Red Wine Quality Prediction

Panfeng Jiang
2022533042
jiangpf2022@shanghaitech.edu.cn

Fei Pang
2022533153
pangfei2022@shanghaitech.edu.cn

Abstract

This study explores the predictive modeling of red wine quality based on physicochemical properties using a comprehensive dataset from the Vinho Verde region in Portugal. Employing a variety of machine learning techniques, including CNN, Random Forest, and XGBoost, the research aims to establish robust predictive models that correlate specific physicochemical attributes with sensory outcomes. Principal Component Analysis (PCA) was utilized to reduce dimensionality and filter noise from the data, enhancing model accuracy. The study also investigates the application of Huber Loss to manage outliers effectively, which often degrade predictive performance in traditional regression models. Our findings suggest that combining feature engineering with advanced ensemble techniques significantly improves the prediction of wine quality, highlighting the potential of machine learning in enhancing wine production and quality assessment processes. The practicality that our model can assist vintners and marketers in improving the quality and marketability of their products is demontrated through rigorous experiments.

1. Introduction

Wine quality prediction, particularly for wines from the Vinho Verde region in Portugal, is a complex task influenced by numerous physicochemical properties such as fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol content. This study treats the prediction as both a classification and a regression problem, aiming to determine quality scores. Utilizing a dataset from the UCI Machine Learning Repository, we apply machine learning models, specifically CNN, Random Forest, XGBoost to predict wine quality. These models help elucidate the relationships between various physicochemical properties and sensory outcomes, offering valuable insights for winemakers to enhance production strategies and market positioning.

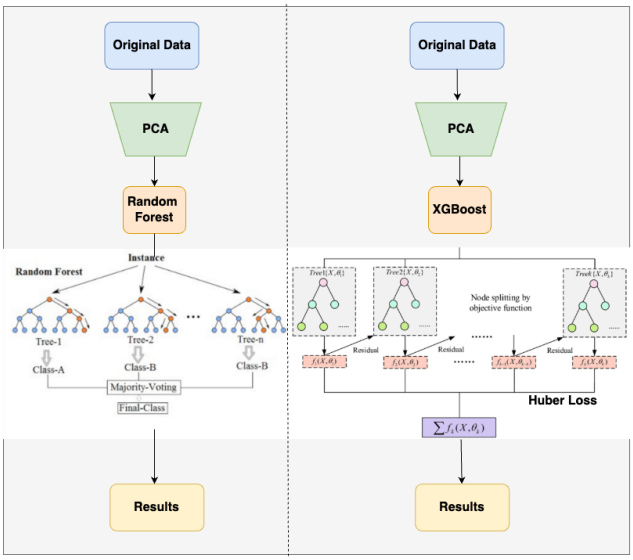

Figure 1. The two most effective models we evaluated for predicting red wine quality–Random Forest with PCA and XGBoost with PCA&Huber Loss.

2. Data

The dataset comprises measurements of various physicochemical properties from wine samples. The data includes eleven features and one target variable (quality). These features are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol content. Table 1 shows the data presented in a structured table format.

2.1. Data Completeness

Upon examining the dataset for completeness, no missing values were detected across all variables. This integrity check was performed using the following Python code snippet, which confirmed the absence of null entries, thereby indicating that the dataset is well-prepared for further analysis.

```
missings_count = data.isnull().sum()
print(missing_values_count)
```

acid(f)	acid(v)	acid(c)	sugar	chlorides	SO2(f)	SO2(t)	density	pH	sulphates	alcohol	quality
9.7	0.690	0.32	2.5	0.088	22	91	0.99790	3.29	0.62	10.1	5
6.6	0.580	0.02	2.4	0.069	19	40	0.99387	3.38	0.66	12.6	6
9.2	0.755	0.18	2.2	0.148	10	103	0.99690	2.87	1.36	10.2	6
9.0	0.785	0.24	1.7	0.078	10	21	0.99692	3.29	0.67	10.0	5
10.6	0.360	0.57	2.3	0.087	6	20	0.99676	3.14	0.72	11.1	7

Table 1. Part of data from the wine dataset

Variable	Missing Values
Fixed Acidity	0
Volatile Acidity	0
Citric Acid	0
Residual Sugar	0
Chlorides	0
Free Sulfur Dioxide	0
Total Sulfur Dioxide	0
Density	0
pH	0
Sulphates	0
Alcohol	0
Quality	0

