A Workflow for Visual Diagnostics of Binary Classifiers using Instance-Level Explanations

Josua Krause, Aritra Dasgupta, Jordan Swartz, Yindalon Aphinyanaphongs, Enrico Bertini

Abstract—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 work is validated through 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 method helps experts derive useful knowledge about the model and the phenomena it describes and also generate useful hypotheses on how a model can be improved.

Index Terms—Machine Learning, Classification, Interpretation, Visual Analytics.

1 Introduction

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 [4, 8, 12, 21, 22, 28]. 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 [8].

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) [12]. 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 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 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 3. We then describe in detail the instance-level explanation algorithm we use in Section 4 and the interfaces we built in Section 5. Section 6 reports on a use case we built to show how the workflow can help perform useful and actionable model diagnostics. Section 7 discusses the results and provides a number of reflections and lessons we have learned from this

- *Josua Krause, is with NYU (josua.krause@nyu.edu)*
- *Aritra Dasgupta, is with PNNL (aritra.dasgupta@pnnl.gov)*
- *Jordan Swartz MD, is with NYU School of Medicine (jordan.swartz@nyumc.org)*
- *Yindalon Aphinyanaphongs MD, is with NYU School of Medicine (yin.a@nyumc.org)*
- *Enrico Bertini, is with NYU (enrico.bertini@nyu.edu)*

collaborative exercise.

2 Related Work

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

2.1 Model Explanations: *Why?* and *How?*

Explanations of behavior of autonomous systems [18] or computational models [11] 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 [26] 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 [19]. 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 [17]. 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.

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.*, infering rules from a neural network [10], or generating explanations [25, 16]. 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-agonstic and generalizable diagnostic interface for inpsecting model quality. In previous work, local explanations have been used to diagnose how models make decisions for single instances of a data set [15, 16, 25]. 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.

2.2 Human-in-the-Loop Inspection of Classification Models

Researchers have recently demonstrated how human interventions can help in greater accuracy in construction of classifiers, when compared with a purely automated approach [27]. 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.* [20]: i) model understanding, ii) model diagnosis, and iii) model refinement. Our proposed diagnostic workflow (Figure 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 [3] 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 [1] and other linked views to show probabilities of items belonging to different classes for multi-class classifiers. Squares [24] 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 [9] 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 [1, 3, 24] or single features [9], 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.

3 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 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.

3.1 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 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 1, which

