Journal of Artificial Intelligence and Engineering Applications

Website: https://ioinformatic.org/

15th February 2026. Vol. 5. No. 2; e-ISSN: 2808-4519

Implementation of Microservice Architecture in a Machine Learning-Based Expert System for Sleep Disorder Diagnosis

Amin Nur Rais[1]*, Warjiyono[2]

[1]*Informatika, Universitas Bina Sarana Informatika*
[2] *Sistem Informasi Akuntansi, Universitas Bina Sarana Informatika*
Amin.arv@bsi.ac.id[1], warjiyono.wrj@bsi.ac.id[2]*

Abstract

Sleep disorders constitute a significant health concern that frequently remains undiagnosed due to restricted access to adequate clinical assessment facilities. This study aims to develop an accurate and accessible early detection system for sleep disorders by implementing the Gaussian Naive Bayes algorithm within a hybrid microservice architecture. The development methodology involves decoupling the intelligent computing service, built on Python (Flask), from the user interface, developed using PHP (CodeIgniter 3), with communication facilitated via an Application Programming Interface (API). The model was trained utilizing a sleep disorder diagnostic dataset comprising 1,000 medical records and evaluated using the 10-Fold Cross-Validation method. Experimental results indicate that the developed model demonstrates superior classification performance, achieving an accuracy of 97.40% and a recall of 99.66%. The high recall value evidences the system's superior sensitivity in detecting positive cases, thereby effectively minimizing the risk of undetected patients (False Negatives). System integration via API proved stable in delivering real-time diagnostic visualization, confirming that this hybrid architecture offers a valid, modular, and responsive solution for the implementation of intelligent healthcare systems.

Keywords: *Gaussian Naive Bayes, Sleep Disorders, Microservice, Python Flask, CodeIgniter 3.*

1. Introduction

Sleep is a fundamental physiological necessity that plays a crucial role in maintaining physical health, mental stability, and overall quality of life. However, sleep disorders have emerged as a global health issue that is frequently overlooked. Recent epidemiological research indicates that the prevalence of sleep disorders, such as insomnia and sleep apnea, continues to rise and possesses a strong correlation with the risk of chronic diseases, including type 2 diabetes mellitus, hypertension, and cardiovascular disorders [1]. The impact of poor sleep quality extends beyond individual health, affecting economic productivity and public safety due to cognitive decline during daytime hours. Despite the high urgency of addressing sleep disorders, access to accurate clinical diagnosis still faces significant constraints. The current gold standard method for diagnosing sleep disorders is Polysomnography (PSG). Although highly precise, PSG is a costly and time-consuming procedure requiring complex equipment and must be conducted in a sleep laboratory under the supervision of expert technicians [2]. This complexity and high cost create a gap in healthcare access, where many patients with sleep disorder symptoms fail to receive timely diagnosis and medical treatment [3]. Consequently, there is a critical need for alternative diagnostic methods that are more efficient, affordable, and accessible to the general public without significantly compromising detection accuracy. The rapid development of Artificial Intelligence (AI) and Machine Learning (ML) technologies offers promising solutions to address these challenges. Machine learning algorithms are capable of analyzing complex clinical data patterns to perform automated predictions with accuracy levels approaching expert diagnosis. One method proven effective in medical classification is Gaussian Naive Bayes (GNB). GNB is known for its high computational efficiency and robust generalization capability, even on datasets with limited feature dimensions, making it an ideal candidate for early diagnosis systems [4].

However, technical challenges arise in implementing AI models into applications usable by end-users. Machine Learning models are generally developed within the Python programming ecosystem due to its robust data science library support, while many industry web application interfaces are still built using PHP due to ease of deployment and the availability of mature frameworks. Traditional monolithic architectural approaches often fail to efficiently bridge this technological disparity. To address this, Microservice architecture becomes a relevant approach. The application of modular architecture in healthcare systems enables high scalability, flexibility in technology selection (polyglot programming), and ease of integration between distinct components [3].

