Data and Web Science Group
Prof. Dr. Heiko Paulheim
B6, 26 – B022
68159 Mannheim

Data Mining I

Exercise 4: Classification

General notice: Please use “local seeds” with X-Validation and a value of “1992” to be comparable.

4.1. Learning a classifier for the Iris Data Set – Part II

Last exercise, you have learned lazy classification models for the Iris dataset. Now try a Decision Tree based approach with 10-fold cross-validation.

1. Let’s try the ID3 tree building algorithm first. Build a process that (1) discretizes all attributes of the Iris data set by frequency into three bins. (2) Afterwards, the process should use the *X-Validation* operator (10 folds, stratified sampling) to generate a training and test data set. (3) As inner operator of the x-validation, the process should use the *ID3* operator to learn a decision tree and the *Performance (Classification)* operator to evaluate the accuracy of the learned model.
2. Remove the discretization operator and change the ID3 operator into RapidMiner’s standard *Decision Tree* building operator. Run the process again. Does the accuracy change? Compare the complexity of the two models. Which model should be preferred according to Occam’s razor?

4.2. Who should get a bank credit?

The German credit data set from the UCI data set library (http://archive.ics.uci.edu/ ml/index.html) describes the customers of a bank in respect whether they should get a bank credit or not. The data set is provided as *credit-g.arff* file on the website. You need to use the RapidMiner *ARFF reader* operator to import the data set. Please also have a look at the data set documentation that is included in the file.

1. Apply the *Compare ROCs* Operator to the dataset and include k-NN (different k values), Decision Tree and Naïve Bayes classification. Which classification approach looks most promising to you?
2. Include the most promising classification approaches and try to optimize the results using a 10-fold X-Validation approach. Which level of accuracy do you reach?
3. What does the precision and recall values for the class “bad” customer tell you? Try to improve the situation by increasing the number of “bad” customers in the training set. For doing this, you first filter all bad customers from the data set and then append these customers to the original set. How does precision and recall change if you apply this procedure twice? Use the *Filter Examples* Operator and *Append* the output to the original dataset.
4. To model a use case-specific evaluation, as observed in the previous example, replace the *Performance (Classification)* operator by the *Performance (Costs)* operator. Set up your cost matrix by assuming that you will lose 1 unit if you refuse a credit to a good customer, but

that you lose 100 units if you give a bad customer a credit. Rerun the experiments from 4.2.2 and evaluate the results.

5. As the creation of training data is mostly a manual task and humans tend to be fallible, training data might include noise. Simulate this behavior by using the *Add Noise* operator and change the parameter “label noise” from 0% to 10% to 20%. Is your preferred classification approach still feasible for this situation? How does the performance of the other classifiers evolve?