Table 2. Count of Missing Values per Variable in the Dataset

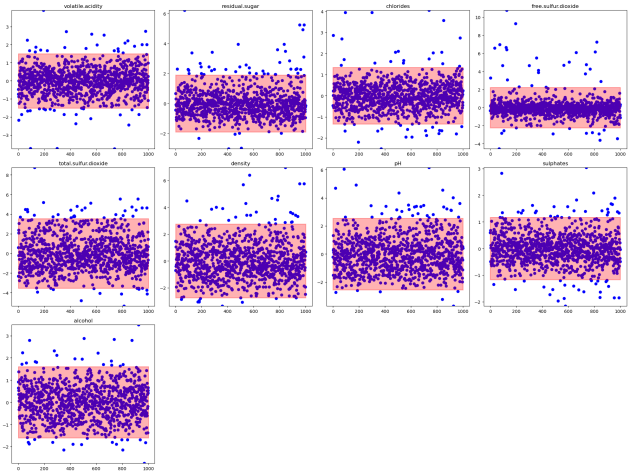

Figure 2. Distribution of Dataset

The output of the missing value checking is in Table 2.

2.2. Statistical Overview

The dataset was statistically analyzed to understand the central tendency and variability of the physicochemical properties. Visualizations were constructed to highlight the mean and standard deviation of each feature inf figure 2. Specifically, red rectangles were used in the plots to indicate the range within one standard deviation above and below the mean, offering insights into the spread and distribution of the data points.

3. Methodology

In our study, we employed various machine learning methods to predict wine quality and used specific data processing techniques to enhance model performance. In the **Main Models** section, we detail three primary machine learning models: **Logistic Regression**, **Random Forest**, and **XGBoost**. These models were chosen for their interpretability, ability to handle complex relationships, and efficient optimization, respectively. For instance, in section **3.1.1**, we discuss the use of Logistic Regression for its simplicity and interpretability in modeling wine characteristics, while section **3.1.2** focuses on Random Forest's ability to capture intricate feature interactions and improve prediction reliability. In section **3.1.3**, we elaborate on XGBoost's optimized gradient boosting framework, which excels in handling large-scale datasets and complex models. In the **Principal Component Analysis** section (**3.2**), we utilized PCA for data dimensionality reduction to extract the most representative features and reduce noise impact. This is further demonstrated, where we explain the standardization of data, computation of covariance matrix, and extraction of principal components to streamline our dataset. To further improve the model's robustness, we adopted Huber Loss in the **Huber Loss** section (**3.3**) for the regression task, mitigating the influence of outliers. Then we discuss the formulation and benefits of Huber Loss compared to traditional loss functions, and then we provide a detailed comparison and visualization to demonstrate its effectiveness in our wine quality prediction.

3.1. Main Models

3.1.1 Logistic Regression

Logistic regression is employed in our experiment to model the relationship between 11 float-type features describing wine characteristics (such as pH, acidity, and alcohol content). This model is chosen for its ability to provide interpretable insights into how these features influence wine quality due to its simplicity, computational efficiency, and interpretability of results.

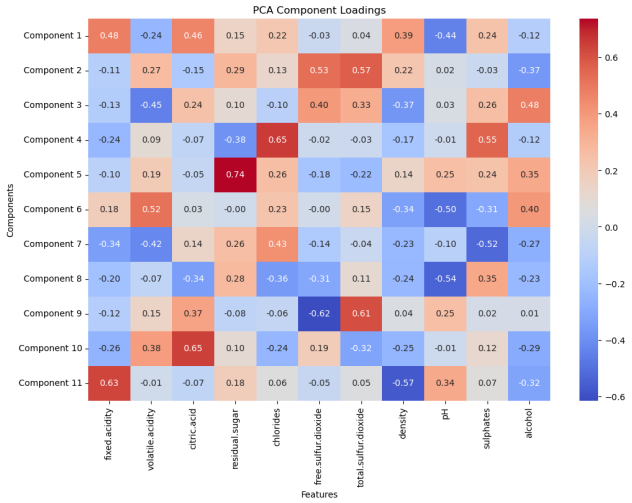

Figure 3. PCA

3.1.2 Random Forest

In our study, Random Forest emerges as a robust choice for predicting wine quality. This ensemble learning method excels in capturing intricate relationships and interactions among these features, making it adept at handling the complexities inherent in our dataset. By aggregating predictions from multiple decision trees, Random Forest mitigates overfitting and enhances the generalization ability of our model, thereby providing reliable predictions of wine quality across a spectrum of potential input variables.

