Wine Quality Prediction Using Support Vector Machines and Logistic Regression

Nikita Olefir
Matriculation Number: 37003A
Machine Learning Projects [24/25]

October 2, 2025

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying.

---

1 Introduction

This project implements binary classification algorithms from scratch to predict wine quality based on physicochemical properties. Wines are classified as "good" (quality ≥ 6) or "bad" (quality < 6). We implement **Support Vector Machines (SVM)** and **Logistic Regression (LR)** with their kernelized variants (polynomial and RBF kernels) to capture non-linear decision boundaries.

Objectives: (1) Implement SVM and LR from scratch in Python; (2) Extend both with kernel methods; (3) Apply rigorous ML methodology with cross-validation; (4) Compare model performance comprehensively.

2 Theoretical Background

2.1 Logistic Regression

Logistic Regression models the probability of binary outcomes using the sigmoid function:

$$\sigma(z) = \frac{1}{1+e^{-z}}, \quad P(y=1|\mathbf{x}) = \sigma(\mathbf{w}^T \cdot \mathbf{x} + b) \tag{1}$$

The model minimizes binary cross-entropy loss with L2 regularization:

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] + \frac{\lambda}{2m}\|\mathbf{w}\|^2 \tag{2}$$

Gradients for parameter updates:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{1}{m}\mathbf{X}^T(\hat{\mathbf{y}} - \mathbf{y}) + \frac{\lambda}{m}\mathbf{w}, \quad \frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i) \tag{3}$$

2.2 Support Vector Machine

SVM finds a maximum-margin hyperplane using hinge loss. For labels $y \in \{-1, +1\}$:

$$\min_{\mathbf{w},b} \frac{1}{2C}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \tag{4}$$

Hinge loss penalizes margin violations: samples with $y \cdot (\mathbf{w}^T\mathbf{x} + b) < 1$ contribute to the gradient, while correctly classified samples beyond the margin do not.

2.3 Kernel Methods

Kernels map data to higher-dimensional spaces without explicit transformation via the kernel trick:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \cdot \phi(\mathbf{x}_j) \tag{5}$$

Decision function in dual space:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) \tag{6}$$

Implemented kernels:

1. **Linear:** $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \cdot \mathbf{y}$
2. **Polynomial:** $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \cdot \mathbf{y} + c)^d$
3. **RBF:** $K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma\|\mathbf{x} - \mathbf{y}\|^2)$

3 Dataset and Preprocessing

3.1 Dataset Description

The UCI Wine Quality dataset contains 6,497 wines (1,599 red, 4,898 white) with 11 physicochemical features plus wine type. Quality scores range from 3 to 9. We converted this to binary classification: bad (quality < 6, 36.7%) vs good (quality ≥ 6, 63.3%), resulting in a 1.73:1 class imbalance.

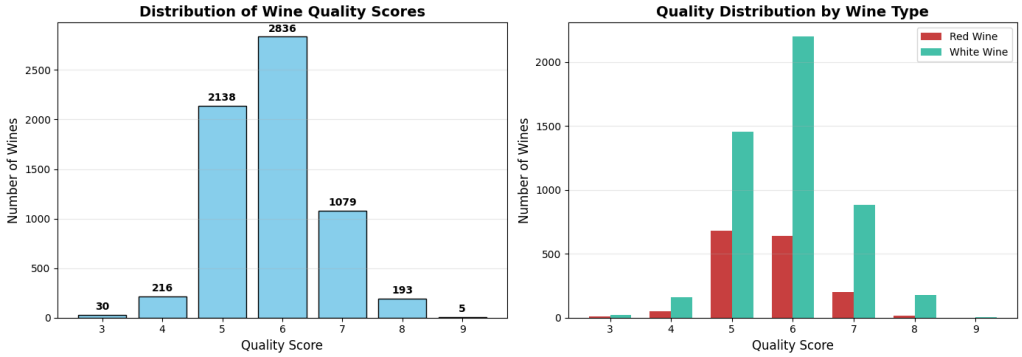

Figure 1: Wine quality distribution showing most wines rated 5 or 6.