Based on this background, this study aims to develop a Smart Sleep Health Check System that integrates the predictive power of the Gaussian Naive Bayes algorithm with the flexibility of modern web interfaces. This study proposes a hybrid microservice architecture connecting Python-based AI services (Flask) with a PHP-based user interface (CodeIgniter 3). This approach is expected to yield an early

diagnosis system that is not only accurate and sensitive in detecting sleep disorders but also responsive and accessible to users, thereby contributing to the improvement of public healthcare service quality.

2. Research Methods

To ensure the validity of the microservice architecture integration, this study applies a systematic software development methodology [5]. The research workflow is organized into three main interconnected phases, as illustrated in Figure 1: (2.1) Modeling, (2.2) API Development, and (2.3) Interface Development.

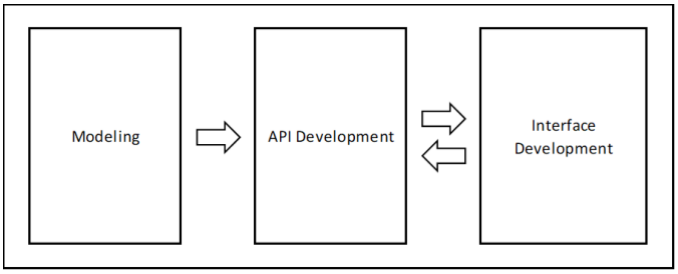

Fig. 1: Research Methods

2.1. Modeling

The modeling phase focuses on transforming raw data into artificial intelligence artifacts ready for deployment within the architecture. This process is conducted using the Python programming language development environment [5]. The modeling workflow involves analyzing the presented raw data, partitioning the dataset into training and testing sets, modeling using the Gaussian Naive Bayes algorithm [6], evaluating the model's performance, and finally, exporting the model for deployment.

2.2. API Development

Building upon the artifacts constructed in the modeling phase, the API development aims to bridge the modeling artifacts with the microservice architecture. This API framework is developed using the Python programming language utilizing the Flask framework [7]. In this phase, the model exported during the modeling stage is loaded into the Flask environment. The process includes formatting input and output data structures and developing specific endpoints accessible by the frontend system.

2.3. Interface Development

The interface development phase is designed to present prediction results to the user through an interactive and educational web interface. This stage utilizes the CodeIgniter 3 framework [8], which implements the Model-View-Controller (MVC) architectural pattern. The controller component plays a critical role in ensuring that data transmission between the API and the View layer functions correctly and adheres to the predefined endpoint structure.

3. Results and Discussion

This study successfully implemented an artificial intelligence-based sleep disorder diagnosis system utilizing a hybrid microservice architecture. The integration between the backend, which handles predictive logic, and the frontend, which presents data visualization, operates according to the methodological design. Test results indicate that the separation of concerns between numerical data processing by Python and interface presentation by PHP produces a modular and high-performance system [8]. The following sections delineate the detailed evaluation results of each development stage, ranging from predictive model performance and API reliability to user interface responsiveness.

3.1. Modeling

The modeling phase commences with Data Understanding, which serves as the primary foundation of the artificial intelligence system. This study utilizes a secondary dataset, the Sleep Disorder Diagnostic Dataset, sourced from the public repository Kaggle. The dataset comprises 1,000 patient medical records covering demographic, clinical, and sleep disorder diagnosis history variables. The data structure employed in model training includes 8 main attributes. From these attributes, the features selected (Feature Selection) as predictor variables (X) include Age, Gender, Sleep Disorder Type, AHI Score (Apnea-Hypopnea Index), and SaO2 Level (Oxygen Saturation Level). Meanwhile, the target variable (y) is Diagnosis_Confirmed, which acts as a binary label to determine the positive or negative status of a sleep disorder.

Prior to algorithmic processing, a Data Preprocessing stage is conducted to handle categorical data. The Gender and Sleep Disorder Type variables are transformed using the One-Hot Encoding technique. This technique converts text categories into binary vectors (0 and 1), thereby enabling the Gaussian Naive Bayes algorithm to perform mathematical probability calculations on these non-numeric features [9] . Model evaluation was conducted using the K-Fold Cross-Validation method with $k=10$. This method partitions the 1,000 data points into 10 subsets, where each iteration alternately utilizes 900 data points for training and 100 data points for testing. This approach was selected to ensure model validity and prevent bias that might arise due to data distribution imbalances [10].

