Random forest visualization

Kuznetsova, N.I.

Award date:
2014

Eindhoven University of Technology

Master Thesis

Random Forest Visualization

Author:
Natalia Kuznetsova

Supervisor:
dr. Michel Westenberg

Defense committee:
dr. K. Buchin
ir. K. Dinkla
ir. S.J. van den Elzen

Visualization group
Department of Mathematics and Computer Science

August 2014

Abstract

Classification is the process of assigning a class label to an observation based on its proprieties or attributes. A classification algorithm is applied to a data set, producing a model. By studying the model, insights about the data set structure can be gained. The benefits that a model can bring depend on the model. In this work, a Random Forest model is used for the analysis of data. A Random Forest model is explored by means of visualization. The results include this report and the prototype of a visualization analysis tool.

The tool, named ReFINE for Random Forest INspEctor, consists of several visualizations for a Random Forest model. ReFINE provides visualizations for Random Forest components - trees, and its special feature: proximity measure, variable importance, interactions and prototypes. Each of these aspects is presented with a different visualization technique; all the visualizations are integrated together to show the connections between them and allow a user to discover patterns in data sets. The effectiveness of the approach is validated with various data sets, including generated and real data.

As a result, ReFINE allows to investigate data, its most importance variables, theirs split points, connection between instances and their distribution.

Acknowledgements

I would like to thank the members of my project group, namely my supervisor Michel Westenberg, Kasper Dinkla and Stef van den Elzen for guidance during the project, valuable ideas and all their comments. Furthermore, I would like to thank Jeroen de Ridder from Delft University of Technology for participation in the project and valuable ideas. My gratitude to Jarke J. van Wijk for the help during the topic choice. Special thank to the members of my assessment committee, the member of my project group and Kevin Buchin, for their time.

And the last but not the least, many thanks for my family and friends for constant support and encouragement.

Contents

List of Figures

Chapter 1

Introduction

Classification is one of the major machine learning methods. Classification can be regarded as the assignment of a label to an observation based on attributes. Classification application examples include spam filtering and face detection. A classification training algorithm takes historical data, referred to as a training set, as input to build a classifier. After the classifier is built, it can be used to assign class labels to new instances. Thus, classification is a form of supervised learning as it requires a training set to be marked by a domain expert. The classification process is illustrated in Figure 1.1. Observations for a training set come from the real world. After that, the training observations are marked by a domain expert and processed by a classification algorithm. The resulting model is used to classify new observations.

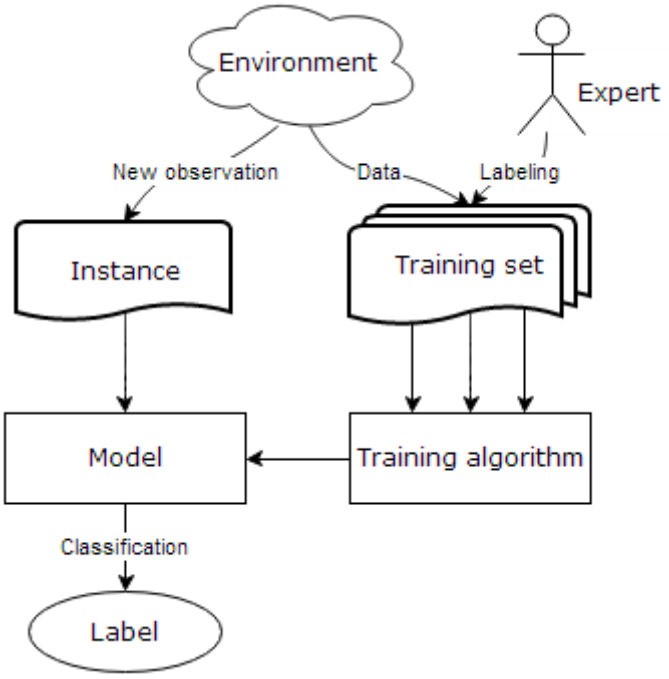


FIGURE 1.1: Supervised classification process. Observations come from the "Environment". After that, the observations are marked by a domain expert, producing a training set. The training set is processed by a classification algorithm to build a model. After the model is built, it is used to classify new instances, coming from the real world

The goal of classification is to assign a class to a new observation with the lowest error. Thus, experts are concerned about the fitness of data. Fitness measures how well a model describes the unseen data. There are different methods to measure the fitness of a model. Moreover, a model brings more advantages to an expert: it can help to understand input data. Every algorithm is classifying data by making decisions based on data values. Those decisions can be used to understand the input data structure, the importance of attributes and the dependence of those attributes to a classification label. The benefits, that a model brings, depend on the algorithm used to construct it.

Supervised classification knows various learning algorithms, including decision trees, neural networks and regression. A decision or classification tree is a classifier that makes a decision based on inspecting one attribute at each internal node. The classifier is built by expecting all attributes and choosing one at every step of construction. This decision is made according to an impurity criterion. For a single data set, multiple trees can be constructed. Constructing an optimal tree is proven to be an NP-complete problem [1]. Several decision trees can be combined to achieve better performance. The idea of combining multiple weak classifiers which individually performs slightly better than random guessing, is implemented in ensemble learning. Ensemble learning method combines multiple classifiers to improve performance. Decision forest, also known as Random Forest, is an algorithm that combines multiple decision trees. The algorithm was introduced by Breiman and Cutler in 1995 [2]. Random Forests can be used for the classification problem as well as for regression problems. The Random Forest classifiers are currently used in many areas of classification and pattern recognition, including medical CT scans [3] and face and body recognition [4].

Data sets usually contain a large number of attributes and it is hard to study them without any preprocessing. A Random Forest model can be used to aggregate the most important structural points of data. Moreover, the Random Forest model can give another advantages for an expert to understand the data structure, such as variable importance (Chapter 2.4.1), prototypes (Chapter 2.4.3) and interactions (Chapter 2.4.4). All those features can help an expert to get insights about data organization, attributes dependence and classification. Although those features give a glance at what the classifier does, it is still a black box. Thus, an expert needs a way to explore it. One of the ways is through the means of visualization.

1.1 Focus

This work is devoted to the visualization of a Random Forest classifier. The goal of this work is to find appropriate visual representations for a Random Forest. The major focus is on the following aspect:

Components and features of Random Forest that can be effectively visualized.

Random Forest is considered as a single classifier. Thus the individual trees are not exposed to the user. The main concern is if Random Forest' components and features can be effectively visualized such that a user can gain insights into a data set structure. Moreover, it can show attribute importance, interactions between attributes and their relation to the classification.

Generalization of the Random Forest visualization.

As the decision forest is applied in various domains, it is necessary to make the visualization not bounded to a particular domain. The generic visualization can be applied to all domains where Random Forests are beneficial.

Scalability of the Random Forest visualization.

Random Forest, as well as training sets, can have different sizes. There are multiple forms of scalability, in terms of number of instances, classes, trees and attributes. The built tool takes that into account.

The developed prototype, named ReFINE for Random Forest INspEctor, is capable of handling synthetic and real data sets with nominal and numeric features. The tool provides several connected panel, one for each aspect of Random Forest. Each panel presents a visualization for a Random Forest model. By interacting with the panels, a user can make observations and gain insights into data organization and structure.

1.2 Outline

The following chapter gives essential background on decision trees and Random Forest. In Chapter 3 the problem statement is given, as well as the overview of existing solutions. Chapter 4 justifies the main design decision made for the visualization solution. In Chapter 5 developed tool with described design decisions is presented, followed by the evaluation of the tool with various use cases in Chapter 6. Chapter 7 concludes the work with the most important results and gives the directions for future work.

Chapter 2

Background

This chapter contains an overview of the background on decision trees and Random Forests. A more detailed description of Random Forest can be found in [2]. First, a short description of decision trees is given. The idea of the decision forest, its algorithm and features are described in this chapter. Finally, a Random Forest model has a number of advantages that are given as a summary for this chapter.

2.1 Decision trees

Decision trees are one of the most widely used models for supervised learning and have been successfully applied in different areas, from medical diagnosis to risk assessment [5]. A decision tree is a supervised learning algorithm. Decision trees can be used for classification or regression. For a classification problem, a data set (or training set) consists of historical data records or *instances*, each marked with a class label, and used to build a decision tree. A classification decision tree learning method predicts the value of a target variable (class variable, *class label* for classification trees) based on other variables (*attributes* or *features*). There are different ways to visualize a classification tree. The most common and intuitive one is a node link diagram. An example of a tree is presented in Figure 2.1 which illustrates a node link diagram for a classification tree for a purchase/no-purchase problem. In this diagram, the internal nodes represent *split points* on attributes and leaf nodes contain the value of a class label. Split points represent decisions based on an attribute. There are two types of attribute values: nominal and numerical. Nominal features take values from a particular set. For example, the education attribute has values of Primary, Secondary and University education. Numerical features' values belong to a continuous scale, such as the age of a person. The edges of a node link diagram represent the partition of the original *data set* into smaller subsets based on split points. For a nominal feature, a split point usually divides the input data set into several subsets depending on the nominal feature's set values. The numerical split point divides the data set into two subsets.

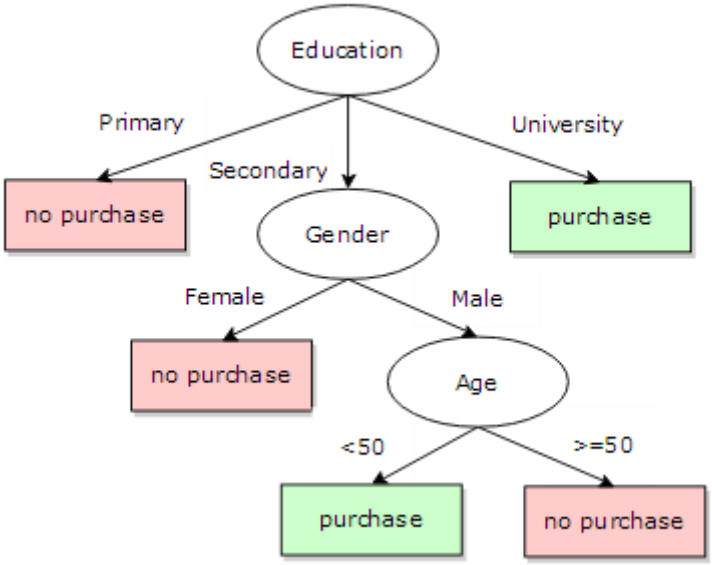


FIGURE 2.1: Example of a decision tree. The decision tree has three internal nodes. At each internal node a decision is made based on an attribute value. To classify a new instance, it is evaluated down the tree, making a decision at each internal node and resulting in one of the leaf nodes.

A prediction can be made whether a customer is going to buy a product or not with the use of the example classification tree in Figure 2.1. Thus, anyone with a primary education is unlikely to buy the product, but a 30-year-old male with secondary education is probably going to purchase the product.

Multiple classification trees can be constructed for a training set. The goal is to build the optimal tree, which is an NP-complete problem [1]. Decision trees are most commonly built in a top-down manner with a greedy algorithm resulting in locally optimal trees [6]. The decision at each step is made according to an impurity measure or splitting criteria. The impurity measure indicates how well a current set of instances is separated by a certain attribute. There are different impurity measures used to construct decision trees, including *Information Gain*[7] and *Gini Measure* [6]. The attribute with the highest value for splitting criteria is selected for the current node. The construction process ends as soon as one of the stopping criteria is met. The stopping criteria can be one of the following:

- All instances of a current subset belong to the same class. A leaf node is marked by the class.
- The number of records have reached a certain threshold. The leaf node is marked by the most frequent class.
- The maximum depth of a tree is reached. The leaf is marked by the most frequent class.

One way to express tree performance is *accuracy*, where the original data is divided into a training set and a test set. The training set is used to construct a decision tree and the test set is used to measure the performance of the built tree. The accuracy is defined as the percentage of correctly classified instances in the test set. There are other methods to measure decision tree performance:

- *size of the tree,*
- *total number of nodes,*
- *depth of the tree,*
- *total number of attributes used.*

For some data sets a decision tree can be very sensitive; if the training data is slightly changed, the resulting tree can be significantly different. Moreover, a decision tree can overfit the data set. Overfitting often results in complex trees that describe each observation of the training set. Overfitting leads to a large *generalization error*. The generalization error is a measure of how well a classifier fits the test data. There are several methods to overcome the overfitting problem. A constructed decision tree can be *pruned*. Pruning is the process of replacing subtrees with leaf nodes [8]. The pruning process results in less structurally complicated trees. Another way is to use an ensemble of classifiers, as will be discussed in the next section.

2.2 Random forest

The idea of using multiple classifiers as a single classifier is implemented with ensembles. In a classification problem, every classifier votes for a particular class. The final decision is the majority of votes. There are different methods of ensemble construction, such as Boosting and Random Forest.

The boosting algorithm is an example of classifier ensembles. Boosting is a general term for iterative algorithms that construct weak classifiers and add them to a final strong classifier with different weights depending of the performance of a weak classifier. A well know example is AdaBoost (Adaptive Boosting). AdaBoost can be built using various classifiers, but decision trees are the most commonly used. AdaBoost inspects all features and selects only the one that improves the prediction accuracy of the model. The method can give a sufficiently low generalization error, but the computational time can be very high for data sets with a large number of features. Random decision forest is another example of ensemble that can help to overcome this.

