

## Section 1: Introduction

Machine learning classification models are essential tools for modeling data and making data-driven decisions across domains. In this project, we applied three supervised learning techniques to the UCI Red Wine Quality dataset. This dataset is publicly available for research and contains physicochemical and sensory data for red and white Portuguese wine, but this project will only perform analysis on red wines. Examples of features include acidity, sugar, pH, and more. Our target variable will be quality, meaning that we will use the 11 features provided to predict the quality of the wine based on those characteristics.

The goal of this project is to design, implement, and test our classification models to achieve the best classification performance on this dataset. Our models include Support Vector Machine, Artificial Neural Networks, and Random Forest methods.

We will evaluate models trained on three preprocessed versions of the dataset, each incorporating different data preparation strategies: (1) cleaned and normalized data, which serves as the baseline model; (2) PCA-transformed data for dimensionality reduction; and (3) data with engineered interaction terms to capture feature relationships.

We will also implement additional methods such as hyper-parameter turning and regularization as necessary. Then, we will select the best performing model for each method and compare the three methods. Programming will be done in Python and the scikit-learn package for machine learning.

[need to add what we achieved once analysis is done]

## Section 2: Exploratory Data Analysis and Data Preparation

### Dataset Overview and Initial Analysis

We begin our analysis with a dataset containing 1,599 wine samples and 11 physicochemical features, along with a target variable representing wine quality. The features include fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. An initial analysis revealed that the dataset contains no missing values, indicating a complete dataset ready for further exploration.

The distribution of feature values is summarized in Table A1 in the appendix, which provides detailed statistical measures for all features. Additional visualizations showing the distribution of each feature can be found in the appendix (Figure A2), allowing us to understand symmetry, potential outliers, and other characteristics that complement the statistical summary.

### Target Variable Analysis and Feature Correlations

Wine quality serves as the target variable, taking discrete values from 3 to 8. Understanding the distribution of wine quality is essential for addressing the challenges inherent in classification tasks, particularly given the class imbalance observed in the dataset. Additionally, analyzing feature relationships is critical for identifying multicollinearity and understanding the underlying data structure that may influence model performance.

The bar chart (left) reveals that this variable exhibits class imbalance, with quality levels 5 and 6 being the most frequent, respectively, while there are very few wines with low quality (such as 3) or extremely high quality (such as 8). The order of magnitude of the differences is very significant, as in our dataset there is a probability of 0.63% of having a wine with quality 3, while the probability is more than 68 times greater for quality 5 (42.59%). Similarly, quality 6 has a probability of 39.90%, which is more than 63 times greater than quality 3.<sup>1</sup>

The correlation matrix (right) provides a comprehensive view of pairwise linear relationships between all physicochemical features, enabling us to assess both feature-feature interactions and feature-target associations simultaneously. Notable pairwise correlations between features include strong positive correlations between fixed acidity and citric acid (0.67), fixed acidity and density (0.67), and total sulfur dioxide and free sulfur dioxide (0.67). Strong negative correlations are observed between fixed acidity and pH (-0.68), citric acid and volatile acidity (-0.55), and citric

---

<sup>1</sup>As a team, we read the codebook and documentation of the data (Cortez et al., 2009; UCI Machine Learning Repository: <https://archive.ics.uci.edu/ml/datasets/wine+quality>) and although in the description of the dictionaries the source states that quality can take values from 0 to 10, in our dataset we only found possible values from 3 to 8.

