Enhancing SVM and KNN Performance Through Preprocessing Pipelines for Interactive mHealth Applications

Btissam Elaziz[1], Charaf Eddine AIT ZAOUIAT[2], Mohamed Eddabbah[3]*, Yassin LAAZIZ[4]
LabTIC-ENSA of Tangier, Abdelmalek Essaadi University, Morocco[1, 4]
Polydisciplinary Faculty of Sidi Bennour, Chouaîb Doukkali University, Morocco[2]
Higher School of Technology – Essaouira, Cadi Ayyad University, Marrakesh, Morocco[3]

***Abstract*—Mobile health (mHealth) applications are increasingly relying on artificial intelligence (AI) to provide accurate and real-time decision support for healthcare delivery. However, achieving the optimal balance between processing time and accuracy remains challenging, especially for interactive applications that rely on cloud computing for scalability and performance. This study investigates the impact of data preprocessing techniques on the performance of two widely used machine learning algorithms, Support Vector Machine (SVM) and k-Nearest Neighbors (KNN), in cloud-based mHealth systems. We evaluate the effects of various scaling methods and dimensionality reduction techniques, on processing time and model accuracy. Our results demonstrate that preprocessing significantly improves model performance, with SVM achieving a precision of 0.72 and a processing time of 0.087 ms using StandardScaler, while KNN demonstrates the fastest processing times when paired with robust preprocessing. These findings underscore the importance of optimizing both data preparation and algorithmic efficiency for interactive mHealth applications. By enhancing model accuracy and reducing latency, this research contributes to the development of cost-effective, real-time mobile health systems that improve user experience and decision-making in healthcare.**

Keywords—Mobile health; cloud computing; machine learning; SVM; KNN; data preprocessing

I. Introduction

The rapid advancements in mobile technology and AI have revolutionized the healthcare sector, giving rise to Mobile Health (mHealth) applications [1]. These applications leverage the portability of mobile devices and the analytical power of AI to provide real-time health monitoring, personalized diagnostics, and predictive analytics [2]. mHealth solutions have become indispensable tools in addressing global healthcare challenges, offering services such as remote patient monitoring, chronic disease management, and telemedicine consultations [3]. At the core of this transformation is the reliance on AI algorithms, which analyze vast amounts of data to deliver actionable insights. However, the computational demands of these algorithms often exceed the capabilities of mobile devices, especially in resource-constrained environments [4].

To address this limitation, cloud computing has emerged as a crucial enabler, providing scalable, cost-effective platforms that offload complex computations from mobile devices to powerful cloud servers [5]. By utilizing cloud platforms, mHealth applications can perform sophisticated AI-driven tasks such as processing high-dimensional data, running predictive models, and delivering real-time decision support without compromising the user experience. This synergy between AI and cloud computing allows for greater efficiency, enabling mHealth applications to cater to the growing demand for fast, reliable, and accurate healthcare solutions [6]. As mHealth continues to expand its reach, optimizing the interplay between AI and cloud platforms will be pivotal in meeting the performance, scalability, and cost requirements of these applications.

In mHealth, real-time interactive mobile systems have become critical in delivering timely and accurate healthcare services, such as chronic disease management, remote patient monitoring, and emergency decision support [7]. These systems rely heavily on AI algorithms to process large volumes of health data, often under stringent time constraints. However, achieving the delicate balance between processing time and accuracy presents significant challenges. Mobile devices, constrained by limited computational resources, depend on cloud platforms to execute complex AI algorithms.

This dependency [8] introduces latency concerns due to data transmission, processing, and response times, which can compromise the system's responsiveness—an essential feature for real-time health applications. Simultaneously, ensuring high accuracy is vital, as errors in AI predictions can lead to incorrect diagnoses or treatment decisions, directly impacting patient safety and trust. Additionally, the performance of AI models is heavily influenced by data preprocessing techniques, which are necessary for improving model efficiency but can also increase computational overhead [9]. For interactive mHealth systems, the trade-offs between optimizing preprocessing pipelines, minimizing cloud processing costs, and maintaining real-time responsiveness remain poorly addressed. Thus, the key challenge lies in optimizing AI model performance to deliver accurate and timely results while ensuring scalability and cost-efficiency in cloud-based environments. Addressing these challenges is essential for advancing mHealth solutions and providing seamless, reliable healthcare services to users worldwide.

This study addresses the key challenges of deploying AI in mHealth by focusing on the optimization of Support Vector Machine (SVM) and k-Nearest Neighbors (KNN) algorithms in terms of data preprocessing, cloud integration, and real-time performance, with the goal of improving both patient outcomes and operational efficiency. SVM and KNN have been selected for their proven effectiveness in medical data classification, low implementation complexity, and robust performance on small to medium-sized datasets, which are common in mHealth applications. Their computational simplicity and compatibility with cloud-based architectures make them particularly suitable for real-time, scalable health systems that require both accuracy and efficiency. Moreover, the contrast between their learning paradigms—margin-based optimization for SVM versus instance-based learning for KNN—offers a valuable comparative framework for assessing preprocessing strategies and deployment trade-offs.

