Grado en Ingeniería Informática

Lab assignment 4: Support Vector Machines (SVMs)

Universidad de Córdoba

Fourth year of "Grado en Ingeniería Informática"
Dpto. de Informática y análisis Numérico
Introduction to computational models
Course 2023-2024

Autor:

Antonio Llamas García i92llgaa@uco.es
31886320V

Docentes:

Pedro A. Gutiérrez Peña pagutierrez@uco.es
Javier Sánchez Monedero jsanchezm@uco.es
César Hervás Martínez chervas@uco.es

Córdoba, 19 de febrero de 2024

Índice

1. 2D representation of SVMs

1.1. Question 1

The provided script performs the following tasks:

1. **Import Statements**: Necessary libraries such as `numpy`, `matplotlib`, `pandas`, and `svm` from `sklearn` are imported for numerical computing, plotting, data handling, and SVM implementation, respectively.
2. **Dataset Loading**: The script loads a dataset from a CSV file named 'dataset1.csv' using `pd.read_csv()` from the `pandas` library. The dataset is divided into features (`X`) and labels (`y`).
3. **SVM Model Training**: An SVM model is trained using the loaded dataset. The chosen parameters for the SVM model are a linear kernel (`kernel='linear'`) and a regularization parameter `C` set to 100.
4. **Data Visualization**: The script visualizes the data points using `plt.scatter()` from `matplotlib`. It also constructs a meshgrid to visualize decision boundaries.
5. **Decision Boundary Calculation**: The decision function values are computed for each point in the meshgrid using `svm_model.decision_function()`. These values are used to plot decision regions and boundaries.
6. **Plotting**: The decision regions and boundaries are plotted using `plt.pcolormesh()` and `plt.contour()`, respectively. The plot is then displayed using `plt.show()`.

Type of Kernel and Training Parameters

- **Kernel Type**: The script uses a linear kernel for the SVM model. This implies that the decision boundary is a hyperplane in the feature space.
- **Training Parameters**:
  - **Kernel**: Linear kernel (`kernel='linear'`).
  - **Regularization Parameter (`C`)**: The parameter `C` is set to 100. This parameter controls the trade-off between maximizing the margin and minimizing the classification error. A higher value of `C` indicates a higher penalty for misclassification, potentially leading to a more complex decision boundary.

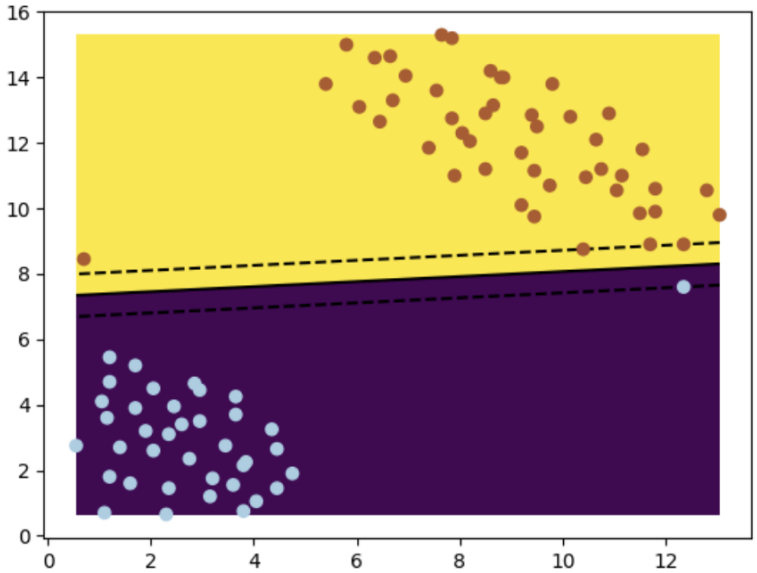

Figura 1: Question 1

2. First example dataset

2.1. Question 2

Intuitively, just by observing the points, we can infer that the most useful separating hyperplane for distinguishing between the two classes is the linear one. This is because a straight line is sufficient to separate both classes.

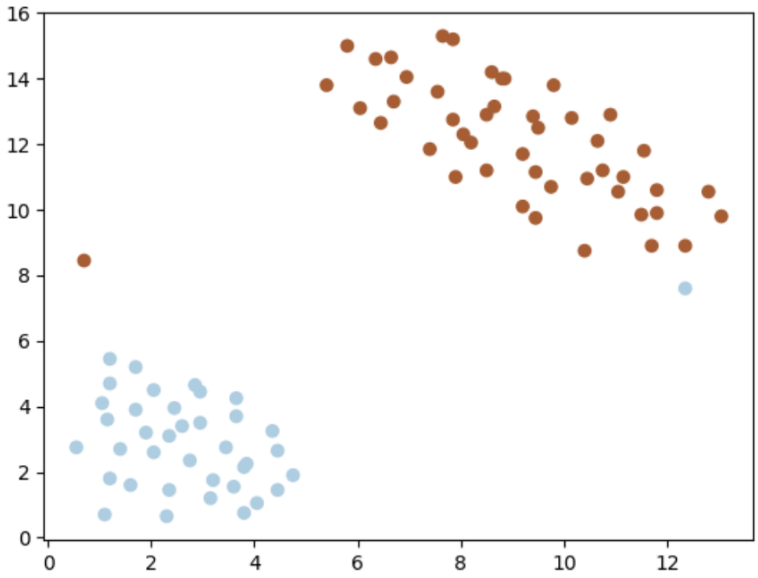

Figura 2: Question 2

2.2. Question 3

If we revisit the original script and experiment with different values of C, we can assess the impact of this parameter. When C is low, the SVM tends to have a wider margin, which could potentially lead to a higher number of classification errors.

