COSC 4010 Practical Machine Learning Fall 2023
Pipeline Optimization Report

Derek Walton

November 2023

1 Introduction

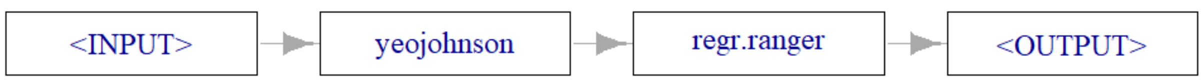
This exercise involves creating a machine-learning pipeline, with the objective of creating a somewhat automated process to preprocess data, tune learner/pipeline hyperparameters, and train a learner to make predictions on a data set. This report will detail the effect of scaling data within the white 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). This exercise utilizes a random forest regression for the mrl3learners package, this model was used as it performed the best in previous exercises. The hyperparameter search space within the pipeline is similar to the default search space with some alterations that are discussed in the next section. The hyperparameter optimization algorithm that was used for the pipeline in this exercise utilizes a Bayesian optimization algorithm in combination with nested cross-validation on the dataset. In addition, run time is the limiting factor within this implementation in order to make program runtimes reasonable. The nested cross-validation consists of 3 outer folds that are implemented using the benchmark_grid function from mlr3, and 5 inner folds that were implemented using the auto_tuner function from mlr3. The run-time limitation was enforced on the inner folds if for example each inner fold is allowed 3 minutes to optimize the learner's hyperparameters the overall runtime of the program would be about 9 minutes. The choice for the number of outer and inner folds was mostly arbitrary, 5 inner folds were selected as this seems to be a common value used within machine learning pipelines, and 3 outer folds were selected to keep run time to a minimum while allowing the pipeline 3 opportunities to produce an accurate learner. The runtimes used in this exercise for the inner folds included 5, 10, and 30 minutes the results of these are discussed in a later section. Leaner prediction accuracy was evaluated using Mean Squared Error regr.mse" "Measure to compare true observed response with predicted response in regression tasks". [Bischl, 2023] There is a single preprocessing step within the pipeline this being a Yeo-Johnson transformation "a statistical method used to stabilize variance and make data more closely follow a normal or Gaussian distribution.". [OpenAI, 2023] In order to determine if this preprocessing step would be beneficial the data for each feature was checked using the skewness function to determine how far it was skewed from normal distribution. All features appeared to be fairly skewed thus, the ranges for the Yeo-Johnson transformation were determined by setting the lower bounds at 5% below the minimum values for each feature and the upper bounds at 5% above the maximum values for each respective feature. Thus the lower bound was 0 - 8.55, and the upper bound was 0.3633-303.45. That was the idea at least, however, this range resulted in missing value errors, so instead the lower bound was set to 0-10 and the upper bound to 10-30. Two untuned learners were used for a comparison of learner performance 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

