COSC 5010-03 Practical Machine Learning Fall 2023 Pipeline Optimization Report

Michael Elgin

November 22, 2023

1 Introduction

Parts of a machine learning pipeline include not only the choice of model (algorithm) and the task of tuning the associated hyperparameters, but various choices about how to preprocess the data. This report explores the effect of normalization, feature reduction, and sample reduction (outlier removal) on the wine quality dataset[1]. The effect of these preprocessing changes is examined on a linear regression model for regression, and a decision tree for classification.

2 Dataset Description

The wine quality dataset is standard tabular data. There are 2 datasets for each color of red and white. Both contain 11 features, all of which are continuous. The target is a discrete value which is the assigned quality of the wine. For the regression tasks, the white wine dataset is used to evaluate models. For classification tasks, both datasets are concatenated, with the target being changed to be the color of the wine, with 0 being assigned to red and 1 to white.

3 Experimental Setup

All computation is done with the Python programming language. Scikit-learn is used to construct models. Pandas is used to load and preprocess the data.

For regression, performance is the mean absolute percentage error. This is defined by taking the difference between the predicted value of the model and the actual value, then dividing by the actual value. Then the mean of all of those is taken. More formally:

$$GE_{Regression\ model}(\hat{f}, \boldsymbol{X}_{test}, \boldsymbol{y}_{test}) = \frac{100}{|\boldsymbol{X}_{test}|} \sum_{i=1}^{|\boldsymbol{X}_{test}|} \left| \frac{\hat{f}(\boldsymbol{X}_{test,i}) - \boldsymbol{y}_{test,i}}{\boldsymbol{y}_{test,i}} \right|$$

[1] https://archive.ics.uci.edu/dataset/186/wine+quality

For classification, the performance metric is standard accuracy:

$$GE_{Classification\ model}(\hat{f}, \boldsymbol{X}_{test}, \boldsymbol{y}_{test}) = \frac{\sum_{i=1}^{|\boldsymbol{X}_{test}|} l_{0,1}(\hat{f}(\boldsymbol{X}_{test,i}), \boldsymbol{y}_{test,i})}{|\boldsymbol{X}_{test}|}$$

$$Acc_{Classification\ model} = (1 - GE_{Classification\ model}) * 100$$

For the linear model, no hyperparameters are tune. For the decision tree, grid-search is used for trying hyperparameter configurations. The grid is exponential in fashion, meaning exponents for 2^x are tried. This allows for a much wider exploration of the hyperparameter space given realistic time constraints, since adjusting hyperparameters in a linear fashion would space all configurations too closely together.

The grid search uses nested resampling. The outer loop (for the unbiased performance estimate) uses 3 folds, the inner loop (for the grid search itself) uses 5 folds.

For the decision tree, the hyperparameters considered are max depth for the tree and minimum samples required for a split. All hyperparameters explored can only have positive numbers. The minimum amount of samples must be 2, hence the first exponent starts at 1.

$$\text{max depth} = 2^x, x \in [0, 7]$$

$$\text{min samples} = 2^x, x \in [1, 15]$$

The first pipeline optimization option is whether or not to normalize the data. This consists of z-scoring the data according to the formula

$$z = \frac{(X - \mu)}{\sigma}$$

Regression will normalize the target (quality) as well, but classification will not.

The second pipeline optimization option is to reduce the amount of features in the data. This step will remove all features except for the top 5 features that are most strongly positively or negatively correlated with the target.

The third pipeline optimization option is to reduce the amount of samples in the data by removing outliers. Outliers are determined by whether or not they are outside of the range

$$[Q1 - 1.5 \cdot IQR, Q3 + 1.5 \cdot IQR]$$

Where Q1 is the first quartile, Q3 is the third quartile, and IQR is the interquartile range. For a sample, if any value in any of the feature columns is out of range, the sample will be removed. The target is not a column considered.

For both models, the settings for the pipeline are optimized together (not independently) along with the hyperparamters in the case of the decision tree.

4 Results

4.1 Regression

Table 1: Regression model pipeline optimization effects

Normalize	Feature Reduction	Sample Reduction	MAPE
			11.566
		✓	11.668
	✓		11.907
	✓	✓	11.930
✓			122.117
✓		✓	123.706
✓	✓		126.153
✓	✓	✓	131.229

From table 1 it can be seen that all pipeline options made performance worse. In the case of feature reduction and sample reduction, trivially so. In the case of Normalization, performance was made drastically worse.

4.2 Classification

Table 2: Classification model pipeline optimization effects

Normalize	Feature Reduction	Sample Reduction	Max Depth	Min Samples Split	Acc
			32	2	0.984
		✓	8	2	0.987
	✓		8	4	0.980
	✓	✓	4	2	0.983
✓			32	2	0.984
✓		✓	8	2	0.988
✓	✓		8	4	0.980
✓	✓	✓	4	2	0.983

In the case of classification, pipeline options had merit. Sample reduction allowed for better performance from the decision tree. Feature reduction was harmful. Normalization had little to no effect.

The main takeaways from these results is that firstly pipeline optimization has the *potential* to improve performance, but likewise is it very easy to cause harm with it. Second, when all parts are optimized together, there is a tendency for changes in the pipeline optimization to affect model hyperparameters as well. For example, when features and samples were reduced in the data for the decision tree, the tree was better off with the more strict depth limits of 8 and 4, but with the full features and samples 8 or 4 would likely be too restrictive, hence the max depth of 32. Third, the value of pipeline optimization is, in principle, heavily

dependent on whether the situation is a regression or classification as well as the model class itself.

5 Code

The associated code is in pipe.ipynb