Pipeline Optimization

COSC5557 – Practical Machine Learning

William Baumchen

12/4/2023

1 - Introduction

Given the cost of fitting machine learning models when computing large-scale or exhaustive optimization and considering the relative necessity of altering portions of the process, optimizing a pipeline – a formalized set of processes that typically import data, conduct preprocessing, conduct hyperparameter optimization, and report a final model – becomes an attractive methodology. In this report, the construction and optimization of a relatively simple pipeline will be conducted using the white wine quality dataset, and the result compared to a model of the same type trained using default hyperparameters.

2 – Dataset Description

The dataset used is the white wine quality dataset, containing 4,898 observations, with 11 features, all of which are real numbers taken from a continuous range. There is one 'target' feature, with seven possible classes. There are no missing values in the observations and target feature.

3 – Experimental Setup

In this exercise all computation was done utilizing MATLAB, more specifically the Statistics and Machine Learning Toolbox. Firstly, the dataset was split twice for nested resampling; a fifth of the original dataset was set aside for outer loop testing, and the remaining 80% of the data was split again, with 90% kept for model training and 10% for model evaluation. The pipeline was constructed in three steps – a feature normalization step, a feature reduction step utilizing principal component analysis, and a hyperparameter optimization step conducted using three different classification machine learning models. This optimization utilized Bayesian optimization to search through the hyperparameter space, which included whether or not to normalize the data, whether or not to conduct feature reduction, and to what degree, and finally what model to use and which hyperparameters to select for that model. The pipeline optimization target and the hyperparameter optimization target was the validation classification loss computed during the optimization.

Normalization in data pre-processing is a term with a number of definitions. Here, the process refers to scaling the data such that it has a mean of zero and standard deviation of one, while retaining the shape properties of the original data set. In this exercise, the normalization of the white wine quality dataset was completed using the normalize function in MATLAB. In this

first step, the pipeline optimization considered whether or not normalizing the dataset was useful.

The second component in the optimization was feature reduction via principal component analysis (PCA). Conducting PCA feature reduction on a dataset results in a transformed set of observations, with features that now correspond in decreasing effective variation in the target feature space of the model. This, more properly defined, is given as an orthogonal linear transformation that transforms the given data to a new coordinate system, such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on [1]. Here, PCA was conducted using the pca function in MATLAB. In the second component of the pipeline the pipeline optimization optimized for the number of transformed features to be removed from the dataset used for the machine learning training, ranging from eleven features used to only one feature. In this model if the features retained was zero, that meant that no feature reduction was to be carried out.

The third component was the selection of the machine learning model and the hyperparameters for that model. This pipeline considered three different classification machine learning models. These models were a binary decision tree classifier, an ensemble classifier, and a k nearest neighbors classifier. For the binary decision tree, the optimization considered the minimum leaf size. The range of parameters used in the hyperparameter optimization was based on the default options set by the fitctree function in MATLAB and reported in Table 1.

Table 1 – Decision Tree Hyperparameter Ranges

Hyperparameter Optimization Ranges – Decision Tree	
MaxNumSplits	Log-scaled integers in [1,max(2,NumObservations-1)]

For the ensemble classifier, the optimization considered the method used by the classifier. The range of parameters used in the hyperparameter optimization was based on the default options set by the fitcensemble function in MATLAB, and reported in Table 2.

Table 2 – Ensemble Hyperparameter Ranges

Hyperparameter Optimization Ranges – Support Vector Machine	
Method	Searches between 'Bag','AdaBoostM2','RUSBoost'

For the k-nearest neighbor model, the optimization considered the distance metric, choosing among a number of predefined metrics, the number of nearest neighbors in the dataset to find for classifying each point when predicting, and whether or not to standardize the training data. The range of parameters used in the hyperparameter optimization was based on the default options set by the fitcknn function in MATLAB, and reported in Table 3.

Table 3 – KNN Hyperparameter Ranges

Hyperparameter Optimization Ranges – K-Nearest Neighbor	
Distance	Searches among 'cityblock', 'chebychev', 'correlation', 'cosine', 'euclidean', 'hamming', 'jaccard', 'mahalanobis', 'minkowski', 'seuclidean', and 'spearman'
NumNeighbors	Searches among positive integer values, by default log-scaled in the range [1, max(2,round(NumObservations/2))]
Standardize	Searches among ‘true’ and ‘false’

For each of models used, k-fold cross validation with 5 folds was used to avoid overfitting and bias. These folds were repartitioned with every individual model fitting. The Bayesian optimization was given a budget of 125 iterations, to allow for a more exhaustive search of the variable space. The acquisition function chosen for the Beyesian optimization was ‘expected-improvement-plus’, which evaluates the expected amount of improvement in the objective function, ignoring values that cause an increase in the objective [2]. In addition, when the function believes that it is overexploiting an area, it will modify its behavior. The Bayesian optimization minimized the average cross-validation classification loss of the machine learning model in question, here defined as the MATLAB cross-validation classification error [3]. In addition, the time elapsed for each optimization was also recorded, but was not part of the optimization.

Once the pipeline optimization was complete, a final evaluation of the model’s validation loss was carried out on the outer set of testing data, and a final comparison to a baseline model was given.

4 – Results

The pipeline optimization algorithm above was run according to the given conditions. Fig. 1 below shows the entirety of the optimization results, where the results are the MATLAB classification loss function for each of the optimization evaluation points. In this exercise, the pipeline arrangement with the best performance was a non-normalized, classification ensemble model with no feature reduction, and utilizing the ‘Bag’ Method.

