COMPARISON OF DATA MINING TOOLS:
KNIME VS RAPIDMINER

Goktug Cengiz
Mehmet Fatih Cagil

April 2019

Contents

1 Introduction

Data Mining is the extraction of interesting knowledge (nontrivial, implicit, previously unknown and potentially useful) that algorithmically detects specific patterns, trends in the data and rules mechanisms (associations between seemingly unconnected data). Data Mining is a multidisciplinary area that includes methods and techniques (including algorithms) from statistics and machine learning but also artificial intelligence, pattern recognition, databases and data visualization [1][2].

Data Mining tasks are related to the objectives. The purpose of the classification task is to build models capable of predicting the class of new cases. Data Mining models are mathematical representations aimed at understanding and studying of the data. Data Mining tasks are a set of processes involved in producing models implemented using techniques (algorithms). The Data Mining software tools combine fundamentals, theories, methods and algorithms. These applications base their operation in algorithms that look for patterns of knowledge by combining a set of tools for interrogation and exploration of data with tools that allow the visualization of results and reporting.

This work presents a comparative study between two free and open source data mining software tools (KNIME and RapidMiner). To evaluate the performance of the tools we used math measurement accuracy. The study aims at providing analysts with the tool and technique they may use to achieve fast and good results.

2 Open Source Data Mining Tools

Based on the most popular open source data mining tools in the market, we choose to analyze the following two ones: Rapidminer and KNIME.

2.1 RapidMiner

RapidMiner [3][4], formerly YALE (Yet Another Learning Environment), is an environment for machine learning, data mining, text mining, predictive analytics, and business analytics. It is used for research, education, training, rapid prototyping, application development, and industrial applications. We chose to use this tool because it is distributed under the AGPL open source license, and it is a data analytic tool used in real projects, ranked second in 2009 and first in 2010 according KDnuggets, a data-mining newspaper. RapidMiner provides a GUI to design an analytical process (reading data from source, transformations, applying algorithm). All GUI changes are stored in an XML (eXtensible Markup Language) file and then this file is read by RapidMiner to run the analyses.

RapidMiner contains all tools for data analysis from data processing (ETL), data modeling, data and result visualization, predictive analytics and statistical modeling, evaluation, and deployment. Other strong points of this tool is that it provides different visualization outputs such as 3D graphs, scattered matrices or maps, the multiple interfaces such as the GUI or the batch processing unit, the accuracy of pre-processing methods and the complete toolbox with over 1500 operations available. RapidMiner divides its Data Mining tasks in 7 groups – Classification and Regression, Attribute Weighting, Clustering and Segmentation, Association and Item Set Mining, Correlation and Dependency Computation, Similarity Computation and finally Model Application. Developed in Java, RapidMiner runs in every major platform and operating system. The open source version is very complete, so it is ideal for our purpose.

To sum up, RapidMiner main advantages are the following:

- Cross-validation at multiple levels using the Batch Cross Validation operator.
- Support for all computer environments.
- API that provides extension capabilities and versatility of configuration.
- Incorporates all of the algorithms available in Waikato Environment for Knowledge.(Weka)
- Provides R scripts.

- Support for in-memory, in-database and cluster processing.
- Wide range of metrics available for model assesments.

2.2 KNIME

In KNIME [5][6], the user can model workflows, which consist of nodes that process data, transported via connections between those nodes. A flow usually starts with a node that reads in data from some data source, which are usually text files, but databases can also be queried by special nodes. Imported data is stored in an internal table-based format consisting of columns with a certain (extendable) data type (integer, string, image, molecule, etc.) and an arbitrary number of rows conforming to the column specifications. These data tables are sent along the connections to other nodes that modify, transform, model, or visualize the data. Modifications can include handling of missing values, filtering of column or rows, oversampling, partitioning of the table into training and test data and many other operators. Following these preparatory steps, predictive models with machine learning or data mining algorithms such as decision trees, Naive Bayes classifiers or support vector machines are built.

For inspecting the results of an analysis workflow numerous view nodes are available, which display the data or the trained models in diverse ways. In contrast to many other workflow or pipelining tools, nodes in KNIME first process the entire input table before the results are forwarded to successor nodes. Intermediate results can be inspected at any time and new nodes can be inserted and may use already created data without preceding nodes having to be re-executed. One of the node's input are the training (or test) patterns, the output are cluster prototypes.

