SAPIENZA
UNIVERSITÀ DI ROMA

Visual Analytics for Machine Learning: Interpreting Black-Box Classifiers through Instance-Level Explanations

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Data Science

Candidate

Paolo Tamagnini
ID number 1536242

Thesis Advisor
Prof. Aris Anagnostopoulos

External Advisor
Prof. Enrico Bertini

Academic Year 2016/2017

The work for this thesis is concurrent to the paper "Interpreting Black-Box Classifiers Using Instance-Level Visual Explanations" published at Workshop on Human-In-the-Loop Data Analytics (HILDA) - May 2017 - SIGMOD - Chicago.

Thesis defended at Sapienza University of Rome in October 2017

Visual Analytics for Machine Learning: Interpreting Black-Box Classifiers through Instance-Level Explanations
Master thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: September 25, 2017

Author's email: paolotamag@gmail.com

Abstract

To realize the full potential of machine learning in diverse real-world domains, it is necessary for model predictions to be readily interpretable and actionable for the human in the loop. Analysts, who are the users but not the developers of machine learning models, often do not trust a model because of the lack of transparency in associating predictions with the underlying data space. To address this problem, we propose *Rivelo*, a visual analytics interface that enables analysts to understand the causes behind predictions of binary classifiers by interactively exploring a set of instance-level explanations. These explanations are model-agnostic, treating a model as a black box, and they help analysts in interactively probing the high-dimensional binary data space for detecting features relevant to predictions. We demonstrate the utility of the interface with a case study analyzing a random forest model on the sentiment of Yelp reviews about doctors.

Acknowledgments

I would like to thank my family, which supported me in those years of studies. Furthermore I would like to thank Prof. Enrico Bertini to welcome me in his research group at NYU Tandon School of Engineering. Working as an intern, I was given the possibility to code the software Rivelo and to write the relative paper, which are the core of this thesis. I also would like to thank Prof. Leonardi and Prof. Anagnostopoulos for offering me the opportunity to study for my thesis abroad. I learned so much during this past year thanks to this wonderful experience and I am looking forward to what will come next.

Contents

Chapter 1

Introduction

This thesis is about the intersection of two different fields within the broad discipline of data science. Those fields are machine learning and visual analytics. Visual analytics comprehends a set of techniques designed to analyze large amount of data by means of visual representations and interactive interfaces. The field of machine learning, instead, regards the design and developing of mathematical models. Those models are used to train a computer with large amount of data in order to perform certain tasks. Through machine learning a computer autonomously looks within the data in order to find patterns. By learning those patterns, the computer will be able to make predictions of unknown variables where the data is incomplete.

Given this framework computers will often be able to succeed more efficiently than humans in predicting the future, for example when foreseeing election outcomes, or in classifying items, like when recognizing handwritten digits in images. Despite this, computers fail in explaining how and why they performed such tasks. In order to trust and improve the machine learning model, the human would like to have total control over the process of performing those tasks. Through user interactions and visual representations, visual analytics can provide useful insights of machine learning autonomous decisions.

In this thesis we present a workflow and a visual interface to help domain experts and machine learning developers explore and understand binary classifiers. The main motivation is the need to develop methods that permit people to inspect what decisions a model makes after it has been trained.

While solid statistical methods exist to verify the performance of a model in an aggregated fashion, typically in terms of accuracy over hold-out data sets, there is a lack of established methods to help analysts interpret the model over specific sets of instances.

This kind of activity is crucial in situations in which assessing the *semantic validity* of a model is a strong requirement. In some domains where human trust in the model is an important aspect (e.g., healthcare, justice, security), verifying the model exclusively through the lens of statistical accuracy is often not sufficient [4, 7, 9, 17, 18, 33]. This need is exemplified by the following quote coming from the recent DARPA XAI program: "*the effectiveness of these systems is limited by the machines current inability to explain their decisions and actions to human users [...] it is essential to understand, appropriately trust, and effectively manage an emerging generation of artificially intelligent machine partners*" [10].

Furthermore, being able to inspect a model and observe its decisions over a data set has the potential to help analysts better understand the data and ultimately the phenomenon it describes.

Unfortunately, the existing statistical procedures used for model validation do not communicate this type of information and no established methodology exists.

In practice, this problem is often addressed using one or more of the following strategies: (1) build a more interpretable model in the first place even if it reduces performance (typically decision trees or logistic regression); (2) calculate the importance of the features used by a model to get a sense of how it makes decisions; (3) verify how the model behaves (that is, what output it generates) when fed with a known set of relevant cases one wants to test.

All of these solutions however have major shortcomings. Building more interpretable models is often not possible unless one is ready to accept relevant reductions in model performance. Furthermore, and probably less obvious, many models that are considered *interpretable* can still generate complicated structures that are by no means easy to inspect and interpret by a human being (e.g., decision trees with a large set of nodes) [9]. Methods that rely on calculation of feature importance, e.g., weights of a linear classifier, report only on the *global* importance of the features and does not tell much about how the classifier makes decisions in particular cases. Finally, manual inspection of specific cases works only with a very small set of data items and does not assure a more holistic analysis of the model.

To address all of these issues we propose a solution that provides the following benefits:

1. Requires only to be able to observe the input/output behavior of a model and as such it can be applied to any existing model without having access to its internal structure.

2. Captures decisions and feature importance at a local level, that is, at the level of single instances, while enabling the user to obtain an holistic view of the model.

3. It can be used by domain experts with little knowledge of machine learning.

The solution we propose leverages *instance-level explanations*: techniques that compute feature importance locally, for a single data item at a time. For instance, in a text classifier such techniques produce the set of words the classifier uses to make a decisions for a specific document. The explanations are then processed and aggregated to generate an interactive workflow that enables the inspection and understanding of the model both *locally* and *globally*.

In Chapter 6, we will describe the workflow (Figure 6.1) in detail which consists of the following steps. The system generates one explanation for each data item contained in the data set and creates a list of features ranked according to how frequently they appear in the explanations. Once the explanations and the ranked list are generated, the user can interact with the results as follows: (1) the user selects one or more features to focus on specific decisions made with them; (2) the system displays the data items explained by the selected features together with information about their labels and correctness; (3) the user inspects them and selects specific instances to compare in more detail; (4) the system provides a visual representation of the descriptors / vectors that represent the selected data items (e.g., words used as descriptors in a document collection) and permits to visually compare them; (5) the user can select one or more of the descriptors / vectors to get access to the raw data if necessary (e.g., the actual text of a document).

In the following chapters we first provide background information on visual analytics and other related work, we then describe the methods used to generate the explanations followed by a more detailed description of the visual user interface and interactions developed to realize the workflow. Then, we provide a small use case to show how the tool works in practice. Furthermore we also describe an indicative user study to show some evidence that different users were actually able to extract useful information from our tool. Finally we conclude with information on the existing limitations and how we intend to extend the work in the future.

Chapter 2

Visual Analytics

In order to better understand our work, it is useful to give a brief overview of what visual analytics is and how it works. Visual analytics is a multi-disciplinary field that relates to the designing of tools and techniques which help explore and comprehend large data-sets. Those techniques mostly alternate automated analysis and interactive visualization to help the user solve problems with the available data. Such problems regards tasks like decision making and reasoning through visual interaction over remarkable amounts of information, which would be overwhelming for the user otherwise. Mostly this implies detecting unexpected and meaningful anomalies and patterns within the data from on demand visualization of a model results.

2.1 The Visual Analytics Process

One of the oldest paper on the topic is from Shneiderman (1996) [28]. Shneiderman foresees the evolution of information visualization methods in the following decades by already making a distinction between two approaches of dealing with data. The first relates to data management of relational databases, where everything is already structured and the aim is the efficiency and security of the safe-keeping of data. The second is relative to the search for patterns in large unstructured ever-growing collection of data, today called *big data*. In the second case as the large amount of data increases, it is more difficult to extract useful information. Therefore, in order to successfully interact with data, Shneiderman suggests the need for new methods related to the human perceptual ability. A user is able in fact to perceive and understand large amount of information through images rather than with plain text. Given all of this, Shneiderman describes the following scheme to apply when dealing with interactive visualization of large amount of data:

"*Overview first, zoom and filter, then details-on-demand.*"

Shneiderman calls this the *information visualization mantra* as the more this process is repeated, the more refined the extracted information will be. ***Overview*** refers to get an overview of the entire data-set. ***Zoom*** means to gather data items of interest and ***filter*** to leave out the ones not of interest. Then ***details-on-demand*** is the last crucial step where the user inspects the data items of interest by retrieving more information about them to detect something useful.
Since Shneiderman provided this approach in 1996, many other refinements have been added to the process. As data get visualized and processed with newer techniques, also new technologies provide extreme dynamic interaction with the visualizations. All of those techniques and technologies today are part of visual analytics.
One of the first definition for visual analytics is from a paper from the Pacific National Northwest Laboratory (PNNL) of 2006. Thomas and Cook [30] report the description of this new field given by the National Visualization and Analytics Center (NVAC). Such institution was established by the Department of Homeland Security (DHS) in order to process large amount of data and reduce the risk of terrorism in the United States. NVAC created a panel of about 40 researchers which conveyed the need of a new technology to prevent and control emergencies: visual analytics. The panel of researchers defined visual analytics: "*the science of analytical reasoning facilitated by interactive visual interfaces*". This science is related to four sets of techniques:

1. ***analytical reasoning techniques*** aiming at planning and decision making;
2. ***visual representations and interaction techniques*** which exploit the human ability of understanding large amount of data through images;
3. ***data representations and transformation techniques*** which alter the raw data so that can be used by the precedent techniques sets 1 and 2;
4. ***techniques for sharing and communicate*** the obtained analytical results.

Visual analytics is about the coexistence of those 4 sets of techniques in new software tools able to help the user in solving a problem.

2.2 Application of Visual Analytics

In order to better understand what visual analytics can do, we will see now its application in tools. A great number of different tools already exists, differing by domain data used, way of processing data and visualization used. What really differs from tool to tool is the problem they were made to solve. We can see in Figure 2.1 some pictures of tools taken from the book *Visualization Analysis and Design* by Tamara Muzner [21]. We will see now what those tool are able to do:

(a) *Scagnostics* [35] is used for understanding better scatter plots matrices (SPLOM). The tool shows a SPLOM visualization made to understand better another SPLOM visualization of actual data. *Scagnostics* shows a data item for each original scatter plot and a single *Scagnostics* scatter plot is used to compare scatter-plot attributes like *convex layout* vs *sparse layout.*

(b) *VisDB* [12] shows an intricate visualization that summarize the data items in a large database. The visualization will color to show how query impacts the database regions.

(c) *Hierarchical Clustering Explorer (HCE)* [26, 27] is used by bioligists to understand similarities among a great number of genes under different experiment condition using a hierarchical clustering.

(d) *PivotGraph* [34] can summarize in a single graph a larger network, where group of nodes and links are merged in single nodes and links of the visualization. This way it is possible to understand quickly the network topology.

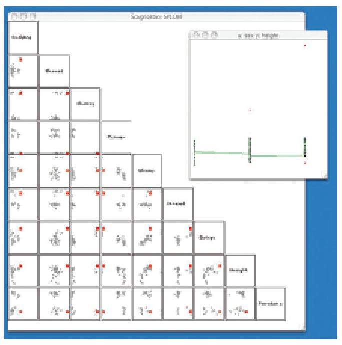
(e) *InterRing* [36] uses a visualization with a radial layout in order to explore data with tree structure.

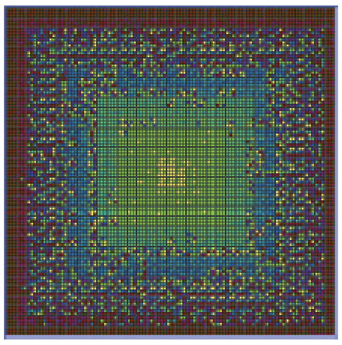
(f) *Constellation* [20] is made for browsing multilevel linguistic networks. Those networks are made to link a single word to its completely different meanings (e.g. "*bank*" with "*financial bank*" and "*river bank*"). It is in fact using a visualization with a graph connecting words on different linguistic layers.

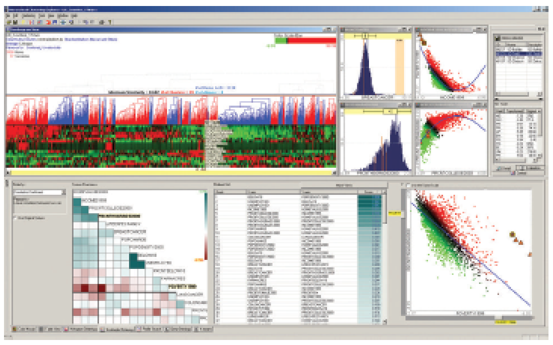
We will know go through *Hierarchical Clustering Explorer (HCE)* interface in detail to see an example more accurately.

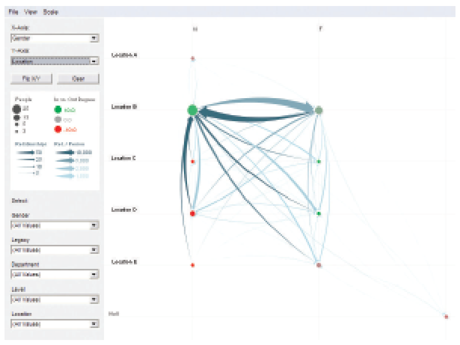
2.2.1 Hierarchical Clustering Explorer (HCE)

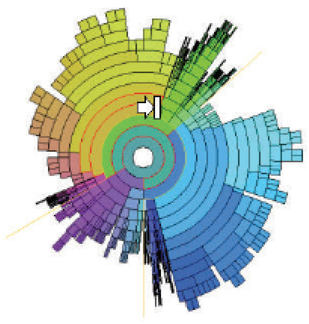
HCE was designed in 2002 by Seo and Shneiderman [26, 27], who we already met introducing visual analytics in Section 2.1. The tool aims at helping bioinformatic researchers at exploring DNA data similarities. The data displayed in the tool comes from DNA microarrays, small chips made of glass on which a robotic device has printed DNA samples in a 2-dimensional array disposition. The sample refers to different genes with various experiment conditions for which the *gene expression level* value was measured. Using this value, bioinformatic researchers compute similarities among different genes. Those similarities measures are used for hierarchical clustering,

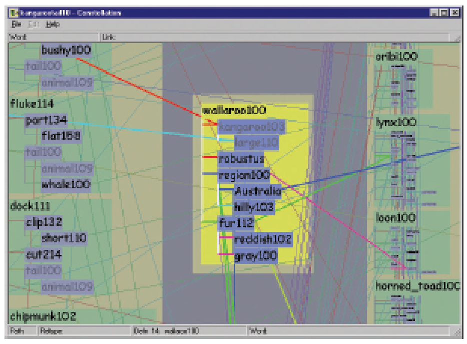

(a)


(b)


(c)


(d)


(e)



(f)

Figure 2.1. Six examples of visual analytics tools interfaces are reported in this picture. Different visualizations took place depending on the domain data and the problem the tools were made to solve. (a) *Scagnostics* [35]. (b) *VisDB* [12]. (c) *Hierarchical Clustering Explorer* [26, 27]. (d) *PivotGraph* [34]. (e) *InterRing* [36]. (f) *Constellation* [20].

represented by a dendogram. A dendogram is a diagram with a tree structure that branch together genes, clusters and sub-clusters given the similarity measures. Unfortunately, with this enormous amount of data, displaying the dendogram in a static view is not possible, since it would not even fit in a computer display and the intricate visualization would overwhelm the user. Therefore the researchers need *HCE*, a visual analytics tool for exploring their research experiment data in order to detect important relations among genes.

The interface of *HCE* is depicted in Figure 2.2 and it is split in two panels. We will call the top panel *overview panel* and the bottom panel the *zoom panel.* The *overview panel* is showing a view over the entire data-set, depicted as a mosaic clustered by a dendogram. The *zoom panel* shows instead the mosaic zoomed on a selected cluster, highlighted in yellow.

