Section 1: Introduction

Machine learning is an application of Artificial Intelligence that allows for computer systems to learn and improve from experience without explicit programming. Computer systems use algorithms and statistical models to perform a specific task, such as classification or regression, using the patterns in data and inference. Algorithms build a mathematical model based on sample (training) data to predict an outcome. 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 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 train on the following versions of the dataset which incorporate data cleaning, feature normalization, and feature extraction: Cleaned, normalized data. This will serve as the baseline model. PCA data Data with interaction terms

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.

By training and fine-tuning three machine learning models on three distinct versions of the wine dataset, we demonstrated how preprocessing choices and hyperparameter optimization can affect predictive performance.

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.[1]

[1]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.

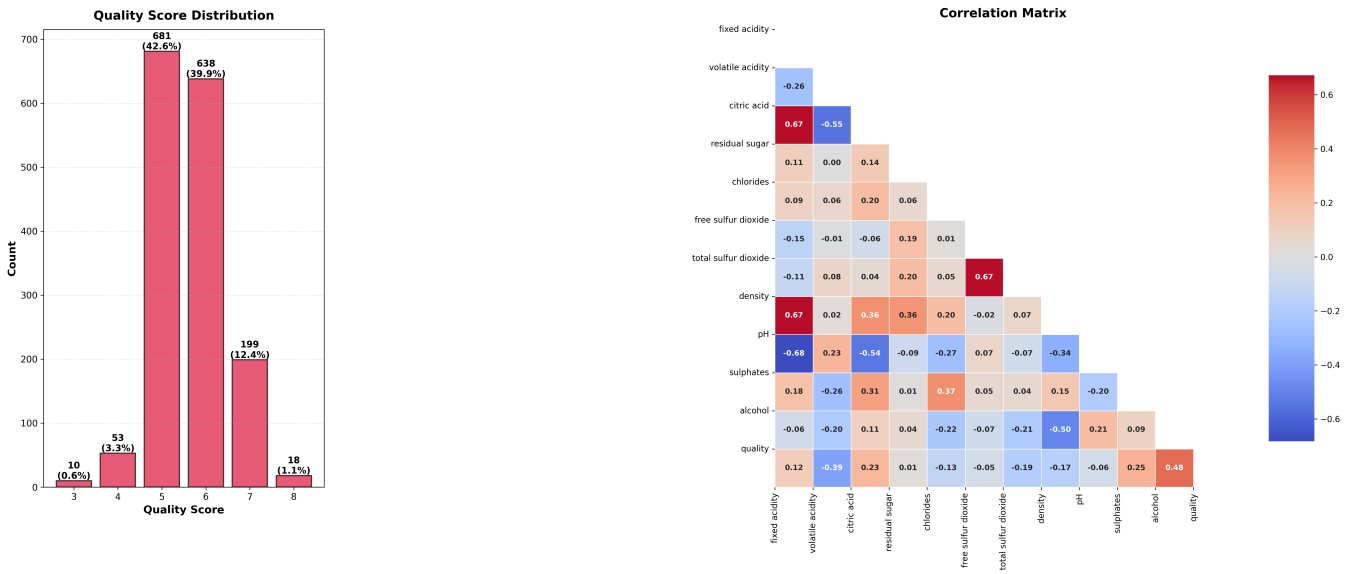

Figure 1: Quality Distribution and Correlation Matrix

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 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, μ is the mean, and σ is the standard deviation of the feature.

The normalized dataset[2] 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.

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 × citric acid (0.67), fixed acidity × density (0.67), total sulfur dioxide × free sulfur dioxide (0.67), fixed acidity × pH (-0.68), citric acid × volatile acidity (-0.55), citric acid × pH (-0.54), and alcohol × density (-0.50). The dataset with interactions[3] 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.

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[4], created 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

All three classification methods (Support Vector Machine, Artificial Neural Network, and Random Forest) were evaluated on the 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

[2]This dataset is available in our GitHub repository (https://github.com/modie25/ml_f25_project) for replication purposes.
[3]This dataset is also available in our GitHub repository (https://github.com/modie25/ml_f25_project).
[4]This dataset is also available in our GitHub repository (https://github.com/modie25/ml_f25_project).