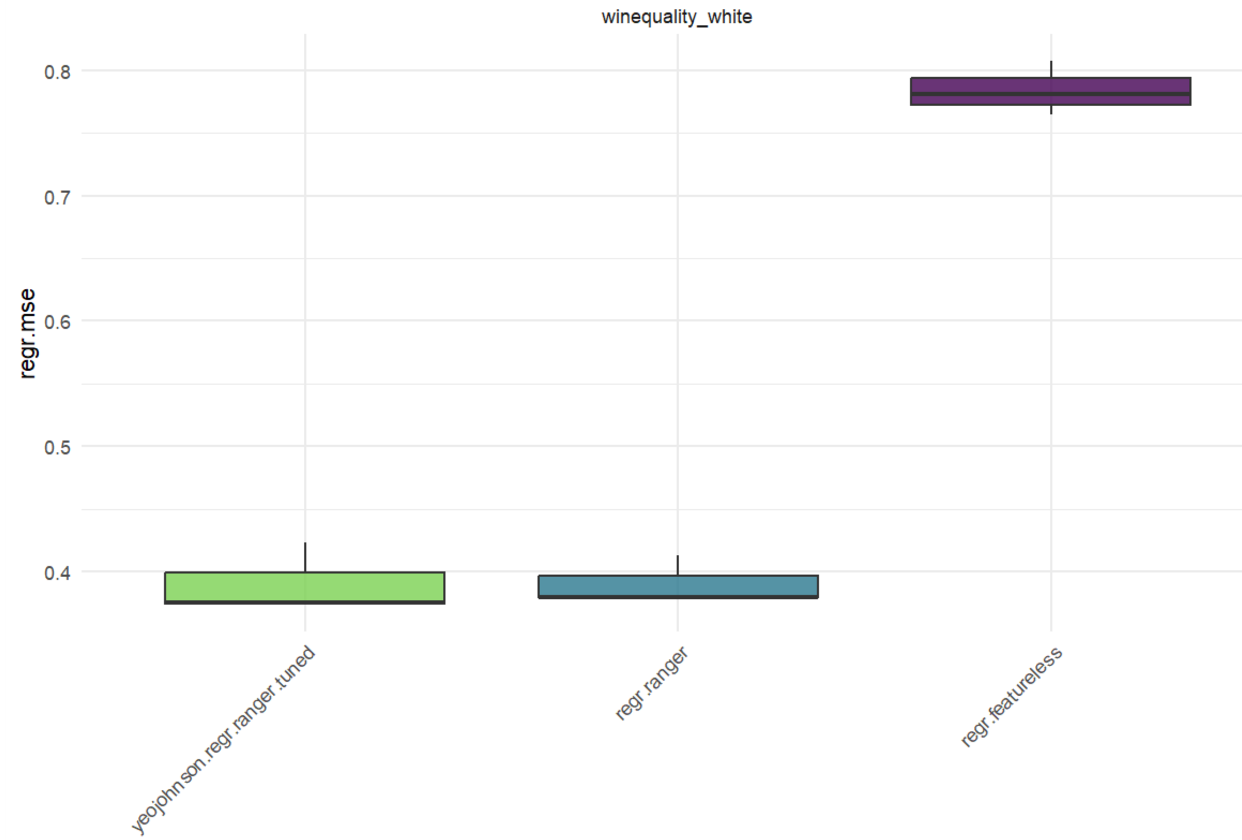
This section will discuss the hyperparameters that were tuned for each learner used in this exercise in addition to the ranges used for tuning.

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.

Learner	Hyperparameter	Description and Range Used
	min.node.size	● The minimum number of samples (observations) required to create a terminal node (leaf) in each tree of the random forest. [OpenAI, 2023] ● For this exercise, the search space utilized a range from 1-20.