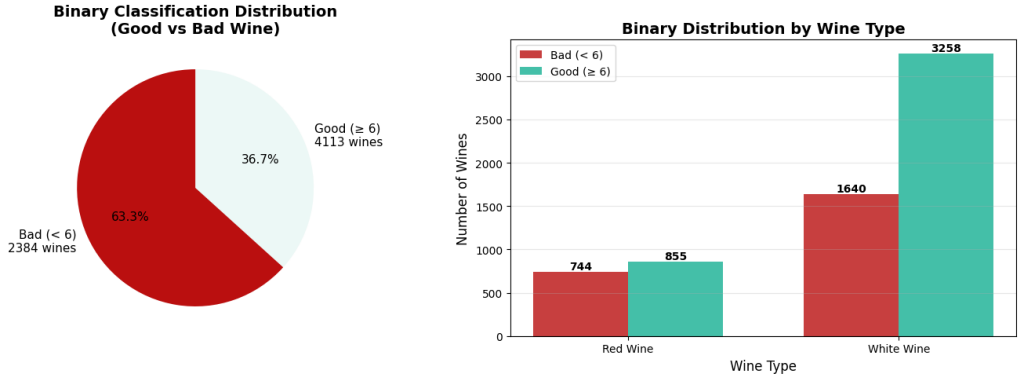

Figure 2: Binary classification distribution (63.3% good vs 36.7% bad).

3.2 Feature Analysis

The dataset contains 12 features describing physicochemical properties and wine type:

Features:

1. **Fixed Acidity** (g/L): Non-volatile acids (tartaric acid)
2. **Volatile Acidity** (g/L): Acetic acid, can cause vinegar taste
3. **Citric Acid** (g/L): Adds freshness and flavor
4. **Residual Sugar** (g/L): Remaining sugar after fermentation
5. **Chlorides** (g/L): Salt content
6. **Free Sulfur Dioxide** (mg/L): Prevents microbial growth
7. **Total Sulfur Dioxide** (mg/L): Total SO_2 (free + bound)
8. **Density** (g/cm^3): Related to alcohol and sugar content

9. **pH**: Acidity level (3-4 for most wines)
10. **Sulphates** (g/L): Wine additive, affects SO_2 levels
11. **Alcohol** (% vol): Alcohol percentage
12. **Wine Type**: Binary (0=red, 1=white)