Decision forest is referred as Random Forest since a random vector is chosen for every tree classifier. The random vector determines a subset of the initial training set for each tree. This method helps to avoid overfitting and decrease the generalization error. Breiman,

the inventor of the Random Forest concept, gives the following formal definition [2], where $\{\Theta_k\}$ is a random vector and $h(x, \{\Theta_k\})$ is a single decision tree:

Definition *A random forest is a classifier consisting of a collection of tree-structured classifiers* $\{h(x, \Theta_k), k = 1, \ldots\}$ *where* $\{\Theta_k\}$ *are independent, identically distributed random vectors and each tree casts a unit vote for the most popular class at input* x*.*

The key element is the independence of each random vector $\{\Theta_k\}$ from $\Theta_1 \ldots \Theta_{k-1}$, but all vectors must share the same distribution. The idea originates from the bagging concept by Breiman. The following definition is given (adopted from [9]):

Definition *Bagging predictors is a method for generating multiple versions of a classifier and using these to get the aggregated classifier.*

This method is frequently used with decision trees. According to the bagging predictors method every tree classifier draws a random bootstrap with replacement from the original set. Bootstrapping is a method of data set resampling. The new sample is formed by randomly picking the samples from the original data set. Some samples are picked several times, others are not selected at all. As a result each classifier is built on a subset of the original training data. The method helps to improve the accuracy of a prediction [9] and provides an additional estimate of performance: *out-of-bag* error (see Section 2.3.1). The out-of-bag error is estimated using the corresponding out-of-bag data. Figure 2.2 shows the partition of data for a Random Forest. First, input data is divided into a training set and a test set. Next, each tree selects a subset from the training set with the bootstrapping method. The tree is trained using the selected data. The remaining part is called out-of-bag data, which is different for every tree.

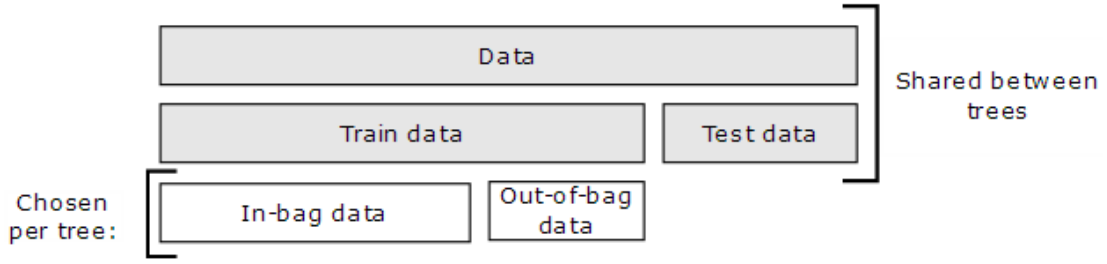


FIGURE 2.2: The data partition for a Random Forest classifier. An original data set is divided into training and test sets. Every tree draws a subset of data from training set, forming "in-bag" data. The remaining part is called out-of-bag data.

Random Forest extends the idea of bagging further by randomizing the feature selection. The original algorithm of Random Forest extends the bagging idea by introducing a Random Feature Selection. The Random Forest algorithm builds multiple decision tree classifiers, each built with Random Tree. The Random Tree algorithm modifies the standard construction of a decision tree with randomization. At every node a subset of

features m are selected from an original set of features M. After this, the best attribute among m is selected as a split point according to an impurity measure. In the original proposal, Breiman used the Gini Index to choose the best variable among randomly selected ones. The trees are built to maximal depth and not pruned. To classify a new instance, the instance is processed to every tree in the ensemble. The resulting class is defined by the majority of "votes".

2.2.1 Construction Algorithm

The generalized algorithm for Random Forest construction is the following (based on [10]):

Algorithm *Random Forest*(x, K, N, M)
1. **for** k = 1 to K
2. Draw a bootstrap sample of size N from the training set
3. Grow a Random Tree Θ_k on the bootstrap data, by recursing on each unvisited node:
4. **while** stopping criteria are not reached
5. Select m variables at random from the set of input variables M
6. Pick the best variable/split-point among the m
7. Split the node into child nodes on the corresponding variable
8. Return the ensemble of trees $\{h(x, \Theta_k), k = 1, \ldots\}$

Where x is a training set and a set of parameters, including total number of trees K, bootstrap sample size N and number of variable selection for each node M.

The aim of combining Bagging with Random Feature Selection is to reduce the correlation between the tree classifiers without reducing the variance too much.

2.2.2 Strength and correlation

Breiman proves that the generalization error converges with increasing number of trees in the forest. For the Random Forest, the upper bound for the generalization error PE is defined as

$$PE \leq \frac{\bar{\rho}(1 - s^2)}{s^2} \tag{2.1}$$

where ρ is the correlation between the classifiers and s is strength of the classifiers. Strength is a fitness measure that shows how accurate the individual trees are. Strength is defined with the use of the margin function mr. The margin function indicates the extent to which the averaged votes for the correct class exceeds the averaged votes for any other class.

$$mr(X, Y) = P_\Theta(h(X, \Theta) = Y) - max_{j \neq Y} P_\Theta(h(X, \Theta) = j) \tag{2.2}$$

for input data X with class Y. P_Θ is the probability over the whole ensemble of the classifiers. Strength is defined as the expected value of the margin function.

$$s = E_{X,Y} mr(X, Y) \tag{2.3}$$

Correlation on the other hand indicates the dependence between the classifiers. The correlation is defined with the use of the raw margin function rmg.

$$rmg(\Theta, X, Y) = I(h(X, \Theta) = Y) - I(h(X, \Theta) = \hat{j}(X, Y)) \tag{2.4}$$

where I is the indicator function (takes value 1 or 0 if the instance belongs to the class or not) and $\hat{j}$ is the most frequent class among incorrect classes, defined as

$$\hat{j}(X, Y) = \arg max_{j \neq Y} P_\Theta(h(X, \Theta) = j) \tag{2.5}$$

Correlation is then defined as the statistical mean between raw margin functions of all pairs of classifiers Θ.

The generalization error will be the lowest if the the ratio $\frac{\bar{\rho}}{s^2}$ is the lowest. The aim is to keep the strength of each classifier as high as possible, while not increasing the correlation of the classifiers too much.

2.3 Random Forest Evaluation

The first thing an expert is interested in is the quality of the model. Random Forest classifier fitness can be evaluated in several ways. The fitness can be determined with test set accuracy, but Random Forest, due to the construction method, has other means to evaluate fitness of the model. *Out-of-bag* error is one of the ways. However, it only gives the estimate of how well the Random Forest performs, but not how it does it.

2.3.1 Out-of-bag error

Each tree is constructed with the use of the bootstrapping with replacement procedure to form a subset of the original data set. Each tree is grown on this subset of the data with random feature selection. The bootstrapping with replacement method selects around two thirds of the original data set for each classifier. So, for an instance there is about one third of classifiers built on the data excluding the instance. For every instance, an out-of-bag (OOB) error is built using the corresponding classifiers. The OOB error estimate is defined as the averaged proportion of misclassified instances over all instances for every class. The error is proven to be unbiased [11].

OOB error is computed during the forest construction and thus requires no additional computation. The error also gives an estimate for the current error rate and tends to overestimate it [2]. The results are compatible with the usage of a training set [11].

2.3.2 Overfitting

The Random Forest tends not to overfit the data, even if the number of trees is relatively high [2]. As the generalization error is defined in terms of correlation and strength ratio, it can be proven using the Law of Large Numbers [2]. As a result, Random Forest can produce a limited value of the generalization error. In other words, after a certain number of trees is built, the generalization error does not decrease by adding more trees. For different data sets, the number of trees required for the generalization error to stabilize is different and dependent on the size of the data set. The result is supported with some case studies [10].

2.3.3 Balancing prediction error

In some data sets the prediction error for different classes can differ dramatically. It can be a result of an unbalanced data set with more instances of one class compared to another. For instance, if the data set contains 90% of class A, then a classifier can achieve pretty good average error rate by always assigning the instance to that class. During Random Forest construction it is possible to assign different weights to classes in order to achieve a better class error rate, but this can result in an increase of overall error rate.

2.4 Random Forest Features

Given a Random Forest certain features can be computed. These features provide insight into data organization and interaction and are used to form the basis for the visualizations in this work.

2.4.1 Variable importance

Using Random Forest, a measure of *variable importance* can be constructed. Variable importance measures the prediction strength of a variable. It is constructed from out-of-bag samples. First, for every tree the OOB samples are classified and the class accuracy is recorded. Next, values of a variable j of OOB data instances are randomly permuted and classified again. The resulting average decrease of the accuracy is used as a variable importance measure for the variable j. Unlike the raw variable occurrence measure, that can be high just due to the random generation, variable importance shows how much the classification depends on a particular variable. Variables can be ranked by relation to the classification using corresponding variable importance measures.

The measure can be used to reduce the number of variables for the classifier. A data set can be trimmed after the first initial run of the classifier on the full data set.

2.4.2 Proximities

An ensemble of decision trees as weak learners gives another advantage. With a decision tree a proximity measure can be calculated for every pair of instances. The proximities are represented as a matrix of size $n \times n$, where n is size of a training set. To construct the proximity matrix, for every tree and every pair of cases (including OOB cases), increase the proximity by one if the cases are classified the same by a tree. The proximity can be calculated as well for the instances of training set and testing set.

A data set for classification is marked by a domain expert. Some instances can be misclassified (given the wrong class label), by mistake or inaccuracy. For a classification problem, it is important to find those misclassified instances, as they contribute negatively to model accuracy. Another source of such noise are *outliers*. Outliers are data instances that are not typical for data distribution. The outliers contribute to the overfitting of a model. Thus, they are likely to be eliminated in order to improve the generalization of a model. With the use of the proximity measure the misclassified instances and outliers can be found, as the proximity matrix shows the connectivity between instances. The ideal case for a training set is when the proximity matrix contains 1 at the $[i, j]$ position if i and j belong to the same class, and 0 otherwise. This means that all the trees agree on the class labels for those two instances. Intermediate values indicate that some trees disagree.

Further more, the proximity matrix can help to handle missing values in the data set more efficiently. The standard way to fill missing values is by averaging for numerical attributes and taking the most frequent value for nominal attributes. All instances are taken into account from the training set. With proximities, only the nearest values are taken to form the missing value.

2.4.3 Prototypes

With the use of a proximity matrix, *prototypes* for every class can be found. A prototype is an abstract instance that represents the class. To find a prototype for a class the following procedure is executed:

- Find a case that has all the k nearest neighbors by proximity marked by the class.
- If the attribute is numeric, the prototype for the attribute is the median among the k nearest instances. The stability of the prototype can be judged by 25th and 75th percentile of the instances.
- If the attribute is nominal, the prototype is the most frequent value among the neighbors.

For every attribute and class, a prototype is computed separately. Thus, a prototype is usually non-existent instance that is common for every attribute in the class.

2.4.4 Interactions

Interaction describes the relation between two or more independent features. Dependence of variables is computed with the correlation measure. Interactions show how attributes are related to each other. In terms of a decision tree, interactions of attributes involve their usage in a particular path. The underlying idea is that highly correlated features are less likely to be used in the same decision tree path. Intuitively, a feature similar (highly correlated) to one being selected already is unlikely to be a good candidate for a splitting point.

2.5 Advantages

In general, a Random Forest classifier has a number of advantages. First, it is suitable for both classification and regression problems. Random Forest can consist of classification or regression trees. It has a high accuracy rate compared to other classification and regression algorithms with low computation cost [12]. For every tree at each node only the subset of initial attributes is selected for a potential split point. As a result, not all features are inspected at every step of tree construction. This can be beneficial if a data set contains a large number of attributes. The classifier gives a sufficient result for different domains and is applied in many of those domains. Every tree in Random Forest is constructed independently. Thus, the construction of a Random Forest can be done in parallel [13].

Moreover, a decision forest classifier provides the estimate of a model fitness without an additional test set. Every tree has OOB data samples, that are used for accuracy testing and additional performance measures. Thus, the classifier does not require an additional test set. OOB data gives another advantage: the variable importance estimate is computed. The variable importance is constructed with OOB data and shows the prediction strength of a variable.

The classifier can help to identify the pathological data, such as misclassified instances and outliers. It can be done with the use of the proximity matrix.

2.6 Limitations

The methods of the Random Forest evaluation in the section 2.3.1 give an estimate to a constructed model accuracy. Furthermore, the features of the model in the section 2.4.1 give additional information about a data set. But none of them provide insights into the Random Forest decision making process. For example, the variable importance gives a rank to each attribute for the classification in general. There is no clear connection to a particular class and valuable split points are not exposed.

Chapter 3

Problem statement

3.1 Problem description

These days the amount of data being produced daily is stunning. Movements are being registered by various cameras, internet activity is being recorded, purchase history is stored. All this data is used in decision making processes. Historical data can help to improve decisions.

The amount of data is growing so rapidly that it can not be processed by a human. Humans are not capable of discovering complex structure in data on their own. Very often it requires to consider thousands of cases or it is necessary to construct relations between dozens of variables. This motivates to use automation of the data exploration process.

Data mining is a process of discovering patterns in datasets. The goal of the field is analysis of data and presenting the results to decision makers. Data mining includes methods of statistics, artificial intelligence, and machine learning. Classification is one technique of data mining. In classification a new instance is assigned with class label based on previously recorded data. The historical data is used to construct a model that explains it.

For the classification problem, data usually consists of instances with a particular set of attributes and a class attribute. The features and aspects of data structure that experts are interested in, can be observed from two points of view: machine learning experts and analysts. An analyst in this case is a domain expert, such a biologist or a doctor.

Analysts are more concerned about the data, using classification algorithms as a tool to explore it. They are interested in data distribution and structure, clusters in data and the relation between attributes and classification. For analysts, two use cases can be identified with focus on:

- *Instances.* The aim is to explore a particular data set, exploring the relations between instances. They are concerned about data noise, such as outliers and misclassified instances.
- *Attributes.* An analyst is aimed at gaining knowledge in the domain area for a particular data set. For example, the relations of genes can be studied based on a data set.

Typically an analyst can solve his tasks by exploring data directly. For example, in a table form or with the use of Data Visualizations. Another way is to use a data mining technique to extract knowledge from the data.

Machine learning experts are focused on classification algorithms and depend on data as the source. Experts are interested in algorithm performance, accuracy and robustness.

