SVM & Logistic Regression for Wine-Quality Prediction

Machine Learning project report

Silvia Guidi - 50761A

Introduction

The present study investigates the application of both linear and kernel-based classification algorithms to predict wine quality. The dataset consists of 6,497 observations with 12 numerical features representing various chemical properties of wine. The primary objective is to classify wines as "good" or "bad" based on a binary transformation of the original quality scores. The analysis follows a structured workflow: first, an exploratory and preprocessing phase examines feature distributions, correlations, and potential issues such as skewness and outliers. Subsequently, models are implemented from scratch, including Support Vector Machines (SVM) and Logistic Regression (LR), both in linear and kernelized forms. Training procedures, regularization strategies, and hyperparameter tuning are thoroughly discussed, with cross-validation employed to evaluate generalization performance. Special attention is given to kernel methods, where Gaussian and polynomial kernels enable nonlinear decision boundaries. Computational optimizations are applied to reduce runtime while preserving theoretical correctness. Model performance is assessed using accuracy, precision, recall, F1-score, training dynamics, and misclassification analysis, providing a comprehensive understanding of model behavior under different configurations.

Data Exploration and Preprocessing

The dataset comprises of 6497 observations and 12 numerical variables. Each variable measures a different component of wine: fixed and volatile acidity, citric acid, residual sugar, chlorides, both free and total sulfur dioxide, density, the ph, sulphates, and alcohol. All these represent the features that will be used to classify each observation, while the target variable used will be 'quality'.

A first exploration of the dataset pictured in figure 1 shows that most variables display skewed distributions, often concentrated toward lower values with long right tails. For instance, residual sugar and sulfur dioxide have several extreme values, suggesting the presence of outliers. Variables such as pH and alcohol are more symmetrically distributed and centered, which makes them more comparable across samples. The quality variable itself is not uniformly spread: most wines are rated around the mid-range, with relatively few cases of very low or very high quality.

Figure 2 gives another perspective on the distributions considered. The boxplots highlight the same tendency: several features exhibit outliers extending far beyond the interquartile range, confirming the asymmetry observed in the histograms. Nevertheless, these points were retained rather than removed, since they represent real chemical variations that could be informative for classification.

Finally, the correlation between the variables is investigated. The heath map in figure 3 shows that most variables are mildly correlated. There are two relations that stand out for their high value: 0.72 for free and total sulfur dioxide and -0.69 for density and alcohol. These two results can be justified using qualitative reasoning: in the first case the variables share the same object measured - sulfur dioxide; in the second case, chemically, ethanol (the element of alcohol) has a lower density compared to the water components of the wine, thus the negative relation. Focusing on the target, all the relations are negative and not strong, with the exception of 'alcohol' which has positive sign and medium strength.