The primary aim of this study is to evaluate and optimize the performance of SVM and KNN algorithms for interactive mHealth AI systems deployed on cloud infrastructure. Specifically, the study seeks to:

- Analyze the Impact of Preprocessing Techniques: Investigate how various data preprocessing methods, such as scaling and dimensionality reduction, influence the accuracy and processing time of SVM and KNN models in a cloud-based environment.
- Optimize Algorithm Efficiency: Identify preprocessing pipelines and configurations that achieve the best trade-off between accuracy and processing time, ensuring the feasibility of real-time decision-making for mHealth applications.
- Enhance Scalability and Responsiveness: Explore how optimized AI models, combined with cloud computing resources, can improve the scalability and responsiveness of mHealth systems, ultimately reducing latency and enhancing user experience.

Through these objectives, the study aims to provide actionable insights for designing cost-effective, accurate, and efficient mHealth systems that improve user experience and leverage the power of cloud-based AI for real-time healthcare delivery.

This study is organized as follows: Following the introduction, the Section II provides a literature review to set the context. Section III presents the methodology. Section IVdiscusses the results; and Section V concludes the study.

II. Literature Review

AI applications in mHealth heavily rely on cloud computing to address the computational constraints of mobile devices, enabling advanced functionalities and seamless performance. Cloud platforms provide scalability, allowing them to process vast datasets and support a growing user base without compromising efficiency or performance. Additionally, cloud computing ensures secure data storage and facilitates easy access to patient information for authorized users, maintaining the privacy and integrity of sensitive health data [10]. These capabilities make cloud computing an indispensable backbone for AI-driven mHealth solutions, supporting their scalability, responsiveness, and security requirements.

SVM and KNN are widely used machine-learning algorithms in health- related research due to their simplicity, robustness, and effectiveness in handling diverse types of medical data [11]. SVM excels in analyzing high-dimensional datasets, a common characteristic of medical records, and performs reliably even with small training datasets, thanks to its ability to create clear class boundaries through kernel functions. SVM has been successfully applied in diagnosing diseases such as diabetes, cancer, and cardiovascular conditions, often achieving high accuracy, as demonstrated with the PIMA Indian Diabetes dataset. Its application extends to medical image analysis, including tumor detection, brain abnormalities, and retinal scans, and to genomic and proteomic data, where it identifies genetic markers and classifies diseases based on gene expression. However, while SVM is highly effective when dataset margins are clear, it may struggle with imbalanced datasets unless enhanced with techniques like cost-sensitive learning [12]. KNN, on the other hand, is a non-parametric algorithm that relies on the proximity of data points, making it particularly suitable for heterogeneous medical datasets. It is valued in clinical settings for its simplicity and interpretability, with applications including disease classification (e.g., diabetes, heart disease), medical decision support through patient record comparisons, and patient similarity assessments for personalized treatment planning. While KNN performs well with smaller datasets and clean, normalized data, it becomes computationally intensive as dataset sizes grow, requiring preprocessing and dimensionality reduction techniques like PCA or LDA [13]. When comparing the two, SVM generally outperforms KNN in high-dimensional and smaller datasets due to its mathematical rigor and margin maximization principle. However, KNN offers simplicity and ease of implementation, making it attractive for specific applications despite its higher computational demands for large datasets. Both algorithms benefit from preprocessing techniques, such as normalization and dimensionality reduction, which significantly enhance their performance and make them highly applicable in health AI applications. Together, SVM and KNN provide complementary strengths, supporting diverse healthcare tasks, from diagnostics to personalized medicine.

Research on SVM and KNN algorithms, and their optimization for mHealth applications challenges include the limited computational resources of mobile devices, the need for real-time processing, and the lack of detailed evaluations on integrating local preprocessing with cloud-based AI execution [14][15]. Existing studies also insufficiently address the application of preprocessing techniques to dynamic and heterogeneous mHealth datasets and overlook scalability and user experience for large user bases [16]. These gaps hinder the effective implementation of SVM and KNN in mHealth, where real-time accuracy, efficiency, and scalability are crucial [17].

III. Methodology

A. Dataset Description: PIMA Indian Diabetes Dataset

The PIMA Indian Diabetes dataset [18], sourced from the National Institute of Diabetes and Digestive and Kidney Diseases, is a widely used benchmark for machine learning studies in healthcare. It comprises medical data collected from 768 female patients of Pima Indian heritage, aged 21 years or older. The dataset aims to predict the presence of diabetes based on diagnostic measurements, making it particularly relevant for studies in mHealth applications. Table I summarizes the dataset's predictor variables.

TABLE I. THE DATASET INCLUDES THE FOLLOWING PREDICTOR VARIABLES AND TARGET VARIABLE

Feature	Description	Data Type
Pregnancies	Number of pregnancies.	int64
Glucose	Plasma glucose concentration (mg/dL) after a 2-hour oral glucose tolerance test.	int64
Blood Pressure	Diastolic blood pressure (mm Hg).	int64
Skin Thickness	Triceps skinfold thickness (mm).	int64
Insulin	2-hour serum insulin (mu U/mL).	int64
BMI	Body Mass Index (kg/m²).	float64
Diabetes Pedigree Function	A score representing the likelihood of diabetes based on family history.	float64
Age	Age of the individual.	int64
Outcome	Binary target variable indicating diabetes diagnosis (1 = diabetic, 0 = non-diabetic).	int64

B. *Data Preprocessing Techniques*

Preprocessing techniques are critical for improving the performance and efficiency of machine learning models as shown in Fig. 1.

This study utilizes various preprocessing methods, including scaling techniques and dimensionality reduction methods, to prepare the data for effective application of SVM and KNN algorithms. These techniques address challenges such as feature imbalance, computational efficiency, and overfitting, ensuring optimized performance in cloud-based mobile health mHealth applications [19].

