DOI: 10.15439/2023F4101
ISSN 2300-5963 ACSIS, Vol. 35

QMAK: Interacting with Machine Learning Models and Visualizing Classification Process

Arkadiusz Wojna, Katarzyna Jachim, Łukasz Kosson, Łukasz Kowalski, Damian Mański, Michał Mański, Krzysztof Mroczek, Krzysztof Niemkiewicz, Robert Piszczatowski, Maciej Próchniak, Tomasz Romańczuk, Piotr Skibiński, Marcin Staszczyk, Michał Szostakiewicz, Leszek Tur, Damian Wójcik, Maciej Zuchniak
University of Warsaw
Faculty of Mathematics, Informatics and Mechanics
Banacha 2, 02-097 Warsaw, Poland
Email: wojna@mimuw.edu.pl

***Abstract*—In various classification problems beside high accuracy data analysts expect often understanding and certain insight into the process of classification. To help them understand why a trained model selects a particular decision, how confident it is in the assigned decision, and to enable interactive improvement of trained models we present QMAK. The tool visualizes not only classification models but also the processes classifying individual objects. Five classical machine learning models and their classification process are visualized with QMAK: neural network, decision tree, k-nearest neighbors, classifier based on principal component analysis (PCA) and rough set based classifier. QMAK provides also exemplary functions enabling users to modify trained models interactively.**

***Index Terms*—explainable machine learning, classification visualization, interactive classifier.**

I. Introduction

As THE field of machine learning matured, beside a predicted value data analysts started to expect some insight into how a trained model made a decision. They expect explanation why decision was selected and how much the model is sure of its decision. Understanding why classification models classify incorrectly some cases can help both authors and analysts improve their models. Yet further step it is to provide the ability of modifying a trained model in an interactive process.

There are many visualization tools dedicated to machine learning but most of them use various charts, sometimes quite advanced, to visualize data, training metrics and classification results, for example Neptune [1]. Many others can visualize classification models based on the graph structure like Graphviz [2] and TensorBoard [3], or on the tree structure like dtreeviz [4], or both, for example Weka [5]. However, only the models having one of these two structures can be visualized with those tools. The tools visualizing classification models reflecting their specific structure are often most advanced and most detailed in how they present a model, but they are usually dedicated to a single model type, for example Netron [6] and NN-SVG [7] that visualize neural networks.

QMAK is a visualization and interactive platform gathering different classification models. It provides a framework not only for visualization and interaction with models but also for visualization of the classification process. It allows users to compare both the structure of different models and how they differ in the classification processes provided with the same object to be classified. QMAK implements visualization of popular machine learning models and gives examples of how users may interact with models to improve their classification accuracy. The platform can be used also as a didactic tool during machine learning courses.

