Predicting Wine Quality

Samuel Olatunde

Click here to find the code

Introduction

The objective of this project is to apply and compare several supervised machine learning algorithms to classify the quality of wine based on its physicochemical properties. The dataset consists of approximately 5,000 samples with 11 numerical features, including fixed acidity, citric acid, residual sugar, and pH, all of which influence perceived wine quality (score between 0 and 10) [1]. By modeling the relationship between these measurable chemical attributes and quality ratings, the study aims to identify the most effective approach for accurate and interpretable classification. Four primary models were implemented and evaluated: K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Machine (SVM), Decision Trees, and Random Forest. Each model was trained, tuned, and tested to assess performance across multiple metrics, including accuracy, precision, recall, and F1-score, while exploring model interpretability through feature importance.

Background

To better understand feature behavior and model interpretability in wine-quality prediction, I reviewed the paper “Modeling Wine Preferences by Data Mining from Physicochemical Properties” by P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis (2009), published in Decision Support Systems. The study demonstrated how data-driven models can predict sensory-based wine quality using measurable chemical attributes and emphasized the importance of systematic feature evaluation. From this work, I learned two key approaches to assessing feature influence: (1) ***traditional backward feature selection***, where each feature is removed in turn and model performance is re-evaluated to determine its predictive contribution, and (2) **sensitivity analysis combined with backward elimination**, which measures how variation in each input affects the model’s output. Partial dependence plots are a common technique for conducting sensitivity analysis; however, they proved somewhat complex to fully interpret and implement within the project’s limited timeframe. As an alternative, I applied **permutation feature importance** to estimate feature influence more efficiently. Specifically, I used permutation feature importance for the more computationally intensive models (SVM and Random Forest) and applied traditional backward feature selection for models that were less computationally demanding (Decision Tree, Logistic Regression, and KNN).

Methodology

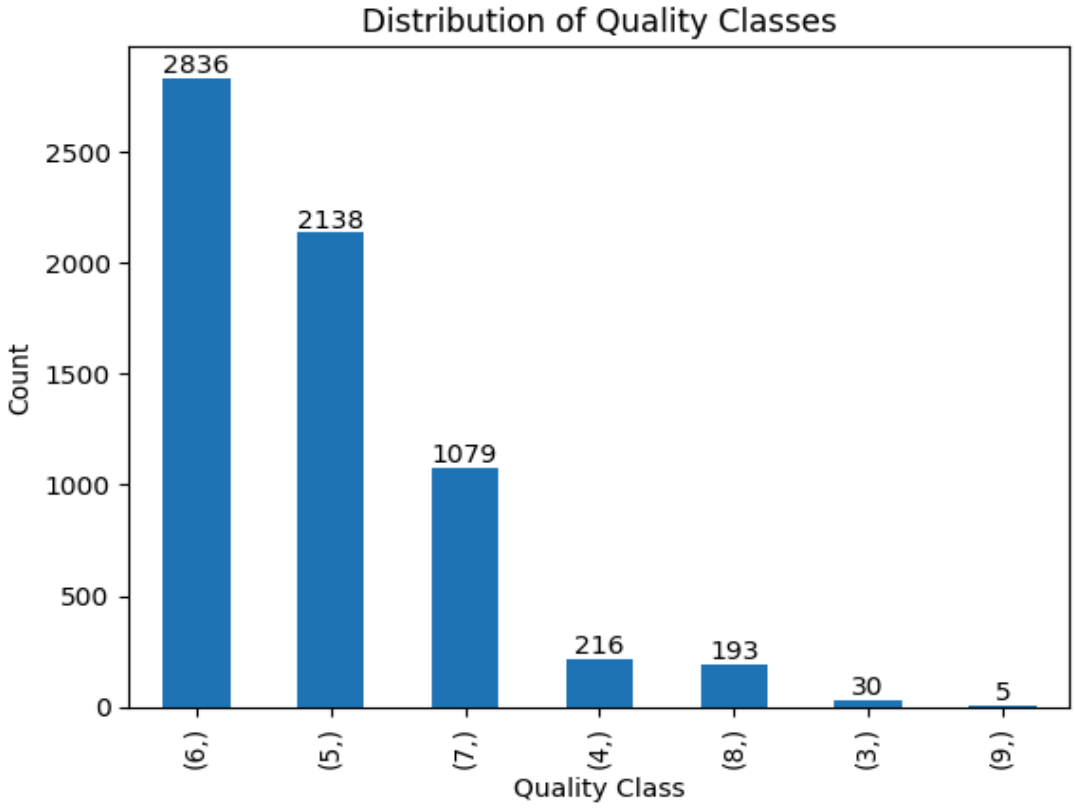
This section outlines the process followed to implement, tune, and evaluate each supervised learning model used for wine-quality classification. The workflow for all models followed a consistent structure: data preprocessing, model training, hyperparameter tuning, selection of

