Wine Quality Prediction using SVM and Logistic Regression

Alessandro Macchi
MSc Data Science for Economics
Università degli Studi di Milano

August 28, 2025

Abstract

This project investigates the effectiveness of machine learning approaches for wine quality classification using physicochemical properties. Four models were implemented and compared: Linear Logistic Regression, Linear Support Vector Machine (SVM), Kernel Logistic Regression, and Kernel SVM. Data preprocessing involved logarithmic transformation to address feature skewness, standardization for scale normalization, and SMOTE (Synthetic Minority Oversampling Technique) to handle class imbalance. Hyperparameter optimization was conducted using 5-fold cross-validation, with models evaluated on accuracy, precision, recall, F1-score, and ROC-AUC metrics. Results demonstrate that kernel methods significantly outperform their linear counterparts, indicating the presence of important non-linear patterns in the data. Kernel Logistic Regression achieved the best overall performance across evaluation metrics, while Kernel SVM showed the fastest convergence and lowest training loss. Linear models exhibited mild underfitting, particularly Linear SVM. These findings suggest that the complexity of kernel methods is justified for wine quality prediction tasks where non-linear relationships are prevalent.

Contents

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. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

1 Introduction

The objective of this project is to investigate the effectiveness Machine Learning approaches for wine quality (low vs. high) using 11 features including acidity, alcohol content and sulfur compounds.

Four models are implemented Logistic Regression, Linear SVM, Kernel Logistic Regression and Kernel SVM. One of the key aspect of the project is to assess whether the complexity of kernel methods is justified by improved performance over simpler linear approaches.

The report starts with Data Exploration and preprocessing techniques. Afterward, the models' structure is presented with the hyperparameter tuning via 5-fold cross validation. In conclusion, a performance evaluation is exhibited using multiple metrics, detailed comparison of learning dynamics and misclassification patterns.

2 Data Exploration and Preprocessing

Before manipulating the datasets, it is necessary to verify the characteristics of the data through Exploratory Data Analysis (EDA).

The most important insights gained from descriptive statistics are the absence of missing values and the diversity of the variable scales in the datasets, which will require normalization in the preprocessing section.

Additionally, the quality distribution demonstrates a slight class imbalance, with 63% of the wines being high quality, while the rest being low quality. The difference is not severe, but data augmentation techniques may help improve the predictions.

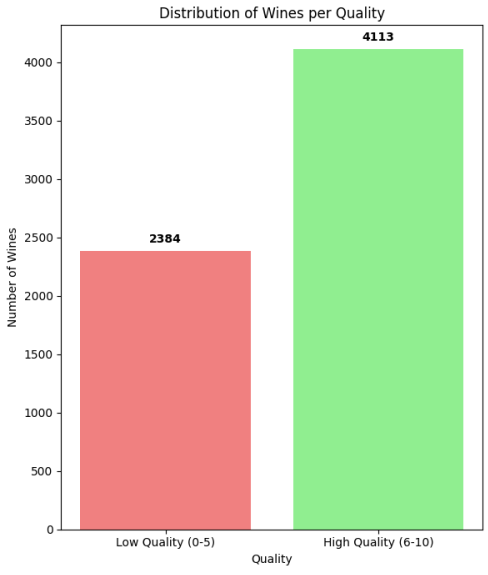

Figure 1: Quality distribution per class

The correlation heatmap shows a strong relationship between some variables, especially *density* and *alcohol*, but it does not justify removing one of them. Moreover, examining the distribution of the variables, the skewness validates using logarithmic transformation in the following section.

Preprocessing begins by creating a unified dataset with a binary label for target, where -1 stands for low quality ($quality < 6$) and $+1$ indicates high quality ($quality \geq 6$). The dataset is then divided in features (X) and label (y). Note that quality is dropped from the features set to avoid perfect collinearity with the binary target and it will not be used from now on.

Afterwards, before transforming any feature, data splitting must be implemented. The test set comprehends 20% of the dataset and the information of this portion will be used only to validate the analysis. This operation must be completed as one of the first steps to avoid data leakage.