The *overview panel* can be zoomed in and out in search of an interesting cluster. The dendogram divides the genes on the x-axis in clusters given a similarity parameter threshold. On the bottom part of the *overview panel* the colored mosaic shows for each gene the *gene expression level* for different experiment conditions. Each gene is linked to a vertical stripe of tiles of the mosaic. Each tile of the same vertical stripe is relative to a different experimental condition where the color resemble the *gene expression level* from green to red. Clusters are visible in the mosaic thanks to white lines diving the vertical stripes (genes) into groups. The user can select one of those cluster which will appear in the *zoom panel.*

The *zoom panel* shows the same mosaic of the *overview panel* zoomed on the selected cluster. This way the user can read gene names on the top x-axis and the experimental conditions on the left y-axis. Experimental conditions are also clustered here by a different dendogram.

Thanks to this tool it is possible for a bionformatic researcher to practically explore the DNA-microarrays data with the work-flow defined by Shneiderman's *information visualization mantra* seen in Section 2.1. The researcher in fact will be **overviewing** the entire data-set and the clustering, **zooming** and **filtering** the clusters by changing the similarity parameter threshold, requesting **on-demand** visualization of a cluster **details**. This way the researcher can get an understanding of genes similarities on different experiment condition.

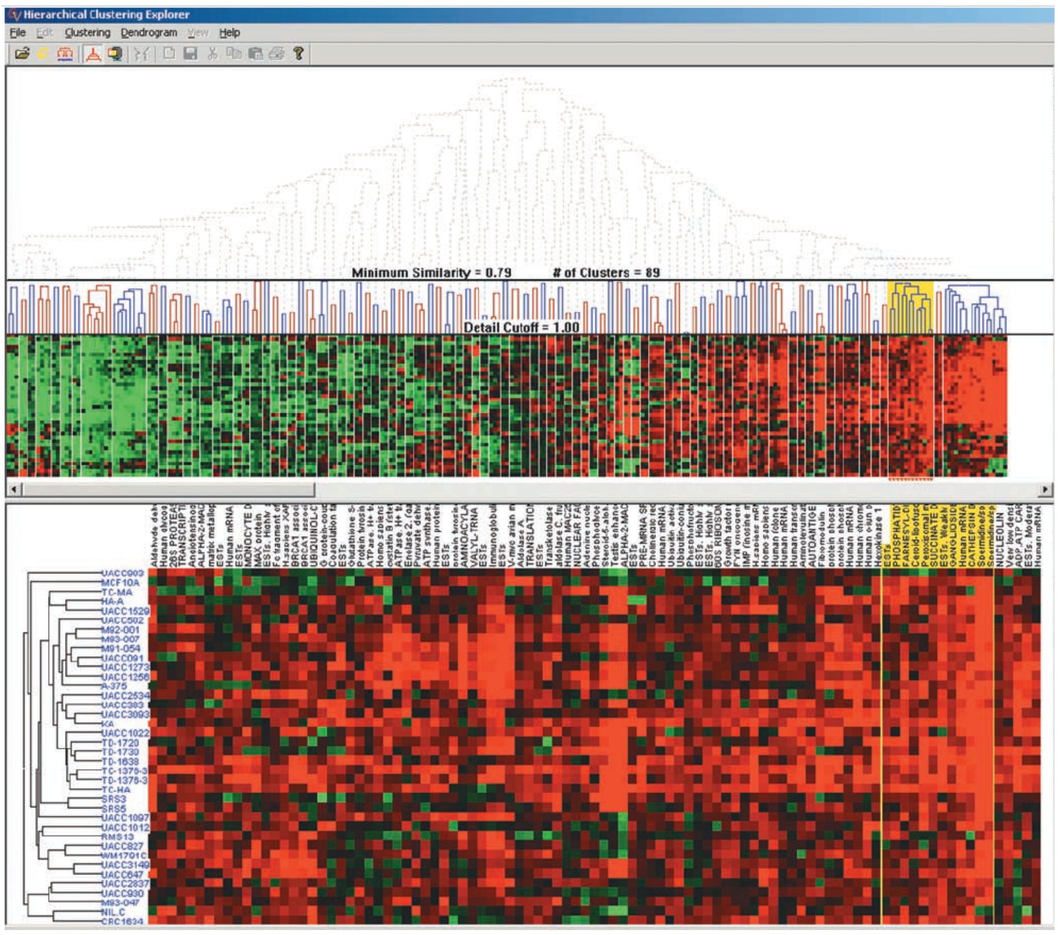


Figure 2.2. We show here an example of application of visual analytics: the interface of *Hierarchical Clustering Explorer (HCE)*. The *overview panel* on the top shows an outline of the clustering of genes. The genes are represented in the colored mosaic by vertical stripes linked to leaves of the dendogram. The vertical stripes keep record of all the different experiment conditions for the same gene. Each tuple (gene - experiment condition) is then represented by a tile of the mosaic with color linked to the *gene expression level* of the relative sample. Genes are divided in clusters by white separations between vertical stripes. The user can select one of those clusters which will be highlighted in yellow and showed in the below *zoomed panel*. In this panel it is possible to read gene names and experiment conditions for the samples of the selected cluster by enlarging the mosaic.

Chapter 3

Related Work

We have seen how visual analytics works for a broad variety of applications. We will see now tools that are specifically made to interpret machine learning models.

Different visual analytics techniques and tools have been developed to inspect the decision-making process of machine learning models and several authors advocated for the need to make models more interpretable [14]. In this chapter, we describe the previous work that is most relevant to our proposed model explanation process.

Those interactive visualizations depend on the required specifications of the explained model. Such specifications can be broad or precise and are depending on the kind of algorithm, input data, and classification used. In each of the related works, we will focus on the visual representation used to explain the model. This representation will depend from other properties of the tool.

Given this framework we made the following categories to distinguish group of tools related to machine learning. Since categories might intersect, the idea is not to give each tool a unique association, but we want instead to highlight their important characteristics. This way we can give a complete overview of all the previous work on the topic.

1. **Visualizing Multi-classification Results**:
   Many machine learning models are designed to classify items not just over two categories, but over multiple categories. In these cases, tools comprehend a visualization which supports all those multiple classification outcomes.

2. **Exploring Predicted Instances**:
   Another approach is to compare and analyze instances in their feature space. Those kind of visualizations help the user to compare instances by the feature relations and predicted labels, while keeping track of model performance.

3. **Use of Feature Importance**:
   An important method in detecting model behaviours is measuring the feature

importance. By visualizing and highlighting features by means of this measure, some tools suggest the user which features are most important for the trained classifier.

4. **Model Dependency in Visualizations**:
   A big differences among tools regard their flexibility towards the different kind of machine learning algorithms. By using a *white-box* approach, some tools are tailored exactly for a specific algorithm to show in detail what happens within the model. Other tools, like ours, use instead a *black-box* approach, which make them more flexible to explain insights from many different models.

5. **Human Interaction Involvement**:
   Another group of existing tool uses the user interaction in an active manner, so that the human is involved in the machine learning process. The user will continuously provide feedback by interacting with a visual interface and the machine learning model will comprehend the human decision in the classification process.

Each of these categories and the relative examples will be discussed in the next sections.

3.1 Visualizing Multi-classification Results

Different techniques are specialized with visualization for multi-classification models. An example is the method described by Rauber *et al.* [23] explaining multi-layer perceptron networks by creating patterns of instances through dimensionality reduction (Figure 3.1). Those patterns are useful to distinguish instances belonging to different classes. This technique would not be a valid replacement for *Rivelo* when handling multi-classification models because it only works with certain types of network models. Another tool able to explain multiclass model is called *Squares* [24]. The related paper shows an example with hand-written digit recognition model. However by using the aggregated representation of predicted instances, *Squares* is able to work with different kind of models as well. As shown in Figure 3.2, the predicted instances are depicted as cells in multiple histograms, one for each class. By interacting with the interface, the user is able to inspect a class performance by looking at a single histogram and a instance misclassification by looking at a single cell within an histogram.

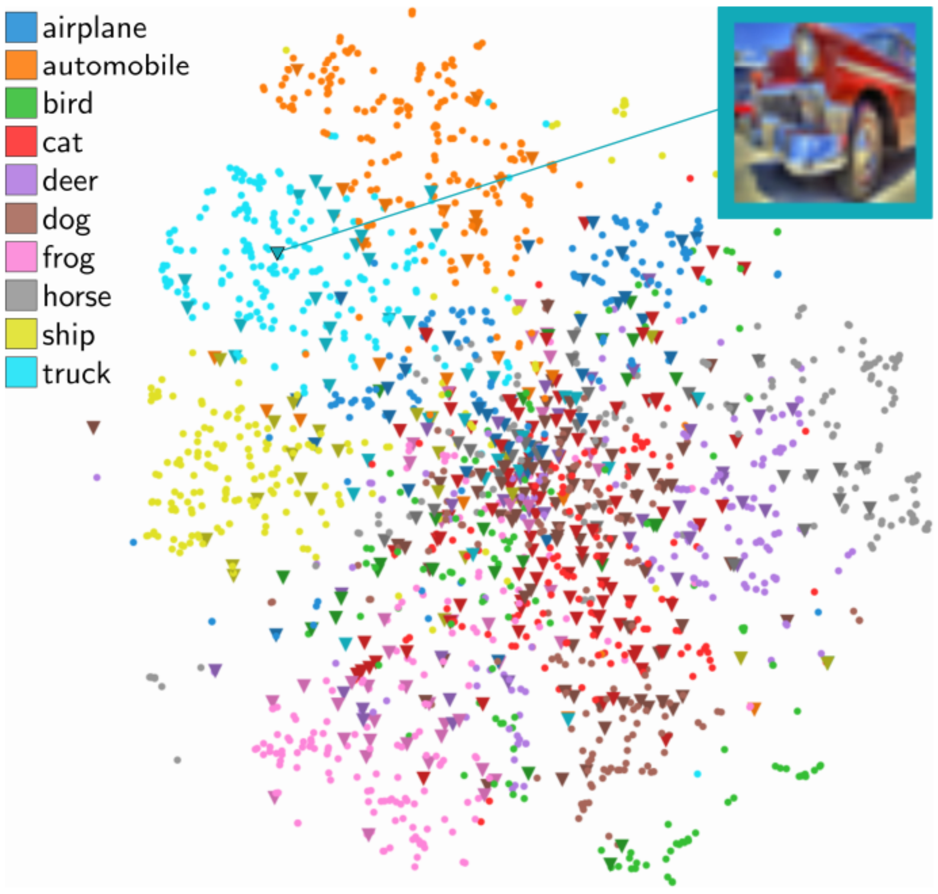


Figure 3.1. In the figure the dimensionality reduction designed by Rauber *et al.* [23] to visualize multi-classification results from a neural network model. In this case the results regard image classifications. Different clusters in the visualization have the same color and match with one of the categories of the image classification.

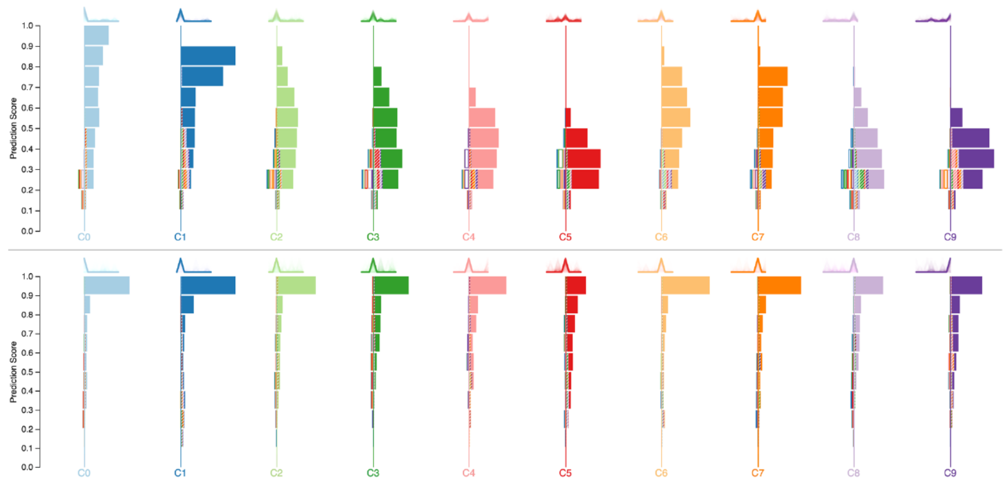


Figure 3.2. We display in this figure *Squares* [24], a visual analytics tool made for analyzing multi-classification models. The visualization depicts the predictions for digits-recognition from two models, one for each row of histograms. Each histogram on the same row has a distinctive color relative to the prediction for a single class, in this case one of the 9 digits.

3.2 Exploring Predicted Instances

Compared to tools for multi-classification, the ones for binary classification deal with less complex output, thus making it possible for visualizations to focus primarily on analyzing the model. For example *ModelTracker* [1] shows predicted instances in a more detailed interactive visualization than *Squares*. As shown in Figure 3.3, the user is able to explore the instance relations, visually represented in cells in the feature space, in order to understand the errors. Common performance statistics are displayed as well. Furthermore *ModelTracker* [1] visualizes predictions and their correctness and how these change when some model parameters are updated. Similarly, *MLCube Explorer* [11] (Figure 3.4) helps users compare model outcomes over various subsets and across multiple models with a data cube analysis type of approach. One main difference between these methods and the one we propose is that we do not base our analysis exclusively on model output, but also use the intermediary representation provided by explanations. This enables us to derive more specific information about *how* a model makes some decisions, and go beyond exploring *what* decisions it makes.

3.3 Use of Feature Importance

Another approach used to explain models is related to feature engineering. *FeatureInsight* [6] (Figure 3.5) leverages both feature importance and human interaction to

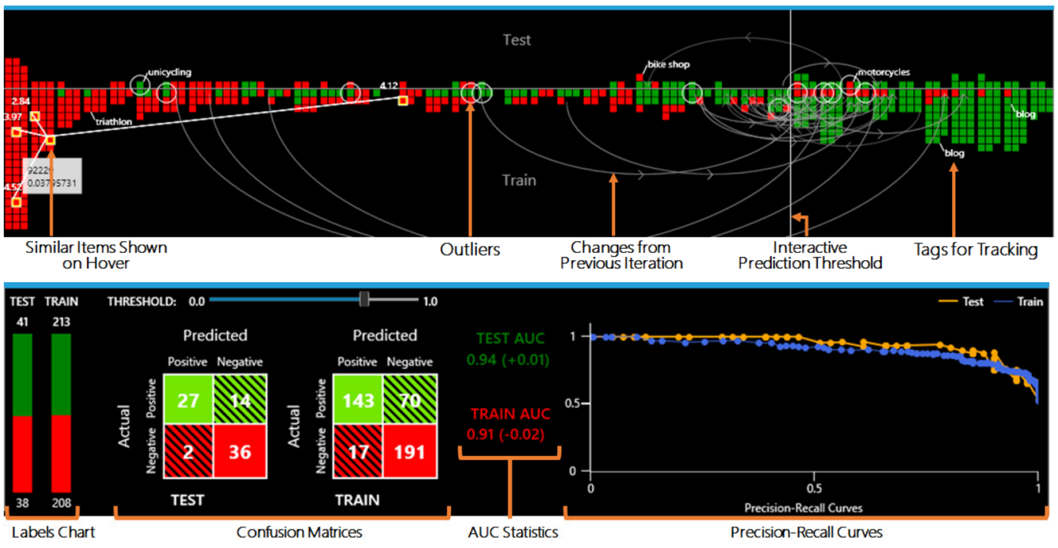


Figure 3.3. The above figure depicts the interface of *ModelTracker* [1]. In the top panel information regarding predicted instances is represented by the position and color of those cells. Given a cell, its x-position and y-position encode respectively the prediction value and which set the instance belongs to (train or test). The color represents instead the ground truth label. By clicking on a single cell, *ModelTracker* will highlight neighbour instances in the feature space. A vertical line represents the interactive threshold. As the user changes the threshold, the panel below shows how the model performance is affected through common statistics.

explain model decisions using feature ideation. This technique consists of searching for new features for a machine learning model to improve its performance. For complex text data, machine learning requires experts to find the set of features by selection and combination of the existing ones. In *FeautureInsight* the user selects a set of features, while a document classifier is trained online. *FeatureInsight* uses feature semantic and rank to help the user selection. *Rivelo* on the other hand ranks features only on the basis of the explanations computed. On a document classifier, *Rivelo*'s feature ranking should match the semantic behind the words. When it doesn't, the user is able to spot the anomaly and inspect it by interacting with the tool.