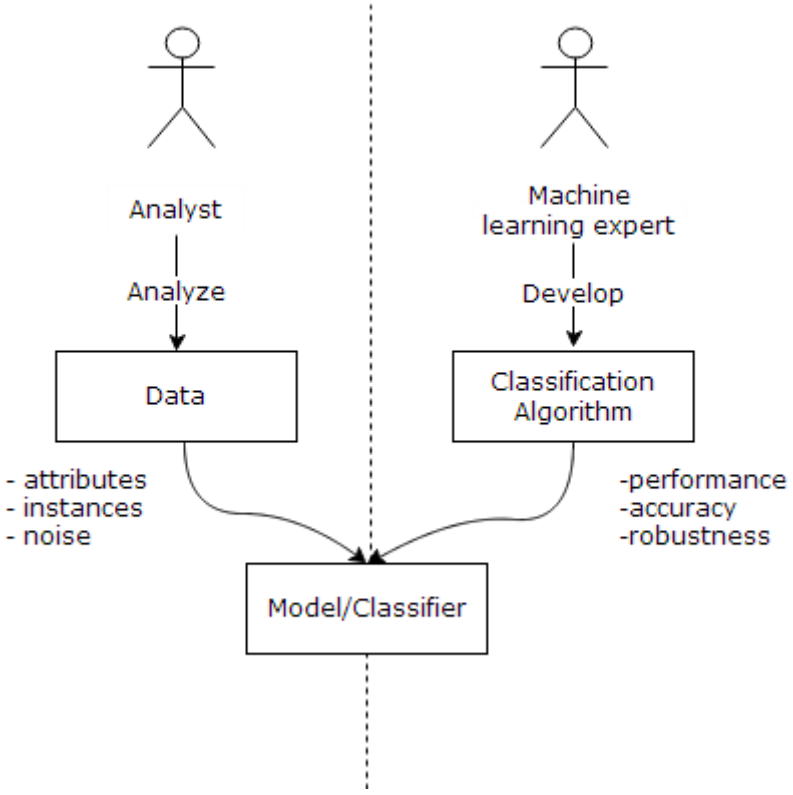


FIGURE 3.1: A classification process with two main stockholders: an analyst and a machine learning expert. An analyst is focused on data and tries to study its attributes, instances and noise. A machine learning expert is focused on a classification algorithm, its performance, accuracy and robustness. A result of applying a classification algorithm to a data set is a model. By studying the built model both stakeholders' aims can be achieved.

Figure 3.1 shows a classification processes with two named actors: an analyst and a machine learning expert. As the figure shows, the result of applying a classification algorithm to a data set is a classification model. By studying the built model both stakeholders (analysts and machine learning experts) can gain insights. First of all, a

classification algorithm is trying to build an accurate model. Accuracy of a model is determined by the way it performs. A classification model makes decisions that separate data into subsets. Of all possible decisions several of them have value. Those valuable decisions separate data in an accurate way, giving a model more prediction power. By exploring those decisions of a model, the structure of the data can be discovered. The most important threshold and splitting point can be identified. Depending on a classification algorithm and correspondingly built classification model, more features of data can be studied with the use of the model.

Thus, studying a model is beneficial for both actors. In some way, a model is a simplified version of a data set. Although classification models are usually complex structures which can be unfamiliar for analysts, as they usually lack knowledge in classification algorithm construction. A solution can be usage of less sophisticated algorithms. The decision trees are one of the easiest classification methods for building a classification, but they lack accuracy in case of very complicated data sets. As Chapter 2 has shown, Random Forest model is a good balance between complexity and performance. In addition, Random Forest provides other ways to get information about data, including proximity, prototypes and interactions. The Random Forest model and its features can be studied by an expert in the raw format. But a better way to study information is by the means of visualization. Moreover, an interactive visualization can reveal complex connection patterns even more easily.

3.2 Goal

This work takes the Random Forest classification model as a basis to study data. Random Forest algorithm is applied in various domains with different data set sizes. The goal is to find a way to visualize a Random Forest model such that insights about data structure and attributes' relations are gained. The aim is to find components and features of Random Forest and their suitable visual representations that are beneficial for the visualization to obtain those insights.

Decision forest is used in different domains. This work is focused on the model itself and its features, rather than specific usage in a certain field. The aim is to apply the visualization to a broader field.

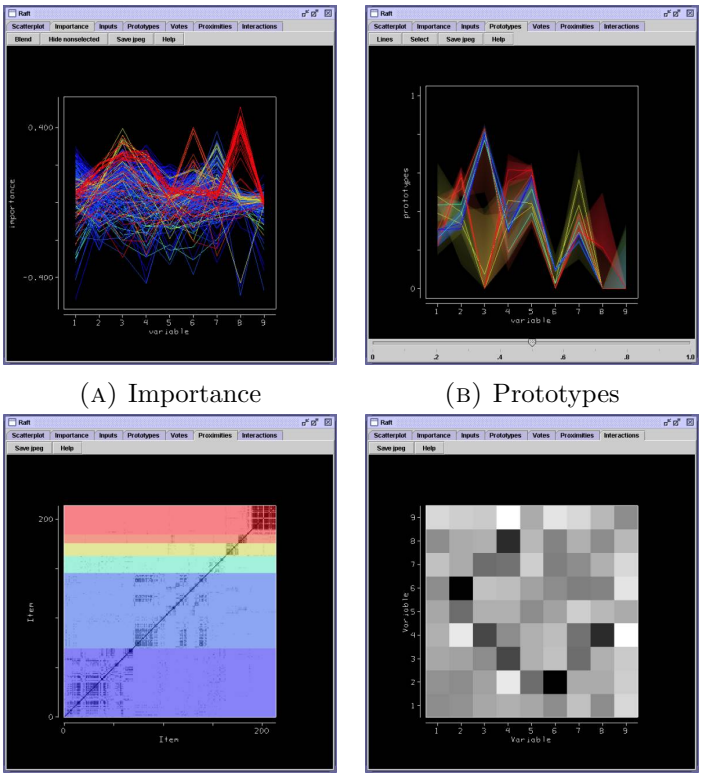
Random Forest algorithm is capable of handling data sets with a large amount of instances and attributes. The visualization should be capable of presenting a forest with large number of instances, attributes and classes.

3.3 Existing analysis tools

There are several solutions to the named problem. During the research two existing analysis tool for Random Forest were found. Both tools present visualizations of a Random Forest model, but have different approaches to it.

3.3.1 Random Forest Tool

Together with the concept of Random Forest, Breiman created a tool to visualize a Random Forest model. The tool is called RAFT, Random Forest Tool. The tool accepts data with numerical features only and presents visualizations for different features of Random Forest:



(A) Importance (B) Prototypes

(C) Proximity (D) Interactions

FIGURE 3.2: Random Forest Tool visualization examples. Colors correspond to classes of an input data set. (A) The plot presents the variable importance visualization. Each line represents the importance of a variable for every instance in a data set. (B) The plot for the prototypes visualization for every attribute of a data set. (C) Scatter plot for the proximity matrix for every pair of instances. The darker color present stronger proximity. (D) The plot shows interactions between variables. The intersection presents the interaction measure for every pair of attributes.

- Scatter plot. RAFT presents data overview with scatter plot, allowing the user to select the attributes for axes. A user can interact with data by selecting subsets of data.

- Importance. Importance visualization presents the variable importance measure case wise. For each case and variable the importance measure is computed and visualized on parallel axes graph. The graph shows the importance for each class, but can result in clutter in case of large number of cases. The example is presented in Figure 3.2 (A).

- Input. The input panel shows data set instances on a parallel axis plot, where axes are the features of a data set. The panel gives an overview of the data set, but also result in clutter with many instances.

- Prototypes. A prototype shows an abstract case that represent the class. The prototypes for each class are visualized as linear traces on parallel axes graph. Figure 3.2 (B) presents an example of prototypes visualization.

- Proximities. Proximities for data instances are visualized as a color coded matrix in Figure 3.2 (C) according to the instance class. The darker color indicates a higher proximity level. It is hard to compare the proximity of different classes with different color coding as the intensity is different for every color.

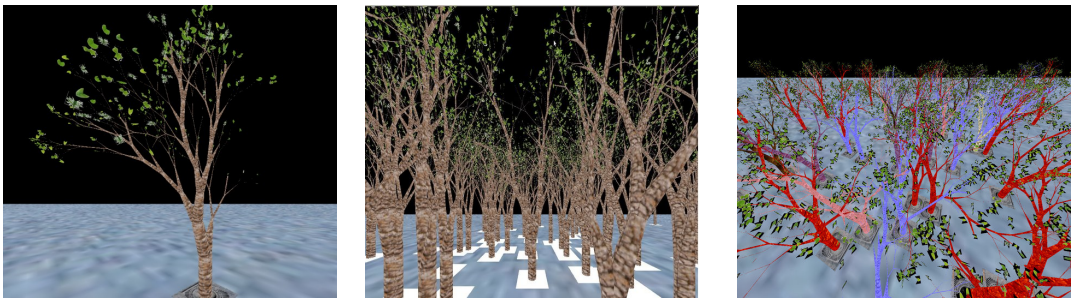
- Interactions. In RAFT interaction visualization is a matrix with a cell corresponding to an intersection of two variables. 3.2 (D) presents an example of the interaction visualization.

RAFT gives visualizations for all important features of Random Forest. The tool uses parallel coordinates to visualize the metrics, but it does not give any representation of trees of a forest or the relation between those metrics and the trees. The used techniques are powerful if the data set size and the number of attributes is relatively small. In addition, the representations do not support nominal features.

3.3.2 3D technique

Another technique is proposed by Min Yang et al. [14]. The technique presents a literal implementation of a tree concept. A single decision tree is presented as a realistic tree. The example is presented in Figure 3.3 (A). The leaf nodes are coded by different types of real leafs. Thus, a decision forest is represented by collection of trees in Figure 3.3 (B). The voting process for a new instance is visualized with the use of color coding. Each tree is colored according to the class color it votes for. The example is in Figure 3.3 (C).

The technique presents an artistic approach to Random Forest visualization. The generated visualization looks interesting, but is hard to interpret. The differences between class leafs are not easily noticeable. The implementation shows trees as separate classifiers and visualizes the classification process for a single instance.



(A) Single tree (B) Forest (C) Votes

FIGURE 3.3: (A) presents a single tree visualization. Leaf style depends on a class label. A decision forest is a collection of such trees (B). Voting process on (C) is color coded according to a selected class.

3.4 Data and Model aspects

The main goal of an analyst is to study data. There are three main aspects that experts are interested in. The aspects include instances of data, its attributes and data noise. With the Random Forest as the model, those aspects can be studied with the Random Forest model features, described in Section 2.4.

Existing implementations try to solve the problem. Two existing implementations were found during the research. The first one by Breiman gives a good visualization support for Random Forest features, but completely ignores the Random Forest components - trees. The 3D technique on other hand focuses on trees' visualization and does not take the metrics into account. The connection between them can help to understand the origin of those features. Thus, another goal of this work is to integrate features of Random Forest with its metrics. The main question is still which features can be effectively visualized and how.

Figure 3.4 shows an overview of important data features connected to Random Forest features. With Random Forest, instances of a data set can be inspected with the use of a proximity matrix and prototypes. The matrix shows how instances are connected to each other class-wise. The disconnected instances indicate misclassification or outliers. The data distribution can be studied with the prototypes. Prototypes show the abstract instance for each class, that is the representative of this class. Attributes of a data set are inspected with the model itself. Split attributes and split points, as parts of a Random Forest decision making process, give information about data organization. They can be studied separately or together using the interactions. Noise in the data set can be identified by high error rate and a proximity matrix. The next chapter presents the proposed solution for the visualization of those features.

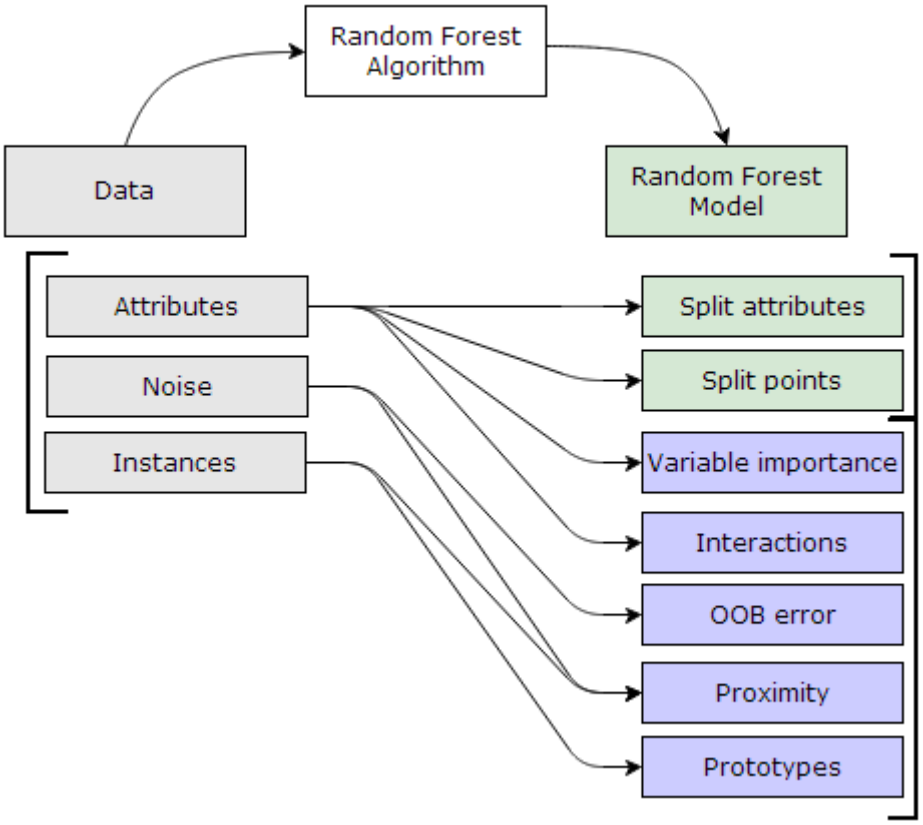


FIGURE 3.4: The connection between data features and Random Forest metrics. Attributes are studied with the model itself by the split attributes and split points and with the model features: variable importance and interactions. Instances are examined though a proximity matrix and prototypes. Noise is identified with OOB error and a proximity matrix.

Chapter 4

Visualization design

This chapter presents the reasoning behind the visualization design and explains how those decisions were made.

