Practical Machine Learning: Pipeline Optimization

Milana M. Wolff

December 14, 2023

1 Introduction

In this assignment, we perform pipeline optimization using the wine quality dataset. This widely used dataset contains a variety of physicochemical input features, such as wine density and acidity, along with expert ratings for red Vinho Verde wines. We approach the pipeline optimization problem by selecting a scaler (from four methods or passthrough), whether or not to use a feature selector and OneHot encoding, and by selecting an estimator from Ridge, K-Neighbors, and Decision Tree classifiers, each with nine possible different hyperparameter configurations evaluated via grid search. We perform grid search cross-validation and nested resampling (3 outer folds, 10 inner folds per hyperparameter configuration).

2 Dataset Description

The dataset used for this assignment contains physicochemical quantitative input features and sensory quantitative output features (i.e., an expert wine score) for the red variant of the Portuguese "Vinho Verde" wine. The dataset includes 1599 observations and eleven input features, including fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol content. According to the UC Irvine Machine Learning Repository website, "the classes are ordered and not balanced (e.g. there are many more normal wines than excellent or poor ones)", with a total of 1319 observations rated as 5 or 6 and a mere 28 observations rated with the highest and lowest scores (3 and 8). This robust dataset includes no missing values to be imputed. We use the eleven listed input features to predict the wine quality measurement as the target.

3 Experimental Setup

We use Python 3.10.12 (GCC 11.4.0) in a Jupyter/interactive Python notebook on Google Colaboratory. Using GridSearchCV, we compare pipelines with the scaler set to StandardScaler(), MinMaxScaler(), Normalizer(), MaxAbsScaler(), or None/passthrough, the encoder set to OneHotEncoder() or passthrough, the feature selector set to VarianceThreshold() or passthrough, and the estimator to Ridge, K-Neighbors, or Decision Tree classifiers. For each classifier, we evaluate 9 different hyperparameter configurations using grid search (1, 5, or 10 neighbors and 'ball_tree', 'kd_tree', or brute force algorithms for KNeighbors(); a tolerance of 0.0001, 0.01, or 0.1 and 'svd', 'cholesky', or 'sparse_cg' solvers for the Ridge classifier; and a maximum tree depth of 1, 2, or 5 and None, 'auto', or 'sqrt' for the maximum features hyperparameter for the Decision tree classifier). We computer 3 outer folds and 10 inner folds of nested resampling/cross-validation.

4 Results

The best model selected is KNeighbors, with a balanced accuracy generalization score of 0.373 ± 0.109, no scaler or OneHot encoding applied in preprocessing, and VarianceThreshold() as the feature selector.

5 Notes

I wasn't able to figure out how to extract the hyperparameters for the KNeighbors model selected at the time of this submission. I spent quite a bit of time trying to figure out how to conduct a hyperparameter search within a pipeline– see the considerable number of StackOverflow links in the repository readme file. I tried several different methods, including a PipelineHelper package and multiple ways of conducting a grid search over the space. I'm not sure how thoroughly these efforts appear in the code I submitted.

6 Code

`https://github.com/COSC5557/pipeline-optimization-mwolff2021/tree/main`