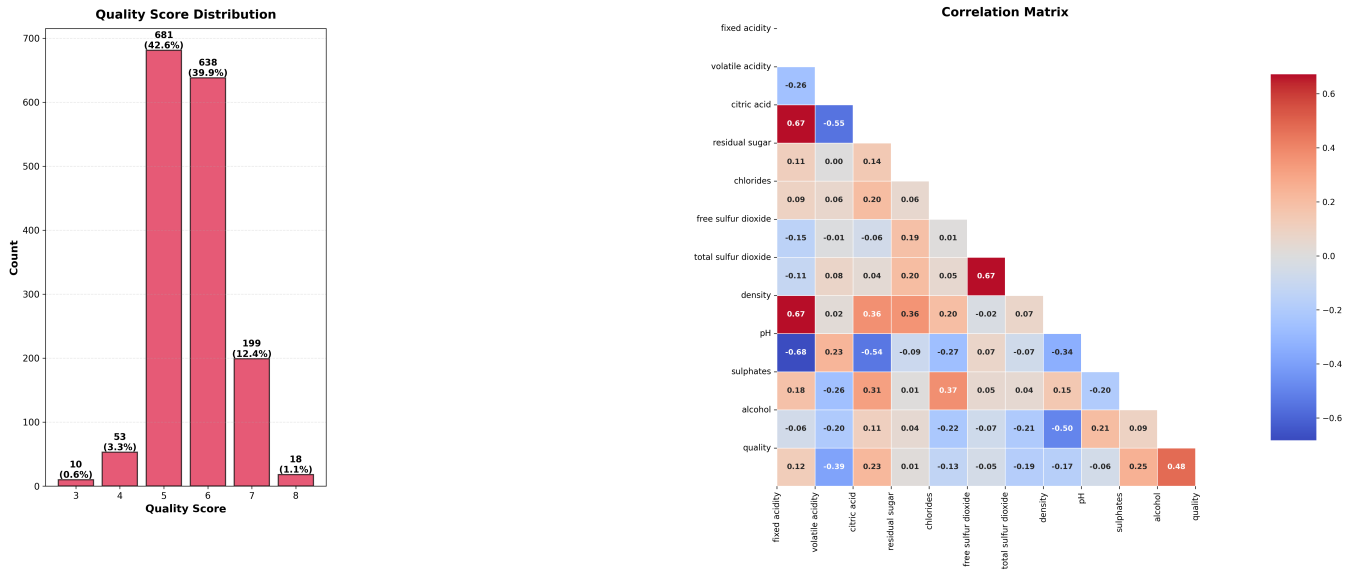


Figure 1: Quality Distribution and Correlation Matrix

acid and pH (-0.54). These relationships indicate potential multicollinearity that may influence model performance. By examining the quality row in the correlation matrix, we identify the strongest associations with wine quality. Alcohol content shows the strongest positive correlation (0.48), followed by sulphates (0.25) and citric acid (0.23). Conversely, volatile acidity exhibits the strongest negative correlation (-0.39), indicating that higher levels are associated with lower quality scores. Other features showing moderate negative correlations include total sulfur dioxide (-0.19) and density (-0.17). While quality is an ordinal variable, we compute Pearson correlation coefficients given its natural ordering, which allows us to capture linear trends. These correlations provide an initial indication of feature importance, though the classification models will ultimately determine the true discriminative power of each feature.

## Feature Engineering and Normalization

Prior to model training, it is essential to prepare the data appropriately. Machine learning algorithms, particularly Support Vector Machines (SVM) and Artificial Neural Networks (ANN), are sensitive to the scale of input features. Features with larger numerical ranges can dominate the learning process, potentially biasing the model toward those features. Therefore, we apply feature normalization to ensure all features contribute equally to the model's learning process.

Our data preparation pipeline consists of two main steps: (1) outlier detection and treatment, and (2) feature standardization. Initial analysis revealed the presence of significant outliers across multiple features, particularly in residual sugar, chlorides, and sulphates. Outliers can significantly affect the mean and standard deviation used in standardization, potentially distorting the transformation. We identify outliers using the Interquartile Range (IQR) method, which is robust to extreme values. Outliers are capped (rather than removed) to preserve the sample size, ensuring we retain all 1,599 samples for model training. Subsequently, we apply standardization (Z-score normalization), which transforms features to have zero mean and unit variance according to the formula:

$$z = \frac{x - \mu}{\sigma}$$

where  $z$  is the standardized value,  $x$  is the original value,  $\mu$  is the mean, and  $\sigma$  is the standard deviation of the feature.

The normalized dataset<sup>2</sup> contains all original features standardized to have zero mean and unit variance, with outliers appropriately treated. This dataset will serve as the baseline for all classification models (SVM, ANN, and Random Forest), ensuring that feature scaling does not bias the learning process and enabling fair comparison across different algorithms.

<sup>2</sup>This dataset is available in our GitHub repository ([https://github.com/modie25/ml\\_f25\\_project](https://github.com/modie25/ml_f25_project)) for replication purposes.

## Feature Interactions

The correlation analysis revealed several strong pairwise relationships between features ( $|r| > 0.5$ ), suggesting that these variables may interact in ways that influence wine quality. While linear models can capture individual feature effects, they may miss how the combination of features together affects the outcome. For example, the effect of fixed acidity on wine quality may depend on the level of citric acid present. To address this, we create multiplicative interaction features for the strongly correlated pairs identified in the correlation matrix. Specifically, we create interaction terms for: fixed acidity  $\times$  citric acid (0.67), fixed acidity  $\times$  density (0.67), total sulfur dioxide  $\times$  free sulfur dioxide (0.67), fixed acidity  $\times$  pH (-0.68), citric acid  $\times$  volatile acidity (-0.55), citric acid  $\times$  pH (-0.54), and alcohol  $\times$  density (-0.50). The dataset with interactions<sup>3</sup> contains all original standardized features plus these seven interaction terms, expanding the feature space from 11 to 18 features. This dataset allows us to evaluate whether explicitly modeling feature interactions improves classification performance compared to the baseline normalized dataset.

## Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms the original features into a new set of uncorrelated variables called principal components. These components are ordered by the amount of variance they explain in the data. PCA serves two purposes in our analysis: (1) visualization of high-dimensional data in lower-dimensional space (2D/3D), and (2) as an alternative dataset for model training, allowing us to compare classification performance with and without dimensionality reduction.

We apply PCA to the normalized dataset to ensure that all features contribute equally to the principal components. The scree plot and cumulative explained variance plot (see Figure A3 in the appendix) reveal how many components are needed to capture the majority of the variance in the data. The analysis shows that 7 components are required to retain 90% of the variance, while 9 components capture 95% of the total variance. This suggests that dimensionality reduction is feasible, as most information can be preserved with fewer components than the original 11 features. The visualization in 2D and 3D space helps us understand the separability of different quality classes in the reduced-dimensional space.