Experimental results demonstrate very high model performance, achieving an Accuracy of 97.40% and a Recall of 99.66%. The high Recall value indicates that the model is highly sensitive in detecting positive cases, a crucial aspect in medical diagnosis systems to minimize the risk of undetected ill patients (False Negatives).

Table 1: Evaluation Matrix Results

Evaluation Metric	Score Value	Percentage
Accuracy	0.9740	97.40%
Precision	0.9746	97.46%
Recall (Sensitivity)	0.9966	99.66%
F1-Score	0.9855	98.55%

After the conclusion of the evaluation phase and the validation of model performance, the ultimate step in the modeling phase is the model export or serialization process. This process aims to save the internal state of the trained algorithm—including prior probabilities and other statistical parameters—into a physical file to facilitate reuse in a production environment without the necessity of retraining. Utilizing the joblib library available within the Python ecosystem, the best model object from the training iterations is converted into a binary format and saved *as model.joblib*. This binary file functions as a static artifact encapsulating the learned "intelligence," which is subsequently loaded by the API service to handle prediction requests in real-time.

3.2. API Development

Following the successful validation of the artificial intelligence artifacts, the subsequent stage involves the development of the Application Programming Interface (API), which functions as a communication bridge between the predictive model and the user interface. This service is constructed using the Python programming language utilizing the Flask micro-framework. The selection of Flask is based on its lightweight characteristics and compatibility with machine learning libraries such as Scikit-learn and Joblib, enabling it to handle HTTP requests with low latency while maintaining reliability in loading complex computational models [11]. During the initialization phase, the application loads the *model.joblib* file into the server memory. This loading process is equipped with error handling mechanisms to ensure the service remains stable and provides notifications if the model artifact is not found.

The most crucial component in this API development is the input data preprocessing function. Given that the Gaussian Naive Bayes model accepts input exclusively as numeric vectors, the API must replicate the data transformation logic identically to that performed during the training phase. The *preprocess_input* function is implemented to handle the conversion of raw data from JSON format into a NumPy array format processable by the model. This transformation logic encompasses:

1. Gender Variable Conversion: Textual data 'Male' is converted to a value of 1, whereas 'Female' becomes 0.
2. Disease Feature Engineering: One-Hot Encoding implementation is performed manually using conditional logic. The system detects sleep disorder types (such as Narcolepsy, Obstructive Sleep Apnea, etc.) and assigns a binary value of 1 to the corresponding variable, while Insomnia is treated as the reference category where all disease indicator variables are set to 0.
3. Feature Vector Arrangement: All transformed variables are arranged into a precise array sequence *([age, ahi, sao2, gender, disorder_types...]*) in accordance with the training data structure.

The API service provides a primary endpoint, */predict*, which accepts requests via the POST method. This endpoint is tasked with receiving the JSON data payload from the client, validating data completeness, executing the preprocessing function, and invoking the model prediction function (*model.predict*). The response returned to the client is formatted in JSON, as this format constitutes a lightweight and uniform data interchange standard for communication between the Python backend and various frontend platforms [12]. The response includes the original input data, the binary prediction code (0 or 1), and the textual interpretation of the diagnostic result ("Diagnosis Confirmed" or "Diagnosis Negative"). Furthermore, the CORS (Cross-Origin Resource Sharing) module is enabled to allow cross-domain access, permitting the web interface (frontend) to communicate with the API server without browser security constraints often encountered in distributed architectures [13].

Table 2: API Endpoint Specifications for Sleep Disorder Diagnosis

Endpoint	HTTP Method	Function Description	Request Body Structure (JSON)	Response Body Structure (JSON)
		Server Status Check		
/	GET	Ensures the Flask service is active and ready to accept requests.	*(None)*	{ "status": "Active", "message": "..." }
		Diagnosis Prediction		
/predict	POST	Receives patient clinical data, performs preprocessing, and returns classification	{ "Age": int, "Gender": "Male"|"Female", "Sleep_Disorder_Type": string, "AHI_Score": float, "SaO2_Level": float }	{ "prediction_code": 0|1, "result": "Diagnosis Confirmed"|"Diagnosis Negative", "input_data": { ... } }

	results from the Naive Bayes model.

