Pipeline Optimization

Farshad Ghorbanishovaneh

University of Wyoming

Abstract

The machine learning pipeline is a sequence of data processing components that run in a specific order. The pipeline is designed to take raw data and transform it into a format that is suitable for machine learning models. The pipeline is a critical component of any machine learning project, as it is responsible for preparing the data that will be used to train the model. In this report, we present a method for optimizing the machine learning pipeline using a Bayesian Optimization algorithm. We demonstrate the effectiveness of our method on a red wine dataset and show that it can improve the performance of the pipeline.

Introduction

Pipeline optimization is the fine-tuning of data pipelines to achieve peak performance, reliability, and scalability. The core objectives of pipeline optimization are to streamline data processing, seamlessly handle growing data volumes. Several key areas benefit from optimization efforts. Performance can be enhanced through techniques like parallelization and strategic data format selection. Scalability is addressed by designing pipelines that can gracefully handle increasing data loads. Building in redundancy and error handling mechanisms strengthens reliability, preventing data loss and ensuring smooth operation. In essence, pipeline optimization is an iterative process that ensures your data pipelines function at their best, consistently delivering the valuable insights.

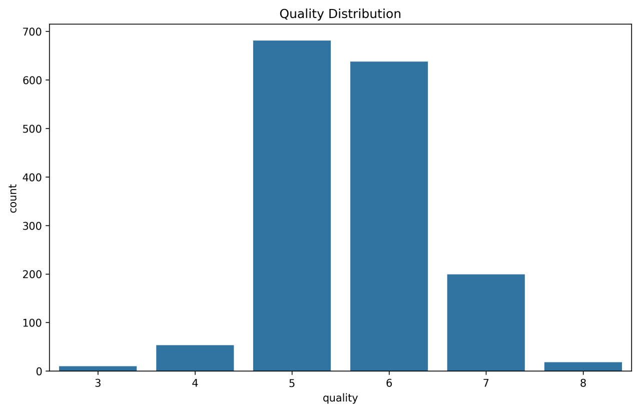

Figure 1: *Distribution of wine quality ratings in the dataset.*

Dataset Description

The Wine Quality Dataset[1] consists of 1,599 samples of wine, each characterized by 12 physicochemical properties and quality ratings. The properties include various acidity measures, sugar, sulfur dioxide levels, and alcohol content. All columns in the dataset are fully populated with non-null values. The quality ratings vary from 3 to 8, with the majority of the wines rated between 5 and 6 (Fig-

[1] `https://archive.ics.uci.edu/dataset/109/wine`

ure 1). Table 1 provides a detailed breakdown of the structural properties of the dataset.

Dataset Description

Besides the 'quality' column, which serves as our target variable, all other attributes in our dataset are of the float data type. The distribution of these data points is depicted in the boxplot Figure 2. In Figure 3, you can observe the correlations between each variable within the dataset, reflecting how each physicochemical property relates to the others and to the overall quality of the wine. The strongest positive correlation with quality is exhibited by alcohol content, suggesting that higher alcohol levels often correspond to higher quality ratings. Conversely, volatile acidity has the strongest negative correlation with quality, indicating that lower volatile acidity is associated with higher quality wines. Other properties, such as sulphates and citric acid, display weaker positive correlations with quality, suggesting a modest relationship to higher quality. In contrast, density and total sulfur dioxide show weaker negative correlations, implying that lower values in these properties might be characteristic of higher quality wines.