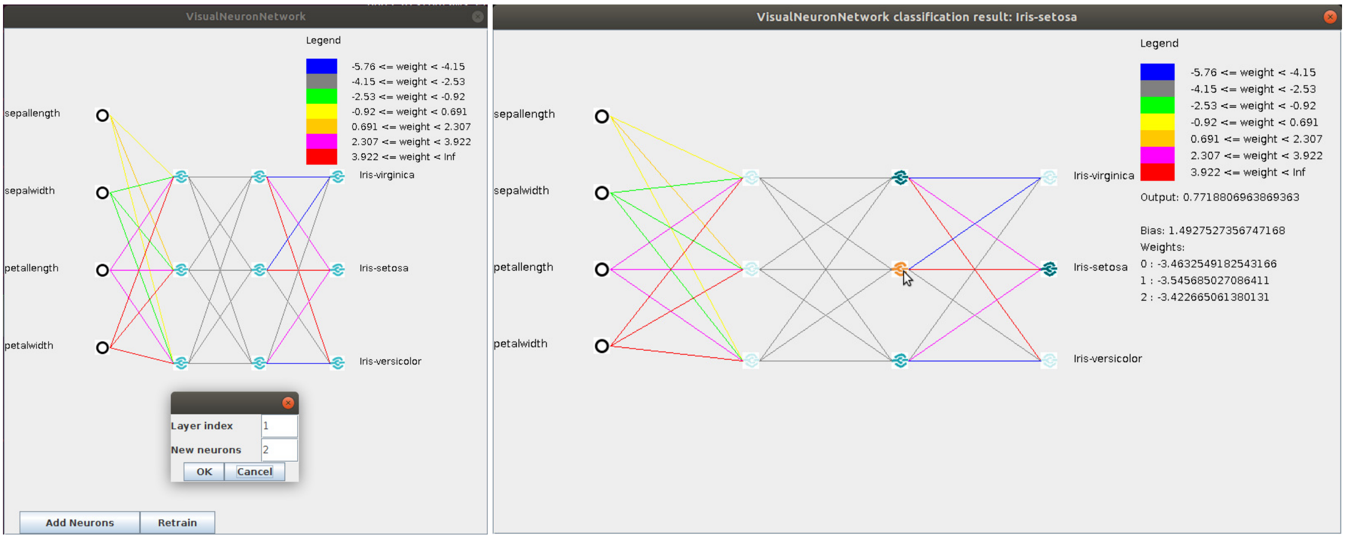
II. System Overview

QMAK is a graphical tool providing the following features: visualization of data, classifiers and single object classification, interactive classifier modification by a user, classification of test data with presentation of misclassified objects, and experiments comparing classification accuracy of different classifiers using different types of tests. It is an open source software issued under the GNU General Public License. The tool and its demo are available at http://rseslib.mimuw.edu.pl/qmak.

QMAK uses Rseslib library [8], [9] as the source of classification models. The version 3.3.0 provides visualization of five classifiers: neural network, decision tree, k-nearest neighbors, classifier based on principal component analysis (PCA) and rough set based rule classifier. Users can implement new classifiers and their visualization and add them easily to QMAK.

Neural network in QMAK is trained with the classical backpropagation algorithm and sigmoid activation functions [10]. Visualization of a neural network presents the neurons and the connections between them (see Figure 1). The neurons from the last layer correspond to decisions. The color of a connection represents its weight as it is defined in the legend. A user can select a neuron to display the exact weights of its input connections and its bias. They can also modify a trained network by adding new neurons in hidden layers and retraining the network. Visualization of classification presents also the strength of the output signal from each neuron with intensity of its color, and the exact value of the output signal after clicking on a selected node.

The decision tree in QMAK is implementation of the well-known C4.5 algorithm [11]. It is visualized by presenting the structure of the tree (see Figure 2). After selection of a node the decision distribution of the training objects entering that



Legend

🟦	-5.76 <= weight < -4.15
▩	-4.15 <= weight < -2.53
🟩	-2.53 <= weight < -0.92
🟨	-0.92 <= weight < 0.691
🟧	0.691 <= weight < 2.307
🟪	2.307 <= weight < 3.922
🟥	3.922 <= weight < Inf

Layer index	1
New neurons	2

Output: 0.7718806963869363

Bias: 1.4927527356747168
Weights:
0 : -3.4632549182543166
1 : -3.545685027086411
2 : -3.422665061380131

Figure 1. Visualization of neural network (left) and its classification process (right)

node is displayed, and the branching condition for an internal node or the assigned decision for a leaf node. A user can cut off the subtree of any internal node and convert it to a leaf. Visualization of classification presents a decision tree with the path from the root to a leaf node highlighted in green corresponding to a classified object.

