Practical Machine Learning

Hyperparameter Optimization

Bimal Pandey

Introduction: This report talks about the hyperparameter optimization of different ML models. The performance of every ML model depends upon their hyperparameter selection. They control the learning algorithm or the structure of the model. First of all, different ml models are applied to predict the wine quality and after that, the hyperparameters of ML algorithms are tuned using Bayesian optimization and Randomized search to obtain the best parameter and best parameters are used for fitting the model to obtain best optimized model.

Dataset Description: The Red Wine dataset consists of 1599 observations and 12 characteristics, out of which 11 are input variables and the remaining one is output variable. Here, the data have only float and integer values (only for the target variable) and there are no null/missing values. The describe() function returns the count, mean, standard deviation, minimum, 25%, 50%, 75%, and maximum values and the qualities of data. The duplicate records are removed using data.drop_duplicates(inplace=True).

Input Variables:

- fixed Acidity
- volatile acidity
- citric acid
- residual sugar
- chlorides
- free sulfur dioxide
- total sulfur dioxide
- density
- pH
- sulphates
- alcohol

Output variable:

- quality

Used libraries and modules: To start with, I imported the following libraries and loaded the dataset, and the original data are separated by ";" in the given data set.

- Numpy: It will provide support for efficient numerical computation
- Pandas: It is a convenient library that supports data frames. Working with pandas will bring ease to many crucial data operations.
- Seaborn: It is a visualization library based on Matplotlib which provides a high-level interface for drawing attractive statistical graphics.
- Sklearn: It is a Python library for data mining, data analysis, and machine learning.
- Matplotlib: It provides a MATLAB-like plotting framework

Experimental Setup: Random Forest Regression and Support Vector Machine Regression algorithms are used for the predictions of the wine quality. The hyperparameters of both algorithms are tuned using Bayesian optimization and the Randomized Search method. The performance of each model was analyzed using mean square error. Finally, the most appropriate algorithm and optimization method was selected among each of two under the consideration. The hyperparameters that were tuned during Randomized Search for random forest classifier are 'n_estimators' with range (50, 1000). The range for 'max_depth' was (10,30). The range for 'min_sample_split' was (2,11) , 'max_features' were ['sqrt', 'log2', None], and 'min_sample_leaf" was selected between (1,8).

Similarly, the ranges of hyperparameters that were tuned using Bayesian optimization for random forest classification were n_estimators= (50, 400), max_depth(1, 30), min_sample_split(2, 11) , min_samples_leaf (1, 7), and max_features': ('sqrt', 'log2', None).

The hyperparameter optimization for support vector machine was also done using RandomizedSearchCV and Bayesian Optimization. At first, the mse was calculated using default hyperparameters and later the optimized parameters were used for assessing the MSE of model. The range of hyperparameter that were tuned for SVM using RandomizedSearchCV were:

```
'C': np.logspace(3, 100),
'epsilon': np.logspace(3, 60),
'gamma': ['scale', 'auto'],
'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
'degree': np.arange(1, 10),
'coef0': np.linspace(0, 15),
'shrinking': [True, False]
```

Similarly, the range of hyperparameters that were used for SVM using Bayesian optimization were:

```
pbounds = {
  'C': (1e-3, 1e3),
  'epsilon': (1e-3, 1e3),
  'gamma': (1e-6, 1e1),
  'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
  'degree': (1, 6),
  'coef0': (0.0, 10.0),
  'shrinking': [True, False]
}
```

For both ML Algorithms, the performance before hyperparameter optimization and after hyperparameter optimization were performed and their results are explained in Results and Discussion sections. Let us discuss the default hyperparameters that were used for Random Forest before optimization, they are n_estimators=20, max_depth=10,min_samples_split=5,min_samples_leaf=5,max_features= None. Similarly, for Support vector Machine model, the default hyperparameters were kernel='rbf', C=1.0, epsilon=0.1, gamma='scale', degree= ‘5’, coef0= ‘7’. Shrinking= ‘True’).

Nested resampling gives a realistic estimate of model performance and prevents overfitting during hyperparameter adjustment. This is especially crucial when comparing models and evaluating their performance and for comparing the suitability of different hyperparameter optimization. The code splits the training and testing data at a ratio of 0.8 to 0.2 to create an outer loop for model evaluation. The training folds are used to tune hyperparameters, while the validation fold evaluates the model's performance using those parameters. An inner loop is used to tune Random Forest Regression and Support Vector Machine model hyperparameters after each outer loop iteration. RandomizedSearchCV and Bayesian Optimization have been used for hyperparameter optimization in the provided code.

RandomizedSearchCV randomly selects hyperparameters from the defined parameter grid ('param_grid') and evaluates performance through cross validation on the Training folds. Both the optimization technique used 5 fold cross validation and 100 iterations were done for both optimization strategies. The best hyperparameters are then used for fitting the model to evaluate the model performance.

Results and Discussions: The mean squared error for RandomForestRegression and Support Vector Mcahine were 0.35 and 0.47 with default hyperparameters. The mean squared error for random forest and support vector machine were found to be 0.31 and 0.44 after hyperparameter optimization was done using Randomized SearchCV. Similarly, the mean squared error for Random Forest Regression and support Vector Machine were found to be 0.32 and 0.45 after using the best hyperparameters obtained using Bayesian Optimization.

ML Model	Best Hyperparameter using Randomized SearchCV	Best Hyperparameter obtained using Bayesian Optimization
Random Forest Regressor	Best Hyperparameters: {'n_estimators': 400, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': 30}	Best Hyperparameters: {'n_estimators': 629, 'min_samples_split': 6, 'min_samples_leaf': 3, 'max_features': 'log2', 'max_depth': 25}
Support Vector Machine	Best Hyperparameters for SVR: {'shrinking': True, 'kernel': 'rbf', 'gamma': 'auto', 'epsilon': 3, 'degree': 5, 'coef0': 1.0204081632653061, 'C': 7}	Best Hyperparameters: OrderedDict([('C', 803.9732910126631), ('coef0', 3.647427224286979), ('degree', 5), ('epsilon', 411.7923777955162), ('gamma', 7.647440714883904), ('kernel', 'rbf'), ('shrinking', False)])

Ml Model	MSE before optimization	MSE after Randomized Search	MSE after Bayesian Optimization
Random Forest Regression	0.35	0.31	0.32

Support Vector Machine	0.47	0. 44	0.45

References:

1. https://towardsdatascience.com/bayesian-optimization-for-hyperparameter-tuning-how-and-why-655b0ee0b399
2. https://github.com/topics/hyperparameter-tuning?l=jupyter+notebook
3. https://youtu.be/xRhPwQdNMss?si=sWANCXFC-u3vR3qQ
4. https://github.com/topics/random-forest-classification?o=asc&s=stars
5. https://www.analyticsvidhya.com/blog/2022/11/hyperparameter-tuning-using-randomized-search/