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 look reveals that the distribution of the variables is overall skewed, as figure 1 depicts. Most of the observations tend to be on the lowest values of the range and around the mean, with the exception of ‘total sulfur dioxide’, ‘ph’, ‘alcohol’ and ‘quality’, which tend to be slightly more centered and spread.

Figure 2 gives another perspective on the distributions considered. Here it is possible to better see how observations tend to be around the mean in most cases, with the four exceptions highlighted before. Moreover, it is evident that in all variables there are potential outliers, which explains why in figure 1 most graphs are shifted on the left and not centered.

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.

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

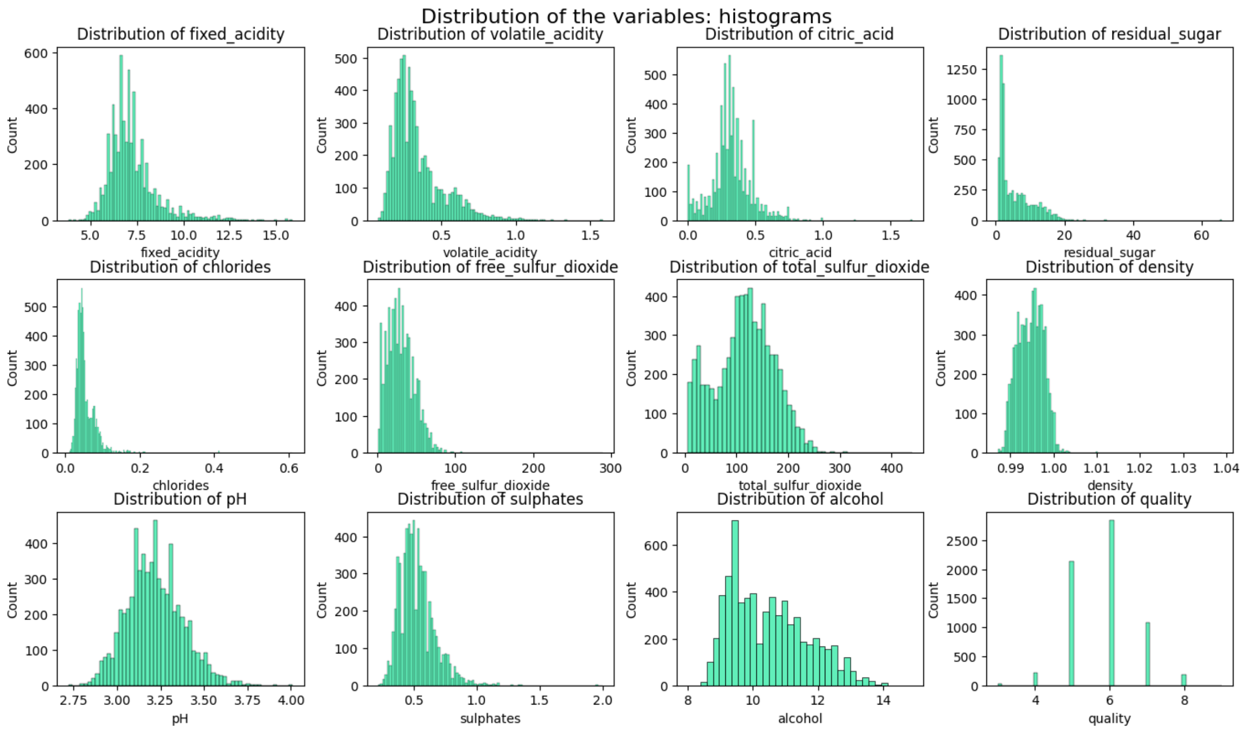

Figure 1: Distribution of variables represented using histograms

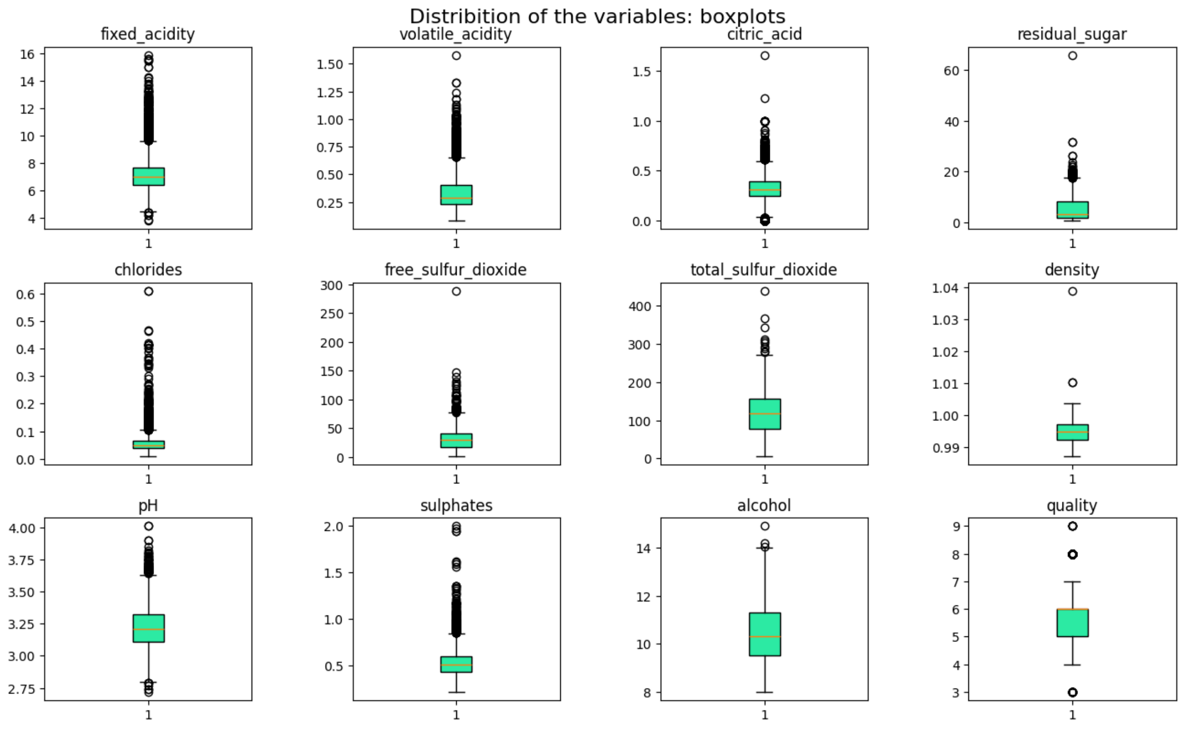

Figure 2: Distribution of variables represented using boxplots

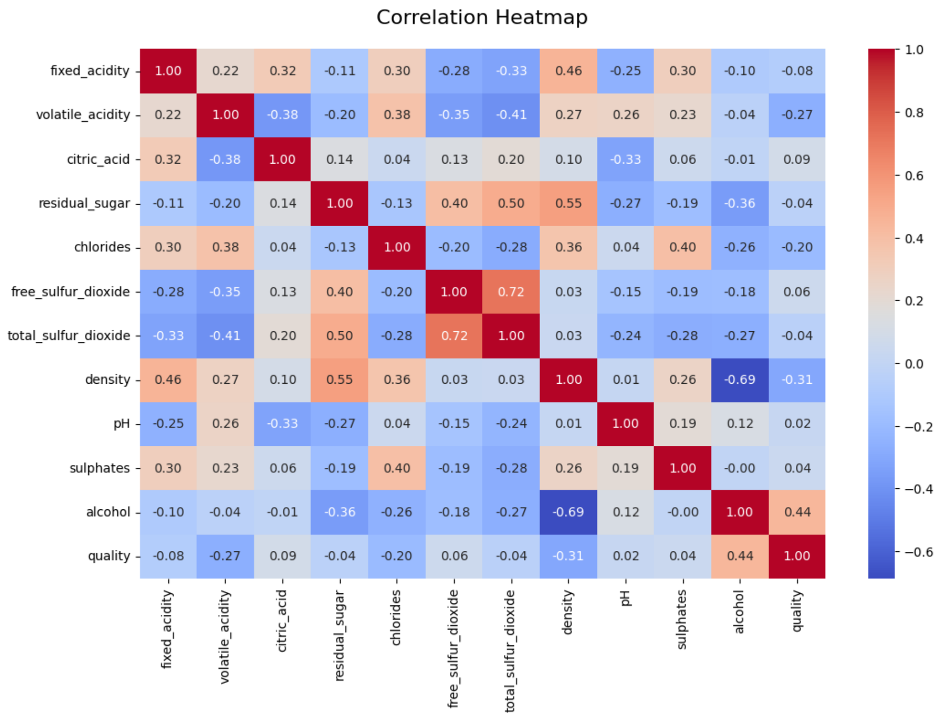

Figure 3: Heathmap of the correlations between variables

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:

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

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.