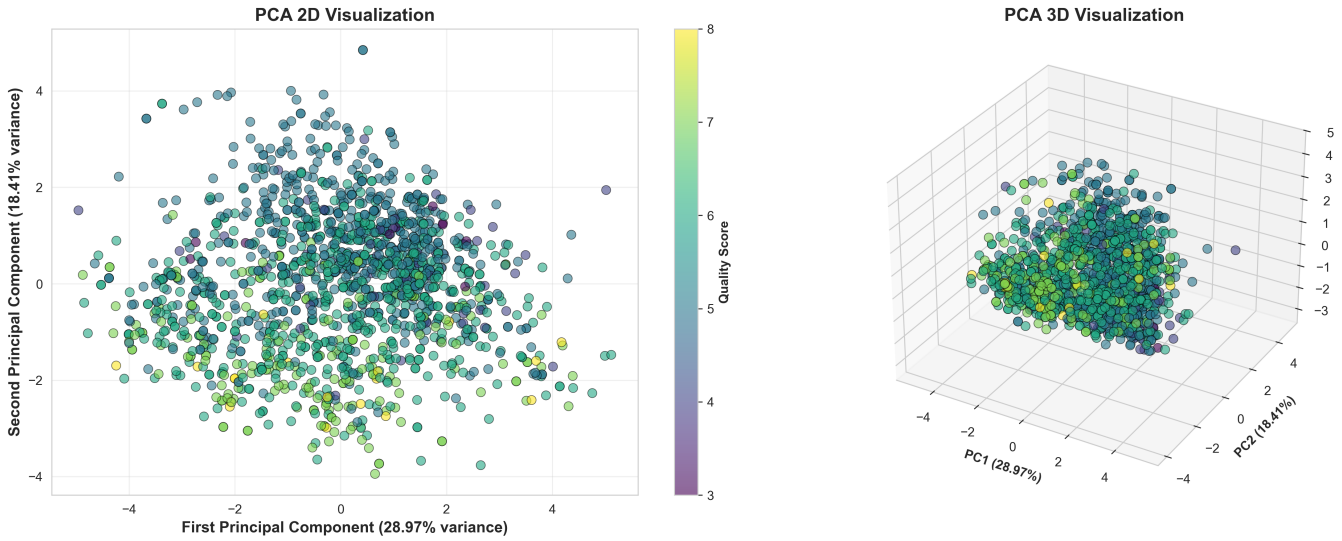

Figure 2: PCA Visualization - Left: 2D projection showing first two principal components; Right: 3D projection showing first three principal components

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, ensuring that model performance is evaluated across all classes rather than primarily optimizing for the majority classes. The best model from cross-validation is evaluated on the held-out test set.

For each classification method, extensive hyperparameter tuning was performed across multiple combinations of kernels, activation functions, solvers, regularization parameters, and other model-specific configurations. As will be shown in the following tables and results sections, only the best-performing models for each method and dataset combination are reported. However, this selection process involved evaluating numerous intermediate configurations through grid search cross-validation, systematically exploring the hyperparameter space to identify optimal settings that maximize cross-validation performance.

3.1 Support Vector Machine

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, evaluating multiple kernel functions to determine the optimal transformation for our data. 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.

The hyperparameters tested using grid search cross-validation are outlined in the table below. We evaluate four kernel types: linear, polynomial, radial basis function (RBF), and sigmoid. The linear kernel creates linear decision boundaries, while polynomial kernels (with degrees 2 and 3) capture polynomial relationships. The RBF kernel, defined as $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$, allows for non-linear decision boundaries and is particularly effective for complex, non-linearly separable data. The sigmoid kernel uses a hyperbolic tangent function and can capture non-linear patterns similar to neural networks.

The regularization parameter C controls 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 γ (for polynomial, RBF, and sigmoid kernels) determines the influence of training examples on the decision boundary. When $\gamma =$ 'scale', it is computed as $\gamma = 1/(n_{\text{features}} \times \text{var}(X))$, adapting to both dimensionality

and data scale. When γ = '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. For polynomial kernels, the degree parameter controls the polynomial order (2 or 3).