the best model based on validation metrics, feature selection using the best model, and final evaluation on the test set.

The remainder of this section describes the imbalance and the workflow of experiments. Separate subsections are dedicated to each algorithm—Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree, and Random Forest—detailing their specific configurations, parameter exploration, and validation results that guided the selection of optimal model settings.

Imbalance

The chart below shows the distribution of each class of wine quality. Observe that there's a lot more moderate quality wines (class: 4,5,6,7) compared to high and low quality wines (class:3,8,9). This data accurately models the real world – average wines are more common than exquisite or terrible wines – however, it's not good for prediction as it biases our models to the majority classes. As a result, resampling techniques such as SMOTE, RandomUndersampling and class weighting were explored to improve generalization and prevent bias toward majority classes.



Logistic Regression Workflow

The logistic regression experiments began by loading the wine-quality dataset from the UCI

Machine Learning Repository and visualizing the distribution of the target variable to assess class imbalance (see Chart Above). The data was split into training, validation, and test sets and standardized using StandardScaler to ensure all physicochemical features were on a comparable scale.

Baseline models were trained using both L1 and L2 regularization without any class weighting to establish reference performance, with parameters set to solver='saga', random_state=17, and max_iter=10000. As shown in Table 1, both variants performed similarly, achieving an **accuracy of approximately 0.54** and an **F1-score near 0.22**, with L1 regularization slightly outperforming L2.

To address class imbalance, additional experiments introduced class weighting, SMOTE, and Random Undersampling. Class weighting applied inverse frequency weights during optimization, while SMOTE generated synthetic samples for minority classes, and Random Undersampling reduced samples from majority classes. Each method was applied independently to the training data, and new models were retrained under identical hyperparameters. The class-weighted models showed lower **accuracy (0.27–0.28**) but improved **recall (up to 0.43)**, indicating increased sensitivity to underrepresented quality classes. SMOTE and undersampling models produced balanced yet modest performance improvements, with the best SMOTE model (L1) achieving an **accuracy of 0.30** and an **F1-score of 0.20,** while the best undersampling model reached **0.27 accuracy** and **0.17 F1-score**.

After comparing all configurations, the L1-regularized logistic regression without resampling was retained for feature analysis, as it maintained the highest validation F1-score with stable convergence. Backward sequential feature selection was then performed using the following configuration: direction='backward', scoring='f1_macro', and cv=StratifiedKFold(n_splits=3) to maintain class proportions. This process identified six key features—*volatile acidity, residual sugar, free_sulfur_dioxide, total_sulfur_dioxide, pH, and alcohol*—as the most influential predictors of wine quality.
The final logistic regression model was retrained on this reduced feature set and evaluated on the test data using both macro- and weighted-averaged metrics to account for class imbalance.

Table 1: Logistic Regression Validation Performance under Different Configurations

Experiment Type	Model Variant	Accuracy	Precision	Recall	F1-Score
Baseline (no weighting or resampling)	Logistic Regression (L1)	0.544	0.511	0.218	0.216

	Logistic Regression (L2)	0.544	0.510	0.218	0.216
Class Weighting	Logistic Regression (wL1)	0.271	0.234	0.271	0.187
	Logistic Regression (wL2)	0.280	0.232	0.432	0.196
SMOTE Oversampling	Logistic Regression (sL1)	0.302	0.236	0.277	0.198
	Logistic Regression (sL2)	0.301	0.236	0.276	0.198
Random Undersampling	Logistic Regression (rL1)	0.273	0.208	0.200	0.170
	Logistic Regression (rL2)	0.241	0.199	0.186	0.158

Note: L1 and L2 refer to the type of regularization. Prefixes w, s, and r indicate weighted, SMOTE, and random undersampling variants, respectively.