Experimental Setup

In this experiment, I embark on a comprehensive exploration of machine learning models to predict wine quality. The environment is configured using Python, with dependencies managed through a virtual environment and installed via a requirements file. The exploratory data analysis indicates no missing values in the dataset. The dataset is then split into training and testing sets, maintaining a training size that is 80% of the total data. Four distinct models are configured for the study: Support Vector Machine (SVM), Gradient Boosting, Random Forest, and Decision Tree. Each model is initially tested with default parameters to establish a baseline for accuracy whihch shown in Figure 4. The models are then optimized using Bayesian Optimization to enhance their performance. To refine these models, Bayesian optimization is utilized to search for the optimal set of hyperparameters that maximize the cross-validation accuracy. This involves defining specific ranges for each parameter of the models and integrating preprocessing steps into a pipeline that includes scaling, median imputation, transformation, feature selection using SelectKBest, and dimensionality reduction via PCA. The optimization process for each model is visualized through plots that show the progression of scores over iterations, highlighting the effectiveness of Bayesian optimization in enhancing model accuracy. After determining the best parameters, these are then applied, and the models are reevaluated on the test set to compare the improvements in performance. Key results include increases in accuracy for each model after hyperparameter tuning, confirmed by a series of visual comparisons between the initial and tuned model performances. Bayesian optimization is used to fine-tune a range of hyperparameters for each model to optimize performance. For the Support Vector Machine (SVM), the parameters C, which controls the penalty of the error term, is varied from 1 to 100, and k, the number of features to select with SelectKBest, ranges from 9 to 11. These parameters help in adjusting the flexibility and feature selection strength of the SVM. The Gradient Boosting model's optimization process involves adjusting *n_estimators* (100 to 400), which specifies the number of boosting stages that will be run, thus affecting the complex-

Table 1: *Wine Quality Dataset Structure.*

Attribute	Description	Data Type
Fixed Acidity	Most acids involved with wine	float64
Volatile Acidity	Amount of acetic acid in wine	float64
Citric Acid	Found in small quantities, citric acid	float64
Residual Sugar	Amount of sugar remaining after fermentation	float64
Chlorides	Amount of salt in the wine	float64
Free Sulfur Dioxide	Free form of SO2; prevents microbial growth	float64
Total Sulfur Dioxide	Amount of free and bound forms of S02	float64
Density	Density of the wine	float64
pH	Describes the acidity or basicity of wine	float64
Sulphates	Wine additive which can contribute to sulfur dioxide gas (S02) levels	float64
Alcohol	Alcohol content of the wine	float64
Quality	Score between 3 and 8 (inclusive)	int64

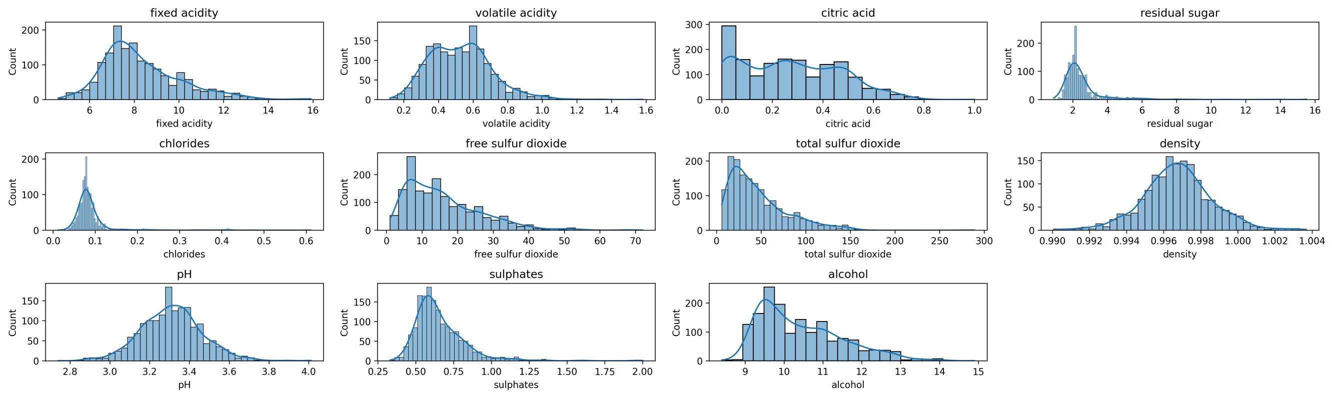

Figure 2: *This figure shows the distribution of the dataset for each attribute in the wine quality dataset.*

