Data Analysis for Medical Applications

Jose A. Govea-Garcia[1], Augusto Ley Rodriguez[2] and Angel Esparza Enriquez[3]

[1] *Tecnologico de Monterrey, Escuela de Ingenieria y Ciencia, Jalisco, Mexico*

Reception date of the manuscript: 10/05/2025
Acceptance date of the manuscript: 10/05/2025
Publication date: 10/05/2025

Abstract— This project explores the use of built-in mobile phone sensors to recognize physical activities in real time using supervised classification models. Motion signals were collected from various participants performing a series of exercises, and several machine learning models were trained to predict the corresponding activity. Throughout the project, techniques such as cross-validation, hyperparameter optimization, and feature selection were applied to improve classifier performance. The final version was integrated into a functional application that receives real-time data from a mobile device. The project concludes that a proper combination of models, fine-tuning, and variable reduction is key to implementing efficient and stable solutions in practical contexts.

Keywords—Modelos de clasificacion, evaluacion de modelos, recoleccion de datos, optimizacion, aplicaciones medicas

I. Introducción

In recent years, the use of embedded sensors in devices such as watches, phones, earbuds, or rings has gained significant importance. This technology has opened up a wide range of opportunities for monitoring human activity, allowing individuals, coaches, and health specialists to access valuable information for planning and executing physical activities, tracking health status, or addressing any other aspect related to well-being.

Commercial applications like Google Fit, Apple Health, or Fitbit already leverage these sensors to identify movement patterns and classify activities such as walking, running, or climbing stairs. However, these solutions are often limited to a small set of predefined actions. Recent studies highlight the value of customizing these tools for specific purposes, which paves the way for more specialized solutions developed by users or researchers.[1].

The objective of this work is to leverage the sensors integrated into mobile phones to detect, in real time, the type of physical activity being performed. To achieve this, we will use classification models that allow us to label the collected data according to the activity. These models will be trained and optimized so that, once deployed, they can accurately assign the correct activity being carried out.

The development of the project will be divided into the following stages:

1. Data collection
2. Evaluation of classification models
3. Model optimization
4. Online classification
5. Report writing

Contact data: Jose A. Govea, goveaangel090@gmail.com

II. Methodology

This section describes the complete process followed to develop the physical activity classification system — from data collection using mobile phone sensors to the training, evaluation, and optimization of various models. The methodology is divided into four main stages: data collection, processing and feature extraction, model evaluation, and optimization with feature selection.

a. Data collection

The first step we carried out was data collection. For this purpose, we used the Phyphox application, which allows access to various sensors integrated into the mobile phone. To streamline and automate the data collection process, we developed a Python script that enabled us to configure all the necessary parameters of the experiment. Additionally, we leveraged Phyphox's real-time IP transmission feature, which allowed the script to retrieve data in object format and store it directly.

First, we defined the conditions of the experiment, selecting six different activities:

1. Jumping jacks
2. Waving
3. Jump rope
4. Boxing
5. Torso rotation
6. Standing still

Next, we configured the necessary parameters for the experiment, including the number of trials per activity and the number of samples recorded in each trial. Once the param-

eters were defined, we proceeded with the data collection. Each team member carried out the experiment in two separate sessions, for reasons that will be explained later in this report.

In the first session, only the accelerometer was used, capturing data along the X, Y, and Z axes. In the second session, in addition to the accelerometer, gyroscope sensors were also used, again capturing data on the X, Y, and Z axes.

b. Processing and Extracted Features

Una vez recolectados los datos, se procedió con su procesamiento previo a la etapa del modelado. Para cada ventana de señal, se extrajeron múltiples características que nos permiten resumir el comportamiento del movimiento en esa ventana de tiempo. Estas características permiten mejorar la capacidad del modelo para diferenciar entre actividades.

Las siguientes características fueron extraídas para cada eje:

1. Mean
2. Standard deviation
3. Kurtosis
4. Skewness
5. Median
6. Maximum value
7. Minimum value
8. Peak-to-peak range
9. Interquartile range (IQR)
10. Zero crossing rate
11. Energy
12. Contribution to overall RMS

The following features were extracted from the acceleration magnitude and gyroscope magnitude:

1. Mean
2. Standard deviation
3. Energy

A Python script was used to extract these features. The script opens the .obj file, computes the features for each signal window, and outputs the results in a .txt file.

c. Evaluation of classification models

With the features computed, the next step in the project was the evaluation of classification models. Initial cleaning tasks were carried out, such as removing incomplete or blank records. Afterwards, the dataset was split into independent variables (X) and the target variable (Y).

Before training the models, a data standardization process was applied using standard scaling to ensure that all features were on the same scale, which is especially important for models sensitive to variable magnitude.