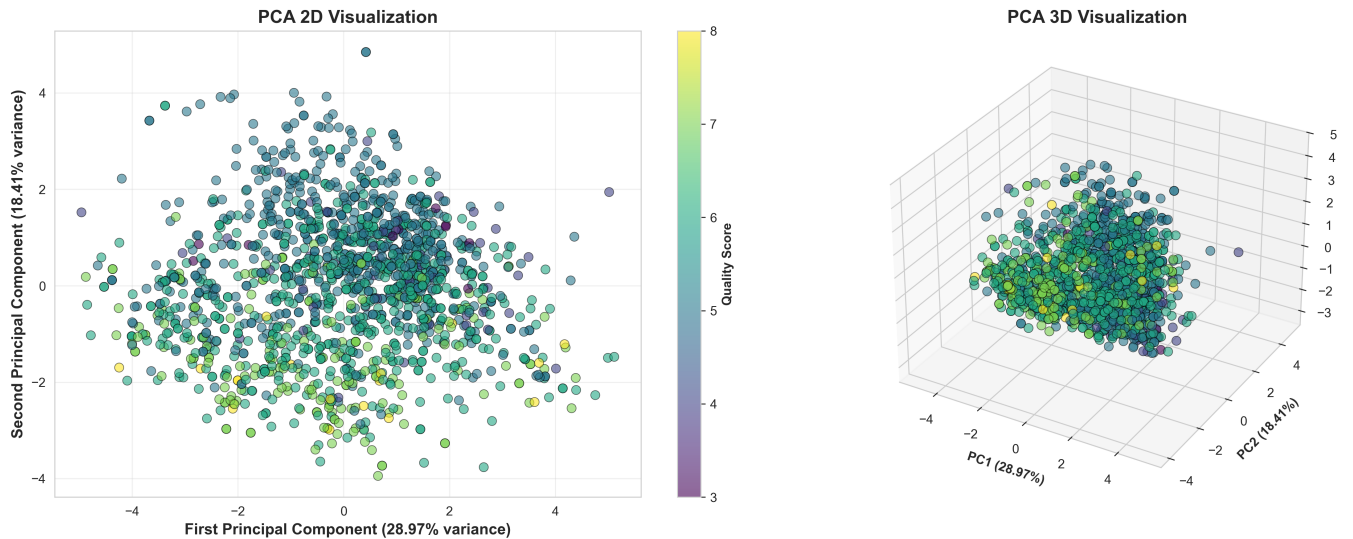


Figure 2: PCA Visualization - Left: 2D projection showing first two principal components; Right: 3D projection showing first three principal components

The 2D and 3D projections show that different quality classes are largely overlapping, with no clear linear separation between quality levels. This indicates that the first few principal components capture variance related to physico-chemical properties but do not directly correspond to quality discrimination, suggesting that non-linear classification methods may be necessary to effectively distinguish between quality classes. The PCA-transformed dataset<sup>4</sup>, created

<sup>3</sup>This dataset is also available in our GitHub repository ([https://github.com/modie25/ml\\_f25\\_project](https://github.com/modie25/ml_f25_project)).

<sup>4</sup>This dataset is also available in our GitHub repository ([https://github.com/modie25/ml\\_f25\\_project](https://github.com/modie25/ml_f25_project)).

using principal components that retain 95% of the total variance, will be used as an alternative input for classification models. This allows us to compare performance with the normalized dataset and assess whether dimensionality reduction improves or hinders classification accuracy.

## Section 3: Methods

Following the exploratory data analysis and data preparation, we now turn to the implementation of three classification methods: Support Vector Machine (SVM), Artificial Neural Networks (ANN), and Random Forest. Each method offers distinct advantages for handling the non-linear relationships and class imbalance observed in the wine quality dataset. We evaluate each method on the three preprocessed datasets (normalized baseline, PCA-transformed, and interaction features) to assess their performance across different feature representations.

### 3.1 Support Vector Machine Method

Support Vector Machine (SVM) is a powerful classification method that seeks to find an optimal separating hyperplane by maximizing the margin between different classes. The method works by identifying support vectors—the data points closest to the decision boundary—and constructing a hyperplane that maximizes the distance to these points. For non-linearly separable data, SVM uses kernel functions to map the data into a higher-dimensional space where linear separation becomes possible. A detailed mathematical formulation of SVM, including the hard margin and soft margin optimization problems, is provided in the appendix (see A5).

We implement SVM using scikit-learn’s `SVC` class with a radial basis function (RBF) kernel, defined as  $k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$ , which allows for non-linear decision boundaries. For our multi-class wine quality classification problem (6 classes: quality scores 3-8), we use the one-versus-rest (OvR) strategy, training one binary classifier per class.

We evaluate SVM performance on three preprocessed datasets: (1) the normalized baseline dataset (11 features), (2) the dataset with interaction features (18 features), and (3) the PCA-transformed dataset (9 principal components retaining 95% variance). For each dataset, we perform an 80-20 train-test split using stratified sampling to preserve class distribution.