Logarithmic transformation is then introduced to reduce the skewness problem and standardization provides an additional assurance that the values operate around the same range. Note that standardization is performed on the test set using the mean and the standard deviation of the training part to prevent data leakage.

Finally, as explained previously, the data augmentation technique SMOTE (Synthetic Minority Oversampling Technique) is used to address class imbalance (63%-37%), even though the difference is not extreme. SMOTE consists in synthetically generated observations that help the less numerous class to be equal to the majority. In this way, it is possible to achieve better classifier performance (in ROC space) than with other techniques [1].

3 Methods

The first implemented model is Logistic Regression. A predictor $g : \mathcal{X} \to \mathbb{R}$ is trained and then used to determine the conditional probability $\eta(x) = P(Y = 1|X = x)$, via the logistic sigmoid function $\sigma(g(w^T x + b))$:

$$\sigma(g(w^T x + b)) = \frac{1}{1+e^{-g(w^T x+b)}} \in (0, 1)$$

where $w \in \mathbb{R}^d$ are the weights (i.e., feature contributes to the log-odds of the positive class), and b is the bias term, a constant term that shifts the decision boundary left or right.

There are three hyperparameters:

1. Learning rate: controls the steps in gradient descent updates. When lower, the updates are slower, but more stable.

2. Regularization strength: controls how much penalty is assigned to large weights, preventing overfitting by shrinking coefficients. An L2 regularization term was chosen (also for the other models), which applies the penalty to the sum of squared weights.

3. Epochs: number of iterations through the training set during learning. A higher value makes it easier for the model to converge, but may cause overfitting.

The loss is obtained by computing the logistic loss with the L2 regularization explained above.

Given the relatively low computational complexity, standard gradient descent can be applied, which continuously adjusts parameters by moving in the direction opposite to the gradient of the loss function. The process continues until convergence, which occurs when the gradients become very small or the loss stops decreasing.

Following Logistic Regression, Support Vector Machines (SVM) are implemented, using the Pegasos algorithm. Unlike the probabilistic approach of logistic regression, SVM focuses on finding the optimal separating hyperplane that corresponds to the maximum margin.

The SVM optimization objective is:

$$\begin{aligned} \min_{(w,\xi)\times\mathbb{R}^{d+m}} \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{t=1}^{m}\xi_t \\ \text{subject to} \quad & y_t w^\top x_t \geq 1-\xi_i, \quad i=1,\ldots,m, \\ & \xi_t \geq 0, \quad i=1,\ldots,m. \end{aligned}$$

Where w is the weight vector, λ is the regularization parameter and ξ_i are slack variables.

The Primal Estimated Sub-Gradient Solver (Pegasos) is a Stochastic Subgradient Descent, which efficiently solve the optimization problem of SVM. Since the run-time does not depend directly on the size of the training set, the resulting algorithm is especially suited for learning from large datasets [2]. Since in Pegasos the learning rate in Pegasos at iteration t is $\eta_t = \frac{1}{\lambda t}$, the learning rate does not need to be tuned separately from the regularization parameter λ.

Therefore, there are only two hyperparameters:

1. Regularization strength (λ): controls overfitting via L2 regularization. Higher values create simpler models by penalizing large weights more heavily.
2. Total number of iterations (max_iter): more iterations allow for better convergence but increase training time.

The update rule changes the solution in every iteration by shrinking weights (regularization). Additionally, when a point violates the margin, it also pushes the decision boundary away from that point.

Next, the kernelized models are analyzed.

Kernelization happens by replacing the feature mapping with a kernel function that directly computes inner products in the (possibly infinite-dimensional) feature space [3].

The model uses a dual representation, which is significantly more efficient than the primal form and allows us to use the kernel trick, enabling nonlinear classification without explicitly computing a feature map $\Phi(x)$.

