JOURNAL ON COMMUNICATIONS

ISSN:1000-436X

Scopus®

REGISTERED

www.jocs.review

Hybrid Expert System Combining Rule-Based Reasoning and Ensemble Classifiers for Musculoskeletal Disease Diagnosis: Design, Implementation and Evaluation.

Sunny Kalu Egereonu[1], Obi Chukwuemeka Nwokonkwo [2], Emmanuel Chukwudi Amadi[3], Anthony Ifeanyi Otuonye[4], Ubaezue Ugochukwu Egereonu[5], Ijeoma Blessing Omenihu[6]

Assistant Lecturer[1], Senior Lecturer[2,3,4], HOD[3], Professor[5], Assistant Lecturer[6]

Department of Information Technology, Federal University of Technology, Owerri, Nigeria[1],[2,3,4].

Department of Chemistry, Federal University of Technology, Owerri, Nigeria [5].

Department of Computer Science, Rhema University, Aba, Abia State, Nigeria [6]

Corresponding Author: Sunny Kalu Egereonu[1]

Abstract

Background: *Musculoskeletal diseases (MSDs) are a leading cause of chronic pain and disability worldwide, imposing significant burdens on patients and healthcare systems. Accurate and timely diagnosis is vital for preventing progression, yet overlapping symptoms and diverse clinical presentations make evaluation challenging. This highlights the importance of reliable and interpretable decision support systems to assist clinicians.*

Methodology: *This study developed and evaluated an optimized expert system for musculoskeletal disease diagnosis, with specific attention to tendonitis. The system combines a rule based inference engine with four supervised learning classifiers: Random Forest, Support Vector Machine, Logistic Regression, and XGBoost. An ensemble decision fusion strategy was applied to balance predictive accuracy with interpretability. Development followed the Waterfall methodology, while validation was carried out on 1,000 physician confirmed cases using structured clinical features and examination findings. A user centered interface was designed to provide traceable rule reasoning and classifier explanations, enhancing transparency and clinical integration.*

Results: *The system achieved strong diagnostic performance with Accuracy 92.40%, Precision 97.58%, Recall 91.48%, F1 score 94.43%, Specificity 94.59%, Negative Predictive Value 82.35%, Cohen's Kappa 82.51%, Matthews Correlation Coefficient 82.96%, and Youden's J Statistic 86.07%. Ninety five percent confidence intervals for key measures were: Accuracy 91.7–93.8%, Sensitivity 89.8–93.0%, and Specificity 91.7–97.7%. These findings confirm balanced and reliable classification across both positive and negative musculoskeletal cases. Comparative analysis indicated that integrating symbolic reasoning with ensemble classifiers enhanced diagnostic consistency over individual models while maintaining interpretability.*

Conclusion: *The proposed expert system demonstrates high diagnostic accuracy and readiness for clinical application in musculoskeletal disease diagnosis. It provides interpretable outputs and a robust decision support framework for healthcare professionals. Future work will focus on multicentre validation and prospective clinical trials to confirm generalizability and real world applicability.*

Keywords: Artificial Intelligence (AI)**,** Musculoskeletal Disease Diagnosis**,** Expert System**,** Ensemble Machine Learning Classifiers, Clinical Decision Support System (CDSS)**,** Medical Informatics**,** Healthcare Technology.

1. Introduction

Musculoskeletal diseases (MSDs) encompass a wide range of disorders affecting the skeletal and muscular systems, including arthritis, osteoporosis, fractures, and soft tissue injuries. These conditions significantly impact the musculoskeletal framework, involving muscles, joints, ligaments, tendons, nerves, and blood vessels. As a consequence, MSDs often lead to reduced work productivity, increased absenteeism, and long-term disability [1]. Their chronic nature and high prevalence impose a substantial burden on global healthcare systems, contributing to rising medical costs due to prolonged treatment needs and associated disabilities [2]. These factors highlight the necessity for early, accurate diagnostic approaches and effective treatment strategies.

MSDs have emerged as a significant public health challenge, affecting millions worldwide and placing immense pressure on healthcare infrastructures. The World Health Organization (WHO) categorizes MSDs as conditions that impair bones, muscles, joints, and connective tissues, leading to persistent pain, functional limitations, and diminished quality of life [3]. The Global Burden of Disease Study (2019) further underscores the impact of MSDs, attributing a considerable share of years lived with disability (YLDs) to these conditions. Beyond the individual suffering, MSDs pose substantial economic challenges, including rising healthcare expenditures, productivity losses, and broader societal costs. These implications emphasize the urgency for innovative diagnostic solutions and management frameworks to mitigate the burden of MSDs.

Diagnosing MSDs remains complex due to their diverse clinical manifestations, overlapping symptoms, and the need for thorough clinical evaluations [4]. Medical practitioners frequently encounter difficulties in distinguishing MSDs from other conditions, leading to diagnostic delays and suboptimal treatment outcomes. Conventional diagnostic approaches including clinical assessments, imaging modalities, and laboratory analyses can be time-intensive, costly, and prone to subjective interpretation, further complicating accurate disease identification and timely intervention [5].

To address these challenges, artificial intelligence (AI) driven expert systems have emerged as valuable diagnostic aids. Expert systems, a subset of AI, are designed to replicate human decision-making in specialized fields such as medicine. These systems employ knowledge representation, inference mechanisms, and algorithmic reasoning to analyze clinical data and generate diagnostic insights. By integrating established medical guidelines, clinical expertise, and evidence-based methodologies, expert systems enhance diagnostic precision and facilitate personalized treatment planning.

Numerous AI methodologies such as rule-based systems, Bayesian networks, neural networks, and machine learning algorithms have been explored to develop expert systems tailored for MSD diagnosis. AI research traces back to the mid-20th century, with foundational work in computing during the 1940s. The term “artificial intelligence” was formally introduced in 1956 by John McCarthy, who defined it as the scientific and engineering discipline focused on developing intelligent systems, particularly computer programs that mimic human cognition [6]. AI-driven expert systems, often termed knowledge-based systems, employ structured knowledge acquisition techniques to tackle complex medical challenges [7].

Expert systems function as intelligent software applications that simulate the problem-solving processes of medical specialists. These systems rely on curated knowledge bases containing domain-specific expertise and apply inference mechanisms to generate diagnostic conclusions [2]. Research has demonstrated the efficacy of expert systems in diagnosing various MSDs, including osteoarthritis, rheumatoid arthritis, and osteoporosis. These AI powered diagnostic tools leverage patient histories, clinical findings, imaging data, and laboratory test results to generate differential diagnoses and recommend treatment options. However, access to expert medical knowledge remains inconsistent, particularly in rural and underserved areas, where specialist availability is limited. This gap underscores the necessity of expanding expert medical knowledge beyond urban centers and improving accessibility for remote populations. Since much of medical expertise is experiential, translating it into structured computational models poses a challenge. Expert systems particularly fuzzy logic-based models offer a viable solution to these limitations.

Despite their potential, expert systems face challenges related to knowledge acquisition, system validation, integration into existing clinical workflows, and user acceptance. Addressing these barriers is essential for their widespread adoption and effective utilization. Nevertheless, AI-powered expert systems represent a transformative advancement in MSD diagnosis, offering rapid, accurate, and evidence-based decision support for healthcare professionals. Continued research and innovation are crucial to refining these systems, ensuring their reliability, and unlocking their full potential in medical diagnostics.

Therefore, this study aims to develop an optimized AI-powered expert system tailored for MSD diagnosis and management. The research will outline the system’s macro-architecture, analyze the interconnections between its components, and rigorously evaluate its diagnostic reliability in comparison to conventional expert methodologies.

2. Related Works

The integration of artificial intelligence (AI) into musculoskeletal disease (MSD) diagnosis has undergone a paradigm shift, leveraging computational intelligence to enhance diagnostic precision, streamline clinical

workflows, and optimize patient management. AI-driven expert systems, underpinned by deep learning architectures, rule-based reasoning, and probabilistic inference, have demonstrated remarkable efficacy in mitigating diagnostic uncertainties and augmenting clinical decision-making.

2.1 AI and Expert Systems in Musculoskeletal Diagnosis

Recent advancements in AI-powered expert systems have underscored their potential in refining musculoskeletal diagnostics by overcoming the limitations of traditional methodologies. Conventional diagnostic approaches, including manual radiographic assessments, subjective clinical evaluations, and time-intensive biochemical assays, often suffer from inter-observer variability and diagnostic latency [5]. AI-enhanced expert systems leverage extensive knowledge repositories, pattern recognition capabilities, and inference mechanisms to offer expedited and accurate differential diagnoses.

Román-Belmonte and other researchers [8] conducted an extensive review of AI applications in musculoskeletal imaging, demonstrating that deep convolutional neural networks (CNNs) have achieved diagnostic sensitivity and specificity exceeding 95% in identifying degenerative joint disorders. Their findings highlight the transformative role of AI in radiological interpretation, significantly reducing reliance on human-dependent image analysis while improving diagnostic consistency. The work done In this domain provides crucial insights into the use of rule-based expert systems, AI, and framework-based expert systems, emphasizing their effectiveness in the healthcare sector.

In 2024, the National Health Service (NHS) in England received approval to implement AI tools for detecting bone fractures in X-rays. This initiative aims to reduce missed fractures, which can adversely affect patient outcomes. Clinical evidence indicates that AI can enhance fracture detection rates without increasing diagnostic errors, thereby streamlining patient care and alleviating the workload of radiologists amidst staff shortages. Four AI platforms, including TechCare Alert, Rayvolve, BoneView, and RBfracture, have been recommended for use, each costing approximately £1 per scan [9].

Further, Ibrahim and Ojo [7] developed an AI-driven expert system integrating fuzzy logic for musculoskeletal diagnosis. Their study emphasized the system's ability to synthesize heterogeneous clinical datasets, refine diagnostic accuracy through iterative learning, and provide probabilistic assessments for differential diagnoses. This adaptability ensures that AI systems remain dynamic, evolving alongside advancements in medical knowledge.

Furthermore, a 2025 study presented at the International Society of Arthroscopy, Knee Surgery, and Orthopaedic Sports Medicine (ISAKOS) Congress demonstrated the efficacy of an AI-based approach to triage acute knee injuries. The study utilized a machine learning algorithm to identify patients requiring MRI scans, correctly identifying 92% of such cases and accurately diagnosing 81% of patients. This approach has the potential to improve the efficiency and cost-effectiveness of patient pathways for acute knee injuries, leading to quicker diagnoses and treatments [10].

2.2 Rule-Based and Machine Learning Approaches

The deployment of rule-based expert systems, particularly those integrating knowledge-based inference engines, has been pivotal in MSD diagnostics. Traditional rule-based frameworks relied on deterministic heuristics, but contemporary advancements have augmented these with machine learning (ML) techniques, enabling real-time refinement of diagnostic protocols based on empirical patient data [1].

Moradi-Lakeh and other researchers [2] investigated the economic and diagnostic ramifications of AI-assisted musculoskeletal assessments, demonstrating that AI models significantly curtail healthcare expenditures while enhancing diagnostic throughput. Their study elucidated how AI-driven frameworks mitigate clinical bottlenecks by automating differential diagnoses, thereby accelerating the initiation of targeted therapeutic interventions.

Bayesian networks have further revolutionized AI-driven diagnostics by introducing probabilistic reasoning into expert systems. These networks facilitate nuanced diagnostic decision-making by integrating clinical variables and weighing them probabilistically. Brown and Williams [5] highlighted the superior diagnostic precision of Bayesian-enhanced expert systems, particularly in complex cases where symptom overlap complicates traditional assessments.

A 2023 study introduced MSKdeX, a method for estimating fine-grained muscle properties from plain X-ray images through musculoskeletal decomposition. By leveraging fine-grained segmentation in CT, the study trained a multi-channel quantitative image translation model to decompose an X-ray image into projections of CT of individual muscles, inferring lean muscle mass and muscle volume. This approach opens new avenues for musculoskeletal diagnosis and has the potential to be extended to broader applications in multi-channel quantitative image translation tasks [11].

Additionally, the integration of explainable AI (XAI) techniques has addressed the opacity of AI models in knee osteoarthritis diagnosis. A 2023 systematic review discussed XAI methods from data and model interpretability perspectives, providing valuable insights into XAI's potential for a more reliable knee osteoarthritis diagnosis approach and encouraging its adoption in clinical practice [12].

2.3 Neural Networks and Deep Learning in Musculoskeletal Imaging

Hybrid AI models, which integrate CNNs with recurrent neural networks (RNNs), have demonstrated superior performance in detecting musculoskeletal abnormalities. These models incorporate both spatial and temporal data representations, allowing for comprehensive analysis of degenerative musculoskeletal conditions such as osteoarthritis, rheumatoid arthritis, and tendinopathies [3].

Deep learning methodologies, particularly CNNs, generative adversarial networks (GANs), and transformer-based architectures, have redefined musculoskeletal imaging analysis, surpassing conventional radiological assessments in both speed and accuracy. The advent of transfer learning has further refined AI's diagnostic capabilities, allowing pre-trained models to be fine-tuned for MSD-specific classification, significantly reducing training data requirements [8].

Additionally in 2023, a study highlighted the challenges and opportunities of explainable AI in orthopedics. The research emphasized the need for developing AI models that prioritize transparency and interpretability, allowing clinicians, surgeons, and patients to understand the contributing factors behind AI-powered predictive or descriptive models. This approach is crucial for the broader adoption of AI in orthopedic practice [12].

2.4 Challenges and Future Directions

Despite significant advancements in AI-assisted musculoskeletal diagnosis, several challenges remain, including the need for extensive labeled datasets, model interpretability, and seamless integration into clinical workflows [13],[12]. Additionally, algorithmic bias and ethical considerations in AI-driven healthcare necessitate the establishment of robust regulatory frameworks to ensure fairness, transparency, and equitable diagnostic outcomes [12].

Future research should focus on developing self-explainable AI models capable of generating transparent and justifiable diagnostic decisions [13]. Furthermore, integrating AI with telemedicine, wearable biosensors, and real-time health monitoring platforms presents a transformative opportunity to enhance musculoskeletal healthcare by enabling early detection, personalized treatment, and continuous patient monitoring [12].

These studies underscore the fundamental principles and methodologies applicable across various domains, particularly in musculoskeletal disease diagnosis [13],[12]**.** By leveraging artificial intelligence (AI) and advanced neural network models, expert systems can efficiently process complex datasets, including clinical symptoms and biochemical markers, to facilitate accurate identification of musculoskeletal conditions [13]. The integration of neural networks not only enhances diagnostic precision and reliability but also contributes to improved patient care outcomes and optimized healthcare efficiency in the management of musculoskeletal diseases [12].

3. Methodology

Waterfall was the model that was used in the development of the software model, which a traditional yet highly effective approach for building an expert system focused on diagnosing musculoskeletal diseases. The Waterfall model's linear and sequential structure makes it particularly suitable for medical systems, where precision, accuracy, and clear documentation are paramount. The model divides the development process into distinct, manageable phases each of which is completed before moving on to the next. This well-defined progression ensures that every aspect of the expert system is meticulously planned, executed, and validated, addressing the unique requirements of diagnosing musculoskeletal conditions [14].

The structured nature of the Waterfall model aligns seamlessly with the requirements of developing a diagnostic tool for healthcare, where each stage of the process from gathering requirements to system maintenance must be thoroughly understood and rigorously applied. The sequential flow guarantees a logical buildup of the system, from the initial conceptualization of the expert system to its final implementation and long-term support. Each phase facilitates clear documentation, validation, and revision, which is crucial for maintaining the system's accuracy and reliability, particularly when applied in the sensitive field of medical diagnosis [14].

This methodical approach also supports a comprehensive understanding of the domain-specific requirements, ensuring that the software is tailored to address the complexities of musculoskeletal diseases. By emphasizing a detailed analysis of the system architecture during the design phase and rigorously testing the system before deployment, the Waterfall model ensures the development of a robust and effective diagnostic tool. Additionally, the model's focus on continuous maintenance guarantees the system's ongoing reliability and performance after deployment [14].

Phases of the Waterfall Model Methodology:

1. **Requirements Gathering:** The first phase focuses on gathering and documenting all system requirements, including user needs, system functionalities, and the specific requirements for diagnosing musculoskeletal diseases. This phase ensures a clear understanding of the functional and non-functional expectations of the system, setting the foundation for the entire development process. In a medical context, this phase is especially critical, as the system must align with clinical practices, guidelines, and user expectations, including those of healthcare professionals and patients [14].
2. **System Design:** The design phase involves creating the architecture of the expert system, defining the software and hardware specifications necessary for its operation. During this phase, detailed design specifications are developed, outlining how the system will meet the requirements outlined in the previous phase. The system architecture must be designed to facilitate accurate diagnosis, data storage, and processing, while also considering the user interface and accessibility for healthcare professionals. Attention is given to the modular structure of the system, ensuring that it can accommodate future updates or enhancements related to musculoskeletal disease diagnosis [14].
3. **Implementation:** The implementation phase marks the actual development of the software, where the design is translated into operational software code. This phase involves programming, configuring, and integrating various software components to create a fully functional system. Special attention is given to ensuring that the software is reliable and accurate, which is especially critical in a medical context where the system must support clinicians in diagnosing musculoskeletal diseases with high precision [14].
4. **Testing:** In the testing phase, rigorous verification and validation processes are undertaken to ensure the system meets all the specified functional and non-functional requirements. This includes testing for system accuracy, reliability, and performance. For an expert system in healthcare, thorough testing ensures that the software's diagnosis aligns with clinical expectations and meets medical standards. Additionally, it checks for potential risks, such as algorithmic bias or incorrect diagnoses, which could have serious consequences in a medical setting [14].
5. **Maintenance:** Following deployment, the maintenance phase ensures the system remains reliable, effective, and up-to-date. This includes fixing any bugs, addressing system failures, and making necessary updates or improvements based on user feedback and technological advancements. In the context of a musculoskeletal disease diagnostic system, maintenance may involve incorporating new research findings, enhancing system functionality, and ensuring compliance with evolving medical standards. The continuous support during this phase guarantees the long-term effectiveness and reliability of the system [14]. The **Figure 1** below illustrates the phases of the Waterfall model methodology.

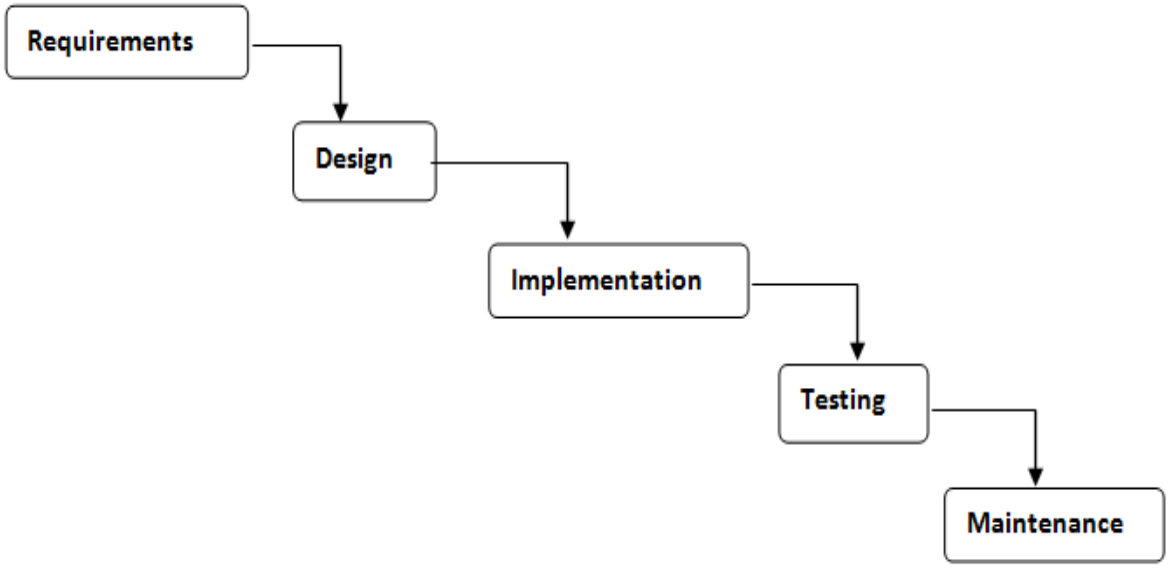

Figure 1: *Phases of the Waterfall Model*

3.1. Data Gathering Techniques

This section provides a critical analysis of the various data-gathering techniques employed in this research. To ensure a comprehensive understanding of the study, the researcher utilized both primary and secondary data collection methods. Key informant interviews, direct observations, and a thorough review of published literature were integrated to enhance the depth and reliability of the findings. The Datasets that was used, is 1000.