On the other hand, a high value of C produces a narrower margin, making the SVM more restrictive in its classification. Finding the optimal value for the margin is crucial to ensure accurate classification of all patterns without overfitting.

In the context of this specific dataset, it is not until C reaches a value of 10 that all patterns are classified correctly. A value of C equal to 100 results in a more restrictive margin, and increasing C further does not lead to significant additional changes in classification. Therefore, it seems that a value of 100 yields the most satisfactory results in this case.

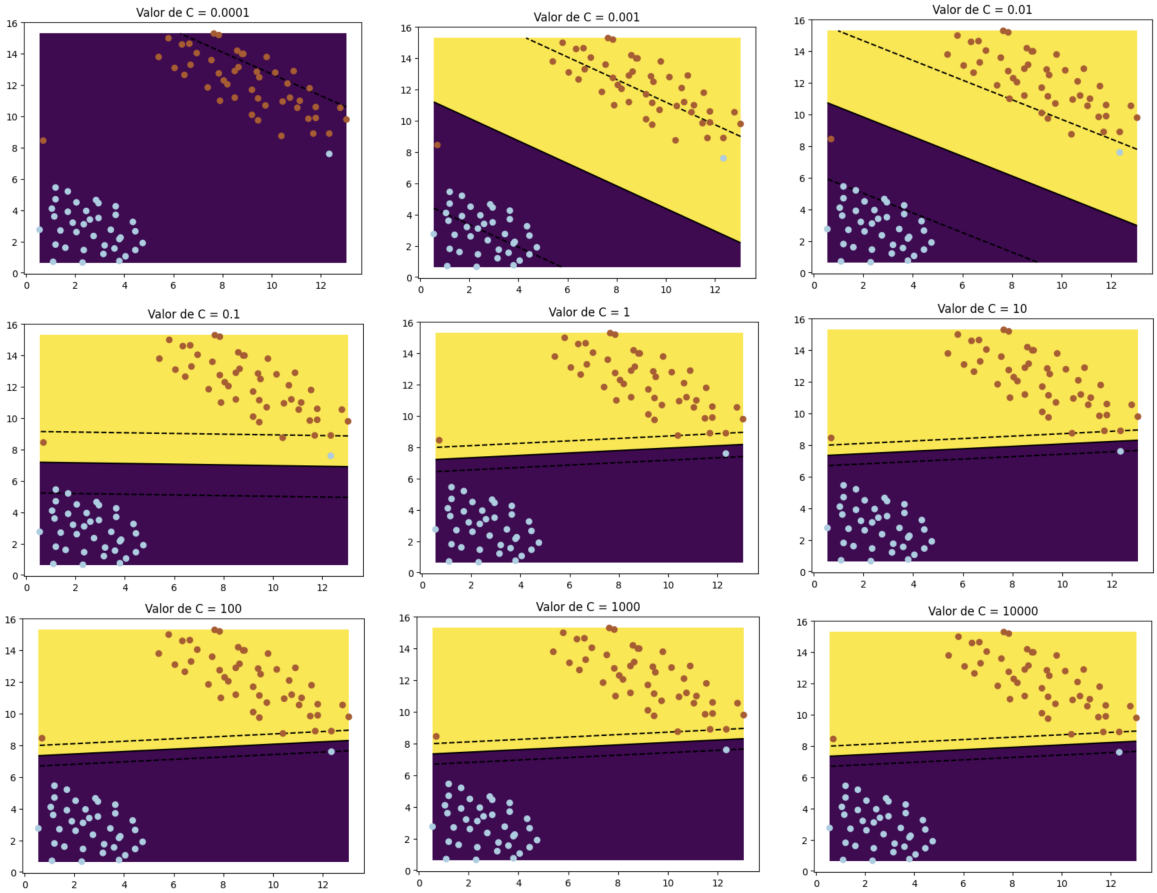

Tabla 1: Modifications C

3. Second example dataset

3.1. Question 4

It becomes apparent that the model struggles to achieve accurate classification. Despite attempting various adjustments to the values of C, the outcomes continue to exhibit similar levels of ineffectiveness. This persistent failure can be attributed to the inherent limitations of the chosen kernel type.