3.3. Interface Development

The development of the User Interface (UI) aims to translate the complexity of the artificial intelligence model into a visual format comprehensible to lay users. This implementation is built upon the CodeIgniter 3 framework, utilizing the Model-View-Controller (MVC) architectural pattern. The application of the MVC architecture in CodeIgniter 3 has proven effective in separating business logic, data, and presentation, resulting in ease of code maintenance and lightweight system execution speed [11]. On the interface side, the system is designed as an interactive Single Page Application (SPA) utilizing the Tailwind CSS utility library to ensure display responsiveness across various devices. The utility-based approach in Tailwind CSS allows for more flexible and consistent design development compared to traditional frameworks, ensuring the display remains adaptive to various user screen sizes [14].

The system interaction flow is divided into four visual stages (states) dynamically managed using JavaScript, comprising:

1. Welcome Screen The initial stage is the homepage display welcoming the user with the title "Smart Sleep Health Check System". This page is designed with minimalism, focusing on the "Mulai Konsultasi" Call to Action (CTA) button. Technically, this interface employs color gradient design elements and subtle animations (animate-slide-up) to provide a modern user experience prior to engaging with technical processes.

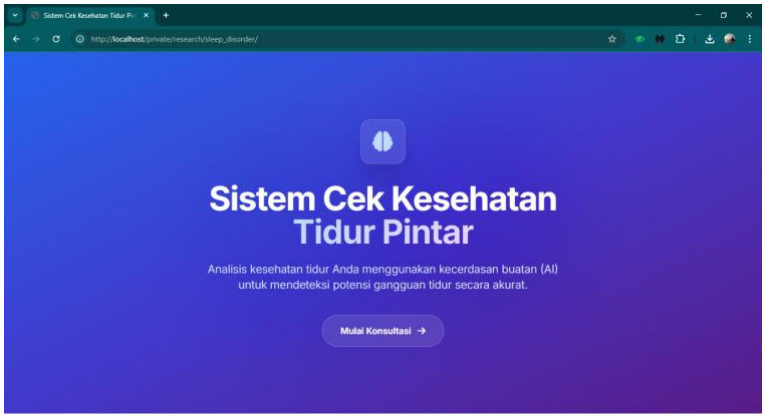

Fig. 2: Welcome Screen

2. Clinical Data Input Form Once the user initiates the session, the system transitions to the data input form. At this stage, users are required to input five vital parameters required by the model:

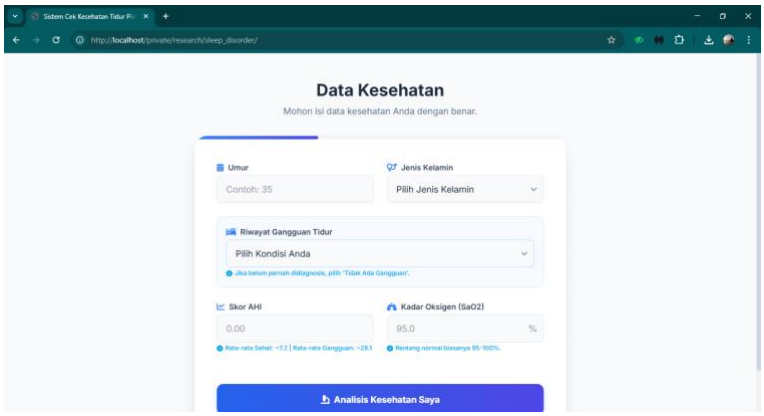

Fig. 3: Input Form

   a. Demographic Data: Age and Gender.
   b. Medical History: Sleep Disorder Type (Dropdown menu covering Insomnia, Sleep Apnea, etc.).
   c. Clinical Parameters: AHI Score (Apnea-Hypopnea Index) and Oxygen Saturation Level (SaO2).

   Input validation is applied directly on the client-side (client-side validation) to ensure all data is populated in a valid numeric format before transmission to the server.

3. Analysis Process (Loading State) When the "Analyze My Health" button is pressed, the interface displays a loading overlay with a spinner animation. In the background, the JavaScript function handleAnalysis aggregates form data and transmits it asynchronously

(asynchronous fetch request) to the Flask API endpoint (/*predict*). This process provides visual feedback to the user that the system is processing data using artificial intelligence, simultaneously preventing double interaction during the computation process.