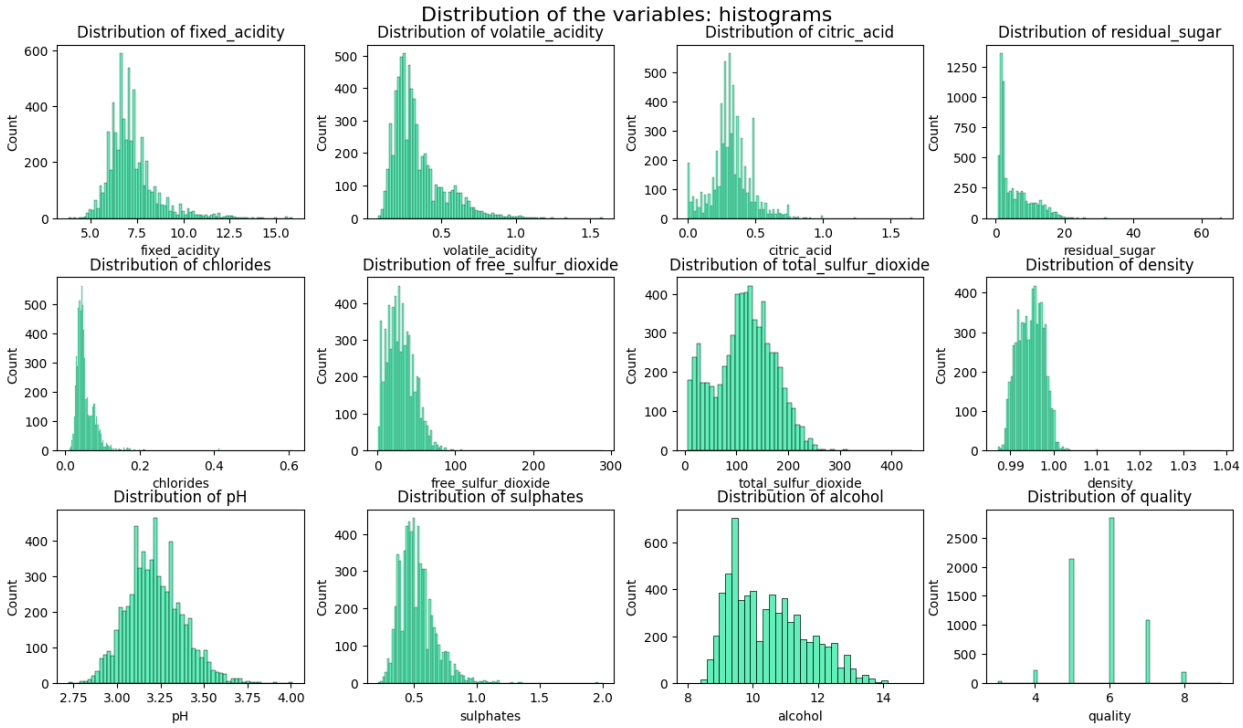

Figure 1: Distribution of variables represented using histograms

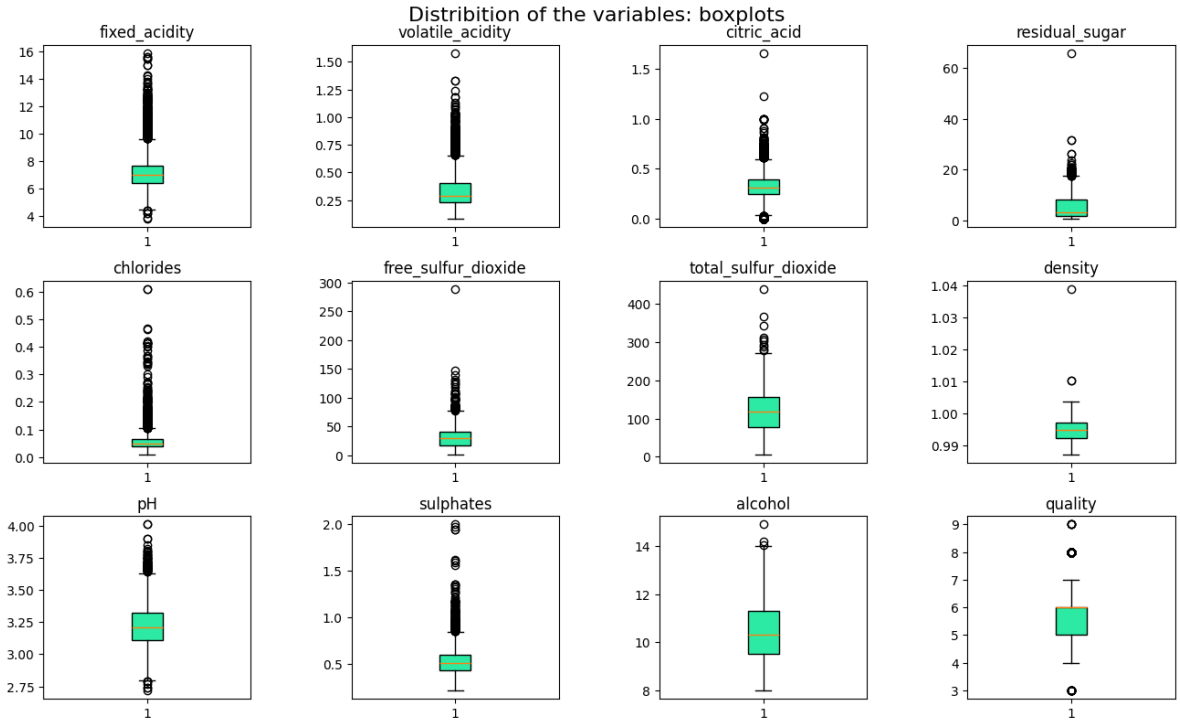

Figure 2: Distribution of variables represented using boxplots

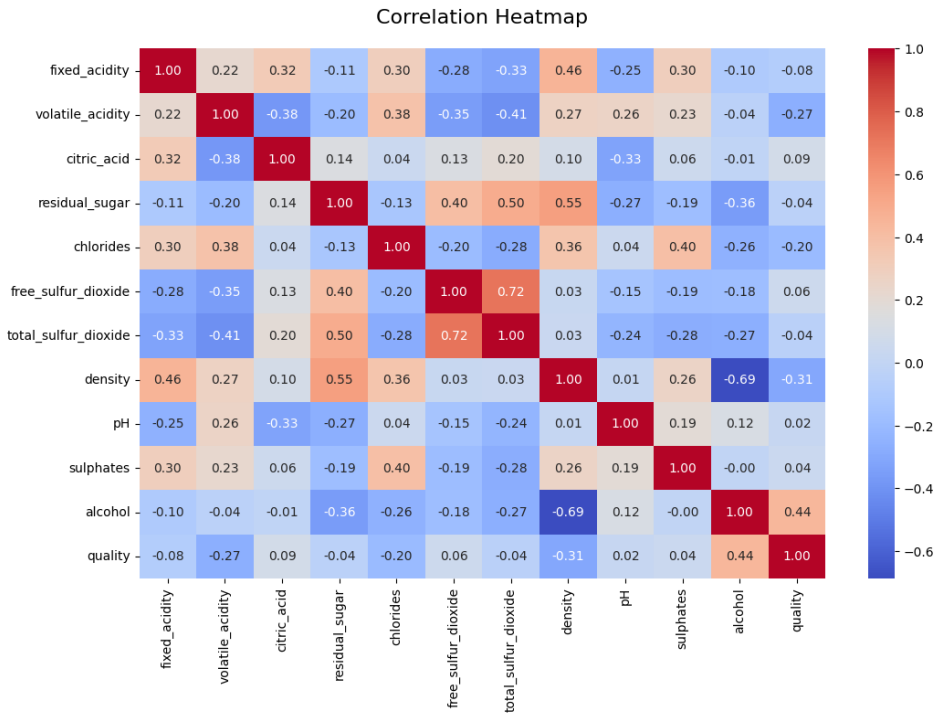

Figure 3: Heat map of the correlations between variables

After this first investigation of the data as they are provided, some manipulation is performed. First, the target variable 'quality' had to be encoded into a binary variable, necessary both for the Support Vector Machine and the Logistic Regression model. The target variable measures the quality of the wine on a scale from 0 to 10, therefore the threshold chosen to consider a wine 'good'(encoded as 1), or 'bad'(encoded as -1) is 6. The variable needed no further adjustments since the distribution between classes is fairly balanced, with slightly more 'good' wines (63.3%) than 'bad' ones (36.7%).

Afterwards, the training and test sets are created. The choice of the size of each set is made in accordance to the size of the data set and the common practices in machine learning to ensure train adequacy and test reliability. As result, 80% of the data were randomly chosen as training and the remaining 20% - approximately 1300 observations - as test. During this split, in addition to randomization, stratification is implemented to ensure that a constant proportion in the classes of the target would be preserved in the two groups. Both the training and the test are standardized to have a common scale using the following formula:

$$\mathrm{X} = \frac{X - \mu_{\text{train}}}{\sigma_{\text{train}}}$$

where μ and σ are respectively the mean and the standard deviation of the train, chosen to avoid problems of data leakage. Figure 4 shows the results of these changes.

Now, the mean of each variable has a value close to zero and standard deviation equals to 1. The range of the variables is now unique and optimal for the models. Distributions of residual_sugar, free_sulfur_dioxide, total_sulfur_dioxide, chlorides remain skewed. Given the quantity of outliers, excluding these points would not be optimal. In this cases it has given priority to preserve the dataset as it is, therefore neither skewness nor outliers have been adjusted.

In summary, the dataset was explored, preprocessed, and prepared for modeling. The main challenges observed are skewed distributions and outliers, but no severe multicollinearity was detected. The binary encoding of quality ensures compatibility with classification algorithms, and standardization guarantees all features are on a comparable scale.

