USING VISUAL ANALYTICS TO EXPLAIN BLACK-BOX MACHINE LEARNING

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Josua Walter Hugo Krause

May 2018

USING VISUAL ANALYTICS TO EXPLAIN BLACK-BOX MACHINE LEARNING

DISSERTATION

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

DOCTOR OF PHILOSOPHY (Computer Science)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Josua Walter Hugo Krause

May 2018

Approved:

____
Department Chair Signature

____
Date

University ID#:	**N11707681**
Net ID#:	**jk4560**

Approved by the Guidance Committee:

Major: Computer Science

Enrico Bertini
Associate Professor of
Computer Science and Engineering

Date

Cláudio T. Silva
Professor of
Computer Science and Engineering

Date

Yindalon Aphinyanaphongs
Assistant Professor of
Population Health and Medicine

Date

Adam Perer
Research Scientist at
IBM Research

Date

Vita

Josua Walter Hugo Krause was born in Isny im Allgäu, Germany in January 1988. He earned his Bachelor's degree in Information Engineering from the University of Konstanz in Germany. The topic of his Bachelor's thesis was about graphically annotating changes in temporally evolving node link diagrams. He expanded this topic to using comic book principles in his Master's thesis. He obtained his Master's degree in Information Engineering also from the University of Konstanz. In September 2013, he started to pursue his graduate studies towards his Ph.D. under the supervision of Prof. Enrico Bertini. His research interests were focused on using visualization to support analyzing machine learning. He published his research on this topic at VAST and CHI, some of the most prestigious conferences in the field of visual analytics and human-computer interaction.

Acknowledgements

This thesis would have not been possible without the support from my advisor, Prof. Enrico Bertini. Thank you for the numerous discussions about my research, being there whenever I had doubts, supporting my good ideas, and dismissing my bad ideas. You are an awesome advisor and I would do my Ph.D. again under your supervision.

I thank our collaborators and co-authors for the exciting paths we ventured. Thank you Adam Perer, Aritra Dasgupta, Prof. Yindalon Aphinyanaphongs, Jordan Swartz, Kenney Ng, and many more for your advice and inputs throughout our collaborations. I would also like to extend my gratitude to Prof. Cláudio Silva, Prof. Juliana Freire, Prof. Rumi Chunara, and Prof. Lisa Hellerstein for their guidance, experience, and feedback at various stages of my Ph.D.

Thanks, Adam and Aritra, for the amazing internships at both IBM and PNNL. I learned a lot. A special thanks goes to my lab-mates, Anshul, Cristian, Jay, and Jun for their constant support, feedback, and always having an open ear for discussions. Also, I thank our interns Paolo and Livia. I wish you all the best. Furthermore, I thank every current or former member of the VIDA family with whom I spent many hours and made so many friends. Thank you Rémi, Vicky, Fernando, Fabio, Kien, Tuan-Anh, Harish, Aline, Bowen, Sungmin, Gromit, Masayo, Yamuna, Raoni, Jorge, Felipe, Aecio, Sonia, and so many others whom I might have missed to mention. I extend my warmest thanks to all the administrators who stayed on top of the administrative processes, making sure that I can focus on my research without any distractions. Thank you Ann, Susana, Judy, Kari, and Eve for your support all these years.

Additionally, I thank Hendrik for encouraging and supporting the best decision of my life, and Narges for being herself. Also, I thank Clément, Nandini, Jeremy, Pauline, Daniel, Leonard, Ingrid and her family, my Volleyball team, Jason, Keishi, Michael, Pavol, Veronica, Dunia, Cerille, Thierry, and many others for fun times and keeping my sanity during the years of my Ph.D. On a personal note, I thank my parents for their love and support throughout my life. And my siblings, Vanessa, Michaela, Swen, and the rest of the family for showing me the important things in life. You are the best.

Josua Krause

May 2018

To my parents for setting up the right gradient to follow.

ABSTRACT

USING VISUAL ANALYTICS TO EXPLAIN BLACK-BOX MACHINE LEARNING

by

Josua Walter Hugo Krause

Advisor: Prof. Enrico Bertini, Ph.D.

Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer Science)

May 2018

As machine learning models increase in complexity, the human ability to understand and interpret decisions made by those models has not been able to keep up. The usefulness of white-box analysis techniques, exposing the internal state of models, is limited to relatively simple models and has trouble with complex models, such as deep neural networks. Recently, black-box machine learning analysis techniques offer model independent insights into the decision making process of machine learning models. In order to quickly and effectively gain insights from those techniques, visual analytics emerges as a powerful set of tools. We use visual analytics to explore both global and local means of explaining and understanding predictive models via black-box techniques. We then propose the Model Diagnostic workflow that uses aggregated instance-level explanations to overcome problems of fully global or local methods. That is, by avoiding global aggregates, finer details of the decision making process are retained, while going beyond individual instances, analysts are not overwhelmed by the quantity of instances to inspect. Finally, we show that the Model Diagnostic workflow can not only help improving the models themselves but offers insights about flaws in the *input data*, thus helping with the task of feature engineering.

Contents

Vita iv

Acknowledgements v

Abstract viii

List of Figures xvii

List of Tables xviii

1 Introduction **1**

1.1 Machine Learning and Predictive Modeling 2

1.1.1 Popular Algorithms 5

1.1.2 Failure Modes 12

1.2 Visual Analytics 14

1.3 Use Case: Machine Learning for Health Care 16

1.4 Motivation 18

1.5 Research Overview and Thesis Contributions 23

1.5.1 INFUSE 24

1.5.2 Prospector 25

1.5.3 Model Diagnostic Workflow 26

1.5.4 User Study on Aggregating Explanations 28

1.6 Thesis Outline 29

2 INFUSE **30**

2.1 Motivation . . . 34

2.1.1 Predictive Modeling in Health Care . . . 34

2.1.2 Running Example: Diabetes Prediction . . . 36

2.1.3 Task Analysis . . . 38

2.2 Related Work . . . 40

2.2.1 Visual Feature Selection . . . 40

2.2.2 Visualization in Predictive Modeling . . . 42

2.3 INFUSE . . . 44

2.3.1 Data and Design . . . 44

2.3.2 Feature View . . . 46

2.3.3 List View . . . 51

2.3.4 Classifier View . . . 53

2.3.5 Interactive Model Builder . . . 54

2.4 Case Study . . . 56

2.4.1 Insight 1: Data Issues . . . 56

2.4.2 Insight 2: Clinically Relevant Features . . . 58

2.5 Future Work and Conclusion . . . 59

2.6 General Discussion . . . 61

3 Prospector **63**

3.1 Motivation . . . 66

3.1.1 Machine Learning for Predictive Modeling . . . 66

3.1.2 Predictive Modeling in Health Care . . . 67

3.1.3 Partial Dependence . . . 70

3.2 Related Work . . . 72

3.2.1 Motivation for Interpretability 72

3.2.2 Algorithm Specific Model Visualization 72

3.2.3 Model Result Visualization 73

3.2.4 Probing Models 74

3.3 System 75

3.3.1 Partial Dependence Plots 76

3.3.2 Local Inspection 79

3.3.3 Workflow 83

3.4 Case Study: Predicting Diabetes 87

3.4.1 Understanding Model Classes 91

3.4.2 Unexpected Effects of Data Imputation 92

3.4.3 The Need for Localized Inspection 92

3.4.4 Impact on Data Scientists' Workflow 94

3.5 Conclusion and Discussion 95

3.6 General Discussion 96

4 Explainer **98**

4.1 Related Work 103

4.1.1 Model Explanations: *Why* and *How* 103

4.1.2 Human-in-the-Loop Inspection of Classifiers 104

4.2 Model Diagnostics 106

4.2.1 User Goals 106

4.2.2 Workflow 108

4.3 Explanation Method 110

4.4 Visual Interface 114

4.4.1 Statistical Summary View 115

4.4.2 Explanation Explorer . . . 117
4.4.3 Item Level Inspector . . . 121
4.5 Case Study: Analysis of Medical Outcomes . . . 123
4.6 Discussion . . . 133
4.7 Conclusion & Future Work . . . 138
4.8 General Discussion . . . 138

5 User Study on Aggregating Instance-level Explanations 140

5.1 Introduction . . . 141
5.2 Related Work . . . 144
5.2.1 Effectiveness of Instance-level Explanations . . . 145
5.2.2 Detecting Biases Using Visual Analytics Methods . . . 146
5.3 Motivation . . . 147
5.4 Design Contribution . . . 149
5.5 Experimental Setup . . . 150
5.5.1 User Interface Conditions . . . 151
5.5.2 Study Design . . . 155
5.5.3 Tasks and Measurements . . . 156
5.5.4 Participants . . . 158
5.6 Results . . . 158
5.6.1 Bias Detection and Trust . . . 159
5.6.2 Comparison Across Conditions . . . 161
5.7 Discussion . . . 163
5.8 Conclusion & Future Work . . . 164
5.9 General Discussion . . . 165

6 Thesis Summary **167**

6.1 Limitations and Assumptions of Black-Box Analysis 168

6.2 Generalization to Other Forms of Machine Learning 170

6.3 Implications 170

7 Conclusion & Future Work **172**

List of Figures

1.1 Example of a confusion matrix. 3
1.2 Example of a ROC curve. 5
1.3 Influence function for "age" of a Generalized Additive Model. . . . 8
1.4 Example Decision Tree. 9
1.5 Anscombe's quartet. 15
1.6 Influence functions of Pneumonia model. 19
1.7 Programmable circuit to automatically design oscillators. 20
1.8 The adversarial patch. 21
1.9 The model diagnostic workflow. 26

2.1 Overview of *INFUSE*. 33
2.2 Steps of a typical predictive modeling pipeline. 34
2.3 Breakdown of the user interface of *INFUSE*. 35
2.4 How to read the glyph representation of *INFUSE*. 44
2.5 Different glyph designs. 45
2.6 Different axis combinations for the scatter-plot layout. 47
2.7 The Classifier View. 52
2.8 A cell of the Classifier View. 53

2.9 *INFUSE*'s Interactive Model Builder. 55

2.10 The scatter-plot view allows to compare multiple types of rankings. 56

2.11 The clinical researchers found an interesting pattern among the glyphs. 60

3.1 An illustration of how partial dependence is computed. 68

3.2 Partial dependence plots. 70

3.3 Different sampling strategies for partial dependence plots. 75

3.4 Explanation of how local inspection works. 79

3.5 The same feature shown as line plot and partial dependence bar. . . 80

3.6 Comparison of three machine learning models. 84

3.7 The interface for selecting a patient. 88

3.8 The summary of one patient. 89

3.9 The user interface of *Prospector*. 90

4.1 The Model Diagnostics workflow. 102

4.2 The Statistical Summary View. 115

4.3 The Explanation Explorer. 116

4.4 The Item Level Inspector. 124

4.5 Showing different orders in the Explanation Explorer. 126

4.6 Showing the second dataset of the case study. 133

5.1 Showing the four study conditions. 143

5.2 The full interface illustrating the aggregated histogram view. 144

5.3 Comparing the distribution of values by "Actual Label". 148

5.4 The full interface illustrating the table view. 152

5.5 Comparison of different subset selections on both models. 154

5.6 Trust and whether the model makes sense. 159

5.7 Change in trust and whether the model makes sense. 160

5.8 Comparison of participants' trust and ability to detect the bias. . . 161

5.9 Interactions with the table and histogram views. 162

5.10 Overall completion time of the study by condition. 163

List of Tables

1.1 Comparison of black-box and white-box explanation strategies. . . . 22
1.2 Locality of decision analyses of the presented approaches. 27

Chapter 1

Introduction

In computer science, machine learning is a set of methods for learning models from examples that accurately perform tasks on new or unseen data without explicitly codifying all instructions. This automation of programming provides a time- and cost-effective alternative to otherwise manually created software solutions, making machine learning popular and widely used for many applications.

However, the relative lack of human supervision in their creation makes it hard to fully understand the inner workings of trained models and limits the ability to verify that the models work correctly. Even though it is possible to statistically verify the correctness of a model by testing it against an unseen data set whose ground truth is known, models can unknowingly utilize latent variables that should not be used. Knowledge about those factors is often implicit in the domain of the task and neither encoded in the data nor the model. Human expertise is needed to detect and encode them explicitly or to block them out.

With the growing complexity of machine learning models and the need for algorithm-agnostic approaches, black-box explanations offer insights into the external behavior of such models. Visual analytics can be used to leverage the full potential of those explanations in a scalable way.

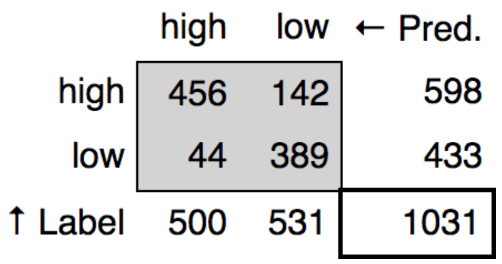
This thesis explores both global and local approaches for black-box explanations. Furthermore, it shows the advantage of the compromise of both techniques: aggregating instance-level explanations. This results in the Model Diagnostic workflow which not only helps improving machine learning models, but also offers insights about problems with the input data, assisting in the task of feature engineering.

Following, we will discuss essentials in both machine learning, starting in Section 1.1, and visual analytics, in Section 1.2. Then, in Section 1.3, we will briefly discuss the common use case, of machine learning for health care, within this thesis. Afterwards, we will motivate the work of the thesis in Section 1.4 followed by an overview of the research and contributions of the thesis in Section 1.5. In Section 1.6 we will then outline the remainder of this thesis.

1.1 Machine Learning and Predictive Modeling

Machine learning enables computers to perform tasks without manually codifying all instructions. This automated programming makes machine learning a popular and widely used approach for many applications as it provides a time- and cost-effective alternative to manually created software solutions. Machine learning relies on algorithms that learn and generalize from examples in order to perform accurately on new or unseen data.

Predictive modeling is a subcategory of machine learning that aims to predict certain properties from given input data. The input data typically consists of *instances* with a given set of *features*. The values of those features can be binary,



	high	low	← Pred.
high	456	142	598
low	44	389	433
↑ Label	500	531	1031

Figure 1.1: Example of a confusion matrix. Each cell represents the number of instances under a certain condition. Rows show the ground truth while columns show the predicted outcomes from the machine learning model. For example, there were 44 instances predicted "high" whose ground truth is actually "low". The edge of the matrix shows the tallies of the rows and columns.

categorical, or numerical. We can think of the data as a table where features are columns and rows are instances. Thus, this kind of data is typically referred to as *tabular* or *structured* data. In the case of *unstructured* data, such as plain text, the data needs to be converted into structured data first, before it can be used as input of predictive models[1]. The predicted property, *outcome* or *label*, can be numerical, in which case the task is called *regression*, or categorical, in which case the task is called *classification*[2]. The different values the label can assume are called *classes*. The task is call *binary classification* if the categorical label has exactly two classes.

The predictive model learns by *training* with a set of example instances (*training set*) and their associated labels. In order to verify that the model did indeed correctly learn from the given examples, a second, previously unseen, set of example instances (testing or *validation set*) with known labels is typically used. The model is used to predict the labels of this set which are then compared to the actual labels of

[1]In this case, the model input corresponds to a fixed-length transformed representation of the original input data. These transformations can be either learned (*i.e.*, via deep learning or similar methods) or are manually specified.

[2]A combination of multiple properties are also possible to predict, in which case the task is called multitask learning, structured learning, *etc.*. Currently, the most common form of predictive modeling involves either a single classification or single regression task.

those instances. Various statistical measures can be computed on this relationship between the predicted and actual labels [1].

For binary classification tasks (for simplicity *positive* and *negative* are used below to describe the outcomes) those statistical measures include:

True Positive / Negative The number of correctly predicted positive (TP) or negative instances (TN).

False Positive / Negative The number of incorrectly predicted positive (FP) or negative instances (FN).

Accuracy $(TP + TN)/(TP + TN + FP + FN)$ The proportion of correctly predicted instances.

Precision $TP/(TP + FP)$ The ratio of correctly predicted positive instances to instances which were predicted to be positive.

True Positive Rate / Recall $TP/(TP + FN)$ The proportion of correctly predicted positive instances.

False Positive Rate $FP/(FP + TN)$ The proportion of incorrectly predicted positive instances.

All of those measures can be read from a *confusion matrix*, as shown in Figure 1.1.

Additionally, a further model quality measurement can be derived from the confidence of a model in its prediction. Typically, classification models allow to output probabilities for each class instead of the predicted class directly. This allows for inspecting the *confidence* of the model. In the case of binary classifiers a *threshold* can be used to adjust predictions to favor confidence in a certain class higher than the other. This is achieved by outputting the positive class as prediction if its probability is higher than the threshold and the negative class otherwise.

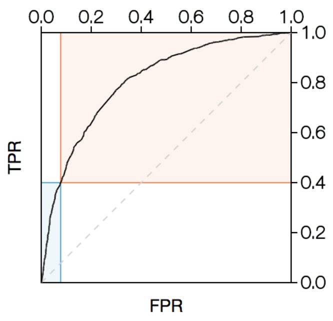


Figure 1.2: Example of a ROC curve. The plot indicates the correctness of the model with the false positive rate as horizontal axis and the true positive rate as vertical axis. The line traces those values for different prediction thresholds. The threshold minimizing incorrect predictions is highlighted by the colored axes.

The *receiver operating characteristic* (ROC) curve for binary classifiers plots the true positive rate against the false positive rate for all thresholds between 0 and 1, as shown in Figure 1.2. The plot indicates the correctness of the model *w.r.t.* its confidence. The ROC curve for a perfect model (100% accuracy) would follow the left vertical axis and the top horizontal axis. The area under the ROC curve (AUC) is a commonly used measure for model quality [19].

1.1.1 Popular Algorithms

There are too many algorithms for predictive models to list all of them here. However, following we will discuss the most commonly used algorithms.

1.1.1.1 Logistic Regression

Logistic Regression [1] is one of the most popular algorithms in machine learning. This is due to its simplicity and its interpretability of the results. Suppose x_i is the value of the feature i of our data, a binary classification using Logistic Regression is equivalent to:

$$l(x) = b + \sum_{i \in \mathcal{C}} w_i x_i$$

where $\mathcal{C}$ is the set of features, b and w_i are the learned weights of the model, and the outcome is determined by whether the weighted sum $l(x)$ is larger or smaller than zero. In order to work with probabilities, have greater precision around the boundary, and to penalize points close to the boundary, a logistic function is used:

$$P(Y = 1|x) = L(x) = \frac{1}{1 + e^{-l(x)}}$$

which gives the likelihood of one of the classes. In order to train the model, *i.e.*, finding the best values of b and w_i, the expression $P(Y = y|x)$, where y represents the ground truth 0 or 1 of the training instances, needs to be maximized. In practice, instead of maximizing $P(Y = y|x)$, the negative logarithm of this expression is minimized. When optimized over the entire training data, this expression is called *negative log likelihood* and it can be optimized via techniques such as gradient descent, to find optimal values of b and w_i.

One of the main advantages of logistic regression is its interpretability. The weights w_i can be directly interpreted as how influential a particular feature is. That is, if the magnitude of a weight is large, a small change in the value x_i of the feature has a big impact on the overall sum. Additionally, the sign of a weight indicates whether a particular feature is positively or negatively correlated with the outcome. For example, a simple (not necessarily correct) predictor for Type 2

Diabetes[3] could look like:

$$l_{\text{diabetes}}(x) = -1.9 + 0.01x_{\text{age}} + 0.02x_{\text{mass}} + (-0.3)x_{\text{height}}$$

with x_{age} in years, x_{mass} in kg, and x_{height} in m. From here, we can see that the model assumes that a high mass leads to higher Diabetes risk, old age also leads to a higher risk but less so than high mass, and large body height leads to a smaller risk. This gives a very intuitive understanding of how the model is reaching its conclusion.

However, this example also illustrates some limitations of logistic regression models. BMI (Body Mass Index; $x_{\text{mass}}/x^2_{\text{height}}$) is a better indicator than mass or height alone, but cannot be expressed or learned by a logistic regression model[4]. Additionally, since the influence of the features is always the same, one feature can force a prediction if its values are large enough. For example, the model will predict a high Diabetes risk with high age independent of the mass or height.

1.1.1.2 Generalized Additive Models

A Generalized Additive Model (GAM [47]) extends the idea of logistic regression to functions instead of weights:

$$g(x) = b + \sum_{i \in \mathcal{C}} f_i(x_i)$$

[3]A disease, affecting the body's ability to regulate blood sugar, which is mostly driven by lifestyle choices and associated with obesity and hypertension. The most common age of onset is between ∼45 and ∼65.

[4]Note, that we could provide BMI directly but only because domain experts had inferred the importance of this relationship in the past. Alternatively, we could also provide features in logarithmic space enabling logistic regression models to learn this relationship, since logarithmic BMI is $\log(x_{\text{mass}}) - 2.0 \log(x_{\text{height}})$.

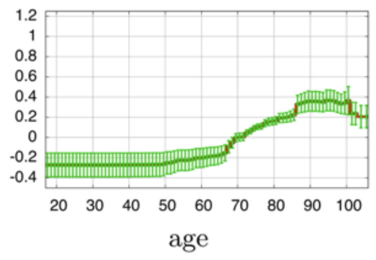


Figure 1.3: An influence function for “age” of a Generalized Additive Model [25].

where $\mathcal{C}$ is the set of features, b is the learned bias of the model, f_i are learned influence functions, and the outcome is determined by whether the sum $g(x)$ is larger or smaller than zero. Having influence functions overcomes the restricted expressiveness of logistic regression models while maintaining its interpretability. The impact of individual features can be seen in the plots of the influence functions.

In the Diabetes example from above, a GAM could solve the issue of always connecting high age to high Diabetes risk by decreasing the influence of the feature age for higher values than ∼65. However, since GAMs only consider one feature at a time, the BMI ($x_{\text{mass}}/x^2_{\text{height}}$) can still not be expressed or learned by the model.

1.1.1.3 Decision Trees

Instead of computing predictions with a mathematical formula, Decision Trees [1] determine predictions by following a sequence of tests on the data. Decision Trees are trees where each node represents a test $x_i > t_i$, with x_i as the value of the feature i and t_i as learned threshold. The prediction algorithm starts at the root of the tree. Depending on the outcome of the test in the root node the algorithm continues with the corresponding sub-tree recursively, until a leaf node is reached. The leaf nodes contain the probabilities for the outcomes of the prediction task and are returned as result by the algorithm.

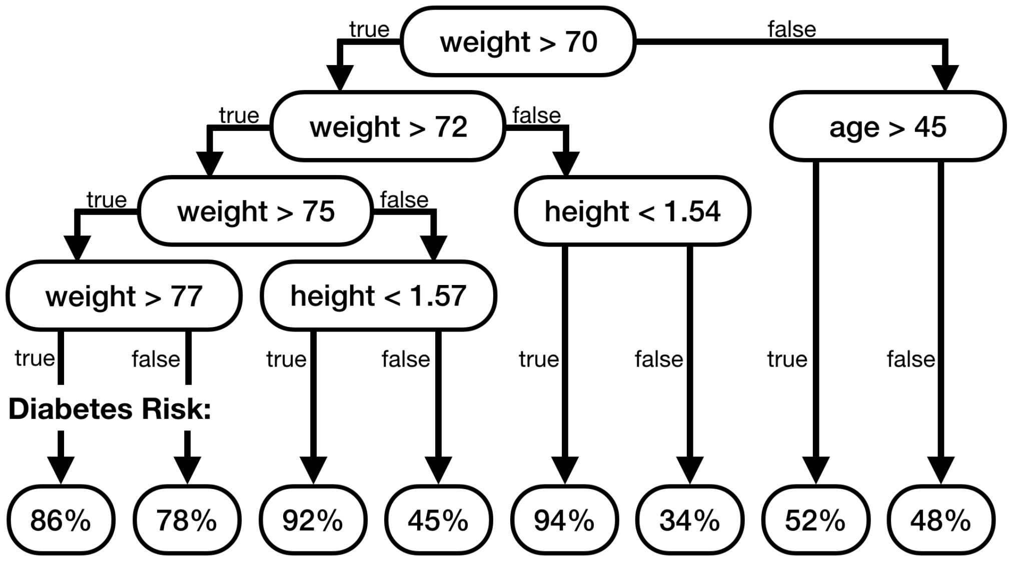


Figure 1.4: A possible decision tree for the Diabetes prediction example[5]. Note, that the tree has interval checks to model the relationship between “weight” in kg and “height” in m.

Decision Trees are usually considered very interpretable, as their behaviour is intuitively understandable. However, this only holds true for trees with a relatively low number of nodes. Trees with hundreds or thousands of nodes are hard to interpret as their complexity makes it hard to gain a mental model.

Decision Trees excel in cases where certain ranges of features have a clear impact on the outcome. For example, in our running Diabetes use case, the risk of becoming diabetic is significantly higher for ages between ∼45 and ∼65. A Decision Tree can then differentiate by age and infer the prediction differently for values inside or outside of that range.

However, Decision Trees cannot model relationships between features very well and have to approximate them. For example, increasing the height of a person also increases that person’s weight. This means that whether a person is overweight,

[5]The tree and its parameters are *not* based on any data and serve only illustratory purposes.

not only depends on the weight of the person, but also how tall that person is. A person weighing 80kg can be considered overweight with a height of 1.7m but would not be with a height of 1.8m. Since a Decision Tree cannot model this relationship ($x_{\text{mass}}/x_{\text{height}}^2 \geq 25$) directly it has to approximate it by breaking the values of x_{mass} and x_{height} into intervals and checking them separately (see Figure 1.4). Because of this, Decision Trees are likely to over-fit on the training data. Over-fitting happens when a machine learning model does not generalize well because of a pattern in the data that is only present in the training data.

1.1.1.4 Ensemble Methods

A powerful method of utilizing several weak models are Ensemble Models. Instead of taking the prediction of one model, Ensembles combine the prediction of many models in order to get a definite result. This is typically done either by using the average prediction of multiple models or by majority voting, *i.e.*, the most common prediction among the individual models is used as final prediction [1].

The main advantage of Ensemble Models is that it can improve the performance of otherwise worse performing models. Given a set of independent models with uncorrelated errors the resulting error reduction of the Ensemble is proportional to the number of individual models [43].

One popular Ensemble Model is the Random Forest [20], which combines several Decision Trees trained on random sub-samples of the training data. This overcomes the tendency of Decision Trees to over-fit on their training data. Another advantage of Random Forests is that they typically produce good results on only few training instances already.

1.1.1.5 Neural Networks

With the recent advance of more powerful computation hardware, especially with highly parallelizable GPUs[6], and availability of larger data sets, (Deep) Neural Networks have become a popular class of machine learning models. The basic building block of a Neural Network is a *layer*, that is a mathematical construct that has N input features and M output features. A layer can also have stored coefficients called *weights*.

A very common layer is the fully-connected layer. For each output feature or *node*, it computes a weighted sum of all input features, much like Linear Regression:

$$y_j(x) = b_j + \sum_{i=1}^{N} w_{ij} x_i$$

where the y_j represents the jth output feature, b_j is the corresponding bias and w_{ij} the corresponding weights.

Other common layers are ReLU (Rectified Linear Unit) and Softmax which both require $N = M$. ReLU computes as $y_j(x) = \max(0, x_j)$ which acts as filter to let only positive values through. Softmax computes as $y_j(x) = e^{x_j} / \sum_{i=1}^{N} e^{x_i}$ and scales the input vector in a way that the output vector sums to one.

With those layers we can build a Multi-Layer-Perceptron (MLP [50]) classifier. A MLP chains a number of fully-connected layers and ReLUs (or other non-linear functions) to each other. This can be interpreted as a sequence of 1. transform the input, 2. filter out negative values, 3. repeat. Softmax is used as final layer to compute the actual classification. Each node here represents one of the classes of the classifier and the class with the highest value "wins".

[6]Graphics Processing Unit

Training a MLP or any Neural Network works as follows. We initialize all weights with random numbers. For each instance of the training data set, we first compute the outcome with the current weights. Then we compute the gradient of the loss function, (which can be either the negative log likelihood of training data labels for classification or the mean of squared errors for regression tasks), with respect to each parameter. The gradient is computed on the predicted outcome $y(x)$ ($i.e.$, values of the final layer) and the *desired* outcome $\hat{y}(x)$ ($i.e.$, value of desired class set to one and zero otherwise). Then, going in reverse order we numerically compute this gradient for each layer and nudge the weights of those layers into the direction of the gradient. This procedure is called *back-propagation*. Ideally, we repeat back-propagation until all weights converge towards fixed numbers, at which point the model is trained.

Neural networks are a powerful machine learning tool, as they enable any function to be modeled [53] and allow for great flexibility through combining different layers. Multi-Layer Perceptrons can potentially be used on our running Diabetes example from above, as they do not have the limitations of any of the previously discussed models. However, Neural Networks typically require many thousand training examples in order to properly converge towards a performant model, making them a sub-optimal choice for tasks where acquiring data is time-consuming or expensive. Additionally, the large number of (hyper-)parameters also makes them difficult to tune.

1.1.2 Failure Modes

Machine learning models do not always perform perfectly. For a given data set, failures to do so typically fall into two main categories: under- and over-fitting [1].

Under-fitting occurs when a model does not have enough *capacity* to fully learn the input data. For example, a Logistic Regression model will never be able to learn to classify whether a point lies inside or outside of a circle, using Cartesian coordinates as input features. This is due to the linear nature of the model. Under-fitting can be prevented by using a more powerful model or increasing the capacity of the current model.

Over-fitting occurs when a model learns its training data set too precisely. For example, a Decision Tree might create one leaf node for every instance in the training data memorizing the outcomes of the training data in full. In this case the model likely will not generalize to new, unseen data. Thus, in order to detect over-fitting, it is common to keep a separate data set, called *hold-out*, *validation*, or *test* data set, at hand that is not used for training the model. The trained model is then tested against this data set. If the model performs well on the training data but not on the test data, then the model is over-fitting. A way to prevent this is to reduce the capacity of the model. For example, reducing the allowed maximum depth of a Decision Tree model is a good way to reduce its capacity.

However, not all failures of machine learning models can be ascribed to the quality of the model. Failures can also stem from the quality of the data. Common cases include:

Biased Data: Biased data might occur if the acquired data stems from a sub-population that insufficiently represents the entire population. For example, if we train our Diabetes model from above only with obese people the resulting model will most likely have a greater tendency to predict Diabetes. Note, that imputing missing values might also result in a biased data set.

Leaking Labels: If a feature of the data is highly correlated to the outcome, a model might only learn this connection, preventing the model from generalizing properly. For example, including an additional feature “Takes Insulin” to the Diabetes model would leak the label, as only people that are already diagnosed with Type 2 Diabetes take the drug Insulin.

1.2 Visual Analytics

Gaining meaningful insights about large data sets is hard. While computing various statistics about the data is often helpful, it might oversimplify or in some cases even mislead information about the data. One example data set often used to demonstrate this is Anscombe’s quartet [11]. The quartet consists of four different data sets that all have the same mean and sample variance for both axes, the same correlation between the axes, and the same regression lines. However, plotting the values of the data sets reveals that each one of those data sets obeys a different, unique characteristic (see Figure 1.5).

Visual Analytics is the area of studying interactive graphical respresentations for analyzing complex data such as Anscombe’s quartet mentioned above. This is often done in addition to and with the help of computational and statistical procedures. Various studies (including but not limited to [26, 30, 118]) have been performed to establish visual principles to effectively take advantage of the high processing power of the human visual system. Additionally, interaction with the graphical representations allows to hide some of the information and reveal it through interaction to not overwhelm the analyst. For example, “overview first; details on demand” [105] is a popular mantra taking advantage of interaction.

In the context of machine learning, visual analytics, at first, seems to be a

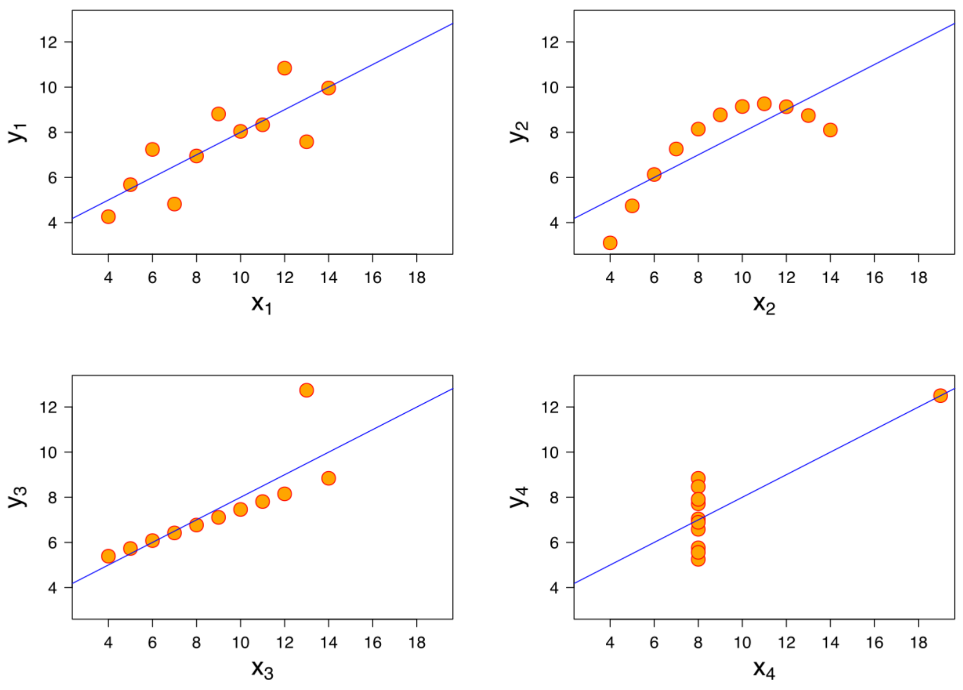


Figure 1.5: Plots of the four data sets of Anscombe's quartet[7]. All data sets have the same mean and sample variance for both axes, the same correlation between the axes, and the same regression lines.

roadblock. Integrating a human in a machine learning workflow inevitable leads to said human being the bottleneck. Humans are magnitudes slower than computers which have to idle while waiting for human input. Thus, human input should only be required for tasks that are impossible without.

For example, with the goal of improving the accuracy of a model a human should not have the task of, *e.g.*, manually picking the order of the features to be included in a decision tree. A good intuition might outperform the computer momentarily but having an objective goal makes it possible to eventually find a suitable algorithm that is en par or even outperforms the human. Additionally, manual decisions are not generalizable to other tasks and not scalable to larger

[7]Image source: Wikipedia

data sets which in turn is the reason to use machine learning to begin with.