K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) workflow closely followed the general preprocessing and evaluation steps outlined previously for Logistic Regression. The dataset was obtained from the UCI Machine Learning Repository using the ucimlrepo library, and exploratory analysis confirmed a moderate class imbalance, with most samples concentrated in the mid-quality range. The data was split into training, validation, and test sets using a 70–15–15 ratio and standardized with StandardScaler to ensure consistent scaling across features.

Initial experiments focused on evaluating KNN models without any resampling to establish baseline performance. Several configurations were tested using different values of k (15–201) and two weighting schemes—uniform and distance. Validation results, summarized in Table 2, show that the best-performing configuration used **n_neighbors=29** and weights='distance', achieving an **accuracy of 0.654**, **precision of 0.776**, and **F1-score of 0.348**. These results

indicate that distance weighting improved classification consistency by giving more influence to closer samples in feature space.

To address class imbalance, two resampling strategies were evaluated: SMOTE and Random Undersampling. Each method was applied independently to the training data, followed by re-scaling and retraining of KNN models using the same hyperparameter grid. The SMOTE-enhanced model achieved moderate performance, with **n_neighbors=16** and weights='distance' yielding an **accuracy of 0.53**8 and **F1-score of 0.328**. The random undersampling configuration, limited to smaller neighborhood sizes due to reduced sample counts, achieved lower overall performance, with its best model (k=2, weights='uniform') reaching **0.310 accuracy** and **0.195 F1-score**.

After evaluating all variants, the Non-resampled KNN model (n_neighbors=29, weights='distance') was selected for feature analysis, as it achieved the highest validation performance across all configurations. Sequential Feature Selection (SFS) with backward elimination was performed using scoring='f1_macro' and a StratifiedKFold cross-validation strategy to maintain class proportions. The selected subset of features—*volatile_acidity, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, pH, and alcohol*—was identified as most predictive of wine quality.

The final KNN model was retrained on this optimized feature set and evaluated on the test set using accuracy, precision, recall, and F1-score (macro and weighted averages).

Table 2: KNN Validation Performance under Different Configurations

Experiment Type	Accuracy	Precision	Recall	F1-Score	n_neighbors	Weights
Baseline (no resampling)	0.654	0.776	0.325	0.348	29	distance
	0.647	0.765	0.318	0.338	17	distance
SMOTE Oversampling	0.538	0.325	0.400	0.328	16	distance
	0.538	0.325	0.402	0.329	20	distance
Random Undersampling	0.310	0.211	0.241	0.195	2	uniform

	0.241	0.191	0.174	0.157	1	uniform

Note: Table 2 shows the top two performing models for each experiment type

Decision Tree Classifier

The Decision Tree experiments followed the standard workflow established in earlier models, including data preprocessing, class distribution assessment, and scaling. A comprehensive grid search was performed over multiple hyperparameters—tree depth, minimum samples per split and leaf, number of features considered at each split, pruning complexity parameter (ccp_alpha), and class weighting. Table X below shows the hyperparameter grid tuned in this experiment.

Table X: Hyperparameter Grid for Decision Tree Classifier

Hyperparameter	Values Tested
max_depth	[8, 10]
min_samples_split	[10, 14, 20]
min_samples_leaf	[3, 5, 8]
max_features	['sqrt', 'log2', 3, 5]
class_weight	[None, 'balanced']
ccp_alpha	[0.0, 0.001, 0.005, 0.01]

For each configuration, training and validation metrics were recorded to identify signs of overfitting, which were evident in several parameter combinations. The best baseline model (shown in Table 3) achieved a **training accuracy of 0.669** and a **validation accuracy of 0.564**. This model used parameters max_depth=10, min_samples_split=14, min_samples_leaf=8, max_features=5, class_weight=None, and ccp_alpha=0.0.

To investigate how resampling affected performance, the workflow applied SMOTE to balance the class distribution, then retrained the model with a new hyperparameter grid. The SMOTE grid search extended max_depth to {8, 20, 30} while keeping the remaining parameters consistent. Although the SMOTE-trained models achieved high training performance (e.g.,

T_accuracy = 0.889, T_f1 = 0.889), their validation accuracy dropped sharply (**V_accuracy = 0.517, V_f1 = 0.296**), further highlighting overfitting due to synthetic sample expansion.