Each of the clusters covers several input patterns. By highlighting one or more clusters in the output table all input patterns which are part of those cluster(s) are highlighted in the input table. Similar translations are, of course, also possible for other summarizing models: branches/leaves of a decision tree, frequent patterns, discriminative molecular fragments, to name just three examples. One of the important design decisions was to ensure easy extensibility, so that other users can add functionality, usually in the form of new nodes (and sometimes also data types). This has already been done by several commercial vendors but also by other university groups or open source programmers. The usage of Eclipse as the core platform means that contributing nodes in the form of plugins is a very simple procedure.

- Each node stores its results permanently and thus workflow execution can easily be stopped at any node and resumed later on.
- Hiliting. In its simplest form, it allows the user to select and highlight several rows in a data table and the same rows are also highlighted in all

other views that show the same data table (or at least the highlighted rows).

- Scalability through sophisticated data handling (intelligent automatic caching of data in the background while maximizing throughput performance)
- Highly and easily extensible via a well-defined API for plugin extensions
- Intuitive user interface
- Import/export of workflows (for exchanging with other KNIME users)
- Parallel execution on multi-core systems
- Command line version for "headless" batch executions

Now that we described the platforms we will use for our study, is now the moment to explain what and how we will do with them.

3 Dataset

The chosen dataset downloaded from the UCI repository (University of California, Irvine) which is famous one called as Iris belong to R. A. Fisher. This is perhaps the best known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are **not** linearly separable from each other.

Predicted attribute: species of the iris plants.

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. species
    - Iris Setosa
    - Iris Versicolour
    - Iris Virginica

4 Methodology

Data Classification is a two steps process: (1) the training (or learning) phase and (2) the test (or evaluation) phase where the actual class of the instance is compared with the predicted class. If the hit rate is acceptable to the analyst, the classifier is accepted as being capable of classifying future instances with unknown class.

4.1 Training Phase

Cross-validation is the technique that we used to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it.

In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data.The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation.

The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once. Based on these advantages, we preferred k-fold cross validation instead of using partitioning node and percentage split method. For all of our tests, we used 10 fold cross validation

4.2 Classification Algorithms

Classification is typically obtained by supervised learning but can also be performed by unsupervised learning, e.g. where the class is not used or unknown as in the Clustering technique. For the test we use algorithms of the following techniques: (1) Decision Tree, (2) Bayesian classifier and (3) a Support Vector Machine.

4.2.1 Naive Bayes Classifier

Given a set of objects, each of which belongs to a known class, and each of which has a known vector of variables, our objective is to construct a rule which will permit us to assign future objects to a class, given the vectors of variables describing the future objects.

4.2.2 Decision Tree

Decision Tree is a non-parametric supervised learning method used for classification. Decision tree learn from data to approximate a sine curve. The deeper the tree, the more complex the decision rules and the fitter the model. Moreover, Decision tree constructs classification models in the form of a tree structure. It splits a data set into smaller and smaller subsets while also an associated decision tree is incrementally developed. In addition, A decision node has two or more branches. Finally, Decision trees can handle both categorical and numerical data.

4.2.3 Support Vector Machine

Nowadays support vector machines are considered one of the most robust and accurate methods among all well-known algorithms. It has a good theoretical foundation, requires only a dozen examples for training and is insensitive to the number of dimensions.

In the aim of SVM is to find the best classification function to distinguish between members of the two classes in the training data. The metric for the concept of the "best" classification function can be realized geometrically.

4.3 Performance Evaluation

The performance evaluation of the classifiers will be assessed by the accuracy metric. This metric is calculated dividing the number of instances correctly classified by the total value of instances. A correctly classified instance is one in which the classifier predicts the correct class of the test's instance.

5 Data Mining Tools Comparison

The experimental work we will do will consist of test an algorithm in the area of classification. The algorithm will be tested and compared following specific parameters we will define at the end of this section. These experiments will be runned in a computer with Windows 10 64 bits Operating System, 16 GB of RAM and i7 7700HQ CPU. Finally, in our experiment we will test two different open source mining platforms, Rapidminer and KNIME; because these platforms are featured as the most popular and most used tools for real big data projects.

5.1 Data Pre-Processing

One of the main advantages of the iris dataset is, it is ready to use. Missing value imputation, outlier detection or any other time consuming applications are not needed; that is why it is practical to use the dataset for comparing two different platforms.

5.2 Test with KNIME

First, we are reading the iris dataset with a *Read CSV* node, then streaming the data to the respective metanodes. We used metanodes for each classification type to keep them organized and easy to manage.

