configuration problem (Methods 5.4.17). 3) Figure S5D shows an example of feature drift over time, not of a processing error. During the training period and early in deployment, using the 'atrial fibrillation' feature lowers the loss; however, the feature becomes gradually less useful over time and ends up hurting the model. We found this drift was caused by significant changes in atrial fibrillation ablation procedure duration, driven by technology and staffing changes (Supplementary Figure S15; Methods 5.4.17).

Current deployment practice is to monitor the overall loss of a model (Figure S5A) over time, and potentially statistics of input features. TreeExplainer enables us to instead directly allocate a model's loss among individual features.

Local explanation embeddings reveal population subgroups relevant to mortality risk and complementary diagnostic indicators in chronic kidney disease

Unsupervised clustering and dimensionality reduction are widely used to discover patterns characterizing subgroups of samples (e.g., study participants), such as disease subtypes [142, 158]. They present two drawbacks: 1) the distance metric does not account for the discrepancies among the units/meaning of features (e.g., weight vs. age), and 2) there is no way for an unsupervised approach to know which features are

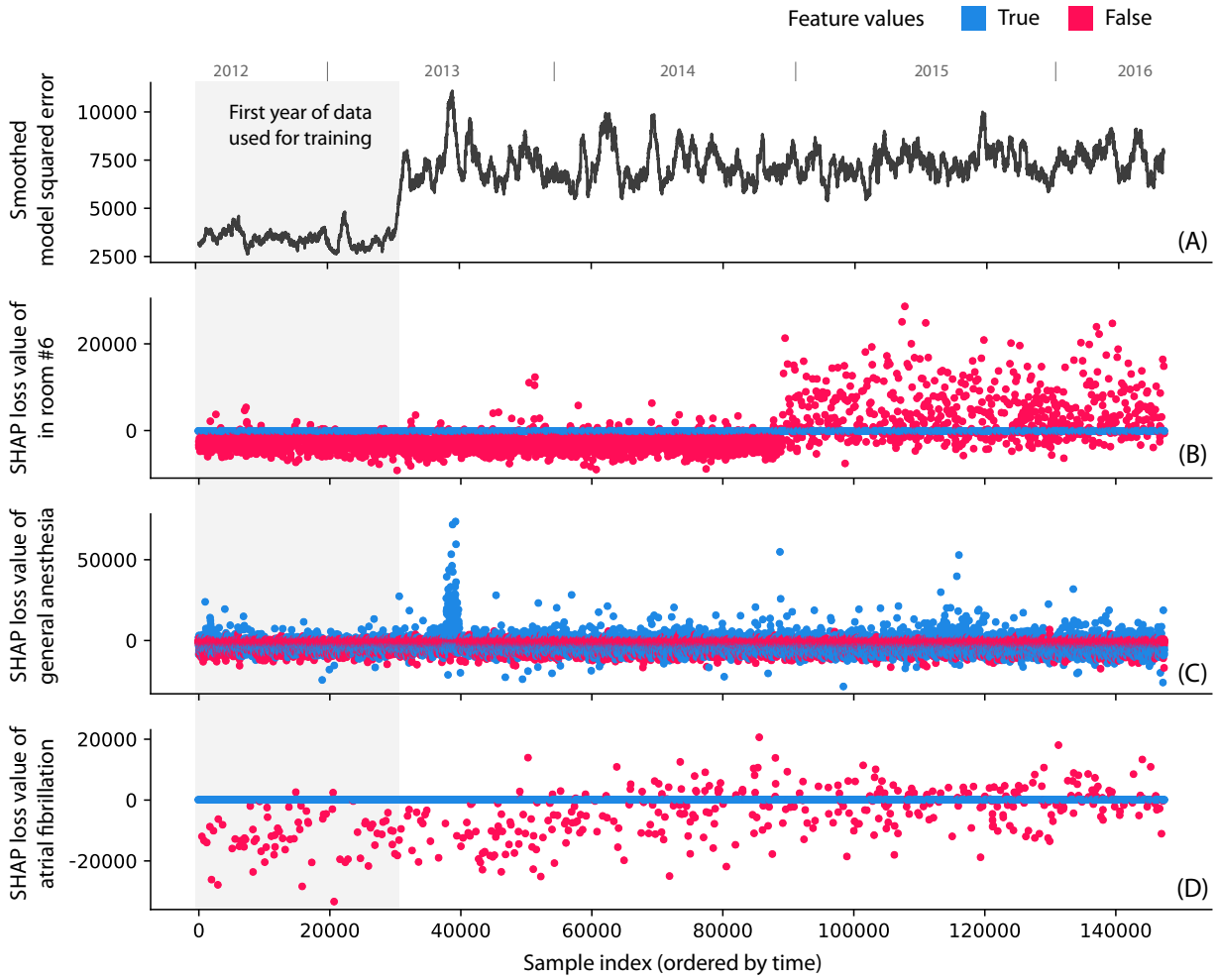


Figure S5: **Monitoring plots reveal problems that would otherwise be invisible in a retrospective hospital machine learning model deployment.** (A) The squared error of a hospital duration model averaged over the nearest 1,000 samples. The increase in error after training is because the test error is (as expected) higher than the training error. (B) The SHAP value of the model loss for the feature indicating if the procedure happens in room 6. The significant change is where we intentionally swapped the labels of room 6 and 13, which is invisible in the overall model loss. (C) The SHAP value of the model loss for the general anesthesia feature; the spike one-third of the way into the data is the result of a previously unrecognized transient data corruption at a hospital. (D) The SHAP value of the model loss for the atrial fibrillation feature. The upward trend of the plot shows feature drift over time (P-value 5.4×10^{-19}).

relevant for an outcome of interest, and so should be weighted more strongly. We can address both of these limitations by using *local explanation embeddings* to embed each sample into a new "explanation space." If we run clustering in this new space, we will get a *supervised clustering* where samples are grouped together based on their *explanations*. Supervised clustering naturally accounts for the differing units of various features, only highlighting changes that are relevant to a particular outcome (Methods 5.4.18).

Running hierarchical supervised clustering using the mortality model results in many groups of people that share a similar mortality risk for similar reasons (Figure S6A; Methods 5.4.18). Analogously, we can also run PCA on local explanation embeddings for chronic kidney disease samples, which uncovers the two primary categories of risk factors that identify unique individuals at risk of end-stage renal disease. This is consistent with the fact that clinically these factors should be measured in parallel (Figures S6B-D; Methods 5.4.18). This type of insight into the overall structure of kidney risk is not at all apparent when just looking at a standard unsupervised embedding (Supplementary Figure S17).

5.3 Discussion

Tree-based machine learning models are used in many domains where interpretability is important. We have sought to significantly improve the interpretability of these models in three main ways: First, we present a new exact method for computing the game-theoretic Shapley values, the only explanations that have several desirable properties. Second, we present a new richer type of local explanation that directly captures interaction effects. Finally, we propose many new tools for model interpretation based on combining local explanations. Local explanations have a distinct advantage over global explanations because by only focusing on a single sample they can remain more faithful to the original model. We anticipate that in the future local explanations will become foundational building blocks for many downstream tasks in machine learning.

Acknowledgements

The results presented in this chapter have been released as a paper in collaboration with my co-authors [104]: Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee.

We are grateful to Ruqian Chen, Alex Okeson, Cassianne Robinson, Vadim Khotilovich, Nao Hiranuma, Joseph Janizek, Marco Tulio Ribeiro, Jacob Schreiber, and members of the Lee lab for the feedback and assistance they provided during the development and preparation of this research. This work was funded by National Science Foundation [DBI-1759487, DBI-1355899, DGE-1762114, and DGE-1256082]; American Cancer Society [127332-RSG-15-097-01-TBG]; and National Institutes of Health [R35 GM 128638].

The Chronic Renal Insufficiency Cohort (CRIC) study was conducted by the CRIC Investigators and supported by the National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK). The data from the CRIC study reported here were supplied by the NIDDK Central Repositories. This manuscript was not prepared in collaboration with Investigators of the CRIC study and does not necessarily reflect the opinions or views of the CRIC study, the NIDDK Central Repositories, or the NIDDK.

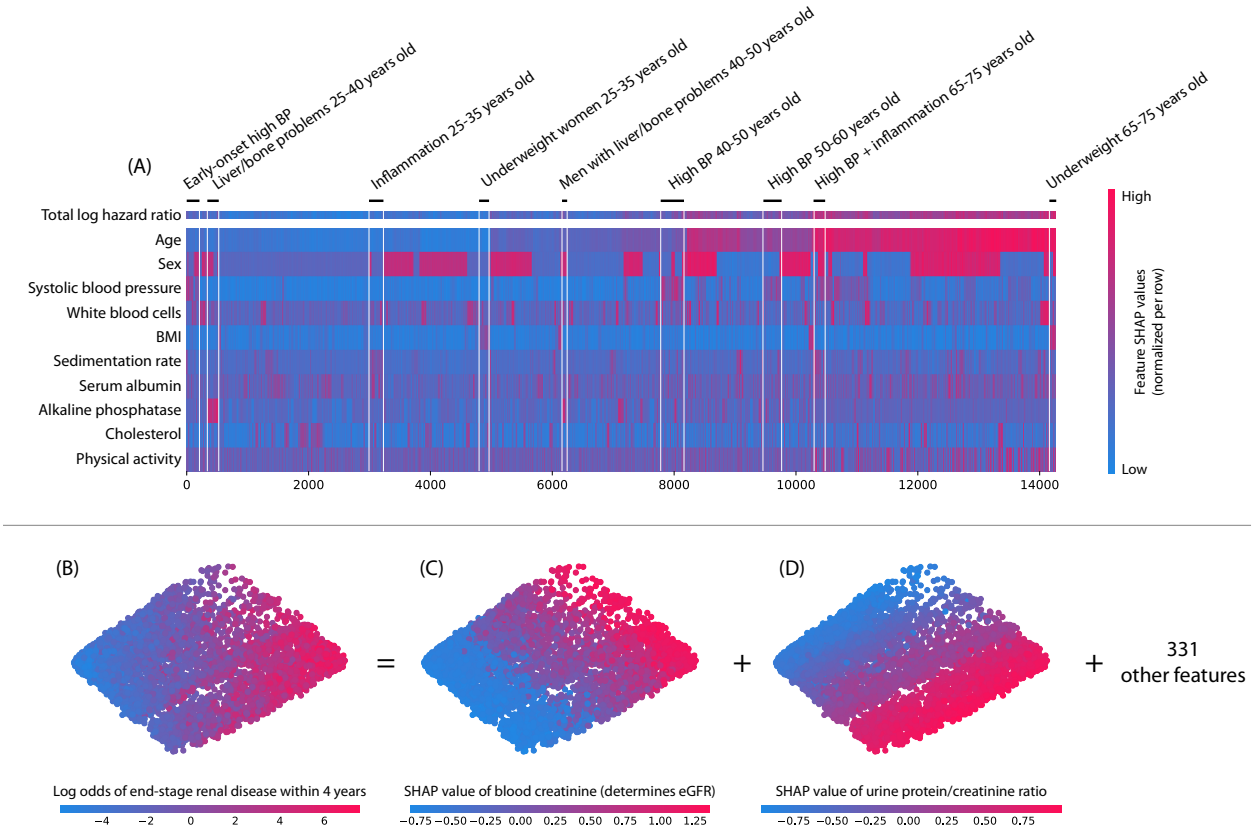


Figure S6: **Local explanation embeddings support both supervised clustering and interpretable dimensionality reduction.** (A) A clustering of mortality study individuals by their local explanation embedding. Columns are patients, and rows are features' normalized SHAP values. Sorting by a hierarchical clustering reveals population subgroups that have distinct mortality risk factors. (B-D) A local explanation embedding of kidney study visits projected onto two principal components. Local feature attribution values can be viewed as an embedding of the samples into a space where each dimension corresponds to a feature and all axes have the units of the model's output. (B) The embedding colored by the predicted log odds of a participant developing end-stage renal disease with 4 years of that visit. (C) The embedding colored by the SHAP value of blood creatinine. (D) The embedding colored by the SHAP value of the urine protein/creatinine ratio. Many other features also align with these top two principal components (Supplementary Figure S16), and an equivalent unsupervised PCA embedding is far less interpretable (Supplementary Figure S17)

5.4 Methods

5.4.1 Institutional review board statement

The chronic kidney disease data used in this study was obtained from the Chronic Renal Insufficiency Cohort (CRIC) study. University Washington Human Subjects Division determined that our study does not involve human subjects because we do not have access to identifiable information (IRB ID: STUDY00006766).

The hospital procedure data used for this study was retrieved from three institutional electronic medical record and data warehouse systems after receiving approval from the Institutional Review Board (University of Washington Human Subjects Division, Approval no. 46889). Protected health information was excluded from the dataset that was used for the machine-learning methods.

5.4.2 The three medical datasets used for experiments

Mortality dataset

The mortality data was obtained from the National Health and Nutrition Examination Survey (NHANES I) conducted by the U.S. Centers for Disease Control (CDC), as well as the NHANES I Epidemiologic Follow-up Study (NHEFS) [30]. Raw tape-format data files were obtained from the CDC website and converted to a tabular format by custom scripts. This reformatted version of the public data has been released at `http://github.com/suinleelab/treexplainer-study`. NHANES I examined 23,808 individuals in the United States between 1971 and 1974, recording a large number of clinical and laboratory measurements. The NHANES I Epidemiologic Followup Study researched the status of the original NHANES I participants as of 1992 to identify when they had died, or alternatively when they were last known to be alive. After filtering NHANES I to subjects that had both followup mortality data as well as common health measurements (such as systolic blood pressure) we obtained 79 features for 14,407 individuals, of which 4,785 individuals had recorded death dates before 1992. This data was used to train several cox proportional hazard ratio model types (Section 5.2.1). Because NHANES I represents a cross section of the United States population, it is a classic dataset that has been often used to understand the association between standard clinical measurements and long term health outcomes [91, 45].

Chronic kidney disease dataset

The kidney dataset is from the Chronic Renal Insufficiency Cohort (CRIC) Study which follows individuals with chronic kidney disease recruited from 7 clinical centers [89]. Participants are assessed at an annual visit and study follow-up is ongoing. We joined both visit and person level attributes to create 333 features for 10,745 visits among 3,939 patients. For each visit, we determined if the patient progressed to 'end-stage renal disease' within the following four years. (End-stage renal disease is the last stage of chronic kidney disease when kidneys are functioning at 10-15% of their normal capacity; dialysis or kidney transplantation is necessary to sustain life.) Predicting this progression outcome results in a binary classification task. Understanding what leads some people with chronic kidney disease to progress to end-stage renal disease while others do not is a priority in clinical kidney care that can help doctors and patients better manage and treat their condition [125, 177, 170, 112]. In the United States chronic kidney disease affects 14% of the population, so improving our management and understanding of the disease can have a significant positive impact on public health [78].

Hospital procedure duration dataset

Our hospital's operating rooms have installed an Anesthesia Information Management System - AIMS (Merge AIMS, Merge Inc., Hartland, WI) that integrates with medical devices and other electronic medical record systems to automatically acquire hemodynamic, ventilation, laboratory, surgery schedule and patient registration data. The automatic capture of data is supplemented by the manual documentation of medications and anesthesia interventions to complete the anesthesia record during a surgical episode. We extracted data from the AIMS database from June 2012 to May 2016. The corresponding medical history data of each patient were also extracted from our electronic medical record data warehouse (Caradigm Inc., Redmond, WA). Patient and procedure specific data available prior to surgery were captured and summarized into

2,185 features over 147,000 procedures. These data consist of diagnosis codes, procedure types, location, free text notes (represented as a bag of words), and various other information recorded in the EMR system. We measured the duration of a procedure as the time spent in the room by the patient. This is an important prediction task since one potential application of machine learning in hospitals is to reduce costs by better anticipating procedure duration (and so improve surgery scheduling). When hospital scheduling systems depend on machine learning models it is important to monitor the ongoing performance of the model. In Section 5.2.7 we demonstrate how local explanations can significantly improve the process of monitoring this type of model deployment.

5.4.3 Model accuracy performance experiments

Modern gradient boosted decision trees often provide state-of-the-art performance on tabular style datasets where features are individually meaningful, as consistently demonstrated by open data science competitions [46, 24]. All three medical datasets we examine here represent tabular-style data, and gradient boosted trees achieve the highest accuracy across all three datasets (Figure S2A). We used 100 random train/test splits followed by retraining to assess the statistical significance of the separation between methods. In the mortality dataset, gradient boosted trees outperformed the linear lasso with a P-value < 0.01 and the neural network with a P-value of 0.03. In the chronic kidney disease dataset, gradient boosted trees trees outperformed the linear lasso with a P-value < 0.01 and the neural network with a P-value of 0.08. In the hospital procedure duration dataset, gradient boosted trees trees outperformed both the linear lasso and neural network with a P-value < 0.01. The details of how we trained each method and obtained the results in Figure S2A are presented below.

Mortality dataset

For NHANES I mortality prediction in Figure S2A, we used a cox proportional hazards loss, and a C-statistic [63] to measure performance. For each of the three algorithms we split the $14,407$ samples according to a 64/16/20 split for train/validation/test. The features for the linear and the neural network models were mean imputed and standardized based on statistics computed on the training set. For the gradient boosted tree models, we passed the original data unnormalized and with missing values.

Gradient Boosted Trees - Gradient boosted trees were implemented using an XGBoost [24] model with the cox proportional hazards objective (we implemented this objective and merged it into XGBoost to support the experiments in this paper). Hyper-parameters were chosen using coordinate descent on the validation set loss. This resulted in a learning rate of 0.001; $6,765$ trees of max depth 4 chosen using early stopping; ℓ_2 regularization of 5.5; no ℓ_1 regularization; no column sampling during fitting; and bagging sub-sampling of 50%.

Linear Model - The linear model for the mortality dataset was implemented using the lifelines Python package [33]. The ℓ_2 regularization weight was chosen using the validation set and set to 215.44.

Neural Network - The neural network model was implemented using the DeepSurv Python package [73]. Running DeepSurv to convergence for mortality data took hours on a modern GPU server, so hyper-parameter tuning was done by manual coordinate decent. This resulted in ℓ_2 regularization of 1.0; the use of batch normalization; a single hidden layer with 20 nodes; a dropout rate of 0.5; a learning rate of 0.001 with no decay; and momentum of 0.9.

Chronic kidney disease dataset

For CRIC kidney disease prediction, we used logistic loss for binary classification, and measured performance using the area under the precision-recall curve in Figure S2A. For each of the three algorithms we split the $10,745$ samples according to a 64/16/20 split for train/validation/test. The features for the linear and the neural network models were mean imputed and standardized based on statistics computed on the training set. For the gradient boosted tree models we passed the original data unnormalized and with missing values.

Gradient Boosted Trees - Gradient boosted trees were implemented using an XGBoost [24] model with the binary logistic objective function. Hyper-parameters were chosen using coordinate descent on the validation set loss. This resulted in a learning rate of 0.003; 2,300 trees of max depth 5 chosen using early stopping; no ℓ_2 regularization; no ℓ_1 regularization; a column sampling rate during fitting of 15%; and bagging sub-sampling of 30%.

Linear Model - The linear model for the kidney dataset was implemented using scikit-learn [133]. Both ℓ_1 and ℓ_2 regularization were tested and ℓ_1 was selected based on validation set performance, with an optimal penalty of 0.1798.

Neural Network - The neural network model was implemented using Keras [26]. We chose to explore various feed-forward network architectures as well as 1D convolution based kernels (that learn a shared non-linear transform across many features). The best performance on the validation data came from a 1D convolution based architecture. After coordinate descent hyper-parameter tuning, we chose 15 1D convolution kernels which then go into a layer of 10 hidden units with rectified linear unit activation functions. Dropout of 0.4 was used during training between the convolution kernels and the hidden layer, and between the hidden layer and the output. Because stochastic gradient descent can have varying performance from run to run, we chose the best model based on validation loss from 10 different optimization runs.

Hospital procedure duration dataset

For hospital procedure duration prediction, we used a squared error loss and measured performance by the coefficient of determination (R^2) in Figure S2A. Two different hospital procedure dataset splits were used in this paper. The first dataset split, used for model comparison in Section 5.2.1 (Figure S2A), consisted of 147,000 procedures divided according to a random 80/10/5/5 split for train/validation/test1/test2. The test2 set was only used to get final performance numbers and not during method development. The features for the linear and the neural network models were mean imputed and standardized based on statistics computed on the training set. For the gradient boosted tree models, we passed the original data unnormalized and with missing values. All hyper-parameter tuning was done with the validation dataset.

The second dataset split, used for the monitoring plots in Section 5.2.7, divided procedures from the two hospitals based on time: the first 1 year of data was used for training and the last 3 years were used for test, in a manner intended to simulate actual model deployment. Models for this data were not hyper-parameter tuned, as the goal was not to achieve perfect performance but to demonstrate the value of monitoring methods; instead, they simply used the best hyper-parameter values found on the first (random-splits) dataset.

Gradient Boosted Trees - Gradient boosted trees were implemented using an XGBoost [24] model with a squared loss objective function. Hyper-parameters were chosen using grid search over the validation set loss. This resulted in a learning rate of 0.1; 530 trees of max depth 10 chosen using early stopping; a column sampling rate during fitting of 50%; and no bagging sub-sampling.

Linear Model - The linear model for the procedure duration dataset was implemented using the `LassoCV` class in scikit-learn [133]. ℓ_1 regularization was tuned based on validation set performance to 0.120.

Neural Network - The neural network model was implemented using Keras [26]. We limited our architecture search to feed-forward networks with up to three hidden layers, and sizes up to 4,096 nodes per layer. The best performance on validation data came from a single-hidden-layer architecture with a 1,024-node layer followed by a dropout probability of 0.75 before the output.

5.4.4 Interpretability comparison of linear models and tree-based models in the presence of non-linearities

Even if linear models appear easier to interpret, their high bias may force the relationships they learn to be farther from the truth than a low-bias tree-based model (Figures S2B-D). We illustrated this using *simulated outcomes* from the NHANES I mortality dataset with varying amounts of non-linearity between certain features and the outcome. The input feature data was used as is. Even if the simulated outcome only

depends on age and body mass index (BMI), the linear model learns features in the mortality dataset that are non-linearly dependent with age and BMI to try to approximate non-linear relationships with the outcome. This increases the test accuracy of the linear model slightly, as seen by the small increase in performance from the two-feature linear model (dashed red) to the all-feature linear model (solid red) (Figure S2C). However, this comes at the cost of placing much of its weight on features that are not actually used to generate the outcome. In contrast, the gradient boosted tree model correctly uses only age and BMI across all levels of function non-linearity (Figure S2D). Below we describe the experimental setup behind the results in Figure S2B-D (Section 5.2.2).

Selection of features

We generated a synthetic label with a known relationship to two input features from the mortality dataset, but to the extent possible, we intended this synthetic relationship to be realistic; while still retaining the ability to control the amount of non-linearity in the relationship. Starting with the classifiers trained on the full mortality (NHANES I) dataset, SHAP dependence plots (Section 5.2.7) were used to find one feature that had a strong linear relationship with mortality (age) and one that had a "U-shaped" relationship with mortality (BMI). These two features were selected and used as the "true" label-generating features in our synthetic model.

Label generation

The synthetic label-generating function was constructed using a logistic function applied to the sum of a linear function of age and a quadratic function of BMI. The functional form allows us to smoothly vary the amount of nonlinearity in the label-generating model. The quadratic function of BMI was parameterized as $(1-p)$ times a linear function, plus p times a quadratic function, where both functions had the same minimum and maximum values (the x-location of the minimum for the quadratic was set to the mean BMI). The contribution in logits was set to range from a minimum of -2.0 to a maximum of 2.0 for age, and -1.0 to 1.0 for BMI, so that even when nonlinear contribution were the strongest ($p = 1$) the linear contribution of the main risk factor was still more important (as true in the original data). The output for each data point was then a probability to be predicted by our models (we did not add noise by binarizing these probabilities to 0-1, so as to avoid the need to average our results over many random replicates). Thus the final label-generating model was

$$y = \sigma\left((1.265(\text{age}) + 0.0233) + (1-p)(0.388(\text{BMI}) - 0.325) + (p)(1.714(\text{BMI})^2 - 1)\right)$$

where $\sigma = \frac{1}{1+e^{-t}}$ is the logistic function. A total of 11 datasets were generated, with $p \in [0.0, 0.1, 0.2...1.0]$, by applying this labeling function to the true data covariates, so that the matrix of predictors X was real data but the labels y were generated synthetically by a known mechanism.

Model training

We trained both gradient boosted trees and linear logistic regression to predict the synthetic labels. For each of the 11 datasets with varying degrees p of nonlinearity, models were tuned on a validation set and evaluated on test data with a 64/16/20 train/validation/test split. The only logistic regression hyper-parameter was the L1 regularization penalty which was optimized over the range $\lambda \in [10^{-4}, 10^{-3}...10^{3}, 10^{4}]$. The tuned gradient boosting model hyper-parameters were optimized over the tree depths of $[1, 2, 4, 8, 10]$ and bagging sub-sampling over the rates $[0.2, 0.5, 0.8, 1.0]$. The learning rate was fixed at 0.01; the minimum loss reduction for splitting to 1.0; the minimum child weight for splitting was 10; and trees we trained for a maximum of $1,000$ rounds with early stopping based on validation set loss.

Feature importance

Per-sample importance values were calculated for each feature in each of the 11 datasets using SHAP values for both the logistic (using Linear SHAP values assuming feature independence [101]) and gradient boosted tree (using TreeExplainer's Tree SHAP algorithm) models. At each value of p, the total weight of irrelevant

samples was calculated by taking the absolute value of all SHAP values for all features other than age and BMI, and summing these values across all samples and features.

5.4.5 Previous Global Explanation Methods for Trees

While local explanation methods for trees have not been extensively studied, interpreting tree-based models by assigning a global importance value to each input feature is a well studied problem and many methods have been proposed [47, 46, 147, 165, 166, 68, 4, 99, 74]. The most basic global approach is to simply count the number of times a feature was used for splitting, but this fails to account for the differing impacts of different splits. A better approach is to attribute the reduction in loss (aka. Gain) provided by each split in each decision tree to the feature that was split on [17, 47]. This "Gain" measure of feature importance was shown to correctly recover the mutual information between the input features and the outcome label in the limit of an infinite ensemble of totally random fully developed trees [99]. However, it becomes biased for finite ensembles of greedily built trees, and so approaches have been designed to account for this bias when using Gain for feature selection [67, 23]. Another popular method for determining feature importance is to permute the data column corresponding to a feature and then observe the change in the model's loss [16]. If the model's loss increases significantly when a feature is permuted then it indicates the model was heavily depending on that feature. This permutation approach can be further extended to account for statistical dependencies between features by permuting only within specified groups of samples [166]. All of these approaches are designed to estimate the global importance of a feature over an entire dataset, so they are not directly applicable to local explanations that are specific to each prediction. If we try to use global methods in place of true local explanations we get significantly worse performance on many benchmark metrics (Figure S3).

While not explicitly designed to be a global feature attribution method, TreeExplainer can be used as a global method by averaging many local explanations. If we do this over all samples in a dataset, then we get a global measure of feature importance that does not suffer from the inconsistencies of the classic Gain method (Supplementary Figure S8), and unlike the permutation method, does not miss high-order interaction effects. Global feature attribution based on TreeExplainer has a higher power to detect important features in the presence of interactions than current state-of-the-art methods (Supplementary Figure S13). This has important implications for the popular task of feature selection based on tree-ensembles.

5.4.6 Previous local explanation methods for trees

As described in Section 5.2.3, we are aware of only two previous tree-specific local explanation methods: reporting the decision path; and an unpublished heuristic difference in expectations method (proposed by Saabas) [146]. Since reporting the decision path is not useful for large tree ensembles we instead focus on the heuristic Saabas method. The Saabas difference in expectations approach explains a prediction by following the decision path and attributing changes in the expected output of the model to each feature along the path. This is efficient since the expected value of every node in the tree can be estimated by averaging the model output over all the training samples that pass through that node. Let f be a decision tree model, x the instance we are going to explain, $f(x)$ the output of the model for the current instance, and $f_x(S) \approx E[f(x) \mid x_S]$ the estimated expectation of the model output conditioned on the set S of feature values (Methods 5.4.10, Algorithm 1), then we can define the *Saabas value* for the i'th feature as

$$\phi_i^s(f, x) = \sum_{j \in D_x^i} f_x(A_j \cup j) - f_x(A_j), \tag{5.1}$$

where D_x^i is the set of nodes on the decision path from x that split on feature i, and A_j is the set of all features split on by ancestors of j. Equation 5.1 results in a set of feature attribution values that sum up to the difference between the expected output of the model and the output for the current prediction being explained (Supplementary Figure S9). When explaining an ensemble model made up of a sum of many decision trees, the Saabas values for the ensemble model are defined as the sum of the Saabas values for each tree.

