Interacting with Predictions: Visual Inspection of Black-box Machine Learning Models

Josua Krause
New York University
New York, NY, USA
josua.krause@nyu.edu

Adam Perer
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
adam.perer@us.ibm.com

Kenney Ng
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
kenney.ng@us.ibm.com

ABSTRACT
Understanding predictive models, in terms of interpreting and identifying actionable insights, is a challenging task. Often the importance of a feature in a model is only a rough estimate condensed into one number. However, our research goes beyond these naïve estimates through the design and implementation of an interactive visual analytics system, *Prospector*. By providing interactive partial dependence diagnostics, data scientists can understand how features affect the prediction overall. In addition, our support for localized inspection allows data scientists to understand how and why specific datapoints are predicted as they are, as well as support for tweaking feature values and seeing how the prediction responds. Our system is then evaluated using a case study involving a team of data scientists improving predictive models for detecting the onset of diabetes from electronic medical records.

Author Keywords
interactive machine learning; predictive modeling; partial dependence; visual analytics; model visualization

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION
In the era of data-driven analytics, there is growing demand to generate and deploy predictive models in a variety of domains so that the patterns unearthed from massive amounts of data can be leveraged and converted into actionable insights. Predictive modeling is defined as the process of developing a mathematical tool or model that generates an accurate prediction [24]. As an example, in healthcare, if one can model the data characteristics of patients who will likely develop diabetes, healthcare institutions could deploy such a model on their patient databases, and automatically flag high risk patients to clinicians to make sure they are being treated appropriately. However, building such models on noisy, real-world data is quite challenging.

Data scientists often turn to machine learning, where the goal is to create predictive models based on information automatically learned from data with ground truth. However, these machine learning techniques are often black-boxes and may be selected based only on performance metrics such as high accuracy scores, and not necessarily based on the interpretability and actionable insights of the model. There has recently been a variety of techniques to inject humans-in-the-loop when building predictive models based on machine learning, from interactive training [1] to interactive feature selection [23]. However, interactive systems to effectively assess and evaluate the interpretability and actionable insights of a predictive model that go beyond simple accuracy metrics is still lacking. We believe bringing humans into the loop at this stage can possibly lead to better models and consequently improved outcomes when the models are deployed.

Our research aims to support data scientists to go beyond judging predictive models solely based on their accuracy scores by also including model interpretability and actionable insights. Towards this goal, we developed *Prospector*, a novel visual analytics system designed to help analysts better understand predictive models. *Prospector* leverages the concept of partial dependence, a diagnostic technique that was designed to communicate how features affect the prediction, and makes this technique fully interactive. *Prospector* also supports localized inspection, so users can understand why certain data results in a specific prediction, and even lets users hypothesize new data by tweaking values and seeing how the predictive model responds. We also demonstrate, through a case study, how the system can lead to important insights for clinical researchers building models that try to predict patient outcomes based on electronic medical records.

Concretely, our contributions include:

- A design and implementation of an interactive visual analytics system, *Prospector*, for assessing the interpretability and actionable insights of trained predictive models by supporting:
  - Interactive partial dependence diagnostics to understand how features affect the prediction overall. There

CHI'16, May 07 - 12, 2016, San Jose, CA, USA

ACM 978-1-4503-3362-7/16/05...$15.00
DOI: http://dx.doi.org/10.1145/2858036.2858529

are novel visual representations, sampling strategies, and support for comparing multiple models.

  - Localized inspection to understand how and why specific data points are predicted as they are. Users can interactively tweak feature values and see how the prediction responds, as well as find the most impactful features using a novel local feature importance metric.

- A case study of data scientists using *Prospector* to improve predictive models for detecting the onset of diabetes trained from electronic medical record data.

MOTIVATION

Machine Learning for Predictive Modeling

Data scientists often use machine learning to create predictive models based on known properties of training data, which acts as the ground truth. Machine learning algorithms typically work in two phases: the training phase and the prediction phase. In the training phase, parameters of the model are learned using training data. The prediction phase then computes predictions using the trained model. Below, we describe several machine learning algorithms that are commonly used and also utilized in the case study.

A decision tree is one such algorithm, which is a tree whose nodes are rules that decide how to proceed down the branches of the tree, according to the range of values for a specific feature. The decision making starts at the root of the tree and leaves carry the prediction results. Decision trees are popular in machine learning as they allow data scientists to model arbitrary functions. However, the more nodes a tree has, the harder it is to understand the reasoning behind outcomes. Logistic regression is another popular algorithm that is easier to understand, as features can only positively or negative influence the prediction, and the rate of influence is fixed. This is achieved by defining a hyper-plane in the feature vector space, where the outcome of the prediction depends on how close to and on which side of the hyper-plane a point is.

Another popular algorithm is random forests, which combine the output of multiple decision trees. A random forest is an example of an ensemble model, which combines the output of several weak machine learning models to yield an overall better result with less bias and variance than a single strong model. However, this makes ensemble models less interpretable since each weak model has only a small influence on the outcome.

Predictive Modeling in Healthcare

In order to make our contributions concrete, we utilize a motivating scenario that emerged from our case study. The case study involves a team of five data scientists interested in using predictive modeling on a longitudinal database of electronic medical records. The research team has a background in healthcare analytics and their database contains 4,000 patients from a major hospital in the United States. The team is interested in building a predictive model to predict if a patient is at risk of developing diabetes, a chronic disease of high blood sugar levels that may cause serious health complications.

Figure 1. An illustration of how partial dependence is computed for the feature "Glucose". On the left are four patients' original feature values. In the middle, the "Glucose" values are all changed to 100, and the corresponding predictions change. Similarly, on the right, the "Glucose" values are all changed to 130, and again the risks are different. This demonstrates the impact of the "Glucose" feature on risk prediction.

The team of data scientists manages to develop a highly accurate predictive model for detecting patients at high risk of developing diabetes. They determine its effectiveness by measuring the common metrics used by predictive models (e.g., accuracy and AUC scores [24]). They also followed the best practices of building predictive models. They worked with clinical researchers to properly define cohorts of patients with diabetes (cases) and matched patients without diabetes (controls) by thoroughly searching through the electronic medical records. They constructed features based on lab tests, diagnosis codes, procedures, demographics, and patient conditions from the records. They used cross-validation to ensure their models were robust. They used a variety of state-of-the-art feature selection methods to utilize the most informative features in the model while keeping it as generalizable as possible. And they used a variety of effective classifiers to do the training and evaluation. After trying various combinations of these techniques, the model with the highest accuracy metrics was selected and presented to the appropriate stakeholders at the healthcare institution.

Their stakeholders were impressed by the high accuracy scores of the model. But when they asked the data scientists for more information about what was inside of the model, the reports only described the top features of the model and their associated "importance" weights. The stakeholders recognized many of the feature names, and it appeared to make clinical sense. However, there were also some surprising ones that led to intellectual discussions. But the stakeholders demanded to know more. They wanted a clearer sense of how certain features impacted the prediction. Furthermore, they wanted to understand how the values associated with the features (e.g., the results associated with lab tests) impacted the prediction. They also were curious to interact with the model to test hypotheses about how the model would react to hypothetical patients of interest. When confronted with these questions, the data scientists shrugged. In the data scientist's defense, it is difficult to summarize and interpret complex models and there are few tools and techniques available to address the stakeholders' requests. So now the stakeholders are faced with a hard decision: do they deploy a predictive model in their institution that appears to have high accuracy but is also somewhat of a black-box?