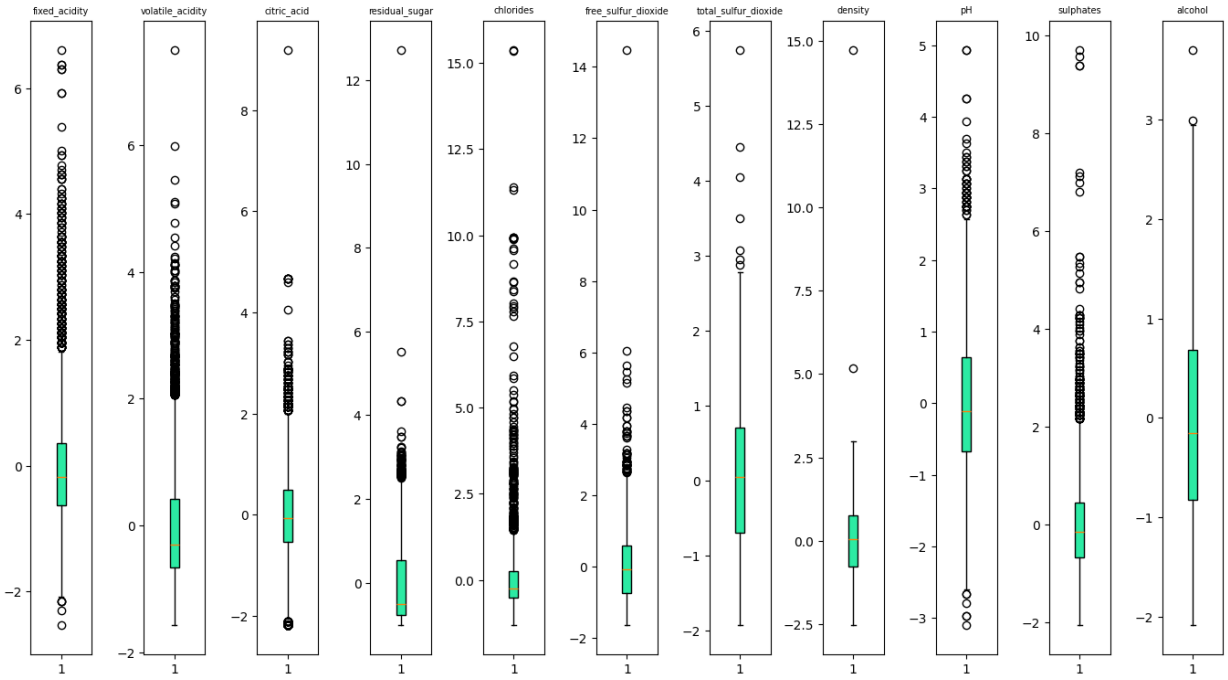

Figure 4: Distribution of standardized variables of the training group represented using boxplots

The models

In this section, each model is presented together with the rationale behind its design and implementation. Starting from the linear SVM, then moving to logistic regression, and finally extending both models to their kernelized variants. All implementations are written following an object-oriented approach, providing a consistent interface ("fit", "predict", "score", and "decision_function") that facilitates evaluation, comparison, and extension to more complex methods.

Support Vector Machine

The Support Vector Machine is trained by minimizing the regularized hinge loss:

$$\min_{w,b} \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i(w^\top x_i + b)\}$$

This formulation does not assume perfect linear separability. Instead, it explicitly allows margin violations by penalizing misclassified or weakly classified points through the hinge loss, leading to a solution that is more robust to noise and outliers while still promoting a large margin when possible. To solve this optimization problem, a Pegasos-style stochastic subgradient descent is adopted. The weight vector w and bias b are initialized to zero and updated in minibatches of size m. At each update step t, the learning rate is set to $\eta_t = \frac{1}{\lambda \cdot t}$. Within a minibatch, the algorithm identifies the set of violators $V_t = \{i : y_i(w^\top x_i + b) < 1\}$ that fail to meet the margin condition. The update proceeds in two parts:

1) Shrinkage step, which accounts for regularization and discourages overly large weights:

$$w \leftarrow (1 - \eta_t \lambda) w$$

2) Correction step (in case of violations):

$$w \leftarrow w + \frac{\eta_t}{|V_t|} \sum_{i \in V} y_i x_i, \quad b \leftarrow b + \frac{\eta_t}{|V_t|} \sum_{i \in V} y_i$$

Thus, the model shrinks the parameters at every step to maintain regularization, while violators push the hyperplane in their favor. The training history records the hinge loss plus regularization at the end of each epoch, providing a diagnostic of convergence. Predictions are made using the decision function $f(x) = w^\top x + b$, with class labels obtained as $\hat{y} = \text{sgn}(f(x))$.

Logistic Regression

Logistic regression is formulated as minimizing the regularized log-loss

$$\min_{w,b} \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-y_i(w^\top x_i + b)})$$

Unlike the hinge loss, the logistic loss arises from a probabilistic model in which labels follow a Bernoulli distribution with success probability given by the sigmoid $\sigma(z) = 1/(1+e^{-z})$. This allows logistic regression to provide calibrated confidence scores in addition to classification. The stochastic gradient for a minibatch is computed as:

$$\nabla_w = -\frac{1}{m}\sum_{i=1}^{m} y_i x_i \sigma(-y_i f(x_i)) + \lambda w, \quad \nabla_b = -\frac{1}{m}\sum_{i=1}^{m} y_i \sigma(-y_i f(x_i))$$

where $f(x) = w^\top x + b$. The weights and bias are then updated with learning rate η: $w \leftarrow w - \eta\nabla_w$ and $b \leftarrow b - \eta\nabla_b$. The training process tracks the logistic regularization at the end of each epoch. Predictions are obtained by thresholding the decision function:

$$\hat{y} = \begin{cases} 1 & \text{if } f(x) \geq 0, \\ -1 & \text{otherwise} \end{cases}$$

Thus, logistic regression closely mirrors the SVM procedure, but leverages a probabilistic fundation and outputs interpretable confidence scores through the sigmoid function.

Kernel methods

Linear classifiers can be extended to nonlinear decision boundaries using kernel functions. Instead of working directly in the input space, the kernel method implicitly maps inputs into a high-dimensional feature space where a linear separation may exist. Two different tricks have been used: the Gaussian ($k(x.x') = \exp(-\gamma\|x - x'\|^2)$) and the polynomial ($k(x.x') = (\gamma x^\top x' + c_0)^d$) both of which enable flexible decision boundaries while keeping computations in the original space. The solution in the kernel space can be expressed as

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) + b$$

where α_i are the coefficients assigned to each training point. Predictions are obtained by applying the sign function for SVM, or the sigmoind probability model for logistic regression.

Kernel SVM

The kernelized SVM minimizes the regularized hinge loss

$$\min_{w,b} \frac{\lambda}{2}\alpha^\top K\alpha + \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i(f(x_i))\}$$

where K is the kernel matrix with entries $K_{ij} = k(x_i, x_j)$ and $f(x_i) = \sum_j \alpha_j y_j K_{ij}$. The optimization follows a Pegasos-style subgradient descend in the dual: at each iteration t with step size $\eta_t = \frac{1}{\lambda t}$, a minibatch is sampled and the set of margin violators is identified as $V_t = \{i \in B_t : y_i f(x_i) < 1\}$. The update rule for violators is

$$\alpha_i \leftarrow \alpha_i + \frac{\eta_t}{|V_t|}, \quad i \in V_t$$

followed by a projection step to enforce the dual box constraints. At the end of each epoch, the hinge loss plus regularization is tracked to monitor convergence.