4.1 Random Forest visualization

One of the goals of the tool is to integrate the metrics of Random Forest with decision trees. Thus, the visualization of individual trees is required. There are many ways of presenting a decision tree, including node-link diagrams, indentation diagrams, icicle plots, tree maps and tree rings. Each method has its advantages and drawbacks. There are some experiments conducted to compare the performance of different methods. Barlow and Nevile handled an experiment to compare the methods to find the most suitable ones for the identification of overall tree structure[15]. The main criteria were: identification of the relation between nodes and their sizes and user preferences. According to the experiment, node link diagrams and icicle plots are preferred for these criteria.

Every tree in Random Forest is treated equally. To present the forest, the idea of *small multiples* by Tufte is used [16]. According to the idea, the whole space is divided into grid subspaces and those subspaces are used for the individual visualizations, which allows easy comparison. Thus, the space for the Random Forest visualization is divided into areas for individual trees. The size of the area depends on the number of trees. Random Forest contains a large number of trees, typically over one hundred. To visualize a large number of trees space efficiently, the following aspects are of importance:

- *Structure of an individual tree.* The structure of a tree gives insight into the complexity of a decision, which reflects the complexity of an input data. Both node link diagrams and icicle plots give a good overview of a decision tree structure.

- *Space filling.* According to the idea of small multiples, the space used by a single tree is restricted. And with larger number of trees in a forest the space is getting smaller. Thus, the space efficiency is of the most importance. Each node in an icicle plot is presented as a rectangle that fits into the idea of a grid. In addition,

unlike a node-link diagram, an icicle plot wastes no space on links; it shows the hierarchy by position. Thus, an icicle plot uses space more efficiently, comparing to a node-link diagram.

- *Scalability.* The number of trees can be large, but can differ greatly based on the data size. Small data sets enables good classification performance with a relatively small number of trees. Other data sets with over thousands of attributes require a comparable size of Random Forest. Thus, a representation of an individual tree must be easily scaled on the number of trees. An icicle plot uses less elements for the representation, as it has no links. The size of an element of an icicle plot can be greater than on a node link diagram. Thus, an icicle plot allows to scale more efficiently than a node-link diagram.

To sum up, it is beneficial to represent each tree of Random forest as an icicle plot as it uses space more efficiently and enables scalability. An icicle plot can be extended to represent the distribution of input data within decision tree nodes. It can be done with different methods, such as width or coloring schemes. But, every tree in Random Forest is learned using a subset from the original dataset; two trees with same structure can look different. One way to keep consistency in the structure and allow users compare trees, is to make them look the same. One solution is to keep the sizes uniform. Thus, fixing a size of a leaf node makes icicle plots more structured and helps to avoid clutter, but gives no information about instances distribution in the nodes. Figure 4.1 presents an example visualization for an individual decision tree. The visualization is constructed from bottom to top, the width of the parent node depends on the size of its child nodes.

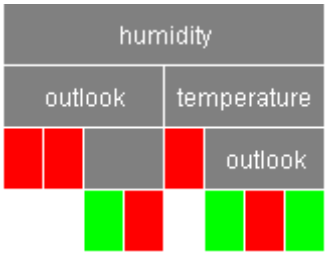
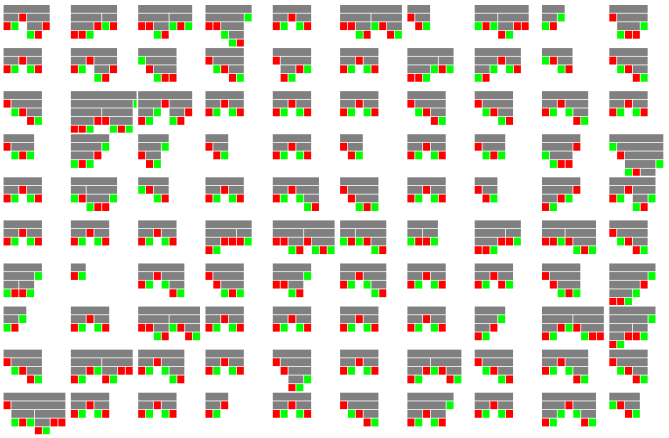


FIGURE 4.1: Icicle plot with fixed leaf size. The leaf nodes are colored according to the class. Internal nodes are marked with the split attribute.

Figure 4.2 compares the visualizations for Random Forest with a different number of trees. The sizes of trees are scaled according to that number. The height and width of a leaf node is scaled to fit the largest tree of a forest. That results in unused space between trees, but it is a necessary sacrifice. As it is clear from Figure 4.2 (A) trees have different sizes. Optimizing every tree for a subspace would lead to confusion in structure and an inability to compare trees. With this visualization a user can compare trees structurally, but unfortunately the split attributes are not visible. Thus a possibility to inspect trees individually is required to be able to fully distinguish trees.



(A) Random Forest with 100 trees

(B) Random Forest with 625 trees

FIGURE 4.2: Examples of the Random Forest visualization with a different number of trees presented with the idea of small multiples. The size of tree leafs are unified to avoid clutter and let user to compare the trees structurally.

4.2 Proximity visualization

One of the most important metrics that Random Forests offer is the proximity measure, which is computed based on decision trees. Proximity shows how strongly instances of a data set are connected to each other. The connection is measured with a percentage of trees that classify a pair of instances equally. The result is a matrix of size $n \times n$, where n is the number of instances. The straightforward idea to visualize a 2-dimensional matrix is to map its values on a XY plot, where indexes are put to the corresponding axes. The method is validated in the case study in the Chapter 6 and, thus, does not require more sophisticated visualization metaphors to be used.

A proximity matrix consist of values from 0 to 1. An expert has two tasks. First, he is interested in distinguishing endpoints (0 and 1, absolute connection and no connection). Second, he is concerned about a middle point (0 and 1) and their closest endpoints (the distance to the end points). One way to visually perceive data on such

a scale is color mapping. There are different color maps used for visualization, including rainbow, heatmap and grayscale. The rainbow color map is most commonly used and unnecessarily complex for this problem. Grayscale map on the other hand has no clearly defined middle, so it is not possible to distinguish affinity towards either endpoint. Consequently, heat map with four main colors in its color spectrum has a clear middle and pronounced extremes. For a proximity matrix visualization with heat map, black indicates no connection and white shows complete similarity. Such coloring can be considered as intuitive for a user.

The proximity is compared within classes of an input data set. Thus, class distribution representation is required. Breiman's solution, described in section 3.3.1, uses class color within the matrix, but it makes it hard to compare the values of different classes. Another solution to the problem will be to introduce the class distribution outside the matrix. Axes of two dimensional plot are colored by class, allowing users to compare every pair of classes. Figure 4.3 shows the resulting solution for proximity matrix visualization. It is clear that the red class is completely separated from green and blue classes.

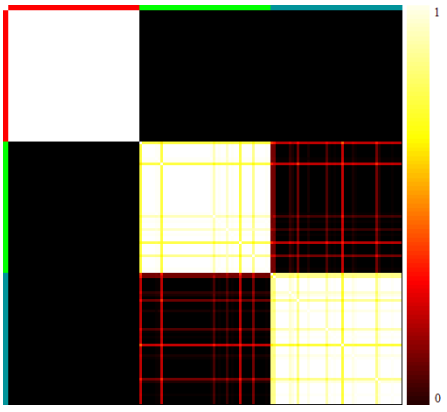
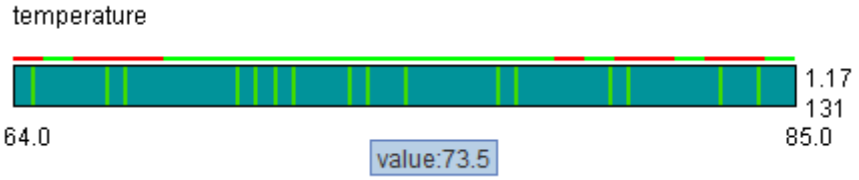


FIGURE 4.3: Proximity matrix solution visualization with class distribution. The values of the matrix are colored according to a heat map. Black values correspond to no connection, white values reflect 100 % connection. The red class is completely separated from the other two. Green and blue classes have several connection between each other.

4.3 Attribute visualization

Another property that is revealed by Random Forest are interactions. The interactions show relations between attributes. Thus, attribute visualization is required. There are two types of attributes: nominal and numerical. An expert is interested in how attributes relate to classification, the variable importance and raw occurrence score. For a numerical attribute, the range of an attribute, its split points and the division of data by those split points is the available data. For numerical and nominal attributes, the class distribution over its values can be studied.

(A) Nominal attribute



(B) Numeric attribute

FIGURE 4.4: Attribute views for numerical and nominal attributes. The width of the representation shows the size of the input data set. The class distribution is added above to see the correlations. (A) For nominal features the values are color coded on a categorical palette. The size reflects the number of instances with the values. (B) For numerical attributes split points are added on a linear scale, showing the class distribution partition.

Figure 4.4 presents the visualization solution for the attribute view. All the measures and the name of an attribute are visualized as plain text, as they are single data points.

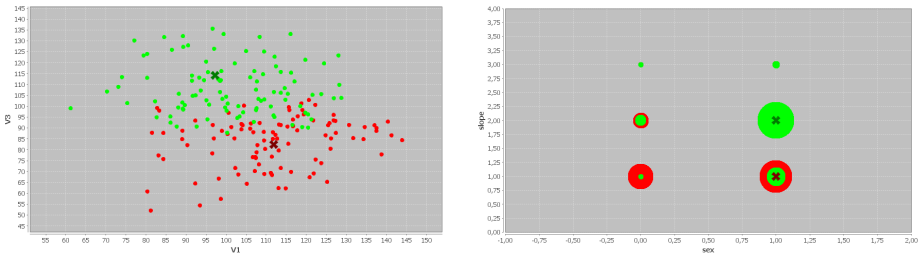
Figure 4.4 (A) is the representation of nominal features. The total width of the attribute represents the number of instances in a data set, and the width of each attribute category reflects the number of its instances. To be able to distinguish values of a nominal attribute, they are color coded with a categorical palette. To show the relation to classification, a class distribution is added above it. The instances are sorted for two attributes. First, according to the current attribute and then according to the class. That allows a user to see the class distribution proportion within each value.

The numerical feature visualization on Figure 4.4 (B) additionally has the range of the attribute placed on both sides. Split points are shown as marks on the attributes, where every value is scaled to its proper position. The class distribution is sorted by the numerical attribute and also added above the attribute visualization.

The value of a split point for a numerical attribute and the specific value for a nominal attribute are dynamically presented as a user hovers over it. The method is used to avoid clutter in case of very densely located split points or small sized nominal values.

4.4 Prototypes visualization

Prototypes are abstract instances that represent each class. Prototypes are hard to interpret as raw values, it is useful to see them in the scope of data. Prototypes together with a proximity matrix give an idea of an attribute connection to the classification.



(A) (B)

FIGURE 4.5: Scatter plots for a pair of attributes. Instances are color coded according to the class. Prototypes for each class are presented as crosses. The size of a point depends on number of instances for the intersection. The nested points correspond to the same values presented within different classes. (A) An example of a scatter plot for two numerical attributes. (B) An example of a scatter plot between two nominal features.

The distance between prototypes let a user estimate how easily classes of a data set are separated.

A direct way to visualize the prototypes is by means of scatter plots among data. The origin of the prototypes are visible on the scatter plot as surrounding data of that same class. To add more value to the nominal attributes visualization, the distribution of data is reflected in the size of a point on the scatter plot. It can not be clearly seen for a numeric attribute on Figure 4.5 (A). But for nominal attributes in Figure 4.5 (B) it is more clear. The methods leads to restriction of only two attributes being considered at a time.

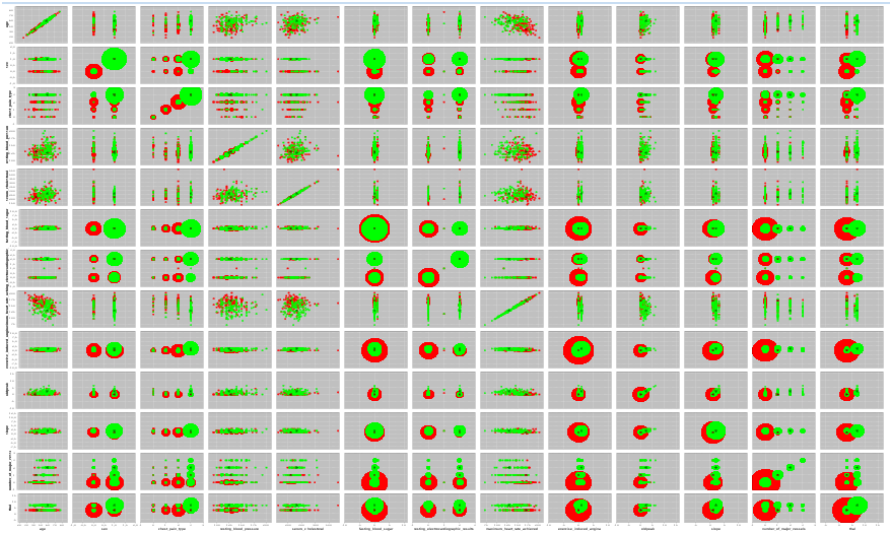


FIGURE 4.6: Scatter plot matrix for all pairs of attributes of a data set. Points are coded according to the class. The size depends on the number of instances with particular values.

To allow a user to consider more than two attributes at a time, a scatter plot matrix is used. Scatter plots give the overview of the input data, allowing a user to see general trends for numerical attributes. The size extension allows a user to see the class distribution over nominal attributes. Figure 4.6 presents an example of the scatter plot matrix. The representation has a drawback: if the number of attributes is large, it is hard or even impossible (in case of 500 attributes results in 250 thousands plots) to perceive the data. In this case, the visualization is limited to one scatter plot at a time.

4.5 Summary

This chapter presented the most important decisions made for the Random Forest components representation. But those visualizations are not used in isolation; they are integrated together. The integration is aimed at giving more insights into data structure and Random Forest decision making processes. The possible interaction between the components is described in the next chapter.

Chapter 5

ReFINE

This chapter presents the solution of the Random Forest visualization developed after studying the main aspects of a Random Forest in Chapter 2 and including the ideas presented in Chapter 4. The developed tool uses the implementation of Random Forest from the Weka package [17]. The standard Weka Random Forest implementation was extended with additional functions, including implementation of variable importance, proximity and prototypes. The tool is named ReFINE which stands for Random Forest INspEctor. ReFINE contains several panels, one for each aspect of the forest. Figure 5.1 presents an overview of two main panels.