Model evaluation was performed using cross-validation to take full advantage of the dataset and ensure more robust results. Specifically, metrics such as accuracy, precision, and recall were used to assess the performance of the various classifiers.

In total, 10 classification models were evaluated, including five models covered in class:

1. SVM
2. SVM with radial basis function (RBF) kernel
3. LDA
4. K-NN
5. MLP (2 layers)

And five additional models were researched and tested independently:

1. Extra Trees
2. Random Forest
3. Logistic Regression
4. Gradient Boosting
5. Naive Bayes

To carry out the evaluation of the classification models, a Python notebook was developed where the previously mentioned algorithms were implemented. Among the models tested, four representative algorithms were selected due to their theoretical significance and methodological diversity: SVM, K-NN, MLP, and Naive Bayes. Each of these offers a different approach to tackling the classification problem.

SVM works by projecting the data into a higher-dimensional space, which allows for the separation of classes that are not linearly distinguishable in their original form. In this new space, it identifies a hyperplane that acts as a boundary between categories. Once trained, the model can use the features of new data to predict which class they belong to.[2].

TABLE 1: REPORTE DE CLASIFICACIÓN PARA SVM (RBF)

Clase	Precision	Recall	F1-Score
1	0.97	0.99	0.98
2	1.00	0.97	0.98
3	1.00	1.00	1.00
4	1.00	0.99	0.99
5	0.97	0.94	0.95
6	0.95	1.00	0.98
Accuracy			**0.98**
Macro avg	0.98	0.98	0.98
Weighted avg	0.98	0.98	0.98

The SVM model with a radial kernel showed highly consistent performance, achieving F1-scores close to or equal to 1.00 for most classes, with an overall accuracy of 98%.

K-NN classifies a new record by comparing it to the closest examples in the training set. The decision is based on the majority class among its k nearest neighbors, determined using a distance metric. This model does not require a complex training process and is particularly useful when seeking an intuitive, similarity-based solutions. [3].

TABLE 2: REPORTE DE CLASIFICACIÓN PARA K-NN

Clase	Precision	Recall	F1-Score
1	0.99	0.99	0.99
2	1.00	0.98	0.99
3	0.99	1.00	1.00
4	1.00	0.99	0.99
5	0.98	0.94	0.96
6	0.94	1.00	0.97
Accuracy			**0.98**
Macro avg	0.98	0.98	0.98
Weighted avg	0.98	0.98	0.98

K-NN achieved outstanding accuracy across all classes, with particularly strong performance in classes 3 and 6, reaching a perfect F1-score (1.00) in some cases. The overall accuracy was also 98

The Multilayer Perceptron (MLP) is an artificial neural network composed of at least one hidden layer between the input and output. Each layer consists of units called neurons, which are fully connected to the next layer. The MLP trains the model using the backpropagation algorithm, adjusting internal weights to minimize the difference between predictions and true labels. This architecture enables the model to learn complex, non-linear relationships between input and output variables. [4].

TABLE 3: REPORTE DE CLASIFICACIÓN PARA MLP (2 CAPAS)

Clase	Precision	Recall	F1-Score
1	0.98	0.99	0.99
2	1.00	0.97	0.98
3	0.99	1.00	1.00
4	0.99	0.98	0.98
5	0.96	0.95	0.95
6	0.95	0.98	0.97
Accuracy			**0.98**
Macro avg	0.98	0.98	0.98
Weighted avg	0.98	0.98	0.98

The multilayer perceptron demonstrated solid performance, with balanced results across classes and a maximum F1-score of 1.00 in class 3. The overall accuracy of the model was 98

The Naive Bayes classifier is an algorithm based on Bayes' Theorem that assumes statistical independence between features. It estimates the probability that a data point belongs to a specific class by considering each attribute separately, making it efficient and scalable even with large or high-dimensional datasets. [5].

TABLE 4: REPORTE DE CLASIFICACIÓN PARA NAIVE BAYES

Clase	Precision	Recall	F1-Score
1	0.98	0.99	0.99
2	1.00	0.98	0.99
3	1.00	1.00	1.00
4	1.00	1.00	1.00
5	0.98	0.95	0.96
6	0.95	0.99	0.97
Accuracy			**0.98**
Macro avg	0.99	0.98	0.99
Weighted avg	0.99	0.98	0.99

The Naive Bayes model achieved an accuracy of 98% and F1-scores of 1.00 in several classes, standing out for its efficiency and generalization capability.

d. Optimization and Feature Selection

Once the overall performance of the classification models was evaluated, the next step was to optimize those that showed the best results. In this stage, key hyperparameters were tuned, and feature selection techniques were applied with the goal of improving performance and reducing model complexity. The models selected for optimization were the radial SVM and MLP.