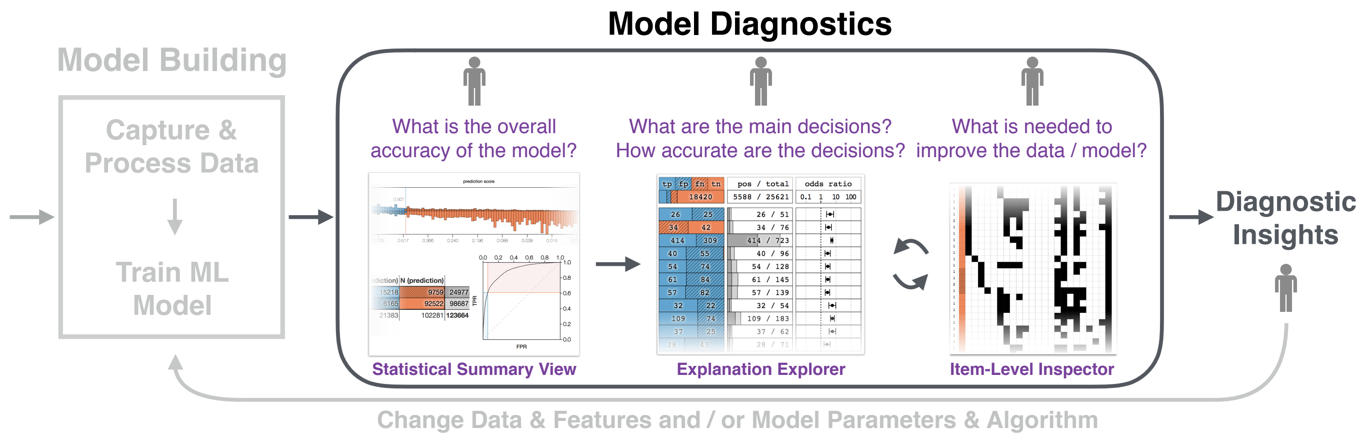

Model Diagnostics
Model Building
Capture & Process Data
Train ML Model
What is the overall accuracy of the model?
What are the main decisions? How accurate are the decisions?
What is needed to improve the data / model?
Statistical Summary View
Explanation Explorer
Item-Level Inspector
Diagnostic Insights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

Fig. 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. An explanation comprises of a set of features that need to be removed to change the label for a given item. 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.

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.

3.2 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 [22], 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 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 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**).

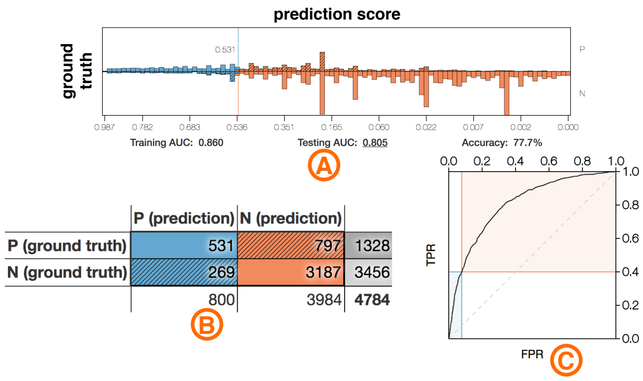
Instance-level. The third step focuses on the analysis of a single user-selected explanation and the instances it explains. Once an interesting explanation has been found in the previous step, it is often useful and necessary to drill-down to the individual instances to observe how the data items contained 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 Explanation Method

The goal of explanations is 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 what features were involved in its decision making process. For binary classifiers this is commonly done by finding the minimal amount of change necessary to change the prediction of the analyzed classifier. The process is performed on the trained model by creating synthetic input values derived from observed data items re-



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

Fig. 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. Selecting the peak left of 0.165 reveals "Sodium Chloride" as its main cause in the Explanation Explorer. (B) The confusion matrix shows the number of correct and incorrect predictions. (C) The ROC curve shows the prediction quality.

vealing 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 with a threshold t is typically used:

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

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 [16] and LIME [25] both propose algorithms that can be used to create explanations. The metric used for minimizing e depends on the explanation technique in use. 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 [23] 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 data items 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 data item 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 few data items each can be an indicator that the model is overfitting as it is trying to memorize labels of individual data items.

- 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 which removes 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 the 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.

