Driver fatigue detection with supervised learning

Andre Guillermo Raymundo Rodriguez
Universidad Panamericana
Universidad Panamericana
Aguascalientes, México
rodriguez.andre.esc9@gmail.com

Gerardo Macias Romo
Universidad Panamericana
Universidad Panamericana
Aguascalientes, México
Geras4215@gmail.com

***Abstract*— One of the most prevalent causes of vehicular accidents is driver fatigue, and early detection of this condition could significantly reduce the incidence of such accidents. This study aims to analyze a dataset comprising images classified as "drowsy" and "not drowsy" to predict driver fatigue using two supervised learning models: Random Forest Classifier (RFC) and Support Vector Machine Classifier (SVMC). The methodology employs image processing techniques, including Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) calculations, and a pre-trained model identifying 68 facial landmarks to extract features like eyes, mouth, nose, and chin, which are used to train both RFC and SVMC models. This research also examines the advantages and limitations of each approach, providing insights into their efficiency and potential to enhance driver fatigue detection. The findings substantiate the superiority of RFC over SVMC in terms of predictive accuracy and performance.**

Keywords—driver fatigue, fatigue detection, supervised learning, RFC, SVMC

I. Introduction

Driver fatigue is a leading cause of vehicular accidents, necessitating robust detection systems to alert drivers and mitigate risks, and this study leverages a dataset [1] of images labeled as "drowsy" and "not drowsy" to develop a model for fatigue detection using two supervised machine learning techniques: Random Forest Classifier (RFC [10]) and Support Vector Machine Classifier (SVMC [11]). The primary objectives are to analyze and process diverse images of fatigued and non-fatigued individuals, train a supervised model to detect fatigue based on facial features, and evaluate its performance in terms of accuracy, runtime, and other metrics, while comparing the strengths and limitations of RFC [10] and SVMC [11].

Existing research highlights the efficacy of vision-based approaches, such as the use of S3FD and FAN algorithms with RFC to analyze facial features like yawns and eye state for real-time fatigue detection [3], and SVM-based systems achieving up to 97.93% accuracy by processing features like PERCLOS, yawn count, and head detection using OpenCV and Dlib [4]. Deep learning approaches, including CNN-based models like AlexNet and ResNet, have also been explored to classify drowsiness behaviors from RGB videos, achieving accuracies around 85% [5], while surveys categorize detection methods into driver state (e.g., PERCLOS, mouth shape), driver performance (e.g., lane tracking), and hybrid approaches, noting advancements in controlled settings but emphasizing the need for robust real-world applications [6, 7]. This study builds on these insights, focusing on feature extraction via Eye Aspect Ratio (EAR), Mouth Aspect Ratio (MAR), and 68 facial landmarks to enhance the accuracy and efficiency of driver fatigue detection.

A key focus of this study is to explain why RFC [10] outperforms SVMC [11] in analyzing and detecting fatigue in images and to demonstrate how the use of a meta-window enhances early detection compared to a standard sliding window approach. Specifically, we address the following research questions:

(1) What are the advantages and disadvantages of RFC [10] versus SVMC [11] in terms of runtime and accuracy?

(2) What are the advantages and disadvantages of the incorporation of a meta-window?

In the subsequent sections of this report, we provide a detailed explanation of the methodology, encompassing the different approaches, evaluation metrics, implementation details—including model performance and training processes—and the results obtained. Finally, we discuss the implications of our findings and identify potential areas for future research and improvement in driver fatigue detection systems.

II. Methodology

The methodology of this study encompasses two distinct approaches: Random Forest Classifier (RFC [10]) and Support Vector Machine Classifier (SVMC [11]). Both methods incorporate image processing techniques applied in two forms: a single sliding window and a meta-window framework.

A. *Image Description – Feature Extraction*

This section outlines the process of extracting features from the dataset [1] images to enable fatigue detection. The methodology leverages facial landmark detection and image processing techniques to compute key metrics, which are aggregated using a single sliding window and a meta-window approach for comprehensive analysis.