SVM Hyperparameter

Hyperparameter	Definition/Purpose	Tested Values
kernel	Type of kernel function used for decision boundaries. Linear creates linear boundaries, polynomial captures polynomial relationships, RBF allows non-linear boundaries, sigmoid uses hyperbolic tangent.	'linear', 'poly', 'rbf', 'sigmoid'
C	Regularization parameter controlling the trade-off between maximizing the margin and minimizing classification errors. Larger values penalize misclassifications more heavily.	0.1, 1, 10, 100
gamma	Kernel coefficient determining the influence of training examples on the decision boundary (for poly, rbf, sigmoid kernels). 'scale' adapts to dimensionality and data scale, 'auto' considers only number of features.	'scale', 'auto', 0.001, 0.01, 0.1, 1 (for rbf); 'scale', 'auto', 0.001, 0.01, 0.1 (for poly, sigmoid)
degree	Polynomial degree for polynomial kernel. Higher degrees create more complex decision boundaries.	2, 3 (for poly kernel only)

3.2 Artificial Neural Network

The multiple layers of nodes characteristic of Artificial Neural Networks (ANN) can handle non-linearly separable datasets. ANNs approximate complex functions by propagating inputs forward through stacked layers of nonlinear transformations. A benefit of an 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 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, airing 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 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 due to saturation at extremes, leading to vanishing gradients. 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.

ANN Hyperparameter

Hyperparameter	Purpose	Tested Values
hidden_layer_sizes	Number of nodes in each hidden layer.	(64,), (128, 64)
activation	activation function used at each node.	'relu', 'tanh', 'logistic'
alpha	Strength of L2 regularization to prevent overfitting.	0.00001,0.0001, 0.001
learning_rate_init	Initial learning rate for weight updates.	0.0005, 0.001, 0.002, 0.005
learning_rate	How the learning rate adjusts during training	'constant', 'adaptive', 'invscaling'
solver	Optimization algorithm used for weight updates.	'adam', 'lbfgs'
max_iter	Maximum number of training iterations.	500, 1000

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 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. This model uses Gini Impurity to measure how mixed classes are within a node. Then, the tree chooses the split that produces the largest reduction in impurity, as measured by information gain. See Appendix for further materials on impurity and gain calculations. 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, as defined by max_features. 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.

The hyperparameters tested for using grid search cross validation are outlined in Table [insert table number]

RF Hyperparameter

Hyperparameter	Definition/Purpose	Tested Values
n_estimators	Specifies the number of trees the model forms. More trees reduce variance but increase training time.	50, 100, 150, 200
max_depth	Controls the maximum depth of each tree the model forms. Limiting depth reduces overfitting by preventing trees from becoming too complex.	None, 15, 25, 50
max_features	Determines how many features each tree considers when searching for the best split. For example, 'sqrt' considers sqrt(number of features).	Sqrt, log2, None

Hyperparameter	Definition/Purpose	Tested Values
class_weight	No class_weight does nothing. A balanced class weight considers each class equally.	None, balanced

Section 4: Results and Analysis

Evaluation Metrics

Performance is assessed using accuracy, macro F1-score, precision, recall, and confusion matrices. Both test set and cross-validation results (mean macro F1-score ± standard deviation) are reported to assess model stability. The scikit-learn classification report provides precision, recall, f1-score, and support for each class and overall. See appendix for normalized confusion matrices.

Accuracy: correct predictions divided by all predictions

Precision: number of true positives divided by all predicted positives (true and false positives)

Recall: number of true positives divided by all true positives (true positives and false negatives)

F1 score: 2 × precision × recall / (precision + recall)

Macro F1 score: considers the F1 scores of each individual class and equally weights them

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.

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 ± std)	Best Kernel	Best C	Best gamma	Best degree
Normalized	0.622	0.316	0.363 ± 0.061	rbf	10	0.1	N/A
PCA	0.638	0.410	0.364 ± 0.081	rbf	100	auto	N/A
Interactions	0.597	0.299	0.355 ± 0.053	poly	10	auto	2

We evaluated four kernel types (linear, polynomial, RBF, and sigmoid) for each dataset. The optimal kernel varied across datasets: RBF kernel achieved the best performance for both the normalized and PCA datasets, while the polynomial kernel (degree 2) performed best for the interactions dataset. This suggests that different feature representations benefit from different kernel transformations. The RBF kernel's ability to create flexible non-linear decision boundaries is particularly effective for the normalized and PCA datasets, while the polynomial kernel's capacity to capture polynomial relationships between features proves more suitable for the interactions dataset, which explicitly encodes multiplicative feature interactions.

