Univ.-Prof. Dr.-Ing. Matthias Boehm
Graz University of Technology
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMK endowed chair for Data Management

1 AMLS SS 2022: Exercise – Wine Quality Prediction

Published: March 10, 2022 (last update: Mar 10)
Deadline: June 17, 2022, 11.59pm

This exercise is an alternative to the AMLS programming projects, and aims to provide practical experience in the exploratory development of machine learning (ML) pipelines. You may use a programming language of your choosing, and utilize existing OSS ML systems and libraries. The expected result is a zip archive named `AMLS_Exercise_<student_ID>.zip`, containing the partial results of the individual sub-tasks, submitted in TeachCenter.

1.1 Data Preparation (15/100 points)

Download and prepare the UCI Wine Quality Dataset[1]. This dataset contains separate CSV files for red and white wine and their quality. Fuse these datasets into a single matrix with an additional feature (0/1) for the type (red or white). Compute meaningful summary statistics for semi-manual data validation (and cleaning if necessary). Then, split the data into train and test datasets, where the train data can be further divided into train and validation sets.

Partial Result: `01_DataPrep.*`

1.2 Modeling (25/100 points)

Read the prepared datasets, and train two predictive models on the train set for (1) predicting the wine quality (regression), and (2) predicting the wine type (classification), as well as evaluate these models—with appropriate evaluation metrics—on the hold-out test set. Extend these ML pipelines by steps for standardization, normalization, and/or outlier removal.

Partial Result: `02_Model.*`

1.3 Tuning (20/100 points)

In order to improve the evaluation metrics of your models on the test set, perform feature engineering (e.g., add non-linear features, remove correlated features, apply binning or other transformations), and tune the hyper-parameters of your models. Evaluate these decisions on the validation set without leaking the test set into the tuning process. Optionally, perform cross validation and data augmentation to further improve the generalization of your models.

Partial Result: `03_Tuning.*`

[1]UCI Machine Learning Repository: `https://archive.ics.uci.edu/ml/datasets/wine+quality`

1.4 Model Debugging (15/100 points)

Systematically evaluate the performance of your trained models by applying basic model debugging techniques. Examples include basic statistics, confusion matrices, and an in-depth analysis of prediction errors (e.g., for strata or slices). Additionally, provide an automated explanation for the made predictions.

Partial Result: `04_Debug.*`

1.5 Parallelization (15/100 points)

Measure the end-to-end runtime of your ML pipeline including model tuning. Then, parallelize parts of this ML pipeline (e.g., task-parallel hyper-parameter tuning) and compare the improved runtime. The source code should be parameterized to allow running your ML pipeline either in the sequential or parallel configurations.

Partial Result: `05_Parallel.*`

1.6 Documentation (10/100 points)

Describe how to run your source code artifacts—of the above tasks—in order to reproduce your results. Furthermore, briefly describe the approach taken, and the obtained results and insights.

Partial Result: `06_README.txt`