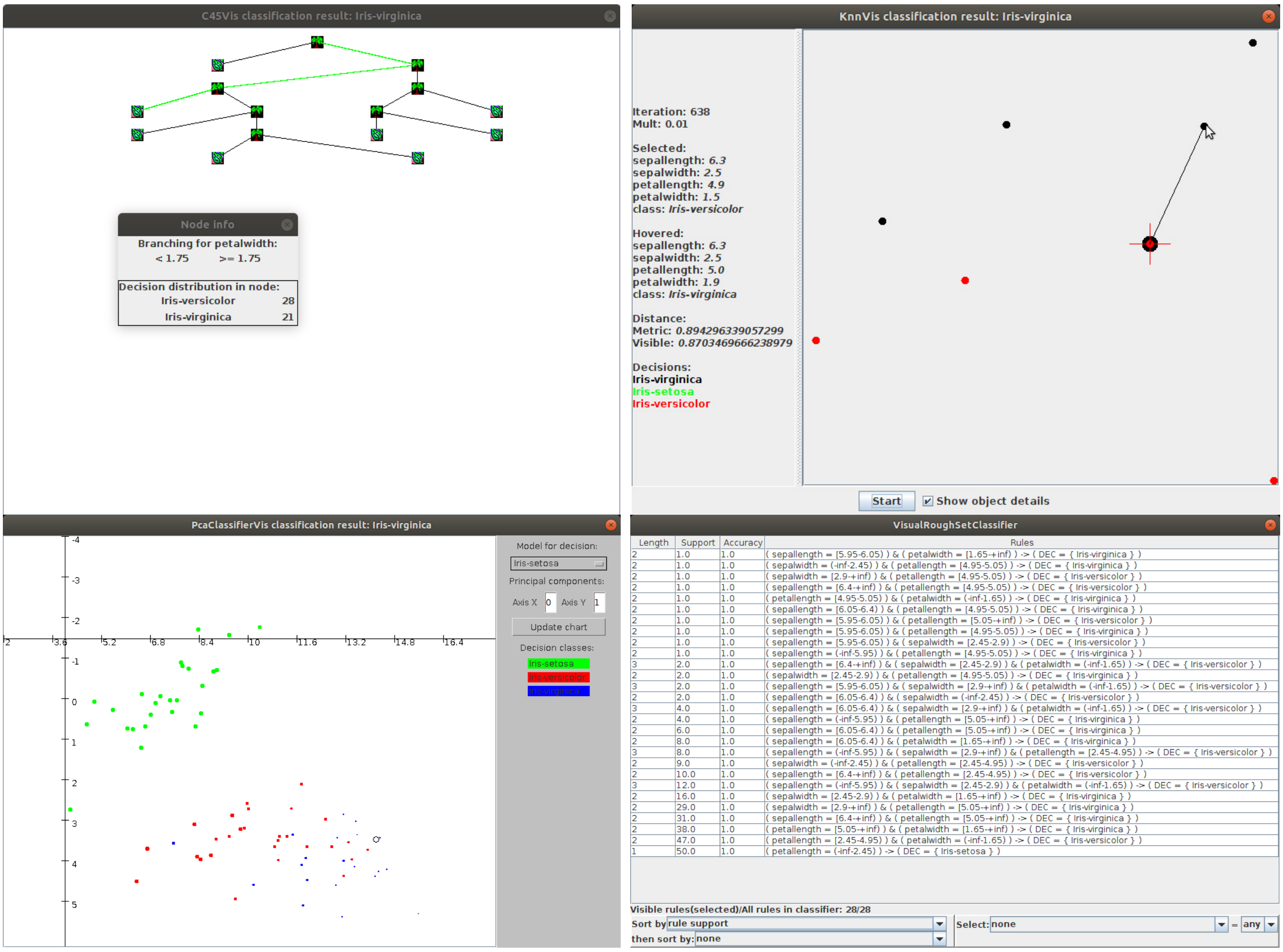
K-nearest neighbours classifier [12] provides distance measures working for data with both numerical and categorical attributes and optimized with attribute weighting. It optimizes automatically also the number k of nearest neighbours and applies weights in voting by nearest neighbours. Visualization of k-NN classifier projects all training objects onto the two-dimensional area of the window, marking the objects of each decision class with a different color. The process of searching for placement of the objects that most faithfully reflects the true distances between them in the induced metric, is displayed live, and can be stopped at any time. A user can select one object and hover the cursor over another one to display the attribute values of both objects and the true distance between them. Visualization of classification by k-NN classifier projects only the classified object and its k nearest neighbors onto the window area, also searching for placement most faithfully reflecting the true distances between them. As in the model visualization, the neighbors from different decision classes have different colors, and a user can display their attribute values and the true distances between them.

The visualized search for the best placement of objects in the two-dimensional area of the QMAK window for k-NN classifier uses an algorithm that combines simulation of spring-line attraction and repulsion with simulated annealing. Each object is assigned random initial coordinates, sampled uniformly from the unit square. The coordinates are then refined in an iterative process. In each iteration two distances are computed for each pair of objects: the true distance in the induced metric and the Euclidean distance between the current coordinates. The difference between these two distances is later applied as the multiplication factor to the vector between the current coordinates of the objects to obtain the force vector. The correction vector for each object is computed as the sum of all force vectors for that object. At the end of each iteration, the correction vectors are multiplied by a scaling factor, reduced to a maximum length, and applied to the current coordinates. The scaling factor decreases exponentially with each epoch to ensure long-term pseudostability of otherwise chaotic N-body problem, while reduction to maximum length reduces instabilities in early iterations.

