PREDICTING RED WINE QUALITY USING MACHINE LEARNING

Zhoujun Cao, Yiying Wang, Bo Peng

ABSTRACT

This project explores the application of machine learning to predict the quality of red wine based on physicochemical attributes. Using a dataset of 1,599 samples from the Vinho Verde region of Portugal, we preprocess the data, select key features, and apply various machine learning models. These include Random Forest, Support Vector Machine (SVM), Gradient Boosting, Artificial Neural Networks (ANN), and an ensemble Voting Classifier. Key techniques include hyperparameter tuning and feature selection to enhance performance. The Voting Classifier achieved the best results, with 87.5% accuracy and 86.66% precision, highlighting the value of ensemble learning for wine quality prediction. This study demonstrates the potential of machine learning in the wine industry to improve quality assessment and decision-making.

Index Terms— Random Forest, SVM, Gradient Boosting, ANN, ensemble Voting Classifier,

1. INTRODUCTION

Machine learning has become a powerful tool for solving complex problems by enabling systems to learn patterns from data and make accurate predictions. Its applications span numerous fields, including healthcare, finance, and manufacturing, where it has been used to optimize processes, classify data, and uncover valuable insights. One critical area where machine learning shows great promise is in the food and beverage industry, where it can be employed to assess quality, predict outcomes, and streamline production.

In this project, we focus on applying machine learning techniques to predict the quality of red wine based on its physicochemical attributes. The dataset, obtained from the UCI Machine Learning Repository [1], contains 1599 red wine samples from the Vinho Verde region of Portugal. By analyzing this dataset, we aim to explore how machine learning can accurately classify wine quality and identify the key factors that influence it. This study not only demonstrates the practical applications of machine learning in wine quality assessment but also highlights its potential for improving decision-making in the wine industry.

The goals of this project are:

- To preprocess and clean the data to improve dataset quality and enhance model performance.
- To implement feature selection methods to reduce dimensionality while preserving key predictive variables.
- To build and optimize machine learning models for classifying wine quality, including hyperparameter tuning to achieve the best possible performance.
- To evaluate model performance using metrics such as accuracy, precision, recall, and F1-score.

The methods used in this project include data preprocessing techniques such as standardization and outlier detection, feature selection using Pearson correlation and multicollinearity analysis, and model development using advanced machine learning algorithms such as Random Forest, Support Vector Machine (SVM), Gradient Boosting, Artificial Neural Networks (ANN), and an ensemble Voting Classifier. Hyperparameter tuning was conducted using grid search with cross-validation to ensure optimal performance.

As a result, the project achieved significant improvements in model accuracy and precision, with the Voting Classifier emerging as the most effective method. These results demonstrate the value of systematic feature selection and ensemble learning in building robust predictive models for wine quality classification.

2. DATA PREPROCESSING

2.1. Dataset Description

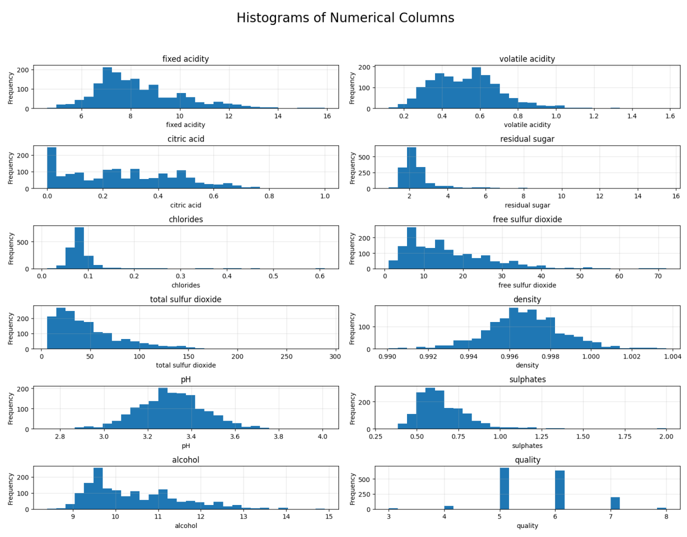

Fig. 1. Histograms of Numerical Columns

This dataset contains 1599 red wine samples from the Vinho Verde region of Portugal, characterized by 11 physicochemical attributes and a sensory quality score ranging from 3 to 8. The physicochemical attributes include fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. The target variable, sensory quality, was determined through expert blind tastings.