1) *Data scaling techniques*

a) Standard Scaler (SS): The Standard Scaler technique is a data preprocessing method used to standardize the features of a dataset. It transforms the data such that it has zero mean and unit variance. This technique is commonly used in machine learning algorithms to ensure that all features are on a similar scale, which can improve the performance of the models. z is the standardized value of a feature; x is the original value of the feature; μ is the mean of the feature and σ is the standard deviation of the feature. A value x is transformed (scaled) into σ using Eq. (1):

$$z = \frac{x-\mu}{\sigma} \quad (1)$$

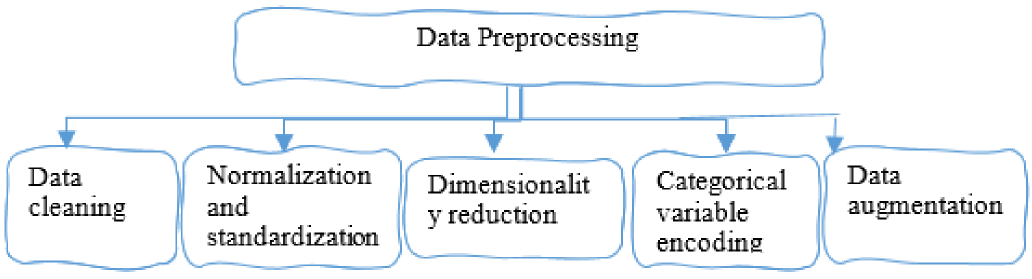

Data Preprocessing
Data cleaning
Normalization and standardization
Dimensionalit y reduction
Categorical variable encoding
Data augmentation

Fig. 1. Overview of various data preprocessing techniques.

b) Min-Max Scaler (MMS): Min-max scaler is a data normalization technique that transforms the values of a dataset to a specific range, typically between 0 and 1. It is calculated using Eq. (2):

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2)$$

where, X_{scaled} is the scaled value, X is the original value, X_{min} is the minimum value in the dataset, and X_{max} is the maximum value in the dataset.

c) Maximum Absolute Scaler (MAS): The Maximum Absolute Scaler is a method used to transform numerical data into a range between 0 and 1. This is achieved by dividing each value by the maximum absolute value present in the dataset. It can be mathematically expressed asashown in Eq. (3):

$$X_{scaled} = \frac{x}{max(|x|)} \quad (3)$$

where, x represents the original value and X_{scaled} represents the transformed value.

d) Robust Scaler (RS): The three scaling techniques mentioned earlier are highly affected by outliers because they rely on each variable's mean or minimum and maximum values. However, the Robust Scaler aims to reduce the impact of outliers by centering the data on the median $Q_2(x)$ and scaling it based on the interquartile range, which is the difference between the first quartile $Q_1(x)$ and the third quartile $Q_3(x)$ of x, as shown in Eq. (4):

$$X_{scaled} = \frac{X - Q_2(x)}{Q_3(x) - Q_1(x)} \quad (4)$$

e) Quantile Transformer (QT): The Quantile Transformer is a statistical technique that transforms a dataset's distribution to a desired target distribution by mapping original data values to new values. It is valuable for scenarios like data normalization, outlier mitigation, and preparation for statistical models. The Quantile Transformer can be represented by Eq. (5):

$$Y = (F_{target}^{-1}(F_{original}(x))) \quad (5)$$

where, $F_{original}$ is the cumulative distribution function (CDF) of the original dataset, and F_{target}^{-1} is the inverse CDF of the desired target distribution.

f) Normalization (N): A normalizer standardizes values to a uniform scale, aiding comparison by removing scale disparities, often expressed as Eq. (6). This equation scales values from 0 to 1, adjusting the minimum to 0 and the maximum to 1. Other techniques may employ varied equation depending on data needs.

$$Normalized\ value = \frac{Value - Minimum\ value}{Maximum\ value - Minimum\ value} \quad (6)$$

2) Dimension reduction techniques

a) Principal Component Analysis (PCA): PCA is a widely used statistical technique in multivariate analysis, primarily employed for dimensionality reduction and data compression [20]. Its main goal is to identify patterns, emphasizing similarities and differences in the data while preserving most of the original variations. PCA achieves four main objectives: extracting relevant information, compressing data by retaining distinguishing features, simplifying data description, and enabling analysis of observational structure. Eq. (7), which calculates variance (var(x)), is integral to PCA.

$$var(x) = \frac{\sum(x_i - \underline{x})^2}{N} \quad (7)$$

b) Linear Discriminant Analysis (LDA): Linear Discriminant Analysis (LDA) is a statistical technique for both dimensionality reduction and classification [21]. It aims to find a linear combination of features that maximizes class separability by maximizing the ratio of between-class and within-class scatters. LDA is widely applied in pattern recognition, machine learning, and data mining, seeking a projection vector that optimizes between-class scatter while minimizing within-class scatter. The between-class scatter matrix S_B is calculated by Eq. (8):

$$S_B = \big(mean(X_c) - mean(X)\big) * \big(mean(X_c) - mean(X)\big)^T \quad (8)$$

where, mean (X_c) is the mean vector of each class and mean(X) is the mean vector of the entire dataset.