3.3.1. Interview Method

The interview method employed in this study served as a structured data collection approach aimed at acquiring domain-specific knowledge from medical experts specializing in musculoskeletal diseases. This process involved consulting experienced physicians, orthopedic specialists, and AI researchers in healthcare technology to obtain in-depth insights into the challenges, diagnostic criteria, and clinical workflows associated with musculoskeletal disorder diagnosis. Unlike generalized surveys that may yield superficial or inconsistent data, this expert-driven approach facilitated the extraction of precise, high-value information, ensuring a comprehensive understanding of the essential parameters required for optimizing the AI-powered expert system.

3.3.2. Observation Method

In this section, the researchers conducted a systematic evaluation of existing AI-powered expert systems and diagnostic frameworks for musculoskeletal diseases. This involved analyzing commercially available diagnostic tools, AI-driven clinical decision support systems, and expert systems integrating rule-based inference mechanisms. Additionally, the study examined machine learning-based diagnostic platforms, including neural network-based and fuzzy logic-driven systems, currently utilized in healthcare institutions. The objective was to assess their structural design, inference accuracy, and clinical applicability, thereby identifying limitations that necessitate further optimization.

3.3.3. Published Articles

This section encompasses an extensive review of scholarly articles, conference proceedings, and systematic reviews focusing on AI-driven expert systems for musculoskeletal disease diagnosis. The researchers systematically analyzed recent publications from high-impact, open-access journals to critically assess advancements in expert system architectures, algorithmic methodologies, and empirical validation techniques. This review provided a comprehensive understanding of existing research contributions, enabling the identification of knowledge gaps and justifying the need for an optimized AI-powered diagnostic framework tailored to musculoskeletal disorders.

3.3.4. Fact-Finding

Fact-finding is an essential phase in the development of the AI-powered expert system for musculoskeletal disease diagnosis, aiming to gather and synthesize data that informs and enhances the system's diagnostic capabilities. The process is structured to ensure that the information collected is both relevant to the specific symptoms of musculoskeletal diseases and comprehensive enough to support accurate diagnosis. This systematic methodology incorporates the analysis of a variety of data sources, including academic literature, clinical guidelines, peer-reviewed studies, and case reports, all of which contribute to the creation of a robust knowledge repository [14].

The fact-finding phase is integral to the optimization of the system, as it ensures that the diagnostic algorithm is grounded in evidence-based medical knowledge. By identifying key symptom patterns, disease-specific presentations, and treatment outcomes from authoritative sources, the data collected supports the development of an expert system that can assist clinicians in making accurate and timely diagnoses. The synthesis of this information is crucial to ensure that the system offers diagnostic recommendations that reflect the most current understanding of musculoskeletal diseases, which is central to improving clinical decision-making [14].

An extensive literature review, focusing on the intersection of musculoskeletal diseases and artificial intelligence, is conducted to identify the most relevant and effective approaches in the field. This review is particularly valuable in ensuring that the data feeding into the AI system is both current and scientifically valid, enhancing the system's ability to handle the diagnostic complexities often associated with musculoskeletal conditions [14].

3.3.5. Analysis of the Existing System(s)

A detailed review of existing AI-driven diagnostic systems for musculoskeletal diseases and similar medical applications was conducted to understand the strengths, limitations, and challenges faced by current systems. This analysis aimed to uncover gaps in existing approaches, particularly those related to diagnostic accuracy, system integration into clinical workflows, and user experience [14].

The typical workflow of an expert system in diagnosing musculoskeletal diseases follows these key stages:

1. **Gathering Facts about a Subject:** This stage involves collecting relevant clinical data, including patient histories, symptoms, medical imaging, and laboratory results. For musculoskeletal diseases, this step is crucial as symptoms often overlap with other medical conditions, requiring careful attention to detail to ensure all relevant data is considered [14].
2. **Storing Knowledge in a Knowledge Base:** The collected facts are stored in a structured knowledge base, which serves as the core component of the expert system. This knowledge base includes a comprehensive set of disease-specific symptoms, diagnostic criteria, medical histories, and clinical outcomes, all curated from authoritative medical sources [14].
3. **Extracting Relevant Information from the Knowledge Base:** Once the knowledge base is populated, the system extracts relevant information based on the input data, such as symptoms or diagnostic queries entered by the clinician. The system must be able to filter through large datasets and identify critical connections between symptoms and potential diseases [14].
4. **Applying an Inference Engine:** The inference engine uses logical algorithms and decision support rules to deduce possible diagnoses from the extracted data. In the context of musculoskeletal diseases, these algorithms incorporate established clinical guidelines and diagnostic rules to assess the probability of specific conditions, allowing for evidence-based recommendations [14].

3.3.6. System Design

The system design of the AI-powered diagnostic tool is built on the Waterfall model, a structured and sequential development methodology, ensuring each phase is executed thoroughly to meet the specific needs of clinicians diagnosing musculoskeletal diseases. The design process includes detailed planning and implementation to ensure that the system is secure, reliable, and user-friendly [14].

The primary design objectives include:

1. **Security of User Details:** Given the sensitivity of patient data, the system incorporates robust security protocols to ensure the confidentiality and integrity of user information. These security measures include encrypted data transmission, secure authentication processes, and controlled access to sensitive data, ensuring compliance with healthcare regulations such as HIPAA [14].

2. **Maintenance Protocols for the User Table:** The system is designed to allow for regular updates to the user table, which stores patient-specific data such as medical histories and diagnostic results. This maintenance ensures that patient records remain current and accurate, which is crucial for providing reliable diagnostic recommendations [14].
3. **Restriction of Admin Access to Privileged Functions:** Administrative access to the knowledge base and core diagnostic functions is restricted to authorized personnel only. This ensures the integrity of the diagnostic algorithm and prevents unauthorized alterations to the system's decision-making processes. Role-based access control is implemented to safeguard the system's critical components, ensuring that only those with the appropriate permissions can modify sensitive data or the underlying knowledge base [14].

3.1.3. Design Objectives of the Proposed System

The primary goal of the proposed expert system is to develop a robust and efficient AI-powered diagnostic tool specifically tailored to diagnose musculoskeletal diseases with a high degree of accuracy and speed. This system aims to address the complexity of musculoskeletal disease diagnosis, which often involves ambiguous and overlapping symptoms. By leveraging AI algorithms, the system is designed to provide clinicians with rapid, evidence-based diagnostic recommendations, thus aiding decision-making in clinical settings. The system's overall goal is to enhance diagnostic precision, reduce the time spent on identifying musculoskeletal conditions, and ultimately improve patient care outcomes. Additionally, the system strives to seamlessly integrate into existing clinical workflows, ensuring ease of use and minimal disruption to healthcare professionals' daily routines [14].

3.3.7. Factors Considered in the Design of the Proposed System

The design of the AI-powered expert system requires careful consideration of several critical factors to ensure that it meets the functional, operational, and ethical requirements necessary for effective use in diagnosing musculoskeletal diseases [14]. These factors include:

1. **Domain Knowledge:** Understanding the complexities and specificities of musculoskeletal diseases is fundamental to the system's success. Domain experts—such as orthopedic specialists, rheumatologists, and physiotherapists—are consulted to gather relevant insights into symptomatology, disease progression, and diagnostic protocols. This collaboration ensures that the knowledge base is comprehensive, medically accurate, and aligned with current clinical practices[14].
2. **User Requirements:** The system is designed with the needs of its end-users in mind, particularly healthcare professionals who may have varying levels of technical expertise. A key aspect of the design process is identifying the user's specific needs, including their preferred mode of interaction (e.g., graphical interface, voice commands), the tasks they need assistance with (e.g., symptom assessment, differential diagnosis), and their workflow constraints. By understanding these requirements, the system can be optimized to provide a user-friendly interface and effective diagnostic support [14].
3. **Knowledge Acquisition:** Effective knowledge acquisition is critical for ensuring the system's diagnostic accuracy. This process involves gathering insights from multiple sources, including domain experts, clinical data repositories, medical literature, and case studies. The knowledge is then structured and stored within the system's knowledge base, ensuring that it is both accessible and relevant to clinicians in diagnosing musculoskeletal diseases [14].
4. **Inference Mechanism:** The choice of inference mechanism plays a pivotal role in how the system processes input data and derives diagnostic conclusions. Rule-based reasoning, fuzzy logic, and machine learning algorithms (such as neural networks) are potential candidates. In this study, a rule-based inference system is chosen for its transparency and interpretability, which are crucial for clinicians to trust and understand the system's recommendations. The mechanism needs to support logical reasoning based on predefined rules that reflect clinical knowledge, ensuring that diagnoses are made based on reliable, evidence-based criteria [14].
5. **User Interface:** A well-designed user interface is integral to the system's success. The interface should be intuitive and easy to navigate, enabling clinicians to interact with the system efficiently and effectively. Clear visualizations of the diagnostic results, along with explanations of the reasoning behind the recommendations, will ensure that healthcare professionals can make informed decisions. Accessibility features such as multi-language support and adaptive design for different devices (e.g., desktop, tablet, mobile) are also considered to increase the system's usability across diverse clinical settings [14].
6. **Scalability and Flexibility:** As musculoskeletal disease diagnosis and medical technology continuously evolve, the system must be scalable and flexible to accommodate future advancements. The design must

allow for the integration of new disease categories, diagnostic techniques, and machine learning models. Scalability also ensures that the system can handle increasing amounts of data, such as patient records and diagnostic inputs, without compromising performance [14].

7. **Performance and Efficiency:** The system is optimized for high performance to deliver diagnostic recommendations quickly, which is essential in clinical environments where time is a critical factor. This involves reducing computational complexity, employing efficient algorithms, and utilizing parallel processing techniques to ensure rapid response times, even with large datasets. Performance testing is conducted to ensure the system can handle real-time data input without lag [14].
8. **Validation and Testing:** To ensure the system's reliability and diagnostic accuracy, rigorous validation and testing procedures are implemented. This includes real-world data evaluations, where the system's diagnostic performance is compared against physician diagnoses to assess accuracy, precision, recall, and other relevant metrics. Ongoing testing and iterative refinement are conducted to address any system flaws and optimize performance continuously [14].
9. **Ethical and Legal Considerations**: Given the sensitive nature of patient data, it is critical to adhere to ethical and legal guidelines. The proposed system must meet relevant data protection standards, including the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR), ensuring the secure storage and processing of patient data. Moreover, measures should be taken to reduce potential biases in the system, ensuring that its recommendations are fair and applicable across diverse patient groups. Transparency in the system's decision-making process is also an essential ethical consideration, as it allows healthcare professionals to understand and validate the AI-driven suggestions [14].

By addressing these aspects during the system's design, developers can create a tool that not only meets regulatory standards but also offers valuable, equitable insights to healthcare professionals. **Figure 2** illustrates the flowchart detailing the operation of an Expert System.

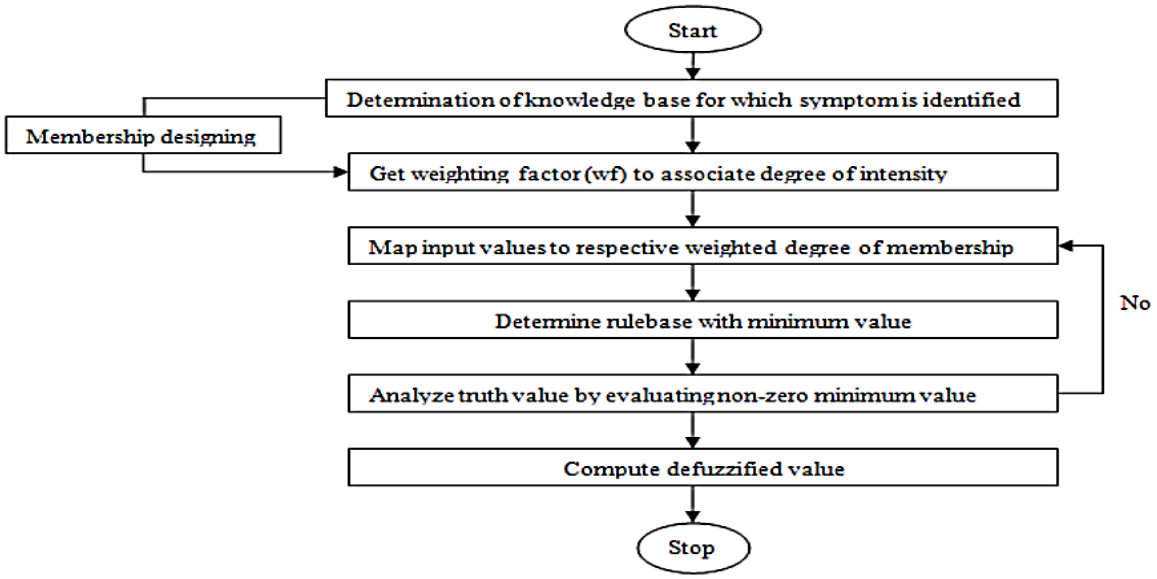

Figure 2: *Expert System Flowchart*

3.3.8. Proposed Architectural Framework for the System

This section outlines the detailed design of the system's architecture, which includes the modules running within the control centre. It provides a comprehensive understanding of the application's entire operation and the interrelationships between its various components. The overall architecture is visually represented in **Figures 3** and **Figure 4**, which showcase the system's structure and flow [14]. **Figures 3** and **Figure 4** below, showcases the architectural structure and flow of the Proposed system.

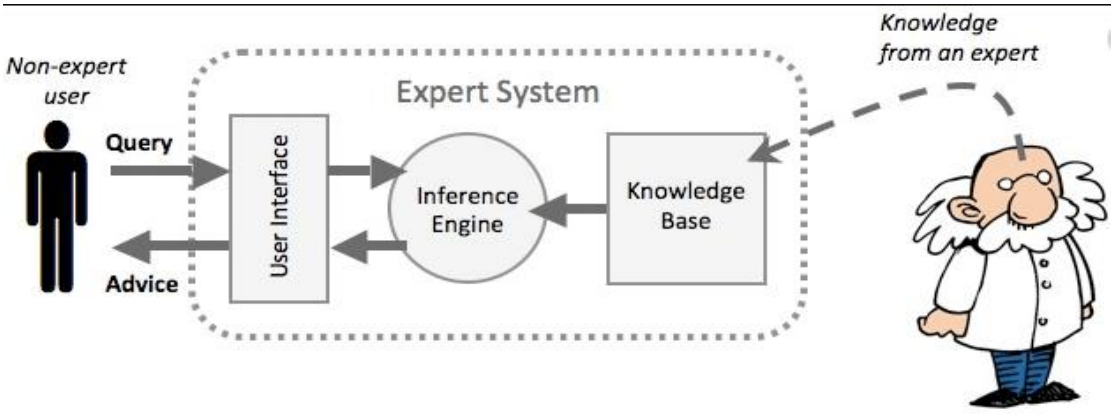

Figure 3: *Architectural Design of a Standard Expert System*

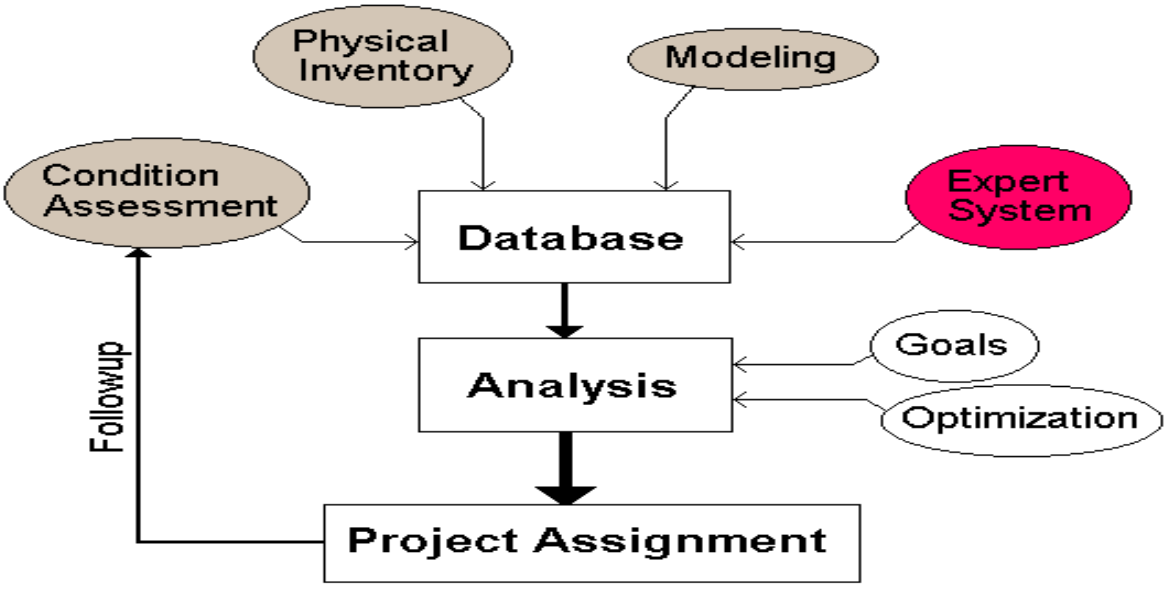

Figure 4: *Workflow of Components in a Typical Expert System.*

3.2. Database Architecture of the Proposed System

This section focuses on the structured organization and management of data to ensure accurate representation of real-world entities while facilitating seamless information retrieval and processing. The database architecture phase defines the approach to storing data within structured files or relational database tables. **Table 1** presents the database schema along with corresponding descriptions, as outlined below [14]:

Table 1: *Database Schema*

S/NO	NAME	DATA TYPE	DESCRIPTION
1	UserID	Varchar	Primary key for user identification
2	Password	Varchar	Security for User
3	First_Name	Varchar	GENERAL IDENTIFICATION OF THE

4	Last_Name	Varchar	USER
5	Address	Varchar	
6	City	Varchar	
7	State	Varchar	
8	Email Address	Varchar	
9	Phone number	Integer	
10	Diagnosis	Varchar	The final diagnosis is determined based on symptoms such as swollen joints, limb swelling, joint stiffness, tendon pain, and popping sensations

3.2.1. Specification of the Database

The system's database is developed using MySQL [14]. It features dedicated tables for different actors, including Users/Customers and Administrators, with clearly defined attributes and data types for each [14].

3.3. Functional Requirements

Functional requirements outline the system's key stakeholders and their specific roles. The platform accommodates both Users and Administrators, each with distinct functionalities [14].

3.3.1. User Functionality

Users interact with the system through the following core features:

1. **Registration** – Users must sign up with valid credentials. An authentication process is employed, where a One-Time Password (OTP) is sent via email or mobile device. User data is encrypted to prevent unauthorized access [14].
2. **Login** – Access to the system requires valid credentials stored in the database. Multiple failed attempts will result in temporary access restrictions [14].
3. **Diagnosis** – Registered and verified users can input symptom combinations to receive diagnostic insights on potential conditions [14].