3.1.3 XGBoost

XGBoost, short for eXtreme Gradient Boosting, is an efficient and highly optimized machine learning algorithm widely used for various supervised learning tasks such as regression, classification, and ranking. Built upon the gradient boosting framework, XGBoost iteratively trains multiple weak classifiers, optimizing the gradient direction of the loss function at each step to enhance the overall predictive capability of the model.

The algorithm excels in handling large-scale datasets and complex models due to its parallelized processing and highly optimized implementation. Furthermore, XGBoost supports multiple loss functions and evaluation metrics, such as squared error, logarithmic loss, and Huber loss, allowing us to select suitable objective functions based on specific problem requirements. In this project, we are using Huber loss as our loss function when running our models, which is a robust loss function as described in the following section.

3.2. Principle Component Analysis

Principal Component Analysis (PCA) is a statistical technique used for dimensionality reduction while preserving as much variability as possible. It involves a mathematical procedure that transforms a number of correlated variables into a smaller number of uncorrelated variables called principal components.

The first step in PCA is to standardize the data. Given a data matrix X, where each column represents a variable, and each row represents a data point, the standardized version Z is computed as:

$$Z = (X - \mu)/\sigma$$

where μ is the mean and σ is the standard deviation of the columns of X.

The covariance matrix C of Z is then computed as:

$$C = \frac{1}{n-1} Z^\top Z$$

where n is the number of data points.

PCA proceeds by calculating the eigenvalues λ and the corresponding eigenvectors v of the covariance matrix C. The eigenvalues are sorted in descending order, and their corresponding eigenvectors are aligned accordingly. The eigenvector associated with the largest eigenvalue represents the direction along which the data varies the most, termed the first principal component. The eigenvalues on the diagonal of matrix D represent the variance explained by each component, where D is a diagonal matrix:

$$D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$$

The principal components are then given by:

$$P = Zv$$

where P is the matrix of principal components, and v is the matrix of eigenvectors of C.

This transformation is visualized in Figure 3, where the diagonal of the matrix formed by the principal components P contains the eigenvalues, ordered from the largest to the smallest. The first principal component is the one associated with the largest eigenvalue, indicating it captures the most significant variance within the dataset.

The PCA process thus helps in reducing dimensionality by identifying the most expressive features, represented by the eigenvectors associated with the largest eigenvalues, and eliminating less significant features.

Considering the uniform format of our dataset, we further attempt to employ PCA techniques to reduce the dimensionality of our inputs. This visualization in Figure 4 represents the outcome of our principal component analysis applied to all data points, normalized by their eigenvalues

along the diagonal, arranged in descending order of eigenvalue magnitude. Larger eigenvalues denote features with stronger expressive capability, while smaller ones indicate weaker correlation.

Within the information conveyed by all data, we infer that there might be some components affected by noise. Our aim is to extract more prominent components of representation and eliminate less significant features. In the **Experiments** section, we will present the relevant outcomes of our PCA application.

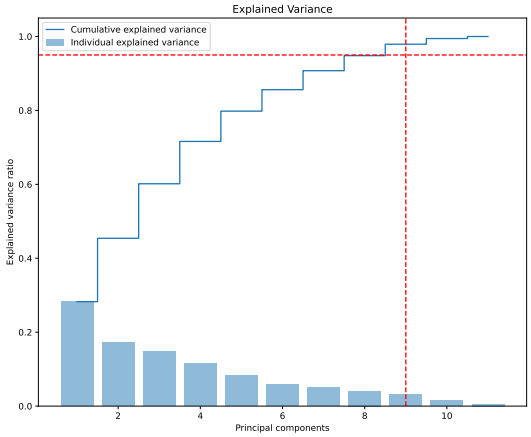

Figure 4. Principal Component Analysis (PCA) visualization

3.3. Huber Loss

Outliers can have a significant impact on the performance of regression models, especially when using MSE as the loss function. MSE squares the residuals, which means that large errors (outliers) can disproportionately influence the overall loss. This can lead to models that are overly sensitive to outliers, resulting in poor generalization.