The dataset under consideration evidently lacks linear separability, rendering the utilization of a linear separating hyperplane ineffective in resolving this classification challenge. This underscores the importance of carefully selecting an appropriate kernel function that can effectively capture the underlying structure of the data for improved classification performance.

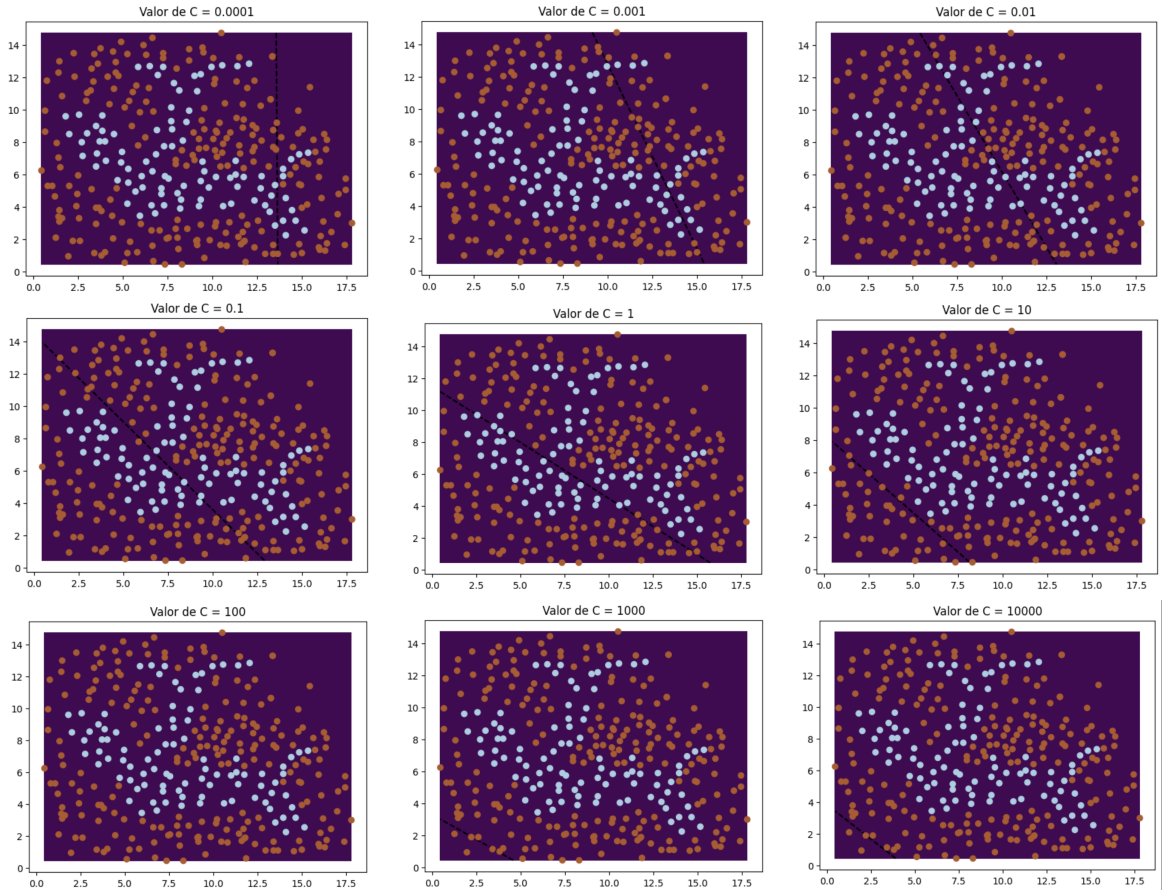

Tabla 2: Modifications in C

3.2. Question 5

As observed, lower values of gamma, such as the first three configurations, tend to result in underfitting, where the model fails to capture the complexities of the data adequately.

Conversely, higher values of gamma, such as the last two, lead to overfitting, where the model becomes overly sensitive to the training data and fails to generalize well to unseen data.

Additionally, the value of C=10 and the range of gamma values [2e-3, 2e-2, 2e-1, 2e0, 2e1, 2e2] were explored to comprehensively evaluate the model's performance under various parameter configurations.