This is possible because Logistic Regression can be expressed in dual form:

$$f(x) = \sum \alpha_i K(x_i, x)$$

where α_i are learned coefficients (analogous to support vector weights), $K(x_i, x)$ is the kernel function measuring similarity and x_i are the support vectors from training data.

In other words, during training, instead of updating weights directly, the kernel matrix between batch and support vectors is computed and the gradients are computed with respect to alphas instead of weights.

In this situation the biggest challenge is making it computationally efficient. To assess this problem, mini-batch with stochastic gradient descent algorithm was used [4]. It works with a small random subsets (here of size 64[1]) for each gradient update, obtained by shuffle all training data each time. The choice must consider the trade-off between memory usage and convergence. Mini-batch was integrated with early stopping with a patience of 20 epochs: patience specifies the maximum number of epochs to continue training without improvement in validation metrics, preventing unnecessary training time. This value was chosen following established regularization practices.

Another technique is subsampling of support vectors, which are randomly selected (to avoid overfitting), and reduces kernel matrix dimensionality. The subsampling ratio was set to 30% based on the need to balance computational efficiency with model representativeness.

The hyperparameters of this model are as follows:

1. Kernel: The Gaussian (RBF) kernel with γ as a parameter that controls the width and smoothness of the kernel. The formula of the RBF kernel is

   $$K_\gamma = exp(-\frac{1}{2\gamma}||x - x'||^2), \quad \gamma > 0$$

   The Polynomial kernel has form

   $$k_n(x, x') = (x \top x' + c)^n, \quad n \in \mathbb{N}$$

   where the parameters are the constant term c (in code $coef0$) and the degree of the polynomial n (in code $degree$)

2. Regularization strength λ

3. Epochs: these also determine the learning rate, which is adaptive and starts at 0.01, decaying over time.

[1] for computational efficiency, it is better to choose power-of-2 batch sizes [5]

The final model is Kernel SVM, implemented using the kernel version of Pegasos. In order to do this, it uses the dual form based on the most significant support vectors, which are filtered based on a threshold that helps to consider only the relevant vectors with high coefficients α.

This version keeps all the good characteristics of Pegasos, such as online learning and efficiency in convergence.

The update rule is analogue to linear SVM: when the margin is violated a new support vector is added and stored in the support set. In addition, all coefficients decay $\alpha_i \leftarrow \alpha_i \times (1 - \eta\lambda)$ for all existing support vectors. When there is no violation, there is just coefficients decay and no new support vector is added. The kernel version implements the same mathematical operation as linear Pegasos, but in coefficient space rather than weight space.

There are three hyperparameters:

1. RBF and polynomial kernels
2. Regularization strength λ
3. Total number of iterations (max_iter)

In conclusion, linear models offer interpretability and computational efficiency, while the kernelized versions handle complex nonlinear patterns at the cost of increased computational complexity.

4 Experimental Setup

Firstly, the training pipeline of each model is

1. Define hyperparameter grids for each model
2. Perform grid search with cross-validation
3. Train final model with best hyperparameters
4. Evaluate on test set with adequate metrics
5. Compare model performance across all variants

All models use 5-fold cross-validation to ensure proper hyperparameter tuning. The dataset is initially split into training and test sets. During cross-validation, only the training set is used, being iteratively divided into training folds (80%) and validation folds (20%).

In each of the five iterations, a different fold serves as the validation set, while the remaining four folds are used for training. The accuracy score represents the performance metric used across all five folds.

The hyperparameter grids are defined in *hyperparameter_tuning/parameters_grid.py*, and the best-performing parameters (those achieving the highest mean CV accuracy[2]) are selected for training the final models on the complete training set.

These models are then evaluated on the test set, producing plots and files to store the principal resulting metrics.

Finally, both plots and performance metrics are saved in the directory *output/timestamp*.