3.4 Model Dependency in Visualizations

Many of the explanation methods investigated by researchers employ a white-box approach, that is, they aim at visualizing the model by creating representations of the internal structures of the models. For instance, logistic regression is often used to create a transparent weighting of the features and visualization systems have been developed to visualize decisions trees [31] and neural networks [16, 23]. These

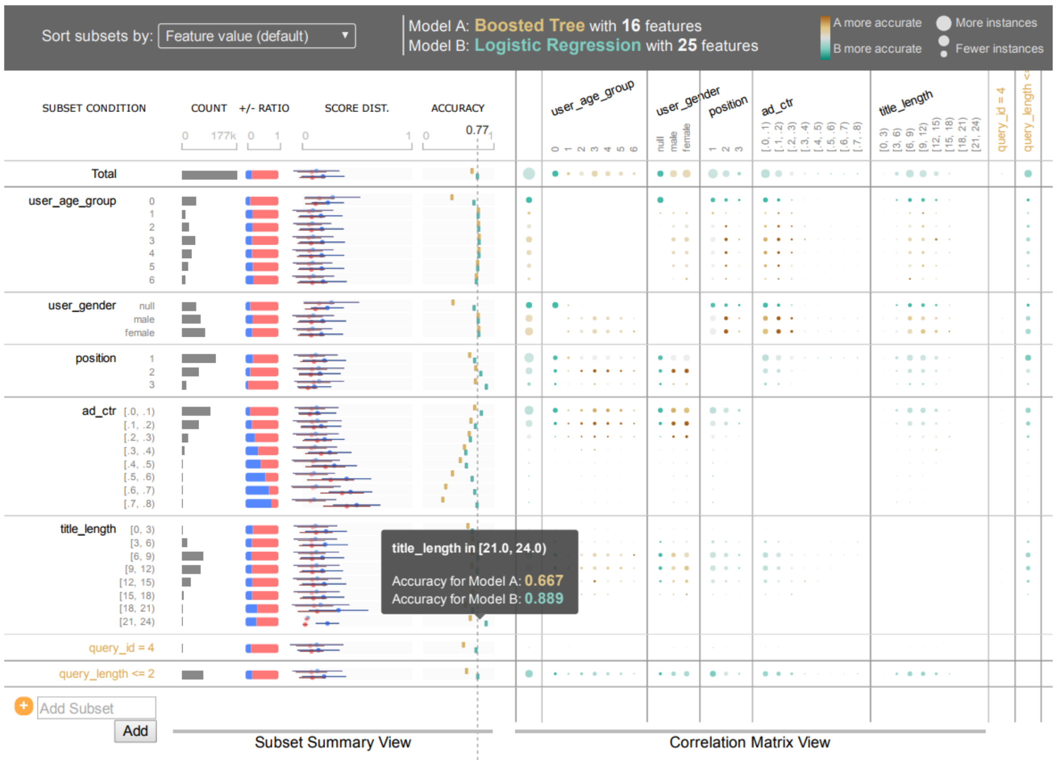


Figure 3.4. *MLCube Explorer* [11] interface is designed for analysis of data cubes, that is data array with 3 dimensions. The interface is made to compare two models over different subsets of instances. Subsets are defined by constraints on the features value. The aim is to understand which model is better for different subsets of instances in the discrete feature space. In the figure each subset is on a different row. Subsets are grouped in larger rows by the common constrained feature. For each subset a summary with some aggregated statistics is shown regarding the relative instances. By fixing the value of the other features on the columns, *MLCube Explorer* shows on the right a correlation matrix. This matrix depicts in colored circles which model is better for each pair of feature constraints.

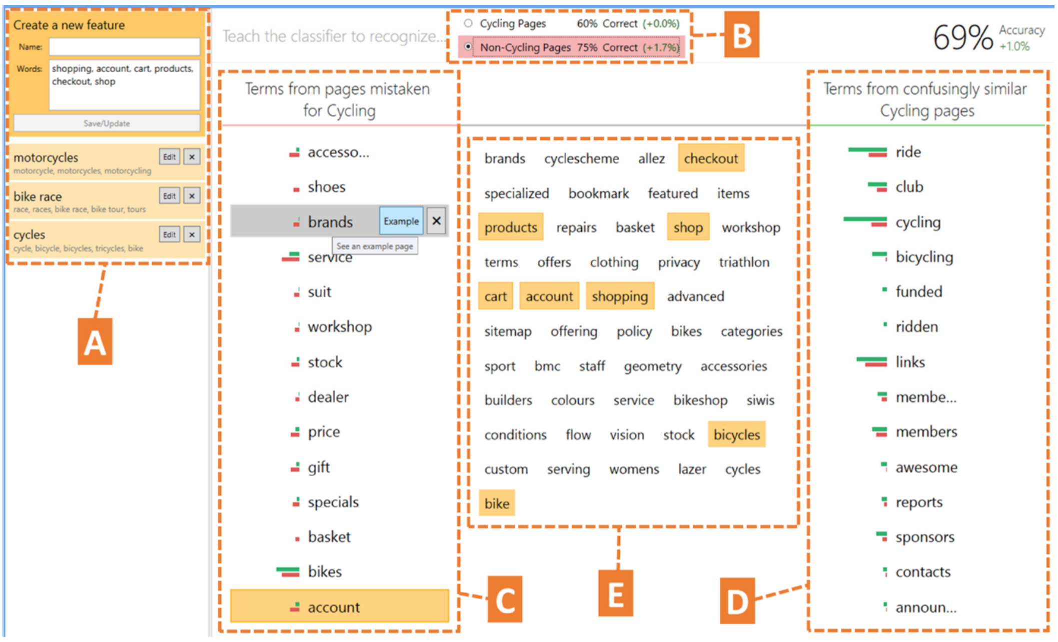

Create a new feature
Name:
Words: shopping, account, cart, products, checkout, shop
Save/Update
motorcycles
motorcycle, motorcycles, motorcycling
bike race
race, races, bike race, bike tour, tours
cycles
cycle, bicycle, bicycles, tricycles, bike
Edit
A
Teach the classifier to recognize...
Cycling Pages 60% Correct (+0.0%)
Non-Cycling Pages 75% Correct (+1.7%)
B
69% Accuracy +1.0%
Terms from pages mistaken for Cycling
accesso...
shoes
brands
Example
See an example page
service
suit
workshop
stock
dealer
price
gift
specials
basket
bikes
account
C
brands cyclescheme allez checkout
specialized bookmark featured items
products repairs basket shop workshop
terms offers clothing privacy triathlon
cart account shopping advanced
sitemap offering policy bikes categories
sport bmc staff geometry accessories
builders colours service bikeshop siwis
conditions flow vision stock bicycles
custom serving womens lazer cycles
bike
E
D
Terms from confusingly similar Cycling pages
ride
club
cycling
bicycling
funded
ridden
links
membe...
members
awesome
reports
sponsors
contacts
announ...

Figure 3.5. *FeatureInsight* [6] helps the user in searching for new features (words and phrases) for a binary document classifier. The interface is divided in 5 panels: (A) where the user can add new features; (B) where you can select the class for which you want to add new features; (C) where recurrent terms from misclassfied documents are suggested; (D) where recurrent terms from correctly classified documents are suggested; (E) where other features similar to the new added are shown. The user will decide which features to add by reading the hints from the panels (C), (D) and (E). Then an online retraining will take place to show any improvement or worsening of the model performance.

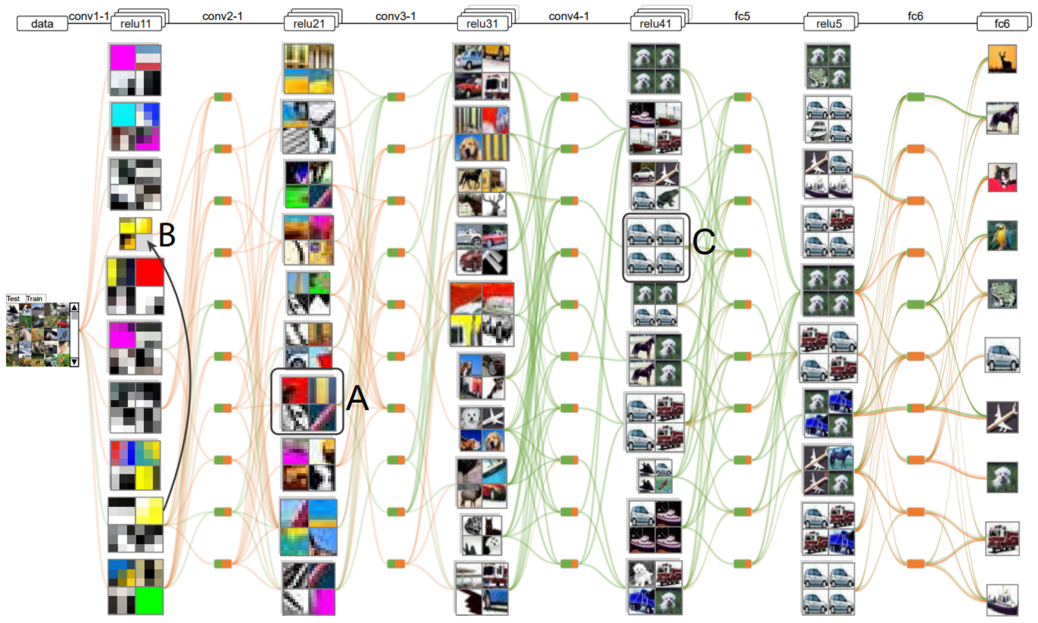


Figure 3.6. The image centric nature of *CNNVis* [16] visualization makes possible to understand how a neural networks classifies image data. In the visualization the user can identify each perceptron role in the classification. This is possible because for image data the tool can visualize shapes and colors to quickly show the user what each perceptron is doing.

methods however can only be applied to specific models and, as such, suffer from limited flexibility.

For example, *CNNVis* [16] (Figure 3.6) is designed to work only with neural networks. The tool uses a network visualization to explain the model which is especially desired for neural networks. This visualization is able to illustrate the decision making process of single neurons and how those decisions are linked. *CNNVis* differs from *Rivelo* because it manage image data instead of binary data. Due to the image centric nature of the visualization *CNNVis* cannot easily be applied to other data types. One solution developed in the past is the idea of training a more interpretable model out of an existing complex model, e.g., inferring rules from a neural network [8].

Another option is to treat the model as a black-box by exclusively inferring from its input-output relations. This more recent solution is the idea of generating local explanations that are able to explain how a model makes a decision for single instances of a data set [25, 13, 15]. These works are excellent solutions to the problem of investigating single instances but there are no established methods to go from single instances back to a global view of the model. This is precisely what our work tries to achieve by using instance-level explanations and embedding them in an interactive system that enables their navigation.

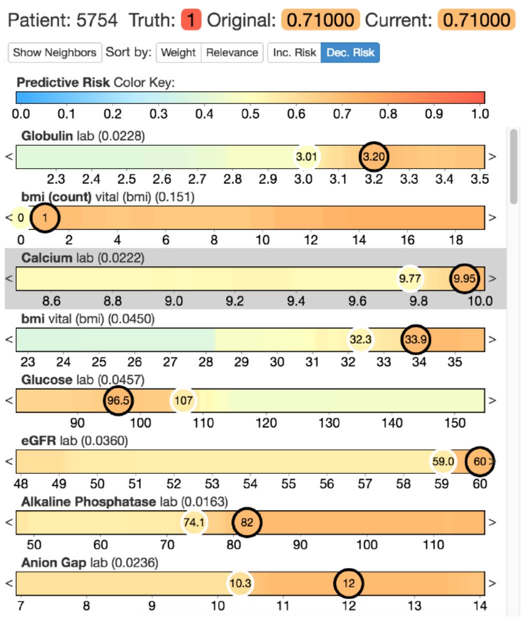

Patient: 5754 Truth: 1 Original: 0.71000 Current: 0.71000
Show Neighbors
Sort by: Weight Relevance
Inc. Risk
Dec. Risk
Predictive Risk Color Key:
Globulin lab (0.0228)
bmi (count) vital (bmi) (0.151)
Calcium lab (0.0222)
bmi vital (bmi) (0.0450)
Glucose lab (0.0457)
eGFR lab (0.0360)
Alkaline Phosphatase lab (0.0163)
Anion Gap lab (0.0236)

Figure 3.7. The depicted interface from *Prospector* [15] shows for each bar a different feature of a chosen instance of a machine learning model with numerical data. The user can use the bars to change the features value and see how the prediction changes. This way the user can understand the feature importance of the model for that given instance.

Therefore Martens and Provost's [19] method, used in our tool, is not the only approach for creating explanations from black-box machine learning models. However, it is optimized for sparse binary input data. *Prospector* [15] (Figure 3.7) on the other hand works with numerical input data. Explanation generation is not the primary purpose of this system, but it can be achieved by repeated application of its proposed procedure of assessing localized feature importance. *LIME* [25] describes a method of explanation generation utilizing a proxy model. That is its method first samples the neighborhood of the instance to be explained. Then a secondary model is trained solely on those sampled instances. This proxy model is chosen to be directly interpretable thus its global feature importance can be used as explanation for the analyzed instance. Those techniques would be valid replacements for the explanation creation method used in *Rivelo* when dealing with numerical data.

3.5 Human Interaction Involvement

Somewhat related to our work are interactive machine learning solutions in which the output of the model is visualized to allow the user to give feedback to the model and improve its decisions [2, 3, 29].

For example *ReGroup* [3] and *CueFlik* [2] are able to involve the user in the decision making process of the model. *ReGroup* shows how a model aggregates nodes within a social network. The user starts by creating groups and then inspects the consequent recommendations of new members provided by the model. *CueFlik* on the other hand uses a model to classify images starting from a set of examples given by the user. Tam *et al.* [29] leverages human interaction for creating decision trees with the help of domain knowledge. Compared to greedy automatically created decision trees, case studies show that manual trees performed by the user return a better performance. In all three of these previous cases the user is directly responsible of the classification performance through interaction. This way the final classification improves and the user becomes more confident with the model. The goal of our work however is to introduce methods and tools that can be easily integrated in existing settings and workflows adopted by domain experts, and as such does not rely on the complex modifications necessary to include explicit user feedback in the process.

Chapter 4

Instance-Level Explanations

An instance level explanation consists of a set of features that are considered the most responsible for the prediction of an instance, i.e., the smallest set of features which have to be changed in the instance's binary vector to alter the predicted label. We used a variant of the instance-level explanation algorithm designed by Martens and Provost [19] for document classification. We will describe first the original algorithm by Martens and Provost [19], then we will see how and why our version differs.

4.1 Martens and Provost's Method

The overall process of the explanation generation is illustrated in Figure 4.1. *Rivelo* computes an explanation for each instance using the input-output relationships of a black-box model.

The technique works by creating artificial instances derived from observed values in order to examine the influence of the features to the output. It assumes binary feature vectors. Starting with the original binary vector x and its predicted label, the algorithm "removes" features from the vector creating an artificial instance $x - e$. Features are added to the vector of "removed" features e until the predicted label of $x - e$ is different than the one of x. The set $E = \{k \,|\, e_k = 1\}$ of "removed" features is then called an *explanation* of the original vector x.