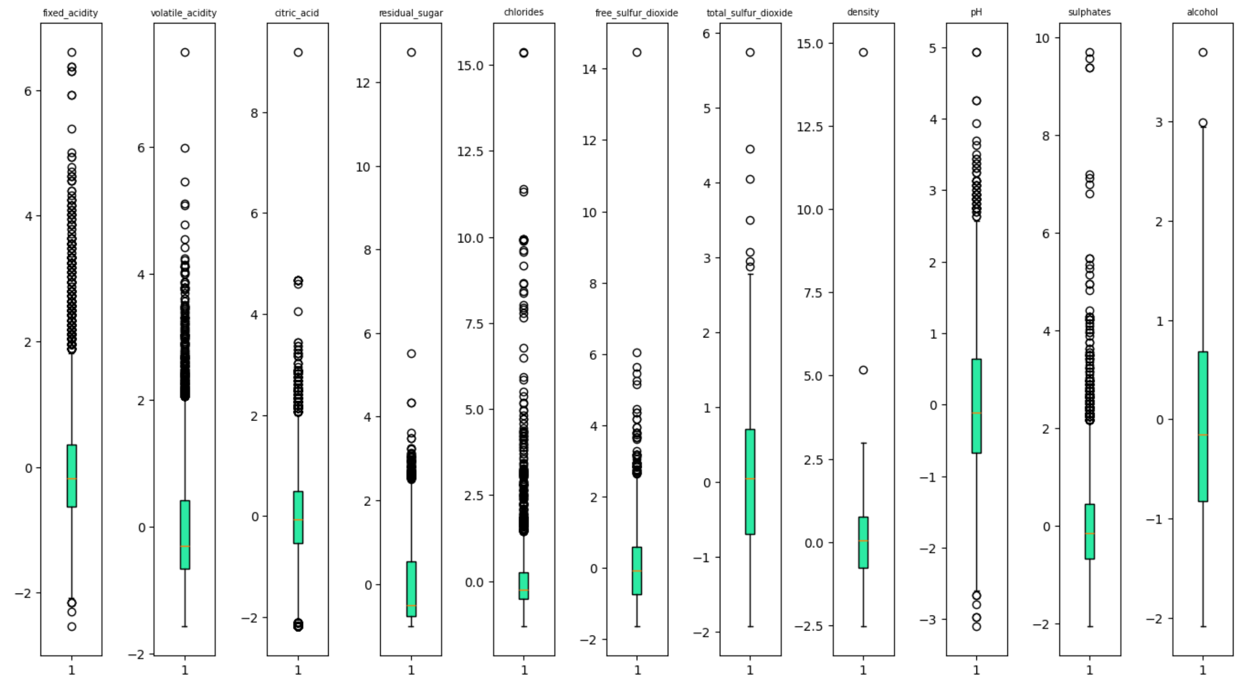

Figure 4: Distribution of standardized variables of the training group represented using boxplots

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 sigmoind 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 hyper-parameters 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

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 (SVM and logistic regression) show very similar and consistent performance. The small gap between training and test indicates that they generalize well and are not overfitting, but the moderate values across all metrics suggest they may be underfitting, as they do not fully capture the complexity of the data. Kernel methods show more varied behaviors. The Gaussian kernel SVM reaches almost perfect performance on the training set but drops sharply on the test set, especially in accuracy and F1-score. This is a clear sign of overfitting, where the model learns the training data too well but fails to generalize. The Gaussian kernel logistic regression, however, performs similarly to the linear models and does not show such overfitting, which suggests it makes only limited use of the kernel's expressive power. Polynomial kernels behave differently: the SVM with polynomial kernel achieves stable and reasonable results across both sets, while the logistic regression with polynomial kernel gives very unbalanced outcomes. In particular, it has low accuracy and precision but perfect recall, meaning it tends to classify nearly all samples as positive, which increases recall but at the cost of many false positives.

In summary, these results illustrate a spectrum of behaviors: linear models are stable but risk underfitting, the Gaussian kernel SVM is powerful but strongly overfits, and the polynomial kernel logistic regression shows poor balance across metrics. The next section on training convergence will further clarify these tendencies, while the following analysis of confusion matrices will provide a more detailed view of the types of errors each model makes.

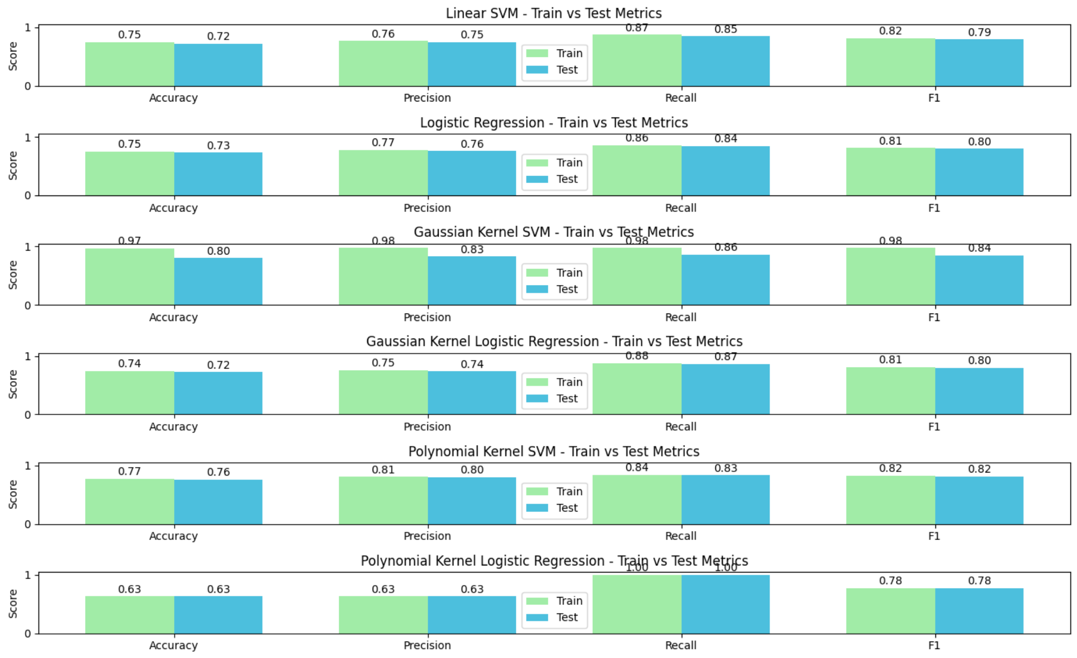

Figure 5: Representation and comaprison of resulting train and test metrics of linear and kernel svm and logistic regression