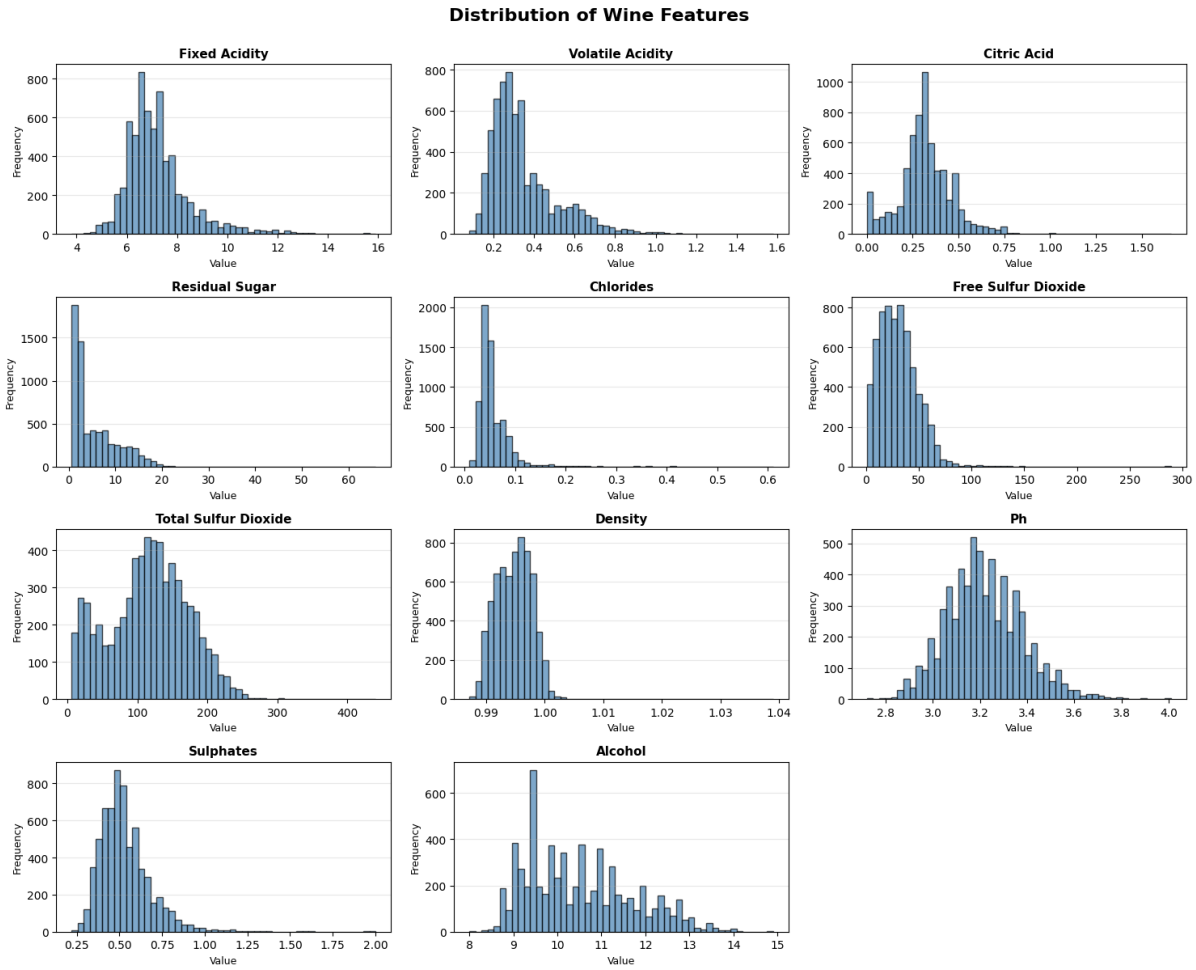

Figure 3: Distribution histograms for all features showing varying skewness patterns.

Distribution patterns observed:

- **Highly right-skewed:** Residual sugar (most wines dry, few sweet), chlorides, free and total sulfur dioxide
- **Moderately right-skewed:** Sulphates, citric acid
- **Approximately normal:** Fixed acidity, volatile acidity, density, pH
- **Multimodal:** Alcohol (distinct peaks at different concentrations)