1) *Facial Landmark Detection*

To extract facial features, a pre-trained face detector and landmark predictor from the DLib [9] library are utilized. The detector, implemented as ‘dlib.get_frontal_face_detector()’, identifies faces within each grayscale image by returning bounding box coordinates for detected faces. Subsequently, the predictor, loaded as ‘dlib.shape_predictor("shape_predictor_68_face_landmarks.d at")’, maps 68 specific facial landmarks on each detected face. These landmarks correspond to key facial features, including the eyes (indices 36–41 for the left eye, 42–47 for the right eye), mouth (indices 48–67), nose, and jawline. The landmark coordinates are used to compute the Eye Aspect Ratio (EAR), Mouth Aspect Ratio (MAR), and head angle, providing a foundation for fatigue detection.

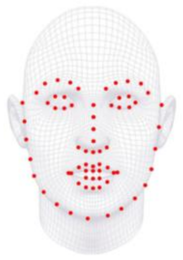


Fig. 1. Facial landmarks

2) Feature Extraction with Single Sliding Window

Features are extracted from each image and aggregated over a single sliding window of 20 frames to capture temporal patterns. The following features are computed for each window:

- **Eye Aspect Ratio (EAR):** EAR measures eye openness by calculating the ratio of the vertical to horizontal distances between eye landmarks.

$$\text{EAR} = \frac{||P_2 - P_6|| + ||P_3 - P_5||}{2 \cdot ||P_1 - P_4||}$$

Eq. 1. EAR formula

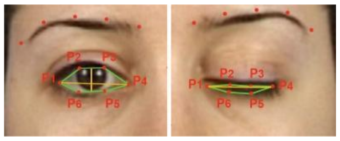


Fig. 2. Eye landmarks

Where P_1 to P_6 are the six landmarks of an eye. The EAR for both eyes is averaged to obtain a single value per frame. Over the 20-frame window, statistical measures are derived: mean (ear_mean), standard deviation (ear_std), and minimum (ear_min).

- **Mouth Aspect Ratio (MAR):** MAR quantifies mouth openness, indicating potential yawning.

$$\text{MAR} = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{2\,\|p_1 - p_5\|}$$

Eq. 2. MAR formula

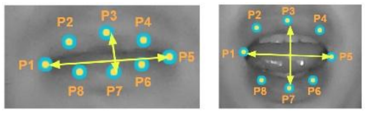


Fig. 3. Mouth landmarks

Where P_1 to P_8 are mouth landmarks. For the sliding window, the mean (mar_mean), standard deviation (mar_std), and maximum (mar_max) are calculated.

- **Head Angle:** The head angle estimates head tilt, a potential fatigue indicator. Using six facial landmarks (nose tip, chin, eye corners, and mouth corners), a 3D head pose is computed via the solvePnP function in OpenCV [8]. A 3D model with predefined points is aligned with the 2D image landmarks, yielding a rotation vector converted to an angle in degrees. The mean head angle (head_angle_mean) over the 20-frame window is used.
- **Blink Detection:** Blinks are detected by monitoring EAR values over the window. A blink is identified when the EAR drops below a threshold of 0.2 (indicating eye closure) and rises above it. Additional metrics include number of blinks in the window (blink_freq), number of frames where EAR $\leqslant$ 0.2 (total_closed_frames) and longest consecutive sequence of frames with EAR $\leqslant$ 0.2 (max_closed_duration).

3) Feature Extraction with Meta-Window

To capture longer-term fatigue patterns, a meta-window approach is applied, grouping five consecutive single sliding windows (equivalent to 100 frames). Two additional features are computed:

- **Percentage of Closed Frames (percent_closed_in_meta):** This measures the proportion of frames with eyes closed across the meta-window.

$$\text{percent\_closed\_in\_meta} = \left(\frac{\sum \text{total\_closed\_frames}}{\text{window\_size} \times \text{meta\_window\_size}} \right) \times 100$$

Eq. 3. Percentage of Closed Frames formula

- **Mostly Closed Windows (mostly_closed_windows):** This counts the number of single windows within the meta-window where more than 50% of frames have eyes closed.

These features (ear_mean, ear_std, ear_min, mar_mean, mar_std, mar_max, head_angle_mean, blink_freq) provide a detailed representation of fatigue indicators within a single 20-frame window. For the meta-window, four additional features are incorporated: total_closed_frames, max_closed_duration, percent_closed_in_meta, and mostly_closed_windows, enhancing the analysis of prolonged fatigue patterns.

