Practical Machine Learning

Pipeline Optimization

Sanjeeb Humagain

May 2, 2024

Introduction:

The main goal of this exercise is to use pipeline optimization for Machine Learning (ML) models which can have significant impact on the model's performance. In this report, Random Forest Regression and Neural Network Regression are optimized by implementing Bayesian Optimization and Random Search Optimization and the results are compared and finally suggesting the most appropriate optimization technique for this specific data and problem.

There are a significantly various methods which could be used for optimization of a ML. However, selection of appropriate technique for a particular dataset can be quite challenging task. After reviewing several resources, it was found that Bayesian optimization and random search optimization methods are widely used technique for hyperparmeter optimization. Therefore, those two methods were chosen in this task.

To select an appropriate ML algorithm and the optimization method, first we need to understand the problem. For instance, if the problem is related to classification we have to choose a different algorithm than an algorithm for prediction. For instance, Linear Regression is used for forecasting and Logistic Regression algorithm is used for classification problems.

Correlation of features were checked and there were some negative correlations as well. A negative correlation is a relationship between two variables that move in opposite directions. From the correlation plot of heatmap, it was fond that some features have negative correlation with system load indicating that system load decreases as the particular feature increases and vice versa. The one advantage of checking the correlation is that we could remove the features which has very less correlation or zero correlation with the data we want to forecast. Removing such features would reduce the complexity and might ultimately help to increase the model performance.

Following libraries are used for this task:

- `pandas`
- `seaborn`
- `matplotlib.pyplot`
- `numpy`
- `train_test_split`
- `BayesianOptimization from bayes_opt`
- `mean_absolute_error, mean_squared_error, r2_score from sklearn.metrics`
- `RandomForestRegressor`
- `RandomizedSearchCV`
- `BayesSearchCV from skopt`

Some of the libraries used are listed above. They are used for data preprocessing, numerical computation, forecasting, optimization and performance evaluation of the model.
In all ML algorithms used, roughly same steps are followed. They are, checking for the missing value, data preprocessing, clean up, scaling, split up, creating model, training the model and running the test, and hyperparameter optimization etc. After performing the hyperparameter optimization, the same model was run with the optimized set of values of the hyperparameters. Finally, the performance of the model and hyperparameters are summarized and discussed by comparing mean square error, mean absolute error and r2 score of each of the options considered in this experiment.

Dataset Description:

In this experiment, the dataset has 9 features which are used to predict the system load. The data set contains 87648 samples of data and the target system load from 2006 to 2011. In this exercise, the system load data is being used to compare machine learning algorithms and optimization methods.

Experimental Setup:

Python is used for this experiment because of the availability of crucial libraries for instance Pandas (for data manipulation), Scikit-learn (for ML) etc. The dataset imported from the csv file and split into training and testing sets in ratio of 80:20 in the same way as performed in previous exercises. The input features are: Date, DryBulb, DewPnt, WetBult, ElecPrice, Day, Month, Year, Minutes, and target or output is SYSLoad.

Random Forest (RF) and Neural Network (NN) are used as ML algorithms for this exercise. Similarly, two optimization techniques, Bayesian optimization and Random Search optimization methods are implemented for hyperparameter optimization process. The performance of each model was assessed using mean square error, mean absolute error and r2 score. The pipeline optimization would give the best score for Random Forest and Neural Network after finishing up the execution. Additionally, we will use the best estimator value for each model to forecast and print the result for comparison.

Nested resampling is a process that provides reliable estimate of a model's performance and assists to prevent overfitting during the entire process of hyperparameter tuning. Specially, it is important when evaluating the performance of various models or even when comparing the different techniques for hyperparameter optimization. In the code, first the training and testing data are split in ratio of 0.8/0.2 which works as outer loop for model evaluation. Hyperparameter tuning is performed on the training folds, while the validation fold is used to evaluate the performance of the model with the selected hyperparameters. Additionally, for each iteration of the outer loop, an inner loop is used to tune the hyperparameters of NN or RF. RandomizedSearchCV and BayesSearchCV functions are used for hyperparameter optimization both of which will accept the cross validation number explicitly. We have done 5 fold cross validation in both of the cases. Finally, the combination of hyperparameters that yields the best performance (i.e. lowest negative MSE) is selected as the best hyperparameters.

During the pipeline optimization, first of all, data is divided into numeric and categorical features. Then, both data are separately handled for missing values. Numeric data uses imputer with strategy of median and scaled accordingly. However, categorical data is transformed by imputer with strategy of most frequency category for the missing data. Additionally, OneHotEncoder is used in categorical transformer. In pipeline optimization, it's important to preprocess both numeric and categorical features appropriately to ensure that the machine learning model can effectively learn from the data.

Furthermore, we define pipeline for RF and NN separately with appropriate range of hyperparameters. BayesSearchCV is used for hyperparameter optimization for both cases as Bayesian Optimization worked well in the previous HPO exercise. We have used number of iteration as 20 and cross validation of 5 in both cases.