Hyperparameter tuning is performed using grid search with 5-fold cross-validation. For each parameter combination, the model is trained on 4 folds and validated on the remaining fold, repeating this process 5 times. The combination yielding the highest average cross-validation macro F1-score is selected as optimal. Grid search optimizes for macro F1-score rather than accuracy to address class imbalance in the dataset. Macro F1-score gives equal weight to all classes, ensuring the model performs well across both majority and minority classes, rather than optimizing primarily for classes 5 and 6 which contain most of the data.

The regularization parameter  $C$  is evaluated over  $\{0.1, 1, 10, 100\}$ , controlling the trade-off between maximizing the margin and minimizing classification errors. Larger values (e.g., 100) penalize misclassifications more heavily, resulting in a smaller margin but fewer training errors, while smaller values (e.g., 0.1) prioritize a larger margin for better generalization.

The kernel parameter  $\gamma$  is evaluated over  $\{\text{'scale'}, \text{'auto'}, 0.001, 0.01, 0.1, 1\}$ , determining the influence of training examples on the decision boundary. When  $\gamma = \text{'scale'}$ , it is computed as  $\gamma = 1/(n_{\text{features}} \times \text{var}(X))$ , adapting to both dimensionality and data scale. When  $\gamma = \text{'auto'}$ , it is set to  $\gamma = 1/n_{\text{features}}$ , considering only the number of features. Numeric values are fixed, with larger values (e.g., 1) creating more complex, localized boundaries and smaller values (e.g., 0.001) producing smoother, more generalized boundaries.

This parameter grid results in  $4 \times 6 = 24$  unique combinations evaluated per dataset. The best model from cross-validation is evaluated on the held-out test set. Performance is assessed using accuracy, macro F1-score, precision, recall, and confusion matrices, with both test set and cross-validation results (mean macro F1-score  $\pm$  standard deviation) reported to assess model stability.

### 3.2 Artificial Neural Network

The multiple layers of nodes characteristic of Artificial Neural Networks (ANN) can handle non-linearly separable datasets. The networks have approximately complex functions by generating a prediction that is the result of staked

layers feed forward from input to hidden layer, to the final output layer. A benefit of a ANN is they do not rely on fixed kernel (SVM) or tree splits (RF). Rather, ANNs can transform multiple input layers with nonlinear activations (see appendix [INSERT WHATEVER APPENDIX] for a more detailed explanation of ANN mechanism). This method has the potential to address complex, nonlinear interactions in the wine quality dataset from continuous chemical features like acidity and sulfates where interactions and combinations of these features might be associated with a distinct wine quality.

The ANN was performed on each of the three preprocessed datasets. It also used 5-fold cross-validation and re-evaluates the models’ best performances across 5 folds for generalization and robustness to prevent overfitting. The regularization parameter alpha is the coefficient for L2 regularization (ridge penalty) and is evaluated over  $\{.00001, .0001, .001\}$ . The alpha helps curb overfitting by discouraging large weight values. We used test values on a logarithmic scale, aiming toward smaller values to account for the small size of the dataset that could likely benefit most from light regularization. Additionally, models were evaluated on over three different activation functions “logistic” (sigmoid), “tahn”, “ReLU”} to determine how the transformations influenced overall model performance. Sigmoid maps the inputs into probabilities, but might exhibit slower learning and difficulty in deeper networks if nets from earlier nodes are in very small ranges. Tanh provides zero-centered outputs to improve hidden layer learning, and ReLU retains strong linear relationships that are likely to affect wine quality (such as alcohol content) by passing positive signals. The learning rate was evaluated to be over  $\{.0005, .001, .002\}$  to include the .001 standard along with a slower lower bound to avoid oscillations, and a larger upper bound to test if accuracy could still be maintained with larger steps. The regularization coefficient, activation function, and hidden-layer configuration were all optimized through grid search to determine which network performed best on each of the datasets.

### 3.3 Random Forest

The Random Forest model is an ensemble model consisting of hundreds or even thousands of decision trees, each trained on a slightly different set of the observations. In order to understand Random Forest, we must first understand the Decision Tree model.

A decision tree recursively splits data based on features that best reduce impurity, for example using Gini impurity or entropy. This model will use Gini impurity. Trees are easy to interpret but tend to overfit – they split until samples are perfectly separated, resulting in trees that memorize the data including noise.

Random Forest addresses this overfitting problem by combining many trees into an ensemble, where each tree is trained on a bootstrap sample of the data, which is a randomly sampled dataset with replacement. It also will only consider a random subset of features at each split. These two methods introduce randomness, ensuring that the trees are diverse, and the final prediction is made by taking the majority prediction across all trees.

The ensemble approach is robust because it significantly reduces variance, leading to a more stable model that is resistant to overfitting. Random Forests also handle nonlinear relationships and outliers well. However, their performance can be sensitive to the hyperparameters they are trained on, which includes things like number of trees, maximum depth, and more.

