Classification of streaming platform images using local binary patterns and machine learning algorithms

J. Álvarez-Borrego[a], E. Guerra-Rosas[b] and L. F. López-Ávila[a]

[a]*Centro de Investigación Científica y de Educación Superior de Ensenada, Carretera Ensenada-Tijuana No. 3918, Zona Playitas, Ensenada, B.C. 22860 México.*
[b]*Facultad de Ingeniería, Arquitectura y Diseño, Universidad Autónoma de Baja California, Km. 103 carretera Tijuana-Ensenada, Ensenada, B.C. 22860 México.*

Received 5 February 2025; accepted 26 April 2025

This research has developed an effective multi-class classification model for images from streaming platforms. Texture features were extracted from the images and used with machine learning algorithms. Two datasets were employed: 'mosaics' comprising 153,488 images across 14 classes and 'descriptors' containing 33,471 images across 11 classes. All images had a resolution of 1280×720 pixels. The local binary pattern (LBP) technique encoded the local texture structure into 59-element feature vectors for each image. Ten algorithms were trained and evaluated on these vectors for each dataset, including support vector machines (SVM) with linear, polynomial, and Gaussian kernels at various scales, as well as ensemble methods like boosted trees, bagging, discriminant analysis, and k-nearest neighbors in subspaces. Training and validation were done via 30 random splits to mitigate bias. Performance metrics like accuracy, sensitivity, specificity, precision, and F1-score were computed per class. The SVM classifier achieved top mean performance: 0.998952 accuracy, 0.992528 sensitivity, 0.999438 specificity, 0.988132 precision, and 0.990280 F1-score. The results validate the proposed LBP feature extraction and machine learning methodology for effectively classifying images across streaming platforms.

Keywords: Images; algorithms; streaming platform; local binary pattern; machine learning; texture features.

DOI: https://doi.org/10.31349/RevMexFis.71.051303

1. Introduction

In the current digital era, characterized by the proliferation of multimedia platforms offering a vast range of content, such as Netflix, Apple TV, Disney, etc. [1], optimizing the user experience has become a primary objective. The exponential growth of the content offering presents significant challenges in facilitating navigation and the discovery of new titles of interest. In this landscape, multimedia content recommendation and classification systems emerge as indispensable tools to improve accessibility and personalization of the user experience [2]. Research in this field is crucial, as it affects user satisfaction and has substantial commercial implications; by collecting detailed information about their audience's consumption habits and preferences, they are uniquely positioned to offer relevant and personalized content. This capability improves user retention and presents opportunities for effective marketing strategies, such as personalized ads and sponsored content [3].

One of the fundamental tasks in this field is the classification of representative images of the content available on these channels. However, as content volume grows, manual classification becomes impractical and error prone. In response to this challenge, machine learning techniques have emerged as a promising solution. By analyzing and extracting relevant visual features from images, these models can classify each image into its corresponding category, which improves efficiency and reduces errors associated with manual labeling. In this context, this work proposes a comprehensive strategy for classifying images from multimedia platforms. The technique of local binary patterns (LBP) is addressed for feature extraction and recognized for its ability to capture texture information in images efficiently. In addition, a variety of machine learning algorithms will be explored, including support vector machines (SVM) with different types of kernels and ensemble classifiers, such as boosted trees and bagging, among others. The most effective approaches to address such classification are identified through an exhaustive evaluation of representative data sets. To do this, performance metrics such as precision, sensitivity, and F1 measure, among others, are analyzed to determine the viability and effectiveness of each approach.

In addition to promoting user experience optimization, this work seeks to contribute to the vision of how multimedia companies can capitalize on user data to expand their reach and improve their marketing strategies. However, it is crucial to take into account the ethical and social considerations associated with the use of user data and the algorithmic influence on content recommendation.

1.1. Background

Image classification is a fundamental process in various fields of knowledge, from scientific to commercial applications. Extensive work has been done on medical image classification for diagnosis, facial recognition for security, object identification in scenes for autonomous vehicles, quality control, and other applications [4,5]. The main goal of these efforts is to automatically categorize the pixels of an image into defined classes or categories according to the features and patterns present in the data [6].

In recent decades, image classification has experienced significant development, driven by rapid technological advancement in data acquisition and increased computational power available. Initially, relatively simple techniques based on features such as color histograms and statistical texture analysis have evolved into much more sophisticated and robust approaches. Among these advances, the emergence of nonparametric classifiers that do not assume specific statistical distributions of the data, such as neural networks and decision trees, stands out. In addition, techniques have been developed to properly handle mixed pixels using sub-pixel and fuzzy logic approaches [4,7].

Likewise, ways to incorporate additional information beyond spectral values have been explored, such as spatial, textural, shape, and context features, multitemporal data, and data integration from multiple sources and sensors. These approaches have significantly improved the accuracy and robustness of classification models. However, accurate image classification remains a challenge, especially in heterogeneous environments. Aspects such as optimal feature selection, uncertainty management, and accuracy assessment are active research areas [7].

In this context [8], compared various approaches for visual feature extraction and classification methods applied to histopathological image analysis. In their study, they examined five different techniques for feature extraction: gray-level co-occurrence matrix (GLCM), local binary patterns, local binary gray-level co-occurrence matrix (LBGLCM), which is a combination of LBP and GLCM, gray-level run-length matrix (GLRLM), and segmentation-based fractal texture analysis (SFTA). The features extracted using these algorithms were classified using four machine learning methods: support vector machine, k-nearest neighbors (KNN), linear discriminant analysis (LDA), and boosted trees. The experimental findings revealed that the combination of the SFTA algorithm for feature extraction and the boosted trees classifier achieved the highest success in classifying histopathological images in normal and cancerous tissues. This study highlights the importance of carefully selecting the most appropriate feature extraction and classification methods to effectively address the challenge of automatic classification of highly complex medical images.

On the other hand [9], proposed a method that combines the local binary pattern texture feature extraction technique with the support vector machine classification algorithm. They used a dataset of 577 X-ray images of COVID-19 patients and healthy people. They applied sharpening techniques as preprocessing and then extracted texture features, which were used as input to train the classifier. Despite having a relatively small dataset, the results showed exceptional performance in image classification on COVID-19 patients and healthy people.

In this topic of image classification, existing methods usually focus on a single type of dataset, such as textures, clothing patterns, buildings, etc. [10] proposes a common approach for the classification of various types of images with a methodology consisting of three phases: obtaining the region of interest using accelerated robust feature points (SURF), extracting LBP features in that region, and clustering the extracted features using a proposed approach called clustering with fixed centers (CFC) to construct a bag of LBP features. They then evaluated algorithms such as support vector machine, decision trees, KNN, linear method, and random forests, where the first of these obtained results that outperformed existing approaches. The experiments were performed on four datasets: Caltech-101, ORL Face, Bengali signatures, and Hindi signatures.

In Ref. [11] developed a novel texture descriptor based on a local binary pattern called invariant to rotation, scale, and illumination LBP (IRSLBP) for image texture analysis and classification. Their proposal involves quantizing RGB images to a single channel with fewer hues to decrease the computational complexity. They then extract features using the IRSLBP descriptor, which considers circular neighborhoods of pixels and decomposes the difference vectors into sign and magnitude components through local difference sign and magnitude transformation. They implemented a multi-core support vector machine for texture classification, which combines the linear and quadratic kernels. Experimental evaluation on three databases showed that the proposed approach, which integrates the IRSLBP descriptor and multi-core support vector machine, outperformed existing methods regarding sensitivity, specificity, accuracy, predictive values, and error rates.

A comparative study of different Local Binary Pattern-based texture descriptor approaches for the task of image classification was conducted by [12]. Their methodology consisted of extensively evaluating various LBP variants such as uniform LBP, dominant LBP, local quinary patterns (LQP), local ternary patterns (LTP), among others, on six benchmark datasets to cover different domains. Techniques such as variance-based band selection, preserving neighborhood embedding (NPE), and random ensembles of support vector machines were also explored. The results showed that the LTP variant outperformed the standard LBP variant, and that the best strategy was to combine a method based on uniform bands of LQP based on rotation-invariant bins of LTP, by using NPE and random ensembles of SVM, thereby achieving high performance on all the evaluated datasets.

1.2. Justification

Automatic image classification is crucial in today's digital age, where visual content is growing exponentially. Multimedia platforms constantly generate massive images and video frames that need to be efficiently organized and labeled. The ability to accurately classify this visual content into different categories or classes has significant implications. It allows for better organization and search of content, enables personalized recommendations to users, facilitates moderation of inappropriate content, and enables more profound analysis of trends and preferences. While various approaches to image classification exist, machine learning algorithms based on local binary patterns have proven efficient and effective. However, the performance of these algorithms depends heavily on the feature extraction technique and the classification algorithm itself. Therefore, studying and implementing different approaches to determine the most suitable and efficient way to classify images from multimedia platforms is essential.

2. Methodology

To carry out this work, a comprehensive strategy was developed to efficiently process images captured by a device called Core Meter, located on the user side that views the content of different platforms. This device takes visual captures every five seconds, thus generating a continuous flow of data that must be classified in real-time. To achieve this task, developing and implementing a robust and efficient image classifier is required, capable of processing the input data as the device acquires it. It is proposed that a machine learning model be trained using the device's own captures to accurately recognize and classify the image classes of the different platforms.

The real-time classification process is essential to ensure a timely response to the system's specific needs, such as optimizing personalized recommendations and improving marketing strategies based on user behavior. Implementing this classifier would not only improve the accuracy of visual content detection but also allow for the system's scalability and efficiency in environments with large volumes of data. Figure 1 illustrates the process described above and the system's main components.

The approach adopted for image analysis and processing included the LBP feature extraction technique and a wide range of machine learning algorithms, including SVM and ensemble classifiers. The most effective approaches to performing this work were identified through an evaluation of representative mosaic and descriptor datasets. Performance metrics such as accuracy, precision, sensitivity, specificity, and the F1 measure were considered to evaluate the performance of the created models. The concepts used and procedures carried out are detailed in the following sections.

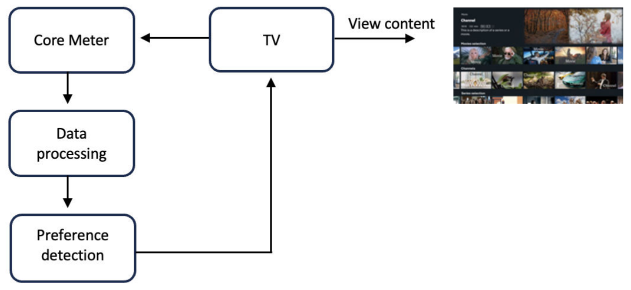

FIGURE 1. Process flow diagram for real-time image classification and generation of personalized recommendations.

2.1. Feature extraction technique

Feature extraction plays a crucial role in image classification through artificial intelligence algorithms. This process involves identifying and selecting the most significant attributes in individual images or sets of images to facilitate their subsequent analysis and processing. The elements extracted can range from aspects such as shape, texture, color, or any other visual feature relevant to the system in question. The importance of this procedure lies in its ability to allow machine learning models to interpret and recognize various visual elements in images, such as objects, individuals, or specific patterns. By isolating these distinctive features, various tasks can be performed, including object identification, face recognition, image classification, and numerous other practical applications.

This project has selected a specific feature extraction methodology called a local binary pattern. This technique is distinguished by its effectiveness in analyzing textures within images and its ability to record detailed information about the image's structure, making it a valuable tool for image classification tasks such as the one addressed in this research.

2.1.1. Local binary pattern

Local binary patterns are an efficient non-parametric technique for texture analysis in grayscale images. This methodology is based on characterizing the local structure of an image by summarizing it in a code. The occurrences of these codes are collected in a histogram, which can be represented as a vector that is useful for classification tasks when combined with machine learning algorithms. One of the main advantages of the LBP operator lies in its attractive properties, such as tolerance to monotonic changes in the grayscale, variations in illumination, and computational simplicity. Additionally, this methodology is highly discriminative, contributing to its wide use and success in texture analysis tasks [13].