Based on these comparisons, the non-resampled Decision Tree was selected as the final model, balancing interpretability and generalization. Subsequent Sequential Feature Selection (SFS) with backward elimination was performed using this estimator, with scoring='f1_macro' and StratifiedKFold(n_splits=3) cross-validation to maintain class balance. The SFS process identified six features—*fixed_acidity, volatile_acidity, chlorides, free_sulfur_dioxide, total_sulfur_dioxide, and alcohol*—as the most informative predictors.

The final Decision Tree model was retrained on the training data using this feature subset and the optimized parameters (max_depth=10, min_samples_split=14, min_samples_leaf=8, max_features=5, ccp_alpha=0.0) and evaluated on the test set. Metrics including accuracy, precision, recall, and F1-score (macro and weighted) were computed to provide a comprehensive assessment of model performance and class-level predictive balance.

Table 3: Decision Tree Classifier — Comparison of Non-SMOTE and SMOTE Experiments

Metric / Parameter	Non-SMOTE (Best Model)	SMOTE (Best Model)
Training Accuracy	0.669	0.889
Training Precision	0.632	0.888
Training Recall	0.352	0.889
Training F1-Score	0.380	0.889
Validation Accuracy	**0.564**	0.517
Validation Precision	**0.416**	0.289
Validation Recall	**0.256**	0.322
Validation F1-Score	**0.265**	0.296
max_depth	10	20
min_samples_split	14	10
min_samples_leaf	8	5
max_features	5	5

class_weight	None	None
ccp_alpha	0.0	0.0

Random Forest Classifier

Building on the Decision Tree experiments, the Random Forest workflow explored both ensemble and imbalance-handling strategies to enhance prediction stability and fairness across wine quality classes. Three main configurations were tested: a standard Random Forest, a Balanced Random Forest (BRF), and a Random Forest trained on SMOTE-resampled data. Each was tuned over multiple hyperparameters—including the number of estimators, maximum depth, minimum samples per split, and number of features per node—and evaluated using accuracy, precision, recall, and F1-score on both training and validation sets. Table Y itemizes the hyperparameter grid for the configurations used in this experiment.

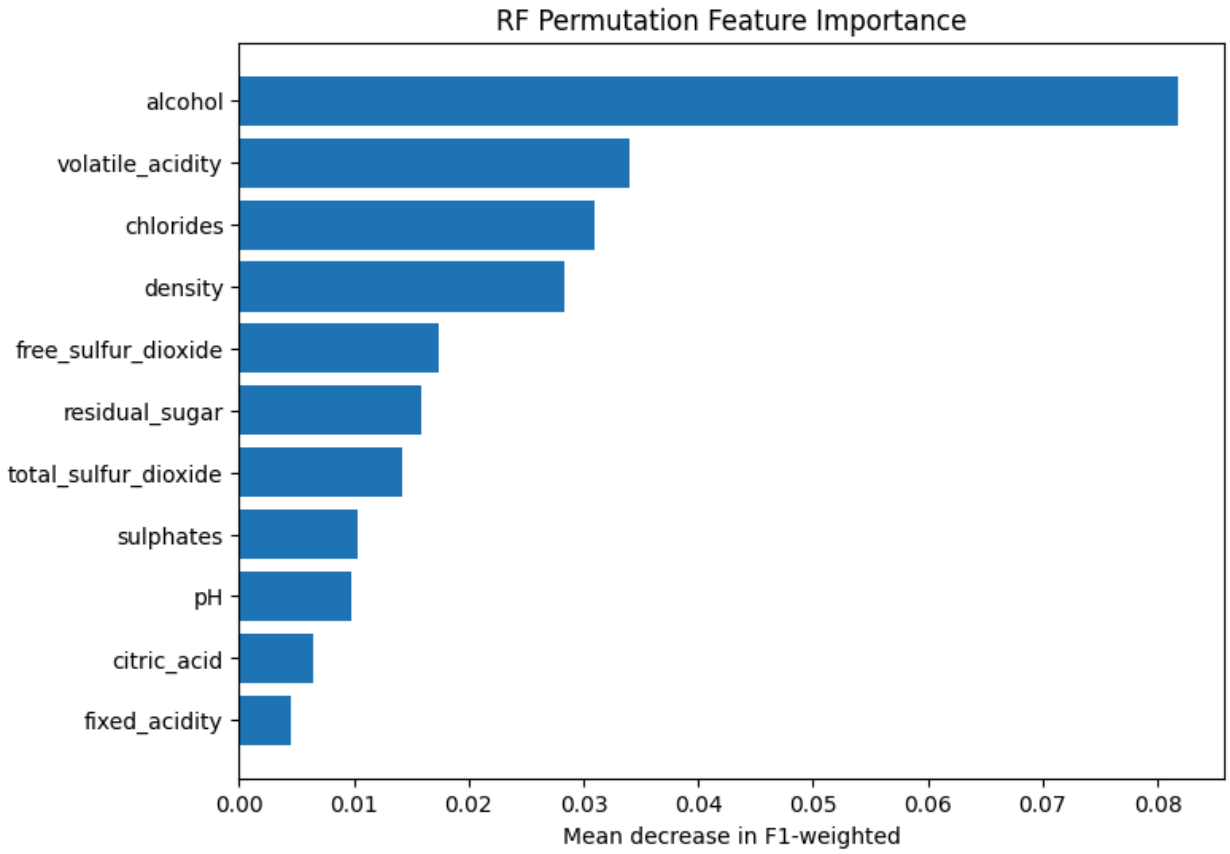
Table Y: Random Forest Hyperparameter Grids Comparison