The PCA-transformed dataset achieves the highest macro F1-score of 0.410, followed by the normalized baseline (0.316) and the interactions dataset (0.299). 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 the most stable cross-validation performance (standard deviation of 0.053), but achieves the lowest macro F1-score.

The optimal hyperparameters vary across datasets: PCA benefits from stronger regularization ($C = 100$), while normalized and interactions perform best with moderate regularization ($C = 10$). For the γ parameter, normalized uses $\gamma = 0.1$, while both PCA and interactions use $\gamma =$ 'auto', which automatically sets $\gamma = 1/n_{\text{features}}$ based on the number of features. The interactions dataset's optimal polynomial kernel uses degree 2, indicating that quadratic relationships between interaction features are most effective for classification. 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 [Insert table number]

Table 1- ANN Performance Comparison Across Datasets

Dataset	Accuracy	Macro F1	CV Macro F1 (mean ± std)	Activation function	Alpha	Hidden layer size	Learning rate	solver
Normalized	0.653	0.395	0.372 ± 0.041	logistic	0.00001	(128,64)	0.0005	lbfgs
PCA	0.638	0.418	0.369± 0.062	Tanh	0.0001	(64,)	0.0005	lbfgs
Interactions	0.594	0.387	0.353 ± 0.039	logistic	0.00001	(64,)	0.0005	lbfgs

Although accuracy remained around 0.60 for all datasets, the macro F1 scores were around half the accuracy (between 0.28 and 0.3). Overall, the ANN achieved the macro F1 (0.418) on the PCA dataset, while the normalized baseline dataset had the best accuracy (0.653). The interactions dataset had the lowest macro F1 and accuracy. It is possible the PCA dataset with reduced noise and redundancy was easier to train and optimize with lower complexity; this is reflected in the model's best performance with the smallest, singular hidden layer size (64,). Fitting a smaller dataset benefited from a less complex network, leading to more generalizability when tested. This is especially helpful for identifying rarer classes in the dataset. The ANN was able to achieve some precision on class "4" (0.18) and "8" (0.5), which are among two of the three lowest represented qualities in the data (see appendix for confusion matrix). However, these classes were still poorly predicted overall.

Early stopping improved convergence. Additionally, the LBFGS solver was optimal for all models. Unlike Adam, LBFGS uses both the gradient and the hessian to assess slope and sharpness of the loss function, aiding in step adjustment and leading to more efficient convergence. This is especially important for small datasets that are sensitive to noise (Zhang et al., 2026). Utilizing LBFGS drastically improved macro F1-Scores for all forms of the preprocessed dataset. F1-scores improved by approximately 54.3% compared to the best run with the default Adam solver.

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, consider (square root of) the number of features when looking for the best split, and 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 0.406 indicates that the model is again, performing poorly across all classes, indicating issues with minority classes.

Table [insert] contains results from the Grid Search Cross Validation on each of the three datasets, including the selected hyperparameters. Of the three models, the normalized dataset yielded the best overall performance as defined by both test accuracy and test macro F1, at 0.6906 and 0.4132, respectively. It also saw the lowest CV Macro F1 standard deviation, indicating stable performance. This makes sense because normalization preserves the original nonlinear structure of the features. It selected unlimited depth, allowing deep trees to exploit these structures. This

model had None for max_features, allowing it to consider all features rather than a random subset. Considering all features makes sense because on the normalized dataset, each feature contributes meaningful information to wine quality. Balanced class weights help minority classes achieve better performance without hurting overall accuracy.

The PCA dataset performed worst of the three with test accuracy of 0.6562 and macro F1 of 0.388, and it also happened to be unstable, with a standard deviation of 0.0693. This makes sense because PCA transforms data into linear, orthogonal components, eliminating much of the nonlinearity in the original data.

The interaction term dataset performed almost nearly as well as the normalized dataset, with a test accuracy of 0.6844 and macro F1 of 0.4018. It was also quite stable, with a CV Macro F1 standard deviation of 0.049. Good performance on this dataset makes sense because interaction terms expand the feature space, giving the Random Forest more nonlinear relationships to learn. However, more features introduce more variance since a subset of features are chosen at each split.