The within-class scatter matrix S_w is calculated using Eq. (9):

$$S_w = \sum\big(X_i - mean(X_i)\big) * \big(X_i - mean(X_i)\big)^T \quad (9)$$

where, Σ represents the summation of all samples in each class.

The goal of LDA is to find the projection vector w that maximizes Fisher's criterion, as given in Eq. (10):

$$\mathrm{J(w) = (w^T * S_B * w)/(w^T * S_w * w)} \quad (10)$$

The solution to this optimization problem can be obtained by finding the eigenvectors corresponding to the largest eigenvalues of the matrix inverse of $(S_w)^{-1} * S_B$. Once the projection vector w is obtained, it can be used to project new samples onto the linear discriminant space for classification or dimensionality reduction purposes.

c) SVM algorithm: SVM is an ML algorithm that finds the best line or plane to separate different classes of data points, as shown in Algorithm 1. It uses a subset of training data called support vectors to determine this line or plane. SVM can handle both linear and non-linear data by using different kernel functions. One of its advantages is its ability to handle high-dimensional data without overfitting. It can also work well with small training samples. SVM has been extensively studied and improved with different techniques such as using different kernels and handling imbalanced datasets. Overall, SVM is a powerful and widely used technique for classification problems [22].

Algorithm 1: Pseudo code of the SVM algorithm

```
Set $Input = (x_i, y_i)$ where $i = 1,2,3,\dots,N$, $x_i = R^n$ and $y_i = \{+1,-1\}$.
Assign $f(X) = \omega^T x_i + b = \sum_{i=1}^{N} \omega Txi + b = 0$.
Minimize the QP problem as, $\varphi(\omega,\varepsilon) = \frac{1}{2}\|\omega\|^2 + C.\left(\sum_i^N \varepsilon_i\right)$.
Calculate the dual Lagrangian multipliers as $minL_p = \frac{1}{2}\|\omega\|^2 - \sum_{i=1}^{N} x_i y_i(\omega xi + b) + \sum_{i=1}^{N} x_i$.
Calculate the dual quadratic optimization (QP) problem as $L_D = \sum_{i=1}^{N} x_i - \frac{1}{2}\sum_{i,j}^{N} x_i x_j \; y_i y_j (x_i, x_j)$.
Solve dual optimization problem as $\sum_{i=1}^{N} y_i x_i = 0$.
Output the classifier as $f(X) = sgn\left(\sum_{i=1}^{N} x_i y_i (x.x_i) + \sum_{i=1}^{N} x_j y_j (x.x_j)\right)$
```

d) KNN algorithm: K-Nearest Neighbors (KNN) is a classification algorithm that assigns the label of a classified data point to an unclassified data point that is closest to it. This is done by considering the majority class of its K nearest data points. The distance between data points is typically measured using Euclidean distance (DE) or Manhattan distance (DM) [23] as shown in Algorithm 2:

Algorithm 2: Pseudo code of the KNN algorithm

```
Input: X: training data, Y: class labels of X, K: number of nearest neighbors.
Output: Class of a test sample x.
Start
Classify $(X, Y, x)$
For each sample x do
Calculate the distance: $d(x, X) = \sqrt{\sum_{i=1}^{n}(x_i - X_i)^2}$
End for
Classify x in the majority class: $C(x_i) = argmax_k \sum x_{j \in KNN} C(X_j, Y_k)$
End
```

e) Experimental design: The experimental design systematically evaluates the performance of SVM and KNN models in cloud-based mHealth systems, focusing on how preprocessing techniques affect accuracy and processing time (Fig. 2). Initially, baseline testing is conducted without preprocessing to establish reference metrics, including accuracy, precision, recall, F1-score, and processing time, highlighting limitations like bias from feature magnitudes and inefficiencies from data redundancy. Next, the impact of scaling techniques, such as Standard Scaler, Min-Max Scaler, Robust Scaler, and Quantile Transformer, is assessed by applying each method to the dataset and evaluating performance metrics. This step identifies effective scaling methods for reducing bias and improving efficiency. Dimensionality reduction techniques, including Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), are then evaluated to understand their influence on computational cost and classification accuracy. PCA retains components with the most variance, while LDA maximizes class separability, both offering insights into trade-offs between efficiency and performance. Finally, combined testing explores the interaction between scaling and dimensionality reduction, with pipelines like Standard Scaler + PCA and Min-Max Scaler + LDA being tested to optimize preprocessing for real-time mHealth applications. This comprehensive approach ensures a balance between accuracy and processing time, enabling scalable and efficient model deployment.

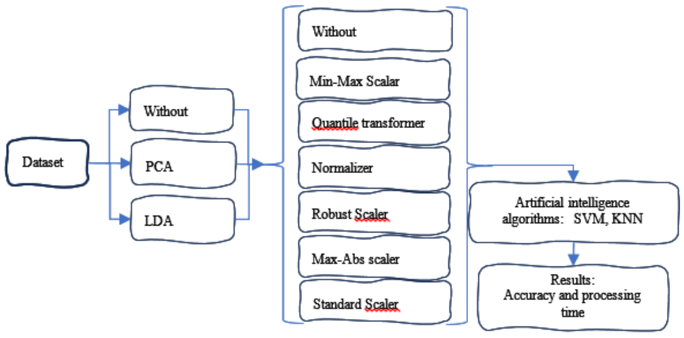


Fig. 2. Experimental Design

f) Metrics collected: Performance Metrics: Accuracy, precision, recall, F1-score.