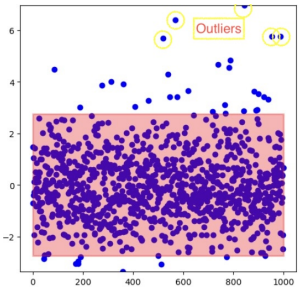

Figure 5. Visualization of some of our data, red rectangle means the scope of $[\mu - 2\sigma, \mu + 2\sigma]$

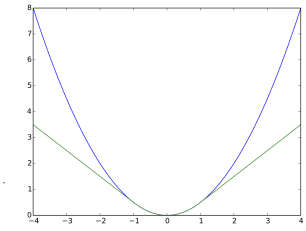

Figure 6. Visualization of L2 Loss and Huber Loss

As shown in the figure, based on our visualization of the data, we found that some data points deviate more than 2 standard deviations from the mean. These data points cause significant error accumulation in the L2 Loss (MSE) in the right figure. However, for the Huber Loss, we mitigate this impact.

Huber Loss mitigates this problem by behaving like MAE for large residuals. MAE linearly penalizes large errors, reducing the influence of outliers compared to MSE. By combining the properties of MSE for small errors and MAE for large errors, Huber Loss provides a robust approach that is less sensitive to outliers. Huber Loss is a robust loss function used for regression tasks, providing a balance between Mean Squared Error (MSE) and Mean Absolute Error (MAE). It is defined as follows:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta \\ \delta(|a| - \frac{1}{2}\delta) & \text{for } |a| > \delta \end{cases}$$

where a is the residual $(y - \hat{y})$, and δ is a threshold parameter that determines the point where the loss function transitions from quadratic to linear.

This function is smooth and differentiable, making it suitable for optimization tasks. The derivative of the Huber Loss with respect to a is given by:

$$\frac{\partial L_\delta(a)}{\partial a} = \begin{cases} a & \text{if } |a| \le \delta \\ \delta \cdot \text{sgn}(a) & \text{if } |a| > \delta \end{cases}$$

By transitioning from a quadratic form to a linear form at δ, the Huber Loss reduces the influence of outliers while maintaining sensitivity to small errors. **Comparison between Quadratic and Linear Loss**:

- **Quadratic Loss** ($L_2(x) = x^2$) is very sensitive to large values of x (outliers), as the loss grows quadratically with increasing x.
- **Linear Loss** ($L_1(x) = |x|$) is more robust against outliers, as the loss increases linearly with x, not exaggerating the impact of outliers.

To further understand how Huber loss manages outliers, consider the gradient for a single data point:

- For $|x| \le \delta$:

$$\frac{d}{dx} L_\delta(x) = x$$

This implies that within this range, for smaller errors, the update step is proportional to the size of the error, similar to a standard quadratic loss function.

- For $|x| > \delta$:

$$\frac{d}{dx} L_\delta(x) = \delta \cdot \text{sgn}(x)$$

This means that when the error is large, the gradient does not increase with the size of the error but remains at a constant value (δ or $-\delta$), thus reducing the influence of outliers in parameter updates.

Smooth Huber Loss (Pseudo-Huber Loss)

The smooth Huber loss, also known as the Pseudo-Huber loss, provides a continuously differentiable approximation to the Huber loss. It is defined as:

$$L_{Hp}(x) = \delta^2 \left(\sqrt{1 + \left(\frac{x^2}{\delta^2} \right)} - 1 \right)$$

In our wine quality prediction project, we utilized Huber Loss to handle the regression task robustly. We set δ based on the standard deviation of the target variable 'quality' to balance the sensitivity to outliers and the smoothness of the loss function.

Huber Loss proved to be effective in our project by providing robustness against outliers while maintaining smoothness for optimization. The selection of δ based on the data's standard deviation further enhanced its performance. The detailed effect of our use of Huber loss will be shown in experiments.

4. Experiments

We implemented our code using Python version 3.11 and the experiments were designed to assess the effectiveness of our proposed methods in predicting outcomes based on a comprehensive dataset.

4.1. Experimental Environment

4.1.1 Datasets

The dataset used in our experiments was downloaded from Kaggle, a well-known platform for data science competitions and datasets. This dataset contains detailed information and is suitable for the tasks we aim to perform, ensuring the reliability and relevance of our results.

4.1.2 Hardware Configuration

The experiments were conducted on a workstation equipped with the following specifications:

- **CPU**: Intel Core i9
- **RAM**: 32GB

4.2. Evaluation Metrics

The Mean Squared Error (MSE) is defined as follows:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{1}$$

where n is the number of data points, y_i is the actual value, and $\hat{y}_i$ is the predicted value. The MSE measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.