However, humans have *soft-knowledge*. That is, they have a general understanding of the world that is typically not codified for machines in all details. Machines can only work with the data they have. Their data is their *universe* and to them nothing else exists. Thus, only humans have the potential to detect and correct when machine learning models acquire incorrect assumptions about their environment. Visual Analytics can help debugging models, as well as, its data.

Additionally, Visual Analytics can help humans, especially domain experts with a problem that is to be solved using machine learning, to verify the *semantic accuracy* of a model. That is, whether decisions made by the model "make sense". Often, the root cause of semantically inaccurate models is due to biased data. Thus, statistical accuracy (*i.e.*, the proportion of correctly predicted instances) of a model can be higher than of a different model, while simultaneously being semantically less accurate (see [25, 65]).

Finally, as humans use machine learning models to speed up and improve decision making, human users need to be able to trust the decisions made by the model. Visual Analytics can help with *understanding* the decision's made by machine learning models and therefore improve the *trust* in the correctness of those decisions.

1.3 Use Case: Machine Learning for Health Care

In this thesis we will mostly explore use cases from the medical domain[8]. It is quite interesting to see parallels between the decision making process of doctors

[8]However, results are easily transferable to other domains as well.

and machine learning.

Over thousands of years the health care domain has evolved into what we now know as evidence-based medicine. That is, new drugs can only be approved after long studies that show significant improvements and methods have been developed and refined over time to improve their measurable effectiveness.

For example, when a patient is examined by a doctor, the doctor follows a structured approach similar to a decision tree. The first few nodes in the tree set up the context: age, ethnicity, weight, visual appearance, *etc.*. A young patient has a vastly different set of potential health risks than an old patient. Likewise, a physically active patient has different health risks than an obese patient. Only after those initial features come symptoms into play. This pattern is also reflected in the documentation of a patient encounter, the doctor's note. Doctor's notes are structured in a way that reinforces this decision tree approach by putting the context establishing features first.

When it comes to symptoms, doctors often have to weigh different plausible diagnoses against each other, as the same symptoms can appear from different causes. This is called *differential diagnosis*. In order to determine which diagnosis has the highest likelihood, Bayesian reasoning is applied [125]. That is, the conditional probabilities of symptoms with respect to different diagnoses are compared to each other. The probabilities are determined through historical records. However, the resulting rules are often simplified to be memorizable by humans.

While residents (recent medical doctorate graduates) follow those rules religiously, attending physicians (more senior medical doctors) might sometimes follow their intuition, acquired through years of patient interactions. Thus, such a medical decision might not be fully explainable through empirical statistics alone anymore.

This is equivalent to, for example, a deep neural network that learned some transformations through its hidden layers that led to hard to interpret interactions of the initial input features.

1.4 Motivation

Modern machine learning algorithms have reached a degree of potential that lets the resulting models model almost any data set. But even though there are statistical measures to guarantee that the model is learning correctly without over-fitting on the training data, the model with the highest accuracy is not always the most usable model or even semantically correct. The following three examples aim to showcase this:

(1) Caruana *et al.* [25] built a Generalized Additive Model with pairwise interaction (GA^2M) to predict mortality risk for hospitalized patients with pneumonia, a potentially life-threatening lung infection. This predictive modeling task used machine learning to predict the mortality risk based on values of several input features, *e.g.*, laboratory measurements, comorbidities, or patient demographics, and was trained from and tested against many instances. It was not feasible to individually check each prediction of the model manually. However, the machine learning algorithm GA^2M used for this study is an intelligible model, that sacrifices parts of its predictive quality in favor of being understandable by humans. This enabled the authors to inspect how features contributed to the predicted outcome.

One of their findings was that the model associated having asthma, a chronic lung disease, with a lower mortality rate for pneumonia patients. However, this combination of diseases is known to have a significantly *increased* mortality rate.

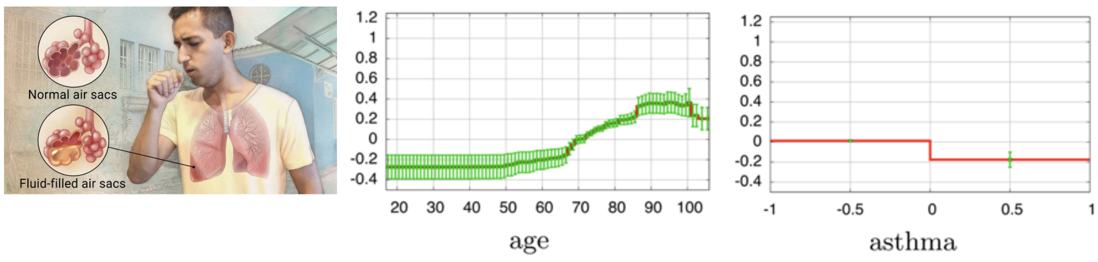


Figure 1.6: Pneumonia is an infectious disease of the lungs that inflames air sacs and likely fills them with fluid, making it hard to breathe and potentially life-threatening (left[9]). The influence functions of the features "age" (middle) and "asthma" (right) of the model created by Caruana *et al.* [25].

The authors double checked their data and found that this phenomenon actually occurred in their data set and thus appeared as correct predictions in their model. After further investigation, it became clear that this bias in the data set stemmed from patients, with those conditions, being treated with intensive care thus lowering their mortality risk below the average. Outside of a hospital the risk of those patients would have been significantly higher. The model, however, only learnt about hospitalized patients which led to it being confident in this false association. (2) Layzell and Bird [17] used machine learning to automatically design an oscillator, an electronic circuit that produces a periodic oscillating signal, using a physical programmable circuit rather than a simulated circuit. After the model completed the task, the scientists looked at the resulting electronic circuit and were surprised to find that instead of the conventional design, using a feedback loop of an amplifier and a filter, the model designed a makeshift radio that responded to the desired frequency and amplified radio signals emitted by nearby computers instead. The output signal of this electronic circuit had the correct frequency and amplitude according to the specification but did not create a valid result due to a latent bias in the available data.

[9]Image source: Google

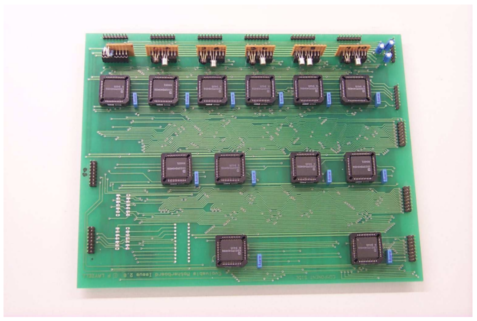


Figure 1.7: The physical programmable circuit used by Bird and Layzell [17] for solving the task of automatically designing oscillators.

(3) Brown *et al.* [22] exploited weaknesses of popular image classification machine learning models in order to create an adversarial patch. This adversarial patch, when added to any image will overwrite the original classification of the model and force it to predict the desired outcome. Concretely, the patch seen in Figure 1.8 forces image classification models to predict any image as toaster. The existence of such patches illustrates the fragility of modern machine learning models to unexpected input data, which leads to surprising and counter-intuitive mispredictions.

Those three examples all exhibit limitations of machine learning models where increasing the statistical accuracy of the model does not result in a semantically more correct model. This is due to inherent biases of the data.

In the pneumonia example (1) patients with exceptionally high mortality risk were removed from the data leading to the model being unable to detect the severity of those cases. In the radio example (2) the model utilized contextual data from the environment that is present but *should* not be used in the model. And in the toaster example (3) the model expected that all input data is coming from physical, real, objects. Those biases cannot be detected from within the model or the data itself. Furthermore, the process of detecting those errors, or even optimizing for semantic accuracy cannot be automated or be formulated as procedure, since there

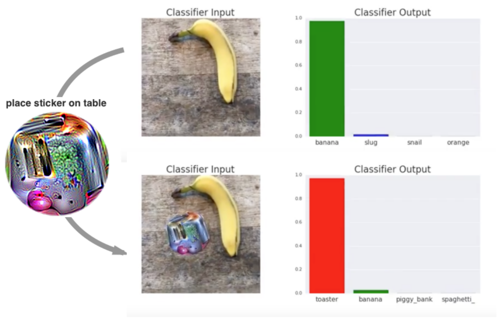


Figure 1.8: The adversarial patch. When put on any image, the image will get classified as toaster [22].

is no objective or measurable optimization criteria for it. This is in contrast to statistical accuracy where a higher number always indicates a better model. As a result, human judgment and contextual knowledge is absolutely necessary.

The above examples illustrate the need for human understanding and supervision in machine learning. However, those cases required intricate understanding of the used machine learning models and are not easily generalized to other techniques. For example, in the case of Caruana *et al.* [25], the incorrect correct result was easily identified since the authors used a special, intelligible, model. However, not all models used in machine learning are, or even attempt to be, easily interpretable by human experts. To counteract this, visual analytics has been successfully used to visualize machine learning processes in order to understand and gain insights into machine learning models [3, 59, 77, 79, 85, 87, 96, 112]. With regards to understanding the decision making of predictive machine learning models, there are two main strategies: *white-box* and *black-box* analysis [67].

In white-box analysis the main focus lies in communicating the internal structure of the model at hand. That is, the analysis is based on the assumption that knowing the full internal state of a model and being able to manually retrace decisions made by it, helps understanding the *behavior* of the model in general.

	White-box		**Black-box**
$\oplus$	Deep understanding of decisions	$\ominus$	External behavior only
$\oplus$	Always reflects the true decision making process	$\ominus$	Not guaranteed to reflect model decisions truthfully in all cases
$\oplus$	Full internal state available	$\ominus$	Limited knowledge of model specifics
$\ominus$	Model specific techniques	$\oplus$	Reusable techniques
$\ominus$	Complexity / interpretability data and model dependent	$\oplus$	Complexity / interpretability independent of data and model
$\ominus$	Switching to interpretable model incurs performance penalty	$\oplus$	No performance considerations necessary

Table 1.1: Comparison of black-box and white-box explanation strategies. $\oplus$ indicates advantages of a technique while $\ominus$ indicates drawbacks.

In this work we will focus on black-box analysis whose focus is on communicating external behavior of a model. That is, no information about the model itself is utilized during the analysis but rather typically inferred from observing how the model reacts to carefully crafted inputs. Compared to white-box analysis there are several advantages.

As white-box analysis relies on the internal structure and thus the nature of a particular model type, such as the GA^2M mentioned above, a method developed for one type might not be applicable to a different type. This limits the usefulness of white-box analysis as every new algorithm demands often an entirely new form of representation. Since black-box analysis is model independent findings are universally applicable.

Another drawback of white-box analysis is its scalability to the complexity of the model. Many solutions do not sufficiently scale as models become more complex. For example, a decision tree with 5 nodes, often used as example when explaining the technique, can easily be visualized and understood in a node-link representation. This representation fails to help understanding a decision tree with hundreds or thousands of nodes. For black-box analysis the internal complexity of a model is irrelevant.

On the other hand black-box analysis also has drawbacks. Especially the limited knowledge of the underlying model allows only for an approximate understanding of its behavior as testing the entire input space would be intractable.

1.5 Research Overview and Thesis Contributions

The main goal of this thesis is to explore how model agnostic feature contribution methods can help to gain a holistic understanding of, as well as, insights about predictive models using visual analytics. In three stages, we will show the progression from initial observations, that led us to utilizing black-box analysis, to the eventual use of aggregated instance-level explanations to understand, trust, and verify predictive models. Additionally, we show that our developed visual analytics techniques can be helpful for feature engineering.

Feature engineering is the task of deciding which features will be collected for a particular machine learning problem and how those features are pre-processed or transformed in order to create favorable model results. It requires both expertise of the problem domain, as well as, experience in picking and manipulating features in “the right way”. For the most part, feature engineering cannot be automated and is typically seen as “black-art” as it relies on intuition and creativity of the modeler [36].

1.5.1 INFUSE

In the first stage, we analyze and compare different feature selection strategies. Feature selection is a pre-processing step that, unlike feature engineering, algorithmically determines the possible predictive impact of a feature. Only the most impactful features are then used as input for the predictive modeling process. This is often necessary as a larger number of features require significantly more training examples to accurately represent the valid input space without losing predictive qualities. As feature selection algorithms use similar statistical tools as many predictive modeling techniques it seems plausible to infer the importance of features in the context of decision making through feature selection in a model agnostic way.

Being able to visually compare different feature selection strategies the machine learning experts noticed that, depending on the chosen feature selection algorithm, different, mostly distinctive, feature sets were deemed to be important while, at the same time, having no significant impact on predictive performance. Additionally, from the view of a domain expert those feature sets were equally reasonable in the context of the prediction task. This demonstrates that **inspecting** and **comparing alternate settings** lets machine learning experts develop insights that **overwrite** their **initial intuitions**. Also, concluding that rankings from feature selection algorithms are not informative enough to provide the importance of features in the context of understanding model decisions, a more powerful model agnostic approach is needed.

1.5.2 Prospector

In the second stage, we explore the influence of features in the decision making process of predictive models through a technique called *partial dependence*. Partial dependence computes feature impact on model outcomes by systematically probing the model with artificial inputs. That is, while keeping the rest of the input values the same, the inspected feature assumes all possible values showing the relationship of feature values to the prediction score. Aggregated over all observed instances, the general behavior of the model with respect to a given feature can be inferred. By computing a novel feature importance score from partial dependence relationships unusual and interesting relations can be found quickly without having to inspect all, often several thousand, features.

Analyzing partial dependency relationships on diabetes prediction models trained on electronic medical records containing patient demographics, diagnoses, medications, and laboratory results, using our method we gained several insights. Firstly, we showed that logistic regression models are not expressive enough to accurately model the complex relationships of certain features to the outcome. Secondly, imputation of missing values, using the population average in laboratory results, caused the model to become uncertain close to the normal value range of laboratory tests. Lastly, we found that the model used a proxy variable indicating the number of doctor visits as predictor of the healthiness of a patient. However, this is *not* a valid predictor. Even though a patient is likely to visit a doctor more often if she is sick, the reverse cannot be said. The above findings show how **partial dependence** with the proposed **feature importance score** allow analysts to effectively **detect model errors** related to **over-** and **under-fitting**, **imputation**, and **leaking labels** caused by incorrect cause-effect relationships. However, a

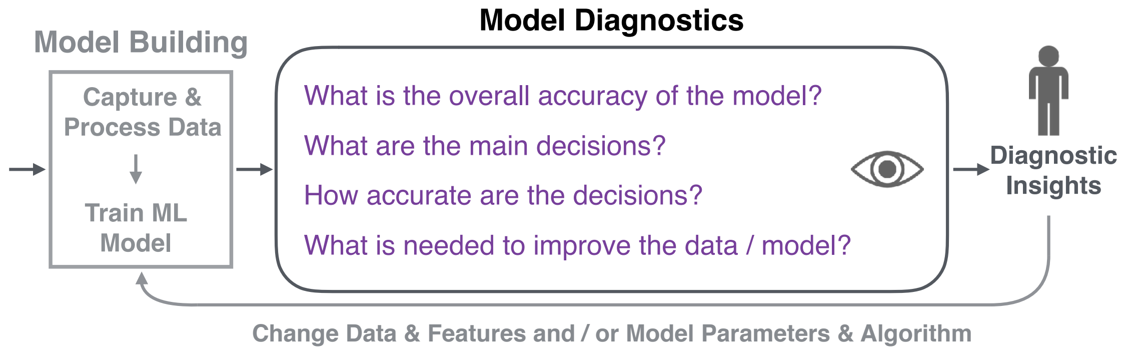

Model Building
Capture & Process Data
Train ML Model
Model Diagnostics
What is the overall accuracy of the model?
What are the main decisions?
How accurate are the decisions?
What is needed to improve the data / model?
Diagnostic Insights
Change Data & Features and / or Model Parameters & Algorithm

Figure 1.9: The proposed *Model Diagnostic* workflow extends the conventional *Model Building* workflow in machine learning for enabling domain experts to reason about the semantic validity of the decisions made by any model through multiple linked visualizations. This ultimately helps to improve data acquisition and model generation processes belonging to the original workflow.

major drawback of using partial dependency is its limitations on extending it to relations of multiple features to the predicted outcome.

1.5.3 Model Diagnostic Workflow

In the third stage, we overcome those limitations by leveraging *instance-level explanations*. Instance-level explanations are defined as the smallest change, to an instance vector, necessary to change the predicted outcome label. Aggregating over those explanations and statistically analyzing the resulting instance subsets, proves as powerful novel approach for understanding the behavior of a model. Additionally, we propose a Model Diagnostic workflow (see Figure 1.9) that helps identify flaws in the *input data*, used to train and test the model.

We use this workflow to analyze a predictive modeling problem revolving around patient admission to a hospital, of patients in the emergency department of said hospital. Accurately predicting whether a patient eventually gets admitted to the hospital helps reducing costs. The major limiting factor of this prediction task

Approach	**Instances**	**Aggregated**	**Global**
INFUSE			X
Prospector	X		X
Model Diagnostic workflow		X	

Table 1.2: Locality of decision analyses of the presented approaches. **Instances** refers to analyzing decisions for one instance at a time. **Aggregated** refers to analyzing decisions for groups of instances and **Global** refers to analyzing decisions globally without inspecting decisions for individual instances.

is the need for input features readily, and electronically, available at the earliest point possible. To this extend we initially used features of prescribed medications as this information is immediately available electronically. Our analysis showed that there are clear groups of patients where the predictive model is very helpful. However, there are some groups of patients where it is impossible for the predictive model to make accurate predictions given the provided information. One of those groups is patients receiving Diatrizoate Meglumine, a contrast medium for CAT or PET scans. This medication only indicates that a scan was performed, but does not carry information about the result of the scan. However, the result of the scan is the deciding factor whether a patient needs further care or can get sent home. Thus, with the given information it is impossible for the predictive model to make a decision that performs better than random guessing. Including more information in the form of additional features does indeed help with this problem but pushes the time when a decision can be made further back. One possible solution is to utilize the predictive model only for the confident groups of patients and wait for the doctors' decisions in other cases. This example illustrates that it is not only possible to **understand decision making** of a predictive model through the **Model Diagnostic workflow** based on **aggregated instance-level**

explanations, but also how it can be applied for **semantic validation** and **feature engineering** on the input data.

1.5.4 User Study on Aggregating Explanations

In order to show the generalizability of the Model Diagnostic Workflow we conducted a user study to explore the effectiveness of aggregated instance-level explanations in detecting biases in the input data.

First, we generalized aggregating instance-level explanations to numerical input data. We achieved this by using histograms of the input features, similar to [64], ordered by the features' aggregated importance. Furthermore, we allowed the comparison of selected, meaningful, subsets to each other. Then, we compared this aggregated representation, in a user study, to the commonly used approach of individually inspecting instance-level explanations one-by-one using a tabular representation.

We found that aggregating instance-level explanations, with our method, significantly **outperforms** inspecting **individual explanations** in its ability to enable detection of biases in the data. However, our method **requires explanations** in order to perform correctly. Additionally, we found that instance-level explanations **hurt** bias detection performance for a **tabular representation**. We hypothesize that the added difficulty of inspecting a table without the help of explanations forces users to **interact** more strongly with the user interface, thus providing better results. This is a known effect (see Hullman *et al.* [54]).

All in all, aggregating instance-level explanations perform as well as explanation-free tabular representations, which require high user engagement. Additionally, our method promises a higher scalability due to its independence from the total number of instances.

Furthermore, we could reproduce findings from Stumpf *et al.* [113], claiming that instance-level explanations might make users over-confident in their understanding of a model. However, we also showed that aggregated instance-level explanations overcome this problem.

1.6 Thesis Outline

We organize the remainder of this thesis in the following manner. In Chapter 2 we will describe and discuss the *INFUSE* system for comparing different feature selection strategies. In Chapter 3 we will describe and discuss the *Prospector* system for inspecting models via partial dependence plots. Following, we will introduce the *Model Diagnostic workflow*, aggregating instance-level explanations, in Chapter 4 and discuss its capabilities in the context of binary input data. Afterwards, in Chapter 5 we will extend this workflow to numerical input data and explore its effectiveness compared to individually inspected instance-level explanations. In Chapter 6, we present a summary of the work while discussing key findings, implications, and open issues, leading to further research opportunities for future researchers. Finally, in Chapter 7, we discuss the conclusions and future work.

Chapter 2

INFUSE

Predictive modeling techniques are increasingly being used by data scientists to understand the probability of predicted outcomes. However, for data that is high-dimensional, a critical step in predictive modeling is determining which features should be included in the models. Feature selection algorithms are often used to remove non-informative features from models. However, there are many different classes of feature selection algorithms. Deciding which one to use is problematic as the algorithmic output is often not amenable to user interpretation. This limits the ability for users to utilize their domain expertise during the modeling process. To improve on this limitation, we developed INFUSE, *a novel visual analytics system designed to help analysts understand how predictive features are being ranked across feature selection algorithms, cross-validation folds, and classifiers. We demonstrate how our system can lead to important insights in a case study involving clinical researchers predicting patient outcomes from electronic medical records.*

Summary:

Research Question:

How do different feature selection strategies compare to each other?

Key Findings:

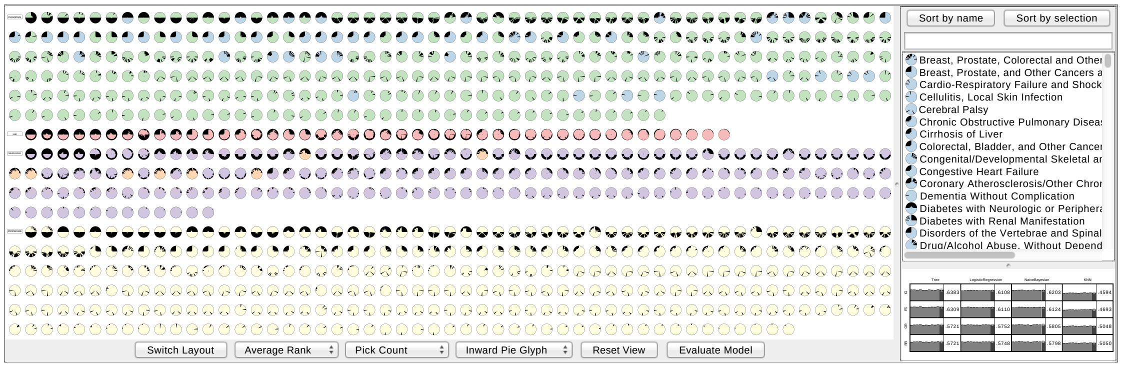
- Different strategies prefer different, equally reasonable, feature sets without having a significant impact on predictive performance.
- Inspecting and comparing alternate settings lets machine learning experts develop insights that overwrite their initial intuitions.
- Rankings from feature selection strategies are not informative enough to help understand model decisions.

Josua Krause, Adam Perer, Enrico Bertini

The visualization research community has usually focused on developing techniques and systems to support the analysis of data sets, with limited analysis of the relationship between data sets and the construction of models on top of them. However, there are a growing number of data scientists interested in more than just interpreting their data: they want to understand their data and predictive probabilities associated with them. Providing visual support for this kind of task has become important as many existing applications on the market and in scientific settings need to solve problems that are predictive in nature, *e.g.*, prediction of customer behavior, diseases, drug effectiveness.

Predictive modeling is defined as the process of developing a mathematical tool or model that generates an accurate prediction [70]. However, building an accurate predictive model is far from trivial. First, modelers must construct cohorts, or distinct groups, to divide their data sets into cases and controls. Then, they must use a feature construction technique to define the feature vector. Next, they must define the parameters for cross-validation to ensure the results are statistically valid and robust. Then, they need to choose a feature selection algorithm to extract the informative features and include them in a model. And finally, they need to choose a classifier to evaluate the predictiveness of the model. For each of these decisions, there are a variety of techniques for cohort construction, feature construction, cross-validation, features selection, and classification to choose from, and there are currently no systematic guidelines to decide which algorithms are most appropriate for which types of data sets. Making the wrong choices can cause predictive models to fail. Kuhn and John argue that many predictive models fail because, "predictive modelers often only explore relatively few models when searching for predictive relationships [...] due to either modeler's preference for, or knowledge of, or expertise in, only a few models or the lack of available software that would enable them to explore a wide range of techniques" [70]. We use these current limitations as motivation to research how visual analytics may improve the process of predictive modeling.

Our proposed research focuses on an important step in the predictive modeling pipeline: feature selection. When data is high-dimensional, feature selection algorithms are often used to remove non-informative features from models. Again, the analyst is confronted with the decision of which feature selection algorithm to utilize, and even if the analyst decides to try out multiple types, the algorithmic



Breast, Prostate, Colorectal and Other
Breast, Prostate, and Other Cancers a
Cardio-Respiratory Failure and Shock
Cellulitis, Local Skin Infection
Cerebral Palsy
Chronic Obstructive Pulmonary Diseas
Cirrhosis of Liver
Colorectal, Bladder, and Other Cancer
Congenital/Developmental Skeletal ar
Congestive Heart Failure
Coronary Atherosclerosis/Other Chron
Dementia Without Complication
Diabetes with Neurologic or Periphera
Diabetes with Renal Manifestation
Disorders of the Vertebrae and Spinal
Drug/Alcohol Abuse, Without Depend

	Tree	LogisticRegression	NaiveBayesian	KNN
IG	.6383	.6108	.6203	.4594
FS	.6309	.6110	.6124	.4693
OR	.5721	.5752	.5805	.5048
RR	.5721	.5748	.5798	.5050

Figure 2.1: An overview of *INFUSE*, a visual analytics tool that supports users to understand the predictive power of features in their models. Each feature is ranked by various feature selection algorithms, and the ranking information is visualized in each of the three views within the system. On the left, the Feature View provides a way to visualize an overview of all features according to their rank using a variety of layouts. On the top-right, the List View provides a sorted list of all features, useful for selections. On the bottom-right, the Classifier View provides access to the quality scores of each model. Each of the views are coordinated, and users can brush between all three views.

output is often not amenable to user interpretation. This limits the ability for users to utilize their domain expertise during the modeling process. To improve on this limitation, we developed *INFUSE* (INteractive FeatUre SElection), a novel visual analytics system designed to help analysts understand how predictive features are being ranked across feature selection algorithms, cross-validation folds, and classifiers. We describe the tasks associated to the feature selection and understanding process and provide a design rationale for our solution. We also demonstrate, through case studies, how the system can lead to important insights for clinical researchers predicting patient outcomes from electronic medical records.

Concretely, our contributions include:

- A design and implementation of a predictive modeling exploration system, *INFUSE*, for understanding how predictive features are being ranked across

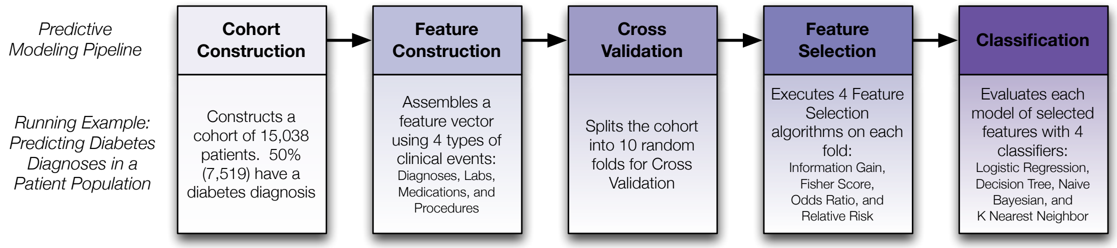

Predictive Modeling Pipeline
Running Example: Predicting Diabetes Diagnoses in a Patient Population
Cohort Construction
Constructs a cohort of 15,038 patients. 50% (7,519) have a diabetes diagnosis
Feature Construction
Assembles a feature vector using 4 types of clinical events: Diagnoses, Labs, Medications, and Procedures
Cross Validation
Splits the cohort into 10 random folds for Cross Validation
Feature Selection
Executes 4 Feature Selection algorithms on each fold: Information Gain, Fisher Score, Odds Ratio, and Relative Risk
Classification
Evaluates each model of selected features with 4 classifiers: Logistic Regression, Decision Tree, Naive Bayesian, and K Nearest Neighbor

Figure 2.2: Steps of a typical predictive modeling pipeline. For each step, we provide the details of the running example we use throughout the paper.

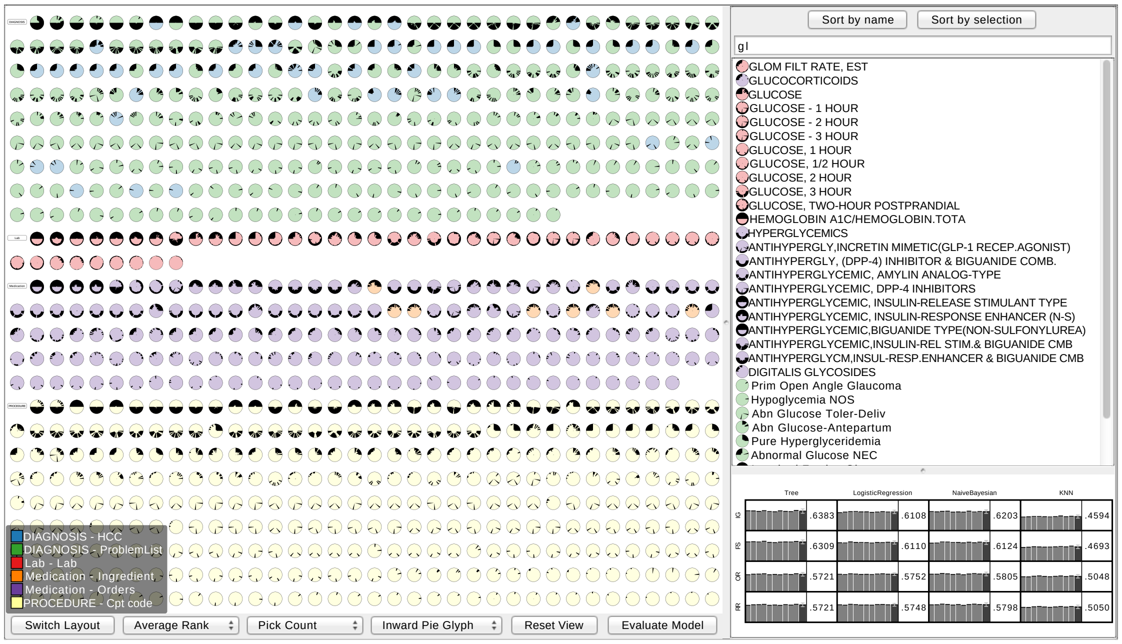
feature selection algorithms, cross-validation folds, and classifiers.

- An Interactive Model Builder, where users can create customized models based on insights reached with *INFUSE*, and then have their results evaluated in comparison to automated methods.
- A case study of domain experts using *INFUSE* to explore predictive models in electronic health records.

2.1 Motivation

2.1.1 Predictive Modeling in Health Care

Predictive modeling is a common and important methodology used in medical informatics and health care research. For instance, it can be used to detect diseases in patients early before they progress [14] and to personalize treatment guidelines to understand which populations will benefit from an intervention [56]. In order to derive such insights and build successful predictive models, it is common for health care researchers to implement, evaluate, and compare many models with different parameters and algorithms. A common workflow for predictive models is a 5-step



GLOM FILT RATE, EST
GLUCOCORTICOIDS
GLUCOSE
GLUCOSE - 1 HOUR
GLUCOSE - 2 HOUR
GLUCOSE - 3 HOUR
GLUCOSE, 1 HOUR
GLUCOSE, 1/2 HOUR
GLUCOSE, 2 HOUR
GLUCOSE, 3 HOUR
GLUCOSE, TWO-HOUR POSTPRANDIAL
HEMOGLOBIN A1C/HEMOGLOBIN.TOTA
HYPERGLYCEMICS
ANTIHYPERGLY,INCRETIN MIMETIC(GLP-1 RECEP.AGONIST)
ANTIHYPERGLY, (DPP-4) INHIBITOR & BIGUANIDE COMB.
ANTIHYPERGLYCEMIC, AMYLIN ANALOG-TYPE
ANTIHYPERGLYCEMIC, DPP-4 INHIBITORS
ANTIHYPERGLYCEMIC, INSULIN-RELEASE STIMULANT TYPE
ANTIHYPERGLYCEMIC, INSULIN-RESPONSE ENHANCER (N-S)
ANTIHYPERGLYCEMIC,BIGUANIDE TYPE(NON-SULFONYLUREA)
ANTIHYPERGLYCEMIC,INSULIN-REL STIM.& BIGUANIDE CMB
ANTIHYPERGLYCM,INSUL-RESP.ENHANCER & BIGUANIDE CMB
DIGITALIS GLYCOSIDES
Prim Open Angle Glaucoma
Hypoglycemia NOS
Abn Glucose Toler-Deliv
Abn Glucose-Antepartum
Pure Hyperglyceridemia
Abnormal Glucose NEC

	Tree	LogisticRegression	NaiveBayesian	KNN
IG	.6383	.6108	.6203	.4594
FS	.6309	.6110	.6124	.4693
OR	.5721	.5752	.5805	.5048
RR	.5721	.5748	.5798	.5050

Figure 2.3: An overview of *INFUSE*, a system for interactive feature selection. On the left, the Feature View provides a way to visualize an overview of all features grouped by type and then sorted by importance. The color key for the feature types and subtypes are shown at the bottom. The buttons and combo boxes at the bottom can be used to switch layouts and define the axes of the scatterplot view shown in Figure 2.6. On the top-right, the List View provides a sorted list of all features, useful for selections. This list can be filtered using the search box above. Currently only features containing the term “gl” are shown. The remaining features are sorted by the number and position of the search term occurrences. On the bottom-right, the Classifier View (Figure 2.7) provides access to the quality scores of each model. Users can also select features and build custom models with the Interactive Model Builder.

process, illustrated in Figure 2.2: (1) cohort construction, (2) feature construction, (3) cross-validation, (4) feature selection, and (5) classification. There are currently few tools that support this complex workflow for predictive modelers.

A recent platform, PARAllel predictive MOdeling (*PARAMO*) [90], enables users to specify a small number of high-level parameters to support this 5-step workflow. *PARAMO* then uses Map-Reduce to execute these many tasks in parallel. After the models have been constructed and evaluated by classifiers, users can compare area under curve (AUC) scores of different models and select the ones with the highest predictive power. While this ability to construct and evaluate models at scale is an important breakthrough for clinical researchers, the clinical experts are still left out of the loop at each of these 5 stages, as each of the algorithms act as a black box.

This type of workflow limits the ability of clinical researchers to use their domain knowledge to assist in the model building phase. While multiple models may have similar performance in terms of prediction accuracy, there is a desire to ensure that models with more clinically meaningful features are selected [28].

