**Introduction:** In order to better organize machine learning workflows, pipelines can be used to make work quicker. In this exercise, two different data sets (Red Wine Quality Data Set and Brain Stroke Data Set) were used to compare performance of an ML pipeline. Within the pipeline, feature selection and a model (Random Forest Classifier algorithm) were combined. Following pipeline setup, a Bayesian Optimization was performed on the pipeline. The test mean accuracy score was then collected. Here, the mean accuracy score describes the mean accuracy of the given data across all subsets (cv).

**Dataset Description:** The first data set worked with was the Red Wine Quality data set. In this data set, red 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.

The second data set worked with was the Brain Stroke data set. In this data set, occurrence of a stroke is measured by eleven features including gender, age, heart disease, ever married, work type, residence type, average glucose level, BMI, smoking status, and presence of stroke. For each of these features, the number of observations was 4981. Regarding preprocessing, I checked for null values by observing the table produced by .describe and also by checking with isnull().values.any() function. There were not any null values, however five features were listed as objects for data type. To change these to int, I used one hot encoding. The features changed from object to int using one hot encoding were gender, ever married, work type, residence type, and smoking status.

**Experimental Setup:** For this exercise, I used the Python programming language via Google Colab. For packages, pandas was used for data manipulation. The sklearn package was used for further data processing such as train_test_split, RandomForestClassifer, Pipeline, VarianceThreshold, make_classification, model_selection, and cross_val_score. For Bayesian Optimization, the sci-kit optimize package was imported and the BayesSearchCV was utilized.

As seen in the Auto Sklearn Fail exercise and the Hyperparameter Optimization exercise, the Wine Quality dataset is very imbalanced. Because of this, the same techniques used in those two exercises were used here. To account for imbalanced data and underrepresented data groups, Synthetic Minority Oversampling Technique (SMOTE) and train_test_split stratification were used before running Random Forest Classifier. To gain reproducible results across multiple executions, the random state was set to 42.

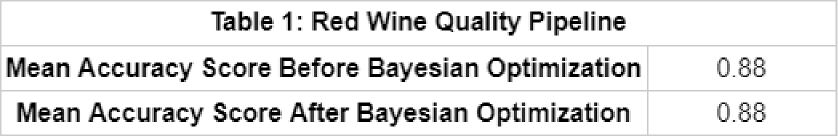
Beginning with Red Wine Quality Data Set, the pipeline was built to include VarianceThreshold as the selector and RandomForestClassifier as the model. This pipeline was fit and the mean accuracy test score was generated before a Bayesian Optimization was conducted on the pipeline. For Bayesian Optimization, the number of iterations was set to 50 initially. Then, the cross_val_score function was used where the number of folds (cv) was set to 3. Initially, just an inner cross-validation was set-up, but later an outer cross-validation was set-up in addition to the inner cross-validation (cv=3) on 50 iterations. Hyperparameters tuned as part of the pipeline here included selector_threshold (range of

0,0.1)) for the selector (Variance Threshold). For the model, max_depth (range of (1,500), max_leaf_nodes (range of (10,100)), max_samples (range of (0.1, 0.8)), min_samples_leaf (range of (1,6)), min_samples_split (range of (2,8)), n_estimators (range of (50,700)), and n_jobs (range of (-1,400) were tuned as part of the pipeline. Regarding hyperparameter function, max_depth describes the depth of the tree, max_leaf_nodes describes the number of leaf nodes used, max_samples describes the number of samples taken from X for training, min_samples_leaf describes the minimum number of samples needed to form a leaf node, min_samples_split describes the minimum sample requirement for a split, n_estimators describes the number of trees, n_jobs describes the number of jobs that run in parallel. After the Bayesian Optimization, the new mean accuracy test score was generated.

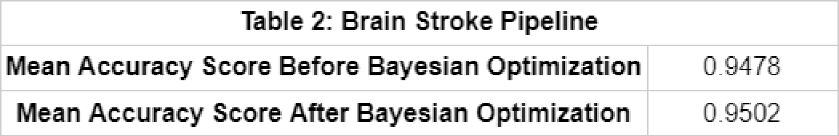
For the second data set, Brain Stroke Data Set, the same pipeline was tested as was generated for the Red Wine Quality Data Set. The hyperparameters tuned as part of the pipeline were the same as tested for the Red Wine Quality dataset. Additionally, the Brain Stroke dataset was imbalanced, so Synthetic Minority Oversampling Technique (SMOTE) and train_test_split stratification were used before running Random Forest Classifier.

**Results:** Beginning with the Red Wine Quality dataset, before the Bayesian Optimization, the test mean accuracy score was 0.88 (Table 1). After Bayesian Optimization of the pipeline, the test score was 0.88 (Table 1). Since the dataset was already adjusted for imbalanced data using stratification and SMOTE, the pipeline was optimized using Bayesian Optimization at different numbers of iterations to see if more iterations could generate a better score. Additional iterations tested were 100,200,and300 iterations. After trying these Bayesian Optimization conditions, the best score produced was still only 0.88 (Table 1). As a result of using Bayesian Optimization, the mean accuracy test score did not prove to be better than the default values.

Next for the Brain Stroke dataset, before the Bayesian Optimization, the test mean accuracy score was 0.9478(Table 2). After Bayesian Optimization of the pipeline, the mean accuracy test score was 0.9502 (Table 2). As a result of using Bayesian Optimization, the mean accuracy test score improved by 0.25%.



| Table 1: Red Wine Quality Pipeline | |
|---|---|
| Mean Accuracy Score Before Bayesian Optimization | 0.88 |
| Mean Accuracy Score After Bayesian Optimization | 0.88 |



| Table 2: Brain Stroke Pipeline | |
|---|---|
| Mean Accuracy Score Before Bayesian Optimization | 0.9478 |
| Mean Accuracy Score After Bayesian Optimization | 0.9502 |

**References:**

1. Saeed, Mehreen. "Modeling Pipeline Optimization with Scikit-Learn." MachineLearningMastery.Com, 21 Oct. 2021, machinelearningmastery.com/modeling -pipeline-optimization-with-scikit-learn/.

2. "Sklearn.Pipeline.Pipeline." Scikit, scikit-learn.org/stable/modules/generated /sklearn.pipeline.Pipeline.html. Accessed 3 Apr. 2024.

3. "Sklearn.Ensemble.Randomforestclassifier." Scikit, scikit-learn.org/stable/modules /generated/sklearn.ensemble.RandomForestClassifier.html. Accessed 15 Apr.

4. Auto-sklearn Fail Exercise wrote in overleaf. https://www.overleaf.com/project /66194f82a4874abfed54600f

5. GfG. "One Hot Encoding in Machine Learning." GeeksforGeeks, GeeksforGeeks, 21 Mar. 2024, www.geeksforgeeks.org/ml-one-hot-encoding/.

6. “Learn Hyperparameter Search Wrapper¶.” Scikit, scikit-optimize.github.io/stable /auto_examples/sklearn-gridsearchcv-replacement.html. Accessed 16 Apr. 2024.