The confusion matrices in Figure 6 help clarify how the different models handle the two classes. The linear models (SVM and logistic regression) show a consistent pattern: they classify the positive class much better than the negative one. This is partly explained by the imbalance in the target variable, with positives being more frequent. Since linear models build a single straight decision boundary, they tend to favor the majority class, which makes them reliable but somewhat limited. This reflects the earlier point about underfitting: they generalize well but do not fully capture the structure of the data. The Gaussian kernel SVM behaves differently. By mapping the data into a richer feature space, it can almost perfectly separate the classes in training. However, this expressive power comes at a cost: the model overfits, and on the test set its accuracy for the negative class drops sharply. The Gaussian kernel logistic regression shows a more conservative behavior. Although it uses the same kernel, its probabilistic formulation regularizes the decision surface, leading to more balanced but overall weaker results. The polynomial kernel SVM finds a middle ground: its performance remains stable across training and test, with decent recognition of both classes, which suggests a better trade-off between bias and variance. Finally, the polynomial kernel logistic regression collapses into a trivial solution, predicting almost everything as positive. This is linked to its design: logistic regression optimizes class probabilities, and under the effect of the data imbalance and a high-dimensional polynomial expansion, it tends toward the majority class, leading to perfect recall but almost no ability to detect negatives.

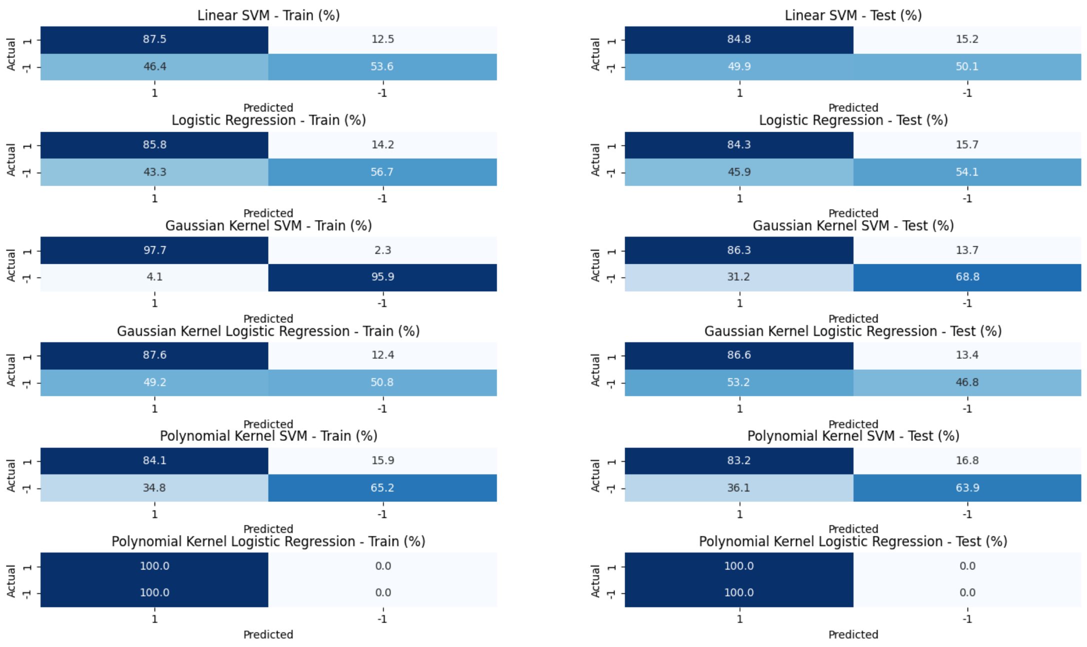

Figure 6: Fonfusion matrices of the four models with percentages per row

Training dynamics

Figure 7 shows the evolution of training and validation loss and accuracy for the linear models. Both the Support Vector Machine (SVM) and logistic regression display similar behaviour, with training and validation curves remaining close to each other throughout the epochs. This pattern confirms what was observed in the evaluation metrics (Figure 5): the linear models generalize well and do not overfit, as indicated by the small gap between training and test performance. At the same time, the moderate values across all metrics, together with the limited improvement in accuracy seen in the learning curves, suggest a certain degree of underfitting. In other words, while these models are stable and consistent, they do not fully capture the complexity of the data.