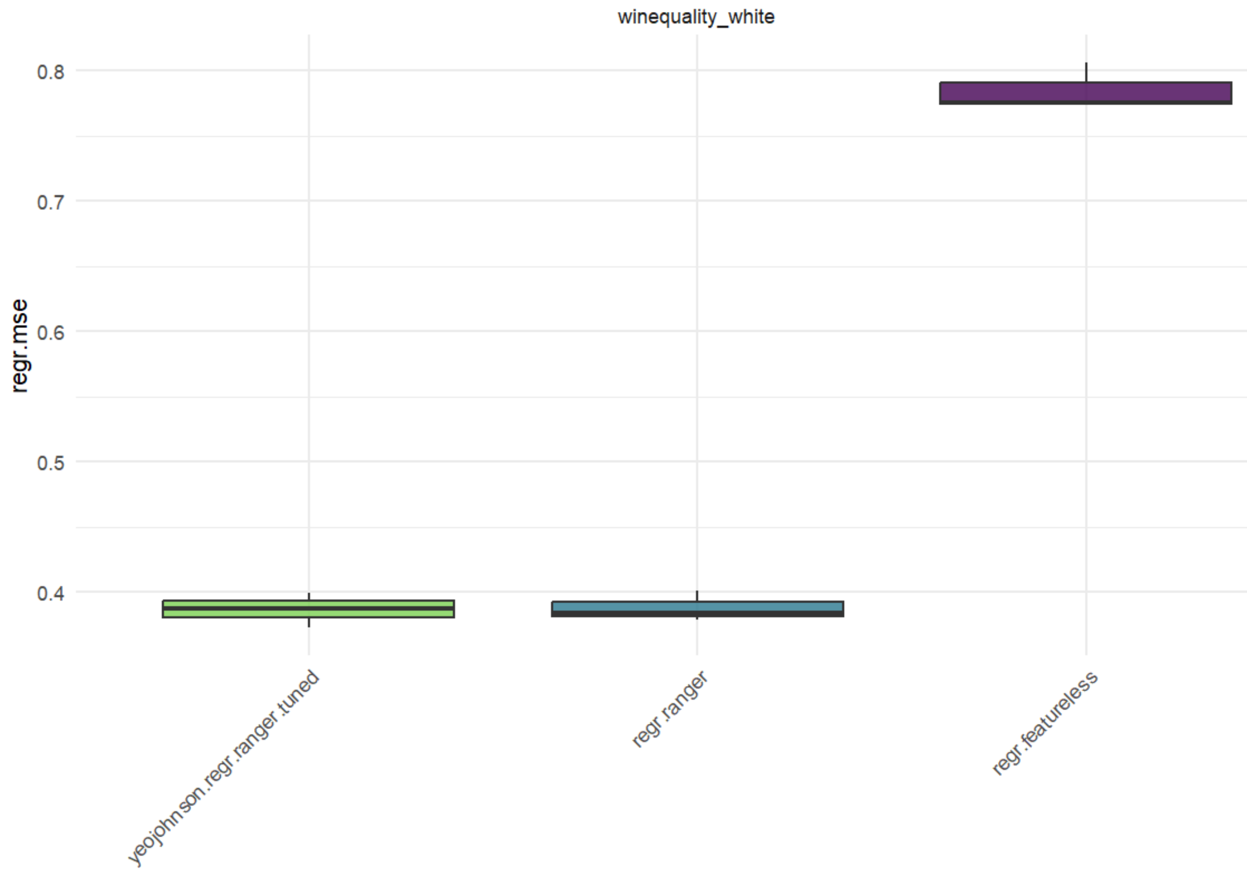
4 Results

The figure below provides learner prediction accuracy for pipelines that were allowed 3 different run_time allowances. Figure 1 is a pipeline given 3 minutes to run per inner cross-validation fold, Figure 2 is a pipeline given 10 minutes per inner cross-validation fold, and Figure 3 30 is a pipeline given 30 minutes per inner cross-validation fold. It can be seen that as tuning time is increased the prediction accuracy of the pipeline increases. Initially, in Figure 1 and even Figure 2 the difference between prediction accuracy for the pipeline and a random forest regressor with default settings is negligible. However, in Figure 3 we begin to see a more meaningful difference in learner prediction performance. It is likely the case that if the pipeline was given more time to train it could produce even better prediction, however, for the purpose of this exercise this will not be done. Figure 4 shows the results of benchmarking several tuned learners within the pipeline. The learners included for this comparison include SVM regressor, KNN regressor, and XGBoost regressor, due to time concerns training time was limited to 5 minutes per inner fold. As expected the best performing learner was the random, however, some what unexpected this particular run resulted in the default random forest regressor having marginally better prediction accuracy. Although the difference in negligible.