5 Results and Evaluation

According to this analysis, the best parameters for each model are:

Model	Learning rate	Kernel	λ	Epochs	Max. Iter.
L. Logistic Regression	0.05	–	0.08	1000	–
Linear SVM	$\frac{1}{\lambda t}$	–	0.1	–	3000
K. Logistic Regression	$\frac{0.01}{1+0.001 \cdot epochs}$	RBF($\gamma = 0.20$)	0.01	1000	–
Kernel SVM	$\frac{1}{\lambda t}$	RBF($\gamma = 0.30$)	0.001	–	3000

Table 1: Hyperparameters for different models[3]

Linear models use more aggressive learning rates compared to their kernel counterparts. This makes sense since kernel methods operate in higher-dimensional feature spaces where smaller steps are typically needed to avoid overshooting optima. However, note that the value for Kernel Logistic Regression is adaptive and reaches very low values at the end of the training.

Additionally, Kernel SVM uses weak regularization ($\lambda = 0.001$) making it more flexible and more likely to overfit in the transformed feature space, while Linear SVM uses a higher regularization ($\lambda = 0.1$). The iteration limits reveal different optimization challenges. Kernel SVM gets a very high budget (3000 iterations), likely due to the complexity of the dual optimization problem.

To compare and evaluate prediction quality, the metrics in Figure 2 have been used.

It can be easily noticed that the best model, according to Figure 2, is Kernel Logistic Regression and that the application of kernels helps to improve the quality of the prediction, both for SVM and Logistic Regression.

Figure 3 shows that all four models clearly perform better than the random classifier, but the kernel methods surpass the linear ones, especially Linear SVM, which, consistently with the previous results, presents some underfitting.

[2]The parameter grids were reduced from more extensive searches to balance computational efficiency with thorough exploration of the hyperparameter space.

[3]The learning rates for SVM, Kernel Logistic Regression and Kernel SVM are adaptive, therefore not tuned.