Figure 1 shows the distributions of these attributes, most of which exhibit distinct patterns. Sensory quality scores demonstrate moderate imbalance, with the majority of wines rated between 5 and 6, indicating that there are many more normal wines than excellent or poor ones.

The original distribution of scores showed that most wines had a quality score of 5 (681 samples) or 6 (638 samples), while fewer samples were rated 7 (199 samples), 4 (53 samples), 8 (18 samples), and 3 (10 samples). To simplify the classification task and better analyze the data, we grouped the quality scores into three categories: low quality (scores of 3 and 4), medium quality (scores of 5 and 6), and high quality (scores of 7 and 8). This reclassification resulted in 63 samples for low quality, 1319 samples for medium quality, and 217 samples for high quality. This grouping is both logical and reflective of real-world wine quality distributions, where medium-quality wines are most common, while high-quality and low-quality wines are relatively rare. The prevalence of medium-quality wines aligns with market trends, as most wines are produced to meet average standards, while high-quality wines are less frequent due to higher production costs and expertise, and low-quality wines are similarly limited due to lower demand. This restructuring simplifies the analysis and effectively captures the practical distribution of wine quality. Figure 2 shows the distribution of the wine quality.

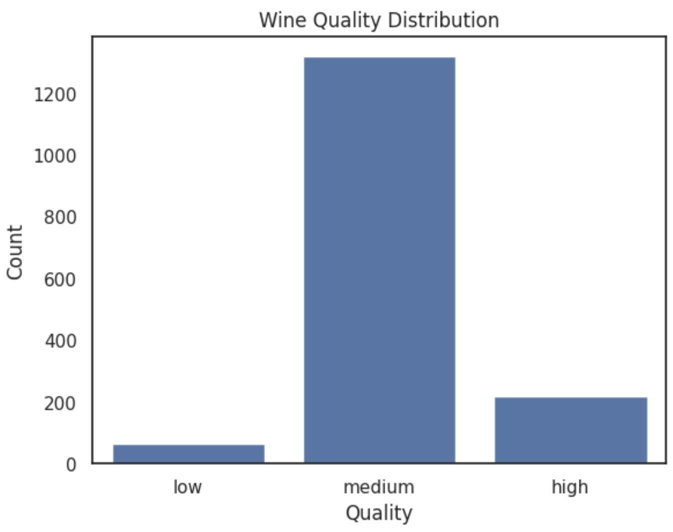

Fig. 2. Distribution of the Wine Quality

2.2. Data Cleaning

2.2.1. *Standardization:*

Standardization is a data preprocessing technique used to scale numerical features in a dataset so that they have a mean of 0 and a standard deviation of 1. This transformation ensures that all features contribute equally to the analysis, preventing those with larger magnitudes or units from dominating the results.

2.2.2. *Outlier Detection:*

We performed outlier detection using the IQR (Interquartile Range) method, where thresholds were defined as 2.5 times the IQR beyond the typical range of the data. This approach identified and removed extreme values that significantly deviated from the central distribution. After detecting and removing 200 outliers, the dataset was reduced to 1,399 rows and 13 columns, resulting in a cleaner and more consistent dataset for analysis.

2.2.3. *Feature Selection:*

- **Multicollinearity**: To address multicollinearity, we performed Variance Inflation Factor (VIF) analysis to measure the degree of multicollinearity among numerical features. Features with VIF values exceeding a predefined threshold (e.g., 10) were iteratively removed, starting with the one having the highest VIF, until all remaining features had acceptable VIF levels. Additionally, we analyzed the correlation matrix to identify highly correlated feature pairs and selectively removed features that were less relevant or exhibited higher VIF values. Figure 3 shows the correlation matrix of red wine features. In this case, no features were removed as the VIF values for all features were within acceptable limits, ensuring the dataset was free of excessive multicollinearity and suitable for modeling.