ity and potential overfitting. The *max_depth* (3 to 10) controls the maximum depth of the individual regression estimators, influencing how the model handles the trade-off between bias and variance. The *learning_rate* (0.01 to 0.3) is used to shrink the contribution of each tree, preventing overfitting by learning gradually. For the Random Forest classifier, hyperparameters include *n_estimators* (100 to 400), setting the number of trees in the forest and therefore the model robustness. The *max_depth* (3 to 10) limits the depth of each tree, and *min_samples_split* (2 to 10) defines the minimum number of samples required to split an internal node. The *min_samples_leaf* (1 to 4) specifies the minimum number of samples a leaf node must have, which can affect the smoothness and generalization of the model. Finally, the Decision Tree model is tuned over parameters such as *max_depth* (3 to 20), which affects the complexity of the decision paths; *min_samples_split* (3 to 20), determining the conditions under which internal nodes can be divided; and *min_samples_leaf* (3 to 8), influencing the minimum size of leaf nodes that controls overfitting by making the rules less detailed.Each model's performance is rigorously evaluated using a 5-fold cross-validation scheme within the training dataset, employing accuracy as the primary metric for assessing improvements. The optimization process for each model is visually repre-

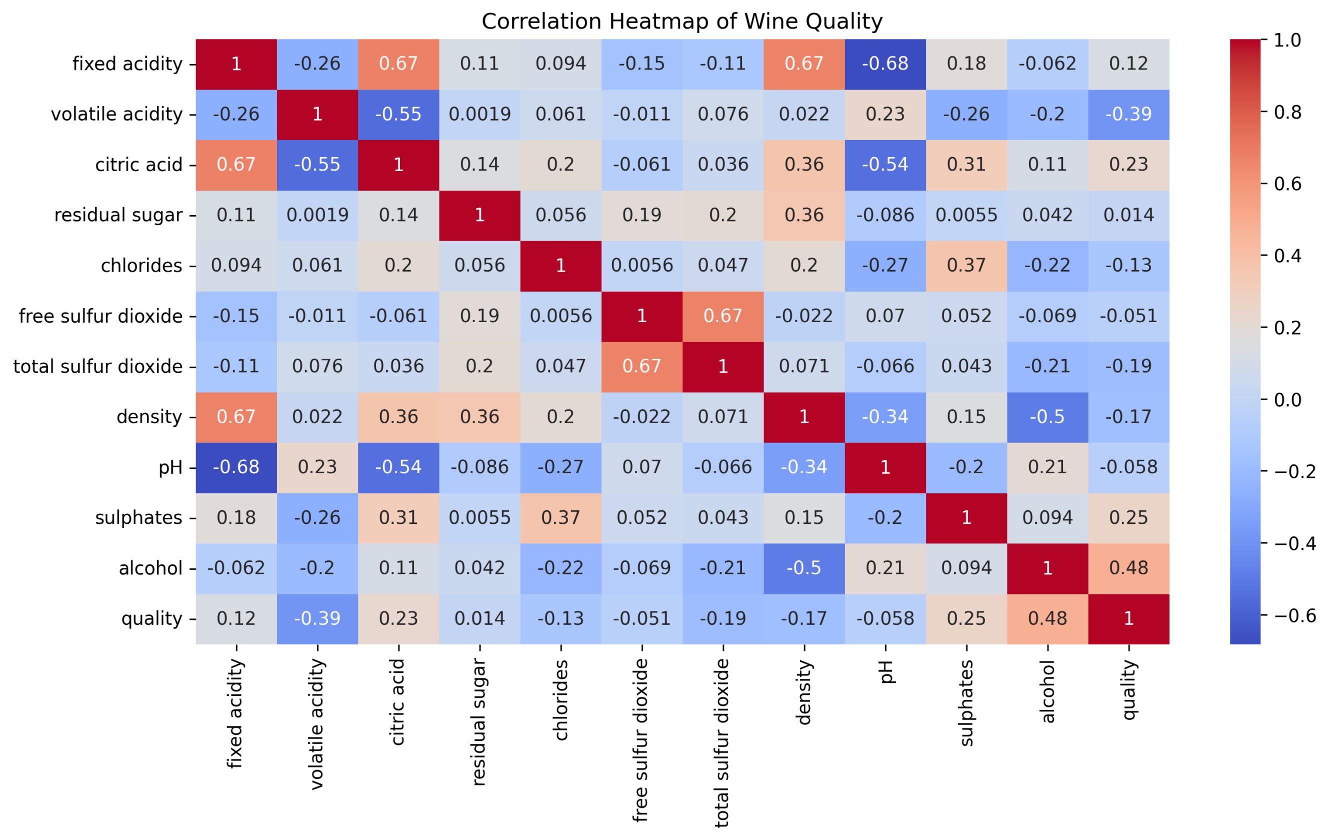

Figure 3: *correlation heatmap of wine quality dataset attributes.*

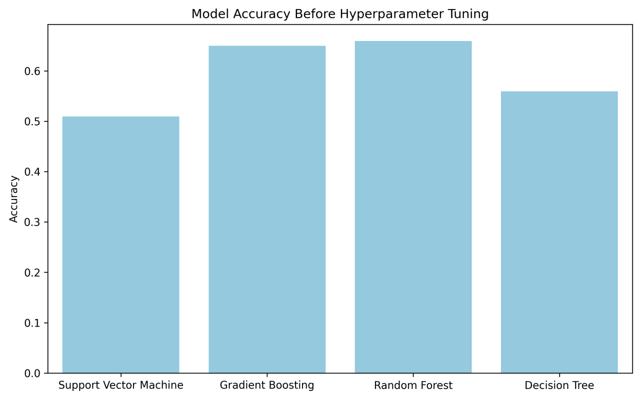