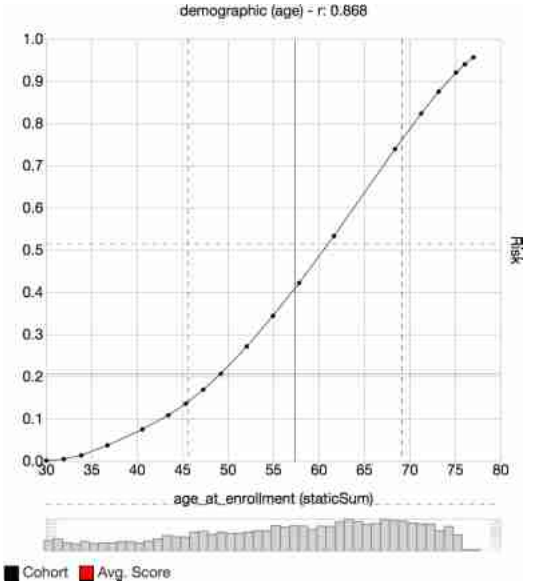

Figure 2. Partial dependence plots. The black curve shows the average predicted risk (probability of a certain outcome) if for all rows in the data the value of this feature was the given value of the horizontal axis. The red line shows the average predicted risk for the original data. The vertical line shows the mean of the observed values and the histogram below the plot shows the distribution of observed values. Dotted lines show the range of one standard deviation around the mean values.

Although this scenario is motivated by our case study, our other projects and interviews suggest these are not atypical requirements. Our work is motivated to support the development of more comprehensible predictive models.

Partial Dependence

The most widely used technique to understand the relationship between a feature and a prediction is by computing partial dependence [14, 16]. The idea of partial dependence plots is to treat predictive models as a black-box and observe how changes in the input affect prediction outcomes. When inspecting only the partial dependence of one input feature at a time, Formula (1) can be used to compute a partial dependence plot.

$$pdp_f(v) = \frac{1}{N}\sum_{i}^{N} pred(x_i) \text{ with } x_{if} = v \qquad (1)$$

N is the number of rows in the input matrix x, *pred* is the prediction function that takes one input row, a feature vector, and returns an outcome probability, and f is the feature used to compute the partial dependence plot. The formula computes the average outcome over all input rows while changing the value of feature f to the input value v for each row x_i. The original input data is kept fixed. This allows for observing the influence of f on the predicted probabilities.

In order to make this function more concrete, consider the illustrative example in Figure 1, where each input row is a patient, and each column is a feature. The last column represents the output of the predictive model that predicts if a patient is at low-risk or high-risk for developing diabetes. In Figure 1a, the patients' original feature values (age, BMI (Body Mass Index, a standard way to quantify obesity), and glucose level (a standard way to determine diabetes)) are shown. If one wants to examine the impact of the glucose feature on the prediction, partial dependence can be applied by keeping all of the other features (age, bmi) as they were, but fixing glucose to a set of fixed values to see how that feature impacts the prediction. For example, in Figure 1b, the glucose values (highlighted in yellow) are fixed to 100, which yields predictions of only 1 patient being high risk, instead of the original 2. Conversely, in Figure 1c, glucose is fixed to 180, and 3 patients are predicted to have high risk. Thus, there appears to be partial dependence of glucose on the prediction.

Partial dependence is typically visualized as a partial dependence plot, as shown in Figure 2,which is a line graph that plots the fixed values for the target feature on the x-axis, and the corresponding predicted risk (probability of a certain outcome) on the y-axis.

RELATED WORK

Motivation for Interpretability

Modern machine learning algorithms are able to create more reliable and precise models but they are increasingly complex and come with the price of being harder and harder to interpret (Breiman [7]). This inverse relation of understandability versus expressiveness of a model introduces the need to find ways to improve the interpretability of complex models to overcome this disadvantage. Lim [28] asks questions such as "*Why* did X happen?", "*Why* not Y?", "*What* happens *if* I do Z?", and "*How* do I make X happen?" to explain complex mechanisms like machine learning models. Our system allows users to interactively ask and answer such questions. Kulesza *et al.* [25] use explanatory debugging by conveying how a model came to its prediction in order to be able to correct mistakes. On the other hand, Patel *et al.* [33] use multiple classifiers to better understand input data. Steeg and Galstyan [45, 41] use total correlation to build a hierarchy of features explaining their role in the data.

Algorithm Specific Model Visualization

In the past, research has primarily focused on understanding and interacting with specific machine learning algorithms. Often the focus is on the internal weights of the trained models. For Bayesian networks, showing probabilities of the nodes (Becker *et al.* [5]) and how the input is propagated (Correa *et al.* [18]) has been used. For Support Vector Machines, projection techniques (Caragea *et al.* [8]) and Nomograms (Jakulin *et al.* [17]) to see the "cut" in the data points were utilized. Visualizing and interpreting the graph of a neural network has also been used by Tzeng and Ma [43]. Kim *et al.* [22, 20] introduce graphical models that allow for interpretability of its features. [20] use a colored matrix of features by category to show distinguishable features computed by their model. Caruana *et al.* [9] use high-performance generalized additive models (GA^2Ms) that allows visual inspection of the influence of its input features on the outcome much like partial dependence.

Model Result Visualization

However, showing only internal, algorithm specific weights is often not enough. Plate *et al.* [36] and Olden [31] show how input features influence the outcome of neural network classifications. Xu *et al.* [46] interprets the graph of a neural network used for image classification to retrieve which part of an image was responsible for a specific classification result. These techniques aim in the same direction as partial dependence but are limited to only neural networks. They cannot be used to support the ability to compare different machine learning models across algorithms.

Having access to the internals of a machine learning algorithm also allows direct interaction with the models and to improve them on the fly. BaobabView (van den Elzen and van Wijk [44]) offers interactive decision tree construction. Steed *et al.* [40, 39] guides regression model creation by enhancing interaction with the input data. EnsembleMatrix (Talbot *et al.* [42]) allows users to combine already computed classification models to build ensemble models. Some recent interactive machine learning tools [1, 2, 3, 4, 19, 21, 26, 29, 32]) are more algorithm agnostic, but depend on general performance measures like confusion matrices, area under ROC curve (AUC) measures, result distribution of data points, and feature weights according to model independent statistics.

Frank and Hall [13] use 2D projections of the data to show results for multiple classification algorithms, as well as Rheingans and desJardins [37] with self-organizing maps.

Probing Models

Partial dependence was proposed by Friedman [14] for analyzing gradient boosting models and has since been used for other models as well (*e.g.* Ehrlinger [12] uses partial dependence to analyze the behavior of random forests in the *R* package *ggRandomForests*).