2.1.2 Running Example: Diabetes Prediction

In order to make our contributions concrete, we utilize a running example from our case study. Our case study involves a team of four clinical researchers interested in using predictive modeling on a longitudinal database of electronic medical records. The research team consisted of one MD researcher with a background in emergency medicine, and three PhD researchers with backgrounds in health care analytics. Their database features over 300,000 patients from a major health care provider 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 causes serious health complications.

From this database, the team constructs a cohort (Step 1) of 15,038 patients. 50% of these patients (7,519) are considered incident cases with a diagnosis of diabetes. Each case was paired with a control patient based on age, gender, and primary care physician resulting in 7,519 control patients without diabetes. From the medical records of these patients, they extract four meaningful types of features (Step 2): diagnoses, lab tests, medications, and procedures. In total, there were 1,627,736 diagnosis events (6,709 unique types), 361,026 lab events (193 types), 818,802 medication events (344 types), and 853,539 procedures (4,403 types). For our visualization, we only consider types of features that were picked by feature selection algorithms which results in 859 features to display.

Next, in order to reduce the bias of the predictive models, the team uses 10 cross-validation folds (i.e. random samples) (Step 3) to divide the population randomly into 10 groups. After cohorts, features, and folds are defined, the clinical researchers are ready to use feature selection. The team has four feature selection algorithms implemented and available to them (Step 4): these include *Information Gain* and *Fisher Score*, which have been used extensively by the researchers, as well as two new ones which were recently implemented by their technologists: *Odds Ratio* and *Relative Risk*. Finally, the team evaluates each selected feature set as a model using four classifiers (Step 5): *Logistic Regression*, *Decision Trees*, *Naive Bayes*, and *K-Nearest Neighbors*.

Typically, this team executes a pipeline of multiple feature selection algorithms, and chooses the model that ends up with the best scores from the classifier. Although this team has an interest in embedding domain knowledge into their models, their current platform for running predictive models does not have a user interface where users can view or edit the specific features that make up each model. Therefore, resulting models are typically not interpretable by domain experts, and do not support bringing in their medical expertise by prioritizing or removing features that may not be relevant to the disease they are modeling.

2.1.3 Task Analysis

The data analysis team initially expressed an interest of having a visual analytics system to aid them in making sense of the complex information generated by the modeling pipeline. During our interviews we agreed to focus on the feature selection and classification steps, as they needed visualizations to reason about the effects of choosing different combinations of the available algorithms. Without such visualizations, the researchers ability to choose among different algorithms is ineffective.

Through our interactions with the analysts we derived three main tasks that guided the design of *INFUSE*:

Task1 - Comparison of feature selection algorithms. In data sets with thousands of features, it is important to have a quick way to understand how feature selection algorithms rank different features differently. Some of the typical questions the researchers ask are: "*Which features are consistently ranked highly by all the algorithms?*"; "*How much do the algorithms differ in their ranking?*"; "*Are there features that have a high rank with some algorithms and a low rank with some others?*"; "*How robust are the rankings with respect to different data samples?*"

Task 2 - Comparison of classification algorithms. The output of each feature selection algorithm is used to feed a series of classification algorithms. At the end of this process, the user is left with a $F \times C$ number of performance comparisons, where F is the number of feature selection algorithms and C the number of classification algorithms. Typical questions our researchers ask are: "*Which combinations of feature selection and classification algorithms give the best scores?*"; "*Are there feature selection algorithms that score consistently better across the set of classification algorithms?*"; "*Are there classification algorithms that score consistently better across the set of feature selection algorithms?*"; "*Which sets of features are selected in the model(s) that give the highest performance?*"

Task 3 - Manual selection and testing of new feature sets. Related to the last question of Task 2, the researchers see value in being able to add or remove features of interest from models. This is desired because there can be additional domain-relevant knowledge, beyond model performance, to introduce a desired feature or remove an undesired one. Typical questions our researchers ask are: "*How does the performance of the model increase or decrease if I remove or add these features?*"; "*How does a new model compare to the models automatically built by the system?*"

INFUSE was designed to support these three tasks by providing a visualization of large sets of features and how these features are used by the modeling algorithms. After several design iterations, we converged on a visual design where features are first-class citizens of the visual representation: that is, each visual object in the main view represents a feature and its design and layout reflects information obtained from the algorithms. A representation centered on features aligns well with the analysts' mental model and makes features easily identifiable through their names. Each feature, in fact, represents real-world entities like medications, lab tests and diagnoses, that have rich semantics and can be easily identified and understood by domain experts.

2.2 Related Work

While visualization of multidimensional data has traditionally focused more on the visualization of the data space, visualizing data features has important applications in real-world scenarios; especially when confronted with hundreds or even thousands of dimensions. In this context, visualization helps data analyst making sense of the feature space while including their background knowledge in the process. Visual feature selection can, for instance, help rank features according to predefined scores, detect similarities among dimensions (thus gauging intrinsic dimensionality of feature spaces), merge or combine features into composite features. In the following we review visualization literature that consider the specific problem of visualizing large sets of features.

2.2.1 Visual Feature Selection

Several approaches to feature selection and dimensionality reduction, in general, exist in visualization. The early work of Guo [45] introduced the idea of visualizing relationships between features sets. His system is based on an interactive matrix view where rows and columns represent features and the cells are colored according to feature similarity (calculated as entropy and χ^2). The matrix is automatically sorted to allow selection of subspaces (feature subsets) where data shows interesting clusters. Visual hierarchical dimension reduction [126] allows detection and grouping of similar features as well. The technique is based on a hierarchical clustering algorithm which clusters dimensions in terms of their similarity and present them in a *sunburst* visualization [131]. Users can interactively choose an aggregation level and use the aggregated dimensions to display data with the reduced set of dimensions. Johansson and Johansson [57] present an integrated environment based on *parallel coordinates* visualization where the number and order of dimensions (axes) presented at any time is guided by a ranking algorithm that takes into account associations as well as intrinsic interestingness of each feature to interactively choose how many features to visualize. Similar in spirit is the *rank-by-feature* framework [104] in which the data features are organized, ranked and visualized in 1D and 2D visual representations (e.g. histograms, bar charts and scatterplots). The user can for instance inspect a matrix of feature pairs, ranked by one of the available ranking functions, and single out those that show interesting associations. A similar mechanism is also used in *scagnostics* [128] a quality metric approach [16] that ranks axis pairs according to the pattern / shape they create in a scatterplot visualization.

More similar to the solution presented in this paper are visualizations that focus on plotting dimensions as data points in the visual representation (rather than, for example, as axes of a visualization where the data items represent records of a data table). *Value and Relation Display* visualizes data features as icons in a scatter plot visualization [130]. The icons are positioned using a *multidimensional scaling* algorithm which positions dimensions with similar distributions close together. The icons are designed to represent the distribution of the data values within the feature.

Such a display allows to detect groups of similar dimensions and to construct multidimensional visualizations by subsetting the original feature space. *Brushing Dimensions* [120] is a similar approach where data features are plotted as dots in a scatter plot using descriptive statistics as axes (e.g. variance, median, kurtosis). The plot is paired with a data item scatter plot which allows for data and feature linking and exploration.

All of the methods described above are based on the calculation of statistical parameters from the data as a way to characterize and expose relationships between the features. Our approach differs in that *INFUSE* interacts directly with feature selection and classification algorithms to help in the discovery of predictive feature sets. A similar approach is found in *SmartStripes* [84], a visual analytics system that allows tight interaction between feature selection algorithms and visualization. Our system differs in that our focus is on the comparison of the output of multiple feature selection algorithms rather than a single one.

2.2.2 Visualization in Predictive Modeling

Visualization has also been used to aid in the creation of predictive models, not only in the selection of features that might be helpful in constructing such models. Visual construction and assessment of decision tree models have been the subject of a good number of works in the field. Ankerst *et al.*, introduced the idea of using pixel-based visualization as a way to manually construct decision trees by giving the user the ability to observe class distributions within each node and to interactively select splitting points [9, 10]. A similar idea is proposed in *PaintingClass* a visualization technique to manually build a decision tree through interaction of parallel coordinates and multidimensional scaling techniques to identify coherent groups of multidimensional data [117]. More recently, *BaobabView* has been presented as a system to inspect and validate a classification model through a tree representation. The paper presents a thorough analysis of the number of tasks that visualization can support in this area and how they are covered by the proposed system [122].

While all the aforementioned systems focus largely on decision trees, visualization has been used in other classification and regression systems that leverage other prediction models. The *iVisClassifier* [29] for instance uses *linear discriminant analysis (LDA)*, a supervised dimensionality reduction method, to project multidimensional data in a scatterplot visualization taking into account information provided by the data labels. The technique allows to visually link the high-dimensional structure to the low-dimensional representation and build clusters. The clusters are then used to classify new data that is progressively introduced into the system to refine the model. Steed *et al.*, in their *cyclone trend analysis* provide a parallel coordinates visualization that leverage computational analysis to identify features with high predictive power in stepwise regression tasks and allows to build predictive models for multidimensional climate data [109, 110]. Recently, a visual analytics system for regression analysis has been proposed by Mühlbacher and Piringer [88]. The system is more similar to our work in nature as it also focuses on the predictive power of feature sets and guides the user in the predictive modeling process. The main difference between this work and ours is our focus on classification rather than regression models and the use of multiple feature selection and classification models to better understand how features score across multiple models.

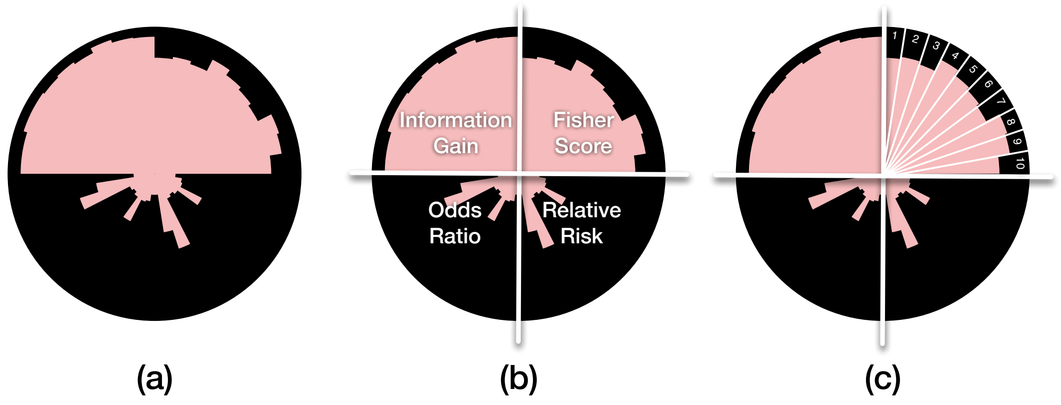


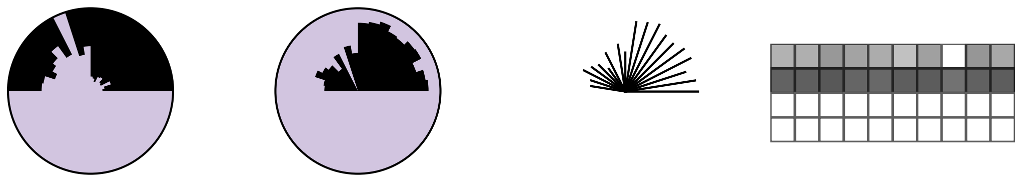
Figure 2.4: (a) The glyph representation of a feature in the *INFUSE* system. (b) Multiple models for each feature are represented as *model sections*. In this example, the feature is divided into four sections, as it was ranked by four feature selection algorithms (Information Gain, Fisher-Score, Relative Risk, and Odds Ratio.). (c) Each section is further divided into *fold slices* representing each of the cross-validation folds. Each fold slices features a inward-filling bar that represents the rank of this feature in that fold. A longer bar implies the feature has a better rank. If no bar appears, the feature was unranked in the fold, and thus did not meet the importance threshold.

2.3 INFUSE

In this section, we describe the design of *INFUSE*, which aims to assist predictive modelers with the tasks introduced in Section 2.1.3. By providing visualizations for users to interpret the results of feature selection algorithms, as well as the ability to customize the models with domain knowledge that may have been missed by the automated algorithms, *INFUSE* provides a user-centric way of manipulating predictive models.

2.3.1 Data and Design

We provide a brief overview of data types utilized by the system. A predictive model, in our setting, is a model trained and validated with machine learning using



(a) (b) (c) (d)

Figure 2.5: Different glyph designs. a shows *fold slices* with bars growing from perimeter to center whereas b grows from center to perimeter. c shows a typical starburst glyph and d shows a matrix using luminance to show the ranks. Note that in b and c it is difficult to see that this feature is unranked in the third fold from the right in the top left quadrant. The values in c are difficult to read because there is no reference to how big the values are. Luminance, as used in d is a harder perceptual attribute for users to interpret and distinguish than length and area are, as used by the other glyphs.

a high number of features as an input to train the model. These features are the primary data items of *INFUSE*. Each feature has a label representing the feature name (*e.g.*, Diabetes), a category to which the feature belongs to (e.g. Diagnosis), and a subtype (*e.g.*, Problem List, the health problems that led to the diagnosis).

Feature selection algorithms receive as an input the whole set of existing features and return a subset of features selected and ranked according to their estimated predictive power. Since in our setting we use the output of multiple algorithms at once, each feature can further be described by the rankings they receive from all these algorithms (where features that are not selected are marked as unranked). Furthermore, since cross-validation is used, each feature actually gets ranked multiple times by each algorithm, leading to a total number of $\#\textit{feature selection algorithms} \times \#\textit{folds}$ ranks that quantitatively describe each feature.

The predictive models built using the output generated by feature selection also provide useful information that we use in our system. Each feature set generated by the process described above is used as an input to a classification algorithm. The algorithm builds a model that corresponds to the specific pair of feature set and classification algorithm used for its training. The classifier, in turn, can be described in terms of its performance using the Area Under Curve (AUC), a measure that is commonly used by modelers to give numerical performance scores to models [70].

The primary goal of *INFUSE* is to visualize this information so that users can understand the predictive power of features in their models. The user interfaces is organized around three main coordinated views as shown in Figure 2.3: the *Feature View* provides a way to visualize an overview of all features providing information about their attributes and ranking received from feature selection; the *List View* provides a sorted list of all features to get easy access to their labels and to assist the user in searching features according to some predefined criteria like their name or category; the *Classifier View* provides access to the quality scores of each model built using the process described above. The views are coordinated so that selections in one view are propagated to all the other views. In the following we provide additional information about the design of each view.

2.3.2 Feature View

The primary component of *INFUSE* is the *Feature View*, a zoomable visualization that displays all features as glyphs. Each glyph represents a feature from the original data set and is designed to provide the information outlined above. The main purpose of the feature view is to allow comparison between features and detection of interesting commonalities and differences in terms of how the algorithms rank them. The view allows the user to display the feature set according to two different configurable layouts: a grid layout (the default), which favors legibility, and a scatter plot layout which aims at laying out and grouping the

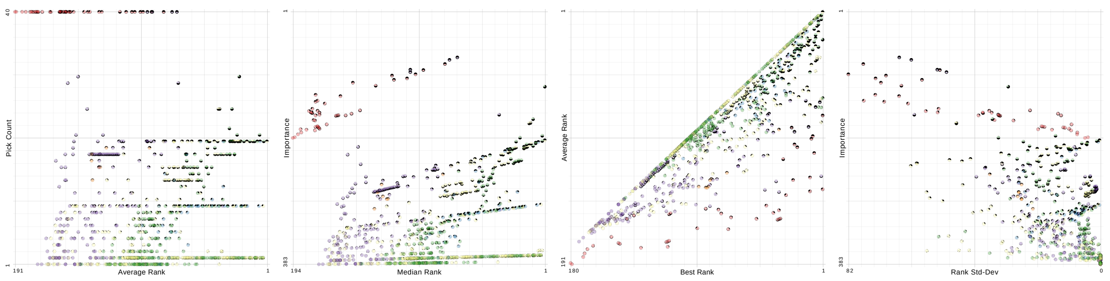


Figure 2.6: Different axis combinations for the scatter-plot layout. In the average rank is plotted against the pick count. Most of the features appear in the lower half because features are rarely picked by more than two algorithms in this example. The bottom-right shows features that are only chosen by two models but were ranked very high by them. shows the median rank plotted against the importance. Notice that the plot looks similar to since importance is a combination of the axes from . The axes in are best rank versus average rank. Features can only appear below the diagonal. The standard deviation of the ranks is plotted against the importance in . The peak to the bottom right corner consists of features that are rarely picked and therefore have lower variance. The peak to the top right consists of features that are consistently high ranked.

features according to various statistics we collect from the ranks. In the following sections, we describe the design of the glyph as well as the different layouts.

2.3.2.1 Feature Glyph Design

As described in Section 2.1.1, the features are ranked by multiple feature selection algorithms and across multiple cross-validation folds. *INFUSE*'s glyph design embeds all of this information in a circular glyph that shows all the rankings obtained from each algorithm/fold pair. As shown in Figure 2.4(a), the glyph is divided into equally-sized circular segments; where each segment represents one of the ranking algorithms. For instance, in Figure 2.4(b), since the feature was ranked by four feature selection algorithms, the circular glyph is divided into four sections. Each of these sections are then divided further into a *fold slice* for each cross-validation fold. For instance, in Figure 2.4(c), each feature selection algorithm was executed on 10 cross-validation folds, therefore there are 10 fold slices.

Within each fold slice, there is an inward-growing bar (that is, starting from the perimeter and growing towards the center) that represents the rank of the feature in a particular fold. For example, in Figure 2.4(c), the feature is higher ranked in Fold 3 than in Fold 4 as the bar in Fold 3 stretches closer towards the center than in Fold 4. Features that are unranked, because their scores are too low to meet the minimum threshold requirement of the algorithm, are represented as empty slices with no bars. We designed fold slices with inward-growing bars on purpose to help distinguishing between slices with empty values from those with low values. During our design iterations we realized in fact that outward pointing bars would make this distinction too hard to make. Since the information of whether a features is picked up by an algorithm is crucial for its interpretation we decided to opt for this design.

Multiple glyph designs were considered and tested within *INFUSE*. For instance, Figure 2.5b shows an example of a glyph where the fold slices grow from the center towards the perimeter. This makes it difficult to identify fold slices with poor ranks. Consider the situation where there is a lowly-ranked feature only ranked in one fold slice section. When zoomed-out, the glyph would just appear as a circle with a dot in the center, and the user would not know which model or fold ranked the feature. Furthermore, it is difficult to see in which fold a feature is unranked when the surrounding models rank the feature. Other glyph designs that were tried involve a star glyph (Figure 2.5c) and a matrix glyph (Figure 2.5d). The star glyph was less effective as users were not afforded a reference point for the maximum ranking and the design leads to some visual artifacts (like high density in the center and lower density in the outer part). The matrix glyph was less effective, as perceiving differences is more difficult when using luminance than length and area as we do in our final design.

Users can gain more details about each section and slice by hovering over the region of interest to view an informative tooltip. Furthermore, an overview key is available to remind users of the position of each model type. The background color of the glyph corresponds to the subtype of the feature and a color key can also be shown as a reference to remember the meaning of the color coding (see bottom-left corner of Figure 2.3).

2.3.2.2 Ranked Layout

The first layout available to users is the ranked layout, which arranges glyphs by their feature type, and sorts them by their overall importance. The name of the feature type is shown at the first position in the group, after that the features are laid out row-first in a grid-like manner, as shown in Figure 2.3. This space-filling approach results in features that are always visible without overlaps.

Features within a group are sorted by their importance. Importance is computed as average rank with penalized unranked features:

$$rank_{best} = \min_{m \in M \times V, f \in F} [rank_m(f)]$$

$$i(f) = \frac{1}{|M \times V|}[2 \cdot rank_{best} \cdot unranked_{M \times V}(f) + \sum_{m \in M \times V} rank_m(f)]$$

where M is the set of models, V is the set of cross-validation folds, F is the set of features, $rank_m(f)$ is the rank of a feature f in the combined model and cross-validation fold m, and $unranked_{M \times V}(f)$ is the number of such combined models that did not choose f. Assume $rank_m(f) = 0$ for unranked features f in the combined model m only when computing $i(f)$. Note that a small value for $i(f)$ means higher importance. The optimal value is 1.

2.3.2.3 Scatterplot Layout

The second layout available to users is the scatterplot layout, where users can select choices for both axes of the scatterplot. The choices for axes include:

- the *average rank* of a feature
  (ignores unranked folds and models)
- the *pick count* of the number of combined models
  and cross-validation folds that picked the feature
- the *importance* of a feature (defined above)
- the *best rank* of the feature
- the *median rank* of the feature
  (ignores unranked folds and models)
- the *standard deviation* of the feature's ranks
  (ignores unranked folds and models)

By default, the average rank is chosen for the horizontal axis and the pick count is chosen for the vertical axis, as shown in Figure 2.10. This combination of axes led to the most insights during the case studies. However, if users choose to select different axes or pivot to a different layout, animation is used for the transition. By using slow-in and slow-out animation, users are given time to anticipate the movement direction of the feature, and are able to track features during the transition easily.

2.3.2.4 Interaction

The Feature View provides a number of interactions. Zooming and panning enables a user to get an overview of the displayed data and focus on the details of a small number of glyphs. This exploration can be reset by clicking on the "Reset View" button, or double clicking on the background. Double clicking on a feature glyph zooms in on the feature so that it fills the viewport. In addition, a tooltip is shown when a user hovers the mouse over a glyph. This tooltip provides information about the name, type, and subtype of the represented feature, as well as all of the statistical information used for the scatterplot layout. Hovering over a fold slice in the glyph gives further information about the feature selection algorithm, the cross-validation fold, and the feature's rank in question. In order to select features for interactive model building (see Section 2.3.5) the user can click on glyphs to toggle the selection or use a lasso gesture to select a group of features. As mentioned in the previous section, users can change the layout of the glyphs with the buttons below the Feature View.

2.3.3 List View

A simple yet important view of features is the List View, which provides a sorted list of all features, useful for selecting features by name. Each list item contains the name of the features along with its glyph. The selection of a feature can be toggled in the list by clicking its list item. As the selection of features is linked between views, this sorted and labeled view supports users finding particular

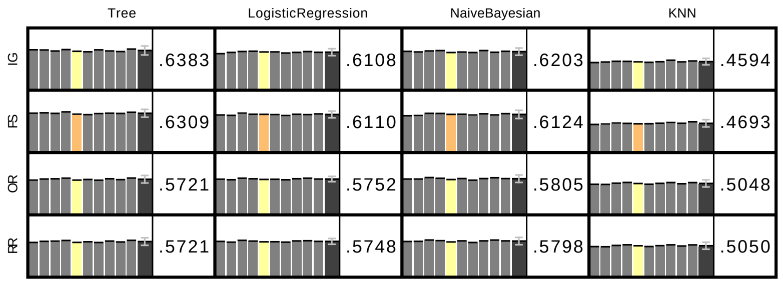


Figure 2.7: The Classifier View displays the results of the classification algorithms for all models. Rows represent feature selection algorithms and columns represent classification algorithms. A more detailed description of the cells can be seen in Figure 2.8. The currently selected model is highlighted in orange, and the results for the same fold in different feature selection algorithms are highlighted in yellow. When users select a model, the features that make up the model are highlighted in the Feature and List views.

features of interest and highlighting them in the complementary views.

The list view can be sorted in a few different ways. By default, features are first sorted by the type of the feature, then by its subtype, and finally by its name. Users can also sort the list by selection, which means currently selected features are displayed at the top and the unselected features appear after them. Within these groups, the features are then sorted by their importance.

In addition to sorting, a user can filter the list view via the search box on the top. Search terms are separated by white-spaces and the list view shows all features that contain all search terms in the name, type, or sub-type. The results are ranked by the sum of the inverse positions of the search terms within the feature description. This favors terms occurring at the beginning of the feature's name and terms that occur multiple times in one feature description (see the top right panel of Figure 2.3 for an example query).

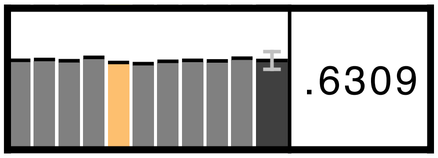


Figure 2.8: Each cell in the Classifier View represents the scores of a particular model by a particular classifier. On the left, there is a bar for each of the validation folds. The height of each bar corresponds to the AUC score for each fold. Immediately to the right of the fold bars, the thicker and darker bar and its height represents the average value across all folds. This bar also features an error bar depicting the standard deviation across the folds. Finally, to the right of the bars, there is a numerical representation of the average AUC score.

2.3.4 Classifier View

The Feature View and the List View both focus on supporting users to interpret the rankings of features across multiple predictive models. However, it is also important for users to understand the quality of each model in predicting the appropriate outcome. The Classifier View, shown in the bottom-right panel of *INFUSE*, is where the quality of each the predictive models can be analyzed.

Typically, predictive models are evaluated using classification algorithms which provide an AUC score (area under ROC curve, the sensitivity as function of the false positive rate). Perfect models will have an AUC score of 1, whereas random guessing will have an AUC score of 0.5. The Classifier View was designed to show AUC scores for each model and fold.

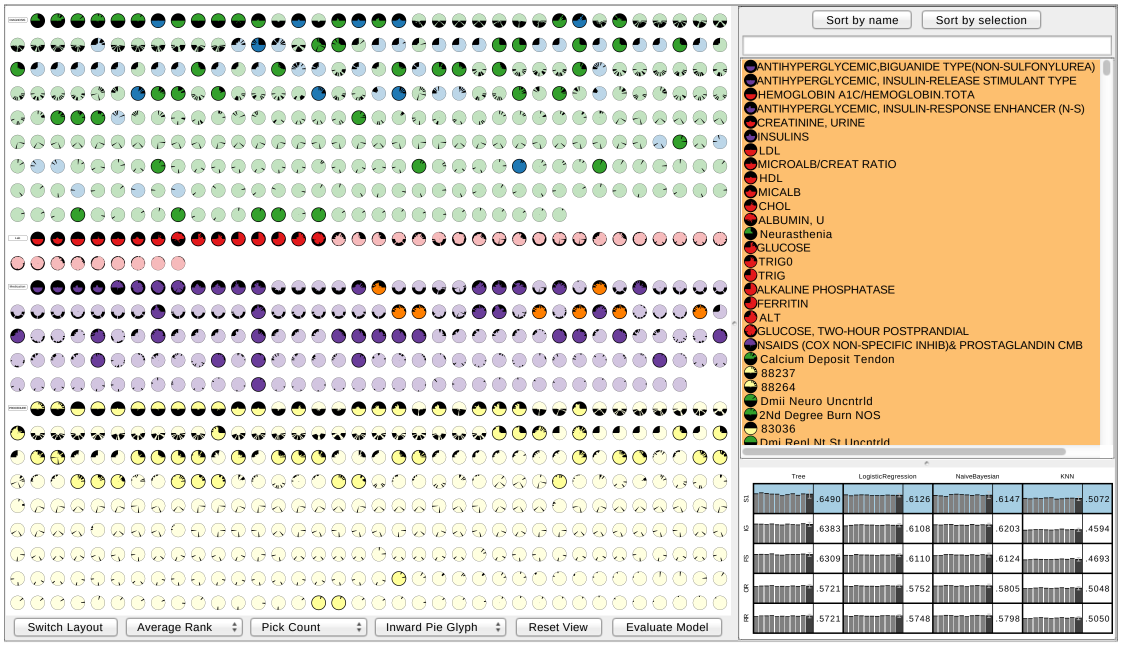
As illustrated in Figure 2.7, each row of the Classifier View represents the predictive model that resulted from each feature selection algorithm. Each column represents a classification algorithm. Multiple classifiers are used because there are a variety of techniques to evaluate models, and in order to avoid biases, *INFUSE* provides the ability to compare the output from multiple classifiers.

Each cell, as shown in Figure 2.8, has several components. On the left, there is a bar for each of the validation folds. The height of each bar corresponds to the AUC score for each fold. There is also a slightly thicker and darker bar immediately to the right of the fold bars, and its height represents the average value across all folds. This bar also features an error bar depicting the standard deviation across the folds. Finally, to the right of the bars, there is a numerical representation of the average AUC score. As this information is important for predictive modelers to reason about the quality of models, these values are given visual prominence. The bars, however, can be used to also reason about the quality across all folds.

Rows are sorted by the average AUC scores across all classification algorithms, so more accurate predictive models appear at the top of this view. Users can interact with this view in several ways. Clicking on a fold bar selects all features that were a part of this model and highlights them in the List and Feature views. The selected fold bar is highlighted in orange, and other scores this fold received by the other classifiers are highlighted in yellow so that they can easily be compared (as shown in Figure 2.7.)

2.3.5 Interactive Model Builder

One of the most important aspects of *INFUSE* is that in addition to allowing the comparison of models, it also enables the creation of new models based on insights. Users can select features for model building in a variety of ways. They can select all of the features from existing models by clicking on a model in the Classifier View. This will highlight and select all of the features that were used in the model. Users can then augment these lists, or start from an empty set, by selecting individual features when clicking on them in the Feature or List view. In order to select multiple features, a lasso selection technique is available in the



ANTIHYPERGLYCEMIC,BIGUANIDE TYPE(NON-SULFONYLUREA)
ANTIHYPERGLYCEMIC, INSULIN-RELEASE STIMULANT TYPE
HEMOGLOBIN A1C/HEMOGLOBIN.TOTA
ANTIHYPERGLYCEMIC, INSULIN-RESPONSE ENHANCER (N-S)
CREATININE, URINE
INSULINS
LDL
MICROALB/CREAT RATIO
HDL
MICALB
CHOL
ALBUMIN, U
Neurasthenia
GLUCOSE
TRIG0
TRIG
ALKALINE PHOSPHATASE
FERRITIN
ALT
GLUCOSE, TWO-HOUR POSTPRANDIAL
NSAIDS (COX NON-SPECIFIC INHIB)& PROSTAGLANDIN CMB
Calcium Deposit Tendon
88237
88264
Dmii Neuro Uncntrld
2Nd Degree Burn NOS
83036
Dmi Renl Nt St Uncntrld

	Tree	LogisticRegression	NaiveBayesian	KNN
S1	.6490	.6126	.6147	.5072
IG	.6383	.6108	.6203	.4594
FS	.6309	.6110	.6124	.4693
CR	.5721	.5752	.5805	.5048
RR	.5721	.5748	.5798	.5050

Figure 2.9: *INFUSE*'s Interactive Model Builder allows users to select sets of features and measure its quality. Selected feature glyphs are highlighted by saturating their color. The List View is sorted to show the selected features by their importance. After a user has made their selections, they can evaluate their model by clicking the "Evaluate Model" button. This adds a new blue row to the the Classifier View, showing the results of the evaluation for the user-defined model.

Ranked and Scatterplot layouts.

After a feature set has been collected, *INFUSE* can automatically evaluate the predictive performance of the user-defined model. By clicking the "Evaluate Model" button, the new model is scored across all cross-validation folds and classifiers, and the results are added in the Classifier panel as a new blue row. In the example in Figure 2.9, the user-defined model out-performed the automated models and it is ranked at the top of the Classifier View. Note that the user created model does not appear in the glyph. This is due to the fact that the user does not need to rank the features in order to obtain a classification result and that the feature set is equal for all cross-validation folds.

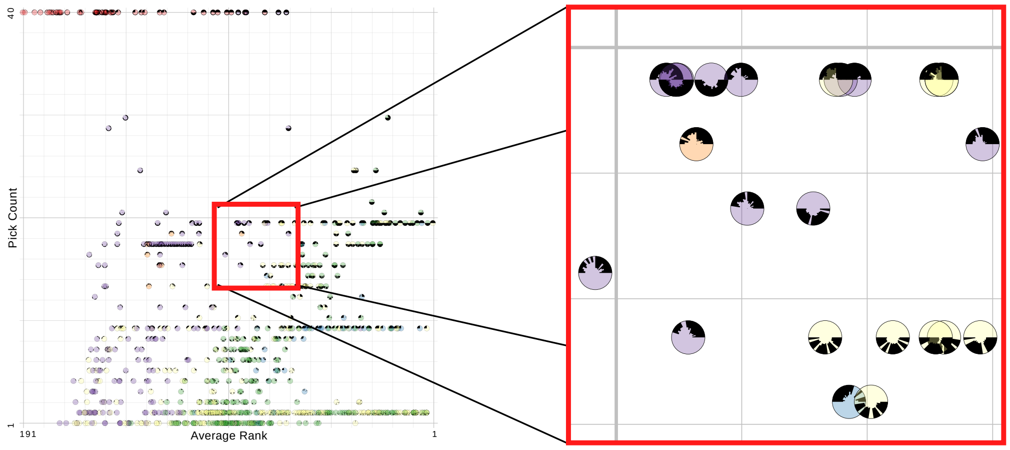


Figure 2.10: The scatter-plot view allows users to compare multiple types of rankings. In the case study, users became curious of the medication features that were chosen by only half of the models. When reviewing these medications with domain experts, it became clear that features picked by the upper-half algorithms were as clinically significant as those picked by the bottom-half. This indicates that merging results from feature selection algorithms makes sense for this predictive model.

2.4 Case Study

Throughout our paper, we have used a running example of a team of clinical researchers using predictive modeling to classify patients at high risk of developing diabetes. In this section, we describe how *INFUSE* has led to a variety of insights when exploring the features of the models.

2.4.1 Insight 1: Data Issues

When the clinical researchers learned of *INFUSE*'s capabilities to compare multiple feature selection algorithms, they decided to expand their pipeline's feature selection algorithms from 2 to 4. The team has used Information Gain and Fisher Score extensively in prior work, and typically uses these same techniques due to their familiarity and past success. Nonetheless, the diabetes data set introduced in Section 2.1.2 was new to them, and they were unsure which techniques would be the most appropriate. So, they asked their technologists to implement two new techniques: Odds Ratio and Relative Risk.

After all four algorithms were available, they executed their modeling pipeline using *PARAMO* [90] and connected the results to *INFUSE*. Instantly, the team was surprised at the patterns that the visualization made evident. The visualization indicated that there seemed to be little agreement between their two old algorithms, and their two new algorithms for the best features. The glyphs clearly indicated that many of the features were highly ranked by two of the four feature selection algorithms, and unranked by the other two, resulting in glyphs resembling alternating half-circles, as shown in Figure 2.11. The team was quick to note that the resulting accuracy across all four models were not significantly different, so this non-overlap would have probably gone unnoticed if the team just looked at resulting predictive scores at the end of the pipeline as they typically do.