- Efficiency Metrics: Processing time (measured in milliseconds).
- Trade-off Analysis: Evaluate the trade-offs between computational efficiency and model accuracy.

g) Computational environment: The experiments were conducted in a computational environment designed to replicate typical resource-constrained scenarios found in mHealth applications. This setup ensures the findings apply to real-world cloud-based systems supporting mHealth platforms.

- Hardware: Processor: AMD Ryzen 5 5500U CPU with Radeon Graphics, 2.10 GHz. Memory: 8 GB of RAM.
- Software: Programming Language: Python 3.8.

IV. Results and Analysis

The study evaluates the performance of SVM and KNN under various preprocessing scenarios, focusing on the following metrics: Processing Time (in milliseconds), Accuracy, Precision, Recall, and F1-Score.

The experiments demonstrate how scaling techniques and dimensionality reduction methods affect the efficiency and accuracy of these algorithms when applied to a health-related dataset.

A. Performance Without Preprocessing

As shown in Table II, SVM performs better than KNN in terms of accuracy (0.71 vs. 0.69) and is faster (0.063 ms vs. 0.13 ms). Without preprocessing, the data may contain bias and unnecessary information, making the model less efficient. This shows that preprocessing is important to improve speed and performance.

TABLE II. Metrics for SVM and KNN

Metric	Accuracy	Precision	Recall	F1-Score	Processing Time (ms)
SVM	0.71	0.70	0.71	0.69	0.063
KNN	0.69	0.69	0.69	0.69	0.13

B. Performance only with Scaling Techniques

Table III presents the impact of different scaling methods on the performance of SVM and KNN. Scaling significantly improves both accuracy and processing time for the two models. For SVM, the Min-Max Scaler achieves the highest accuracy (0.73) with a moderate processing time of 0.12 ms, while the Standard Scaler offers a good trade-off between accuracy (0.72) and faster processing (0.087 ms). In the case of KNN, the Standard Scaler provides the best accuracy (0.75) along with efficient processing (0.085 ms), making it particularly suitable for real-time applications. Although the Quantile Transformer yields moderate accuracy, its higher computational cost reduces its practicality. These results demonstrate that the choice of scaling method directly influences model performance and efficiency.

TABLE III. Metrics for SVM and KNN with Scaling Techniques

Scaling Technique	Standard Scaler		MinMax Scaler		Robust Scaler		Quantile Transformer	
Metric	Accuracy	Processing Time (ms)	Accuracy	Processing Time (ms)	Accuracy	Processing Time (ms)	Accuracy	Processing Time (ms)
SVM	0.72	0.087	0.73	0.12	0.66	0.13	0.71	0.42
KNN	0.75	0.085	0.73	0.12	0.68	0.063	0.74	0.13

C. Performance only with Dimensionality Reduction

Table IV shows that LDA outperforms PCA for both SVM and KNN. SVM with LDA achieves higher accuracy (0.74) and faster processing (0.085 ms) than with PCA. KNN also performs better and faster with LDA (0.047 ms). LDA's strength in maximizing class separability explains its superior results, making it the preferred choice for this dataset.

TABLE IV. Metrics for SVM and KNN with PCA and LDA

Metric	SVM + PCA	SVM + LDA	KNN + PCA	KNN + LDA
Precision	0.70	0.73	0.64	0.69
Recall	0.71	0.74	0.66	0.69
F1-score	0.69	0.73	0.65	0.69
Accuracy	0.71	0.74	0.66	0.69
Processing time (ms)	0.092	0.085	0.1	0.047

D. Performance with Dimensionality Reduction and Scaling Techniques

Table V shows that combining scaling and dimensionality reduction improves performance. For SVM with PCA, Min-Max Scaler gives the best accuracy (0.73), while Standard Scaler is faster (0.087 ms). For KNN with PCA, Min-Max Scaler has the highest precision (0.75), but Robust Scaler is more efficient (0.063 ms). With LDA, Standard Scaler provides the best accuracy and fastest processing, especially for SVM (0.041 ms).

TABLE V. METRICS FOR SVM WITH PCA AND SCALING TECHNIQUES

Scaling Technique	Max-Abs Scaler	Min-Max Scaler	Normalizer	Quantile Transformer	Robust Scaler	Standard Scaler
Precision	0.71	0.72	0.42	0.71	0.66	0.72
Recall	0.71	0.73	0.65	0.71	0.67	0.73
F1-Score	0.71	0.73	0.65	0.71	0.67	0.73
Processing time (ms)	0.12	0.12	0.14	0.42	0.13	0.087

TABLE VI. METRICS FOR KNN WITH PCA AND SCALING TECHNIQUES

Scaling Technique	Max-Abs Scaler	Min-Max Scaler	Normalizer	Quantile Transformer	Robust Scaler	Standard Scaler
Precision	0.71	0.75	0.66	0.74	0.68	0.75
Recall	0.69	0.73	0.66	0.74	0.68	0.75
F1-Score	0.69	0.73	0.66	0.74	0.68	0.75
Processing time(ms)	0.13	0.12	0.11	0.13	0.063	0.085

Table VI shows that for KNN with PCA, Min-Max and Standard Scalers achieve the best metrics (0.75), with Standard Scaler being faster (0.085 ms). Robust Scaler is the most efficient (0.063 ms) but slightly less accurate. Normalizer performs worst (0.66). This highlights Standard Scaler as the best balance of accuracy and speed.

