Section 3: Methods

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. 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.

Implementation: We implement SVM using scikit-learn's `SVC` class with a radial basis function (RBF) kernel, defined as $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$, 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 accuracy is selected as optimal.

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 γ 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, weighted F1-score, precision, recall, and confusion matrices, with both test set and cross-validation results (mean accuracy $\pm$ standard deviation) reported to assess model stability.

Section 4: Results and Analysis

4.1 Support Vector Machine Results

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

Table 1: SVM Performance Comparison Across Datasets

Dataset	Accuracy	F1-Score (weighted)	CV Accuracy (mean $\pm$ std)	Best C	Best gamma
Normalized	0.675	0.656	0.644 $\pm$ 0.030	10	1
PCA	0.669	0.651	0.640 $\pm$ 0.023	100	1
Interactions	0.606	0.579	0.622 $\pm$ 0.033	1	auto

The normalized baseline dataset achieves the highest performance with 67.5% accuracy and a weighted F1-score of 0.656. The weighted F1-score calculates the F1-score for each class independently and then averages them, weighted by the number of true instances per class. This metric is particularly important for imbalanced datasets, as it accounts for class distribution when evaluating model performance. The PCA-transformed dataset performs similarly (66.9% accuracy, F1-score 0.651), demonstrating that dimensionality reduction from 11 to 9 features retains most discriminative information. The dataset with interaction features shows lower performance (60.6% accuracy, F1-score 0.579), suggesting that the engineered interaction terms do not improve classification for this problem.

Cross-validation results show consistent performance across folds, with standard deviations below 3.5% for all datasets, indicating stable model behavior. The optimal hyperparameters vary across datasets: the normalized dataset benefits from moderate regularization ($C = 10$), while PCA requires stronger regularization ($C = 100$), and interactions perform best with minimal regularization ($C = 1$). For the γ parameter, both normalized and PCA datasets use $\gamma = 1$, while interactions uses $\gamma = \text{'auto'}$, which automatically sets $\gamma = 1/(n_{\text{features}} \times \text{var}(X))$ based on the number of features and variance of the data. A detailed analysis of class-wise performance, including confusion matrices, is provided in the appendix (Figure A1).

Appendix: Additional SVM Visualizations

A1. Confusion Matrices for SVM

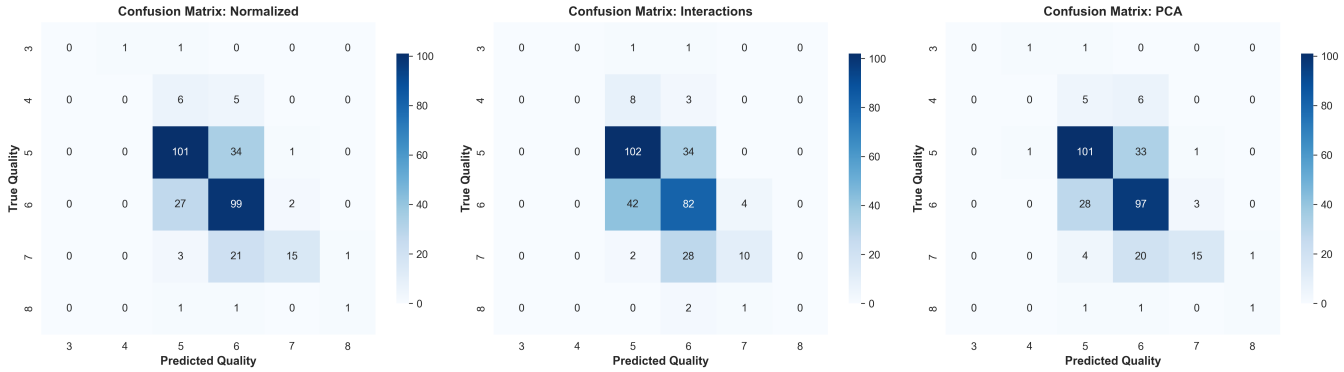

Figure 1: Confusion Matrices

Figure A1: Confusion Matrices for SVM - Left: Normalized dataset; Middle: Interactions dataset; Right: PCA dataset

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. The model's conservative predictions for class 7 (high precision but low recall) further highlight the challenge of distinguishing between adjacent quality levels.

The normalized and PCA datasets show nearly identical confusion patterns, with both correctly classifying approximately 100-101 samples of quality 5 and 97-99 samples of quality 6. The interactions dataset exhibits slightly more confusion between classes 5 and 6, with 42 misclassifications of true quality 6 as quality 5, compared to 27-28 in the other datasets. This increased confusion, combined with lower overall accuracy, suggests that the interaction features do not provide additional discriminative power for this classification task.