As *INFUSE* gave them the opportunity to examine multiple algorithms at the feature-level, they were curious as to why this trend of non-overlapping feature rankings occurred. They investigated the scores associated with each feature rank and noticed that many of the features had scores of ∞ from the Relative Risk algorithm. It turned out there was a bug within the Relative Risk implementation where a divide by zero error could happen if a feature did not occur in any of the control patients. After fixing this bug, they noticed that much of the non-overlap still was evident. Looking more closely at the algorithms provided a reason why the two new algorithms behave very differently: they realized that both of the new algorithms only look at the presence and absence of the feature between the case and controls, and do not pay attention to the feature values in any other way (*e.g.*, distribution of values). This is in contrast to the Fisher Score and Information Gain algorithms that take the actual feature values into account. This means that for features that are present in both case and control groups in the same proportion, there is no discrimination value.

One of the team members mentioned, “Each feature selection algorithm captures different types of information. *INFUSE* allows you to see what the effect of that information is being captured and gives you insight into the robustness of your predictive model.”

As different algorithms will make sense for different purposes depending on the data set and goals, *INFUSE* provides an ability to inspect the features and determine which algorithms produce ranked sets of domain-relevant features.

2.4.2 Insight 2: Clinically Relevant Features

After the data issues were solved, the researchers began investigating the content of the predictive features. Using the scatterplot view, they inspected all of the medications that were ranked by all feature selection algorithms and folds and found that they were *antihyperglycemic* medications, which are common treatments to lower the blood sugar of diabetes patients, and made clinical sense to be ranked high.

However, looking towards the center of the scatterplot, where the features are only ranked by half of the algorithms and folds, the researchers noticed a cluster of medications that had half-circle patterns like those described above. This region is highlighted in Figure 2.10. By mouse-hovering these features to read their names, it became clear that those ranked high by the upper-half of the circle (Information Gain and Fisher Score) were as clinically relevant and similar as those ranked by the bottom-half algorithms (Relative Risk and Odds Ratio). This provided feedback that in predictive modeling it is not safe to assume that one single feature selection algorithm is able to detect all possible interesting features and also that having a system like *INFUSE* allows them to build a much richer picture of what kind of feature sets may lead to effective modeling. Without such a tool they would be restricted at evaluating one single algorithm at a time or, at best, restricting the comparison to a small number of features.

After interacting with the system one of the team members said, "*If you just use one feature selection algorithm, you're only getting certain types of features.* INFUSE *gives you a guide to what you might be missing. Using a combination type approach [with the Interactive Model Builder] will lead to stronger predictive models.*"

The clinical team is now going to re-think their strategy about how they build predictive models and may consider using features by merging top ranked features from different types of feature selection algorithms. The researchers are convinced that by merging features, in addition to the interactive model building capability, their predictive models will be improved.

2.5 Future Work and Conclusion

There remains a great deal of research to further improve the analytical process of predictive modelers. *INFUSE* only focuses on the feature selection step of predictive modeling. Each of the other steps would benefit from a visual interface to explore and parameterize the pipeline as well.

The search capabilities also have room for improvement by allowing more

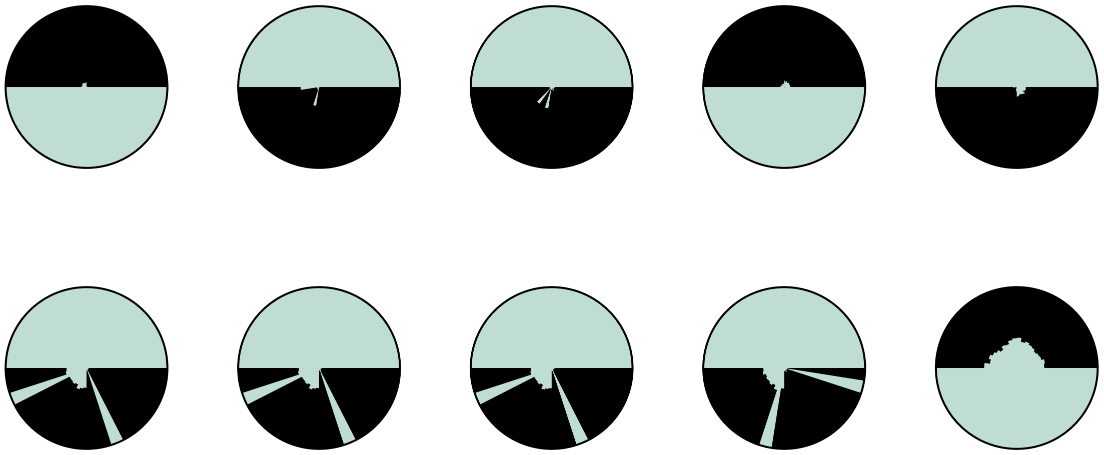


Figure 2.11: The clinical researchers found an interesting pattern among the glyphs indicating non-overlap of feature selection algorithm results. These features were highly ranked by 2 of the 4 feature algorithms, and unranked by the other 2, resulting in glyphs that resemble half-circles.

complex queries like features with a given range of ranks or features picked by a given algorithm, which would ease the task of finding relevant features for a user. Also, expanding the range of the search box to filter also in the Feature View may reduce the number of overlapping glyphs in the scatterplot view. Other clutter reduction techniques could also be available to users, such as a semantic zooming overlap resolution strategy that can jitter glyphs that overlap when the view is zoomed in.

Finally, to date, this tool has been used extensively for predictive modeling on clinical data. However, *INFUSE* was designed to be domain-independent and can easily be used for other domains in need of high-dimensional predictive modeling. Our future work includes additional case studies in other domains to ensure the robustness of our tools. This would also give the opportunity to explore the scalability of the design. Typically, the number of cross-validation folds is not more than ten. However, certain analysts may wish to compare a larger number of feature selection algorithms which would decrease the amount of space available per algorithm in the glyph. While similarly-ranked features would still appear visually alike, it may become difficult to identify certain algorithms or folds without the help of interaction. The overall number of features also plays a role in scalability concerns.

In conclusion, predictive modeling techniques are increasingly being used by data scientists to understand the probability of predicted outcomes. We present *INFUSE*, a tool that lets users interactively create predictive models. Typically, the predictive modeling pipeline leaves users out of the loop, and the algorithms operate as a black box. By giving users the power to interact with the results of feature selection, cross validation folds, and classifiers, *INFUSE* has shown promise to improve the predictive models of analysts. We further demonstrated how our system can lead to important insights in a case study involving clinical researchers predicting patient outcomes from electronic medical records.

The authors thank Kenney Ng for providing his expertise in predictive modeling. The authors also thank the anonymous healthcare institution who provided the data for the clinical researchers experiments.

2.6 General Discussion

INFUSE [66] explored the viability of the popular method of using feature importance measures to understand model behavior. Even though there is no *direct* influence from the model itself feature importance gives valuable insights due to its role in feature selection for the model and the similarity of its computation and the model's construction. However, our experiments showed that there is little consensus between different feature importance algorithms, even though equal model performances were achieved. This is likely due to redundancies in features in the input data but shows that those methods are not able to properly point out those occurrences. Furthermore, it also shows that those feature importance methods are too far from the actual model to gain sufficiently strong insights into the model's behavior. This result encouraged us to explore techniques of model dependent feature importance metrics, such as *Prospector* described in the next Chapter.

Chapter 3

Prospector

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 instances 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.*

Summary:

Research Question:

How can partial dependence be leveraged to gain diagnostic insights into machine learning models?

Key Findings:

- The partial dependence based feature importance score allows to effectively detect model errors.
- Detectable errors include: over-fitting, under-fitting, biases in the data caused by imputation, and leaking labels caused by incorrect cause-effect relationships.
- Localized inspections help to understand the how and why of specific instance predictions by finding locally impactful features.

Josua Krause, Adam Perer, Kenney Ng

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 [70]. As an example, in health care, if one can model the data characteristics of patients who will likely develop Diabetes, health care 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 [5] to interactive feature selection [66]. 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 are novel visual representations, sampling strategies, and support for comparing multiple models.
  - Localized inspection to understand how and why specific instances 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.

3.1 Motivation

3.1.1 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 an instance 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.

3.1.2 Predictive Modeling in Health Care

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 health care

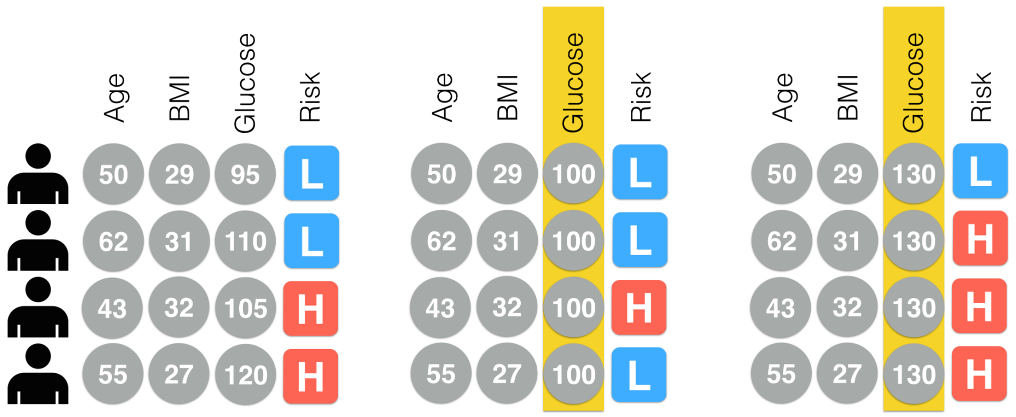


Figure 3.1: An illustration of how partial dependence is computed for the feature "Glucose". On the left are four patients' original feature values. In the middle, the "Glucose" values are all changed to 100, and the corresponding predictions change. Similarly, on the right, the "Glucose" values are all changed to 130, and again the risks are different. This demonstrates the impact of the "Glucose" feature on risk prediction.

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.

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 [70]). 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 health care 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?

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.

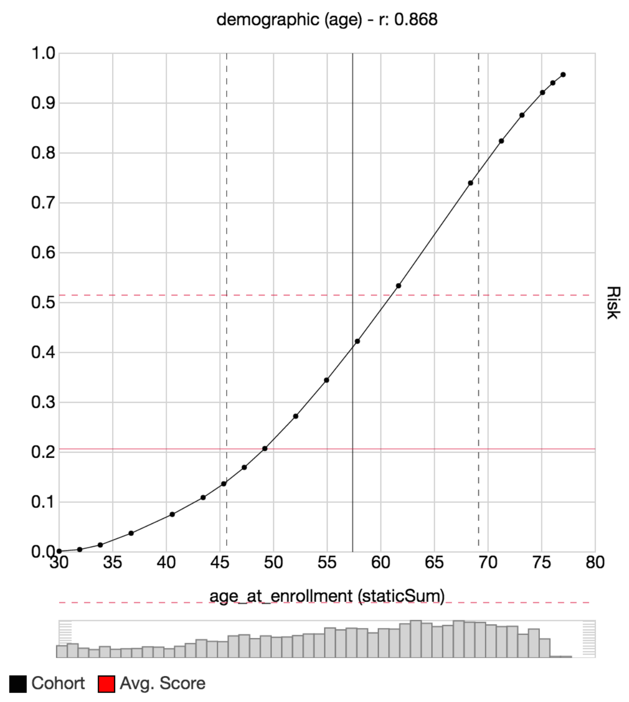


Figure 3.2: Partial dependence plots. The black curve shows the average predicted risk (probability of a certain outcome) if for all rows in the data the value of this feature was the given value of the horizontal axis. The red line shows the average predicted risk for the original data. The vertical 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.

3.1.3 Partial Dependence

The most widely used technique to understand the relationship between a feature and a prediction is by computing partial dependence [41, 48]. 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 (3.1) can be used to compute a partial dependence plot.

$$pdp_f(v) = \frac{1}{N}\sum_{i}^{N} pred(x_i) \text{ with } x_{if} = v \tag{3.1}$$

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 3.1, 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 3.1a, 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 3.1b, 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 3.1c, 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 3.2, which is a line graph that plots the fixed values of the target feature on the x-axis, and the corresponding predicted risk (*i.e.*, probability of a certain outcome) on the y-axis.

3.2 Related Work

3.2.1 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 [21]). 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 [74] 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.* [71] 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.* [93] use multiple classifiers to better understand input data. Steeg and Galstyan [111, 124] use total correlation to build a hierarchy of features explaining their role in the data.

3.2.2 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.* [13]) and how the input is propagated (Correa *et al.* [58]) has been used. For Support Vector Machines, projection techniques (Caragea *et al.* [24]) and Nomograms (Jakulin *et al.* [55]) to see the "cut" in the instances were utilized. Visualizing and interpreting the graph of a neural network has also been used by

Tzeng and Ma [121]. Kim *et al.* [61, 63] introduce graphical models that allow for interpretability of its features. [61] use a colored matrix of features by category to show distinguishable features computed by their model. Caruana *et al.* [25] use high-performance generalized additive models (GA^{2}Ms) that allows visual inspection of the influence of its input features on the outcome much like partial dependence.

3.2.3 Model Result Visualization

However, showing only internal, algorithm specific weights is often not enough. Plate *et al.* [98] and Olden [91] show how input features influence the outcome of neural network classifications. Xu *et al.* [129] 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 [122]) offers interactive decision tree construction. Steed *et al.* [109, 110] guides regression model creation by enhancing interaction with the input data. EnsembleMatrix (Talbot *et al.* [114]) allows users to combine already computed classification models to build ensemble models. Some recent interactive machine learning tools [5, 6, 7, 8, 60, 62, 72, 86, 92]) are more algorithm agnostic, but depend on general performance measures like confusion matrices, area under ROC curve (AUC) measures, result distribution of instances, and feature weights according to model independent statistics.

Frank and Hall [39] use 2D projections of the data to show results for multiple classification algorithms, as well as Rheingans and desJardins [101] with self-organizing maps.

3.2.4 Probing Models

Partial dependence was proposed by Friedman [41] for analyzing gradient boosting models and has since been used for other models as well (*e.g.*, Ehrlinger [37] uses partial dependence to analyze the behavior of random forests in the *R* package *ggRandomForests*).

Cortez and Embrechts [32, 33] and Kim *et al.* [63] 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 instances. 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.* [42] 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 instance.

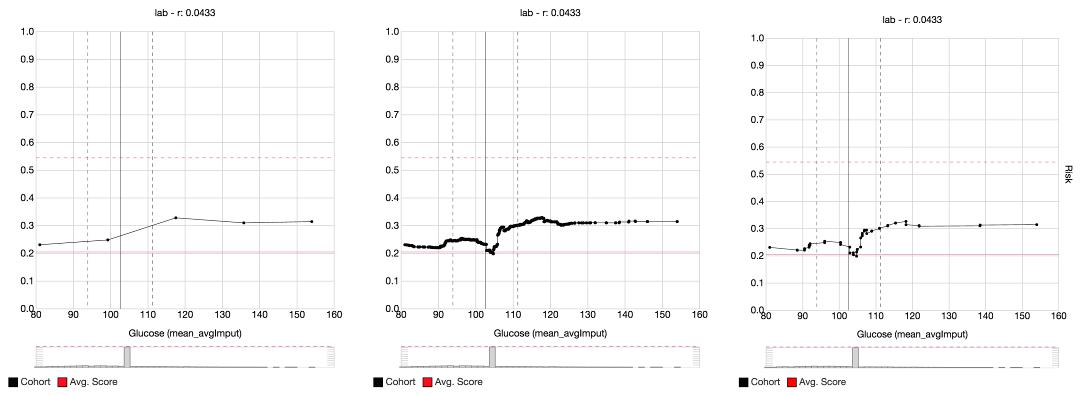


Figure 3.3: Different sampling strategies for partial dependence plots. The leftmost plot uses a naïve sampling which misses the dip in the predicted risk for a Glucose value around 105. Using the thresholds of the trees in the random forest the middle plot shows all details of the displayed model. The 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.

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 instances 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.

3.3 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 [90]. 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 [73].

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.

3.3.1 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 3.2. 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.

3.3.1.1 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 [37] 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:

- Some sampled prediction values are not included in the interpolation.
- The interpolation is a curve where it should be a constant which alludes to values that are impossible to achieve with the prediction function.
- 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 3.3). 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 end-points. 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.

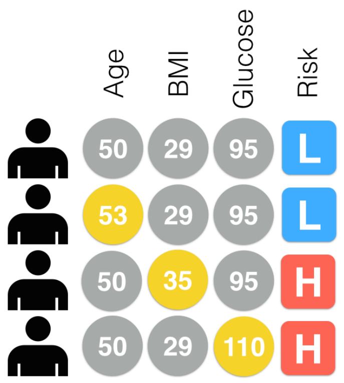


Figure 3.4: This illustration provides an explanation of how local inspection works. On the top row are the patient's original feature values and the corresponding prediction. On the bottom three rows, users changed certain values of the patient, highlighted in yellow, and such values impacted the prediction.

3.3.2 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 3.4 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 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 3.4 we do this for all features independently yielding local partial dependence plots for each feature using a single input row.

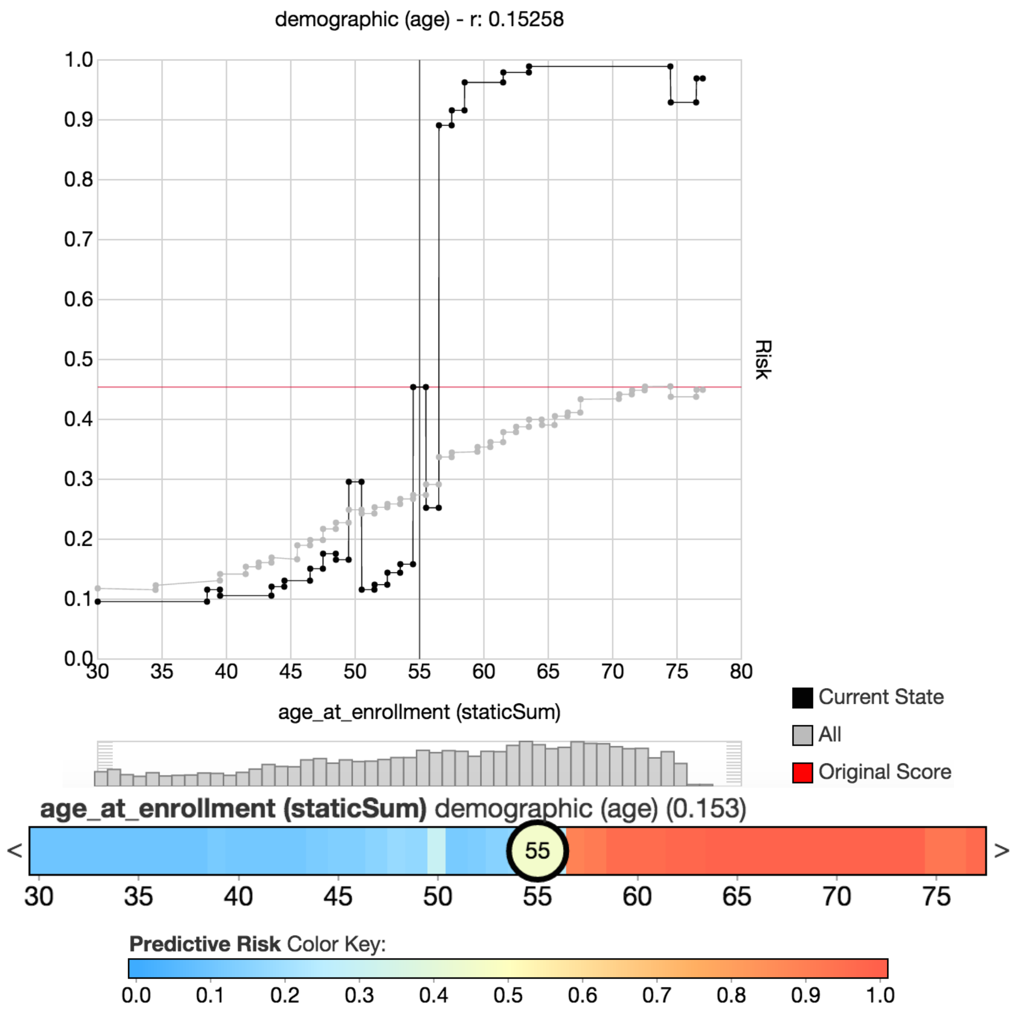


Figure 3.5: The same feature shown as line plot (top) and partial dependence bar (middle). Color indicates the predicted risk for the outcome. The color mapping is shown at the bottom.

3.3.2.1 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 3.5. 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.

3.3.2.2 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 (3.2) 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.

$$I_f(p) = \int_{-\infty}^{\infty} [pred(p^*) - pred(p)]\ \omega(v,\, p_f)\ dv \tag{3.2}$$

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

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

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.

$$\arg\max_{v} \left[s \;\; [pred(p^*) - pred(p)] \;\; \omega(v, p_f) \right] \tag{3.3}$$

Formula (3.3) 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 (3.2). The changes yielding the highest impact can be interpreted as suggestions for changing a data point.

3.3.2.3 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 3.6. This is useful for comparing the expressiveness of models and seeing which models are possibly under- or over-fitting the input data.

3.3.3 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.

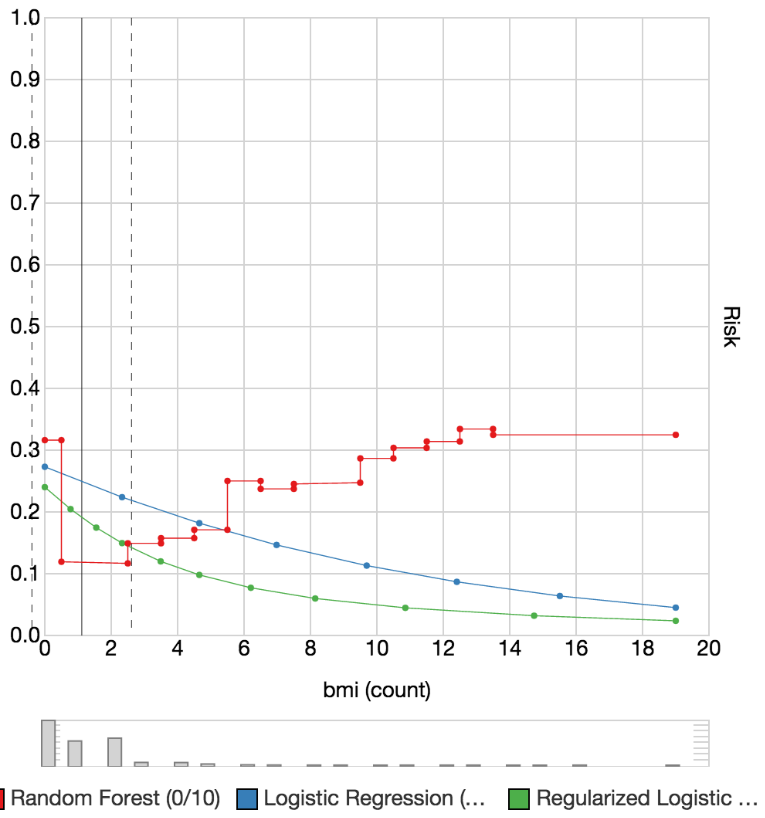


Figure 3.6: Comparison of three machine learning models on the number of measured BMI values. The two regression models (logistic regression in blue and regularized logistic regression in green) can express only a single slope (downwards or upwards) whereas the random forest in red can model the strong 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.

3.3.3.1 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 3.7, 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 3.8. 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.

3.3.3.2 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 3.9. 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 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 order, the suggested changes to each feature are indicated with a white circular label in the partial dependence bar, shown on the bottom left of Figure 3.9.

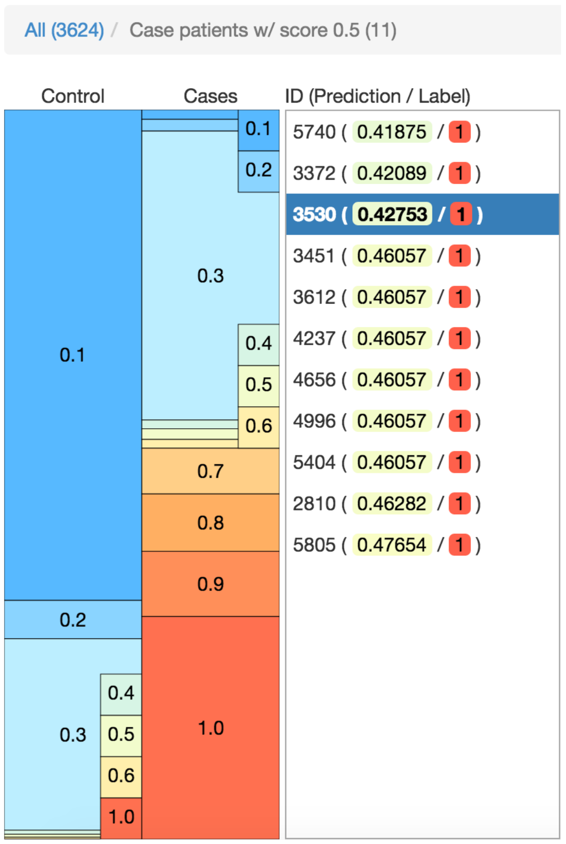
Often after analysts have inspected a particular patient, they may wish to find other patients similar to them to see how they react to the predictive model. If users wish to find patients similar to the one they are inspecting, they can click on the "Neighborhood" button and *Prospector* will automatically 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.

3.3.3.3 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 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.

3.4 Case Study: Predicting Diabetes

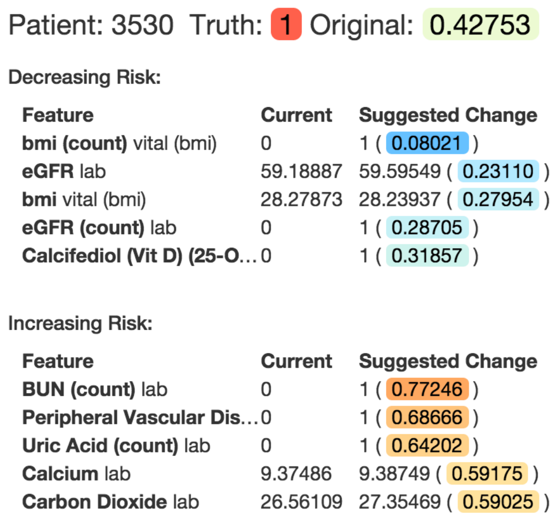
In order to evaluate the utility of *Prospector*, we chose to conduct a case study utilizing a team of real data scientists building their own predictive models on their own real-world datasets to demonstrate its effectiveness at reaching insights in practice. There is a growing belief in the visualization community that traditional evaluation metrics (*e.g.*, measuring task time completion or number of errors) are often insufficient to evaluate visualization systems [15, 97, 106]. Using the evaluation methodology developed by Perer and Shneiderman [94], 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


All (3624) / Case patients w/ score 0.5 (11)

ID (Prediction / Label)

5740 (0.41875 / 1)
3372 (0.42089 / 1)
3530 (0.42753 / 1)
3451 (0.46057 / 1)
3612 (0.46057 / 1)
4237 (0.46057 / 1)
4656 (0.46057 / 1)
4996 (0.46057 / 1)
5404 (0.46057 / 1)
2810 (0.46282 / 1)
5805 (0.47654 / 1)

Figure 3.7: 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.



Patient: 3530 Truth: 1 Original: 0.42753

Decreasing Risk:

Feature	Current	Suggested Change
bmi (count) vital (bmi)	0	1 (0.08021)
eGFR lab	59.18887	59.59549 (0.23110)
bmi vital (bmi)	28.27873	28.23937 (0.27954)
eGFR (count) lab	0	1 (0.28705)
Calcifediol (Vit D) (25-O…	0	1 (0.31857)

Increasing Risk:

Feature	Current	Suggested Change
BUN (count) lab	0	1 (0.77246)
Peripheral Vascular Dis…	0	1 (0.68666)
Uric Acid (count) lab	0	1 (0.64202)
Calcium lab	9.37486	9.38749 (0.59175)
Carbon Dioxide lab	26.56109	27.35469 (0.59025)

Figure 3.8: 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.

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

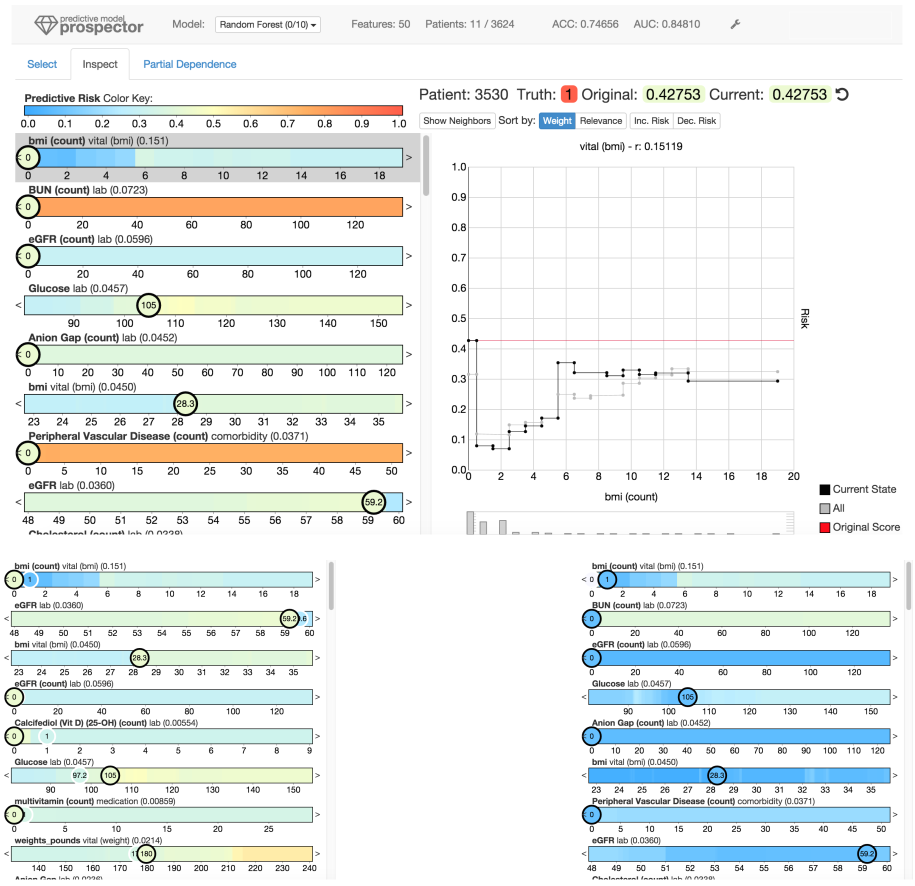

predictive model prospector
Model: Random Forest (0/10)
Features: 50
Patients: 11 / 3624
ACC: 0.74656
AUC: 0.84810
Select
Inspect
Partial Dependence
Patient: 3530 Truth: 1 Original: 0.42753 Current: 0.42753
Predictive Risk Color Key:
Show Neighbors
Sort by: Weight Relevance Inc. Risk Dec. Risk
bmi (count) vital (bmi) (0.151)
BUN (count) lab (0.0723)
eGFR (count) lab (0.0596)
Glucose lab (0.0457)
Anion Gap (count) lab (0.0452)
bmi vital (bmi) (0.0450)
Peripheral Vascular Disease (count) comorbidity (0.0371)
eGFR lab (0.0360)
vital (bmi) - r: 0.15119
Risk
bmi (count)
Current State
All
Original Score
Calcifediol (Vit D) (25-OH) (count) lab (0.00554)
multivitamin (count) medication (0.00859)
weights_pounds vital (weight) (0.0214)

Figure 3.9: 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.

or using ICE plots [42]. 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.

3.4.1 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 3.6, 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 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 instances 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.

3.4.2 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 3.3, 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.

3.4.3 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.

3.4.4 Impact on 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.

3.5 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 instances 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 instances 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.

3.6 General Discussion

Prospector [69] explored model dependent feature importances that allow for a fine grained value influence analysis. From that we derived local feature importances that apply to single instances and give insights into the decision making process of the machine learning model in the given case. We showed that this method can be used to verify strengths and understand short-comings of models. However, the method focuses on individual features preventing insights related to combinations of features. Furthermore, the user has to choose between a heavily aggregated global view of the model's decision making or an individual instance-level view which is too fine grained to be useful for models with many instances. To overcome those problems we developed the workflow presented in the next Chapter.

Chapter 4

Explainer

Human-in-the-loop data analysis applications necessitate greater transparency in machine learning models for experts to understand and trust their decisions. To this end, we propose a visual analytics workflow to help data scientists and domain experts explore, diagnose, and understand the decisions made by a binary classifier. The approach leverages "instance-level explanations", measures of local feature relevance that explain single instances, and uses them to build a set of visual representations that guide the users in their investigation. The workflow is based on three main visual representations and steps: one based on aggregate statistics to see how data distributes across correct / incorrect decisions; one based on explanations to understand which features are used to make these decisions; and one based on raw data, to derive insights on potential root causes for the observed patterns. The workflow is derived from a long-term collaboration with a group of machine learning and healthcare professionals who used our method to make sense of machine *learning models they developed. The case study from this collaboration demonstrates that the proposed workflow helps experts derive useful knowledge about the model and the phenomena it describes, thus experts can generate useful hypotheses on how a model can be improved.*

Summary:

Research Question:

How can black-box explanations help diagnose model errors?

Key Findings:

- The Model Diagnostic workflow enables a scalable understanding of the local decision making of a predictor.
- Aggregation of instance-level explanations allows for semantic validation of the input data.
- A model can only perform as well as its input data.
- Gaining insights about the limitations of a model in turn helps with feature engineering on the input data.

Josua Krause, Aritra Dasgupta, Enrico Bertini

In this paper we propose an interactive workflow and a visual user interface to help data scientists and domain experts diagnose and validate binary classifiers. The approach we suggest is based on a mix of automated and interactive methods that guide the user towards understanding what decisions a model makes, which ones are correct or incorrect, and potential strategies to improve them.

Being able to explore the decisions a model makes and identifying potential issues is crucial in application areas where experts need to get a sense of how the model works and build trust in its decisions. While common practice in much of the machine learning endeavors is to focus on model accuracy, many researchers have voiced the need for more transparency when the application domain requires it [12, 25, 40, 80, 82, 123]. A recent DARPA (Defense Advanced Research Projects Agency) program called "Explainable AI (XAI)", for example, calls for more research in this area and declares, as the main motivation for the program that "*the effectiveness of these systems is limited by the machines current inability to explain their decisions and actions to human users*" and that "*it is essential to understand, appropriately trust, and effectively manage an emerging generation of artificially intelligent machine partners*".