Cortez and Embrechts [10, 11] and Kim *et al.* [22] use sensitivity analysis to analyze and compare machine learning models of different algorithms. Sensitivity analysis is similar to partial dependence except that it uses a few base vectors (usually the mean, median, or quartiles of all observed values) instead of computing the probabilities over all input points. This method is faster than partial dependence but may miss critical details about the prediction function especially if the function is strongly non-linear.

Goldstein *et al.* [15] extends the idea of partial dependence by using Individual Conditional Expectation (ICE) plots which show one line for each row of the input data. We found, however, that this often clutters the plots too much and makes them harder to interpret. We experimented further with showing standard deviations and quartiles of the partial dependence line but discarded this approach since the spread of the partial dependence results is always expected to be large unless one feature dominates the classification significantly and is able to solely change the classification even for the furthest data points.

By not restricting ourselves to sampling only the observed input space, our approach on partial dependence enables a deeper analysis of the machine learning model. Furthermore, accepting the costs of computing partial dependence over all input points yields proper results even for highly non-linear models, while also not overwhelming users with too much detail. This is strengthened even further by our novel approach of using implementation details of the inspected models to improve the sampling and the representation of the results.

SYSTEM

In order to integrate partial dependence and localized inspection into our pipeline we propose *Prospector*, a web-based interface. The server side of *Prospector* can load any machine learning model accessible via python, or integrate with existing predictive modeling pipelines such as PARAMO [30]. Although this paper demonstrates the system on clinical data, the tool is able to handle predictive models for other domains. For example, the tool has also been used with exploring models that predict real estate prices, as well as classic data sets from the UCI Machine Learning Repository [27].

In this section, we first describe *Prospector*'s novel enhancements of partial dependence plots. Then, we describe how *Prospector* leverages partial dependence to support localized inspection. Finally, we describe the workflow of how these techniques are integrated into *Prospector*'s UI.

Partial Dependence Plots

Partial dependence is typically visualized as a partial dependence plot, which is a line graph that plots the fixed values for the target feature on the horizontal axis, and the corresponding predicted risk (probability of a certain outcome) on the vertical axis. In *Prospector*, we enhance the plot by adding a red reference line with the average predicted risk of the model on the original data, as shown in Figure 2. In addition, a black vertical line indicates the average observed value of the input data for the current feature. Both reference lines are accompanied with dotted lines showing one standard deviation in both directions from the mean value. To help with validating insights and assigning importance, a histogram of the observed values in the original data is also shown below the plot.

Sampling Partial Dependence

One of our core contributions is the ability to effectively treat partial dependence as a black-box for inspection. However, naïvely treating the predictive model as black-box may lead to inaccuracies in the generated plot. Often only observed input values are used for sampling which leaves the prediction function for other values undefined, even though those values might be of particular interest for understanding the internals of the model. Furthermore, the interpolation between those sample points may ignore inherent features of the prediction function. For example, Ehrlinger [12] shows the usage of partial dependence plots via random forest models. The prediction function for those models can only change between thresholds appearing in the nodes of the trees. For all other values the prediction remains constant. However, the interpolation between sample points used in the examples is polynomial. This leads to the following misrepresentations of the prediction function: (i) some sampled prediction values are not included in the interpolation; (ii) the interpolation is a curve where it should be a constant which alludes to values

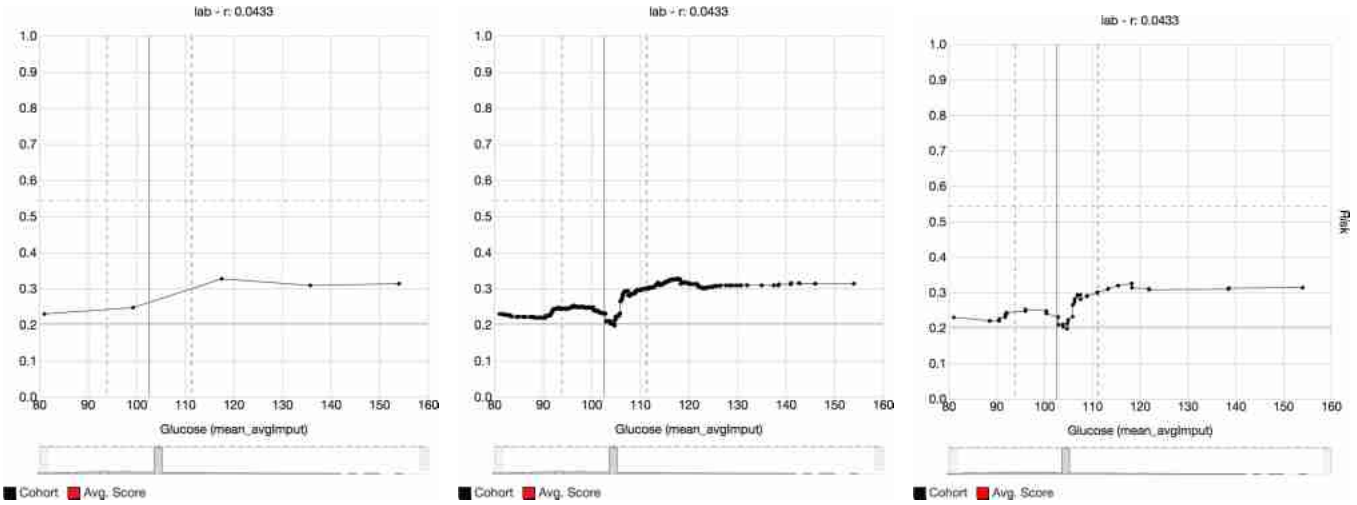

Figure 3. Different sampling strategies for partial dependence plots. The leftmost plot uses a naïve sampling which misses the dip in the predicted risk for a Glucose value around 105. Using the thresholds of the trees in the random forest the middle plot shows all details of the displayed model. The rightmost plot simplifies this by detecting co-linear points and summarizing them into lines improving readability. The dip in the predicted risk is due to imputation of missing values to the mean of the observed values. This increase in local noise shifts the prediction towards the overall average predicted risk (the horizontal red line). Most patients have never had their Glucose value measured.

that are impossible to achieve with the prediction function; (iii) steps are interpolated as curves which gives the impression of a smooth ascent of values when it should be a series of sudden jumps.

We overcome those inaccuracies by acknowledging inherent properties of the prediction functions of our models. This is only possible by leveraging the internal design of model algorithms and therefore, *Prospector* must more effectively sample the range of the input features. For example, in decision tree or forest models, where the predicted risk will not change between thresholds of the nodes of those models' trees *Prospector* utilizes the knowledge that the plot will be a step function by only sampling values at the thresholds of the given feature to accurately compute the complete plot (see Figure 3). It does this by inspecting the decision rules in the nodes of the model to retrieve those thresholds.

However some models, such as random forest models, produce a large number of points where the outcome might change which leads to a cluttered plot that impairs readability. One solution to this is to simplify the generated plot by finding almost co-linear points and reducing them to the endpoints. Such visual optimizations support more comprehensible plots that are easier to read while still being accurate representations. Other machine learning algorithms may not require such optimizations.