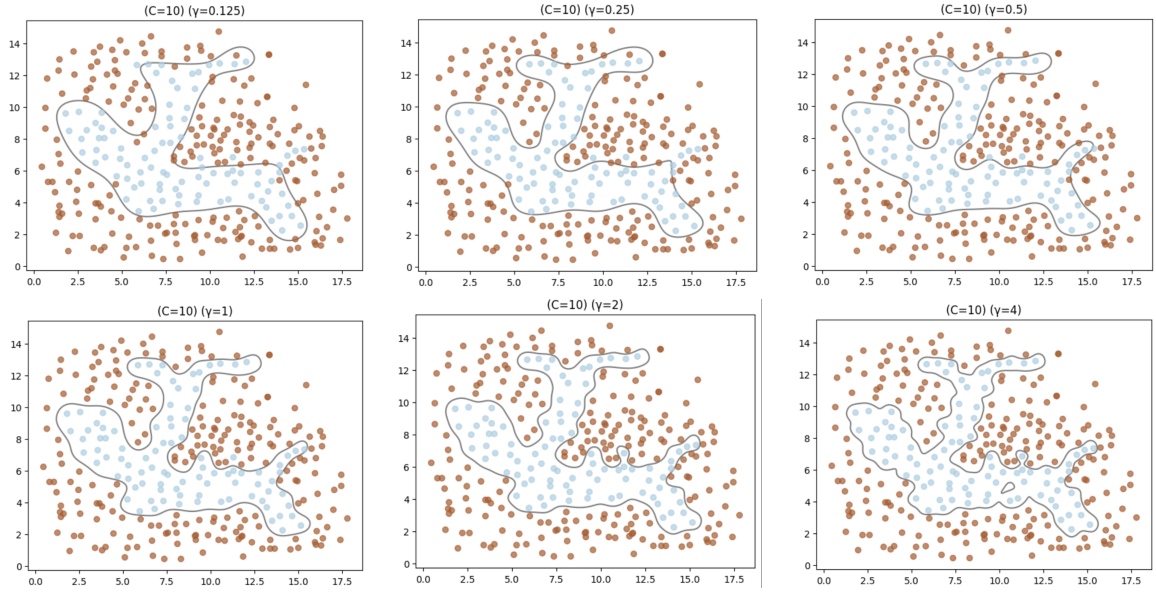

Tabla 3: Modifications in gamma

It's noteworthy that upon evaluation, the optimal value for gamma was found to be 2. Building upon this revelation, we will proceed to explore various configurations for the parameter C, including the values of 0.01, 0.1, 1, 100, 1000, and 10000, in order to determine the optimal model configuration.

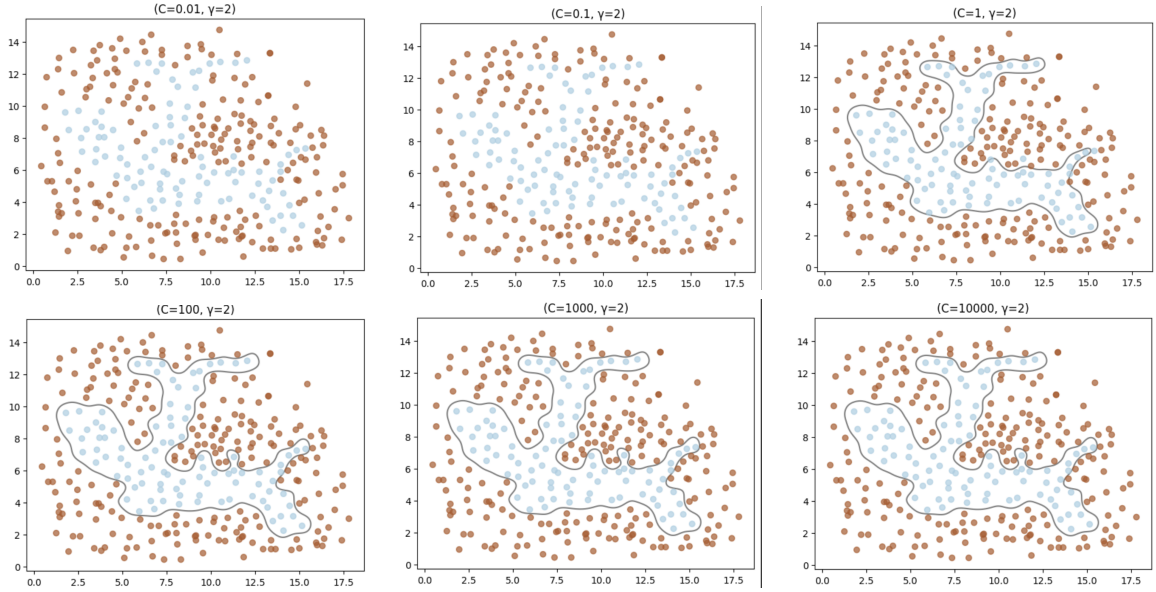

Tabla 4: Modifications in C

4. Third example dataset

4.1. Question 6

It's evident that a linear kernel is insufficient for accurately separating this dataset. Upon closer inspection of the points, it becomes apparent that there are a few outliers, situated within the region of one class but erroneously classified as the other.

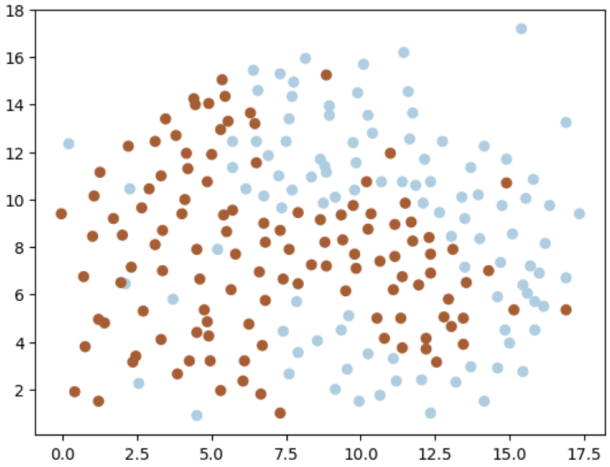

Figura 3: Dataset 3

4.2. Question 7

Therefore, we will employ a non-linear SVM to effectively classify the data. Building upon the findings from previous evaluations, we will utilize the best configuration obtained earlier to prove the optimal model performance.