5.4.7 Model agnostic local explanation methods

Many different local explanation methods have been proposed in the literature [7, 167, 141, 32, 101, 140]. The most well known is simply taking the gradient of the model's output with respect to its inputs at the current sample. This is common in the deep learning literature, as is the related approach of multiplying the gradient times the value of the input features. Relying entirely on the gradient of the model at a single point though can often be misleading [155]. To provide a better allocation of credit for deep learning models, various methods have been proposed that either modify the standard gradient back propagation rule to instead propagate attributions [160, 181, 6, 155, 82, 1], or integrate the gradient along a path based on fair allocation rules from economics [169].

In contrast to deep learning methods, model-agnostic methods make no assumptions about the internal structure of the model. These methods rely only on observing the relationship between changes in the model inputs and model outputs. This can be done by training a global mimic model to approximate the original model, then locally explaining the mimic model by either taking its gradient [7], or fitting a local linear model as in MAPLE [138]. Alternatively, the mimic model can be fit to the original model locally for each prediction. For a local linear mimic model the coefficients can be used as an explanation, as in the popular LIME method [141]. For a local decision rule mimic model the rules can be used as the explanation as in Anchors [140]. Another class of approaches do not explicitly fit a mimic model, but instead perturb sets of features to measure their importance, then use methods from game theory to fairly allocate the importance of these sets among the input features, this class includes IME [167] and QII [32].

Perhaps surprisingly, despite the seeming variety of different local explanation methods, two back propagation-style deep learning methods [6, 155], local linear mimic models [141], and several game theoretic methods [95, 167, 32] were recently unified into a single class of *additive feature attribution methods* in our prior study [101]. This class is of particular interest since results from cooperative game theory imply there is a unique optimal explanation approach in the class (the Shapley values) that satisfies several desirable properties [151, 145]. Unfortunately computing the Shapley values is NP-hard in general [111], with a runtime cost exponential in the number of input features. When faced with an NP-hard optimization problem it is typical to build approximate methods, which exactly IME [167] or QII [32] do. However, here we take an alternative approach and restrict our focus specifically to tree-based machine learning models. By doing this we are able to show constructively that solving for the exact Shapley values in trees is not NP-hard, and can be solved by TreeExplainer in low-order polynomial time (Methods 5.4.10).

5.4.8 Convergence experiments for model agnostic Shapley value approximations

In Supplementary Figures S7C-D we generated random datasets of increasing size and then explained (over)fit XGBoost models with 1,000 trees. The runtime and standard deviation of the local explanations are reported for Kernel SHAP [101], IME [167], and TreeExplainer; except that for Kernel SHAP and IME the reported times are only a lower bound. Both the IME and Kernel SHAP model-agnostic methods must evaluate the original model a specific number of times for each explanation, so the time spent evaluating the original model represents a lower bound on the runtime of the methods (note that the QII method [32] is not included in our comparisons since for local feature attribution it is identical to IME). In Supplementary Figure S7C we report this lower bound, but it is important to note that in practice there is also additional overhead associated with the actual execution of the methods that depends on how efficiently they are implemented. We also only used a single background reference sample for the model-agnostic approaches. This allows them to converge faster at the expense of using less accurate estimates of conditional expectations. Increasing the number of background samples would only further reduce the computational performance of these methods. Each method is run ten times, then the standard deviation for each feature is divided by the mean of each feature to get a normalized standard deviation (Supplementary Figure S7D). In order to maintain a constant level of normalized standard deviation, Kernel SHAP and IME are allowed a linearly increasing number of samples as the number of features in a dataset, M, grows. In Supplementary Figure S7C, TreeExplainer is so much faster than the model-agnostic methods that it appears to remain unchanged as we scale M, though in reality there is a small growth in its runtime. In Supplementary Figure S7D there is truly no variability since the TreeExplainer method is exact and not stochastic.

In Supplementary Figures S7E-F, the different explainers are compared in terms of estimation error (absolute deviation from the ground truth) on the chronic kidney disease dataset. We chose this dataset model for our comparisons because it is much smaller than the hospital procedure duration dataset, and has a more common loss function than the mortality dataset model (logistic loss vs. a cox proportional hazards loss). Ground truth is obtained via TreeExplainer's exact Independent Tree SHAP algorithm with the reference set fixed to be the mean of the data. The plots are obtained by increasing the number of samples allowed to each explainer and reporting the max and mean estimation error. For IME, we tune the minimum samples per feature, a hyper-parameter that is utilized to estimate which features' attributions have larger variability. After the minimum samples per feature has been achieved, the rest of the samples are allocated so as to optimally reduce the variance of the sum of the estimated values (by giving more samples to features with high sampling variance). As expected this is beneficial to the max evaluation error, but can potentially lead to bias (as for IME (min 10) in Supplementary Figure S7E). While often helpful, ℓ_1 regularization was not useful to improve Kernel SHAP for this dataset, so we report the results from unregularized regression.

To measure the cost of computing low variance estimates of explanations for the chronic kidney disease dataset we defined "low variance" as 1% of the tenth largest feature impact (out of 333 features), then measured how many samples it took on average to reach a standard deviation below that level (where standard deviation is measured across repeated runs of the explanation method). This was done for both the maximum standard deviation across all features (Supplementary Figure S7E), and the mean standard deviation (Supplementary Figure S7F). Calculating low variance estimates for the experiments presented in this paper on the chronic kidney disease dataset would have taken almost 2 CPU days for basic explanations, and over 3 CPU years for interaction values (Section 5.2.6).

5.4.9 Unifying previous heuristics with Shapley values

Here we review the uniqueness guarantees of Shapley values from game theory as they apply to local explanations of predictions from machine learning models [151]. We then detail how to reinterpret previous local explanation heuristics for trees, and so connect them with Shapley values.

As applied here, Shapley values are computed by introducing each feature, one at at time, into a conditional expectation function of the model's output, $f_x(S) \approx E[f(x) \mid x_S]$ (Methods 5.4.10, Algorithm 1), and attributing the change produced at each step to the feature that was introduced; then averaging this process over all possible feature orderings (Supplementary Figure S9). Shapley values represent the only possible method in the broad class of *additive feature attribution methods* [101] that will simultaneously satisfy three important properties: *local accuracy*, *consistency*, and *missingness*.

Local accuracy (known as *additivity* in game theory) states that when approximating the original model f for a specific input x, the explanation's attribution values should sum up to the output $f(x)$:

Property 4 (Local accuracy / Additivity)**.**

$$f(x) = \phi_0(f) + \sum_{i=1}^{M} \phi_i(f, x) \tag{5.2}$$

The sum of feature attributions $\phi_i(f, x)$ matches the original model output $f(x)$, where $\phi_0(f) = E[f(z)] = f_x(\emptyset)$.

Consistency (known as *monotonicity* in game theory) states that if a model changes so that some feature's contribution increases or stays the same regardless of the other inputs, that input's attribution should not decrease:

Property 5 (Consistency / Monotonicity)**.** *For any two models f and f', if*

$$f'_x(S) - f'_x(S \setminus i) \geq f_x(S) - f_x(S \setminus i) \tag{5.3}$$

for all subsets of features $S \in \mathcal{F}$, then $\phi_i(f', x) \geq \phi_i(f, x)$.

Missingness (similar to *null effects* in game theory) requires features with no effect on the set function f_x to have no assigned impact. All local previous methods we are aware of satisfy missingness.

Property 6 (Missingness)**.** *If*

$$f_x(S \cup i) = f_x(S) \tag{5.4}$$

for all subsets of features $S \in \mathcal{F}$*, then* $\phi_i(f, x) = 0$*.*

The only way to simultaneously satisfy these properties is to use the classic Shapley values:

Theorem 3. *Only one possible feature attribution method based on* f_x *satisfies Properties 1, 2 and 3:*

$$\phi_i(f, x) = \sum_{R \in \mathcal{R}} \frac{1}{M!} \left[f_x(P_i^R \cup i) - f_x(P_i^R) \right] \tag{5.5}$$

where $\mathcal{R}$ *is the set of all feature orderings,* P_i^R *is the set of all features that come before feature* i *in ordering* R*, and* M *is the number of input features for the model.*

The equivalent of Theorem 3 has been previously presented in [101] and follows from cooperative game theory results [179], where the values ϕ_i are known as the Shapley values [151]. Shapley values are defined independent of the set function used to measure the importance of a set of features. Since here we are using f_x, a conditional expectation function of the model's output, we are computing the more specific *SHapley Additive exPlanation (SHAP) values* [101].

Surprisingly, there is a close parallel between the Saabas values (Equation 5.1) and the SHAP values (Equation 5.5). While SHAP values average the importance of introducing a feature over all possible feature orderings, Saabas values only consider the single ordering defined by a tree's decision path (Supplementary Figure S9). This mean that Saabas values satisfy the local accuracy property since they always sum up to the difference between the expected value of the model $E[f(x)]$ and the current output $f(x)$. But since they do not average over all orderings they do not match the SHAP values, and so must violate consistency. Examples of such inconsistencies can be found for even very small trees, where changing the model to depend more on one feature can actually cause the Saabas attribution values for that feature to decrease (Supplementary Figure S8). The difference this makes can be seen by examining trees representing multi-way AND functions. If we let all the input features be independent and identically distributed then no feature in an AND function should have any more credit than another, yet for Saabas values, splits near the root are given much less credit than splits near the leaves (Supplementary Figures S7A). This means that while mathematically all the features play an equal role in the model, the features near the root of the tree get little or no credit. This is particularly troubling since features near the root are likely the most important as they were chosen first by greedy splitting.

There is a close parallel between Saabas values and the classic "Gain" method for global feature attribution (sometimes known as Gini importance) [47]. Just as Gain attributes the change in loss after each split to the feature that was split on, so Saabas attributes the change in conditional expectation after each split to the feature that was split on. Both methods only consider a single order for introducing features into the model, the order defined by paths in the tree. Choosing to use only a single ordering leads to inconsistent allocation of credit (Supplementary Figure S7A and S8). Shapley values guarantee a consistent allocation of credit by averaging the change after each split over all possible orderings. It has been previously shown through a connection with mutual information (which is consistent) that the Gain method becomes consistent in the limit of an infinite ensemble of totally random fully developed trees [99]. This suggests that the Saabas method may also become consistent in the limit of an infinite ensemble of totally random fully developed trees. This is indeed the case, and we show in Theorem 4 that for binary features the Saabas values converge to the SHAP values in the limit of an infinite ensemble of totally random fully developed trees.

Theorem 4. *In the limit of an infinite ensemble of totally random fully developed trees on binary features the Saabas values equal the SHAP values [101] (which are Shapley values of a conditional expectation function of the model's output).*

Proof. Assume all features are binary, then the decision path of a single input instance x for a single tree will be a random ordering of all the input features. The Saabas values for that tree will be equivalent to a single permutation from the formula for Shapley values (Equation 5.5). Since there is an infinite ensemble of trees, all possible feature orderings will be represented in equal proportions. Given a finite output domain, there will furthermore be all possible feature orderings represented for each possible leaf value. Taking the

average of the Saabas values over the ensemble of trees then becomes the same as the averaging function in the definition of the Shapley values (Equation 5.5). □

5.4.10 TreeExplainer algorithms

Here, we describe the algorithms behind TreeExplainer in three stages. First, we describe an easy to understand (but slow) version of the main Tree SHAP algorithm, then we present the complex polynomial time version of Tree SHAP, and finally we describe the Independent Tree SHAP algorithm used for explaining non-linear model output transformations (such as the model's loss). While solving for the Shapley values is in general NP-hard [111], these algorithms show that by restricting our attention to trees, we can find exact algorithms that run in low-order polynomial time.

Tree SHAP

Tree SHAP, the main algorithm behind TreeExplainer, can exactly compute the Shapley values, and so guarantee consistent explanations (Property 5). Tree SHAP exactly computes Equation 5.5 in low order polynomial time, where the conditional expectation function, f_x, is defined using tree traversal (Algorithm 1). Letting T be the number of trees, D the maximum depth of any tree, and L the number of leaves, Tree SHAP has worst case complexity of $O(TLD^2)$. This represents an exponential complexity improvement over previous exact Shapley methods, which would have a complexity of $O(TLM2^M)$, where M is the number of input features. By directly computing the Shapley values we are able to guarantee that the explanations will always be consistent and locally accurate.

Estimating SHAP values directly in $O(TLM2^M)$ time If we ignore computational complexity then we can compute the SHAP values for a tree by estimating $E[f(x) \mid x_S]$ and then using Equation 5.5 (Methods 5.4.9). For a tree model, $E[f(x) \mid x_S]$ can be estimated recursively using Algorithm 1, where *tree* contains the information of the tree. v is a vector of node values; for internal nodes, we assign the value *internal*. The vectors a and b represent the left and right node indexes for each internal node. The vector t contains the thresholds for each internal node, and d is a vector of indexes of the features used for splitting in internal nodes. The vector r represents the cover of each node (i.e., how many data samples fall in that sub-tree).

Algorithm 1 finds $E[f(x) \mid x_S]$ by recursively following the decision path for x if the split feature is in S, and taking the weighted average of both branches if the split feature is not in S. The computational complexity of Algorithm 1 is proportional to the number of leaves in the tree, which when used on all T trees in an ensemble and plugged into Equation 5.5 leads to a complexity of $O(TLM2^M)$ for computing the SHAP values of all M features.

Algorithm 1 Estimating $E[f(x) \mid x_S]$

```
1:  procedure EXPVALUE(x, S, tree = {v, a, b, t, r, d})
2:      procedure G(j)                          ▷ Define the G procedure which we will call on line 10
3:          if v_j ≠ internal then              ▷ Check if node j is a leaf
4:              return v_j                      ▷ Return the leaf's value
5:          else
6:              if d_j ∈ S then                 ▷ Check if we are conditioning on this feature
7:                  return G(a_j) if x_{d_j} ≤ t_j else G(b_j)   ▷ Use the child on the decision path
8:              else
9:                  return [G(a_j)·r_{a_j} + G(b_j)·r_{b_j}]/r_j   ▷ Weight children by their coverage
10:     return G(1)                             ▷ Start at the root node
```

Estimating SHAP values in $O(TLD^2)$ time Now we calculate the same values as above, but in polynomial time instead of exponential time. Specifically, we propose an algorithm that runs in $O(TLD^2)$

time and $O(D^2 + M)$ memory, where for balanced trees the depth becomes $D = \log L$. Recall T is the number of trees, L is the maximum number of leaves in any tree, and M is the number of features.

The intuition of the polynomial time algorithm is to recursively keep track of what proportion of all possible subsets flow down into each of the leaves of the tree. This is similar to running Algorithm 1 simultaneously for all 2^M subsets S in Equation 5.5. Note that a single subset S can land in multiple leaves. It may seem reasonable to simply keep track of how many subsets (weighted by the cover splitting of Algorithm 1 on line 9) pass down each branch of the tree. However, this combines subsets of different sizes and so prevents the proper weighting of these subsets, since the weights in Equation 5.5 depend on $|S|$. To address this we keep track of each possible subset size during the recursion, not just single a count of all subsets. The *EXTEND* method in Algorithm 2 grows all these subset sizes according to a given fraction of ones and zeros, while the *UNWIND* method reverses this process and is commutative with *EXTEND*. The *EXTEND* method is used as we descend the tree. The *UNWIND* method is used to undo previous extensions when we split on the same feature twice, and to undo each extension of the path inside a leaf to compute weights for each feature in the path. Note that *EXTEND* keeps track of not just the proportion of subsets during the recursion, but also the weight applied to those subsets by Equation 5.5. Since the weight applied to a subset in Equation 5.5 is different when it includes the feature i, we need to *UNWIND* each feature separately once we land in a leaf, so as to compute the correct weight of that leaf for the SHAP values of each feature. The ability to *UNWIND* only in the leaves depends on the commutative nature of *UNWIND* and *EXTEND*.

In Algorithm 2, m is the path of unique features we have split on so far, and contains four attributes: i) d, the feature index, ii) z, the fraction of "zero" paths (where this feature is not in the set S) that flow through this branch, iii) o, the fraction of "one" paths (where this feature is in the set S) that flow through this branch, and iv) w, which is used to hold the proportion of sets of a given cardinality that are present weighted by their Shapley weight (Equation 5.5). Note that the weighting captured by w does not need to account for features not yet seen on the decision path so the effective size of M in Equation 5.5 is growing as we descend the tree. We use the dot notation to access member values, and for the whole vector $m.d$ represents a vector of all the feature indexes. The values p_z, p_o, and p_i represent the fraction of zeros and ones that are going to extend the subsets, and the index of the feature used to make the last split. We use the same notation as in Algorithm 1 for the tree and input vector x. The child followed by the tree when given the input x is called the "hot" child. Note that the correctness of Algorithm 2 (as implemented in the open source code) has been validated by comparing its results to the brute force approach based on Algorithm 1 for thousands of random models and datasets where $M < 15$.

Complexity analysis: Algorithm 2 reduces the computational complexity of exact SHAP value computation from exponential to low order polynomial for trees and sums of trees (since the SHAP values of a sum of two functions is the sum of the original functions' SHAP values). The loops on lines 6, 12, 21, 27, and 34 are all bounded by the length of the subset path m, which is bounded by D, the maximum depth of a tree. This means the complexity of *UNWIND* and *EXTEND* is bounded by $O(D)$. Each call to *RECURSE* incurs either $O(D)$ complexity for internal nodes, or $O(D^2)$ for leaf nodes, since *UNWIND* is nested inside a loop bounded by D. This leads to a complexity of $O(LD^2)$ for the whole tree because the work done at the leaves dominates the complexity of the internal nodes. For an entire ensemble of T trees this bound becomes $O(TLD^2)$. If we assume the trees are balanced then $D = \log L$ and the bound becomes $O(TL \log^2 L)$. □

Independent Tree SHAP: Estimating SHAP values under independence in $O(TRL)$ time

The Tree SHAP algorithm provides fast exact solutions for trees and sums of trees (because of the linearity of Shapley values [151]), but there are times when it is helpful to explain not the direct output of the trees, but also a non-linear transform of the tree's output. A compelling example of this is explaining a model's loss function, which is very useful for model monitoring and debugging (Section 5.2.7).

Unfortunately, there is no simple way to adjust the Shapley values of a function to exactly account for a non-linear transformation of the model output. Instead, we combine a previously proposed compositional approximation (Deep SHAP) [101] with ideas from Tree SHAP to create a fast method specific to trees, *Independent Tree SHAP*. The compositional approach requires iterating over each background sample from the dataset used to compute the expectation. This means that conditional expectations can no longer be computed using Algorithm 1, since only the path corresponding to the current background reference sample will be defined. To solve this we enforce independence between input features then develop Independent Tree

Algorithm 2 Tree SHAP

procedure TREESHAP(x, $tree = \{v, a, b, t, r, d\}$)
  ϕ = array of $len(x)$ zeros
  procedure RECURSE(j, m, p_z, p_o, p_i)
    $m =$ EXTEND(m, p_z, p_o, p_i) ▷ Extend subset path with a fraction of zeros and ones
    if $v_j \neq internal$ **then** ▷ Check if we are at a leaf node
      for $i \leftarrow 2$ to $len(m)$ **do** ▷ Calculate the contributions from every feature in our path
        $w = sum(\text{UNWIND}(m, i).w)$ ▷ Undo the weight extension for this feature
        $\phi_{m_i} = \phi_{m_i} + w(m_i.o - m_i.z)v_j$ ▷ Contribution from subsets matching this leaf
    else
      $h, c = (a_j, b_j)$ **if** $x_{d_j} \leq t_j$ **else** (b_j, a_j) ▷ Determine hot and cold children
      $i_z = i_o = 1$
      $k = \text{FINDFIRST}(m.d, d_j)$
      if $k \neq$ nothing **then** ▷ Undo previous extension if we have already seen this feature
        $i_z, i_o = (m_k.z, m_k.o)$
        $m = \text{UNWIND}(m, k)$
      RECURSE(h, m, $i_z r_h / r_j$, i_o, d_j) ▷ Send both zero and one weights to the hot child
      RECURSE(c, m, $i_z r_c / r_j$, 0, d_j) ▷ Send just zero weights to the cold child
  procedure EXTEND(m, p_z, p_o, p_i)
    $l, m = len(m), copy(m)$
    $m_{l+1}.(d, z, o, w) = (p_i, p_z, p_o, (1 \textbf{ if } l = 0 \textbf{ else } 0))$ ▷ Init subsets of size l
    for $i \leftarrow l$ to 1 **do** ▷ Grow subsets using p_z and p_o
      $m_{i+1}.w = m_{i+1}.w + p_o \cdot m_i.w \cdot (i/l)$ ▷ Subsets that grow by one
      $m_i.w = p_z \cdot m_i.w \cdot (l - i)/l$ ▷ Subsets that stay the same size
    return m ▷ Return the new extended subset path
  procedure UNWIND(m, i) ▷ The inverse of the ith call to EXTEND($m, ...$)
    $l, n, m = len(m), m_l.w, copy(m_{1 \ldots l-1})$
    for $j \leftarrow l - 1$ to 1 **do** ▷ Shrink subsets using $m_i.z$ and $m_i.o$
      if $m_i.o \neq 0$ **then**
        $t = m_j.w$
        $m_j.w = n \cdot l/(j \cdot m_i.o)$
        $n = t - m_j.w \cdot m_i.z \cdot (l - j)/l$
      else
        $m_j.w = (m_j.w \cdot l)/(m_i.z(l - j))$
    for $j \leftarrow i$ to $l - 1$ **do**
      $m_j.(d, z, o) = m_{j+1}.(d, z, o)$
    return m
  RECURSE(1, [ ], 1, 1, 0) ▷ Start at first node with all zero and one extensions
  return ϕ

SHAP as a single-reference version of Tree SHAP (Algorithm 3).

Independent Tree SHAP enforces an independence assumption between the conditional set S and the set of remaining features ($x_S \perp x_{\bar{S}}$). Utilizing this independence assumption, Shapley values with respect to R individual background samples can be averaged together to get the attributions for the full distribution. Accordingly, Algorithm 3 is performed by traversing hybrid paths made up of a single foreground and background sample in a tree. At each internal node, *RECURSE* traverses down the tree, maintaining local state to keep track of the set of upstream features and whether each split went down the path followed by the foreground or background sample. Then, at each leaf, two contributions are computed – one positive and one negative. Each leaf's positive and negative contribution depends on the feature being explained. However, calculating the Shapley values by iterating over all features at each leaf would result in a quadratic time algorithm. Instead, *RECURSE* passes these contributions up to the parent node and determines whether to assign the positive or negative contribution to the feature that was split upon based on the directions the foreground and background samples traversed. Then the internal node aggregates the two positive contributions into a single positive contribution and two negative contributions into a single negative contribution and passes it up to its parent node.

Note that both the positive and negative contribution at each leaf is a function of two variables: 1) U: the number of features that matched the foreground sample along the path and 2) V: the total number of unique features encountered along the path. This means that for different leaves, a different total number of features V will be considered. This allows the algorithm to consider only $O(L)$ terms, rather than an exponential number of terms. Despite having different U's at each leaf, Independent Tree SHAP exactly computes the traditional Shapley value formula (which considers a fixed total number of features $\geq V$ for any given path) because the terms in the summation group together nicely.

Complexity Analysis: If we assume *CALCWEIGHT* takes constant time (which it will if the factorial function is implemented based on lookup tables), then Algorithm 3 performs a constant amount of computation at each node. This implies the complexity for a single foreground and background sample is $O(L)$, since the number of nodes in a tree is of the same order as the number of leaves. Repeating this algorithm for each tree and for each background sample gives us $O(TRL)$. □

Note that for the experiments in this paper we used $R = 200$ background samples to produce low variance estimates.

5.4.11 Benchmark evaluation metrics

We used 21 evaluation metrics to measure the performance of different explanation methods. These metrics were chosen to capture practical runtime considerations, desirable properties such as local accuracy and consistency, and a range of different ways to measure feature importance. We considered multiple previous approaches and based these metrics off what we considered the best aspects of prior evaluations [1, 66, 101, 155]. Importantly, we have included three different ways to hide features from the model. One based on mean masking, one based on random sampling under the assumption of feature independence, and one based on imputation. The imputation based metrics attempt to avoid evaluating the model on unrealistic data, but it should be noted that this comes at the cost of encouraging explanation methods to assign importance to correlated features. After extensive consideration, we did not include metrics based on retraining the original model since, while informative, these can produce misleading results in certain situations.

All metrics used to compute comprehensive evaluations of the Shapley value estimation methods we consider are described below (Figure S3, Supplementary Figures S10 and S11). Python implementations of these metrics are available online `https://github.com/suinleelab/treeexplainer-study`. Performance plots for all benchmark results are also available in Supplementary Data 1.

Runtime

Runtime is reported as the time to explain 1,000 predictions. For the sake of efficiency the runtime for each explanation method was measured using 100 random predictions, and then scaled by 10 to represent the time to explain 1,000 predictions. Both the initialization time of each method and the per-prediction time was measured, and only the per-prediction time was scaled.

Algorithm 3 Independent Tree SHAP

procedure INDTREESHAP(x, $refset$, $tree = \{v, a, b, t, r, d\}$)
  ϕ = array of $len(x)$ zeros
  procedure CALCWEIGHT(U, V) ▷ Shapley value weight for a set size and number of features
    return $\frac{U!(V-U-1)!}{V!}$
  procedure RECURSE(j, U, V, $xlist$, $clist$)
    if $v_j \neq internal$ **then** ▷ Calculate possible contributions at leaf
      $pos = neg = 0$
      if $U == 0$ **then return** (pos, neg)
      if $U \neq 0$ **then** $pos = calcweight(V, U-1) * v_j$
      if $U \neq V$ **then** $neg = -calcweight(V, U) * v_j$
      return (pos, neg)
    $k = None$ ▷ Represents the next node
    if $(x_{d_j} > t_j)$ and $(c_{d_j} > t_j)$ **then** $k = b_j$ ▷ Both x and c go right
    if $!(x_{d_j} > t_j)$ and $!(c_{d_j} > t_j)$ **then** $k = a_j$ ▷ Both x and c go left
    if $xlist_{d_j} > 0$ **then** ▷ Feature was previously x
      if $x_{d_j} > t_j$ **then** $k = b_j$
      else $k = a_j$
    if $clist_{d_j} > 0$ **then** ▷ Feature was previously c
      if $c_{d_j} > t_j$ **then** $k = b_j$
      else $k = a_j$
    if $k \neq None$ **then** ▷ Recurse down a single path if next node is set
      return RECURSE(k, U, V, $xlist$, $clist$)
    if $(x_{d_j} > t_j)$ and $!(c_{d_j} > t_j)$ **then** ▷ Recurse x right and c left
      $xlist_{d_j} = xlist_{d_j} + 1$
      ($posx$,$negx$) = RECURSE(b_j, $U+1$, $V+1$, $xlist$, $clist$)
      $xlist_{d_j} = xlist_{d_j} - 1$
      $clist_{d_j} = clist_{d_j} + 1$
      ($posc$,$negc$) = RECURSE(a_j, U, $V+1$, $xlist$, $clist$)
      $clist_{d_j} = clist_{d_j} - 1$
    if $!(x_{d_j} > t_j)$ and $(c_{d_j} > t_j)$ **then** ▷ Recurse x left and c right
      $xlist_{d_j} = xlist_{d_j} + 1$
      ($posx$,$negx$) = RECURSE(a_j, $U+1$, $V+1$, $xlist$, $clist$)
      $xlist_{d_j} = xlist_{d_j} - 1$
      $clist_{d_j} = clist_{d_j} + 1$
      ($posc$,$negc$) = RECURSE(b_j, U, $V+1$, $xlist$, $clist$)
      $clist_{d_j} = clist_{d_j} - 1$
    $\phi_{d_j} = \phi_{d_j} + posx + negc$ ▷ Save contributions for d_j
    return ($posx + posc$, $negx + negc$) ▷ Pass up both contributions
  for c **in** $refset$ **do**
    RECURSE(0, 0, 0, array of $len(x)$ zeros, array of $len(x)$ zeros)
  return ϕ/len($refset$)

Local accuracy

Local accuracy strictly holds only when the sum of the attribution values exactly sum up from some constant base value to the output of the model for each prediction (Property 4). This means $E_x[(f(x) - \sum_i \phi_i)^2] = 0$. But to also capture how close methods come to achieving local accuracy when they fail, we compute the normalized standard deviation of the difference from the model's output over 100 samples

$$\sigma = \frac{\sqrt{E_x[(f(x) - \sum_i \phi_i)^2]}}{\sqrt{E_x[f(x)^2]}} \tag{5.6}$$

then define nine cutoff levels of σ for reporting a positive score between 0 and 1:

$$\sigma < 10^{-6} \implies 1.00 \tag{5.7}$$
$$10^{-6} \leq \sigma < 0.01 \implies 0.90 \tag{5.8}$$
$$0.01 \leq \sigma < 0.05 \implies 0.75 \tag{5.9}$$
$$0.05 \leq \sigma < 0.10 \implies 0.60 \tag{5.10}$$
$$0.10 \leq \sigma < 0.20 \implies 0.40 \tag{5.11}$$
$$0.20 \leq \sigma < 0.30 \implies 0.30 \tag{5.12}$$
$$0.30 \leq \sigma < 0.50 \implies 0.20 \tag{5.13}$$
$$0.50 \leq \sigma < 0.70 \implies 0.10 \tag{5.14}$$

Consistency guarantees

Consistency guarantees are a theoretical property of an explanation method that ensure pairs of cases will never be inconsistent (Property 5). We broke agreement with this property into three different categories: an exact guarantee, a guarantee that holds in the case of infinite sampling, and no guarantee. Note that while inconsistency could be tested computationally, it would require enumerating a search space exponential in the number of input features, which is why we chose to directly report the theoretical guarantees provided by different methods.