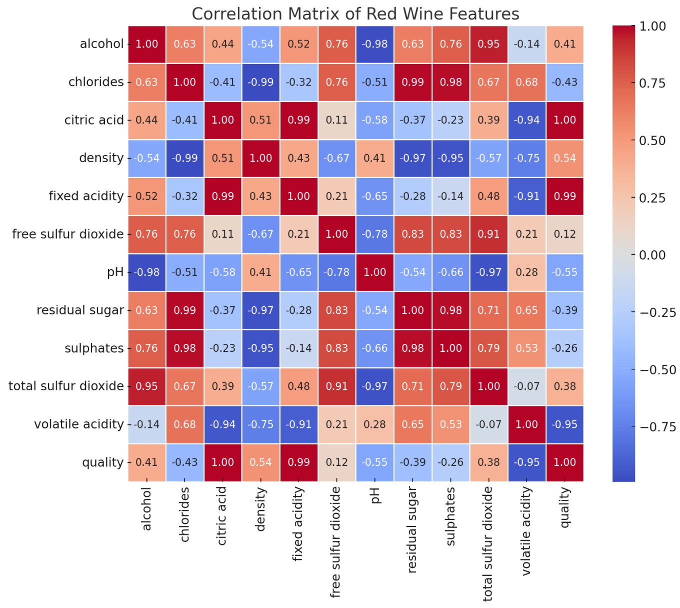

Fig. 3. Correlation Matrix of Red Wine Features

- **Pearson Correlation Analysis**: To evaluate the relationship between features and the target variable, we performed a Pearson correlation analysis. This method calculates the Pearson correlation coefficient and P-value for each feature to assess its linear relationship and statistical significance with the target variable. Features with P-values less than the significance threshold ($\alpha = 0.05$) were considered statistically significant and retained for further analysis. Features with P-values greater than or equal to α were considered insignificant and removed from the dataset.

Feature Selection Results:

- **Significant Features** (P-$value < 0.05$): ['alcohol', 'chlorides', 'citric acid', 'density', 'fixed acidity', 'free sulfur dioxide', 'pH', 'sulphates', 'total sulfur dioxide', 'volatile acidity'].
- **Removed Features** (P-$value \geq 0.05$): ['residual sugar'].

After feature selection, we evaluated model performance using the **Voting Classifier**, and the results showed a significant improvement:

- **Before Feature Selection**:
  Accuracy: 0.8656, Precision: 0.6403, Recall: 0.5364, F1 Score: 0.5597.
- **After Feature Selection**:
  Accuracy: 0.8750, Precision: 0.8666, Recall: 0.5466, F1 Score: 0.5732.

This improvement demonstrates the importance of feature selection. By reducing irrelevant or insignificant features, the **Voting Classifier** achieved a more focused dataset, reduced noise, and better predictive performance across all metrics.

3. METHODOLOGY

In this section, we describe the methodology used to develop and evaluate the machine learning models for wine quality classification. The models applied include Random Forest, Support Vector Machine (SVM), Gradient Boosting, Artificial Neural Networks (ANN), and an ensemble Voting Classifier.

Hyperparameter tuning is crucial for optimizing model performance. All models were tuned using grid search with 5-fold cross-validation, with accuracy as the scoring metric. This approach systematically tests combinations of hyperparameters and selects the configuration that achieves the highest average accuracy across the folds. To address class imbalance, techniques such as `class_weight='balanced'` or resampling were applied where applicable. Key parameters optimized of each model will be listed in each section.

3.1. Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees using Bootstrap sampling, where samples are randomly selected with replacement. Each training subset may include duplicate samples and exclude some original samples, introducing randomness to reduce overfitting. For each subset, a decision tree is trained by splitting nodes based on the feature with the highest Information Gain, which quantifies a feature's ability to reduce uncertainty and improve classification. The Information Gain is computed using entropy, defined as:

$$\text{Entropy}(S) = -\sum_{i=1}^{C} p_i \log_2 p_i, \tag{1}$$

where S is the set of all samples at the current node, p_i is the proportion of samples belonging to class i, and C is the total number of classes. The Information Gain is then calculated as:

$$\text{Gain}(A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v), \tag{2}$$

where A is the feature being evaluated, Values(A) represents the set of all possible values for feature A, S_v is the subset of S where feature A takes the value v, and $|S_v|/|S|$ is the proportion of samples in S that belong to subset S_v. The feature with the highest Information Gain is selected to split the node, as it provides the largest reduction in uncertainty.