Hyperparameter	Random Forest (No Resampling)	Balanced Random Forest	Random Forest with SMOTE
n_estimators	[150, 300]	[150, 160, 96]	[150, 300]
max_features	['sqrt', 'log2', 3, 5]	[2, 1.0, 'sqrt', 'log2']	[1.0, 2, 5]
max_depth	[3, 5, 7]	[None]	[None, 10]
min_samples_split	[2, 4, 8, 25]	[2, 6, 8, 12, 24]	[2, 4, 8]

As shown in Table 4, the standard Random Forest achieved the strongest balance between training and validation performance, with a **training accuracy of 0.667** and **validation accuracy of 0.586**, alongside a validation F1-score of 0.549. In contrast, the Balanced Random Forest—designed to internally weight minority classes—recorded lower validation performance (**validation accuracy of 0.368, F1-score of 0.383**), indicating that algorithm-level balancing alone did not improve generalization. The SMOTE-based Random Forest achieved perfect training metrics (**T_accuracy = 1.0, T_F1 = 1.0)** but suffered from mild overfitting, with validation metrics slightly higher (**V_accuracy = 0.637, V_F1 = 0.636**) but not significantly outperforming the standard model.

Considering both generalization and interpretability, the standard Random Forest was selected as the final model for further analysis. Its optimized parameters were (n_estimators=300, max_depth=7, min_samples_split=8, max_features=5), representing a configuration that offered competitive performance while maintaining reasonable complexity.

Feature selection was conducted using Permutation Importance, which quantifies each feature's contribution by measuring the decrease in macro F1-score when values are randomly permuted. The chart below shows the importance plots.



A threshold of 0.01 was used to remove features with minimal predictive influence. The resulting subset included eight key physicochemical features: *alcohol, volatile_acidity, chlorides, density, free_sulfur_dioxide, residual_sugar, total_sulfur_dioxide, and sulphates.* The refined model trained on this subset was used for final test evaluation.

Table 4: Random Forest Classifier — Comparison of Best Non-SMOTE, Balanced, and SMOTE Experiments

Metric / Parameter	Standard RF (Non-SMOTE)	Balanced RF	RF (SMOTE)
Training Accuracy	0.667	0.382	1.000

Training Precision	0.701	0.492	1.000
Training Recall	0.667	0.382	1.000
Training F1-Score	0.643	0.398	1.000
Validation Accuracy	**0.586**	0.368	0.637
Validation Precision	**0.616**	0.462	0.640
Validation Recall	**0.586**	0.368	0.637
Validation F1-Score	**0.549**	0.383	0.636
n_estimators	300	150	300
max_depth	7	None	None
min_samples_split	8	8	4
max_features	5	sqrt	5

Support Vector Machine (SVM)

The Support Vector Machine experiments were conducted following the same standardized workflow as previous models, including data splitting, scaling, and class imbalance assessment. Two experiments were conducted, one using the raw data and another using resampled data via SMOTE. Both approaches aimed to determine how imbalance handling affects SVM generalization.