Keep positive (mask)

The Keep Positive (mask) metric measures the ability of an explanation method to find the features that increased the output of the model the most. For a single input the most positive input features are kept at their original values, while all the other input features are masked with their mean value. This is done for eleven different fractions of features ordered by how positive an impact they have as estimated by the explanation method we are evaluating (those that have a negative impact are always removed and never kept). Plotting the fraction of features kept vs. the model output produces a curve that measures how well the local explanation method has identified features that increase the model's output for this prediction. Higher valued curves represent better explanation methods. We average this curve over explanations of 100 test samples for 10 different models trained on different train/test splits. To summarize how well an explanation method performed we take the area under this curve. Masking features and observing their impact on a model's output is a common method for assessing local explanation methods [1, 101, 155]. An example plot of this metric for a random forest model of the chronic kidney disease model is available in Supplementary Figure S22.

Keep positive (resample)

The Keep Positive (resample) metric is similar to the Keep Positive (mask) metric, but instead of replacing hidden features with their mean value, this resample version of the metric replaces them with values from a random training sample. This replacement with values from the training dataset is repeated 100 times and the model output's are averaged to integrate over the background distribution. If the input features are independent then this estimates the expectation of the model output conditioned on the observed features. The mask version of this metric described above can also be viewed as approximating the conditional expectation

of the model's output, but only if the model is linear. The resample metric does not make the assumption of model linearity. An example plot of this metric for a random forest model of the chronic kidney disease model is available in Supplementary Figure S23.

Keep positive (impute)

The Keep Positive (impute) metric is similar to the Keep Positive (mask) and (resample) metrics, but instead of replacing hidden features with their mean value or a sample from the training set, this impute version of the metric replaces them with values imputed based on the data's correlation matrix. The imputations match the maximum likelihood estimate under the assumption that the inputs features follow a multivariate normal distribution. Unlike the mask and resample versions of this metric, the impute version accounts for correlations among the input features. By imputing we prevent the evaluation of the model on invalid inputs that violate the correlations observed in the data (for example having an input where 'hematocrit' is normal but 'anemia' is true). An example plot of this metric for a random forest model of the chronic kidney disease model is available in Supplementary Figure S24.

Keep negative (mask)

The Keep Negative (mask) metric measures the ability of an explanation method to find the features that decreased the output of the model the most. It works just like the Keep Positive (mask) metric described above, but keeps the most negative impacting features as computed by the explanation method (Supplementary Figure S25).

Keep negative (resample)

The Keep Negative (resample) metric measures the ability of an explanation method to find the features that decreased the output of the model the most. It works just like the Keep Positive (resample) metric described above, but keeps the most negative impacting features as computed by the explanation method (Supplementary Figure S26).

Keep negative (impute)

The Keep Negative (impute) metric measures the ability of an explanation method to find the features that decreased the output of the model the most. It works just like the Keep Positive (impute) metric described above, but keeps the most negative impacting features as computed by the explanation method (Supplementary Figure S27).

Keep absolute (mask)

The Keep Absolute (mask) metric measures the ability of an explanation method to find the features most important for the model's accuracy. It works just like the Keep Positive (mask) metric described above, but keeps the most important features as measured by the absolute value of the score given by the explanation method (Supplementary Figure S28). Since removing features by the absolute value of their effect on the model is not designed to push the model's output either higher or lower, we measure not the change in the model's output, but rather the change in the model's accuracy. Good explanations will enable the model to achieve high accuracy with only a few important features.

Keep absolute (resample)

The Keep Absolute (resample) metric measures the ability of an explanation method to find the features most important for the model's accuracy. It works just like the Keep Absolute (mask) metric described above, but uses resampling instead of mean masking (Supplementary Figure S29).

Keep absolute (impute)

The Keep Absolute (impute) metric measures the ability of an explanation method to find the features most important for the model's accuracy. It works just like the Keep Absolute (mask) metric described above, but uses imputing instead of mean masking (Supplementary Figure S30).

Remove positive (mask)

The Remove Positive (mask) metric measures the ability of an explanation method to find the features that increased the output of the model the most. It works just like the Keep Positive (mask) metric described above, but instead of keeping the most positive features, it instead removes them, which should lower the model output (Supplementary Figure S31).

Remove positive (resample)

The Remove Positive (resample) metric measures the ability of an explanation method to find the features that increased the output of the model the most. It works just like the Keep Positive (resample) metric described above, but instead of keeping the most positive features, it instead removes them, which should lower the model output (Supplementary Figure S32).

Remove positive (impute)

The Remove Positive (impute) metric measures the ability of an explanation method to find the features that increased the output of the model the most. It works just like the Keep Positive (impute) metric described above, but instead of keeping the most positive features, it instead removes them, which should lower the model output (Supplementary Figure S33).

Remove negative (mask)

The Remove Negative (mask) metric measures the ability of an explanation method to find the features that decreased the output of the model the most. It works just like the Keep Negative (mask) metric described above, but instead of keeping the most negative features, it instead removes them, which should raise the model output (Supplementary Figure S34).

Remove negative (resample)

The Remove Negative (resample) metric measures the ability of an explanation method to find the features that decreased the output of the model the most. It works just like the Keep Negative (resample) metric described above, but instead of keeping the most negative features, it instead removes them, which should raise the model output (Supplementary Figure S35).

Remove negative (impute)

The Remove Negative (impute) metric measures the ability of an explanation method to find the features that decreased the output of the model the most. It works just like the Keep Negative (impute) metric described above, but instead of keeping the most negative features, it instead removes them, which should raise the model output (Supplementary Figure S36).

Remove absolute (mask)

The Remove Absolute (mask) metric measures the ability of an explanation method to find the features most important for the model's accuracy. It works just like the Keep Absolute (mask) metric described above, but instead of keeping the most important features, it instead removes them, which should lower the model's performance (Supplementary Figure S37).

Remove absolute (resample)

The Remove Absolute (resample) metric measures the ability of an explanation method to find the features most important for the model's accuracy. It works just like the Keep Absolute (resample) metric described above, but instead of keeping the most important features, it instead removes them, which should lower the model's performance (Supplementary Figure S38).

Remove absolute (impute)

The Remove Absolute (impute) metric measures the ability of an explanation method to find the features most important for the model's accuracy. It works just like the Keep Absolute (impute) metric described above, but instead of keeping the most important features, it instead removes them, which should lower the model's performance (Supplementary Figure S39).

5.4.12 User study experiments

Here we explore how consistent different attribution methods are with human intuition. While most complex models cannot be easily explained by humans, very simple decision trees can be explained by people. This means we can run user study experiments that measure how people assign credit to input features for simple models that people understand, then compare people's allocations to allocations given by different local explanation methods. If an explanation method gives the same result as humans, then we say that method is consistent with human intuition.

To test human intuition, we use four simple models and ask 33 english speaking individuals in the U.S. on Amazon Mechanical Turk to explain three different samples for each model. Each model is a simple depth-two decision tree that only depends on two binary features. To make the model more approachable we called the model a "sickness score", and used three binary input features: *fever*, *cough*, and *headache*. The models we used were: *AND*, *OR*, *XOR*, and *SUM* (study participants were not told these names). The headache feature was never used by any of the models. The fever and cough features always each contributed a linear effect of +2 when they were on, but for models other than SUM there were also non-linear effects. For AND +6 was given when both features were true. For OR +6 was given when either feature was true. For XOR +6 was given when either feature was true but not both. For each model we explained three samples: 1) fever false, cough false, headache true; 2) fever false, cough true, headache true; and 3) fever true, cough true, headache true.

Users were asked to allocate blame for a sickness score output value among each of the three input features. No constraints were placed on the input fields used to capture user input, except that the fields were not left blank when there was a non-zero sickness score. This experiment resulted in twelve different consensus credit allocations (Supplementary Figures S40-S43). We then took these twelve consensus credit allocations and used them as ground truth for twelve metrics. Each metric is the sum of the absolute differences between the human consensus feature attribution and attribution given by an explanation method (Supplementary Figures S44-S46). Note that we used a healthy population with independent input features as the background reference dataset (the background is used for computing conditional expectations by the explanation methods).

The results show that all the local explanation methods based on Shapley values agree with the human consensus feature attribution values across all twelve cases. However, the heuristic Saabas method differs significantly from the human consensus in several of the nonlinear cases (Supplementary Figure S12). Not only do Shapley values have attractive theoretical guarantees, and strong quantitative performance (Figure S3), but these experiments show they also match human intuition on small example models (Supplementary Figure S12). Python implementations of these study scenarios are available online `https://github.com/suinleelab/treeexplainer-study`. Performance plots for all user study results are also available in Supplementary Data 1.

5.4.13 SHAP interaction values

Here we describe the new richer explanation model we proposed to capture local interaction effects; it is based on the Shapley interaction index from game theory. The Shapley interaction index is a more recent concept

than the classic Shapley values, and follows from generalizations of the original Shapley value properties [50]. It can allocate credit not just among each player of a game, but among all pairs of players. While standard feature attribution results in a vector of values, one for each feature, attributions based on the Shapley interaction index result in a matrix of feature attributions. The main effects are on the diagonal and the interaction effects on the off-diagonal. If we use the same definition of f_x that we used to get Sabaas values and SHAP values, but with the Shapley interaction index, we get *SHAP interaction values* [50], defined as:

$$\Phi_{i,j}(f,x) = \sum_{S \subseteq \mathcal{M} \setminus \{i,j\}} \frac{|S|!(M-|S|-2)!}{2(M-1)!} \nabla_{ij}(f,x,S), \tag{5.15}$$

when $i \neq j$, and

$$\nabla_{ij}(f,x,S) = f_x(S \cup \{i,j\}) - f_x(S \cup \{i\}) - f_x(S \cup \{j\}) + f_x(S) \tag{5.16}$$

$$= f_x(S \cup \{i,j\}) - f_x(S \cup \{j\}) - [f_x(S \cup \{i\}) - f_x(S)]. \tag{5.17}$$

where $\mathcal{M}$ is the set of all M input features. In Equation 5.15 the SHAP interaction value between feature i and feature j is split equally between each feature so $\Phi_{i,j}(f,x) = \Phi_{j,i}(f,x)$ and the total interaction effect is $\Phi_{i,j}(f,x) + \Phi_{j,i}(f,x)$. The main effects for a prediction can then be defined as the difference between the SHAP value and the off-diagonal SHAP interaction values for a feature:

$$\Phi_{i,i}(f,x) = \phi_i(f,x) - \sum_{j \neq i} \Phi_{i,j}(f,x) \tag{5.18}$$

We then set $\Phi_{0,0}(f,x) = f_x(\emptyset)$ so $\Phi(f,x)$ sums to the output of the model:

$$\sum_{i=0}^{M} \sum_{j=0}^{M} \Phi_{i,j}(f,x) = f(x) \tag{5.19}$$

While SHAP interaction values could be computed directly from Equation 5.15, we can leverage Algorithm 2 to drastically reduce their computational cost for tree models. As highlighted in Equation 5.17, SHAP interaction values can be interpreted as the difference between the SHAP values for feature i when feature j is present and the SHAP values for feature i when feature j is absent. This allows us to use Algorithm 2 twice, once while ignoring feature j as fixed to present, and once with feature j absent. This leads to a run time of $O(TMLD^2)$ when using Tree SHAP, since we repeat the process for each feature. A full open-source implementation is available online `https://github.com/suinleelab/treeexplainer-study`.

SHAP interaction values have uniqueness guarantees similar to SHAP values [50], and allow the separate consideration of main and interaction effects for individual model predictions. This separation can uncover important interactions captured by tree ensembles that might otherwise be missed (Section 5.2.7). While previous work has used global measures of feature interactions [106, 69], to the best of our knowledge SHAP interaction values represent the first local approach to feature interactions beyond simply listing decision paths.

5.4.14 Model summarization experiments

Here, we describe in more detail the model summarization results introduced in Section 5.2.7. One of the most basic ways to understand a model is to display the global importance of each feature, often as a bar chart (Figure S4A left). For tree-based models such as gradient boosted trees a basic operation supported by any implementation is providing the total "Gain" over all splits for a feature as a global measure of feature importance [47]. Computing SHAP values across a whole dataset, we can improve on this basic task of displaying global feature importance in two ways: 1) By averaging the SHAP values across a dataset, we can get a single global measure of feature importance that retains the theoretical guarantees of SHAP values. This avoids the troubling inconsistency problems of the classic Gain method (Supplementary Figure S8), and also provides better global feature selection power than either Gain or permutation testing (Supplementary Figure S13). This is particularly important since tree-based models are often used in practice for feature

selection [55, 127]. 2) A limitation of traditional global explanations for trees is that reporting a single number as the measure of a feature's importance conflates two important and distinct concepts: the magnitude of an effect, and the prevalence of an effect. By plotting many local explanations in a beeswarm-style *SHAP summary plot* we can see both the magnitude and prevalence of a feature's effect (Figure S4A right), and by adding color, we can also display the effect's direction.

The value of summary plots based on many local explanations is illustrated in Figure S4A for the NHANES I mortality dataset, where an XGBoost cox proportional hazards model was trained using hyper-parameters optimized on a validation dataset (Methods 5.4.3), its predictions were explained using TreeExplainer, and then compiled into a summary plot. On the left of Figure S4A is a familiar bar-chart, not based on a typical heuristic global measure of feature importance, but on the average magnitude of the SHAP values. On the right of Figure S4A is a set of beeswarm plots where each dot corresponds to an individual person in the study. Each person has one dot for each feature, where the position of the dot on the x-axis corresponds to the impact that feature has on the model's prediction for that person (as measured by the prediction's SHAP value for that feature). When multiple dots land at the same x position they pile up to show density.

Unsurprisingly, the dominating factor for risk of death in the U.S. in the 1970s (which is when the NHANES I data was collected) is age. By examining the top row of the summary plot we can see that a high value for the age feature (red) corresponds to a large increase in the log hazard ratio (i.e., a large positive SHAP value), while a low value for age (blue) corresponds to a large decrease in the log hazard ratio (i.e., a large negative SHAP value). The next most important feature for mortality prediction is sex, with men having about a 0.6 increase in log-hazards relative to women, which corresponds to about 7 years of change in the age feature. Interestingly, the impact of being a man vs. woman on the model's output is not constant across individuals, as can be seen by the spread of the blue and red dots. The differences of effect within the same sex are due to interactions with other features in the model that modulate the importance of sex for different individuals (we explore this in more detail in Methods 5.4.15).

An important pattern in the mortality data revealed by the summary plot, but not by classic global feature importance, is that features with a low global importance can still be some of the most important features for a specific person (Figure S4A). Blood protein, for example, has a low global impact on mortality, as indicated by its small global importance (Figure S4A left). However, for some individuals, high blood protein has a very large impact on their mortality risk, as indicated by the long tail of red dots stretching to the right in the summary plot (Figure S4A right). This trend of rare high magnitude effects is present across many of the features, and always stretches to the right. This reflects the fact that there are many ways to die abnormally early when medical measurements are out of range, but there not many ways to live abnormally longer (since there are no long tails stretching to the left). Summary plots combine many local explanations to provide a more detailed global picture of the model than a traditional list of global feature importance values. In many cases this more detailed view can provide useful insights, as demonstrated in our medical datasets (Figure S4A, Supplementary Figures S18 and S19).

5.4.15 Feature dependence experiments

Here we describe in more detail the feature dependence results introduced in Section 5.2.7. Just as summary plots provide richer information than traditional measures of global feature importance (Figure S4A), dependence plots based on SHAP values can provide richer information than traditional partial dependence plots by combining the local importance of a single feature across many samples (Figure S4B-G). Plotting a feature's value on the x-axis vs. the feature's SHAP value on the y-axis produces a *SHAP dependence plot* that shows how much that feature impacted the prediction of every sample in the dataset.

For the mortality model a SHAP dependence plot reproduces the standard risk inflection point known for systolic blood pressure between 120 mmHg and 140 mmHg [57] (Figure S4B). This highlights the value of a flexible model that is able to capture non-linear patterns, and also shows how interaction effects can have a significant impact on a feature. Standard partial dependence plots capture the general trends, but do not provide any information about the heterogeneity present within people that have the same measured systolic blood pressure (Supplemental Figure S14). Each dot in a SHAP dependence plot represents a person, and the vertical dispersion of the dots is driven by interaction effects with other features in the model. Many different individuals have a recorded blood pressure of 180 mmHg in the mortality dataset, but the impact that measurement has on their log-hazard ratio varies from 0.2 to 0.6 because of other factors that differ

among these individuals. We can color by another feature to better understand what drives this vertical dispersion. Coloring by age in Figure S4B explains most of the dispersion, meaning that early onset high blood pressure is more concerning to the model than late onset high blood pressure.

For the chronic kidney disease (CKD) model a SHAP dependence plot again clearly reveals the previously documented non-linear inflection point for systolic blood pressure risk, but in this dataset the vertical dispersion from interaction effects appears to be partially driven by differences in blood urea nitrogen (Figure S4E). Correctly modeling blood pressure risk is important, since blood pressure control in select CKD populations may delay progression of kidney disease and reduce the risk of cardiovascular events. Lower blood pressure has been found to slow progression of CKD and decrease overall cardiovascular mortality in some studies [176, 143, 156, 136, 149, 8]. For example, long-term follow-up of the Modification of Diet in Renal Disease (MDRD) study suggested that lower systolic blood pressure led to improved kidney outcomes in patients with CKD [149]. The SPRINT trial, which randomized patients to treatment to systolic blood pressure <120 vs. <140 mmHg found that treatment to lower systolic blood pressure was associated with lower risk of cardiovascular disease; though no difference was seen in rates of CKD progression between the treatment groups [57, 25].

5.4.16 Interaction effect experiments

Here we describe in more detail the interaction effect results introduced in Section 5.2.7. As mentioned in Section 5.2.7, using SHAP interaction values we can decompose the impact of a feature on a specific sample into a main effect and interaction effects with other features. SHAP interaction values allow pairwise interaction effects to be measured at an individual sample level. By combining many such samples we can then observe patterns of interaction effects across a dataset.

In the mortality dataset, we can compute the SHAP interaction values for every sample and then decompose the systolic blood pressure dependence plot into two components. One component contains the main effect of systolic blood pressure and interaction effects with features that are not age, and the other component is the (symmetric) SHAP interaction value of systolic blood pressure and age. The main effect plus the interaction effects equals the original SHAP value for a sample, so we can add the y-values of Figure S4C and Figure S4D to reproduce Figure S4B. Interaction effects are visible in dependence plots through vertical dispersion of the samples, and coloring can often highlight patterns likely to explain this dispersion, but it is necessary to compute the SHAP interaction values to confirm the causes of vertical dispersion. In the systolic blood pressure dependence plot from the mortality model the vertical dispersion is primarily driven by an interaction with age (Figure S4D), as suggested by the original coloring (Figure S4B).

Plotting interaction values can reveal interesting relationships picked up by complex tree-ensemble models that would otherwise be hidden, such as the interaction effect between age and sex described in Section 5.2.7. In the chronic kidney disease model an interesting interaction effect is observed between 'white blood cells' and 'blood urea nitrogen' (Figure S4F). This means that high white blood cell counts are more concerning to the model when they are accompanied by high blood urea nitrogen. Recent evidence has suggested that inflammation may be an important contributor to loss of kidney function [15, 44]. While there are numerous markers of inflammation, white blood cell count is one of the most commonly measured clinical tests available, and this interaction effect supports the notion that inflammation may interact with high blood urea nitrogen to contribute to faster kidney function decline.

5.4.17 Model monitoring experiments

Here we describe in more detail the model monitoring results introduced in Section 5.2.7. As noted in Section 5.2.7, deploying machine learning models in practice is challenging because they depend on a large range of input features, any of which could change after deployment and lead to degraded performance. Finding problems in deployed models is difficult because the result of bugs is typically not a software crash, but rather a change in an already stochastic measure of prediction performance. It is hard to determine when a change in a model's performance is due to a feature problem, an expected generalization error, or random noise. Because of this, many bugs in machine learning pipelines can go undetected, even in core software at top tech companies [186].

A natural first step when debugging model deployments is to identify which features are causing problems. Computing the SHAP values of a model's loss function directly supports this by decomposing the loss among the model's input features. This has two important advantages over traditional model loss monitoring: First, it assigns blame directly to the problematic features so that instead of looking at global fluctuations of model performance, one can see the impact each feature has on the performance. Second, by focusing on individual features we have higher power to identify problems that would otherwise be hidden in all the other fluctuations of the overall model loss. Improving our ability to monitor deployed models is an important part of enabling the safe use of machine learning in medicine.

As mentioned in Section 5.2.7, to simulate a model deployment we used the hospital procedure duration prediction dataset. It contains four years of data from two large hospitals. We used the first year of data for training and ran the model on the next three years of data to simulate a deployment. This is a simple batch prediction task, and so is far less prone to errors than actual real-time deployments, yet even here we observed dataset issues that had not been previously detected during data cleaning (Figure S5).

Figure S5A shows the smoothed loss of the procedure duration model over time, and represents the type of monitoring used widely in industry today. There is a clear increase in the model's error once we switch to the test set, which was expected; then there are short spikes in the error that are hard to differentiate from random fluctuations. To test the value of using monitoring plots based on local explanations we intentionally swapped the labels of operating rooms 6 and 13 two-thirds of the way through the dataset. This is meant to represent the type of coding change that can often happen during active development of a real-time machine learning pipeline. If we look at the overall loss of the model's predictions two-thirds of the way through we see no indication that a problem has occurred (Figure S5A), which means this type of monitoring would not be able to catch the issue. In contrast, the *SHAP monitoring plot* for the room 6 feature clearly shows when the room labeling error begins (Figure S5B). The y-axis of the SHAP monitoring plot is the impact of the room 6 feature on the loss. About two-thirds of the way through the data we see a clear shift from negative values (meaning using the feature helps accuracy) to positive values (meaning using the feature hurts accuracy). The impact on accuracy is substantial, but because procedures occurring in room 6 and 13 are just a small fraction of the overall set of procedures, the increased error is invisible when looking at the overall loss (Figure S5A).

In addition to finding our intentional error, SHAP monitoring plots also revealed problems that were already present in the dataset. Two of these are shown Figure S5C and D.

In Figure S5C we can see a spike in error for the general anesthesia feature shortly after the deployment window begins. This spike represents a transient configuration issue in our hospital revealed by a SHAP monitoring plot. It corresponds to a subset of procedures from a single hospital where the anesthesia type data field was left blank. After going back to the hospital system we found that this was due to a temporary electronic medical record configuration issue whereby the link between the general anesthesia induction note and the anesthesia type got broken. This prevented the anesthesia type data field from being auto-populated with "general anesthesia" when the induction note was completed. This is exactly the type of configuration issue that impacts machine learning models deployed in practice, and so needs to be detected during model monitoring.

In Figure S5D we see an example, not of a processing error, but of feature drift over time. The atrial fibrillation feature denotes if a patient is undergoing an atrial fibrillation ablation procedure. During the training period, and for some time into deployment, using the atrial fibrillation feature lowers the loss. But then over the course of time the feature becomes less and less useful until it begins hurting the model by the end of the deployment window. Based on the SHAP monitoring plot for atrial fibrillation we went back to the hospital to determine possible causes. During an atrial fibrillation ablation, the goal is to electrically isolate the pulmonary veins of the heart using long catheters placed from the groin area. Traditionally, the procedure was completed with a radiofrequency ablation catheter delivering point-by-point lesions to burn the left atrium around the pulmonary veins. During the deployment window, the hospital began to use the second generation cryoballoon catheter (Arctic Front Advance, Medtronic Inc., Minneapolis, MN), which freezes the tissue and has been demonstrated to have a shorter procedure duration compared to radiofrequency ablation [85]. At the same time, there were improvements in radiofrequency ablation catheter technology including the use of contact force sensing which allowed the operator to determine how strongly the catheter was touching the left atrial wall. This technology ensures that ablation lesions are delivered with enough force to create significant lesion size. With noncontact force catheters, the operator may think the catheter is touching the

atrial wall but it may, in actuality, simply be floating nearby. Contact force sensing is also associated with shorter procedure times [110, 81]. Cryoballoon versus radiofrequency ablation is chosen based on patient characteristics and physician preference. Lastly, during this time there were staffing changes including the use of specialized electrophysiology technologists which decreased procedural preparation time. All of these changes led to a significant decrease in atrial fibrillation ablation procedural and in-room time during the test period (Supplementary Figure S15), which translated into high model error attributable to the atrial fibrillation feature. We quantified the significance of the SHAP monitoring plot trend using an independent t-test between atrial fibrillation ablation procedures that appear in the first 30,000 samples of the test time period vs. those that appear in the last 60,000 samples of the test time period. This lead to a P-value of 5.4×10^{-19}. The complexity of the reasons behind the feature drift in Figure S5D illustrate why it is so difficult to anticipate how assumptions depended on by a model might break in practice. Using SHAP values to monitor the loss of a model allow us to retroactively identify how model assumptions may be changing individually for each input feature, even if we cannot a priori anticipate which features are likely to cause problems.

Explaining the loss of a model is not only useful for monitoring over time, it can also be used in dependence plots to understand how a specific feature helps improve model performance. For hospital procedure duration prediction we can plot the dependence between the time of the day feature and it's impact on the model's output. The time of day feature indicates when a procedure started, and is particularly effective at reducing the model's loss just after 7:30am and 8:30am in the morning (Supplementary Figure S21). These two times are when long-running elective surgeries are scheduled in this hospital system, so the dependence plot reveals that the model is using the time of the day to detect routine long-running surgeries.

Current practice is to monitor the overall loss of the model, and also potentially monitor the statistics of the features for changes over time. This is problematic because just monitoring the overall loss of the model can hide many important problems, while monitoring the changes in the statistics of features is essentially an unsupervised anomaly detection problem that is prone to both false positives and false negatives. Swapping the names of operating rooms that are used equally often would be invisible to such an unsupervised method. By directly attributing the loss to the features we can highlight precisely which features are impacting the loss and by how much. When changes show up in the monitoring plot, those changes are in the units of the model's loss and so we can quantify how much it is impacting our performance. These monitoring plots represent a compelling way that many local explanations can be combined to provide richer and more actionable insights into model behavior than the current state of the art.