TABLE VII. METRICS FOR SVM WITH LDA AND SCALING TECHNIQUES

Scaling Technique	Max-Abs Scaler	Min-Max Scaler	Normalizer	Quantile Transformer	Robust Scaler	Standard Scaler
Precision	0.73	0.73	0.65	0.73	0.73	0.73
Recall	0.74	0.74	0.68	0.74	0.74	0.74
F1-Score	0.73	0.73	0.64	0.73	0.73	0.73
Processing time(ms)	0.063	0.08	0.1	0.12	0.085	0.041

Table VII shows that for SVM with LDA, all scaling techniques except Normalizer achieve strong and consistent metrics (precision, recall, F1-score ≈ 0.73–0.74). Standard Scaler offers the best processing time (0.041 ms), making it the most efficient. Normalizer performs the worst across all metrics. These results highlight Standard Scaler as the most effective option when combining LDA with SVM.

TABLE VIII. METRICS FOR KNN WITH LDA AND SCALING TECHNIQUES

Scaling Technique	Max-Abs Scaler	Min-Max Scaler	Normalizer	Quantile Transformer	Robust Scaler	Standard Scaler
Precision	0.69	0.69	0.69	0.69	0.69	0.69
Recall	0.69	0.69	0.69	0.71	0.69	0.69
F1-Score	0.69	0.69	0.69	0.69	0.69	0.69
Processing time(ms)	0.07	0.072	0.063	0.084	0.047	0.032

Table VIII shows that KNN with LDA performs consistently across all scaling techniques, with precision, recall, and F1-score around 0.69. Quantile Transformer slightly improves recall (0.71), while Standard Scaler provides the fastest processing time (0.032 ms). These results indicate that although performance is similar, Standard Scaler is the most efficient.

Fig. 3(a) compares SVM and KNN accuracy and processing time with various scalers. Standard and Min-Max Scalers yield the highest accuracy, but Standard Scaler is fastest. Quantile Transformer is accurate but much slower.

Fig. 3(b) presents the performance of SVM and KNN using PCA and LDA. Both dimensionality reduction techniques yield comparable results across precision, recall, F1-score, and accuracy. However, LDA shows better efficiency, especially with KNN, offering lower processing time.

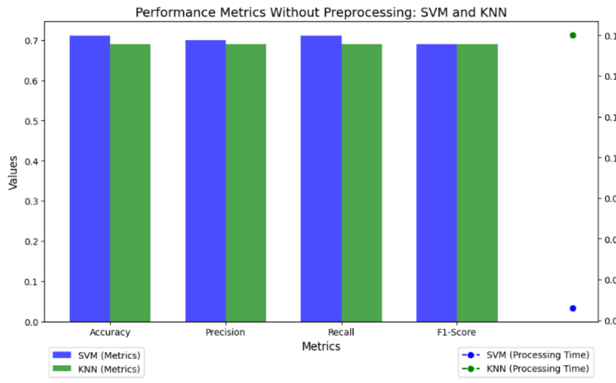
Fig. 3(c) shows SVM performance using different scaling techniques, with PCA Standard Scaler and MinMax Scaler achieving high precision, recall, and F1-score, while Standard Scaler has the lowest processing time. In contrast, Quantile Transformer offers good metrics but with the highest time cost.

Fig. 3(d) shows that QuantileTransformer offers the best KNN performance but is the slowest, while RobustScaler is fastest with lower accuracy StandardScaler and MinMaxScaler provide a good balance between performance and processing time.

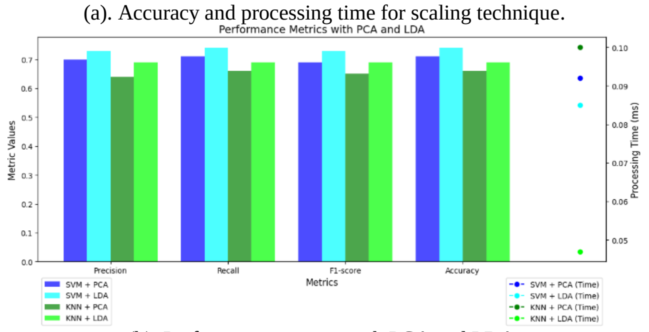
Fig. 3(e) shows that QuantileTransformer gives the best SVM performance but with the highest processing time, while StandardScaler is the fastest with strong accuracy. Other scalers like MinMaxScaler and RobustScaler offer a balanced trade-off between performance and speed.

Fig. 3(f) shows that QuantileTransformer gives the highest KNN performance with LDA but takes the longest time to process. StandardScaler is the fastest and still maintains strong performance. Other scalers like MinMaxScaler and RobustScaler also show consistent accuracy with lower processing times, indicating good overall efficiency.