The baseline SVM pipeline combined StandardScaler with an SVC classifier. A grid search using GridSearchCV with StratifiedKFold cross-validation tuned parameters such as kernel type, regularization strength (C), kernel coefficient (gamma), polynomial degree, and class weighting. Table 5(a) highlights the parameter grid used for the non-SMOTE SVM.

To address imbalance, a second pipeline was developed using SMOTE embedded within an imblearn pipeline. This configuration applied oversampling before SVM fitting to generate synthetic samples of minority wine quality classes. The hyperparameter grid is also highlighted in Table 5(a).

Table 5(a): Hyperparameter Grid for Non-SMOTE SVM and SMOTE-SVM

Hyperparameter	Values Tested (Non-SMOTE)	Values Tested (SMOTE)

kernel	['linear', 'rbf', 'poly']	
C	[0.5, 1, 3]	[0.1, 0.5]
gamma	['scale', 'auto']	
degree	[2, 3]	[2, 4]
coef0	[0, 0.5, 1]	[0, 0.5, 0.7]
class_weight	[None, 'balanced']	[None]

The best-performing non-SMOTE model used a polynomial kernel with parameters (C=3, degree=3, gamma='scale', coef0=1, class_weight=None). It achieved a **training accuracy of 0.650 and validation accuracy of 0.559**, with a macro **F1-score of 0.303** as shown in Table 5(b).

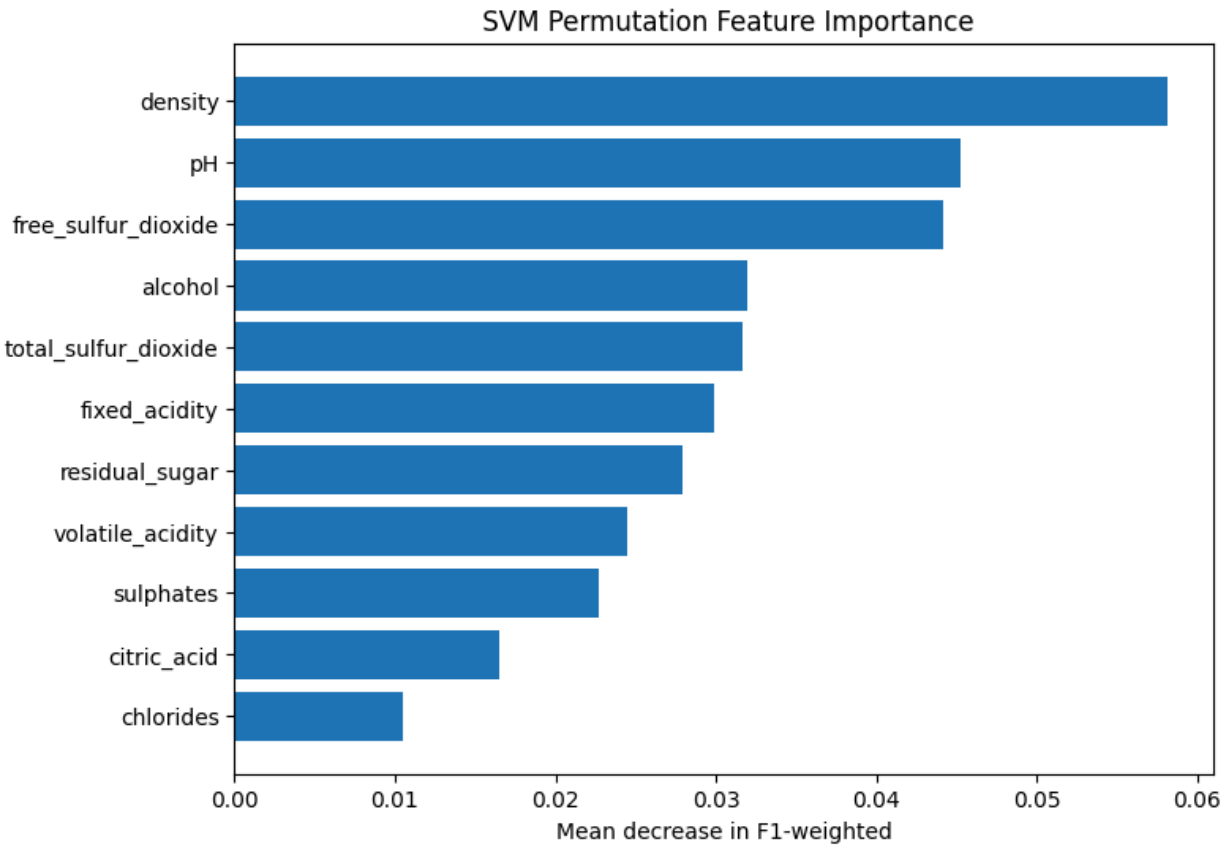
The SMOTE-enhanced SVM achieved its best performance using a polynomial kernel (C=0.5, degree=4, gamma='auto', coef0=0.5). It reached a **training accuracy of 0.595** and v**alidation accuracy of 0.442**, with a **macro F1-score of 0.290**, indicating that while SMOTE slightly improved minority recall, it reduced overall generalization.