B. *Image classification - Model training*

This section describes the process of training the Random Forest Classifier (RFC [10]) and Support Vector Machine Classifier (SVMC [11]) models for fatigue detection, using the features extracted from the dataset [1] as outlined in the previous section. The training pipeline incorporates data preprocessing, class balancing, dataset [1] splitting, and hyperparameter optimization to ensure robust model performance.

1) Data Preprocessing

The feature set is first processed to ensure consistency and compatibility with the machine learning models. To normalize the feature values and mitigate the impact of varying scales, the MinMaxScaler from scikit-learn is applied. This scaler transforms each feature to a range of [0, 1].

2) Class Balancing with SMOTE [13]

The Synthetic Minority Over-sampling Technique (SMOTE [13]) is employed to balance the classes. SMOTE [13] generates synthetic samples for the minority class by interpolating between existing minority samples and their nearest neighbors, ensuring an equal number of instances for both classes. This step is critical to prevent the models from being biased toward the majority class and to improve their ability to generalize across both fatigue states. The balanced dataset [1] is then used for subsequent training steps.

3) Dataset [1] Splitting

To evaluate the performance of the model on unseen data, the balanced dataset [1] is split into training and testing sets using the train_test_split function from scikit-learn. The split allocates 85% of the data to the training set and 15% to the testing set.

4) Model Training and Hyperparameter Optimization

Both RFC [10] and SVMC [11] models are trained using the training set, with hyperparameter optimization performed via grid search to maximize performance. A grid search (GridSearchCV, 5-fold cross-validation) is conducted. The best model is selected based on the F1-score, balancing precision and recall to account for class importance in fatigue detection.

After training, the models are evaluated on the test set using classification metrics such as accuracy, precision, recall, and F1-score, along with a confusion matrix to analyze true positives, true negatives, false positives, and false negatives.

III. Experimental setup

A. Dataset [1]

This study utilizes a dataset [1] divided into two distinct categories: "drowsy" and "not drowsy." The "drowsy" category comprises 35,935 images, while the "not drowsy" category contains 30,491 images, yielding a total of 66,426 images. Each image is labeled with specific identifiers reflecting the conditions captured, as follows:

- **Drowsy category:**
  1) glasses_sleepyCombination
  2) glasses_slowBlinkWithNodding
  3) glasses_yawning
  4) noglasses_sleepyCombination
  5) noglasses_slowBlinkWithNodding
  6) noglasses_yawning
- **Not drowsy category:**
  1) glasses_nonsleepyCombination
  2) glasses_sleepyCombination
  3) glasses_slowBlinkWithNodding
  4) glasses_yawning
  5) noglasses_nonsleepyCombination
  6) noglasses_sleepyCombination
  7) noglasses_slowBlinkWithNodding
  8) noglasses_yawning

In Figure 4. we can observe some of the samples of the dataset [1].

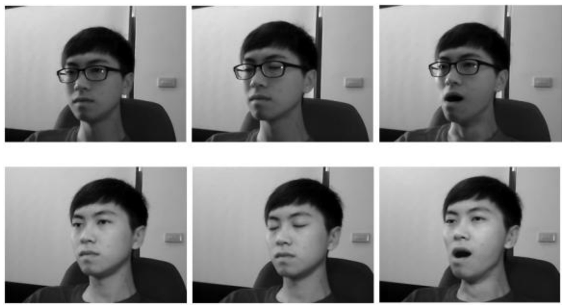


Fig. 4. Dataset samples

Images sharing the same label belong to a sequence extracted from a single video, with these video sequences repeated across different individuals. This structure ensures a diverse representation of facial behaviors under various fatigue and non-fatigue states, facilitating robust model training and evaluation.

B. Validation Metrics

For this driver fatigue detection study, we employed several validation metrics to assess the performance and reliability of our models:

1) Accuracy

This metric measures the overall proportion of correct predictions (both true positives and true negatives) relative to the total number of instances in the test set.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Eq. 4. Accuracy formula

2) Precision

Evaluates the proportion of true positive predictions among all instances predicted as positive (drowsy), reflecting the model's ability to avoid false positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Eq. 5. Precision formula

3) *Recall*

Recall measures the proportion of actual positive instances (drowsy) correctly identified by the model, emphasizing its ability to detect fatigue effectively.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Eq. 6. Recall formula

4) *F1-Score*

The F1-score provides a harmonic mean of precision and recall, offering a balanced measure of the performance of a model when dealing with imbalanced classes or when both false positives and false negatives are significant.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Eq. 7. F1-Score formula

5) *Confusion Matrix*

A confusion matrix was generated to provide a detailed breakdown of the predictions of the model, displaying TP, TN, FP, and FN counts. This matrix identifying the balance between correctly classified instances and misclassifications.