After constructing the decision trees, Random Forest predicts the final output using majority voting for classification tasks, where each tree independently provides its classification. Optimized hyperparameters are:

- **Number of estimators (`n_estimators`)**: Number of decision trees in the forest, tested with values [100, 200, 300].
- **Maximum depth (`max_depth`)**: Maximum depth of individual trees, tested with [5, 10, 20].
- **Minimum samples split (`min_samples_split`)**: Minimum number of samples required to split an internal node, tested with [2, 5, 10].
- **Minimum samples leaf (`min_samples_leaf`)**: Minimum number of samples required at a leaf node, tested with [1, 2, 4].

Feature importance was assessed using the `feature_importances` attribute after training, enabling the selection of features exceeding a specified importance threshold.

3.2. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm designed to find an optimal hyperplane for classifying data points into different categories. For binary classification, the decision boundary is expressed as:

$$f(x) = \mathbf{w}^T \phi(x) + b, \tag{3}$$

where $\mathbf{w}$ is the weight vector, b is the bias term, and $\phi(x)$ maps the input $\mathbf{x}$ into a higher-dimensional space. SVM aims to maximize the margin, defined as the distance between the hyperplane and the nearest data points, known as support vectors.

To improve generalization on noisy or non-separable data, SVM employs the soft margin formulation, allowing some misclassifications. The optimization problem is defined as:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i, \quad \text{subject to} \begin{array}{r} y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \\ \xi_i \geq 0. \end{array} \tag{4}$$

where C controls the trade-off between margin maximization and classification errors, and ξ_i are slack variables that allow margin violations.

To handle non-linear relationships, SVM uses kernel functions to compute similarities directly in the input space without explicitly computing $\phi(x)$. Commonly used kernels include:

- **Linear kernel**: $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$,
- **Polynomial kernel**: $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + \theta)^d$,
- **Radial Basis Function (RBF)**: $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$.

The parameters optimized include:

- **Kernel type (`kernel`)**: Explored options were `linear`, `rbf`, and `poly`.
- **Regularization parameter (`C`)**: Ranges from 0.1 to 100, controlling the trade-off between margin maximization and misclassification.
- **Kernel coefficient (`gamma`)**: Options included `scale` (default) and `auto`, determining the shape of the RBF and polynomial kernels.

3.3. Gradient Boosting Classification

Gradient Boosting Classification is an ensemble learning method that iteratively combines weak learners, typically decision trees, to build a strong predictive model. The final model is expressed as:

$$F_M(x) = F_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x), \tag{5}$$

where $F_0(x)$ is the initial prediction, $h_m(x)$ represents the m-th weak learner, γ_m is the learning rate, and M is the total number of boosting iterations. The model optimizes a differentiable loss function $L(y, F(x))$, typically log loss for classification:

$$L(y, F(x)) = -\sum_{i=1}^{N}\sum_{k=1}^{K} 1(y_i = k) \cdot \log\left(\frac{e^{F_k(x_i)}}{\sum_{j=1}^{K} e^{F_j(x_i)}}\right), \tag{6}$$

where N is the number of samples, K is the number of classes, $F_k(x_i)$ is the predicted score for class k, and $1(y_i = k)$ is an indicator function for the true class. At each iteration, pseudo-residuals are computed as:

$$r_{i,k}^{(m)} = 1(y_i = k) - \frac{e^{F_k(x_i)}}{\sum_{j=1}^{K} e^{F_j(x_i)}}, \tag{7}$$

and a weak learner $h_m(x)$ is trained to approximate these residuals. The model is updated as:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x). \tag{8}$$

The parameters optimized include:

- **Number of estimators (`n_estimators`)**: Number of boosting iterations, tested with 100 and 200.
- **Learning rate (`learning_rate`)**: Step size for updates, tested with 0.01 and 0.1.
- **Maximum depth (`max_depth`)**: Depth of decision trees, tested with 3 and 5.
- **Subsample (`subsample`)**: Fraction of samples used for training, tested with 0.8 and 1.0.
- **Minimum samples for splits and leaves (`min_samples_split, min_samples_leaf`)**:
  Tree complexity parameters, tested with [2, 5, 10] and [1, 2, 4].