5.4.18 Local explanation embedding experiments

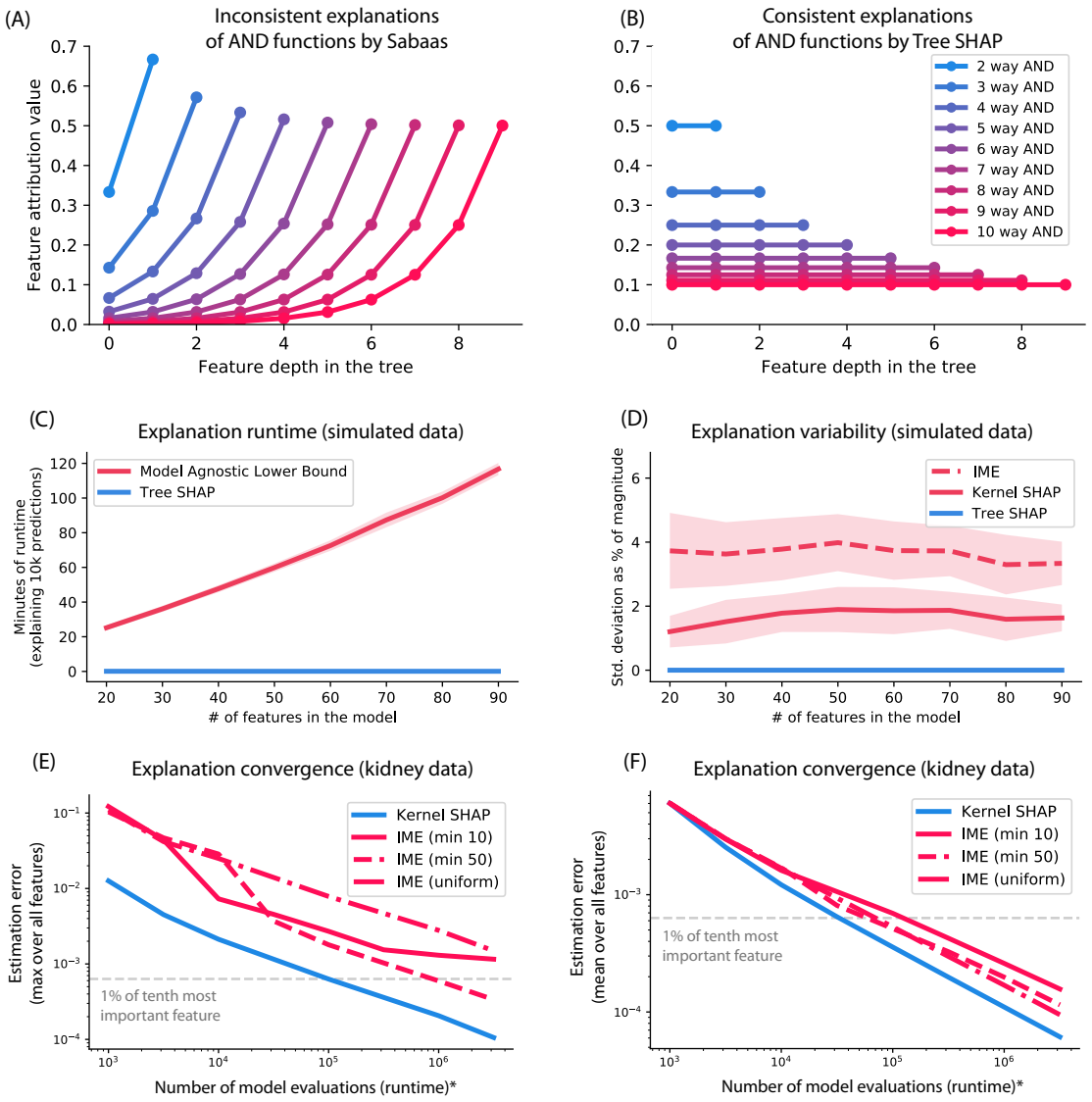
Here we describe in more detail the local explanation embedding results introduced in Section 5.2.7. There are two challenges with standard unsupervised clustering methods: 1) The distance metric does not account for the discrepancies among the units and meaning of features (e.g., units of years vs. units of cholesterol), and simple standardization is no guarantee the resulting numbers are comparable. 2) Even after a distance metric is defined, there is no way for an unsupervised approach to know which features are relevant for an outcome of interest, and so should be weighted more strongly. Some applications might seek to cluster patients by groups relating to kidney disease, and another by groups relating to diabetes. But given the same feature set, unsupervised clustering will give the same results in both cases.

As mentioned in Section 5.2.7, we can address both of the above problems in traditional unsupervised clustering by using local explanation embeddings to embed each sample into a new "explanation space." If we then run clustering in this new space, we will get a *supervised clustering* where samples are grouped together that have the same model output for the same reason. Since SHAP values have the same units as the model's output they are all comparable within a model, even if the original features were not comparable. Supervised clustering naturally accounts for the differing scales of different features, only highlighting changes that are relevant to a particular outcome. Running hierarchical supervised clustering using the mortality model results in groups of people that share a similar mortality risk for similar reasons (Figure S6A). The heatmap in Figure S6A uses the leaf order of a hierarchical agglomerative clustering based on the SHAP values of each sample. On the left are young people, in the middle are middle aged people, and on the right are older people. Within these broad categories many smaller groups are of particular interest, such as people with early onset high-blood pressure, young people with inflammation markers, and underweight older people. Each of these

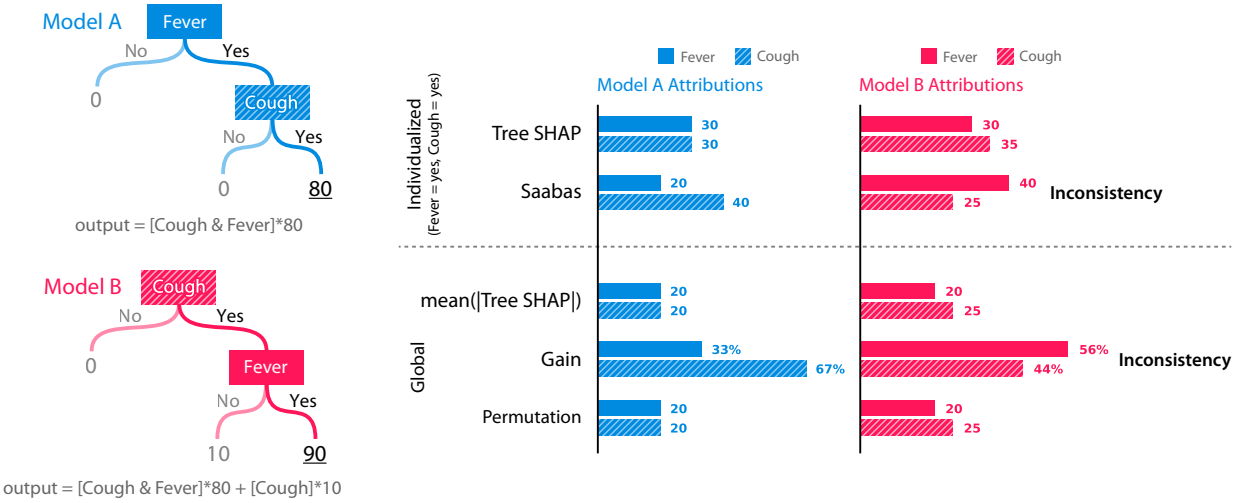
recapitulate known high risk groups of people, which would not have been captured by a simple unsupervised clustering approach (Supplementary Figure S20).

In addition to making explicit clusters, we can also use dimensionality reduction to directly visualize the explanation space embedding produced by the SHAP values. This gives a continuous representation of the primary directions of model output variation in the dataset. For the kidney disease dataset, the top two principal components of the explanation embedding highlight two distinct risk factors for disease progression (Figures S6B-D). The first principal component aligns with blood creatinine levels, which are used to compute the estimated glomerular filtration rate (eGFR) of the kidneys. High levels of creatinine are a marker of lower eGFR and are the primary test for detection of kidney disease. The second principal component aligns with higher urine protein concentration in the urine. Quantified as the urine protein to urine creatinine ratio (PCR), this is a marker of kidney damage and is used in conjunction with eGFR to quantify levels of kidney disease. If we color the explanation space embedding by the risk of kidney disease progression, we see a roughly continuous increase in risk from left to right (Figure S6B). This is largely explained by a combination of the two orthogonal risk directions described above: One direction follows the blood creatinine level feature (which determine the eGFR) (Figure S6C) and the other direction follows the urine protein feature (Figure S6D). Several of the other top features in the chronic kidney disease model also align with these two orthogonal embedding directions (Supplementary Figure S6B-D). It is well established that eGFR and urine PCR are the strongest predictors of progression to end-stage renal disease among patients with chronic kidney disease [113, 28]. Physiologically, eGFR and PCR are likely complimentary, yet distinct kidney disease markers. Figure S6B-D shows that eGFR and PCR each identify unique individuals at risk of end-stage renal disease; thus confirming that clinically they should be measured in parallel. This type of insight into the overall structure of kidney risk is not at all apparent when just looking at a standard unsupervised embedding (Supplementary Figure S17).

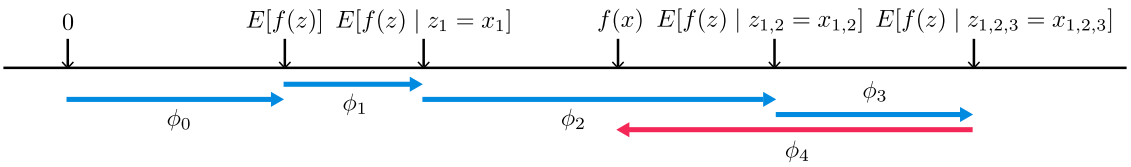
Supplementary Figures



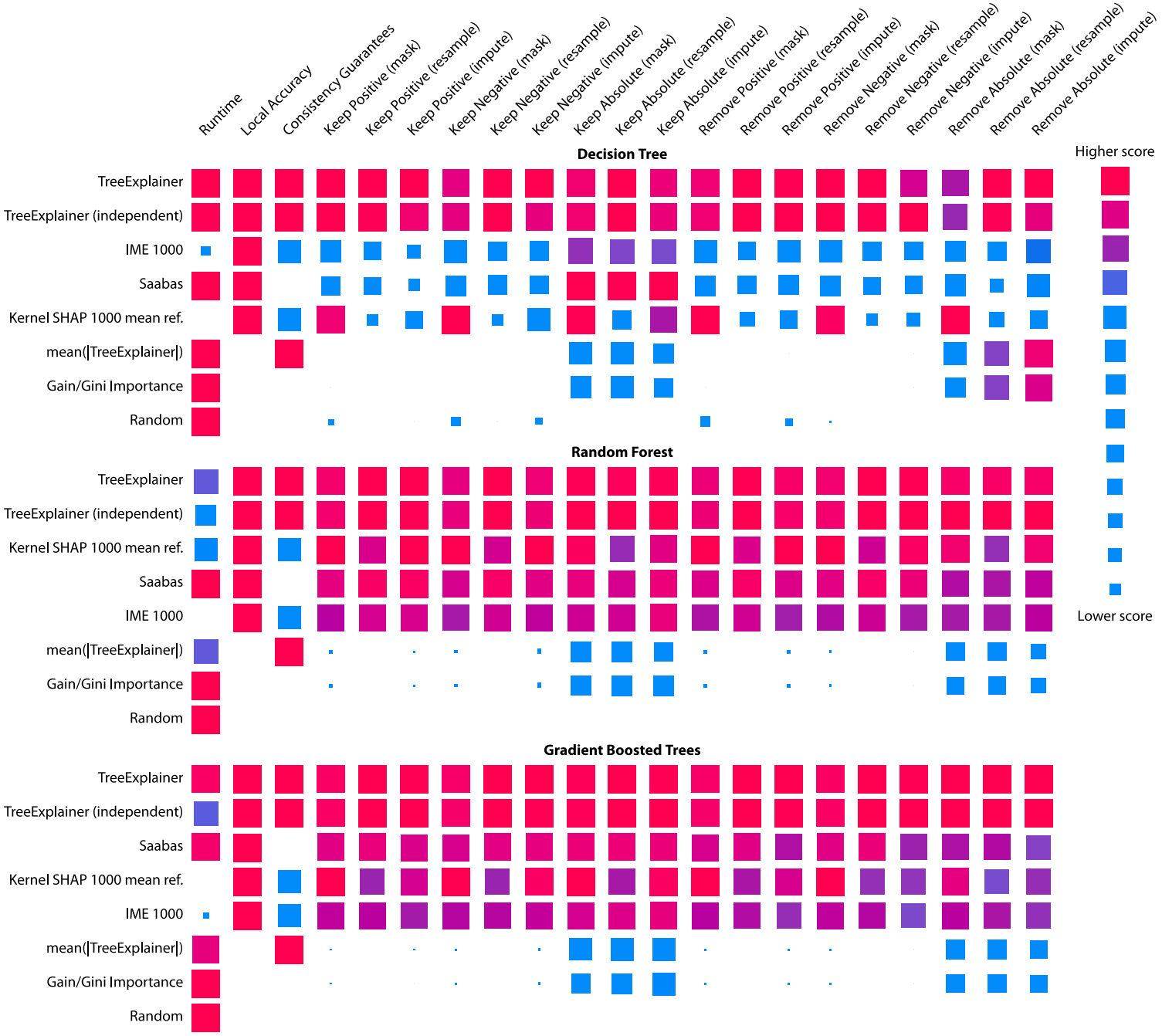
Supplementary Figure S7: **(A-B) TreeExplainer avoids the consistency problems of previous tree-specific approaches.** For tree models that represent a multi-way AND function the Saabas method gives little credit to features near the root, while Tree SHAP evenly distributes credit among all the features involved in the AND function. **(C-F) TreeExplainer represents a dramatic performance improvement over model agnostic approaches.** All model agnostic approaches rely on sampling, so their runtime is lower bounded by the run-time to evaluate the model, and they always have some sampling variability. TreeExplainer's Tree SHAP algorithm runs thousands of times faster, and has no sampling variability since it is exact. We consider both the Kernel SHAP [101] and IME [167] model agnostic estimators for Shapley values. (C-D) Using simulated data we can train XGBoost models over datasets of different sizes and observe that the number of samples required to maintain estimates with constant variance grows linearly with the number of features in the model. The reported runtime is a lower bound since it only accounts for the time to execute the model, not execute the explanation method itself. (E-F) As we increase the number of model evaluations used by model agnostic approaches we converge towards the exact solution. However, achieving low variance estimates requires days or years of CPU time even on the smallest of our medical datasets. We also only used a single background reference sample for computing conditional expectations in the model agnostic methods (instead of an entire dataset) so these represent lower bounds on the runtime. The estimation error on the y-axis represents the difference from the exact solution with a single background reference sample. Details of the experimental setup for C-F are described in Methods 5.4.8.



Supplementary Figure S8: **Two simple tree models that demonstrate inconsistencies in the Saabas and gain feature attribution methods.** The Cough feature has a larger impact in Model B than Model A, but is attributed less importance in Model B. Similarly, the Cough feature has a larger impact than Fever in Model B, yet is attributed less importance. The individualized attributions explain a single prediction of the model (when both Cough and Fever are Yes) by allocating the difference between the expected value of the model's output (20 for Model A, 25 for Model B) and the current output (80 for Model A, 90 for Model B). Inconsistency prevents the reliable comparison of feature attribution values. The global attributions represent the overall importance of a feature in the model. "Gain" is the most common way of measuring feature importance in trees and is the sum of the reductions in loss that come from all splits using that feature. "Permutation" is the change in the model's accuracy when a single feature is permuted.

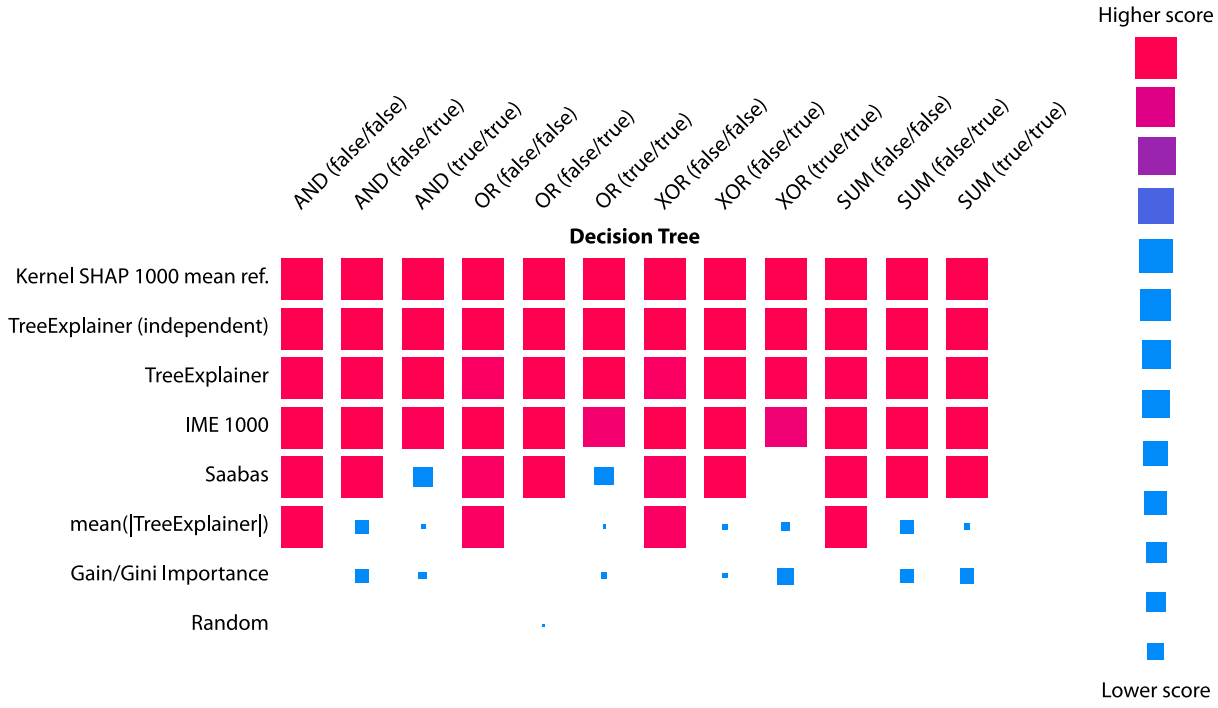


Supplementary Figure S9: The Sabaas values, ϕ_i^s, attribute feature importance by measuring differences in conditional expectations along the order defined by the decision path. This is very similar to SHAP (SHapley Additive exPlanation) values, ϕ_i, except SHAP values result from averaging over all possible orderings. This is important since for non-linear functions the order in which features are introduced matters. Proofs from game theory show that averaging over all orderings is the only possible consistent approach where $\sum_{i=0}^{M} \phi_i = f(x)$ (Methods 5.4.10).

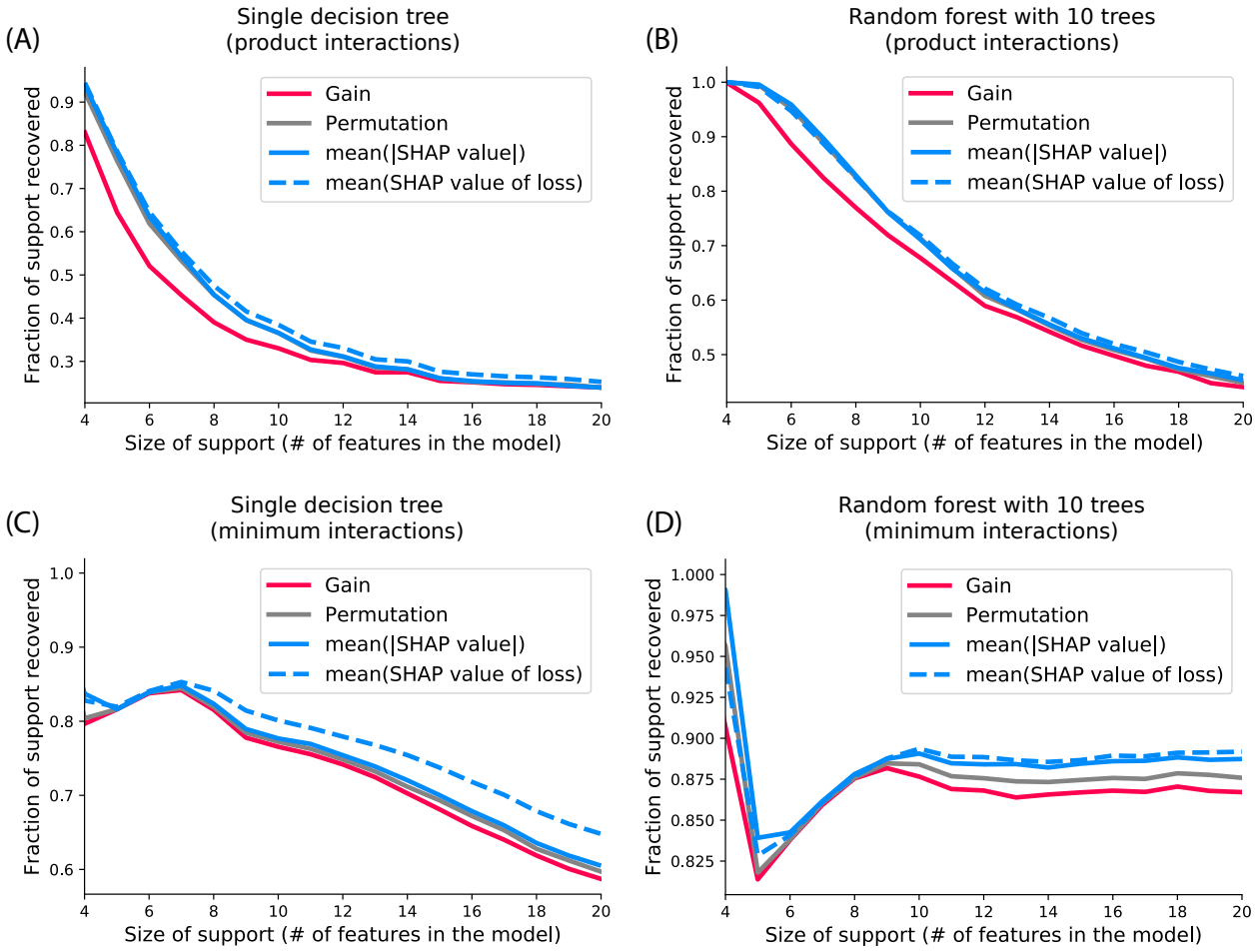


Supplementary Figure S10: **Explanation method performance across thirteen different evaluation metrics and three regression models in a simulated dataset with 60 features divided into tightly correlated groups.** Each tile represents the performance of a feature attribution method on a given metric for a given model. Within each model the columns of tiles are scaled between the minimum and maximum value, and methods are sorted by their overall performance. Some of these metrics have been proposed before and others are new quantitative measures of explanation performance that we are introducing (see Methods).

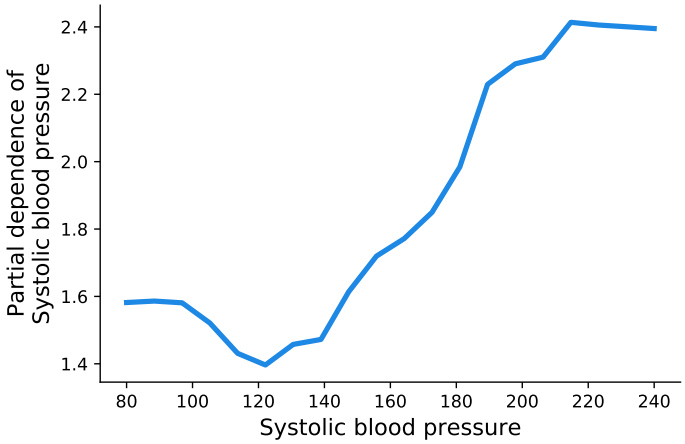
Supplementary Figure S11: **Explanation method performance across thirteen different evaluation metrics and three regression models in a simulated dataset with 60 independent features.** Each tile represents the performance of a feature attribution method on a given metric for a given model. Within each model the columns of tiles are scaled between the minimum and maximum value, and methods are sorted by their overall performance. Some of these metrics have been proposed before and others are new quantitative measures of explanation performance that we are introducing (see Methods).



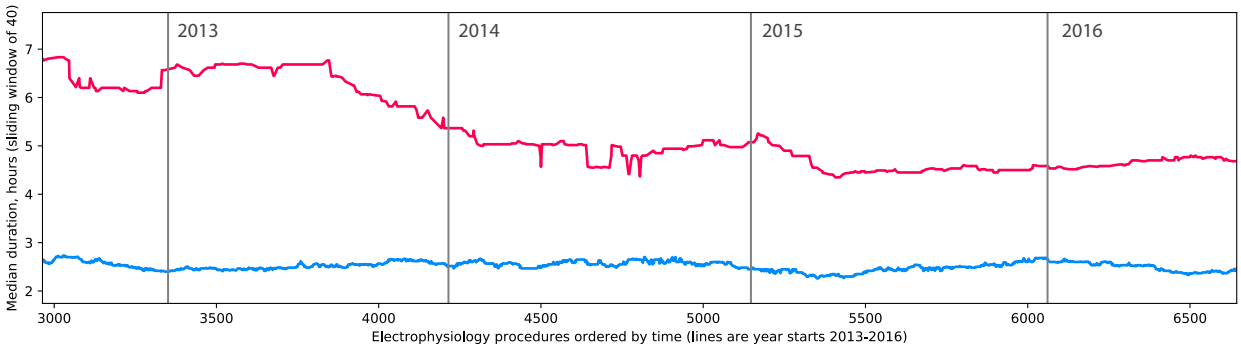
Supplementary Figure S12: **Shapley value based methods agree with human intuition.** We measured the most intuitive way to assign credit among input features by asking users to assign credit for predictions from four simple models (three predictions per model). The consensus allocation observed from a user study was then used as the ground truth and compared with the allocations given by different explanation methods. The sum of absolute differences was used to quantify agreement. All the Shapley value based methods had nearly perfect agreement across all the scenarios. The raw allocations for the cases where Saabas fails are shown in Supplementary Figures S44-S46 (Methods 5.4.12). Note that since these small (human understandable) models have only three features, model agnostic Shapley methods are accurate and so comparable with TreeExplainer.



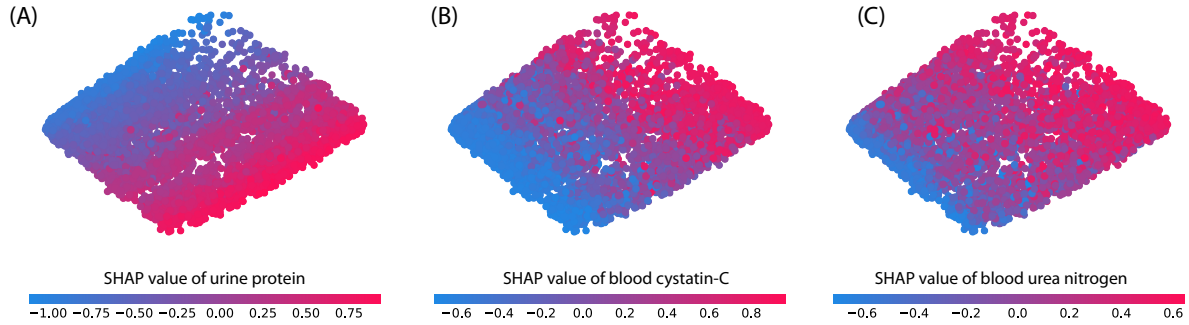
Supplementary Figure S13: **SHAP values can be combined to provide better feature selection power than traditional approaches.** Feature selection methods based on trees or ensembles of trees typically either use the total gain (reduction in train error when splitting) or a permutation test (scramble each feature and observe the change in error) to do feature selection. To compare feature selection power we reproduce the same simulated independent features setting as [74] (but with strong 3rd order interactions) and compare the mean SHAP value of the loss vs. mean SHAP value magnitude vs. gain vs. permutation for feature selection. We also repeat the experiment replacing the product style interactions from [74] with minimum functions. For both a single tree (A,C) and an ensemble of ten trees (B,D) the SHAP values provide a better ranking of features. Perhaps because, unlike gain, they guarantee consistency as defined by Property 5, and unlike permutations, they account for high-order interaction effects. The x-axis of the plots represents the number of features used in the true model (out of 200 total features), while the y-axis represents the fraction of those true features recovered in the set of top ranked features of the same size. Results are averages over performance on 1000 simulated datasets. Both SHAP value based methods outperform gain and permutation testing in every figure, with all paired t-test P-values being $< 10^{-7}$.



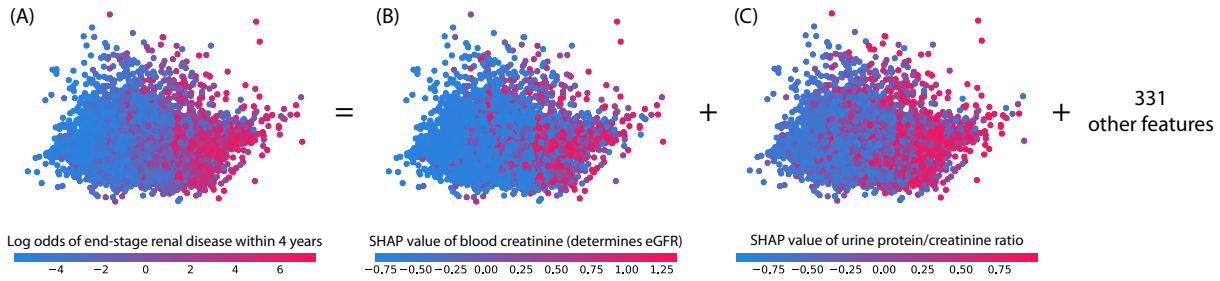
Supplementary Figure S14: **Partial dependence plot of systolic blood pressure in the mortality model.** Unlike the corresponding SHAP dependence plot in Figure S4B, the partial dependence plot gives no indication of the heterogeneity between individuals caused by interaction effects in the model.