This method's original texture analysis approach is based on a 3×3 pixel neighborhood. The procedure involves taking the value of the central pixel and comparing it to the values of its eight neighboring pixels using a thresholding process to determine whether each neighbor is greater or less than the central pixel. These results are converted to binary values (1 if the neighbor is greater than or equal to 0 if it is less). The binary results of these comparisons are weighted by powers of two and summed to obtain the LBP code corresponding to the central pixel. Figure 2 illustrates an example of how this operator is applied. Formally, let g_c be the gray value of the central pixel and g_p the values of its eight neighbors ($p = 0, ..., 7$), then the LBP code for the pixel with coordinates (x, y) is calculated as follows [13,14]:

$$LBP(x,y) = \sum_{p=0}^{7} S(g_p - g_c)2^p, \tag{1}$$

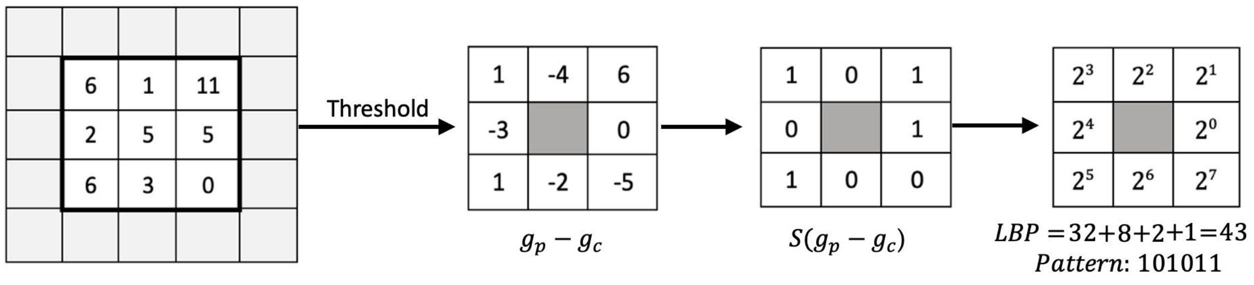

FIGURE 2. Example of calculating the original LBP (Guerra-Rosas *et al.*, 2023).

where p traverses the eight neighboring pixels of the central pixel, $S(z)$ represents the threshold function:

$$S(z) = \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} , \qquad (2)$$

In practical applications, the standard approach to representing an image's local binary patterns consists of constructing an LBP histogram (LBPH). This technique involves collecting the LBP codes computed for each input image pixel and grouping them into a histogram that captures their frequency distribution. In this way, the LBPH histogram acts as a texture descriptor, providing a compact statistical representation of the local pattern information present in the image [13]:

$$LBPH(i) = \sum_{x,y} \delta\{i, LBP(x,y)\}, \qquad (3)$$

where $i = 0, ..., 2^7$ and $\delta(\cdot)$ represents the Kronecker product function. This histogram-based characterization is widely used in texture analysis and classification tasks due to its effectiveness and simplicity.

To adapt more effectively to practical applications, the original local binary patterns approach, shown above, has been generalized in two main ways. One proposed extension for the LBP operator may employ neighborhoods of different sizes. This generalization, denoted as $LBP_{P,R}$, allows us to define a circular neighborhood around the central pixel, where P represents the number of neighbors considered, and R is the radius of the neighborhood. When the sampling points do not precisely coincide with the pixel centers, bilinear interpolation is used to determine the corresponding values. For example, $LBP_{16,2}$ refers to a neighborhood with 16 neighbors located at a radius of 2 pixels from the center. On the other hand, uniform pattern versions of the LBP operator have been developed, introducing a variant in calculating the codes that improve their descriptive capacity and computational efficiency [13,14].

Another relevant extension is the introduction of uniform patterns, denoted as $LBP_{P,R}^{u2}$. An LBP binary code is considered uniform if it exhibits at most two transitions from 0 to 1 or vice versa when represented as a circular string. For example, the patterns 00000000, 00011110, and 10000011 are uniform. In calculating the LBPH histogram, these properties are exploited by assigning an individual band to each uniform pattern, while all non-uniform patterns are grouped into a separate band. Thus, the standard LBPH has 256 bands with eight neighbors, while the uniform pattern version only requires 59 bands. With 16 neighbors, the numbers are 65,536 and 243 bands, respectively. Evidently, the uniform pattern approach manages to significantly reduce the length of the histogram vectors, leading to computational benefits [13,15]. For this work, uniform vectors of 59 elements were used.

2.2. Classification methods for image analysis

Classification methods allow the identification and categorization of patterns and features within images. These models vary in complexity and approach, from traditional statistical techniques to advanced machine learning algorithms. This section examines various classification methodologies used for image analysis, highlighting their fundamental principles. The idea is to provide a detailed overview of the tools used for classifying images from the multimedia platforms selected in this work.

2.2.1. Linear discriminant analysis

Linear discriminant analysis is one of the most widely used methodologies in discriminant analysis, especially in classification and dimensionality reduction tasks. This technique seeks to determine an ideal projection W that transforms a high-dimensional space d_1 into a lower-dimensional space d_2. This transformation allows samples from the same class to be grouped while samples from different classes are separated. The search for this optimal projection can be expressed as a mathematical maximization problem [16-18]:

$$W_{\text{opt}} = \arg_W \max \frac{W^{\intercal} S_B W}{|W^{\intercal} S_W W|}. \qquad (4)$$

Maximizing this expression is equivalent to finding the vector W that projects the data to maximize the separation between the projected class means (numerator) while minimizing the variance within each projected class (denominator). The training set is assumed to contain N classes, then

S_B and S_W are defined as [17,18]:

$$S_B = \sum_{j=1}^{C} N_j (\overline{\mathbf{X}}_j - \overline{\mathbf{X}})(\overline{\mathbf{X}}_j - \overline{\mathbf{X}})^T , \quad (5)$$

$$S_W = \sum_{j=1}^{C} \sum_{i=1}^{N_j} (\mathbf{x}_i^{(j)} - \overline{\mathbf{x}}_j)(\mathbf{x}_i^{(j)} - \overline{\mathbf{x}}_j)^T . \quad (6)$$

In this context, the interclass matrix S_B evaluates the separability between class centers, while the intraclass scattering matrix S_W quantifies the within-class variation of each class in low-dimensional space. $\{\mathbf{x}_i^{(j)}, i = 1, \dots, N_j\}j = 1, \dots, C$ denotes the feature vectors of the training samples. N_j represents the number of samples belonging to class j, $\underline{x_i^j}$, is the vector of the i-th sample belonging to the j-th class, $\overline{X}_j$ is the vector defining the mean of the j-th class, and $\overline{X}$ is the mean computed from all available examples.

2.2.2. *K-nearest neighbor*

The k-nearest neighbor (k-NN) algorithm is a straightforward yet highly effective classification method. Despite its simplicity, it excels in scenarios involving multimodal classes and situations where objects may belong to multiple class labels. The k-NN classifier identifies a set of k nearest objects in the training data that are closest to a given test object and assigns a label based on the majority class in this neighborhood. This approach addresses the challenge of datasets where exact matches between test and training objects are rare. Furthermore, it manages conflicting information from nearby objects effectively [19-21].

Given a training dataset $D = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \dots, (\mathbf{x}_n, c_n)\}$, where x_i represents an object's feature vector and c_i its corresponding class label, and a test object z with an unknown label, the k-NN algorithm follows these steps:

1. **Compute Distances**: Calculate the distance $d(\mathbf{z}, \mathbf{x}_i)$ between the test object $\mathbf{z}$ and each object $\mathbf{x}_i$ in $\mathcal{D}$. Common distance metrics include:

- Euclidean Distance:

$$d(\mathbf{z}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{m} (z_j - x_{i,j})^2} \quad . \quad (7)$$

- Manhattan Distance:

$$d(\mathbf{z}, \mathbf{x}_i) = \sum_{j=1}^{m} |z_j - x_{i,j}| \, . \quad (8)$$

- Select Nearest Neighbors: Identify the k objects with the smallest distances to z , forming the neighborhood $N_k \subseteq D$.

- Determine Class Label: Assign the class c_z to z based on the majority class among the neighbors, using the following rule:

$$c_z = \arg\max_{v \in L} \sum_{y \in N_k} I(v = c_y) \, . \quad (9)$$

where L is the set of possible classes, and $I(\cdot)$ is an indicator function defined as:

$$I(\text{condition}) = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{otherwise} \end{cases} . \quad (10)$$

Key considerations

- **Choice of** k: The value of k significantly impacts the algorithm's performance:
  - A small k (*e.g.*, $k = 1$) is sensitive to noise and outliers.
  - A large k may include objects from different classes, reducing classification precision.
  - Optimal values of k are typically determined through cross-validation.

- **Weighted Voting**: When the distances to the nearest neighbors vary greatly, a weighted voting scheme can improve accuracy. A common weighting function is the inverse squared distance:

$$W_i = \frac{1}{d(\mathbf{z}, \mathbf{x}_i)^2}, \quad \text{for} \quad x_i \in N_k \, . \quad (11)$$

The class assignment then becomes:

$$c_z = \arg\max_{v \in L} \sum_{y \in \mathcal{N}_k} W_y \cdot I(v = c_y) \, . \quad (12)$$

The k-NN algorithm's simplicity and adaptability make it a powerful tool, particularly for smaller datasets and multimodal distributions. However, it is computationally intensive for large datasets, as the distance calculation scales with the size of the training data [20,21].

2.2.3. *Decision trees*

Decision trees are widely used in machine learning, image processing, and pattern detection due to their interpretability and flexibility. They model data through a hierarchical structure of decisions, where each decision corresponds to a test on an input feature. These trees effectively handle both discrete and continuous variables and require no assumptions about the underlying data distribution, making them robust for diverse and complex datasets [22,23].

In creating a decision tree model, fundamental elements such as nodes and branches must be considered, and it is crucial to carry out actions such as splitting, stopping, and pruning to obtain an optimal model. These concepts are detailed below:

1. Root Node: The starting point of the tree, representing the first decision based on a selected feature.

2. Internal Nodes: Represent intermediate decisions, where a feature is compared to a threshold, directing the flow to child nodes.
3. Leaf Nodes: Terminal nodes that provide the final classification or decision outcome.

Branches

Branches represent the possible outcomes of tests performed at the nodes. Each path from the root to a leaf forms a classification rule, which can be expressed in "if-then" form. For example:

$$\text{If } x_1 \leq a \text{ and } x_2 > \beta, \text{ then class } C. \tag{13}$$

Splitting

At each node, the data is split based on an input feature and a threshold. The goal is to maximize the homogeneity of the resulting subsets. The following metrics are commonly used to evaluate splits:

1. Entropy (H): Measures the impurity of a set S:
$$H(S) = -\sum_{i=1}^{k} p_i \log_2 p_i, \tag{14}$$
where p_i is the proportion of examples in class i, and k is the number of classes.
2. Information Gain ($InfGain$): Quantifies the reduction in entropy after splitting:
$$\begin{aligned}\text{InfGain}(S, A) &= H(S) \\ &- \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v),\end{aligned} \tag{15}$$
where A is the splitting feature, S_v is the subset of S where A takes value v, and $|S|$ is the size of S.
3. Gini Index (G): Another impurity measure:
$$G(S) = 1 - \sum_{i=1}^{k} p_i^2, \tag{16}$$
splitting continues recursively, selecting the feature and threshold that maximize $InfGain$ or minimize $G(S)$.

Stopping criteria

To prevent overfitting, stopping rules are employed to control the tree's complexity:

- Minimum Node Size: Stop splitting if the number of records in a node falls below a threshold, typically between 0.25% and 1% of the dataset size.
- Maximum Tree Depth: Limit the tree's depth to ensure generalization.

Pruning

When stopping rules are insufficient, pruning can optimize tree size:

- Pre-Pruning (Forward Pruning): Limits tree growth during construction based on criteria such as minimum impurity reduction.
- Post-Pruning (Backward Pruning): Builds a large tree and reduces its size by removing nodes with little additional information.

The optimal subtree is chosen based on metrics like validation accuracy or cross-validation error. For example, pruning aims to minimize the misclassification error :

$$E = \frac{\text{Number of misclassified samples}}{\text{Total number of samples}}. \tag{17}$$

Learning process

The goal of a decision tree is to iteratively partition the feature space to isolate regions dominated by a single class. At each step:

1. Select the feature x_j and split point θ that optimize the chosen metric.
2. Partition the dataset into subsets:

$$S_{\text{left}}=\{x \in S \mid x_j \leq \theta\},\ S_{\text{right}}=\{x \in S \mid x_j > \theta\}. \tag{18}$$

3. Repeat the process for S_{left} and S_{right} until a stopping criterion is met or the subsets are pure.

Decision trees focus on one feature at a time, creating a simple and interpretable model. However, their sensitivity to overfitting highlights the importance of strategies like pruning and the use of ensemble methods, such as random forests, for improved performance [6,23,24].

2.2.4. Support vector machines

Support Vector Machines are the most widely adopted kernel learning algorithm. The solution this methodology offers is theoretically sophisticated, computationally efficient, and perceived as highly effective on a wide range of large-scale practical problems. The method originated from certain concepts of statistical learning theory related to the generalization abilities of learning systems [25,26].

A) Linearly separable case

Consider a dataset consisting of elements represented as pairs (x_i, y_i), where $x_i \in \mathbb{R}^n$ is a vector containing the features of a sample, and $y_i \in \{-1, 1\}$ represents the class label. If a hyperplane $w \cdot x + b = 0$ exists such that all samples are correctly classified, the dataset is said to be linearly separable.

This condition implies:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \forall i. \tag{19}$$

The optimal hyperplane maximizes the margin, defined as the distance between the hyperplane and the nearest data points (support vectors). The optimization problem is:

$$\min_{\mathbf{w},b} \tfrac{1}{2}\|\mathbf{w}\|^2, \text{ subject to } y_i(\mathbf{w} \cdot x_i + b) \geq 1,\ \forall i. \tag{20}$$

B) Quasi-linearly separable case

In practical scenarios, strict linear separability may not be feasible due to noise or overlapping data [25,26]. To address this, a soft margin is introduced, allowing some misclassification by using slack variables $\xi_i \geq 0$. The modified constraints are:

$$y_i\big(\mathbf{w} \cdot \mathbf{x}_i + b\big) \geq 1 - \xi_i, \quad \forall i. \tag{21}$$

The optimization problem becomes:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i, \quad \text{subject to } \xi_i \geq 0. \tag{22}$$

Here, $C > 0$ is a regularization parameter that controls the trade-off between maximizing the margin and minimizing classification errors.