Prospector also enhances partial dependence plots by taking into account the context of the original data values. For example, certain features only make sense as integer values (*e.g.* the number of times a laboratory test was performed) and it does not make sense to show non-integer values in the plots. Such features can be heuristically detected by inspecting the set of values in the original data set and *Prospector* restricts those features to only have integer values as input. Similarly, in the partial dependence plot, only integer values are computed. Furthermore, for predictive models using step functions the plot is horizontally shifted by 0.5 so that jumps happen between values. This eases reading the actual value at the integer points. Even though this improvement leads to a slight misrepresentation of the prediction function for non-observable values the readability of the plot is significantly improved to support user tasks. For non-integer data types, these optimizations are not necessary.

Local Inspection

Our second core contribution is leveraging our implementation of partial dependence to support the user task of local inspection. Users can use *Prospector* to inspect specific data points of interest and see how the models predict how they behave. In addition, if the users are curious about how a particular data point's risk might change if it had different values, a user can explore this as well. The idea of localized inspection is illustrated in Figure 4 using our running example of diabetes prediction. At the top, the original patient's feature values are shown, along with the patient's original predicted low risk of having diabetes. Suppose the analyst was curious to see how the patient's risk would change if his BMI was

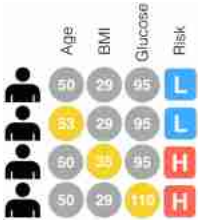

Figure 4. This illustration provides an explanation of how local inspection works. On the top row are the patient's original feature values and the corresponding prediction. On the bottom three rows, users changed certain values of the patient, highlighted in yellow, and such values impacted the prediction.

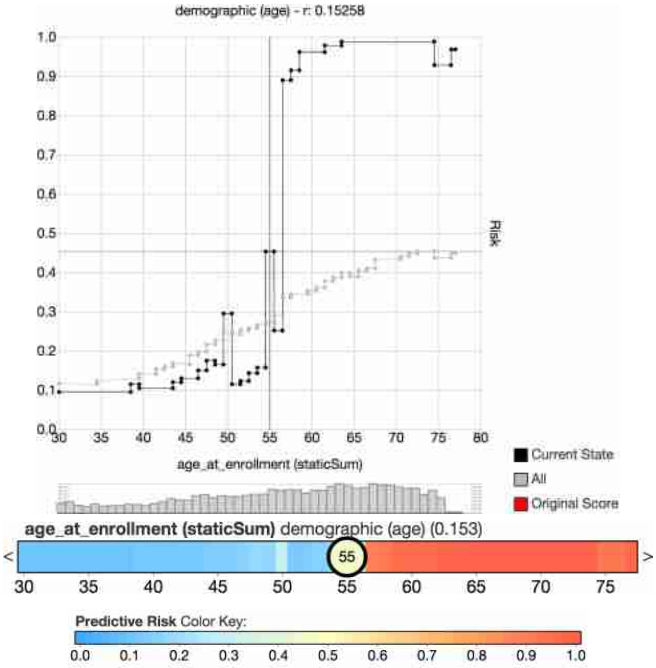

Figure 5. The same feature shown as line plot (top) and partial dependence bar (middle). Color indicates the predicted risk for the outcome. The color mapping is shown at the bottom.

increased to 35. Localized inspection allows users to interactively change this value, and see the corresponding prediction. In order to streamline this kind of exploration we fully compute the predicted risk for all values of BMI similar to partial dependence. As seen in Figure 4 we do this for all features independently yielding local partial dependence plots for each feature using a single input row.

Partial Dependence Bars

In order to increase interactivity, encourage exploration, and display a larger number of features at once, we use a novel visual encoding, *partial dependence bars*, a color bar representation of a partial dependence plot, shown in Figure 5. The horizontal axis of a partial dependence bar represents the range of values of a feature, and the color at a given position represents the predicted risk for that value. Color is mapped to a three-point color scale, where blue represents low risk, yellow represents medium risk, and red represents high risk. As these bars are meant to aid local inspection, the current feature value of the datapoint being inspected is positioned along the horizontal axis and annotated as a circular label. Users can drag this circular label left or right to inspect how changes in the feature value affect the predicted risk as well as the local partial dependence of other features.

Local feature importance

The fourth novel contribution of our research is a technique to simplify the exploration of the predicted risk space by automatically finding features where a small change in value yields a significantly large change on the predicted risk.

While manipulating values of specific features allows users to test hypotheses on how features of interest may impact the prediction, if users wish to simply understand how to most impact the prediction, manipulating features one-by-one to test impact is an inefficient process. Instead, *Prospector* can employ local feature importance, a novel technique that computes the impact of a given feature on the model according to the current values. This localized feature importance comes in two different flavors: as a feature importance number and as actual suggestions for value changes.

A straight-forward way to define a localized importance of features is to look at the range of possible predicted risks the feature can create starting from the given data point. Formula (2) computes the local importance I of a given feature f for the given feature vector p. It sums up the entirety of changes in outcomes for all values v for feature f. The outcome changes are weighted by ω the likelihood of changing the value from p_f to v. p^* is the modified feature vector where its value for f is set to v and *pred* is the prediction function.

$$I_f(p) = \int_{-\infty}^{\infty} \left[pred(p^*) - pred(p)\right]\ \omega(v,\ p_f)\ dv \qquad (2)$$

$$\text{with } p^*_f = v \text{ and } p^*_g = p_g \text{ for } g \neq f$$

$$\omega(v,\ p_f) = \frac{1}{\sigma_f\sqrt{2\pi}} \exp\left(-\frac{(v-p_f)^2}{2\sigma_f^2}\right)$$

In order for different features to be comparable, ω takes the distribution of values in the input data into account. In features with a high spread, a larger change is more likely than in a feature with a narrow value range. We model the likelihood of the change using a normal distribution with the reference value p_f as mean and the standard deviation σ_f of the observed values of f as standard deviation. Ordering the features according to this local importance yields features that are likely to decrease the predicted risk first, then features that have a low impact on the predicted risk, and finally features that are likely to increase the predicted risk.

Instead of computing local feature importance for all possible changes, it is more practically useful to compute the importance according to the most *impactful* change for a feature. An impactful change is the smallest change needed to have the largest change in the predicted outcome. Note that this is different from the slope of the function since an impactful change might be after a valley or ridge. Again, in order to have comparable results, the distribution of values in the input data is taken into account.

$$\arg\max_v \left[s\ \left[pred(p^*) - pred(p)\right]\ \omega(v, p_f)\right] \qquad (3)$$

Formula (3) finds the most impactful change of a feature. s is either 1 or -1 depending on whether to search for the largest increase or decrease. All other variables are the same as in Formula (2). The changes yielding the highest impact can be interpreted as suggestions for changing a data point.