In addition to evaluating a model in terms of accuracy, we propose the idea of *semantic validation*, the need for domain experts to verify that the decisions a model makes are plausible when compared against their mental models of the problem. For instance, in healthcare settings, medical doctors often want to see examples of recommendations the model provides and need to gain trust in it before they feel comfortable with deploying it in real-world settings. Such reservations in deploying models without having an opportunity to manually verify what decisions they make are well justified as it is entirely possible for a model to achieve high accuracy and yet provide dramatically erroneous recommendations [25].

Another important factor to consider is that domain experts and data scientists are often working in collaboration to solve a particular problem (or they are actually the same person covering both roles). Being able to manually inspect a model can give them an opportunity to generate useful insights on how a model can be improved. While commonly used aggregate statistics such as area under the curve (AUC) give a sense of the overall accuracy of the model, and can be used as a parameter to compare between different models, they do not provide insights on how or why a model fails to capture important phenomena accurately.

Some existing methods do provide more transparency and useful information for enabling better understanding and diagnostic purposes, but they tend to be limited and specific to a particular kind of model. For example, *logistic regression* and *decisions trees* are commonly regarded as more interpretable models thanks to their ability to provide information on feature weights and / or specific decisions the model makes (decision trees) [40]. These solutions are however limited by a number of factors. Since they are specific to the selected method, they are hard to generalize and cannot be applied transparently to other types of models. Furthermore, they only provide a limited picture of what decisions the model makes. Feature weights provide a highly coarse summary of how relevant features are *globally*, but they do not provide information on how the model makes decisions *locally*, for a selected set of instances. Even more transparent methods, like decision trees, tend to grow very large and are not easy to parse visually, especially for data sets with a high number of dimensions / features.

To address these issues we propose a workflow, aimed at machine learning experts and data scientists, based on *instance-level explanations*, computational methods to derive a description of how a model makes decisions on single data items, without having access to the internal logic of the model (*i.e.*, using the model as a *black box*). These explanations are then aggregated and used as input to a visualization system that enables the browsing of model decisions and assessment of their quality.

The work we describe in the paper stems from a one year collaboration with a group of domain and machine learning experts from the *NYU Langone Medical*

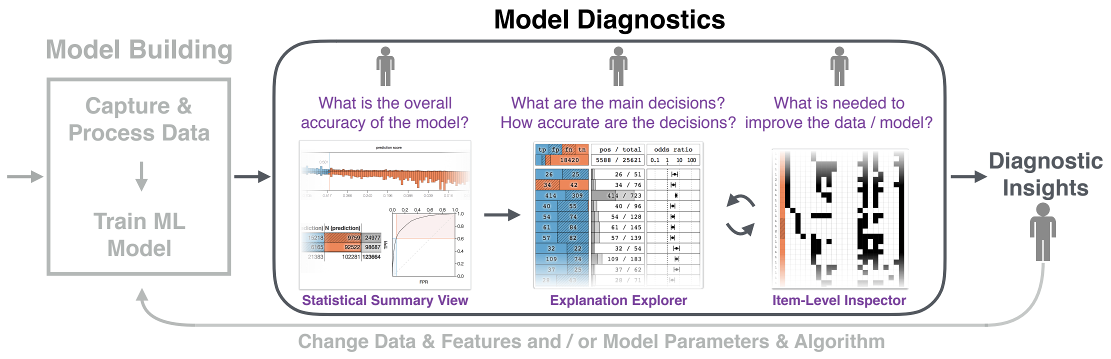

Model Diagnostics
Model Building
Capture & Process Data
Train ML Model
What is the overall accuracy of the model?
What are the main decisions? How accurate are the decisions?
What is needed to improve the data / model?
Statistical Summary View
Explanation Explorer
Item-Level Inspector
Diagnostic Insights
Change Data & Features and / or Model Parameters & Algorithm

Figure 4.1: Our proposed **Model Diagnostics** workflow extends the conventional *Model Building* workflow in machine learning for enabling domain experts to reason about the semantic validity of the decisions made by any model through multiple linked visualizations of statistical performance summaries, explanations, and item-level distribution of features. By iterating through explanation-level summaries and item-level details, experts are able to generate diagnostic insights about the quality of both the data and the model. This ultimately helps to improve data acquisition and model generation processes belonging to the original workflow.

Center. In our collaboration, we worked together to make sense of models built to understand how patients are handled in the hospital and to figure out whether important outcomes of interest can be predicted correctly. This resulted in the development of an interactive model diagnostic workflow using visual explanations of model behavior that is the main contribution of this work. The rest of the paper is organized as follows. We present related work in the next section. We provide an overview of model diagnostics goals and of the proposed workflow in Section 4.2. We then describe in detail the instance-level explanation algorithm we use in Section 4.3 and the interfaces we built in Section 4.4. Section 4.5 reports on a use case we built to show how the workflow can help perform useful and actionable model diagnostics. Section 4.6 discusses the results and provides a number of reflections and lessons we have learned from this collaborative exercise.

4.1 Related Work

In the following, we discuss model explanations and visual analytics techniques used for interacting with classification models.

4.1.1 Model Explanations: *Why* and *How*

Explanations of behavior of autonomous systems [75] or computational models [35] can lead to a high degree of human-machine trust. In machine learning, model explanations are beginning to be used in human-in-the-loop data analysis applications for communicating information about model behavior and predictions. While there are some studies [113] that show that explanations can lead to over-reliance on the system, generally it has been posited that model explanations lead to a high degree of human interpretability and trust [76]. Similar to the latter, our goal in this work was to develop a visual analytic workflow for model explanations and to work closely with data scientists and domain experts to understand how that could lead them to understand and trust model behavior.

In the literature, we find two contrasting purposes behind generating model explanations. The first approach is embedded within the interactive machine learning pipeline and helps end users in refining a model's predictions by interacting with the model structure. This helps users to build a mental model about the model reasoning process [71]. Through the EluciDebug approach, Kulesza *et al.* lay out a set of principles for the process of explanatory debugging using a Naïve Bayes classifier model. Although the principles are generally applicable, the explanation technique is specifically applicable only to a particular model. Furthermore, additive models enable intuitive explanations through feature contributions that allow both to visualize the decision making process for single instances [99] and feature contributions on a population level [25]. However, this solution requires the use of an additive model and as such it is not generally applicable.

To overcome this limitation, a second approach for explanation generation is to treat the machine learning model as a black box, bypassing the model structure, while communicating the input-output relationships and their relevance to a model's decisions to an analyst, *e.g.*, inferring rules from a neural network [34], or generating explanations [69, 102]. We adopt this black-box approach in our workflow for benefiting domain experts, who are not trained in machine learning, and also for providing data scientists with a model-agnostic and generalizable diagnostic interface for inspecting model quality. In previous work, local explanations have been used to diagnose how models make decisions for single instances of a data set [66, 69, 102]. In contrast, we provide an interactive workflow where users can explore aggregated representations of explanations and better understand the context of model decisions by iterating across explanation-level and instance-level visual summaries of prediction quality.

4.1.2 Human-in-the-Loop Inspection of Classifiers

Researchers have recently demonstrated how human interventions can help in greater accuracy in construction of classifiers, when compared with a purely automated approach [115]. In this work, Tam *et al.* used information theory to show how soft knowledge of model developers can be encoded in decision trees, and they advocate a tighter integration between human and machine-centric processes for model development. The goals for integrating visual analytic techniques and classification methods fall broadly into three categories, as proposed by Liu *et al.* [78]: i) model understanding, ii) model diagnosis, and iii) model refinement. Our proposed diagnostic workflow (Figure 4.1) encompasses the goals of understanding model behavior and diagnosing the model decision space for enabling data scientists and domain experts to generate insights about potential inadequacies in the data and in the model quality. The refinement step is an obvious action as a result of these insights, but is outside the scope of our work.

Analyzing summary statistics of model performance through the lens of visualization techniques is the most common approach for finding matches and mismatches between model predictions and ground truth data. To this end, ModelTracker [5] provides a unified interface for error detection and debugging for binary classifiers showing item-wise distributions of prediction scores. Bilal *et al.* propose the confusion wheel visualization [2] and other linked views to show probabilities of items belonging to different classes for multi-class classifiers. Squares [100] provides a single, unified visualization of performance metrics and easy accessibility to the data for debugging multi-class classifiers. For enhancing the interpretability of classifier predictions, Cortez and Embrechts [32] use a sensitivity analysis approach for letting users understand the effects of variation of input values on model outputs. While these methods are able to diagnose performance issues at the level of a single item [2, 5, 100] or single features [32], they lack a holistic summary of the entire decision space that exposes associations among subsets of items and features, and communicates the reasons behind the model decisions. Through an explanation-based approach, we can let analysts explore these associations for a large, high-dimensional data set, drill-down to individual items, and diagnose potential problems with respect to both global and local decisions. This leads to actionable insights about the limits to which model quality can be improved, and ultimately, hints about how to improve the data.

4.2 Model Diagnostics

We use the term *Model Diagnostics* to indicate the steps necessary for a domain expert or a model developer to semantically validate the decisions made by a model using their domain knowledge. In this section we outline the different goals for a user when using a model diagnostic interface and provide an overview of the implementation of the resulting workflow (Figure 4.1). The workflow was derived through a long term collaboration among visual analytic researchers and model developers and domain experts in the medical field, specifically in the application scenario of hospital visits. The over-arching goal in this scenario is to use predictive modeling for reducing patient wait time and optimizing the hospital resources needed for admitted patients.

4.2.1 User Goals

In the course of our interactions with domain and machine learning experts and analyzing a variety of model building problems, we realized that the model diagnostics problem can be decomposed into the following main goals; which we express as a set of questions as shown in Figure 4.1.

G1: What is the overall accuracy of the model? In this step, experts need to get an overview of the distribution of prediction scores across the data items, derive an understanding about the uncertainty associated with predictions of certain items, and generally where the predictions are correct or incorrect.

G2: What are the main decisions the model makes? A trained classifier creates a decision space that maps a (potentially high-dimensional) input space into the output space defined by the two labels *true* and *false*. Understanding what these decisions are and how frequently they are made is a crucial piece of knowledge domain experts want to draw from the classifier. For instance, in the healthcare scenario we explore in this paper it is crucial for domain experts to know that the vast majority of decisions the classifier makes are based on a small set of drugs (features). They also want to ensure that different sets of drugs are used by the classifier to make decisions about different sets of patients (*e.g.*, a group of patients is characterized by *Ondansetron* and *Sodium Chloride*, whereas another is characterized by antibiotic drugs).

G3: How accurate are the decisions the model makes? Together with knowing what decisions the model makes, it is crucial to also know how accurate these decisions are. Using the same example as above, it is not sufficient to know that the model classifies a group of patients according to the drugs they received, but also how often this decisions are correct or incorrect.

G4: How can one change the data or the model to improve its decisions? Understanding decisions and assessing their accuracy is relatively useful, but the ultimate goal for a model developer is to actually gain *actionable* insights on how the model can be *improved*. Some of the insights experts want to derive include: whether the model parameters should be tuned or a better set of features should be derived.

In this work we do not provide specific support for the actual parameter tuning or data processing steps necessary to improve the model. The black-box nature of our approach is illustrated in Figure 4.1, which shows that the model diagnostic workflow is an extension of (and not a part of) the existing model building workflows that data scientists follow as part of their routine. Modelers have specific ways and tools to perform these steps and intervening on their established practices is out of the scope of this work. Rather, in this work we focus on providing support for the diagnostic part experts may want to execute at the end of each modeling round and which is currently not well supported by existing tools and practices. The diagnostic insights produced by our workflow provides hints about whether the input data or the model structure needs to be changed for improving the prediction quality.

4.2.2 Workflow

The workflow we propose results from two pre-processing operations: *explanation generation* and *visual mapping*.

Explanation generation takes as an input a data set and a trained binary classifier and creates for each instance in the data set an *explanation*. An explanation is a description of the logic (or rule) the classifier uses to assign a given label to the instance. For this purpose, we leverage a method developed by Martens and Provost [82], which computes, for a given instance which features need to be "removed" in order to change the classification outcome. For instance, in a text classification problem, an explanation for a document consists of the words that need to be removed in order to change the label originally assigned by the classifier. In Section 4.3 we describe in more detail how the explanation method works.

Visual mapping takes as an input the data set and the set of explanations, and builds a set of interactive visualizations (Figure 4.1) that support the user goals we outlined above. The interactive workflow revolves around three main linked interfaces; each one supporting the analysis of model decisions at different levels of granularity and addressing the user goals.

Outcome-level. The first step focuses on overall accuracy of the model, using a representation similar to a confusion matrix. The main goal of this step is to get a sense of how data distributes across the prediction score computed by the classifier (typically a score between $[0, 1]$), and the four possible outcomes: true or false positive and true or false negative. By visualizing how data distributes across the four possible outcomes the user can gain a sense of how accurate the model is (**G1**) and whether errors cluster around particular sets of scores.

Feature-level. The second step uses the computed explanations to generate an overview of decisions made by the classifier and their accuracy. Each explanation is described by the set of features it uses to explain an instance and, as such, it provides a description of how the model makes its decisions. In this step, we group together all the explanations (and thus the instances) that contain the same set of features, compute accuracy statistics on top of them, and use these groups as a visual interactive summary of the decisions the model makes. By visualizing the explanations and their accuracy the user can get a sense of what are the major decisions the model makes and how accurate they are (**G2, G3**).

Instance-level. The third step focuses on the analysis of a single user-selected explanation and the instances it explains. Once an interesting explanation has been found in the previous step, it is often useful and necessary to drill-down to the individual instances to observe how the data items contained in an explanation distributes in the original data space. Being able to observe their actual data values and the decisions the model enables experts in formulating hypotheses about why the classifier fails to make correct decisions with some instances. In other words, when it is possible to visually compare the data values of instances that have the same explanation but different outcomes, users can draw inferences on the root cause of the diverging outcomes. Therefore, by visualizing single instances the user can reason on how the model makes decisions and derive potentially useful hypotheses about how they can be improved (**G4**).

These three steps are linked in a sequence by user-driven filtering mechanisms. The user can select specific sets of values at the outcome-level and visualize them at the features-level. While observing the main set of decisions at the feature-level, she or he can select specific explanations and inspect individual instances in the instance-level interface.

It is important to stress the key role explanations play in the workflow. By computing the explanations and computing statistics on top of them we can effectively provide a description of the main set of decisions the model makes *without* having access to the internal logic of the model. The relevant aspect of explanations is that they compute a compact description of which features the model uses to make *local* decisions for a *subset* of instances. For example, in the medical data analysis explored in this work, where each patient is described by the medications he or she received (features) and the classifier predicts whether the patient will be admitted or not, an explanation can identify a group of patients characterized by a small set of medications; that is, the medications the classifier uses to make its prediction.

4.3 Explanation Method

Using explanations, we intend to group data items from the perspective of the machine learning model being analyzed. In order to do so without relying on a particular model, that is, treating the model as black box, we can estimate which features were involved in its decision making process. In our initial approach, we had explored alternative methods for grouping the data by looking only at prediction scores of the model [67]. However, we realized that those methods mostly reflect the intrinsic structures of the data set instead of the decision making process of the model. Therefore, in this work we build explanations by finding the minimal amount of change necessary to change the prediction of the analyzed model, specifically, a binary classifier. Also, contrary to our previous approach of using explanations to detect only the commonly used features by a model [116], here we focus on explanations as a way for experts to diagnose correct or problematic model behavior and address the goals *G1, G2, and G3*, that were outlined in Section 4.2.

Explanations are created using a trained model by creating synthetic input values derived from observed data items revealing this input-output relationship. The set of changes to the values that swayed the outcome of the prediction is then called explanation e for the given original data item:

$$\min_{e} L(v - e) \neq L(v)$$

where L is the label function with "positive" or "negative" as result and v is the data item to be explained. In order to compute e, the prediction function P of the classifier is used with a threshold t:

$$L(v) = P(v) > t$$

The output of P, the prediction score, is a number between 0 and 1 indicating the confidence of a classifier in the predicted outcome. The threshold t is chosen to yield the most correctly predicted items on the training data.

Prospector [69] and LIME [102] both propose algorithms that can be used to create explanations. The metric used for minimizing e depends on the explanation technique. Prospector assumes feature independence and thus minimizes e by combining one-dimensional impactful changes of the prediction score. LIME on the other hand creates a local new simpler model by sampling the neighborhood of the analyzed data item and extracts e from this transformed local space. Those two methods aim to approximate minimal explanations in real valued high-dimensional input data spaces.

In our case we are dealing with high-dimensional *binary* input data. For most applications, like text analysis or movie recommendation, binary input data is sparse, *i.e.*, almost all feature values are 0 instead of 1. Therefore, binary data can also be interpreted as a bag of features. That is, a data item can be treated as set of features whose value is 1.

Martens and Provost [83] provide an algorithm for computing minimal explanations for binary input data. As Prospector and LIME only generate approximate explanations for data items we adopt and extend this method instead. The method allows for only removing features from the bag of features. This restriction comes from the observation that allowing additions to the bag of features can “tone out” the original item by adding unrelated features with high impact on the prediction score.

The algorithm to generate explanations using this method consists of successively removing features from the bag of features until the prediction outcome changes. The order of the removal is determined by the largest change in prediction score when removing a feature. *The set of features that are removed from the bag of features in order to change the outcome of the prediction is then called an explanation of the original data item.*

One problem with the original algorithm is that it contains a series of conditions that make it give up on explaining some of the instances when these conditions are met. In our case however we want to be able to provide a full picture of the data set and as a consequence we want to create an explanation for every instance provided in the data. For this purpose we decided to introduce a few modifications to the original algorithm:

- The original algorithm enforces a maximum length of explanations and declares an item as unexplainable if it fails to find an explanation that is shorter than the limit. In our implementation we removed this restriction. The main consequence of this modification is that sometimes the algorithm may produce explanations that are very long and unintelligible. Those explanations however are interesting because they can help us detect and visualize edge cases which may reveal surprising information. In addition, having many long explanations that explain only a few data items can be an indicator of a highly complex model with few similar instances or that the model is overfitting as it is trying to memorize individual labels.
- The algorithm can run into plateaus where removing any feature does not change the prediction score. The original algorithm gives up in this case. We circumvent this problem using the following two-step strategy: in this case we select a feature at random and let the algorithm work as usual. Once an explanation has been computed, we follow-up with a "clean-up" step removing features that do not contribute to a change of the prediction in the end. This extra-step can be very computationally expensive if the input data is not sparse, however, it is necessary to ensure that the resulting explanation is minimal.

- The original algorithm skips data items whose prediction outcome never changes. As we use explanations as estimate of which features were involved in the decision making process of the model we assign the explanation of those cases to be the original data item. That is, all features present in the original item make up the explanation as all of them were necessary for the model to compute the predicted label.

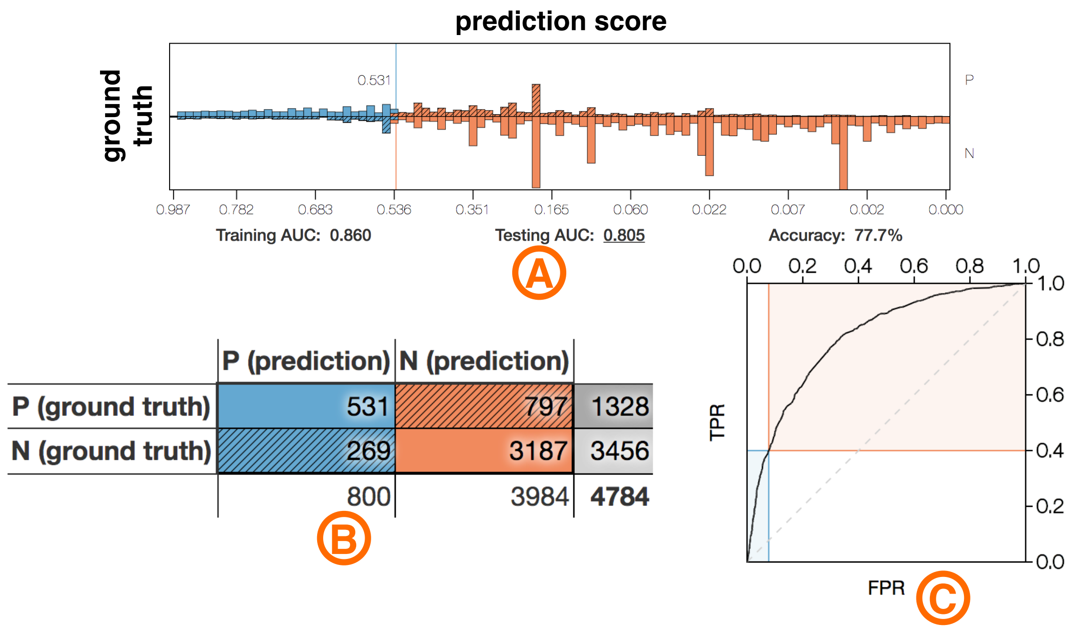
The explanation algorithm can take several hours to compute even for small data sets depending on the sparseness of the data. This requires the generation to be performed offline before analyzing a model. In order to shorten the computation time we utilized caching of partial explanation results in order to reduce the number of queries to the machine learning model.

4.4 Visual Interface

Our proposed user interface[1] consists of three different panels, each corresponding to the different goals of our proposed workflow that we described in Section 4.2. By interacting with each panel and navigating across these panels, experts can diagnose different aspects of model behavior.

In the visualizations that are a part of our interface, the colors orange and blue are used to show negative and positive *prediction* quantities. A hatching pattern is used for quantities where those predictions are *incorrect* according to the ground truth of the data. In this section, we describe each panel according to the order of the workflow: Statistical Summary View of the machine learning model, the Explanation Explorer, and the Item Level Inspector.

[1] https://github.com/nyuvis/explanation_explorer



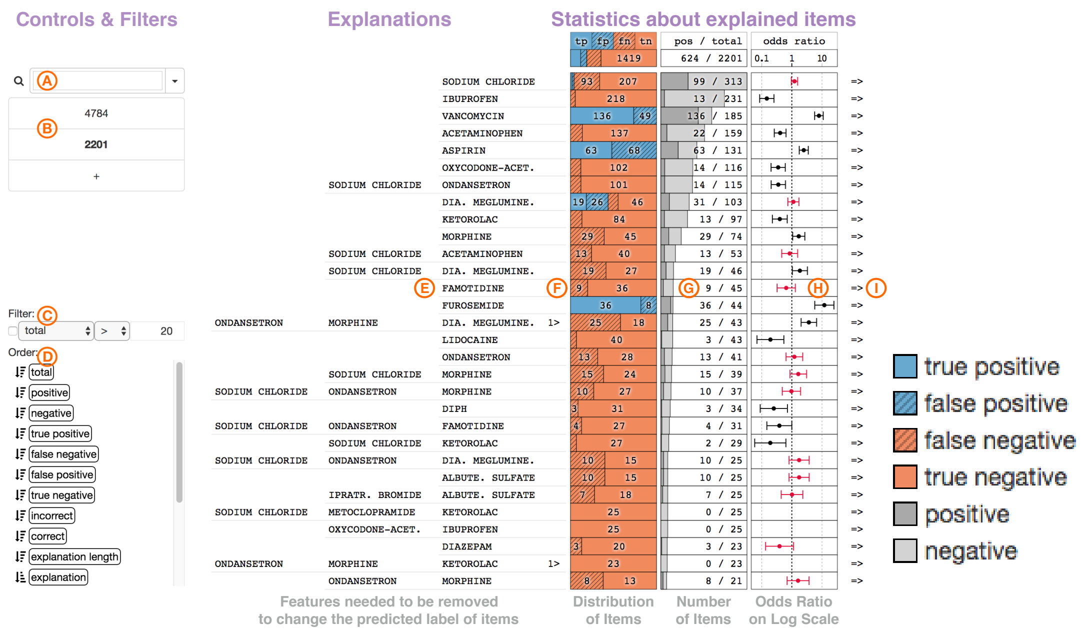
	P (prediction)	N (prediction)	
P (ground truth)	531	797	1328
N (ground truth)	269	3187	3456
	800	3984	**4784**

Figure 4.2: The **Statistical Summary View**. (A) Histograms showing the distribution of prediction scores. The direction of the bars indicates the ground truth and their position relative to the threshold line (at 0.531) indicates the predicted label. (B) The confusion matrix shows the number of correct and incorrect predictions. (C) The ROC curve shows the prediction quality.

4.4.1 Statistical Summary View

The purpose of this panel (Figure 4.2) is to address **G1** by providing a quick summary of the performance of a trained model that can help detect shortcomings before proceeding with further analyses of the model. The view consists of multiple components.

The histograms (Figure 4.2A) show the distribution of data items over prediction scores. The chosen threshold is shown as vertical line. Bars going up indicate the number of predicted positive labels while bars going down show predicted negative labels as emphasized by the color of the bars. The prediction score goes from 1 to 0 from left to right to match the order of cells in the confusion matrix. Likewise, bars at the bottom, left of the threshold, and at the top, right of the threshold, depict



				tp fp fn tn	pos / total	odds ratio	
				1419	624 / 2201	0.1 1 10	
		SODIUM CHLORIDE		93 207	99 / 313		=>
		IBUPROFEN		218	13 / 231		=>
		VANCOMYCIN		136 49	136 / 185		=>
		ACETAMINOPHEN		137	22 / 159		=>
		ASPIRIN		63 68	63 / 131		=>
		OXYCODONE-ACET.		102	14 / 116		=>
	SODIUM CHLORIDE	ONDANSETRON		101	14 / 115		=>
		DIA. MEGLUMINE.		19 26 46	31 / 103		=>
		KETOROLAC		84	13 / 97		=>
		MORPHINE		29 45	29 / 74		=>
	SODIUM CHLORIDE	ACETAMINOPHEN		13 40	13 / 53		=>
	SODIUM CHLORIDE	DIA. MEGLUMINE.		19 27	19 / 46		=>
		FAMOTIDINE		9 36	9 / 45		=>
		FUROSEMIDE		36 8	36 / 44		=>
ONDANSETRON	MORPHINE	DIA. MEGLUMINE.	1>	25 18	25 / 43		=>
		LIDOCAINE		40	3 / 43		=>
		ONDANSETRON		13 28	13 / 41		=>
	SODIUM CHLORIDE	MORPHINE		15 24	15 / 39		=>
SODIUM CHLORIDE	ONDANSETRON	MORPHINE		10 27	10 / 37		=>
		DIPH		3 31	3 / 34		=>
SODIUM CHLORIDE	ONDANSETRON	FAMOTIDINE		4 27	4 / 31		=>
	SODIUM CHLORIDE	KETOROLAC		27	2 / 29		=>
SODIUM CHLORIDE	ONDANSETRON	DIA. MEGLUMINE.		10 15	10 / 25		=>
		ALBUTE. SULFATE		10 15	10 / 25		=>
	IPRATR. BROMIDE	ALBUTE. SULFATE		7 18	7 / 25		=>
SODIUM CHLORIDE	METOCLOPRAMIDE	KETOROLAC		25	0 / 25		=>
	OXYCODONE-ACET.	IBUPROFEN		25	0 / 25		=>
		DIAZEPAM		3 20	3 / 23		=>
ONDANSETRON	MORPHINE	KETOROLAC	1>	23	0 / 23		=>
	ONDANSETRON	MORPHINE		8 13	8 / 21		=>

Features needed to be removed to change the predicted label of items — Distribution of Items — Number of Items — Odds Ratio on Log Scale

Figure 4.3: In the **Explanation Explorer** each row represents a group of data items explained by a set of features (E). An indicator is shown for explanations longer than 3 features. Column (F) shows the distribution of true / false positive / negative data items within the group. Colors show the predicted label ("blue" for positive and "orange" for negative) and a hatching pattern indicates incorrect predictions. Column (G) shows the number of items captured by the explanation. The bars are relative to the size of the largest explanation. Column (H) shows the odds ratio of the group on a logarithmic scale. Whiskers show the confidence interval. The arrows on the right (I) navigate to the Item Level Inspector focusing on the given explanation. The controls of the Explanation Explorer are shown on the left. The first entry of the list of filtered data items (B) represents the full dataset and following entries show sizes after filter steps are applied. The "+" creates a new filter according to the current selection of explanations. Explanations can be selected satisfying a condition (C) or by searching for features in the search box (A). The sort order of explanations is defined by the list at the bottom (D).

incorrectly predicted data items as indicated by their hatching pattern. Selecting a particular bar lets the user navigate to the Explanation Explorer for inspecting items that fall in the given range of prediction scores.

The confusion matrix (Figure 4.2B) splits data items by their ground truth (vertical) and the predicted label (horizontal). The edge of the matrix shows the sums of its columns and rows. The predicted label depends on a threshold that divides prediction scores into positive and negative. We choose the threshold to minimize incorrect predictions (*i.e.*, the threshold with the smallest number of false positive and false negative predictions).

The ROC curve on the testing data (Figure 4.2C) shows the false positive rate ($\frac{FP}{FP+TN}$) plotted against the true positive rate ($\frac{TP}{TP+FN}$). The thresholds for those values are implicit in the plot. However, the position for the chosen optimal threshold (as described above) is indicated in the plot via two crossing lines.

The area under the ROC curve (AUC) is also shown for both the testing and the training data set. An AUC of 1 indicates optimal prediction while an AUC of 0.5 equals classification by flipping a coin. Comparing the training AUC to the test AUC is a good estimator of how well the given model generalizes the training data. A very high training AUC with a much lower test AUC indicates overfitting of the training data. In addition to the AUC the accuracy of the model with the chosen threshold is also shown.

4.4.2 Explanation Explorer

The second panel, the Explanation Explorer (Figure 4.3), addresses **G2** and **G3** by encoding a list of explanations based on the method we described in Section 4.3. The explanations are representative of the main model decisions and the associated statistics about explained items provide insight into the accuracy of those decisions. Each row in the list represents one subgroup of data items explained using a single explanation set. The rows can be filtered based on different criteria for user exploration which we describe below. The row on top shows information for the full set of current data items.

The first column of the list shows this explanation (Figure 4.3E). In order to make this information quickly readable we only show the first three features of an explanation and indicate if there are more features present by adding a marker, showing the number of remaining features, on the right side of the feature names. Furthermore, the feature descriptions are abbreviated in a way that each feature takes up the same amount of space. With this the complexity of an explanation *i.e.*, the number of features used in an explanation, can be seen at a glance. The full description of all features can be seen in the tooltip when hovering over the features. The design decision to show only up to three features stems from the fact that only short explanations can be easily interpreted and having many long explanations is usually a sign of problems with the classifier, like overfitting, and in that case, the actual features involved are less interesting.

The next column shows the relative distribution of predicted labels of the explained subset of data items as stacked bars (Figure 4.3F). The colors blue and orange are used to indicate a positive and negative prediction respectively while a hatching pattern indicates incorrect predictions. The actual numbers are shown in the bars as well.

The bars in the third column show the size of the subset relative to the largest explanation subset of the current data items (Figure 4.3G). The bars are split according to the distribution of the ground truth labels. Two shades of gray are used to avoid confusion with distributions of predicted labels. The total number of items in the subset along with the number of positive items according to the ground truth is written in the column as well.

The fourth column shows the odds ratio of the subset on a logarithmic scale (Figure 4.3H). Whiskers indicate its confidence interval. Odds ratio is a popular metric for determining effectiveness in evidence based medicine and clinical trials. It is computed by comparing the subset explained by the given explanation with the full set of current data items. This way we can detect whether an explanation describes a consistent subset of instances or if the subset appears like a random sample. With this the odds ratio is:

$$\frac{p_e/n_e}{p_t/n_t}$$

where p_e and n_e is the ratio of positive and negative items respectively in the explanation subset and p_t and n_t is the ratio of those items in the remaining data set. The confidence interval of the odds ratio is then computed as:

$$\exp\left(\log\left(\text{odds ratio}\right)\ \pm\ 1.96\sqrt{P_e^{-1} + N_e^{-1} + P_t^{-1} + N_t^{-1}}\right)$$

where P and N are the actual number of positive and negative items in the explanation subset e and the remaining data t.

An odds ratio larger than one indicates that the explained subset is significantly positive with respect to the rest of the current data items. Likewise, a value smaller than one indicates that it is significantly negative. However, if the confidence interval crosses one the subset is not significantly different. To highlight this important special case the odds ratio and the whiskers are drawn in red in this case.

At the right end of each row is a button (Figure 4.3I) to inspect the explained subset more closely in the Item Level Inspector as described in Section 4.4.3.

The rows shown by the Explanation Explorer can be reordered as well as filtered. As shown in Figure 4.3, the panel features various controls on the left hand side to accomplish those operations . Filtering works by first selecting affected rows (either by clicking on a row or by using widgets on the left) and then clicking on the "+" in the list of filtered data items (Figure 4.3B) Each new filtering of data items creates a new entry in this list showing the current number of data items. By selecting entries higher up in the list the user can go back to this filter. The topmost entry always contains all data items of the entire data set.

Besides getting a filter for a given prediction score range from the Statistical Summary View there are two ways of filtering items: searching and conditioning. The search field (Figure 4.3A) can be used to select rows whose explanation matches the query specified by the user. While typing or using arrow keys suggestions for feature names are shown in a dropdown list. Those suggestions are sorted by how often that feature appears in explanations and how closely it matches the already specified query. The query can contain multiple features that need to appear in the explanation separated by a comma ",". The conditioning widget (Figure 4.3C) allows to filter by quantities.

As shown in Figure 4.3D, different metrics can be used to filter or reorder the list of explanations. A good use for the conditioning filter is to remove explanations that only explain a small subset of the data when looking for unusual or significant subsets. The explanation rows can also be reordered using these metrics. The widget contains a list that shows the order in which explanations get sorted. Each element has a symbol next to it indicating the sort direction which can be clicked on to change the sort direction. Selecting an element brings it to the top of the list.

The metrics used for reordering are the same as those used for conditioning with the additional option of lexicographical sorting by using the feature names of the explanations. Common metrics to use for sorting or reordering, besides total amount of items, are "uncertainty" and "odds ratio". "Uncertainty" (the closeness of the odds ratio to one: $-|\log(OR)|$) provides a view into problematic areas of the machine learning model sometimes even unpredictable items when items with the same value configuration have different ground truth labels. "Odds ratio", based on the computation mentioned earlier, points to especially strong predictive areas of the machine learning model.