C) Non-linear case

For datasets that are not linearly separable, a kernel function $\phi(x)$ is used to map the data into a higher-dimensional feature space where linear separability might be achieved [27]. The optimization problem remains similar, but computations are performed using the kernel trick:

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}'). \tag{23}$$

Common kernel functions include:

Polynomial: $K(x, x') = (1 + \langle x, x' \rangle)^d$,

Radial Basis Function (RBF): $K(x, x') = \exp\left(-\gamma \|x_i - x_j\|^2\right)$,

Sigmoid: $K(x, x') = \tanh\left(k \langle x, x' \rangle + \Theta\right)$.

D) Multiclass Classification: The challenge becomes more complicated when dealing with classification tasks involving multiple classes. This is because the input instances can belong to any of the C available classes, which must be mutually exclusive. Several strategies are proposed to address this situation with support vector machines. Among them are approaches based on the one-against-all methodology, the one-against-one approach, and the error-correcting output encoding method. Each method offers a different perspective to decompose the multiclass problem into a series of binary subproblems, thus facilitating its resolution [28,29].

E) One-against-all:

This approach builds C binary classifiers. For each classifier, samples from one class are labeled as positive $+1$, and samples from all other classes are labeled as negative (-1). A new sample is classified by the classifier with the highest output:

$$\hat{y} = \text{arg} \max_{c \in \{1,\ldots,C\}} f_c(x), \tag{24}$$

where $f_c(x)$ is the decision function for class c [29].

F) One-vs-one

The one-vs-one strategy creates $C(C-1)/2$ binary classifiers, each trained on data from a pair of classes. During classification, a majority voting scheme is used:

$$\hat{y} = \underset{c \in \{1,\ldots,C\}}{\arg\max}\ \text{votes}(c), \tag{25}$$

where votes(c) represents the number of classifiers that predict class c [29].

G) Error-correcting output coding

In this method, each class is represented by a unique binary code of L bits. Each of the L classifiers predict one bit of the code. For a new sample, the classifiers produce a binary string, and the class with the closest code (based on Hamming or Euclidean distance) is assigned:

$$\hat{y} = \underset{c \in \{1,\ldots,C\}}{\arg\min}\ d(b, b_c), \tag{26}$$

where b is the predicted bit string, b_c is the code for class c, and d is the distance metric [28,29].

2.2.5. Ensemble classifiers

There is a statistical concept or phenomenon known as "the wisdom of the crowd," which suggests that combining (ensembling) multiple average-performing predictors makes it possible to obtain superior results compared to a single highly accurate predictor acting alone. Ensemble learning, or ensemble systems, is a general term for methods combining multiple base learners to make a decision, typically in supervised machine learning tasks. A base learner, or inducer, is an algorithm that inputs labeled examples and produces a model that generalizes this data. The produced model (*e.g.*, a decision tree, neural network, linear regression model, etc.) can make classifications or predictions for new unlabeled examples. The central premise of ensemble learning is that by combining multiple models, the errors of a single base learner will likely be compensated by others, and as a result, the overall prediction performance of the ensemble will be better than that of a single model [30].

Three fundamental strategies must be considered for constructing an efficient assembled system: data sampling or selection, training of the classifiers that are part of the ensemble, and the combination of these classifiers [30,31].

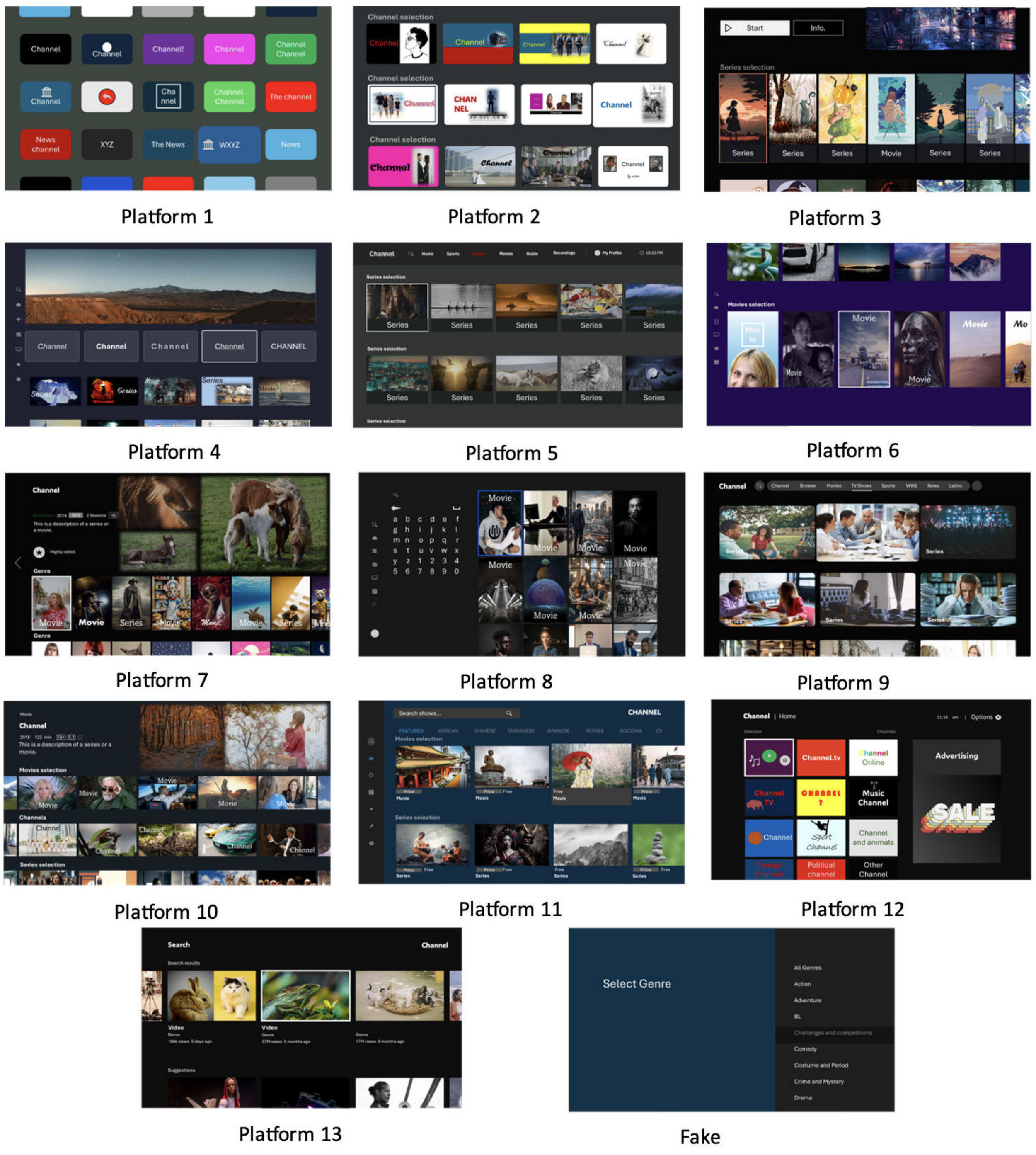

FIGURE 3. Mosaics.

2.3. Performance measures

Evaluating the machine learning method is a fundamental stage in any project. Although a model may show satisfactory results when evaluated using a specific metric, such as the accuracy score, its performance may be poor when other relevant metrics are analyzed, such as precision, F1 measure, or other alternatives. Often, we rely solely on classification accuracy as an indicator of model success, but more is needed. In addition, it is essential to evaluate the model on independent validation and test data sets to ensure its generalization and robustness in real-world situations [32].

In this section, we discuss the various types of metric analysis used to thoroughly evaluate the models' performance.

2.3.1. Confusion matrix

The confusion matrix is a fundamental tool for predictive analysis in machine learning. When working with classification models, it evaluates their performance and helps understand the nature of the errors made. In essence, it is a table or matrix that summarizes the number of instances correctly classified and incorrectly classified by the model for each class of the problem [33].

2.3.2. Metrics

Metrics used in this numerical experiment are:

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN}, \tag{27}$$

$$Precision = \frac{VP}{VP + FP}, \tag{28}$$

$$Sensitivity = \frac{VP}{VP + FN}, \tag{29}$$

$$Specificity = \frac{VN}{VN + FP}, \tag{30}$$

$$\text{Measure F1} = \frac{2 \cdot Sensitivity \cdot Precision}{Sensitivity + Precision}, \tag{31}$$

where

- VP (True Positive): Corresponds to the instances in which the model correctly predicted the positive or affirmative class.
- FP (False Positive): Refers to those cases in which the model wrongly predicted the positive class when, in fact, the instance belonged to the negative class.
- FN (False Negative): Represents the instances in which the model failed to predict the positive class, incorrectly classifying them as unfavorable.
- VN (True Negative): Encompasses the instances belonging to the negative class that were correctly predicted as such by the model.

2.4. Data preparation, feature extraction, and model development for classification

In this work, image sets from various multimedia platforms were used. On the one hand, mosaic images were processed, and on the other hand, images were called descriptors. Both sets are explained in detail below.

2.4.1. Mosaics

The mosaic represents an overview of various available options, such as the menu of some multimedia platforms that show thumbnails of movies or series. Each thumbnail in the mosaic represents a different option, and the main objective is to provide a quick overview of what is available to the user. Figure 3 shows an example of the images used by each channel.

This set is composed of 153,488 images of 1280×720 pixels represented in 14 different channels or classes. A false channel allows the model to identify and classify instances that do not fit any of the channels of interest. Four thousand randomly chosen and representative images were used for each channel for training, and the rest were reserved for validation. Table I shows the distribution by channel.

After the data set was separated, the Binary Local Pattern technique, previously described, was used to extract discriminative features that captured the structure and texture of the grayscale images. This approach allowed for obtaining compact and meaningful numerical representations of images where a vector of 59 elements represented each image. This laid the groundwork for the development of machine learning models.

TABLE I. Distribution of the set of images by channel for the analysis of the mosaics.

Channels	Total	Training	Validation
Platform_1	11064	4000	7064
Platform_2	10540	4000	6540
Platform_3	11300	4000	7300
Platform_4	10478	4000	6478
Platform_5	5994	4000	1994
Platform_6	10002	4000	6002
Platform_7	11516	4000	7516
Platform_8	10524	4000	6524
Platform_9	10000	4000	6000
Platform_10	28884	4000	24884
Platform_11	10000	4000	6000
Platform_12	9962	4000	5962
Platform_13	11906	4000	7906
Fake	5278	4000	1278
Total	157448	56000	101448

TABLE II. The configuration of the ten algorithms was implemented and validated for mosaic classification.

Models	Description
SVM linear	Type: Support vector machine Kernel function: Linear Multiclass method: one vs one
SVM quadratic	Type: Support vector machine Kernel function: Quadratic polynomial. Multiclass method: one vs one
SVM cubic	Type: Support vector machine Kernel function: Cubic polynomial. Multiclass method: one vs one
SVM fine Gaussian	Type: SVM Kernel: Gaussian Scale: 1.9 Multiclass: one vs one
SVM mean Gaussian	Type: SVM Kernel: Gaussian Scale: 1.9 Multiclass: one vs one
SVM thick Gaussian	Type: SVM Kernel: Gaussian Scale: 31 Multiclass: one vs one
Ensemble boosted trees:	Type: Ensemble; Method: AdaBoost Classifier type: Decision tree Maximum number of divisions: 20; Number of classifiers: 30
Ensemble bagged trees:	Type: Ensemble; Method: Bag Classifier type: Decision tree Maximum number of divisions: 55999; Number of classifiers: 30
Ensemble subspace discriminant:	Type: Ensemble; Method: Subspace Classifier type: Discriminant Subspace dim: 30 Number of classifiers: 30
Ensemble subspace KNN:	Type: Ensemble; Method: KNN Classifier type: Nearest neighbor Subspace dim: 30 Number of classifiers: 30

In this research, ten algorithms were implemented and evaluated to address the proposed classification problem. These models included variants of support vector machines with linear, polynomial, and Gaussian kernels of various scales and ensemble systems. SVMs used the one-versus-one method to handle multiclass problems. The ensemble systems included boosted trees, in which AdaBoost was used as an ensemble method with decision trees as base classifiers. The bagging ensemble method was implemented with decision trees. Discriminant and KNN in subspaces used the subspace ensemble method, with linear discriminant and nearest neighbor type base classifiers, respectively. A description of these models is presented in Table II. This diversity of models allowed us to evaluate their performance thoroughly and determine the most suitable approaches for the specific classification task. It should be noted that, in the training stage, the holdout validation technique [34] was used to evaluate the performance of the models. In this method, the dataset was divided into two subsets:

75% of the data was used to train the model, while the remaining 25% was separated as a test or validation set, which allowed obtaining an unbiased estimate of the model's performance on unseen data. This technique is recommended for large datasets, such as those used in this case. Once the process was complete, the model's performance metrics were calculated with the previously separated validation data, *i.e.*, with images the model had never interacted with.

2.4.2. *Descriptors*

Once the user selects a movie or series from the mosaic and enters it, we could consider that image a descriptor. In this context, the image would represent the specific movie or series that the user has chosen, and it contains more detailed and specific information about the content compared to the thumbnails in the mosaic. This image could contain scenes, characters, and plot details, among other aspects, that are not captured in the mosaic.