Removing in this context indicates the change of a feature that is present to not being present. We chose the term "removing" because it is particularly intuitive for sparse binary data. The technique works only for sparse binary data where it is possible to assign prediction responsibility to components relative to present features. This is the case for bag of words in document classifiers, but also for any kind of data where instances represent a set of items, like medications in the medical domain and products bought or liked in market basket analysis. Bag of words can

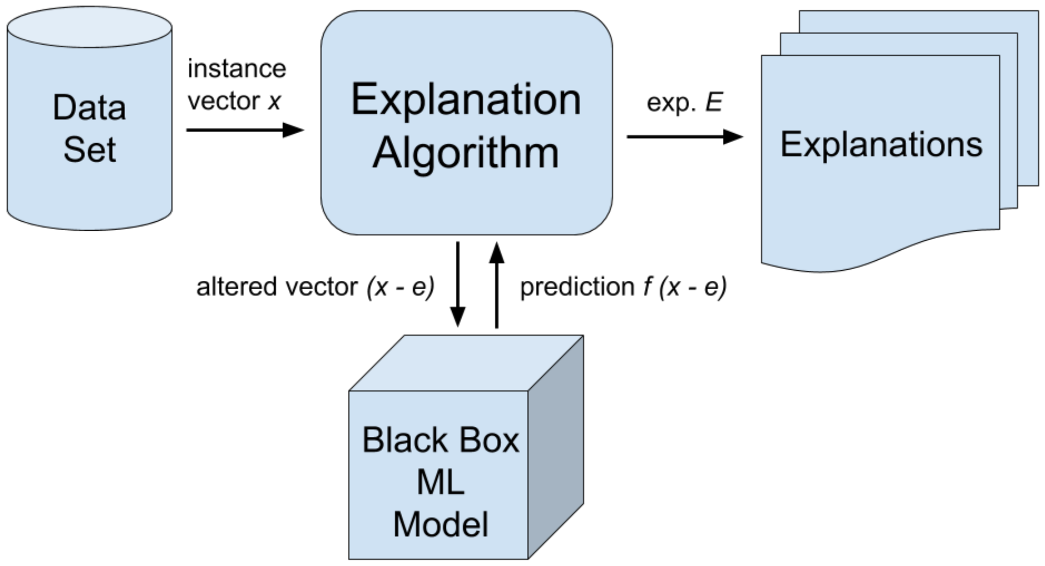


Figure 4.1. Illustrating the process of explanation computation. x is an observed instance vector. We treat the ML model as function f that maps a vector to a prediction score. The explanation algorithm tries to find the shortest vector e for which $f(x - e)$ is below the threshold. This vector e serves as an *explanation* for the prediction.

also contain *n-grams* where features represent expressions of multiple terms that can be used to explain a prediction. In the case instead of not sparse binary data, the components most responsible for a prediction are given by not present features. It is possible to select those features while building explanations by providing the tool with the binary complement data and model.

The machine learning model is assumed to deterministically compute a prediction score $f(x)$ between 0 and 1 for a given input vector x. The predicted label is then computed by comparing this score to a given optimal threshold. This threshold is computed on the training data minimizing the number of misclassifications.

The process of removing features from a vector is the core of the explanation algorithm. The algorithm consists of successively removing the feature with the highest impact on the prediction score towards the threshold until the label changes. This results in the most compact explanation for a given instance. If the prediction score cannot be changed towards the threshold by removing a feature the instance is discarded without outputting an explanation. Despite this in most cases it is always possible to remove a feature that changes the prediction towards the threshold. In these cases the algorithm keeps removing features until either one of the three conditions is met: (i) there a no features left to remove in the vector; (ii) the explanation set is finally able to change prediction to the original instance vector; (iii) the candidate explanation reached already a maximum size. The last condition is needed because explanations are useless when they are too long to read. Explanations that are long are not intuitively interpretable by a user thus they are discarded in this original version of the algorithm.

4.2 Our Implementation

Martens and Provost [19] designed an algorithm which we first implemented in *Python* in its original version. This original algorithm had strict rules for instances that would not smoothly achieve an explanation. Therefore we added some extra steps to the original version aiming at relaxing this rules in order to reduce the number of discarded instances without losing the compactness property of the explanations. We needed more explained instances in order to achieve a more complete visualization.

The original algorithm was discarding instances when it was not possible to find a feature able to change the prediction score towards the threshold. To increase the number of explained instances, in order to provide a fuller picture of the model's decisions in our visualization, we relax this restriction by removing random features with no impact on the prediction score. Oftentimes, after removing some features randomly the prediction score starts changing again leading to an explanation for this otherwise discarded instance. However, removing features with no impact on the prediction score violates the compactness property of the resulting explanation. In order to restore this property we add a post-processing step that re-adds features that do not contribute to a favorable prediction score change. This process is time consuming requiring us to pre-compute explanations offline.

Another difference of our approach to the algorithm by Martens and Provost [19] is the handling of explanations that grow too long. The original algorithm was also discarding instances when the explanations was growing more than a fixed size, usually 5 features. As we are adding random features, in some cases the explanation might grow too long before the compacting step. In order to not discard explanations that are only temporarily too long we perform the length check after this step.

To summarize we report here all those extra steps which are the differences with Martens and Provost's method [19].

1. Sometimes we also add features to the explanation that once removed from the vector do not cause any change in the scorer function. Those score irrelevant features do not represents an improvement in building the explanation, but they allow us to not stop the search. By not stopping the search we can avoid discarding some instances from their explanation computation.

2. While looking for features to add to the explanation set, we do not stop if the explanation set reaches a maximum size. Our search for features terminates

only when either we removed all features from the vector or the removal of any feature will cause a worsening in the scorer function.

3. Because of the presence of explanations that are too long and explanations that contain score irrelevant features, we need to post-process all computed explanations, in order to reduce their size if possible. Post-processing consists in disregarding explanations longer than 5 after an attempt of shortening them.

To understand how all of this works in practice, in the following pages we will go through the details of the actual algorithm.

Chapter 5

The Algorithm

In order to describe more in detail the algorithm of *Rivelo*, we need first a more strict definition of instance-level explanation.

Definition 5.0.1. Instance-Level Explanation
Given a machine learning model with scorer function $f(x) = p \in [0, 1]$ and an instance $x : \{x_i \in \{0, 1\} \;\; i = 1, .., n\}$, we can denote the set $D = \{i : x_i = 1\}$ that represents the set of features of x. We state that the x has class c because $f(x) \geq l$ where $l \in [0, 1]$ is the threshold of model.
We define $E : \{i : e_i = 1\}$ where $e : \{e_i \in \{0, 1\} \;\; i = 1, .., n\}$ is the binary vector representing the explanation. E is an explanation of the instance x if:

- $E \subseteq D$
- $f(x - e) < l$

Given any machine learning model able to classify binary vectors x with a prediction p with the following scorer function f: $f(x) = p$, $p \in [0, 1]$, $\{x_i \in \{0, 1\} \;\; i = 1, .., n\}$ we are able to compute an explanation for each instance. For this reason we have a model-independent technique that allows to deal with models as black boxes. The only requirement is to use as input only binary data, but this means we can compute explanations for document classification models. This kind of model converts text data into binary data treating documents as bag of words, after removing stop-words and stemming all remaining terms. Especially in document classification models, the number of features is high but thanks to an hill-climbing approach we are able to compute explanations even for models with thousands of features. This hill-climbing approach is present in our alternative version of the algorithm by Martens and Provost [19].

5.1 The Explanation Computation

Before displaying the algorithm we give more details about the prediction of the classification. Given x instance of the data-set:

$$D = \{i : x_i = 1\}$$

$$x_{A_i} = \begin{cases} 1 & i \in A \\ 0 & i \notin A \end{cases}$$

$$f(x_A) = p \in [0, 1]$$

$$0 \leq t \leq 1$$

$$C_M(D) = \begin{cases} c_1 & p \geq t \\ c_2 & p \leq t \end{cases}$$

We need to point out that not always is possible to find a small explanation E that changes the prediction for the instance x. As E increases in size, it loses its capability to explain properly the prediction. It becomes completely useless when the cardinality of E is equal to $\sum_i x_i$ as it contains all the possible features present in the binary vector. For this reason we fix z_{max} as the maximum size of E after which we give up and we move to the next instance in the data-set.
The algorithm works on a data-set DS of P instances described as follows:

$$DS = \{x^p \quad p = 1, .., P\}$$

where for each $x^p : C_M(D^p) \in \{c_1, c_2\}$ we try to compute a different explanation E. The pseudo-code for the explanation computation is provided at Algorithm 1. Keep in mind that the actual implementation is a bit different since we take advantage of the representations of sets in binary vectors so that we are able to compute prediction changes in an aggregated manner. It follows now a description of the steps in the procedure.

1. For each instance x of DS, we iter over all the elements in the set D.

2. In the beginning our set that will later represent the explanation is empty: $E_0 = \{\emptyset\}$

3. For each element i in the current set D/E_0 we compute the prediction change

$\Delta_i f$. We do this by removing from the current set the element i and subtracting the new prediction value on the new set with the old one on the current set.

4. After all $\Delta_i f$ are computed, depending on which was the originally predicted class c, we pick the element i^* where the difference is more relevant. That is:

   (a) the more negative $\Delta_i f$ if $c = c_1$

   (b) the more positive $\Delta_i f$ if $c = c_2$

   If there is no suitable $\Delta_i f$ to pick, we exit the *while* cycle. That is when:

   (a) $\Delta_i f > 0 \;\forall\; i$ if $c = c_1$

   (b) $\Delta_i f < 0 \;\forall\; i$ if $c = c_2$

5. If we could find a suitable $\Delta_{i^*} f$, we check if it is relative to a null prediction change ($\Delta_{i^*} f = 0$). If it is not the case we just go directly to point 6. Otherwise, we retrieve all i relative to null prediction changes and we pick a random one among them. We set this random picked i to our new i^*.

6. We store i^* in the originally empty set E_0.

7. We compute the class $\hat{c}$ of D/E_0 by comparing the new prediction value with a fixed threshold t: $C_M(D/E_0) = \hat{c}$.

8. If the class already changed, $\hat{c} \neq c$, it means we have the explanation $E = E_0$ and we exit the *while* cycle. Otherwise, if $\hat{c} = c$, we start again from point 3 trying to remove another element from the current set D/E_0. This procedure will iter until either there are no elements left to remove or the condition $c \neq \hat{c}$ is met.

9. When we exit the *while* cycle we return the set $E = E_0^k$.

This way we are returning sets that do not always return a change of class. These faulty explanations are relative to two cases:

- the current set D/E_0 is empty;
- no suitable element left to remove: by removing any of the element left, the prediction change would have taken us even further from the threshold t value.

The algorithm was implemented to store faulty explanation anyway in order to keep record of the faulty cases during the development of the software. Those faulty cases are not input of the postprocess computation we will see in Section 5.2.

This algorithm is done in such a way that sometimes for some $\hat{k}$ we add to E features i such that $\Delta_i^{\hat{k}} f = 0$. Those features do not represents an improvement in our research for E but they allow us to not stop the search. Eventually for $k > \hat{k}$ we will find $\Delta_i^k f \neq 0$ such that we are pointed again in the right direction for a change of class. For the same reason we do not bound the size for explanation at this moment, leaving them grow until either the class change or there is no element to remove left. In the next Section 5.2 we will describe a second algorithm able to postprocess all computed too long and not-faulty explanations, in order to reduce their size if possible.

5.2 The Postprocessing Computation

In this part we try to reduce the size of each computed explanation E. For each E, we check if there is any feature in the set that, if removed from E, does not compromise the change of class. As explained before, we know that such features exist because, when we built previously the explanations, we also added the ones that didn't involve any change in the scorer function. Basically this computation resembles the process of going backwards compare to how we built the explanations. In practice we are trying to re-add elements in the set E to the set D/E and see if the change of class ($\hat{c} \neq c$) is still guaranteed. The Algorithm 2 of explanation postprocessing describes the procedure taken for each not faulty explanation E and it comprehends the following steps:

1. The algorithm takes as input an instance D, its predicted class c and its computed explanation E.
2. We set the set $E_0 = E$ and we enter the *while* cycle.
3. For each element j of the set E_0, we compute the prediction change $\Delta_j f$ by subtracting the prediction value on the new set $(D/E_0) \cup \{j\} = D/(E_0/\{j\})$ with the old one on the current set D/E_0.
4. Depending on the original class, we are picking the more suitable j^*. That is the j which keeps or takes us as further as possible from the threshold t. Such an element is usually linked to the less relevant feature in the explanation. To do this we perform the following:
   (a) If $c = c_1 : j^* \mid \Delta_{j^*} f \leq \Delta_j f \ \forall\ j$
   (b) If $c = c_2 : j^* \mid \Delta_{j^*} f \geq \Delta_j f \ \forall\ j$
5. At this point there are 2 cases:
   (a) If $C_M(D/(E_0/\{j^*\})) = \hat{c} \neq c :$ removing j^* from the E_0 did not compromise the change of class. We can therefore remove j^* from E_0

$(E_0 = E_0/\{j^*\})$ and go back to point 3 and try to reduce even more the size of the explanation.

(b) If $C_M(D/(E_0/\{j^*\})) = \hat{c} = c$: removing j^* from the E_0 compromised the change of class. We need therefore to exit the *while* cycle.

6. If we exit the cycle we return the new explanation $E' = E_0$ of reduced size only if the decrease in size is high enough. That is when $size(E') \leq z_{max}$ where z_{max} is a user-defined parameter.

After Algorithm 2, we obtain the final set of explanations E' that will be used for *Rivelo* visualization. We list here three considerations regarding our result.

- Each explanation is as small as possible. We do not want features that are irrelevant to the prediction score that were added just to not get stuck in Algorithm 1. Those features were removed from the explanations in Algorithm 2.

- Each explanation size does not excess the user desired size z_{max}. Long explanation are not practicable since they should usually point directly to few features that are the key for that prediction outcome.

- We increased the final number of explanations as we could produce more explanations than the original algorithm by Martens and Provost [19]. This is extremely desired since the more explanation we have, the more useful will be *Rivelo*'s interactive visualization. This can be measured by checking the percentage of explained instances in the test set over their the total number. In Chapter 7 we will see an exact scenario for this increase in number of explained instances.

Algorithm 1: Explanation Computation

```
Input: $\{x_i^p \in \{0,1\} \quad i = 1,..,n\}$
$D = \{i : x_i = 1\}$
$E_0^0 = \{\emptyset\}$
$c^0 = c = C_M(D)$
$k = 0$
Output: E
while $c^k = c$ or $size(D/E_0^k) \neq 0$ do
    for $i \in D/E_0^k$ do
        $E_i^k = E_0^k \cup \{i\}$
        $\Delta_i^k f = f(x_{D/E_i^k}) - f(x_{D/E_0^k})$
    end
    if $c = c_1$ then
        $i^* = \arg\min_i \Delta_i^k f$
        if $\Delta_{i^*}^k f > 0$ then
            break
        end
    end
    if $c = c_2$ then
        $i^* = \arg\max_i \Delta_i^k f$
        if $\Delta_{i^*}^k f < 0$ then
            break
        end
    end
    if $\Delta_{i^*}^k f = 0$ then
        $i^* = Random\ pick(\{\ i \mid \Delta_i^k f = 0\ \})$
    end
    $E_0^{k+1} = E_{i^*}^k$
    $c^{k+1} = C_M(D/E_0^{k+1})$
    $k = k + 1$
end
return $E = E_0^k$
```

Algorithm 2: Explanation Postprocessing

```
Input: $E_0^0 = E$
$D = \{i : x_i = 1\}$
$\hat{c}^0 = \hat{c} = C_M(D/E)$
$c = C_M(D)$
$k = 0$
Output: $E'$
while $True$ do
    for $j \in E_0^k$ do
        $E_j^k = E_0^k/\{j\}$
        $\Delta_j^k f = f(x_{D/E_j^k}) - f(x_{D/E_0^k})$
    end
    if $c = c_1$ then
        $j^* = \arg\min_j \Delta_j^k f$
    end
    if $c = c_2$ then
        $j^* = \arg\max_j \Delta_j^k f$
    end
    if $C_M(D/E_{j^*}^k) = c$ then
        break
    end
    else
        $E_0^{k+1} = E_{j^*}^k$
        $k = k + 1$
    end
end
if $size(E_0^k) \leq z_{max}$ then
    return $E' = E_0^k$
end
```

Chapter 6

The Explanation Interface

We implemented the workflow (Figure 6.1), which we already briefly described in the introduction (Chapter 1), in an interactive visual interface we call *Rivelo*[1]. The application in its current implementation works exclusively with binary classifiers and binary features and it assumes to receive as an input a data set and a trained classifier. The data set must also contain ground truth information, that is, for each data item what is the correct label the classifier is supposed to predict. Once the system is launched, it automatically computes the following information: one explanation for each data item; information about whether the prediction made by the classifier is correct or incorrect (including false positives and false negatives); the list of features, ranked according to how frequently they appear in the explanations. For each feature, it also computes the ratio between positive and negative labels, that is whether the feature tends to predict more positive or negative outcomes, and the number of errors the classifier makes when predicting items whose explanations contain that feature.