ReFINE is aimed at integrating the features of Random Forest with its decision trees. Thus, the main view is the Forest View. That shows the representation of all trees. The tool contains Proximity, Interactions and Prototypes panels. Each panel is bound to

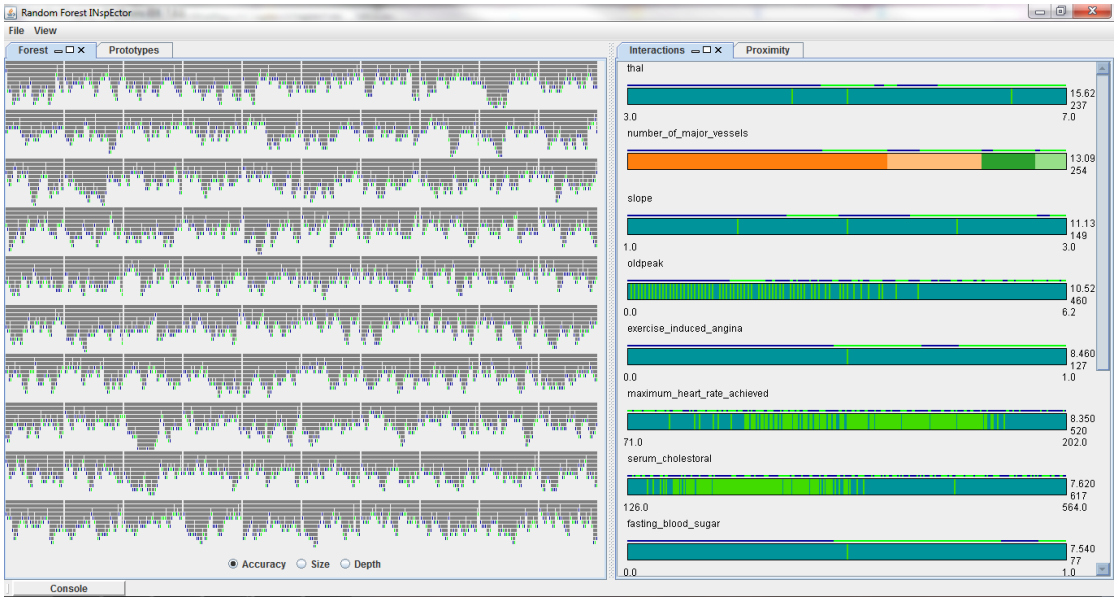


FIGURE 5.1: Overview of the two main panels. Random Forest on the left, as a collection of decision trees. Each tree is represented with an icicle plot. The attributes view on the right shows attributes of a data set with (from top to bottom) the name, variable importance, raw occurrence and range for each attribute.

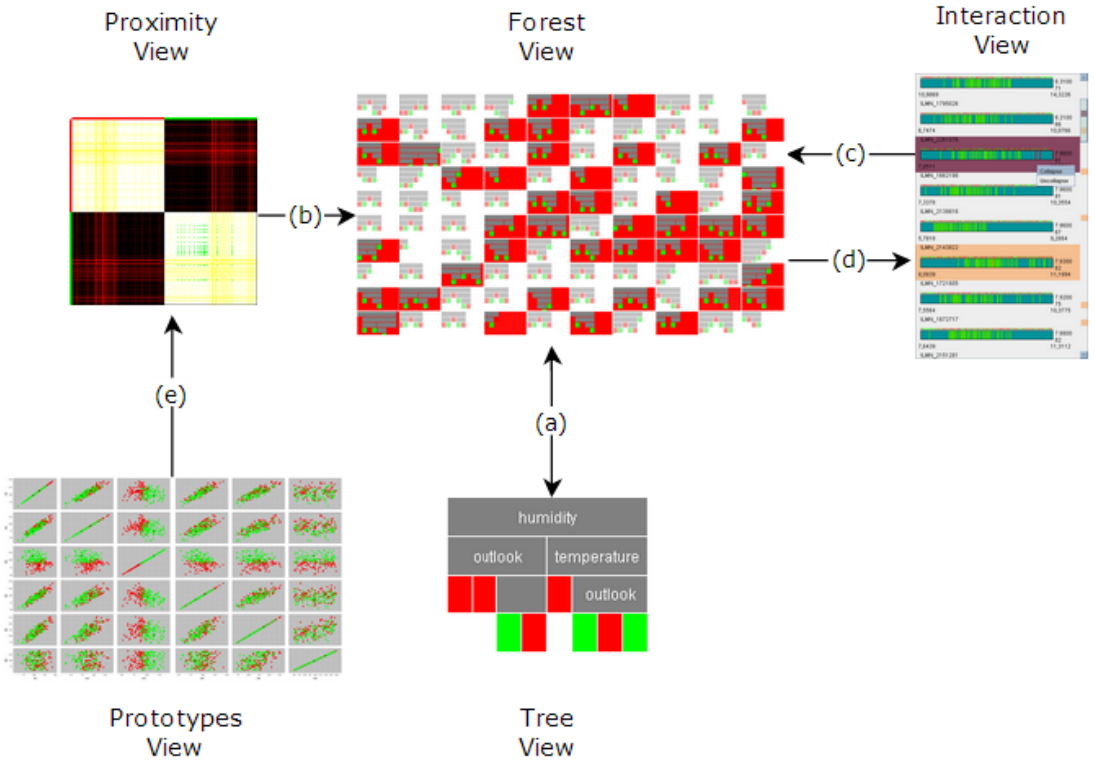


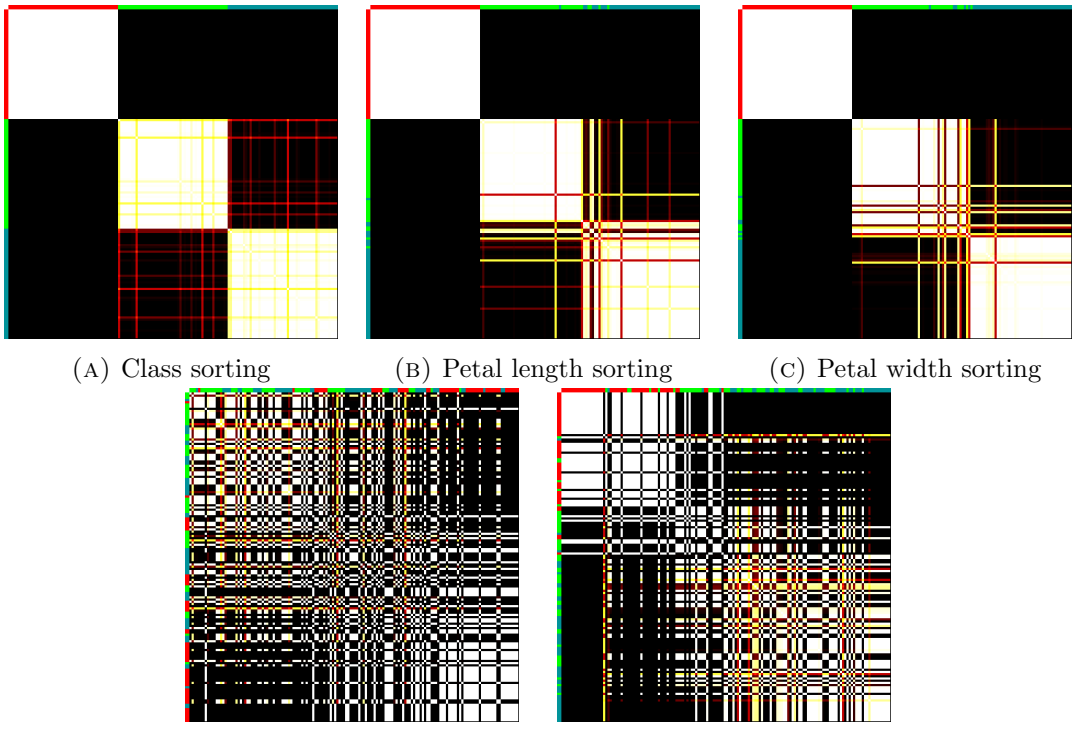
FIGURE 5.2: Diagram shows the relations between the main panels of the ReFINE. (a) A user can inspect each tree separately. (b) The selection in a proximity matrix shows the trees that are involved. The involved trees are highlighted with red. (c) A selection of an attribute highlights the trees that use the attribute as a split with red. The other splits points of the trees are reflected back into the attribute domain (d). With prototype scatter plots a user can see (e) the instances used for prototype calculation in the proximity matrix. The corresponding instances are highlighted within the proximity matrix with a class color.

the other panels, such that a selection in the panel is reflected on the others. Figure 5.2 shows the dependencies of the panels.

The Forest View in Figure 5.1 on the left gives a general overview of trees' structure. Each tree is visualized with an icicle plot, scaled according to the number of trees and their depths. A user can explore a single tree by selecting it on the panel, as presented in Figure 5.2 (a). In addition, a user can rearrange the tree by certain criteria:

- *Accuracy.* Each tree has its own OOB error rate.
- *Size.* In this case the size of a tree is determined by the total number of nodes, both internal and leaf nodes.
- *Depth.* Depth of a node is the distance from it to the root node. Depth of a tree is the maximal distance of all the nodes.

The idea behind different arrangements of trees is to see if different properties of trees (accuracy, size and depth) has correlation to the forest features (proximity, interactions). Thus a user needs to see which trees are involved in a corresponding feature. One way



(A) Class sorting (B) Petal length sorting (C) Petal width sorting

(D) Sepal width sorting (E) Sepal length sorting

FIGURE 5.3: The solution for the proximity matrix visualization for Iris data set. The matrix is colored according to the heat color map. The black color indicates no connection of instances. White color shows that all trees classify the instances equally.

to enable it, is querying trees based on a certain criterion. That requires the interaction between different panels of the tool.

5.1 Proximity View

One of the panels represents the proximity matrix. The proximity matrix is visualized as an XY plot with color coded values. The values of the matrix can be sorted by an attribute, presenting a different perspective on the proximity. A user has possibility to choose an attribute for sorting. The results of different sortings are compared in Figure 5.3 (A-E). An example of the proximity matrix is computed for the Iris data set with a Random Forest of 100 trees. The default class sorting shows the connections within classes. Other arrangements help to estimate the connection between an attribute and the class label. Thus, petal length and width are strongly connected to the class label (Figure 5.3 B and C) and there is no strong connection for the sepal size (Figure 5.3 D and E).

The main goal of the tool is to show the connection between Random Forest metrics, such as proximity, and its components: decision trees. With ReFINE, a user can interact with a proximity matrix by selecting an element in it. As a result Figure 5.2 (b), the trees that agree on the classification of those two instances will be highlighted in the

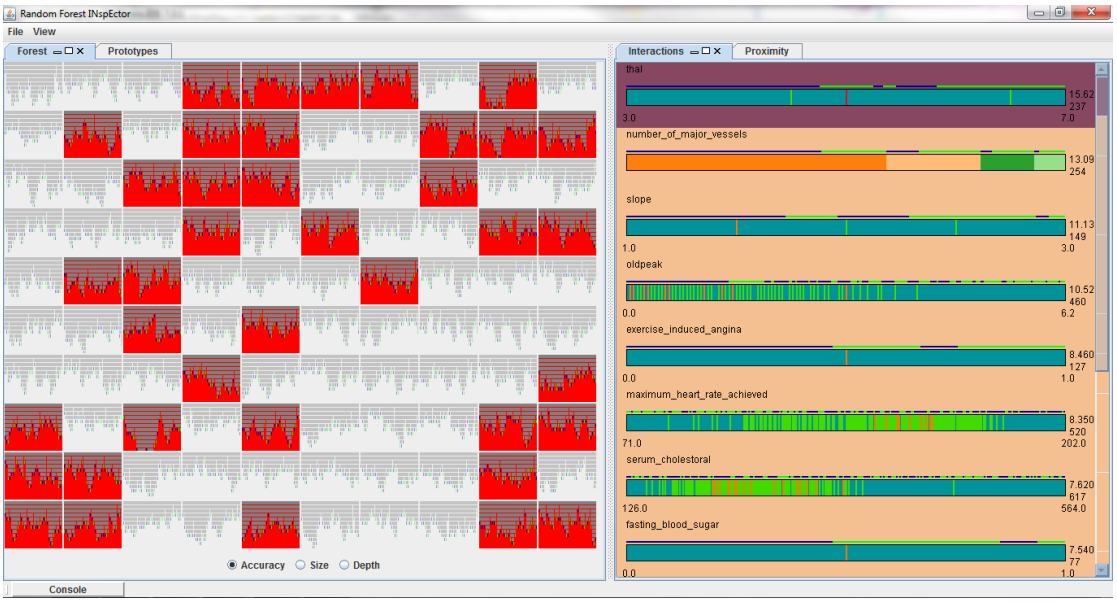


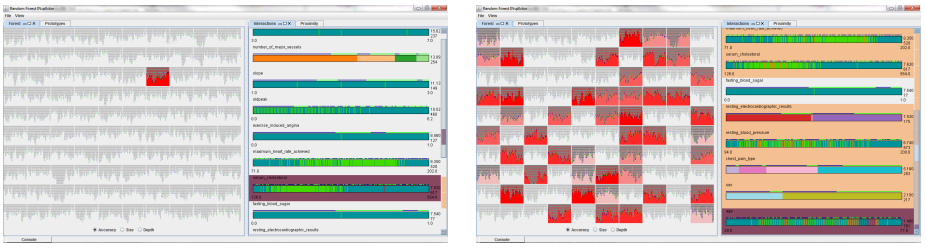
FIGURE 5.4: A result of selecting a split point. Trees that use the split point are highlighted in red. Other attributes that are used in those trees are highlighted in orange. Purple color indicates high correlation (> 0.8)

Forest View. The non-highlighted trees can be inspected Figure 5.2 (a) to understand the cause of the misclassification.