Table 5(b): Comparison of SVM Performance Across Configurations

Metric / Parameter	Non-SMOTE SVM (Best Model)	SMOTE-SVM (Best Model)
Training Accuracy	0.650	0.595
Validation Accuracy	**0.559**	0.442
Macro F1-Score	**0.303**	0.290
Kernel Type	poly	poly
C	3	0.5
Degree	3	4

Gamma	scale	auto
coef0	1	0.5
class_weight	None	None

The non-SMOTE SVM was selected as the final model due to its superior validation accuracy and more stable performance. Feature selection was then performed using Permutation Importance; the importance chart is shown below.



Applying a drop threshold of 0.02 to remove less influential features, the following eight features were retained for final training and testing: *density, pH, free_sulfur_dioxide, alcohol, total_sulfur_dioxide, fixed_acidity, residual_sugar, volatile_acidity, and sulphates*.
This refined feature subset helped maintain model interpretability while improving computational efficiency.

Final Results and Discussion

This section provides the results from the evaluation of all best-performing validation models on the test set. An interesting observation is that none of the resampled models reached this stage, likely due to noise introduced by oversampling and information loss from undersampling.

Note: We mostly refer to macro recall because it's a better reflector of how the model handles imbalance.

Logistic Regression

The final Logistic Regression model used L1 regularization with the SAGA solver (max_iter=10000) without resampling, chosen for its stability and interpretability. On the test set, it achieved an **accuracy of 0.532**, **macro F1-score of 0.209**, and **weighted F1-score of 0.497**. **Precision** was moderate (**macro = 0.521**, **weighted = 0.528**), while macro recall remained low (**macro = 0.215, weighted = 0.53**), showing difficulty in identifying minority quality classes.

These results reflect the model's limited ability to capture non-linear relationships in the data. Nevertheless, L1 regularization enhanced feature sparsity and interpretability, making Logistic Regression a reliable baseline against which more complex models were compared.

K-Nearest Neighbors(KNN)

The final KNN model used **29 neighbors** with **distance-based weighting** and no resampling, which provided the strongest validation performance among the configurations tested. On the test set, it achieved an **accuracy of 0.629**, **macro F1-score of 0.334**, and **weighted F1-score of 0.612**. **Precision** was relatively high (**macro = 0.779**, **weighted = 0.653**), but **recall** remained modest (**macro = 0.307**), indicating that while predictions were often correct, the model missed several minority cases.

Overall, the KNN model performed better than Logistic Regression, benefiting from its non-parametric nature and flexibility in capturing non-linear relationships. However, its reliance on distance metrics also made it sensitive to class imbalance and high-dimensional variability, which slightly limited its generalization.

Decision Tree

The final Decision Tree model used moderate complexity parameters (max_depth = 10, min_samples_split = 14, class_weight=None, min_samples_leaf = 8, max_features = 5) to balance interpretability and overfitting risk. On the test set, it achieved an **accuracy of 0.523**, **macro F1-score of 0.246**, and **weighted F1-score of 0.505**. **Precision** was moderate (**macro = 0.410**, **weighted = 0.505**), while recall was relatively low (**macro = 0.237**), indicating uneven performance across classes.

The Decision Tree's predictive power was limited by overfitting to the training data. As a result, it served primarily as an interpretable but less generalizable benchmark for subsequent ensemble methods.