5 Visual Interface

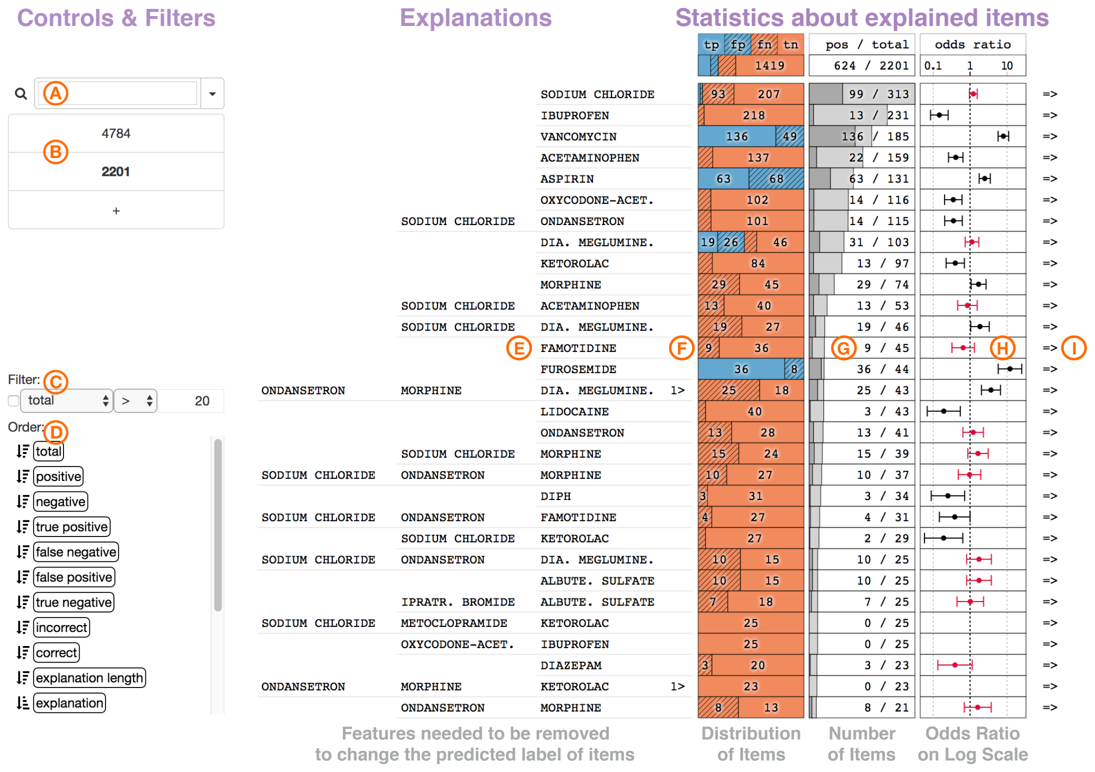
Our proposed user interface consists of three different panels each corresponding to the different goals of our proposed workflow that we described in Section 3. By interacting with each panel and navigating across these panels, experts can diagnose different aspects of model behavior.

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 the interfaces of each panel in the workflow's order: Statistical Summary View of the machine learning model, the Explanation Explorer, and the Item Level Inspector.

5.1 Statistical Summary View

The purpose of this panel (Figure 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 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,



				tp / fp / fn / tn	pos / total	
				1419	624 / 2201	
		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	=>

Fig. 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).

depict 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 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 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 or below 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.

5.2 Explanation Explorer

The second panel, the Explanation Explorer (Figure 3), addresses **G2** and **G3** by encoding a list of explanations based on the method we described in Section 4. 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 3E). In order to make this information quickly readable we only show up to three of potentially more features of the explanation. 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. If the explanation has more than three features an indicator with a number on the right shows the number of remaining features. 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 3F). Again, 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 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 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. 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:

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

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 3I) to inspect the explained subset more closely in the Item Level Inspector as described in Section 5.3.

The rows shown by the Explanation Explorer can be reordered as well as filtered. As shown in Figure 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 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 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 3C) allows to filter by quantities.

As shown in Figure 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.

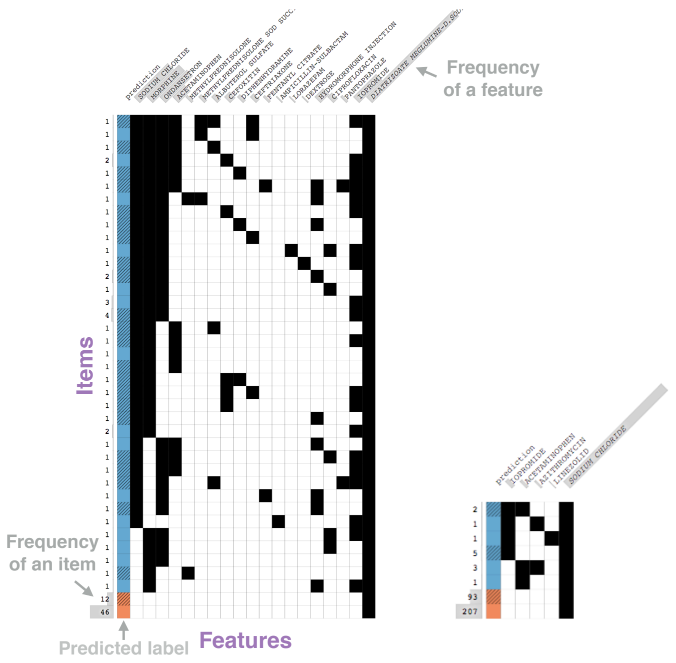
The metrics used for reordering are the same as those used for conditioning with the additional option of lexicographical sorting by using the feature names of the explanations. Common metrics to use for sorting or reordering, besides total amount of items, are "uncertainty" and "odds ratio". "Uncertainty" (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.