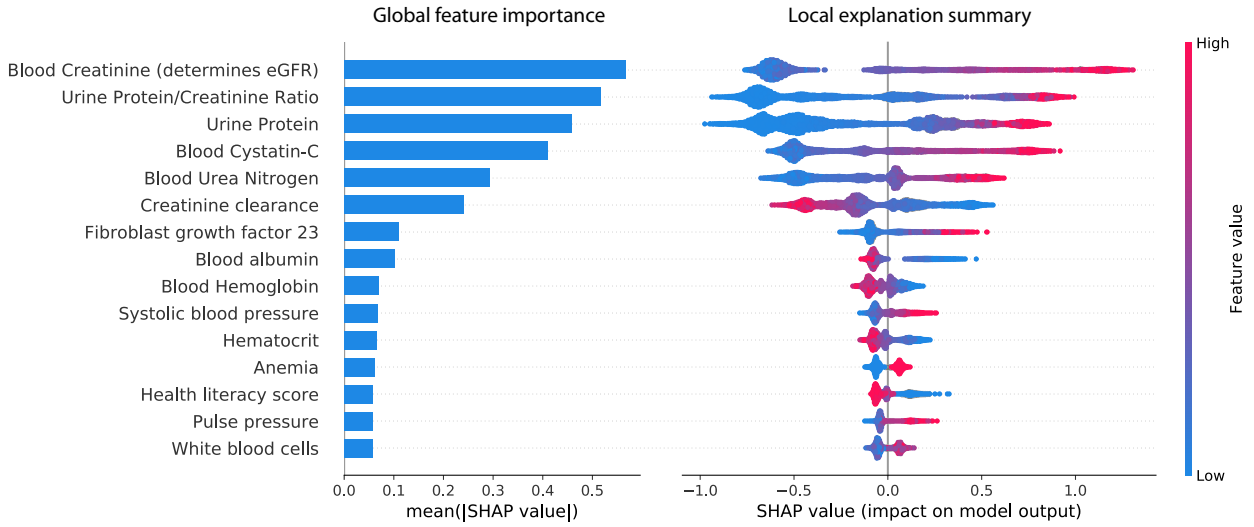
Supplementary Figure S15: **The average duration of ablation procedures for atrial fibrillation dropped significantly during 2014.** This data was obtained directly from the electrophysiology lab to diagnose why the atrial fibrillation feature was found (using a SHAP monitoring plot) to degrade model performance (Figure S5D). The reason is that around 2014 (which was after the simulated model deployment) the duration of these procedures dropped significantly.



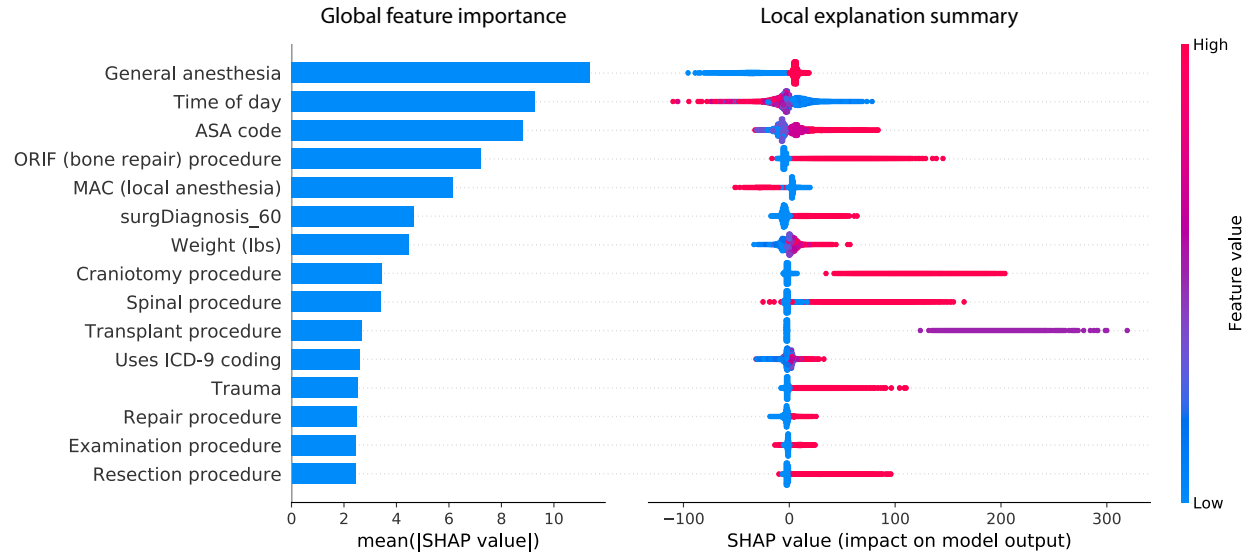
Supplementary Figure S16: **A local explanation embedding of kidney visits projected onto its first two principle components.** This shows the next three top features beyond those shown in Figures S6B-D. The fact that these features also align with the top principal components shows how many of the important features in the data set are capturing information along two largely orthogonal dimensions.



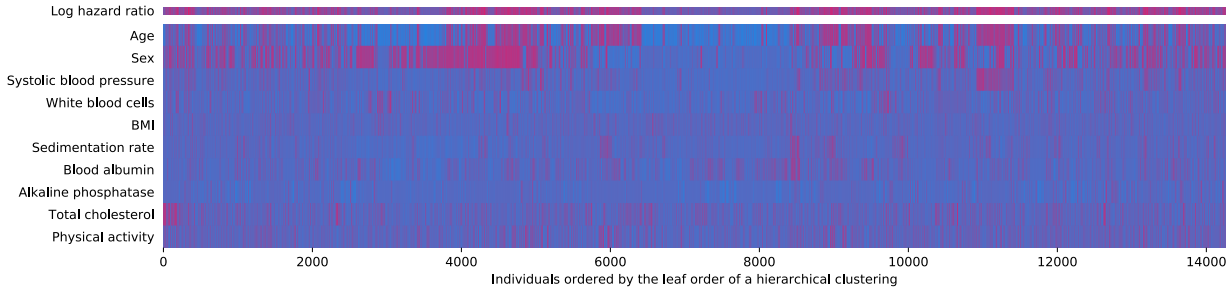
Supplementary Figure S17: **Principle component embedding of the chronic kidney disease dataset.** Unlike an embedding based in the local explanation space (Figures S6B-D), an unsupervised embedding of the data does not necessarily align with the outcome of interest in a dataset.



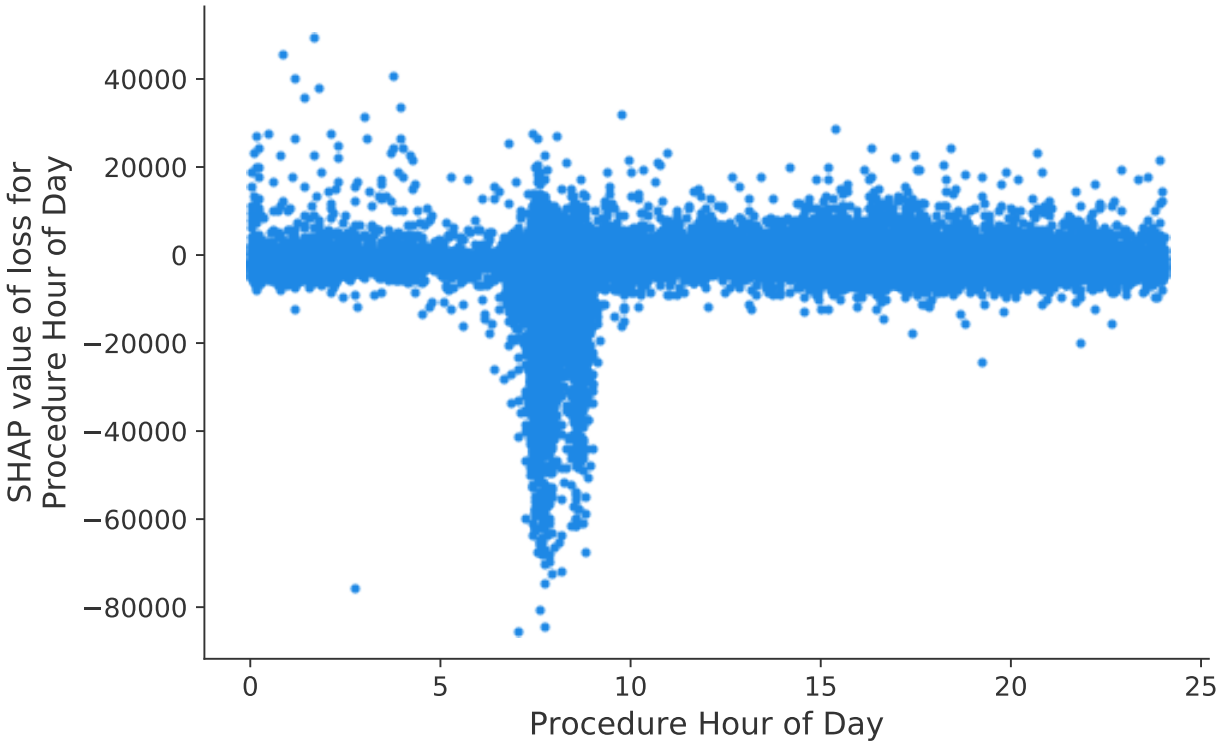
Supplementary Figure S18: **Bar chart (left) and summary plot (right) for a gradient boosted decision tree model trained on the chronic kidney disease data.** For the summary plot red dots indicate a high value of that feature for that individual, while blue dots represent a low feature value. The x-axis is the SHAP value of a feature for each individual's prediction, representing the change in the log-hazard ratio caused by observing that feature. High blood creatinine increases the risk of end-stage kidney disease. Conversely, low creatinine clearance increases the risk of end-stage kidney disease.



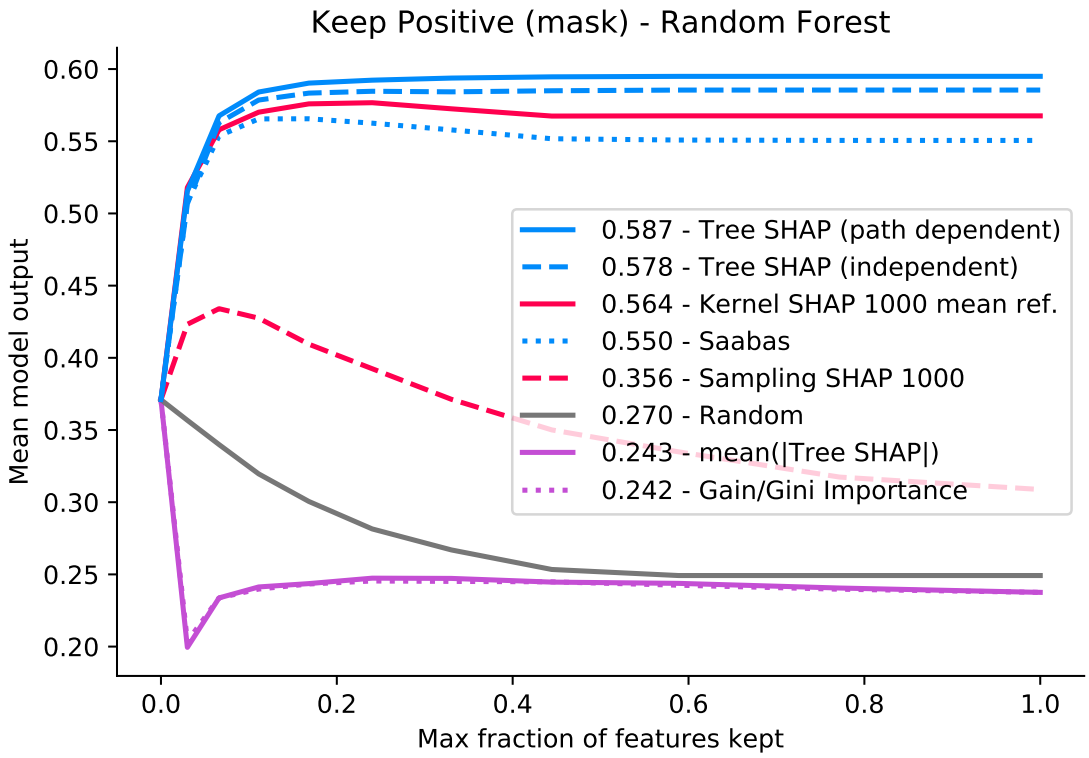
Supplementary Figure S19: **Bar chart (left) and summary plot (right) for a gradient boosted decision tree model trained on the hospital procedure duration data.** For the summary plot red dots indicate a high value of that feature for that individual, while blue dots represent a low feature value. The x-axis is the SHAP value of a feature for each individual's prediction, representing the change in the log-hazard ratio caused by observing that feature. Many of the features are bag-of-words counts that have only a few non-zero values. Because the model is very nonlinear, the impact of a flag being on (such as the trauma flag) can have very different effects for different procedures (as shown for trauma by the horizontal dispersion of the red dots).



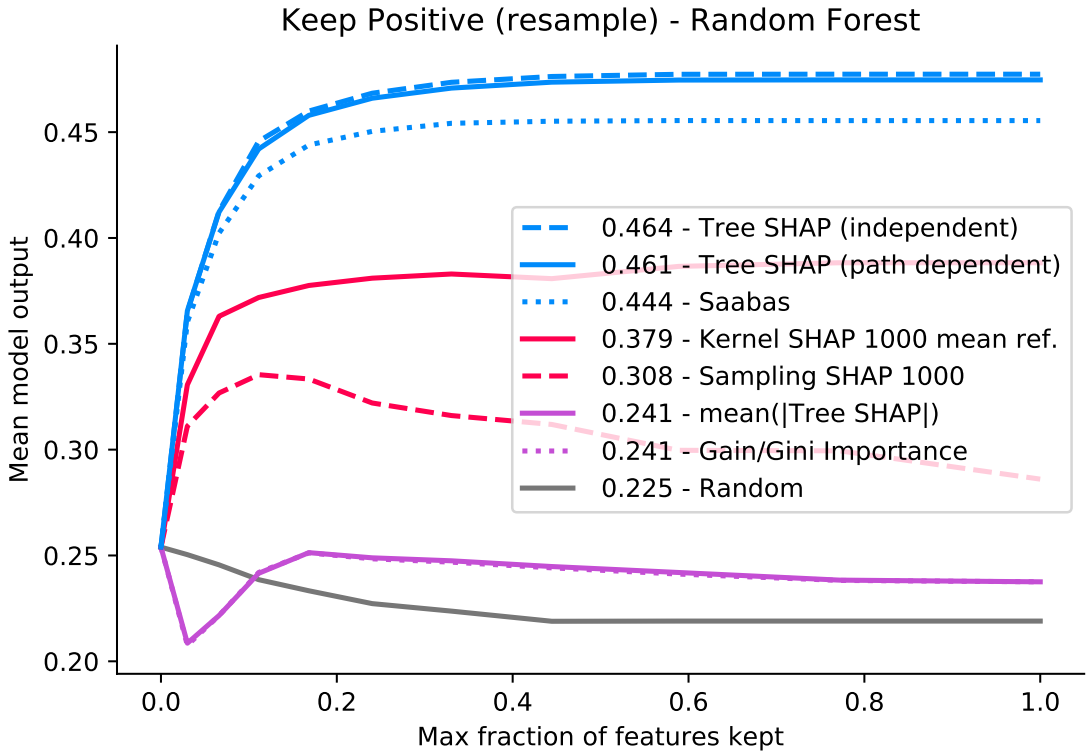
Supplementary Figure S20: **A clustering of 14,264 mortality study individuals by their normalized data values.** Standard complete linkage hierarchical clustering reveals fewer groups consistency with model risk than supervised clustering (Figure S6A). This is because unsupervised clustering has no bias towards clusters that share common risk characteristics. Row-normalized feature SHAP values are used for coloring, as in Figure S6A.



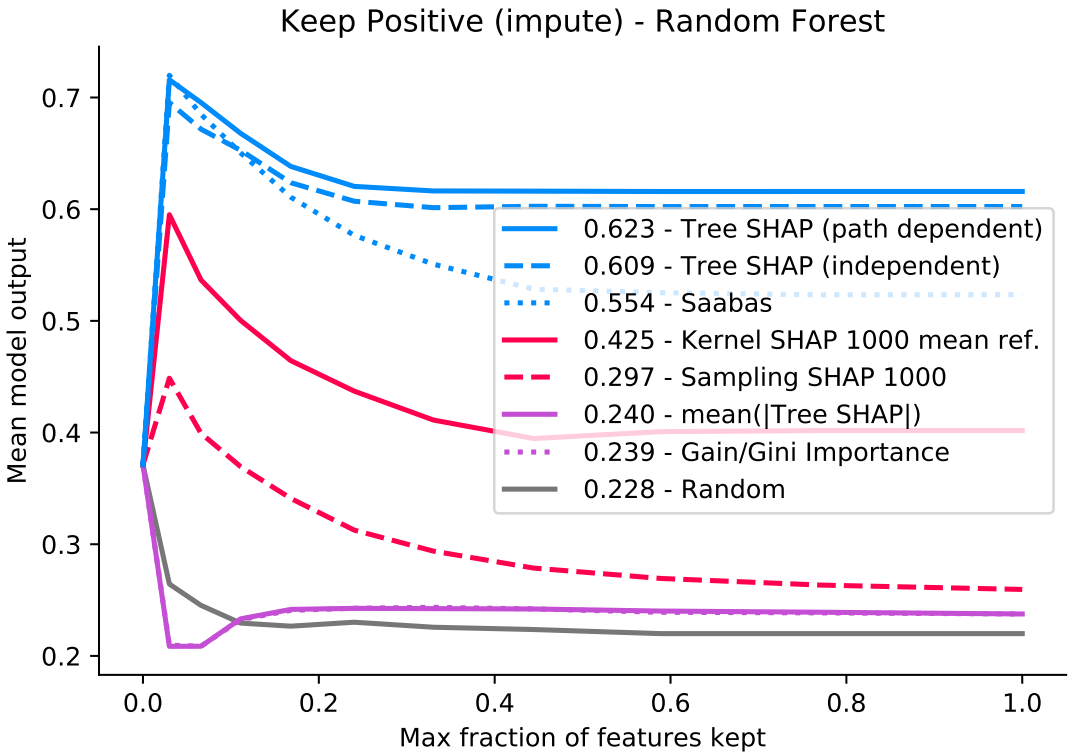
Supplementary Figure S21: **A dependence plot for time of day vs the SHAP value of time of day with respect to the model loss.** In our hospital system long running elective surgeries are scheduled at 7:20-7:30 AM on Monday/Tuesday/Thursday/Friday and at 8:20-8:30 AM on Wednesday. This plot shows that the primary way model is using time of day to reduce the model's loss is by detecting these surgery scheduling times. Each dot is a procedure, the x-axis is the time that procedure was scheduled to begin, and the y-axis is the impact knowing the time of day had on the model's loss for predicting that procedure's duration (lower is better).



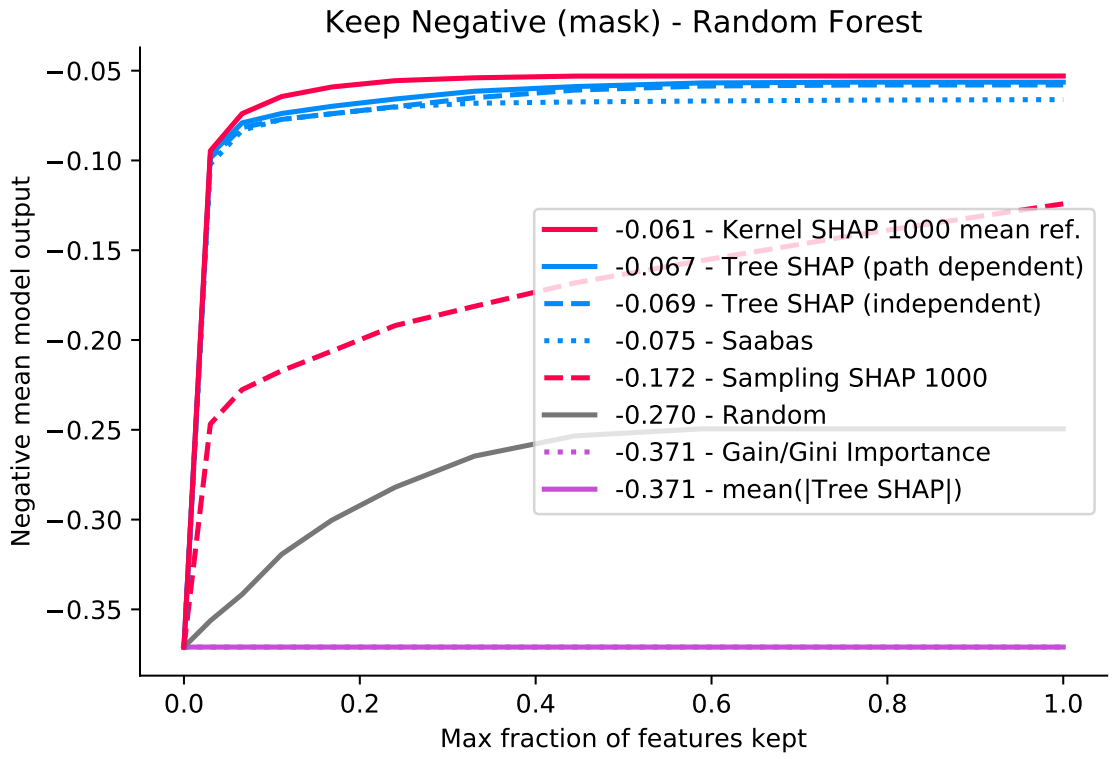
Supplementary Figure S22: **Keep positive (mask) metric for a random forest trained on the chronic kidney disease dataset.** Sorting the attribution values of an explanation method provides an ordering of the features for each prediction made by the model. Here we keep a fraction of the features ordered by how much they increase the model's output. Features that are not kept are masked with their mean value. If the ordering is good, as we include more and more features we push the model's output higher. Note that features with a negative contribution are always masked. The x-axis is the maximum fraction of features kept and the y-axis is the mean increase of the model over 100 predictions (averaged over 10 model's trained on different train/test splits of the dataset). Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



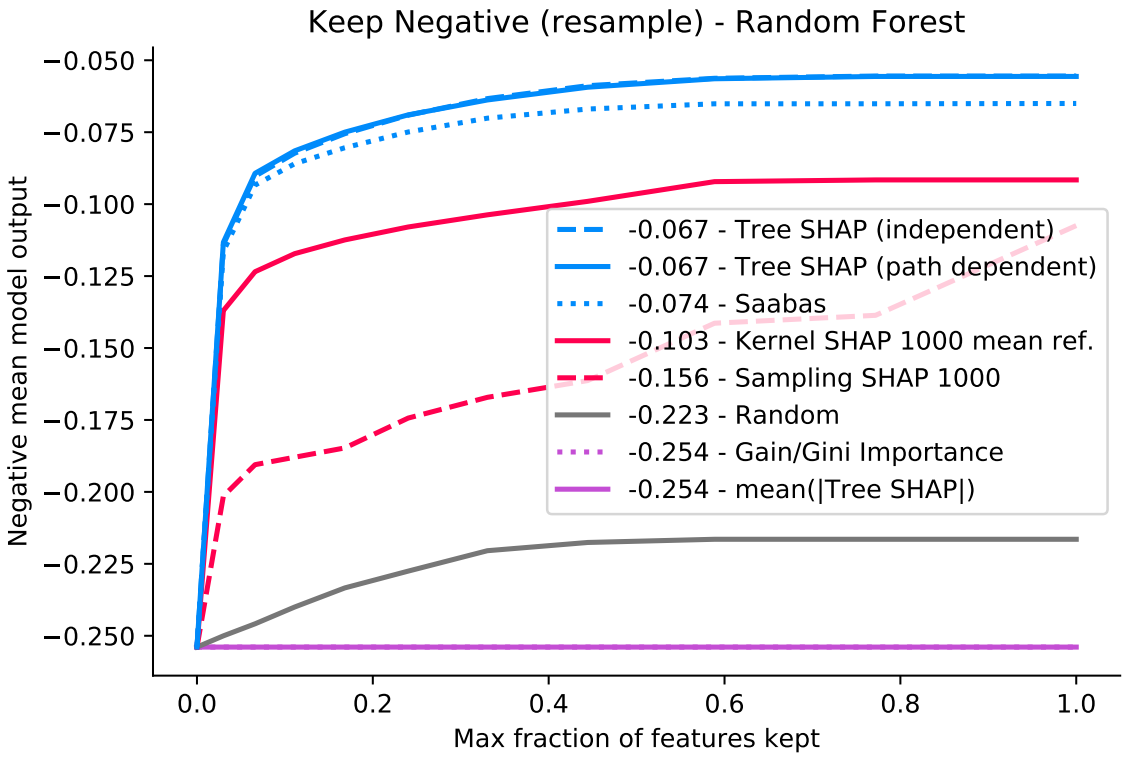
Supplementary Figure S23: **Keep positive (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S22 except that instead of masking the hidden features with their mean value, we instead replace them with a random sample from the training dataset. This resampling process is averaged over 100 times to integrate over the distribution of background samples. If the input features are independent of one another then this effectively computes the conditional expectation of the model output conditioned on only the observed features. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



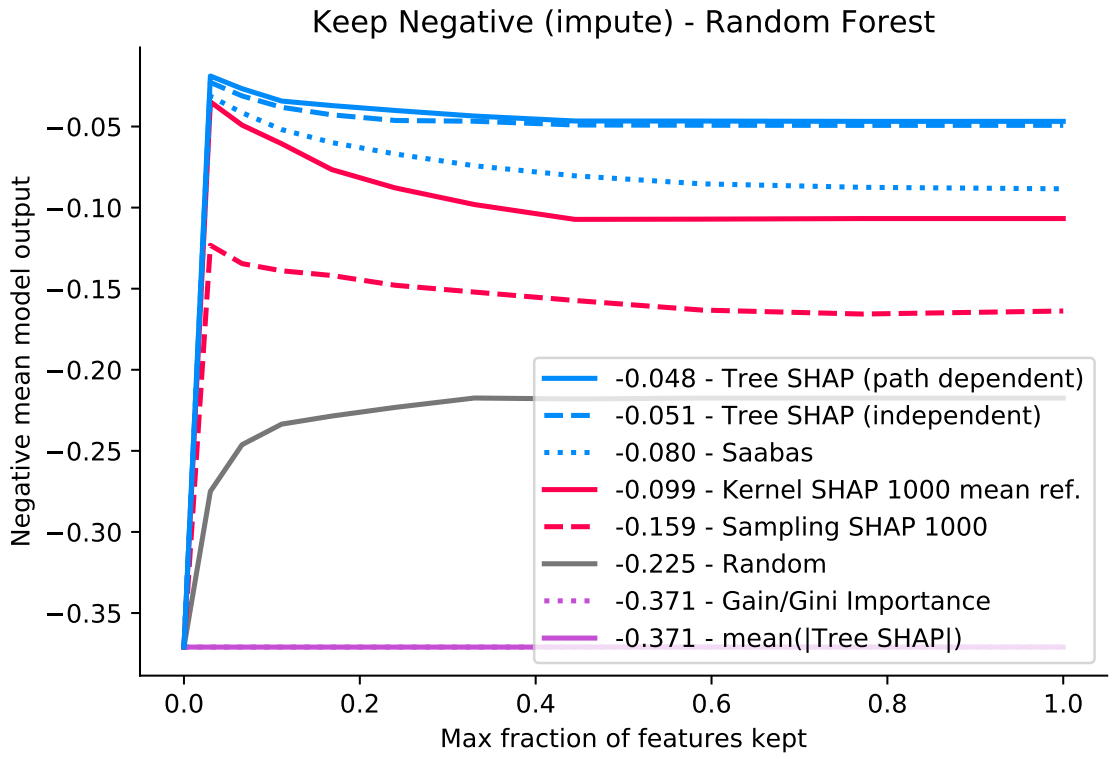
Supplementary Figure S24: **Keep positive (impute) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figures S22 and S23 except that instead of mean masking or resampling the hidden features, we instead impute them using the covariance matrix of the training data (this is maximum likelihood imputation if we assume the input features are multivariate normal). This imputation process seeks to avoid evaluating the model on unrealistic input data. In contrast with mean imputation or resampling we assume the input feature are independent and so many provide unrealistic combinations of input features (such as pregnant men). Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