Kernel Logistic Regression

In kernel space, logistic regression minimizes the regularized log-loss

$$\min_{\alpha,b} \frac{\lambda}{2}\alpha^\top K\alpha - \frac{1}{n}\sum_{i=1}^{n}[y_i \log \sigma(f(x_i)) + (1 - y_i)\log(1 - \sigma(f(x_i)))]$$

where $\sigma(z) = 1/(1 + e^{-z})$ is the sigmoid and $f(x) = \sum_j \alpha_j K_{ij} + b$. Here labels are mapped to $0, 1$ before computing the loss.

The gradient updates at iteration t are:

$$\nabla_\alpha = \frac{1}{n}K(\sigma(f) - y) + \lambda K\alpha, \qquad \nabla_b \frac{1}{n}\sum_{i=1}^{n}(\sigma(f(x_i)) - y_i)$$

The parameters are updated with learning rate η:

$$\alpha \leftarrow \alpha - \eta\nabla_\alpha, \qquad b \leftarrow b - \eta\nabla_b$$

At each epoch, the log-loss plus regularization is tracked, together with training and validation accuracy.

Both kernelized models preserve the same object-oriented interface (fit, predict, score, decision_function), ensuring a seamless comparison between linear and nonlinear variants. In practice, kernel methods enable flexible, nonlinear decision boundaries while maintaining the theoretical structure of their linear counterparts.

Code Optimization for Kernel Models

To reduce the runtime of kernel-based models, several strategies were applied. First, batch updates were implemented in the gradient-based training loops, allowing computations on subsets of the data rather than the full kernel matrix at each step. Second, precomputation of kernel matrices for all candidate hyperparameters significantly reduced redundant computations during cross-validation. Third, parallelization of cross-validation folds using joblib enabled simultaneous evaluation across CPU cores, effectively reducing total CV time. Finally, early stopping based on validation loss was introduced, terminating training once improvements plateaued and preventing unnecessary epochs. Importantly, these optimizations were applied without altering the theoretical foundation of the models or the cross-validation procedure, ensuring that results remain theoretically valid.

Hyperparameter tuning

The performance of both linear and kernelized models depends critically on the choice of hyperparameters. The k-fold cross-validation technique is employed. The training data is split into k folds of equal size; at each iteration, one fold is used as validation set while the model is trained on the remaining $k-1$ folds. The mean and standard deviation of the validation scores across all folds are recorded, providing both an estimate of generalization performance and its variability. The hyperparameter combination yielding the highest mean validation accuracy is selected.

For linear models, the main hyperparameter is the regularization strength λ, which controls the trade-off between minimizing the loss and penalizing parameter values. A logarithmic grid is explored: $\lambda \in \{10^{-4}, 10^{-3.33}, 10^{-2.66}, 10^{-2}, 10^{-1.33}, 10^{-1.66}, 10^{0}\}$.

For kernel methods, in addition to λ, kernel-specific parameters must be tuned. With the Gaussian kernel, the scale parameter γ determines the effective radius of influence of each data point. The grid explored is: $\gamma \in \{10^{-3}, 10^{-2.2}, 10^{-1.4}, 10^{-0.6}, 10^{-0.2}, 10^{1}\}$. With the polynomial kernel, both γ and the polynomial degree d are tuned, as the polynomial order strongly influences the complexity of the induced feature space. The degree is selected from a discrete grid: $d \in \{2, 3, 4\}$, while γ follows the same logarithmic grid as before. The offset parameter c_0 is kept fixed.

This tuning procedure in implemented in the cross_val_score and cross_val_score_kernel functions, which iterate over all combinations of candidate hyperparameters, evaluate their performance via 5-fold cross validation, and return the configuration achieving the highest validation score.

Results

Prior to the detailed examination of performance metrics, confusion matrices, and learning dynamics, it is useful to highlight the general trends observed across models. Linear approaches (SVM and logistic regression) deliver stable but moderate performance, reflecting their limited representational capacity and a tendency to underfit. Gaussian kernel models exhibit markedly different behaviors: the SVM attains near-perfect accuracy on the training set but generalizes poorly, whereas logistic regression yields smoother, less complex decision boundaries, resulting in only marginal improvements over the linear baseline. Polynomial kernels produce mixed outcomes: the SVM achieves a favorable balance between flexibility and stability, while logistic regression is characterized by unstable convergence and biased predictions.

Models performance

Figure 5 reports the evaluation metrics (accuracy, precision, recall, and F1-score) for all models, comparing training and test results. The linear models, namely SVM and logistic regression, perform very similarly, displaying consistent outcomes across both datasets. In contrast, kernel-based approaches produce more heterogeneous results. The Gaussian kernel SVM, for instance, reaches almost perfect accuracy on the training set but suffers a sharp decline on the test set, particularly in accuracy and F1-score, pointing to its difficulty in maintaining generalization. The Gaussian kernel logistic regression behaves differently: despite relying on the same kernel transformation, it performs much like the linear models, suggesting that the probabilistic formulation constrains its flexibility and limits overfitting. Polynomial kernels, on the other hand, lead to less uniform patterns. The polynomial SVM delivers stable and reasonable performance across training and test, whereas the polynomial logistic regression exhibits highly unbalanced behavior, with low accuracy and precision but perfect recall. This reflects a strong bias toward predicting the majority class as positive, which maximizes recall at the expense of a large number of false positives.

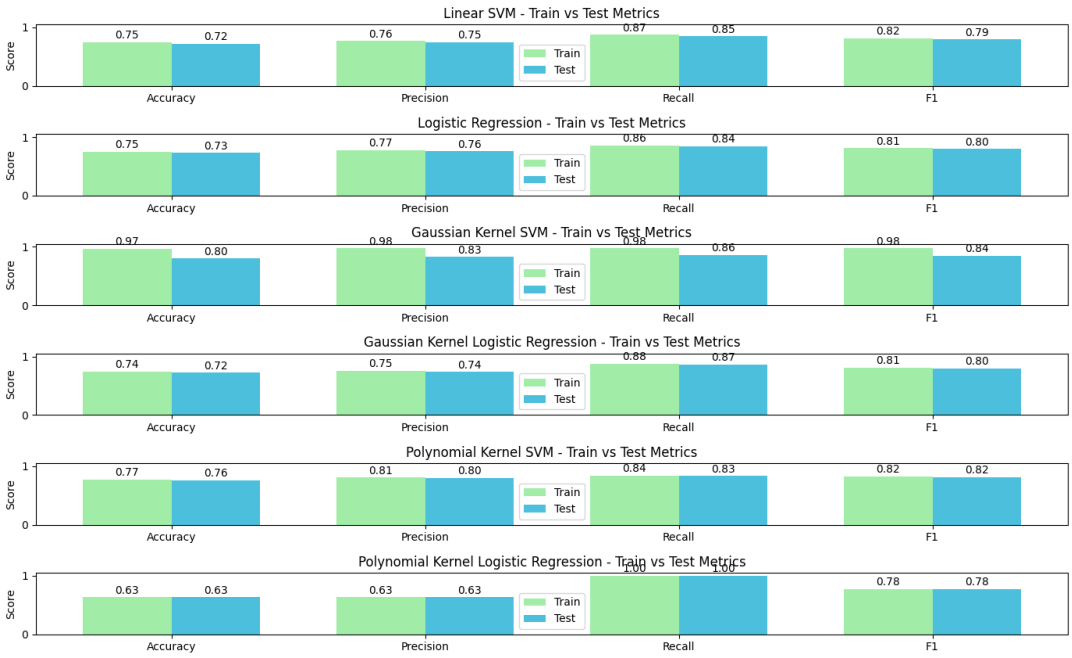

Figure 5: Representation and comaprison of resulting train and test metrics of linear and kernel svm and logistic regression