When we streamed the data to each metanode, we are partitioning the data with *X-Partitioner* node because we are using cross validation method. After executing the respective learner and predictor nodes for each classification methods (details explained in following sections), we are aggregating the data with *X-Aggregator* and gathered the results with *Scorer* node.

5.2.1 Naive Bayes

We used *Species* as the classification column for the *Naive Bayes Learner* node, with default probability = 0.0001 which is the default value for default probability. Default probability is used when the attribute is nominal and was not seen by the learner or continuous and its probability is smaller than the default probability; which is not true for our case.

Results of *Naive Bayes Learner* node is combined with the *X-Partitioner* at the *Naive Bayes Predictor*. Default values are used for the predictor. After executing the prediction node, data aggregated at the *X-Aggregator* node to conclude the cross validation. In order to see the results, data streamed to the *Scorer* node.

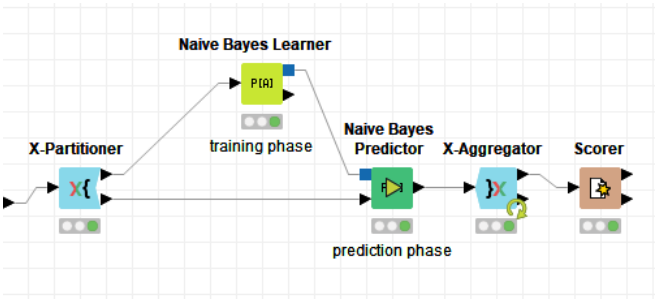

Figure 1: Workflow of Naive Bayes Prediction

Prediction is pretty accurate with 95.33%, details can be seen at *Figure 2*

Species \ N...	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	50	0	0
Iris-versicolor	0	47	3
Iris-virginica	0	4	46

Correct classified: 143 Wrong classified: 7

Accuracy: 95.333 % Error: 4.667 %

Cohen's kappa (κ) 0.93

Figure 2: Accuracy of Naive Bayes Prediction

5.2.2 Support Vector Machine

We used *Species* as the prediction class again, *SVM Learner* node configured with RBF kernel and sigma 0.1 options. Then similar process followed as the Naive Bayes section. *SVM Predictor* node concludes the prediction and *Scorer* node collects the results.

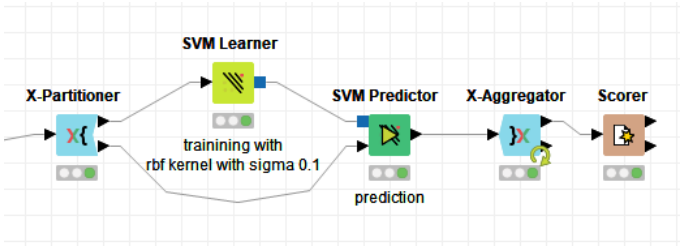

Figure 3: Workflow of Support Vector Machine

Accuracy of SVM is lower than the Naive Bayes, details can be seen at the *Figure 4*

Species \ S...	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	34	0	16
Iris-versicolor	0	35	15
Iris-virginica	0	1	49

Correct classified: 118 Wrong classified: 32

Accuracy: 78.667 % Error: 21.333 %

Cohen's kappa (κ) 0.68

Figure 4: Accuracy of Support Vector Machine

5.2.3 Decision Tree

Default values are used for *Decision Tree Learner* and *Decision Tree Predictor* nodes. Quality measure for the learner node selected as *Gini Index* and reduced error pruning is active. Starting at the leaves, each node is replaced with its most popular class, but only if the prediction accuracy doesn't decrease. Reduced error pruning has the advantage of simplicity and speed.

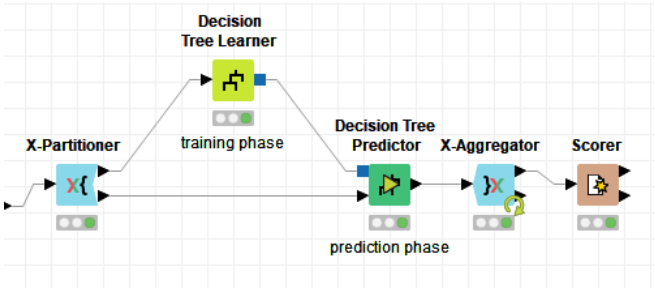

Figure 5: Workflow of Decision Tree

As you see in *Figure 13*, decision tree method gives us a decision tree as an output. We can ask why algorithm starts PetalWidthCm instead of PetalLengthCm. This is because we should require to look at a situation involving the possibilities of change that we have explained with entropy and that the data is distributed. Entropy is a measure of unpredictability of the state, or equivalently, of its average information content. Entropy is calculated by the formula as you see below.

