Pipeline Optimization

Introduction:

In this assignment, I am going to optimize several ML models using Pipeline Optimization integrated with Bayesian optimization to find the best and optimum model for wine quality dataset. I am going to select several ML algorithms, apply the pipeline optimization on them, optimize them for their hyperparameters, and use them to predict the results. This configuration is supposed to be highly complicated. By comparing the results, we will find out which ML model shows better performance as well as providing insight for pipeline optimization process. Here, I used the regression learners including k-Nearest Neighbors (hyperparameters: k and distance), Random Forest (hyperparameters: num.trees, min.node.size, and mtry), Support Vector Machine (hyperparameters: cost, epsilon, and gamma), and Gradient Boosting (hyperparameters: nrounds and max_depth). Also, I applied scaling which scales features and factor encoding to each one of the learners. Root Mean Square Error was used as objective function and calculated for each iteration to find the model with minimum RMSE.

Dataset Description:

I am using the data from white wine which has 12 features and 4898 observations. Features include fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality. I used all of the mentioned attributes to predict wine quality which has a numeric scale between 0 to 10. The dataset has no missing value. A summary of the dataset is presented in table 1.

Table 1. statistical summary of the white wine dataset

Feature	Min	Mean	Max	SD
Fixed acidity	3.80	6.85	14.20	0.84
Volatile acidity	0.08	0.27	1.10	0.10
Citric acid	0.00	0.33	1.66	0.12
Residual sugar	0.60	6.39	65.80	5.07
Chlorides	0.01	0.04	0.34	0.02
Free sulfur dioxide	2.00	35.31	289.0	17.00
Total sulfur dioxide	9.00	138.4	440.0	42.50
Density	0.98	0.99	1.04	0.003
pH	2.72	3.19	3.82	0.15
Sulphates	0.22	0.49	1.08	0.11
Alcohol	8.00	10.51	14.20	1.23
Quality	3	5.87	9	0.88

Experimental Setup:

I used R programming language and the libraries such as mlr3 (to implement ML tasks), mlr3pipelines (to set up the pipeline), mlr3learners (to train the models), mlr3tuning (to tune the hyperparameters), mlr3mbo (to implement Bayesian Optimization), mlr3viz (to visualize the results), readr (to read the data), caret (to split the data), and dplyr (to manage the data). I decided to fit the regression models to predict the wine quality, therefore, regression learners including k-Nearest Neighbors ("regr.kknn"), Random Forest ("regr.ranger"), Support Vector Machine ("regr.svm"), and Gradient Boosting ("regr.xgboost") were used in this exercise. I applied the scaling and factor encoding to each one of learners and made a graph for them (Figure 1). Furthermore, I tuned some hyperparameters of the preprocessing steps including robust scaling (true or false) and method ("one-hot", "treatment", "sum") for encoding.

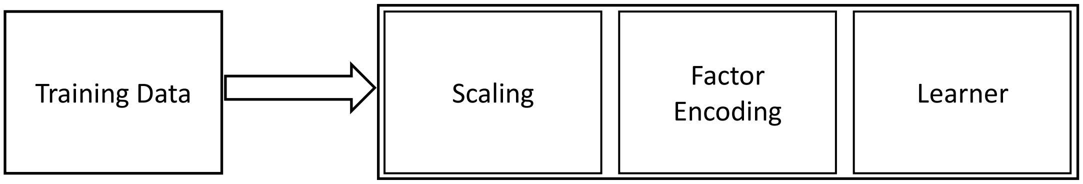

Figure 1. Schematic structure of pipeline optimization

The hyperparameters for each ML model and related value ranges which were selected to be tuned are presented in table 2. The Bayesian optimization approach was used in combination with pipeline optimization to minimize the objective function (which is RMSE).

Table 2. The hyperparameters and their related value range for ML models

Model	Hyperparameter	Range
k-Nearest Neighbors	k	1-20
	distance	1-20
Random Forest	num.trees	100-1000
	min.node.size	1-20
	mtry	1-10
Support Vector Machine	cost	0.1-10
	epsilon	0.01-2
	gamma	0.01-2
Gradient Boosting	nrounds	10-300
	max_depth	1-20

To tackle this problem and select the best ML algorithm after optimization process, I used the nested resampling by dividing the data into two sets (training and validation). I applied 5-fold cross validation to the training dataset and ran the experiment with 100 iterations for each model.