C. *Implementation details*

1) *Python:* This programming language was selected owing to its inherent flexibility and the straightforward nature of its logical structure, as well as its extensive collection of libraries and packages that enhance development capabilities.

2) *Libraries:*

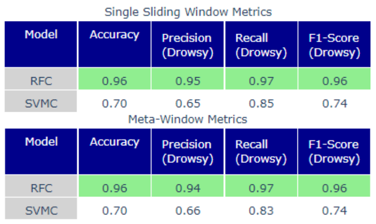
- cv2 (OpenCV) [8]: Utilized for image processing tasks, including reading video frames and converting images to grayscale.
- Dlib [9]: Employed for facial detection and landmark prediction, leveraging 'get_frontal_face_detector()' to identify faces and 'shape_predictor()' with the pre-trained "shape_predictor_68_face_landmarks.dat" model to map 68 facial landmarks for feature extraction.
- sklearn.ensemble (RandomForestClassifier) [10]: Provided the Random Forest Classifier (RFC [10]) implementation.
- sklearn.svm (SVC) [11]: Supplied the Support Vector Machine Classifier (SVMC [11]) with an RBF kernel.
- sklearn.model_selection [12]: Facilitated dataset [1] splitting into training and testing sets with 'train_test_split' and performed exhaustive hyperparameter searches with 'GridSearchCV' using 5-fold cross-validation to optimize model performance.
- imblearn.over_sampling (SMOTE) [13]: Applied to balance the dataset [1] by generating synthetic samples for the minority class.
- joblib [14]: Used to save and load trained models (RFC [10], SVMC [11]) and the MinMaxScaler object, enabling their reuse in real-time applications without retraining.

IV. RESULTS

This study evaluates the performance of two supervised learning models—Random Forest Classifier (RFC [10]) and Support Vector Machine Classifier (SVMC [11])—trained using feature extraction from two approaches: single sliding window and meta-window. In total, four training scenarios were conducted: RFC [10] and SVMC [11] with single sliding window features, and RFC [10] and SVMC [11] with meta-window features. The following sections compare the models and feature extraction methods, analyzing their effectiveness in detecting driver fatigue based on the provided metrics.

1) *Comparison Between RFC [10] and SVMC [11]*

Both RFC [10] and SVMC [11] were trained and evaluated on the test set, using the single sliding window and meta-window features. The performance metrics for each model are presented in the tables below, followed by an analysis of each metric and an explanation of RFC's [10] superiority over SVMC [11].



Single Sliding Window Metrics

Model	Accuracy	Precision (Drowsy)	Recall (Drowsy)	F1-Score (Drowsy)
RFC	0.96	0.95	0.97	0.96
SVMC	0.70	0.65	0.85	0.74

Meta-Window Metrics

Model	Accuracy	Precision (Drowsy)	Recall (Drowsy)	F1-Score (Drowsy)
RFC	0.96	0.94	0.97	0.96
SVMC	0.70	0.66	0.83	0.74

Table 1. Comparison of model's performance

- Accuracy: RFC [10] consistently achieves higher accuracy than SVMC [11] in both scenarios (0.96 vs. 0.70 for single sliding window, 0.96 vs. 0.70 for meta-window).
- Precision (Drowsy): RFC's [10] precision (0.95 and 0.94) surpasses SVMC's [11] (0.65 and 0.66), meaning RFC [10] is better at avoiding false positives
- Recall (Drowsy): RFC [10] exhibits superior recall (0.97 and 0.97) compared to SVMC [11] (0.85 and 0.83). This metric is critical in fatigue detection, as higher recall indicates fewer false negatives (missed drowsy instances), enhancing safety by ensuring fatigued drivers are more likely to be detected.
- F1-Score (Drowsy): The F1-score, which balances precision and recall, is higher for RFC [10] (0.96 and 0.96) than for SVMC [11] (0.74 and 0.74). This reflects RFC's [10] better overall performance in

handling the trade-off between false positives and false negatives.