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i) \tag{1}$$

where b is the base of the logarithm used. Common values of b are 2, Euler's number e, and 10, and the corresponding units of entropy are the bits for b = 2, nats for b = e, and bans for b = 10.

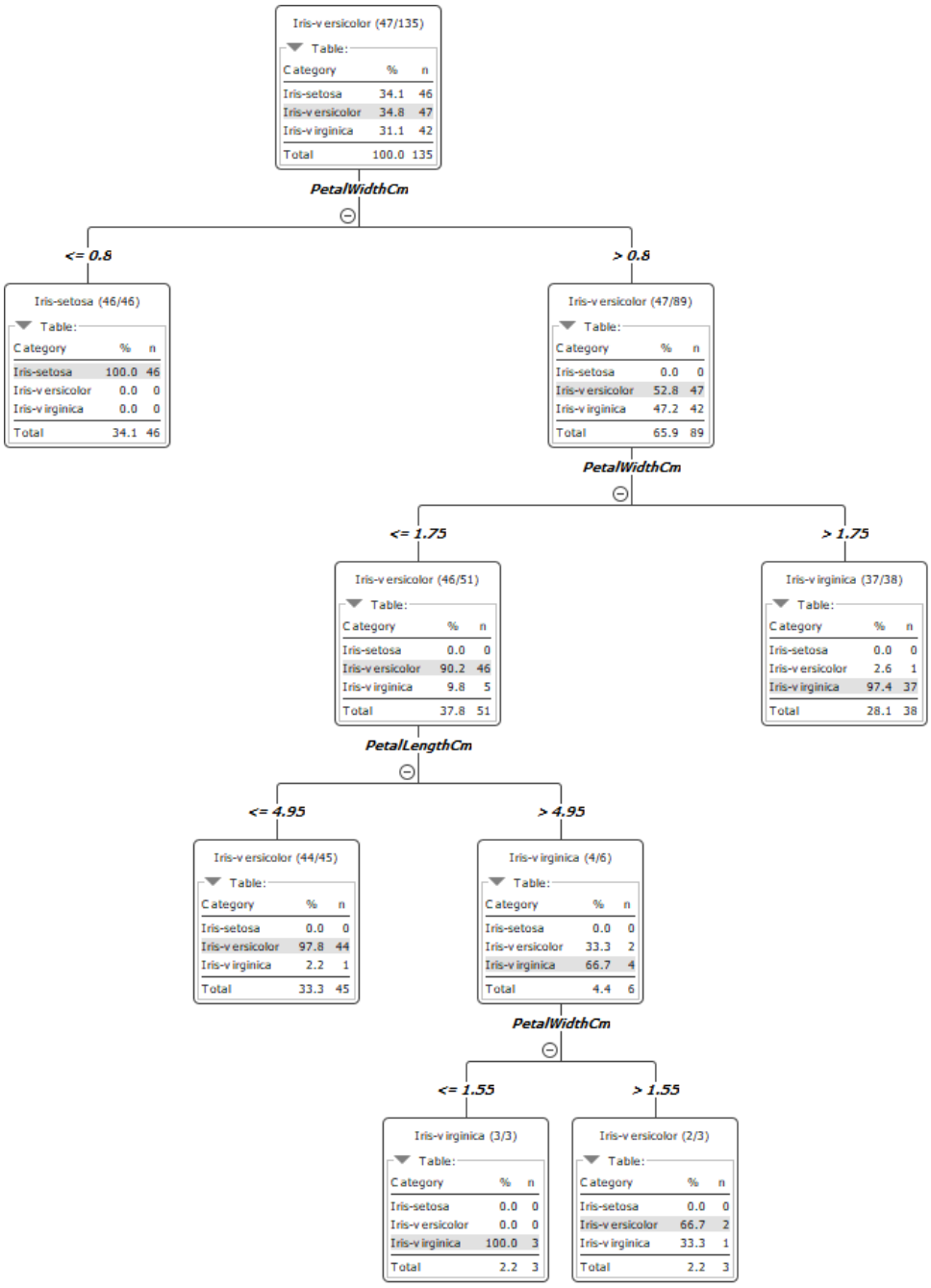

Figure 6: Visualization of Decision Tree

Accuracy of the deicison tree method is very similar to naive bayes.