The confusion matrices in Figure 6 provide additional insight into how the different models handle class imbalance. For the linear SVM and logistic regression, the same tendency emerges: both are more successful at identifying positive cases than negatives. This asymmetry can be partly explained by the distribution of the target variable, where positives are more frequent. Since linear models construct a single separating hyperplane, they naturally favor the majority class, leading to reliable but limited discrimination power. The Gaussian kernel SVM departs from this pattern: by projecting the data into a richer feature space, it is able to almost perfectly separate the classes in the training set. However, this expressiveness is not fully beneficial, as the improvement in training accuracy does not translate to the test set. By contrast, the Gaussian kernel logistic regression exhibits a more conservative profile. While still relying on nonlinear mappings, the probabilistic formulation acts as a regularizer, producing smoother decision boundaries that achieve more balanced, though overall weaker, results. Among polynomial models, the SVM again represents an intermediate case, maintaining stability across training and test and providing a fair recognition of both classes. The polynomial logistic regression, however, collapses into a trivial solution: it predicts nearly all instances as positive, a consequence of the imbalance combined with the high-dimensional feature expansion. The result is perfect recall but almost no ability to detect negative cases.

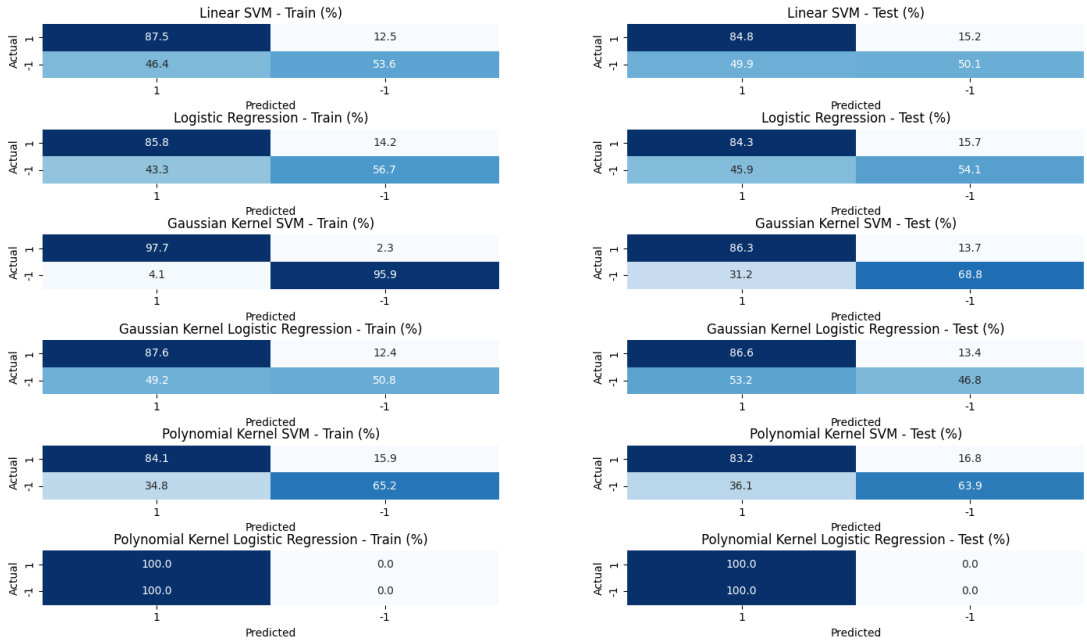

Figure 6: Confusion matrices of the four models with percentages per row

Training dynamics

Figure 7 illustrates the evolution of training and validation loss and accuracy for the linear models. Both SVM and logistic regression show similar behavior, with training and validation curves remaining closely aligned across epochs. This confirms that these models generalize well and do not overfit, consistent with the small gap observed in the evaluation metrics. At the same time, moderate metric values and limited improvement in accuracy suggest some degree of underfitting, indicating that while these models are stable, they do not fully capture data complexity.

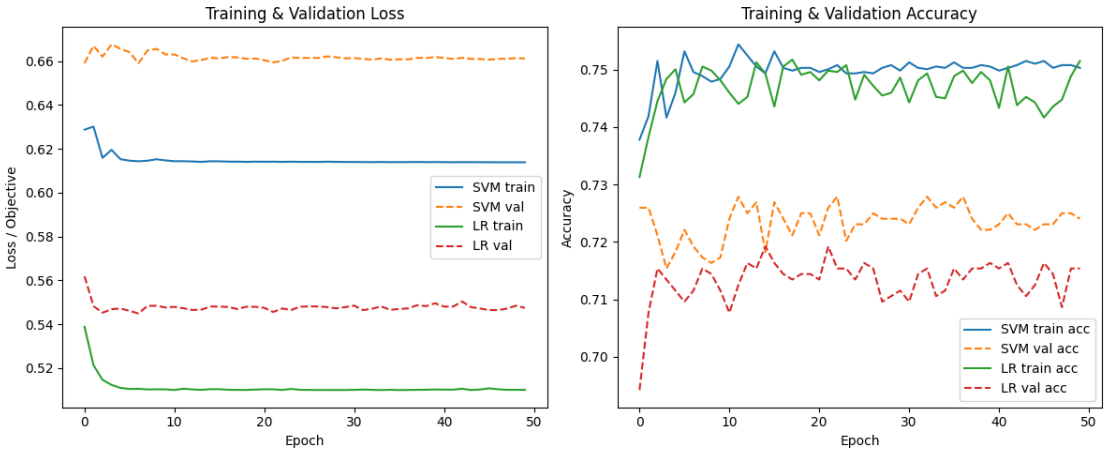

Figure 7: Comparison of training and validation loss (left) and accuracy (right) for linear SVM and Logistic Regression over 50 epochs.

Training dynamics for the Gaussian kernel models (Figure 8) reveal a stark contrast. The Gaussian kernel SVM quickly reaches high training accuracy, exceeding 0.95 within the first few epochs, while validation

accuracy plateaus around 0.8. The gap between training and validation loss indicates that the model fits the training data extremely well but struggles to generalize, consistent with overfitting. Conversely, the Gaussian kernel logistic regression exhibits smoother learning: training and validation losses decrease steadily, remaining close throughout, with accuracy rising to a moderate plateau between 0.7 and 0.75. This pattern indicates reasonable generalization without overfitting, though moderate accuracy reflects some underfitting.