5.2 Interactions View

The Interactions View contains an attribute panel for every attribute of a data set. Every panel is built as described in Section 4.3. The attributes are sorted according to the variable importance measure, such that the most important ones are on top. A user can explore the interactions between attributes by interacting with them with the cursor. A user can select a nominal attribute or a certain split point of a numerical attribute, as presented in Figure 5.2 (c). Trees of the built Random Forest are highlighted in the corresponding panel. Then the split point of the selected trees are reflected back to the attributes domain, highlighting the corresponding attributes and split points, Figure 5.2 (d). The attributes and the split points are highlighted only if they occur down the path from the selected criterion. For example, if the gender attribute is selected in the tree in Figure 2.1 then age attribute is highlighted with the split point 50. The highlighted attributes are color coded. The orange background color indicates that an attribute is used with the same trees. The purple color shows that the attribute is used in the trees and the correlation between this attribute and selected attribute is greater than 0.8, but a user can change this threshold.

With the use of the Interaction View a user can investigate the Random Forest decision making process. A user can see which attributes are of the most importance for the classification, based on variable importance measures and number of raw occurrences.



(A) (B)

FIGURE 5.5: Results of selecting different split points. (A) Selecting an outlier. (B) Highlight with different color intensity, depending on the selected split value. The more intense colored trees contain split points closer to the value.

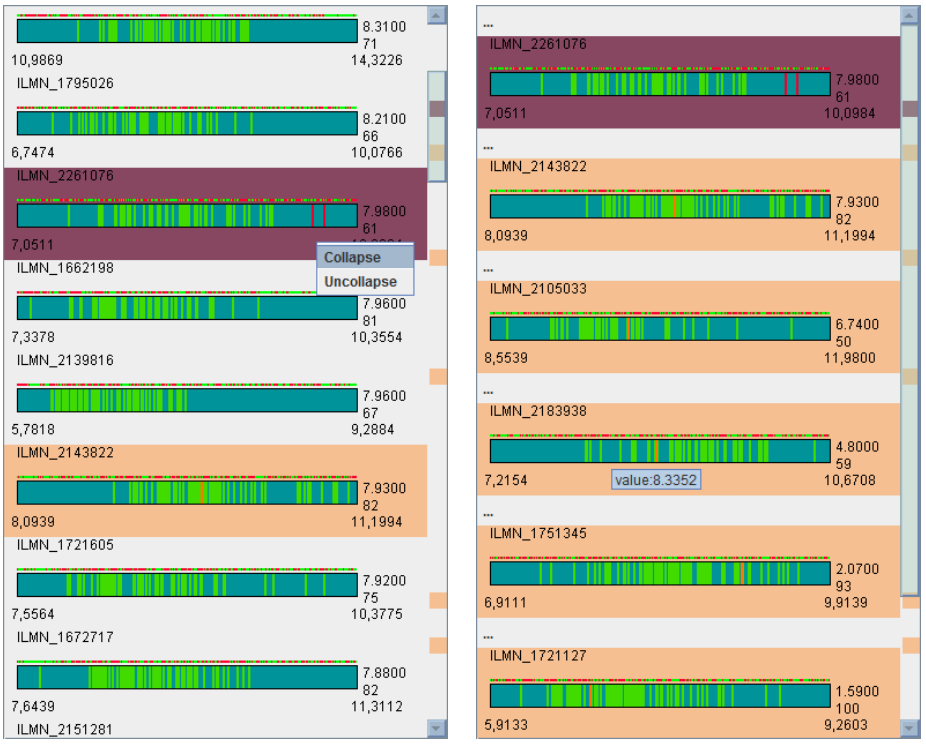
The view also presents all the split points. By interacting with the Attribute Views valuable split points in data can be found. Some split points are good candidates for subdividing data and used in many trees, others are outliers and are used by a single tree. In Figure 5.5 (A) an example of such an outlier is presented. For numerical attributes the scale is continuous. All trees are built on a subset of the data, thus the probability for them to use the exact same split point is very low. Moreover, some numerical attributes' split points are very unevenly densely populated (the example is in Figure 5.4 *maximum_heart_rate_archived* attribute). Thus, a user can select a split range for an attribute. The trees are highlighted according to the distance from the middle point of the range. The intensity of the background corresponds to this distance, where the middle point is complete red. Figure 5.5 (B) shows an example of differential forest highlights.

A user can select more than one attribute at a time. It is possible to narrow the selection down by selecting more attributes. In case of two or more attributes, only the trees that use all the selected attributes in a single path will be highlighted. A user simply selects additional attributes with the cursor. The selection creates more complex queries, allowing a user to investigate the models' decision making process.

To cope with data sets with a large number of attributes, some additional functions are added. Attributes are contained within a scroll panel. The standard scrollbar is extended to reflect the selection of attributes. Figure 5.6 (A) illustrates an example of the scrollbar. The coloring helps a user estimate the number of attributes selected and allows to quickly navigate to them. To improve the navigation even further, a user can collapse unused attributes. As a result, in Figure 5.6 (B) only highlighted attributes remain visible.

5.3 Prototypes View

Prototypes are visualized as scatter plots for all pairs of attributes. The prototypes for each class are marked within the scope of data on the scatter plots. Prototypes are calculated using proximity. The prototype view is connected to the trees through the



(A) (B)

FIGURE 5.6: The user interface for the Interaction View. The scroll bar representation is extended to reflect the selection in the panel, allowing a user to navigate through it easier. In addition, a user can hide all nonhighlighted attributes

proximity matrix. A user can select a prototype on the scatter plot and the corresponding instances are highlighted in the proximity matrix with a class color. Those instances are the sources to calculate the proximity for each class. After, a user can inspect the highlighted instances with the proximity matrix. Figure 5.2 (e) and (b) shows the relation. A user can select any of the highlighted instances to inspect the relation to the decisions trees. Figure 5.7 illustrates the result of such an interaction. The instances for the green class prototypes are selected in the proximity matrix, showing the origin of the prototype.

The Prototype View presents the overview of a data set, showing the data distribution over all its attributes. Prototypes themselfs help to establish the connection between attributes and the classification. If the distance between prototypes is significant, then classes are well separated with the attribute. Just like a proximity matrix, prototypes help to see how the attributes are connected to the classification. Moreover, with the scatter plot matrix the most valuable split points and outliers can be noticed and inspected with the Interaction View.

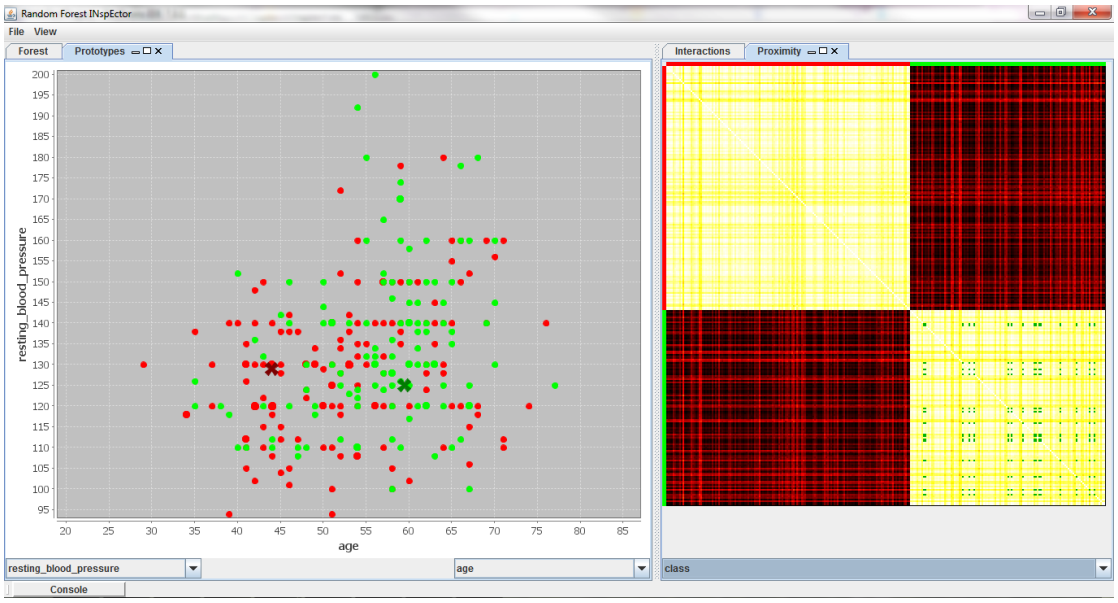


FIGURE 5.7: The Prototype View on the left with a scatter plot for two attributes. The Proximity View on the right. A user can select a prototype in its scatter plot to see the instances used to calculate it in the proximity matrix. The instances are highlighted with the class color.

5.4 Summary

The resulting tool allows a user to interact with the Random Forest features, showing their origin. The most important link is the connection of those features to the built forest. That helps a user to understand how they are obtained from the forest. The Interaction View allows to see the decision making process of the forest. The Proximity and Prototypes Views allow to see the connection between different variables and the classification.

The next chapter is dedicated to the validation of ReFINE with several case studies, including generated and real data.

Chapter 6

Case studies

The possibilities of ReFINE are demonstrated with several case studies of various data sets. First, a generated data set is used to prove that it is possible to discover some generated particularities of a data set. Then, the tool is applied to several real data sets.

6.1 Generated data

First the tool is applied to a set of generated data to test the abilities of ReFINE to analyze data. The data set contains 6 numeric attributes, a1 through a6, and 3 classes: A, B and C. Each class has 50 instances. The attribute values were generated with the normal distribution under the following assumptions:

- a1 and a2 have good prediction power to separate class C from classes A and B.
- a3 and a4 have no strong correlation between any of the class labels.
- a5 and a6 can separate classes A and B well.
- None of the attributes are strongly correlated.
- Class C is disconnected from classes A and B, as one of attributes is generated in a separate part of the space. Table 6.1 shows the parameters for the normal distribution that were used to generate the data.

First, a Random Forest with 100 trees is generated for the data set and processed by ReFINE. Each tree is constructed while considering three random features. The OOB error is 0.3716 . Generating more than one hundred trees results in no change in the

Set	a1		a2		a3		a4		a5		a6	
	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2
class A	30	10	50	15	10	5	30	5	80	20	90	20
class B	30	10	50	15	10	5	30	5	100	20	100	30
class C	60	10	60	15	10	5	30	5	80	5	110	50

TABLE 6.1: Parameters for the normal distribution used for the data set generation.

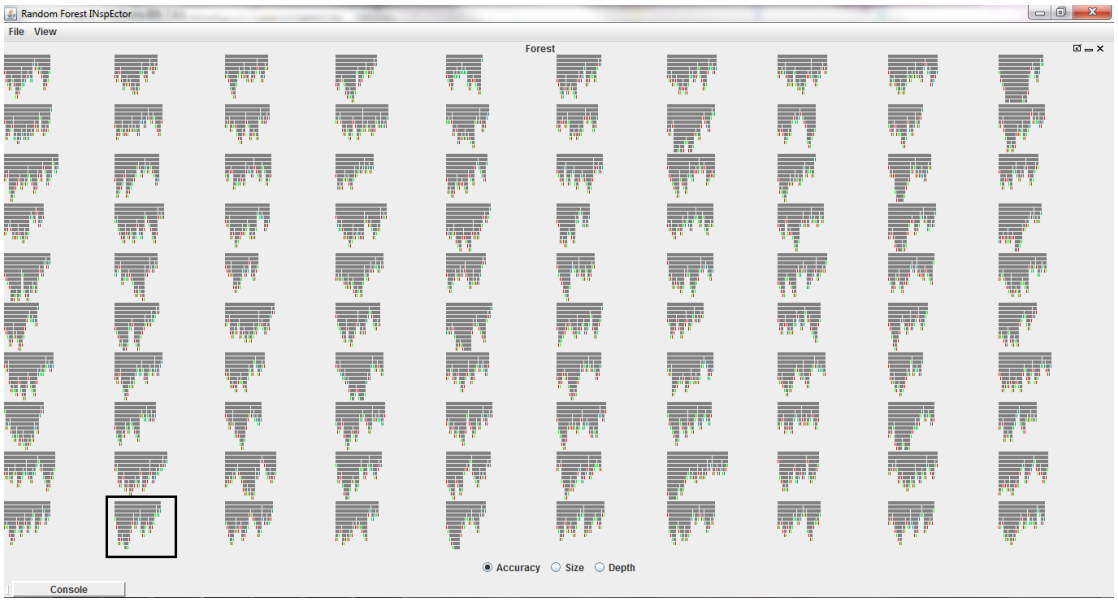


FIGURE 6.1: A Random Forest of 100 trees for the generated data set with 3 classes and 6 variables. The general structure of the trees shows the complexity of the data set. The selected tree is presented on the next figure.

OOB error. The resulting trees' visualization is presented in Figure 6.1. Already from the trees' structure it can be concluded that class C is easier to separate, as the trees are asymmetric. Figure 6.2 shows an icicle plot for a single tree. Class C, represented as blue, is separated at a lower level. Classes A and B on the other hand are harder to distinguish and require more levels to be separated. That is also supported by the proximity matrix. Figure 6.3 (A) presents the proximity matrix for the data set with sorted by class. Class C is totally disconnected from the other classes, as the majority of values for the blue class are colored as white, while classes A and B have connections between each other, as the values are not white. To see the relations to other classes, a user needs to consider the corresponding intersections. Thus, class A has connections to class B and vice versa.

The attribute a1 is considered to be the most important with the variable importance of 6.36. The variable a5 is also treated as a good predictor, while the attributes a3 and a4 have negative variable importance. Negative variable, which is possible for an attribute, importance shows that an attribute can have detrimental impact on the classification.

While it is not clear with the variable importance measure which class a variable has the most impact on, it can be explored with the proximity matrix. Figure 6.4 shows different sortings for the matrix. It is visible that the attributes a1 and a5 have a good prediction power for the blue class C, thus the variable importance is high for those attributes. The attributes a3 and a4 have no clear patterns and sorting with them results in clutter. The conclusion is also supported by the Prototypes View. The prototype on the scatter plot in Figure 6.5 (A) for the blue class is separated from the other two prototypes.