The hyperparameters for each model and for each preprocessing steps were optimized. At each iteration, the RMSE value was calculated and the settings with the lower RMSE value were selected. Afterward, I predicted the result for the validation set using optimized setting to calculate the RMSE. Thus, we have RMSE for optimized model (training using 5-fold CV) and RMSE for validation to compare the models together, considering that the validation set prevents bias and overfitting.

Results:

The best configurations for the preprocessing steps and tuned hyperparameters for each ML model are presented in table 3.

Table 3. The best configurations of hyperparameters for ML models and preprocessing steps

Model	Hyperparameter	Optimum value	Scale	Method
k-Nearest Neighbors	k distance	12 1.0	robust True	treatment
Random Forest	num.trees min.node.size mtry	995 1 2	robust False	sum
Support Vector Machine	cost epsilon gamma	1.67 0.08 0.40	robust True	sum
Gradient Boosting	nrounds max_depth	77 7	robust True	sum

The results of the optimization for training dataset (5-fold CV) without tuning preprocessing steps (figure 2) and with tuning them (figure 3) were presented. These plots illustrated the RMSE for each iteration and model. We can conclude that the Random Forest model is showing better performance and Gradient Boosting is the second best based on these figures. In addition, figure 2 demonstrates more fluctuation in comparison to the plot of hyperparameter optimization. Also, we can see improvement in the results of SVM method after roughly 50 iterations in figure 3.

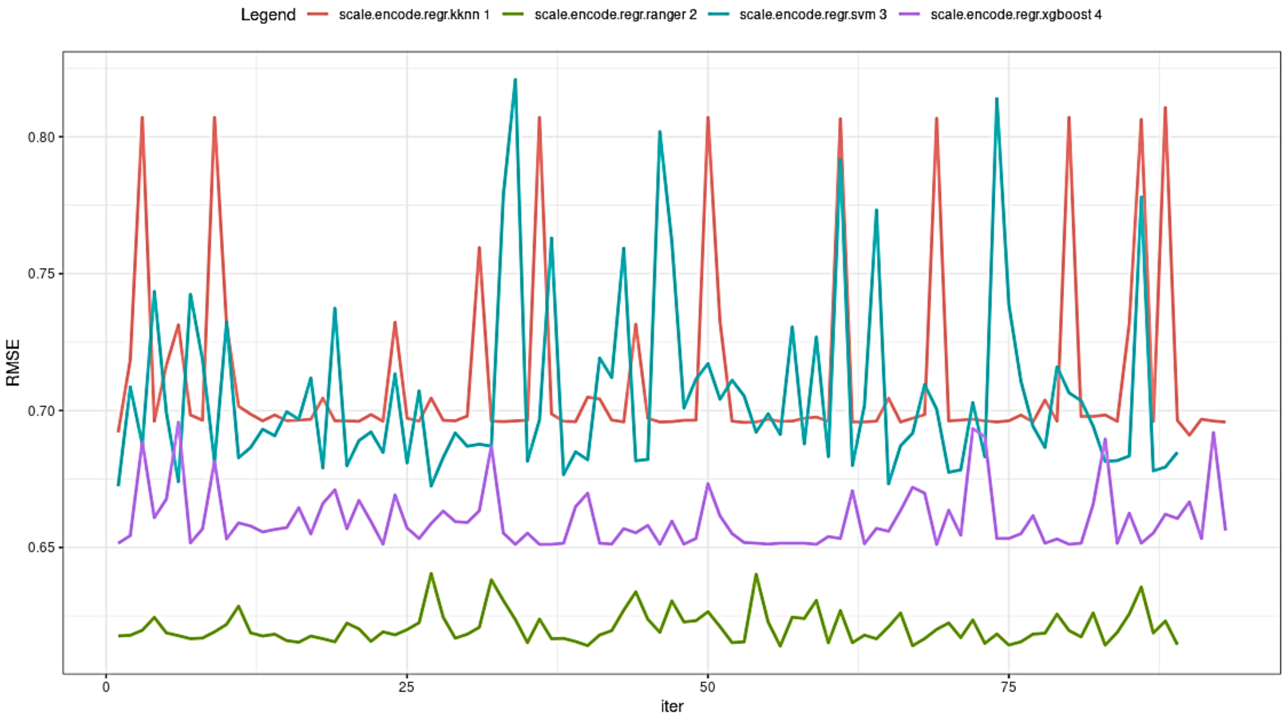

Figure 2. Optimization process for pipeline optimization without tuning preprocessing steps.