5.3 Item Level Inspector

The third panel (Figure 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, however, 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. Also, 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



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

Fig. 4: The **Item Level Inspector** showing a matrix of data items as rows against features as columns for the explanations "Diatrizoate Meglumine" and "Sodium Chloride" in the initial dataset of the case study (Section 6). Rows group identical data items together and shows the count on the left side. Features are sorted by "relative feature importance" showing from left to right how labels can be separated.

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 the exact same feature vectors but with different labels that cannot be separated exists as well.

As some features have no discriminating impact with regards to the "relative feature importance" they can be ignored by deactivating the "show all features" check box at the top left in order to simplify the matrix view.

6 Case Study

The proposed workflow and user interface described above stem from a one year collaboration with a machine learning expert and a medical doctor from the *NYU Langone Medical Center*. 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 (patients sent to further observation are included here as they follow a different path through the hospital). 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 to be 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 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.*, arrival mode, triage score) [5, 7, 13].

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 interface and workflow we describe in the paper result from numerous iterations over the methods used to derive information from the model and 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.

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.

The table below shows a summary of the models we trained and their performance.

Model	Training	Test AUC
Gaussian Naïve Bayes (GNB) [30]	0.58	0.52
Logistic Regression (LR) [29]	0.85	0.79
Random Forest (RF) [6]	0.88	0.79
Multi Layer Perceptron (MLP) [14]	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.

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 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 and, as part of a medication order 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.

When we look at the Item Level Inspector for this explanation (Figure 4b) we can in fact see that hospital admission is the predicted outcome when "Sodium Chloride" appears in the context of other drugs and that the patient is sent home when this is the only medication the patient received.

When we look 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.

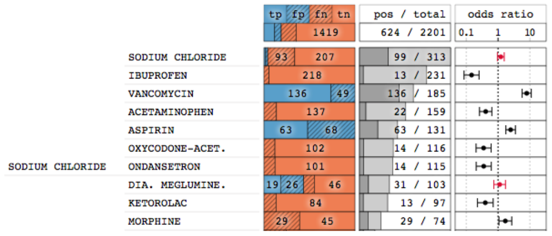
Another common explanation is "Ibuprofen" a pain relieving drug. It is predictive for non-admissions which is likely due to patients with pain symptoms that turned out to be benign. The odds ratio indicates a significant relation to the outcome. On the other hand "Vancomycin", an antibiotic used for 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 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 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 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 5d) shows explanations whose predictions are not significant. This is often the case when it is impossible to correctly predict a given set of identical data items that have contradicting ground truth labels.

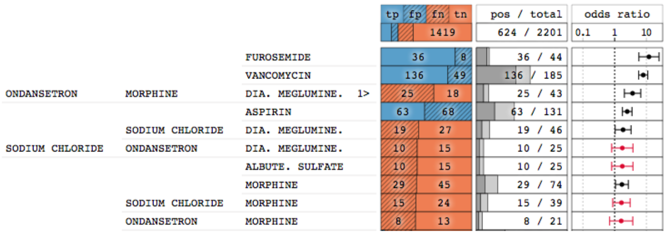
The first two explanations "Ipratropium Bromide", "Albuterol Sulfate" (medication for treating chronic obstructive pulmonary disease and asthma, lung diseases that can have chronic and acute symptoms the latter of which requires immediate attention) and "Sodium Chloride", "Ondansetron", "Morphine" are 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