4.4.3 Item Level Inspector

The third panel (Figure 4.4) allows for a more granular inspection of items explained by a given explanation set. This addresses **G4** by providing hints about the extent to which a model can be improved and if changing the data is necessary for that purpose. The panel consists of a matrix showing the actual feature vectors of the given items. Each row represents a unique feature vector pattern while columns represent features. Rows can be expanded so that each row represents exactly one item. Cells in the matrix are filled if the corresponding feature vector contains the feature represented by the column. As rows are aggregates of multiple data items, the number of items is shown as a bar with the number indicated on the left side of the matrix. The feature names for the columns are shown slanted on top of the matrix. Bars behind the names show how often the feature is present. The very first column in the matrix shows the predicted label (using the colors blue and orange) and its correctness (hatching pattern for incorrect predictions) of the given data item. Items with the same feature vector configuration but different labels are shown in different rows.

Rows and columns can be reordered using different options, similar to Explanation Explorer. One of the important reordering criteria for rows is the **feature order**, where items are ordered by seeing whether the first feature of the columns is present and then if the next feature is present, and is repeated for all the columns. An important reordering criteria for the columns is the **relative feature importance**: the gini feature importance with respect to the current subset of data items and their predicted labels and correctness.

The combination of the “feature order” and the “relative feature importance” criteria provide a particularly interesting view on the subset of data items. Using this order, the most discriminating features with respect to predicted and actual labels are shown first. Since the rows are ordered by those features, a user can follow those orderings to see how to separate different predicted and actual labels. This guidance of the user to relevant associations in the item subset are useful for quickly understanding the raw data. Note that it is sometimes possible to fully separate data items this way. However, utilizing this separation would be overfitting on the validation set. Furthermore, the opposite situation with exact same feature vectors but with different labels that cannot be separated exists as well.

Some features are not discriminative in terms of “relative feature importance”. They can be ignored to simplify the matrix view.

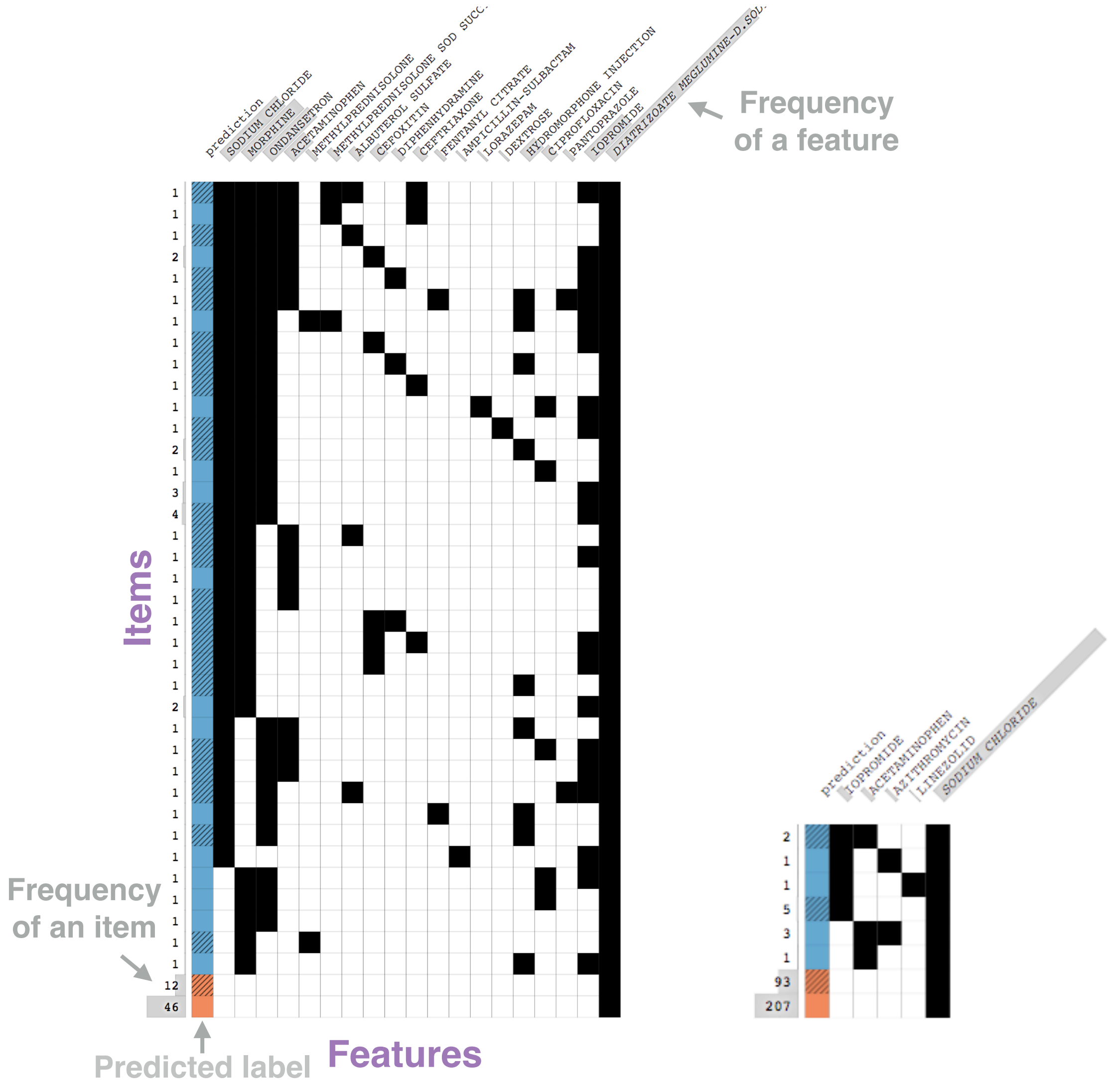
4.5 Case Study: Analysis of Medical Outcomes

The proposed workflow described above stem from a one year collaboration with a machine learning expert and a medical doctor from the *NYU Langone Medical Center*, both co-authors of this paper. The medical machine learning team at the medical center works in tight collaboration with doctors and hospital management to derive novel methods to automate medical procedures, provide diagnostics support, improve efficiency and gain novel insights on medical procedures and processes.

Domain Problem Description. Our collaboration focused on the analysis and improvement of models built to optimize processing times in the emergency department of the hospital. The crucial decision here is whether a patient coming to the emergency room will end up being admitted to the hospital or sent home. In the case of a patient being admitted to the hospital, a bed has to be prepared for the patient which results in a 2-hour waiting period where the patient occupies a bed in the emergency room preventing other patients from being processed. If the waiting time can be reduced by knowing early if a given patient will be admitted, the throughput of the emergency room can be increased.

The idea to reduce this wait time is to use predictive modeling at the earliest time possible so that an admitted patient can be moved sooner. The amount of data available to make this decision however is very limited. When first presented with a patient the emergency doctor orders medication for treating, stabilizing, or preparing the patient for procedures or tests and eventually will conclude a diagnosis and decide whether the patient is in need of admission.

As medication is the earliest recorded indicator of the admission result and also is recorded before lengthy procedures or tests it is the most promising candidate for a predictive model. The main machine learning task is therefore to verify whether



(a) Inspection of "Diatrizoate Meglumine" and (b) "Sodium Chloride".

Figure 4.4: The **Item Level Inspector** showing a matrix of data items as rows and features as columns for the explanations *Diatrizoate Meglumine* and *Sodium Chloride* in the initial data set of the case study (Section 4.5). Rows group identical instances together and show the count on the left side. Features are sorted by "relative feature importance" showing from left to right how labels can be separated.

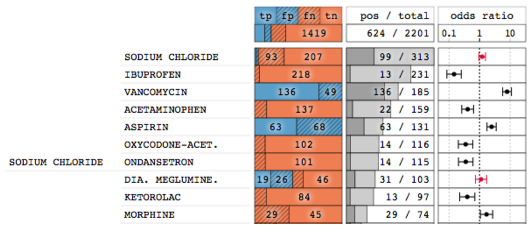
a viable model can be built by using exclusively the limited information available.

Other work has been done in this regard, however, using input features that are not readily available (*e.g.*, information from medical notes that are written after the fact) or are hospital specific (*e.g.*, mode of arrival, triage score) [18, 23, 46].

During our collaboration the team of visualization experts met with the medical team regularly to understand the problem and the data, and to develop collaboratively visual analytics solutions for model diagnostics and interpretation. The workflow we described in the paper resulted from numerous iterations over the methods used to derive information from the model and the methods used to enable their interactive visual exploration.

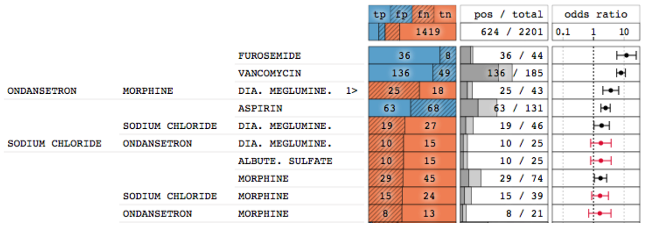
In this section, we describe one particular example that showcases the capabilities of the proposed method and provides insights on how it is able to support diagnostic analysis of complex machine learning model used in a relevant real-world scenario. In the following, the term "we" is used to refer to the team of visual analytic experts, a machine learning expert, and a medical doctor, who collaboratively worked on the usage scenarios described below.

Selecting Initial Data and Model (G1). We initially gathered a dataset of 5980 patients (28% admitted) with binary vectors indicating medications given to the patient. Those patients were randomly split into a training (1196 patients with 30% admitted) and test (4784 patients with 27% admitted) dataset. We then computed several models and tweaked them using mostly the Statistical Summary View and model specific approaches. This initial dataset contains 398 unique medications. The table below shows a summary of the models we trained and their performance.



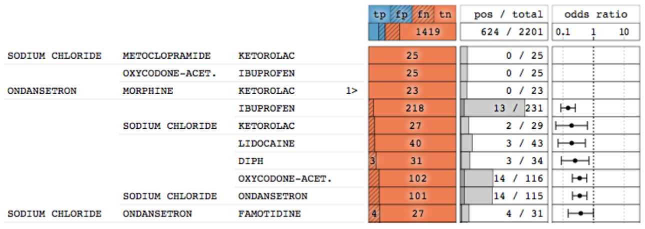
				tp fp fn tn	pos / total	odds ratio
				1419	624 / 2201	0.1 1 10
		SODIUM CHLORIDE		93 207	99 / 313	
		IBUPROFEN		218	13 / 231	
		VANCOMYCIN		136 49	136 / 185	
		ACETAMINOPHEN		137	22 / 159	
		ASPIRIN		63 68	63 / 131	
		OXYCODONE-ACET.		102	14 / 116	
	SODIUM CHLORIDE	ONDANSETRON		101	14 / 115	
		DIA. MEGLUMINE.		19 26 46	31 / 103	
		KETOROLAC		84	13 / 97	
		MORPHINE		29 45	29 / 74	

(a) Ordered by "total" size showing the most common explanations.



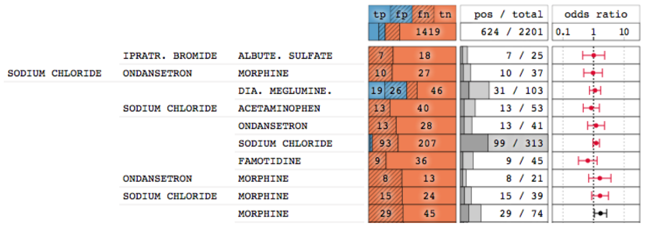
				tp fp fn tn	pos / total	odds ratio
				1419	624 / 2201	0.1 1 10
		FUROSEMIDE		36 8	36 / 44	
		VANCOMYCIN		136 49	136 / 185	
ONDANSETRON	MORPHINE	DIA. MEGLUMINE.	1>	25 18	25 / 43	
		ASPIRIN		63 68	63 / 131	
	SODIUM CHLORIDE	DIA. MEGLUMINE.		19 27	19 / 46	
SODIUM CHLORIDE	ONDANSETRON	DIA. MEGLUMINE.		10 15	10 / 25	
		ALBUTE. SULFATE		10 15	10 / 25	
		MORPHINE		29 45	29 / 74	
	SODIUM CHLORIDE	MORPHINE		15 24	15 / 39	
	ONDANSETRON	MORPHINE		8 13	8 / 21	

(b) Ordered by "odds ratio" showing significantly positive explanations.



				tp fp fn tn	pos / total	odds ratio
				1419	624 / 2201	0.1 1 10
SODIUM CHLORIDE	METOCLOPRAMIDE	KETOROLAC		25	0 / 25	
	OXYCODONE-ACET.	IBUPROFEN		25	0 / 25	
ONDANSETRON	MORPHINE	KETOROLAC	1>	23	0 / 23	
		IBUPROFEN		218	13 / 231	
	SODIUM CHLORIDE	KETOROLAC		27	2 / 29	
		LIDOCAINE		40	3 / 43	
		DIPH		3 31	3 / 34	
		OXYCODONE-ACET.		102	14 / 116	
	SODIUM CHLORIDE	ONDANSETRON		101	14 / 115	
SODIUM CHLORIDE	ONDANSETRON	FAMOTIDINE		4 27	4 / 31	

(c) Ordered by reverse "odds ratio" showing significantly negative explanations.



			tp fp fn tn	pos / total	odds ratio
			1419	624 / 2201	0.1 1 10
	IPRATR. BROMIDE	ALBUTE. SULFATE	7 18	7 / 25	
SODIUM CHLORIDE	ONDANSETRON	MORPHINE	10 27	10 / 37	
		DIA. MEGLUMINE.	19 26 46	31 / 103	
	SODIUM CHLORIDE	ACETAMINOPHEN	13 40	13 / 53	
		ONDANSETRON	13 28	13 / 41	
		SODIUM CHLORIDE	93 207	99 / 313	
		FAMOTIDINE	9 36	9 / 45	
	ONDANSETRON	MORPHINE	8 13	8 / 21	
	SODIUM CHLORIDE	MORPHINE	15 24	15 / 39	
		MORPHINE	29 45	29 / 74	

(d) Ordered by "uncertainty" showing item subsets whose predictions are not significant.

Figure 4.5: Showing different orders in the **Explanation Explorer** for addressing the goals (G2 & G3) in the case study (Section 4.5). The initial dataset is filtered for explanations with > 20 data items.

Model	Training	Test AUC
Gaussian Naïve Bayes (GNB) [133]	0.58	0.52
Logistic Regression (LR) [132]	0.85	0.79
Random Forest (RF) [20]	0.88	0.79
Multi Layer Perceptron (MLP) [49]	0.85	**0.80**

As we can see most of the models achieve similar performance on the test data. In the following we focus exclusively on the *Multi Layer Perceptron* model but the same kind of analysis can be performed on any of the other models with similar results for the models with similar predictive power.

Exploring model decisions and spotting problems (G2 & G3). To start the analysis we compute all the explanations and visualize them in the Explanation Explorer shown in Figure 4.5a, which by default is sorted by frequency of explanations. The first thing we notice is that *Sodium Chloride* is the most common explanation and that it contains a considerable number of misclassified instances.

Sodium Chloride represents an intravenous therapy, the infusion of a liquid directly into a vein. As part of a medication order it is used to increase the effectiveness and response time of a drug and also to apply medication if a patient is unconscious. Used by itself it has the only purpose of hydrating a patient.

The distribution for the explanation shows both positive and negative predicted outcomes, which may seem paradoxical at first. This result however stems from the fact that the context of an explanation (that is, whether features co-occur with the features used in the explanation; note that certain co-occurring features form other explanations as they have a direct influence on the outcome) matters in terms of which outcome it explains.

The Item Level Inspector can help us clarify this situation. We can see that hospital admission is the predicted outcome when *Sodium Chloride* appears together with other drugs, whereas when this is the only medication the patient received, the patient is predicted to get sent home (Figure 4.4b).

Looking at the odds ratio value for this explanation we also notice that this subset is not significantly predictive and that the misclassification rate is high (weak signal). Note that even though *Sodium Chloride* is the most common explanation it cannot be used as a significant indicator of the outcome. From a medical perspective this makes sense as *Sodium Chloride* is mostly used as supporting medication, however, the machine learning model still assigned predictive power to it. This indicates that the data did not contain a strong enough signal to make a more informed decision in those cases.

Another common explanation is *Ibuprofen* a pain relieving drug. It is predictive for non-admissions which is likely due to patients with pain symptoms that turned out to be benign. The odds ratio indicates a significant relation to the outcome. On the other hand *Vancomycin*, an antibiotic used for treating infections, is significantly linked to hospital admission which is expected.

After filtering out uncommon explanations ($<$ 20 explained items) ordering the explanations by "odds ratio" reveals significant indicators for admission and non-admission (Figure 4.5b). In addition to the already discovered significant explanations we can see *Furosemide*, a drug for treating congestive heart failure, as being strongly indicative for admission and certain drugs in combination with *Sodium Chloride* strongly linked to non-admission (Figure 4.5c). The drugs in question are pain-relievers (*Morphine* and *Ketorolac*) and drugs to help with stomach problems (*Ondansetron* and *Metoclopramide*). Note that using an IV (*Sodium Chloride*) for stomach related problems helps both hydrate the patient and ensures the intake of the medication (after *e.g.*, vomiting).

Finding Weaknesses (G4). Ordering explanations by “uncertainty” (Figure 4.5d) shows explanations whose predictions are not significant. This is often the case when it is impossible to correctly predict a set of identical instances that have a contradicting ground truth.

The first two explanations *Ipratropium Bromide*, *Albuterol Sulfate* (medication for treating chronic obstructive pulmonary disease and asthma, lung diseases that can have chronic and acute symptoms the latter of which requires immediate attention) and *Sodium Chloride*, *Ondansetron*, *Morphine* are both predicted negative. However, the ground truth of those subset has the same distribution as the overall dataset (thus an odds ratio close to 1). This means the true admission rate of those two subsets is independent of the medication in question as the admission rate matches the admission rate of the dataset. If more patients would be observed in the data this rate would likely stay the same. Through Item Level Inspector we can see that the features of the explanations are the only features in the respective data items. No further information is provided that could help swaying those subsets in a definite direction of admission or non-admission.

Another problematic drug is *Diatrizoate Meglumine* which has a high misclassification rate and an odds ratio close to 1. The drug is a contrast medium that is given in preparation of PET (positron emission tomography) or CT (computerized tomography) scans. As the outcome of the scan is not known it cannot be determined whether the test was positive for the hypothesis made by the attending physician. Furthermore, even the presence of other drugs is no indicator for admission as it only shows the doctor’s risk assessment *before* the test was ordered and therefore does not include whether the doctor’s assumption was correct. Note, that Figure 4.4a shows how outcomes can be better separated using available features. However, doing so would result in overfitting on the validation data set which should be avoided in any case.

Faced with this revelation we explored how we could provide more information to reduce those ambiguities. In order to properly deal with cases like *Ipratropium Bromide* and *Albuterol Sulfate* or *Sodium Chloride* and *Diatrizoate Meglumine* more information is needed. Through domain expertise we can reason about the underlying shortcomings of the current dataset, *e.g.*, the nature of the limitations of *Diatrizoate Meglumin*. In order to overcome those limitations we need to include additional information in our dataset. For example, including information about the final diagnosis of a patient resolves the ambiguities of patients explained by *Diatrizoate Meglumin* and other problematic explanations mentioned above, and likely improves the overall quality of the prediction[2]. However, this also moves the time of the prediction closer to the point in time when the actual decision, whether the patient is admitted to the hospital, is made thus reducing the time-gain for preparing a bed in case of admission.

Changing Data and Model. In the following we describe how we could improve the prediction task by including additional information to our dataset. This additional information, *i.e.* final diagnoses, was added to overcome limitations posed by medications not strongly linked to an outcome, as described above. In order to include those diagnosis features in the data we had to capture new data which also allowed for capturing a bigger dataset. The new dataset contains 154580 patients (20% admitted) and was randomly split into a training (30916 patients with 20% admitted) and test (123664 patients with 20% admitted) dataset. It

[2]Including other information, such as, mode of arrival, gender, or age, might improve accuracy but would not solve the issues mentioned above.

contains 1709 unique medications and 15422 unique diagnoses.

The best results of different models on the new dataset are:

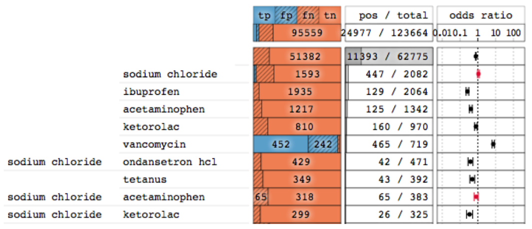
	no diagnoses		incl. diagnoses	
Model	Training	Test AUC	Training	Test AUC
GNB	0.51	0.49	0.75	0.66
LR	0.71	0.67	0.93	0.88
RF	0.69	0.68	0.98	0.83
MLP	0.71	**0.68**	0.95	**0.88**

Maximum values are chosen using digits not shown.

Again we are focusing solely on the Multi Layer Perceptron model for further analyses (even though similar results can be found with the other equally well performing models). In order to compare our new data to the previous dataset we first created models that do not utilize the newly added diagnoses. However, the resulting AUC is much lower than for the initial data. Looking at the Statistical Summary View reveals a strong concentration of data points at a specific prediction score. Focusing on this prediction score in the Explanation Explorer (Figure 4.6a) shows that it corresponds to the 62776 patients that did not receive any medication at all. This configuration predicts non-admission as it is more likely to get sent home when not receiving any medication. The unusual large number of such cases ($\sim 50\%$), however, hints at a possible capturing error which would also explain the 11394 cases where patients were admitted. This failure rate severely affects the machine learning models. For comparison the next largest explanation of *Ibuprofen* in the new dataset consists only of 2011 patients. In fact patients without medication were not captured in the original dataset and removing them from the new dataset increases the best train / test AUC to 0.83 / 0.80 similar to the original dataset. For further analysis we include patients without medication. Utilizing diagnoses in the models strongly increase the possible AUC.

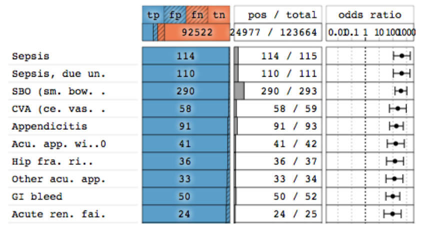
How Did Diagnoses Features Change the Model? The Explanation Explorer of the best model utilizing diagnoses features, Multi Layer Perceptron, can be seen in Figure 4.6b. Noticeably, almost all explanations now consist of diagnoses. This also means that medication features have now become almost irrelevant except for medications, like *Ibuprofen* and *Vancomycin*, that were strong indicators before. The most significant diagnoses, using odds ratio, for admission are *Sepsis*, *Sepsis due to unidentified organism*, and *Small Bowel Obstruction*. Diagnoses that require antibiotics (*e.g.*, *Vancomycin*) and pain medication (*e.g.*, *Ibuprofen*) respectively. Contradictory or insignificant medications, like *Diatrizoate Meglumine* or *Ipratropium Bromide* and *Albuterol Sulfate*, do not show up anymore as they can be more effectively replaced by their diagnoses. The largest explanation, with 2619 patients, is *Unspecified* which, after some research, turns out to be due to a policy change before which doctors were allowed to omit a diagnosis if the patient got admitted to the hospital. Why only 2105 (∼80%) were actually admitted to the hospital remains unclear.

Diagnostic Insights. By adding diagnoses to the dataset a strong increase in predictive quality was achieved. However, seeing that diagnoses effectively replace medication in their predictive power suggests that the “labels are leaking”. That is, since doctors make the decision of whether to admit a patient at the time of the final diagnosis there is a strong correlation between the label and the features. This is an undesired effect as the model is not predicting the outcome anymore but merely building an approximate lookup table for diagnosis admission rates. If the model would have kept using medication and only consulted diagnoses for ambiguous cases



		tp fp fn tn	pos / total	odds ratio
		95559	24977 / 123664	0.01 0.1 1 10 100
		51382	11393 / 62775	
	sodium chloride	1593	447 / 2082	
	ibuprofen	1935	129 / 2064	
	acetaminophen	1217	125 / 1342	
	ketorolac	810	160 / 970	
	vancomycin	452 242	465 / 719	
sodium chloride	ondansetron hcl	429	42 / 471	
	tetanus	349	43 / 392	
sodium chloride	acetaminophen	65 318	65 / 383	
sodium chloride	ketorolac	299	26 / 325	

(a) The second dataset without diagnoses ordered by “total” size.



	tp fp fn tn	pos / total	odds ratio
	92522	24977 / 123664	0.01 0.1 1 10 100 1000
Sepsis	114	114 / 115	
Sepsis, due un.	110	110 / 111	
SBO (sm. bow. .	290	290 / 293	
CVA (ce. vas. .	58	58 / 59	
Appendicitis	91	91 / 93	
Acu. app. wi..0	41	41 / 42	
Hip fra. ri..	36	36 / 37	
Other acu. app.	33	33 / 34	
GI bleed	50	50 / 52	
Acute ren. fai.	24	24 / 25	

(b) The second dataset using diagnoses ordered by “odds ratio”.

Figure 4.6: Showing the second dataset of the case study (Section 4.5) with and without using diagnoses features in the **Explanation Explorer**.

the usability of the model would have been improved due to diagnoses. This is not the case. Despite its lower objective quality the model using only medications as input emerged as the more practically useful model. Since experts know about the strengths and weaknesses of the model, they can distinguish between confident and ambiguous cases early and decide whether to accept the prediction or wait for the final decision made by the doctor. This demonstrates that a statistically weaker model can be more useful in practice.

4.6 Discussion

Through our case study of patient visits, we have shown that by aggregating model decisions through explanations, we are able to make sense of a large number of interesting decisions: some expected and some unexpected; some useful and some less useful; and finally some leading to actionable knowledge and some requiring more introspection on the part of domain experts. This level of transparency is necessary for experts and data scientists to built trust in a model and, especially, generate ideas on how it can be improved.

In our interactions we have also noticed the usefulness of using explanations as the main method to make sense of model decisions. As long as the features used for the problem can be interpreted by the user, the concepts expressed in the visualization are easy to grasp and learn. During our collaboration we have experimented with other structures such as trees and rules but we often found that these were either too complicated or hard to use for modeling complex phenomena reliably and succinctly.

As we observed in the case study presented in Section 4.5 it is important to understand which decisions a model is most certain about and also find the decisions about which it is uncertain. When issues are detected there are several possibilities: training a better model, finding better data, introducing new and more informative features, or deciding that the model can make decisions only for the subset of cases the experts are most certain about. One possible outcome is also deciding that the problem is simply too complex and that expert judgment is, at the current stage, preferable.

From the experience we gained in this project we drew a number of important lessons, which we outline below.

Lessons Learned. In our work we noticed that many of the issues we spot in our analysis cannot be corrected simply by training a better model with the same data, but need some major redesign of the feature space and a careful analysis of the biases contained in the data. In turn, while diagnosing one or more models built on one data set and set of features can bring useful knowledge, ultimately solutions often have to come from better data engineering. We believe visual analytics can and should play a major role in this regards and find ways to support analysts explore alternative data and feature spaces. This is even more relevant when we observe that visual analytics systems and research tends to focus on one single data set and one single set of features. Focusing on supporting external changes of data and models offers many challenges and opportunities for visual analytics.

Another important observation pertains to the practical value of developing a visual validation system separated from and not interfering with the existing modeling pipeline. From Figure 4.1 it may seem natural to envision visual analytics methods able to support the user in closing the loop and apply direct modifications to the model in order to improve it. This is the type of solution advocated by the *interactive machine-learning* paradigm [4], in which the user can directly instruct the model on how to improve its decisions.

However, through our collaboration, we realized that modelers and experts often have very specific tools they use for model development and refinement and it is often hard to intervene on their familiar processes and infrastructure. A much more viable solution is to develop a methodology that does not require a substantial modification of their existing workflow and infrastructure.

We also observe that while this type of paradigm is useful to provide better examples to the model, it cannot solve the data acquisition shortcomings we have outlined above. Fixing these problems requires domain experts to rethink the whole approach of the stated machine learning problem. For example, improving the input data might require to capture new features from different sources or rethinking of pre-processing steps. It seems important to figure out in future research which particular settings are the most appropriate for the “out of the loop” solution we proposed here and which are more amenable to the interactive machine learning paradigm.

A final observation is how the process of validating the model often leads to generating insights that pertain more to the reality being modelled than the model itself. In several occasion, our collaborators ended up spotting potential issues with how their patients are handled in the hospital. Typical examples include situations in which some patients are discharged and at the same time are given medications that represent a strong signal for a serious condition for the doctor. These kind of mismatches between the mental model of the doctor and the reality modeled is a potential source of process improvement and can be used to take important actions.

In relation to this last observation, it seems interesting to reflect on how visual analytics can further leverage the power of modeling for exploratory data analysis and data sense making. While many systems focus on direct visualization of raw data as overview, there seem to be relevant opportunities on using modeling as a preparatory step so that the resulting visualization contains more signals about hidden associations among features and items in the data.

Limitations. The workflow and its implementation we described work exclusively with sparse binary data and binary classification. Although, explanation generation can be extended to other input data types the visual representation of those explanations has to be redesigned in order to accommodate other data types. Similarly, handling classification for more than two classes is also not trivial.

Our method works only with interpretable features, that is, features have a direct connection to a reality the user can easily understand. Many relevant machine learning problems however require the use of highly non-interpretable features. Classification of images, audio, and video, is a classic example of this case. In these settings the single features used by the model do not have any direct interpretation the user can directly use for model understanding.

Our solution works best with analyzing one single model at a time but it does not provide direct support for *model comparison*. In many practical cases modelers like to train multiple models and then figure out how they compare. While in practice most of these comparisons are currently performed on statistical aggregations, it would be useful to develop methods able to compare multiple models in terms of the *decisions* they make and how they differ. This is even more important in those cases in which models display a similar performance but actually differ in the way the decisions they make.

Merging same explanations with different outcomes, like in the case of *Sodium Chloride*, was done to make a user aware of this case. However, merging penalizes the odds ratio. In the cases presented in this paper the odds ratio did not get affected as the correctness for both outcomes were similar. If, for example, the positive prediction were always right but the negative prediction equivalent to a random guess both cases would be underrepresented by the odds ratio.

With respect to scalability, neither the total number of features nor the total number of instances is limiting, since only a subset of available features appear in explanations and many instances are aggregated. However, it can happen that explanations are consistently long or do not aggregate well. This is mostly dependent on the model. Long explanations can be a sign of overfitting or a highly complex model with few similar instances. Explanations in the latter case are less interpretable which demands for a strategy to simplify or shorten explanations.

4.7 Conclusion & Future Work

We demonstrated how visual explanations can be effectively leveraged by data scientists and medical experts for diagnosing model decisions and for ultimately making informed judgment about associations among medications and patients' diagnoses. We will extend our method to non-binary data and multi-class problems. We will also extend our solution for letting data scientists compare explanations from multiple models and leverage our model-agnostic workflow for making informed choices about choosing machine learning models in real-world application scenarios.

We thank Prof. Foster Provost for his help in understanding and using his instance-level explanation technique. The research described in this paper is part of the Analysis in Motion Initiative at Pacific Northwest National Laboratory (PNNL). It was conducted under the Laboratory Directed Research and Development Program at PNNL, a multi-program national laboratory operated by Battelle. Battelle operates PNNL for the U.S. Department of Energy (DOE) under contract DE-AC05-76RLO01830. The work has also been partially funded by the Google Faculty Research Award "Interactive Visual Explanation of Classification Models".

4.8 General Discussion

The presented workflow [65] shows how instance-level explanations can be leveraged to get insights into a model's decision making process without overwhelming an analyst with the quantity of such individual explanations. We demonstrated the feasibility of the workflow using an example with exclusively binary features. While many problems can be expressed this way it is a limitation to the presented implementation of the proposed workflow. Furthermore, we evaluated the effectiveness of the workflow on the insights gained through it. Those problems illustrate the need for a more generalizable implementation and a formal study on the effectiveness of the workflow whose results are described in the next Chapter.

Chapter 5

User Study on Aggregating Instance-level Explanations

Recently, there is growing consensus of the critical need to have better techniques to explain machine learning models. However, many of the popular techniques are instance-level explanations, which explain the model from the point of view of a single data point. While local explanations may be misleading, they are also not human-scale, as it is impossible for users to read explanations for how the model behaves on all of their data points. Our paper explores the effectiveness of providing instance-level explanations in aggregate, by demonstrating that such aggregated explanations have a significant impact on users' ability to detect biases in data. This is achieved by comparing meaningful subsets, such as differences between ground truth labels, predicted labels, and correct and incorrect predictions, which provide necessary navigation to explain machine learning models.

Summary:

Research Question:

What is the impact of aggregating instance-level explanations?

Key Findings:

- Histograms can be used to aggregate instance-level explanations and compare subsets effectively.

- Aggregating instance-level explanations significantly outperforms inspecting individual explanations or unassisted aggregation for detecting biases in the data.

- Inspecting individual instance-level explanations can be misleading and hurts detecting biases in the data.

- Aggregated instance-level explanations allow to detect biases equally well as in a table without explanations while being more scalable.

Josua Krause, Adam Perer, Enrico Bertini

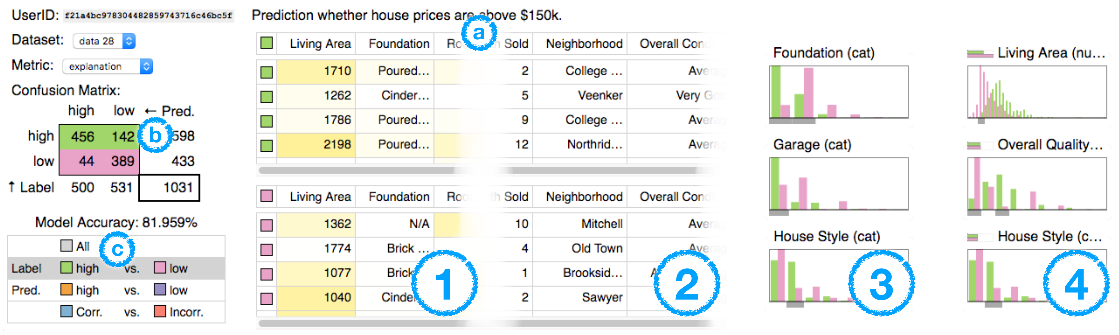
5.1 Introduction

As data continues to increase in complexity and scale, data scientists are increasingly turning to machine learning to automatically make decisions. However, when these decisions are applied to high-stakes domains such as medicine, law enforcement, and financial lending, it is critical for humans to understand the basis for these decisions.