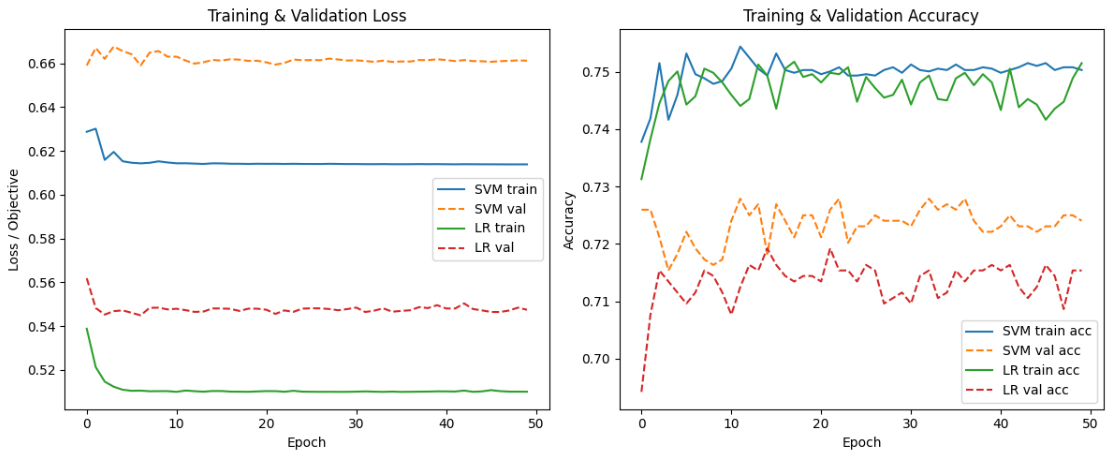

Figure 7: Comparison of training and validation loss (left) and accuracy (right) for linear SVM and Logistic Regression over 50 epochs.

Figure 8 reports the training dynamics for the Gaussian kernel models, highlighting a contrast in learning behavior compared to the linear models. The Gaussian kernel SVM exhibits rapid early improvement but reaches early stopping at epoch 5. Training accuracy climbs above 0.95 while validation accuracy stabilizes around 0.8, with the training loss remaining well below the validation loss. The slight upward drift and plateauing of both loss and accuracy curves indicate that the model has learned the training data extremely well but struggles to generalize, confirming the overfitting observed in the evaluation metrics (Figure 5).

In comparison, the Gaussian kernel logistic regression demonstrates more stable and balanced training. Both training and validation losses decrease steadily and remain very close throughout the epochs, while accuracy rises smoothly from around 0.65 to a plateau between 0.7 and 0.75. The small gap between curves suggests that the model generalizes reasonably well without overfitting. Nevertheless, the moderate final accuracy indicates a certain degree of underfitting, showing that the kernel's expressive power is not fully exploited, similar to what was observed for the linear models.

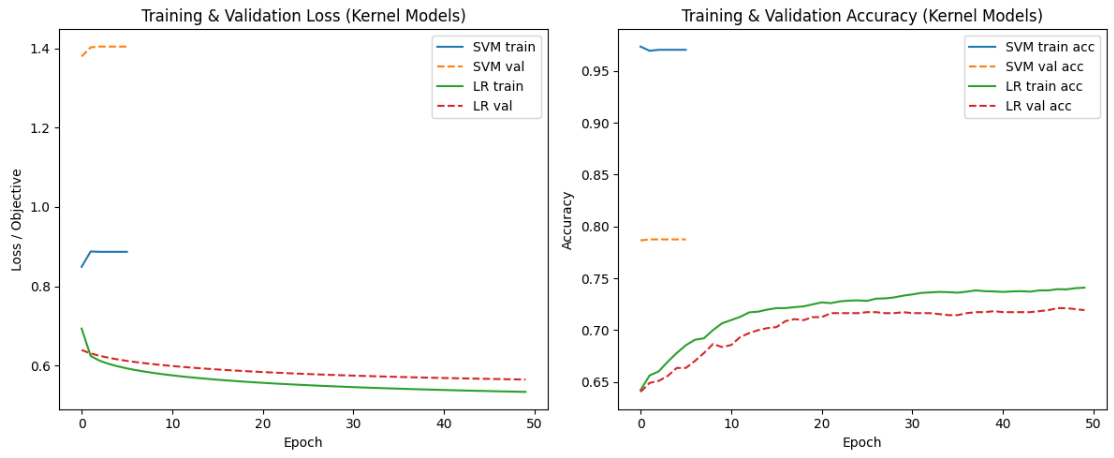

Figure 8: Comparison of training and validation loss (left) and accuracy (right) for Gaussian Kernel SVM and Logistic Regression over 50 epochs.

Figure 9 illustrates the training and validation dynamics for the polynomial kernel models, which display markedly different behaviors for SVM and logistic regression. The SVM with polynomial kernel shows smooth and stable learning: training and validation losses decrease steadily and stabilize, while accuracy rises and levels off for both sets. This indicates that the model generalizes well, balancing the ability to fit the training data while capturing meaningful patterns in the data.

In contrast, the logistic regression with polynomial kernel exhibits highly unstable behavior. Both losses and accuracies fluctuate repeatedly during training, with ups and downs occurring simultaneously for training and validation. This erratic pattern suggests that the model struggles to converge and cannot reliably learn the structure of the data. The instability in the curves, coupled with the moderate performance, indicates underfitting and highlights the model's limited ability to exploit the kernel's expressive power.