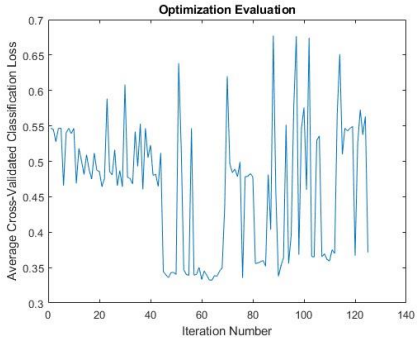

Figure 1 – Observed Validation Loss

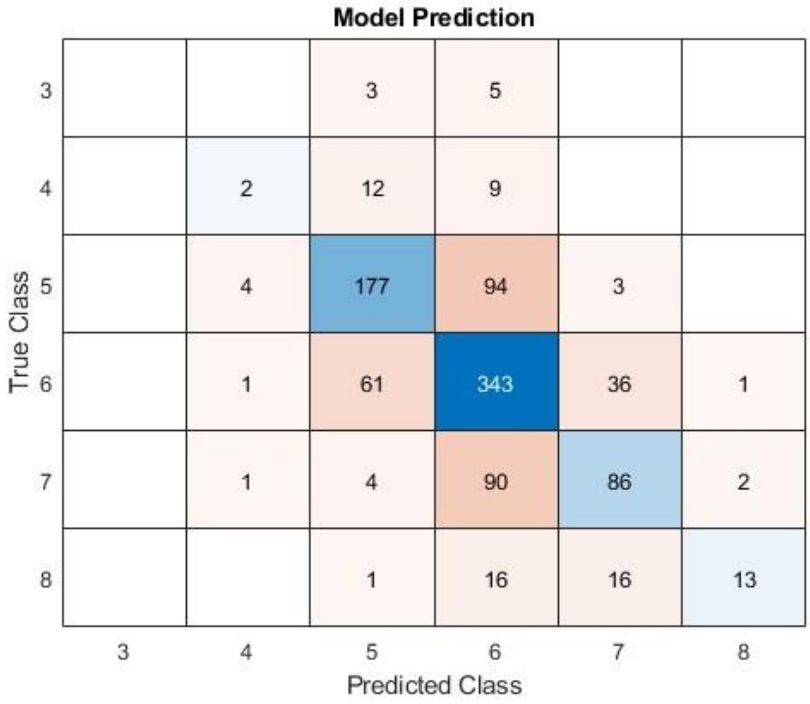

Figure 2 – Model Prediction

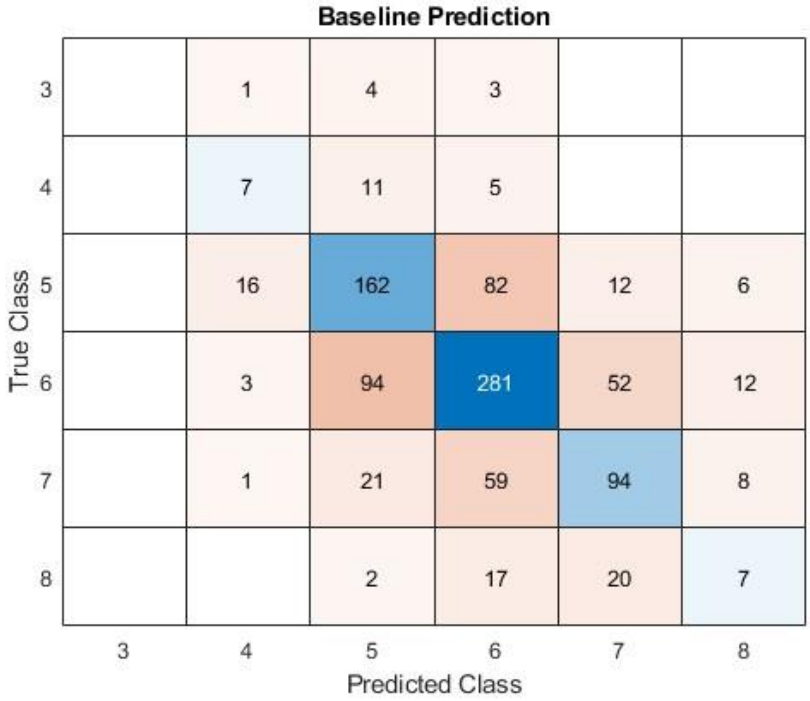

Figure 3 – Baseline Prediction

Fig. 2 shows the confusion plot of the optimized pipeline model's predictions for the outer test data, while Fig. 3 shows the same for the baseline model. As expected, the optimized pipeline

performs better than the nonoptimized model. The optimized ensemble had a minimum observed evaluation average cross-validation loss of 0.33215, and an average cross-validated loss on the test data of 0.35807. This is in comparison with the baseline, which had an average cross-validation loss of 0.47448 and an average cross-validated loss on the test data of 0.43212.

Investigation of the evaluation trace of the optimization revealed that out of the sorted values, the first 24 had the same pipeline hyperparameters as the chosen model. The next best results came from the same model and model-specific hyperparameters, but with variation in the utilization of the normalization and feature reduction steps. The next best model was the decision tree, which had a worse performance in comparison. As such, it's clear from the results of this evaluation that the best performing model was the ensemble model, with the 'Bag' Method, and with no normalization or feature reduction.

References

[1] Jolliffe, I. T. *Principal Component Analysis*. 2nd ed., Springer, 2002.

[2] The MathWorks, Inc. (n.d.). Bayesian Optimization Algorithm. MATLAB Bayesian Optimization Algorithm Explanation - MATLAB. https://www.mathworks.com/help/stats/classificationknn.loss.html#btar64m-2_head

[3] The MathWorks, Inc. (n.d.). Classification Loss. Loss of k-nearest neighbor classifier - MATLAB. https://www.mathworks.com/help/stats/bayesian-optimization-algorithm.html