Data and Web Science Group
Prof. Dr. Heiko Paulheim
B6 – B1.16
68159 Mannheim

Data Mining

Exercise 4: Classification

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. Discretise the Iris data set into three bins. Then use the DecisionTreeClassifier with a 10-fold stratified cross validation and compute the accuracy. Afterwards plot the decision tree.
2. Remove the discretization and adjust the max_depth parameter of DecisionTreeClassifier to increase the accuracy. 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 with respect to whether they should get a bank credit or not. The data set is provided as *credit-g.arff* file in ILIAS.

1. Plot ROC curves for k-NN (different k values), Decision Tree and Naïve Bayes classification (you can use the given avg_roc function) . Which classification approach looks most promising to you?
2. For the two most promising classification approaches, compute the accuracy and confusion matrix in a 10-fold cross-validation setup (use cross_val_predict function). Which level of accuracy do you reach?
3. What do 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 (in the cross-validation!). How do precision and recall change if you apply this procedure?
4. To model a use-case specific evaluation, as observed in the previous example, compute the cost of all misclassifications. 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. Re-run the experiments from 4.2 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* function and change the parameter “percentage” from 0% over 10% to 20%. Is your preferred classification approach still feasible for this situation? How does the performance of the other classifiers evolve?