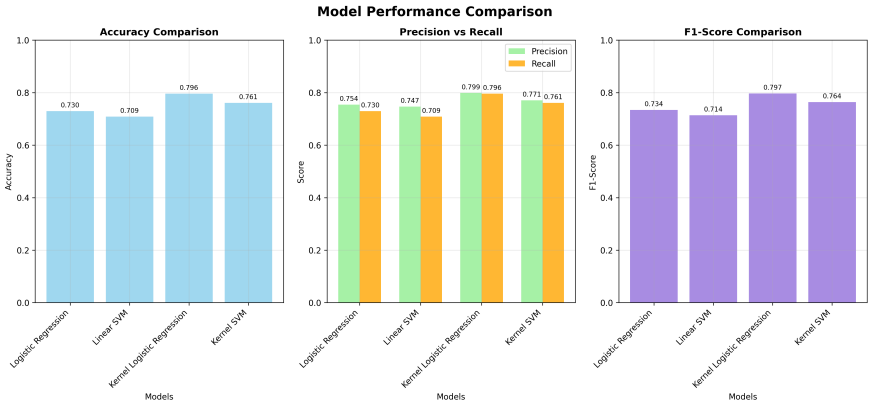

Figure 2: Accuracy, Precision, Recall and F1-score for each model

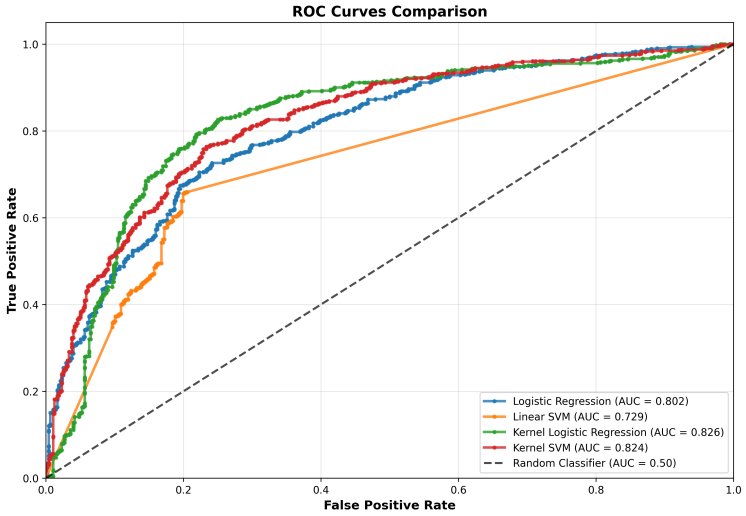

Figure 3: ROC curves

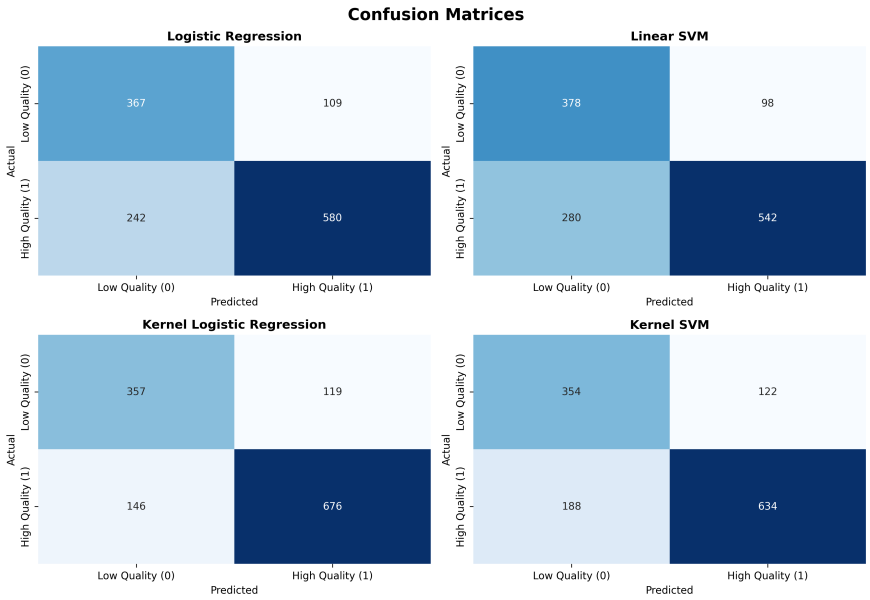

Figure 4: Confusion matrices

Figure 4 demonstrates that the model with less misclassifications is Kernel Logistic Regression.

Notice that all the models tend to overestimate the low quality class. This happens because the test set is imbalanced, while the application of SMOTE to training data made the class numerically equal. Linear models struggle more with this problem, while kernels methods have more balanced errors, suggesting a better handling of the imbalance.

6 Discussion

Figure 5 shows that Kernel SVM has the steepest drop and lowest final loss and converges very quickly (within 20–30 epochs), clearly showing the best performance in terms of minimizing loss. Kernel Logistic Regression reaches a stable loss, better than linear models but worse than Kernel SVM.

Linear SVM drops sharply, but then oscillates with noisy fluctuations. Logistic Regression decreases initially but then gradually increases again over epochs, reaching the worst loss among the four.

Note that Linear SVM and Kernel SVM use hinge loss, while Linear Logistic Regression and Kernel Logistic Regression the logistic loss.

While Figure 4 showed a general view of the misclassifications of the four models, Figure 6 provides insights about which variables influence errors the most.

Figure 6 shows that *chlorides, fixed_ acidity and sulphates* are the most problematic features across all models, consistently showing the highest misclassification pattern scores. This

