COSC 5557
Hyperparameter Optimization

Almountassir Bellah Aljazwe

March 2024

1 Introduction

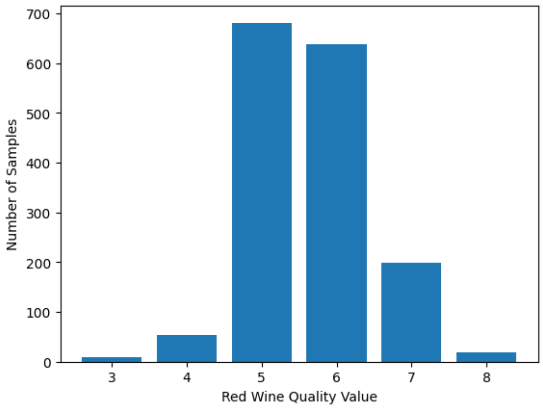
The main objective of finding an optimal predictive model for our "Red Wine Quality" dataset remains in progress. For context, the performance accuracy of five different predictive models were evaluated and compared, in our previous report. What was done, however, is not sufficient in finding an optimal predictive model; this is due to the fact that no hyper-parameter optimization was performed during the previous step. With this in mind, optimizing the hyper-parameters is the necessary upcoming step in accomplishing our main objective.

2 Dataset Description

The 'Red Wine Quality' dataset contains 1599 rows (samples) and twelve columns; eleven are feature columns, while the final twelfth is the target column. The target column contains categorical values; each value is an integer which corresponds to a 'quality' ranking from '1' to '10'. The dataset, however, ranges from '3' to '8' only, with all six values appearing. The dataset contains no missing values for all the provided samples.

A defining characteristic of the 'Red Wine Quality' dataset is its imbalanced distribution of target values. This can be concluded from the following :

3	4	5	6	7	8
10/1559	53/1559	681/1559	638/1559	199/1559	18/1559
$\approx 0.6\%$	$\approx 3\%$	$\approx 44\%$	$\approx 41\%$	$\approx 13\%$	$\approx 1\%$



From the figures, we can clearly observe that the data distribution is cluttered around the middle target values. This, in reality, is reasonable as it may be rare to encounter wine with extremely low quality or extremely high quality; it is reasonable to expect wine with average quality. With regards to our model, the imbalanced nature of this dataset may present obstacles in the performance of our predictions; this is due to the low number of samples for quality values, other than '5' and '6'.

3 Experimental Set-up

The experimental set-up was mostly similar for all the predictive models. Each model was imported through the Sklearn Python library. Depending on the model, some basic data processing was required to allow proper model training and hyper-parameter optimization; the models concerned are the K-Nearest Neighbors, Logistic Regression and Support Vector Machine Classifier; these models either required the data to be normalized, or benefit by converging quicker such as the Logistic Regression model. The models then enter the hyper-parameter optimization stage. To optimize the hyper-parameters of the mentioned five models, three optimization algorithms were used : Random Search, Bayesian Optimization, and Bayesian Optimization with Hyper Band (BOHB). Three out of the five predictive models optimized their hyper-parameters using two independent trials of Random Search and Bayesian Optimization; the mentioned models are K-Nearest Neighbors, Random Forest, Decision Tree Classifier. Two of the five optimized their hyper-parameters using two independent trials of Bayesian Optimization and Bayesian Optimization with Hyper Band; the mentioned models are Logistic Regression and Support Vector Machine Classifier. Testing of the models and

their different hyper-parameters relied on 10-fold cross validation and were evaluated through an accuracy score.

The following tables describe the hyper-parameter space of which the hyper-parameter optimization algorithms received as inputs, for each predictive model :

Table 1: K-Nearest Neighbors

Hyper-Parameter	Space Type	Range
Number of Neighbors	Integer	[1, 1000]
Weights	Categorical	Uniform or Distance
Algorithm	Categorical	Auto, Ball Tree, KD Tree, Brute
Leaf Size	Integer	[1, 100000]
p	Integer	[1, 100000]

Table 2: Random Forest

Hyper-Parameter	Space Type	Range
Number of Estimators	Integer	[50, 1000]
Criterion	Categorical	Gini, Entropy, Log Loss
Max Depth	Integer	[1, 100]
Minimum Samples Split	Integer	[2, 100]
Minimum Samples Leaf	Integer	[1, 100]
Minimum Weight Fraction Leaf	Real Number	[0.0, 0.5]
Maximum Features	Categorical	Square Root, Log2, None

Table 3: Decision Tree Classifier

Hyper-Parameter	Space Type	Range
Criterion	Categorical	Gini, Entropy, Log Loss
Splitter	Categorical	Best, Random
Max Depth	Integer	[1, 1000]
Minimum Samples Split	Integer	[2, 100]
Minimum Samples Leaf	Integer	[1, 100]
Minimum Weight Fraction Leaf	Real Number	[0.0, 0.5]
Maximum Features	Categorical	Square Root, Log2, None

