Pipeline Optimization

Bimal Pandey

Introduction: This report introduces pipeline optimization of different ML models to predict wine quality. The method includes the classification task to predict the wine quality. Random Forest and Support vector Machine classifiers are used for the task of classification.

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.

Input Variables:

- Fixed Acidity
- Volatile acidity
- Citric acid
- Residual sugar
- Chlorides
- Free sulfur dioxide
- total sulfur dioxide
- density
- Ph
- Sulphates
- Alcohol

Output variable:

- Quality

Experimental Setup: First of all, I imported different libraries needed for implementing and running the program.

```
import pandas as pd

from sklearn.ensemble import RandomForestClassifier

from sklearn.pipeline import Pipeline

from sklearn.impute import SimpleImputer

from sklearn.compose import ColumnTransformer

from sklearn.preprocessing import StandardScaler, MinMaxScaler, OneHotEncoder

from sklearn.pipeline import Pipeline

from sklearn.model_selection import train_test_split, RandomizedSearchCV

from sklearn.metrics import accuracy_score

from sklearn.svm import SVC
```

```
from sklearn.compose import ColumnTransformer

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import KFold

!pip install optuna

import optuna
```

Then, dataset are split into training and test sets. After that, pipeline preprocessing is carried out.

```
numerical_features= X.select_dtypes(include=['int64', 'float64']).columns

categorical_features= X.select_dtypes(include=['object']).columns
```

Pipeline Preprocessing:

The ML pipeline optimization tasks tunes the preprocessing options. Num_strategy is tuned using mean, median, and most_frequent for the strategy of numerical transformation/preprocessor. Similarly, StandardScalera and MinMaxScaler are used for tuning the standardization process. The first part of pipeline uses imputer as a tool for the imputation of missing values if any and replaces the missing values with mean of data. The simple imputer is used in this process. Similarly, categorical features are encoded if there exists any. Furthermore, cat_strategy is defined as most_frequent, and constant for the strategy of categorical preprocessing and error and ignore are tuned for OneHotencoder. The Column Transformer is used for combining numerical and categorical preprocessing. The pipeline includes either RandomForestClassifier or SVM classifier in the optimization process. The hyperparameters of these classifiers are systematically tuned using Randomized Search method and Bayesin Optimization using optuna.

For the RandomForestClassifier, the default hyperparameters used are

```
n_estimators = trial.suggest_int('n_estimators', 100, 200, step=100)

 max_depth = trial.suggest_int('max_depth', 2, 25, log= True)
```

The hyperparameter optimization for random forest and SVM classifiers was performed using Randomized search and Bayesian optimization using optuna. The parameters that were tuned for random forest classification using Randomized Search method are n_estimators with range of (100, 1000), max_depth between (3, 30), min_sample_split (2, 11), min_sample_leaf (1, 5) and num_strategy mean. Median and most_frequent are used. StandardScaler and MinMax Scaler are used for standardization. Similarly, for SVM the hyperparameters that were tuned are: 'classifier__C': uniform (0.1, 10), 'classifier__gamma': uniform(0.001,0.1),'kernel':['linear','rbf','poly','sigmoid'],'preprocessor__num__imputer__strategy':['mean', 'median','most_frequent'],'preprocessor__num__scaler': [StandardScaler(), MinMaxScaler()]. Furthermore, for hyperparameter optimization of random forest using optuna, the hyperparameters that were tuned are: n_estimators(100, 1000), max_depth(3, 30), min_samples_split(2, 11), and min_samples_leaf (1, 5).

Nested resampling (nested cross validation) addresses the issue of model selection bias. It helps in preventing overfitting during hyperparameter tuning. Random search is established using RandomizedSearchCV to create 5 folds for the training data. So, the cross-validation is performed using

RandomizedSearchCV with 5-fold cross validation. The training and test data are divided in the ratio of 0.8/0.2 making the outer loop of the data. Training folds and validation folds are established using cross validation for the tuning of hyperparameters. For each iteration of outer loop, an inner loop is used to tune the hyperparameters. The inner loop is responsible for inner resampling and splits the training data again into 5 folds for cross-validation. For each combination of hyperparameters, it splits the training data into 'cv' number of folds, where each fold is used once as a validation while k-1 remaining folds form the training set. The randomized search is used for the tuning of hyperparameters for both classifier at first stage and best parameters are obtained which are used for training the classifier to get best optimized accuracy. At, the next stage Bayesian optimization using optuna is used for tuning the hyperparameters of both classifier.

The best hyperparameters that were obtained for random forest classification using Randomized Search are: n_estimators=960, max_depth= 21, min_sample_split= 4, min_sample_leaf= 3, strategy= most_frequent, and preprocessor_num_scaler is StandardScaler. The best hyperparameters that were obtained for SVM were: c= 6, gamma= 0.01, kernel 'rbf' and strategy mean and num_scaler as StandardScaler. The best hyperparameters that were obtained for random forest classifier using optuna were: n_estimators= 169, max_depth=21, min_samples_split= 7, min_samples_leaf= 3.