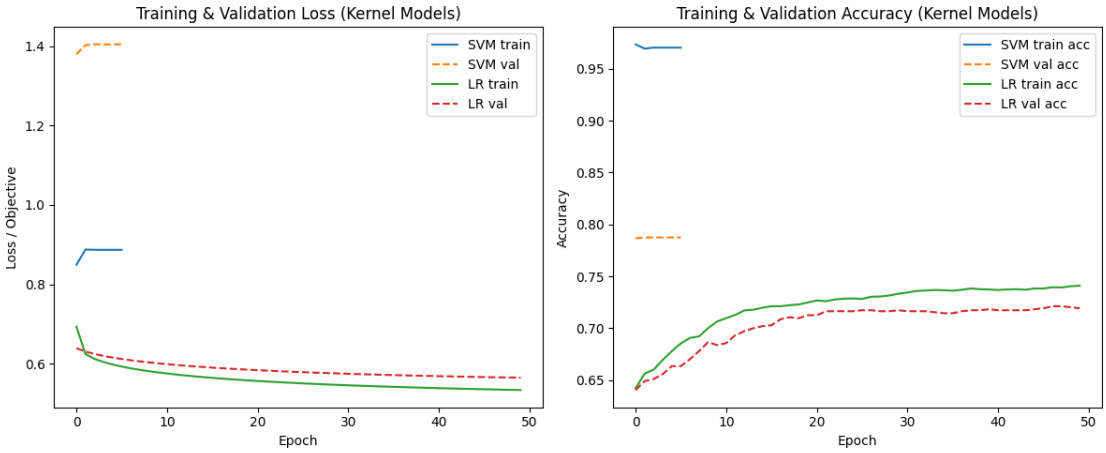

Figure 8: Comparison of training and validation loss (left) and accuracy (right) for Gaussian Kernel SVM and Logistic Regression over 50 epochs.

Polynomial kernel models display distinct behaviors for SVM and logistic regression (Figure 9). The polynomial SVM learns steadily, with both losses and accuracies stabilizing across training and validation, suggesting good generalization and reliable performance. The polynomial logistic regression, however, exhibits erratic fluctuations in loss and accuracy, indicating difficulty in convergence. The oscillatory patterns highlight the model's limited capacity to exploit the kernel's expressive power and suggest underfitting.

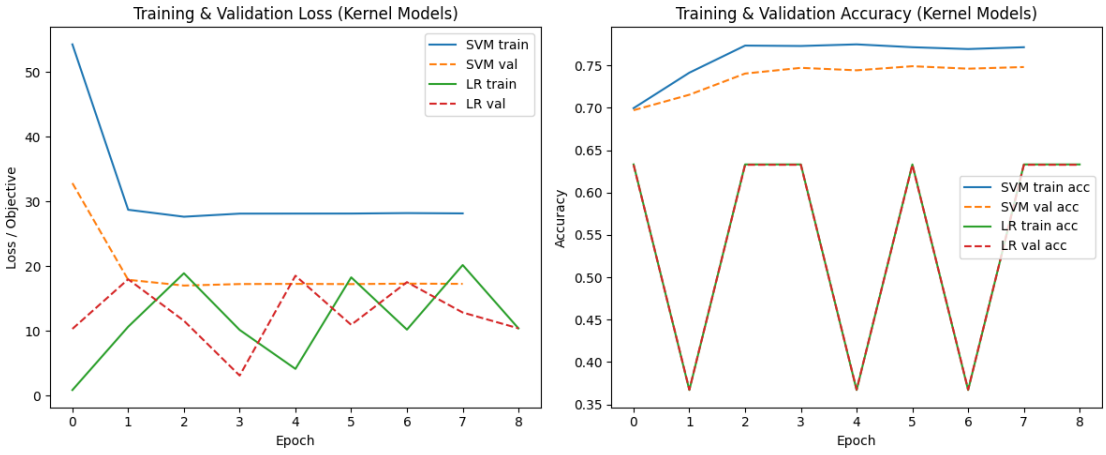

Figure 9: Comparison of training and validation loss (left) and accuracy (right) for Polynomial Kernel SVM and Logistic Regression over 8 epochs.

Cross-Validation and Hyperparameter Analysis

Cross-validation reveals important differences in how the models respond to hyperparameter tuning. For the linear models (Figure 10), the SVM achieves its highest validation accuracy at a regularization parameter of $\lambda = 10^{-2}$. Accuracy decreases for both stronger and weaker regularization, forming a characteristic bell-shaped curve. This pattern reflects the role of λ: higher regularization overly constrains the margin, reducing the model's ability to fit the data, while too little regularization allows excessive flexibility, increasing variance and risk of overfitting. Logistic regression, by contrast, achieves its best performance at the minimal tested regularization $\lambda = 10^{-4}$, with validation accuracy remaining relatively stable for small λ and declining only under stronger regularization. This indicates that logistic regression benefits from almost unregularized fitting, allowing it to capture patterns in the data while remaining robust to overfitting due to its linear, probabilistic formulation.

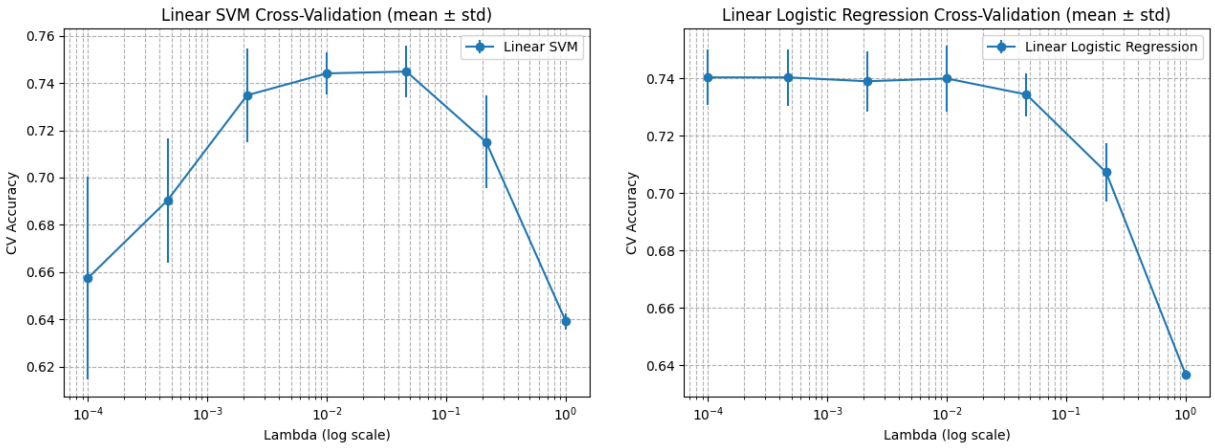

Figure 10: Cross-validation curves for the linear models: SVM (left), logistic regression (right)