Species \ D...	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	50	0	0
Iris-versicolor	0	45	5
Iris-virginica	0	4	46

Correct classified: 141 Wrong classified: 9

Accuracy: 94 % Error: 6 %

Cohen's kappa (κ) 0.91

Figure 7: Accuracy of Decision Tree

5.3 Test with RapidMiner

RapidMiner works a little different than KNIME, we have to put our training and prediction nodes inside of the *Cross Validation* node and we don't have different prediction nodes for different methods. We connect training part to *Apply Model* node and it applies the respective prediction method according to the training part.

5.3.1 Naive Bayes

Default values are used for Naive Bayes training, *Species* is the classification column again.

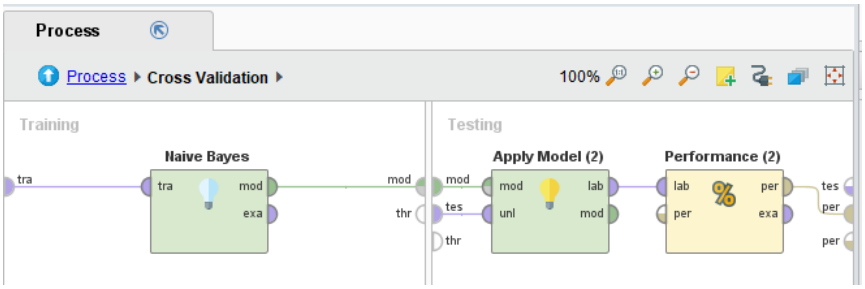

Figure 8: Workflow of Naive Bayes Prediction

Results of Naive Bayes are identical with respective KNIME results.

accuracy: 95.33% +/- 4.27% (micro average: 95.33%)

	true Iris-setosa	true Iris-versicolor	true Iris-virginica	class precision
pred. Iris-setosa	50	0	0	100.00%
pred. Iris-versicolor	0	47	4	92.16%
pred. Iris-virginica	0	3	46	93.88%
class recall	100.00%	94.00%	92.00%	

Figure 9: Accuracy of Naive Bayes Prediction

5.3.2 Support Vector Machine

Process is very similar to Naive Bayes one, all we had to to replace the Naive Bayes training node with SVM training node, we used default values and rdf kernel with sigma = 0.1

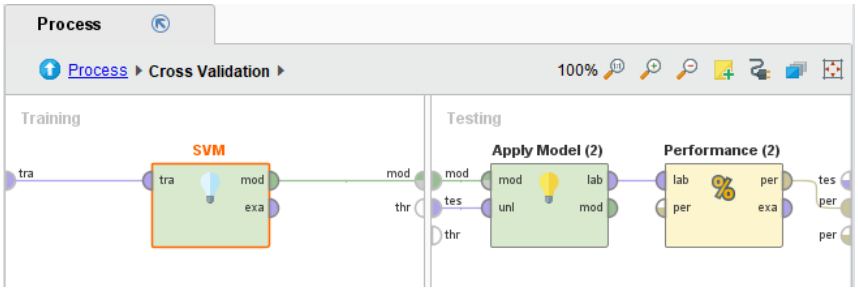

Figure 10: Workflow of Support Vector Machine

Accuracy of SVM is not better than other two methods; but when you compare the results of RapidMiner-SVM with KNIME-SVM, it can be seen that RapidMiner gives us better results.

accuracy: 92.67% +/- 5.54% (micro average: 92.67%)

	true Iris-setosa	true Iris-versicolor	true Iris-virginica	class precision
pred. Iris-setosa	50	0	0	100.00%
pred. Iris-versicolor	0	47	8	85.45%
pred. Iris-virginica	0	3	42	93.33%
class recall	100.00%	94.00%	84.00%	

Figure 11: Accuracy of Support Vector Machine

5.3.3 Decision Tree

Idea behind the decision tree is identical with respective KNIME application. Default values are used.

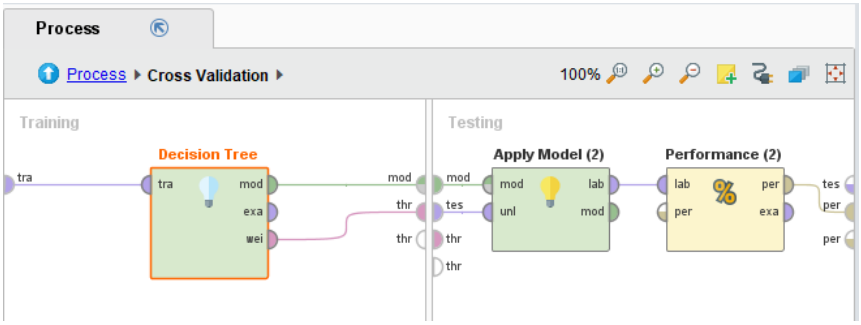

Figure 12: Workflow of Decision Tree

Figure 13 shows the visualization of the decision tree and its decide making structure. Reasoning behind it is identical to the one explained at the KNIME decision tree.