Results:

Randomized Search Method

RandomForestClassifier

Accuracy before optimization: 0.65

Accuracy after optimization: 0.7

Support Vector Machine

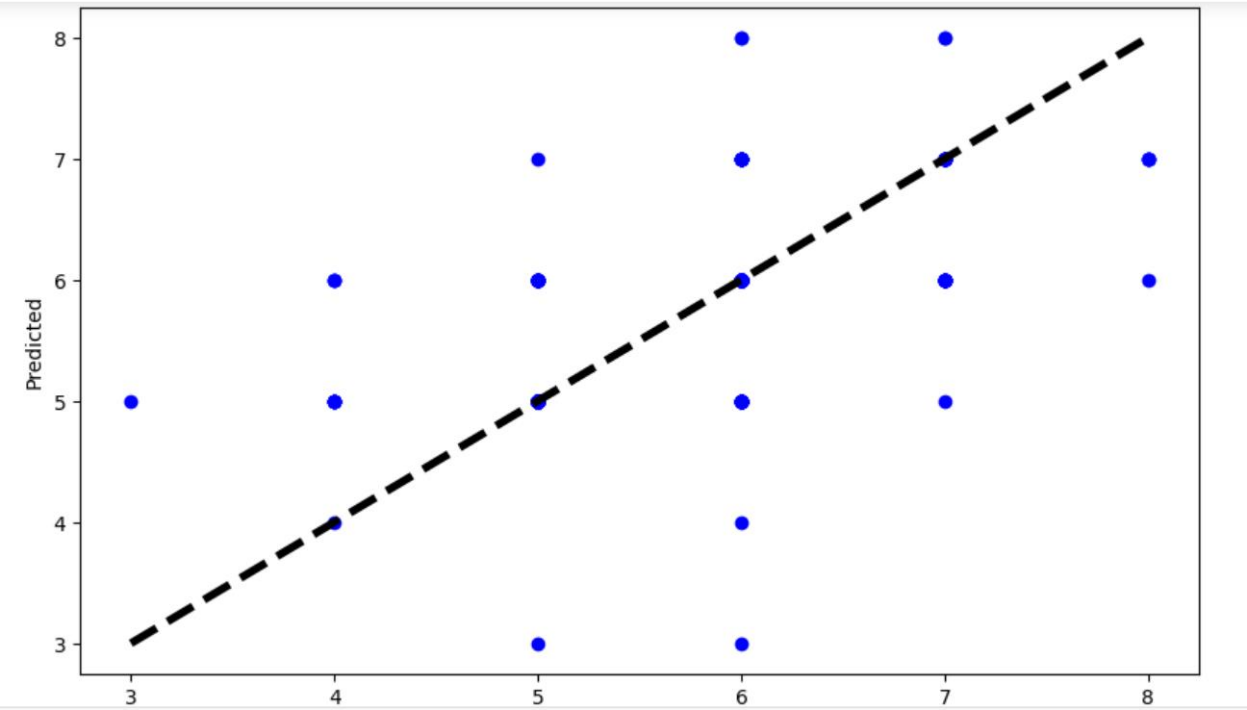
Accuracy before: 0.60

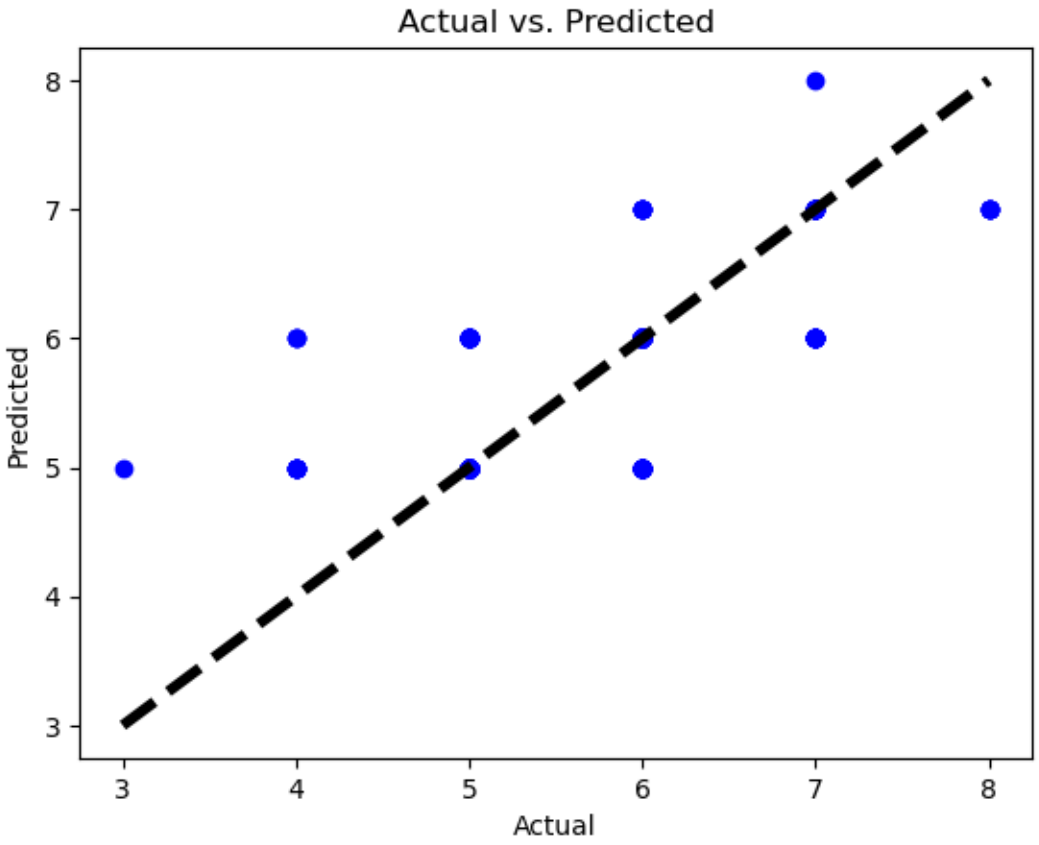
Accuracy after optimization: 0.68

Using Optuna

Random Forest Classifier

accuracy: 0.678


Predicted
8
7
6
5
4
3
3
4
5
6
7
8


Actual vs. Predicted
Predicted
Actual
8
7
6
5
4
3
3
4
5
6
7
8