## Section 4: Results and Analysis

The performance metrics chosen included accuracy, macro F1, and the scikit-learn classification report, which includes precision, recall, f1-score, and support for each of the classes and overall. Accuracy is correct predictions divided by all predictions. Precision is the number of true positives divided by all predicted positives (true and false positives). Recall is the number of true positives divided by all true positives (true positives and false negatives). F1 score is  $2 * \text{precision} * \text{recall} / (\text{precision} + \text{recall})$ . Macro F1 score considers the F1 scores of each individual class and equally weights them. This was selected because a class imbalance is present in the dataset, so we should consider the performance of our models with respect to all classes, rather than optimizing for the classes with most data.

### 4.1 Support Vector Machine Results

We evaluate SVM performance across three preprocessed datasets to assess the impact of different feature representations on classification performance. Table 1 summarizes the performance metrics for each dataset.

**Table 1: SVM Performance Comparison Across Datasets**

Dataset	Accuracy	Macro F1-Score	CV Macro F1 (mean $\pm$ std)	Best C	Best gamma
Normalized	0.622	0.316	$0.363 \pm 0.061$	10	0.1
PCA	0.638	0.410	$0.364 \pm 0.081$	100	auto
Interactions	0.616	0.383	$0.349 \pm 0.081$	100	0.1

The PCA-transformed dataset achieves the highest macro F1-score of 0.410, followed by the interactions dataset (0.383) and the normalized baseline (0.316). The macro F1-score calculates the F1-score for each class independently and then averages them with equal weight, making it particularly important for imbalanced datasets as it ensures all classes contribute equally to the overall metric. This optimization strategy prioritizes balanced performance across all quality classes rather than maximizing accuracy on the majority classes (5 and 6).

While PCA achieves the best macro F1-score, it also shows higher variability in cross-validation results (standard deviation of 0.081). The normalized dataset provides the most stable performance (standard deviation of 0.061), though with lower macro F1-score. The interactions dataset shows intermediate performance in both macro F1-score and stability (standard deviation of 0.081).

The optimal hyperparameters vary across datasets: PCA and interactions both benefit from stronger regularization ( $C = 100$ ), while normalized performs best with moderate regularization ( $C = 10$ ). For the  $\gamma$  parameter, normalized uses  $\gamma = 0.1$ , interactions uses  $\gamma = 0.1$ , and PCA uses  $\gamma = \text{'auto'}$ , which automatically sets  $\gamma = 1/n_{\text{features}}$  based on the number of features. A detailed analysis of class-wise performance, including confusion matrices, is provided in the appendix (Figure A4).

## 4.2 Artificial Neural Network

The performances of each of the preprocessed datasets using ANN are shown in Table 2.

**Table 2: ANN Performance Comparison Across Datasets**

Dataset	Accuracy	Macro F1	CV Macro F1 (mean $\pm$ std)
Normalized	0.597	0.292	$0.297 \pm 0.059$
PCA	0.597	0.276	$0.289 \pm 0.020$
Interactions	0.606	0.299	$0.299 \pm 0.058$

Although accuracy remained around 0.60 for all datasets, the macro F1 scores were around half the accuracy (between 0.28 and 0.3). The class imbalance of the original set emphasizes ANN’s difficulty in predicting minority wine classes (INSERT THE MINORITY WINE CLASSES, TALK ABOUT CONFUSION MATRIX). Overall, the ANN achieved the best performance on the interaction dataset with an accuracy of 0.606, and macro F1 of 0.299 followed by the normalized baseline (0.292) and PCA (0.276). Due to ANNs’ reliance on hidden layers to learn nonlinear combinations, leveraging the interactions data with the highest correlations likely boosted the model with richer raw features that explicitly encode potential relationships. This provides richer signals for activation functions to map into class-specific outputs. Interaction terms can also make class boundaries more distinct, making patterns easier to identify and learn for an ANN with only 1,600 samples.

The best parameters on both the normalized and interaction dataset were tanh activation function with low regularization and a slightly faster learning rate (alpha=.00001 and learning rate= 0.002). This is compatible with the centered, standardized dataset because Tanh is symmetric around zero (outputs [-1,1]) and aligns well with those properties. Alternately, the PCA dataset achieved its best performance with ReLU activation but required stronger regularization (alpha=0.001).

## 4.3 Random Forest

We first ran a baseline random forest model on our baseline (cleaned and normalized) dataset. This was done using the default scikit-learn parameters, which notably include: - 100 trees - No maximum depth of tree - Considers

sqrt(number of features) when looking for the best split - Gini impurity for Shannon information gain - No class weights

The random forest baseline model performs well on the most common wine qualities (5 and 6), achieving F1 scores of above 0.66, meaning that it's doing a pretty good job at correctly identifying the class while balancing false positives and false negatives. It struggles more with the minority classes (3, 4, and 8) due to the class imbalance present. The overall accuracy of 68.4% is pretty strong as a baseline for this dataset, but further fine-tuning is needed to improve performance further. However, the overall macro F1 score of 0.40 indicates that the model is again, performing poorly across all classes, indicating issues with minority classes.