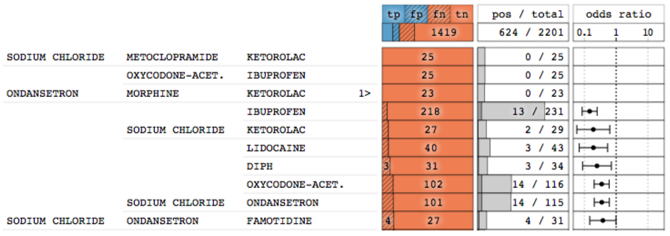
			tp fp fn tn	pos / total	odds ratio
			1419	624 / 2201	0.1 1 10
	SODIUM CHLORIDE		93 207	99 / 313	
	IBUPROFEN		218	13 / 231	
	VANCOMYCIN		136 49	136 / 185	
	ACETAMINOPHEN		137	22 / 159	
	ASPIRIN		63 68	63 / 131	
	OXYCODONE-ACET.		102	14 / 116	
SODIUM CHLORIDE	ONDANSETRON		101	14 / 115	
	DIA. MEGLUMINE.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

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



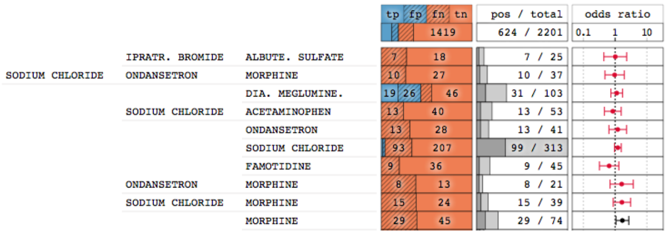
				tp fp fn tn	pos / total	odds ratio
				1419	624 / 2201	0.1 1 10
		FUROSEMIDE		36 8	36 / 44	
		VANCOMYCIN		136 49	136 / 185	
ONDANSETRON	MORPHINE	DIA. MEGLUMINE.	1>	25 18	25 / 43	
		ASPIRIN		63 68	63 / 131	
	SODIUM CHLORIDE	DIA. MEGLUMINE.		19 27	19 / 46	
SODIUM CHLORIDE	ONDANSETRON	DIA. MEGLUMINE.		10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

Fig. 5: Showing different orders in the **Explanation Explorer** for addressing the goals (G2 & G3) in the case study (Section 6). The initial dataset is filtered for explanations with > 20 data items.

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 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 with "Ipratropium Bromide" and "Albuterol Sulfate" or "Diatrizoate Meglumine" more information is needed. One option is to include information about the final diagnosis which has both benefits and drawbacks. On one hand it likely improves the overall quality of the prediction. On the other hand it 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. For including the diagnoses 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.

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.

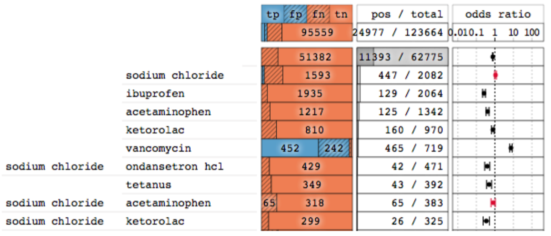
Again we are focusing solely on the Multi Layer Perceptron model for further analyses. In order to compare our new data to the previous dataset we first created models that do not utilize the newly added diagnoses. However, the resulting AUC is much lower than for the initial data. Looking at the Statistical Summary View reveals a strong concentration of data points at a specific prediction score. Focusing on this prediction score in the Explanation Explorer (Figure 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 (∼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 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.