Predictive modeling is an area of supervised machine learning which aims to predict outcomes from data. Such models are trained on examples with a known ground truth. In order to verify that a model generalizes well to unseen data, a hold-out data set with known ground truth is typically used to test the model after training. This allows to detect problems with the model, such as over-fitting on the training data, *i.e.*, the model learned a phenomenon that is only present in the training data, by measuring the gap in the accuracy between the training and the testing data. However, sometimes a bias in the collected data affects both the training and the test data which makes it impossible to detect through accuracy alone. A human understanding of the underlying data is needed.

For example, Caruana *et al.* [25] built an interpretable machine learning model to analyze mortality risk in patients diagnosed with Pneumonia. After analyzing the model's behavior, Caruana *et al.* detected that patients that additionally suffered from Asthma had a significantly lower mortality risk, according to the *model* and supported by the data. However, this finding goes against current medical knowledge, as the combination of Pneumonia and Asthma are associated with a significantly increased mortality risk. In fact, the data was biased because these high-risk patients with Asthma were given special attention during their hospital visits which contributed to their lower mortality. The presence of Asthma was not responsible for their improvement in health, but rather a systematic bias.

Using the interpretable model and human expert knowledge, it was possible to detect this systematic bias in the data before deploying the model. However, using interpretable machine learning algorithms typically penalizes their capacity, thus lowering the potential accuracy of the model [25] or is only superficially more interpretable by being interpretable on a small scale but not for more complex



UserID: f21a4bc9783044828597437l6c46bc5f

Dataset: data 28

Metric: explanation

Confusion Matrix:

	high	low	← Pred.
high	456	142	598
low	44	389	433
↑ Label	500	531	1031

Model Accuracy: 81.959%

	☐ All		
Label	🟩 high	vs.	🟪 low
Pred.	🟧 high	vs.	🟪 low
	🟦 Corr.	vs.	🟥 Incorr.

Prediction whether house prices are above $150k.

🟩	Living Area	Foundation	Ro	Sold	Neighborhood	Overall Con
🟩	1710	Poured…		2	College …	Ave
🟩	1262	Cinder…		5	Veenker	Very G
🟩	1786	Poured…		9	College …	Ave
🟩	2198	Poured…		12	Northrid…	Ave

🟪	Living Area	Foundation	Ro	Sold	Neighborhood	Overall Con
🟪	1362	N/A		10	Mitchell	Ave
🟪	1774	Brick …		4	Old Town	Ave
🟪	1077	Brick		1	Brooksid…	A
🟪	1040	Cinde		2	Sawyer	

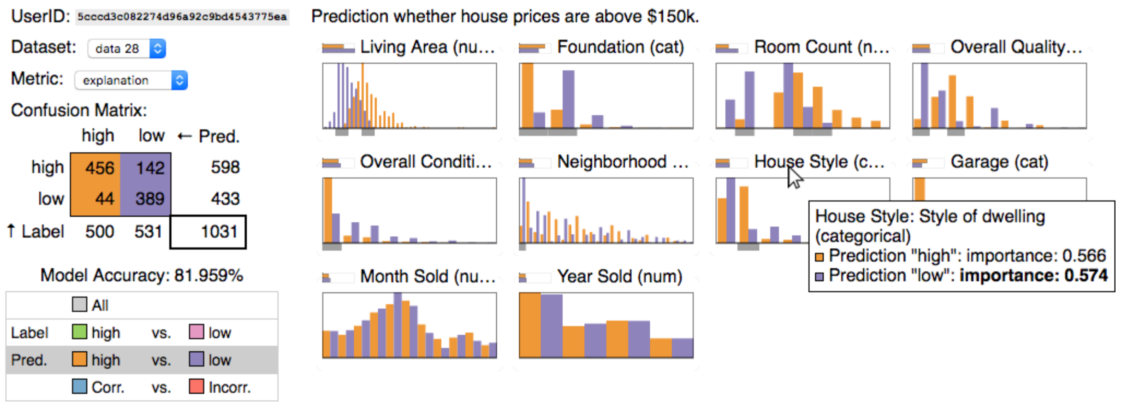
Figure 5.1: Showing the four conditions of our study: (1) instance-level explanations; (2) only instances; (3) only aggregated features; (4) aggregated features with explanations. On the left and the top are consistent parts of the user interface showing: (a) the problem description; (b) the confusion matrix; (c) the subset selector.

tasks ([68, 76]). As a way to interpret the behavior of machine learning models *independently* from the used algorithm, black-box and more precisely, instance-level explanations recently became popular [69, 83, 102].

However, such explanations are commonly reviewed by experts one-at-a-time. This task becomes infeasible when dealing with thousands or more instances, also typical of real-world datasets. To that extent, we propose a visual way of reviewing instance level explanations with the help of aggregation in combination with navigation. This is implemented through the comparison of subsets of the test data under different conditions.

We conducted a study comparing aggregated instance-level explanations to their individual counterparts. Under both conditions different subsets of the test data could be compared by participants. By providing models with both biased and unbiased data we were able to measure the trust of participants in the decisions made by the models and their ability to detect flaws in the underlying data for both methods.

Concretely, our contributions include a method for effectively comparing subset


UserID: 5cccd3c082274d96a92c9bd4543775ea

Dataset: data 28

Metric: explanation

Confusion Matrix:

	high	low	← Pred.
high	456	142	598
low	44	389	433
↑ Label	500	531	1031

Model Accuracy: 81.959%

	⬜ All		
Label	🟩 high	vs.	🟪 low
Pred.	🟧 high	vs.	🟪 low
	🟦 Corr.	vs.	🟥 Incorr.

Prediction whether house prices are above $150k.

House Style: Style of dwelling (categorical)
🟧 Prediction "high": importance: 0.566
🟪 Prediction "low": **importance: 0.574**

Figure 5.2: The full interface illustrating the aggregated histogram view. The user is comparing the model's prediction of "high" house prices (orange) to the prediction of "low" prices (purple). The user hovers over the feature "House Style" revealing a more detailed description, whether the feature is categorical or numeric, and the importance / feature weight for each of the subsets.

of a data set using histograms; using this method, a way to effectively aggregate instance-level explanations; and a study showing that this aggregation overcomes the potential harmfulness of instance-level explanations.

Following, we will first discuss related work in Section 5.2 and then motivate the circumstances of our study further in Section 5.3. We will propose our design for aggregating and comparing subsets of instance-level explanations in Section 5.4. Afterwards we will describe the experimental setup in Section 5.5. The results of the study are provided in Section 5.6 and their implications are discussed in Section 5.7. We then conclude in Section 5.8 and discuss future work.

5.2 Related Work

We broadly divide related work into two parts. Studies focusing on the effectiveness of instance-level explanations and attempts in detecting bias, using visual analytics, in data provided to machine learning models.

5.2.1 Effectiveness of Instance-level Explanations

When introducing their algorithm, LIME (Ribeiro *et al.* [102, 103, 119]), the authors conducted experiments to show the effectiveness of their method. However, instance-level explanations were only inspected individually and not in aggregate form.

Kulesza *et al.* [71] introduced explanatory debugging. Users are presented individual decisions, made by the model, in a list. Those can then can be used to "personalize" the model and improve its statistical performance by finding and giving feedback on incorrect decisions.

Zhou *et al.* [134] analyzes how uncertainty and cognitive load affects trust in a machine learning model. Here models are compared that predict the risk of pipe failure in a sewer systems according to several features. In addition to the expected failure rate according to model, the length of the observed part of the pipes is shown to the user aggregated over all instances. The study found that showing the uncertainty of the model significantly decreased the trust of participants. Additionally, adding cognitive load in terms of limited decision time trust in the model decreased significantly as well.

Narayanan *et al.* [89] explored how humans understand explanations from a machine learning model. Explanations for individual instances, in the form of simple rules, were presented and participants were asked to determine the predicted outcome of the underlying model. The study found that greater complexity, more rules and more variables, resulted in a higher response time and decreased accuracy.

Note, that the works presented so far always assume that errors stem from the shortcomings of the model and not from incorrect or biased data.

Stumpf *et al.* [113] found that under some circumstances, explanations can be harmful to the end user, by invoking false confidence. This is on one hand due to the user extrapolating from few instance-level explanations, making their mental model seem correct. And, on the other hand, trust in the machine learning model *overrides* their initial intuition: “I guess this thing knows more than me. The system knows more than me. I’ll accept [the diagnosis]”. The study investigated inspecting individual instances one after the other, however, in our experiments we could confirm both of those findings even when showing instances in tables.

5.2.2 Detecting Biases Using Visual Analytics Methods

Hohman *et al.* [52] identifies detecting biased data as one of their five use cases for visual analytics for machine learning. However, their examples focus on work that only looks at the data without the help of machine learning models [44] or simple models where humans adjust the thresholds of the model manually [127].

Chang *et al.* [27] uses crowd-sourcing to label data and ensure its integrity. However, this approach does not work if domain expertise in the field is required to label data correctly.

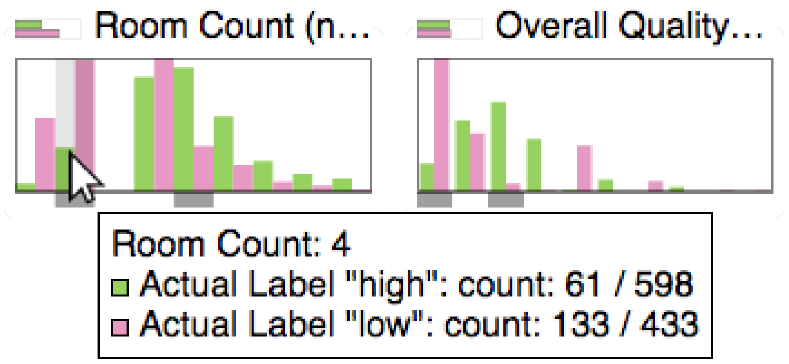
Simard *et al.* [108] introduces Machine Teaching. This paradigm uses an already labeled data set for training a machine learning model. It then presents predicted instances to a domain expert who then can either, fix an incorrect label, manipulate features, change constrains, or post-pone a decision if the instance is ambiguous. This way an expert can ensure that the final model is correct and remove biases. However, finding biases is not scalable as the experts has to go through many examples and might miss problems, especially if the performance of the model increases but the underlying data is incorrect.

Krause *et al.* [65] demonstrates, how aggregated instance-level explanations can be used to find biases in hospital data. They used an instance-level algorithm optimized for sparse binary input data (Martens and Provost [83]). Through aggregation, filtering, and reordering, they found biases in their data used for predicting hospital admission that made it impossible for the machine learning model to correctly predict admission in some cases. For example, the model knew about a CET or PET scan happening but was unaware of their results. Thus, the model was unable to predict the diagnosis since the result of the scan directly influences the outcome.

5.3 Motivation

Experiments for instance-level explanations typically focus on use cases where the explanation is presented to the user one instance at a time. This is helpful when monitoring the continuous performance of a machine learning model in production. However, it limits one's ability to gather a holistic view of a model's behavior (*i.e.*, a global explanation). Looking at many instances is very time consuming and potentially ineffective. It is not clear whether people can build a coherent understanding of a model by looking at a series of instances: comparison between many instances overloads memory and does not leverage the data compression capabilities of aggregate representations.

The main goal of our study is therefore to explore the idea of aggregating data about many instances and their explanations and verify its effects on model comprehension. More precisely, we want to study the effect of aggregation on what we call "semantic validation": the ability of a human to validate the decisions of a model according to his or her knowledge of the domain.



Room Count: 4
🟩 Actual Label "high": count: 61 / 598
🟪 Actual Label "low": count: 133 / 433

Figure 5.3: Comparing the distribution of values by "Actual Label" (*i.e.*, ground truth). The height of the bars show the percentage of values within the respective subset (green for "high" outcomes and pink for "low" outcomes). The average feature weight of each subset is shown next to the feature name. This is only visible in the condition including explanations.

For this purpose, somebody knowledgeable with the domain has to verify that the model and the data are consistent with their mental model and, if necessary, override information coming from statistical aggregates on accuracy. This is an important task, especially for models making critical decisions such as those employed in health care [25] and security.

A second goal of this study is to better understand how explanations contribute to semantic validation. Explanations typically provide, for each instance, a weight or score that conveys information about how important each feature is, for a given decision, and for a given instance. An important question therefore is to better understand what particular benefits, if any, explanations bring to human validation; whether this is conducted using an instance-level exploration strategy or a more compact aggregation. Our hypothesis is that explanations may bring value if they manage to direct the user's attention to instances and features where biases and mistakes reside.

In summary, our experiment aims at studying the effect of two main factors:

aggregation level, that is, instances vs. aggregations, and explanations, that is, whether feature weights are present or not.

5.4 Design Contribution

In order to effectively analyze machine learning behavior we allow users to compare subsets of the data set to each other. These subsets are defined by different combinations of cells in the confusion matrix of the machine learning model. We selected subsets that help understanding the behavior of the model:

All. The full data set is shown and no comparison occurs. This is the initial view of the data.

Ground truth. By comparing rows of the confusion matrix to each other a user can explore the actual labels of the data.

Predicted labels. By comparing columns of the confusion matrix to each other a user can explore the predicted labels of the model.

Correctness. By comparing the diagonals of the confusion matrix to each other a user can explore when the model's prediction is correct or incorrect.

It is thinkable to allow more freedom in selecting subsets to, *e.g.*, compare only errors of a certain predicted label, however, this would increase the complexity of the user interface and a user has to understand when to use each of those subsets in order to be effective.

When aggregating instances, comparing subsets to each other is not trivial. The naïve solution of showing the actual amount of instances with respect to the full data set disadvantages the smaller subset. However, it is not as important to know the actual distribution, but rather where one subset has a significantly higher or lower concentration of instances compared to the other. To this extend we propose a novel approach of scaling each subset separately with respect to their own magnitude (see Figure 5.3). Note, that we then compare percentages of instances within the respective subsets. We further indicate strong differences in the subsets by showing a gray bar at the bottom of the histogram. We are not aware of any literature that uses or explored this way of comparing subsets with histograms and claim it as a design contribution. We demonstrate the effectiveness of this design in Section 5.6.

5.5 Experimental Setup

In order to see whether explanations are helpful in detecting biases of the training data we used the publicly available housing price data [31] and created a version that has a bias that needs to be detected. Originally a regression task, we converted the data set into a classification task predicting whether the house price is above $150k (598 instances above; 433 instances below; 1031 instances in total). We also reduced the number of features in the data set to 10 in order to make it possible to see all features at once in all conditions, without the need to scroll, so data otherwise hidden off-screen would not be a confounding effect. The biased dataset needed to have a bias detectable in both the aggregated and instance-level version, thus we chose to manipulate the outcome of the biased data set to be dependent on the value of one feature ("Living Area") with some random perturbations. The biased outcome was chosen so that a *larger* "Living

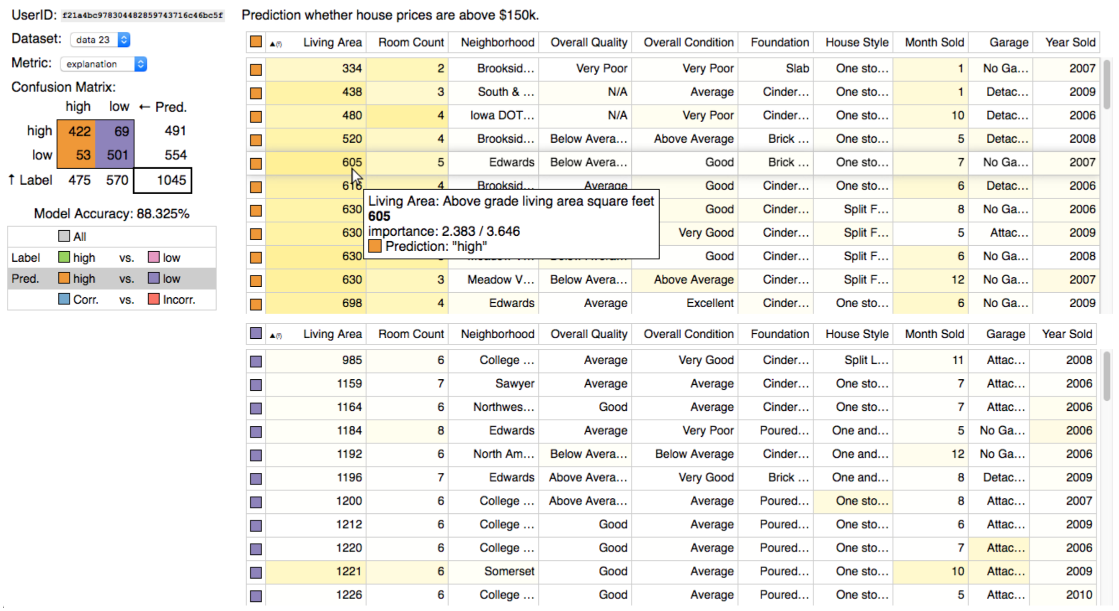
Area" results in a *lower* house price. This relationship does not reflect reality (an increased "Living Area" generally results in a higher house price). The bias is present to the same degree in both the training and the testing data.

Furthermore, by controlling the degree of randomness while creating the biased data set we controlled the accuracy of the prediction when training a machine learning model, such that the model using the biased data has a *higher* accuracy than the model on the real data. We trained Multi-Layer Perceptrons [50] on both data sets resulting in test accuracies of 81.96% for the real data and 88.33% for the biased data.

5.5.1 User Interface Conditions

For explaining the model behavior we computed the explanations using the LIME algorithm [102] on the test data. LIME computes feature weights for each instance in the data. A weight of zero indicates that the feature was not used in the prediction whereas a non-zero weight indicates that the feature was used. A feature weight with larger magnitude indicates that the feature was more important to the prediction than a feature with a smaller magnitude of its weight. However, in order to simplify the user interface and understanding, we computed the absolute value of the feature weights. Thus, participants will only see if a feature has influence on the prediction, not whether this influence is towards a "low" or "high" house price prediction. This additional information is not relevant for the given task and would make the user interface confusing.

For comparing instance-level and aggregated conditions we developed two user interfaces. Both interfaces share two major components, the confusion matrix of the current model alongside the model's accuracy and a list for selecting different



UserID: f21a4bc978304482859743716c46bc5f

Dataset: data 23

Metric: explanation

Confusion Matrix:

	high	low	← Pred.
high	422	69	491
low	53	501	554
↑ Label	475	570	1045

Model Accuracy: 88.325%

	⬜ All		
Label	🟩 high	vs.	🟪 low
Pred.	🟧 high	vs.	🟪 low
	🟦 Corr.	vs.	🟥 Incorr.

Prediction whether house prices are above $150k.

🟧	▲ Living Area	Room Count	Neighborhood	Overall Quality	Overall Condition	Foundation	House Style	Month Sold	Garage	Year Sold
🟧	334	2	Brooksid…	Very Poor	Very Poor	Slab	One sto…	1	No Ga…	2007
🟧	438	3	South & …	N/A	Average	Cinder…	One sto…	1	Detac…	2009
🟧	480	4	Iowa DOT…	N/A	Very Poor	Cinder…	One sto…	10	Detac…	2006
🟧	520	4	Brooksid…	Below Avera…	Above Average	Brick …	One sto…	5	Detac…	2008
🟧	605	5	Edwards	Below Avera…	Good	Brick …	One sto…	7	No Ga…	2007
🟧	616	4	Brooksid…	Average	Good	Cinder…	One sto…	6	Detac…	2006
🟧	630				Good	Cinder…	Split F…	8	No Ga…	2006
🟧	630				Very Good	Cinder…	Split F…	5	Attac…	2009
🟧	630				Good	Cinder…	Split F…	6	No Ga…	2008
🟧	630	3	Meadow V…	Below Avera…	Above Average	Cinder…	Split F…	12	No Ga…	2007
🟧	698	4	Edwards	Average	Excellent	Cinder…	One sto…	6	No Ga…	2009

Living Area: Above grade living area square feet
605
importance: 2.383 / 3.646
🟧 Prediction: "high"

🟪	▲ Living Area	Room Count	Neighborhood	Overall Quality	Overall Condition	Foundation	House Style	Month Sold	Garage	Year Sold
🟪	985	6	College …	Average	Very Good	Cinder…	Split L…	11	Attac…	2008
🟪	1159	7	Sawyer	Average	Average	Cinder…	One sto…	7	Attac…	2006
🟪	1164	6	Northwes…	Good	Average	Cinder…	One sto…	7	Attac…	2006
🟪	1184	8	Edwards	Average	Very Poor	Poured…	One and…	5	No Ga…	2006
🟪	1192	6	North Am…	Below Avera…	Below Average	Cinder…	One and…	12	No Ga…	2006
🟪	1196	7	Edwards	Above Avera…	Very Good	Brick …	One and…	8	Detac…	2009
🟪	1200	6	College …	Above Avera…	Average	Poured…	One sto…	8	Attac…	2007
🟪	1212	6	College …	Good	Average	Poured…	One sto…	6	Attac…	2009
🟪	1220	6	College …	Good	Average	Poured…	One sto…	7	Attac…	2006
🟪	1221	6	Somerset	Good	Average	Poured…	One sto…	10	Attac…	2009
🟪	1226	6	College …	Good	Average	Poured…	One sto…	5	Attac…	2010

Figure 5.4: The full interface illustrating the table view showing individual instances. The user is comparing the model's prediction of "high" house prices (orange) to the prediction of "low" prices (purple). The feature "Living Area" is ordered by ascending values and the user hovers over the cell with the value "605".

subsets to compare to each other (see Figure 5.1). Those subsets can be: comparing instances with different ground truth labels, instances with different predicted labels, instances with different correctness, or the full dataset in which case no comparison occurs. How comparing subsets looks like is dependent on the which condition is used. The selections use different colors to distinguish the subsets in order to prevent participants getting confused about which subset comparison is currently selected (we also indicate the selection in the list). The colors are also used to highlight the confusion matrix cells corresponding to the current selection. Note, that all selections always represent all instances in the data and no two instances from the same confusion matrix cell can appear in opposing subsets.

The design of the user interface lets users iterate through multiple useful slices of the data (such as getting an overview of the data or comparing different, meaningful, subsets to each other). This design, inspired by SYF [95], provides users with a systematic guide to iterate through meaningful views while also supporting flexible diversions to pursue insights.

5.5.1.1 Instance-level Condition

The user interface for the instance-level condition is a table showing the values of each feature for each instance in its cells, as seen in Figure 5.4. This is a change from how instance-level explanations are usually studied in the literature, where each instance is presented in isolation. However, this way of showing instances is limiting as it becomes time consuming to inspect more instances so a participant would only see very few instances in total.

The columns of the table, representing features, are ordered by the average weight of this feature, if the condition includes explanations. If the condition does not include explanations the columns are sorted alphabetically. The cells of the table reflect the feature weight of the corresponding instance using a yellow color scale. In addition to that, hovering over a cell with the mouse shows a tool-tip indicating the actual feature weight number and the full value of the cell and the full feature name in case those values got abbreviated due to cell size restrictions.

Columns can be sorted by clicking on the table header. This cycles through sorting the feature values by ascending and descending value. If explanations are available, the feature can also be sorted by ascending and descending feature weights.

For comparing different subsets of the data we show two aligned tables. The colors representing the different subsets are shown on the far left side of the table.

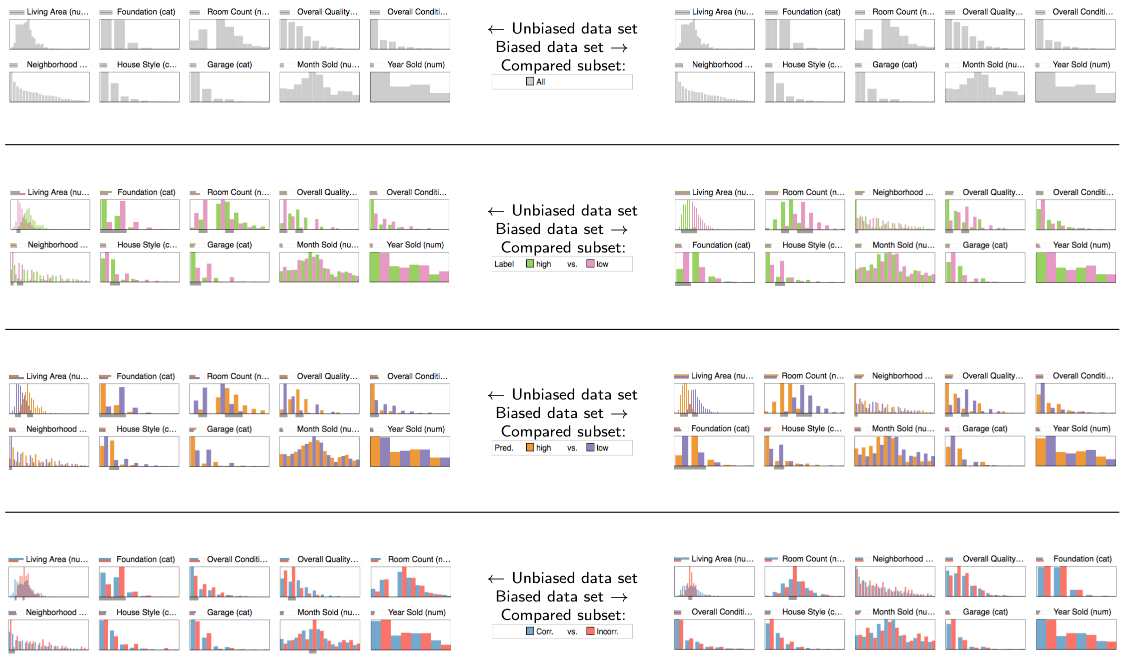


Figure 5.5: Comparison of different subset selections on both the unbiased model (left side) and the biased model (right side). Features are sorted per segment by most important at the top-left, row-wise to least important at the bottom-right. Note, the feature "Living Area", in both the "Label" (*i.e.*, ground truth) and "Pred." (*i.e.*, model prediction), has flipped outcomes for the biased model (right side). Each subset has a different color palette to not confuse different selections with each other.

5.5.1.2 Aggregated Condition

The user interface for the aggregated condition represents the distribution of feature values as histograms, similar to [64]. Feature names are shown above the histogram and the histograms are arranged left to right row-wise and top to bottom. For the condition with explanations available, a small bar chart next to the feature name indicates the average weight of this feature. In this condition, the histograms are ordered by descending magnitude of average feature weights. The averages are computed for each subset separately. The order is, more specifically, based on the average of the subsets' average weights, which allows features to appear first that are only important under certain conditions. If no explanations are provided, the order is alphabetically.

Hovering over a histogram with the mouse reveals tool-tips showing the actual instance count of the hovered histogram segment, as well as, the full feature name and the feature weight number. For categorical features the bars of the histograms are slimmer so that distinct values are more easily separable. Additionally, the order of the values indicate their quantity in the data set with the most common categorical value first on the left.

When comparing different subsets of the data, as seen in Figure 5.5, bars of each color are shown next to each other in the histogram. The height of the bars are scaled by their relative proportion *within* each subset. This means the height indicates the percentage of instances in the respective subset. The vertical scale ranges to the highest percentage across both subsets. This allows for seeing where one subset is more concentrated than the other independent of the total size of each subset. In order to indicate big differences in the distribution we show a gray rectangle at the bottom of the histogram if one subset is strongly more concentrated at this value range than the other (see Figure 5.3).

5.5.2 Study Design

We study four conditions which result from combining different representations with the inclusion or exclusion of explanations. The different representations are:

- A view of the model through individual instances. Instances are listed in a table. This is an extension of the approach of inspecting instances one-by-one.
- A view of the model through aggregated instances. Instances are aggregated in histograms for individual features.

In each of those conditions we compare the ability to detect biases in the data by comparing the unbiased data set to the data set with the manufactured bias.

We also explore the impact of those conditions on whether explanations and aggregation improve trust in the model's decisions. Particularly, trust in the *right* model.

5.5.3 Tasks and Measurements

In order to test conditions against each other we created a questionnaire. After asking the participant about the knowledge of machine learning and basic terminology, we have a training section for participants to familiarize themselves with the interface. First, an introductory video explains all components of the interface. The video uses an example model from a different data set which is designed to predict whether a room is occupied or not based on predictions from various sensors [81]. Then, a series of questions about this example model are asked and the participant can and has to use the interface to answer them correctly. The questions ask about the values of features under certain conditions, such as "What is the model's prediction for high values of 'CO_2'?", "What is the lowest value of 'Humidity' that predicts 'occupied'?", "Are the predictions for low values of 'Light' correct?". The questions are constructed in a way to be easily answerable under all conditions given an understanding of the user interface and basic principles of machine learning. An incorrect answer leads back to the beginning of the section and the participant is given the chance to correct the mistake. We did not use those questions to exclude any participant but rather for giving them an opportunity to get comfortable with the user interface. Note, that it is not necessary for participants to have a deep knowledge in *how* machine learning algorithms work, as long as, the basic principles of prediction, ground truth, or accuracy are clear. This reflects that domain experts would often not necessarily be trained in *developing* machine learning models but rather *using* them.

As final question of this segment the participant has to detect, in a hypothetical, scenario that a prediction does not make *sense* from a semantic standpoint, even though that prediction is *correct* from the perspective of the model. This question aims to prime the participants for the upcoming task and teaches that model correctness is not necessarily equivalent to semantic correctness. After this, we ask some common sense questions about how house prices are supposed to correspond to certain features. This ensures that participants have enough domain knowledge for the upcoming task.

In the main part of the study we present the participant with both housing data models one after the other and encourage them to explore the models with the end goal of determining which model can be trusted more. The order of the data sets is random. The participant then has to answer the following questions about each model: “Do you think the predictions of the model make sense?”, “How well does the model perform in terms of accuracy?”, “How much do you trust the model?” on a five-point Likert scale; and explain the reasoning for their answers. For each question we provide a more in-depth explanation with examples.

After inspecting both models, we ask the participants to state which model performs better in terms of accuracy, which model can be trusted more, and whether the model they trust more had the higher or lower accuracy, or no model can be trusted more than the other. We ask participants to describe their reasoning and also state their confidence in their decision on a five-point Likert scale.

5.5.4 Participants

We ran all four conditions of the study on Prolific[1], an online survey recruitment system. Participants in online recruitment aim to increase their payout to effort ratio. Thus, we took several measures to ensure high data quality. Firstly, we only allowed participants with a high rating on the platform and an interest in computer science. Secondly, we excluded all data from participants that had a suspiciously fast completion times (less than 10 minutes after watching the introductory video) which would not allow them to establish well thought out answers. We also excluded participants with too little interaction with the interface, determined by the number of histograms or table cells they inspected and how often they changed the subset comparison selection. As attention question we used the question "Which model had the higher accuracy?" to remove participants. This question has an objective answer that had to be determined during the study as well. By reading the full text answers we could exclude participants that had little understanding of machine learning or the assigned task, or were giving non-sense answers. We retained 100 eligible participants divided evenly across the four conditions. This represents less than 47% of total participants, not counting participants that stopped the study before submitting.

5.6 Results

Following, we will perform an exploratory analysis of our study results.

[1]https://www.prolific.ac/

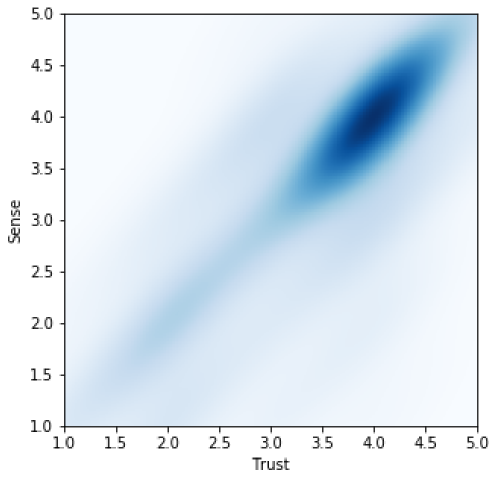
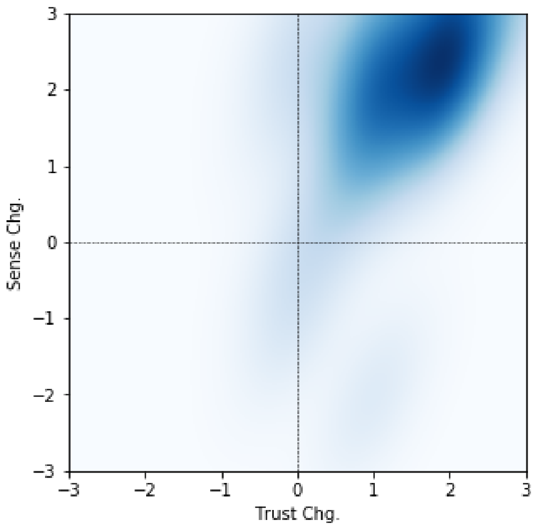


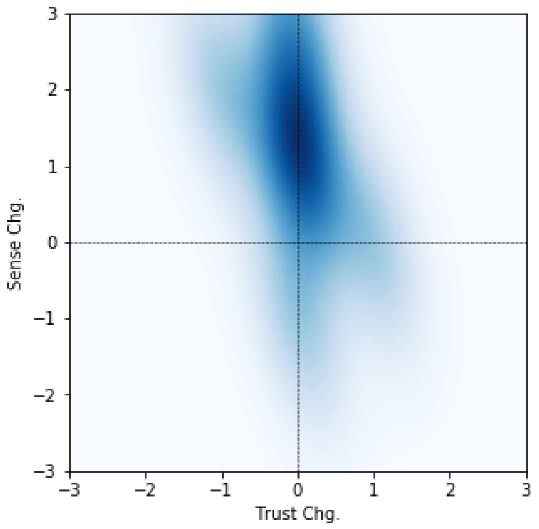
Figure 5.6: A kernel density estimation of the responses of participants on a five-point Likert scale about how much they trust the biased model and whether it makes sense to them. Note, that the majority of participants did not detect the bias of the model.

5.6.1 Bias Detection and Trust

When comparing how much participants trust a particular model and whether they think this model makes sense, one can see that those responses are typically correlated (see for example Figure 5.6 with a Pearson correlation coefficient of 0.759 and Spearman rank-order correlation coefficient of 0.745). This also extends to plain-text answers, which allow to detect whether participants correctly found the bias in the correct data set unambiguously. Participants were very verbose about their findings, if they found something: "*It has higher accuracy so should be more trustworthy than the other one. However some of the results don't make sense to me. Maybe this is just an atypical property market.*"; "*It is accurate, yet the predictions do not make much sense. Higher quality houses having a larger amount of low priced houses, percentage-wise? More rooms, area, or stories resulting in lower prices? The logic does not work out.*"; "*larger houses are valued lower than others which are smaller*" (sic).



(a) Bias detected and correct preference.