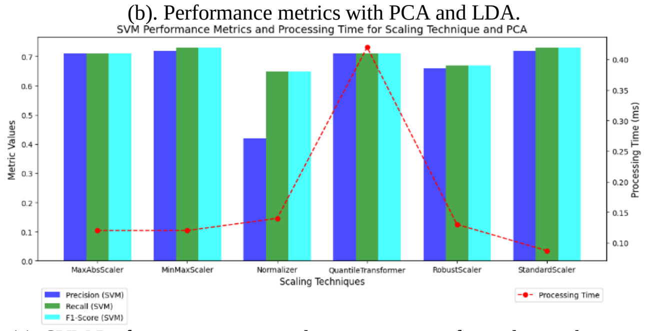
The findings in Fig. 3(a) to Fig. 3(f), highlight the practical implications of combining preprocessing with SVM and KNN to optimize mHealth applications. This approach enhances real-world applicability in areas such as remote patient monitoring, enabling real-time analysis of wearable data; chronic disease management, supporting early detection and personalized care; and telemedicine, improving the speed and reliability of virtual consultations. Faster processing times, such as 0.041 ms for SVM with LDA, ensure responsive systems that reduce delays in delivering critical insights, improving user trust and scalability through efficient resource utilization. Moreover, optimized preprocessing pipelines facilitate real-time predictions, enabling immediate feedback in critical scenarios like glucose monitoring or irregular heart rhythm detection. This responsiveness empowers healthcare professionals to make timely interventions, enhancing patient outcomes and advancing the efficiency of mHealth solutions.



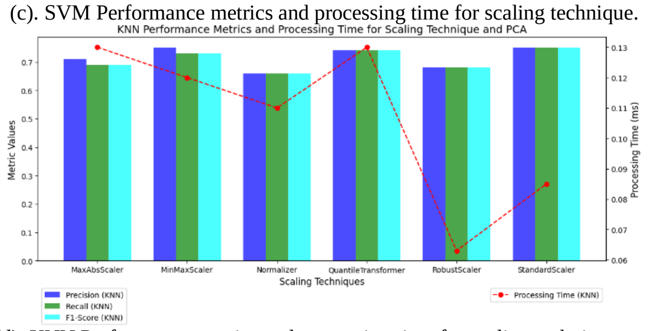
(a). Accuracy and processing time for scaling technique.



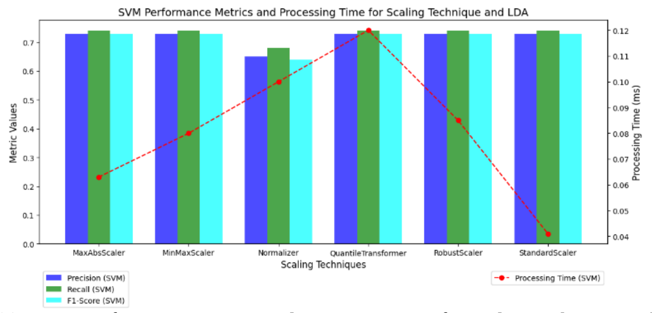
(b). Performance metrics with PCA and LDA.



(c). SVM Performance metrics and processing time for scaling technique.



(d). KNN Performance metrics and processing time for scaling technique and PCA.



(e). SVM Performance metrics and processing time for scaling technique and LDA.

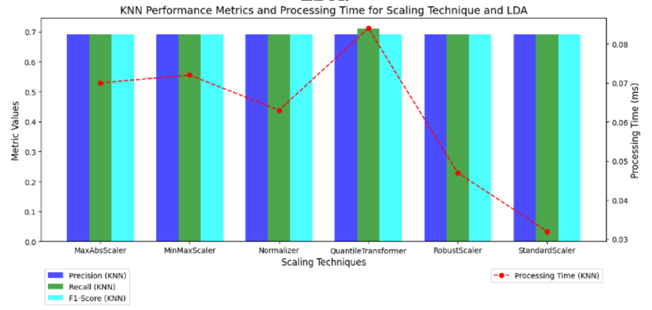


Fig. 3. (f) KNN Performance metrics and processing time for scaling technique and LDA.

V. Conclusion

This study provides a comprehensive evaluation of SVM and KNN algorithms, emphasizing their optimization for mobile health applications in cloud environments. Through systematic testing with various preprocessing techniques, such as scaling (Standard Scaler, Min-Max Scaler) and dimensionality reduction (PCA, LDA), the findings demonstrate significant improvements in both accuracy and processing efficiency. These optimizations address the unique constraints of mHealth systems, such as the need for real-time responsiveness and resource efficiency, making SVM and KNN suitable for applications like remote patient monitoring, chronic disease management, and telemedicine. The study highlights the critical role of preprocessing in reducing computational overhead, improving latency, and enhancing the scalability of cloud-based healthcare systems. While this work primarily focuses on preprocessing-driven optimization, it opens several avenues for future research. These include integrating deep learning models with classical classifiers, performing systematic hyperparameter tuning (e.g., grid search, Bayesian optimization), and applying statistical validation techniques to reinforce the robustness of results. Additionally, exploring adaptive preprocessing strategies that evolve in real-time with streaming health data could further improve model responsiveness in dynamic mHealth scenarios. Expanding the evaluation to include diverse datasets and multi-device deployment environments would also contribute to validating the generalizability of the proposed approach.

References

[1] A. A. Salameh, I. A. Abu-AlSondos, N. H. Abu, and A. Nahar Harun, “Current Knowledge and Future Possibilities of Medical Digital Technologies based on Mobile Health”, Int. J. Interact. Mob. Technol., vol. 17, no. 17, pp. pp. 134–147, Sep. 2023.

[2] R. Jeya, Hezerul Abdul Karim, and Sarina Binti Mansor, “Artificial Intelligence and Mobile Apps Support Intelligent Healthcare Systems for Mental Health Services ”, Int. J. Interact. Mob. Technol., vol. 18, no. 20, pp. pp. 157–168, Oct. 2024.