In order to select a descriptor that allows the platform to be identified, the idea of focusing on specific regions of the image that remain relatively constant across different data sets or changes in the environment was adopted. This approach allows the model to be trained with relevant features from these stable regions, which helps mitigate the impact of variations in other parts of the image. This can reduce the need to completely retrain the model in case of changes in the data set or the environment, resulting in a more efficient and less computationally intensive process. Figure 4 shows examples of the images that make up the different channels and the region selected for processing. This dataset consists of 33,471 images of 1280×720 pixels divided into 11 different channels, including a fake category that allows the model to identify instances that do not belong to any of the ten target classes. One thousand representative images, randomly chosen from each channel, were used for training, while the rest were used for validation. This distribution is detailed in Table III.

The same procedures as with the mosaic set were followed: separation into training and validation data, feature extraction using the LBP technique, and development of classification models such as support vector machines with dif-

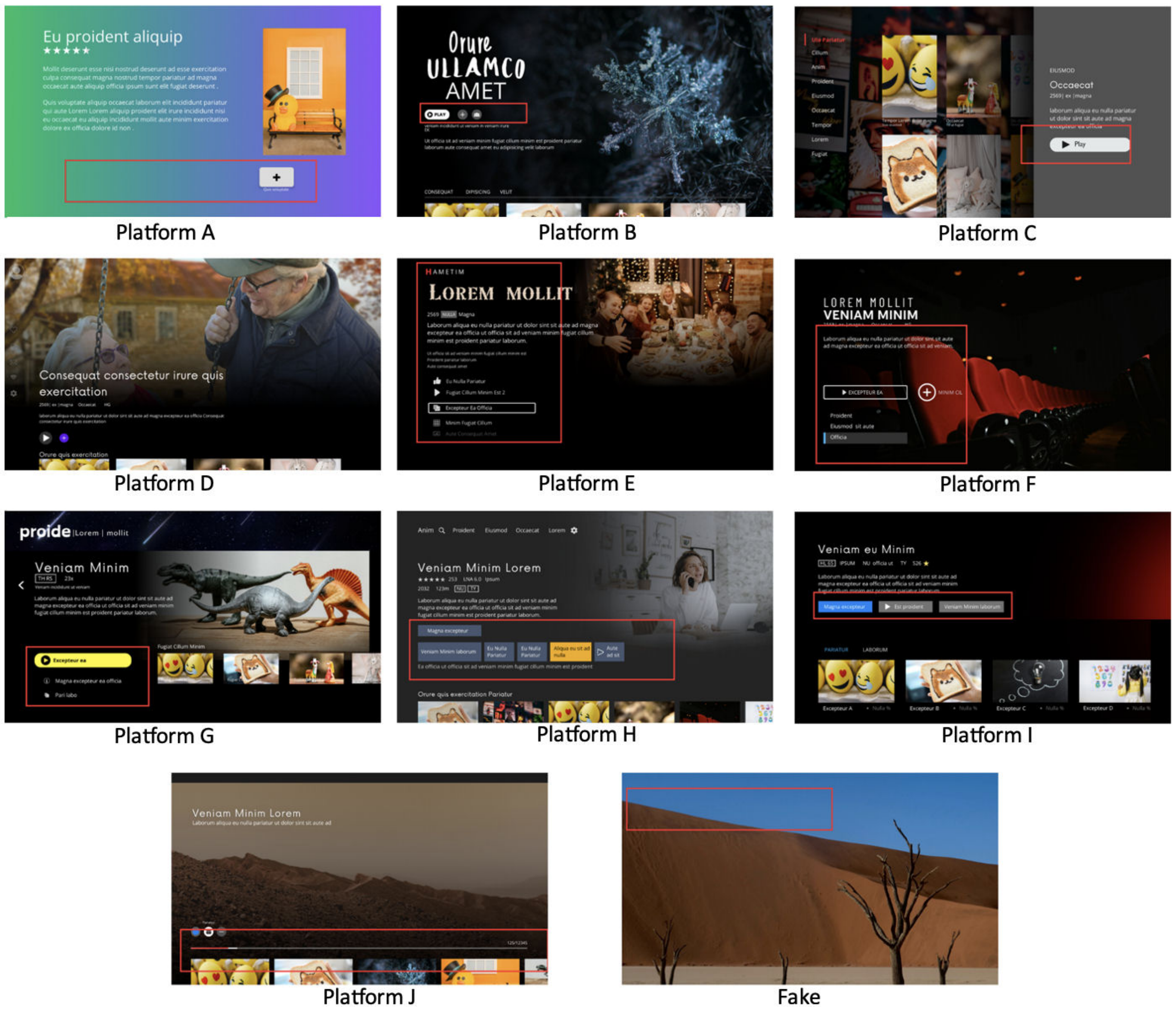

FIGURE 4. Descriptors: Sample images from various platforms focusing on the region of interest for processing.

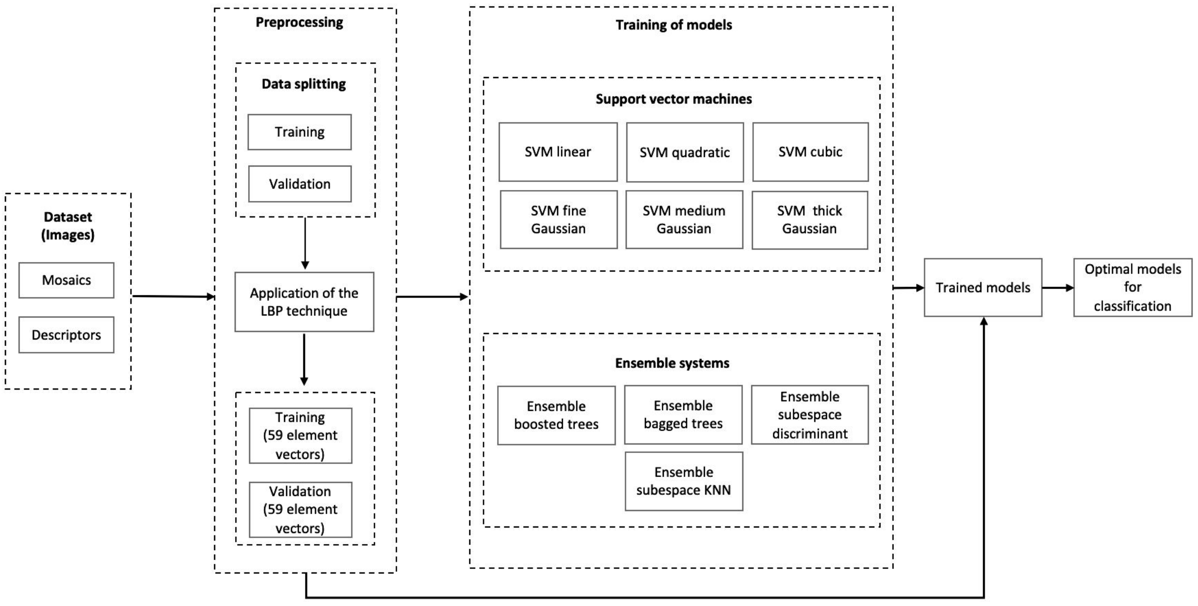

FIGURE 5. Overview of the proposed method.

TABLE III. Distribution of images in each category for the study of descriptors.

Channels	Total	Training	Validation
Platform_1	1307	1000	307
Platform_2	1943	1000	943
Platform_3	1633	1000	633
Platform_4	5378	1000	4378
Platform_5	5657	1000	4657
Platform_6	1423	1000	423
Platform_7	5060	1000	4060
Platform_8	3130	1000	2130
Platform_9	1939	1000	939
Platform_10	5744	1000	4744
Fake	5000	1000	4000
Total	38214	11000	27214

ferent kernels and ensemble classifiers, as shown in Table IV. In the training stage, k-fold cross-validation [35] was implemented with $K = 5$. The dataset was divided into five partitions of the same size during this process. Each iteration used one of these partitions as the test set, while the other four were used as the training set. This ensured a thorough evaluation of the model, allowing for a more accurate and generalizable estimate of its performance on unseen data. Finally, performance metrics were calculated where the previously separated validation data was used.

2.5. Experimental design

The experimental tests were conducted on a computer with 8.00 GB of RAM and an Intel® Core^{TM} i5-1035G1 processor capable of reaching 1.19 GHz in turbo mode. The software used to perform the classification process was MATLAB version 2020b. Two data sets were used, as described above: mosaics and descriptors. Both data sets were used to evaluate the performance of the ten machine learning algorithms applied to the classification problem addressed in this work.

The equipment specifications guaranteed an adequate environment to process the volumes of data handled during the experiments efficiently. MATLAB was chosen due to its powerful numerical calculation and matrix algebra capabilities, as well as its optimized libraries for machine learning and artificial vision tasks, which facilitated the implementation of the different algorithms evaluated. Figure 5 shows the outline of the proposed strategy.

TABLE IV. Characteristics of the ten algorithms implemented and evaluated for classifying the data set belonging to the descriptors.

Models	Description
SVM linear	Type: Support Vector Machine Kernel function: Linear Multiclass: one vs one
SVM quadratic	Type: Support Vector Machine Kernel function: Quadratic polynomial. Multiclass: one vs one
SVM cubic	Type: Support Vector Machine Kernel function: Cubic polynomial. Multiclass: one vs one
SVM fine Gaussian	Type: SVM Kernel function: Gaussian Kernel scale: 1.9 Multiclass: one vs one
SVM mean Gaussian	Type: SVM Kernel function: Gaussian Kernel scale: 7.7 Multiclass: one vs one
SVM thick Gaussian	Type: SVM Kernel function: Gaussian Kernel scale: 31 Multiclass: one vs one
Ensemble boosted trees;	Type: Ensemble system Method: AdaBoost Classifier type: Decision tree Maximum number of divisions: 20; Number of classifiers: 30
Ensemble bagged trees	Type: Ensemble system Method: Bag Classifier type: Decision tree Maximum number of divisions: 10999; Number of classifiers: 30
Ensemble subspace discriminant	Type: Ensemble system Method: Subspace Classifier type: Discriminant Subspace dim: 30; Number of classifiers: 30
Ensemble subspace KNN	Type: Ensemble system Method: Subspace Classifier type: Nearest neighbor Subspace dim: 30; Number of classifiers: 30

3. Results

Several widely accepted metrics in this field were used to evaluate the performance of the ten machine learning models implemented in this study, detailed in the performance measures section. The metrics used include precision, specificity, sensitivity, accuracy, and the F1 measure, each of which provides:

- Valuable information about different aspects of the model's performance.

- Allowing an objective and detailed analysis of their performance.

- The identification of their strengths and weaknesses.

Since the splitting, training, and validation process explained above was carried out randomly, it was repeated 30 times to prevent bias, in line with the central limit theorem. Below, the models' accuracy and average training time are presented, as well as the mean and standard deviation for each metric, to validate the consistency of each trained algorithm for each data set: mosaics and descriptors.

3.1. Mosaics

Table V shows the accuracy and training average of the mosaic classification algorithms. Table VI shows the evaluation metrics of the linear support vector machine methodology on the mosaic validation set. Table VII shows the evaluation metrics with the same method used in Table VI but with the quadratic kernel; Table VIII shows the evaluation metrics but with the cubic kernel; Table IX with fine Gaussian kernel; Table X with mean Gaussian kernel, and Table XI with thick Gaussian kernel. Table XII shows the metrics with the boosted tree ensemble classifier. Table XIII shows the metrics with the same ensemble as in Table XII using bagging. The evaluation metrics of the subspace discriminant analysis are shown in Table XIV. The evaluation metrics using the k-nearest neighbors' classifier in subspaces are shown in Table XV

3.2. Descriptors

Table XVI shows the accuracy and training average of the descriptors classification algorithms. Table XVII shows the evaluation metrics of the linear support vector machine methodology on the descriptors validation set. Table XVIII shows the evaluation metrics with the same method used in Table XVII but with the quadratic kernel; Table XIX shows the evaluation metrics but with the cubic kernel; Table XX with fine Gaussian kernel; Table XXI with mean Gaussian kernel, and Table XXII with thick Gaussian kernel. Table XXIII shows the metrics with the boosted tree ensemble classifier. Table XXIV shows the metrics with the same en semble as in Table XXIII using bagging.

TABLE V. Accuracy and average training time of mosaic classification algorithms.