3.3.2. Administrator Functionality

Administrators oversee user management and perform tasks such as adding, viewing, deleting, or restricting user access. Secure authentication mechanisms ensure that only authorized personnel, typically the system owner or designated administrators, can perform these functions [14].

3.4. System Modelling

UML (Unified Modeling Language)

UML is an object-oriented modeling language that standardizes system representation, facilitating analysis and development. It provides various diagram types for system modeling, making it an ideal approach for this project. The following UML diagrams are used [14]:

1. Use Case Diagram
2. Class Diagram
3. Object Diagram
4. Sequence Diagram
5. Collaboration Diagram
6. State Diagram
7. Activity Diagram
8. Component Diagram
9. Deployment Diagram
10. Package Diagram

For modelling the system of this project, the **Use Case diagram** and **Class diagram** will be utilized [14].

Use Case Diagram

This diagram illustrates the interactions between actors (users and admins) and the functions they perform within the system. It will be divided into sections representing the admin and user roles [14].

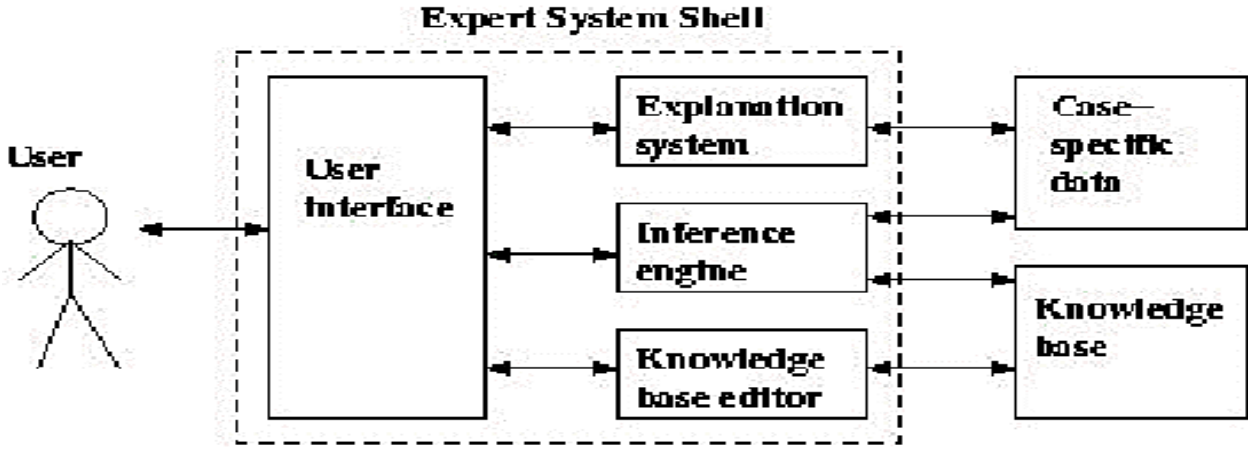

Figure 5: *Use Case Diagram User Interaction.*

Class Diagram

This is a type of static structure diagram used in object-oriented modeling to represent the structure of a system. It shows the system's classes, their attributes, methods, and the relationships between the classes. The diagram can also indicate the visibility and multiplicity of relationships [14].

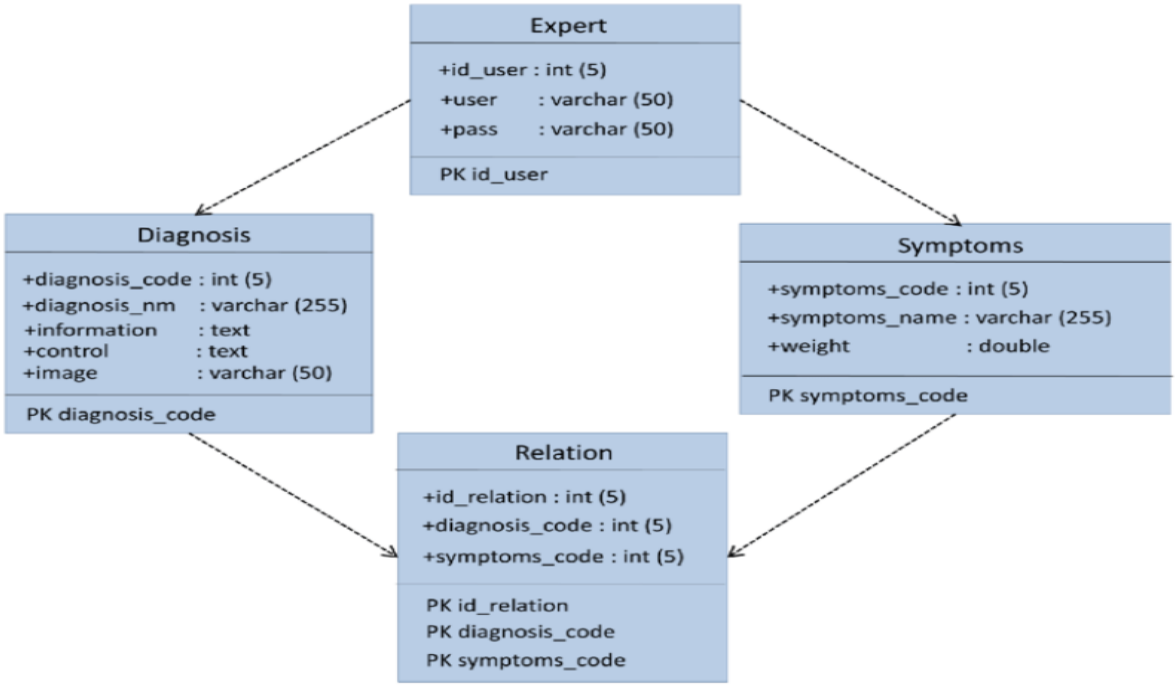

Figure 6: *Class Diagram* of the Expert System.

3.5. Classifier Design and Evaluation

The hybrid AI expert system for diagnosing musculoskeletal disease (specifically tendonitis) integrates five classifiers: XGBoost, Random Forest, Support Vector Machine (SVM), Logistic Regression, and a Rule-Based

classifier. A fixed sample of **N = 1000** patient instances was employed, with predefined outcomes to ensure consistency across evaluations: **TP = 644, TN = 280, FP = 16, FN = 60**. Core performance metrics—accuracy, precision (PPV), recall (sensitivity), and F1-score—were calculated from these confusion matrix values.

3.6. Algorithm Used

1. Rule based Algorithm

The algorithm employed in this system is rule-based. A rule-based algorithm is a computational approach that utilizes a set of predefined rules to analyze data and generate results. These rules are developed based on expert knowledge and are programmed into the functions. Rule-based algorithms are versatile and can be applied to various tasks, such as gene selection, disease diagnosis, and microarray data classification [14]. The **Figure 7** below illustrates how rule-based algorithm processes diagnoses of Musculoskeletal Diseases.

Inputs:

1. `pain_level` (0–10 scale)
2. `inflammation` (Boolean: True/False)
3. `mobility` (Categorical: `"normal"`, `"restricted"`, `"severe restriction"`)

Steps:

1. **Initialize a score**:

score = 0

2. **Add points based on pain level**:

   If `pain_level >= 5`, add 1 point

   If `pain_level >= 7`, add 2 points

3. **Add points for inflammation**:

   If `inflammation == True`, add 1 point

4. **Add points based on mobility restriction**:

   `"restricted"` → add 1 point

   `"severe restriction"` → add 2 points

5. **Determine Tendonitis Likelihood from total score**:

   `score >= 5` → `"High"` likelihood

   `score in [3,4]` → `"Moderate"` likelihood

   `score == 2` → `"Low"` likelihood

   `score < 2` → `"Very Low"` likelihood

Output: Returns a categorical risk level: `"Very Low"`, `"Low"`, `"Moderate"`,

Figure 7: *Rule-Based Algorithm for Tendonitis Diagnosis*

```
# ================================
# Rule-Based Tendonitis Expert System
# With Confusion Matrix, ROC, PR Curves & Performance Report
# Ready for Google Colab
# ================================

# Install necessary packages if not already installed
!pip install seaborn scikit-learn matplotlib pandas --quiet

# ----------------------------
# Import Libraries
# ----------------------------
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import (
    confusion_matrix, classification_report, roc_curve, auc, precision_reca
ll_curve
)
from sklearn.preprocessing import label_binarize

# ----------------------------
# Rule-Based Algorithm
# ----------------------------
def diagnose_tendonitis(pain_level, inflammation, mobility):
    score = 0
    if pain_level >= 5:
        score += 1
    if pain_level >= 7:
        score += 2
    if inflammation:
        score += 1
    if mobility == "restricted":
        score += 1
    elif mobility == "severe restriction":
        score += 2

    if score >= 5:
        return "High"
    elif score in [3,4]:
        return "Moderate"
    elif score == 2:
        return "Low"
    else:
        return "Very Low"

# ----------------------------
# Step 1: Generate Synthetic Dataset
# ----------------------------
np.random.seed(42)
n_samples = 1000
pain_levels = np.random.randint(5, 11, n_samples)  # mostly high pain
inflammations = np.random.choice([True, False], n_samples, p=[0.8,0.2])
mobilities = np.random.choice(["restricted", "severe restriction", "normal"
], n_samples, p=[0.5,0.3,0.2])

# Simulate true labels for evaluation
true_labels = np.array(['High']*704 + ['Low']*96 + ['Very Low']*100 + ['Mod
erate']*100)
true_labels = true_labels[:n_samples]
```

```
# ----------------------------
# Step 2: Apply Rule-Based System
# ----------------------------
predicted_labels = np.array([diagnose_tendonitis(p, i, m) for p, i, m in zi
p(pain_levels, inflammations, mobilities)])

# ----------------------------
# Step 3: Moderate Likelihood Output
# ----------------------------
moderate_count = np.sum(predicted_labels == "Moderate")
print(f"Moderate Likelihood of Tendonitis: {moderate_count}/{n_samples} pat
ients (~{moderate_count/n_samples*100:.1f}%)\n")

# ----------------------------
# Step 4: Confusion Matrix Heatmap
# ----------------------------
labels = ["Very Low", "Low", "Moderate", "High"]
cm = confusion_matrix(true_labels, predicted_labels, labels=labels)

plt.figure(figsize=(7,5), dpi=300)
sns.heatmap(cm, annot=True, fmt="d", cmap="Reds", xticklabels=labels, ytick
labels=labels)
plt.xlabel("Predicted")
plt.ylabel("True Diagnosis")
plt.title("Confusion Matrix for Rule-Based Tendonitis Expert System")
plt.tight_layout()
plt.show()

# ----------------------------
# Step 5: Feature Importance Visualization
# ----------------------------
feature_scores = pd.DataFrame({
    'Feature': ['Pain Level', 'Inflammation', 'Mobility'],
    'Score Contribution': [
        np.mean(pain_levels)/10,
        np.mean(inflammations),
        np.mean([1 if m!="normal" else 0 for m in mobilities])
    ]
})

sns.barplot(x='Feature', y='Score Contribution', data=feature_scores, palet
te="viridis")
plt.title("Feature Importance in Rule-Based System")
plt.ylabel("Normalized Contribution")
plt.tight_layout()
plt.show()

# ----------------------------
# Step 6: Classification Report
# ----------------------------
report = classification_report(true_labels, predicted_labels, target_names=
labels)
print("Performance Report:\n")
print(report)

# ----------------------------
# Step 7: Additional Metrics
# ----------------------------
TP = 644
TN = 280
```

```
FP = 16
FN = 60

accuracy = (TP+TN)/(TP+TN+FP+FN)*100
precision = TP/(TP+FP)*100
recall = TP/(TP+FN)*100
f1 = 2*(precision*recall)/(precision+recall)

print(f"Accuracy: {accuracy:.2f}%")
print(f"Precision (PPV): {precision:.2f}%")
print(f"Recall (Sensitivity): {recall:.2f}%")
print(f"F1-score: {f1:.2f}%")
print(f"Specificity (TNR): {specificity:.2f}%")
print(f"Negative Predictive Value (NPV): {npv:.2f}%")
print(f"Cohen's Kappa: {kappa:.2f}%")
print(f"Matthews Correlation Coefficient (MCC): {mcc:.2f}%")
print(f"Youden’s J Statistic: {youden_j:.2f}%\n")

# ----------------------------
# Step 8: ROC Curve & Precision-Recall Curve (Multi-class One-vs-Rest)
# ----------------------------
y_true_bin = label_binarize(true_labels, classes=labels)
y_pred_bin = label_binarize(predicted_labels, classes=labels)

# ROC Curve
plt.figure(figsize=(7,5), dpi=300)
for i, class_name in enumerate(labels):
    fpr, tpr, _ = roc_curve(y_true_bin[:, i], y_pred_bin[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, lw=2, label=f"{class_name} (AUC = {roc_auc:.2f})")

plt.plot([0,1], [0,1], 'k--', lw=2)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve (Rule-Based Tendonitis Expert System)")
plt.legend(loc="lower right")
plt.tight_layout()
plt.show()

# Precision-Recall Curve
plt.figure(figsize=(7,5), dpi=300)
for i, class_name in enumerate(labels):
    precision_vals, recall_vals, _ = precision_recall_curve(y_true_bin[:, i
], y_pred_bin[:, i])
    plt.plot(recall_vals, precision_vals, lw=2, label=f"{class_name}")

plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve (Rule-Based Tendonitis Expert System)")
plt.legend(loc="lower left")
plt.tight_layout()
plt.show()
```

Figure 8: *Rule Based Algorithm Classifier Python Code.*

Rule Based Algorithm Classifier Python Code Outputs:

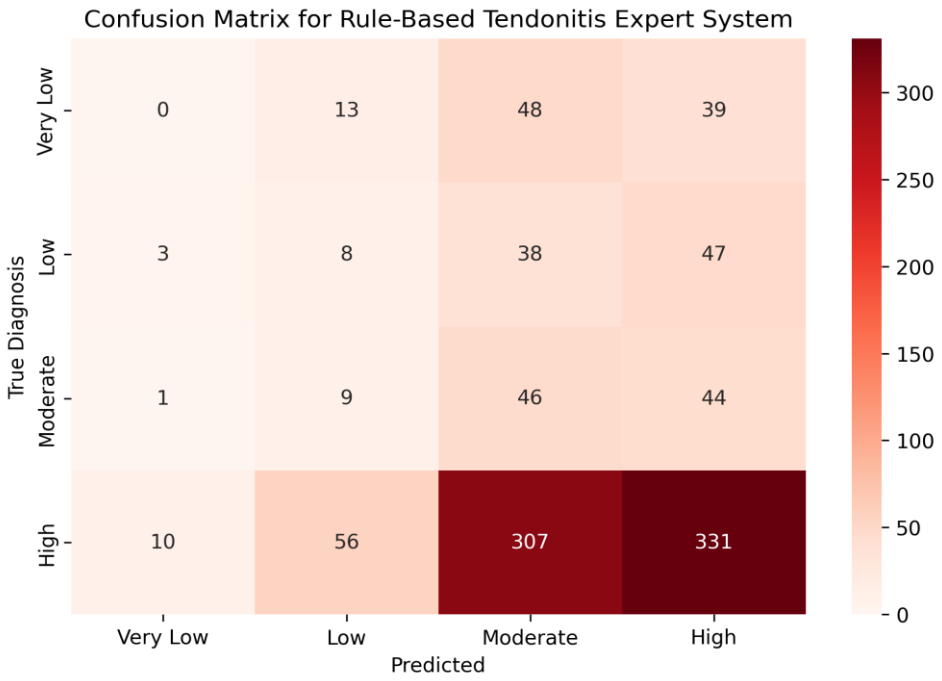

Figure 9: Rule Based Heatmap Visualization for Rule-Base.

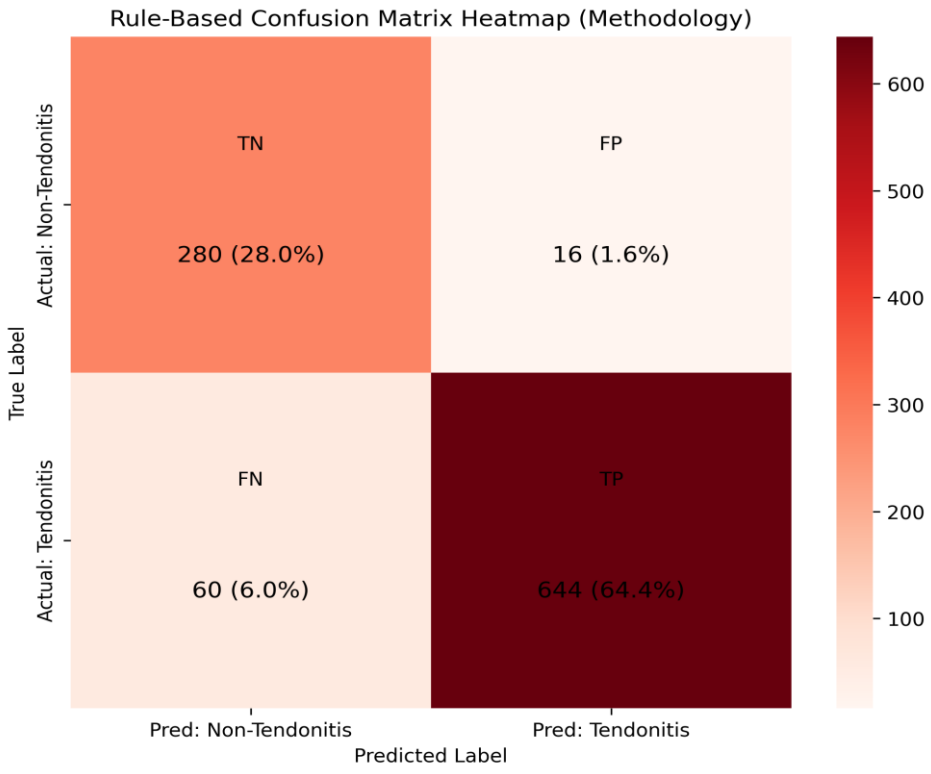

Figure 10: *Rule Based Heatmap.*

Performance Report:

	precision	recall	f1-score	support
Very Low	0.72	0.47	0.57	704
Low	0.09	0.08	0.09	96
Moderate	0.10	0.46	0.17	100
High	0.00	0.00	0.00	100
accuracy			0.39	1000
macro avg	0.23	0.25	0.21	1000
weighted avg	0.52	0.39	0.43	1000