The user interface is made of the following main panels that reflect the steps of the workflow (check labels in Figure 6.2):

- a *feature list panel* (Labels 1, 2) on the left, to show the list of ranked features;
- the *explanations panel* (Labels 3, 4, 5) next to it, to show the explanations and data items containing the selected features;
- the *descriptors panel* (Label 6), containing a visual representation of the vectors / descriptors of the data items selected in the explanations panel;
- the *raw data panel* (Labels 7, 8) containing the raw data of selected vectors;
- the *data stats panel* (Figure 6.3) to understand aggregated statics regarding the predictions and their explanations.

[1] *Rivelo* GitHub repository at: `https://github.com/nyuvis/Rivelo`.

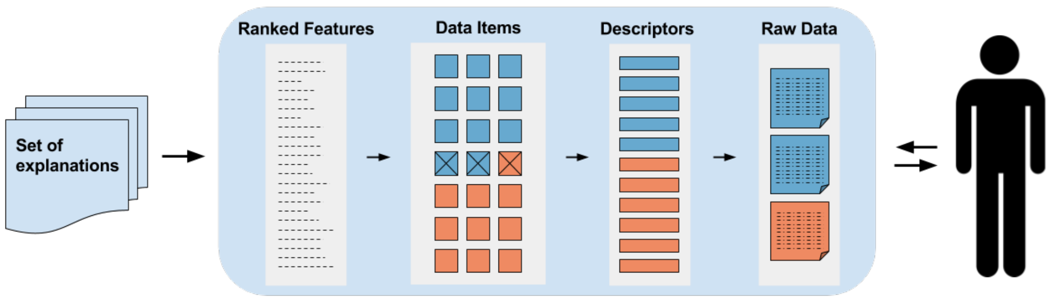
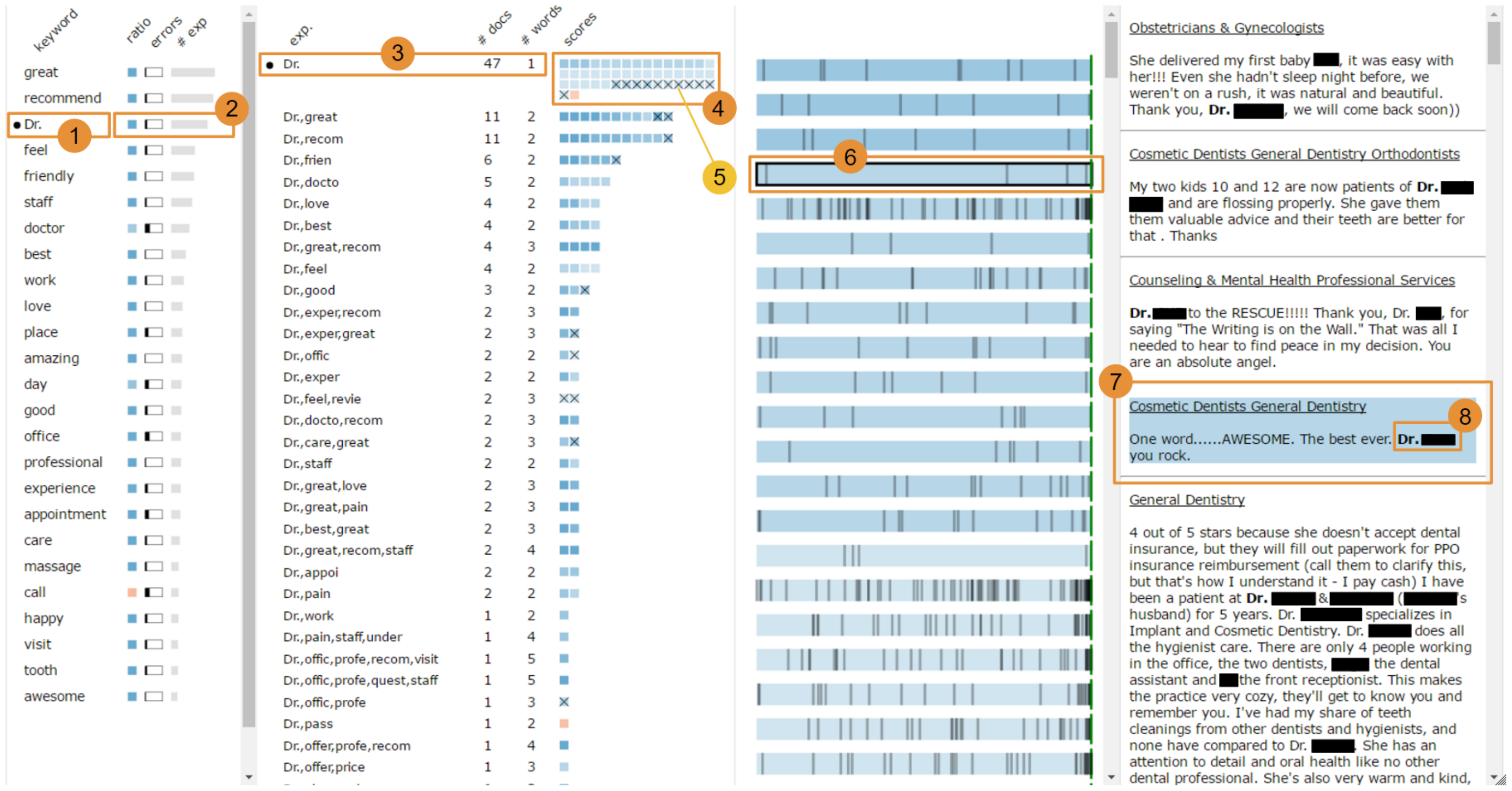


Figure 6.1. The *Rivelo* workflow involves the following user interactions: selection of features, selection of the explanation, vectors inspection, and exploration of the raw data. By switching back and forth between those steps, the user can freely change the selections in a smooth and animated visualization. This explanation-driven workflow can extract global behaviours of the model from patterns of local anomalies through human interaction.

We now describe each panel and interaction with them in more details.



great
recommend
● Dr.
feel
friendly
staff
doctor
best
work
love
place
amazing
day
good
office
professional
experience
appointment
care
massage
call
happy
visit
tooth
awesome

exp.	# docs	# words
● Dr.	47	1
Dr.,great	11	2
Dr.,recom	11	2
Dr.,frien	6	2
Dr.,docto	5	2
Dr.,love	4	2
Dr.,best	4	2
Dr.,great,recom	4	3
Dr.,feel	4	2
Dr.,good	3	2
Dr.,exper,recom	2	3
Dr.,exper,great	2	3
Dr.,offic	2	2
Dr.,exper	2	2
Dr.,feel,revie	2	3
Dr.,docto,recom	2	3
Dr.,care,great	2	3
Dr.,staff	2	2
Dr.,great,love	2	3
Dr.,great,pain	2	3
Dr.,best,great	2	3
Dr.,great,recom,staff	2	4
Dr.,appoi	2	2
Dr.,pain	2	2
Dr.,work	1	2
Dr.,pain,staff,under	1	4
Dr.,offic,profe,recom,visit	1	5
Dr.,offic,profe,quest,staff	1	5
Dr.,offic,profe	1	3
Dr.,pass	1	2
Dr.,offer,profe,recom	1	4
Dr.,offer,price	1	3

Obstetricians & Gynecologists

She delivered my first baby [redacted], it was easy with her!!! Even she hadn't sleep night before, we weren't on a rush, it was natural and beautiful. Thank you, **Dr.** [redacted], we will come back soon))

Cosmetic Dentists General Dentistry Orthodontists

My two kids 10 and 12 are now patients of **Dr.** [redacted] and are flossing properly. She gave them them valuable advice and their teeth are better for that . Thanks

Counseling & Mental Health Professional Services

Dr. [redacted] to the RESCUE!!!!! Thank you, Dr. [redacted], for saying "The Writing is on the Wall." That was all I needed to hear to find peace in my decision. You are an absolute angel.

Cosmetic Dentists General Dentistry

One word......AWESOME. The best ever. **Dr.** [redacted] you rock.

General Dentistry

4 out of 5 stars because she doesn't accept dental insurance, but they will fill out paperwork for PPO insurance reimbursement (call them to clarify this, but that's how I understand it - I pay cash) I have been a patient at **Dr.** [redacted] & [redacted] ([redacted] s husband) for 5 years. Dr. [redacted] specializes in Implant and Cosmetic Dentistry. Dr. [redacted] does all the hygienist care. There are only 4 people working in the office, the two dentists, [redacted] the dental assistant and [redacted] the front receptionist. This makes the practice very cozy, they'll get to know you and remember you. I've had my share of teeth cleanings from other dentists and hygienists, and none have compared to Dr. [redacted]. She has an attention to detail and oral health like no other dental professional. She's also very warm and kind,

Figure 6.2. Showing the user interface of *Rivelo*. (1) is a selected feature and (2) are its relative indicators for average prediction, errors and frequency within the explanations set. Some of the features on this list are grayed out because they wouldn't return any explanation if added to the query. (3) is the selected explanation with the number of explained documents and the number of used features. (4) are the cells representing each explained instance, with color resembling the prediction value. Among them, (5) represent some false positive instances. (6) is the descriptor representing visually one of the explained instances. (7) is the raw data related to the same descriptor. (8) shows in bold how the explanation feature is used in the text data.

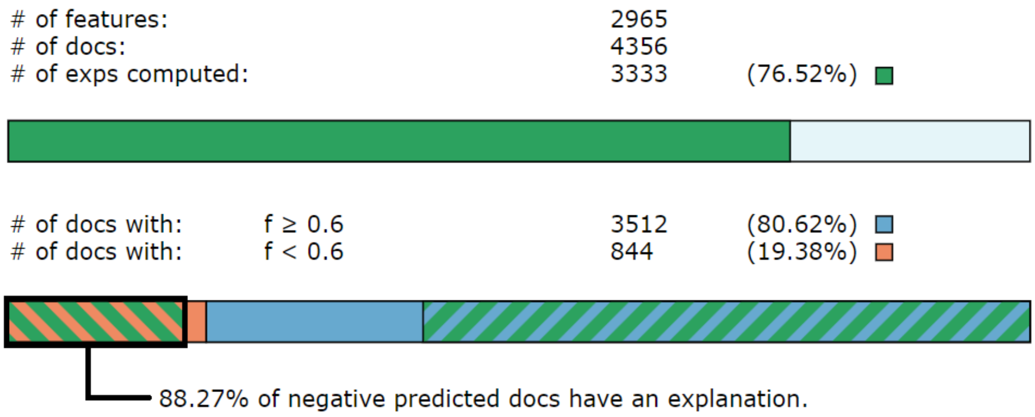
The **feature list panel** displays the features computed in the beginning and displays the following information in a list: the name of the feature, the ratio of positive labels, the number of errors and the number of explanations it belongs to. The ratio is displayed as a colored dot with shades that go from *blue* to *red* to depict ratios between 0% and 100% true outcomes. The number of errors and the number of explanations are depicted using horizontal bar charts. Users can sort the list according to three main parameters: (1) *number of explanations*, which enables them to focus on importance, that is, how many decisions the classifier actually makes using that feature. (2) *ratio of positive labels*, which enables them to focus on specific sets of decisions, and (3) *number of errors*, which enables them to focus on correctness, that is, what features and instances create most of the problems.

Once one or more features are selected, the **explanations panel** displays all explanations containing the selected features. The explanations are sorted according to the number of explained instances. Each explanation has a group of colored cells next to it ((4) in Figure 6.2) that represent one instance each. The color of the cell indicates the value of its prediction (blue for positive and red for negative) and different shades depending on the prediction value. When the cell represents an instance that is classified incorrectly it is marked with a small cross, to indicate the error ((5) in Figure 6.2).

By selecting an explanation or a single cell we can display the explained instances in the **descriptors panel**. The panel displays a list of visual “descriptors”, each depicting the vector of values used to represent an instance in the data set ((6) in Figure 6.2). The descriptor is designed as follows. Each rectangle is split into as many (thin) cells as the number of features in the data set. A cell is colored with a dark shade (small vertical dark segments in the image) when a feature is present in the instance and left empty when it is not present. A cell is colored in green when it represents a feature contained in the explanation. The background color represent the predicted label.

The main use of descriptors is to help the user visually assess the similarity between the selected instances according to which (binary) features they contain. When looking at the list of descriptors, one can at a glance learn how many features are contained in a instance (that is, how many dark cells are in it) and how similar the distribution of features is. Descriptors are particularly useful in the case of instances with sparse features, that is, when the number of features present in a given instance is much smaller than the number of those which are not present.

Next to the vectors panel, the **raw data panel** displays the actual raw data corresponding to the selected descriptors. This is useful to create a “semantic bridge” between the abstract representation used by the classifier and the descriptor



```
# of features:                      2965
# of docs:                          4356
# of exps computed:                 3333      (76.52%) 🟩

# of docs with:     f ≥ 0.6         3512      (80.62%) 🟦
# of docs with:     f < 0.6         844       (19.38%) 🟧
```

Figure 6.3. The Data Stats panel shows statistics for explained instances. The top bar shows the overall coverage while the bar below shows the distribution over the two labels.

representation, and the original data. In our example, the data shown is text from a document collection but different representations can be used in this panel to connect the abstract vector to the actual original data contained in the data set (e.g., images). In this case we also highlight, in each text snippet, the words that correspond to the features contained in the vector. Similar solutions can be imagined for other data types.

One additional panel of the user interface is accessible on demand to obtain aggregate statistics about the explanation process and the classifier (Figure 6.3). The **data stats panel** provides several pieces of information including: percentage of explained instances over the total number of instances, number of instances predicted correctly and incorrectly for each label, number of features, number of data instances as well as the prediction threshold used by the classifier to discriminate between positive and negative cases. As you can see in Figure 6.3 two bars are displayed. In the first bar the percentage of explained instances over the total number of instances can be inspected. Below, in the second bar, it is possible to assess the real coverage of explanations over positive and negative predictions. The different percentages are shown by interacting with the different areas in the bar.

An extra visualization of *Rivelo* is the *performance histogram*, shown in Figure 6.4, which appears if requested. The *performance histogram* lets you perceive how the threshold impacts the predicted labels. The x-axis of this plot represents the prediction score decreasing from 1 to 0 on the right. Each bin represents the share of instances predicted within a range. The horizontal line in the plot divides each bin in two parts: above the line we have positive ground truth instances and below negative ground truth instances. Predicted positive and negative instances will be represented by blue and red bins respectively. From this visualization it is possible

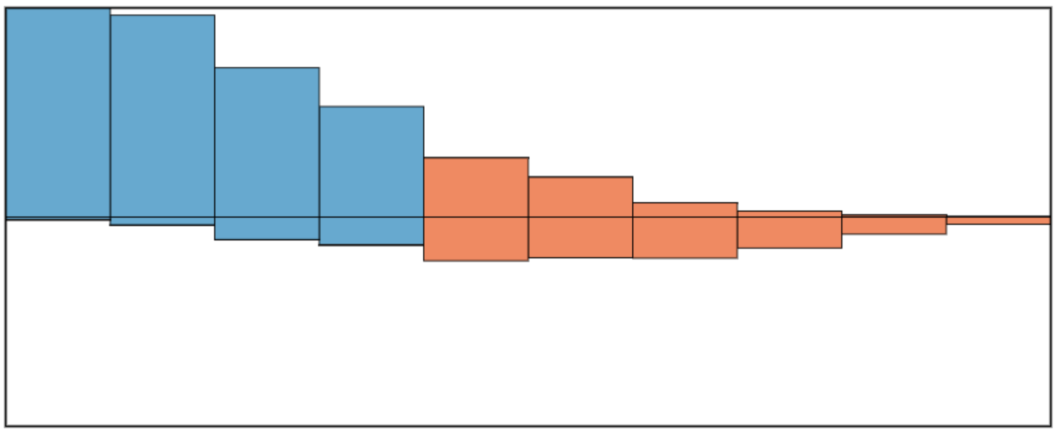


Figure 6.4. The performance histogram shows the distributions of prediction scores and misclassifications. Each bin is relative to a different prediction score decreasing from 1 to 0 to the right. The threshold (in this case 0.6) is determined by the change of color between bins: all blue bins represent positive predicted instances and all red bins negative predicted instances. The horizontal line divides the bins by ground truth percentage. The portion of a bin above the horizontal line is relative to positive ground truth instances, below to negative ones. For example the first red bin to the left represents all instances with prediction $p : 0.5 < p \leq 0.6$. Being red, this bin represents negative predictions and the portion above the horizontal line is relative to its false negatives, below the line to its true negatives. In this case the portions of the bin related to misclassifications (false negatives) is dangerously close to 50% of the bin, but this percentage will decrease by moving to bins further away from the threshold.

to perceive a quick idea of instances distribution by prediction score and an overview of misclassifications. Given the automatically computed optimal threshold, it is possible to quickly compare distributions of ground truth labels, encoded by the y-axis, with predicted labels, encoded with the color, for different prediction scores.