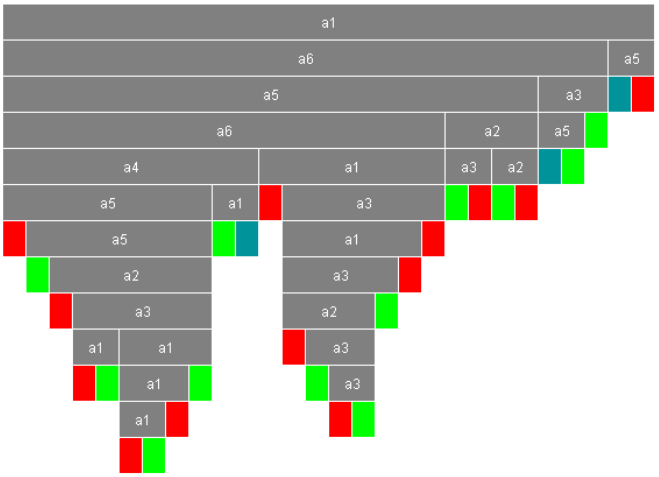
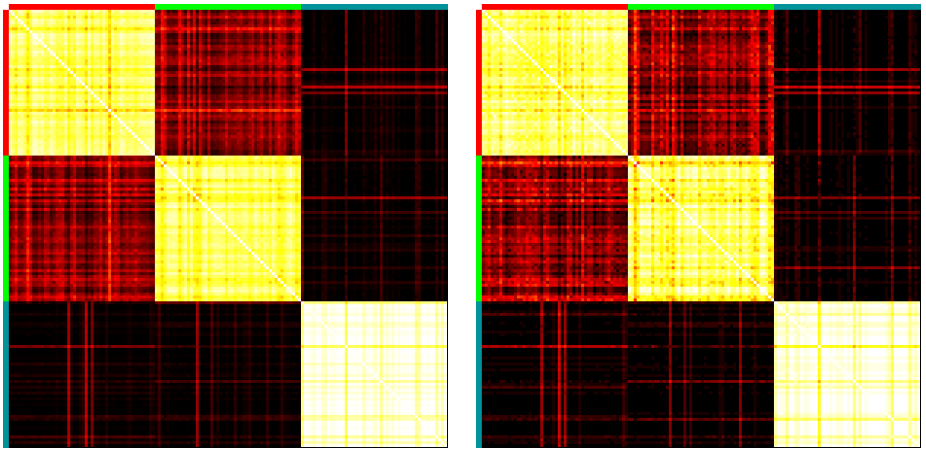


FIGURE 6.2: One of the 100 trees of the Random Forest built for the generated data set example. The blue class is easier to separate from the red and green classes. The red and green classes require deep branches to be separated.



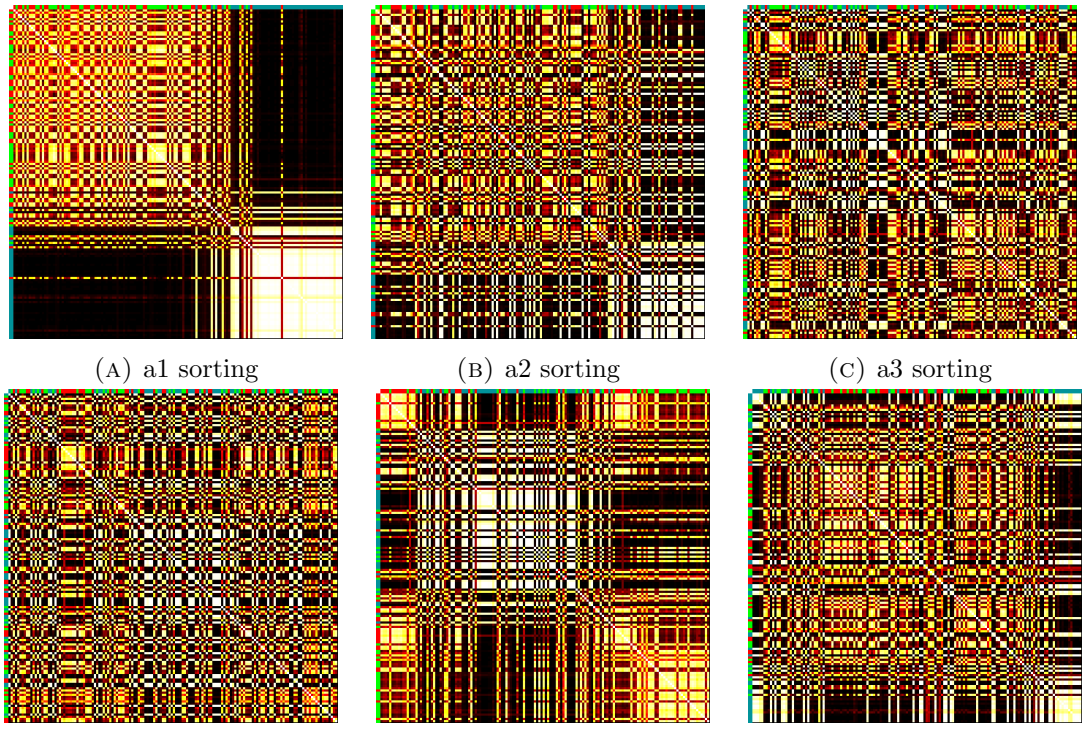
(A) A Random Forest with 100 trees (B) A Random Forest with 30 trees

FIGURE 6.3: The proximity matrix for the generated data set with 3 classes with 50 instances each. The blue class is disconnected from the other two. Red and green classes have stronger connection and are more often misclassified by the trees of the Random Forest.

While prototypes in Figure 6.5 (B) are close to each other which strengthens that the attributes a3 and a4 to be inaccurate predictors.

Some outliers can be noticed on the proximity matrix visualization in Figure 6.3 (A). There is an outlier among the red class's instances that is more often classified as a green class instance rather than red. Some instances are confused within red and blue classes. The scatter plots help to locate the outliers. Thus, it is possible to locate the red outlier with the scatter plot for the a4 and a5 attributes.

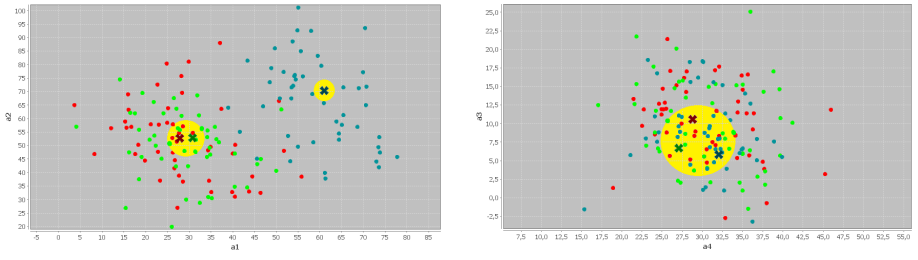
As for the split points, for 100 trees' forest the split points are densely located. The important variable a1 is used twice as often as the detrimental variables a3 and a4. With the Interaction View it is possible to find the split points for every important attribute.



(A) a1 sorting (B) a2 sorting (C) a3 sorting

(D) a4 sorting (E) a5 sorting (F) a6 sorting

FIGURE 6.4: Different sorting of the proximity matrix for the case. Different sortings help to establish the connection between variables and the classification. The attributes a1 (A) and a5 (E) have good prediction power for the blue class since the proximity matrix with the sorting still have a structure.



(A) The scatter plot for a1 and a2. (B) The scatter plot for a4 and a5.

FIGURE 6.5: (A) The scatter plot for a1 and a2 with 3 classes and their prototypes. The variable a1 is a good predictor for the blue class, since its prototype is separated. But not for the red and green classes, as the prototypes are close together. (B) The scatter plot for a4 and a5 and their prototypes. None of the two variables are good predictors, as prototypes are badly distinguishable.

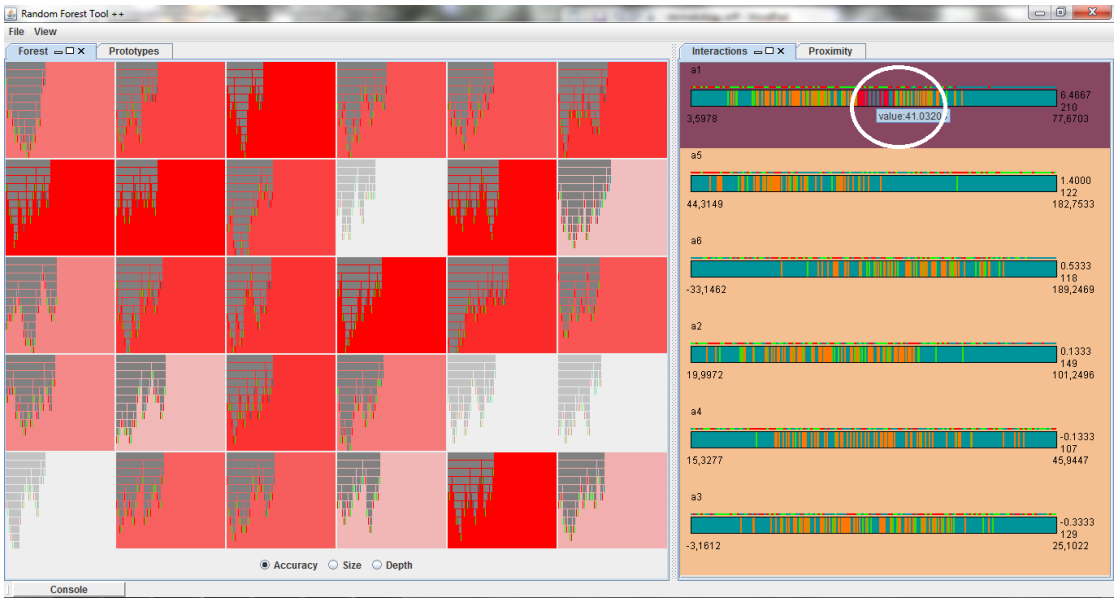
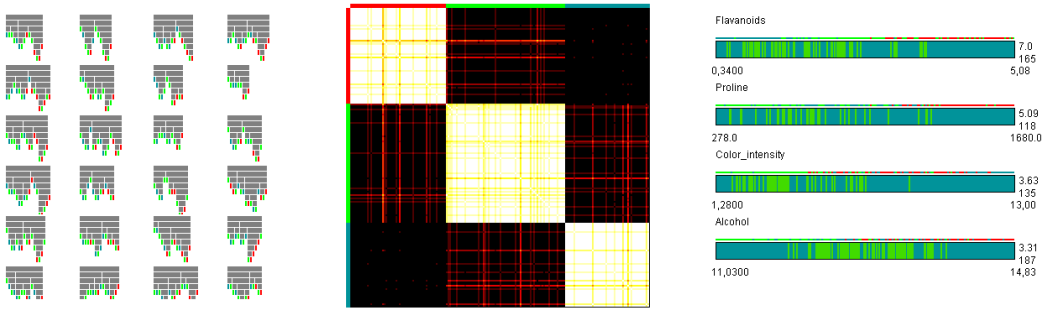


FIGURE 6.6: A Random Forest of 30 trees for the synthetic data set. The trees are highlighted according to the selection on the Interaction View: usage of corresponding split point of the attribute a1. The highlighted trees use a split point around value 41 for the attribute a1.

For this example, there are only 6 variables, so a user can inspect the scatter plots to find potential values for split points. Thus, for the attribute a1 in Figure 6.5 (A) a good candidate will be around 42. In case of a data set with a large number of attributes, a user can inspect the scatter plot between the most important attributes to find the split points.

To make the Interaction View less cluttered, the number of trees can be reduced. Thus, with 30 trees for the current data set, the OOB error is 0.3711. For this particular data set, a Random Forest with less trees outperforms a larger Random Forest, but the proximity matrix becomes less useful. Thus, on the proximity matrix in Figure 6.3 (B) it is harder to distinguish outliers.

Figure 6.6 shows the split points for the Random Forest of 30 trees. After inspecting the split points on the Interaction View, the split point of 41.03 was found for the attribute a1, as it is used within many trees. The different arrangements of the trees were tried to see if there is any relation among them. It seems that there is no pattern that can be recognized by a human within any of those arrangements. In the similar manner as Figure 6.6, a good split point for the attribute a5 is 97.8. The attributes a3 and a4 were not inspected, as they have been shown to be insignificant for the classification. To sum up, all initially generated features of the example data set were found with the use of a Random Forest model and ReFINE. The tool helps to identify all intended particularities of this synthetic data set. In addition, it helped to discover outliers in the data set. The tool seems to be useful for simple artificial datasets, the next step involves usage of ReFINE with real data sets.



(A) The Random Forest for the wine case (B) The proximity matrix for the case (C) Variable importance for the attributes

FIGURE 6.7: The resulting visualization for the case 2. (A) Forest visualization for the case. Structure of the tree is quite complex. (B) A proximity matrix for the case. Red class and green class have connections between each other, as well as green and blue classes. (C) The Interaction panel for the case, showing the most important variable and their split points.

6.2 Wine data set

The next data set is obtained from the UCI Machine Learning Repository [18] and belongs originally to the Institute of Pharmaceutical and Food Analysis and Technologies [19]. The data set contains wine recognition data. The data is the result of chemical analysis of different wine sorts grown in the same region in Italy. The data set contains 13 attributes and 178 instances, 59 for the first class, 71 and 48 for second and third classes correspondingly. Each class has a very defined structure, that is why the set is commonly used to design and test classifiers. For the same reason, most classifiers have good performance for the set. In this case, an expert wants to investigate what makes this data set so easy for the classification.

First, a Random Forest with 50 trees is built for the wine data set. The OOB error is low (0.1234) for the constructed forest, which is the first sign of well-structured set. From the tree structure in Figure 6.7 (A), it can be concluded that classes 1 and 3 are separated from each other, while class 1 has connections to class 2 and class 2 has connections to class 3. That conclusion is also supported by the proximity matrix and the prototypes visualization in Figure 6.7 (A) and (B). According to the built Random Forest model, the most important attributes are Flavanoids, Proline and Color_intensity. Figure 6.7 (C) shows a part of the Interaction View. The first attribute has good prediction power to separate all the classes. It has the range of [0.34,5.08] and values below 1.5 belong to the third class, between 1.5 and 3.1 to the second class, and the rest belong to the first one. Proline has possibility to distinguish the first class from the rest. Color_intensity attribute separates the second class.