We next evaluated random forest model performance across our three preprocessed datasets (normalized, PCA, and interaction term) and fine-tuned and chose hyperparameter values using grid search cross validation. The hyperparameter grid allows us to control aspects of the random forest model, like the ones mentioned previously. The grid chosen was: - "n\_estimators": [50, 100, 150, 200]. This is the number of trees the model forms. - "max\_depth": [None, 15, 25, 50]. This is the depth of tree the model forms (how many layers) - "class\_weight": [None, "balanced"]. No class\_weight does nothing. A balanced class weight considers each class equally.

Table 3 contains results from the Grid Search Cross Validation on each of the three datasets. Of the three datasets, the dataset containing interaction terms performed best in terms of test accuracy and test macro F1 score. Normalized performed next best, and our dataset using PCA performed worse of the three. However, the normalized dataset provides the most stable results (0.02 macro F1 standard deviation). Interaction term data was the next best, at 0.05 standard deviation, and PCA was least stable with 0.07 standard deviation. From this information, we can conclude that PCA seems to hurt random forest performance, whereas interaction terms help. PCA reduces dimensionality and turns the feature space into linear components, which removes the nonlinear structure that random forest relies on. Interaction terms expand the feature space with more nonlinear relationships, which random forest can exploit to improve performance (although at the cost of slightly higher variance).

The model trained on the interaction term dataset chose the following hyperparameters: - Class\_weight: balanced - Max\_depth: 15 - N\_estimators: 100

This tells us that using a balanced class weight helps with our problem where the model was performing well on the classes with more data, but not as prioritizing the classes with lesser data. Interaction terms increase risk of overfitting, but the maximum depth of 15 helps combat this.

Overall, Random Forest performed best on the interaction term dataset and balanced class weighting, confirming that adding nonlinear features and using class weights improves performance without hurting accuracy.

**Table 3: Random Forest Performance Comparison Across Datasets**

Dataset	CV Macro F1 (mean)	CV Macro F1 (std)	Test Accuracy	Test Macro F1
Normalized	0.3359	0.0208	0.6594	0.3923
PCA	0.3625	0.0693	0.6562	0.3880
Interactions	0.3477	0.0491	0.6844	0.4018

## Section 5: Conclusions

Although test accuracy and Macro F1 were lowest for the ANN model, the variability across folds was among the lowest consistently for each dataset

Smaller dataset with high class imbalance

"Scaling consistency: Ensure features are standardized (e.g., StandardScaler) fit only on the training split."??

Include overall thoughts on benefits of interaction term

## References

Zhang, Jinsong . (2025, Oct. 14). Support Vector Machine [PowerPoint slides]. Washington University in St.Louis.

Zhang, Jinsong . (2025, Oct. 28). Artificial Neural Netowrks. [PowerPoint slides]. Washington University in St.Louis.  
Zhang, Jinsong . (2025, Nov. 18). Random Forest Model. [PowerPoint slides]. Washington University in St.Louis.

## Appendix: Additional Visualizations

### A1. Statistical Summary Table

The following table provides comprehensive descriptive statistics for all features in the dataset, including count, mean, standard deviation, minimum, maximum, and quartiles.

Statistical Summary of Features												
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
Count	1599.0	1599.0	1599.0	1599.0	1599.0	1599.0	1599.0	1599.0	1599.0	1599.0	1599.0	1599.0
Mean	8.32	0.53	0.27	2.54	0.09	15.87	46.47	1.0	3.31	0.66	10.42	5.64
Std Dev	1.74	0.18	0.19	1.41	0.05	10.46	32.9	0.0	0.15	0.17	1.07	0.81
Min	4.6	0.12	0.0	0.9	0.01	1.0	6.0	0.99	2.74	0.33	8.4	3.0
25%	7.1	0.39	0.09	1.9	0.07	7.0	22.0	1.0	3.21	0.55	9.5	5.0
Median	7.9	0.52	0.26	2.2	0.08	14.0	38.0	1.0	3.31	0.62	10.2	6.0
75%	9.2	0.64	0.42	2.6	0.09	21.0	62.0	1.0	3.4	0.73	11.1	6.0
Max	15.9	1.58	1.0	15.5	0.61	72.0	289.0	1.0	4.01	2.0	14.9	8.0

Figure 3: Statistical Summary of Features

### A2. Feature Distributions

The following visualization provides a comprehensive view of the distribution of each feature, including symmetry, potential outliers, and other characteristics that complement the statistical summary table.

### A3. PCA Variance Analysis

The scree plot and cumulative explained variance plot provide detailed information about the variance explained by each principal component and the number of components needed to retain a specified percentage of the total variance.

### A4. Confusion Matrices for SVM

The confusion matrices reveal consistent patterns across all three datasets. The model performs well on the majority classes (quality 5 and 6), which together represent approximately 82% of the dataset. However, the model struggles significantly with minority classes (quality 3, 4, and 8), achieving near-zero precision and recall for these classes. This reflects the class imbalance inherent in the dataset, where quality scores 3, 4, and 8 represent only 0.6%, 3.3%, and 1.1% of samples, respectively.

For the normalized dataset, the model correctly classifies 95 samples of quality 5 and 81 samples of quality 6. The PCA dataset shows similar performance with 99 correct classifications for quality 5 and 79 for quality 6, and achieves the best performance for quality 7 with 24 correct classifications. The interactions dataset correctly classifies 99 samples of quality 5 and 74 samples of quality 6.