Table 4: Logistic Regression

Hyper-Parameter	Space Type	Range
Solver	Categorical	LBFGS, Liblinear, Newton-CG, Newton-Cholesky, SAG, SAGA
Penalty	Categorical	L1, L2, Elastic Net, None
C	Real Number	[0.1, 1000]
Maximum Iterations	Integer	[1000, 5000]

Table 5: Support Vector Machine Classifier

Hyper-Parameter	Space Type	Range
C	Real Number	[0.1, 100]
Kernel	Categorical	Linear, Poly, RBF, Sigmoid
Degree	Integer	[1, 20]
Gamma	Categorical	Scale, Auto
Coef0	Real Number	[0, 1]
Tolerance	Real Number	[0.001, 1]

The three implementations of hyper-parameter optimization were possible thanks to three helpful Python packages :

- Random Search - sklearn
- Bayesian Optimization - baytune
- Bayesian Optimization with Hyperband - hpbandster_sklearn

4 Results

After the hyper-parameters, of each model, were optimized, the corresponding accuracy scores and the corresponding hyper-parameters of the best results were kept and a comparison is made between the different hyper-parameter optimization algorithms :

Table 6: Random Search

Iterations	Model	Best Accuracy Score	Best Hyper-parameters
200	K-Nearest Neighbors	$\approx 55\%$	Neighbors : 403 Weights : Uniform Algorithm : Ball Tree Leaf Size : 18717 p : 168
200	Decision Tree Classifier	$\approx 59\%$	Splitter : Best Criterion : Log Loss Max. Depth : 557 Min. Samples Leaf : 92 Min. Samples Split : 48 Min. Weight Fraction Leaf : ≈ 0.03 Max Features : None
200	Random Forest	$\approx 59\%$	Estimators : 647 Criterion : Entropy Max Depth : 95 Min. Samples Split : 72 Min. Samples Leaf : 25 Min. Weight Fraction Leaf : ≈ 0.03 Max. Features : Log2

Table 7: Bayesian Optimization

Iterations	Model	Best Accuracy Score	Best Hyper-parameters
200	K-Nearest Neighbors	$\approx 56\%$	Neighbors : 241 Weights : Distance Algorithm : KD Tree Leaf Size : 39403 p : 836
200	Decision Tree Classifier	$\approx 56\%$	Splitter : Best Criterion : Gini Max. Depth : 204 Min. Samples Leaf : 3 Min. Samples Split : 40 Min. Weight Fraction Leaf : ≈ 0.14 Max Features : None
300	Logistic Regression	$\approx 59\%$	Solver : SAGA C : ≈ 0.1 Max Iterations : 1000 Penalty : L1
200	SVM Classifier	$\approx 60\%$	C : ≈ 0.68 Coef0 : ≈ 0.61 Degree : 13 Gamma : Scale Kernel : RBF Tolerance : ≈ 0.45
200	Random Forest	$\approx 60\%$	Estimators : 185 Criterion : Gini Max Depth : 27 Min. Samples Split : 32 Min. Samples Leaf : 25 Min. Weight Fraction Leaf : ≈ 0.02 Max. Features : Log2

Table 8: Bayesian Optimization with Hyperband

Iterations	Model	Best Accuracy Score	Best Hyper-parameters
90	Logistic Regression	$\approx 59\%$	Solver : LBFGS, Newton-CG (tie) C : ≈ 1.3 Max Iterations : 4101 Penalty : L2
50	SVM Classifier	$\approx 60\%$	C : ≈ 0.98 Coef0 : ≈ 0.95 Degree : 19 Gamma : Scale Kernel : RBF Tolerance : ≈ 0.32

Using the best results for each model, the following table compares the performance of the non-optimized predictive models with the optimized predictive models :

Table 9: Non-optimized vs. Optimized Model Comparison

Model	Non-optimized Accuracy Score	Optimized Accuracy Score
K-Nearest Neighbors	$\approx 51\%$	$\approx 56\%$
Random Forest	$\approx 59\%$	$\approx 60\%$
Decision Tree Classifier	$\approx 56\%$	$\approx 59\%$
Logistic Regression	$\approx 58\%$	$\approx 59\%$
SVM Classifier	$\approx 59\%$	$\approx 60\%$

NOTE : Non-optimized models were evaluated again, with normalized data to allow for unbiased comparisons with the optimized models. Thus, the results are different from the previous report. These updated results can be found in the folder containing all the code files.

Given the results obtained, we can observe that each predictive model improves (although not by much) after hyper-parameter optimization. It is, however, not a clear decision as to which predictive model is the optimal model as the performance is nearly similar for four of the five models. In addition, the performance is still not good enough; this may indicate a need for data processing in the initial stages of the model development.