Gaussian kernel models display more pronounced differences (Figure 11). For the Gaussian SVM, the optimal parameters are $\gamma = 1.5848$ and $\lambda = 10^{-4}$, which yield a peak mean accuracy of 0.788. Other combinations of γ and λ produce lower accuracy, confirming that this pair represents the best trade-off between flexibility and regularization. The interpretation is consistent with the meaning of the parameters: larger γ increases the influence of individual training points, which can lead to overfitting if not controlled by sufficient regularization. Indeed, higher λ would reduce training accuracy by penalizing large weights and smoothing the decision boundary, while lower λ allows near-perfect training fit but worsens test performance. Gaussian kernel logistic regression reaches its best mean accuracy of 0.74 at $\gamma = 0.23$ and $\lambda = 10^{-4}$. Across other parameter combinations, accuracy is consistently lower, showing that the model is less sensitive to the kernel's nonlinearity and relies on moderate γ to balance flexibility and stability.

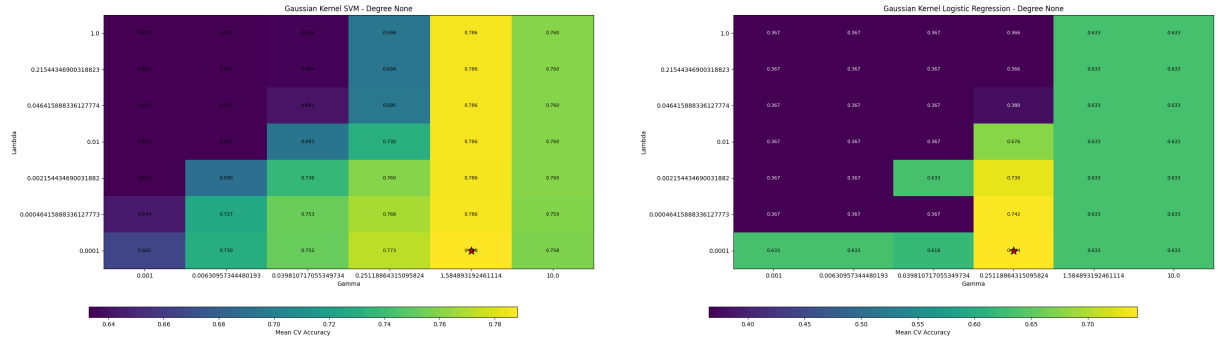

Figure 11: Cross-validation mean accuracy for Gaussian kernel SVM and logistic regression across different gammas and lambda values.

Polynomial kernels further illustrate these contrasting behaviors (Figure 12). The polynomial SVM achieves maximum performance at degree $d = 5$ with $\gamma = 0.0398$ and $\lambda = 0.0001$, yielding a mean accuracy of 0.756, and similarly at degree 5 with slightly higher $\lambda = 0.0004$ (0.753). The graphs show that moderate degree, intermediate γ, and light regularization allow the SVM to capture nonlinear patterns effectively, while increasing the degree or γ too much generally reduces accuracy, as the model becomes overly flexible and sensitive to individual training points. On the other hand, polynomial logistic regression reaches its best performance only at very low $\gamma = 0.006$ and $\lambda = 0.0001$, with a mean accuracy of 0.633, and performs worse for higher degrees. The increase in degree does not benefit logistic regression because, unlike SVM, it cannot exploit the additional complexity; higher degree amplifies instability in class probability estimation, especially when combined with stronger γ or low regularization. Across all other parameter combinations, logistic regression remains underfitted, demonstrating that it is sensitive to degree, γ, and λ, but cannot leverage the polynomial kernel as effectively as the SVM.

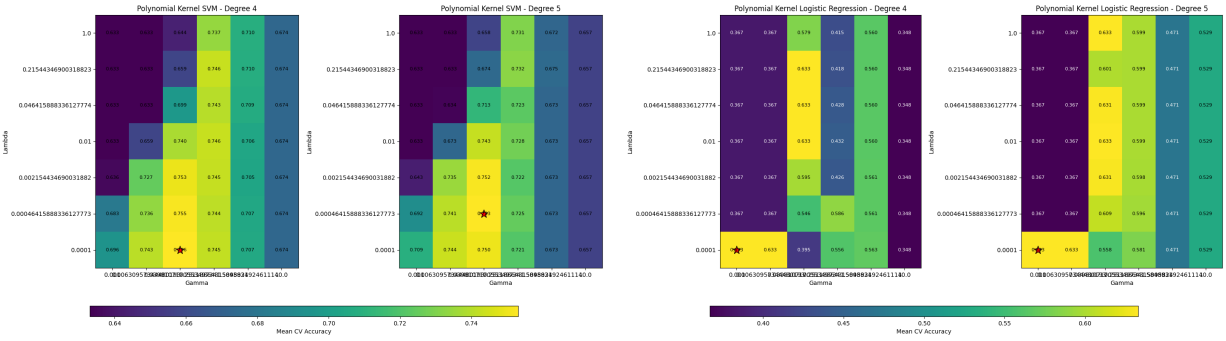

Figure 12: Cross-validation mean accuracy for Polynomial kernel SVM and logistic regression across different gammas and lambda values.

Misclassified Example Analysis

The analysis of misclassified observations reveals systematic differences among models (Figure 13). Most approaches, including linear SVM and logistic regression, Gaussian kernel logistic regression, and polynomial kernel variants, misclassify wines with low alcohol content and high density, while features such as pH, citric acid, and residual sugar remain close to average. This pattern suggests that errors occur primarily in borderline cases where chemical properties provide weaker signals, and the similarity of misclassification profiles between training and test sets indicates structural rather than random errors.

The Gaussian kernel SVM, in contrast, exhibits markedly different patterns. Multiple features deviate strongly from the average, and misclassification profiles differ substantially between training and test sets. These discrepancies reflect overfitting: the model memorizes irregularities in the training data, producing sharply defined misclassification profiles for training but broader and shifted profiles in test data. Overall, while most models show stable and interpretable error patterns, the Gaussian kernel SVM is highly sensitive to specific feature combinations and demonstrates poor generalization.