Table [insert]

Dataset	Test Accuracy	Test Macro F1	CV Macro F1 (mean ± std.)	n_estimators	max_depth	max_features	class_weight
Normalized	0.6906	0.4132	0.3523 ± 0.04	100	None	None	balanced
PCA	0.6562	0.3880	0.3625 ± 0.0693	50	15	sqrt	None
Interactions	0.6844	0.4018	0.3477 ± 0.0491	100	15	sqrt	balanced

Section 5: Conclusions

The purpose of this analysis was to apply three machine learning methods using wine-related features from the dataset to assess which method (and their corresponding combination of hyperparameters) would optimize prediction performance of wine quality. Each method presented unique results, strengths, and limitations. The model that achieved the highest macro F1 was the ANN on the PCA dataset (0.418), which marginally outperformed both FR and SVM. This suggests that dimensionality reduction improved ANN's ability to balance predictions across classes. However, Random Forest model using the interactions data set had the highest two test accuracies of 69.1% on the normalized dataset, followed but 68.4% on the interaction dataset. Random forests are often considered a powerful metric due to their ensemble structure and ability to capture nonlinear relationships. RF tends to favor majority classes, making minority classes ("3" and "4") difficult to predict when the model displayed poor F1-scores. Moreover, with fewer samples, the trees were less diverse. RF's strength in handling nonlinear structures might also explain the weaker performance on the dataset with a PCA transformation; reducing features to linear components undermined the nonlinear structure that RF thrives on, while the addition of interaction terms strengthened it. For SVM and ANN, the interaction dataset might have introduced redundancy and noise that could affect the higher variability in the interaction model for SVM (std=0.081).

Support Vector Machine, Artificial Neural Network, and Random Forest each provided diverse implications and results in their prediction. These results highlight the importance of matching preprocessing strategies to leverage the strengths and potential pitfalls of model mechanisms. Most models assume a uniform distribution of data across all categories, so the presence of class imbalance hurt the performance of all the models. Although interaction terms enrich the feature space, Future iterations of this analysis should explore other strategies such as resampling, or deep learning models like Generative Adversial Networks (GANs) that can simulate realistic samples of minority classes (Jiang et al., 2025).

Method Comparison

Method	Highest Accuracy	Dataset (Accuracy)	Highest Macro F1	Dataset (Macro F1)
SVM	0.638	PCA	0.410	PCA
ANN	0.653	Normalized	0.418	PCA
RF	0.691	Normalized	0.413	Normalized

Contributions: Carlos conducted preprocessing & exploratory data analysis, SVM methods, analysis, and results, and compiled the final paper. Jasmine completed the introduction and contributed to methods, analysis, and results

for random forests. Odalis conducted ANN methods, analysis, results, the conclusion section, and organized meetings and GitHub documentation. All group members contributed to planning, discussion, and editing.

References

Jiang, J., Zhang, C., Ke, L., Hayes, N., Zhu, Y., Qiu, H., Zhang, B., Zhou, T., & Wei, G.-W. (2025). A review of machine learning methods for imbalanced data challenges in chemistry. Chemical Science, 16(18), 7637–7658. https://doi.org/10.1039/d5sc00270b

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.

Zhang, Z., Yuan, G., Qin, Z., & Luo, Q. (2026). An improvement by introducing LBFGS idea into the Adam optimizer for machine learning. Expert Systems with Applications, 296, 129002. https://doi.org/10.1016/j.eswa.2025.129002

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 (RBF kernel), the model correctly classifies 70% of quality 5 samples and 63% of quality 6 samples. The PCA dataset (RBF kernel) achieves the best performance for quality 5 with 73% correct classifications, followed by 62% for quality 6, and 60% for quality 7. The interactions dataset (polynomial kernel, degree 2) correctly classifies 66% of quality 5 samples and 65% of quality 6 samples, but shows lower performance for quality 7 with 45% correct classifications.