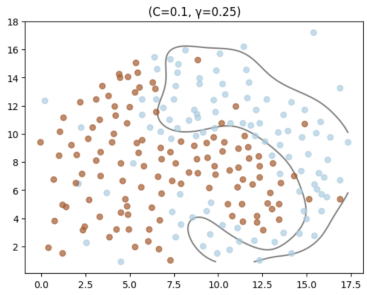

Tabla 5: Under-fitting model

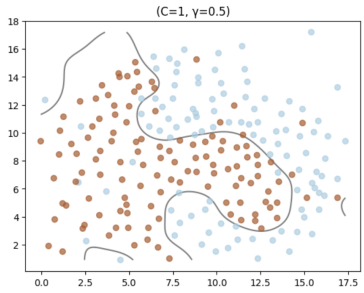

Tabla 6: Good model

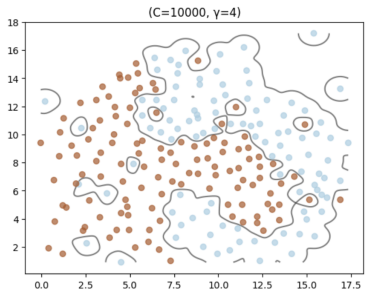

Tabla 7: Over-fitting model

5. Console interface

5.1. Question 8

Dividing the dataset into training and test sets is a fundamental strategy to evaluate the model's performance impartially. By splitting the data into training and test sets, we ensure that the model hasn't seen the test data during training, providing a more realistic assessment of its ability to generalize to new data.

Using the train_test_split method with a 75 % proportion for the training set and 25 % for the test set stratifies the data to maintain the same class proportion in both sets. This is crucial to prevent class imbalance that could bias the model evaluation.

We perform multiple repetitions of the training and evaluation process with different random seeds for data splitting. This allows us to observe how the model's performance varies across different data partitions. The variation in model performance due to different seeds gives us insight into the model's robustness and how it may behave on different datasets.

After repeating the process several times, we observe that the best-performing hyperparameter configuration was C=10 and gamma=1. This means that the SVM model with an RBF kernel and these hyperparameters achieved the highest accuracy in classifying the test set among all the combinations tested.

In summary, by splitting the data into training and test sets, performing multiple repetitions with different seeds, and optimizing the hyperparameters, we obtain a comprehensive evaluation of the SVM model's performance on the given dataset.

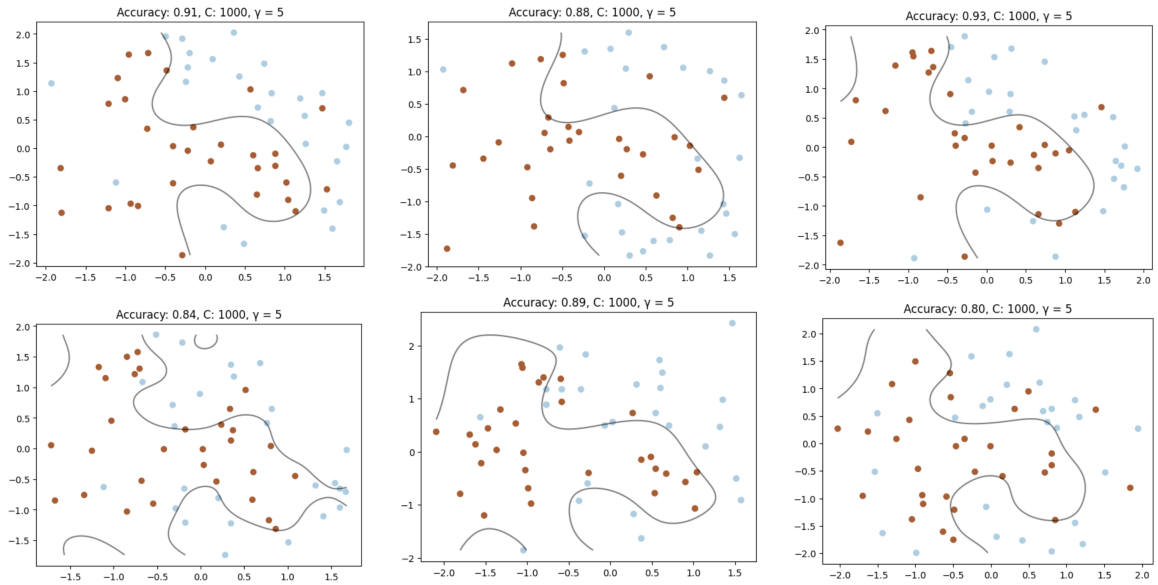

Tabla 8: Question 8: Adjusting by hand

5.2. Question 9

We conducted the previous training using the GridSearchCV object, enabling us to execute a nested K-Fold cross-validation process and identify the optimal values for C and gamma parameters.