These two models were chosen due to their high performance. For both, the most relevant hyperparameters were identified, and different combinations were tested using cross-validation to evaluate the performance of each configuration.

As the first step in the optimization process, we focused exclusively on tuning the hyperparameters of the selected models, leaving feature selection for a later stage.

For the SVM model with a radial basis function (RBF) kernel, different values of the parameter C were explored. The best performance was observed with C = 0.1. However, the variations in accuracy across configurations were minimal, all ranging between approximately 98.3% and 98.5%.

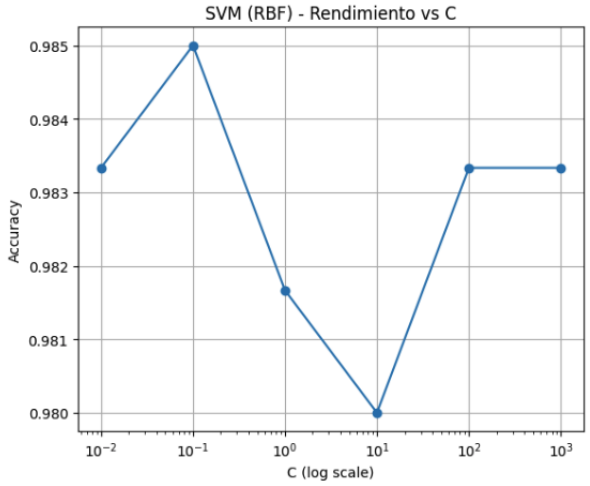

Fig. 1: Performance of the SVM Model with RBF Kernel for Different Hyperparameter Values *C*.

In the case of the Multilayer Perceptron (MLP), five different hidden layer configurations were tested. Surprisingly, the simplest architecture, (10,), delivered the highest accuracy. However, the (50,) configuration, which initially yielded slightly lower performance, was ultimately selected for the live demonstration.

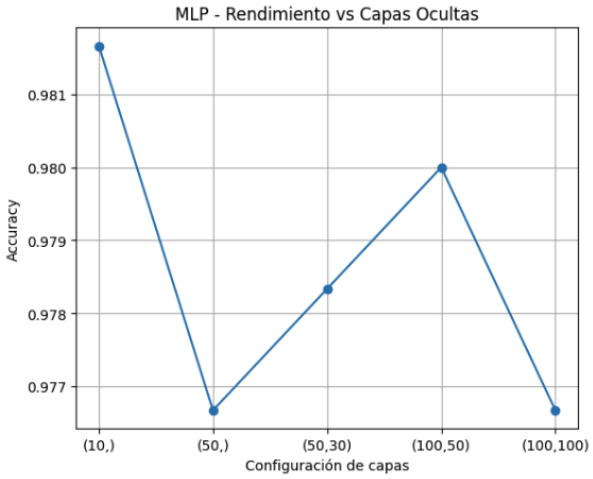

Fig. 2: Performance of the MLP Model with Different Hidden Layer Configurations

This decision was based on observations made during the real-time testing phase. Although the (10,) architecture achieved the highest accuracy in cross-validation, it exhibited inadequate behavior during the live demonstration: the model was unable to correctly recognize all six activities, limiting its predictions to only one or two classes. In contrast, the (50,) configuration—which had slightly lower accuracy (a difference of less than 0.2

As the second step of the optimization process, a univariate feature selection technique was applied using SelectKBest with the $f_classif$ scoring function. The objective was to identify the minimum number of features required to maximize the model's accuracy, thereby reducing computational complexity and potentially enhancing generalization by eliminating irrelevant or redundant features.

Various values of k (number of selected features) were evaluated: 5, 10, 15, 20, 25, 30, 35, and 39. For each k value, cross-validation was conducted using the previously optimized models: SVM with C = 10 and MLP with hidden layer configuration (50,).

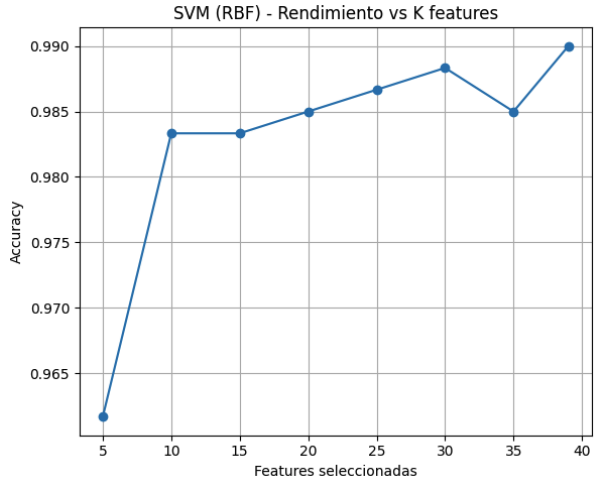

Fig. 3: Performance of the SVM (RBF) Model with Configuration $C = 10$ Using Different Numbers of Selected Features

In the case of SVM, performance consistently improved until reaching its peak with 39 features, achieving an accuracy of 99.0% (see Figure 3). However, very similar values were observed starting from 25 features, suggesting that dimensionality can be reduced without significantly affecting performance.

