Practical Machine Learning: Pipeline Optimization

Milana M. Wolff

May 05, 2024

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 Bayesian optimization. We perform 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.

Scaler	Selector	Estimator
StandardScaler(), MinMaxScaler(), Normalizer(), MaxAbsScaler(), passthrough	VarianceThreshold(), passthrough	SVCClassifier(), RidgeClassifier(), KNeighborsClassifier(), DecisionTreeClassifier(), BaggingClassifier(), RandomForestClassifier()

Classifier	Support Vector Classifier	K Neighbors Classifier	Ridge Classifier	Decision Tree Classifier	Bagging Classifier	Random Forest Classifier
Hyperparameter Search Space	'C' : (1e-6, 1e+6, prior='log-uniform'), 'gamma' : (1e-6, 1e+1, prior='log-uniform'), 'degree': (1, 8), 'kernel' : 'linear', 'poly', 'rbf'	'n_neighbors' : Integer(1, 100, prior = 'log-uniform'), 'algorithm' : 'ball_tree', 'kd_tree', 'brute', 'leaf_size' : (1, 50)	'tol' : (0.01, 0.1, prior = 'log-uniform'), 'solver' : 'svd', 'cholesky', 'sparse_cg', 'sag', 'lsqr', 'alpha' : (0.1, 1.0, prior = 'log-uniform')	'max_depth' : (1, 10, prior = 'log-uniform'), 'max_features' : None, 'auto', 'sqrt', 'log2', 'min_samples_split' : (0.1, 1.0, prior = 'log-uniform')	'n_estimators' : (50, 500, prior = 'log-uniform'), 'max_features' : (0.1, 5, prior = 'log-uniform')	'n_estimators' : (100, 100000, prior = 'log-uniform'), 'criterion' : 'gini', 'entropy', 'log_loss', 'max_depth' : (1, 10, prior = 'log-uniform')

3 Experimental Setup

We use Python version 3.10.9 running on the `teton-gpu` partition of the Beartooth cluster. (Using Python 3.10.12 (GCC 11.4.0) in a Jupyter/interactive Python notebook on Google Colaboratory repeatedly timed out). We use classifiers, preprocessors, and other pipeline elements from the `scikit-learn` package. We use GridSearchCV with 3 outer folds and 10 inner folds for nested resampling/cross-validation over the entire pipeline parameter space, with Bayesian optimiation (implemented via `BayesSearchCV`) applied to search for optimal hyperparameter settings of individual classifiers in the last stage of the pipeline. The pipeline includes a scaler, a selector, and a final estimator component. As the features in the wine quality dataset are real-numbered values, we do not use an encoder on this dataset. The possible components are tabulated below.

The hyperparameter search spaces for each estimator are tabulated below.

We evaluate the results using balanced accuracy, as the wine quality dataset is highly imbalanced.

4 Results

We obtained a training set score of 1.0 over the entire dataset, indicating a high degree of overfitting. We obtain a test set score of 0.3238. The generalization score is 0.363 $\pm$ 0.091. The best pipeline used a MinMaxScaler(), the VarianceThreshold() selector, and a BaggingClassifier() as the estimator. (Note: I wasn't able to figure out how to extract the optimal hyperparameter settings of the BaggingClassifier() used at the time of submission.)

5 Code

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