A common pattern across all datasets is confusion between adjacent quality levels, particularly between classes 5 and 6. The normalized dataset misclassifies 34 true quality 5 samples as quality 6, and 35 true quality 6 samples as quality 5. The PCA dataset shows 26 misclassifications of quality 5 as 6, and 37 misclassifications of quality 6 as 5.

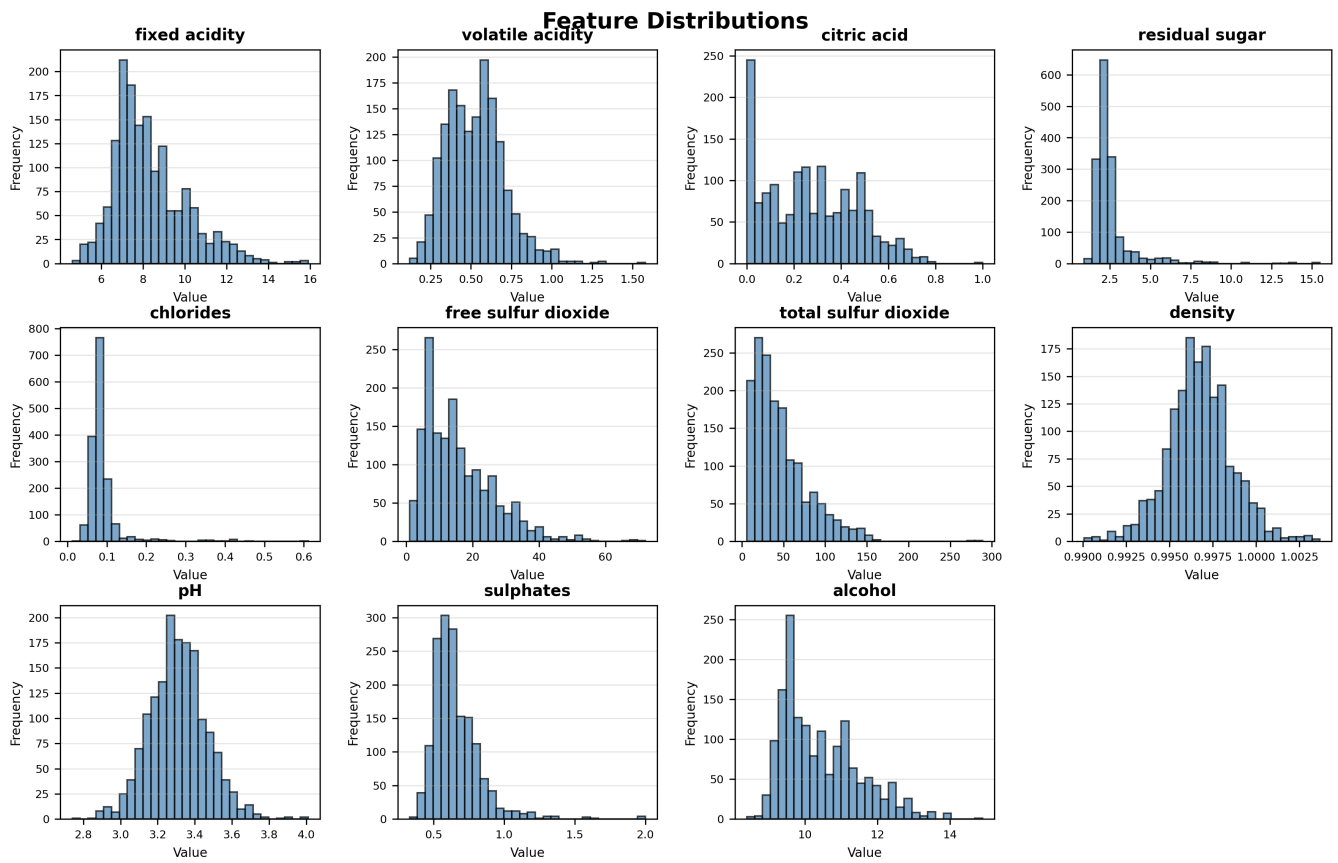


Figure 4: Feature Distributions

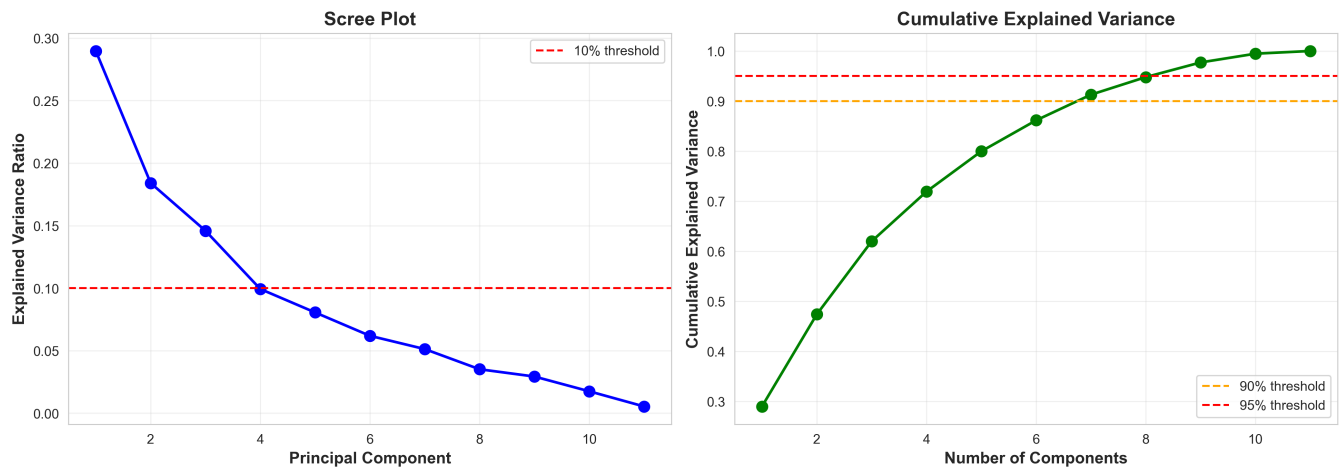


Figure 5: PCA Variance Analysis

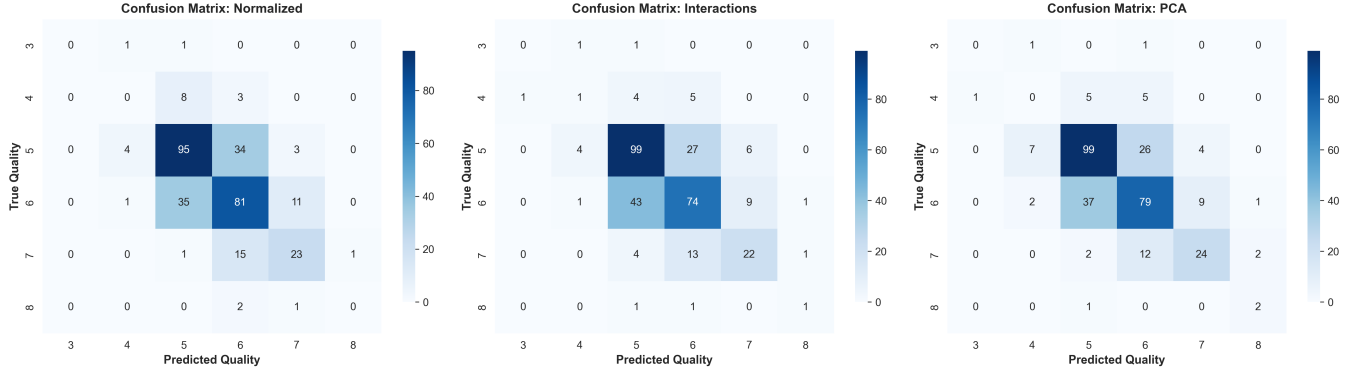


Figure 6: Confusion Matrices

The interactions dataset exhibits the highest confusion between these classes, with 27 misclassifications of quality 5 as 6, and 43 misclassifications of quality 6 as 5. This pattern suggests that distinguishing between adjacent quality levels remains challenging even with macro F1 optimization.

The model’s conservative predictions for class 7 (23-24 correct classifications across datasets) further highlight the challenge of distinguishing between adjacent quality levels. Classes 3, 4, and 8 show near-complete misclassification, with most instances being predicted as adjacent classes (primarily 4, 5, or 6).

## A5. Mathematical Formulation of Support Vector Machine

Support Vector Machine (SVM) seeks to find an optimal separating hyperplane by maximizing the margin between different classes. Given a training data set  $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ , where  $\mathbf{x}_i \in \mathcal{R}^d$  and  $y_i \in \{-1, +1\}$  for binary classification, we seek a linear model classifier in the form:

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b$$

where  $\phi(\mathbf{x})$  denotes a fixed feature-space transformation,  $\mathbf{w}$  is the weight vector, and  $b$  is the bias parameter. For a binary linearly separable data set, there exists at least one choice of  $\mathbf{w}$  and  $b$  that satisfies:

$$y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) > 0, \quad i = 1, \dots, N$$

The margin of a hyperplane is defined as the geometric distance of the closest point in the data set to the hyperplane, given by:

$$\gamma = \min_i \frac{y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b)}{\|\mathbf{w}\|}$$

Since rescaling of  $\mathbf{w}$  and  $b$  does not change the hyperplane, we can use this freedom to produce constraints such that the margin becomes  $\gamma = 1/\|\mathbf{w}\|$ . The maximum margin solution is found by solving the optimization problem:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$

subject to:

$$y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) \geq 1, \quad i = 1, 2, \dots, N$$

This is a quadratic programming problem. For non-linearly separable data sets, we extend this to the soft margin formulation by introducing slack variables  $\xi_i \geq 0$ :

$$\arg \min_{\mathbf{w}, b, \boldsymbol{\xi}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i$$

subject to:

$$y_i(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, 2, \dots, N$$

where  $C > 0$  is a regularization parameter that controls the trade-off between maximizing the margin and minimizing classification errors.