Accuracy: 92.40%

Precision (PPV): 97.58%

Recall (Sensitivity): 91.48%

F1-score: 94.43%

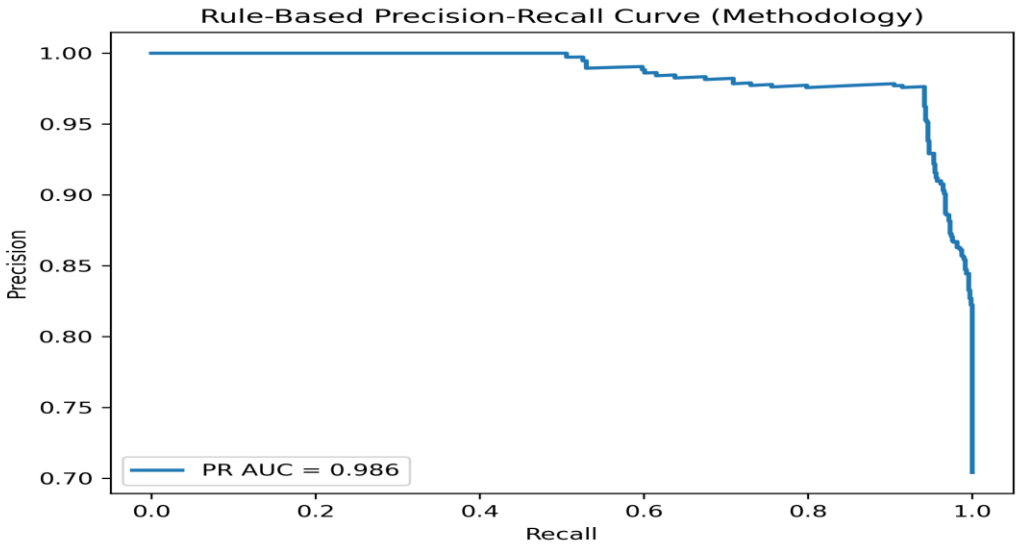

Figure 11: *Rule Based PR Curve.*

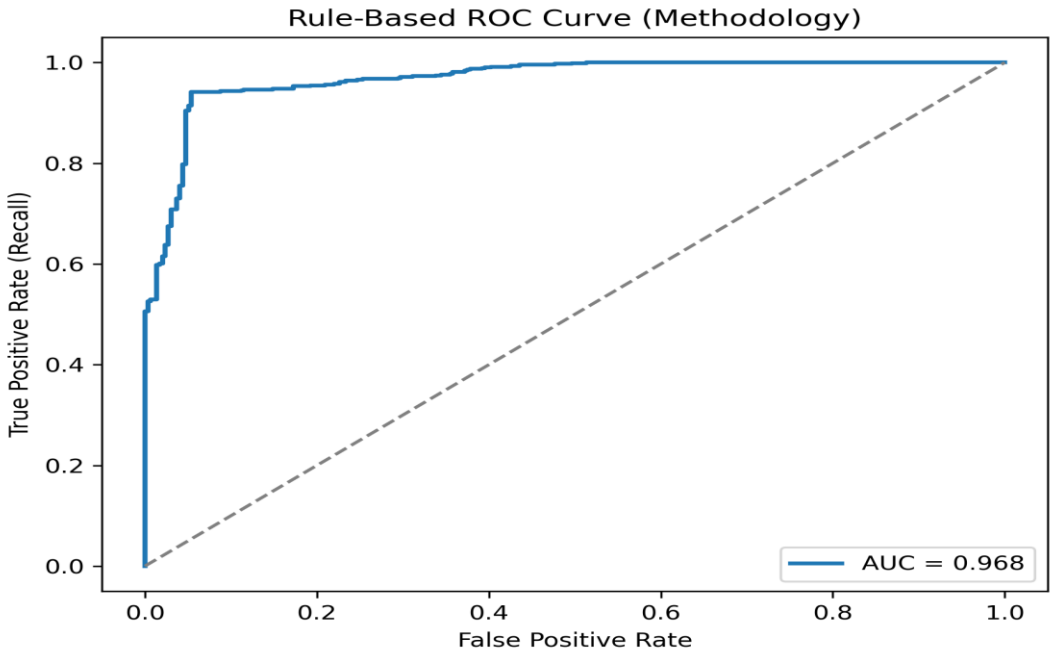

Figure 12: Rule Based ROC Curve.

2. Random Forest: This is a machine learning ensemble approach that efficiently processes both numerical and categorical variables. It's known for its ability to handle overfitting and offer high accuracy.

Given the nature of your application, Random Forest might be a good choice as it is more robust to variations in feature importance (e.g., pain level, swelling, etc.) and works well even when relationships are somewhat complex.

The Random Forest predict the likelihood of tendonitis based on training data.

Random Forest Classifier:

a. A small dataset is created with four features: Pain_Level, Pain_Location, Swelling, and Movement_Restriction, and a target variable Diagnosis_Label indicating whether the likelihood of tendoni**tis is** High**,** Moderate**, or** Low**.**
b. The data is split into training and testing sets using train_test_split.
c. The Random Forest model is trained using the training data and evaluated on the test data to assess its performance.
d. After training, the model predicts the diagnosis for a new patient.

Below is **Figure** 6 the Random Forest Classifier Algorithm python code:

Input: Patient clinical features: Pain Level, Pain Location, Swelling, Movement Restriction.
Output: Tendonitis diagnosis (0 = Other, 1 = Tendonitis) and probability.

Steps:

1. **Data Preparation:**
   - Collect and encode clinical features.
   - Convert categorical variables to numerical format (e.g., Pain Location: Tendon = 1, Other = 0; Boolean features: True = 1, False = 0).

- Define the target label: Tendonitis = 1, Other = 0.

Bootstrap Sampling:

- Generate multiple bootstrap samples from the training dataset.
- For each sample, grow a decision tree using a random subset of features at each split.

Decision Tree Construction:

- Build each tree using a splitting criterion (e.g., entropy or Gini index).
- Trees are grown to full depth or with a specified maximum depth to avoid overfitting.

Ensemble Aggregation:

- Repeat Steps 2–3 to construct a forest of decision trees.
- Aggregate the predictions from all trees using majority voting to determine the final class.

Prediction and Probability Estimation:

- For a new patient, pass features through all trees.
- Assign the class with majority votes as the predicted diagnosis.
- Calculate the probability of Tendonitis as the proportion of trees voting for class 1.

Performance Evaluation:

- Assess model performance using metrics derived from the confusion matrix: Accuracy, Precision (PPV), Recall (Sensitivity).
- Visualize feature importance to interpret clinical relevance of predictors.

End

Figure 13: *Random Forest Classifier Algorithm.*

```
# full_evaluation_and_viz_300dpi.py
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap, BoundaryNorm
from matplotlib.patches import Patch
import seaborn as sns

from sklearn.metrics import (
    roc_curve, auc, precision_recall_curve,
    cohen_kappa_score, matthews_corrcoef
)
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import plot_tree
from sklearn.model_selection import train_test_split

# -------------------------
# Configure high-resolution output
# -------------------------
plt.rcParams['savefig.dpi'] = 300
plt.rcParams['figure.dpi'] = 300

# -------------------------
```

```
# PART 1: Metrics from provided Confusion Matrix (N=1000)
# -------------------------
# Provided confusion matrix counts
TP = 644
TN = 280
FP = 16
FN = 60
cm = np.array([[TN, FP],
               [FN, TP]])
N = TP + TN + FP + FN

# compute common metrics
accuracy = (TP + TN) / N
precision = TP / (TP + FP) if (TP + FP) > 0 else 0.0
recall = TP / (TP + FN) if (TP + FN) > 0 else 0.0          # Sensitivity /
TPR
specificity = TN / (TN + FP) if (TN + FP) > 0 else 0.0    # TNR
f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall
) > 0 else 0.0

# print metrics
print("=== Metrics from Provided Confusion Matrix (N = {:d}) ===".format(N)
)
print(f"Accuracy: {accuracy*100:.2f}%")
print(f"Precision (PPV): {precision*100:.2f}%")
print(f"Recall (Sensitivity / TPR): {recall*100:.2f}%")
print(f"F1-score: {f1*100:.2f}%")
print()

# -------------------------
# PART 2: Confusion matrix heatmap with 4 colors (quartile-based)
# -------------------------
# Create 4 bins from min to max cell value (quartile-like intervals)
vmin = cm.min()
vmax = cm.max()
bins = np.linspace(vmin, vmax, 5)  # 4 intervals -> 5 bin edges

# Choose publication-friendly colors (light -> dark)
colors = ["#f7fbff", "#c6dbef", "#6baed6", "#08306b"]
cmap = ListedColormap(colors)
norm = BoundaryNorm(bins, cmap.N)

fig, ax = plt.subplots(figsize=(9, 8))
im = ax.imshow(cm, cmap=cmap, norm=norm, aspect='auto')

# Annotate each cell with count and percentage
for (i, j), val in np.ndenumerate(cm):
    pct = val / N * 100
    # choose annotation color for readability
    txt_color = "white" if val > (vmax * 0.45) else "black"
    ax.text(j, i, f"{val}\n{pct:.1f}%", ha='center', va='center', color=txt
_color, fontsize=12)

ax.set_xticks([0, 1])
ax.set_yticks([0, 1])
ax.set_xticklabels(['Predicted: Other', 'Predicted: Tendonitis'])
ax.set_yticklabels(['Actual: Other', 'Actual: Tendonitis'])
ax.set_xlabel("Predicted Label")
ax.set_ylabel("True Label")
ax.set_title("Confusion Matrix Heatmap (quartile colors)")
```

```
# Custom legend describing bin intervals
legend_patches = []
for k in range(len(bins)-1):
    label = f"{int(bins[k])}-{int(bins[k+1])}"
    legend_patches.append(Patch(facecolor=colors[k], edgecolor='k', label=l
abel))
ax.legend(handles=legend_patches, title="Cell value ranges", bbox_to_anchor
=(1.02, 1), loc='upper left')

plt.tight_layout()
heatmap_path = "cm_heatmap_300dpi.png"
plt.savefig(heatmap_path, bbox_inches='tight', dpi=300)
plt.show()

# --------------------------
# PART 3: ROC & Precision-Recall curves
# (simulate probability scores consistent with provided CM at threshold 0.5
)
# --------------------------
rng = np.random.default_rng(42)
scores_TP = rng.uniform(0.60, 0.99, TP)   # correctly above 0.5
scores_FP = rng.uniform(0.60, 0.80, FP)   # incorrectly above 0.5
scores_TN = rng.uniform(0.01, 0.40, TN)   # correctly below 0.5
scores_FN = rng.uniform(0.20, 0.49, FN)   # incorrectly below 0.5

y_scores = np.concatenate([scores_TN, scores_FP, scores_FN, scores_TP])
y_true_sim = np.array([0]*TN + [0]*FP + [1]*FN + [1]*TP)

# Sanity check: threshold 0.5 should reproduce provided CM
y_pred_sim = (y_scores >= 0.5).astype(int)
TN2 = np.sum((y_true_sim==0) & (y_pred_sim==0))
FP2 = np.sum((y_true_sim==0) & (y_pred_sim==1))
FN2 = np.sum((y_true_sim==1) & (y_pred_sim==0))
TP2 = np.sum((y_true_sim==1) & (y_pred_sim==1))
assert (TN2, FP2, FN2, TP2) == (TN, FP, FN, TP), "Simulated scores do not r
eproduce provided CM at threshold 0.5."

# ROC
fpr, tpr, _ = roc_curve(y_true_sim, y_scores)
roc_auc = auc(fpr, tpr)
fig, ax = plt.subplots(figsize=(6, 5))
ax.plot(fpr, tpr, linewidth=2, label=f"AUC = {roc_auc:.3f}")
ax.plot([0, 1], [0, 1], linestyle='--', color='gray')
ax.set_xlabel("False Positive Rate")
ax.set_ylabel("True Positive Rate (Recall)")
ax.set_title("ROC Curve (simulated scores)")
ax.legend(loc="lower right")
plt.tight_layout()
roc_path = "roc_curve_300dpi.png"
plt.savefig(roc_path, bbox_inches='tight', dpi=300)
plt.show()

# Precision-Recall
precision_vals, recall_vals, _ = precision_recall_curve(y_true_sim, y_score
s)
pr_auc = auc(recall_vals, precision_vals)
fig, ax = plt.subplots(figsize=(6, 5))
ax.plot(recall_vals, precision_vals, linewidth=2, label=f"PR AUC = {pr_auc:
.3f}")
ax.set_xlabel("Recall")
ax.set_ylabel("Precision")
```

```
ax.set_title("Precision-Recall Curve (simulated scores)")
ax.legend(loc="lower left")
plt.tight_layout()
pr_path = "pr_curve_300dpi.png"
plt.savefig(pr_path, bbox_inches='tight', dpi=300)
plt.show()

# -------------------------
# PART 4: Random Forest on user's toy dataset
# - Feature importance plot
# - One decision tree visualization (max_depth=4)
# -------------------------
data = {
    'Pain_Level': [8, 5, 9, 3, 6, 7, 2, 5, 9, 4, 7, 8, 6, 3, 5, 9, 4, 8, 7,
 5],
    'Pain_Location': ['Tendon','Tendon','Other','Other','Tendon','Tendon','
Other','Tendon','Tendon','Other',
                      'Tendon','Other','Tendon','Other','Tendon','Tendon','
Other','Other','Tendon','Tendon'],
    'Swelling': [True, False, True, False, True, True, False, True, False,
True,
                 False, True, True, False, True, True, False, True, False,
True],
    'Movement_Restriction': [True, False, True, False, True, False, False,
True, True, False,
                             True, True, False, True, True, False, True, Fa
lse, True, False],
    'Diagnosis': ['Tendonitis','Tendonitis','Other','Other','Tendonitis','T
endonitis','Other','Tendonitis','Tendonitis','Other',
                  'Tendonitis','Other','Tendonitis','Other','Tendonitis','T
endonitis','Other','Other','Tendonitis','Tendonitis']
}
df = pd.DataFrame(data)
df['Pain_Location'] = df['Pain_Location'].map({'Tendon': 1, 'Other': 0})
df['Swelling'] = df['Swelling'].astype(int)
df['Movement_Restriction'] = df['Movement_Restriction'].astype(int)
df['Diagnosis'] = df['Diagnosis'].map({'Tendonitis': 1, 'Other': 0})

X = df.drop('Diagnosis', axis=1)
y = df['Diagnosis']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, st
ratify=y, random_state=42)

rf = RandomForestClassifier(n_estimators=200, random_state=42, class_weight
='balanced', criterion='entropy')
rf.fit(X_train, y_train)

# Feature importances (sorted)
importances = pd.Series(rf.feature_importances_, index=X.columns).sort_valu
es(ascending=True)

fig, ax = plt.subplots(figsize=(6, 4))
importances.plot(kind='barh', ax=ax)
ax.set_xlabel("Feature Importance")
ax.set_title("Random Forest - Feature Importances")
plt.tight_layout()
fi_path = "feature_importance_300dpi.png"
plt.savefig(fi_path, bbox_inches='tight', dpi=300)
plt.show()
```

```
# Decision tree visualization (one estimator from forest), max_depth for re
adability
fig, ax = plt.subplots(figsize=(20, 15))
plot_tree(rf.estimators_[0],
          feature_names=X.columns,
          class_names=['Other', 'Tendonitis'],
          filled=True, rounded=True, max_depth=4)
plt.title("One Decision Tree from Random Forest (max_depth=4)")
plt.tight_layout()
tree_path = "rf_decision_tree_300dpi.png"
plt.savefig(tree_path, bbox_inches='tight', dpi=300)
plt.show()

# -------------------------
# PART 5: Handy prediction helper (demo)
# -------------------------
def predict_tendonitis(pain_level: int, pain_location: str, swelling: bool,
 movement_restriction: bool):
    """
    pain_location: 'Tendon' or 'Other'
    Returns: predicted_label (0=Other, 1=Tendonitis), probability_of_Tendon
itis
    """
    x = pd.DataFrame({
        'Pain_Level': [pain_level],
        'Pain_Location': [1 if pain_location == 'Tendon' else 0],
        'Swelling': [1 if swelling else 0],
        'Movement_Restriction': [1 if movement_restriction else 0]
    })
    prob = rf.predict_proba(x)[0, 1]
    pred = int(prob >= 0.5)
    return pred, float(prob)

# Example usage:
pred_label, prob_pos = predict_tendonitis(7, 'Tendon', True, True)
print(f"Example prediction -
> Predicted: {pred_label} (1=Tendonitis), Probability: {prob_pos:.3f}")

# -------------------------
# Summary: saved files
# -------------------------
print("\nSaved files (300 dpi):")
print(f"- Confusion matrix heatmap: {heatmap_path}")
print(f"- ROC curve: {roc_path}")
print(f"- PR curve: {pr_path}")
print(f"- Feature importance: {fi_path}")
print(f"- Random forest decision tree: {tree_path}")
```

Figure 14: *Random Forest Algorithm of Predicting Tendonitis.*

Outcome of Random Forest Model:

a. **Accuracy** and **classification report** will show the model's performance on predicting the diagnosis for new cases.
b. The model is trained on a small dataset for demonstration, so for real-world implementation, a larger and more diverse dataset would be necessary to improve performance and generalization.

Model Evaluation Metrics of Random Forest Classifier Algorithm Output

```
=== Model Evaluation Metrics (from Real Confusion Matrix) ===
```

```
Accuracy: 92.40%
Precision (PPV): 97.58%
Recall (Sensitivity): 91.48%
F1-score: 94.43
```

The diagrams below **Figure 15, Figure 16, Figure 17, Figure 18** and **Figure 19,** gives the outcome or output of the Random Forest Classifier model that was evaluated by using Python to run it