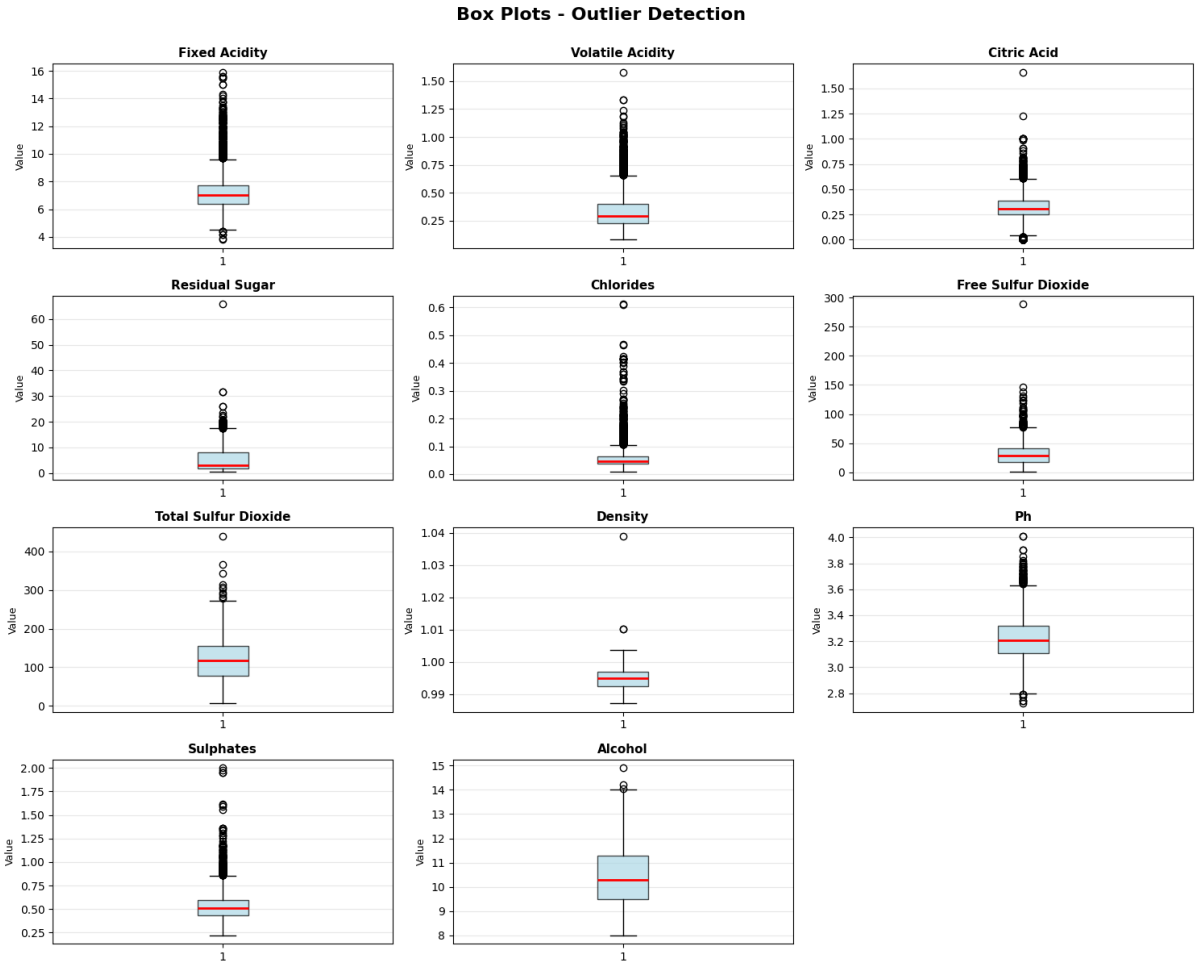

Figure 4: Box plots revealing outliers across features. Residual sugar, chlorides, and sulfur compounds show numerous extreme values.

Outlier analysis: Box plots reveal significant outliers in multiple features. Residual sugar shows the most extreme outliers (some wines with 60+ g/L vs median 2-3 g/L), indicating a few dessert wines. Chlorides and sulfur dioxide compounds also exhibit many outliers. Fixed acidity shows some high outliers (15+ g/L). These outliers likely represent legitimate wines with unusual characteristics rather than measurement errors. Our z-score normalization mitigates outlier impact while preserving relative relationships.

Key observations: Most wines clustered around quality 5-6; very few extreme ratings (30 wines at quality 3, only 5 at quality 9). The right-skewed distributions suggest most wines have moderate levels of sugar, chlorides, and sulfur compounds, with occasional wines having much higher concentrations.

3.3 Data Preprocessing

Stratified train-test split: 80% training (5,199 samples), 20% test (1,298 samples). Stratification maintained class balance in both sets.

Z-score normalization: Features standardized using training set statistics only to prevent data leakage:

$$\mathbf{X}_{norm} = \frac{\mathbf{X} - \mu_{train}}{\sigma_{train}} \tag{7}$$

3.4 Hyperparameter Tuning

5-fold stratified cross-validation used for hyperparameter selection. For Logistic Regression regularization $\lambda \in \{0.001, 0.01, 0.1, 1.0\}$, we found $\lambda = 1.0$ optimal (CV accuracy: 73.84% $\pm$ 1.46%).

Other hyperparameters: LR learning rate 0.1, SVM learning rate 0.001, 1,000 iterations for standard models, 500 for kernel models, SVM $C = 1.0$. Kernel parameters tested: polynomial degrees 2 and 3, RBF gamma values 0.1 and 0.5.

4 Results

4.1 Model Performance

Table 1 summarizes test set performance for all models.

Table 1: Test set performance comparison

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	75.65%	78.24%	85.28%	81.61%
SVM	69.34%	74.04%	79.44%	76.64%
LR - Poly ($d = 2$)	63.87%	83.16%	53.89%	65.39%
LR - Poly ($d = 3$)	65.79%	72.88%	72.88%	72.96%
LR - RBF ($\gamma = 0.1$)	**76.19%**	**77.71%**	**87.47%**	**82.31%**
LR - RBF ($\gamma = 0.5$)	74.19%	74.72%	89.54%	81.46%
SVM - Poly ($d = 2$)	57.47%	81.40%	42.58%	55.91%
SVM - Poly ($d = 3$)	68.80%	75.79%	75.79%	75.47%
SVM - RBF ($\gamma = 0.1$)	68.03%	68.78%	90.63%	78.22%
SVM - RBF ($\gamma = 0.5$)	68.80%	68.57%	92.82%	79.03%