Figure 4: *Model accuracy before hyperparameters tuning.*

sented through plots that depict the evolution of scores over iterations, highlighting the effectiveness of Bayesian optimization in refining model accuracy. After identifying the best parameters, they are applied to re-evaluate the models on the test set, demonstrating significant performance improvements compared to the baseline.

Reults

Support Vector Machine

The Support Vector Machine (SVM) model underwent several stages of optimization and assessment, yielding varying accuracy levels throughout the process. Initially, before any hyperparameter tuning or pipeline implementation, the SVM achieved an accuracy

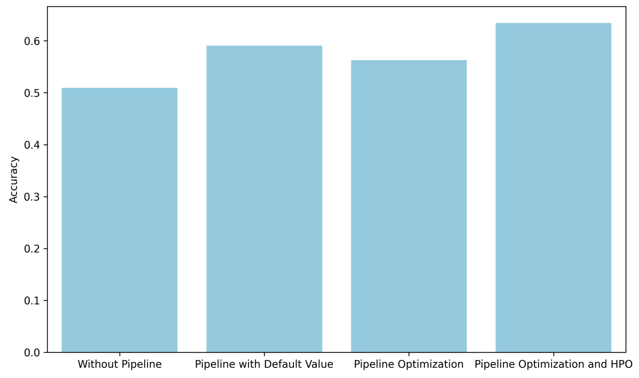

Figure 5: *Support Vector Machine accuracy before and after hyperparameters tuning.*

of approximately 50.94%. This performance was improved to 59.06% when the SVM was re-evaluated with a default preprocessing pipeline that included scaling, imputation, and feature transformation. Further enhancements were made through pipeline parameter optimization based on validation data, which increased accuracy to about 64.82%. However, when tested, the pipeline optimization alone yielded a lower accuracy of 56.25%. Incorporating both hyperparameter tuning and pipeline optimization led to higher accuracies, reaching 62.24% on validation data and 63.44% in the final testing phase. These stages show a clear progression and improvement in model performance, reflecting the effectiveness of systematic tuning and preprocessing in enhancing the SVM's predictive capabilities (Figure 5).