Random Forest

The final Random Forest model, configured with (n_estimators = 300, max_depth = 7, min_samples_split = 8, max_features = 5), demonstrated stronger validation performance than the Decision Tree but experienced a notable performance drop on the test data. It achieved a **test accuracy of 0.564**, **macro F1-score of 0.238**, and **weighted F1-score of 0.534**, compared to higher validation metrics (**validation F1 ≈ 0.55, validation precision ≈ 0.62**).

This decline indicates mild overfitting to the validation folds or distributional differences between validation and test sets. While **precision** remained relatively high (**macro = 0.678, weighted = 0.583**), **recall** dropped significantly (**macro = 0.236**), showing that the model tended to favor dominant classes.

Despite this, the Random Forest still outperformed the single Decision Tree, although it's just as prone to overfitting, if not more. Its ensemble nature produced more consistent predictions, and its feature importance analysis offered interpretable insights into the most influential physicochemical variables affecting wine quality.

Support Vector Machine (SVM)

The final SVM model used a polynomial kernel with parameters (C = 3, degree = 3, gamma = 'scale', coef0 = 1, class_weight = None), selected for its stable validation performance. On the test set, it achieved an **accuracy of 0.547**, **macro F1-score of 0.239**, and **weighted F1-score of 0.520**. **Precision** was moderate **(macro = 0.394, weighted = 0.545**), while **recall** remained low (**macro = 0.234**), showing that the model predicted majority classes more reliably than minority ones.

Compared to the tree-based models, the SVM achieved a similar generalization level with less evidence of overfitting. Its polynomial kernel captured moderate non-linear relationships while maintaining smoother decision boundaries, though at a higher computational cost. Overall, the SVM performed competitively with Random Forest but did not surpass it in predictive strength or balance across classes.

Conclusions

As shown in Table 6 below, the K-Nearest Neighbors (KNN) classifier achieved the highest macro-averaged F1-score among all evaluated models, indicating superior generalization on the wine quality prediction task. While this result may appear unexpected, it aligns with the intuition that wines of similar quality often share closely related physicochemical properties. Consequently, KNN's distance-based learning approach is well-suited for datasets where

samples naturally cluster by similarity — such as wines of comparable chemical composition and sensory profiles.

Table 6: Model Comparisons Based on Macro Metrics

Rank by Accuracy	Model	Accuracy	Macro Precision	Macro Recall	Macro F1
4	**Logistic Regression**	0.532	0.521	0.215	0.209
1	**K-Nearest Neighbors (KNN)**	**0.629**	**0.779**	**0.307**	**0.334**
5	**Decision Tree**	0.523	0.410	0.237	0.246
2	**Random Forest**	0.564	0.678	0.236	0.238
3	**Support Vector Machine (SVM)**	0.547	0.394	0.234	0.239

Nevertheless, certain limitations likely influenced overall model performance. The Random Forest model showed signs of overfitting, potentially due to the lack of pruning mechanisms that could have constrained tree depth without reducing expressiveness. Similarly, Logistic Regression was not fully optimized, as time constraints limited tuning of its regularization parameters. More comprehensive hyperparameter searches could also improve SVM and Random Forest, which are theoretically better equipped to model complex, high-dimensional relationships than KNN.

Beyond algorithmic improvements, incorporating richer contextual variables could further enhance predictive accuracy. Features such as price, region, or even pairings of physicochemical and sensory indicators — for example, identifying patterns like “high quality with high alcohol content” or “high quality with low alcohol content” — might provide more nuanced relationships that aid in classification. Integrating such relational and economic attributes could allow the models to capture aspects of wine quality that go beyond chemistry alone.

For future work, diagnostic and interpretability analyses are recommended. Constructing confusion matrices for each model would clarify misclassification patterns across wine quality categories. Additionally, decision path visualizations for tree-based methods and support vector plots for SVMs could improve transparency, helping to interpret how the models differentiate between wine quality levels.

References

[1] “UCI Machine Learning Repository.” Archive.ics.uci.edu, 10 June 2009, archive.ics.uci.edu/dataset/186/wine+quality.

[2] “Scikit-Learn: Machine Learning in Python — Scikit-Learn 0.24.2 Documentation.” Scikit-Learn.org, scikit-learn.org/stable/index.html#.