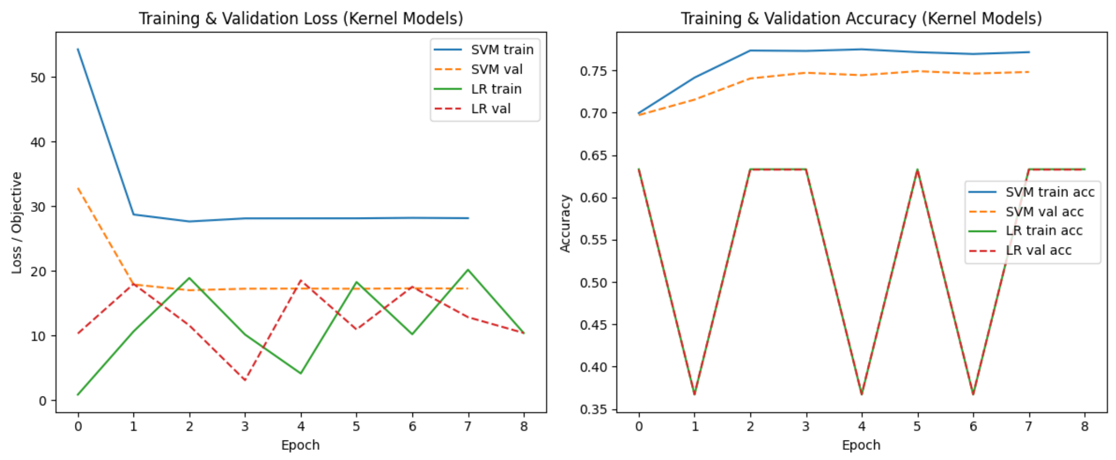

Figure 9: Comparison of training and validation loss (left) and accuracy (right) for Polynomial Kernel SVM and Logistic Regression over 8 epochs.

In summary, the training dynamics corroborate the patterns observed in the evaluation metrics. Linear models remain stable throughout training, with close alignment between training and validation curves, reflecting good generalization but a tendency toward underfitting. The Gaussian kernel SVM, in contrast, achieves near-perfect training performance while validation lags behind, highlighting clear overfitting. The Gaussian kernel logistic regression maintains stable and balanced learning, similar to the linear models, but does not fully exploit the kernel's capacity. Polynomial kernels display contrasting behaviors: the SVM shows steady and reliable learning with consistent performance across training and validation, while the logistic regression exhibits highly unstable and oscillatory curves, indicating underfitting and poor convergence. Overall, these results illustrate a spectrum of model behaviors, from stable but limited linear models to highly expressive but overfitting Gaussian kernels and inconsistent polynomial kernel logistic regression. The following section on cross-validation and hyperparameter analysis will further explore how tuning model parameters influences these training dynamics and generalization patterns.

Cross-Validation and Hyperparameter Analysis

The cross-validation analysis highlights different behaviors for the two linear models (Figure 10). For the SVM, validation accuracy shows a bell-shaped curve, peaking at $\lambda = 10^{-2}$. This reflects the sensitivity of the margin-based formulation: excessive regularization reduces the ability to fit the data, while too little leads to overfitting. The confusion matrix confirms this behavior, with slightly better recognition of the positive class than the negative. Logistic regression, by contrast, selects the minimal $\lambda = 10^{-4}$, with accuracy remaining flat for small values and decreasing only under stronger regularization. This explains its stable performance across training and test and the same asymmetry in the confusion matrix. Overall, the SVM demonstrates a clear trade-off between bias and variance, efficiently capturing patterns without overfitting, whereas logistic regression is more robust to small regularization but slightly underfits, particularly on the minority class.

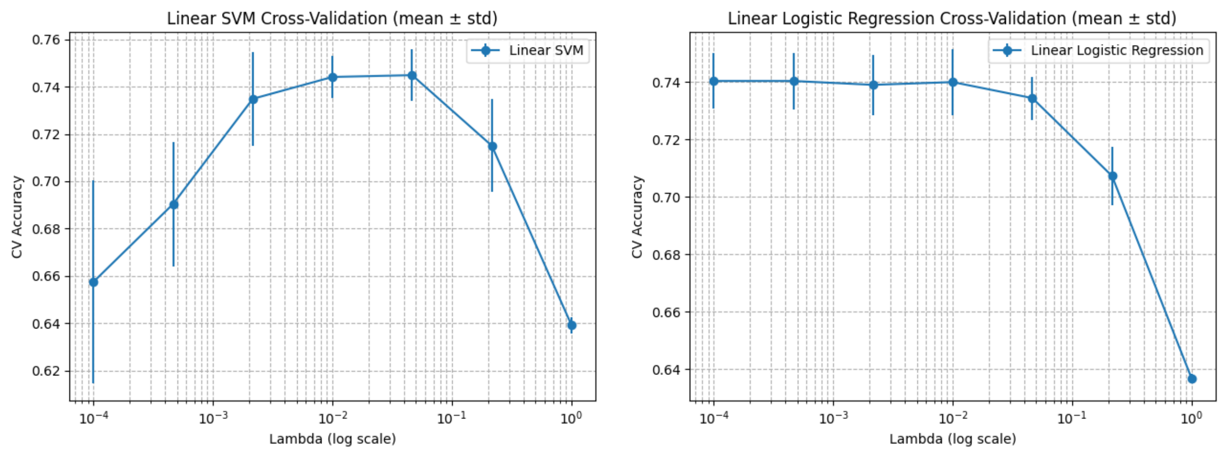

Figure 10: Cross-validation curves for the linear models: SVM (left), logistic regression (right)