[3] M. Eddabbah, M. Moussaoui, and Y. Laaziz, "A smart architecture design for health remote monitoring systems and heterogeneous wireless sensor network technologies: a machine learning breathlessness prediction prototype," *International Journal of Intelligent Enterprise*, vol. 6, no. 2-4, pp. 293-310, 2019.

[4] T. S. Ajani, A. L. Imoize, and A. A. Atayero, "An overview of machine learning within embedded and mobile devices—optimizations and applications," *Sensors*, vol. 21, no. 13, pp. 4412, 2021.

[5] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Applied Computing and Informatics*, vol. 14, no. 1, pp. 1-16, 2018.

[6] N. S. Hussien, S. Sulaiman, A. Aborujilah, M. Wibowo, and H. Samma, “Scalability of Mobile Cloud Storage”, Int. J. Interact. Mob. Technol., vol. 15, no. 21, pp. pp. 199–206, Nov. 2021.

[7] A. S. Albahri, A. A. Zaidan, O. S. Albahri, B. B. Zaidan, and M. A. Alsalem, "Real-time fault-tolerant mHealth system: Comprehensive review of healthcare services, opens issues, challenges and methodological aspects," *Journal of Medical Systems*, vol. 42, pp. 1-56, 2018.

[8] Y. Deng, "Deep learning on mobile devices: a review," in *Mobile Multimedia/Image Processing, Security, and Applications 2019*, vol. 10993, pp. 52-66, 2019.

[9] C. Fan, M. Chen, X. Wang, J. Wang, and B. Huang, "A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data," *Frontiers in Energy Research*, vol. 9, pp. 652801, 2021.

[10] R. Sivan and Z. A. Zukarnain, "Security and privacy in cloud-based e-health system," *Symmetry*, vol. 13, no. 5, pp. 742, 2021.

[11] Q. An, S. Rahman, J. Zhou, and J. J. Kang, "A comprehensive review on machine learning in healthcare industry: classification, restrictions, opportunities and challenges," *Sensors*, vol. 23, no. 9, pp. 4178, 2023.

[12] R. Guido, M. C. Groccia, and D. Conforti, "A hyper-parameter tuning approach for cost-sensitive support vector machine classifiers," *Soft Computing*, vol. 27, no. 18, pp. 12863-12881, 2023.

[13] A. AlKarawi and K. AlJanabi, "Data Reduction Techniques: A Comparative Study," *Journal of Kufa for Mathematics and Computer*, vol. 9, no. 2, pp. 1-17, 2022.

[14] M. Sudha, M. R. Muthukathan, G. B. H. Malini, R. Sankar, M. Mythily, P. S. Kumaresh, M. N. V. Mageshkumar, and S. Shanmugam, "Predictive modeling for healthcare worker well-being with cloud computing and machine learning for stress management," *International Journal of Electrical & Computer Engineering*, vol. 15, no. 1, 2025.

[15] A. H. Abdulazeez and A. M. Abdulazeez, "Integration of Machine Learning with Fog Computing for Health Care Systems Challenges and Issues: A Review," The Indonesian Journal of Computer Science, vol. 13, no. 3, 2024.

[16] A. Deniz-Garcia, H. Fabelo, A. J. Rodriguez-Almeida, G. Zamora-Zamorano, M. Castro-Fernandez, M. D. P. Alberiche Ruano, T. Solvoll, C. Granja, T. R. Schopf, G. M. Callico, and C. Soguero-Ruiz, "Quality, usability, and effectiveness of mHealth apps and the role of artificial intelligence: current scenario and challenges," Journal of Medical Internet Research, vol. 25, p. e44030, 2023.

[17] H. K. Bharadwaj, A. Agarwal, V. Chamola, N. R. Lakkaniga, V. Hassija, M. Guizani, and B. Sikdar, "A review on the role of machine learning in enabling IoT based healthcare applications," IEEE Access, vol. 9, pp. 38859-38890, 2021.

[18] V. Chang, J. Bailey, Q. A. Xu, and Z. Sun, "Pima Indians diabetes mellitus classification based on machine learning (ML) algorithms," Neural Computing and Applications, vol. 35, no. 22, pp. 16157-16173, 2023.

[19] Cetin, V., and O. Yildiz, "A comprehensive review on data preprocessing techniques in data analysis," Pamukkale University Journal of Engineering Sciences, vol. 28, no. 2, pp. 299-312, 2022.

[20] A. G. Oladepo, A. O. Bajeh, A. O. Balogun, H. A. Mojeed, A. A. Salman, and A. I. Bako, “Heterogeneous Ensemble with Combined Dimensionality Reduction for Social Spam Detection”, Int. J. Interact. Mob. Technol., vol. 15, no. 17, pp. pp. 84–103, Sep. 2021.

[21] C. Zhang, M. Mei, Z. Mei, J. Zhang, A. Deng, and C. Lu, "PLDANet: reasonable combination of PCA and LDA convolutional networks," International Journal of Computers Communications & Control, vol. 17, no. 2, 2022.

[22] D. M. Abdullah and A. M. Abdulazeez, "Machine learning applications based on SVM classification a review," Qubahan Academic Journal, vol. 1, no. 2, pp. 81-90, 2021.

[23] M. Suyal and P. Goyal, "A review on analysis of K-nearest neighbor classification machine learning algorithms based on supervised learning," International Journal of Engineering Trends and Technology, vol. 70, no. 7, pp. 43-48, 2022.