COSC 4010 Practical Machine Learning Fall 2023
Pipeline Optimization Report

Derek Walton

November 2023

1 Introduction

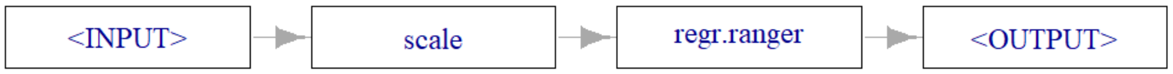
This exercise involves creating a machine-learning pipeline, with the objective of creating a somewhat automated process to preprocess data, tune learner/pipeline hyperparameters, and train a learner to make predictions on a data set. This report will detail the effect of scaling data within the white wine dataset. In addition to tuning hyperparameters utilizing Bayesian optimization and nested cross-validation. The effects of preprocessing the data and tuning the hyperparameters will be examined on a Randon Forest used for regression.

2 Data Description

The wine quality dataset consists of tabular data. There are two datasets included one for white wine and the other for red, both datasets consist of wine samples from northern Portugal. The datasets consist of 11 features, these features are continuous values for various wine characteristics. The target data are discrete values used to describe the quality of the wine. This repository makes use of the white wine dataset which consists of 4898 observations. The data was divided into a training and test split, 67% of the data was used for training and the remaining 33% for testing.

3.1 Experiment Setup

This exercise was completed using the R programming language and a number of packages including mlr3, mlr3benchmark, mlr3learners, mlr3mbo, mlr3tuning, mlr3tuningspaces, and mlr3viz (dependencies for these packages have been omitted for the sake of brevity). As stated in the introduction the learner used for this exercise is the regression Random Forest from the mlr3package. Random Forest was selected for this exercise as it was the best-performing learner from the hyperparameter optimization exercise previously completed. The choice to model this exercise as a regression task remains the same as previous exercises being that target values are numeric, and feature data consists of continuous values. Evaluation of learner performance was done using Mean Squared Error *"regr.mse"* "Measure to compare true observed response with predicted response in regression tasks". [Bischl, 2023] The search space for the learner's hyperparameters was selected from the recommended mlr3 search spaces, in this case, "regr.ranger.default", though there were some slight modifications that were made that will be discussed in the next section. The tuning algorithm used for this exercise was Bayesian optimization, in the form of mlr3mbo's *"mbo"*. This algorithm was selected due to the way that it selects hyperparameter candidates, it invokes more confidence as its selection of hyperparameters is "decided" instead of randomly selected. There is only a single preprocessing step that was used in this exercise, as many of the other options did not seem to be applicable to the dataset. This step utilizes *scale* from the mlr3pipeline, this preprocessing step “Centers all numeric features to mean = 0 (if center parameter is TRUE) and scales them by dividing them by their root-mean-square (if scale parameter is TRUE).”. [Bischl, 2023] Nested resampling was done using *auto_tuner* in the mlr3 package(automated hyperparameter tuning), using a resampling method of 5 cross-validation folds. In addition to 3 cross-validation folds used as an argument to the *benchmark_gird* function in mlr3benchmark. The termination condition for hyperparameter optimization in this exercise was *run_time* prior attempts to allow the pipeline to execute until completion, which resulted in wait times of up to 12 hours without finishing the learner training process. Because of this run time was used to limit the amount of time needed to finish training the learner while still tuning the hyperparameters within a given time constraint. Finally, two unoptimized learners were included in the benchmark_grid function in order to evaluate the benefits of the pipeline these included a random forest regressor and a featureless regressor.



The figure above provides a visualization of the pipeline used in this exercise.

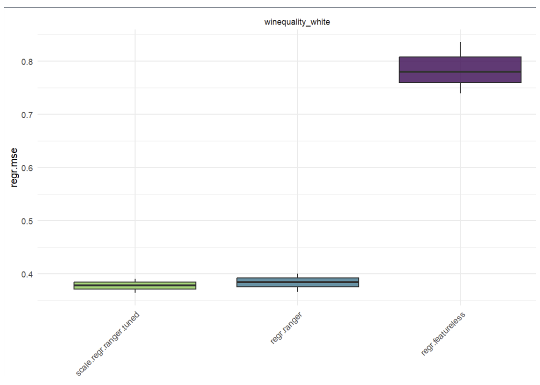
3.2 Hyperparameter Ranges

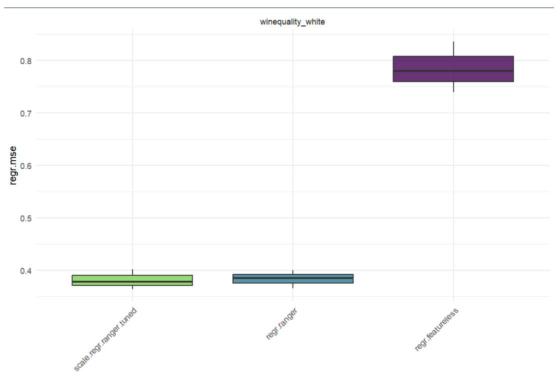
This section will discuss the hyperparameters that were tuned for each learner used in this exercise in addition to the ranges used for tuning.