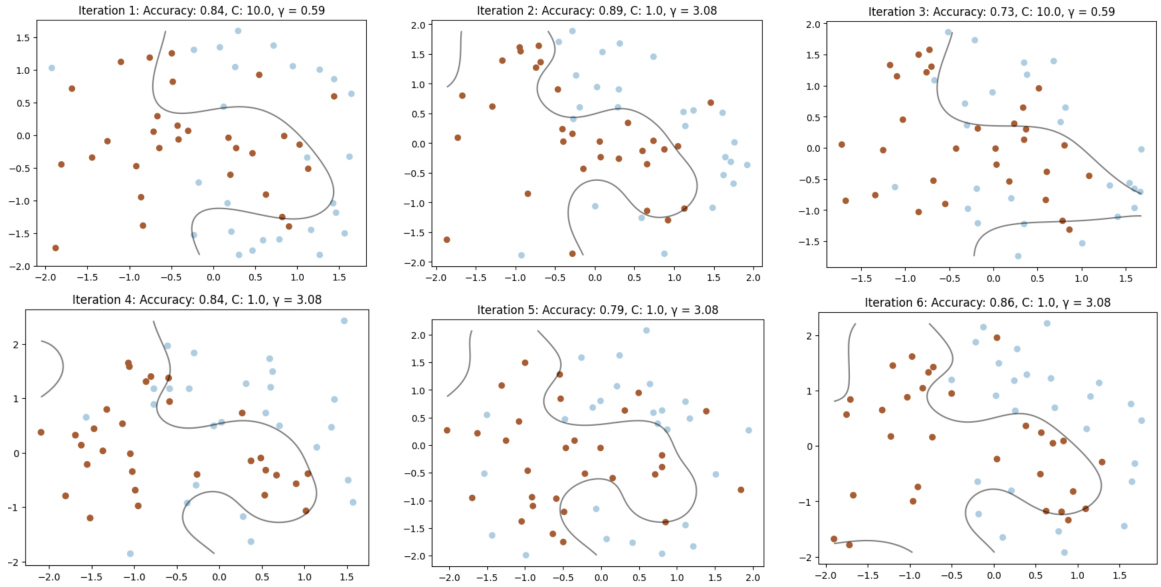

Tabla 9: Question 8: Adjusting by hand

The obtained results appear to be as promising as before, indicating that the initially selected values for C and gamma were indeed optimal for addressing this specific problem. This finding underscores the effectiveness of the hyperparameter search strategy we employed initially.

Ultimately, it is evident that cross-validation provides a more convenient and reliable way to determine optimal values for different parameters. This approach largely eliminates the trial-and-error process associated with manual hyperparameter selection, resulting in a more precise and efficient model fit

5.3. Question 10

Adjusting the parameters manually and evaluating the model's accuracy on the test set, as done in question 8, has several drawbacks:

1. **Subjectivity**: Manual parameter tuning relies heavily on the researcher's intuition and experience. It may lead to biased selections based on prior knowledge or assumptions about the dataset.

2. **Time-consuming**: Manually adjusting parameters involves repeated trial and error, which can be time-consuming and inefficient, especially when dealing with large datasets or complex models.

3. **Limited exploration**: Manual tuning often explores only a limited range of parameter values based on the researcher's discretion, potentially overlooking better-performing combinations that fall outside of this range.

4. **Overfitting**: The risk of overfitting is higher when tuning parameters manually. Without rigorous validation, there is a possibility of selecting parameter values that optimize performance on the test set but fail to generalize well to unseen data.

5. **Lack of reproducibility**: Manual parameter tuning may lack reproducibility since the chosen parameters are based on the researcher's judgment, making it difficult to replicate the exact procedure and achieve consistent results.

6. **Difficulty in optimization**: It can be challenging to find the optimal combination of parameters manually, especially in high-dimensional parameter spaces. This approach may overlook subtle interactions between parameters that can significantly impact model performance.

5.4. Question 11

In this section, we delve into implementing K-fold nested cross-validation manually, without relying on the GridSearchCV utility from Scikit-learn. Despite resembling the methodology employed in Question 8, this manual approach allows for a more granular understanding of how parameters are optimized through cross-validation.