Models	Accuracy	Time (seconds)
SVM linear	0.968	271.04
SVM quadratic	0.989	456.09
SVM cubic	0.989	425.25
SVM fine Gaussian	0.951	792.15
SVM mean Gaussian	0.973	512.19
SVM thick Gaussian	0.918	724.60
Ensemble boosted trees	0.691	848.08
Ensemble bagged trees	0.960	591.24
Ensemble subspace discriminant	0.892	614.58
Ensemble subspace KNN	0.961	2740.80

TABLE VI. Evaluation metrics of the linear support vector machine on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Channels	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_1	0.990348	0.000361	0.999271	0.000027	0.998783	0.000071	0.999237	0.000024	0.994547	0.000174
Platform_2	0.987989	0.000292	0.999174	0.000020	0.986488	0.000432	0.998356	0.000029	0.987238	0.000223
Platform_3	0.984406	0.000333	0.998803	0.000026	0.974575	0.000329	0.997060	0.000036	0.979466	0.000249
Platform_4	0.956499	0.000878	0.997047	0.000062	0.951739	0.000898	0.994154	0.000088	0.954113	0.000691
Platform_5	0.736821	0.006997	0.995592	0.000158	0.967032	0.002438	0.995232	0.000164	0.836358	0.004814
Platform_6	0.950660	0.002537	0.998976	0.000055	0.984203	0.001123	0.998685	0.000069	0.967139	0.001684
Platform_7	0.985920	0.000368	0.999117	0.000023	0.983100	0.000378	0.998170	0.000034	0.984508	0.000289
Platform_8	0.971816	0.000594	0.997805	0.000046	0.946004	0.001290	0.993967	0.000125	0.958736	0.000872
Platform_9	0.959489	0.000497	0.997205	0.000037	0.963003	0.001228	0.995006	0.000076	0.961242	0.000613
Platform_10	0.953855	0.001106	0.997047	0.000078	0.970894	0.002033	0.995500	0.000097	0.962297	0.000845
Platform_11	0.997726	0.000046	0.999272	0.000015	0.982556	0.000659	0.995172	0.000158	0.990083	0.000328
Platform_12	0.977468	0.000550	0.998566	0.000036	0.989322	0.000239	0.998020	0.000041	0.983359	0.000339
Platform_13	0.979935	0.000542	0.998748	0.000034	0.979442	0.000809	0.997613	0.000065	0.979688	0.000560
Fake	0.960869	0.001339	0.996701	0.000115	0.958361	0.001149	0.993713	0.000178	0.959613	0.001136
Mean	0.956700	0.001174	0.998095	0.000052	0.973965	0.000934	0.996420	0.000085	0.964171	0.000916

TABLE VII. Evaluation metrics of the support vector machine with quadratic kernel on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Channels	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_1	0.998026	0.000236	0.999852	0.000018	0.999344	0.000087	0.999817	0.000018	0.998684	0.000128
Platform_2	0.996933	0.000216	0.999789	0.000015	0.997197	0.000355	0.999621	0.000033	0.997065	0.000258
Platform_3	0.994749	0.000224	0.999595	0.000017	0.989607	0.000329	0.998876	0.000029	0.992172	0.000200
Platform_4	0.986164	0.000455	0.999054	0.000032	0.988777	0.000446	0.998398	0.000037	0.987469	0.000288
Platform_5	0.920788	0.005057	0.998917	0.000075	0.986228	0.001613	0.998757	0.000078	0.952378	0.002879
Platform_6	0.983444	0.000910	0.999664	0.000019	0.994952	0.000843	0.999572	0.000024	0.989164	0.000599
Platform_7	0.995421	0.000453	0.999712	0.000029	0.997234	0.000254	0.999565	0.000027	0.996327	0.000224
Platform_8	0.985638	0.000398	0.998857	0.000032	0.980060	0.000743	0.997465	0.000062	0.982841	0.000425
Platform_9	0.984565	0.001342	0.998942	0.000093	0.981601	0.000837	0.997827	0.000131	0.983081	0.001012
Platform_10	0.981465	0.000517	0.998820	0.000034	0.993711	0.000200	0.998518	0.000032	0.987550	0.000267
Platform_11	0.999182	0.000043	0.999736	0.000014	0.993188	0.000254	0.998130	0.000063	0.996176	0.000129
Platform_12	0.993122	0.000461	0.999566	0.000029	0.997833	0.000356	0.999463	0.000035	0.995472	0.000294
Platform_13	0.994401	0.000389	0.999650	0.000024	0.994912	0.000405	0.999372	0.000037	0.994656	0.000318
Fake	0.984080	0.000959	0.998656	0.000081	0.982760	0.001202	0.997417	0.000147	0.983419	0.000947
Mean	0.985570	0.000833	0.999344	0.000037	0.991243	0.000566	0.998771	0.000054	0.988318	0.000569

TABLE VIII. Evaluation metrics of the support vector machine with cubic kernel on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Channels	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_1	0.999052	0.000092	0.999929	0.000007	0.999221	0.000097	0.999880	0.000011	0.999137	0.000080
Platform_2	0.997105	0.000212	0.999801	0.000015	0.997105	0.000165	0.999627	0.000019	0.997105	0.000150
Platform_3	0.995167	0.000318	0.999626	0.000025	0.991886	0.000246	0.999069	0.000031	0.993524	0.000213
Platform_4	0.989857	0.000521	0.999310	0.000036	0.987759	0.000656	0.998572	0.000045	0.988807	0.000352
Platform_5	0.941333	0.003306	0.999213	0.000047	0.989567	0.000993	0.999091	0.000051	0.964845	0.001900
Platform_6	0.982559	0.000671	0.999646	0.000014	0.994483	0.000543	0.999545	0.000019	0.988485	0.000473
Platform_7	0.995424	0.000267	0.999712	0.000017	0.997934	0.000155	0.999606	0.000021	0.996678	0.000174
Platform_8	0.983085	0.000738	0.998644	0.000060	0.984611	0.000670	0.997605	0.000084	0.983847	0.000567
Platform_9	0.989404	0.000487	0.999276	0.000034	0.983216	0.000887	0.998243	0.000068	0.986300	0.000532
Platform_10	0.983949	0.000742	0.998979	0.000048	0.995778	0.000298	0.998790	0.000054	0.989828	0.000453
Platform_11	0.998939	0.000092	0.999657	0.000030	0.994212	0.000183	0.998321	0.000055	0.996570	0.000113
Platform_12	0.995106	0.000504	0.999691	0.000032	0.998628	0.000136	0.999628	0.000031	0.996864	0.000262
Platform_13	0.995778	0.000278	0.999736	0.000017	0.995661	0.000281	0.999497	0.000022	0.995720	0.000188
Fake	0.987085	0.000972	0.998910	0.000083	0.985336	0.000569	0.997852	0.000088	0.986209	0.000560
Mean	0.988132	0.000657	0.999438	0.000033	0.992528	0.000420	0.998952	0.000043	0.990280	0.000430

TABLE IX. Evaluation metrics of the support vector machine with fine Gaussian kernel on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Channels	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_1	0.998720	0	0.999905	0	0.994196	0	0.999507	0	0.996453	0
Platform_2	0.996194	0	0.999747	0	0.960550	0	0.997220	0	0.978048	0
Platform_3	0.994643	0	0.999607	0	0.941096	0	0.995397	0	0.967129	0
Platform_4	0.681987	0	0.968379	0	0.994134	0	0.970024	0	0.808994	0
Platform_5	0.877218	0	0.998273	0	0.967136	0	0.997881	0	0.919985	0
Platform_6	0.972320	0	0.999447	0	0.968907	0	0.998847	0	0.970610	0
Platform_7	0.980763	0	0.998806	0	0.968344	0	0.997003	0	0.974514	0
Platform_8	0.972729	0	0.998009	0	0.887440	0	0.989817	0	0.928129	0
Platform_9	0.945946	0	0.996250	0	0.954936	0	0.993593	0	0.950420	0
Platform_10	0.969702	0	0.998104	0	0.965500	0	0.996175	0	0.967596	0
Platform_11	0.996074	0	0.998785	0	0.948320	0	0.986407	0	0.971611	0
Platform_12	0.996407	0	0.999780	0	0.970500	0	0.998048	0	0.983283	0
Platform_13	0.997222	0	0.999832	0	0.963267	0	0.997684	0	0.979951	0
Fake	0.963505	0	0.996985	0	0.941690	0	0.992676	0	0.952472	0
Mean	0.953102	0	0.996565	0	0.959001	0	0.993591	0	0.953514	0

TABLE X. Evaluation metrics of the support vector machine with mean Gaussian kernel on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Channels	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_1	0.992395	0	0.999428	0	0.997593	0	0.999300	0	0.994988	0
Platform_2	0.991411	0	0.999410	0	0.988379	0	0.998699	0	0.989893	0
Platform_3	0.990537	0	0.999278	0	0.975068	0	0.997536	0	0.982742	0
Platform_4	0.967935	0	0.997799	0	0.973912	0	0.996274	0	0.970914	0
Platform_5	0.767456	0	0.996276	0	0.963224	0	0.995860	0	0.854268	0
Platform_6	0.942446	0	0.998793	0	0.985456	0	0.998531	0	0.963471	0
Platform_7	0.989474	0	0.999340	0	0.986671	0	0.998590	0	0.988070	0
Platform_8	0.979711	0	0.998414	0	0.957291	0	0.995367	0	0.968371	0
Platform_9	0.970878	0	0.998009	0	0.965819	0	0.995939	0	0.968342	0
Platform_10	0.966018	0	0.997842	0	0.976000	0	0.996550	0	0.970983	0
Platform_11	0.996613	0	0.998916	0	0.981554	0	0.994657	0	0.989027	0
Platform_12	0.980538	0	0.998764	0	0.990833	0	0.998295	0	0.985659	0
Platform_13	0.989720	0	0.999361	0	0.985072	0	0.998521	0	0.987391	0
Fake	0.953529	0	0.996002	0	0.970655	0	0.994026	0	0.962016	0
Mean	0.962762	0	0.998402	0	0.978395	0	0.997010	0	0.969724	0

TABLE XI. Evaluation metrics of the support vector machine with thick Gaussian kernel on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Channels	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_1	0.970738	0	0.997754	0	0.995612	0	0.997605	0	0.983018	0
Platform_2	0.949377	0	0.996491	0	0.954893	0	0.993810	0	0.952127	0
Platform_3	0.954274	0	0.996506	0	0.940548	0	0.992479	0	0.947361	0
Platform_4	0.896096	0	0.993050	0	0.878666	0	0.985746	0	0.887295	0
Platform_5	0.485880	0	0.987641	0	0.915493	0	0.986732	0	0.634835	0
Platform_6	0.763180	0	0.994218	0	0.929288	0	0.992942	0	0.838082	0
Platform_7	0.957164	0	0.997328	0	0.949350	0	0.994490	0	0.953241	0
Platform_8	0.931981	0	0.994752	0	0.898749	0	0.987639	0	0.915064	0
Platform_9	0.952702	0	0.997040	0	0.867566	0	0.988713	0	0.908143	0
Platform_10	0.902750	0	0.993777	0	0.919000	0	0.989354	0	0.910803	0
Platform_11	0.986442	0	0.995781	0	0.944382	0	0.983174	0	0.964954	0
Platform_12	0.939623	0	0.996071	0	0.972667	0	0.994687	0	0.955859	0
Platform_13	0.962794	0	0.997780	0	0.920161	0	0.993218	0	0.940995	0
Fake	0.882446	0	0.989620	0	0.921958	0	0.984347	0	0.901769	0
Mean	0.895389	0	0.994843	0	0.929167	0	0.990353	0	0.906682	0

TABLE XII. Evaluation metrics of the boosted tree ensemble classifier on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Channels	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_1	0.974655	0	0.998072	0	0.990798	0	0.997565	0	0.982661	0
Platform_2	0.883236	0	0.993689	0	0.692813	0	0.974292	0	0.776521	0
Platform_3	0.757856	0	0.981094	0	0.763151	0	0.965411	0	0.760494	0
Platform_4	0.630372	0	0.974550	0	0.636308	0	0.952951	0	0.633326	0
Platform_5	0.308373	0	0.979714	0	0.708920	0	0.976303	0	0.429791	0
Platform_6	0.314857	0	0.969674	0	0.695085	0	0.964277	0	0.433396	0
Platform_7	0.475747	0	0.954592	0	0.655282	0	0.936884	0	0.551265	0
Platform_8	0.775607	0	0.984957	0	0.649814	0	0.960127	0	0.707160	0
Platform_9	0.812856	0	0.990645	0	0.591202	0	0.964957	0	0.684533	0
Platform_10	0.537860	0	0.970906	0	0.538667	0	0.945341	0	0.538263	0
Platform_11	0.804491	0	0.958949	0	0.519732	0	0.851214	0	0.631494	0
Platform_12	0.687374	0	0.974007	0	0.909167	0	0.970172	0	0.782865	0
Platform_13	0.864870	0	0.992753	0	0.742872	0	0.978068	0	0.799242	0
Fake	0.461356	0	0.922516	0	0.785226	0	0.911817	0	0.581219	0
Mean	0.663536	0	0.974723	0	0.705645	0	0.953527	0	0.663731	0

TABLE XIII. Evaluation metrics of the bagging tree ensemble classifier on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Channels	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_1	0.994491	0.000434	0.999586	0.000033	0.998282	0.000297	0.999495	0.000038	0.996383	0.000273
Platform_2	0.970401	0.001511	0.997935	0.000109	0.982268	0.001100	0.996925	0.000103	0.976297	0.000789
Platform_3	0.969596	0.001157	0.997705	0.000090	0.943708	0.001520	0.993820	0.000140	0.956476	0.001001
Platform_4	0.951357	0.002018	0.996635	0.000147	0.964619	0.001653	0.994591	0.000164	0.957940	0.001262
Platform_5	0.863776	0.005633	0.998025	0.000094	0.981273	0.001550	0.997814	0.000098	0.918774	0.003380
Platform_6	0.850140	0.006040	0.996539	0.000164	0.979020	0.001772	0.996194	0.000164	0.910028	0.003559
Platform_7	0.965233	0.002122	0.997809	0.000139	0.967255	0.001800	0.996001	0.000168	0.966241	0.001408
Platform_8	0.954514	0.001483	0.996455	0.000121	0.929688	0.001728	0.991508	0.000171	0.941936	0.001183
Platform_9	0.955924	0.001548	0.997040	0.000108	0.933987	0.001504	0.992985	0.000143	0.944827	0.001122
Platform_10	0.909474	0.002902	0.993939	0.000213	0.968606	0.001339	0.992440	0.000219	0.938106	0.001714
Platform_11	0.989255	0.000461	0.996625	0.000146	0.956035	0.000958	0.986669	0.000268	0.972361	0.000565
Platform_12	0.962966	0.001644	0.997623	0.000109	0.983117	0.001089	0.996765	0.000128	0.972936	0.001063
Platform_13	0.985414	0.001314	0.999091	0.000083	0.983820	0.000965	0.998193	0.000101	0.984615	0.000858
Fake	0.931189	0.002010	0.994049	0.000184	0.952791	0.001630	0.990834	0.000232	0.941865	0.001455
Mean	0.946695	0.002163	0.997075	0.000124	0.966033	0.001350	0.994588	0.000153	0.955628	0.001402