Chapter 7

Use Case: Doctor reviews on Yelp

In this chapter we present a use case based on the analysis of a text classifier used to predict the rating a user will give to a doctor in Yelp, the popular reviews aggregator. To generate the classifier, we used a collection of 5000 reviews submitted by Yelp users. We first processed the collection using stemming and stop-word removal and then created a binary feature for each term extracted, for a total of 2965 binary features. We also created a label out of the rating field, grouping together reviews with 1 or 2 stars for negative reviews and those with 4 and 5 stars for positive reviews. This way we created 4356 binary vectors, one for each review, of which 3334 (76%) represent positive reviews and the remaining (24%) negative reviews. Then, we trained a random forest model [5] based on the data and labels just described and obtained a classifier with an area under the ROC curve score equal to $AUC = 0.875$. We exported the scorer function of the model, the predicted label and the ground truth label for each review of test data to generate the input for *Rivelo*. We also computed the optimal threshold for the scorer function 0.6. The value is closer to 1 to compensate for the high amount of positive reviews in the data set.

After the first computation of explanations, which took around 1.5 hours to complete, we explained 64% of the reviews with explanations of length up to 5 words. By post-processing, which took about 1.5 hours as well, we were able to reduce the size of 34.5% of the 1563 explanations that were longer than 5 features. This way we increased the number of explained instances by 542, explaining 76.52% of instances of our test set. By compacting long explanations in the post-processing, we are able to explain 12% more instances than the original algorithm by Martens and Provost [19]. This statistic results are visualized in the *data stats panel* of Figure 6.3, while Figure 6.2 shows the results obtained by the entire procedure.

Looking at the set of features sorted by frequency one can readily see that most

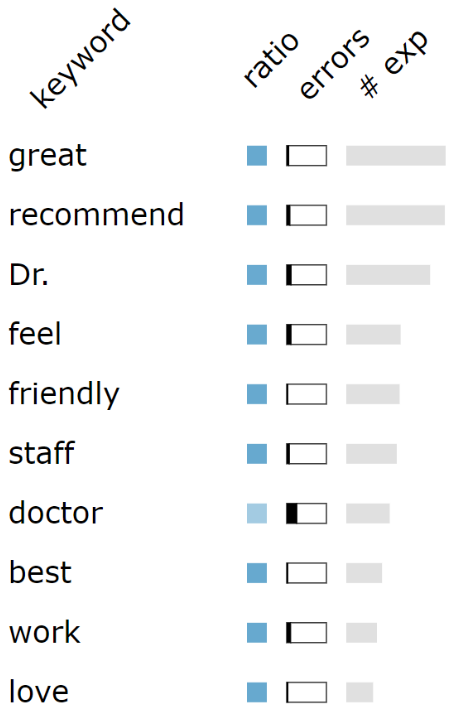
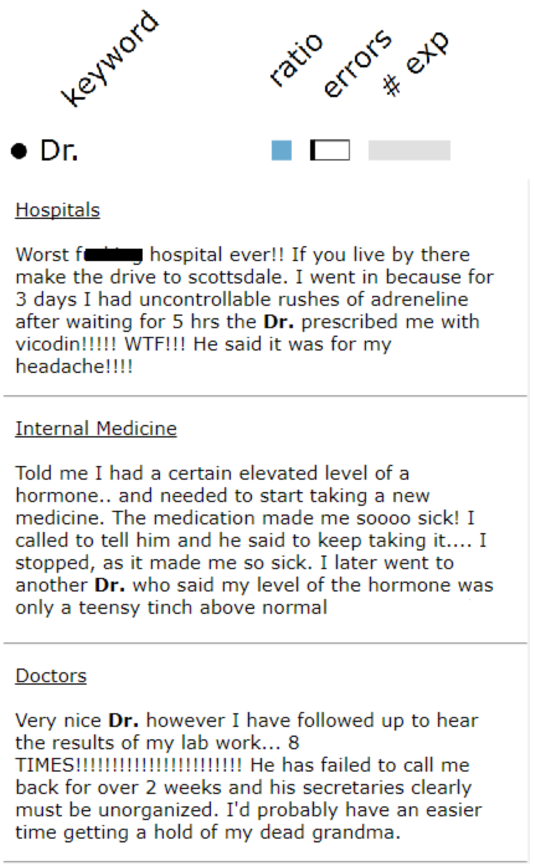


Figure 7.1. Reported in this figure we have the top 10 most used features used in explanations for the Yelp case study. Most of the reviews in the data-set are positive, therefore the majority of the computed explanation are relative to positive predictions. This is the reason why all of those most frequent explanation terms have a positive ratio (blue label).

of the model decisions take place to predict the positive label and that many of the words used capture adjectives that represent positive sentiments such as, "*friendly*", "*love*" and "*recommend*", which have been labeled correctly as positive words (Figure 7.1). By taking a closer look we also see less obvious words, that do not seem to have straightforward role in the classification, even if they are used more than others in explanations. For example, the word "*Dr.*" is used frequently to explain positive reviews. Using the error bar indicator next to "*Dr.*" (Figure 7.1), we can also see that the feature has a very low false positive rate.

When we select this feature, the interface shows all the explanations and instances containing it. We can then see that the large majority of cases is predicted by the word "*Dr.*" alone or a combination of this word with some other positive property such as "*great*" and "*recommend*". We can also see that some of the instances are classified incorrectly, especially those in which the explanation contains exclusively the word "*Dr.*". To better understand this trend, we inspect several raw text documents associated to these instances and figure out that reviewers tend to use the name of the doctor preceded by the word "*Dr.*" whenever they have to say



Hospitals

Worst f[redacted] hospital ever!! If you live by there make the drive to scottsdale. I went in because for 3 days I had uncontrollable rushes of adreneline after waiting for 5 hrs the **Dr.** prescribed me with vicodin!!!!! WTF!!! He said it was for my headache!!!!

Internal Medicine

Told me I had a certain elevated level of a hormone.. and needed to start taking a new medicine. The medication made me soooo sick! I called to tell him and he said to keep taking it.... I stopped, as it made me so sick. I later went to another **Dr.** who said my level of the hormone was only a teensy tinch above normal

Doctors

Very nice **Dr.** however I have followed up to hear the results of my lab work... 8 TIMES!!!!!!!!!!!!!!!!!!!!!! He has failed to call me back for over 2 weeks and his secretaries clearly must be unorganized. I'd probably have an easier time getting a hold of my dead grandma.

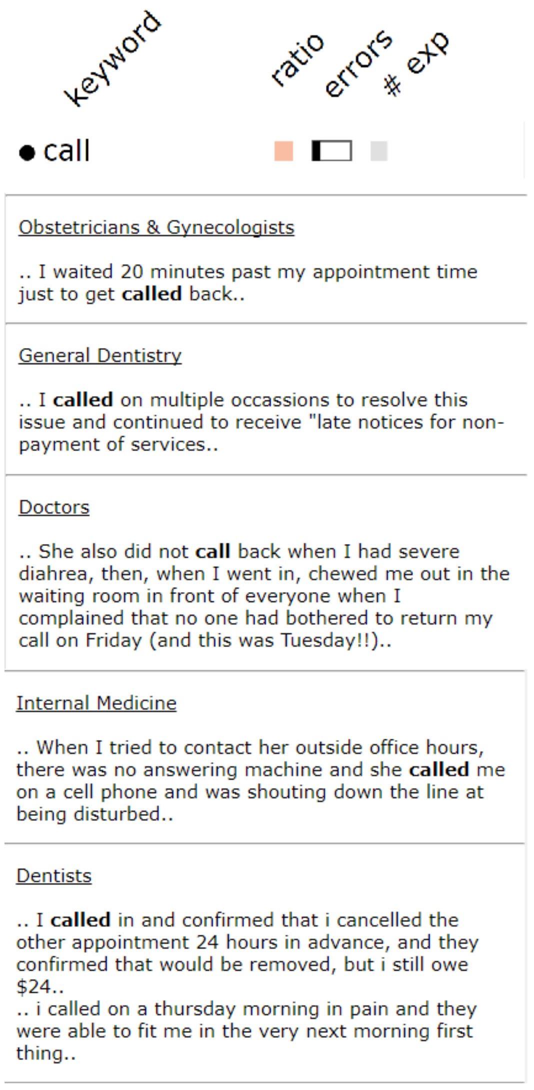
Figure 7.2. This figure shows some false positive explained by the explanation "*Dr.*". As you can read, those are negative reviews that use the title "*Dr.*" without mentioning the doctor name, making an exception to the rule and confusing the machine learning classifier.

something positive, but they tend to refer to the practitioner as "*the doctor*", using a more generic terminology, when writing a negative review.

Through this inspection, we can also better understand how the few false positives explained by "*Dr.*" happen. When we look at the raw text of selected false positive cases, displayed in Figure 7.2, we see that most of them represent rare cases where the patient is using the word "*Dr.*" without using the doctor name (e.g. "*Dr. is very nice but the staff is rude.*"). The model therefore tends to be confused by outlier cases in which a contradiction of the rule is present.

Another interesting word is the word "*call*", which, as shown in the figure, tends to predict negative reviews, even though with a somewhat high error rate. While not immediately obvious why this word leads to negative reviews, we figure out, through the visual inspection enabled by our application, that reviewers tend to mention cases in which they have called the doctor's office and received poor assistance (Figure 7.3). A similar case is the feature "*dentist*" (shown in 7.4), which also tends to predict negative reviews. The feature however has also a somewhat higher error rate which means many positive reviews are misclassified as negative. Through closer inspection, we realize that the classifier is not able to disambiguate cases in which the word "*dentist*" is used in a positive context such as "*fantastic dentist*" (Figure 7.4).

The most common words in explanations with hundreds of associated documents are "*great*" and "*recommend*", top explanation term in Figure 7.1. The majority of those documents are true positive, but we can still find and select the few false positive the model generates. Selecting all the misclassified positive reviews containing "*great*" we spot an interesting problem: some of the reviewers sometime use the word "*great*" in a sarcastic way, making the detection of a negative connotation too hard for the classifier. An example of a miscalssified sarcastic negative review is shown in Figure 7.5. Similarly, we also notice that the word "*recommend*" is sometime used in conjunction with a negation, that is, "*not recommend*", making it once again too hard for the classifier to make the correct prediction with its current configuration. Some example of these cases are shown in Figure 7.6. An interesting aspect of this last case is that the manual inspection of misclassified instances can lead to ideas on how the classifier could be improved. For instance, in this last case equipping the classifier with means to detect negation may lead to improved performance.



● call

Obstetricians & Gynecologists

.. I waited 20 minutes past my appointment time just to get **called** back..

General Dentistry

.. I **called** on multiple occassions to resolve this issue and continued to receive "late notices for non-payment of services..

Doctors

.. She also did not **call** back when I had severe diahrea, then, when I went in, chewed me out in the waiting room in front of everyone when I complained that no one had bothered to return my call on Friday (and this was Tuesday!!)..

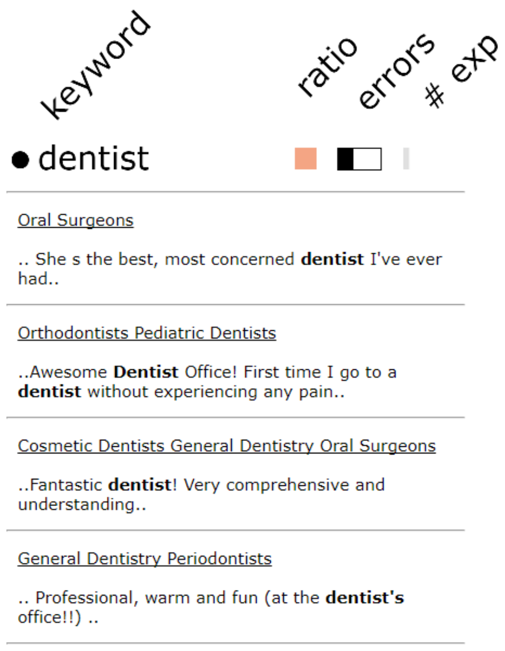
Internal Medicine

.. When I tried to contact her outside office hours, there was no answering machine and she **called** me on a cell phone and was shouting down the line at being disturbed..

Dentists

.. I **called** in and confirmed that i cancelled the other appointment 24 hours in advance, and they confirmed that would be removed, but i still owe $24..
.. i called on a thursday morning in pain and they were able to fit me in the very next morning first thing..

Figure 7.3. In this figure some of the true negative reviews explained by "*call*" are listed. By reading you see how the reviewers are complaining about issues they had during the doctor phone service.



● dentist

Oral Surgeons

.. She s the best, most concerned **dentist** I've ever had..

Orthodontists Pediatric Dentists

..Awesome **Dentist** Office! First time I go to a **dentist** without experiencing any pain..

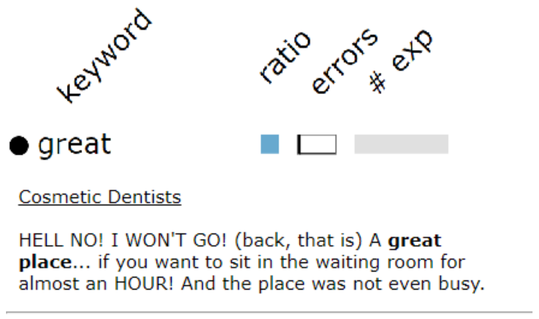
Cosmetic Dentists General Dentistry Oral Surgeons

..Fantastic **dentist**! Very comprehensive and understanding..

General Dentistry Periodontists

.. Professional, warm and fun (at the **dentist's** office!!) ..

Figure 7.4. This figure shows false negative reviews explained by "*dentist*". It is unclear why the classifier used the term "*dentist*" to predict those negative labels and thanks to *Rivelo* we are able to recognize the classifier mistake. First "*dentist*" is incorrectly used to explain many false negatives, as shown in the error bar of which a great portion is black. Second those false negatives reported in figure are clearly positive reviews with strong positive terms like "*awesome*", "*best*" and "*fantastic*". From this observation a machine learning developer could investigate what went wrong in the prediction of those instances, which should have easily been predicted positive.