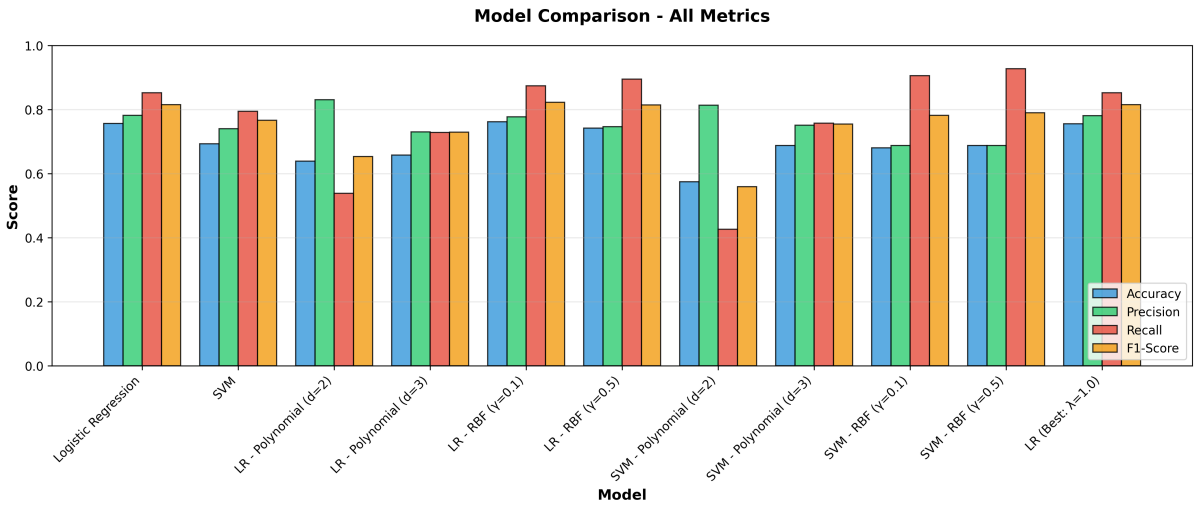

Figure 5: Performance comparison across all models and metrics.

Best model: Kernel LR with RBF ($\gamma = 0.1$) achieved 82.31% F1-score and 76.19% accuracy.

Key findings: (1) LR consistently outperformed SVM (6.3% accuracy advantage); (2) RBF kernels significantly better than polynomial kernels; (3) Lower gamma (0.1) generalized better than higher gamma (0.5); (4) Polynomial degree 2 severely underfit, degree 3 showed

improvement but remained suboptimal; (5) Standard LR competitive with kernel variants, suggesting linear relationships dominate.

4.2 Best Model Analysis

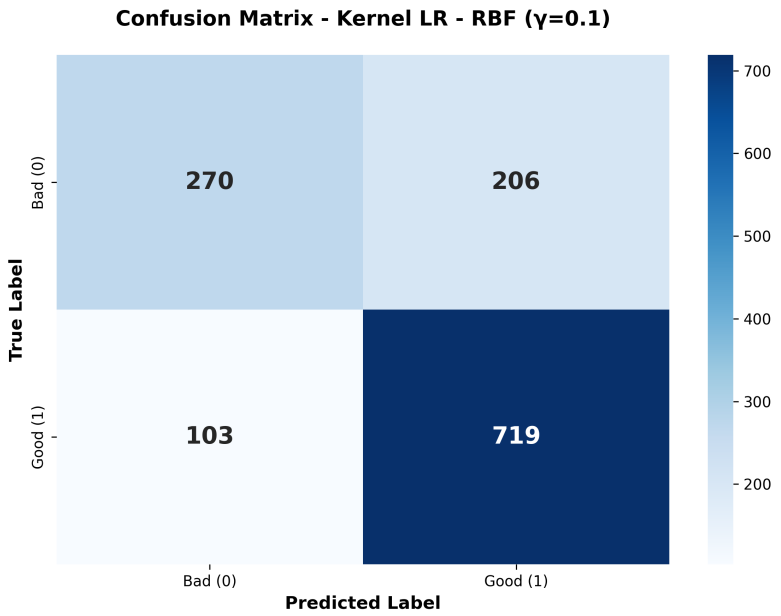

Figure 6: Confusion matrix for best model (Kernel LR with RBF, $=0.1\gamma = 0.1 = 0.1$).