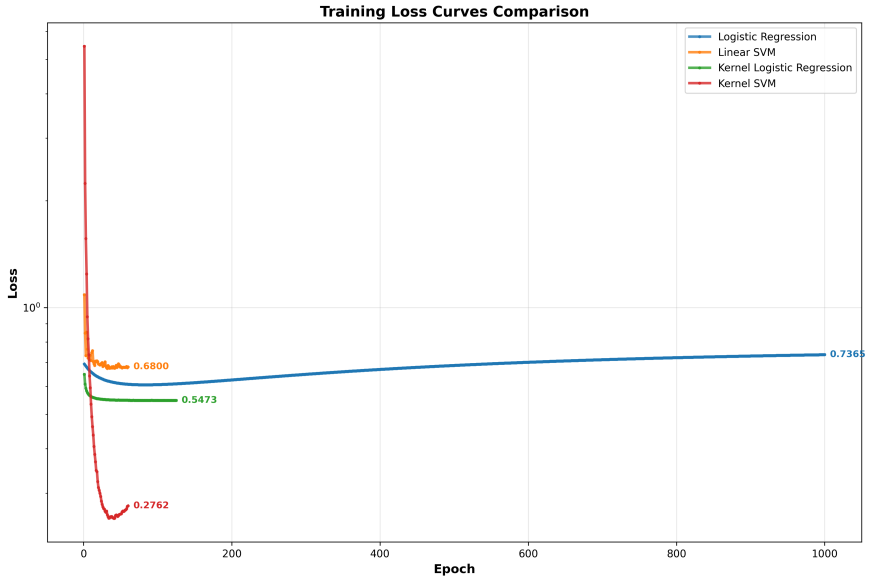

Figure 5: Loss curves

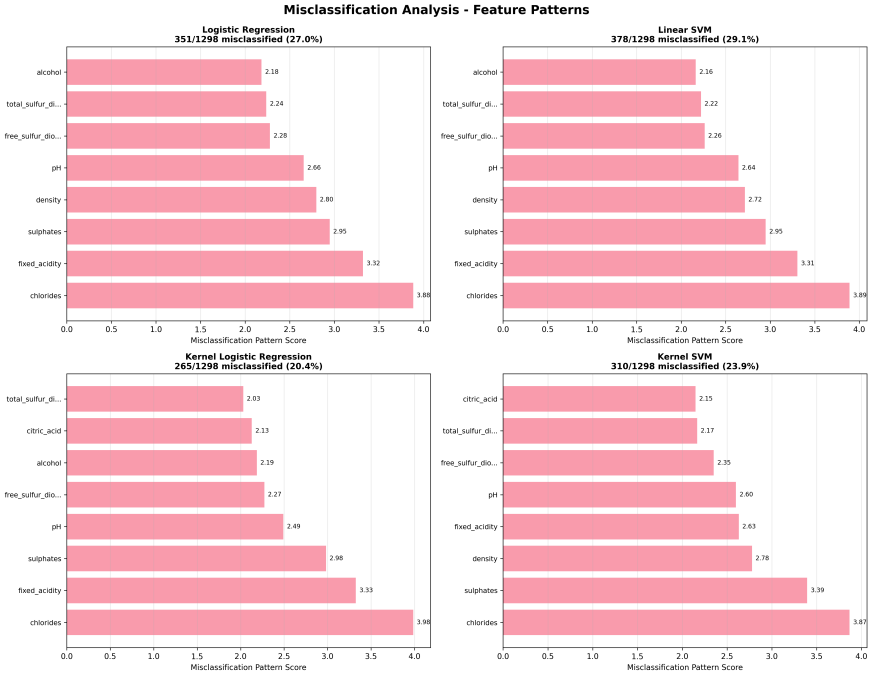

Figure 6: Feature analysis

suggests these features may be noisy and may require feature engineering and that the problem is probably more data-level rather than model-level.

To address the problem of underfitting and overfitting, Figure 7 presents the learning curves.