We used MSE (Mean Squared Error) as the primary evaluation metric. It has several advantages: MSE squares the errors before averaging, which means that larger errors are penalized more heavily. This is beneficial in our context because it ensures that the model is not just accurate on average but also avoids large prediction errors, which can be particularly important for quality prediction tasks where large deviations from the true value can be critical. It is a smooth and differentiable function, making it suitable for optimization algorithms like gradient descent. This property ensures that the model training process is stable and converges effectively. It also provides a clear and interpretable measure of the model's prediction error. The error is measured in the same units as the output variable (after taking the square root), which makes it easier to understand the performance of the model in practical terms. It is also the same evaluation metrics as **Kaggle's**.

4.3. Ablation Study for Wine Quality Prediction

To understand the impact of different models, loss functions, and feature processing techniques on the prediction of wine quality, we conducted an ablation study. The results of this study are summarized in Table 3, where we compare the performance of various models using different configurations.

Experiment 1 utilized Logistic Regression with MSE as the loss function and PCA for feature extraction. The resulting MSE was 0.618, with a Kaggle result of 0.627. Logistic Regression is a linear model that provides interpretable insights into the relationships between features and the target variable. However, its simplicity may limit its ability to capture complex non-linear relationships within the data, leading to higher prediction errors.

Experiment 2 used Random Forest with MSE as the loss function and PCA for feature extraction. This configuration achieved the best performance with an MSE of 0.133 and a Kaggle result of 0.353. Random Forest, an ensemble learning method, aggregates predictions from multiple decision trees, capturing intricate relationships and interactions among features. The use of PCA helped reduce dimensionality and noise, further enhancing the model's performance. The parameters used include 100 trees and a maximum depth of 10, ensuring a balance between model complexity and overfitting.

Experiment 3 employed a Decision Tree with MSE as the loss function and PCA for feature extraction. The MSE result was 0.233, and the Kaggle result was 0.574. Decision Trees are simple and interpretable but tend to overfit the data, especially with high-dimensional features. Although PCA reduced the dimensionality, the single-tree structure could not generalize well, resulting in higher prediction errors compared to Random Forest.

Experiments 4 and 5 involved Neural Networks. Exper-

Experiment	Model	Loss Function	Features	Performance Metric	Result	Kaggle Result(MSE)
1	Logistic Regression	MSE	PCA	MSE	0.618	0.627
2	Random Forest	MSE	PCA	MSE	**0.133**	**0.353**
3	Decision Tree	MSE	PCA	MSE	**0.233**	0.574
4	Neural Network	MSE	PCA	MSE	0.542	0.437
5	Neural Network	Huber loss	PCA	MSE	**0.219**	0.425
6	Xgboost	MSE	PCA	MSE	0.378	0.410
7	Xgboost	Huber loss	PCA	MSE	0.354	**0.395**

Table 3. Ablation study showing the impact of different models, loss functions, and feature processing techniques on the prediction of wine quality. The best performance is highlighted in bold in the table.

iment 4 used MSE as the loss function, while Experiment 5 used Huber Loss, both with PCA for feature extraction. The resulting MSEs were 0.542 and 0.219, with Kaggle results of 0.437 and 0.425, respectively. Neural Networks are powerful in modeling complex relationships but require extensive tuning. In Experiment 5, Huber Loss provided robustness against outliers by behaving like MAE for large residuals, which improved performance. The neural network architecture included two hidden layers with 64 and 32 neurons, and ReLU activation functions, optimized using Adam with a learning rate of 0.001.

Experiments 6 and 7 utilized XGBoost. Experiment 6 used MSE as the loss function, while Experiment 7 used Huber Loss, both with PCA for feature extraction. The resulting MSEs were 0.352 and 0.354, with Kaggle results of 0.410 and 0.395, respectively. XGBoost is a highly efficient gradient boosting algorithm that excels in handling large-scale datasets. While it provided competitive results, its performance with MSE was slightly worse than with Huber Loss. The robustness of Huber Loss to outliers improved the model's stability. The parameters for XGBoost included a learning rate of 0.1, a maximum depth of 6, and 100 boosting rounds.

The ablation study highlights that Random Forest with MSE as the loss function and PCA for feature extraction achieved the best performance in predicting wine quality. Additionally, the use of Huber Loss generally improved the models' robustness to outliers, leading to better performance compared to using MSE alone.