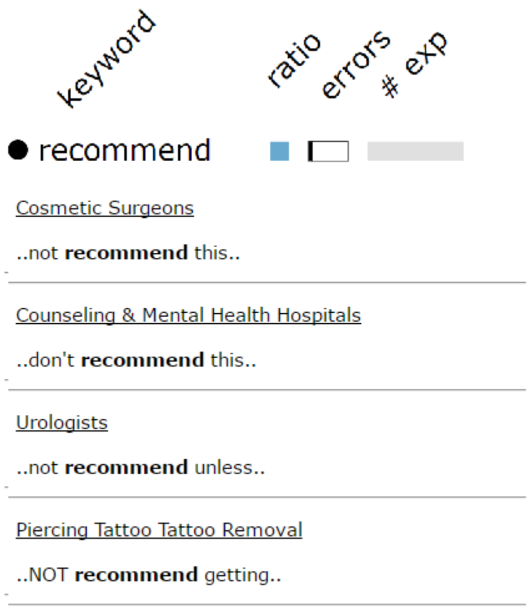


● great

Cosmetic Dentists

HELL NO! I WON'T GO! (back, that is) A **great place**... if you want to sit in the waiting room for almost an HOUR! And the place was not even busy.

Figure 7.5. This figure shows an interesting case that *Rivelo* was able to point out. The shown review explained by "*great*" is false positive because the reviewer used "*great*" in a sarcastic way, which the classifier was not able to detect.



Cosmetic Surgeons

..not **recommend** this..

Counseling & Mental Health Hospitals

..don't **recommend** this..

Urologists

..not **recommend** unless..

Piercing Tattoo Tattoo Removal

..NOT **recommend** getting..

Figure 7.6. *Rivelo* is able to point problems by showing interesting prediction cases, prompting ideas to improve the classifier. For example in this figure we can read some false positive explained by "*recommend*". The classifier missclassified those reviews because it was not able to detect negation in those sentences. This way *Rivelo* is suggesting to improve the classifier with negation detection.

Chapter 8

User Study

In this chapter we describe a small user study on the visual interface of *Rivelo*. Given the scarcity of resources, this user study does not present complete results. Therefore the outcome is just indicative, aiming at pointing out evidence that *Rivelo* can be used and understood by different users. The process followed in this study comprehended the following steps: collecting demographic information of the users (5 minutes), explaining how *Rivelo* works (10 minutes), training the users with a supervised practical tutorial on the interface (15 minutes), leaving the users to freely interact with the tool (20 minutes), making a quick interview asking questions regarding their experience with the tool (15 minutes).

8.1 Recruitment and Setting

Participants have been recruited from graduate students of Sapienza University of Rome, of which around 40% is not familiar with machine learning and the other portion is a becoming expert. A total of 14 individuals have been recruited (9 males, 5 females) through personal messages. The average age is 24.78 years. The studies always occurred on the same system with a 15-inch screen with full-HD resolution (1920x1080).

8.2 The Procedure

The user study starts by collecting general demographic information such as gender, age, education level and country of origin. After that the author gives an explanation of how *Rivelo* works. *Rivelo* is loaded with the *Yelp* data-set we have already seen in Chapter 7. Then the author gives also a quick live demonstration by interacting with the interface. After the demo is over, the participant is asked to try the tool and ask questions. When the training has ended the author leaves the room to avoid distractions and biases. The subject has 20 minutes to find 5 observations on the classifiers global behaviours through interaction with Rivelo. When the time is over an interview takes place to discover what the user was able to find. The entire time length of the user study is of about 1 hour.

8.3 Results

The results from the final interviews with the user study participants are satisfying. Most of the subjects were able to find not only the classifier behaviours already mentioned in Chapter 7, but also some decision patterns the author was not able to notice.

Most of the subjects correctly start by inspecting terms from the *features panel* that do not give an immediate reason to why they were used by the explanations. Then they select an explanation relative to many documents and read through the raw data in search for recurrent situations which could describe a global behaviour of the classifier.

In Figure 8.2 we see a grid displaying the terms that were most selected by the user study participants. Most of those selected terms are verbs or nouns and not adjectives. Some examples of adjectives used in explanations are “*great*”, “*friendly*”, “*rude*” or “*horrible*”, which all have a comprehensible role in the classifier. The user study participants were more focused in inspecting unexpected explanation terms such as: “*money*”, “*Dr.*”, “*desk*”, “*staff*”, “*doctor*” and “*day*”. Those terms, mostly nouns, were apparently used by the classifier, but it is not directly clear on why they are meaningful for a classification. Therefore users check if the classifier used those terms correctly or if there is a recurrent mistake in those classification.

Except the cases we have already seen in Chapter 7, we can add some new observations made by the users. Some of the users noticed that the term “*fine*” is used unexpectedly to identify negative reviews. By reading through the raw data, participants realized that “*fine*” is used in sentences describing something that was “*just fine*”, while if the reviewer wants to be positive about some service she would use the adjective “*great*”. Another example regards the use of two apparently similar features: “*bill*” and “*price*”. From inspecting the *features panel*, a user was interested in those two features because, even though they are semantically similar, the model is using “*bill*” to identify negative reviews while “*price*” to identify positive ones. From inspecting the *raw data panel*, the user realized that generally people use “*expensive bill*” when they consider the doctor service expensive and “*low price*” otherwise.

In Figure 8.3 we can see another heatmap describing the model bugs detected by the user study participants. Each column of the matrix representation is relative to a bug category which we will describe now from the most detected to the least detected.

- **Syntax detection**:
  The most common observation regarding the model regards the syntax detection. That is the ability of the model to understand the logic structure within the text data. Even to participants not familiar with machine learning, *Rivelo* was easily showing this lack of the model in detecting the role of a term in its context. That derives from the text data representation in *bags of words*, which take into account just which words are present but not where in the sentence are placed. The more expert participants suggested that the model should use *natural language processing* (NLP) techniques.

- **Need for *n-grams* features**:
  In checking out terms like "*desk*", "*cover*", "*doctor*" and other noun features, the users see that the greatest amount of missclassifications is relative to the explanation containing just that single term (for example the explanation "*desk*" or "*cover*" in Figure 8.1). Where instead one of those terms is together with another different term, for example "*rude*" + "*desk*" or "*insurance*" + "*cover*", the amount of mistaken predictions decreases. To overcome this problem, without changing the approach given by the bag of words representation, several participants suggested the use of *n-grams* features. That is features representing not just single terms but phrases. This way it is possible to understand for example to what noun an adjective is related to. For example a *2-gram* feature might be "*great doctor*" which is quite different from "*great pain*".

- **Need for capitalization of features**:
  By inspecting the *raw data panel*, the user study participants noticed that oftentimes reviewers tend to write in caps when they are upset or enthusiastic about something. Therefore they suggested that the model should differentiate uppercase features, rather than merging them with the lowercase ones.

- **Negation detection**:
  This case has been already covered by the Use Case in Chapter 7 and it regards the ability of the model to detect negation within a sentence. For example by inspecting false positives explained by "*recommend*", participants were able to find many case in which the reviewers wrote "*I do not recommend*".

- **Overfitting the train set**:

The participants with more knowledge of machine learning were able to find features used by the model that were clearly a symptom of overfitting the train set. That is features linked to names of doctors like "*Kaiser*", "*April*" or "*Lake*". Apparently those doctors had many reviews with same label in the train set and as a consequence the model used their names to identify positive or negative reviews.

- **Stemming bugs**:
  Some of the users localized problems in the stemming of the model. This technique basically is the removal of the suffix and prefix of words. This is done in order to recognize the same word when it appears in different conjugation or forms. In certain cases the stemming algorithm might make mistakes and this is exactly what users could detect. For example a negative word like "*careless*" is transformed into "*care*", a positive word. By showing false positives explained by "*care*", *Rivelo* points the bug to the user showing "*careless*" in the raw data. Theoretically the user could fix the stemming bug and reinspect the new predictions with *Rivelo*, until no cases like this one are present.

- **Punctuation detection**:
  In some cases punctuation is useful to understand emphasis given by the reviewers. For example if a reviewers writes "*great!!!*" instead of just "*great.*", the model should detect this and give an even higher positive prediction. Three of the participants noticed that the model is not able to detect such emphasis and that "*!!!*" is used as a separate features confusing even more the model.

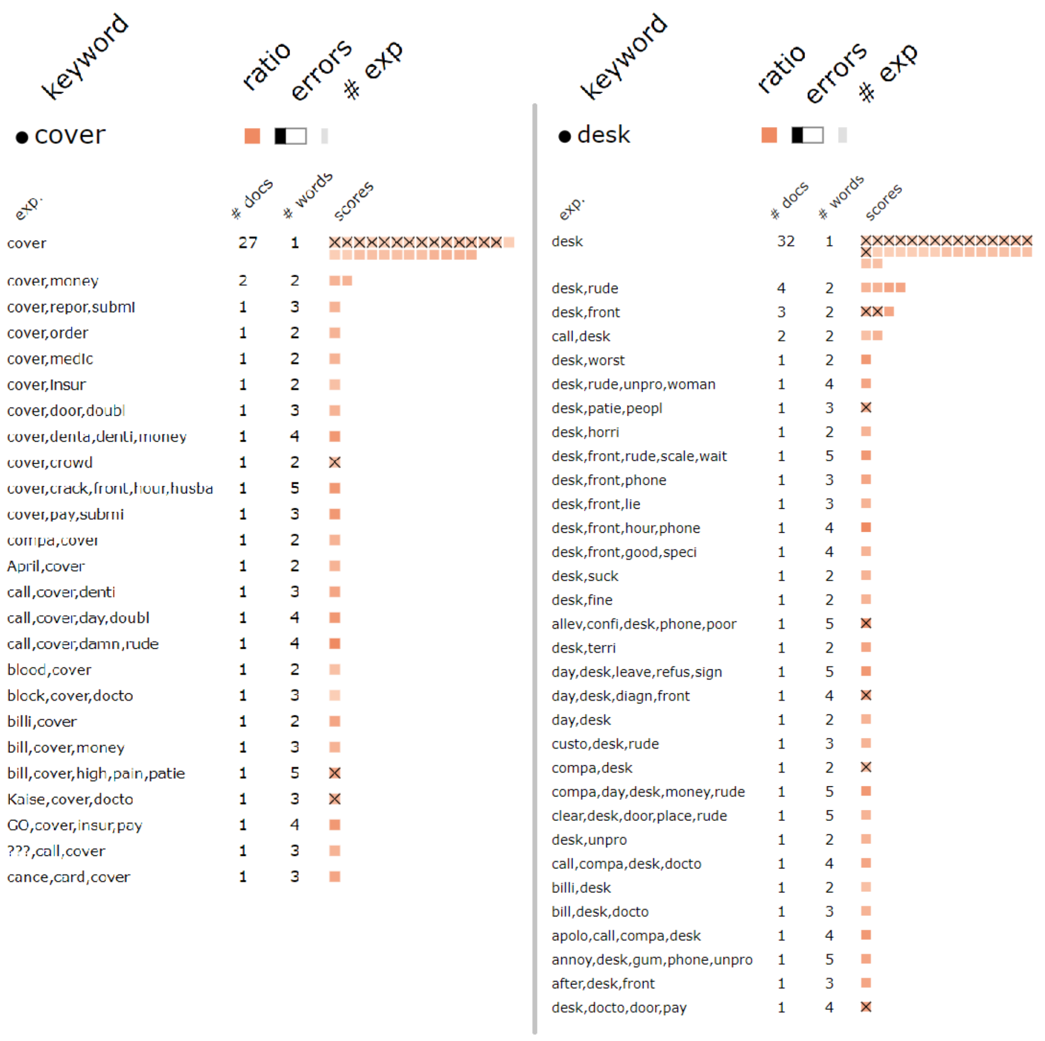
- **Predictions for long reviews**:
  Oftentimes reviews are really long, containing a great amount of features and complicated statements. Participants were able to notice that many miscalssifications are linked to those reviews, where the model is not able to understand the prolix argument. Furthermore two users noticed that the model explanations for such long reviews were oftentimes made of a single feature. The users linked this phenomenon to the model inaccuracy.

- **Time dependency**:
  An interesting observation by two users is related to the time context in which features are used. For example features like "*problem*" and "*pain*" if used with a past verb they are positive features (e.g. "*I had pain, now it's gone*"), while if used with a present verb they become negative features (e.g. "*I still have pain*"). Instead features like "*happy*" and "*love*" have positive meaning in the present while negative one in the past (e.g. "*I used to love this place*" vs "*I love this place*"). User study participants noticed that the underlying model

is not able to detect time dependency of a used features and this is source of misclassifications.

- **Sarcasm detection**:
  Just one of the participant found a case of undetected sarcasm, of which we have already talked about in the Use Case of Chapter 7.

We have seen a practical example of *Rivelo* users not just being able to find the global behaviours shown in the use case of Chapter 7, but also to add new interesting observations on the *Yelp* use case. Therefore we concluded the user study gave a positive result. Despite this, many are the upgrades we still need to make to *Rivelo*. In the next chapter we will see some of those possible improvements.



● cover ■ ◧ ▯

exp.	# docs	# words	scores
cover	27	1	☒☒☒☒☒☒☒☒☒☒☒☒☒☒■ ■■■■■■■■■■■■
cover,money	2	2	■■
cover,repor,submI	1	3	■
cover,order	1	2	■
cover,medIc	1	2	■
cover,Insur	1	2	■
cover,door,doubl	1	3	■
cover,denta,denti,money	1	4	■
cover,crowd	1	2	☒
cover,crack,front,hour,husba	1	5	■
cover,pay,submi	1	3	■
compa,cover	1	2	■
April,cover	1	2	■
call,cover,denti	1	3	■
call,cover,day,doubl	1	4	■
call,cover,damn,rude	1	4	■
blood,cover	1	2	■
block,cover,docto	1	3	■
billi,cover	1	2	■
bill,cover,money	1	3	■
bill,cover,high,pain,patie	1	5	☒
Kaise,cover,docto	1	3	☒
GO,cover,insur,pay	1	4	■
???,call,cover	1	3	■
cance,card,cover	1	3	■

● desk ■ ◧ ▯

exp.	# docs	# words	scores
desk	32	1	☒☒☒☒☒☒☒☒☒☒☒☒☒☒☒ ☒■■■■■■■■■■■■■■ ■■
desk,rude	4	2	■■■■
desk,front	3	2	☒☒■
call,desk	2	2	■■
desk,worst	1	2	■
desk,rude,unpro,woman	1	4	■
desk,patie,peopl	1	3	☒
desk,horri	1	2	■
desk,front,rude,scale,wait	1	5	■
desk,front,phone	1	3	■
desk,front,lie	1	3	■
desk,front,hour,phone	1	4	■
desk,front,good,speci	1	4	■
desk,suck	1	2	■
desk,fine	1	2	■
allev,confi,desk,phone,poor	1	5	☒
desk,terri	1	2	■
day,desk,leave,refus,sign	1	5	■
day,desk,diagn,front	1	4	☒
day,desk	1	2	■
custo,desk,rude	1	3	■
compa,desk	1	2	☒
compa,day,desk,money,rude	1	5	■
clear,desk,door,place,rude	1	5	■
desk,unpro	1	2	■
call,compa,desk,docto	1	4	■
billi,desk	1	2	■
bill,desk,docto	1	3	■
apolo,call,compa,desk	1	4	■
annoy,desk,gum,phone,unpro	1	5	■
after,desk,front	1	3	■
desk,docto,door,pay	1	4	☒

Figure 8.1. The participants of the user study, while selecting terms with an high error rate (black and white bar), noticed that the higher amount of misclassifications often is relative to the explanation with a single noun or verb. They were able to detect this from the visualization in the *explanations panel*, where misclassifications are depicted as little cells with a cross symbol on top. In the figure we can see this for the terms "*cover*" and "*desk*". This describes an overall behaviour of the classifier: oftentimes the classifier better detects, within the review context, the role of single nouns or verbs when it is also present another meaningful term. For example the explanation "*desk*" is relative to many misclassifications, while the explanation "*desk*" + "*rude*" relates to all true negatives.