Thus, ReFINE can be applied to the real data and can bring insights in the data structure. Although the data set has a pretty simple structure, next the tool is checked on more sophisticated data sets.

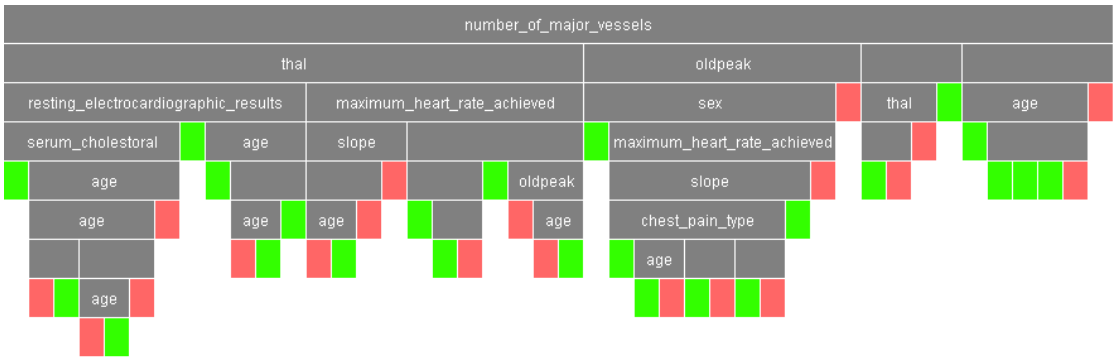


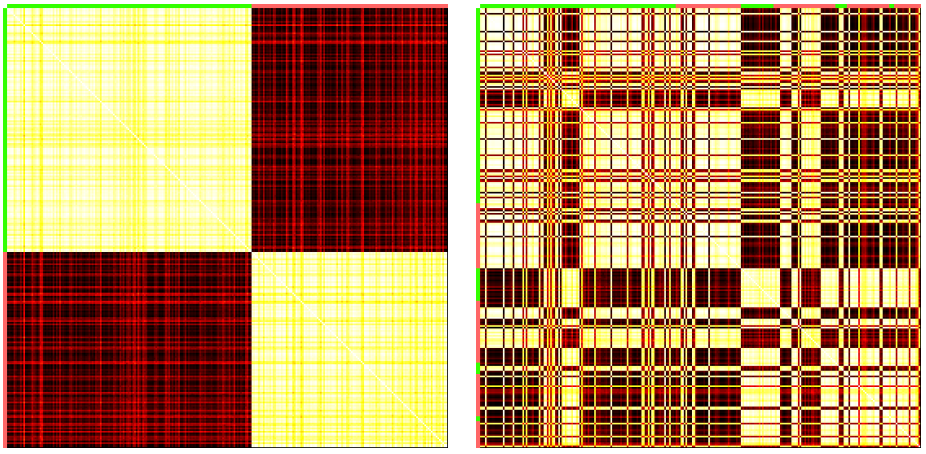
FIGURE 6.8: A random tree in the Random Forest for the heart disease log data set. The data set has 2 classes with numeric and nominal attributes. The structure of the tree is complex as there is no single attribute that separates the two classes well.

6.3 Heart disease log

The medicine area has many data sets since it is essential to record all observations. Thus, it is a common area where classification is practiced. The next data set presents a log for a heart disease problem. The set was also obtained from the UCI repository [18]. This data set contains 270 records for patients and has 13 attributes, including age, gender, blood pressure, blood sugar to name a few. The set has numerical and nominal attributes. There are two classes in the data set: absence and presence of heart diseases. A medical expert may want to know which recorded factors can predict presence of heart diseases.

First, a Random Forest model is built for the data set. The model has an OOB error rate of 0.2733. The data set has more complex structure compared to previous cases which results in complicated trees as well. There are only two classes in the data set, but the trees' structure in Figure 6.8 is complex. That shows that classes are hard to separate and there is a connection between classes. The generated proximity matrix, presented in Figure 6.9, confirms the conclusion as well. The classes are not separated from each other, as the values in the proximity matrix are the middle values and, thus, a user can see yellow and red in the matrix.

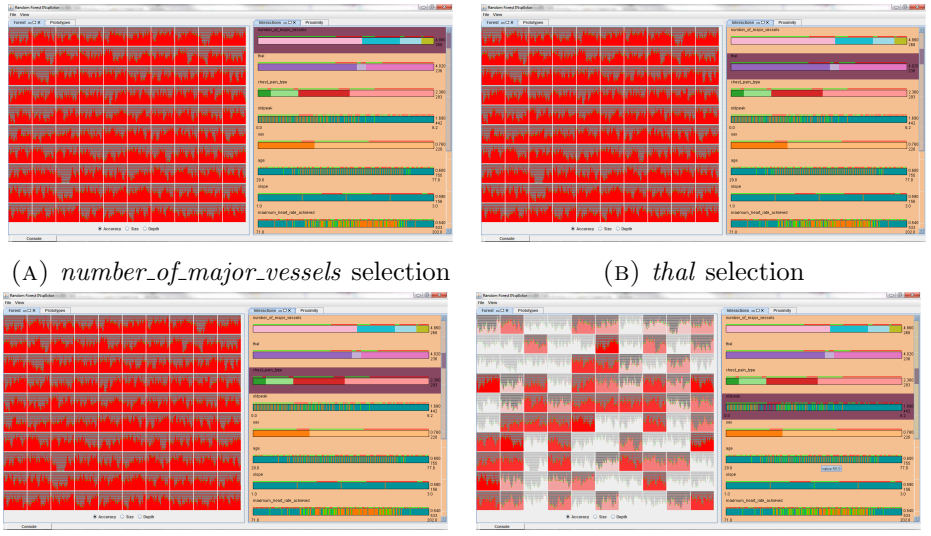
The most important attributes, according to the variable importance measure, are *number_of_major_vessels*, *thal* and *chest_pain_type*. The results are comparable to the result of feature selection method for the same data set [20]. The attributes are used in every tree of the built Random Forest, as Figure 6.10 (A), (B) and (C) shows. Figure 6.9 (B) shows the proximity matrix for this case sorted by the most important attribute. It is harder to see the connection of a nominal attribute to the classification on the proximity matrix. For nominal attributes it is easier to see the connection on the Attribute View. For instance, for the *chest_pain_type* attribute there is a visible connection in Figure 6.11. There are four types of chest pain according to the data set. People with either of the first three types of the chest pain are unlikely to have the heart disease, while the fourth type most probably leads to heart diseases.



(A) Proximity matrix with class sorting

(B) Proximity matrix with number_of_major_vessels sorting

FIGURE 6.9: The classes have connections between each other, as many instances are sometimes confused by the forest.



(A) *number_of_major_vessels* selection

(B) *thal* selection

(C) *number_of_major_vessels* selection

(D) *oldpeak* selection

FIGURE 6.10: The two most important attributes are used in all the trees of the Random Forest for the case.

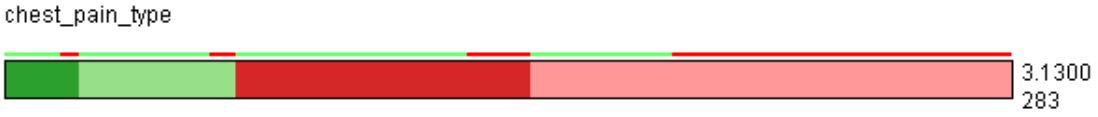


FIGURE 6.11: The *chest_pain_type* attribute visualization. From the class distribution it follows that the fourth type is most likely to lead to a heart disease, while types one, two and three are less likely to cause it.

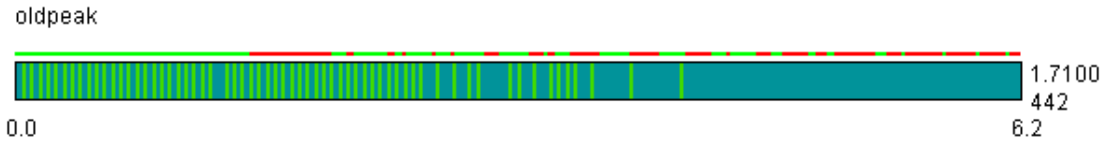


FIGURE 6.12: The *oldpeak* attribute visualization. The trees of the forest do not agree on the similar split points which result into the split point of the attribute being widely distributed. Still it can be noticed that the right side of the attribute range is not used. The values greater than approximately 4 of the attribute are not used as split points.

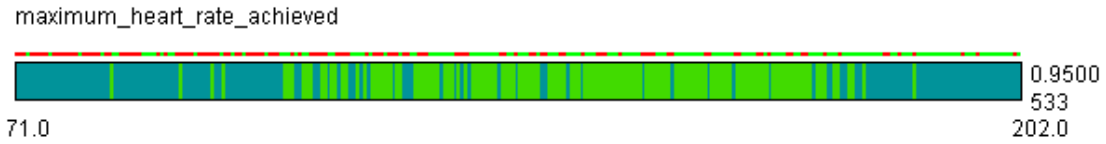


FIGURE 6.13: The *maximum_heart_rate_achieved* attribute visualization. The trees also do not agree on the split points. The split points are located in the range [120;170].

The *thal* attribute shows predisposition of a person to thalassemia, which is a blood disorder caused by weakness and destruction of red blood cells. The attribute take 3 values: 3, 6 and 7. In the similar way as for *chest_pain_type*, according to the *thal* Attribute View a person is likely to have a heart disease with *thal* greater than 6.

Medical data sets are hard to analyze without background knowledge in the field. All the conclusions made are based on the results given by the forest and common sense without any domain expertise.

6.4 Summary

The capabilities of ReFINE were tested on artificial and real data sets. The tool shows to be useful in data exploration process. The following aspects were noticed:

- The Forest View gives a general overview of the trees' structure, allowing a user to make observations about the complexity of a data set.
- The proximity matrix visualization shows the connection between classes of a data set. In addition, it is possible to see the connection of numerical attributes to the classification with the proximity matrix. Although, for nominal attributes it is not that obvious. The connection for those can be estimated with the Attribute View.
- The Prototype View is useful for data sets with small number of attributes, such that all pairs can be visualized. The view gives support for the attribute importance for the classification as well.
- The Interaction Views together with the Prototype View helps to find valuable split points for numerical attributes.

- Different arrangements for the Random Forest trees did not show to have a meaningful pattern even for a large number of trees.
- ReFINE is capable of handling both nominal and numerical attributes.

Chapter 7

Conclusions

In this work a prototype of a visualization tool for a Random Forest model was built after studying the construction and features of the model. Random Forest is a widely used approach for the classification problem. It is capable of giving sufficiently high results for the problem with relatively lower computation time. In addition, the forest model has some advantages over other models: proximity measure, prototypes and interactions.

The tool allows to visualize the listed features of the Random Forest. There are few existing visualizations for the Random Forest. ReFINE presents the components of the forest (trees) and allows a user to explore them. Moreover, the tool is the first to achieve useful visualization of the components, compared to the artistic way of presenting by Min Yang et al [14]. Furthermore, contrary to the original RAFT tool by Breiman, ReFINE shows the connection of the Random Forest feature to the trees. That gives a user the possibility to fully understand the origin of those features. In addition, it was demonstrated that the tool is able to help in the data exploration process. With the use of ReFINE some particularities of data sets were discovered and described.

7.1 Limitations and future work

The developed tool mostly focuses on the components and features of a Random Forest model. The tool takes an existing model. It can be useful to let a user interactively construct the forest to find a lower number of trees. It is easier for a human to understand the simplified model. The visualizations of the tool are more comprehensive for smaller models. ReFINE presents the model as-is, another solution in this case would be filtering. For example, let a user select subset of instances or attributes for the analysis.

It is possible to make some conclusions with the use of ReFINE, but the lack of knowledge in a domain area can lead to obvious or wrong decisions. Thus, the tool should be tested by domain experts with real data cases.

Bibliography

[1] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15 – 17, 1976.

[2] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[3] Antonio Criminisi, Jamie Shotton, Duncan Robertson, and Ender Konukoglu. Regression forests for efficient anatomy detection and localization in ct studies. In *Proceedings of the 2010 International MICCAI Conference on Medical Computer Vision: Recognition Techniques and Applications in Medical Imaging*, MCV'10, pages 106–117, Berlin, Heidelberg, 2011. Springer-Verlag.

[4] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124, January 2013.

[5] Tom Mitchell. *Machine Learning*. McGraw Hill, 1997.

[6] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. CRC Press, July 1984.

[7] John Ross Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[8] J. R. Quinlan. Simplifying decision trees. *Int. J. Man-Mach. Stud.*, 27(3):221–234, September 1987.

[9] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[10] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[11] Leo Breiman. Out-of-bag estimation. URL `http://www.stat.berkeley.edu/~breiman/OOBestimation.pdf`.

[12] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 161–168, New York, NY, USA, 2006. ACM.

[13] Henrik Boström. Concurrent learning of large-scale random forests. In Anders Kofod-Petersen, Fredrik Heintz, and Helge Langseth, editors, *SCAI*, volume 227 of *Frontiers in Artificial Intelligence and Applications*, pages 20–29. IOS Press, 2011.

[14] Min Yang, Hexin Xu, Dingju Zhu, and Huijuan Chen. Visualizing the random forest by 3d techniques. In Yongheng Wang and Xiaoming Zhang, editors, *Internet of Things*, volume 312 of *Communications in Computer and Information Science*, pages 639–645. Springer Berlin Heidelberg, 2012.

[15] Todd Barlow and Padraic Neville. A comparison of 2-d visualizations of hierarchies. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, INFOVIS '01, pages 131–, Washington, DC, USA, 2001. IEEE Computer Society.

[16] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, USA, 1986.

[17] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[18] K. Bache and M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[19] M Forina, S Lanteri, C Armanino, R Leardi, and G Drava. Parvus: An extendable package of programs for data explorative analysis. *Classification and Regression Analysis, Version*, 1, 1994.

[20] N.Ramaraj B. Sarojini. Enhancing medical prediction using feature selection. *International Journal of Artificial Intelligence and Expert Systems (IJAE)*, 1, 2011.