TABLE XIV. Evaluation metrics of the discriminant analysis in subspaces on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Channels	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_1	0.860311	0.002270	0.987853	0.000230	0.999500	0.000096	0.988664	0.000215	0.924695	0.001320
Platform_2	0.917884	0.002305	0.994136	0.000177	0.951142	0.001452	0.991364	0.000207	0.934215	0.001531
Platform_3	0.957446	0.002604	0.996759	0.000201	0.940306	0.002319	0.992697	0.000328	0.948798	0.002296
Platform_4	0.926250	0.002449	0.995790	0.000158	0.775069	0.003088	0.981696	0.000187	0.843936	0.001757
Platform_5	0.395416	0.004772	0.984001	0.000345	0.819953	0.003186	0.981934	0.000323	0.533516	0.004181
Platform_6	0.685908	0.005722	0.991545	0.000226	0.920695	0.001447	0.990153	0.000217	0.786132	0.003710
Platform_7	0.963106	0.000856	0.997804	0.000053	0.911735	0.002230	0.992712	0.000133	0.936715	0.001220
Platform_8	0.913851	0.001919	0.993311	0.000163	0.886717	0.002385	0.985414	0.000227	0.900077	0.001587
Platform_9	0.915225	0.003374	0.995409	0.000193	0.721040	0.002253	0.977765	0.000275	0.806608	0.002334
Platform_10	0.830257	0.003014	0.988222	0.000256	0.916356	0.001850	0.983972	0.000240	0.871180	0.001744
Platform_11	0.994552	0.000364	0.998386	0.000108	0.906411	0.000887	0.975826	0.000242	0.948438	0.000535
Platform_12	0.940046	0.001963	0.996113	0.000135	0.969528	0.000792	0.994540	0.000135	0.954558	0.001087
Platform_13	0.992690	0.001121	0.999579	0.000065	0.916203	0.001038	0.994679	0.000082	0.952913	0.000725
Fake	0.781083	0.003754	0.977905	0.000497	0.932646	0.000798	0.974378	0.000425	0.850156	0.002061
Mean	0.862430	0.002606	0.992630	0.000201	0.897664	0.001702	0.986128	0.000231	0.870853	0.001863

TABLE XV. Evaluation metrics of the k-nearest neighbors' classifier in subspaces on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Channels	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_1	0.992743	0.000529	0.999453	0.000040	0.999203	0.000160	0.999436	0.000038	0.995962	0.000271
Platform_2	0.979872	0.001101	0.998604	0.000078	0.986259	0.000722	0.997808	0.000091	0.983055	0.000695
Platform_3	0.961398	0.001193	0.996981	0.000095	0.969708	0.001270	0.995018	0.000153	0.965535	0.001057
Platform_4	0.964808	0.001599	0.997637	0.000111	0.949630	0.001292	0.994572	0.000146	0.957158	0.001140
Platform_5	0.841988	0.004189	0.997653	0.000072	0.980125	0.001702	0.997432	0.000082	0.905816	0.002826
Platform_6	0.747696	0.004908	0.993332	0.000172	0.985490	0.001045	0.993177	0.000174	0.850272	0.003295
Platform_7	0.958581	0.001889	0.997346	0.000126	0.976624	0.001079	0.996120	0.000142	0.967517	0.001168
Platform_8	0.951460	0.001821	0.996203	0.000147	0.930051	0.001457	0.991302	0.000206	0.940633	0.001392
Platform_9	0.965747	0.001192	0.997706	0.000081	0.940946	0.001340	0.994056	0.000139	0.953185	0.001099
Platform_10	0.919981	0.001630	0.994655	0.000117	0.977606	0.000895	0.993646	0.000133	0.947918	0.001057
Platform_11	0.988554	0.000403	0.996443	0.000125	0.945338	0.001273	0.983907	0.000359	0.966463	0.000765
Platform_12	0.968541	0.001364	0.997983	0.000090	0.987872	0.000658	0.997385	0.000092	0.978111	0.000761
Platform_13	0.993208	0.000398	0.999576	0.000025	0.993268	0.000408	0.999205	0.000037	0.993238	0.000317
Fake	0.941483	0.001729	0.995103	0.000148	0.932161	0.001960	0.990198	0.000251	0.936798	0.001628
Mean	0.941147	0.001711	0.997048	0.000102	0.968163	0.001090	0.994519	0.000146	0.952976	0.001248

TABLE XVI. Accuracy and average training time of the classification algorithms for descriptors.

Models	Accuracy	Time (seconds)
SVM linear	0.988	18.676
SVM quadratic	0.991	18.358
SVM cubic	0.993	18.482
SVM fine Gaussian	0.934	54.981
SVM mean Gaussian	0.987	20.687
SVM thick Gaussian	0.927	33.285
Ensemble boosted trees	0.900	45.507
Ensemble bagged trees	0.988	11.309
Ensemble subspace discriminant	0.917	11.094
Ensemble subspace KNN	0.982	132.810

The evaluation metrics of the subspace discriminant analysis are shown in Table XXV. The evaluation metrics using the k -nearest neighbors' classifier in subspaces are shown in Table XXVI.

Tables XXVII and XXVIII present a comparison of the evaluation metrics for each model on the mosaic and descriptor validation sets.

4. Discussion

The main objective of this research was to address image classification in the context of multimedia platforms. Specifically, it focused on classifying two types of images: mosaics and descriptors, as mentioned above. Local Binary Patterns technique was used, which obtained vectors of 59 elements containing detailed information about the texture of the images. These feature vectors served as input to implement ten machine-learning algorithms. The results obtained mainly were exceptional.

For mosaic classification, in the training stage, the options that stood out the most were support vector machines with cubic and quadratic kernels with adequate execution times, maintaining good computational performance. In the validation phase, the support vector machine with cubic kernel obtained the best performance with an average precision of 0.988132, specificity of 0.999438, sensitivity of 0.992528, accuracy of 0.998952, and F1 measure of 0.990280. The quadratic kernel followed with very close values. The bagging algorithm showed outstanding performance of the assembled classifiers, with an average accuracy of 0.99458827 and an F1 measure of 0.955628, contrary to the boosted trees that yielded the lowest values in the metrics in general.

Regarding the classification of descriptors, the results were equally outstanding for the support vector machines with quadratic and cubic kernels with averages above 98 percent in all their metrics. On the assembled systems, the bagging algorithm also showed solid performance, as did the boosted trees model, which contrasts with its performance on the mosaic dataset, where it performed much worse. For this data, the subspace discriminant analysis algorithm obtained the lowest score, which in metrics such as precision and the F1 measure reached a value of 85 percent.

TABLE XVII. Evaluation metrics of the linear support vector machine on the descriptor validation set.

Channels	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_A	0.992627	0.000183	0.998767	0.000031	0.963667	0.002282	0.993607	0.000335	0.977931	0.001183
Platform_B	0.936417	0.017865	0.999228	0.000235	0.990771	0.001235	0.999133	0.000237	0.962746	0.009768
Platform_C	0.888001	0.003407	0.997033	0.000101	0.987730	0.000796	0.996817	0.000105	0.935212	0.002015
Platform_D	0.994665	0.000228	0.998974	0.000044	0.997937	0.000042	0.998807	0.000038	0.996298	0.000118
Platform_E	0.999569	0	0.999911	0	0.996400	0.000092	0.999310	0.000016	0.997982	0.000046
Platform_F	0.960137	0.002866	0.999351	0.000049	0.990544	0	0.999214	0.000048	0.975101	0.001478
Platform_G	0.999081	0.000111	0.999839	0.000019	0.999236	0.000163	0.999749	0.000032	0.999159	0.000108
Platform_H	0.968594	0.000843	0.997249	0.000076	0.999061	0	0.997391	0.000070	0.983592	0.000435
Platform_I	1	0	1	0	0.994675	0	0.999816	0	0.997330	0
Platform_J	0.991626	0.000194	0.998251	0.000041	0.981036	0.000365	0.995250	0.000076	0.986303	0.000221
Fake	0.979769	0.001202	0.999272	0.000044	0.982538	0.001298	0.998692	0.000051	0.981151	0.000732
Mean	0.973681	0.002445	0.998898	0.000058	0.989418	0.000570	0.997981	0.000092	0.981164	0.001464

TABLE XVIII. Evaluation metrics of the support vector machine with quadratic kernel on the descriptor validation set.

Channels	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_A	0.991868	0.000273	0.998606	0.000049	0.986875	0.003510	0.996882	0.000504	0.989362	0.001740
Platform_B	0.981948	0.002287	0.999792	0.000027	0.991965	0.001653	0.999704	0.000032	0.986929	0.001400
Platform_C	0.949012	0.002642	0.998736	0.000068	0.987836	0.001029	0.998482	0.000074	0.968033	0.001535
Platform_D	0.997034	0	0.999431	0	0.998013	0.000136	0.999203	0.000022	0.997523	0.000068
Platform_E	0.999166	0.000167	0.999829	0.000034	0.994138	0.000362	0.998855	0.000063	0.996645	0.000186
Platform_F	0.916423	0.005513	0.998574	0.000102	0.989835	0.001102	0.998438	0.000106	0.951708	0.003158
Platform_G	0.997313	0.000062	0.999528	0.000011	0.999507	0	0.999525	0.000009	0.998409	0.000031
Platform_H	0.991762	0.005937	0.999293	0.000513	0.998122	0	0.999201	0.000473	0.994923	0.002992
Platform_I	1	0	1	0	0.994675	0	0.999816	0	0.997330	0
Platform_J	0.995314	0.000429	0.999022	0.000090	0.983354	0.000632	0.996291	0.000107	0.989298	0.000312
Fake	0.968018	0.002476	0.998833	0.000094	0.984093	0.001500	0.998322	0.000090	0.975987	0.001262
Mean	0.980714	0.001799	0.999240	0.000090	0.991674	0.000902	0.998611	0.000135	0.986013	0.001153

TABLE XIX. Evaluation metrics of the support vector machine with cubic kernel on the descriptor validation set.

Channels	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_A	0.993366	0.000203	0.998861	0.000035	0.989483	0.000660	0.997483	0.000099	0.991421	0.000341
Platform_B	0.977253	0.001504	0.999736	0.000018	0.993485	0	0.999666	0.000018	0.985302	0.000765
Platform_C	0.960524	0.001411	0.999029	0.000036	0.991680	0.000711	0.998858	0.000037	0.975853	0.000776
Platform_D	0.997487	0.000001	0.999518	0	0.997495	0.000228	0.999193	0.000037	0.997491	0.000114
Platform_E	0.999526	0.000243	0.999902	0.000050	0.995942	0.000221	0.999225	0.000064	0.997731	0.000186
Platform_F	0.963877	0.002376	0.999411	0.000040	0.994405	0.001159	0.999334	0.000044	0.978901	0.001374
Platform_G	0.997771	0.000062	0.999608	0.000011	0.999507	0	0.999593	0.000009	0.998638	0.000031
Platform_H	0.996610	0.000515	0.999712	0.000044	0.998388	0.000294	0.999608	0.000045	0.997498	0.000284
Platform_I	1	0	1	0	0.995705	0.000341	0.999852	0.000012	0.997848	0.000171
Platform_J	0.993955	0.000629	0.998733	0.000133	0.986509	0.000288	0.996602	0.000115	0.990218	0.000328
Fake	0.964634	0.002538	0.998701	0.000097	0.987098	0.000563	0.998299	0.000092	0.975735	0.001283
Mean	0.985909	0.000862	0.999383	0.000042	0.993609	0.000406	0.998883	0.000052	0.989694	0.000514

TABLE XX. Evaluation metrics of the support vector machine with fine Gaussian kernel on the descriptor validation set.

Channels	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_A	0.982312	0	0.997028	0	0.958000	0	0.991291	0	0.970004	0
Platform_B	0.978339	0	0.999777	0	0.882736	0	0.998457	0	0.928082	0
Platform_C	0.959350	0	0.999059	0	0.932070	0	0.997501	0	0.945513	0
Platform_D	0.997862	0	0.999606	0	0.959342	0	0.993129	0	0.978223	0
Platform_E	1	0	1	0	0.904445	0	0.983648	0	0.949825	0
Platform_F	0.628571	0	0.991751	0	0.884161	0	0.990079	0	0.734774	0
Platform_G	1	0	1	0	0.977340	0	0.996619	0	0.988540	0
Platform_H	0.998101	0	0.999841	0	0.986854	0	0.998824	0	0.992446	0
Platform_I	1	0	1	0	0.974441	0	0.999118	0	0.987055	0
Platform_J	0.997855	0	0.999599	0	0.882378	0	0.979165	0	0.936570	0
Fake	0.422783	0	0.951201	0	0.995758	0	0.952745	0	0.593552	0
Mean	0.905925	0	0.994351	0	0.939775	0	0.989143	0	0.909508	0

TABLE XXI. Evaluation metrics of the support vector machine with mean Gaussian kernel on the descriptor validation set.