PCA classifier finds a separate model of principal components for each decision class using Oja-RLS rule [13]. Its visualization projects all training objects onto the plane spanned by a selected pair of principal components of the model for a selected decision class. The objects of each decision class are marked with a different color. The objects closer to the plane are represented by larger dots, the ones more distant from the plane are represented by smaller dots. A user can switch between different decision classes and different pairs of principal components. Visualization of classification by PCA classifier marks additionally the position of the classified object on the presented plane.

Rough set classifier uses the algorithms computing discernibility matrix, reducts and rules generated from reducts [14], [15], [8]. Its visualization presents the decision rules with their length, support and accuracy. A user can filter and sort the rules by attribute occurrence, attribute values, rule length, support or accuracy. Visualization of classification shows only the rules matching the classified object enabling the same filtering and sorting criteria as visualization of the classifier.

Beside visualization of classification QMAK integrates also other features. Users can run experiments comparing the accuracy of different classification models, or the accuracy of the same classification model for different parameter settings,


C45Vis classification result: Iris-virginica
Node info
KnnVis classification result: Iris-virginica
Start
☑ Show object details
PcaClassifierVis classification result: Iris-virginica
-4
-3
-2
-1
0
1
2
3
4
5
2
3.6
5.2
6.8
8.4
10
11.6
13.2
14.8
16.4
Update chart
VisualRoughSetClassifier

Branching for petalwidth:
< 1.75 >= 1.75

Decision distribution in node:	
Iris-versicolor	28
Iris-virginica	21

Iteration: 638
Mult: 0.01

Selected:
sepallength: *6.3*
sepalwidth: *2.5*
petallength: *4.9*
petalwidth: *1.5*
class: *Iris-versicolor*

Hovered:
sepallength: *6.3*
sepalwidth: *2.5*
petallength: *5.0*
petalwidth: *1.9*
class: *Iris-virginica*