The best model correctly classified 989/1,298 samples (76.19**Error breakdown:**

- True Positives: 719 (correctly identified good wines)
- True Negatives: 270 (correctly identified bad wines)
- False Positives: 206 (bad wines misclassified as good)
- False Negatives: 103 (good wines misclassified as bad)

The 2:1 FP:FN ratio indicates bias toward predicting "good," reflecting the 63.3

4.2.1 Misclassification Analysis

We examined misclassified samples to understand model limitations. Of 309 errors:

- **False Positives (206):** Bad wines predicted as good
- **False Negatives (103):** Good wines predicted as bad

Sample misclassified examples: *Example 1 - False Positive (Bad predicted as Good):*

- Actual: Bad (quality ¡ 6)
- Features: High volatile acidity (0.558), high pH (1.270), high sulphates (1.230), red wine
- **Issue:** High sulphates and certain acid levels may correlate with good wines, but high volatile acidity (vinegar taste) makes it bad

Example 2 - False Positive (Bad predicted as Good):

- Actual: Bad (quality ¡ 6)
- Features: Very high residual sugar (3.596), high chlorides (1.831), high alcohol (1.931), white wine
- **Issue:** Model likely associates high alcohol with quality, but excessive sugar and salt create imbalance

Example 3 - False Negative (Good predicted as Bad):

- Actual: Good (quality ≥ 6)
- Features: High volatile acidity (0.925), low citric acid (-0.965), low alcohol (-1.079), red wine
- **Issue:** Low alcohol strongly associated with bad wines, but other factors compensate in reality

Patterns identified: *False Positives characteristics:*

1. **Conflicting features:** Wines with some good indicators (e.g., high alcohol, proper pH) but critical flaws (e.g., high volatile acidity, excessive chlorides)
2. **Imbalanced chemistry:** High sugar with high salt, or high alcohol with poor acidity balance
3. **Single strong features:** Model overweights individual positive features, ignoring negative combinations

False Negatives characteristics:

1. **Low alcohol bias:** Wines with alcohol ¡ 9.5
2. **Threshold proximity:** Many are borderline quality=6 wines where small errors matter
3. **Atypical profiles:** Good wines with unusual feature combinations not well-represented in training data

Model limitations revealed:

1. **Feature interaction blindness:** Linear (even kernelized) models struggle with complex feature interactions. A wine might have high alcohol (good) and high volatile acidity (bad), but the model doesn't capture that this specific combination is unacceptable.
2. **Threshold artifacts:** Binary classification at quality=6 creates a sharp boundary where wines rated 5.5 and 6.5 (if continuous) would be very similar but are forced into different classes.
3. **Majority class bias:** With 63.3% good wines, the model defaults to "good" when uncertain, explaining the 2:1 FP:FN ratio.
4. **Rare profile handling:** Wines with unusual but valid feature combinations (e.g., low alcohol but excellent acid balance) are often misclassified due to insufficient training examples.
5. **Missing domain knowledge:** The model doesn't understand that certain feature combinations are chemically incompatible with quality, treating all features independently.

These limitations suggest that future improvements should focus on: (1) explicit feature interaction terms; (2) ensemble methods to capture different decision patterns; (3) class-weighted loss to reduce majority bias; (4) multi-class formulation to preserve gradual quality transitions.

4.3 Training Curves

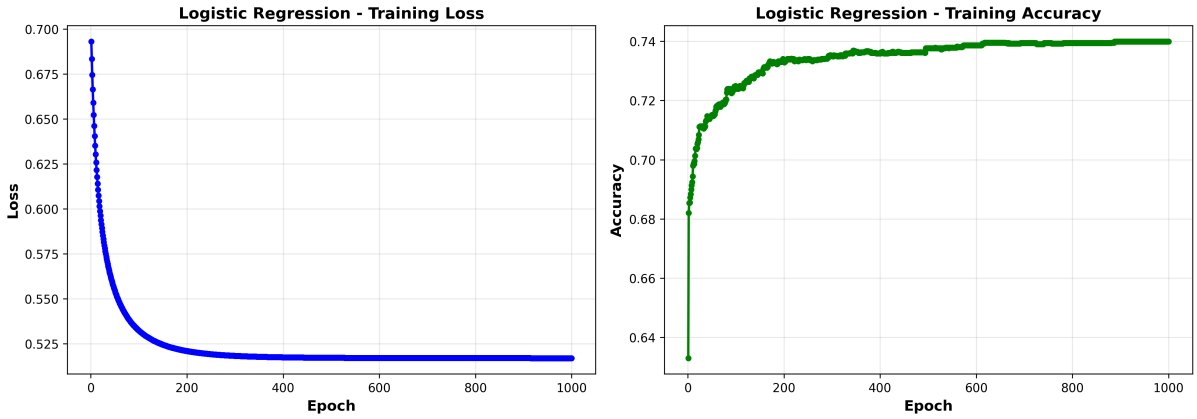

Figure 7: LR training curves showing smooth convergence without overfitting.