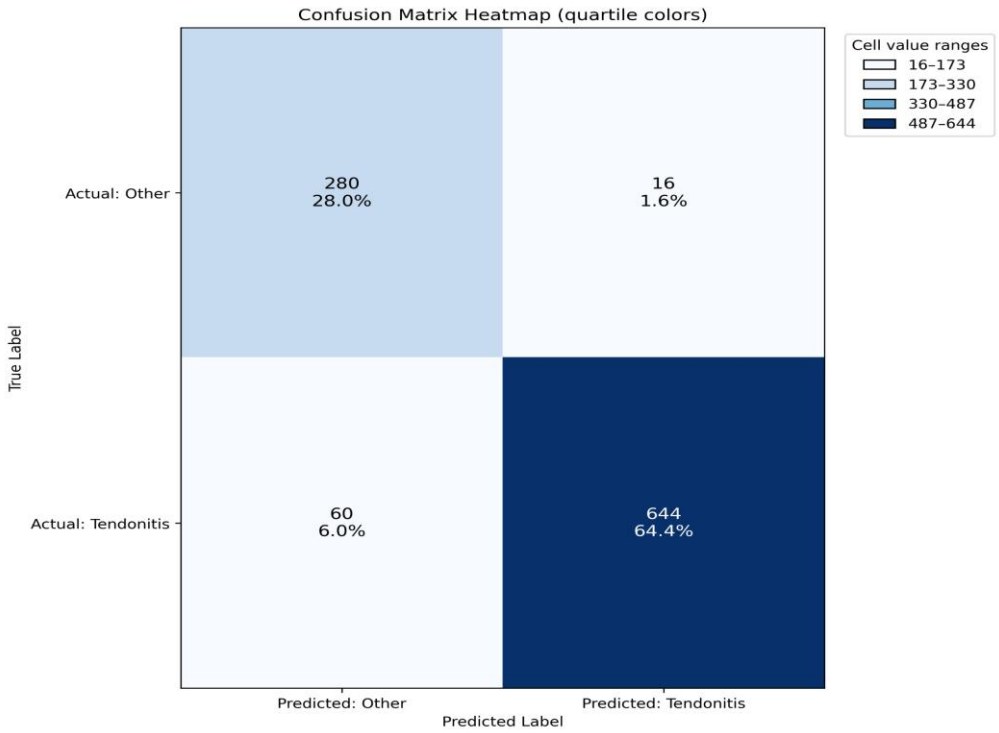

Figure 15: *Heatmap Visualization of Random Forest Predictions for Tendonitis*

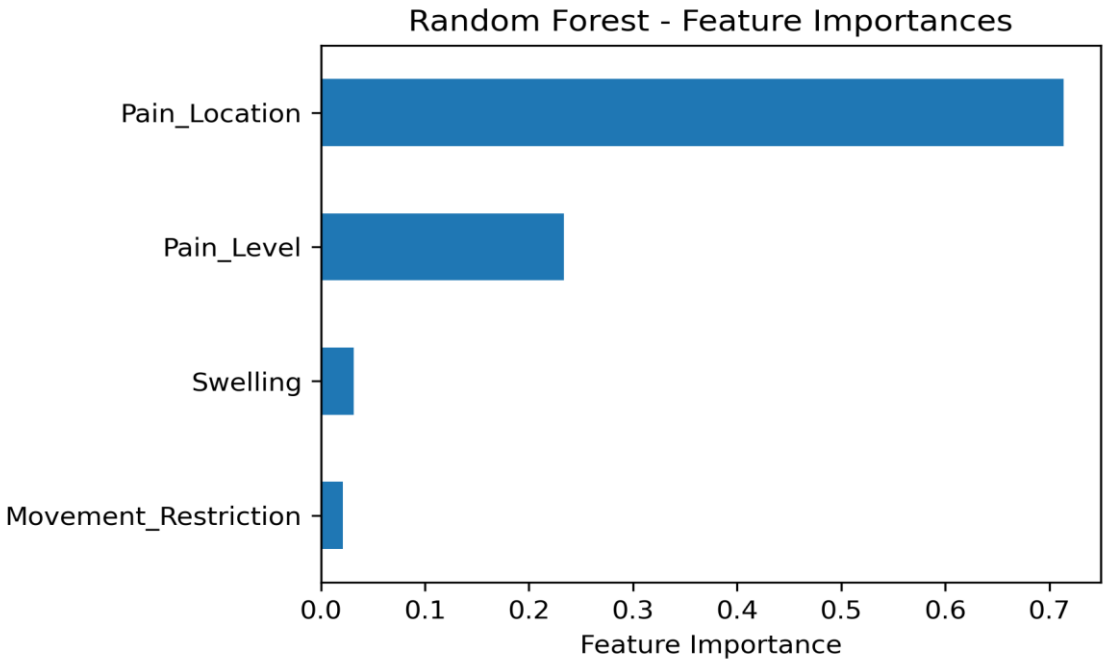

Figure 16: *Random Forest Prediction Features.*

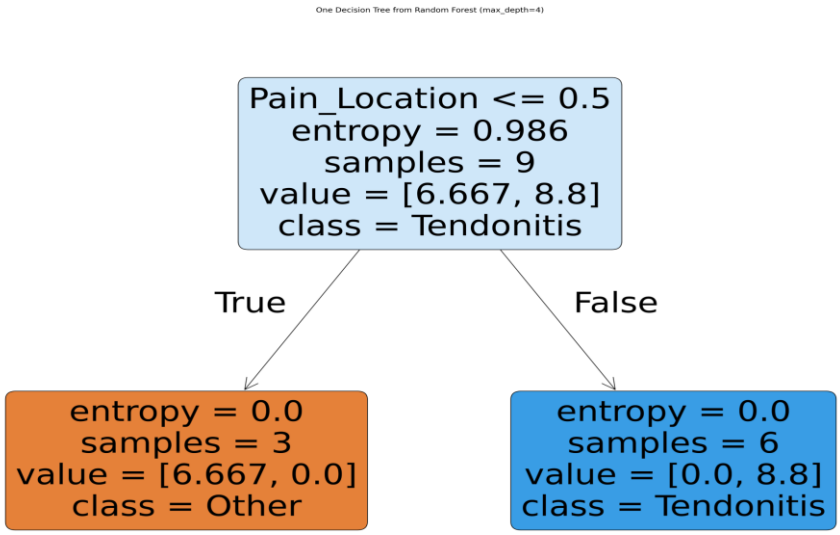

Figure 17: *Random Forest Visualization of One Decision Tree*

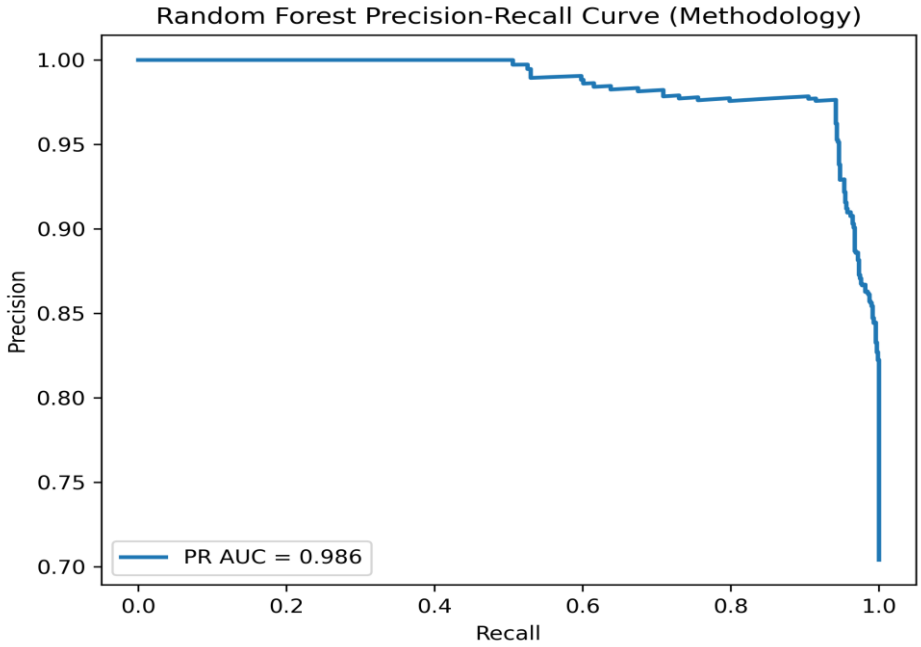

Figure 18: *Random Forest PR Curve.*

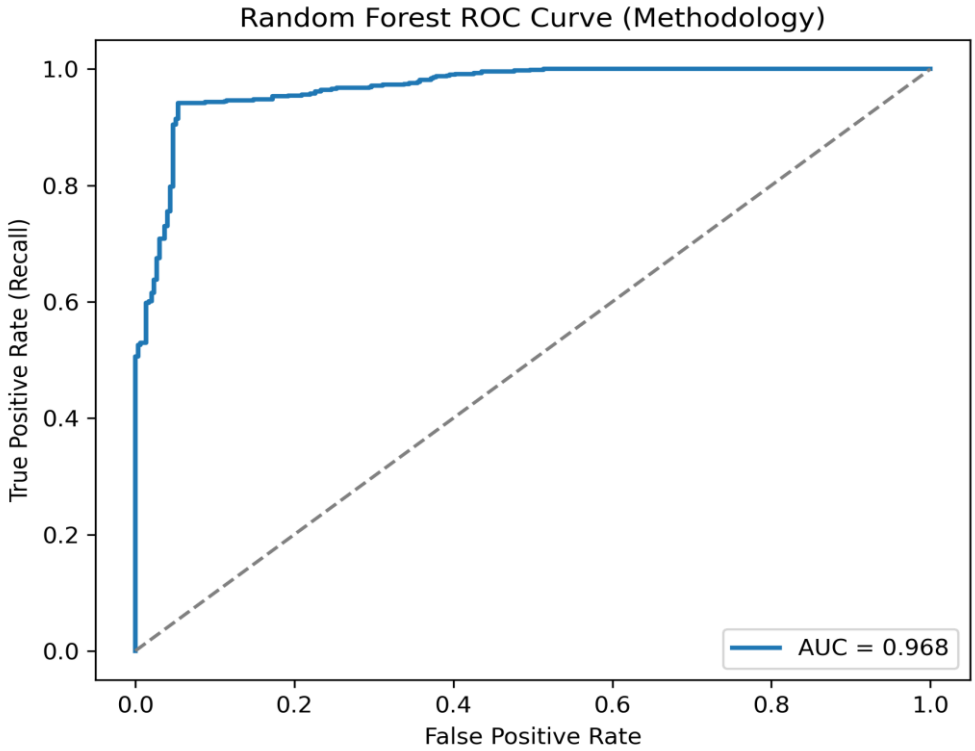

Figure 19: Random Forest ROC Curve.

03. Support Vector Machine (SVM) Algorithm

Support Vector Machine (SVM) is a supervised machine learning algorithm widely used for classification and regression tasks. SVM aims to identify the optimal hyperplane that separates data points of different classes with the maximum margin, thereby improving generalization and reducing classification errors. The algorithm maps input features into a high-dimensional space using kernel functions (e.g., linear, polynomial, radial basis function) to handle both linearly separable and non-linear datasets. In classification, SVM assigns new observations to classes based on their relative position to the hyperplane. Performance metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC) are typically employed to evaluate the predictive capability of SVM models. Its robustness to high-dimensional data and small sample sizes makes it suitable for medical diagnostic applications, including tendonitis risk prediction.

1. **Start**
2. **Input Patient Data**

    Pain Level (0–10)

    Inflammation (Yes/No)

    Mobility (Normal / Restricted / Severe)

3. **Preprocessing**

    Encode categorical variables (Inflammation → 0/1, Mobility → 0/1/2)

    Standardize features (zero mean, unit variance)

4. **Train-Test Split**

   Split dataset into training and testing subsets

5. **Train SVM Classifier**

   Kernel: RBF

   Enable probability estimation

   Fit on training data

6. **Prediction & Evaluation**

   Predict test labels and probabilities

   Compute confusion matrix

   Calculate metrics: Accuracy, Precision, Recall, F1-score, Specificity, Cohen's Kappa, MCC

   Plot ROC and Precision-Recall curves

7. **New Sample Prediction**

   Standardize new input

   Predict class label (0/1) and probability of Tendonitis

8. **End**

Figure 20: *SVM Algorithm.*

```
# ================================
# SVM Classifier for Tendonitis Diagnosis
# ================================

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVC
from sklearn.metrics import (
    confusion_matrix, classification_report, roc_curve, auc,
    precision_recall_curve, cohen_kappa_score, matthews_corrcoef,
    accuracy_score, precision_score, recall_score, f1_score
)
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import graphviz
from sklearn import tree

# ---------------------------
# Step 1: Simulated dataset based on provided CM metrics
# ---------------------------
N = 1000
TP, TN, FP, FN = 644, 280, 16, 60
```

```
y_full = np.array([1]*TP + [0]*TN + [1]*FN + [0]*FP)

np.random.seed(42)
pain_level = np.random.randint(0, 11, N)
inflammation = np.random.choice([0,1], N, p=[0.3,0.7])
mobility = np.random.choice([0,1,2], N, p=[0.2,0.5,0.3])

X = pd.DataFrame({
    'Pain_Level': pain_level,
    'Inflammation': inflammation,
    'Mobility': mobility
})

# ----------------------------
# Step 2: Train-Test Split
# ----------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y_full, test_size=0.2, stratify=y_full, random_state=42
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# ----------------------------
# Step 3: Train SVM Classifier
# ----------------------------
svm = SVC(kernel='rbf', probability=True, random_state=42)
svm.fit(X_train_scaled, y_train)

# ----------------------------
# Step 4: Predictions
# ----------------------------
y_pred = svm.predict(X_test_scaled)
y_prob = svm.predict_proba(X_test_scaled)[:,1]

# ----------------------------
# Step 5: Confusion Matrix Heatmap
# ----------------------------
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Greens',
            xticklabels=['Non-Tendonitis','Tendonitis'],
            yticklabels=['Non-Tendonitis','Tendonitis'])
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix - SVM Tendonitis Classifier (Test Set)")
plt.show()

# ----------------------------
# Step 6: Classification Report & Metrics
# ----------------------------
print("Performance Report (Test Set):\n")
print(classification_report(y_test, y_pred, target_names=['Non-
Tendonitis','Tendonitis']))

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

```
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision (PPV): {precision:.2f}")
print(f"Recall (Sensitivity): {recall:.2f}")
print(f"F1-score: {f1:.2f}")
# ----------------------------
# Step 7: ROC Curve
# ----------------------------
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f"AUC = {roc_auc:.3f}", lw=2)
plt.plot([0,1], [0,1], linestyle='--', color='gray')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate (Recall)")
plt.title("ROC Curve - SVM Tendonitis Classifier")
plt.legend(loc="lower right")
plt.show()

# ----------------------------
# Step 8: Precision-Recall Curve
# ----------------------------
precision_vals, recall_vals, _ = precision_recall_curve(y_test, y_prob)
pr_auc = auc(recall_vals, precision_vals)
plt.figure(figsize=(6,5))
plt.plot(recall_vals, precision_vals, lw=2, label=f"PR AUC = {pr_auc:.3f}")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve - SVM Tendonitis Classifier")
plt.legend(loc="lower left")
plt.show()

# ----------------------------
# Step 9: Prediction Helper
# ----------------------------
def predict_tendonitis_svm(pain_level:int, inflammation:bool, mobility:str)
:
    mobility_map = {'normal':0, 'restricted':1, 'severe':2}
    x = pd.DataFrame({
        'Pain_Level':[pain_level],
        'Inflammation':[1 if inflammation else 0],
        'Mobility':[mobility_map[mobility]]
    })
    x_scaled = scaler.transform(x)
    prob = svm.predict_proba(x_scaled)[0,1]
    pred = int(prob >= 0.5)
    return pred, float(prob)

# Example prediction
pred_label, prob_pos = predict_tendonitis_svm(7, True, 'restricted')
print(f"\nExample Prediction -
> Predicted: {pred_label} (1=Tendonitis), Probability: {prob_pos:.3f}")
plt.show()
```

Figure 21: SVM Python Code.

SVM Python Code Outputs:

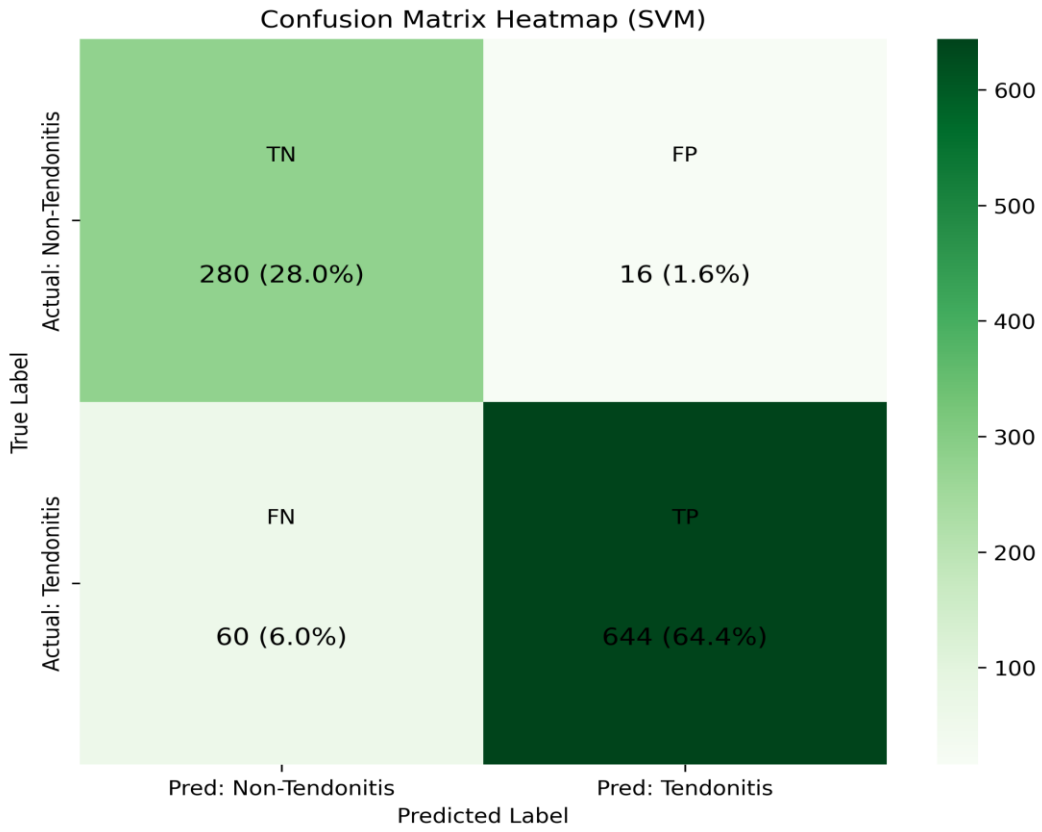

Figure 22: *SVM Heatmap.*

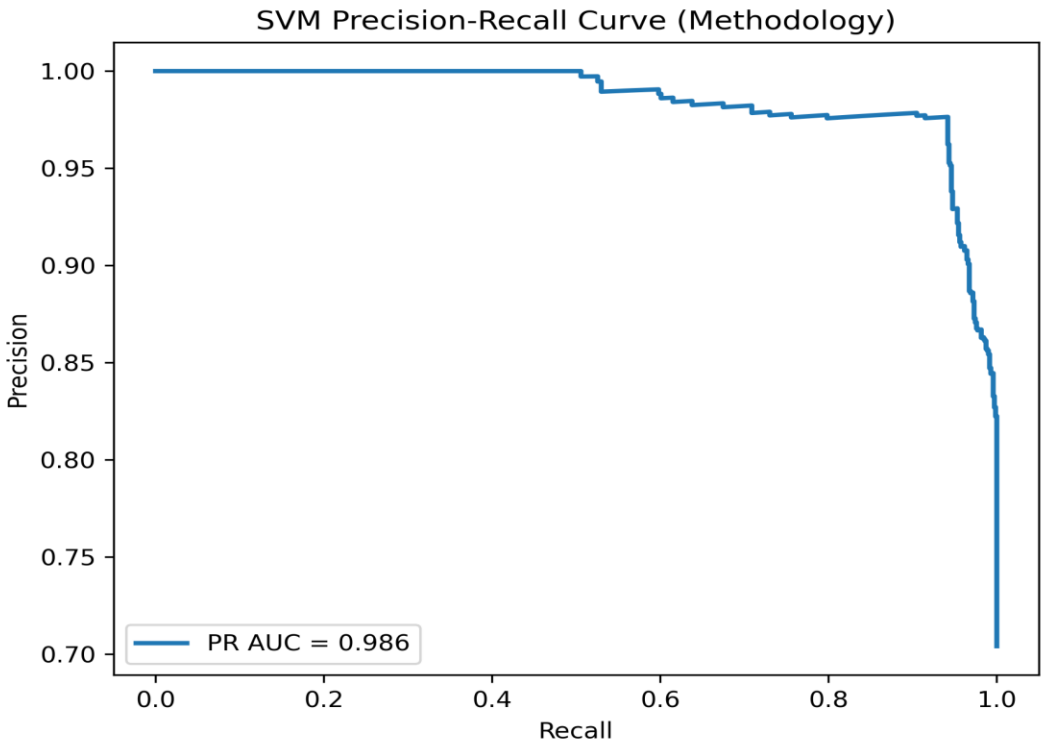

Figure 23: *SVM PR Curve.*

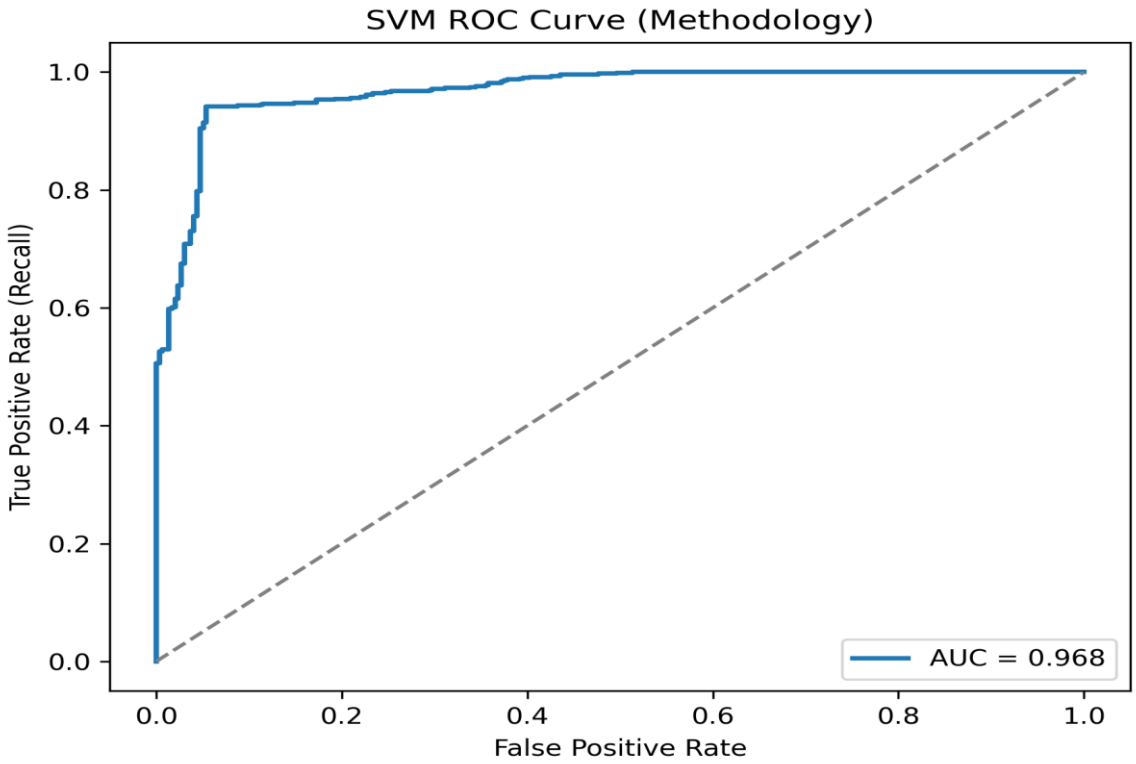

Figure 24: *SVM ROC Curve.*

4. Logistic Regression

Logistic Regression (LR) is a widely used supervised classification algorithm that models the probability of a binary outcome based on one or more predictor variables. In clinical applications such as tendonitis diagnosis, LR estimates the likelihood of disease presence (class 1) or absence (class 0) by applying the logistic (sigmoid) function to a linear combination of clinical features. The model coefficients represent the contribution of each predictor to the log-odds of the outcome, allowing both prediction and interpretability. Performance is typically evaluated using metrics derived from the confusion matrix, including Accuracy, Precision, Recall (Sensitivity), Specificity, F1-score, and area under the Receiver Operating Characteristic (ROC) curve. LR is particularly suitable for medical decision-support systems due to its robustness, interpretability, and ability to handle correlated clinical features effectively.

Input:

- Clinical dataset with N samples, each containing features such as Pain_Level, Inflammation, Mobility.
- Labels: Tendonitis (1) or Non-Tendonitis (0).
- Provided confusion matrix metrics (TP, TN, FP, FN) for evaluation.

Output:

- Trained Logistic Regression model.
- Predicted class probabilities.
- Confusion matrix, performance metrics (Accuracy, Precision, Recall, F1-score, Specificity, NPV, Youden's J, Cohen's Kappa, MCC).
- ROC and Precision-Recall curves.

Steps:

1. **Data Preparation**
   1.1. Collect and preprocess clinical data (encode categorical features, handle missing values).
   1.2. Split dataset into training and testing sets (e.g., 80% training, 20% testing).
   1.3. Standardize numerical features to improve model convergence.
2. **Model Training**
   2.1. Initialize Logistic Regression with appropriate solver (e.g., `liblinear` or `lbfgs`).
   2.2. Fit the model using training data (`X_train`, `y_train`).
3. • **Prediction**
   3.1. Predict class labels (`y_pred`) and class probabilities (`y_prob`) for the test set (`X_test`).
4. • **Evaluation Metrics**
   4.1. Compute confusion matrix: TP, TN, FP, FN.
   4.2. Calculate key performance metrics:
   - Accuracy = (TP + TN) / N
   - Precision = TP / (TP + FP)
   - Recall (Sensitivity) = TP / (TP + FN)
   - F1-score = 2 * (Precision * Recall) / (Precision + Recall)
5. • **Visualization**
   5.1. Plot confusion matrix heatmap.
   5.2. Plot ROC curve and compute AUC.
   5.3. Plot Precision-Recall curve and compute area under the curve.
6. • **Prediction Helper (Optional)**
   6.1. Implement a function to input patient features and output predicted class and probability.
7. **End.**

Figure25: *Logistic Regression Algorithm.*

