**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.ensemble import GradientBoostingClassifier

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, GridSearchCV

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
```

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. The data are standardized by the Standard Scaler. 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 Grid Search method and Bayesin Optimization using optuna.

Nested resampling (nested cross validation) addresses the issue of model selection bias. The outer cross-validation is performed using GridSearchCV with 5-fold cross validation. This outer loop evaluates model performances and their hyperparameters. For each combination of hyperparameters in 'param_grid', 'GridSearchCV' performs the nested round of cross validation on the training data. For each combination of hyperparameters, 'GridSearchCV' internally splits the training data into five folds and train the model on the subsets of data to evaluate performance.

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 grid 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.

**Results:**

**Grid Search Method**

**RandomForestClassifier**

Accuracy before optimization: 0.65

Accuracy after optimization: 0.67

Support Vector Machine

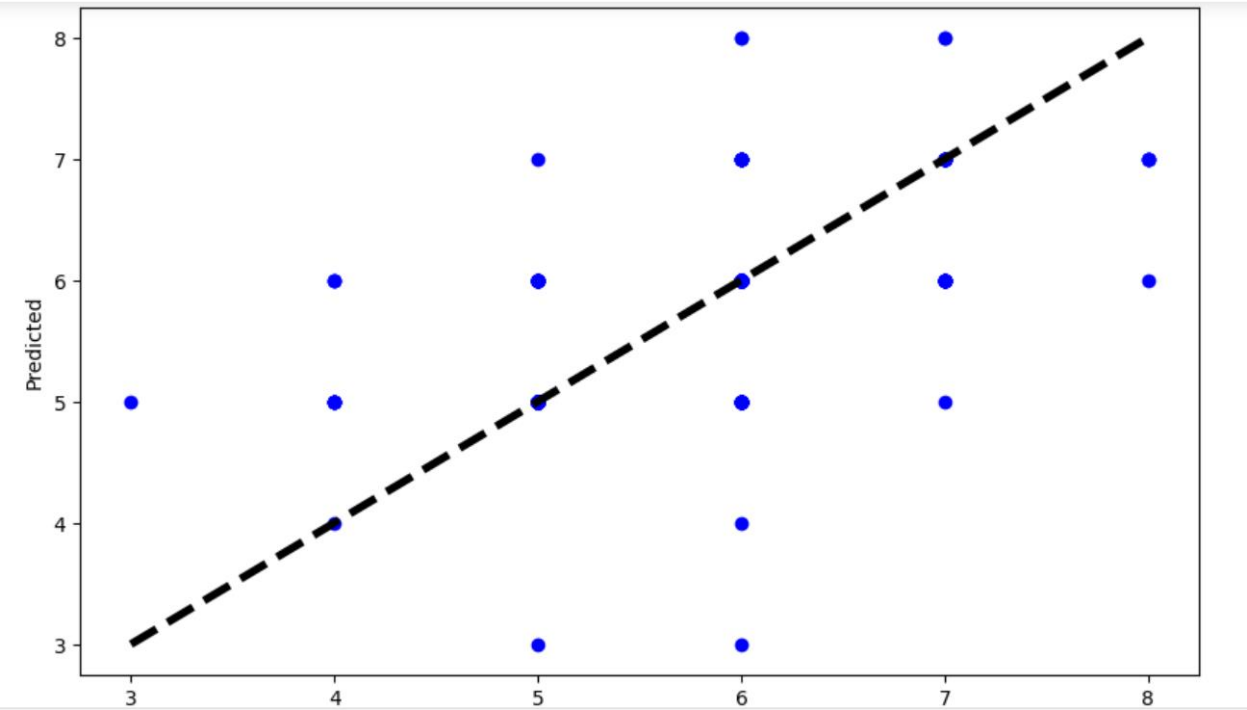
Accuracy before: 0.64

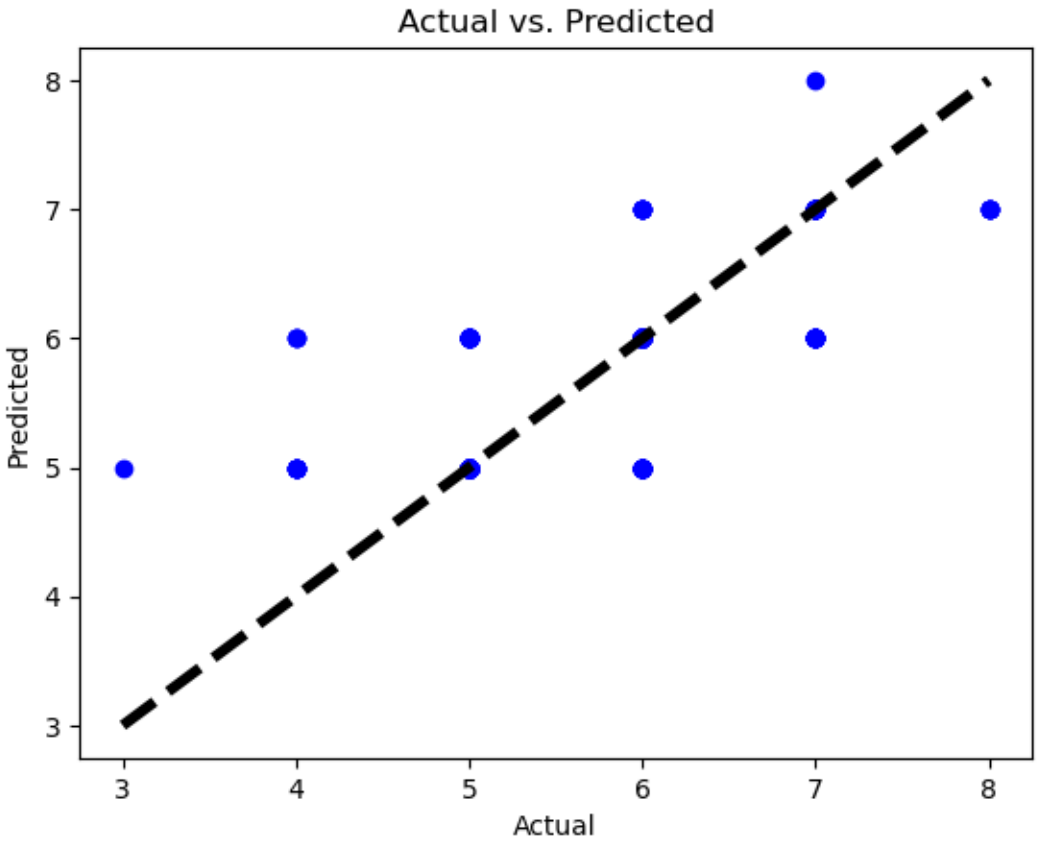
Accuracy after optimization: 0.68

**Using Optuna**

**Random Forest Classifier**

Best accuracy: 0.67




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