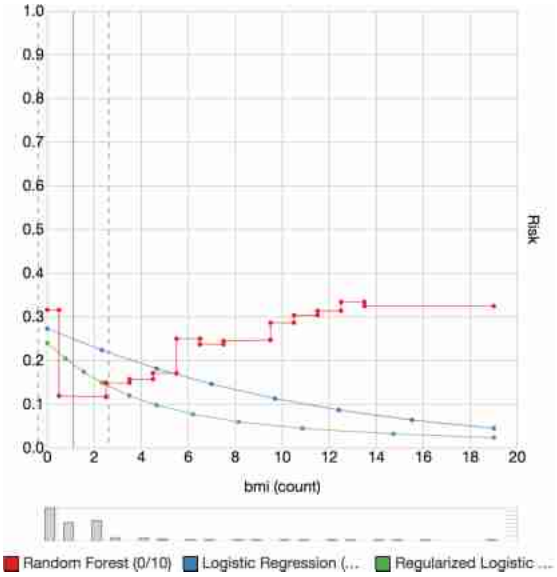

Figure 6. Comparison of three machine learning models on the number of measured BMI values. The two regression models (logistic regression in blue and regularized logistic regression in green) can express only a single slope (downwards or upwards) whereas the random forest in red can model the strong decrease in predicted risk going from no BMI measures to one measure as well as the later increase again if a patient has several BMI measures. The random forest is more expressive, but the distribution of input values in the histogram below the plot hint the model might be overfitting as most of the observed values are 2 or less.

Comparing Multiple Models

Prospector also supports plotting multiple models in the same plot. As the input domain and the output range are the same across different machine learning models on the same data, partial dependence plots can also be used to compare multiple models as shown in Figure 6. This is useful for comparing the expressiveness of models and seeing which models are possibly under- or over-fitting the input data.

Workflow

In order to support the workflow of clinical researchers, as described above in the Motivation, *Prospector*'s UI is organized into three main tabs: patient selection, patient inspection, and partial dependence plots.

Patient Selection

The patient selection tab allows users to find patients they may want to inspect, based on their ground truth (*e.g.* whether they actually had diabetes) and their predicted risk (*e.g.* assessed likelihood by the predictive model of having the disease). *Prospector* provides a visual summary of the patient population by providing a patient selection visualization. The visualization, as seen in Figure 7, consists of two columns dividing the population according to their ground truth (case patients being those actually diagnosed with diabetes and control patients being not). Each column is then separated into bins of predicted probabilities in steps of 0.1 which can be clicked to select the group of patients fitting those criteria. For instance, if a case patient was predicted with a low risk score, that patient would appear in the top of the right column. If a bin is too small to provide a clickable area, a box at the side of the column is displayed to allow choosing even small populations. The selected population is then shown next to the visualization with the individual prediction results shown for each entry.

In order to get more details about patients before selecting, users can hover over a patient in the list and see a summary for the patient, as shown in Figure 8. In addition to the predicted risk and the ground truth, the interface shows the top 5 most impactful features, for both increasing and decreasing predicted risk, according to the local feature importance described above. For each impactful feature, the original data value is shown as well as the suggested change and what the resulting predicted risk would be if such a change was made. This summary provides a preview of how amenable a particular patient's predicted risk is to changing and which features are mostly responsible for their current predicted risk.

Patient Inspection

After users select a patient of interest, the UI switches to the patient inspection tab with the selected patient's data loaded, as shown in Figure 9. All of the features' partial dependence bars are shown in a scrollable list, with the patient's feature values selected with circular labels. Users can drag the circular label to change the value of any feature and see the predicted risk change in real-time. Users can also select a feature and see the corresponding typical partial dependence plot of the feature. In this plot the local partial dependence of the current patient is shown as black curve and the global partial dependence of the whole population is shown in gray. The partial dependence plot is also clickable and users can change the feature value here as well, changing the black vertical line that, in this plot, shows the current value.

Users can change the sort order of the partial dependence bars by using the buttons at the top. In addition to sorting by the feature weight and relevance as determined by the predictive model, users can also sort according to our local feature importance and impactful changes as described above. If impactful changes are chosen as the sort order, the suggested changes to each feature are indicated with a white circular label in the partial dependence bar, as shown in the bottom left of Figure 9.

Often after analysts have inspected a particular patient, they may wish to find other patients similar to them to see how they react to the predictive model. If users wish to find patients similar to the one they are inspecting, they can click on the "Neighborhood" button and *Prospector* will automatically find the closest patients to the current set of values using feature-wise Euclidean distance. This similar set of patients are then used as the population in the patient selection tab that users can navigate.

Partial Dependence Plots

If users wish to browse the global partial dependence plots of a feature of interest, they can navigate to the third tab. Users can view multiple models at once by using a combo-box at the top of the user interface to select the models they wish to view in the plot. Each selected model is assigned a unique color using a quantitative color scale, and a color-coded key

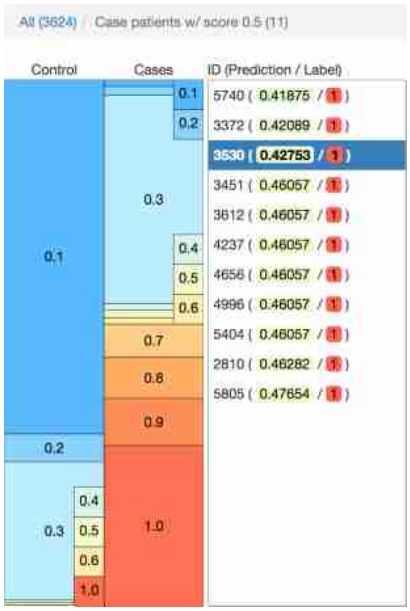

Figure 7. The interface for selecting a patient. The left side shows the distribution of patients within different ranges of predicted risk. The columns indicate the ground truth. On the right side a list shows the patients of the currently selected range.

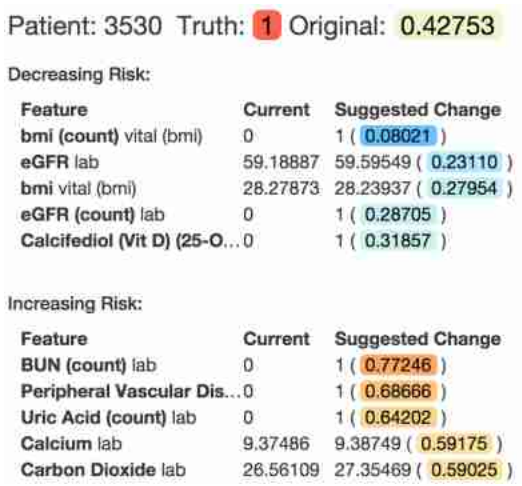

Figure 8. The summary of one patient. The header line indicates the patient id, the ground truth, and the predicted risk. For both decreasing and increasing the predicted risk the top 5 most impactful features are shown. Each feature shows its current value and the suggested change with the highest impact along with how the predicted risk would change.

is displayed beneath the plot. If more than one model is selected, the red "Avg. Score" helper line is not shown. Users can also filter the global population to a subset population of interest by using a predicted probability bin or the "Neighborhood" of a patient in the patient selection tab. This alters the plot accordingly allowing for a more focused analysis for *e.g.* mispredicted patients, outliers, or patient neighborhoods.