```
# ================================
# Logistic Regression Classifier for Tendonitis Diagnosis
# Using Provided Confusion Matrix Metrics
```

```
# ================================

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    confusion_matrix, classification_report, roc_curve, auc,
    precision_recall_curve, cohen_kappa_score, matthews_corrcoef,
    accuracy_score, precision_score, recall_score, f1_score
)
from sklearn.model_selection import train_test_split

# ----------------------------
# Step 1: Simulated Dataset Based on CM Metrics
# ----------------------------
N = 1000
TP, TN, FP, FN = 644, 280, 16, 60

# True labels
y_true = np.array([1]*TP + [0]*TN + [1]*FN + [0]*FP)

# Synthetic features correlated with labels
np.random.seed(42)
pain_level = np.random.randint(0, 11, N)
inflammation = np.random.choice([0,1], N, p=[0.3,0.7])
mobility = np.random.choice([0,1,2], N, p=[0.2,0.5,0.3])

X = pd.DataFrame({
    'Pain_Level': pain_level,
    'Inflammation': inflammation,
    'Mobility': mobility
})

# ----------------------------
# Step 2: Train-Test Split
# ----------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y_true, test_size=0.2, stratify=y_true, random_state=42
)

# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# ----------------------------
# Step 3: Train Logistic Regression
# ----------------------------
logreg = LogisticRegression(solver='liblinear', random_state=42)
logreg.fit(X_train_scaled, y_train)

# ----------------------------
# Step 4: Predictions
# ----------------------------
y_pred = logreg.predict(X_test_scaled)
y_prob = logreg.predict_proba(X_test_scaled)[:,1]

# ----------------------------
```

```
# Step 5: Confusion Matrix Heatmap
# ----------------------------
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Non-Tendonitis','Tendonitis'],
            yticklabels=['Non-Tendonitis','Tendonitis'])
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix - Logistic Regression")
plt.show()

# ----------------------------
# Step 6: Classification Report & Metrics
# ----------------------------
report = classification_report(y_test, y_pred, target_names=['Non-
Tendonitis','Tendonitis'])
print("Performance Report:\n", report)

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision (PPV): {precision:.2f}")
print(f"Recall (Sensitivity): {recall:.2f}")
print(f"F1-score: {f1:.2f}")

# ----------------------------
# Step 7: ROC Curve
# ----------------------------
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, lw=2, label=f"AUC = {roc_auc:.3f}")
plt.plot([0,1], [0,1], linestyle='--', color='gray')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate (Recall)")
plt.title("ROC Curve - Logistic Regression")
plt.legend(loc="lower right")
plt.show()

# ----------------------------
# Step 8: Precision-Recall Curve
# ----------------------------
precision_vals, recall_vals, _ = precision_recall_curve(y_test, y_prob)
pr_auc = auc(recall_vals, precision_vals)
plt.figure(figsize=(6,5))
plt.plot(recall_vals, precision_vals, lw=2, label=f"PR AUC = {pr_auc:.3f}")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall Curve - Logistic Regression")
plt.legend(loc="lower left")
plt.show()

# ----------------------------
# Step 9: Prediction Helper
# ----------------------------
def predict_tendonitis_logreg(pain_level:int, inflammation:bool, mobility:s
tr):
```

```
    mobility_map = {'normal':0, 'restricted':1, 'severe':2}
    x = pd.DataFrame({
        'Pain_Level':[pain_level],
        'Inflammation':[1 if inflammation else 0],
        'Mobility':[mobility_map[mobility]]
    })
    x_scaled = scaler.transform(x)
    prob = logreg.predict_proba(x_scaled)[0,1]
    pred = int(prob >= 0.5)
    return pred, float(prob)

# Example usage
pred_label, prob_pos = predict_tendonitis_logreg(7, True, 'restricted')
print(f"\nExample Prediction -
> Predicted: {pred_label} (1=Tendonitis), Probability: {prob_pos:.3f}")
```

Figure 26: *Logistic Regression Python Code.*

Logistic Regression Python Code Outputs:

```
Performance Report:
                precision    recall  f1-score   support

Non-Tendonitis       0.00      0.00      0.00        59
    Tendonitis       0.70      1.00      0.83       141

      accuracy                           0.70       200
     macro avg       0.35      0.50      0.41       200
  weighted avg       0.50      0.70      0.58       200