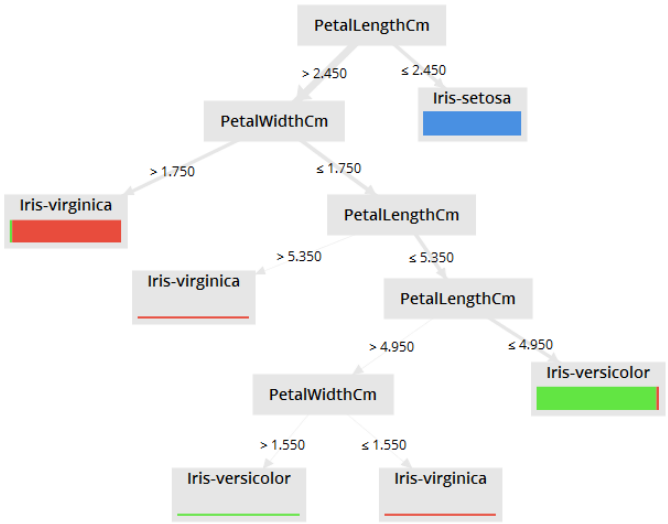

Figure 13: Visualization of Decision Tree

Again, results is almost same with the KNIME decision tree results and Naive Bayes results.

accuracy: 95.33% +/- 4.27% (micro average: 95.33%)

	true Iris-setosa	true Iris-versicolor	true Iris-virginica	class precision
pred. Iris-setosa	50	0	0	100.00%
pred. Iris-versicolor	0	46	3	93.88%
pred. Iris-virginica	0	4	47	92.16%
class recall	100.00%	92.00%	94.00%	

Figure 14: Accuracy of Decision Tree

5.4 Comparison

Naive Bayes Comparison			
Data Mining Tool	Accuracy	Correct Classified	Wrong Classified
KNIME	95.333%	143	7
RapidMiner	95.333%	143	7
SVM Comparison			
Data Mining Tool	Accuracy	Correct Classified	Wrong Classified
KNIME	78.667%	118	32
RapidMiner	92.67%	139	11
Decision Tree Comparison			
Data Mining Tool	Accuracy	Correct Classified	Wrong Classified
KNIME	94%	141	9
RapidMiner	95.333%	143	7

6 Conclusion

In this brief case study, we compared KNIME with RapidMiner. Results are very close to each other, RapidMiner has a little advantage on the support vector machine method over the KNIME. Using just one dataset is not sufficient enough to say that RapidMiner is better than KNIME, we can just conclude that RapidMiner gives better results with the dataset for three methods that we used. However, in terms of user interface and user experience, KNIME looks better than RapidMiner.

Namings are more understandable and clear at KNIME, color palettes are also more organized. And the traffic lights concept which indicates whether a node is ready to execute or not, is very helpful for beginners. If the light is red, some configurations are required, if its yellow, its ready to execute. If it is green, it has already executed. It is possible to link a node to multiple different nodes in KNIME but in order to achieve the same thing in RapidMiner, you have to use a multiplier node. Also its possible to run nodes individually in KNIME but not in RapidMiner.

References

[1] Santos, M., Ramos, I. 2009. Business Intelligence - Tecnologias da Informação na Gestão de Conhecimento. FCA - Editora de informática, Lda., 2nd edition

[2] Santos, M.F., Azevedo, C. 2005. Data Mining - Descoberta de conhecimento em bases de dados. FCA - Editora de Informática, Lda., 1st edition

[3] Admin. (n.d.). Lightning Fast Data Science Platform for Teams | RapidMiner©. Retrieved April 26, 2019, from https://rapidminer.com/

[4] RapidMiner Studio - EduTech Wiki. (n.d.). Retrieved April 26, 2019, from http://edutechwiki.unige.ch/en/RapidMiner_Studio

[5] KNIME – Open for Innovation. (n.d.). Retrieved April 26, 2019, from https://www.knime.com/

[6] Knime – EduTech Wiki. (n.d.). Retrieved January April 26, 2019, from http://edutechwiki.unige.ch/en/Knime