CASE STUDY: PREDICTING DIABETES

In order to evaluate the utility of *Prospector*, we chose to conduct a case study utilizing a team of real data scientists building their own predictive models on their own real-world datasets to demonstrate its effectiveness at reaching insights in practice. There is a growing belief in the visualization community that traditional evaluation metrics (e.g. measuring task time completion or number of errors) are often insufficient to evaluate visualization systems [6, 35, 38]. Using the evaluation methodology developed by Perer and Shneiderman [34], we conducted a 4-month long-term case study with a team of five data scientists interested in using predictive modeling on a longitudinal database of electronic medical records. The research team is interested in building a predictive model to predict if a patient is at risk of developing diabetes using a database of 4,000 patients from a major hospital in the United States. Due to sensitive data agreements, this team wished to remain anonymous.

The initial phase of the case study involved understanding the data scientists' current tools and needs. They presented their typical results after building predictive models, sharing stories of success when their stakeholders were pleased, as well as examples of less successful results when their stakeholders demanded answers they couldn't provide with existing tools. Their use cases and experiences shaped the design and requirements of the tool.

After the tool was developed, there were bi-weekly meetings with the data science team in which we discussed the current interface and identified shortcomings of the interface, necessary UI enhancements, and components that were not worth developing further. Some of the elements originally proposed turned out not to be useful, such as overlaying distributions of risk in the partial dependence plots or using ICE plots [15]. However, focusing the meetings on examining the team's predictive models together using *Prospector* allowed us to determine which elements helped improve both the models and their comprehension of the models.

Understanding Model Classes

Initially, the data scientists were unsure which type of predictive models to build. Although they had used simpler models in the past, such as decision trees and logistic regression, they were eager to use random forest models due the promise of higher accuracy but were worried about how interpretable the results would be. After building models using both logistic regression and random forests, they were curious to use partial dependence plots to understand the trade-offs of both approaches. An example of such trade-offs can be seen in Figure 6, which is a partial dependence plot of two logistic regression models and one random forest. Interestingly, for this feature (which refers to the number of times patients got their BMI recorded), the model types disagree substantially for higher values in the plot. This is surprising since the inspected feature is most important for all models and all models perform equally well using standard statistics like accuracy or AUC.

While all three models have a decrease in average predicted risk when the count goes from 0 to 2, the logistic regression models continue to trend downward. However, the random forest model (in red), illustrates the predictive risk increases

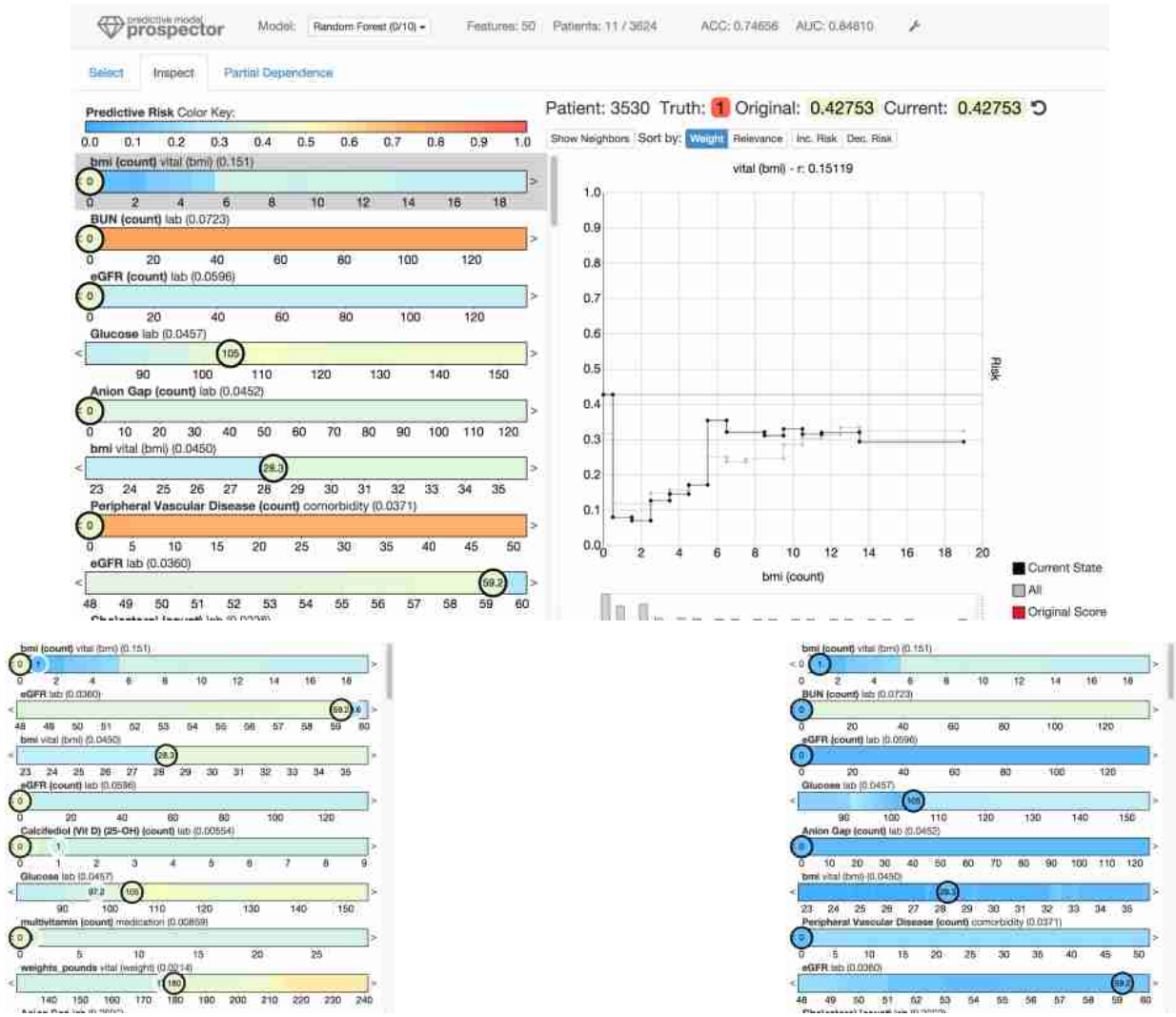

Figure 9. The user interface of *Prospector* is shown at the top. The bottom left shows suggestions on what changes (white circles) would decrease the predicted risk the most. The bottom right shows how the color plots change due to changing a value (namely changing the bmi value from 0 to 1). Fully white circles show the original value of the given patient.

as the count gets higher than two. The data scientists were surprised to learn how differently the model classes treated this feature, but using the tool, they were able to devise a two-fold explanation. On one hand, logistic regression models are not expressive enough to model the late increase after the initial drop, as logistic regression models are bound by a single curve. On the other hand, most of the observed data points of the feature are zero, one, and two while the higher values occur extremely rare, as the histogram below the plot clearly illustrates. This led the data scientist team to question whether to model everything as precisely as possible or using a simpler model for the sake of generality.