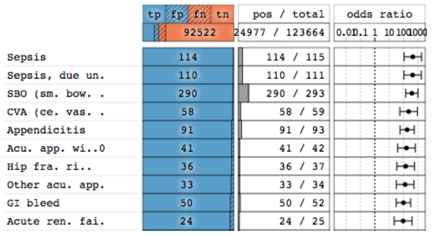
Diagnostic Insights. By adding diagnoses to the dataset a strong increase in predictive quality was achieved. However, seeing that diagnoses effectively replace medication in their predictive power suggests that the "labels are leaking". That is, since doctors make the decision of whether to admit a 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 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. By knowing its strengths and weaknesses it can be applied by detecting and deciding confident cases early while letting ambiguous cases undecided until a doctor makes the decision. This demonstrates that a statistically weaker model can be more useful in practice.

7 Discussion

Through our case study of patient visits, we have shown that by aggregating model decisions through explanations, we are able to make



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



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

Fig. 6: Showing the second dataset of the case study (Section 6) with and without using diagnoses features in the **Explanation Explorer**.

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 6 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.

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 that is separated from and does not interfere with the existing modeling pipeline. Looking at Figure 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 [2], 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 leverage more the power of modeling for exploratory data analysis and data sense making. While many visualization systems focus on direct visualization of raw data as an 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.

8 Limitations & Future Work

We would like to point out some important limitations of our current work. First, the workflow and interfaces 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 cases with more than two classes is also not trivial. Second, 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. Third, our solution works best with analysing 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.

On a concluding note, we demonstrated how visual explanations can be effectively leveraged by medical experts for diagnosing model decisions and for ultimately making informed judgment about associations among medications and patients’ diagnoses. In our future work we plan to focus on extending our method to non-binary data and multi-class problems. At the same time, we will work on a new solution to address the model comparison problem.

References

[1] 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. 2

[2] 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. 9

[3] 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 Conference on Human Factors in Computing Systems (CHI '15)*, pages 337–346, April 2015. 2

[4] 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. 1

[5] 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. 7

[6] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001. 7

[7] 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. 7

[8] 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 *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1721–1730. ACM, 2015. 1

[9] 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, 2011. 2

[10] M. W. Craven and J. W. Shavlik. Using neural networks for data mining, 1998. 2

[11] 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. 2

[12] A. A. Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations*, 15(1):1–10, 2014. 1

[13] 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. 7

[14] 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. 7

[15] J. Krause, A. Perer, and E. Bertini. INFUSE: interactive feature selection for predictive modeling of high dimensional data. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):1614–1623, Dec 2014. 2

[16] J. Krause, A. Perer, and K. Ng. Interacting with predictions: Visual inspection of black-box machine learning models. *CHI*, pages 5686–5697, 2016. 2, 4

[17] 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. 2

[18] 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. 2

[19] Z. C. Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016. 2

[20] S. Liu, X. Wang, M. Liu, and J. Zhu. Towards better analysis of machine learning models: A visual analytics perspective. *Visual Informatics*, 2017. 2

[21] Y. Lou, R. Caruana, and J. Gehrke. Intelligible models for classification and regression. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 150–158. ACM, 2012. 1

[22] 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. 1, 3

[23] D. Martens and F. Provost. Explaining data-driven document classifications. *MIS Q.*, 38(1):73–100, Mar. 2014. 4

[24] 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. 2

[25] M. T. Ribeiro, S. Singh, and C. Guestrin. ”why should I trust you?”: Explaining the predictions of any classifier. *CoRR*, abs/1602.04938, 2016. 2, 4

[26] S. Stumpf, A. Bussone, and D. O'Sullivan. Explanations considered harmful? user interactions with machine learning systems. In *Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2016. 2

[27] 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. 2

[28] A. Vellido, J. D. Martín-Guerrero, and P. J. Lisboa. Making machine learning models interpretable. In *Proc. of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, volume 12, pages 163–172, 2012. 1

[29] 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. 7

[30] H. Zhang. The Optimality of Naive Bayes. In V. Barr and Z. Markov, editors, *FLAIRS Conference*. AAAI Press, 2004. 7