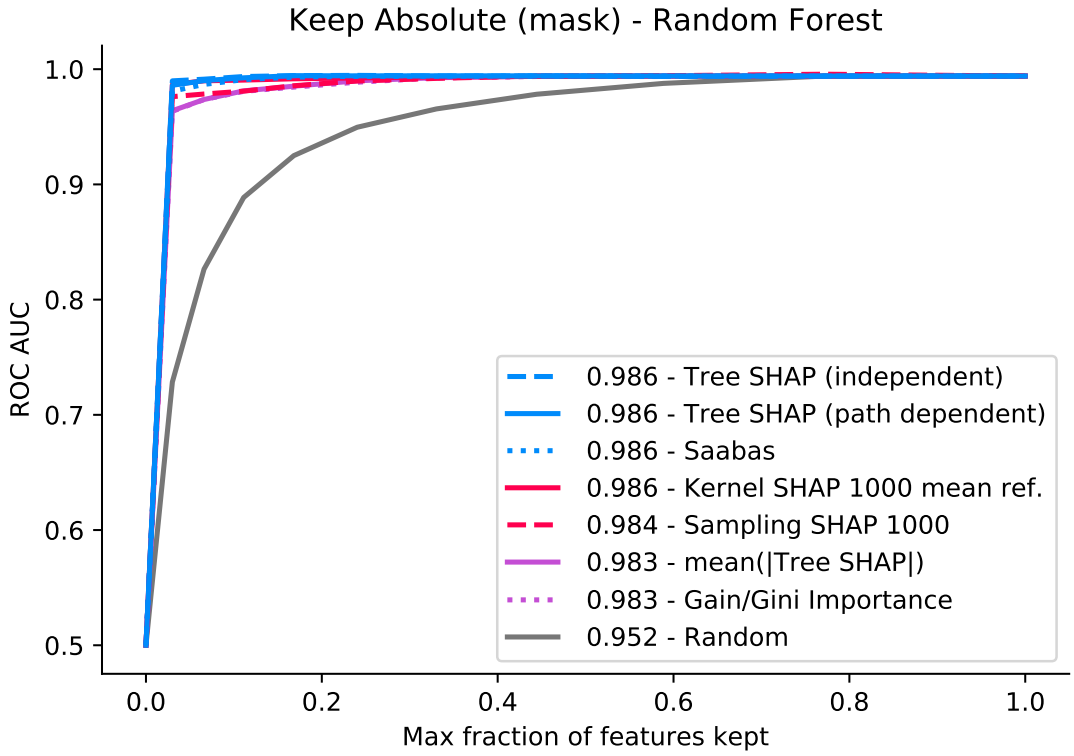
Supplementary Figure S25: **Keep negative (mask) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S22 except that we keep the most negative features instead of the most positive. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



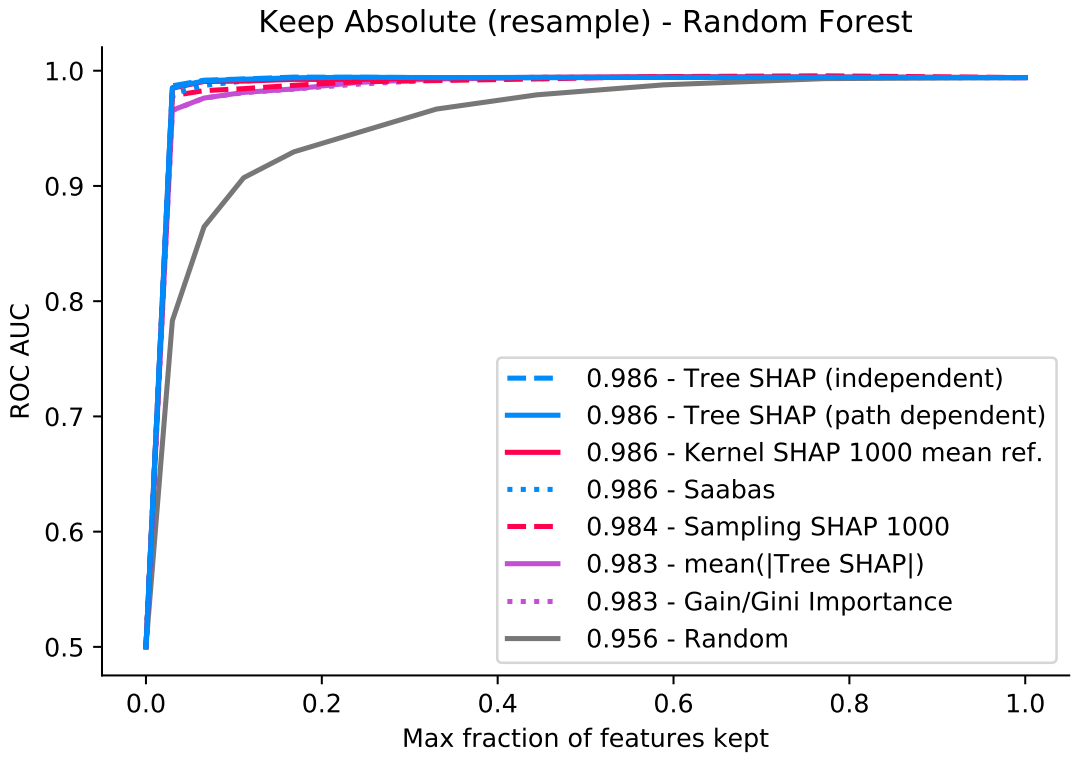
Supplementary Figure S26: **Keep negative (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S23 except that we keep the most negative features instead of the most positive. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



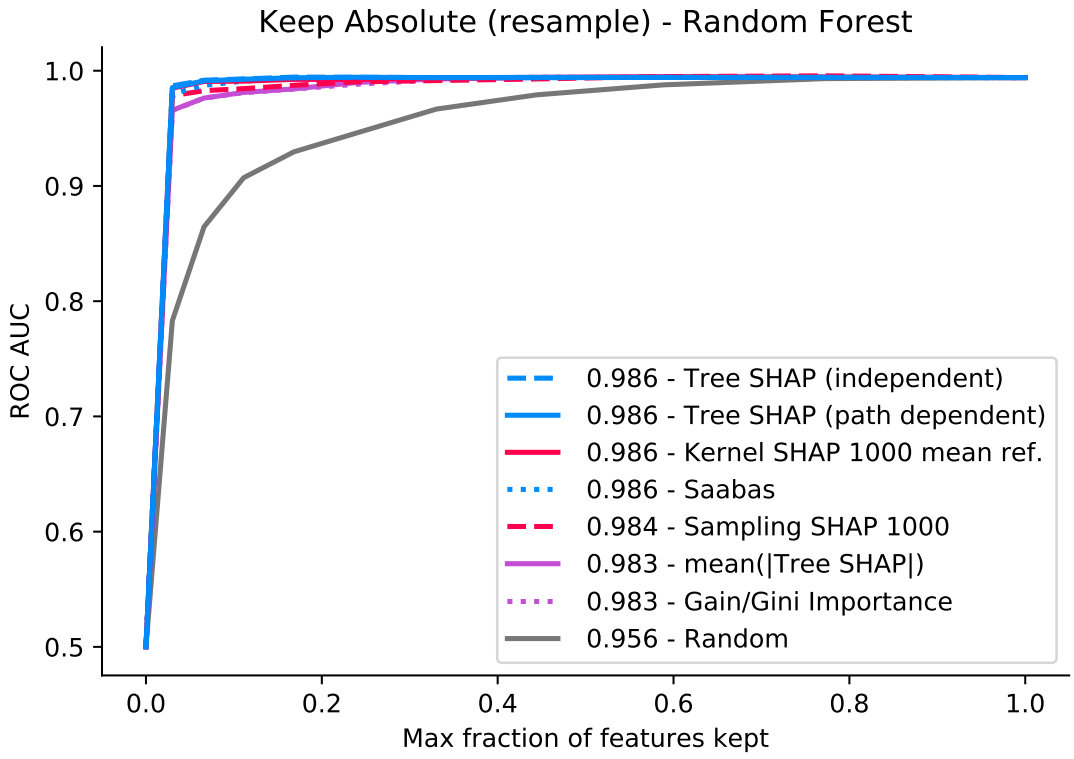
Supplementary Figure S27: **Keep negative (impute) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S24 except that we keep the most negative features instead of the most positive. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



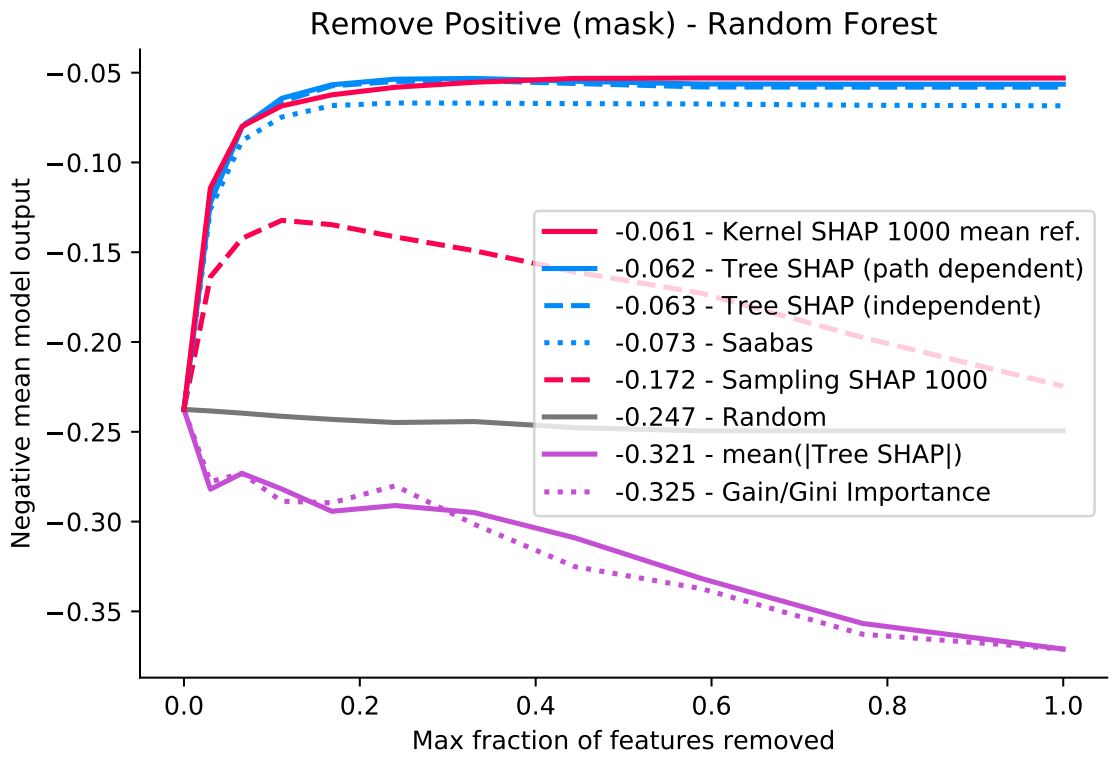
Supplementary Figure S28: **Keep absolute (mask) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figures S22 and S25 except that we keep the most important features by absolute value instead of the most positive or negative. Since this no longer specifically pushes the model output higher or lower, we instead measure the accuracy of the model. Good attribution methods will identify important features that when kept will result in better model accuracy, measured in this case by the area under the receiver operating characteristic (ROC) curve. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



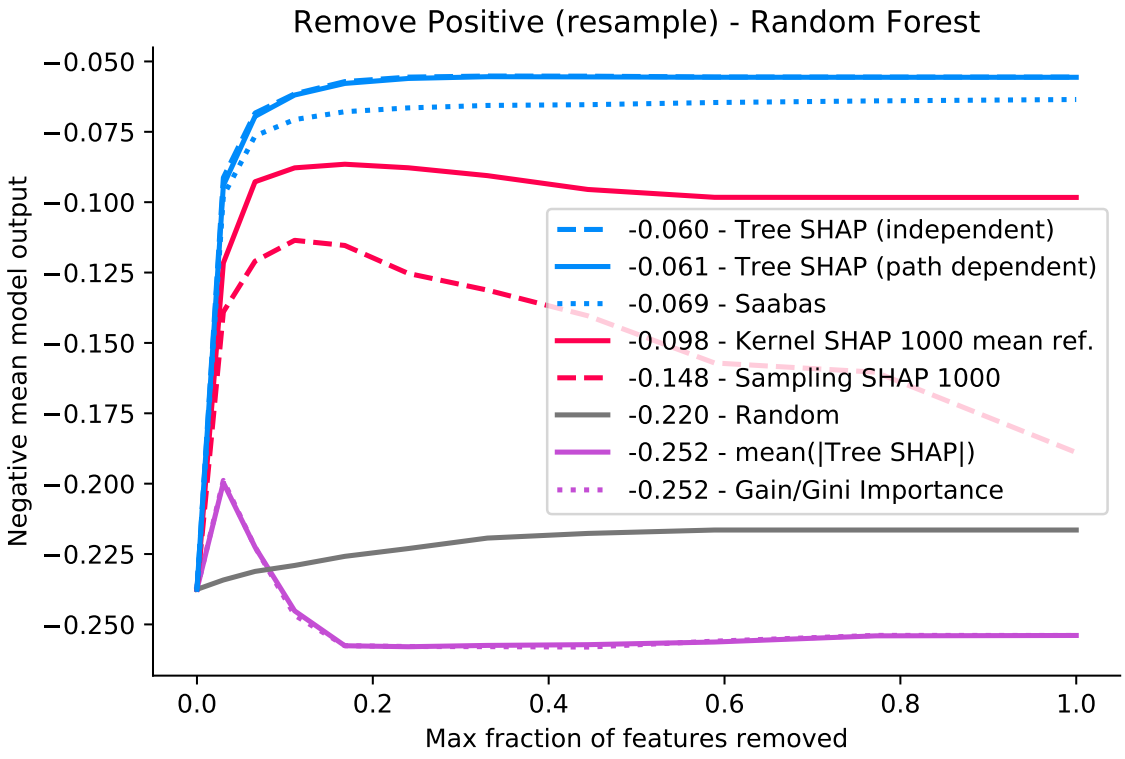
Supplementary Figure S29: **Keep absolute (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figures S23 and S26 except that we keep the most important features by absolute value instead of the most positive or negative (as in Figure S28). Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



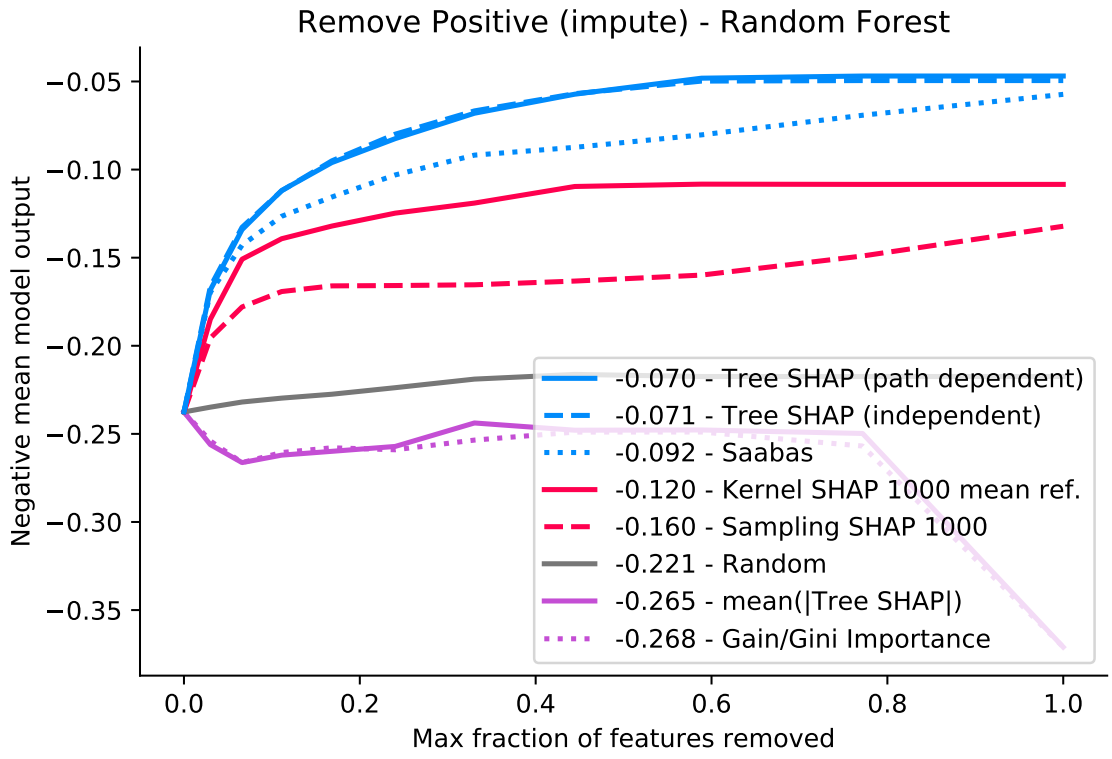
Supplementary Figure S30: **Keep absolute (impute) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figures S24 and S27 except that we keep the most important features by absolute value instead of the most positive or negative (as in Figure S28). Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



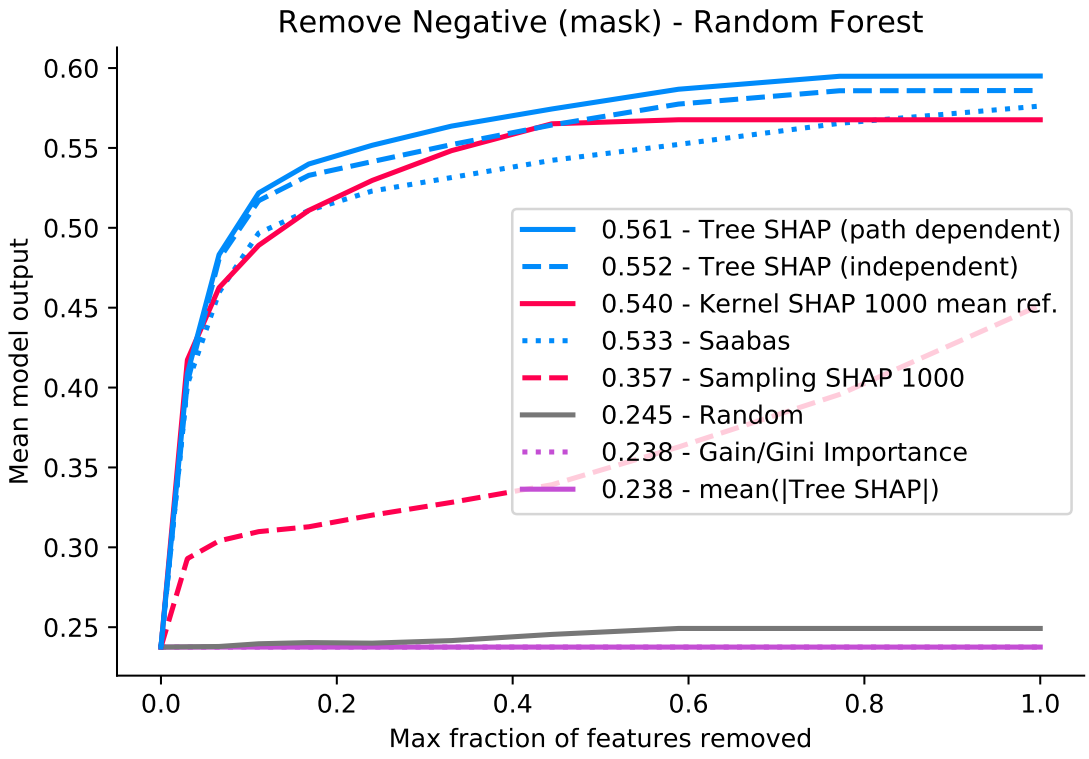
Supplementary Figure S31: **Remove positive (mask) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S22 except that we remove the most positive features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



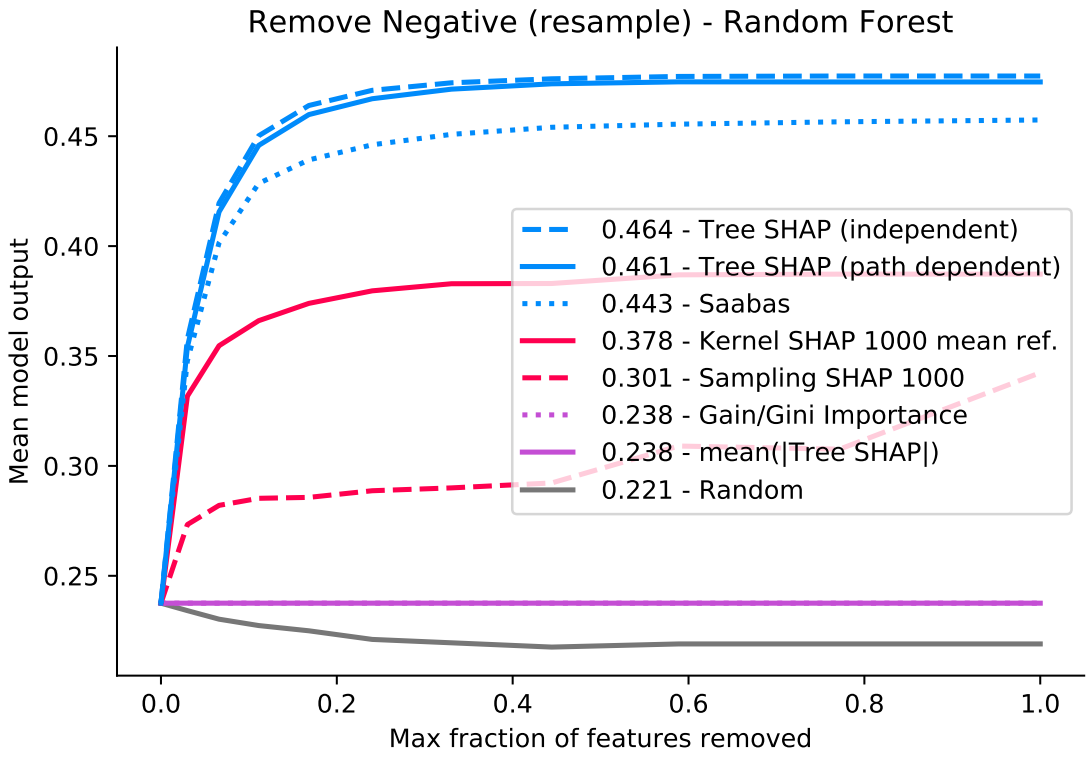
Supplementary Figure S32: **Remove positive (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S23 except that we remove the most positive features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



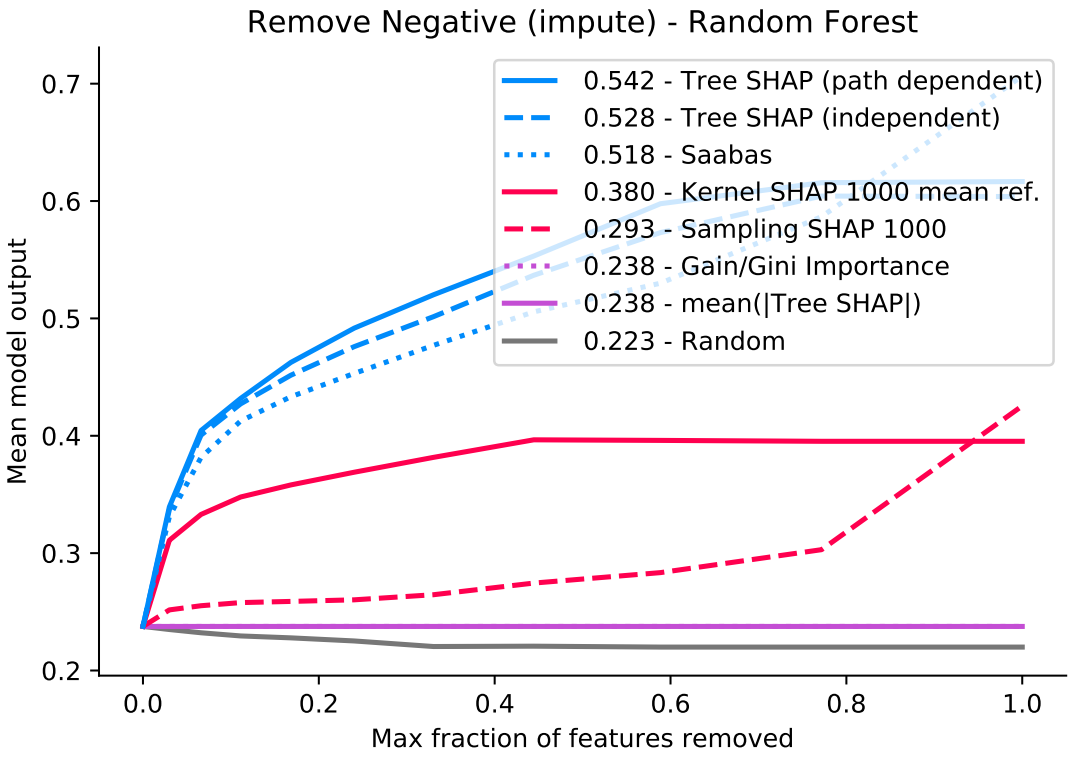
Supplementary Figure S33: **Remove positive (impute) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S24 except that we remove the most positive features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



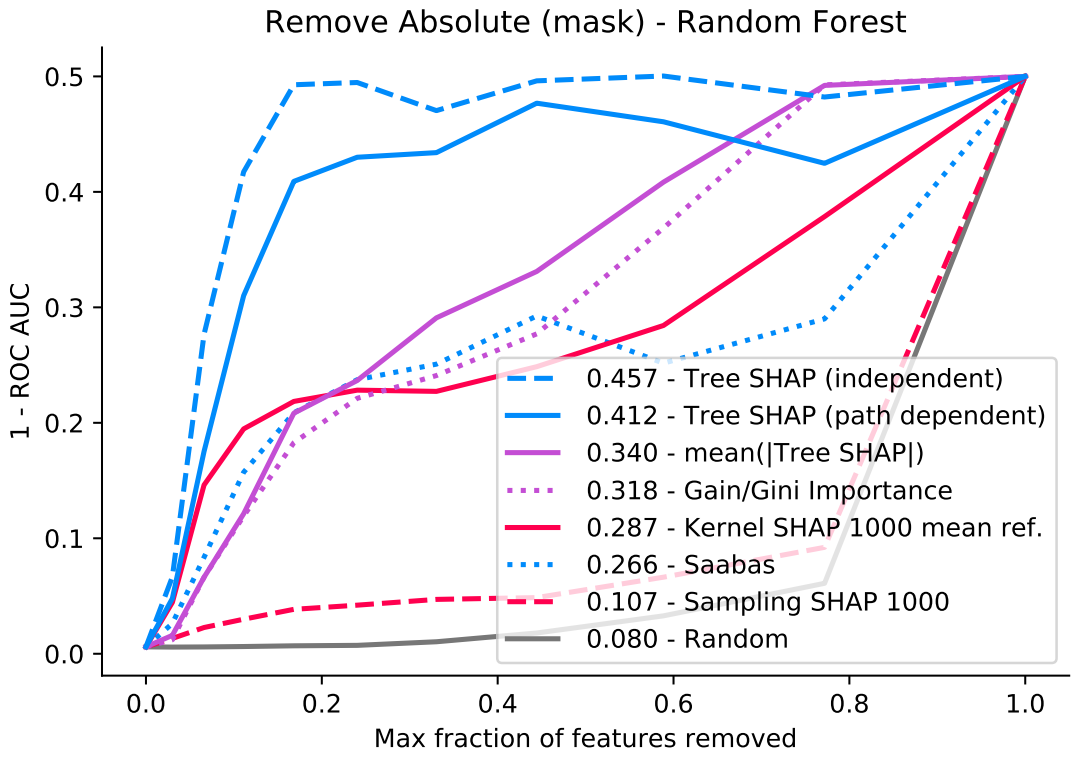
Supplementary Figure S34: **Remove negative (mask) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S25 except that we remove the most negative features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



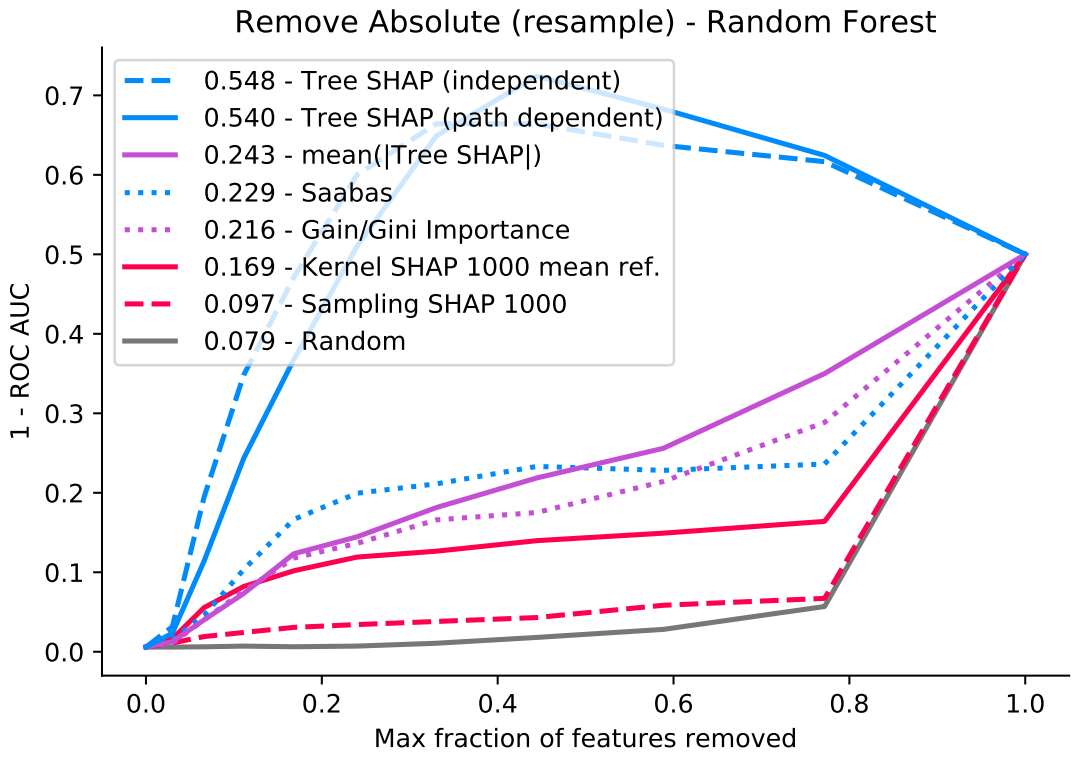
Supplementary Figure S35: **Remove negative (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S26 except that we remove the most negative features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