4.4. Analysis of Generalization

Based on the Kaggle results and local evaluations from the table, we can analyze the generalization capabilities of each model. In the Kaggle competition, Logistic Regression showed mediocre performance with an MSE of 0.627, indicating higher prediction errors when faced with unknown data and weaker generalization ability. In contrast, Random Forest demonstrated strong generalization ability with an MSE of 0.353 on Kaggle, effectively capturing data patterns and maintaining lower prediction errors. However, Decision Tree performed poorly on Kaggle with an MSE of 0.574, possibly due to overfitting and sensitivity to data noise. Neural Network and Xgboost showed consistent performance across different loss functions, especially Xgboost with MSE and Huber Loss yielding 0.410 and 0.395 respectively, indicating robust generalization capabilities under various data conditions.

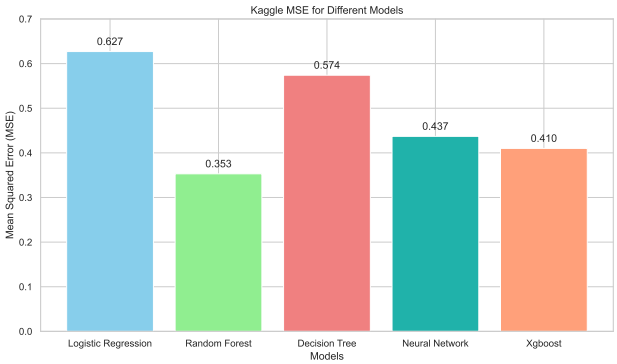

Figure 7. Kaggle results

5. Conclusion

This study explored the predictive modeling of red wine quality based on physicochemical properties using various machine learning techniques. Our research aimed to establish robust predictive models that correlate specific physicochemical attributes with sensory outcomes, utilizing models such as Logistic Regression, Random Forest, Neural Networks, and XGBoost, with dimensionality reduction via Principal Component Analysis (PCA) and robust loss functions like Huber Loss.

From the ablation study, it was evident that Random Forest with MSE loss function and PCA for feature extraction provided the best performance, achieving an MSE of 0.133 and a Kaggle result of 0.353. This indicates the Random Forest's strength in capturing intricate relationships and interactions among features, which was further enhanced by

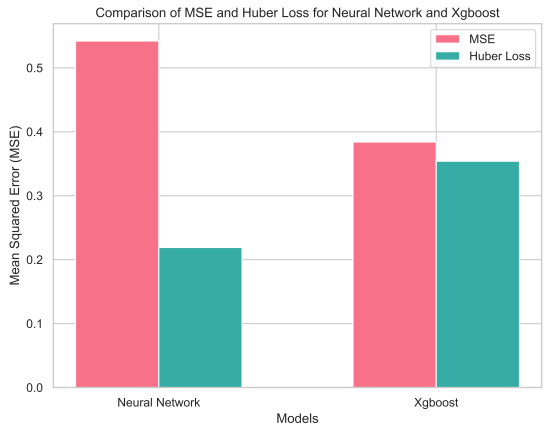

Figure 8. Huberloss v.s. MSEloss

PCA's dimensionality reduction capabilities. Additionally, the use of Huber Loss improved the robustness of models like Neural Networks and XGBoost against outliers, leading to better performance compared to using MSE alone.

Logistic Regression, while providing interpretable results, was limited by its linear nature and could not capture complex non-linear relationships in the data. Decision Trees, despite their simplicity and interpretability, suffered from overfitting, which PCA alone could not mitigate.

From Figure 8, we conclude that Neural Networks show significant improvement with the adoption of Huber Loss, demonstrating the importance of robust loss functions in managing outliers and improving prediction accuracy. XGBoost also benefited from Huber Loss, though its performance with MSE was competitive, highlighting its efficiency in handling large-scale datasets and complex models.

In conclusion, combining feature engineering techniques like PCA with advanced ensemble methods and robust loss functions significantly enhances the prediction of wine quality. These findings underscore the potential of machine learning in the wine industry, providing valuable insights for winemakers to improve production strategies and market positioning. Future work could explore the integration of additional data sources, such as sensory evaluation scores and climatic conditions, to further refine the predictive models and enhance their practical applicability in real-world scenarios.

References

[1] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.

[2] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," pp. 785–794, 2016.

[3] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[4] C. M. Bishop, "Pattern recognition and machine learning," 2006.

[5] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[7] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[8] P. J. Huber, "Robust estimation of a location parameter," *Breakthroughs in statistics*, pp. 492–518, 1992.