A common pattern across all datasets is confusion between adjacent quality levels, particularly between classes 5 and 6. The normalized dataset misclassifies 25% of true quality 5 samples as quality 6, and 27% of true quality 6 samples

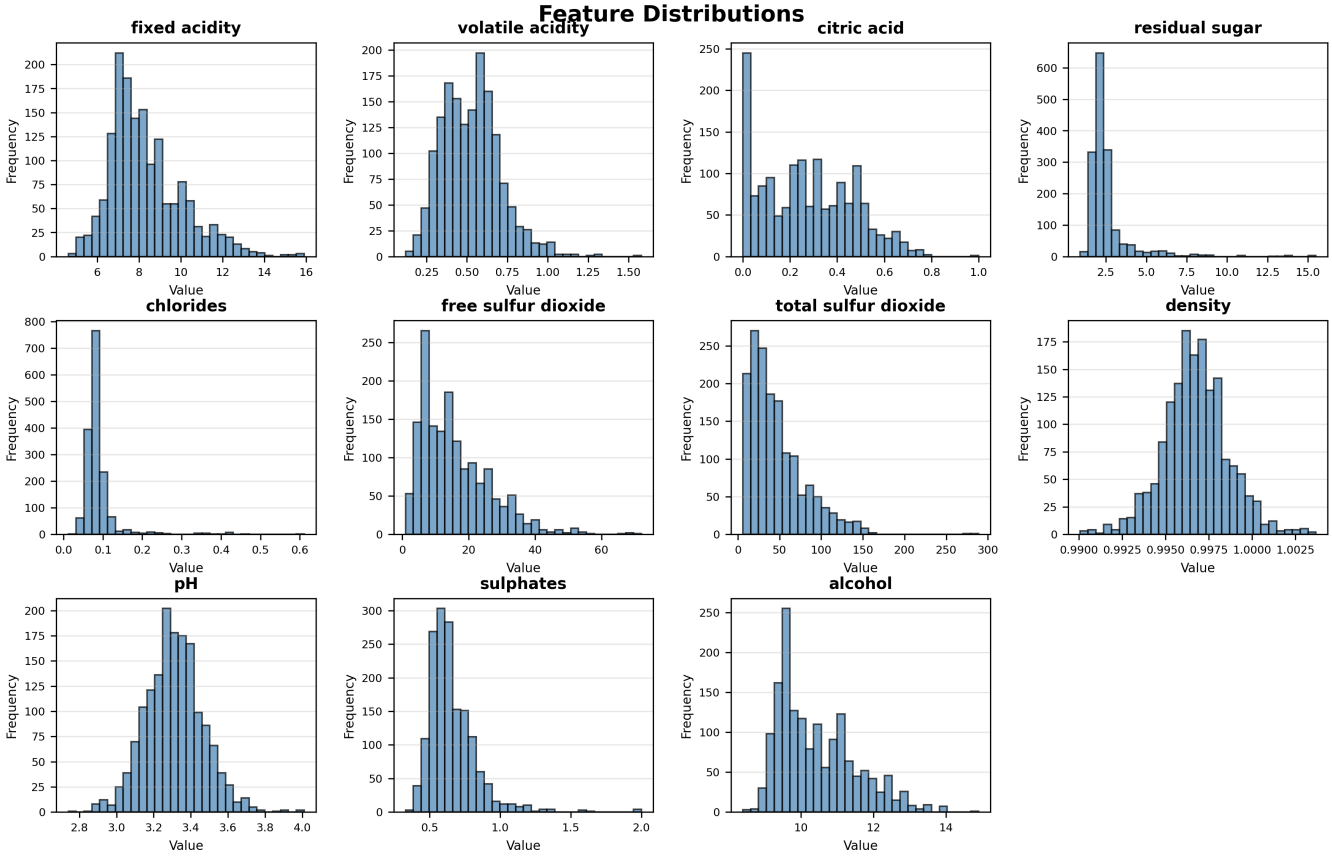

Figure 4: Feature Distributions

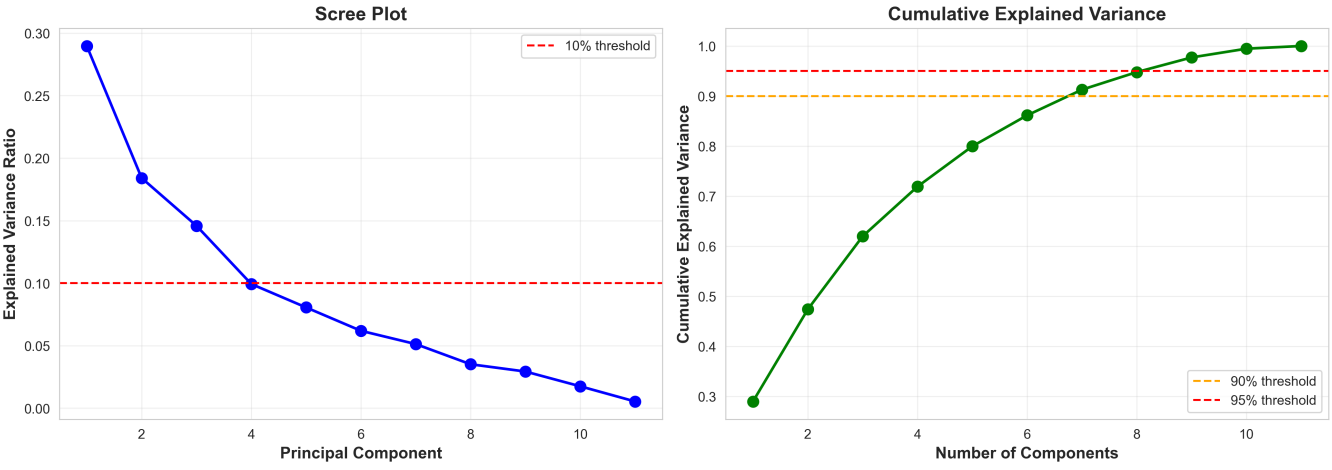

Figure 5: PCA Variance Analysis

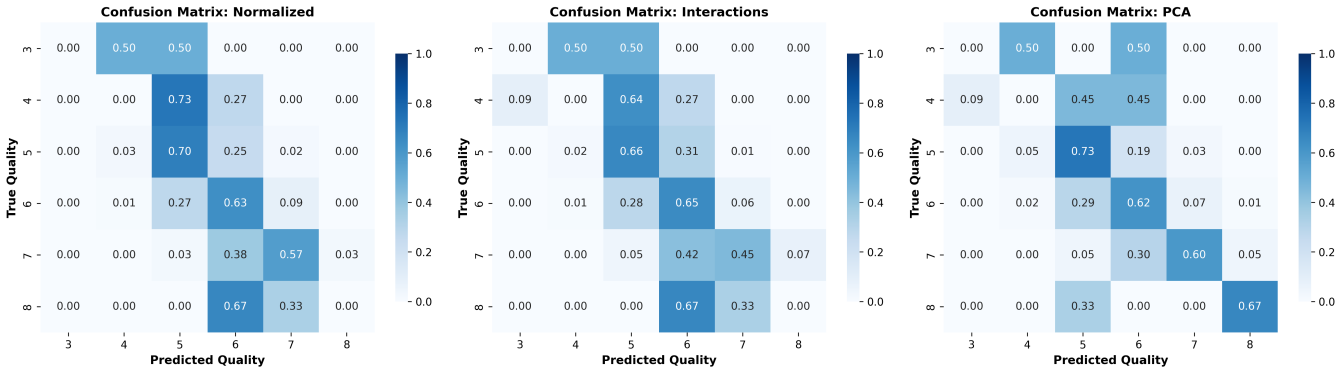

Figure 6: Confusion Matrices

as quality 5. The PCA dataset shows 19% misclassifications of quality 5 as 6, and 29% misclassifications of quality 6 as 5. The interactions dataset exhibits the highest confusion between these classes, with 31% misclassifications of quality 5 as 6, and 28% 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 performance for class 7 varies across datasets: the normalized dataset achieves 58% correct classifications, the PCA dataset achieves 60%, while the interactions dataset shows lower performance at 45%. Classes 3, 4, and 8 show near-complete misclassification, with most instances being predicted as adjacent classes (primarily 4, 5, or 6). Notably, the PCA dataset achieves 67% correct classification for quality 8 (2 out of 3 samples), though this result should be interpreted cautiously given the very small sample size.

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 $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathcal{R}^d$ and $y_i \in \{-1, +1\}$ for binary classification, we seek a linear model classifier in the form:

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

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

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

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

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

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

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

subject to:

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

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

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

subject to:

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

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