Channels	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_A	0.991899	0	0.998622	0	0.979500	0	0.995811	0	0.985660	0
Platform_B	0.950000	0	0.999405	0	0.990228	0	0.999302	0	0.969697	0
Platform_C	0.921131	0	0.998006	0	0.977883	0	0.997538	0	0.948659	0
Platform_D	0.987780	0	0.997635	0	0.997031	0	0.997538	0	0.992384	0
Platform_E	0.999566	0	0.999911	0	0.988405	0	0.997942	0	0.993954	0
Platform_F	0.861570	0	0.997499	0	0.985816	0	0.997318	0	0.919515	0
Platform_G	0.999507	0	0.999914	0	0.999507	0	0.999853	0	0.999507	0
Platform_H	0.987454	0	0.998924	0	0.997653	0	0.998824	0	0.992527	0
Platform_I	1	0	1	0	0.994675	0	0.999816	0	0.997330	0
Platform_J	0.992665	0	0.998487	0	0.969857	0	0.993496	0	0.981128	0
Fake	0.951170	0	0.998173	0	0.991516	0	0.997942	0	0.970924	0
Mean	0.967522	0	0.998780	0	0.988370	0	0.997762	0	0.977390	0

TABLE XXII. Evaluation metrics of the support vector machine with thick Gaussian kernel on the descriptor validation set.

Channels	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_A	0.992846	0	0.999009	0	0.798000	0	0.969464	0	0.884823	0
Platform_B	0.315297	0	0.975545	0	0.986971	0	0.975674	0	0.477918	0
Platform_C	0.616240	0	0.987021	0	0.875197	0	0.984420	0	0.723238	0
Platform_D	0.986573	0	0.997416	0	0.990178	0	0.996252	0	0.988372	0
Platform_E	0.999343	0	0.999867	0	0.980030	0	0.996472	0	0.989592	0
Platform_F	0.710579	0	0.994588	0	0.841608	0	0.992210	0	0.770563	0
Platform_G	0.998274	0	0.999698	0	0.997044	0	0.999302	0	0.997659	0
Platform_H	0.904600	0	0.991070	0	0.997183	0	0.991548	0	0.948638	0
Platform_I	1	0	1	0	0.994675	0	0.999816	0	0.997330	0
Platform_J	0.979096	0	0.995861	0	0.918212	0	0.982325	0	0.947678	0
Fake	0.974186	0	0.999125	0	0.920467	0	0.996399	0	0.946565	0
Mean	0.861549	0	0.994473	0	0.936324	0	0.989444	0	0.879307	0

TABLE XXIII. Evaluation metrics of the boosted tree ensemble classifier on the descriptor validation set.

Channels	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_A	0.988983	0.000676	0.998087	0.000119	0.996375	0.000575	0.997836	0.000120	0.992665	0.000407
Platform_B	0.932667	0.015727	0.999180	0.000207	0.991531	0.001835	0.999094	0.000205	0.961131	0.008426
Platform_C	0.954354	0.006893	0.998883	0.000178	0.979831	0.002443	0.998440	0.000171	0.966910	0.003522
Platform_D	0.995414	0.000676	0.999120	0.000130	0.996482	0.000611	0.998696	0.000156	0.995948	0.000485
Platform_E	0.999061	0.000391	0.999808	0.000080	0.989908	0.000691	0.998114	0.000143	0.994463	0.000422
Platform_F	0.870819	0.013219	0.997692	0.000270	0.983688	0.003476	0.997474	0.000268	0.923764	0.007508
Platform_G	0.999056	0.000366	0.999834	0.000064	0.999171	0.000278	0.999735	0.000074	0.999113	0.000247
Platform_H	0.993373	0.000990	0.999435	0.000085	0.996823	0.000936	0.999231	0.000095	0.995095	0.000602
Platform_I	1	0	1	0	0.995030	0.000511	0.999829	0.000018	0.997509	0.000257
Platform_J	0.997467	0.000637	0.999478	0.000132	0.973848	0.001559	0.995010	0.000301	0.985515	0.000882
Fake	0.962761	0.004676	0.998626	0.000179	0.989148	0.002374	0.998297	0.000197	0.975770	0.002759
Mean	0.972178	0.004023	0.999104	0.000131	0.990167	0.001390	0.998341	0.000159	0.980717	0.002320

TABLE XXIV. Evaluation metrics of the tree-assembled classifier using bagging on the descriptor validation set.

Channels	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_A	0.989188	0.000575	0.998123	0.000101	0.996467	0.000759	0.997880	0.000135	0.992814	0.000460
Platform_B	0.937946	0.012576	0.999249	0.000163	0.991748	0.001861	0.999165	0.000164	0.964055	0.006778
Platform_C	0.955953	0.008849	0.998923	0.000227	0.979726	0.002119	0.998476	0.000209	0.967669	0.004281
Platform_D	0.995475	0.000719	0.999131	0.000139	0.996566	0.000541	0.998719	0.000111	0.996020	0.000343
Platform_E	0.999069	0.000386	0.999809	0.000079	0.990244	0.000777	0.998173	0.000132	0.994637	0.000388
Platform_F	0.872949	0.012583	0.997738	0.000255	0.982821	0.003773	0.997506	0.000262	0.924585	0.007407
Platform_G	0.999146	0.000364	0.999850	0.000064	0.999187	0.000349	0.999751	0.000080	0.999167	0.000268
Platform_H	0.993253	0.001521	0.999425	0.000131	0.997277	0.000657	0.999257	0.000128	0.995260	0.000810
Platform_I	1	0	1	0	0.994959	0.000681	0.999826	0.000024	0.997473	0.000342
Platform_J	0.997483	0.000537	0.999481	0.000111	0.974564	0.002099	0.995137	0.000368	0.985889	0.001082
Fake	0.963770	0.005435	0.998664	0.000209	0.989325	0.001646	0.998340	0.000213	0.976372	0.002975
Mean	0.973112	0.003959	0.999127	0.000134	0.990262	0.001387	0.998385	0.000166	0.981267	0.002285

TABLE XXV. Evaluation metrics of discriminant analysis in subspaces on the descriptor validation set.

Channels	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_A	0.993694	0.000220	0.999167	0.000028	0.761617	0.003931	0.964251	0.000580	0.862308	0.002543
Platform_B	0.286337	0.005471	0.972181	0.000798	0.977742	0.004110	0.972243	0.000769	0.442920	0.006430
Platform_C	0.434852	0.027867	0.971653	0.003502	0.908794	0.007958	0.970191	0.003337	0.587650	0.025303
Platform_D	0.979539	0.001090	0.996364	0.000238	0.907469	0.016227	0.982063	0.002477	0.942049	0.008587
Platform_E	0.986911	0.000962	0.997284	0.000202	0.991905	0.000430	0.996363	0.000189	0.989401	0.000547
Platform_F	0.887974	0.012041	0.998547	0.000176	0.728526	0.010897	0.994350	0.000240	0.800316	0.008605
Platform_G	0.951280	0.007304	0.991318	0.001388	0.965361	0.003053	0.987445	0.000924	0.958245	0.002895
Platform_H	0.928862	0.007925	0.993564	0.000783	0.988451	0.001963	0.993164	0.000611	0.957707	0.003562
Platform_I	0.989386	0.001590	0.999618	0.000058	0.995740	0	0.999484	0.000056	0.992552	0.000800
Platform_J	0.988847	0.001792	0.997775	0.000361	0.934205	0.001514	0.986693	0.000392	0.960748	0.001153
Fake	0.989163	0.002600	0.999640	0.000088	0.915730	0.002711	0.996732	0.000122	0.951027	0.001835
Mean	0.856077	0.006260	0.992465	0.000693	0.915958	0.004799	0.985725	0.000882	0.858629	0.005660

TABLE XXVI. Evaluation metrics of the k-nearest neighbors classifier in subspaces on the descriptor validation set.

Channels	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Platform_A	0.967001	0.000323	0.994173	0.000060	0.990958	0.001164	0.993701	0.000170	0.978833	0.000581
Platform_B	0.887431	0.011852	0.998564	0.000171	0.990554	0.000994	0.998474	0.000170	0.936121	0.006670
Platform_C	0.944029	0.002882	0.998611	0.000076	0.983939	0.002119	0.998269	0.000090	0.963568	0.001862
Platform_D	0.997199	0.000271	0.999476	0.000051	0.973024	0.000290	0.995221	0.000073	0.984963	0.000231
Platform_E	0.998774	0.000199	0.999749	0.000041	0.991482	0.000248	0.998334	0.000062	0.995115	0.000182
Platform_F	0.911636	0.005870	0.998519	0.000109	0.966982	0.003750	0.998029	0.000111	0.938479	0.003337
Platform_G	0.998458	0.000258	0.999729	0.000045	0.999491	0.000062	0.999694	0.000040	0.998974	0.000134
Platform_H	0.993758	0.000495	0.999467	0.000043	0.999077	0.000314	0.999437	0.000048	0.996410	0.000303
Platform_I	0.999252	0.000498	0.999973	0.000018	0.995740	0	0.999827	0.000017	0.997493	0.000248
Platform_J	0.995081	0.000489	0.998978	0.000102	0.979251	0.000547	0.995539	0.000126	0.987102	0.000364
Fake	0.965281	0.002442	0.998732	0.000092	0.981654	0.001765	0.998141	0.000116	0.973397	0.001642
Mean	0.968900	0.002325	0.998725	0.000073	0.986559	0.001023	0.997697	0.000093	0.977314	0.001414

TABLE XXVII. Comparison of model evaluation metrics on the mosaic validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Models	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
SVM linear	0.956700	0.001174	0.998095	0.000052	0.973965	0.000934	0.996420	0.000085	0.964171	0.000916
SVM quadratic	0.985570	0.000833	0.999344	0.000037	0.991243	0.000566	0.998771	0.000054	0.988318	0.000569
SVM cubic	0.988132	0.000657	0.999438	0.000033	0.992528	0.000420	0.998952	0.000043	0.990280	0.000430
SVM fine Gaussian	0.953102	0	0.996565	0	0.959001	0	0.993591	0	0.953514	0
SVM mean Gaussian	0.962762	0	0.998402	0	0.978395	0	0.997010	0	0.969724	0
SVM thick Gaussian	0.895389	0	0.994843	0	0.929167	0	0.990353	0	0.906682	0
Ensemble boosted trees	0.663536	0	0.974723	0	0.705645	0	0.953527	0	0.663731	0
Ensemble bagged trees	0.946695	0.002163	0.997075	0.000124	0.966033	0.001350	0.994588	0.000153	0.955628	0.001402
Ensemble subspace discriminant	0.862430	0.002606	0.992630	0.000201	0.897664	0.001702	0.986128	0.000231	0.870853	0.001863
Ensemble subspace KNN	0.941147	0.001711	0.997048	0.000102	0.968163	0.001090	0.994519	0.000146	0.952976	0.001248

TABLE XXVIII. Comparison of model evaluation metrics on the descriptor validation set.

	Precision		Specificity		Sensitivity		Accuracy		F1 Measurement	
Models	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
SVM linear	0.973681	0.002445	0.998898	0.000058	0.989418	0.000570	0.997981	0.000092	0.981164	0.001464
SVM quadratic	0.980714	0.001799	0.999240	0.000090	0.991674	0.000902	0.998611	0.000135	0.986013	0.001153
SVM cubic	0.985909	0.000862	0.999383	0.000042	0.993609	0.000406	0.998883	0.000052	0.989694	0.000514
SVM fine Gaussian	0.905925	0	0.994351	0	0.939775	0	0.989143	0	0.909508	0
SVM mean Gaussian	0.967522	0	0.998780	0	0.988370	0	0.997762	0	0.977390	0
SVM thick Gaussian	0.861549	0	0.994473	0	0.936324	0	0.989444	0	0.879307	0
Ensemble boosted trees	0.972178	0.004023	0.999104	0.000131	0.990167	0.001390	0.998341	0.000159	0.980717	0.002320
Ensemble bagged trees	0.973112	0.003959	0.999127	0.000134	0.990262	0.001387	0.998385	0.000166	0.981267	0.002285
Ensemble subspace discriminant	0.856077	0.006260	0.992465	0.000693	0.915958	0.004799	0.985725	0.000882	0.858629	0.005660
Ensemble subspace KNN	0.968900	0.002325	0.998725	0.000073	0.986559	0.001023	0.997697	0.000093	0.977314	0.001414

In addition to the metrics of accuracy, sensitivity, specificity, precision, and F1 measure, the inference time of the SVM classifier with cubic kernel was preliminarily evaluated. The results indicated that this algorithm had an average inference time of less than two seconds. This performance suggests that the algorithm is fast enough to be used in real-time applications on multimedia platforms.

The results in Tables XXVII and XXVIII highlight key differences in model performance across the mosaic and descriptor validation sets. Overall, SVM-based models demonstrated consistently high accuracy, precision, and sensitivity, particularly the cubic and quadratic variants, which achieved the best performance across both datasets.