Unexpected Effects of Data Imputation

Due to limitations of their database, many of the patients were missing Glucose lab test results. During the feature construction phase, the team made a decision that in order to work around the missing values, each patient who did not have a value would be given the average observed value of all other patients. This imputation technique is popular among predictive modelers, as simply removing all patients without such data would make the data quite small. However, once the data science team began to explore the Glucose feature in the tool, as shown in Figure 3, they began to realize the dramatic effects their imputation strategy can have. Due to the imputation, patients that are either cases or controls often have the same lab test values which increases the noise of the predicted risk. The partial dependence plot illustrates that, as noise increases, the predicted probability gets closer to the population average leading to a valley in the machine learning model. Exploring this feature in *Prospector* suggested that a better strategy for handling missing values would be needed to overcome this problem.

The Need for Localized Inspection

Discussions in bi-weekly interviews also led to the development of the localized inspection of patients which aims to answer the following questions:

1. What impact does a feature have on an actual patient?
2. Does the model behave correctly on a case-to-case basis?
3. What are the most important features for a given patient?
4. Why are certain patients not being accurately predicted?
5. Can we identify high impact actionable features?

The last question about identifying actionable features was of particular importance to the data science team. They were interested to know if the model could be used to learn features that could be acted upon by the patients or their doctors to reduce the risk of diabetes. However, the data science team were disappointed to learn, via *Prospector*, that many of the highest ranking features were not actionable. For instance, some of the most predictive features for a high risk of diabetes involved having a high count of the number of lab tests. Informing patients to get fewer lab tests would likely not correlate to lower risk of diabetes. Instead, these lab test counts were likely a proxy for other features that correlated to more complicated or more sickly patients seeing their doctors more regularly and thus getting more lab tests. Other demographic features that were highly predictive, like age, simply have no intervention as well. The data science team then reconsidered which features should be a part of the predictive model, by creating features that are actionable and omitting others. Of course, the model cannot know this by itself – no matter what sophisticated feature selection algorithms are used – so the access *Prospector* provides is critical for this process.

Impact into Data Scientists' Workflow

In addition to learning new insights about their predictive models, the tool also impacted the team's workflow. Prior to *Prospector*, after each new predictive model was built, a data scientist would manually generate a set of reports describing the model. Typically, this would involve exporting a list of the model's top features and their weights, and generating a bar chart for the other team members to review. They would present these bar chart summaries during review meetings and discuss if the model seemed sensible enough. If they believe the list of features made sense, they would then present this chart to their stakeholders. If it didn't, they would brainstorm how to improve the predictive model (such as changing the classification or feature selection algorithms) and then repeat this process. While this manual approach led to the deployment of predictive models in the past, many iterations were required and understanding impactful values of features were rarely considered.

Once *Prospector* was integrated into their workflow, many of these shortcomings were overcome. No longer did a data scientist need to generate a set of manual reports. Instead, the predictive model can be loaded into the tool directly. Since the tool is interactive, it also allows the team to ask questions that may have not been considered when static charts were created. The tool also allowed them to ask questions beyond the top features that contributed to the models. They could ask more patient-centric questions such as "Why is this patient not being classified correctly?" by drilling down to incorrectly predicted patients and exploring the most impactful features for them. Beyond exploration, *Prospector* was also used to communicate models to the stakeholders, which allowed stakeholders to ask questions and see the results in real-time. This rapid feedback helped gain support for deploying predictive models in future projects. As a result of these successes, *Prospector* is now a part of their predictive modeling workflow and is used for other work than predicting the onset of diabetes.

CONCLUSION AND DISCUSSION

In this paper, we demonstrated how the design and implementation of an interactive visual analytics system, *Prospector*, can help data scientists assess the interpretability and actionable insights of trained predictive models. *Prospector* accomplishes this by supporting interactive partial dependence diagnostics for understanding how features affect the prediction overall by featuring novel visual representations, sampling strategies, and support for comparing multiple models. Furthermore, *Prospector* supports localized inspection so data scientists can understand how and why specific datapoints are predicted as they are. With support to interactively tweak feature values and see how the prediction responds, as well as finding the most impactful features using a novel local feature importance metric, data scientists can interact with models on a deeper level than possible with common tools. Finally, we presented a case study, in the spirit of #chi4good, which involved a team of data scientists using *Prospector* to improve predictive models for detecting the onset of diabetes. Their extended use of the tool led to better predictive models, as well as better communication of their models to their stakeholders.

Despite the novel features of *Prospector* and its successful case study, there is still much future work to continue to give users full comprehension of predictive models. *Prospector* relies on partial dependence for one input feature at a time, but this approach relies on the orthogonality of input features. However, in real-world data, this is not often the case, as features may be correlated. *Prospector* can only model changes along one axis at a time as it cannot take correlations or influences between features into account. We plan to address this limitation in future work by modeling valid sets of input points and visualizing how they react to changes in one or more features. Another limitation is that *Prospector* was built to view predictive models after they had been built using users' own predictive modeling pipeline of choice. However, this flexibility limits the ability for users to directly impact their predictive models based on insights reached during exploration. Also, *Prospector* can only handle single-class predictions, but we plan to extend this functionality to multi-class predictions in the future. Our future work also intends to integrate *Prospector* more directly into the predictive modeling pipeline so users can directly modify features for feature construction and feature selection and see how their models improve in a single user interface. Despite these limitations, providing users with advanced visual tools to inspect black-boxes of machine learning shows great promise and helps users comprehend and retain control of their predictive models without sacrificing accuracy.