3.4. Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) consist of interconnected layers, where each node computes an output as:

$$z_j = f\left(\mathbf{w}_j^T \mathbf{x}\right), \tag{9}$$

with $\mathbf{x}$ as the input vector, $\mathbf{w}_j$ as the weight vector, and $f(\cdot)$ as the activation function (e.g., ReLU, sigmoid, or tanh). Outputs from each layer are passed to the next, and the final output is:

$$\mathbf{y} = f\left(\mathbf{W}^T \mathbf{z}\right), \tag{10}$$

where $\mathbf{z}$ is the last hidden layer's output, and $\mathbf{W}$ is the weight matrix of the output layer.

Training minimizes the error function, such as mean squared error or cross-entropy loss:

$$E = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{t}_n\|^2, \tag{11}$$

where $\mathbf{y}_n$ and $\mathbf{t}_n$ are the predicted and true outputs, respectively. Weights are updated iteratively using gradient descent:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}), \tag{12}$$

with $\eta > 0$ as the learning rate. Backpropagation computes gradients efficiently.

The parameters optimized include:

- **Hidden layer sizes (`hidden_layer_sizes`)**: Tested configurations include (50), (100), (100, 50), and (50, 50).
- **Activation functions (`activation`)**: Evaluated ReLU, tanh, and logistic.
- **Optimizers (`solver`)**: Considered Adam and SGD.
- **Regularization strength (`alpha`)**: Explored values of 0.0001, 0.001, and 0.01.
- **Maximum iterations (`max_iter`)**: Set to 200 and 500.

3.5. Voting

The Voting classifier is an ensemble method that combines the predictions of multiple base models to improve classification performance. In this study, we used **hard voting**, where the final prediction is determined by the majority vote among the selected models. This approach allows the classifier to benefit from the strengths of multiple models while potentially mitigating the weaknesses of any single model.

3.5.1. Principle of Hard Voting

Given N base models, each model predicts a class label $\hat{y}_i$ for an input X, where $i \in \{1, 2, \ldots, N\}$. The hard voting classifier predicts the final label $\hat{y}$ as:

$$\hat{y} = \arg\max_k \sum_{i=1}^{N} 1(\hat{y}_i = k).$$

Here:

- $1(\hat{y}_i = k)$ is an indicator function that equals 1 if model i predicts class k, and 0 otherwise.
- $\arg\max_k$ selects the class k with the highest number of votes.

In the event of a tie, the class with the lowest index is chosen by default.

3.5.2. Model Selection Process

We evaluated four base models—Random Forest, Support Vector Machine (SVM), Gradient Boosting Classifier (GBC), and Artificial Neural Network (ANN)—as candidates for the Voting classifier. The final set of models used in the Voting ensemble was determined based on their individual performance during testing. Models with poor classification results were excluded to optimize the overall ensemble performance. This flexible selection process ensures that only the best-performing models contribute to the Voting classifier.

3.5.3. Evaluation

The Voting classifier was trained on the training set and tested on the test set. Its performance was assessed using accuracy, precision, recall, F1-score, and AUC metrics. The model also generated a confusion matrix to analyze the classification results across the three labels (*high*, *medium*, *low*). By combining predictions from the selected models, the Voting classifier demonstrated improved performance compared to most individual classifiers. Figure 4 shows the Flowchart of the Voting Classifier.

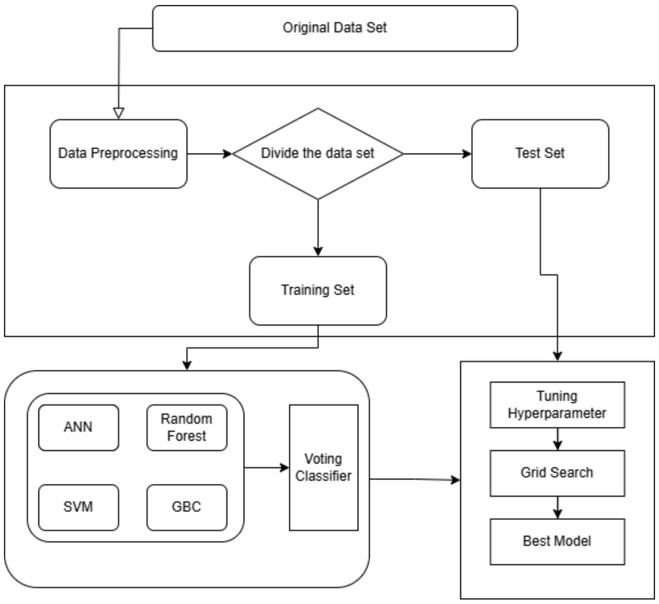

Fig. 4. Flowchart of the Voting Classifier

4. RESULT

4.1. Model Training

In Section 3, we discussed the logic and underlying principles of each model. In this section, we present detailed information, including feature selection results, optimal parameters, and the implementation of ensemble methods, to demonstrate their performance and effectiveness.