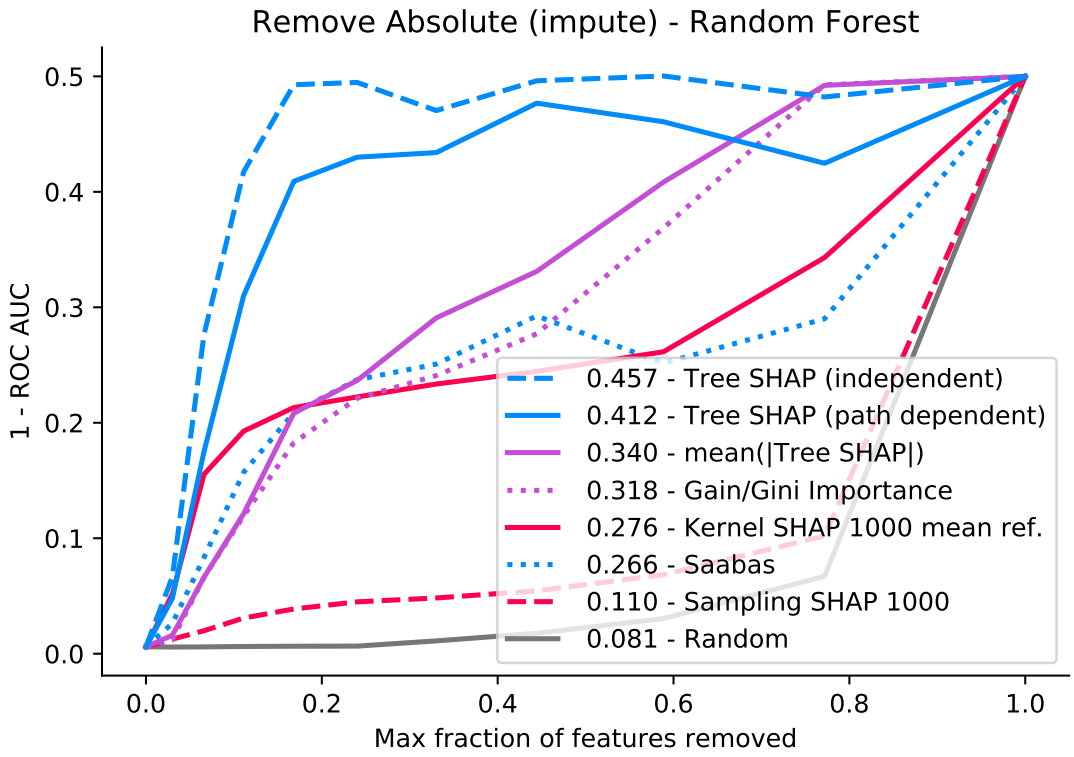
Supplementary Figure S36: **Remove negative (impute) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S27 except that we remove the most negative features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



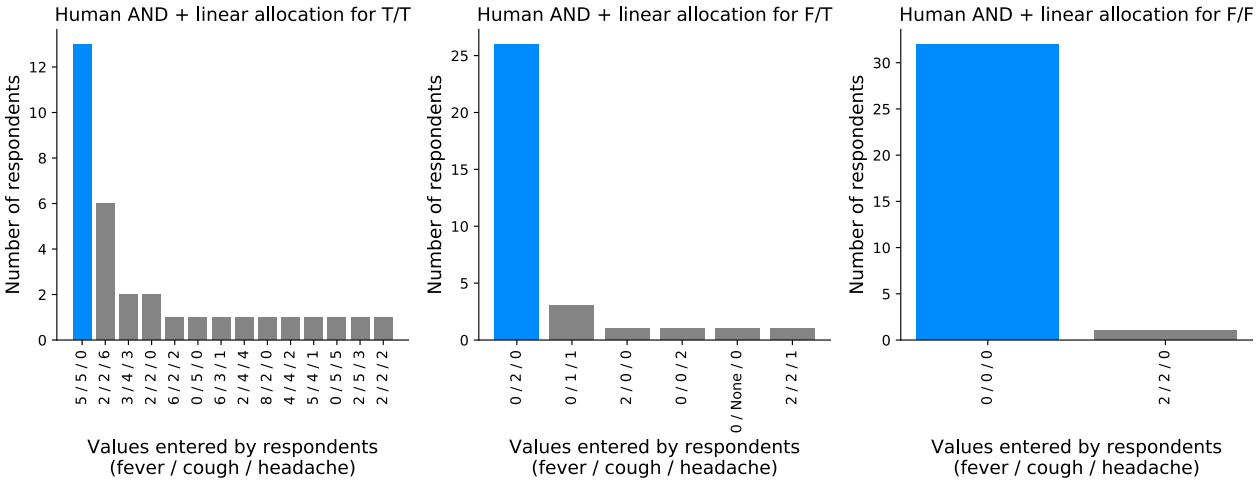
Supplementary Figure S37: **Remove absolute (mask) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S28 except that we remove the most important features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



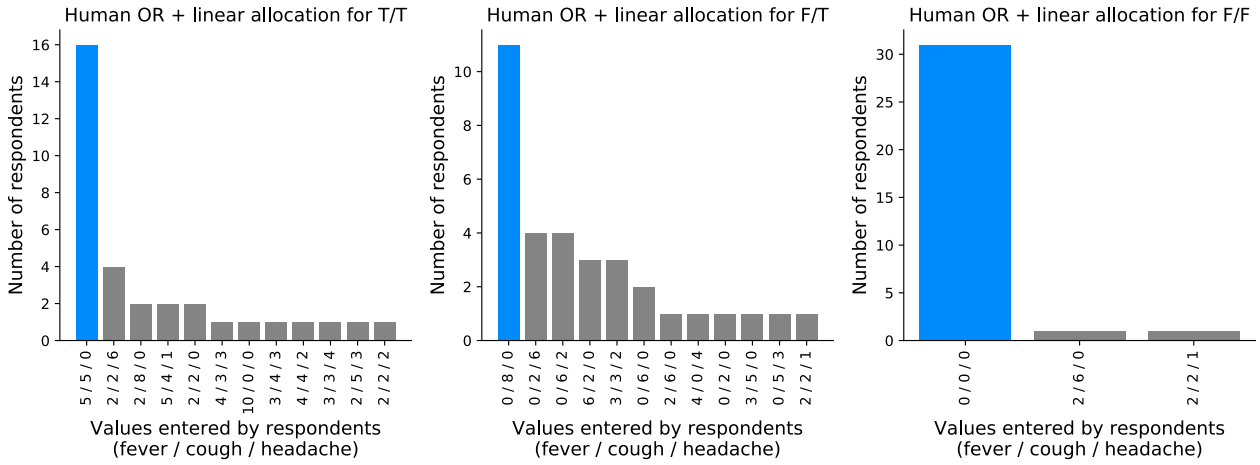
Supplementary Figure S38: **Remove absolute (resample) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S29 except that we remove the most important features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



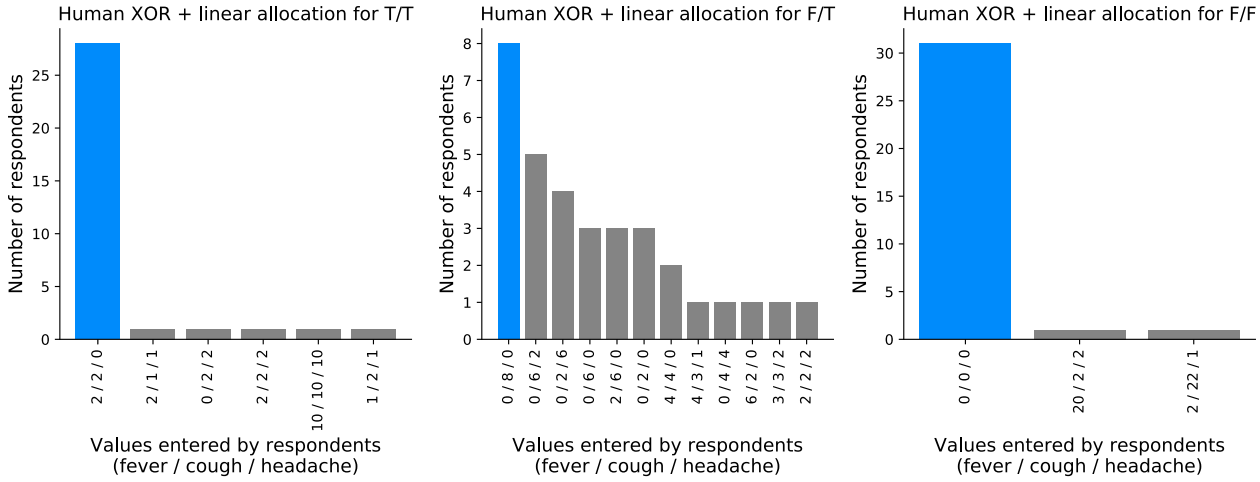
Supplementary Figure S39: **Remove absolute (impute) metric for a random forest trained on the chronic kidney disease dataset.** This is just like Supplementary Figure S30 except that we remove the most important features instead of keeping them. Note that the Tree SHAP and Sampling SHAP algorithms correspond to TreeExplainer and IME [167], respectively.



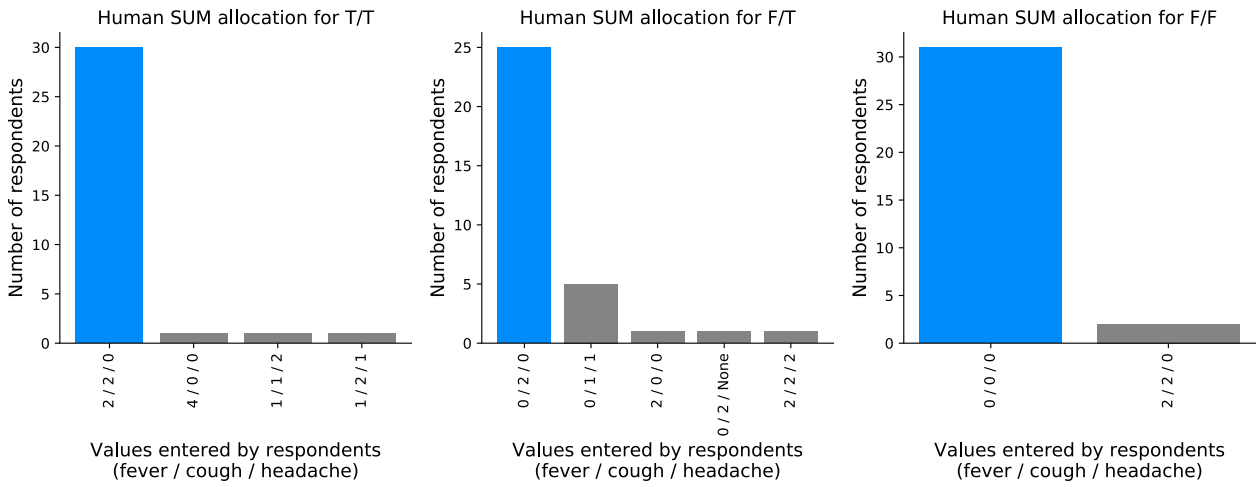
Supplementary Figure S40: **Consensus human intuition values for an AND function.** Human consensus values were measured by a user study over 33 participants for a simple AND-based function. The most popular allocation was chosen as the consensus (Methods 5.4.12). The labels denote the allocation given by people as "fever / cough / headache". The title gives the input values for sample being explained as "fever value/cough value", where 'T' is true and 'F' is false; note that headache is always set to true.



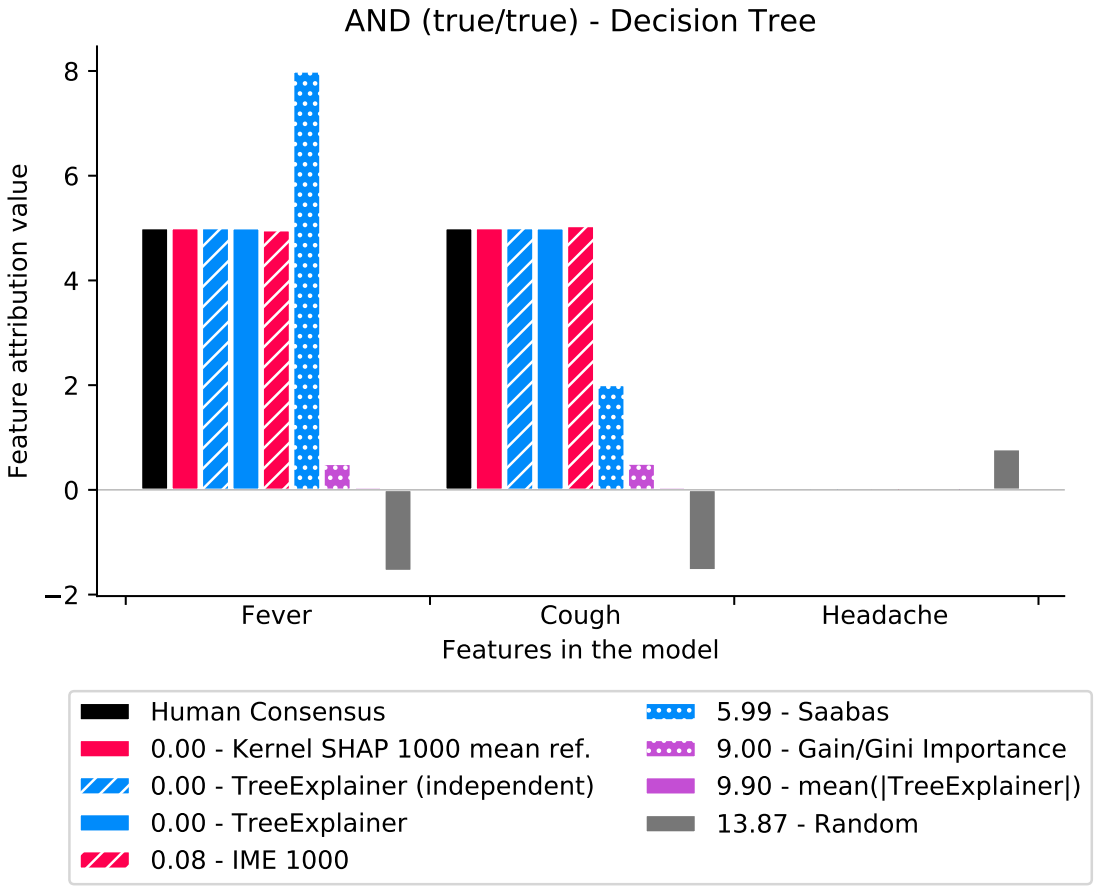
Supplementary Figure S41: **Consensus human intuition values for an OR function.** Human consensus values were measured by a user study over 33 participants for a simple OR-based function. The most popular allocation was chosen as the consensus (Methods 5.4.12). The labels denote the allocation given by people as "fever / cough / headache". The title gives the input values for sample being explained as "fever value/cough value", where 'T' is true and 'F' is false; note that headache is always set to true.



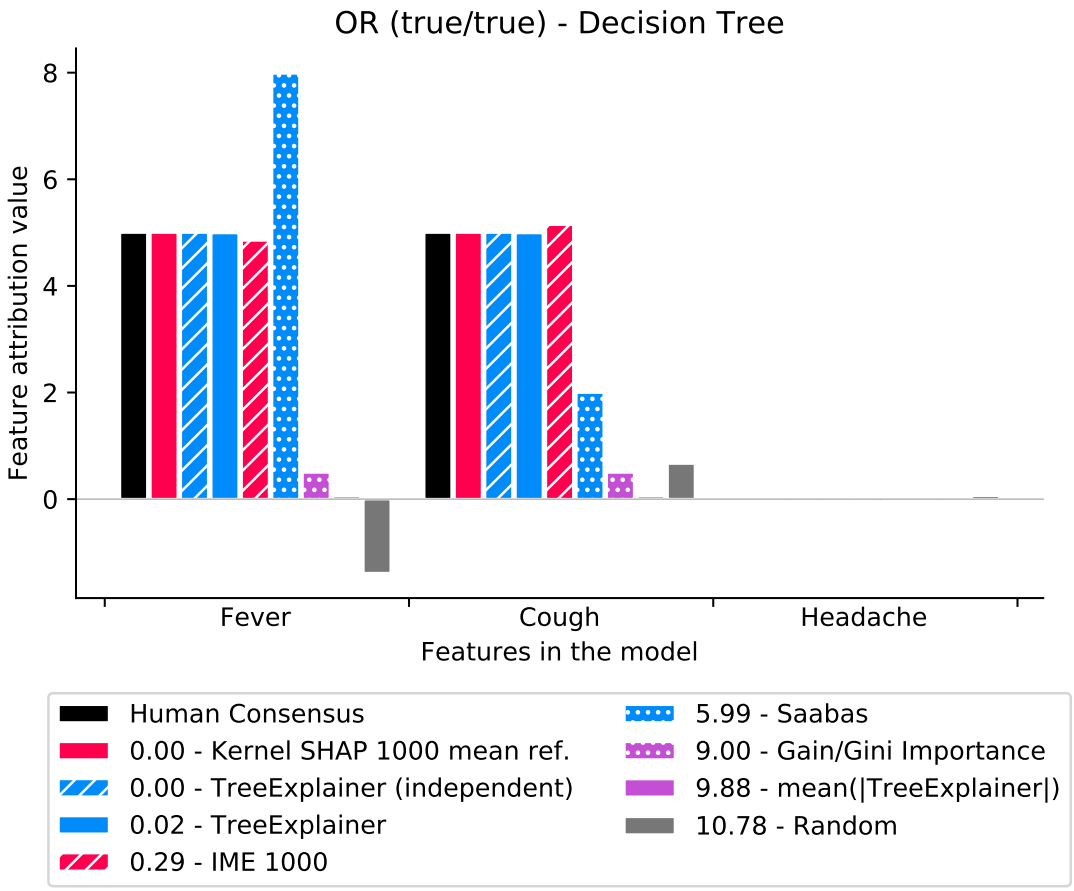
Supplementary Figure S42: **Consensus human intuition values for an eXclusive OR (XOR) function.** Human consensus values were measured by a user study over 33 participants for a simple XOR-based function. The most popular allocation was chosen as the consensus (Methods 5.4.12). The labels denote the allocation given by people as "fever / cough / headache". The title gives the input values for sample being explained as "fever value/cough value", where 'T' is true and 'F' is false; note that headache is always set to true.



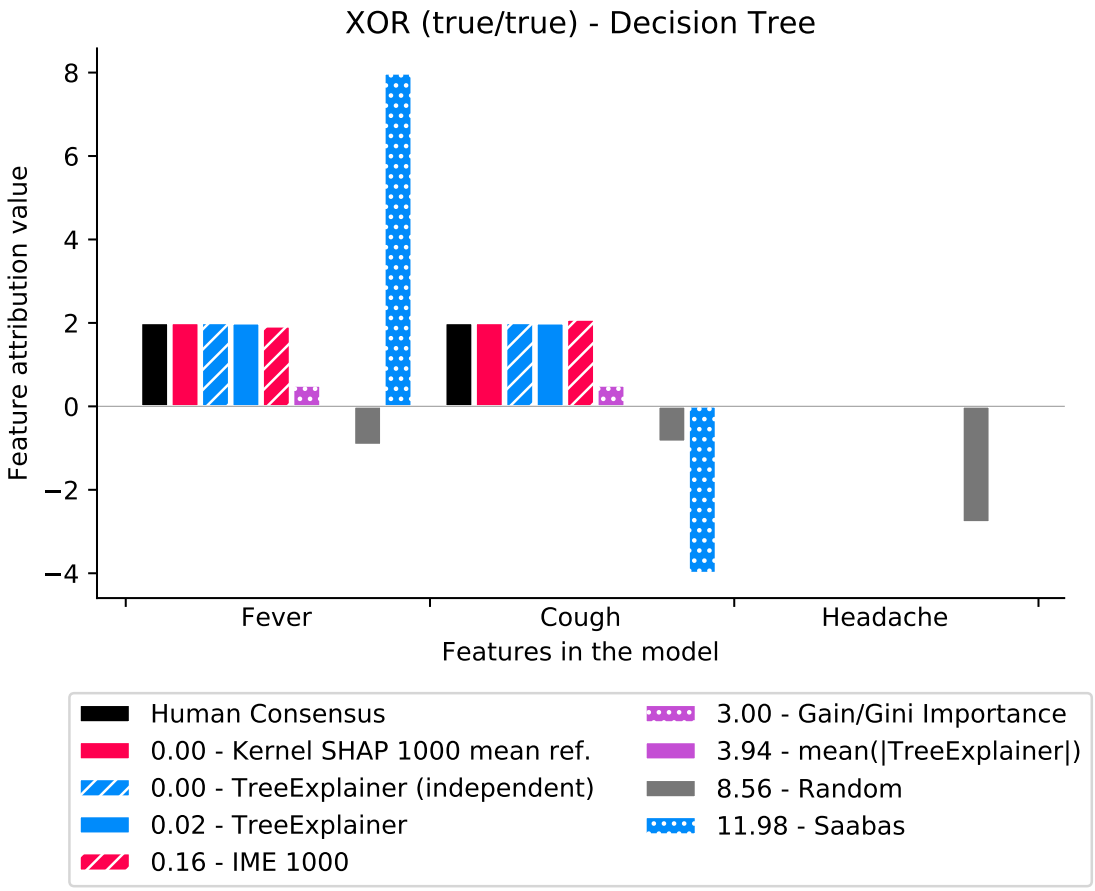
Supplementary Figure S43: **Consensus human intuition values for a SUM function.** Human consensus values were measured by a user study over 33 participants for a simple SUM function. The most popular allocation was chosen as the consensus (Methods 5.4.12). The labels denote the allocation given by people as "fever / cough / headache". The title gives the input values for sample being explained as "fever value/cough value", where 'T' is true and 'F' is false; note that headache is always set to true.



Supplementary Figure S44: **Comparison with human intuition for an AND function.** Human consensus values were measured by a user study for a simple AND-based function evaluated when all inputs were set to 'true' (Supplementary Figure S40). The results of different explanation methods were then compared to these consensus values to measure their consistency with human intuition (Methods 5.4.12).



Supplementary Figure S45: **Comparison with human intuition for an OR function.** Human consensus values were measured by a user study for a simple OR-based function evaluated when all inputs were set to 'true' (Supplementary Figure S41). The results of different explanation methods were then compared to these consensus values to measure their consistency with human intuition (Methods 5.4.12).



Supplementary Figure S46: **Comparison with human intuition for an eXclusive OR (XOR) function.** Human consensus values were measured by a user study for a simple XOR-based function evaluated when all inputs were set to 'true' (Supplementary Figure S42). The results of different explanation methods were then compared to these consensus values to measure their consistency with human intuition (Methods 5.4.12).

Chapter 6

Conclusion

Explainable machine learning is a broad area of research that contains many sub-problems. In this dissertation I have focused on two specific types of explanations: graph-based explanations that reveal relationships between variables, and feature attribution based explanations that assign a real valued measure of impact to each input feature of a supervised model. Feature attribution based explanations are by no means the only way to explain a prediction made by a supervised machine learning model, but they are the most widely used type of explanation. My work in Chapters 3-5 has sought to advance the state of the art for feature attribution based explanations, and so improve on this fundamental type of explanation.

There are many future directions to improve feature attribution based explanations. Future theoretical directions include exploring and quantifying explainability trade-offs that exist in the presence of correlated input features, as well as using explanations to guide model training. Directions for future practical contributions include efficient and general model monitoring tools, as well as the integration of causal assumptions and information to better inform the interpretation of the models we are explaining. The number of future application areas for feature attribution based explanations is enormous, but a couple of particular interest to me are in-the-loop high stakes decision making, and better understanding of machine learning models in biology and medicine. As the field of explainable machine learning grows, I think our ability to evaluate the tools we build should also grow. The comprehensive benchmark proposed in Chapter 5 is a first step in this direction, but there is much more research needed in this area.

In this dissertation strong classic theoretical results from Shapley values were combined with fast new algorithms and applied to specific real-world problems. I look forward to seeing the field of explainability continue to progress along all three of these dimensions: theoretical rigor, computational performance, and innovative applications.

Bibliography

[1] Marco Ancona et al. "Towards better understanding of gradient-based attribution methods for Deep Neural Networks". In: *6th International Conference on Learning Representations (ICLR 2018)*. 2018.

[2] American Medical Association. "Current procedural terminology: CPT". In: (2007).

[3] Sandy Leung-Kuen Au et al. "EZH2-mediated H3K27me3 is involved in epigenetic repression of deleted in liver cancer 1 in human cancers". In: *PloS One* 8.6 (2013), e68226.

[4] Lidia Auret and Chris Aldrich. "Empirical comparison of tree ensemble variable importance measures". In: *Chemometrics and Intelligent Laboratory Systems* 105.2 (2011), pp. 157–170.

[5] Michael S Avidan et al. "Anesthesia awareness and the bispectral index". In: *New England Journal of Medicine* 358.11 (2008), pp. 1097–1108.

[6] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.

[7] David Baehrens et al. "How to explain individual classification decisions". In: *Journal of Machine Learning Research* 11.Jun (2010), pp. 1803–1831.

[8] George L Bakris et al. "Effects of blood pressure level on progression of diabetic nephropathy: results from the RENAAL study". In: *Archives of internal medicine* 163.13 (2003), pp. 1555–1565.

[9] David A Belsley, Edwin Kuh, and Roy E Welsch. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. Vol. 571. Hoboken, NJ: John Wiley & Sons, 2005.

[10] Joke G van Bemmel et al. "A network model of the molecular organization of chromatin in Drosophila". In: *Molecular cell* 49.4 (2013), pp. 759–771.

[11] Shelley L Berger. "The complex language of chromatin regulation during transcription". In: *Nature* 447.7143 (2007), pp. 407–412.

[12] Bradley E Bernstein et al. "A bivalent chromatin structure marks key developmental genes in embryonic stem cells". In: *Cell* 125.2 (2006), pp. 315–326.

[13] Elizabeth M Blackwood and Robert N Eisenman. "Max: a helix-loop-helix zipper protein that forms a sequence-specific DNA-binding complex with Myc". In: *Science* 251.4998 (1991), pp. 1211–1217.

[14] Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Berlin, Germany: Springer Science & Business Media, 2005.

[15] Benjamin Bowe et al. "Association between monocyte count and risk of incident CKD and progression to ESRD". In: *Clinical Journal of the American Society of Nephrology* 12.4 (2017), pp. 603–613.

[16] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[17] Leo Breiman et al. *Classification and regression trees*. CRC press, 1984.

[18] Ru Cao and YI Zhang. "SUZ12 is required for both the histone methyltransferase activity and the silencing function of the EED-EZH2 complex". In: *Molecular Cell* 15.1 (2004), pp. 57–67.

[19] Raymond J Carroll and David Ruppert. *Transformation and weighting in regression*. Vol. 30. Boca Raton, FL: CRC Press, 1988.

[20] Rich Caruana et al. "Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission". In: *KDD*. 2015.

[21] CDC. “Defining Adult Overweight and Obesity (https://www.cdc.gov/obesity/adult/defining.html)”. In: (2017).

[22] A Charnes et al. “Extremal principle solutions of games in characteristic function form: core, Chebychev and Shapley value generalizations”. In: *Econometrics of Planning and Efficiency* 11 (1988), pp. 123–133.

[23] S Chebrolu, A Abraham, and J Thomas. “Feature deduction and ensemble design of intrusion detection systems”. In: *Computers & security* 24.4 (2005), pp. 295–307.