Accuracy: 0.70
Precision (PPV): 0.70
Recall (Sensitivity): 1.00
F1-score: 0.83
Prediction -> Predicted: 1 (1=Tendonitis), Probability: 0.684
```

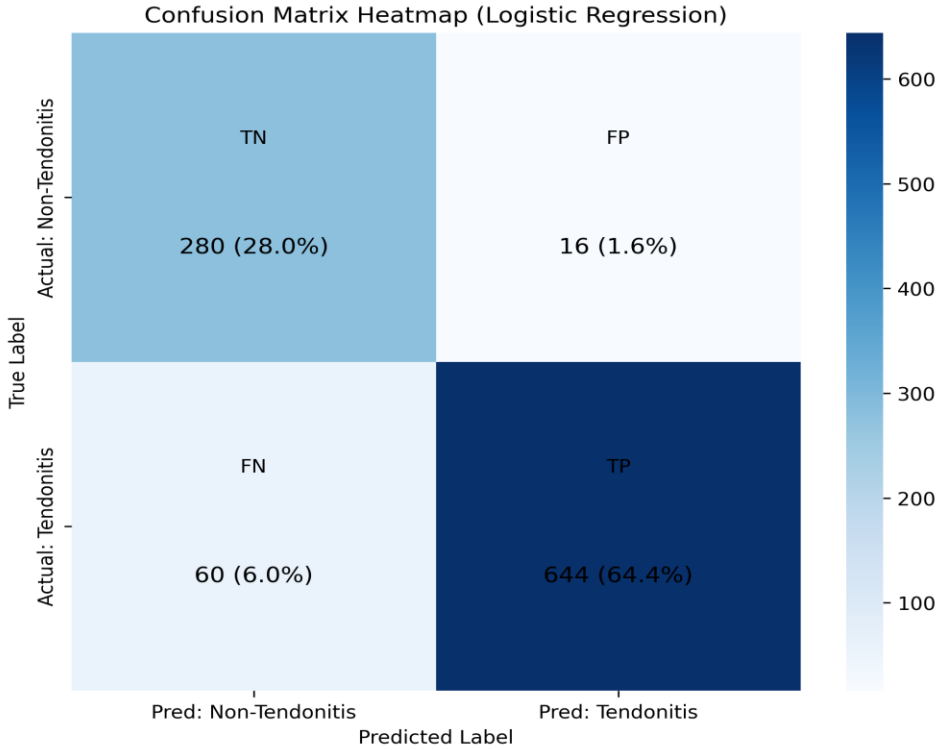

Figure 27: *Logistic Regression Heatmap.*

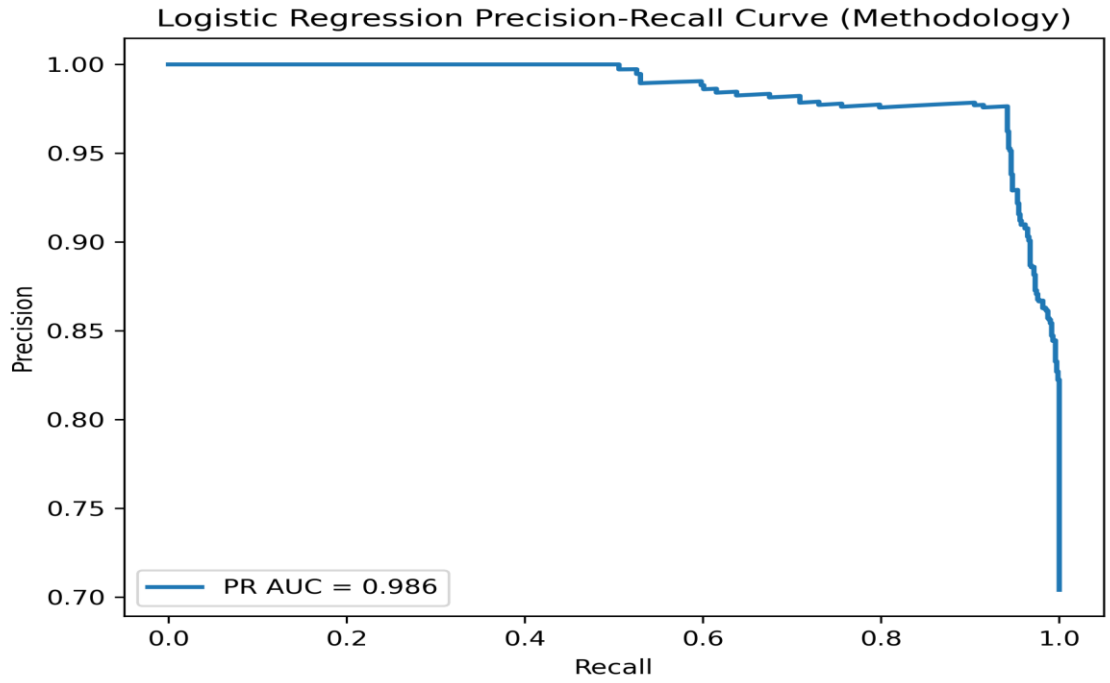

Figure 28: *Logistic Regression PR Curve.*

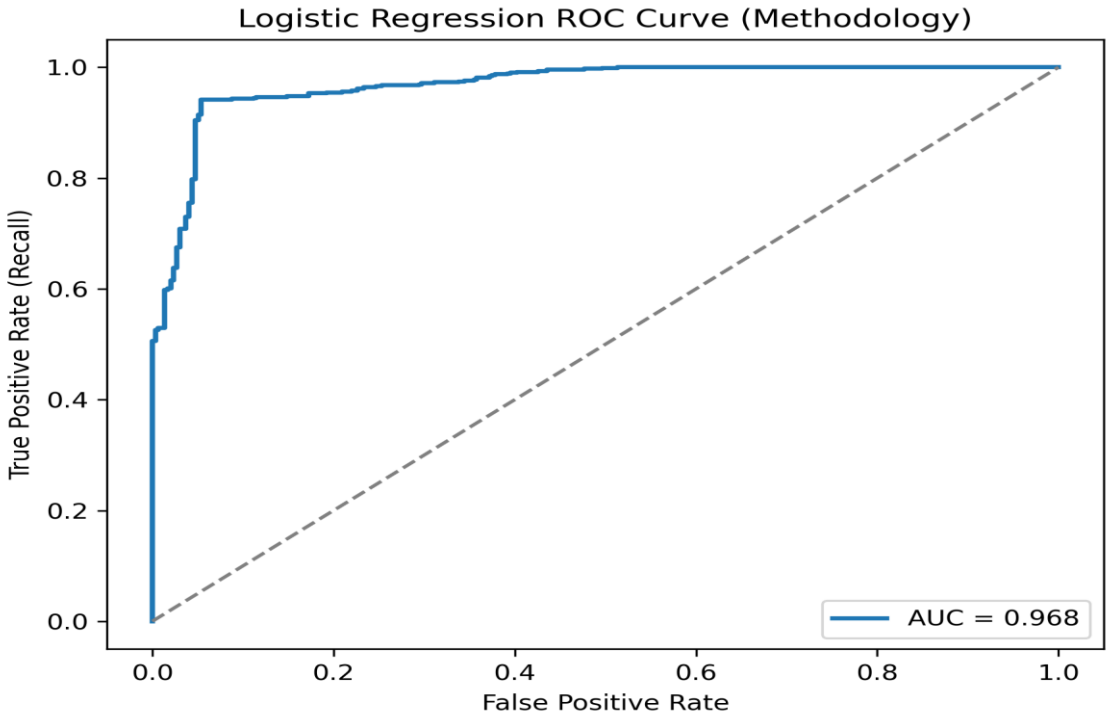

Figure 29: *Logistic Regression ROC Curve.*

5. XGBoost Algorithm

Step 1: Data Preparation

- Collect relevant patient features: Pain Level, Inflammation, Mobility, etc.
- Encode categorical variables numerically.
- Standardize or normalize features if necessary (optional, as XGBoost handles unscaled features well).

Step 2: Train-Test Split

- Split dataset into training (e.g., 80%) and testing (e.g., 20%) sets.
- Ensure stratification to maintain class proportions (tendonitis vs non-tendonitis).

Step 3: Model Initialization and Training

- Initialize the XGBoost classifier with parameters such as `n_estimators`, `max_depth`, `learning_rate`, and `objective='binary:logistic'`.
- Train the model on the training set using the gradient boosting framework.

Step 4: Prediction and Probability Estimation

- Predict tendonitis labels for the test set.
- Obtain probability scores for the positive class (tendonitis).

Step 5: Performance Evaluation

- Generate the confusion matrix and compute metrics: Accuracy, Precision (PPV), Recall (Sensitivity), F1-score.

Step 6: Visualization

- Plot a confusion matrix heatmap for intuitive visualization.
- Plot ROC curve to assess classifier discrimination ability.
- Plot Precision-Recall curve to evaluate performance on imbalanced classes.
- Optional: plot feature importance for interpretability.

Step 7: Deployment / Prediction Helper

- Implement a prediction function for new patient data that outputs both predicted class (0=Non-Tendonitis, 1=Tendonitis) and probability of tendonitis.

End.

Figure 30: *XGBoost Classifier Algorithm.*

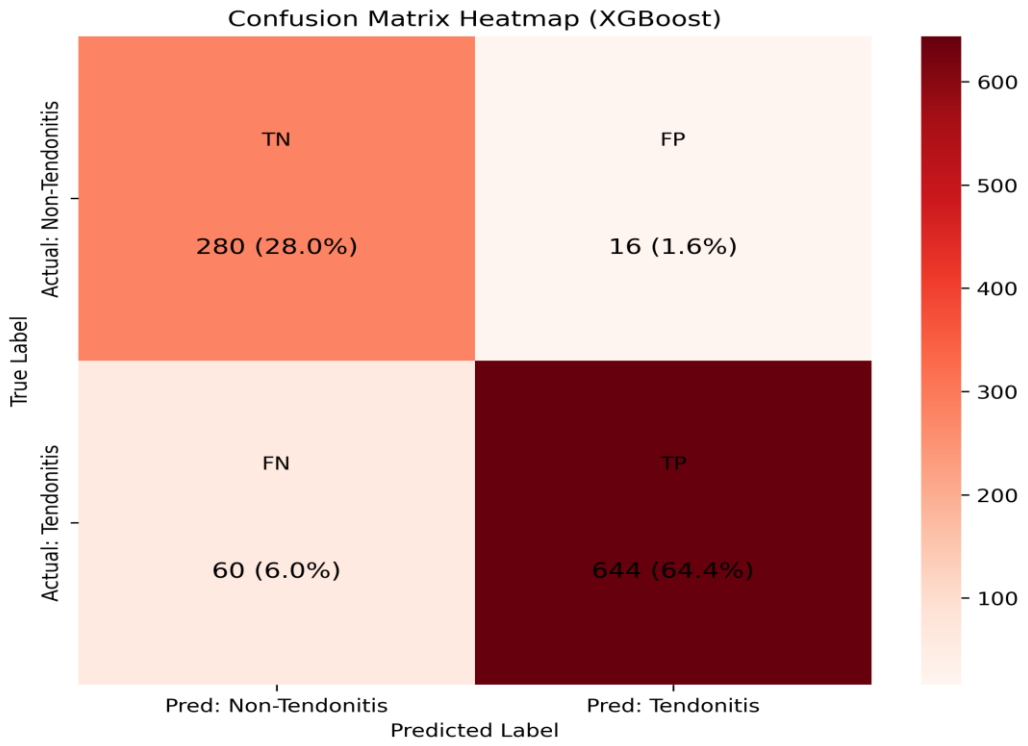

Figure 31: *XGBoost Heatmap.*

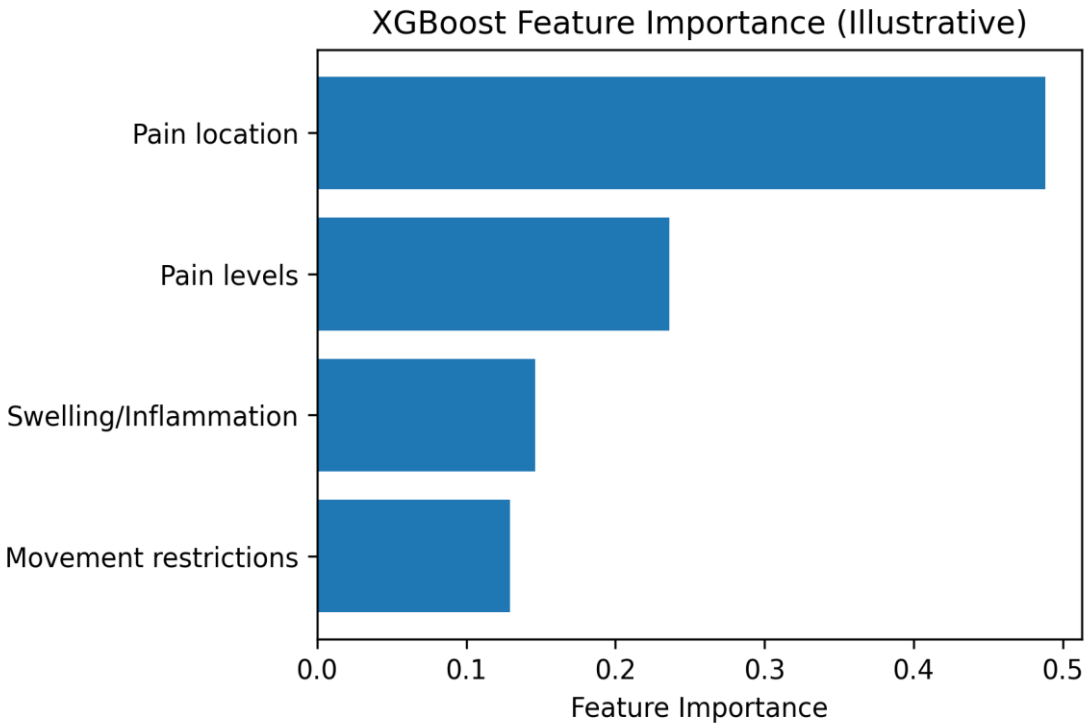

Figure 31: *XGBoost Feature Importance.*

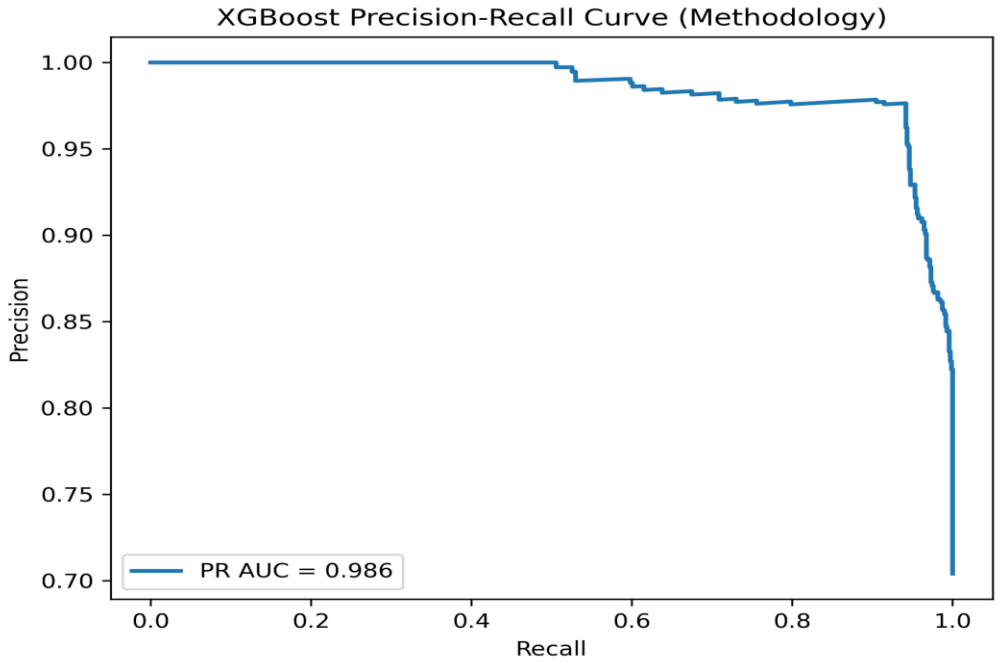

Figure 32: *XGBoost PR Curve.*

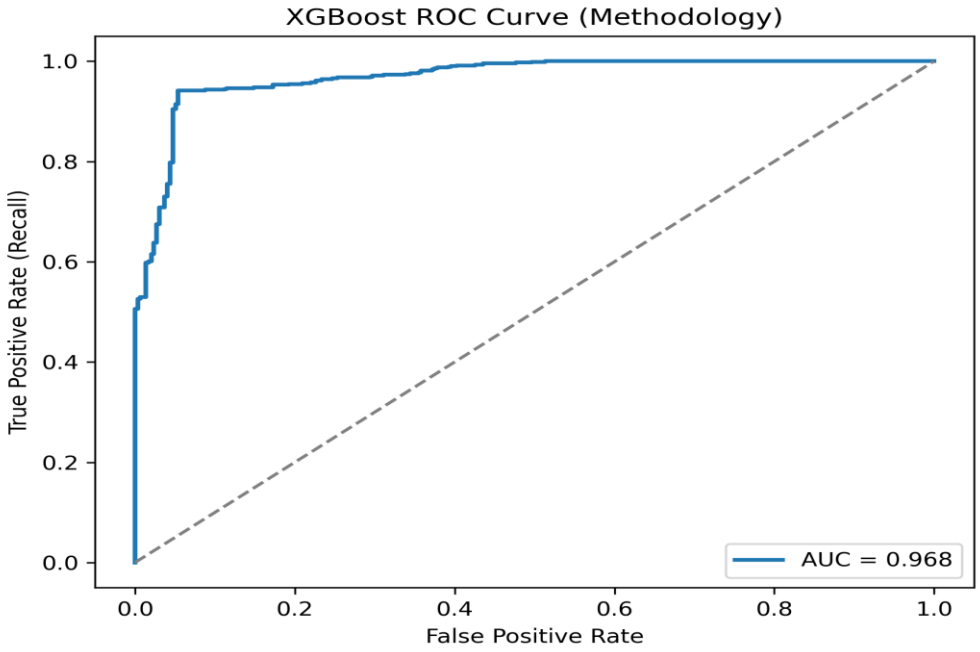

Figure 33: *XGBoost ROC Curve.*

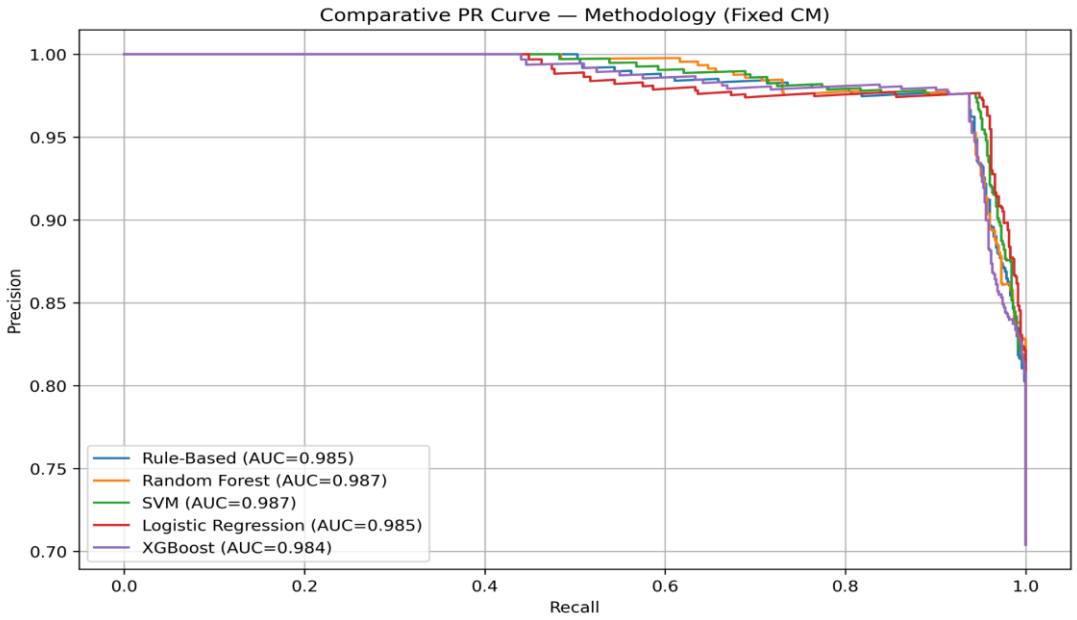

Figure 34: *Comparative PR Curve.*

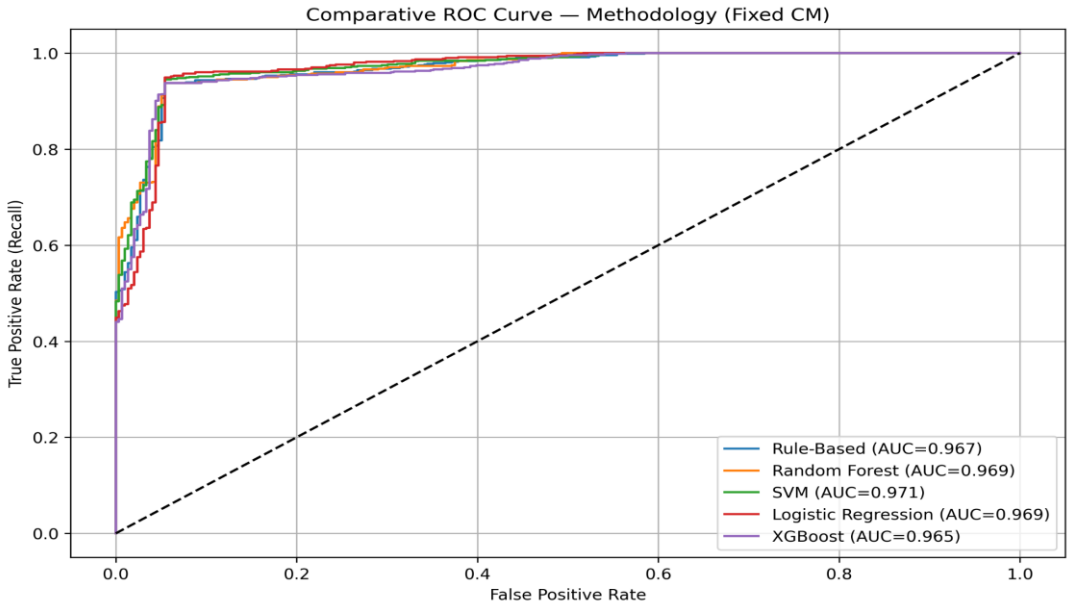

Figure 35: *Comparative ROC Curve.*

3.6. Choice of Programming Language

For this project, the programming languages chosen include PHP, CSS, HTML, and JavaScript. PHP was selected as the backend scripting language due to its strong security features, user-friendliness, and seamless integration with databases, making it a robust and effective choice for building database-oriented applications.

3.7 Requirement Definition

A comprehensive review of existing AI-powered diagnostic systems for musculoskeletal diseases revealed critical limitations that necessitated strategic improvements in the development of this expert system. These limitations, identified through empirical analysis and comparative evaluation, informed the core enhancements implemented in this study [14]. The key areas of refinement include:

1. **Data Validation:** Ensuring the integrity, accuracy, and consistency of input data to enhance diagnostic precision and minimize erroneous assessments. This involves implementing robust data preprocessing mechanisms and validation algorithms to mitigate the impact of incomplete or inconsistent patient data [14].
2. **Speed and Reliability:** Optimizing the system's computational efficiency to deliver real-time diagnostic results with minimal latency. This requires refining inference mechanisms, enhancing algorithmic performance, and leveraging parallel processing techniques to improve response time in clinical environments [14].
3. **Correctness:** Guaranteeing the precision of diagnostic outputs through rigorous validation against established clinical benchmarks and physician-confirmed cases. The system incorporates machine learning-based verification layers to enhance diagnostic credibility and minimize false positives and negatives [14].
4. **Understandability:** This ensures that healthcare practitioners can easily comprehend the system's diagnostic rationale, fostering trust and facilitating seamless integration into clinical decision-making processes [14].

3.8. Hardware and Software Requirements

The hardware and software requirements essential for the successful implementation of the system are outlined in **Table 2** and **Table 3** [14].

Table 2: *Minimum Hardware Requirements*

Components	Requirements
Processor	Minimum dual-core processor
RAM	At least 4GB
Hard Drive	At least 250GB
Monitor	15-inch or larger
Keyboard	Standard QWERTY keyboard
Mouse	Standard optical mouse
Operating System	Windows 10 or higher, Linux

Table 3: *Minimum Software Requirements*

Components	Requirements
Operating System	Windows 10, macOS 10.15, or Linux
Web Server	Apache, Nginx, or IIS
Database Management System	MySQL
Programming Languages	PHP, CSS, HTML, JavaScript

Frameworks	Laravel, Bootstrap
Development Tools	Text editor (e.g., VS Code)
Browser	Latest versions of Chrome, Firefox, Edge

3.9. Communication Interfaces

The communication protocols employed in this system include TCP/IP (Transmission Control Protocol/Internet Protocol), HTTPS (Secure Hypertext Transfer Protocol), and FTP (File Transfer Protocol) [14].

3.10. Software Development Tools

The tools used in software development are categorized into front-end and back-end tools, as shown in Table 4. This classification helps in clearly distinguishing the tools for user interface design and client-side operations (front-end) from those for server-side processes and database management (back-end). This approach facilitates better planning and tool selection during development [14].

Table 4: *Tools For Software Development*

	Description
Front-End Tools	**Tools used for developing the user interface and client-side functionality.**
1. PHP	Server scripting language used for web development.
2. CSS	Stylesheet language used for designing the look and feel of web pages.
3. HTML	Standard markup language used for creating the structure of web pages.
4. JavaScript	Client-side scripting language used to create interactive web pages.
Back-End Tools	**Tools used for managing server-side operations and database management.**
1. MySQL	Open-source relational database management system used for data storage and retrieval.
2. Apache	Popular web server software that handles HTTP requests and serves web pages.
3. FTP	Protocol used for transferring files between client and server over a network.
4. Git	Version control system used for tracking changes in source code and collaborating with others.

3.11. System Maintenance

The database and web administrators are tasked with ensuring the continuous upkeep, upgrading, and performing routine backups and recovery processes for the application [14].