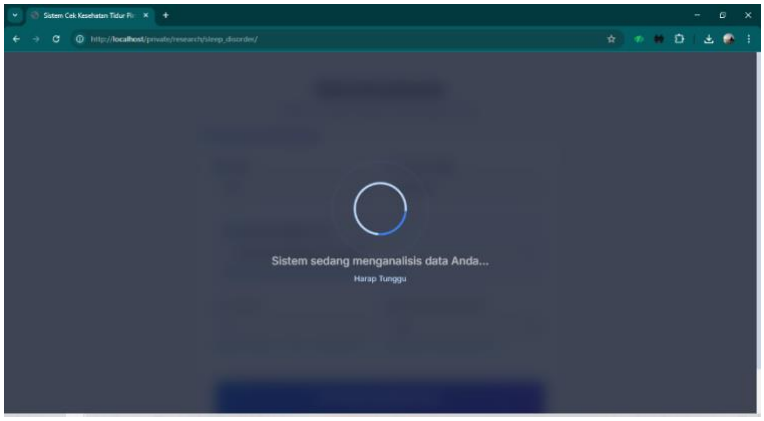

Fig. 4: Analiysis Process

4. Diagnostic Result Visualization (Result Dashboard) The final stage is the presentation of diagnostic results appearing in a modal window (pop-up modal) after the response is received from the API. This display presents three main information components:

a. Diagnosis Status: A visual label colored red (if a disorder is detected) or green (if normal), providing a quick conclusion to the user.

b. Visual Meter: Dynamic bar charts to visualize the user's health position. The AHI Score chart displays the user's position relative to the "Safe Zone" (<15) and "Danger Zone". Meanwhile, the SaO2 chart visualizes blood oxygen levels with color indicators changing according to severity.

c. Educational Card: Contextual information explaining the significance of these medical numbers, such as heart health risks if oxygen levels fall below 90%, to enhance user health literacy.

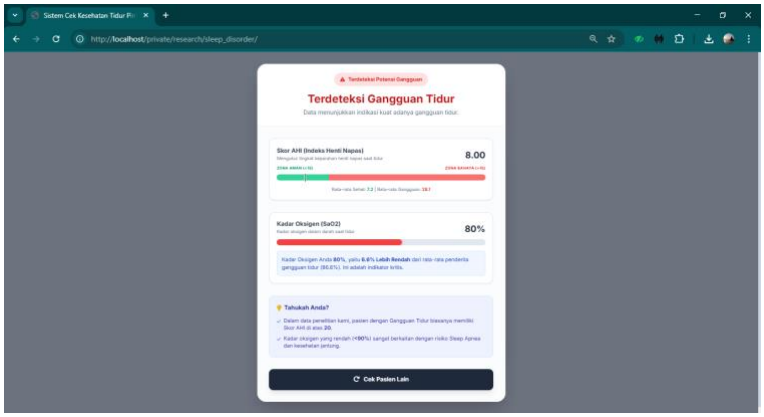

Fig. 5: Diagnostic Result

4. Conclusion

This study concludes that the development of the Smart Sleep Health Check System utilizing a hybrid microservice architecture has successfully integrated the analytical reliability of Python with the interface flexibility of PHP. The implementation of the Gaussian Naive Bayes algorithm demonstrated superior performance, achieving an Accuracy of 97.40% and a Recall of 99.66%. These results indicate the system's high sensitivity in detecting sleep disorder risks, thereby minimizing False Negatives. The synergy between the Flask API backend, tasked with performing clinical data inference, and the CodeIgniter 3 frontend, which presents real-time diagnostic visualization, confirms that this solution is not only statistically valid but also ready for deployment as a responsive, modular, and user-friendly medical screening instrument.

References

[1] F. S. Al-qahtani, “Prevalence of sleep disturbance and its associated factors among diabetes type-2 patients in Saudi Arabia,” no. November, pp. 1–12, 2024, doi: 10.3389/fpubh.2024.1283629.