On the mosaic validation set (Table XXVII), models such as SVM cubic and SVM quadratic achieved precision and sensitivity values above 0.98, indicating strong classification performance. The ensemble-based methods, particularly boosted trees, exhibited lower sensitivity, suggesting a trade-off between specificity and recall.

In contrast, on the descriptor validation set (Table XXVIII), performance remained high across most models, with ensemble methods, such as boosted trees and subspace KNN, showing competitive results. The improved F1-score of these models suggests that ensemble techniques may generalize better for this dataset.

The differences in model performance across datasets emphasize the importance of selecting the appropriate classification model based on the data characteristics. While SVM models excel in maintaining high sensitivity and precision, ensemble methods may provide a more balanced performance depending on the dataset. These findings highlight the need to consider model selection for different validation scenarios carefully.

It is important to note that the images used to evaluate the presented methodological approach were not part of the set of images previously used for training the different artificial intelligence models. This allows for assessing the capacity to generalize these to new data. In both training and test groups, 30 executions of the algorithms were carried out, which allowed for calculating and presenting the average value and standard deviation for the validation set.

This study aligns with previous research on image classification using artificial intelligence algorithms. In the work presented by [8], where analyzed histopathological images, it was found that support vector machines and boosted trees proved to be the most effective of the classification algorithms evaluated. In particular, the combination of SFTA and boosted trees achieved an accuracy of 94.3%, the most successful combination [10]. Developed a bag of LBP features. They trained several machine learning models and obtained the best performance of the SVM, with an average accuracy of 81.7% in the different data sets used.

The results obtained in this study have significant practical implications for classifying images from multimedia platforms. Classification is essential to improving the browsing experience and content recommendation for users. By having accurate classification systems, platforms can offer more personalized and relevant suggestions, facilitating navigation, the discovery of new titles, and the expansion of advertising with ads or sponsored content. However, it is important to consider that this study focused on a specific set of images and platforms.

If we want a more comprehensive and generalized understanding, further research and exploration of different datasets, platforms, and types of multimedia-related images are necessary. Implementing an online ranking system could be explored, where images are ranked as they are added to platforms. This would allow for a constant update of the recommendation and navigation systems. Furthermore, evaluating additional feature extraction techniques and integrating deep learning models such as convolutional neural networks would be possible. In this way, we could thoroughly compare all possible solutions with the results obtained in this study.

Beyond the models evaluated in this study, alternative approaches such as Least Squares Support Vector Machines (LS-SVM), Linear Programming Support Vector Machines (LP-SVM), Robust Vector Machines (RVM), Bayesian Support Vector Machines (B-SVM), and Committee Machines could serve as additional points of comparison. These models offer various advantages: LS-SVM improves computational efficiency, LP-SVM and RVM enhance robustness to noise, B-SVM provides probabilistic insights, and Committee Machines leverage ensemble strategies for improved classification performance. Future work could explore these approaches to broaden the comparative analysis and assess their effectiveness in different validation scenarios.

These findings emphasize the importance of model selection based on dataset characteristics, as different models may excel in different aspects of classification performance.

5. Conclusions

The results of this research confirm the effectiveness of the proposed methodology for image classification in multimedia platforms. The use of the local binary pattern technique for feature extraction proved to be adequate since the generated vectors were effective for feeding the machine learning algorithms.

It was observed that support vector machine algorithms with cubic and quadratic kernels excelled in classifying mosaics and descriptors, while the thick Gaussian kernel had the lowest performance. This confirms the importance of choosing the appropriate algorithm for each type of data and the value of parameter optimization, such as the kernel in SVMs.

In the assembled classifiers, the bagging algorithm achieved outstanding performance in both data sets, contrary to the behavior of the boosted trees that showed the lowest performance. This reinforces that model assembly can improve the robustness of the classification system. The evaluation of the models on a separate dataset, not previously used in training, demonstrated the generalization capacity of the models to new data. This is crucial to ensure that classification systems are effective in real-world deployments.

This work can complement and improve content recommendation systems on multimedia platforms. Its accuracy in content classification, ability to adapt in real-time, optimization of marketing and content strategies, user experience improvement, scalability, and efficiency highlight its practical and commercial relevance.

Future research is suggested to explore different combinations of algorithms, datasets, and feature extraction techniques for image classification from such platforms.

Acknowledgements

We thank technician Rosa Lilia Pérez-Arce, technical of the Applied Physics Division, CICESE, for creating Fig. 4.

Conflicts of interest

The authors declare no conflict of interest related to this study.

Funding

This research was funded by Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE), Baja California, grant number F0F181.

Data statement

The data supporting is not publicly available, but they can be shared if you send an Email to the corresponding author.

1. M. Jenner, Netflix and the Re-invention of Television / Mareike Jenner. Cham: Palgrave Macmillan (in English), 2023, `https://doi.org/10.1007/978-3-031-39237-5`.
2. D. Jannach, A. Manzoor, W. Cai, and L. Chen, A Survey on Conversational Recommender Systems, *ACM Comput. Surv.*, **54** (2021) `https://doi.org/10.1145/3453154`.
3. E. C. Malthouse and H. Li, Opportunities for and Pitfalls of Using Big Data in Advertising Research, *Journal of Advertising*, **46** (2017) 227, `https://doi.org/10.1080/00913367.2017.1299653`.
4. O. Castillo, P. Melin, and SpringerLink, Hybrid Intelligent Systems in Control, Pattern Recognition and Medicine, 1st 2020. ed. (Studies in Computational Intelligence 827). Cham: Springer International Publishing: Imprint: (Springer in English, 2020), `https://doi.org/10.1007/978-3-030-34135-0`.
5. A. A. Hopgood, Intelligent systems for engineers and scientists : a practical guide to artificial intelligence, Fourth edition. ed. (An Auerbach Book). (CRC Press in English, 2022), `https://doi.org/10.1201/9781003226277`.
6. W. Du and Z. Zhan, Building decision tree classifier on private data, *Electrical Engineering and Computer Science - All Scholarship.* **8** (2002) `https://surface.syr.edu/eecs/8`.
7. D. Lu and Q. Weng, A survey of image classification methods and techniques for improving classification performance, *International Journal of Remote Sensing*, **28** (2007) 823, `https://doi.org/10.1080/01431160600746456`.
8. Ş. Öztürk and B. Akdemir, Application of Feature Extraction and Classification Methods for Histopathological Image using GLCM, LBP, LBGLCM, GLRLM and SFTA, *Procedia Computer Science*, **132** (2018) 40, `https://doi.org/10.1016/j.procs.2018.05.057`.
9. S. Al-Jumaili, A. Al-Azzawi, A. D. Duru, and A. A. Ibrahim, Covid-19 X-ray image classification using SVM based on Local Binary Pattern, in 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies *(ISMSIT)*, **21-23** (2021) 383, `https://doi.org/10.1109/ISMSIT52890.2021.9604731`.
10. D. Srivastava, R. Bakthula, and S. Agarwal, Image classification using SURF and bag of LBP features constructed by clustering with fixed centers, *Multimedia Tools and Applications*, **78** (2019) 14129, `https://doi.org/10.1007/s11042-018-6793-8`.
11. S. Veerashetty and N. B. Patil, Novel LBP based texture descriptor for rotation, illumination and scale invariance for image texture analysis and classification using multi-kernel SVM, *Multimedia Tools and Applications*, **79** (2020) 9935, `https://doi.org/10.1007/s11042-019-7345-6`.
12. L. Nanni, A. Lumini, and S. Brahnam, Survey on LBP based texture descriptors for image classification, *Expert Systems with Applications*, **39** (2012) 3634, `https://doi.org/10.1016/j.eswa.2011.09.054`.
13. B. Yang and S. Chen, A comparative study on local binary pattern (LBP) based face recognition: LBP histogram versus LBP image, *Neurocomputing*, **120** (2013) 365, `https://doi.org/10.1016/j.neucom.2012.10.032`.
14. C. Shan and T. Gritti, Learning Discriminative LBP-Histogram Bins for Facial Expression Recognition, in British Machine Vision Conference, (2008).
15. E. Guerra-Rosas, L. F. L ópez- ávila, E. Garza-Flores, C. A. Vidales-Basurto, and J. álvarez-Borrego, Classification of Skin Lesion Images Using Artificial Intelligence Methodologies through Radial Fourier- Mellin and Hilbert Transform Signatures, *Applied Sciences*, **13** `https://doi.org/10.3390/app132011425`.
16. R. A. Fisher, The use of multiple measurements in Taxonomic problems, *Annals of Eugenics*, **7** (1936) 179, `https://doi.org/10.1111/j.1469-1809.1936.tb02137.x`.
17. Y. Lu and Q. Tian, Discriminant Subspace Analysis: An Adaptive Approach for Image Classification, *IEEE Transactions on Multimedia*, **11** (2009) 1289, `https://doi.org/10.1109/TMM.2009.2030632`.

18. X. Zhang and Y. Jia, A linear discriminant analysis framework based on random subspace for face recognition, *Pattern Recognition*, **40** (2007) 2585, https://doi.org/10.1016/j.patcog.2006.12.002.

19. G. Batista and D. F. Silva, How k-nearest neighbor parameters affect its performance, in Argentine symposium on artificial intelligence, 2009: Citeseer, pp. 1-12, https://api.semanticscholar.org/CorpusID:16606615.

20. M. Steinbach and P.-N. Tan, kNN: k-Nearest Neighbors, in The Top Ten Algorithms in Data Mining, V. K. Xindong Wu Ed.: Chapman and Hall/CRC, 2009, ch. kNN: k-Nearest Neighbors, pp. 20-28.

21. S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, Learning k for kNN Classification, *ACM Trans. Intell. Syst. Technol*., **8** (2017) 43, https://doi.org/10.1145/2990508.

22. B. Taha Jijo and A. Mohsin Abdulazeez, Classification Based on Decision Tree Algorithm for Machine Learning, *Journal of Applied Science and Technology Trends*, **2** (2021) 20, https://doi.org/10.38094/jastt20165.

23. Y. Y. Song and Y. Lu, Decision tree methods: applications for classification and prediction, (in eng), *Shanghai Arch Psychiatry*, **27** (2015) 130, https://doi.org/10.11919/.issn.1002-0829.215044.

24. S. Suthaharan and SpringerLink, Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning, 1st 2016. ed. (Integrated Series in Information Systems 36). Boston, MA: (Springer US in English, 2015), https://dx.doi.org/10.1007/978-1-4899-7641-3.

25. E. Campo Le ón, Introducci ón a las m áquinas de vector soporte (SVM) en aprendizaje supervisado, Matem áticas, Facultad de Ciencias, Universidad de Zaragoza, (Zaragoza, España, 2016).

26. P. S. Sastry, An introduction to support vector machines, in Computing and Information Sciences: Recent Trends, J. C. Misra Ed.: Alpha Science Int'l Ltd., 2003, ch. 3, pp. 53-85.

27. H. Xue, Q. Yang, and S. Chen, SVM: Support vector machines, in The Top Ten Algorithms in Data Mining, V. K. Xindong Wu Ed.: Chapman and Hall/CRC, 2009, ch. SVM: Support vector machines, pp. 37-60.

28. C. Angulo Bah ón, Aprendizaje con m áquinas núcleo en entornos de multiclasificación, P. H., Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial, Universitat Politècnica de Catalunya, (Catalunya, 2001), https://hdl.handle.net/10803/6178.

29. F. F. Chamasemani and Y. P. Singh, Multi-class Support Vector Machine (SVM) Classifiers – An Application in Hypothyroid Detection and Classification, in 2011 Sixth *International Conference on Bio- Inspired Computing: Theories and Applications*, **27-29** (2011) 351, https://doi.org/10.1109/BIC-TA.2011.51.

30. O. Sagi and L. Rokach, Ensemble learning: A survey, *WIREs Data Mining and Knowledge Discovery, 8* (2018) e1249, https://doi.org/10.1002/widm.1249.

31. R. Polikar, A. J. Ferreira, M. A. T. Figueiredo, M. Di Marzio, and C. C. Taylor, Ensemble Learning Boosting Algorithms: A Review of Methods, Theory, and Applications Boosting Kernel Estimators, in Ensemble Machine Learning Methods and Applications, C. Zhang and Y. Ma Eds.: (Springer New York, NY, 2012) 1, https://doi.org/10.1007/978-1-4419-9326-7.

32. B. Abdualgalil and S. Abraham, Applications of Machine Learning Algorithms and Performance Comparison: A Review, in 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), **24-25** (2020) 1-6, https://doi.org/10.1109/ic-ETITE47903.2020.490.

33. R. Borja-Robalino, A. Monleon-Getino, and J. Rodellar, Estandarizaci ón de m étricas de rendimiento para clasificadores machine y deep learning, presented at the VI Congreso Internacional de Ciencia, Tecnolog ía e Innovaci ón para la Sociedad (CITIS), Guayaquil Ecuador, (2020) https://www.researchgate.net/publication/339943922.

34. S. Yadav and S. Shukla, Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification, in 2016 IEEE 6th International Conference on Advanced Computing *(IACC)*, **27-28** (2016) 78, https://doi.org/10.1109/IACC.2016.25.

35. C. A. Ramezan, T. A. Warner, and A. E. Maxwell, Evaluation of Sampling and Cross-Validation Tuning Strategies for Regional-Scale Machine Learning Classification, Remote Sensing, **11** https://doi.org/10.3390/rs11020185.