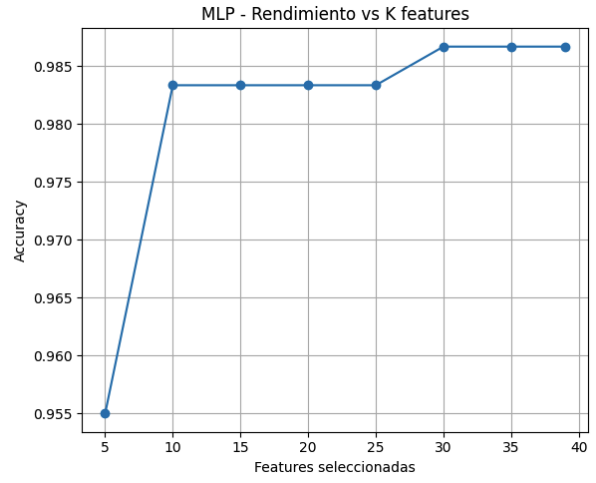

Fig. 4: Performance of the MLP Model with Hidden Layer Configuration $(50,)$ Using Different Numbers of Selected Features

On the other hand, the MLP model achieved its best performance with 30 or more features, maintaining a stable accuracy of 98.6% for any higher value (see Figure 4). Notably, even with only 10 features, the model already achieved an accuracy above 98.3%, highlighting the algorithm's ability to adapt to lower-dimensional spaces without losing generalization capability.

These results demonstrate that both SVM and MLP can benefit from a reduction in the number of variables, as long as the most predictive features are retained. This not only improves efficiency but can also facilitate the practical implementation of the model.

e. App Model

For the real-time application implementation, an optimized model was built using a scikit-learn pipeline. This pipeline includes three main components: StandardScaler to normalize the data, SelectKBest to select the 10 most relevant features, and an MLPClassifier with a (50,) architecture.

A grid search (GridSearchCV) with stratified cross-validation was applied to optimize the hyperparameters: hidden layer sizes, alpha (regularization), and learning_rate_init. The best-performing model, with high average performance, was serialized using joblib for direct use in the application.

III. Results

The selected model was successfully integrated into the real-time classification application. During live testing, the system was able to accurately detect all six predefined exercises, with response times suitable for immediate use following data capture.

For this live version, a custom script was developed to receive data from the mobile device in real time, following a process similar to the original data collection. The signal was captured using the Phyphox application, which enabled direct acquisition of accelerometer and gyroscope data from the mobile device via local connection.

Although the MLP-based model with a single hidden layer of 50 neurons did not achieve the highest accuracy during cross-validation, it proved to be the most stable and reliable under real-world conditions. More complex configurations tended to overfit and, in live tests, failed to accurately distinguish the six exercises, often misclassifying multiple repetitions into only one or two classes.

In contrast, the selected model maintained near 98

It is important to note that while data was collected from all three team members during the experimental stage, the model used in the real-time implementation was specifically trained on the data of one individual. This is because, for the system to function ideally, the model must be trained with data representative of the person performing the activities live. Although the system retains some generalization capability, subtle differences in how exercises are executed between individuals may affect recognition accuracy. This observation highlights the importance of developing future versions of the model that include personalization mechanisms or more robust generalization strategies to broaden its applicability.

For more details, please refer to the project repository.

References

[1] Harvard Health Publishing. (2021) Exercising with your smartphone: How activity trackers and health apps help. https://www.health.harvard.edu/staying-healthy/exercising-with-your-smartphone-how-activity-trackers-and-health-apps-help. Consultado el 8 de junio de 2025.

[2] IBM Corporation. (2024) Cómo funciona svm (support vector machine). https://www.ibm.com/docs/es/spss-modeler/saas?topic=models-how-svm-works. Consultado el 8 de junio de 2025.

[3] I. Corporation. (2024) ¿qué es knn? https://www.ibm.com/mx-es/think/topics/knn. Consultado el 8 de junio de 2025.

[4] Scikit-learn Developers, *Neural network models (supervised)*, https://scikit-learn.org/stable/modules/neural_networks_supervised.html#mlp, scikit-learn, 2024, consultado el 8 de junio de 2025.

[5] IBM Corporation. (2024) Naive bayes. https://www.ibm.com/think/topics/naive-bayes. Consultado el 8 de junio de 2025.