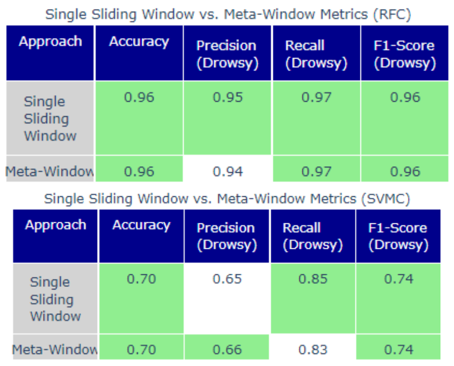
The superiority of RFC [10] over SVMC [11] can be attributed to several factors:

- RFC [10], being an ensemble of decision trees, effectively captures non-linear relationships and interactions between features (e.g., EAR, MAR, and head angle), which are prevalent in facial fatigue indicators. SVMC [11], even with an RBF kernel, may struggle to model these complex patterns as effectively.
- RFC [10] is less sensitive to noise and outliers in the dataset [1], as it aggregates predictions across multiple trees. In contrast, SVMC [11] can be heavily influenced by outliers
- RFC [10] inherently provides feature importance scores, allowing better insight into which features (e.g., blink_freq, percent_closed_in_meta) contribute most to fatigue detection, aiding interpretability and potential model improvement.

SVMC [11], while effective, is more computationally intensive and less adaptable to the dataset's [1] complexity, resulting in lower performance across all metrics.

2) Comparison Between Single Sliding Window and Meta-Window

The performance of RFC [10] and SVMC [11] was also compared across the two feature extraction methods: single sliding window (20 frames) and meta-window (5 windows of 20 frames each, totaling 100 frames). The metrics for each approach are summarized below.



Single Sliding Window vs. Meta-Window Metrics (RFC)

Approach	Accuracy	Precision (Drowsy)	Recall (Drowsy)	F1-Score (Drowsy)
Single Sliding Window	0.96	0.95	0.97	0.96
Meta-Window	0.96	0.94	0.97	0.96

Single Sliding Window vs. Meta-Window Metrics (SVMC)

Approach	Accuracy	Precision (Drowsy)	Recall (Drowsy)	F1-Score (Drowsy)
Single Sliding Window	0.70	0.65	0.85	0.74
Meta-Window	0.70	0.66	0.83	0.74

Table 2. Comparison between approaches

As we can observe, the improvement is relatively modest, likely because the dataset [1] consists of static images extracted from videos rather than continuous video streams. Both approaches effectively detect immediate fatigue indicators, but they may not fully capture cumulative fatigue over longer periods. Analyzing more frames to detect cumulative fatigue (e.g., over several minutes) would be computationally expensive in a static image dataset. In such a scenario, the meta-window approach would likely show a more significant advantage, as it is designed to aggregate temporal patterns across a broader timeframe, enhancing the detection of sustained fatigue states.

V. Discussion

A. *Answer of the research questions*

(1) What are the advantages and disadvantages of RFC [10] versus SVMC [11] in terms of runtime and accuracy? The Random Forest Classifier (RFC [10]) offers significant advantages over the Support Vector Machine Classifier (SVMC [11]) in terms of runtime and accuracy. RFC [10] trains substantially faster due to its parallelizable ensemble of decision trees, which efficiently handle independent tree construction, reducing computational overhead. This structure also enables RFC [10] to capture non-linear relationships and complex feature interactions inherent in fatigue detection, leading to higher accuracy. However, RFC [10] may overfit on noisy data if hyperparameters like tree depth are not carefully tuned, though this was mitigated using GridSearchCV. In contrast, SVMC [11], with its margin maximization approach via an RBF kernel, generalizes well in high-dimensional spaces with clear class boundaries but is hindered by slower training due to the quadratic optimization problem and lower accuracy due to its sensitivity to noise and outliers, making it less effective for the complex, noisy data typical in this study.