Gradient Boosting

The Gradient Boosting model showed an interesting pattern in its performance through various stages of optimization and pipeline integration. Initially, the model started with a baseline accuracy of 65% before any tuning or adjustments. This performance slightly dropped to 61.88% after implementing a default preprocessing pipeline, possibly indicating that the initial settings may have been more aligned with the raw data. Subsequent pipeline parameter optimization based on validation data slightly improved the model's accuracy to 64.43%, but when this optimized pipeline was tested, the accuracy decreased to 56.25%. However, the most significant improvements were observed after combining hyperparameter tuning with pipeline optimization. This comprehensive approach led to an accuracy of 66.38% on validation data and closely matched this with a final test accuracy of 66.25%. These results underscore the effectiveness of finely tuned preprocessing and model parameters in enhancing the predictive power of Gradient Boosting (Figure 6).

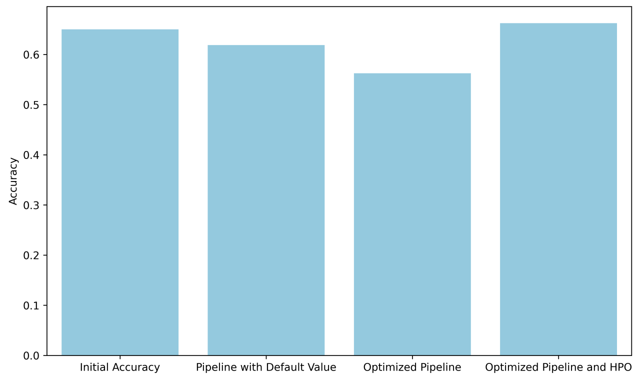

Figure 6: *Gradient Boosting accuracy before and after hyperparameters tuning.*

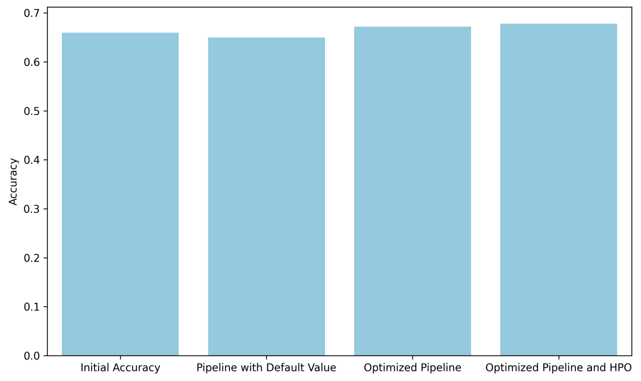

Figure 7: *Random Forest accuracy before and after hyperparameters tuning.*

Random Forest

The Random Forest model's performance trajectory exhibits gradual enhancements at various stages of tuning and optimization. Initially, the model displayed a baseline accuracy of 65.94% without any hyperparameter tuning or preprocessing interventions. A slight decline in accuracy to 65% was observed after the integration of a default preprocessing pipeline.

Upon implementing pipeline parameter optimization based on validation dataset feedback, accuracy improved to 67.95%. This indicates that tailored preprocessing steps helped the model better capture the underlying patterns in the data. Testing this pipeline-optimized model on unseen data yielded a slightly lower, yet robust, accuracy of 67.19%, demonstrating the model's effectiveness and generalization capabilities.

Further enhancements were achieved by combining hyperparameter tuning with pipeline optimization, where the model reached an accuracy of 67.17% on validation data, closely mirrored by a final accuracy of 67.81% on the test set. This incremental improvement highlights the benefits of a finely adjusted model and preprocessing pipeline, ensuring that the Random Forest classifier maintains strong predictive performance and generalization across different data subsets (Figure 7).