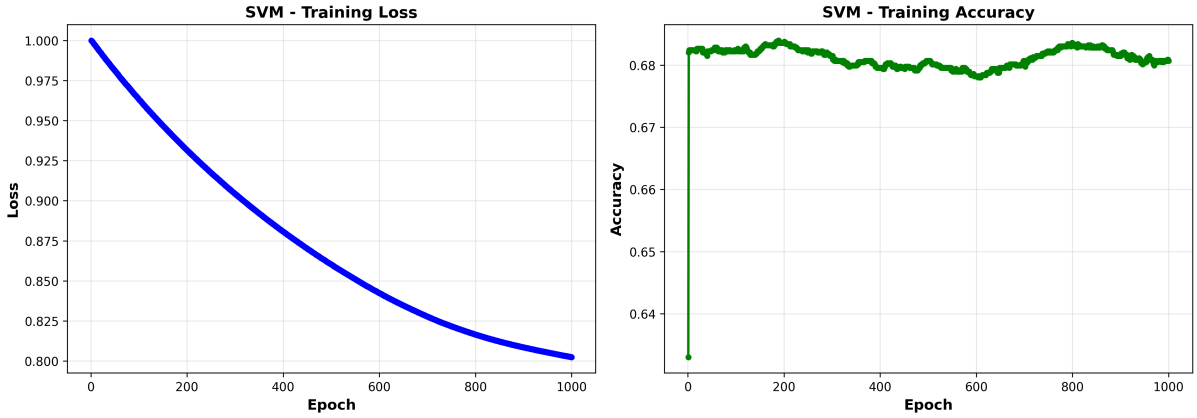

Figure 8: SVM training curves showing slower convergence with more oscillation.

LR converged rapidly (loss 0.70 $\rightarrow\rightarrow$ ß0.52$by epoch 300) with stable accuracy plateau at 74\%. SVM converged more$
ß0.80)$with greater oscillation, plateauing at 68\%. Both models showed no overfitting signs.$

5 Discussion

5.1 Why Logistic Regression Outperformed SVM

LR's superior performance (75.65**Theoretical:** (1) LR's smooth cross-entropy loss enables more stable gradient descent than SVM's sharp hinge loss; (2) LR provides calibrated probabilities suitable for overlapping class distributions, while SVM's hard margins may be too rigid; (3) LR's uniform weight penalization differs from SVM's support-vector-focused approach. **Practical:** LR converged 3-5$\times$ faster (0.16s vs 4.91s) and showed less sensitivity to hyperparameters. SVM required careful learning rate tuning and exhibited training instability.

5.2 Kernel Analysis

RBF success: The 82.31**Polynomial failure:** Degree 2 insufficient for complex interactions (63.87**Gamma selection:** Lower $=0.1\gamma = 0.1 = 0.1(broader influence, smoother boundaries) generalized bette$
$0.5\gamma = 0.5 = 0.5(tighter influence, overfitting risk)$.

5.3 Model Complexity Assessment

No overfitting: Test accuracy (76.19**No underfitting:** 74The ∼76

6 Conclusion

6.1 Summary

We successfully implemented SVM and Logistic Regression from scratch with kernel extensions, achieving 82.31**Key findings:**

- LR superior to SVM (6.3
- RBF kernel most effective for non-linearity
- Lower gamma optimal for generalization
- Polynomial kernels underperformed

6.2 Limitations and Future Work

Limitations: (1) Kernel SVM computationally expensive (34-145s training); (2) class imbalance addressed only via stratification; (3) binary classification discards fine-grained quality information; (4) limited hyperparameter search space. **Future directions:**

- Class-weighted loss functions or SMOTE for imbalance
- Multi-class classification preserving full quality scale
- Feature engineering (domain-specific interactions, PCA)
- Ensemble methods and deep learning
- Bayesian hyperparameter optimization