The Gaussian kernel highlights distinct behaviors for the two models (Figure 11). For the SVM, the optimal parameters are $\gamma = 1.5848$ and $\lambda = 10^{-4}$, achieving a peak mean accuracy of 0.788. The high accuracy at low λ combined with sensitivity to both γ and λ suggests overfitting: the model captures complex patterns but may not generalize well beyond the training set. Logistic regression reaches its best performance at $\gamma = 0.23$ and $\lambda = 10^{-4}$, with mean accuracy around 0.74. Unlike the SVM, it underfits for most other parameter combinations, indicating limited ability to exploit the kernel's nonlinear transformations. Overall, the SVM is more sensitive to parameter changes and can overfit the data, while logistic regression is steadier but less flexible, trading some accuracy for stability.

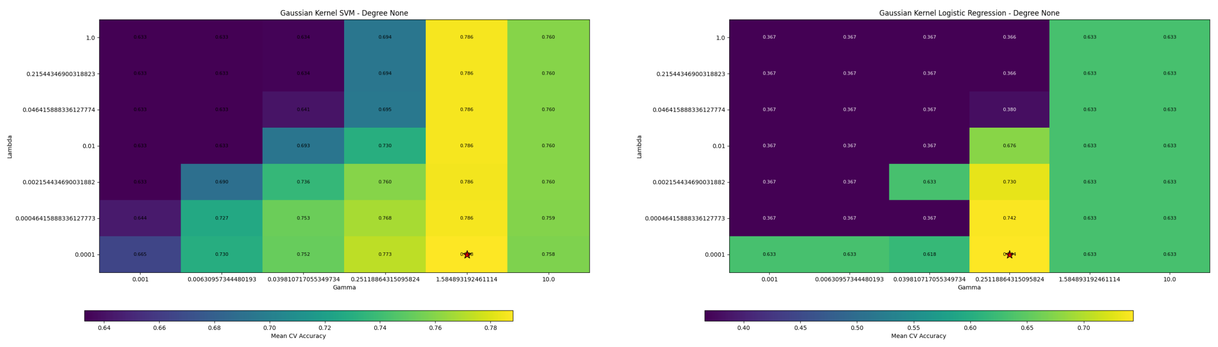

Figure 11: Cross-validation mean accuracy for Gaussian kernel SVM and logistic regression across different gammas and lambda values.

For polynomial kernels, SVM and logistic regression show markedly different behavior(Figure 12). The SVM reaches its peak performance at degree 4 with $\gamma = 0.0398$ and $\lambda = 0.0001$ (mean accuracy 0.756), and at degree 5 with $\gamma = 0.0398$ and $\lambda = 0.0004$ (0.753). These results indicate that the SVM can effectively capture complex nonlinear patterns, but its high sensitivity to gamma and lambda suggests a risk of overfitting if parameters are not carefully tuned. Logistic regression, in contrast, achieves its best performance only at very low gamma and lambda ($\gamma = 0.006$, $\lambda = 0.0001$, mean accuracy 0.633) for both degrees, with most other parameter combinations leading to underfitting. This highlights the limited flexibility of logistic regression under polynomial transformations: it remains stable but cannot fully exploit the kernel's expressive power. Overall, SVM demonstrates strong adaptability with moderate overfitting risk, while logistic regression prioritizes stability at the cost of lower accuracy.

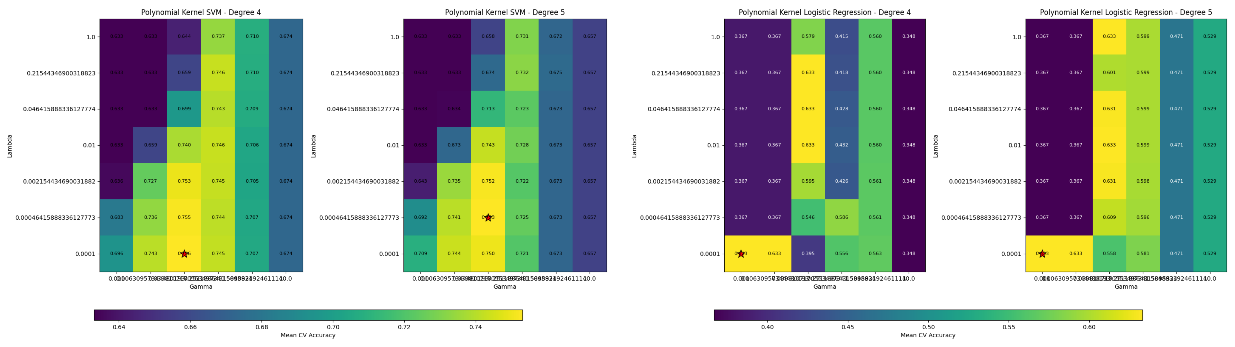
Figure 12: Cross-validation mean accuracy for Polynomial kernel SVM and logistic regression across different gammas and lambda values.