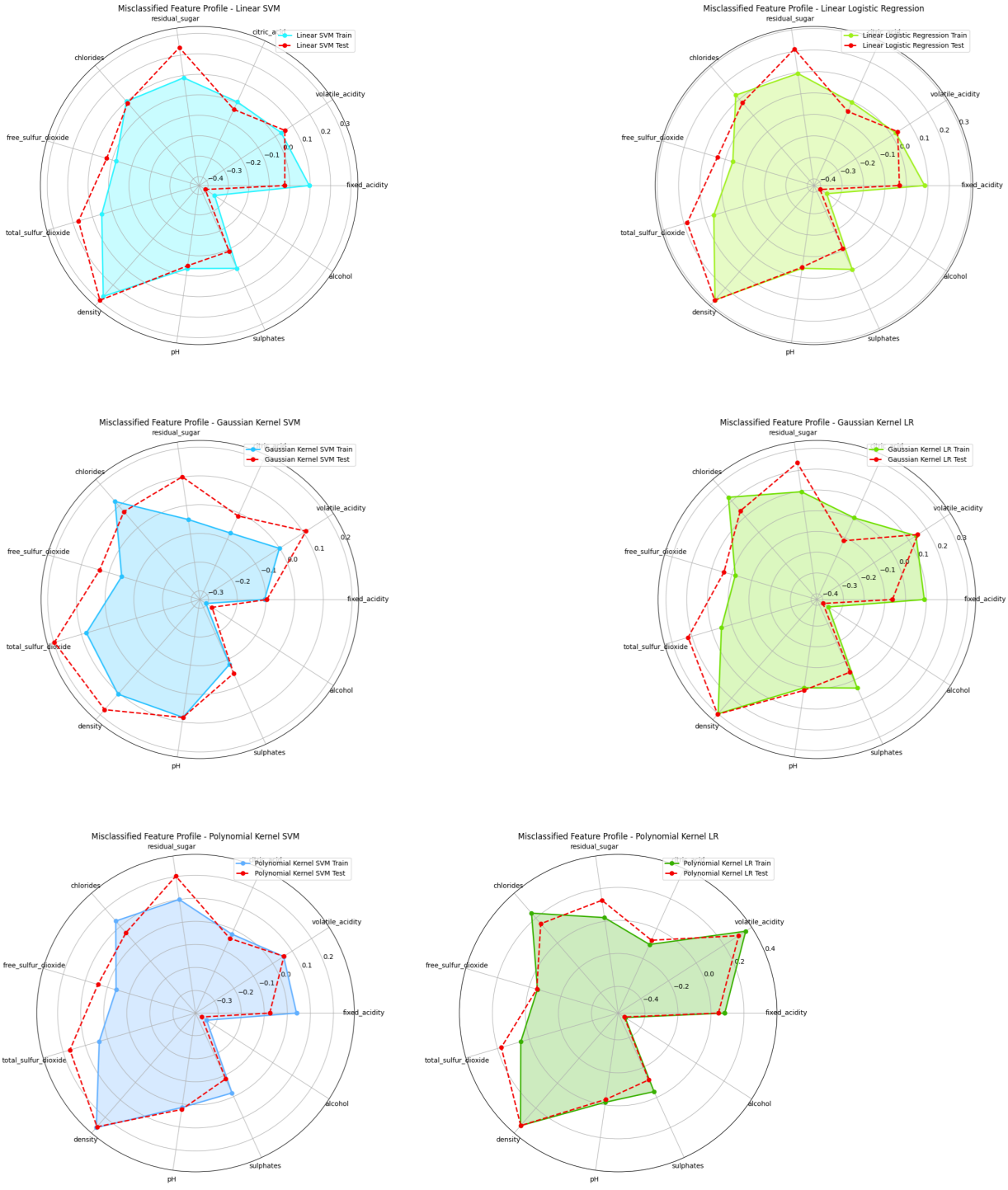

Figure 13: Radar plots of the average profiles of misclassified observations for training and test sets across all models

Generalization Analysis

Across models, generalization behavior reflects the interplay between model flexibility, regularization, and data complexity. Linear models (SVM and logistic regression) maintain stable but moderate metrics (Figure 5) across training and test sets, with a small gap between them. Their validation curves (Figure 7) show closely aligned training and validation loss and accuracy, indicating good generalization but limited capacity to capture nonlinear patterns. Regularization parameters for both models are low-to-moderate (λ selected via cross-validation, Figure 10), helping prevent overfitting, though the resulting performance reflects mild underfitting, particularly for the minority class as seen in the confusion matrices (Figure 6).

Gaussian kernel models reveal a stronger contrast. The SVM attains near-perfect training metrics (Figure 5) but drops substantially on test data, particularly for minority-class accuracy, highlighting overfitting. Cross-validation identifies optimal parameters at moderate γ and very low λ (Figure 11), which maximize training fit but amplify sensitivity to training noise. Training curves (Figure 8) plateau quickly, with low training loss but higher validation loss, confirming memorization of training patterns. In contrast, Gaussian kernel logistic regression achieves moderate metrics comparable to linear models, with training and validation curves closely aligned and limited sensitivity to γ and λ, reflecting stable learning with only mild underfitting.

Polynomial kernel models further illustrate the importance of parameter tuning. The SVM performs best at degree 4–5 with moderate γ and light λ (Figure 12), showing balanced training and test metrics (Figure 5) and smooth, stable convergence (Figure 9). Increasing degree or γ generally reduces test accuracy, as the model becomes overly flexible and sensitive to individual points. Logistic regression with polynomial kernels achieves highest metrics only at very low degree, γ, and λ, with most other parameter combinations producing underfitting, erratic learning curves, and poor class balance (Figures 5 and 9).

The analysis of misclassified observations (Figure 13) complements these findings. Linear models, Gaussian kernel logistic regression, and polynomial kernel SVM show stable and consistent misclassification patterns, concentrated near borderline cases, indicative of underfitting. The Gaussian kernel SVM, however, exhibits sharply different profiles between training and test sets, reflecting overfitting and heightened sensitivity to specific feature combinations.

Overall, these results illustrate a spectrum of generalization behaviors: linear models generalize stably but underfit, Gaussian kernel SVM overfits, Gaussian kernel logistic regression balances stability and flexibility, and polynomial kernels reveal nuanced interactions between kernel complexity, regularization, and model capacity.

Conclusion

The analysis demonstrates a clear spectrum of model behaviors. Linear SVM and logistic regression show stable performance and generalization, but tend to underfit due to their limited capacity to capture complex nonlinear patterns. Kernelized models offer greater flexibility: Gaussian kernel SVM achieves near-perfect training performance but overfits, whereas Gaussian kernel logistic regression balances learning stability and generalization without fully exploiting nonlinear features. Polynomial kernel SVM provides a reliable trade-off between flexibility and stability, while polynomial kernel logistic regression struggles with convergence and class imbalance, resulting in suboptimal predictions. Cross-validation and misclassification analyses further highlight the importance of careful hyperparameter tuning and the influence of feature distributions on model performance. Overall, the study emphasizes that model selection involves balancing expressiveness, regularization, and robustness to dataset characteristics. These findings illustrate the trade-offs inherent in linear versus kernel-based approaches and provide guidance for the application of these techniques to similar classification tasks in practice.