4.1.1. Random Forest

In Random Forest models, feature selection enhances classification performance by identifying the most critical predictors. By setting an importance threshold of 0.01, the model selectively retains features with the highest predictive significance. These key features are then ranked from most to least important. Table 1 presents the features selected through Random Forest feature importance analysis.

4.1.2. Voting

Based on the comparative performance analysis of the four classifiers, we will implement a hard voting ensemble strategy using the three top-performing models: Random Forest, Artificial Neural Network (ANN), and Gradient Boosting Classifier (GBC).

Table 1. Selected Features

Index	Feature Name
9	volatile acidity
0	alcohol
7	sulphates
2	citric acid
8	total sulfur dioxide
1	chlorides
6	pH
3	density
4	fixed acidity
5	free sulfur dioxide

4.1.3. Model Parameters

To achieve the best performance for each model, we tuned their hyperparameters. Table 2 summarizes the optimal parameters for each model, determined through grid search.

4.2. Performance Indices

Table 3 illustrates the confusion matrix for the label "low," providing a breakdown of predicted versus actual classifications. It highlights True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN) across all three predicted categories (*low*, *medium*, *high*). The calculations for the labels "medium" and "high" follow the same process as those for the label "low".

The performance of the classification models was evaluated using the following metrics:

1. **Accuracy:** Measures the overall correctness of the classification.
$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$
2. **Precision:** Indicates the proportion of positive identifications that were actually correct.
$$\text{Precision} = \frac{TP}{TP + FP}$$
3. **Recall:** Represents the ability of the model to find all the relevant cases.
$$\text{Recall} = \frac{TP}{TP + FN}$$
4. **F1 Score:** A harmonic mean of precision and recall, balancing their trade-off.
$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$
5. **AUC (Area Under the Curve):**
   - **Macro-Averaged AUC:** Calculates the AUC for each class (using the One-vs-Rest method) and then averages the AUC values across all classes. Assigns equal weight to all classes.
$$\text{Macro-Averaged AUC} = \frac{1}{K}\sum_{i=1}^{K} \text{AUC}_i$$
     where K is the total number of classes, and AUC_i is the AUC for class i.
     Parameter: `average='macro'`.

Table 2. Model Parameters

Model	**Parameter**
Random Forest	`'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 200`
SVM	`'C': 100, 'gamma': 'auto', 'kernel': 'rbf'`
GBC	`'learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 200, 'subsample': 0.8`
ANN	`'activation': 'relu', 'alpha': 0.0001, 'hidden_layer_sizes': (50, 50), 'max_iter': 500, 'solver': 'adam'`

Table 3. Confusion Matrix for label "low"

	Predicted low	**Predicted medium**	**Predicted high**
Actually low	TP (True Positive)	FN (False Negative)	FN (False Negative)
Actually medium	FP (False Positive)	TN (True Negative)	TN (True Negative)
Actually high	FP (False Positive)	TN (True Negative)	TN (True Negative)

- **Micro-Averaged AUC:** Aggregates the True Positives (TP) and False Positives (FP) across all classes to compute an overall AUC.

$$\text{Micro-Averaged AUC} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + \sum_{i=1}^{N} FP_i}$$

  where N is the total number of samples in the test set. Parameter: `average='micro'`.

These metrics were chosen to provide a comprehensive evaluation of the model's performance, particularly for imbalanced datasets.

4.3. Performance Comparison

We compare the performance of the models with and without feature selection and conclude that the model performed better with feature selection. Table 4 summarizes the performance metrics of each model after applying feature selection. The metrics include accuracy, precision, recall, F1-score, and both macro and micro AUC scores.

The comparative analysis reveals notable variations in performance metrics across five models: Random Forest, Support Vector Machine (SVM), Gradient Boosting Classifier (GBC), Artificial Neural Network (ANN), and the Voting Classifier.