Misclassified Example Analysis

The analysis of misclassified observations highlights systematic differences between models (Figure 13). For most models, including linear SVM, linear logistic regression, Gaussian kernel logistic regression, and polynomial kernel models, misclassifications tend to occur in samples with low alcohol content and high density, while pH, citric acid, and residual sugar remain close to the average. Other features show only small deviations, and the profiles of misclassified observations are very similar between training and test sets, suggesting that these errors are consistent and reflect mild underfitting rather than overfitting. In contrast, the Gaussian kernel SVM exhibits a very different behavior: several features deviate more strongly, and the profiles of training and test misclassifications differ noticeably. This mirrors the observations from the metrics and cross-validation, where the Gaussian SVM achieved nearly perfect training performance but a substantial drop on the test set, and the confusion matrix showed lower accuracy for the minority class. Overall, while most models display consistent and interpretable error patterns, the Gaussian kernel SVM is highly sensitive to specific combinations of feature values, producing sharp misclassification profiles on training and broader, shifted profiles on test data, confirming its tendency to overfit.

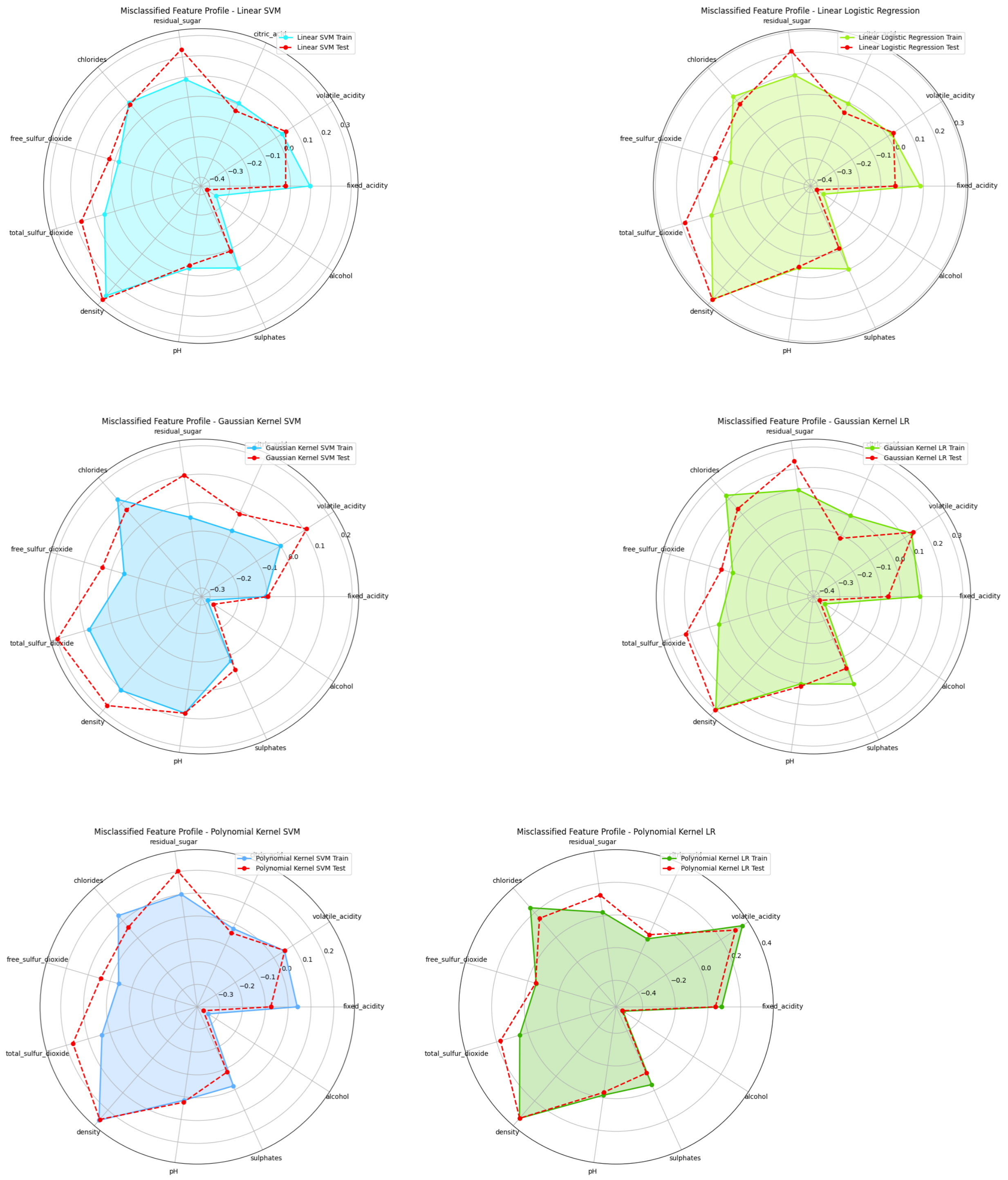

Figure 13: Radar plots of the average profiles of misclassified observations for training and test sets across all models

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.