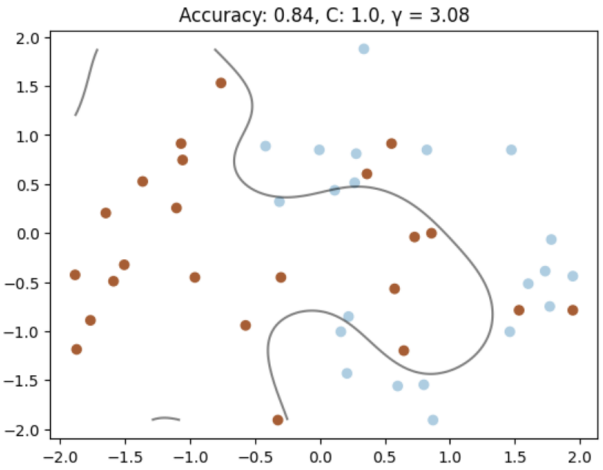

Figura 4: Question 11 Results

The conclusion that the result is the same indicates that we have accurately simulated the operation of GridSearchCV. This suggests that our manual implementation has been effective and reliable in finding the best values for the C and gamma parameters in the Support Vector Machine (SVM) model. By obtaining consistent results, we can have confidence in the validity of our approach and its ability to identify optimal hyperparameters for the given dataset.

Furthermore, this consistency reinforces our understanding of the hyperparameter search process and provides us with greater confidence in our ability to adjust machine learning models effectively and accurately. In summary, the comparison between the manual implementation and the use of GridSearchCV has demonstrated the validity and reliability of our approach.

6. COMPAS dataset

6.1. Question 12

We followed the instructions to train the SVM model on the provided dataset without splitting it into training and test sets and without standardizing the features. Instead, we used the entire dataset for training. We utilized a range of values for the hyperparameters C and γ, ranging from 10^{-3} to 10^{3}, and applied grid search to find the optimal values.

After training the model, we obtained the following results:

- Best parameters: $\{'C' : 1000{,}0, 'gamma' : 0{,}01\}$
- Accuracy on the test set: 0,6679

Comparing this accuracy (66.80 %) with the previous lab assignment, where the best accuracy achieved was 67.45 %, we observe a minimal change. The difference in performance is insignificant, suggesting that the adjustment of hyperparameters did significantly impact the model's generalization ability.

Overall, the optimal values obtained for the parameters were $C = 1000{,}0$ and $\gamma = 0{,}01$, indicating that the model performed best with these parameter settings on this dataset.

6.2. Question 13

Al examining the implemented script, we can observe that the value of K is specified in the cv parameter of the GridSearchCV function, while the parameters C and gamma are specified in the param_grid parameter of the same function.

We experimented with different values of K and compared the computational times and the test results.

Iteración	**Valor de k**	**CCR (%)**	**Segundos**
1	3	66.79	147.79
2	5	66.79	338.42
3	10	66.79	844.57

Tabla 10: Comparative of the k values

As we can see in the results, the variation of K does not influence the result obtained, but it does influence the execution time, so we can conclude that we are not interested in increasing the K in the relationship between success and time.

6.3. Question 14

Use these methods to calculate the performance of the model for each class and by groups. This is, for the best model, calculate confusion matrices for train and test, and display some given performance metrics for test (for example, accuracy and FPR, but you can use others that you consider valid). Analyse the expected behaviour according to this information and briefly discuss the limitations of the metrics when representing the confusion matrix in this problem.

We are going to perform the initial pre-processing by swapping the class values, assigning the positive class as 1 and vice versa

6.4. Question 15

We can adapt this idea to our problem by calculating the GM for the groups (blacks and whites), so that we can guide the optimization by the GM of the worst performance group or the average of GM for all the groups. Try these two strategies and compare the results with the standard approach (overall accuracy).

7. Spam Dataset

7.1. Question 16

A linear SVM model with the values $C = 10^{-2}$, $C = 10^{-1}$, $C = 10^{0}$ and $C = 10^{1}$ must be trained. For this, use a script similar to the one used for question 9. Compare the results and establish the best configuration.

- Best parameters: 'C': 0.1
- Accuracy on the test set: 97.4 %

La optimal configuration achieved an impressive test accuracy of 97.4 %, with a corresponding C value of 0.1. These results are remarkably high, making it challenging to further improve upon them at this stage.

7.2. Question 17

For the best configuration, it builds the confusion matrix and establishes the misclassification emails. Check the input variables for the emails incorrectly classi- fied and find out the reason behind it. Note that for each pattern, when xi is equal to 1 it means that the i-th word in the vocabulary appears, at least once, in the email.