[2] Title, P. Versus, P. Devices, A. In, T. H. E. Diagnosis, and O. F. Obstructive, “Polysomnography Versus Portable Devices And Mobile Applications In The Diagnosis Of Obstructive Sleep Apnea: A Narrative Review Of Current Evidence,” Int. J. Innov. Technol. Soc. Sci., pp. 0–10, 2025.

[3] M. M. Monowar et al., “Advanced sleep disorder detection using multi-layered ensemble learning and advanced data balancing techniques”.

[4] R. Alazaidah, G. Samara, M. Aljaidi, M. Alshammari, M. H. Qasem, and A. A. 3, “Potential of Machine Learning for Predicting Sleep Disorders : A Comprehensive Analysis of Regression and Classification Models,” MDPI, 2024.

[5] D. Irawan, “Implementasi Arsitektur Microservices pada Pengembangan Aplikasi Absensi Web Terdistribusi,” vol. 7, no. 3, 2025, doi: 10.32877/bt.v7i3.2401.

[6] D. I. Saputra and D. L. Hakim, “Implementasi Algoritma Gaussian Naive Bayes Classifier Untuk Prediksi Potensi Tsunami Berbasis Mikrokontroler,” Epsil. J. Electr. Eng. Inf. Technol., vol. 2, pp. 122–138, 2022.

[7] H. L. Walingkas and P. O. N. Saian, “Penerapan Framework Flask pada Pembangunan Sistem Informasi Pemasok Barang,” J. JTIK (Jurnal Teknol. Inf. dan Komunikasi), vol. 7, no. 2, pp. 227–234, 2023, doi: 10.35870/jtik.v7i2.729.

[8] M. M. Bani, S. Dwi, I. Mau, and S. G. Balimema, "Implementasi Sistem Pelayanan Kesehatan Berbasis Online Menggunakan Codeigniter 3 dan Metode MVC CodeIgniter 3 , diharapkan dapat tercipta solusi yang efektif dalam menunjang layanan Delone And Mclean Website Sistem Informasi Akademik STIKES Sukabumi , de," Modem J. Inform. dan Sains Teknol., vol. 3, no. September, 2025.

[9] M. Guntara and F. D. Astuti, "Komparasi Kinerja Label-Encoding dengan One-Hot-Encoding pada Algoritma K-Nearest Neighbor menggunakan Himpunan Data Campuran Performance Comparison of Label-Encoding with One-Hot-Encoding Methode on K-Nearest Neighbor Algorithm with Mixed Type Data Set," JIKO (JURNAL Inform. DANKOMPUTER), vol. 9, no. 2, pp. 352–360, 2025, doi: 10.26798/jiko.v9i2.1605.

[10] H. Hafid, "Penerapan K-Fold Cross Validation untuk Menganalisis Kinerja Algoritma K-Nearest Neighbor pada Data Kasus Covid-19 di Indonesia," J. Math. Comput. Stat., vol. 6, no. 2, pp. 161–168, 2023.

[11] E. Suherlan, S. Arti, S. Nabilah, Z. Hazimah, F. Teknik, and U. N. Jakarta, "PENERAPAN FLASK FRAMEWORK UNTUK DEPLOYMENT MODEL MACHINE LEARNING DALAM MENDUKUNG ANALISIS ADAPTASI Available at :," vol. 9, no. 1, 2025.

[12] Supria et al., "Seminar Nasional Industri dan Teknologi (SNIT), Politeknik Negeri Bengkalis," Semin. Nas. Ind. dan Teknol. (SNIT), Politek. Negeri Bengkalis Perbandingan, no. November, pp. 17–23, 2024.

[13] M. I. Arief, D. S. Anwar, and A. Supriatman, "Analisis Kerentanan Website Melalui Pendekatan Penetration Testing Berdasarkan Standar Owasp Top 10 Studi Kasus Simpelmas Universitas XYZ," J. ELEKTRO Inform. SWADHARMA, vol. 05, 2025.

[14] M. F. Santoso, "Perbandingan Efektivitas Bootstrap dan Tailwind CSS dalam Pengembangan UI Web Responsif," J. Teknol. Dan Sist. Inf. Bisnis, vol. 7, no. 4, pp. 489–497, 2025.