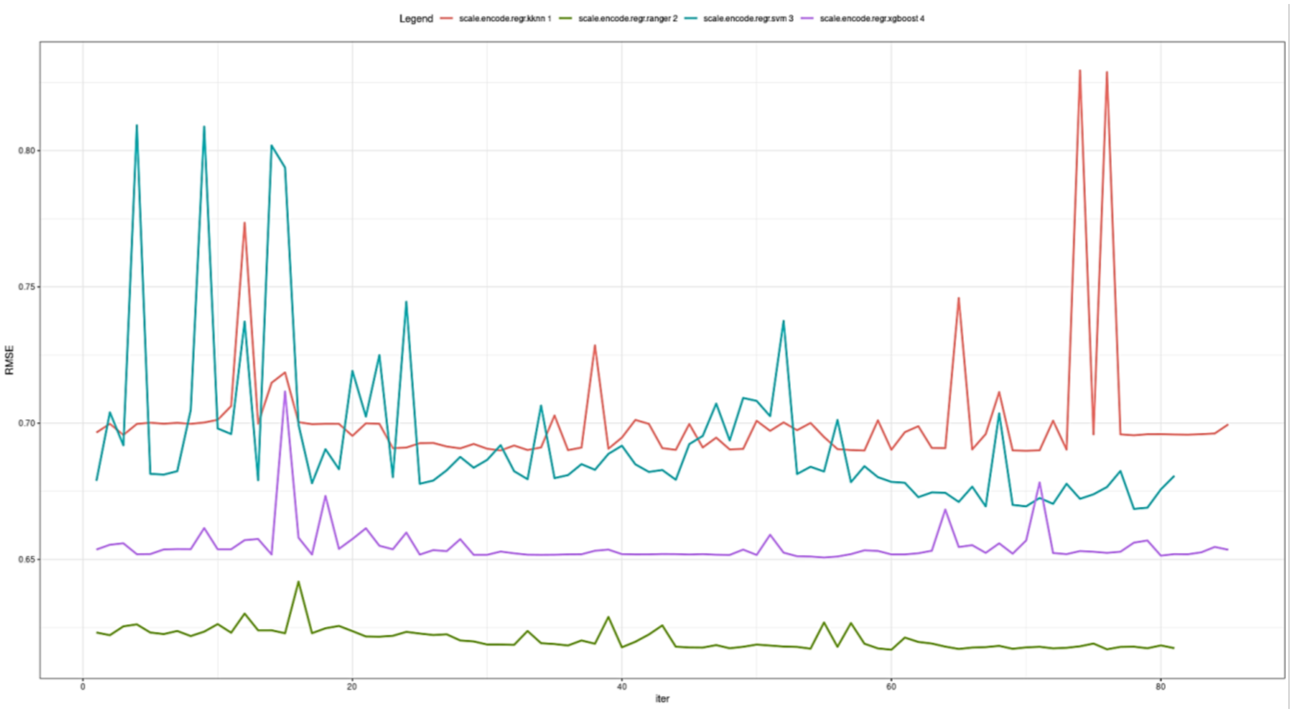

Figure 2. Optimization process for pipeline optimization with tuning preprocessing steps.

The results for the best run after optimization for training (5-fold CV) and validations using optimum setting were presented in table 4.

Table 4. The RMSE for the training and validation sets using optimum setting.

Model	RMSE for 5-fold CV	RMSE for validation
k-Nearest Neighbors	0.69	0.70
Random Forest	0.61	0.60
Support Vector Machine	0.67	0.66
Gradient Boosting	0.65	0.64

We know the lower RMSE represents the better fit. Based on table 3, Random Forest shows better performance with RMSE = 0.60 in comparison with the other models. The RMSE of 5-fold CV and validation were approximately equal for all the models. All in all, we cannot observe any improvement in the results after combining pipeline and hyperparameter optimizations compared to previous exercise (hyperparameter optimization). It is suggested to repeat the experiment with specific pipeline for each ML algorithm.