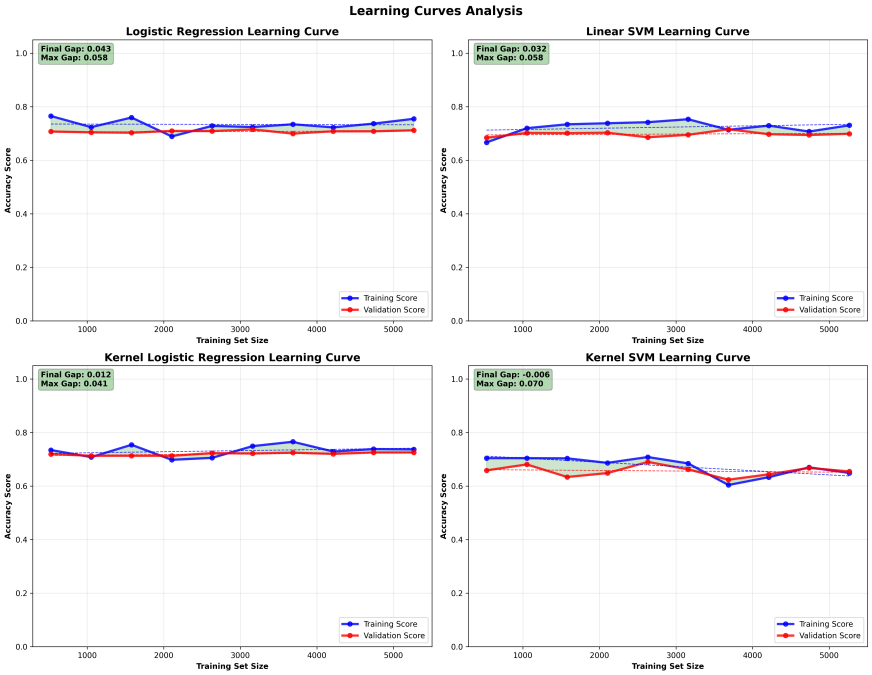

Figure 7: Learning curves

Kernel Logistic Regression stabilizes, reaching a very low final gap. Kernel SVM initially shows signs of overfitting, but later converges well. In contrast, both linear methods consistently display higher values in training and validation score, exhibiting the limits of linearity in prediction.

In Figure 5, Logistic Regression showed a relatively high training loss that even increased after many epochs, suggesting underfitting, due to its limited capacity. This interpretation is supported by Figure 7.

Linear SVM reached a plateau in loss at a high level, confirming persistent underfitting, as both training and validation accuracy remained modest. This thesis is confirmed by the ROC curve, which indicates poor generalization.

Kernel Logistic Regression performs well, both in training loss and in learning curves, showing improved generalization as more data were used. It consistently achieves the best scores on all metrics and shows stable performance, making it the best model.

Finally, Kernel SVM achieves the lowest training loss, suggesting an excellent fit on the training set and obtaining results almost as good as Kernel Logistic Regression, as shown in Figure 3. However, its slightly worse misclassification rate (3% higher than Kernel Logistic Regression) makes it the second-best model.

7 Conclusion

Kernel methods significantly outperform their linear counterparts, suggesting the dataset contains important non-linear patterns that simpler linear models cannot capture effectively.

Kernel Logistic Regression is the best in almost all the evaluation criteria, but with more work Kernel SVM may surpass it, as shown by Figure 5. Linear Logistic Regression is consistently better than Linear SVM, which, as illustrated in Figures 2 and 3, exhibits underfitting.

The greatest problem is the trade-off in computational time. Kernel models are more than ten times slower than linear ones, while receiving an improvement of around 6% in misclassifications. This creates a contrast between efficiency and accuracy that must be carefully assessed.

References

[1] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[2] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: primal estimated sub-gradient solver for svm. *Mathematical Programming*, 127(1):3–30, 2011.

[3] Gustav Eriksson and Emil Belin. A brief introduction to reproducing kernel hilbert spaces, 2024.

[4] Kenneth Ezukwoke and Samaneh Zareian. Logistic regression and kernel logistic regression—a comparative study of logistic regression and kernel logistic regression for binary classification. *University Jean Monnet: Saint-Etienne, France*, 2019.

[5] Vincent Peter C Magboo and Patricia Angela R Abu. Analysis of batch size in the assessment. In *Agents and Multi-agent Systems: Technologies and Applications 2023: Proceedings of 17th KES International Conference, KES-AMSTA 2023, June 2023*, volume 354, page 221. Springer Nature, 2023.