REFERENCES

1. Saleema Amershi, Max Chickering, Steven M. Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. ModelTracker: Redesigning Performance Analysis Tools for Machine Learning. (April 2015), 337–346.
2. Saleema Amershi, James Fogarty, Ashish Kapoor, and Tan Desney. 2011a. Designing for Effective End-user Interaction with Machine Learning. In *Proceedings of the 24th Annual ACM Symposium Adjunct on User Interface Software and Technology (UIST '11 Adjunct)*. ACM, New York, NY, USA, 47–50.
3. Saleema Amershi, James Fogarty, and Daniel Weld. 2012. Regroup: Interactive Machine Learning for On-demand Group Creation in Social Networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 21–30.
4. Saleema Amershi, Bongshin Lee, Ashish Kapoor, Ratul Mahajan, and Blaine Christian. 2011b. CueT: Human-guided Fast and Accurate Network Alarm Triage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 157–166.
5. Barry Becker, Ron Kohavi, and Dan Sommerfield. 2002. Information Visualization in Data Mining and Knowledge Discovery. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Chapter Visualizing the Simple Baysian Classifier, 237–249.
6. Enrico Bertini, Heidi Lam, and Adam Perer. 2011. Summaries: a special issue on evaluation for information visualization. *Information Visualization* 10, 3 (2011).
7. Leo Breiman. 2001. Statistical Modeling: The Two Cultures. *Statist. Sci.* 16, 3 (08 2001), 199–231.
8. Doina Caragea, Dianne Cook, and Vasant Honavar. 2001. Gaining Insights into Support Vector Machine Pattern Classifiers Using Projection-Based Tour Methods. In *Proceedings of the KDD Conference*. 251–256.
9. Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 1721–1730.
10. Paulo Cortez and Mark J. Embrechts. 2011. Opening black box Data Mining models using Sensitivity Analysis. In *Computational Intelligence and Data Mining (CIDM), IEEE Symposium on*. 341–348.
11. Paulo Cortez and Mark J. Embrechts. 2013. Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences* 225 (2013), 1–17.
12. John Ehrlinger. 2015. ggRandomForests: Random Forests for Regression. (2015). R package version 1.1.4.
13. Eibe Frank and Mark Hall. 2003. Visualizing Class Probability Estimators. In *Knowledge Discovery in Databases (PKDD)*, Nada Lavra, Dragan Gamberger, Ljupo Todorovski, and Hendrik Blockeel (Eds.). Lecture Notes in Computer Science, Vol. 2838. Springer Berlin Heidelberg, 168–179.
14. Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29, 5 (10 2001), 1189–1232.
15. Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. 2014. Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. *Journal of Computational and Graphical Statistics* (March 2014).
16. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The elements of statistical learning: data mining, inference, and prediction*. Springer Verlag.
17. Aleks Jakulin, Martin Možina, Janez Demšar, Ivan Bratko, and Blaž Zupan. 2005. Nomograms for Visualizing Support Vector Machines. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD '05)*. ACM, New York, NY, USA, 108–117.
18. David Correa Martins Jr., Evaldo A. de Oliveira, Ulisses de Mendonça Braga Neto, Ronaldo Fumio Hashimoto, and Roberto Marcondes Cesar Jr. 2013. Signal propagation in Bayesian networks and its relationship with intrinsically multivariate predictive variables. *Information Sciences* 225 (March 2013), 18–34.
19. Ashish Kapoor, Bongshin Lee, Desney Tan, and Eric Horvitz. 2010. Interactive Optimization for Steering Machine Classification. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10)*. ACM, New York, NY, USA, 1343–1352.
20. Been Kim, Finale Doshi-Velez, and Julie A Shah. 2015a. Mind the Gap: A Generative Approach to Interpretable Feature Selection and Extraction. In *Advances in Neural Information Processing Systems*.
21. Been Kim, Kayur Patel, Afshin Rostamizadeh, and Julie Shah. 2015b. Scalable and interpretable data representation for high-dimensional complex data. In *Association for the Advancement of Artificial Intelligence (AAAI)*.
22. Been Kim, Cynthia Rudin, and Julie A Shah. 2014. The Bayesian Case Model: A Generative Approach for Case-Based Reasoning and Prototype Classification. In *Advances in Neural Information Processing Systems*. 1952–1960.
23. Josua Krause, Adam Perer, and Enrico Bertini. 2014. INFUSE: Interactive Feature Selection for Predictive Modeling of High Dimensional Data. *Visualization and Computer Graphics, IEEE Transactions on* 20, 12 (Dec 2014), 1614–1623.
24. Max Kuhn and Kjell Johnson. 2013. *Applied Predictive Modeling*. Springer London, Limited.

25. Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of Explanatory Debugging to Personalize Interactive Machine Learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI '15)*. ACM, New York, NY, USA, 126–137.
26. Carson K. Leung and Kyle W. Joseph. 2014. Sports Data Mining: Predicting Results for the College Football Games. *Procedia Computer Science* 35 (September 2014), 710–719. Knowledge-Based and Intelligent Information & Engineering Systems 18th Annual Conference (KES), Proceedings of.
27. Moshe Lichman and Kevin Bache. 2013. UCI Machine Learning Repository. (2013). `http://archive.ics.uci.edu/ml`
28. Brian Y. Lim. 2012. *Improving Understanding and Trust with Intelligibility in Context-aware Applications*. Ph.D. Dissertation. Pittsburgh, PA, USA. Advisor(s) Dey, Anind K. AAI3524680.
29. Shubhanshu Mishra, Jana Diesner, Jason Byrne, and Elizabeth Surbeck. 2015. Sentiment Analysis with Incremental Human-in-the-Loop Learning and Lexical Resource Customization. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media (HT '15)*. ACM, New York, NY, USA, 323–325.
30. Kenney Ng, Amol Ghoting, Steven R. Steinhubl, Walter F. Stewart, Bradley Malin, and Jimeng Sun. 2013. PARAMO: A PARAllel predictive MOdeling platform for healthcare analytic research using electronic health records. *Journal of Biomedical Informatics* (2013).
31. Julian D. Olden and Donald A. Jackson. 2002. Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling* 154, 12 (2002), 135–150.
32. Kayur Patel, Naomi Bancroft, Steven Drucker, James Fogarty, and James Landay. 2010. Gestalt: Integrated Support for Implementation and Analysis in Machine Learning. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology (UIST)*. ACM, New York, NY, 37–46.
33. Kayur Patel, Steven M. Drucker, James Fogarty, Ashish Kapoor, and Desney S. Tan. 2011. Using Multiple Models to Understand Data. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, Toby Walsh (Ed.). IJCAI/AAAI, 1723–1728.
34. Adam Perer and Ben Shneiderman. 2008. Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*. 265–274.
35. Catherine Plaisant. 2004. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*. ACM, 109–116.
36. Tony Plate, Joel Bert, John Grace, and Pierre Band. 1997. Visualizing the Function Computed by a Feedforward Neural Network. *Progress in Connectionist-Based Information Systems: Proceedings of The Fourth International Conference on Neural Information Processing (ICONIP'97)* 1 (1997), 306–309.
37. Penny Rheingans and Marie desJardins. 2000. Visualizing high-dimensional predictive model quality. In *Visualization*. 493–496.
38. Ben Shneiderman and Catherine Plaisant. 2006. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Proceedings of the 2006 BELIV Workshop*. 1–7.
39. Chad A. Steed, Patrick J. Fitzpatrick, J. Edward Swan II, and T.J. Jankun-Kelly. 2009a. Tropical Cyclone Trend Analysis Using Enhanced Parallel Coordinates and Statistical Analytics. *Cartography and Geographic Information Science* 36, 3 (2009), 251–265.
40. Chad A. Steed, J. Edward Swan II, T.J. Jankun-Kelly, and Patrick J. Fitzpatrick. 2009b. Guided analysis of hurricane trends using statistical processes integrated with interactive parallel coordinates. In *Visual Analytics Science and Technology (VAST), IEEE Symposium on*. 19–26.
41. Greg Ver Steeg and Aram Galstyan. 2015. Maximally Informative Hierarchical Representations of High-Dimensional Data. In *AISTATS'15*.
42. Justin Talbot, Bongshin Lee, Ashish Kapoor, and Desney S. Tan. 2009. EnsembleMatrix: Interactive Visualization to Support Machine Learning with Multiple Classifiers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1283–1292.
43. Fan-Yin Tzeng and Kwan-Liu Ma. 2005. Opening the Black Box - Data Driven Visualization of Neural Networks. In *Proceedings of IEEE Visualization '05 Conference*. IEEE, 383–390.
44. Stef van den Elzen and Jarke J. van Wijk. 2011. BaobabView: Interactive construction and analysis of decision trees. In *Visual Analytics Science and Technology (VAST), IEEE Conference on*. 151–160.
45. Greg Ver Steeg and Aram Galstyan. 2014. Discovering Structure in High-Dimensional Data Through Correlation Explanation. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger (Eds.). Curran Associates, Inc., 577–585.
46. Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 2048–2057.