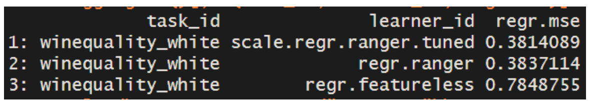
Learner	**Hyperparameter**	**Description and Range Used**
Random Forest	sample.fraction	● Specifies the fraction of observations to be used in each tree. Smaller fractions lead to greater diversity, and thus less correlated trees which often is desirable. [Lovelace, 2020] ● For this exercise, the search space utilized a range from 0.1-1.
	num.trees	● The number of trees in the forest. ● For this exercise, the search space utilized a range from 1-2000.
	mtry.ratio	● The parameter determines the ratio of variables to be sampled as candidates at each split when constructing each tree in the random forest model. [OpenAI, 2023] ● For this exercise, the search space utilized a range from 0-1.
	min.node.size	● The minimum number of samples (observations) required to create a terminal node (leaf) in each tree of the random

Learner	Hyperparameter	Description and Range Used
		forest. [OpenAI, 2023] ● For this exercise, the search space utilized a range from 1-20.

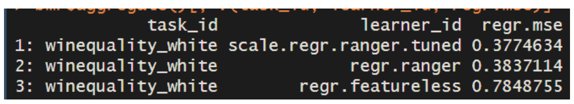
4 Results





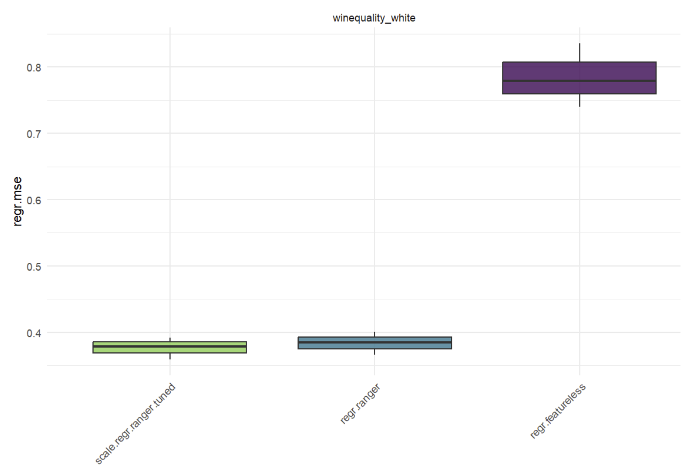


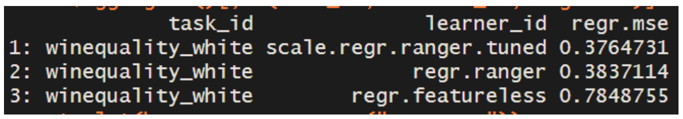
```
             task_id              learner_id  regr.mse
1: winequality_white scale.regr.ranger.tuned 0.3814089
2: winequality_white             regr.ranger 0.3837114
3: winequality_white        regr.featureless 0.7848755
```



```
             task_id              learner_id  regr.mse
1: winequality_white scale.regr.ranger.tuned 0.3774634
2: winequality_white             regr.ranger 0.3837114
3: winequality_white        regr.featureless 0.7848755
```

The figure on the left provides training results for the learner when provided 5 minutes per benchmark cross-validation fold, and the one on the right provides training results for the learner when provided 10 minutes per cross-validation fold. Clearly both the random forest with hyperparameter tuning and without have similar results. This should come as no surprise since both learners used very similar hyperparameter search spaces. However, it is clear to see that both learners performed significantly better than their featureless counterpart, which is also to be expected. The one note to this is that there appears to be marginally better results for the learner that was optimized, and that is allowed time the degree of improvement increases. For the purpose of this exercise, the pipeline was limited via runtime but it is possible that longer runtimes could have resulted in more significant performance improvement for learner prediction accuracy.





```
           task_id               learner_id  regr.mse
1: winequality_white scale.regr.ranger.tuned 0.3764731
2: winequality_white             regr.ranger 0.3837114
3: winequality_white        regr.featureless 0.7848755
```

On the note of longer times allotted for hyperparameter optimization, the figure above provides the results for a learner that was given 30 minutes, it can be seen that there is a marginal improvement compared to its 10-minute counterpart, it may be the case that any further time allotted will provide diminishing returns in learner prediction accuracy.

References

Bischl, B., Sonabend, R., Kotthoff, L., & Lang, M. (2023). R squared - MLR_MEASURES_REGR.RSQ. - mlr_measures_regr.rsq • mlr3. https://mlr3.mlr-org.com/reference/mlr_measures_regr.rsq.html

Bischl, B., Sonabend, R., Kotthoff, L., & Lang, M. (Eds.). (2024).

"Applied Machine Learning Using mlr3 in R". CRC Press. https://mlr3book.mlr-org.com

Bischl, B., Sonabend, R., Kotthoff, L., & Lang, M. (2023). Center and Scale Numeric Features • mlr3. https://mlr3pipelines.mlr-org.com/reference/mlr_pipeops_scale.html

Lovelace, R., Muenchow, J., &amp; Nowosad, J. (2020). 15 Ecology. In Geocomputation with R. essay, CRC press.

ChatGPT (version 2), an AI language model developed by OpenAI