Decision Tree

The Decision Tree model's performance through various stages of optimization reflects a mixture of improvements and consistent outcomes. Initially, the model started with a baseline accuracy of 55.94%. After integrating a default preprocessing pipeline, a slight increase in accuracy to 56.56% was observed, indicating minor improvements from initial preprocessing steps such as scaling and imputation.

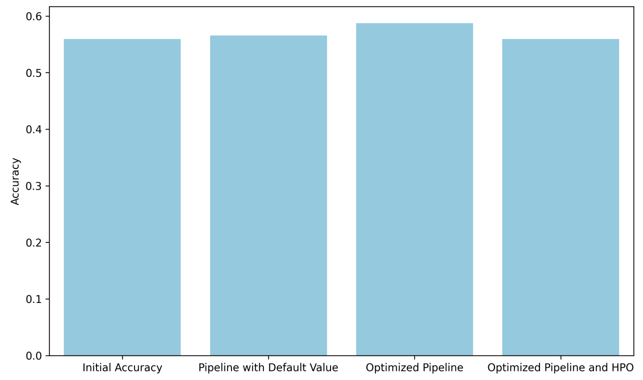

Figure 8: *Decision Tree accuracy before and after hyperparameters tuning.*

Significant enhancement was achieved with pipeline parameter optimization based on validation data, where the accuracy jumped to 62.47%. This improvement suggests that the tailored preprocessing and feature selection better suited the Decision Tree's structure, helping it to capture more nuanced patterns in the data. However, when this pipeline-optimized model was tested on unseen data, the accuracy decreased to 58.75%, implying some overfitting to the validation set.

The subsequent stage involved hyperparameter tuning combined with pipeline optimization, where the model reached an accuracy of 60.44% on validation data. Surprisingly, when this fully optimized model was assessed on the test dataset, the accuracy reverted to its initial baseline of 55.94%. This outcome highlights potential challenges in the model's ability to generalize beyond the training data, despite the optimizations, and underscores the Decision Tree's sensitivity to specific data distributions and parameter settings (Figure 8).

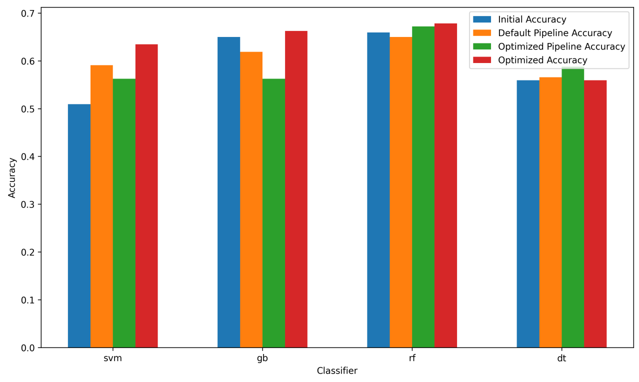

Figure 9: *Model accuracy before and after hyperparameters tuning.*

Conclusion

ChatGPT

In this report, we have demonstrated a robust machine learning pipeline optimization using Bayesian Optimization, specifically tailored for enhancing model performance. This methodology has been systematically applied to a red wine dataset, showcasing its capability to significantly improve model accuracy across several algorithms, including Support Vector Machine, Gradient Boosting, Random Forest, and Decision Tree. Our approach not only involves fine-tuning the hyperparameters but also meticulously optimizing the preprocessing steps integral to the pipeline. The resulting improvements in model accuracy highlight the potential of Bayesian Optimization to refine model parameters and adapt preprocessing techniques to maximize predictive performance. Comprehensive results and comparisons are detailed in Figure 9, which visually summarizes the enhancements achieved at various stages of the optimization process. This analysis confirms the effectiveness of our optimization strategy in elevating the predictive accuracy of traditional machine learning models on complex datasets.