[24] Tianqi Chen and Carlos Guestrin. “XGBoost: A scalable tree boosting system”. In: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 785–794.

[25] Alfred K Cheung et al. “Effects of intensive BP control in CKD”. In: *Journal of the American Society of Nephrology* 28.9 (2017), pp. 2812–2823.

[26] François Chollet et al. *Keras*. `https://github.com/fchollet/keras`. 2015.

[27] Cedric R Clapier and Bradley R Cairns. “The biology of chromatin remodeling complexes”. In: *Annual Review of Biochemistry* 78 (2009), pp. 273–304.

[28] Chronic Kidney Disease Prognosis Consortium et al. “Association of estimated glomerular filtration rate and albuminuria with all-cause and cardiovascular mortality in general population cohorts: a collaborative meta-analysis”. In: *The Lancet* 375.9731 (2010), pp. 2073–2081.

[29] ENCODE Project Consortium et al. “An integrated encyclopedia of DNA elements in the human genome”. In: *Nature* 489.7414 (2012), pp. 57–74.

[30] Christine S Cox et al. “Plan and operation of the NHANES I Epidemiologic Followup Study, 1992”. In: (1997).

[31] Menno P Creyghton et al. “Histone H3K27ac separates active from poised enhancers and predicts developmental state”. In: *Proceedings of the National Academy of Sciences* 107.50 (2010), pp. 21931–21936.

[32] Anupam Datta, Shayak Sen, and Yair Zick. “Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems”. In: *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE. 2016, pp. 598–617.

[33] Cameron Davidson-Pilon. *Lifelines*. `https://github.com/camdavidsonpilon/lifelines`. 2016.

[34] Rahul C. Deo. “Machine Learning in Medicine.” In: *Circulation* 132 20 (2015), pp. 1920–30.

[35] Anwesha Dey et al. “Loss of the tumor suppressor BAP1 causes myeloid transformation”. In: *Science* 337.6101 (2012), pp. 1541–1546.

[36] Luciano Di Croce and Kristian Helin. “Transcriptional regulation by Polycomb group proteins”. In: *Nature structural & molecular biology* 20.10 (2013), pp. 1147–1155.

[37] Marc I Diamond et al. “Transcription factor interactions: selectors of positive or negative regulation from a single DNA element”. In: *Science* 249.4974 (1990), pp. 1266–1272.

[38] C Michael Dunham et al. “Perioperative hypoxemia is common with horizontal positioning during general anesthesia and is associated with major adverse outcomes: a retrospective study of consecutive patients”. In: *BMC anesthesiology* 14.1 (2014), p. 43.

[39] Kirill Dyagilev and Suchi Saria. “Learning (predictive) risk scores in the presence of censoring due to interventions”. In: *Machine Learning* 102.3 (2016), pp. 323–348.

[40] Jesse M Ehrenfeld et al. “The incidence of hypoxemia during surgery: evidence from two institutions”. In: *Canadian Journal of Anesthesia/Journal canadien d’anesthésie* 57.10 (2010), pp. 888–897.

[41] Hisham ElMoaqet, Dawn M Tilbury, and Satya Krishna Ramachandran. “Multi-step ahead predictions for critical levels in physiological time series”. In: *IEEE transactions on cybernetics* 46.7 (2016), pp. 1704–1714.

[42] *Encode Project Data*. encodeproject.org.

[43] Richard H Epstein, Franklin Dexter, and Neil Patel. "Influencing anesthesia provider behavior using anesthesia information management system data for near real-time alerts and post hoc reports". In: *Anesthesia & Analgesia* 121.3 (2015), pp. 678–692.

[44] Fangfang Fan et al. "White blood cell count predicts the odds of kidney function decline in a Chinese community-based population". In: *BMC nephrology* 18.1 (2017), p. 190.

[45] Jing Fang and Michael H Alderman. "Serum uric acid and cardiovascular mortality: the NHANES I epidemiologic follow-up study, 1971-1992". In: *Jama* 283.18 (2000), pp. 2404–2410.

[46] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[47] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin, 2001.

[48] Seth Frietze et al. "ZNF274 recruits the histone methyltransferase SETDB1 to the 3' ends of ZNF genes". In: *PloS one* 5.12 (2010), e15082–e15082.

[49] Haiqing Fu et al. "Methylation of histone H3 on lysine 79 associates with a group of replication origins and helps limit DNA replication once per cell cycle". In: (2013).

[50] Katsushige Fujimoto, Ivan Kojadinovic, and Jean-Luc Marichal. "Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices". In: *Games and Economic Behavior* 55.1 (2006), pp. 72–99.

[51] Pablo Garcia-Sanz et al. "Sin3b interacts with Myc and decreases Myc levels". In: *Journal of Biological Chemistry* 289.32 (2014), pp. 22221–22236.

[52] Amit X Garg et al. "Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review". In: *Jama* 293.10 (2005), pp. 1223–1238.

[53] Atul A Gawande et al. "The incidence and nature of surgical adverse events in Colorado and Utah in 1992". In: *Surgery* 126.1 (1999), pp. 66–75.

[54] *Genome Reference Consortium*. genomereference.org.

[55] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. "Variable selection using random forests". In: *Pattern Recognition Letters* 31.14 (2010), pp. 2225–2236.

[56] Mark B Gerstein et al. "Architecture of the human regulatory network derived from ENCODE data". In: *Nature* 489.7414 (2012), pp. 91–100.

[57] SPRINT Research Group. "A randomized trial of intensive versus standard blood-pressure control". In: *New England Journal of Medicine* 373.22 (2015), pp. 2103–2116.

[58] Joanne Guay, Edward A Ochroch, and Sandra Kopp. "Intraoperative use of low volume ventilation to decrease postoperative mortality, mechanical ventilation, lengths of stay and lung injury in adults without acute lung injury". In: *Cochrane Database of Systematic Reviews* 7 (2018).

[59] Kalpana Gupta et al. "Mmip1: a novel leucine zipper protein that reverses the suppressive effects of Mad family members on c-Myc." In: *Oncogene* 16.9 (1998), pp. 1149–1159.

[60] Stephen R Hann. "MYC Cofactors: Molecular Switches Controlling Diverse Biological Outcomes". In: *Cold Spring Harbor Perspectives in Medicine* 17.4 (2014), p. 9.

[61] Trevor Hastie et al. *The Elements of Statistical Learning*. Vol. 2. 1. Berlin, Germany: Springer, 2009.

[62] Alex B Haynes et al. "A surgical safety checklist to reduce morbidity and mortality in a global population". In: *New England Journal of Medicine* 360.5 (2009), pp. 491–499.

[63] Patrick J Heagerty, Thomas Lumley, and Margaret S Pepe. "Time-dependent ROC curves for censored survival data and a diagnostic marker". In: *Biometrics* 56.2 (2000), pp. 337–344.

[64] Katharine Henry et al. "A targeted real-time early warning score (TREWScore) for septic shock". In: *Science Translational Medicine* 7 (2015), 299ra122–299ra122.

[65] Michael M Hoffman et al. "Unsupervised pattern discovery in human chromatin structure through genomic segmentation". In: *Nature Methods* 9.5 (2012), pp. 473–476.

[66] Sara Hooker et al. "Evaluating feature importance estimates". In: *arXiv preprint arXiv:1806.10758* (2018).

[67] A Irrthum, L Wehenkel, P Geurts, et al. "Inferring regulatory networks from expression data using tree-based methods". In: *PloS one* 5.9 (2010), e12776.

[68] Hemant Ishwaran et al. "Variable importance in binary regression trees and forests". In: *Electronic Journal of Statistics* 1 (2007), pp. 519–537.

[69] Rui Jiang et al. "A random forest approach to the detection of epistatic interactions in case-control studies". In: *BMC bioinformatics* 10.1 (2009), S65.

[70] David Juan et al. "Epigenomic co-localization and co-evolution reveal a key role for 5hmC as a communication hub in the chromatin network of ESCs". In: *bioRxiv* (2015). DOI: 10.1101/008821.

[71] AK Kable, RW Gibberd, and AD Spigelman. "Adverse events in surgical patients in Australia". In: *International Journal for Quality in Health Care* 14.4 (2002), pp. 269–276.

[72] Kaggle. *The State of ML and Data Science 2017*. 2017. URL: https://www.kaggle.com/surveys/2017.

[73] Jared L Katzman et al. "DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network". In: *BMC medical research methodology* 18.1 (2018), p. 24.

[74] Jalil Kazemitabar et al. "Variable Importance using Decision Trees". In: *Advances in Neural Information Processing Systems*. 2017, pp. 426–435.

[75] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in Neural Information Processing Systems*. 2017, pp. 3146–3154.

[76] Samir M Kendale and Jeanna D Blitz. "Increasing body mass index and the incidence of intraoperative hypoxemia". In: *Journal of clinical anesthesia* 33 (2016), pp. 97–104.

[77] Satyajeet P Khare et al. "HIstome—a relational knowledgebase of human histone proteins and histone modifying enzymes". In: *Nucleic acids research* 40.D1 (2012), pp. D337–D342.

[78] *Kidney Disease Statistics for the United States*. 2018. URL: https://www.niddk.nih.gov/health-information/health-statistics/kidney-disease.

[79] Joomyeong Kim and Hana Kim. "Recruitment and biological consequences of histone modification of H3K27me3 and H3K9me3". In: *ILAR Journal* 53.3-4 (2012), pp. 232–239.

[80] Tae Wan Kim et al. "Ctbp2 Modulates NuRD-Mediated Deacetylation of H3K27 and Facilitates PRC2-Mediated H3K27me3 in Active Embryonic Stem Cell Genes During Exit from Pluripotency". In: *STEM CELLS* (2015).

[81] Masaomi Kimura et al. "Comparison of lesion formation between contact force-guided and non-guided circumferential pulmonary vein isolation: a prospective, randomized study." In: *Heart rhythm* 11.6 (June 2014), pp. 984–91. ISSN: 1556-3871. DOI: 10.1016/j.hrthm.2014.03.019. URL: https://linkinghub.elsevier.com/retrieve/pii/S1547527114003002%20http://www.ncbi.nlm.nih.gov/pubmed/24657428.

[82] Pieter-Jan Kindermans et al. "Learning how to explain neural networks: PatternNet and PatternAttribution". In: *arXiv preprint arXiv:1705.05598* (2017).

[83] D.J. Klein and M. Randić. "Resistance distance". English. In: *Journal of Mathematical Chemistry* 12.1 (1993), pp. 81–95. ISSN: 0259-9791. DOI: 10.1007/BF01164627. URL: http://dx.doi.org/10.1007/BF01164627.

[84] Fabian O Kooij et al. "Decision support increases guideline adherence for prescribing postoperative nausea and vomiting prophylaxis". In: *Anesthesia & Analgesia* 106.3 (2008), pp. 893–898.

[85] Karl-Heinz Kuck et al. "Cryoballoon or Radiofrequency Ablation for Paroxysmal Atrial Fibrillation". In: *New England Journal of Medicine* 374.23 (June 2016), pp. 2235–2245. ISSN: 0028-4793. DOI: 10.1056/NEJMoa1602014. URL: http://www.nejm.org/doi/10.1056/NEJMoa1602014.

[86] Clarence CY Kwan. "A Regression-Based Interpretation of the Inverse of the Sample Covariance Matrix". In: *Spreadsheets in Education (eJSiE)* 7.1 (2014), p. 3.

[87] Monika Lachner, Roderick J O'Sullivan, and Thomas Jenuwein. "An epigenetic road map for histone lysine methylation". In: *Journal of Cell Science* 116.11 (2003), pp. 2117–2124.

[88] Ben Langmead and Steven L Salzberg. "Fast gapped-read alignment with Bowtie 2". In: *Nature Methods* 9.4 (2012), pp. 357–359.

[89] James P Lash et al. "Chronic Renal Insufficiency Cohort (CRIC) Study: baseline characteristics and associations with kidney function". In: *Clinical Journal of the American Society of Nephrology* 4.8 (2009), pp. 1302–1311.

[90] Julia Lasserre, Ho-Ryun Chung, and Martin Vingron. "Finding associations among histone modifications using sparse partial correlation networks". In: *PLoS Computational Biology* 9.9 (2013), e1003168.

[91] Lenore J Launer et al. "Body mass index, weight change, and risk of mobility disability in middle-aged and older women: the epidemiologic follow-up study of NHANES I". In: *Jama* 271.14 (1994), pp. 1093–1098.

[92] Steffen L Lauritzen. *Graphical Models*. Oxford, UK: Oxford University Press, 1996.

[93] Ngoc T Le et al. "A nucleosomal approach to inferring causal relationships of histone modifications". In: *BMC Genomics* 15.Suppl 1 (2014), S7.

[94] Chun-Chung Lee et al. "TCF12 protein functions as transcriptional repressor of E-cadherin, and its overexpression is correlated with metastasis of colorectal cancer". In: *Journal of Biological Chemistry* 287.4 (2012), pp. 2798–2809.

[95] Stan Lipovetsky and Michael Conklin. "Analysis of regression in game theory approach". In: *Applied Stochastic Models in Business and Industry* 17.4 (2001), pp. 319–330.

[96] Zachary Chase Lipton, David C. Kale, and Randall C. Wetzel. "Phenotyping of Clinical Time Series with LSTM Recurrent Neural Networks". In: *CoRR* abs/1510.07641 (2015).

[97] Po-Ling Loh, Martin J Wainwright, et al. "Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses". In: *The Annals of Statistics* 41.6 (2013), pp. 3022–3049.

[98] Ana Losada, Tomoki Yokochi, and Tatsuya Hirano. "Functional contribution of Pds5 to cohesin-mediated cohesion in human cells and *Xenopus* egg extracts". In: *Journal of Cell Science* 118.10 (2005), pp. 2133–2141.

[99] Gilles Louppe. "Understanding random forests: From theory to practice". In: *arXiv preprint arXiv:1407.7502* (2014).

[100] F Lumachi et al. "Relationship between body mass index, age and hypoxemia in patients with extremely severe obesity undergoing bariatric surgery". In: *in vivo* 24.5 (2010), pp. 775–777.

[101] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 4768–4777. URL: `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

[102] Scott M. Lundberg and Su-In Lee. "Consistent feature attribution for tree ensembles". In: *CoRR* abs/1706.06060 (2017).

[103] Scott M Lundberg et al. "ChromNet: Learning the human chromatin network from all ENCODE ChIP-seq data". In: *Genome biology* 17.1 (2016), p. 82.

[104] Scott M Lundberg et al. "Explainable AI for Trees: From Local Explanations to Global Understanding". In: *arXiv preprint arXiv:1905.04610* (2019).

[105] Scott M Lundberg et al. "Explainable machine learning predictions to help anesthesiologists prevent hypoxemia during surgery". In: *Nature Biomedical Engineering* 2 (2018), pp. 749–760.

[106] Kathryn L Lunetta et al. "Screening large-scale association study data: exploiting interactions using random forests". In: *BMC genetics* 5.1 (2004), p. 32.

[107] Lena Maier-Hein et al. "Surgical data science for next-generation interventions". In: *Nature Biomedical Engineering* 1.9 (2017), p. 691.

[108] Kantilal Varichand Mardia, John T Kent, and John M Bibby. *Multivariate Analysis*. Cambridge, MA: Academic Press, 1979.

[109] Adam A Margolin et al. "ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context". In: *BMC bioinformatics* 7.Suppl 1 (2006), S7.

[110] Eloi Marijon et al. "Real-time contact force sensing for pulmonary vein isolation in the setting of paroxysmal atrial fibrillation: procedural and 1-year results." In: *Journal of cardiovascular electrophysiology* 25.2 (Feb. 2014), pp. 130–7. ISSN: 1540-8167. DOI: 10.1111/jce.12303. URL: http://doi.wiley.com/10.1111/jce.12303%20http://www.ncbi.nlm.nih.gov/pubmed/24433324.

[111] Yasuko Matsui and Tomomi Matsui. "NP-completeness for calculating power indices of weighted majority games". In: *Theoretical Computer Science* 263.1-2 (2001), pp. 305–310.

[112] Kunihiro Matsushita et al. "Comparison of risk prediction using the CKD-EPI equation and the MDRD study equation for estimated glomerular filtration rate". In: *Jama* 307.18 (2012), pp. 1941–1951.

[113] Kunihiro Matsushita et al. "Estimated glomerular filtration rate and albuminuria for prediction of cardiovascular outcomes: a collaborative meta-analysis of individual participant data". In: *The lancet Diabetes & endocrinology* 3.7 (2015), pp. 514–525.

[114] Negar Memarian et al. "Multimodal data and machine learning for surgery outcome prediction in complicated cases of mesial temporal lobe epilepsy". In: *Computers in biology and medicine* 64 (2015), pp. 67–78.

[115] Natalie Meyer and Linda Z. Penn. "Reflecting on 25 years with MYC". In: *Nat Rev Cancer* 8.12 (2008), pp. 976–990.

[116] Joëlle Michaud et al. "HCFC1 is a common component of active human CpG-island promoters and coincides with ZNF143, THAP11, YY1, and GABP transcription factor occupancy". In: *Genome Research* 23.6 (2013), pp. 907–916.

[117] Martin Renqiang Min et al. "Interpretable sparse high-order boltzmann machines". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. 2014, pp. 614–622.

[118] Dariush Mozaffarian et al. "Heart disease and stroke statistics—2016 update: a report from the American Heart Association". In: *Circulation* (2015), CIR–0000000000000350.

[119] PS Myles et al. "Bispectral index monitoring to prevent awareness during anaesthesia: the B-Aware randomised controlled trial". In: *The lancet* 363.9423 (2004), pp. 1757–1763.

[120] Bala G Nair et al. "Intraoperative clinical decision support for anesthesia: a narrative review of available systems". In: *Anesthesia & Analgesia* 124.2 (2017), pp. 603–617.

[121] Mark Newman. *Networks: an introduction*. Oxford, UK: Oxford University Press, 2010.

[122] Wei Niu et al. "Diverse transcription factor binding features revealed by genome-wide ChIP-seq in *C. elegans*". In: *Genome Research* 21.2 (2011), pp. 245–254.

[123] Government Printing Office. "National Center for Health Statistics. Health, United States, 2016, With chartbook on long-term trends in health. No. 2017." In: (2017).

[124] World Health Organization. "Pulse Oximetry Training Manual". In: (2011).

[125] Paula F Orlandi et al. "Hematuria as a risk factor for progression of chronic kidney disease and death: findings from the Chronic Renal Insufficiency Cohort (CRIC) Study". In: *BMC nephrology* 19.1 (2018), p. 150.

[126] Sharon L. Paige et al. "A temporal chromatin signature in human embryonic stem cells identifies regulators of cardiac development". In: *Cell* 151.1 (2012), pp. 221–232.

[127] Feng Pan et al. "Feature selection for ranking using boosted trees". In: *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM. 2009, pp. 2025–2028.

[128] Aswini K. Panigrahi et al. "A cohesin-RAD21 interactome". In: *Biochemical Journal* 442.3 (2012), pp. 661–670.

[129] Vania Parelho et al. "Cohesins functionally associate with CTCF on mammalian chromosome arms". In: *Cell* 132.3 (2008), pp. 422–433.

[130] Brandon J Parker et al. "A transcriptional regulatory role of the THAP11–HCF-1 complex in colon cancer cell function". In: *Molecular and Cellular Biology* 32.9 (2012), pp. 1654–1670.

[131] Brandon J Parker et al. "Host Cell Factor-1 recruitment to E2F-bound and cell-cycle-control genes is mediated by THAP11 and ZNF143". In: *Cell Reports* 9.3 (2014), pp. 967–982.

[132] Jagruti H. Patel et al. "Analysis of genomic targets reveals complex functions of MYC". In: *Nat Rev Cancer* 4.7 (2004), pp. 562–568.

[133] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[134] Samuel Peña-Llopis et al. "BAP1 loss defines a new class of renal cell carcinoma". In: *Nature Genetics* 44.7 (2012), pp. 751–759.

[135] Juliane Perner et al. "Inference of interactions between chromatin modifiers and histone modifications: from ChIP-Seq data to chromatin-signaling". In: *Nucleic Acids Research* 42.22 (2014), pp. 13689–13695.

[136] H Mitchell Perry Jr et al. "Early predictors of 15-year end-stage renal disease in hypertensive patients". In: *Hypertension* 25.4 (1995), pp. 587–594.

[137] David Piluso, Patricia Bilan, and John P Capone. "Host cell factor-1 interacts with and antagonizes transactivation by the cell cycle regulatory factor Miz-1". In: *Journal of Biological Chemistry* 277.48 (2002), pp. 46799–46808.

[138] Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. "Model Agnostic Supervised Local Explanations". In: *Advances in Neural Information Processing Systems*. 2018, pp. 2520–2529.

[139] Liudmila Prokhorenkova et al. "CatBoost: unbiased boosting with categorical features". In: *Advances in Neural Information Processing Systems*. 2018, pp. 6637–6647.

[140] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-precision model-agnostic explanations". In: *AAAI Conference on Artificial Intelligence*. 2018.

[141] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD*. ACM. 2016, pp. 1135–1144.

[142] Stephanie M van Rooden et al. "The identification of Parkinson's disease subtypes using cluster analysis: a systematic review". In: *Movement disorders* 25.8 (2010), pp. 969–978.

[143] Steven J Rosansky et al. "The association of blood pressure levels and change in renal function in hypertensive and nonhypertensive subjects". In: *Archives of internal medicine* 150.10 (1990), pp. 2073–2076.

[144] Kate R Rosenbloom et al. "ENCODE data in the UCSC Genome Browser: year 5 update". In: *Nucleic Acids Research* 41.D1 (2013), pp. D56–D63.

[145] Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[146] A. Saabas. *treeinterpreter Python package*. URL: `https://github.com/andosa/treeinterpreter`.

[147] Marco Sandri and Paola Zuccolotto. "A bias correction algorithm for the Gini variable importance measure in classification trees". In: *Journal of Computational and Graphical Statistics* 17.3 (2008), pp. 611–628.

[148] Suchi Saria et al. "Integration of early physiological responses predicts later illness severity in preterm infants." In: *Science translational medicine* 2 48 (2010), 48ra65.

[149] Mark J Sarnak et al. "The effect of a lower target blood pressure on the progression of kidney disease: long-term follow-up of the modification of diet in renal disease study". In: *Annals of internal medicine* 142.5 (2005), pp. 342–351.

[150] Caroline A Schneider, Wayne S Rasband, and Kevin W Eliceiri. "NIH Image to ImageJ: 25 years of image analysis". In: *Nature Methods* 9.7 (2012), pp. 671–675.

[151] Lloyd S Shapley. "A value for n-person games". In: *Contributions to the Theory of Games* 2.28 (1953), pp. 307–317.

[152] Daria Shlyueva, Gerald Stampfel, and Alexander Stark. "Transcriptional enhancers: from properties to genome-wide predictions". In: *Nature Reviews Genetics* 15.4 (2014), pp. 272–286.

[153] Edward H Shortliffe and Martin J Sepúlveda. "Clinical Decision Support in the Era of Artificial Intelligence". In: *Jama* 320.21 (2018), pp. 2199–2200.

[154] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning Important Features Through Propagating Activation Differences". In: *arXiv preprint arXiv:1704.02685* (2017).

[155] Avanti Shrikumar et al. "Not Just a Black Box: Learning Important Features Through Propagating Activation Differences". In: *arXiv preprint arXiv:1605.01713* (2016).

[156] Neil B Shulman et al. "Prognostic value of serum creatinine and effect of treatment of hypertension on renal function. Results from the hypertension detection and follow-up program. The Hypertension Detection and Follow-up Program Cooperative Group." In: *Hypertension* 13.5 Suppl (1989), p. I80.

[157] Ola Söderberg et al. "Characterizing proteins and their interactions in cells and tissues using the in situ proximity ligation assay". In: *Methods* 45.3 (2008), pp. 227–232.

[158] Therese Sørlie et al. "Repeated observation of breast tumor subtypes in independent gene expression data sets". In: *Proceedings of the national academy of sciences* 100.14 (2003), pp. 8418–8423.

[159] François Spitz and Eileen EM Furlong. "Transcription factors: from enhancer binding to developmental control". In: *Nature Reviews Genetics* 13.9 (2012), pp. 613–626.

[160] Jost Tobias Springenberg et al. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).

[161] Chris Stark et al. "BioGRID: a general repository for interaction datasets". In: *Nucleic Acids Research* 34.suppl 1 (2006), pp. D535–D539.

[162] Bas van Steensel et al. "Bayesian network analysis of targeting interactions in chromatin". In: *Genome research* 20.2 (2010), pp. 190–200.

[163] Guy VG Stevens. "On the inverse of the covariance matrix in portfolio analysis". In: *The Journal of Finance* 53.5 (1998), pp. 1821–1827.

[164] L Strachan and DW Noble. "Hypoxia and surgical patients–prevention and treatment of an unnecessary cause of morbidity and mortality." In: *Journal of the Royal College of Surgeons of Edinburgh* 46.5 (2001), pp. 297–302.

[165] Carolin Strobl et al. "Bias in random forest variable importance measures: Illustrations, sources and a solution". In: *BMC bioinformatics* 8.1 (2007), p. 25.

[166] C Strobl et al. "Conditional variable importance for random forests". In: *BMC bioinformatics* 9.1 (2008), p. 307.

[167] Erik Štrumbelj and Igor Kononenko. "Explaining prediction models and individual predictions with feature contributions". In: *Knowledge and information systems* 41.3 (2014), pp. 647–665.

[168] Richard L. Summers et al. "Functionality of empirical model-based predictive analytics for the early detection of hemodynamic instabilty." In: *Biomedical sciences instrumentation* 50 (2014), pp. 219–24.

[169] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *arXiv preprint arXiv:1703.01365* (2017).

[170] Navdeep Tangri et al. "Multinational assessment of accuracy of equations for predicting risk of kidney failure: a meta-analysis". In: *Jama* 315.2 (2016), pp. 164–174.

[171] Lionel Tarassenko, Alexander Hann, and J. Duncan Young. "Integrated monitoring and analysis for early warning of patient deterioration." In: *British journal of anaesthesia* 97 1 (2006), pp. 64–8.

[172] Lance R Thomas et al. "Interaction with WDR5 promotes target gene recognition and tumorigenesis by MYC". In: *Molecular Cell* 58.3 (2015), pp. 440–452.

[173] LR Thomas et al. "Interaction of MYC with host cell factor-1 is mediated by the evolutionarily conserved Myc box IV motif". In: *Oncogene* (2015).

[174] Juan M Vaquerizas et al. “A census of human transcription factors: function, expression and evolution”. In: *Nature Reviews Genetics* 10.4 (2009), pp. 252–263.

[175] Emmanuelle Viré et al. “The Polycomb group protein EZH2 directly controls DNA methylation”. In: *Nature* 439.7078 (2006), pp. 871–874.

[176] W Gordon Walker et al. “Renal function change in hypertensive members of the Multiple Risk Factor Intervention Trial: racial and treatment effects”. In: *JAMA* 268.21 (1992), pp. 3085–3091.

[177] F Perry Wilson et al. “Urinary creatinine excretion, bioelectrical impedance analysis, and clinical outcomes in patients with CKD: the CRIC study”. In: *Clinical Journal of the American Society of Nephrology* 9.12 (2014), pp. 2095–2103.

[178] Winkelmann. *Correlated Count Data*. English. Springer Berlin Heidelberg, 2008, pp. 203–239. ISBN: 978-3-540-77648-2. DOI: 10.1007/978-3-540-78389-3_7. URL: http://dx.doi.org/10.1007/978-3-540-78389-3_7.

[179] H Peyton Young. “Monotonic solutions of cooperative games”. In: *International Journal of Game Theory* 14.2 (1985), pp. 65–72.

[180] Wen Yuan et al. “H3K36 methylation antagonizes PRC2-mediated H3K27 methylation”. In: *Journal of Biological Chemistry* 286.10 (2011), pp. 7983–7989.

[181] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[182] Daniel R Zerbino et al. “The Ensembl Regulatory Build”. In: *Genome Biology* 16.1 (2015), p. 56.

[183] Yong Zhang et al. “Model-based analysis of ChIP-Seq (MACS)”. In: *Genome Biology* 9.9 (2008), R137.

[184] Jian Zhou and Olga G Troyanskaya. “Global quantitative modeling of chromatin factor interactions”. In: *PLoS Computational Biology* 10.3 (2014), e1003525.

[185] Vicky W Zhou, Alon Goren, and Bradley E Bernstein. “Charting histone modifications and the functional organization of mammalian genomes”. In: *Nature Reviews Genetics* 12.1 (2011), pp. 7–18.

[186] Martin Zinkevich. *Rules of Machine Learning: Best Practices for ML Engineering*. 2017.

[187] Eytan Zlotorynski. “Chromatin: ZNF143 in the loop”. In: *Nature Reviews Molecular Cell Biology* 16.3 (2015), pp. 127–127.