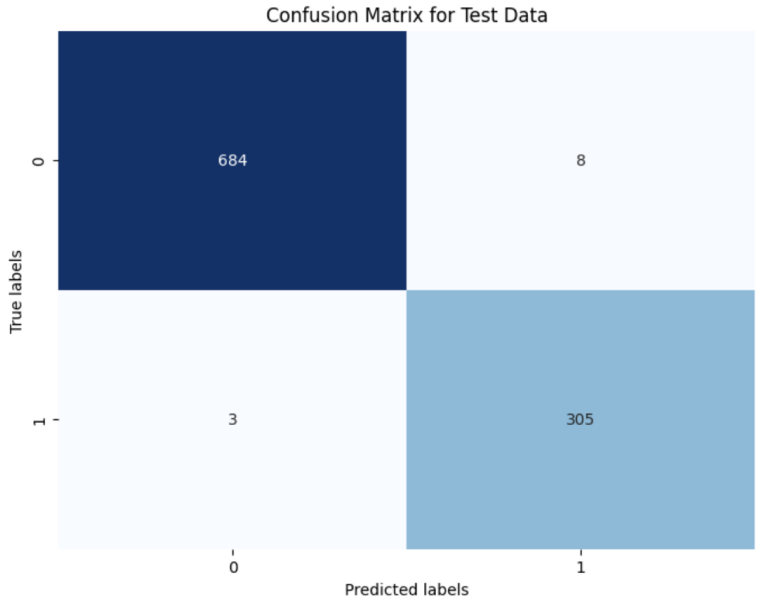

Figura 5: Caption

In our analysis, we observed that a total of 11 emails were wrongly predicted by the model. These emails are identified by their index numbers: [9, 21, 58, 73, 147, 328, 407, 526, 560, 842, 881].

Total of matched words detected in email number 9: 13

- Word number 530: email
- Word number 665: food
- Word number 734: ground
- Word number 798: http
- Word number 839: info
- Word number 875: ireland
- Word number 959: link
- Word number 960: linux
- Word number 994: main
- Word number 1609: subscrib
- Word number 1743: ultim
- Word number 1772: useless
- Word number 1899: zip

Prediction: 0 (not spam). **Expected value:** 1 (spam)

In this case, according to my judgment, the email has been misclassified since it does not contain typical spam words such as "prize,credit card,money,sexual offers,.^{et}c. Given the bag of words, it becomes challenging to detect if it is spam.

Total of matched words detected in email number 21: 46

- Word number 70: amp
- Word number 73: analyst
- Word number 123: assur
- Word number 161: bd
- Word number 226: button
- Word number 233: california
- Word number 237: campaign
- Word number 351: consum
- Word number 529: els
- Word number 530: email
- Word number 705: gcc
- Word number 765: heart
- Word number 798: http
- Word number 809: ie
- Word number 818: imagin
- Word number 876: irish
- Word number 960: linux
- Word number 991: mai
- Word number 1031: merchant
- Word number 1092: necessari
- Word number 1094: neg
- Word number 1112: north
- Word number 1119: null
- Word number 1161: octob
- Word number 1164: offer
- Word number 1190: otherwis
- Word number 1191: our
- Word number 1402: repositori
- Word number 1411: respect
- Word number 1416: retail
- Word number 1500: sexual
- Word number 1560: spam
- Word number 1571: spend
- Word number 1583: star
- Word number 1637: taken
- Word number 1665: that
- Word number 1672: theori
- Word number 1675: these
- Word number 1678: think
- Word number 1698: tm
- Word number 1758: unsubscrib
- Word number 1829: weight
- Word number 1836: whatev
- Word number 1851: wife
- Word number 1892: york
- Word number 1894: young

Prediction: 1 (spam). **Expected value:** 0 (not spam)

In this case, the email has been predicted as spam, likely due to the presence of words like "sexual.^{a}nd "wife."However, it appears to be an unethical conversation between computer professionals. It can be inferred that it is a conversation between computer professionals based on the keywords that appear, such as "gcc,linux,.^{et}c.

7.3. Question 18

Compare the results obtained with the results achieved using an RBF network. To do this, use the programme developed in the previous practice. Use only one seed (the one with the best results).

The search for the best configuration and the final execution will be conducted locally, and the results will be performed in the Colab notebook.

The best hyperparameters found are:

- Best seed: 1
- Best learning rate: 1e-09
- Best ratio: 0.05
- Best regularization: L1
- Best test CCR: 95.2

The results obtained using SVM are clearly superior to those obtained with RBF. Although it is true that the accuracy in tests is slightly lower, with barely any difference, the execution time is clearly lower in SVM compared to RBF. Therefore, SVM is undoubtedly the clear winner.

- RBF (tests): 95.2 %
- SVM (tests): 97.4 %

7.4. Question 19

Train a non-linear SVM and compare the results obtained.

Hyperparameter	**Value**
C	10.0000
Gamma	0.0100
Accuracy	
Training Set	0.9765
Test Set	0.9900

Tabla 11: Grid Search Results

In this case, the results are considerably good, with a similar result in test accuracy to around 99 %, along with the optimization time required by GridSearchCV.

Model	**Accuracy**
SVM Linear	99.0 %
SVM No-Linear	97.4 %
RBF Network	95.2 %

Tabla 12: Results of SPAM Dataset

From these results, we can conclude that SVM Linear outperformed both SVM Non-Linear and RBF Network in terms of accuracy. SVM Linear had the highest accuracy, followed by SVM Non-Linear and then RBF Network. Therefore, for this particular dataset and task, using a linear SVM model seems to be the most effective approach in terms of achieving the highest accuracy.

8. Bibliografía

- Curso Introducción a los Modelos Computacionales
- scratch-multilayer-perceptron
- davidalbertonogueira/MLP: Simple multilayer perceptron
- mLEARn: An Implementation of Multi-layer Perceptron in C++