(2) What are the advantages and disadvantages of the incorporation of a meta-window? The incorporation of a meta-window offers significant advantages for fatigue detection by aggregating features over an extended sequence of frames, enabling the capture of prolonged fatigue patterns such as micro-sleeps that might be missed by shorter analysis windows. This approach enhances the ability to detect gradual fatigue onset by providing valuable temporal context, which is crucial for improving the reliability of fatigue alerts in safety-critical applications. However, it also presents disadvantages, including increased computational complexity and higher memory demands due to the need to maintain and process a history of multiple windows, potentially impacting real-time performance. Additionally, its benefits may be limited in datasets with static or short-duration data, where the detection of cumulative fatigue is less pronounced, suggesting that its effectiveness could be more fully realized with longer, continuous data streams.

B. *Limitations of the work*

- Dataset [1]: The dataset [1] utilized in this study consisted of short-duration data, which limited the effectiveness of the meta-window approach. The meta-window, designed to capture prolonged fatigue patterns over an extended sequence of frames, was hindered by the lack of sufficient temporal data, reducing its ability to detect cumulative fatigue effectively.
- Central Processing Unit: The Central Processing Unit (CPU) posed a significant constraint, impacting the meta-window approach and overall computational performance. The available processing power proved inadequate for handling the complex and resource-

intensive tasks required, resulting in extended processing times and delays in generating results, which compromised the efficiency of the analysis.

C. Future work

For future research and implementations, we aim to explore the use of a convolutional neural network (CNN) to enhance accuracy metrics and enable more effective implementation of the meta-window approach. The CNN's ability to automatically extract hierarchical features from images could improve fatigue detection performance, while its capacity to process larger temporal sequences may better leverage the meta-window's potential for capturing prolonged fatigue patterns.

VI. Conclusions

This investigation presents a comparative analysis of two supervised learning algorithms, Random Forest Classifier (RFC [10]) and Support Vector Machine Classifier (SVMC [11]), alongside an evaluation of two feature extraction methods, single sliding window and meta-window, demonstrating the superior performance of RFC [10] in detecting driver fatigue. The results underscore RFC's [10] effectiveness in accurately identifying fatigue states, attributed to its robust handling of complex feature interactions. In summary, various computer vision techniques exist for detecting driver fatigue, and combining feature extraction strategies with the implementation of these models offers promising avenues to enhance the performance and timeliness of fatigue detection systems.

References

[1] A. Basva, “nthuddd2.” https://www.kaggle.com/datasets/banudeep/nthuddd2, 2022.

[2] S. Virahonda, “shape predictor 68 face landmarks.dat.” https://www.kaggle.com/datasets/sergiovirahonda/shape-predictor-68-face-landmarksdat, 2021.

[3] B.-T. Dong, H.-Y. Lin, and C.-C. Chang, “Driver fatigue and distracted driving detection using random forest and convolutional neural network,” Applied Sciences, vol. 12, no. 17, 2022.

[4] B. K. Sava¸s and Y. Becerikli, “Real time driver fatigue detection based on svm algorithm,” in 2018 6th International Conference on Control Engineering Information Technology (CEIT), pp. 1–4, 2018.

[5] Dua, M., Shakshi, Singla, R. et al. “Deep CNN models-based ensemble approach to driver drowsiness detection”. Neural Comput & Applic 33, 3155–3168 (2021).

[6] G. Sikander and S. Anwar, “Driver fatigue detection systems: A review,” IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 6, pp. 2339–2352, 2019.

[7] Q. Wang, J. Yang, M. Ren, and Y. Zheng, “Driver fatigue detection: A survey,” in 2006 6th World Congress on Intelligent Control and Automation, vol. 2, pp. 8587–8591, 2006.

[8] Bradski, G., & Kaehler, A. (2025). OpenCV: Open Source Computer Vision Library (Versión 4.x) [Software]. Retrieved from https://opencv.org/

[9] King, D. E. (2025). Dlib: A Machine Learning Toolkit (Versión 19.x) [Software]. Retrieved from http://dlib.net/

[10] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[11] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[12] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. Retrieved from https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection

[13] Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. Journal of Machine Learning Research, 18(17), 1-5. Retrieved from https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

[14] Joblib Development Team. (2025). Joblib: Running Python Functions as Pipeline Jobs (Versión 1.x) [Software]. Retrieved from https://joblib.readthedocs.io/