Introduction to Statistical Computing in Scala - an Implementation of the K-Nearest Neighbors classifier

Roxana Tesileanu, *Research Assistant, National Institute of Forest Research and Management (INCDS), Romania*

***Abstract*—Statistical computing in ecology evolves at a high speed, mainly because researchers have recognized the advantage of being able to design their algorithms according to their needs. The present paper introduces the implementation in Scala of the k-Nearest Neighbors (kNN) classifier based on Euclidean distances, which can be applied also on small datasets, a situation commonly encountered in ecological research.**

***Index Terms*—machine learning in ecology, k-Nearest Neighbors classification, Scala**

I. Introduction

One of the drivers of the machine-learning progress is the great amount of data born within and gathered by networked and mobile computing systems which necessitates further processing in order to gain insights into the specific fields from which it originates (Jordan and Mitchell 2015, p. 256). The "Big Data phenomenon" is real but, unfortunately, it can not be generalized to all research areas, especially some areas of ecological research which investigate systems which are by nature data-poor and will probably remain as such unless cost-intensive data collection projects are being proposed and financed. This by no means implies that such areas cannot take advantage of the progress of machine-learning and take the best out of the existing datasets. On the contrary, machine-learning, being a study field which "sits at the crossroads of computer science, statistics and a variety of other disciplines concerned with automatic improvement over time, and inference and decision-making under uncertainty" (Jordan and Mitchell 2015, p. 256), is welcomed in real-world environmental decision-making facing the need of proposing courses of action under the cloud of uncertainties and regarding issues characterized by multiple attributes (Vatn and Bromley, 1994). Moreover, machine-learning is used not only in analyzing data from observational studies (which might benefit from larger time series datasets), but also from experimental studies (which can't produce large amounts of data but benefit from an experimental design, delivering data which provides stronger inference about effects and causal relationships - Quinn and Keough, 2002) (Harrington 2012). Indeed, a combination of machine-learning classifiers with the benefits of a MANOVA experimental design used for collecting training data, might produce useful research results, in ecological research too. Thus, in order to be able to use the power of machine-learning algorithms adapted to small ecological datasets, a statistical package written in Scala is currently under development at INCDS. The idea of adapting the use of machine-learning algorithms to real-world data requirements is not new. Hastie and Tibshirani (1996), describe the Discriminant Adaptive Nearest Neighbor (DANN) procedure as a locally adaptive form of nearest neighbors classification using a modified linear discriminant analysis (LDA) procedure to estimate an effective metric for computing neighborhoods. The main idea of combining distance-based approaches with approaches based on the association between variables is driven by the fact that a multivariate dataset can be analysed in both ways (Quinn and Keough 2002): calculating scores for the derived variables (components) for each object, and calculating dissimilarity measures (distances) for every possible pair of objects. The present paper introduces the Scala functions used to run the kNN algorithm and then it delineates the role of the classifier within a real-world classification pipeline for reducing error.

II. Running the kNN algorithm

In order to be able to run the kNN algorithm written using the Scala language, one should consider which tools to choose for the different components of a development system - which are according to Rehman and Paul (2003): the hardware platform, the operating system, editors, compilers and assemblers, debuggers, version control system and bug tracking. For implementing Scala projects, working with the Scala Build Tool (SBT) (http://www.scala-sbt.org/index.html) condenses the list of the components of a development system but also requires a default structure of the project directory with the source files of user-defined packages located in the src subdirectory (Suereth and Farwell 2016) following a standard pattern. This is the reason for which the source files of the scalaML package are placed at the end of a chain of subdirectories within the project directory (see appendix A through E).

III. The machine-learning pipeline for reducing error

IV. Conclusion

Appendix A
Source file - src/main/scala/mai/scalaML/BasicVectorOP.scala

.....

Appendix B
Source file - src/main/scala/mai/scalaML/ReadFile.scala

.....

APPENDIX C
SOURCE FILE - SRC/MAIN/SCALA/MAI/SCALAML/KNN.SCALA

.....

APPENDIX D
SOURCE FILE - SRC/MAIN/SCALA/MAI/SCALAML/AUTONORM.SCALA

.....

APPENDIX E
SOURCE FILE - SRC/MAIN/SCALA/MAI/SCALAML/TESTFRAMEWORKCLASSIF.SCALA

.....

ACKNOWLEDGMENT

The author would like to thank Jeff Druce and Mike Reposa from CRA for very useful learning tips and Georgeta Ionescu from INCDS for giving me the financial support needed.

Roxana Tesileanu My bio.