Distance:
Metric: *0.894296339057299*
Visible: *0.8703469666238979*

Decisions:
Iris-virginica
Iris-setosa
Iris-versicolor

Model for decision: Iris-setosa
Principal components:
Axis X 0 Axis Y 1

Decision classes:
🟩 Iris-setosa
🟥 Iris-versicolor
🟦 Iris-virginica

Length	Support	Accuracy	Rules
2	1.0	1.0	(sepallength = [5.95-6.05)) & (petalwidth = [1.65-+inf)) -> (DEC = { Iris-virginica })
2	1.0	1.0	(sepalwidth = (-inf-2.45)) & (petallength = [4.95-5.05)) -> (DEC = { Iris-virginica })
2	1.0	1.0	(sepalwidth = [2.9-+inf)) & (petallength = [4.95-5.05)) -> (DEC = { Iris-versicolor })
2	1.0	1.0	(sepallength = [6.4-+inf)) & (petallength = [4.95-5.05)) -> (DEC = { Iris-versicolor })
2	1.0	1.0	(petallength = [4.95-5.05)) & (petalwidth = (-inf-1.65)) -> (DEC = { Iris-virginica })
2	1.0	1.0	(sepallength = [6.05-6.4)) & (petallength = [4.95-5.05)) -> (DEC = { Iris-virginica })
2	1.0	1.0	(sepallength = [5.95-6.05)) & (petallength = [5.05-+inf)) -> (DEC = { Iris-versicolor })
2	1.0	1.0	(sepallength = [5.95-6.05)) & (petallength = [4.95-5.05)) -> (DEC = { Iris-virginica })
2	1.0	1.0	(sepallength = [5.95-6.05)) & (sepalwidth = [2.45-2.9)) -> (DEC = { Iris-versicolor })
2	1.0	1.0	(sepallength = (-inf-5.95)) & (petallength = [4.95-5.05)) -> (DEC = { Iris-virginica })
3	2.0	1.0	(sepallength = [6.4-+inf)) & (sepalwidth = [2.45-2.9)) & (petalwidth = (-inf-1.65)) -> (DEC = { Iris-versicolor })
2	2.0	1.0	(sepalwidth = [2.45-2.9)) & (petallength = [4.95-5.05)) -> (DEC = { Iris-virginica })
3	2.0	1.0	(sepallength = [5.95-6.05)) & (sepalwidth = [2.9-+inf)) & (petalwidth = (-inf-1.65)) -> (DEC = { Iris-versicolor })
2	2.0	1.0	(sepallength = [6.05-6.4)) & (sepalwidth = (-inf-2.45)) -> (DEC = { Iris-versicolor })
3	4.0	1.0	(sepallength = [6.05-6.4)) & (sepalwidth = [2.9-+inf)) & (petalwidth = (-inf-1.65)) -> (DEC = { Iris-versicolor })
2	4.0	1.0	(sepallength = (-inf-5.95)) & (petallength = [5.05-+inf)) -> (DEC = { Iris-virginica })
2	6.0	1.0	(sepallength = [6.05-6.4)) & (petallength = [5.05-+inf)) -> (DEC = { Iris-virginica })
2	8.0	1.0	(sepallength = [6.05-6.4)) & (petalwidth = [1.65-+inf)) -> (DEC = { Iris-virginica })
3	8.0	1.0	(sepallength = (-inf-5.95)) & (sepalwidth = [2.9-+inf)) & (petallength = [2.45-4.95)) -> (DEC = { Iris-versicolor })
2	9.0	1.0	(sepalwidth = (-inf-2.45)) & (petallength = [2.45-4.95)) -> (DEC = { Iris-versicolor })
2	10.0	1.0	(sepallength = [6.4-+inf)) & (petallength = [2.45-4.95)) -> (DEC = { Iris-versicolor })
3	12.0	1.0	(sepallength = (-inf-5.95)) & (sepalwidth = [2.45-2.9)) & (petalwidth = (-inf-1.65)) -> (DEC = { Iris-versicolor })
2	16.0	1.0	(sepalwidth = [2.45-2.9)) & (petalwidth = [1.65-+inf)) -> (DEC = { Iris-virginica })
2	29.0	1.0	(sepalwidth = [2.9-+inf)) & (petallength = [5.05-+inf)) -> (DEC = { Iris-virginica })
2	31.0	1.0	(sepallength = [6.4-+inf)) & (petallength = [5.05-+inf)) -> (DEC = { Iris-virginica })
2	38.0	1.0	(petallength = [5.05-+inf)) & (petalwidth = [1.65-+inf)) -> (DEC = { Iris-virginica })
2	47.0	1.0	(petallength = [2.45-4.95)) & (petalwidth = (-inf-1.65)) -> (DEC = { Iris-versicolor })
1	50.0	1.0	(petallength = (-inf-2.45)) -> (DEC = { Iris-setosa })

Visible rules(selected)/All rules in classifier: 28/28
Sort by rule support
then sort by: none
Select: none = any

Figure 2. Decision tree (upper left), k-nearest neighbors classifier (upper right), PCA classifier (lower left) and rough set classifier (lower right) visualized in QMAK

for example, using cross-validation or multiple random split and test. In those experiment all five visualized classification models are available for testing as well as other non-visualized classifiers: Support Vector Machine, AQ15, Naive Bayes, RIONIDA, and others.

The tool provides also three kinds of data graphs presenting different types of correlations between categorical and numerical attributes.

III. System Usage

QMAK can help users understand why a particular decision was selected. For example, the classification path for an object in a decision tree shows the attributes and the conditions on these attributes determining the decision. K-nearest neighbors model shows the training objects identified as the most similar to the classified object and used to vote for the predicted decision. Rough set classifier shows the decision rules matching the classified object.

Using QMAK users can also find out how confident a classifier is in the assigned decision. In neural network it is indicated by the difference between the strength of the output signal from the winning neuron and the strength of the signals from other neurons. In k-nearest neighbors model one can compare the number and the placement of the nearest neighbors with the winning decision with the number and the placement of the neighbors with other decisions.

Users can also find out which element of an already trained classifier needs to be improved. QMAK highlights misclassified objects in red, and for each misclassified object a user can command the classifier to visualize its classification. In some cases, the user can later modify the classifier interactively. QMAK demonstrates a few examples of such interaction.

Similar weights of the connections between two layers in a neural network like between the two hidden layers in Figure 1, may indicate the first of those two hidden layers was given too few neurons and the second of the two layers was unable to learn the desired functions on its neurons. A user can either train a new network with a different structure or add new neurons to hidden layers of the existing network and retrain it.

In a decison tree if a branch misclassifies many test objects that branch can be the result of overfitting to a training set. In QMAK a user can prune such a branch by turning the inner node from which the branch comes out into a leaf node.