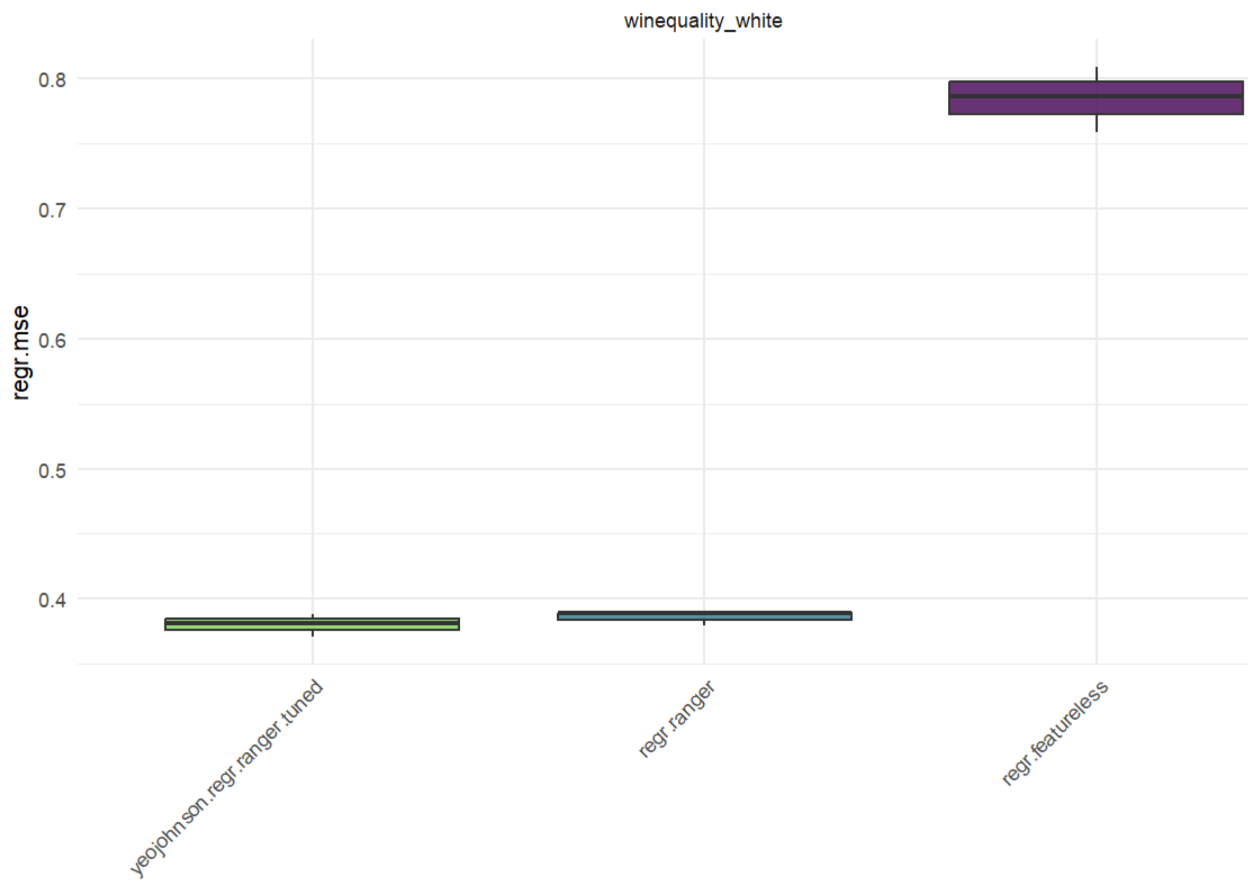



```
              task_id                   learner_id  regr.mse
1: winequality_white yeojohnson.regr.ranger.tuned 0.3905393
2: winequality_white                  regr.ranger 0.3906961
3: winequality_white            regr.featureless 0.7842073
```