4.3.1. Voting Classifier Performance

The Voting Classifier demonstrated superior performance across critical evaluation metrics:

- Highest accuracy at 87.5%
- Significantly improved precision (0.867 compared to $\sim$0.5 in individual models)
- Competitive recall and F1-score
- Excellent AUC scores (Macro: 0.824, Micro: 0.957)

4.3.2. Individual Model Contributions

Each base classifier brought unique strengths to the ensemble:

- Random Forest and Gradient Boosting Classifier (GBC) exhibited strong, consistent performance
- Artificial Neural Network (ANN) provided balanced metrics
- Support Vector Machine (SVM) showed relatively weaker predictive capabilities

4.3.3. Strategic Recommendation

We recommend adopting the Voting Classifier as the primary predictive model. This approach leverages the collective strengths of Random Forest, GBC, and ANN, effectively mitigating individual model limitations and enhancing overall predictive performance.

4.4. Limitations and Potential Solutions

4.4.1. Class Imbalance Challenge

Observation:

- F1-score and recall hovering around 0.5 indicate significant classification difficulties.
- The imbalanced dataset across the three labels (low, medium, high) confirms this issue.

Current Approach:

- **Class Weights Adjustment:** Adjusting class weights in the model (e.g., `class_weight='balanced'` in scikit-learn) to penalize misclassification of minority classes more heavily.

Potential Solutions:

1. **Oversampling Minority Classes:**
   Use techniques like SMOTE (Synthetic Minority Oversampling Technique) to generate synthetic samples for underrepresented labels (low and high), thereby balancing the dataset.
2. **Undersampling the Majority Class:**
   Reduce the number of medium-labeled samples to ensure a more balanced distribution of labels.

Table 4. Model performance with feature selection

Model	Accuracy	Precision	Recall	F1-Score	Macro AUC	Micro AUC
Random Forest	0.85625	0.507564	0.495214	0.499446	0.816556	0.956062
SVM	0.784375	0.487511	0.542466	0.506756	0.770427	0.942744
GBC	0.853125	0.505197	0.539376	0.521331	0.759919	0.950562
ANN	0.825	0.527766	0.519945	0.52331	0.764449	0.93853
Voting	0.875	0.866622	0.546636	0.573199	0.823697	0.957402

4.4.2. Label Range Sensitivity

Observation:

- Model performance is highly sensitive to how labels are defined. Experimental results show the best classification performance with the following label boundaries:
  - Low labels: 3–4
  - Medium labels: 5–6
  - High labels: 7–8
- This sensitivity highlights the model's vulnerability to label boundary definitions.

Impact:

- Improper label definitions can result in:
  - Overlapping feature distributions across boundaries
  - Exacerbated class imbalance
  - Reduced model accuracy and generalization ability

Potential Solutions:

Leverage domain expertise to define label boundaries based on real-world considerations, such as consumer taste preferences or industry standards for wine quality.

5. CONCLUSION

This project applied machine learning techniques to classify red wine quality based on physicochemical attributes. The dataset was carefully preprocessed to guarantee experimental results. This included cleaning the data, standardizing features, removing outliers, and performing feature selection. Feature selection was particularly important as it reduced the dataset's dimensionality while retaining the most relevant predictors.

Several machine learning models were implemented, including Random Forest, Support Vector Machine (SVM), Gradient Boosting, Artificial Neural Networks (ANN), and an ensemble Voting Classifier. Each model underwent hyperparameter tuning. Among these models, the Voting Classifier achieved the best results, with an accuracy of 87.5% and a precision of 86.66%. This showed that combining predictions from multiple models can lead to better overall performance compared to individual models.

Challenges were also recognized during the project. Recall and F1-scores were impacted by class imbalance. To solve this problem, future research could investigate techniques like oversampling and class weighting. The definition of quality labels also had an impact on the model's performance. This emphasizes how crucial it is to carefully choose labels in order to enhance categorization outcomes..

Overall, the study showed that wine quality can be accurately predicted by machine learning. It also included insightful information about how to actually use these methods. Addressing data imbalance, improving feature selection techniques, and investigating alternative ensemble methodologies could be the main goals of future research. These actions could strengthen the strategy and increase model accuracy.

6. REFERENCES

[1] Paulo Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Wine Quality," UCI Machine Learning Repository, 2009, DOI: https://doi.org/10.24432/C56S3T.