Introduction: For this exercise, I took a look at the Wine Quality data set to perform hyperparameter optimization to get a better understanding of hyperparameter tuning process. Optimizing hyperparameters can help improve the model performance and produce more optimal results.

Data Set Description: For this analysis, I used the Red Wine Quality Data. Here, In this data set, wine quality is measured by twelve features including fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol, and quality. For each of these features, the number of observations was 1599. Regarding preprocessing, I checked for null values using a isnull().values.any() function. There were not any null values. In this exercise, hyperparameter optimization was performed to improve the predicting model. The feature chosen to predict was quality.

Experimental Setup: To work with hyperparameter optimization, two ML algorithms were chosen to optimize: Random Forest and SVC.

Beginning with Random forest, the pandas package was imported to aid in data analysis and importation of the dataset. Further, the sklearn package was used to utilize machine learning models such as Random Forest Regressor and SVC. Additionally, the sklearn package was used to implement randomly splitting the Red Wine Quality dataset into train and test subsets. For hyperparameter optimization, Bayesian Optimization was utilized from the package bayes_opt.

Once packages were loaded, the Red Wine Quality Data set was downloaded via the pandas package and checked for null values using the isnull().values.any() function. Since there were none, the input features (x) and output targets (y) were defined accordingly. Next, the dataset was split into training and testing data using the train_test_split function from sklearn. Here, I chose the test size to be 20% and the train size to be 80%.

Once the data was loaded, I continued with Bayesian Optimization. First, I did not know what hyperparameters were included in the Random Forest algorithm so I used a get_param_names() function to look at hyperparameters. Then, to further understand what each hyperparanter measured, I went to the sklearn information page about Random Forest before picking the hyperparameters to tune. From the sklearn library page, I chose the hyperparameters with integer values. Once I looked at these, I chose to optimize n_estimators, max_depth, min_samples_split, max_features, max_leaf_nodes, max_samples, min_impurity_decrease, min_samples_leaf, min_weight_fraction_leaf, n_jobs, and verbose. Next, I defined an objective function using the parameters I chose from the list I generated. When defining my objective function, I defined my model to be Random Forest Regressor and listed all of the hyperparameters as integer values. Then, I added a line to return the negative mean cross validation score. This was the score I was trying to improve upon.

Next for Random Forest, I defined the parameter bounds and then ran the Bayesian Optimization. When optimization was complete, I returned the best hyperparameters by using the opt.max['params'] function. Then, I went on to train the final model using the hyperparameters that were optimized using Bayesian optimization. The model was fit and a R^2 Test score was generated.

Next, I ran a Bayesian Optimization on a SVC algorithm. For this portion, the sklearn package was utilized for importation of SVC and preprocessing using the MinMaxScaler. The pandas package was used for data importation and the bayes_opt was used to run Bayesian Optimization.

Just as in Random Forest Regressor, data was downloaded using pandas and the input and output were set accordingly. The data was then split into train and test data subsets using the train_test_split function.

Next, I did some preprocessing by using the MinMaxScaler to linearly scale the values into a set range. Then, I defined an objective function and set the model to be SVC. Before I listed hyperparameters I used get_param_names() function to look at hyperparameters. Then, I included every hyperparamter in my objective function and ended with a return of the negative mean cross validation score. This was the score I was trying to improve upon.

Next for SVC, I defined the parameter bounds and then ran the Bayesian Optimization. When optimization was complete, I returned the best hyperparameters by using the opt.max['params'] function. Then, I went on to train the final model using the hyperparameters that were optimized using Bayesian optimization. The model was fit and a R^2 Test score was generated.

Results: Beginning with Random Forest this is what I observed. As I was using Bayesian Optimization, many parameters came back with an error either explaining that the hyperparameter had to be boolean or fall within a specific range of numbers. When I observed the "needs to be boolean" argument, I eliminated this hyperparameter from the optimization. When I observed the "needs to fall in this range" argument, I adjusted the parameter boundaries. In the end, the hyperparameters I optimized included n_estimators, max_depth, min_samples_split, max_features, max_leaf_nodes, max_samples, min_decrease, min_samples_leaf, and min_weight_fraction_leaf. Once optimized, the final R^2 test score was -0.024 for Random Forest.

For SVC, many parameters came back with an error either explaining that the hyperparameter had to be boolean or fall within a specific range of numbers as well. This resulted in C, cache_size, coef0, and gamma being tuned. Once optimized, the final R^2 test score was 0.516 for SVC.

Of these two models after Bayesian Optimization, SVC appears to work better for the Red Wine Quality Data set due to a higher R^2 test score. The negative R^2 test score for Random Forest implies the fit was not very good and or did not fit the shape of the data well.

References:

1. Lee, Dr. Ernesto. "Step-by-Step Guide: Bayesian Optimization with Random Forest." Medium, Medium, 31 Aug. 2023, drlee.io/step-by-step-guide-bayesian-optimization-with-random-forest-fdc6f329db9c.

2. "Sklearn.Preprocessing.MinMaxScaler." Scikit, scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html. Accessed 25 Mar. 2024.

3. "Sklearn.Ensemble.Randomforestregressor." Scikit, scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. Accessed 25 Mar. 2024.

4. "Sklearn.Svm.SVC." Scikit, scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. Accessed 25 Mar. 2024.

5. "Get_all_params - Catboostregressor." CatBoost, catboost.ai/en/docs/concepts/python-reference_catboostregressor_get_all_params. Accessed 25 Mar. 2024.