Figure 1

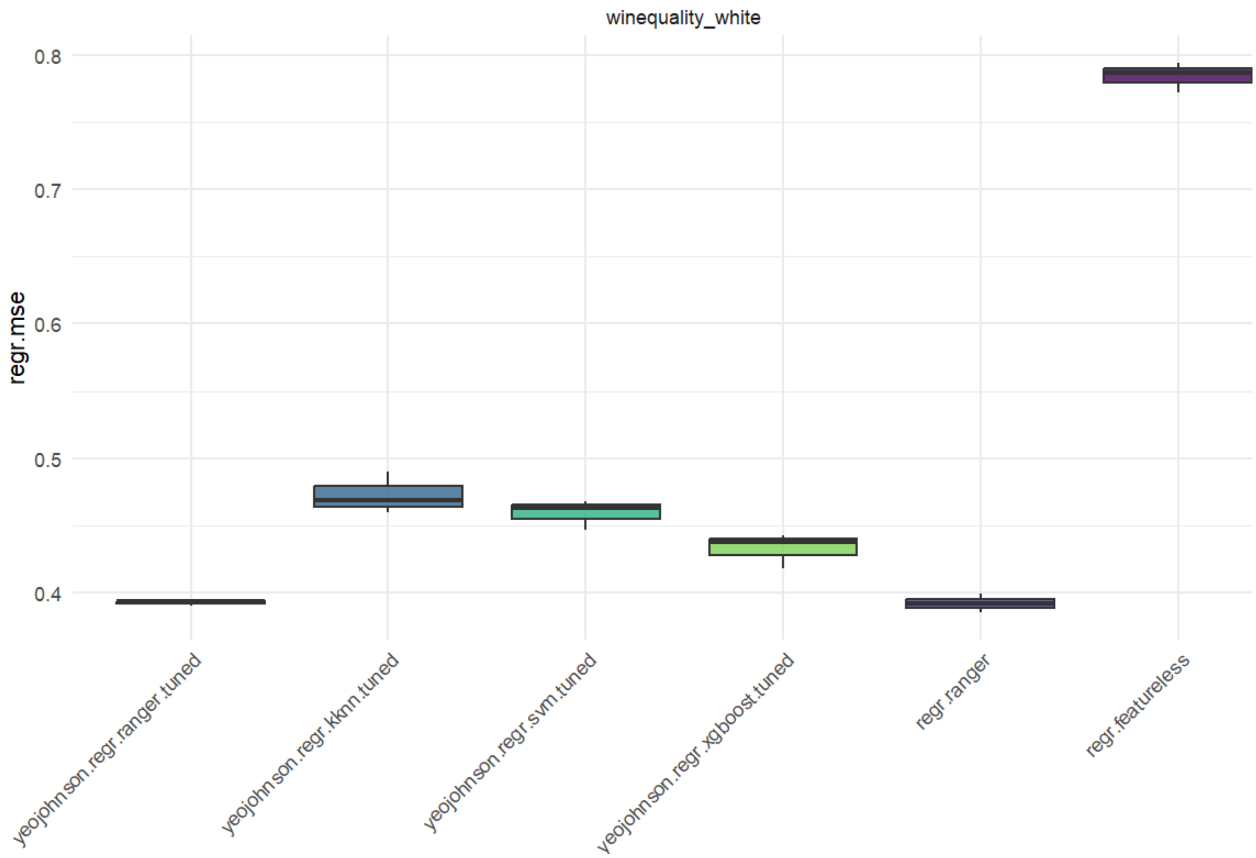



```
                task_id                    learner_id  regr.mse
1: winequality_white yeojohnson.regr.ranger.tuned 0.3866025
2: winequality_white                  regr.ranger 0.3879021
3: winequality_white             regr.featureless 0.7851038
```
Figure 2



	task_id	learner_id	regr.mse
1:	winequality_white	yeojohnson.regr.ranger.tuned	0.3804449
2:	winequality_white	regr.ranger	0.3858057
3:	winequality_white	regr.featureless	0.7843299

Figure 3




```
             task_id                    learner_id  regr.mse
1: winequality_white  yeojohnson.regr.ranger.tuned 0.3926178
2: winequality_white    yeojohnson.regr.kknn.tuned 0.4727279
3: winequality_white     yeojohnson.regr.svm.tuned 0.4591358
4: winequality_white yeojohnson.regr.xgboost.tuned 0.4328008
5: winequality_white                   regr.ranger 0.3918292
6: winequality_white              regr.featureless 0.7843576
```

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

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