In k-nearest neighbors model one can check whether changing the number of neighbors selected to vote or changing the voting method fixes classification of misclassified objects. The number of neighbors and the voting method can be changed without retraining an already trained classifier. Moreover, if many test objects are misclassified by a k-NN classifier because of a small subset of training objects, a user may consider removal of such objects from the training set.

IV. System Extensibility

The platform is designed to make addition of both new classification models and their visualization as simple as possible. It is intended to allow users implement visualization of their classifiers in such way that they do not need to have any knowledge of how a tool presenting visualization, in particular QMAK, is implemented. To achive that, a simple interface with two methods is defined.

The first method implements visualization of the structure of a classification model:

```
void
draw(JPanel canvas)
```

The second method implements visaualization of the process classifying a single test object:

```
void
drawClassify(JPanel canvas, DoubleData obj)
```

In both methods the author of a classifier uses the provided `canvas` object to draw whatever is best to visualize the classifier.

Many visualization tools provide graph-based framework for implementing visualization. But many classification models, for example: k-nearest neighbors, rule-based classifiers, support vector machine, do not fit such framework. In QMAK the author gets a general object-frame in which they can draw any graphical representation of a classifier or classification process. Decision trees and neural networks have graph-based visualization, k-NN and PCA classifiers each presents a certain type of projection of selected objects represented as points onto two-dimensional area, and rough set classifier present text description of the components of the model. Moreover, visualization can be animated like in case of k-nearest neighbors classifier.

After implementation of the two interface methods in the source code of a classifier a user can easily add it to QMAK using menu or by adding an entry in the configuration file. It does not require any change in QMAK itself.

V. Summary

The paper presents QMAK, an open platform for visualization of classification models and classification process.

Two main features that distinguish QMAK from other machine learning visualizers are that it integrates visualization of five different classification models in one tool and that all five models visualize also the very process of classifying a single object. This unique value of QMAK can help data analysts interpret their data and the models built from data in a way that has hitherto been unavailable. The platform can also be an important alternative as a didactic tool for teaching of machine learning.

References

[1] "Neptune: metadata store for machine learning operations," accessed: 2023-05-15. [Online]. Available: https://neptune.ai

[2] "Graphviz: open source graph visualization software," accessed: 2023-05-15. [Online]. Available: https://graphviz.org

[3] "Tensorboard: Tensorflow's visualization toolkit," accessed: 2023-05-15. [Online]. Available: https://www.tensorflow.org/tensorboard

[4] "dtreeviz: python library for decision tree visualization and model interpretation," accessed: 2023-05-15. [Online]. Available: https://github.com/parrt/dtreeviz

[5] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witen, "The weka data mining software: An update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009. doi: 10.1145/1656274.1656278

[6] "Netron: visualizer for neural network, deep learning, and machine learning models," accessed: 2023-05-15. [Online]. Available: https://github.com/lutzroeder/netron

[7] A. LeNail, "Nn-svg: Publication-ready neural network architecture schematics," *Journal of Open Source Software*, vol. 4, no. 33, p. 747, 2019. doi: 10.21105/joss.00747

[8] A. Wojna and R. Latkowski, "Rseslib 3: Library of rough set and machine learning methods with extensible architecture," *LNCS Transactions on Rough Sets XXI*, vol. 10810, pp. 301–323, 2019. doi: 10.1007/978-3-662-58768-3_7

[9] A. Wojna, R. Latkowski, and Ł. Kowalski, *RSESLIB: User Guide*, accessed: 2023-05-15. [Online]. Available: http://rseslib.mimuw.edu.pl/rseslib.pdf

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 2020.

[11] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[12] A. Wojna, "Analogy-based reasoning in classifier construction (phd thesis)," *LNCS Transactions on Rough Sets IV*, vol. 3700, pp. 277–374, 2005. doi: 10.1007/11574798_11

[13] K. I. Diamantaras and S. Y. Kung, *Principal Component Neural Networks: Theory and Applications*. New York: John Wiley & Sons, Inc., 1996.

[14] Z. Pawlak, *Rough Sets - Theoretical Aspects of Reasoning about Data*. Dordrecht: Kluwer Academic Publishers, 1991.

[15] A. Skowron and C. Rauszer, "The discernibility matrices and functions in information systems," in *Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory*, R. Slowinski, Ed. Dordrecht: Kluwer Academic Publishers, 1992, pp. 331–362.