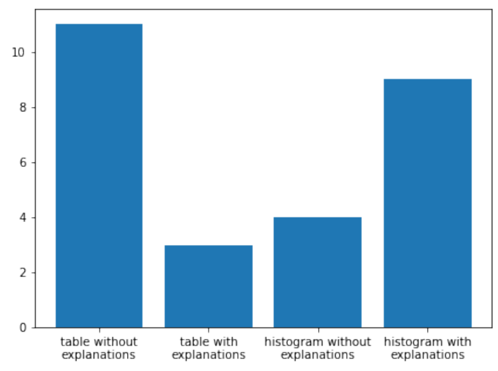


(b) Bias detected but no preference change.

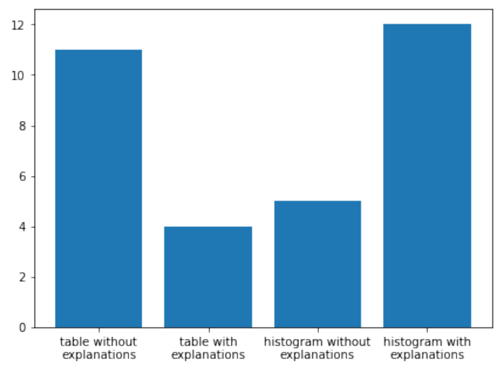
Figure 5.7: How participants changed their responses comparing the unbiased model to the biased model. A positive value indicates that the response was higher in the unbiased model. (a) shows the case when participants detected the bias and subsequently preferred the unbiased model. (b) shows the case when participants detected the bias but still chose the biased model.

However, the above mentioned correlation is not perfect. This is likely due to some participants not being convinced, that their correct discovery of the flaw in the data is enough that the corresponding model cannot be trusted: "*If the data says it's true, then it's true I suppose and it's more trustworthy than my common sense.*"; "*I feel like the results of [the biased model] where strange even though they where correct according to the dataset.*"; "*I'm drawn to trusting the model which was more accurate even though it didn't entirely make sense to me.*" (sic).

This divergence in trust and the finding of flaws in the data can also be seen in Figure 5.7. If finding the flaw in the data swayed the participant to not trust the model an increase in trust for the unbiased model compared to the biased model corresponds to an increase in the perception that the unbiased model makes more



(a) Correct trust



(b) Detected bias

Figure 5.8: (a) compares how many participants trusted the unbiased, thus correct, model more. (b) compares how many participants correctly identified the bias in the data, determined from plain-text answers. Note, that in both cases adding explanations to the table view hurt performance, whereas adding explanations to the histogram view improved performance.

sense (Figure 5.7a). However, if the finding did not influence the preference, trust between both models stayed the same (Figure 5.7b).

In total, 25% of people who correctly identified the bias still opted to trust the biased model more, due to the higher reported accuracy of the model. A further 8% who identified the bias trusted both models equally. This aligns with the findings of Stumpf *et al.* [113] that trust in the machine learning model may *override* people's initial intuition about its performance.

5.6.2 Comparison Across Conditions

Comparing the correctness across all four conditions can be seen in Figure 5.8. First, we can see a strong improvement both in correctness and whether the participant trusted the unbiased model more, when switching from tables to histograms while having access to explanations (p-value Figure 5.8a: Fisher's 0.0477, χ^2 0.0489; p-value Figure 5.8b: Fisher's 0.0161, χ^2 0.0169). When adding

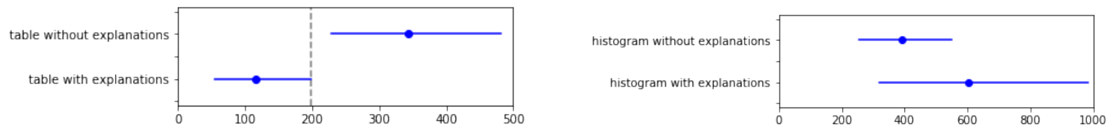


Figure 5.9: The left side shows interactions with the table view measured by counting how many table cells were hovered by the mouse. The right side shows interactions with the histogram view measured by counting how many histogram bars were hovered by the mouse. The plot shows the bootstrapped mean and confidence interval for each setting.

explanations to histograms (p-value Figure 5.8b: Fisher's 0.0359, χ^2 0.0366) we can see a significant improvement when comparing correctness. We hypothesize that explanations are a necessity for histograms to work effectively, since they point out which, of the possibly many, pattern seen in the distributions are meaningful. We can also see an improvement in whether the participant trusted the unbiased model, however, it is not significant (p-value Figure 5.8a: Fisher's 0.0982, χ^2 0.0986).

Furthermore, we see a strong decline in correctness when adding explanations to tables (p-value Figure 5.8a: Fisher's 0.0127, χ^2 0.0137; p-value Figure 5.8b: Fisher's 0.0311, χ^2 0.0320). At first, we were puzzled at this counter-intuitive result and we double and triple checked that those results were not a simple mix-up in conditions. We hypothesize that, having explanations at hand in a table, focuses the attention of participants to fewer instances and additionally makes them more confident that they fully understood the model. This extrapolation from few instances aligns with the findings of Stumpf *et al.* [113], who found that explanations can be harmful in certain circumstances, and shows that the findings also apply to a tabular representation of the explanations.

In order to investigate this hypothesis further, we can look at the number of interactions of the participants performing the tasks. We can see in Figure 5.9 that participants engaged with the table view significantly more when no explanations

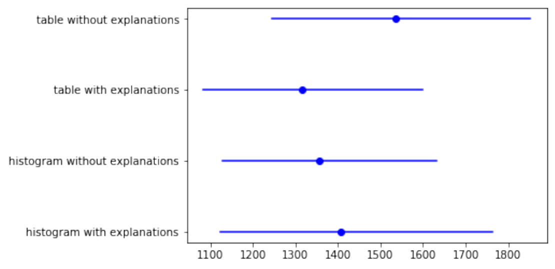


Figure 5.10: Overall completion time of the study by condition. The plot shows the bootstrapped mean and confidence interval for each setting. There is no significant difference between the conditions.

were present. This might be an example of Hullman *et al.* [54], who state that information visualization might benefit from visual difficulties, since people are forced to interact more with the visualization. This seems to be the case for a table, without any further help from the interface about what to look at.

Despite that, we found no significant difference in the time participants took to complete the study, as can be seen in Figure 5.10. Even though the histogram view with explanations and the table view without explanations do not have a significant timing difference, we hypothesize that an aggregated representation of the model is a more effective method for finding biases. This hypothesis is rooted in both conditions performing equally well and histograms being a more scalable data representation than tables, due to their independence from the magnitude of the data.

5.7 Discussion

We showed that aggregating instance-level explanations can be an effective way of enabling humans to identify biases in the input data of machine learning tasks. Even though, assisted aggregation is as effective as unassisted individual inspection it scales better with large data sets. Individually inspecting instances in the data is only possible on a sample of the data and requires extrapolation of findings to the whole data set. Aggregation does not suffer from this, as the representation of the data is independent from its size. Even though, it requires dedication, our test data set was small enough to still be able to scan in full if necessary.

Furthermore, the bias planted in the data was simple enough to be able to be found under *all* conditions. This might not be true for real-world data sets with more complex biases. Even though, histograms are advantageous with respect to tables in finding arbitrary patterns, they are still limited to only one dimension. Biases that are present only through combinations of features will not be detectable.

In our study, we could confirm findings from Stumpf *et al.* [113] and overcome their limitations by using instance-level explanations with aggregation. However, we could not overcome trust in machine learning model authority, despite being confronted with contradictory evidence, in all cases. Speculatively, this might stem from people being used to being presented with cleaned up and validated data, as this cumbersome process is often hidden from the end result.

5.8 Conclusion & Future Work

We presented a novel way of aggregating and comparing instance-level explanations. We found that this method can help humans identify biases in the input data to machine learning models. However, this is only the case in combination. Aggregation alone or individual instance-level explanations might lead to worse performance in this regard. We demonstrated, that an aggregated instance-level explanation approach is as effective as going through the data unassisted. This is promising, as the proposed method is independent of the size of the data set and thus likely more scalable than its non-aggregated counterpart. However, specifically confirming this hypothesis remains future work.

As we were conducting an exploratory analysis of the study, individual findings remain to be tested in-situ in future work. Furthermore, experimenting with more complex forms of data biases opens up additional research opportunities.

All in all, we presented a usable method for effectively utilizing instance-level explanations on a large scale. As machine learning models become more complex and opaque, this becomes an important stepping stone in tackling the behemoth of effectively improving machine learning models and their data alike.

5.9 General Discussion

In our study we found that aggregating instance-level explanations using histograms and subset comparison provides a scalable alternative to inspecting tabular data manually for finding biases in the input data of a predictive model. We only tested finding biases that were possible to find under all of our four conditions. It would be easy to construct a bias that is impossible to find looking at a table of the data, but this would be an unfair comparison. Thus, our method is more scalable, since it enables to find a super-set of biases detectable in tables.

However, our method requires both aggregation and instance-level explanations. Without either of those, performance in finding biases decreased significantly. This also brings light to the issue that individual instance-level explanations can be misleading and thus harmful. We could confirm, that the findings of Stumpf *et al.* [113], which show that inspecting individual instance-level explanations can lead to harmful extrapolations of the model's behavior, also apply to a tabular representation. However, our method overcomes this harmfulness.

Furthermore, even when confronted with evidence of an erroneous data set a quarter of the participants still preferred the model with higher accuracy but incorrect data. This implies a strong trust in statistical accuracy whose perception needs to be reevaluated.

The presented work shows the effectiveness of aggregated instance-level explanations, and in extension of the previously presented Model Diagnostic workflow, but it also opens up further research opportunities.

Chapter 6

Thesis Summary

In this thesis, we discussed the role of visual analytics to explain black-box machine learning. We saw that global approaches, like feature selection methods, are not informative enough to help understand model decisions. Different strategies preferred different, equally reasonable, feature sets without having a significant impact on predictive performance. This showed, that inspecting and comparing alternate settings may let machine learning experts develop insights that overwrite their initial intuitions.

Next, we saw that partial dependence with a derived feature importance score allowed to effectively detect model errors. Detectable errors included: over-fitting, under-fitting, biases caused by imputation, and leaking labels caused by incorrect cause-effect relationships. Furthermore, localized inspections helped to understand the how and why of specific instance predictions by finding locally impactful features.

Then, we proposed and discussed the Model Diagnostic workflow, which is based on aggregating local instance-level explanations in order to gain insights about a model with the intent of improving it. We showed that the Model Diagnostic workflow enables a scalable understanding of local decision making of a predictive model and that aggregating instance-level explanations allows for semantic validation of the input data. As a model can only perform as well as its input data, gaining insights about the limitations of a model in turn help with feature engineering on said data.

Finally, we showed how histograms can be effectively used to analyze aggregated instance-level explanations. With the help of comparing meaningful subsets, aggregated instance-level explanations significantly outperformed inspecting individual explanations or aggregation without explanations in detecting biases in the input data while being more scalable than a tabular representation. Furthermore, we showed that inspecting individual instance-level explanations can be misleading and hurts detecting biases in the input data.

All in all, we demonstrated a workflow for effectively utilizing black-box instance-level explanations in order to improve model correctness via semantic validation. However, this approach has some limitations to overcome which we will explore next. Note, that those limitations are not an argument against the presented techniques but merely a starting point for future research.

6.1 Limitations and Assumptions of Black-Box Analysis

Analyzing machine learning models using black-box techniques can only approximate the true behavior of the model. Looking at more instances or increasing the sample rate for both partial-dependence plots or instance-level explanations increases the granularity and precision of the analysis but cannot express the underlying relationships of the input in full. On the other hand, those relationships can be too complex to be understood by a human. The challenge is to find a middle ground between the fidelity of explanations and their interpretability.

Additionally, some tasks have complex interactions between input features that cannot be described by interpreting individual features. For example, instance-level explanations provide weights for the features for a given instance. However, those weights can be completely different for other instances. Instance-level explanations alone do not provide enough information to reason about why the weights changed in this particular way. A possible solution could be to expand the concept of explanations to allow for representing non-trivial relationships between features, such as using rules to express partial behaviors of the model or a higher dimensional partial dependence.

Black-box analysis methods provide the convenience of using the same algorithm for multiple models. However, with regards to data some decisions still need to be made. For example, for the instance-level explanations in Chapter 4 we determined that Martens and Provost's [83] algorithm provided consistently better results than Riberio *et al.*'s[102] algorithm for our use case of highly sparse binary data. This is due to the fact that different data types require different approaches for explanations. A binary feature can have an explanation that describes the presence or absence of the feature whereas for a numeric feature the actual value is important. Our Model Diagnostic workflow is agnostic to the used instance-level explanation algorithm but this choice still has to be made.

6.2 Generalization to Other Forms of Machine Learning

In this thesis, we only focused on predictive modeling tasks with structured inputs. Even though this is a large and popular subset, it does not cover the entirety of machine learning or even predictive modeling. Presented techniques are not immediately transferable to tasks such as online learning or streaming input data, which requires recurrent models, such as LSTM (Long Short Term Memory) models [51]. Those areas pose further challenges and offer a variety of research opportunities.

On a different note, the recent popularity of neural networks offers an additional opportunity in terms applicability of the work in this thesis. Neural networks are differentiable models, which makes it possible to compute gradients towards desired outcomes for given inputs. As instance-level explanations aim to approximate this gradient to some degree (LIME [102] is essentially a Monte-Carlo approximation of the above mentioned gradient) gradients might be used as drop-in replacements for instance-level explanations in the techniques proposed in this thesis. However, whether gradients are a computationally faster alternative to instance-level explanations and whether they can achieve similar results poses an interesting open research question.

6.3 Implications

Meta-learning, *i.e.*, using machine learning to learn model architectures for solving the actual task, is recently growing in popularity. With systems, such as auto-sklearn [38] or AlphaGo [107], the role of humans in machine learning shifts away from being the architect of models to being a domain expert of the modeled problem. Humans have vast contextual knowledge that is hard to communicate directly to machine learning models. As such, understanding the behavior of black-box machine learning models and semantic validation of their performance, as presented in this thesis, is becoming more and more relevant.

Interestingly, because humans are used to having contextual knowledge about real world problems machine learning is trying to solve, it often seems surprising when and why a model failed, since the correct prediction “is so obvious”. Thus, it is crucial for machine learning models to correctly communicate their behavior to humans in order to prevent such fallacies to go unnoticed. This thesis provided an important step towards achieving this goal.

Chapter 7

Conclusion & Future Work

In this thesis we explored how visual analytics can be used in order to leverage the full potential of black-box machine learning. To this extend we used a variety of techniques to explain the behavior of machine learning models, both globally and locally, and developed a workflow to utilize those explanations to diagnose models. We also studied the effect of aggregating local explanations in order to detect biases in the input data of machine learning models. However, each finding shown in this thesis posed further questions, providing research opportunities to be investigated.

This thesis identified problems in the area of machine learning that can only be fully solved with human expertise. In this context, visual analytics proved to be a powerful tool in assisting humans in performing those tasks.

Bibliography

[1] E. Alpaydin. *Introduction to Machine Learning.* Massachusetts Institute of Technology, 2010.

[2] B. Alsallakh, A. Hanbury, H. Hauser, S. Miksch, and A. Rauber. Visual methods for analyzing probabilistic classification data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1703–1712, 2014.

[3] B. Alsallakh, A. Jourabloo, M. Ye, X. Liu, and L. Ren. Do convolutional neural networks learn class hierarchy? *CoRR*, abs/1710.06501, 2017.

[4] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.

[5] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 337–346, April 2015.

[6] S. Amershi, J. Fogarty, A. Kapoor, and T. Desney. 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, pages 47–50, New York, NY, USA, 2011. ACM.

[7] S. Amershi, J. Fogarty, and D. Weld. 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, pages 21–30, New York, NY, USA, 2012. ACM.

[8] S. Amershi, B. Lee, A. Kapoor, R. Mahajan, and B. Christian. Cuet: Human-guided fast and accurate network alarm triage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 157–166, New York, NY, USA, 2011. ACM.

[9] M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel. Visual classification: an interactive approach to decision tree construction. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 392–396. ACM, 1999.

[10] M. Ankerst, M. Ester, and H.-P. Kriegel. Towards an effective cooperation of the user and the computer for classification. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 179–188. ACM, 2000.

[11] F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1):17–21, 1973.

[12] B. Baesens, R. Setiono, C. Mues, and J. Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3):312–329, 2003.

[13] B. Becker, R. Kohavi, and D. Sommerfield. Visualizing the simple baysian classifier. In U. Fayyad, G. G. Grinstein, and A. Wierse, editors, *Information Visualization in Data Mining and Knowledge Discovery*, pages 237–249. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[14] R. Bellazzi and B. Zupan. Predictive data mining in clinical medicine: current issues and guidelines. *International journal of medical informatics*, 77(2):81–97, 2008.

[15] E. Bertini, H. Lam, and A. Perer. Summaries: a special issue on evaluation for information visualization. *Information Visualization*, 10(3), 2011.

[16] E. Bertini, A. Tatu, and D. Keim. Quality metrics in high-dimensional data visualization: an overview and systematization. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2203–2212, 2011.

[17] J. Bird and P. Layzell. The evolved radio and its implications for modelling the evolution of novel sensors. In *Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress - Volume 02*, CEC '02, pages 1836–1841, Washington, DC, USA, 2002. IEEE Computer Society.

[18] J. Boyle, M. Jessup, J. Crilly, D. Green, J. Lind, M. Wallis, P. Miller, and G. Fitzgerald. Predicting emergency department admissions. *Emerg Med J*, 29(5):358–365, May 2012.

[19] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145 – 1159, 1997.

[20] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct. 2001.

[21] L. Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, Aug 2001.

[22] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer. Adversarial Patch. *ArXiv e-prints*, Dec. 2017.

[23] A. Cameron, K. Rodgers, A. Ireland, R. Jamdar, and G. A. McKay. A simple tool to predict admission at the time of triage. *Emerg Med J*, 32(3):174–179, Mar 2015.

[24] D. Caragea, D. Cook, and V. Honavar. Gaining insights into support vector machine pattern classifiers using projection-based tour methods. In *Proceedings of the KDD Conference*, pages 251–256, 2001.

[25] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. 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, pages 1721–1730, New York, NY, USA, 2015. ACM.

[26] D. Chang, L. Dooley, and J. E. Tuovinen. Gestalt theory in visual screen design: A new look at an old subject. In *Proceedings of the Seventh World Conference on Computers in Education Conference on Computers in Education: Australian Topics - Volume 8*, CRPIT '02, pages 5–12, Darlinghurst, Australia, Australia, 2002. Australian Computer Society, Inc.

[27] J. C. Chang, S. Amershi, and E. Kamar. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI*

Conference on Human Factors in Computing Systems, CHI '17, pages 2334–2346, New York, NY, USA, 2017. ACM.

[28] H. Chen, S. S. Fuller, C. Friedman, and W. Hersh. *Medical informatics: knowledge management and data mining in biomedicine*, volume 8. Springer, 2006.

[29] J. Choo, H. Lee, J. Kihm, and H. Park. ivisclassifier: An interactive visual analytics system for classification based on supervised dimension reduction. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 27–34. IEEE, 2010.

[30] W. S. Cleveland and R. McGill. Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, 79(387):531–554, 1984.

[31] D. D. Cock. Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. *Journal of Statistics Education Volume 19, Number 3*, 2011.

[32] P. Cortez and M. J. Embrechts. Opening black box data mining models using sensitivity analysis. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 341–348. IEEE, April 2011.

[33] P. Cortez and M. J. Embrechts. Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences*, 225:1–17, 2013.

[34] M. W. Craven and J. W. Shavlik. Using neural networks for data mining, 1998.

[35] A. Dasgupta, J.-Y. Lee, R. Wilson, R. A. Lafrance, N. Cramer, K. Cook, and S. Payne. Familiarity vs trust: A comparative study of domain scientists' trust in visual analytics and conventional analysis methods. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):271–280, 2017.

[36] P. Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55(10):78–87, Oct. 2012.

[37] J. Ehrlinger. ggrandomforests: Random forests for regression, 2015. R package version 1.1.4.

[38] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc., 2015.

[39] E. Frank and M. Hall. Visualizing class probability estimators. In N. Lavra, D. Gamberger, L. Todorovski, and H. Blockeel, editors, *Knowledge Discovery in Databases (PKDD)*, volume 2838 of *Lecture Notes in Computer Science*, pages 168–179. Springer Berlin Heidelberg, 2003.

[40] A. A. Freitas. Comprehensible classification models: a position paper. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 15(1):1–10, 2014.

[41] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 10 2001.

[42] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, March 2014.

[43] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Massachusetts Institute of Technology, 2016.

[44] Google PAIR. Facets. [Online]. Available: `https://pair-code.github.io/facets/`, 2017.

[45] D. Guo. Coordinating computational and visual approaches for interactive feature selection and multivariate clustering. *Information Visualization*, 2(4):232–246, Dec. 2003.

[46] N. Handly, D. A. Thompson, J. Li, D. M. Chuirazzi, and A. Venkat. Evaluation of a hospital admission prediction model adding coded chief complaint data using neural network methodology. *Eur J Emerg Med*, 22(2):87–91, Apr 2015.

[47] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman and Hall/CRC, June 1990.

[48] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Verlag, 2001.

[49] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[50] G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1):185–234, 1989.

[51] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[52] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers. *ArXiv e-prints*, Jan. 2018.

[53] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257, 1991.

[54] J. Hullman, E. Adar, and P. Shah. Benefitting infovis with visual difficulties. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2213–2222, Dec 2011.

[55] A. Jakulin, M. Možina, J. Demšar, I. Bratko, and B. Zupan. Nomograms for visualizing support vector machines. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 108–117, New York, NY, USA, 2005. ACM.

[56] P. B. Jensen, L. J. Jensen, and S. Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13(6):395–405, 2012.

[57] S. Johansson and J. Johansson. Interactive dimensionality reduction through user-defined combinations of quality metrics. *IEEE transactions on visualization and computer graphics*, 15(6):993–1000, 2009.

[58] D. C. M. Jr., E. A. de Oliveira, U. de Mendonça Braga Neto, R. F. Hashimoto, and R. M. C. Jr. Signal propagation in bayesian networks and its relationship

with intrinsically multivariate predictive variables. *Information Sciences*, 225:18–34, March 2013.

[59] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau. ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models. *ArXiv e-prints*, Apr. 2017.

[60] A. Kapoor, B. Lee, D. Tan, and E. Horvitz. Interactive optimization for steering machine classification. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1343–1352, New York, NY, USA, 2010. ACM.

[61] B. Kim, F. Doshi-Velez, and J. A. Shah. Mind the Gap: A generative approach to interpretable feature selection and extraction. In *Advances in Neural Information Processing Systems*, 2015.

[62] B. Kim, K. Patel, A. Rostamizadeh, and J. Shah. Scalable and interpretable data representation for high-dimensional complex data. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

[63] B. Kim, C. Rudin, and J. A. Shah. The Bayesian Case Model: A generative approach for case-based reasoning and prototype classification. In *Advances in Neural Information Processing Systems*, pages 1952–1960, 2014.

[64] J. Krause, A. Dasgupta, J.-D. Fekete, and E. Bertini. SeekAView: an intelligent dimensionality reduction strategy for navigating high-dimensional data spaces. *Large Data Analysis and Visualization (LDAV), IEEE Symposium on*, Oct 2016.

[65] J. Krause, A. Dasgupta, J. Swartz, Y. Aphinyanaphongs, and E. Bertini. A workflow for visual diagnostics of binary classifiers using instance-level explanations. *IEEE VAST*, 2017.

[66] J. Krause, A. Perer, and E. Bertini. INFUSE: interactive feature selection for predictive modeling of high dimensional data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1614–1623, Dec 2014.

[67] J. Krause, A. Perer, and E. Bertini. Using visual analytics to interpret predictive machine learning models. *ICML Workshop on Human Interpretability in Machine Learning (WHI)*, Jun 2016.

[68] J. Krause, A. Perer, and E. Bertini. Using Visual Analytics to Interpret Predictive Machine Learning Models. *ArXiv e-prints*, June 2016.

[69] J. Krause, A. Perer, and K. Ng. Interacting with predictions: Visual inspection of black-box machine learning models. *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 5686–5697, 2016.

[70] M. Kuhn and K. Johnson. *Applied Predictive Modeling*. Springer London, Limited, 2013.

[71] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, pages 126–137, New York, NY, USA, 2015. ACM.

[72] C. K. Leung and K. W. Joseph. Sports data mining: Predicting results for the college football games. *Procedia Computer Science*, 35:710–719, September

2014. Knowledge-Based and Intelligent Information & Engineering Systems 18th Annual Conference (KES), Proceedings of.

[73] M. Lichman and K. Bache. UCI machine learning repository, 2013.

[74] B. Y. Lim. *Improving Understanding and Trust with Intelligibility in Context-aware Applications*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2012. AAI3524680.

[75] B. Y. Lim and A. K. Dey. Assessing demand for intelligibility in context-aware applications. In *Proceedings of the 11th International Conference on Ubiquitous Computing*, UbiComp '09, pages 195–204, New York, NY, USA, 2009. ACM.

[76] Z. C. Lipton. The Mythos of Model Interpretability. *ArXiv e-prints*, June 2016.

[77] M. Liu, J. Shi, K. Cao, J. Zhu, and S. Liu. Analyzing the training processes of deep generative models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):77–87, Jan 2018.

[78] S. Liu, X. Wang, M. Liu, and J. Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 2017.

[79] S. Liu, J. Xiao, J. Liu, X. Wang, J. Wu, and J. Zhu. Visual diagnosis of tree boosting methods. *IEEE Trans. Vis. Comput. Graph.*, 24(1):163–173, 2018.

[80] Y. Lou, R. Caruana, and J. Gehrke. Intelligible models for classification and regression. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 150–158. ACM, 2012.

[81] V. F. Luis M. Candanedo. Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models. *Energy and Buildings. Vol. 112, p. 28–39*, 2016.

[82] D. Martens, B. Baesens, T. Van Gestel, and J. Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, 183(3):1466–1476, 2007.

[83] D. Martens and F. Provost. Explaining data-driven document classifications. *MIS Q.*, 38(1):73–100, Mar. 2014.

[84] T. May, A. Bannach, J. Davey, T. Ruppert, and J. Kohlhammer. Guiding feature subset selection with an interactive visualization. *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 111–120, Oct. 2011.

[85] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. *CoRR*, abs/1710.10777, 2017.

[86] S. Mishra, J. Diesner, J. Byrne, and E. Surbeck. Sentiment analysis with incremental human-in-the-loop learning and lexical resource customization. In *Proceedings of the 26th ACM Conference on Hypertext &#38; Social Media*, HT '15, pages 323–325, New York, NY, USA, 2015. ACM.

[87] T. Muhlbacher, L. Linhardt, T. Moller, and H. Piringer. Treepod: Sensitivity-aware selection of pareto-optimal decision trees. *IEEE Transactions on Visualization & Computer Graphics*, 24(1):174–183, Jan. 2018.

[88] T. Muhlbacher and H. Piringer. A partition-based framework for building and

validating regression models. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):1962–1971, 2013.

[89] M. Narayanan, E. Chen, J. He, B. Kim, S. Gershman, and F. Doshi-Velez. How do Humans Understand Explanations from Machine Learning Systems? An Evaluation of the Human-Interpretability of Explanation. *ArXiv e-prints*, Feb. 2018.

[90] K. Ng, A. Ghoting, S. R. Steinhubl, W. F. Stewart, B. Malin, and J. Sun. PARAMO: A PARAllel predictive MOdeling platform for healthcare analytic research using electronic health records. *Journal of Biomedical Informatics*, 2013.

[91] J. D. Olden and D. A. Jackson. Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling*, 154(12):135–150, 2002.

[92] K. Patel, N. Bancroft, S. Drucker, J. Fogarty, and J. Landay. 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)*, pages 37–46, New York, NY, 2010. ACM.

[93] K. Patel, S. M. Drucker, J. Fogarty, A. Kapoor, and D. S. Tan. Using multiple models to understand data. In T. Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1723–1728. IJCAI/AAAI, July 2011.

[94] A. Perer and B. Shneiderman. 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, pages 265–274, 2008.

[95] A. Perer and B. Shneiderman. Systematic yet flexible discovery: guiding domain experts through exploratory data analysis. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 109–118. ACM, 2008.

[96] N. Pezzotti, T. Hllt, J. V. Gemert, B. P. F. Lelieveldt, E. Eisemann, and A. Vilanova. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):98–108, Jan 2018.

[97] C. Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, pages 109–116. ACM, 2004.

[98] T. Plate, J. Bert, J. Grace, and P. Band. 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)*, 1:306–309, 1997.

[99] B. Poulin, R. Eisner, D. Szafron, P. Lu, R. Greiner, D. S. Wishart, A. Fyshe, B. Pearcy, C. MacDonell, and J. Anvik. Visual explanation of evidence in additive classifiers. In *Proceedings of the 18th Conference on Innovative Applications of Artificial Intelligence - Volume 2*, IAAI'06, pages 1822–1829. AAAI Press, 2006.

[100] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. Squares: Supporting

interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):61–70, 2017.

[101] P. Rheingans and M. desJardins. Visualizing high-dimensional predictive model quality. In *Visualization*, pages 493–496, Oct 2000.

[102] M. T. Ribeiro, S. Singh, and C. Guestrin. ”why should I trust you?”: Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016.

[103] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[104] J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. *Information Visualization*, 4(2):96–113, 2005.

[105] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IN IEEE SYMPOSIUM ON VISUAL LANGUAGES*, pages 336–343, 1996.

[106] B. Shneiderman and C. Plaisant. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Proceedings of the 2006 BELIV Workshop*, pages 1–7, 2006.

[107] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[108] P. Y. Simard, S. Amershi, D. M. Chickering, A. E. Pelton, S. Ghorashi, C. Meek, G. Ramos, J. Suh, J. Verwey, M. Wang, and J. Wernsing. Machine teaching: A new paradigm for building machine learning systems. *CoRR*, abs/1707.06742, 2017.

[109] C. A. Steed, P. J. Fitzpatrick, J. E. S. II, and T. Jankun-Kelly. Tropical cyclone trend analysis using enhanced parallel coordinates and statistical analytics. *Cartography and Geographic Information Science*, 36(3):251–265, 2009.

[110] C. A. Steed, J. E. S. II, T. Jankun-Kelly, and P. J. Fitzpatrick. Guided analysis of hurricane trends using statistical processes integrated with interactive parallel coordinates. In *Visual Analytics Science and Technology (VAST), IEEE Symposium on*, pages 19–26, Oct 2009.

[111] G. V. Steeg and A. Galstyan. Maximally informative hierarchical representations of high-dimensional data. In *AISTATS'15*, 2015.

[112] H. Strobelt, S. Gehrmann, B. Huber, H. Pfister, and A. M. Rush. Visual analysis of hidden state dynamics in recurrent neural networks. *CoRR*, abs/1606.07461, 2016.

[113] S. Stumpf, A. Bussone, and D. O'Sullivan. Explanations considered harmful? user interactions with machine learning systems. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2016.

[114] J. Talbot, B. Lee, A. Kapoor, and D. S. Tan. Ensemblematrix: Interactive visualization to support machine learning with multiple classifiers. In *Proceed-*

ings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09, pages 1283–1292, New York, NY, USA, 2009. ACM.

[115] G. K. Tam, V. Kothari, and M. Chen. An analysis of machine-and human-analytics in classification. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):71–80, 2017.

[116] P. Tamagnini, J. Krause, A. Dasgupta, and E. Bertini. Interpreting black-box classifiers using instance-level visual explanations. *Workshop on Human-In-the-Loop Data Analytics*, May 2017.

[117] S. T. Teoh and K.-L. Ma. PaintingClass: Interactive Construction, Visualization and Exploration of Decision Trees. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 667–672, New York, NY, USA, 2003. ACM.

[118] A. Treisman. Preattentive processing in vision. *Comput. Vision Graph. Image Process.*, 31(2):156–177, Aug 1985.

[119] M. Tulio Ribeiro, S. Singh, and C. Guestrin. Nothing Else Matters: Model-Agnostic Explanations By Identifying Prediction Invariance. *ArXiv e-prints*, Nov. 2016.

[120] C. Turkay, P. Filzmoser, and H. Hauser. Brushing dimensions–a dual visual analysis model for high-dimensional data. *IEEE transactions on visualization and computer graphics*, 17(12):2591–9, Dec. 2011.

[121] F.-Y. Tzeng and K.-L. Ma. Opening the black box - data driven visualization of neural networks. In *Proceedings of IEEE Visualization '05 Conference*, pages 383–390. IEEE, 2005.

[122] S. van den Elzen and J. J. van Wijk. BaobabView: Interactive construction and analysis of decision trees. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on*, pages 151–160. IEEE, 2011.

[123] A. Vellido, J. D. Martín-Guerrero, and P. J. Lisboa. Making machine learning models interpretable. In *Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, volume 12, pages 163–172, 2012.

[124] G. Ver Steeg and A. Galstyan. Discovering structure in high-dimensional data through correlation explanation. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 577–585. Curran Associates, Inc., 2014.

[125] R. Wachter. *The Digital Doctor*. McGraw-Hill Education, 2015.

[126] J. Wang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, pages 105–112. IEEE, 2003.

[127] M. Wattenberg, F. Viegas, and M. Hardt. Attacking discrimination with smarter machine learning. [Online]. Available: https://research.google.com/bigpicture/attacking-discrimination-in-ml/, 2016.

[128] L. Wilkinson, A. Anand, and R. L. Grossman. Graph-Theoretic Scagnostics. In *INFOVIS*, volume 5, page 21, 2005.

[129] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with

visual attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2048–2057, 2015.

[130] J. Yang, A. Patro, S. Huang, N. Mehta, M. O. Ward, and E. A. Rundensteiner. Value and Relation Display for Interactive Exploration of High Dimensional Datasets. In *IEEE Symposium on Information Visualization*, pages 73–80. IEEE Computer Society, 2004.

[131] J. Yang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, pages 105–112. IEEE, 2003.

[132] H.-F. Yu, F.-L. Huang, and C.-J. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Mach. Learn.*, 85(1-2):41–75, Oct. 2011.

[133] H. Zhang. The Optimality of Naive Bayes. In V. Barr and Z. Markov, editors, *FLAIRS Conference*. AAAI Press, 2004.

[134] J. Zhou, S. Z. Arshad, S. Luo, and F. Chen. Effects of uncertainty and cognitive load on user trust in predictive decision making. In *16th IFIP TC 13 International Conference on Human-Computer Interaction — INTERACT 2017 - Volume 10516*, pages 23–39, New York, NY, USA, 2017. Springer-Verlag New York, Inc.