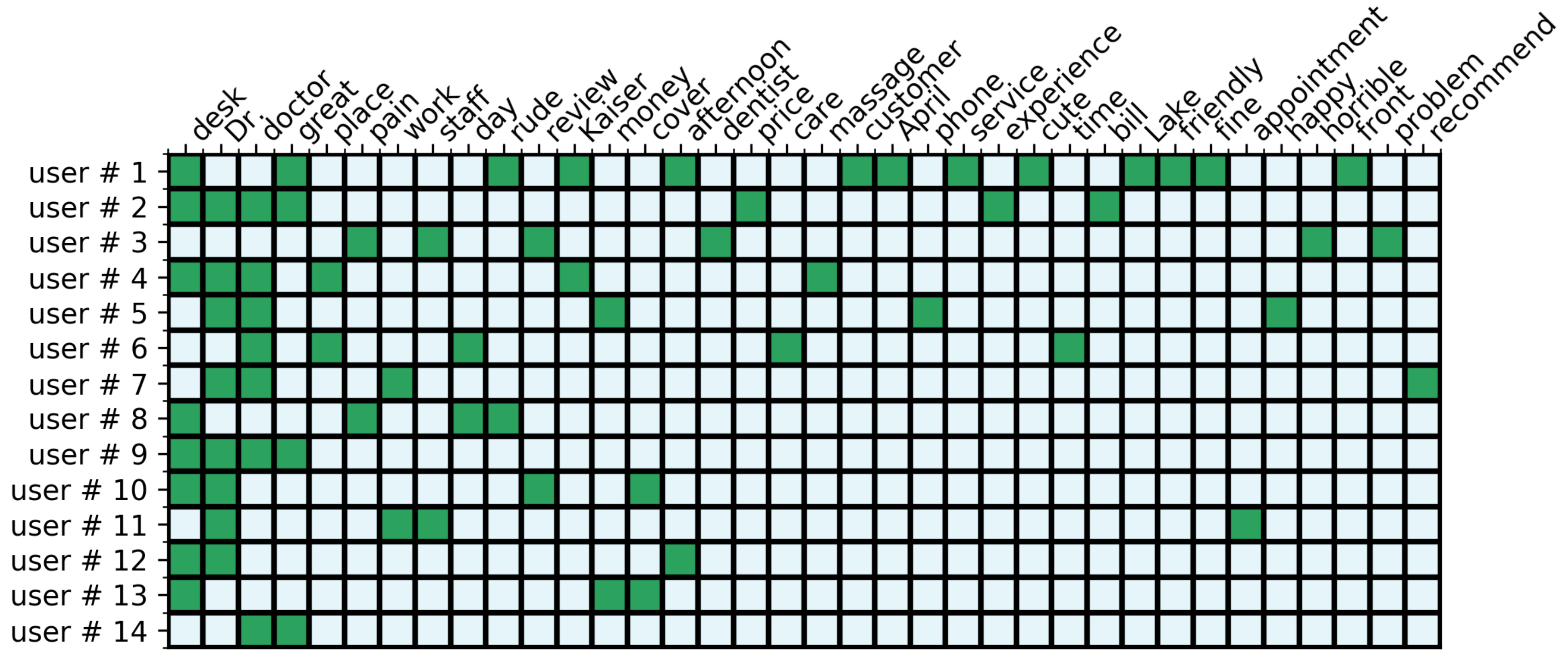


Figure 8.2. In this figure we display all the features used by the 12 participants. Even though participants could choose from a broad variety of options from the *features panel*, this heatmap shows some recurrent features picked by users to make observations on the model. Users are sorted on rows from the one picking more features to the one picking less features, while columns are representing the features sorted from the most picked to the least picked. A green cell shows when a given row-user is picking that column-feature. As we can see the three most picked features are “*Dr.*”, “*desk*” and “*doctor*”.

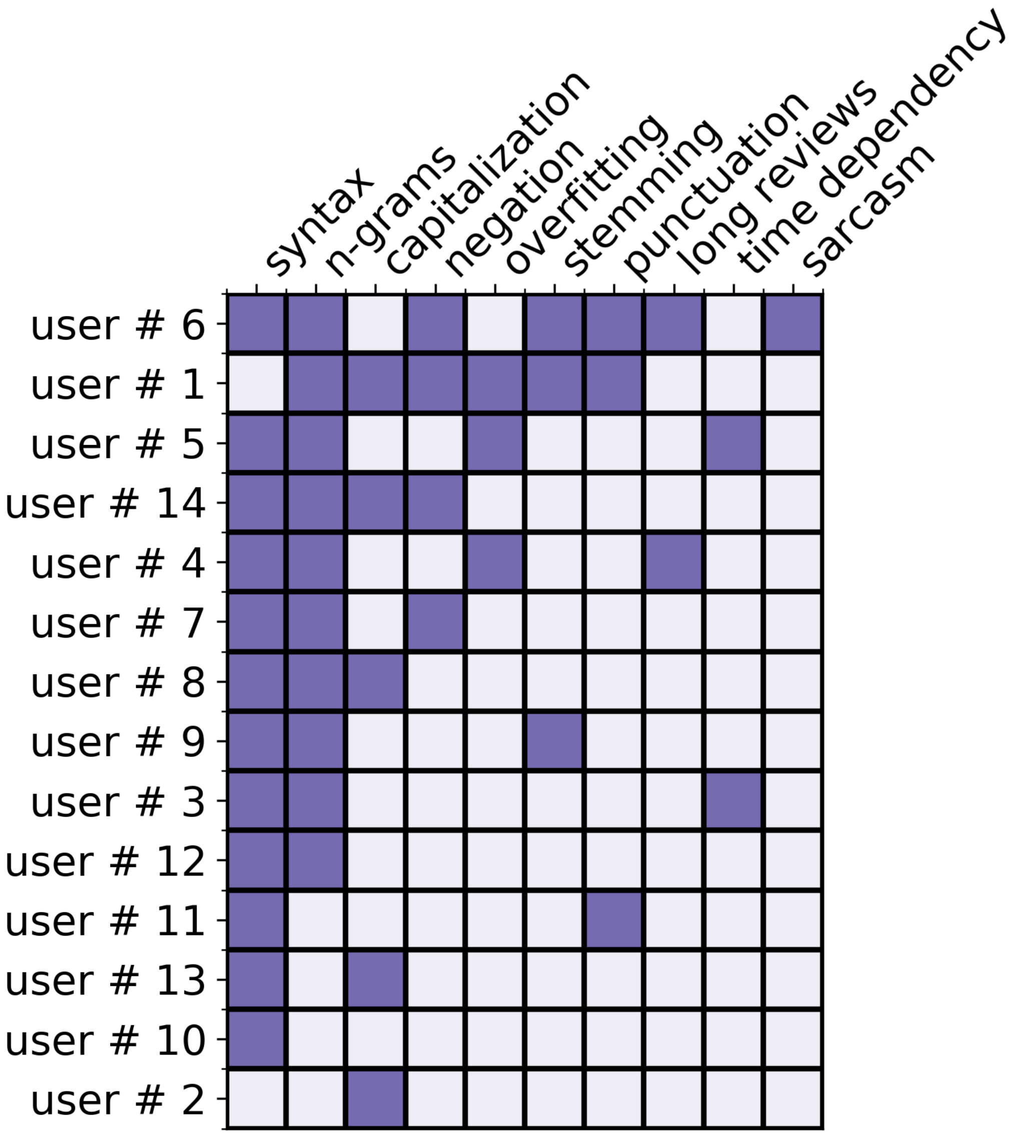


Figure 8.3. In this figure we display all the category bugs detected by the 12 participants. Users are sorted on rows from the one detecting more bug types to the one detecting less bug types, while columns are representing the bug type sorted from the most detected to the least detected. A purple cell shows when a given row-user is detecting that column-bug type. The bug categories are described in detail in Section 8.3.

Chapter 9

Discussion and Future Works

Our use case shows how the proposed solution can help figure out major decisions made by the classifier, spot potential issues and possibly also help derive insights on how problems can be solved. The system, however has, in its current implementation, a number of relevant limitations, that we discuss below.

First, *Rivelo* works exclusively with binary classification and binary feature sets. While this specific configuration covers a large set of relevant cases (e.g., we tested the system with a medical data set describing drugs administered to patients in emergency rooms to predict admissions), many other relevant cases are not covered; notably cases in which features or the predicted outcome are not binary. To solve this problem we will need to develop explanations and design visual representations able to handle the more general case of non-binary features and multiclass outcomes. While an extension of the technique to multiclass classification is not trivial extending to numerical input data could be achieved by adopting techniques like those proposed by Prospector [15] and LIME [25].

Second, the descriptors we use to compare selected instances in terms of their similarity do not lead to an optimal solution. Ideally, we would like to more directly explain what makes instances with similar configurations have differing outcomes and vice-versa. One potential solution we will investigate is to train local models to rank the features in the descriptors in ways that highlight the differences that lead to different outcomes (e.g., instances with the same explanation but different outcome). Another possible extension is to provide a scatter-plot visualization using a projection of the selected instances. This task raises various problems. One of those is visually representing a great quantity of instances without overwhelming the user. In the dense visualization of instances the user would need to differentiate them by number of features, length of explanations, prediction value, predicted label, ground truth label and more properties. A solution might be a technique similar to the technique used by Piava *et al.* [22]. To better compare a large number of instances we could use a projection that uses dimensionality reduction, as for example t-SNE [32]. This way we could visually correlate instance similarities and outcomes. Multidimensional projections however often suffer of several distortion effects that reduce the trustworthiness of the visualization. In addition, they do not provide a direct relationship between the original data space and the trends observed in the projections, making it harder to understand the root causes of observed issues. A possible similarity metric could be the Jaccard distance which uses the ratio of the vector intersection over their union. At the same time the visualization could branch instances with similar score together.

Another important limitation is our focus on understanding one single model at a time. Often important insights can be generated by comparing multiple models. We plan to explore how our technique can be extended to multiple model comparisons and see what advantages it may provide. That is challenging because it will require to deal with multiple explanations for each instance. In fact, given a single instance, we would have a different explanation for each model we want to compare.

Finally we tailored Rivelo on an explantaion-driven approach. However explanations corresponding to only single instances occur quite often. This reduces the effectiveness of aggregation. Therefore a future work might be to add an optional instance-driven approach. By selecting a feature in a first panel the tool would display a single aggregation of all instances containing that feature in their explanation, sorting them by different properties and not just by explanation. Furthermore the user would also be able to select and compare instances with different explanations.

Chapter 10

Conclusion

In this thesis, we have shown how a visual explanation workflow can be used to help people make sense of and assess a classifier using a black-box approach based on instance-level explanations and interactive visual interfaces. Our system called *Rivelo* aggregates instances using explanations and provides a set of interactions and visual means to navigate and interpret them. The work has not been validated with a complete and professional user study, given the lack of resources. Further investigation is needed to fully understand the effectiveness of our approach, potential limitations and the extent to which it helps analysts reach useful and accurate conclusions. For this reason, we intend to evaluate the system in the near future through a series of detailed user studies involving analysts and domain experts with a range of expertise in machine learning and in the application domain.

By looking at all these possible future works, we see how *Rivelo* gave us the opportunity to dive in this expanding field. As the demand for transparency of model decisions grows, more techniques are needed to build trust and confidence for its users. We presented a work-flow enabling domain experts and machine learning experts to gain insights into the decision making process of complex models. Even though there is room for improvements we showed that visual analytics can be used to explore explanations effectively. By proposing this explanation-driven approach, we showed how machine learning explanations are useful in visual analytics.

Bibliography

[1] Amershi, S., Chickering, M., Drucker, S. M., Lee, B., Simard, P., and Suh, J. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 337–346. ACM (2015).

[2] Amershi, S., Fogarty, J., Kapoor, A., and Tan, D. Effective end-user interaction with machine learning. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pp. 1529–1532. AAAI Press (2011).

[3] Amershi, S., Fogarty, J., and Weld, D. 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, pp. 21–30. ACM (2012).

[4] Baesens, B., Setiono, R., Mues, C., and Vanthienen, J. Using neural network rule extraction and decision tables for credit-risk evaluation. *Manage. Sci.*, **49** (2003), 312.

[5] Breiman, L. Random forests. *Mach. Learn.*, **45** (2001), 5.

[6] Brooks, M., Amershi, S., Lee, B., Drucker, S. M., Kapoor, A., and Simard, P. Featureinsight: Visual support for error-driven feature ideation in text classification. *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*, **00** (2015), 105.

[7] Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N. 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, pp. 1721–1730. ACM (2015). ISBN 978-1-4503-3664-2.

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

[9] Freitas, A. A. Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, **15** (2014), 1.

[10] Gunning, D. Explainable Artificial Intelligence (XAI). `http://www.darpa.mil/program/explainable-artificial-intelligence` (2017). [Online; accessed 2017-April-7].

[11] Kahng, M., Fang, D., and Chau, D. H. P. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, HILDA '16. ACM (2016).

[12] Keim, D. A. and Kriegel, H.-P. VisDB: Database Exploration using Multidimensional Visualization. *IEEE Computer Graphics & Application Journal*, **14** (1994), 40. Reprinted in: Readings in Information Visualization, edited by S. Card, J. Machinaly, B. Scheidermann, Morgan Kaufmann, pages 126-139, 1999.

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

[14] Krause, J., Perer, A., and Bertini, E. Using Visual Analytics to Interpret Predictive Machine Learning Models. *ArXiv e-prints*, **1** (2016). `arXiv:1606.05685`.

[15] Krause, J., Perer, A., and Ng, K. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pp. 5686–5697. ACM (2016). ISBN 978-1-4503-3362-7.

[16] Liu, M., Shi, J., Li, Z., Li, C., Zhu, J., and Liu, S. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, **23** (2017), 91.

[17] Lou, Y., Caruana, R., and Gehrke, J. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pp. 150–158. ACM (2012). ISBN 978-1-4503-1462-6.

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

[19] Martens, D. and Provost, F. Explaining data-driven document classifications. *MIS Q.*, **38** (2014), 73.

[20] Munzner, T., Guimbretiere, F., and Robertson, G. Constellation: A visualization tool for linguistic queries from mindnet. In *Proceedings of the 1999 IEEE Symposium on Information Visualization*, INFOVIS '99, pp. 132–. IEEE Computer Society, Washington, DC, USA (1999). ISBN 0-7695-0431-0.

[21] Munzner, T. and Maguire, E. *Visualization analysis and design.* AK Peters visualization series. CRC Press, Boca Raton, FL (2015).

[22] Paiva, J. G. S., Carlos, S., and Minghim, R. An approach to supporting incremental visual data classification. *IEEE Transactions on Visualization and Computer Graphics*, **21** (2015), 4.

[23] Rauber, P. E., Fadel, S. G., Falcao, A. X., and Telea, A. C. Visualizing the hidden activity of artificial neural networks. *IEEE Transactions on Visualization and Computer Graphics*, **23** (2017), 101.

[24] Ren, D., Amershi, S., Lee, B., Suh, J., and Williams, J. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics*, **23** (2016).

[25] Ribeiro, M. T., Singh, S., and Guestrin, C. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144. ACM (2016).

[26] Seo, J. and Shneiderman, B. Interactively exploring hierarchical clustering results [gene identification]. *Computer*, **35** (2002), 80.

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

[28] Shneiderman, B. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pp. 336 –343 (1996).

[29] Tam, G. K. L., Kothari, V., and Chen, M. An analysis of machine- and human-analytics in classification. *IEEE Trans. Vis. Comput. Graph.*, **23** (2017), 71.

[30] Thomas, J. J. and Cook, K. A. A visual analytics agenda. *IEEE Comput. Graph. Appl.*, **26** (2006), 10.

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

[32] van der Maaten, L. and Hinton, G. E. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, **9** (2008), 2579.

[33] Vellido, A., Martín-guerrero, J., and Lisboa, P. J. G. Making machine learning models interpretable. In *In Proc. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, vol. 12, pp. 163–172 (2012).

[34] Wattenberg, M. Visual exploration of multivariate graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pp. 811–819. ACM, New York, NY, USA (2006). ISBN 1-59593-372-7.

[35] Wilkinson, L., Anand, A., and Grossman, R. Graph-theoretic scagnostics. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, pp. 21–. IEEE Computer Society, Washington, DC, USA (2005). ISBN 0-7803-9464-x.

[36] Yang, J., Ward, M. O., and Rundensteiner, E. A. Interring: An interactive tool for visually navigating and manipulating hierarchical structures. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, INFOVIS '02, pp. 77–. IEEE Computer Society, Washington, DC, USA (2002). ISBN 0-7695-1751-X.