For Random Forest algorithm following hyperparameters are tuned:

1. n_estimators
2. max_depth
3. min_samples_split
4. min_samples_leaf

For Neural Network algorithm following hyperparameters are tuned:

1. hidden_layer_sizes
2. activation
3. alpha
4. max_iter
5. learning_rate_init

Results:

The metrics after pipeline optimization are tabulate in table 1. We can see that the R2 score values for Random Forest is higher than that of Neural Network.

In table 2, hyperparameters ranged defined and best hyperparemeter values after Bayesian Optimization is summarized.

The MSE, MAE and R2 score of both algorithms using both optimization methods are tabulated below.

S.N.	Metrics	ML algorithms	
		Random Forest (RF)	Neural Network (NN)
		Bayesian Optimization	Bayesian Optimization
1	Mean Squared Error (MSE)	130663.8219	309325.9310
2	Mean Absolute Error (MAE)	259.6662	443.2211
3	R2 Score	0.9350	0.8462

Table 1: MSE, MAE and R2 score for RF and NN using Bayesian HPO

ML algorithm					
Neural Network (NN)			Random Forest (RF)		
Hyperparameters	Range defined	Tuned value	Hyperparameters	Range defined	Tuned value
regressor__hidden_layer_sizes	Integer(8, 256, prior='log-uniform')	136	regressor__n_estimators	Integer(10, 1000)	1000
regressor__activation	['identity', 'logistic', 'tanh', 'relu']	relu	regressor__max_depth	Integer(3, 40)	40
regressor__alpha	Real(1e-5, 1e-2, prior='log-uniform')	2.1142e-05	regressor__min_samples_split	Integer(2, 10)	2
regressor__max_iter	Integer(10, 100)	'logistic'	regressor__min_samples_leaf	Integer(1, 5)	1
regressor__learning_rate_init	Real(0.0001, 0.1, prior='log-uniform')	0.09759			

Table 2: Hyperparameter values ranges defined and hyperparameters after Optimization

The presence of a ConvergenceWarning in neural network algorithm suggests, in my view, that additional iterations are necessary to reach an optimal solution. This indicates that Neural Network would take considerable amount of time and the model performance is considerably less than RF.

The overall execution time was 112 minutes which is less time need collectively to run Bayesian optimization for both RF and NN algorithms in previous exercise. This suggests that pipeline optimization increases the efficiency in terms of time.

In conclusion, pipeline optimization increase efficiency in time and based up on R2 score random forest algorithm is more suitable than neural network algorithm in this case.

The Plots of all the algorithms used are listed below.

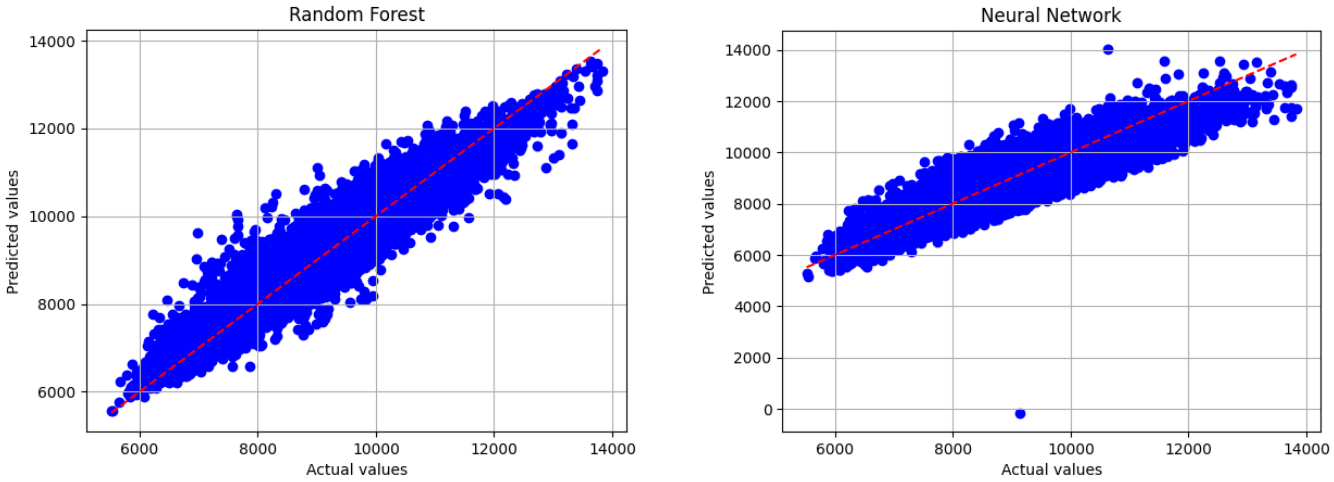

Fig1: Actual vs predicted value using RF and NN with Bayesian HPO

Code is available in github repository: https://github.com/COSC5557/pipeline-optimization-Sanjeeb-PL.git