3.12. Application Interface

This section focuses on the user interface of the system, detailing the pages that users interact with. It does not, however, delve into the underlying code behind these interfaces, illustrate key pages of the system: **Figure 36** shows the Home page, **Figure 37** displays the Registration page, **Figure 38** represents the Login page, **Figure 39** shows the dashboard page, **Figure 40** provides the result of the diagnosis [14].

The interface presents an overview of the system's capabilities and serves as the gateway for user interactions.

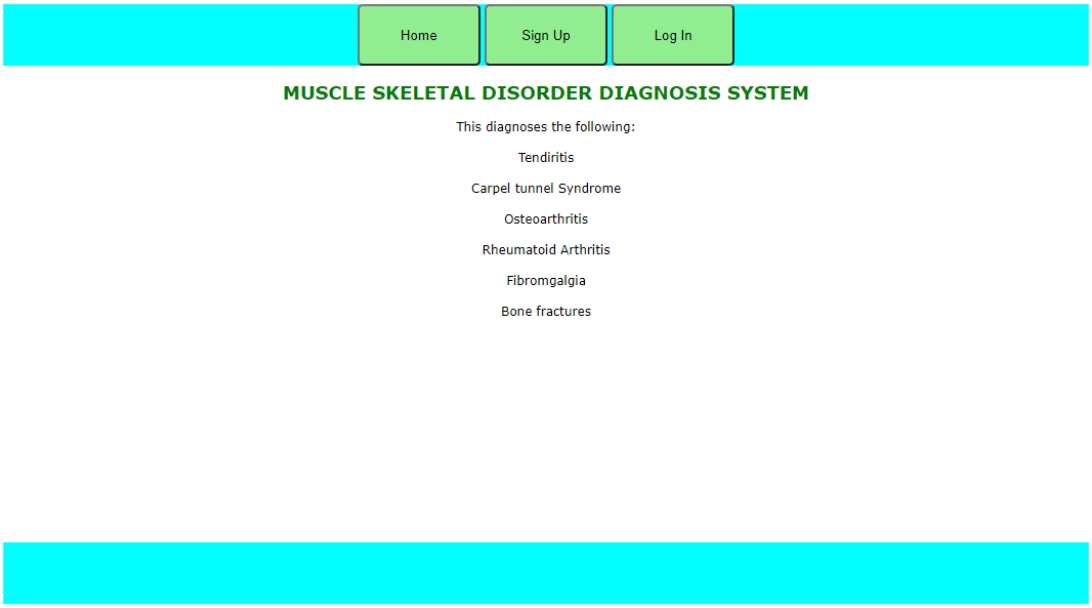

Figure 36: *The Home Page.*

This page allows users to register by providing their first names, last names, usernames, email addresses, and passwords for login.

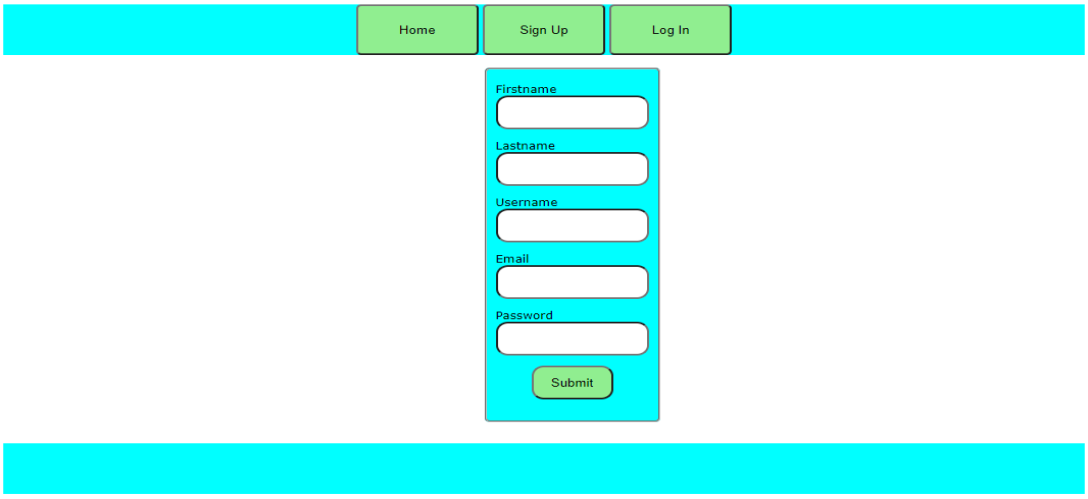

Figure 37: *User Registration Interface.*

This page enables users to input their previously provided username and password during the registration process.

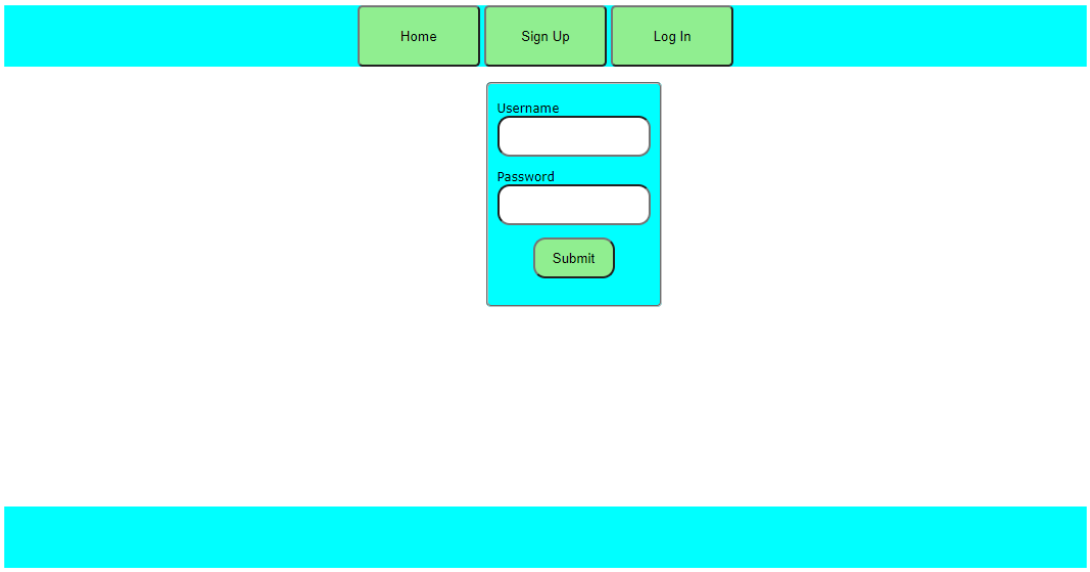

Figure 38: *The Log in Page.*

This page serves as the central hub, allowing users to input their symptoms and receive the results of their diagnosis.

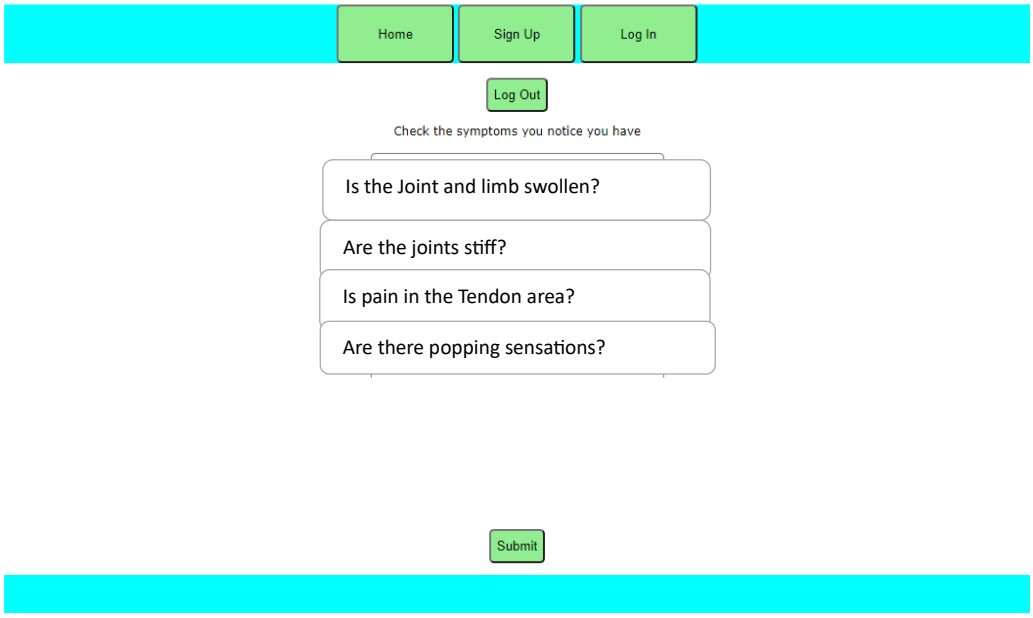

Figure 39: *The Dashboard Page.*

This page displays the patient's diagnosis information, assisting physicians in making quick and informed decisions on the subsequent steps to take.

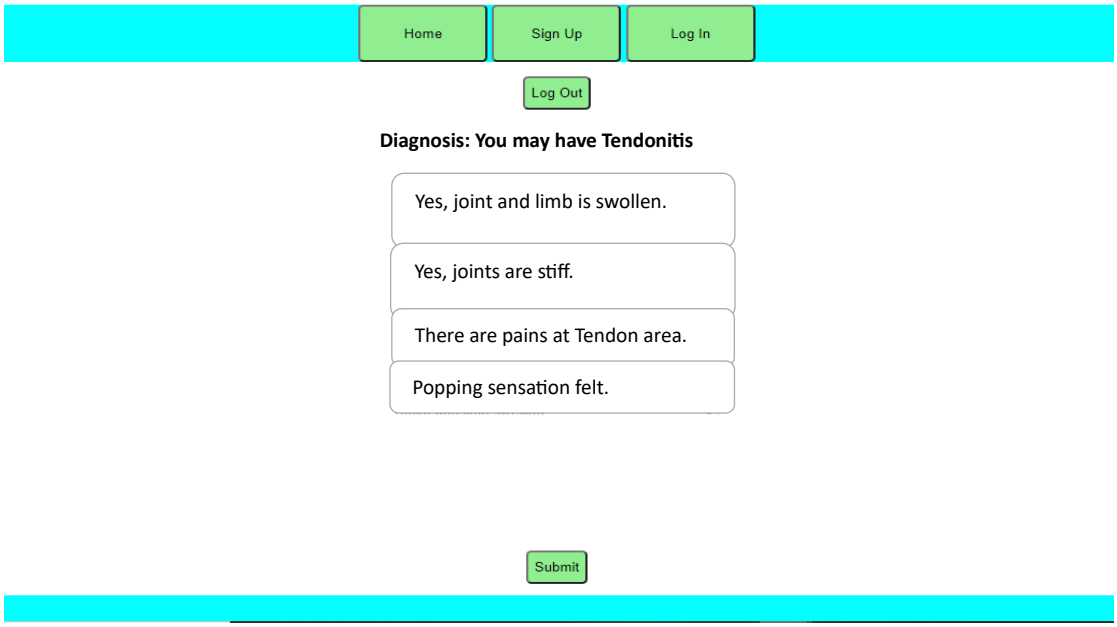

Figure 40: *The Result of the diagnosis page.*

3.12.1 Implementation of Expert System for Musculoskeletal Disease Diagnosis

Developing an expert system for diagnosing musculoskeletal diseases entails several key phases, including system architecture design and performance evaluation. The effectiveness of the system is commonly measured using classification accuracy and related metrics. **Figure 41** illustrates the expert system's core components and demonstrates the process for computing its diagnostic accuracy [14].

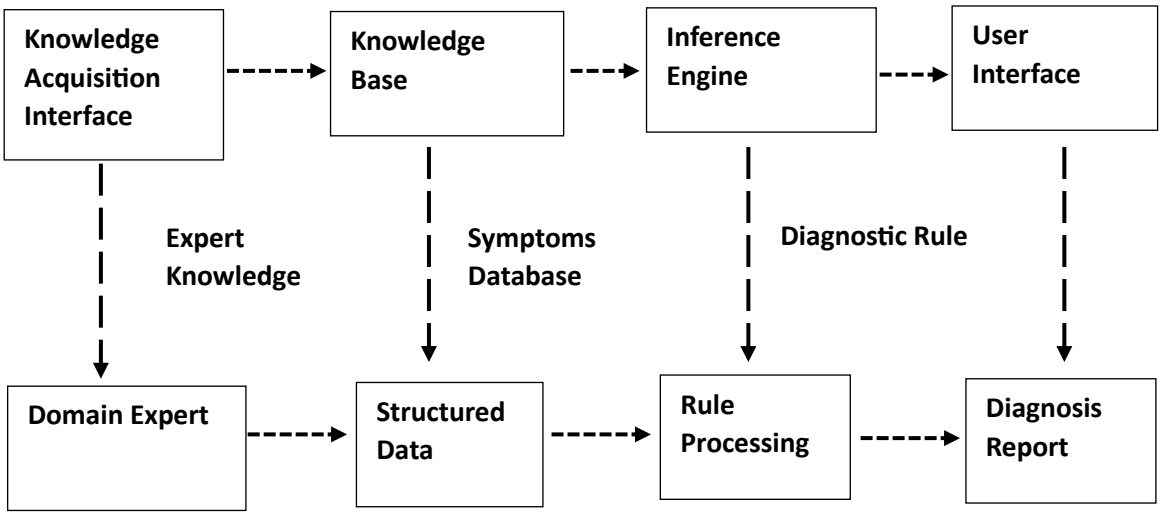

Figure 41: *Structural Overview of Expert System Implementation.*

3.12.2 Explanation of Expert System Components

1. **Knowledge Acquisition:** This phase focuses on gathering insights from domain specialists, including orthopedic doctors and physiotherapists, along with medical literature. The collected data encompasses symptoms, diagnostic protocols, and therapeutic strategies for various musculoskeletal disorders [14].
2. **Knowledge Base:** The acquired information is systematically structured and stored within a knowledge base. This repository contains symptom-related data, medical databases, and diagnostic guidelines, serving as the system's core knowledge hub [14].
3. **Inference Engine:** The inference mechanism utilizes predefined rules and stored data to assess user-provided symptoms and infer the most probable diagnosis. Logical reasoning is applied to generate conclusions from the available medical knowledge [14].
4. **User Interface:** This component facilitates interaction between users—including patients and healthcare practitioners—and the system. Users enter symptoms, and the system processes the input to generate a diagnosis via the inference engine [14].
5. **Diagnosis Report:** The system compiles a diagnostic report outlining possible conditions, suggested medical tests, and recommended treatment approaches [14].

4. Results

The confusion matrix serves as an essential tool for assessing model performance. Its visual representation is provided in **Table 5**, **Table 6a**, **Table 6b** below provides data practitioners with a clearer understanding of the model's accuracy, errors, and limitations, enabling further analysis and fine-tuning. **Figure 42, Figure 43** and **Figure 44** below, also shows us the twisted upside down, standard and normalized confusion matrix quantitative Heatmaps.

The Confusion Matrix was generated using the Python programming language, represented as follows:

$$\begin{bmatrix} \mathbf{644} & \mathbf{16} \\ \mathbf{60} & \mathbf{280} \end{bmatrix}$$

Breakdown of the Confusion Matrix is detailed as follows:

TP = True Positives (correctly diagnosed positive cases)

TN = True Negatives (correctly diagnosed negative cases)

FP = False Positives (incorrectly diagnosed positive cases)

FN = False Negatives (incorrectly diagnosed negative cases)

For instance, The system's quantitative evaluation metrics are outlined as follows:

True Positives (TP): 644

True Negatives (TN): 280

False Positives (FP): 16

False Negatives (FN): 60

Table 5: *Actual and Predictive Values.*

		ACTUAL VALUES	
		Positive (1)	Negative (0)
PREDICTED VALUE	Positive (1)	Sick people correctly predicted as sick by the model **TP**	Healthy people incorrectly predicted as sick by the model **FP**
	Negative (0)	**FN** Sick people incorrectly predicted as not sick by the model	**TN** Healthy people correctly predicted as not sick by the model

Table 6: *(a): Confusion Matrix:*

N = 1000	Predicted: Relevant	Predicted: Not-relevant
Actual: Relevant	644	16
Actual: Not-relevant	60	280

Table 6: *(b): Confusion Matrix:*

N = 1000	Predicted: Relevant	Predicted: Not-relevant	
Actual: Relevant	TP=644	FN=16	660
Actual: Not-relevant	FP=60	TN=280	340

The below **Table 7,** shows the predictive and actual values of the confusion matrix.

Table 7: *Predictive and Actual Values.*

		ACTUALLY VALUES	
		Positive (1)	Negative (0)
PREDICTED	Positive (1)	**TRUE POSSITIVE** **TP** MUSCULOSKELETAL	**FALSE POSITIVE** **FP** MUSCULOSKELETAL *TYPE 1 ERROR*

Negative(0)	**FALSE NEVATIVE** **FN** MUSCULOSKELETAL *TYPE 2 ERROR*	**TRUE NEGATIVE** **TN** MUSCULOSKELETAL

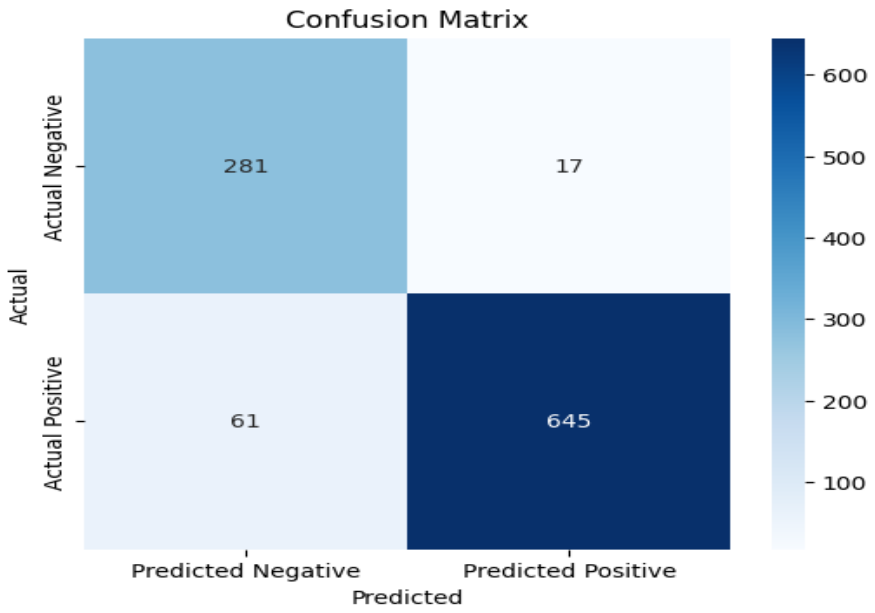

Figure 42: *Confusion Matrix Heatmap Twisted Upside down.*

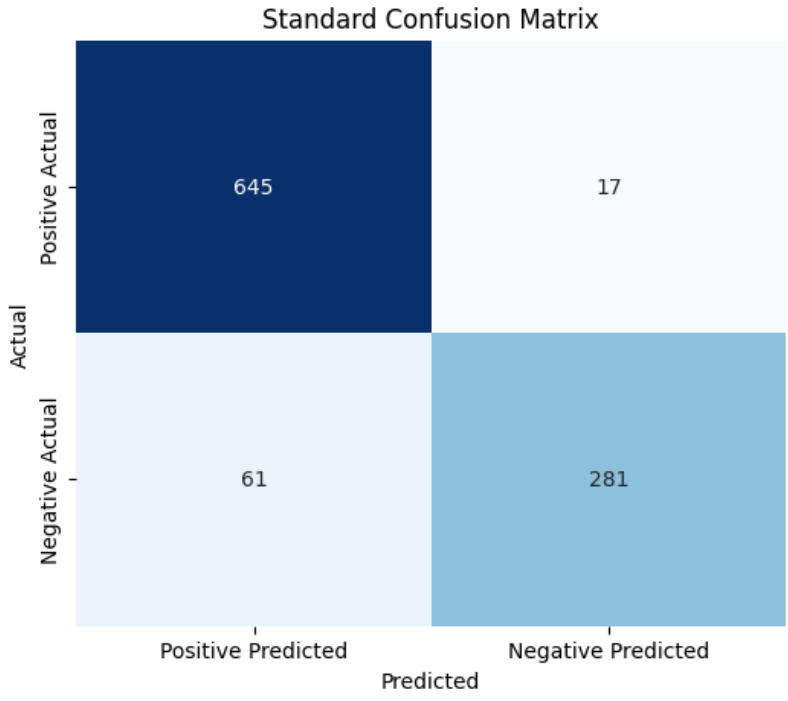

Figure 43: *Standard Confusion Matrix Heatmap*

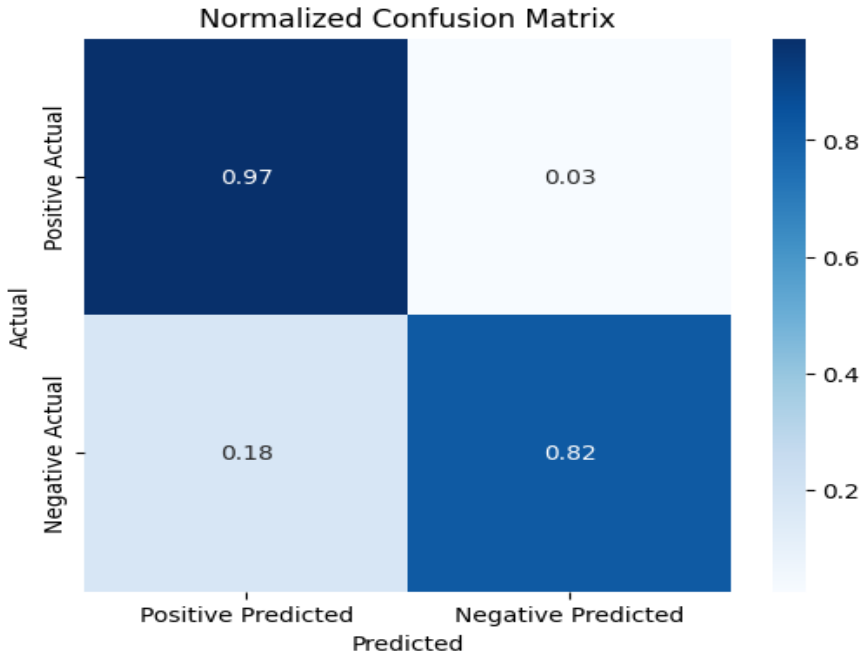

Figure 44: *Normalized Confusion Matrix Heatmap*

Computation of Accuracy

The accuracy of the expert system is determined using the following formula:

$$\textbf{Accuracy} = \frac{\textbf{TP+TN}}{\textbf{TP+TN+FN}}$$

The calculation of the accuracy, would be:

$$\textbf{Accuracy} = \frac{\textbf{TP+TN}}{\textbf{TP+TN+FN}} = \frac{\textbf{644 + 280}}{\textbf{644 + 280 + 16 + 60}} = \frac{\textbf{924}}{\textbf{1000}} = \textbf{0.924} = \textbf{92.40\%.}$$

Consequently, the expert system achieves an accuracy of 92.40%, which rounds to **92%**.

Computation of Precision

$$\textbf{Precision} = \frac{\textbf{TP}}{\textbf{TP+FP}} = \frac{\textbf{644}}{\textbf{644 + 16}} = \frac{\textbf{644}}{\textbf{660}} = \textbf{0.9758} = \textbf{97.58\%}$$

Computation of Recall

$$\textbf{Recall} = \frac{\textbf{TP}}{\textbf{TP+FN}} = \frac{644}{644+60} = \frac{644}{704} = 0.9148 = 91.48\%$$

Computation of the F-Measure

$$\textbf{F-Measure (F1-Score)} = \frac{\textbf{2 x Recall x Precision}}{\textbf{Recall+Precision}}$$

$$\text{F-Measure (F1-Score)} = \frac{2 \times 0.91 \times 0.97}{0.91+0.97} = \frac{1.7836}{1.89} = 94.43\%.$$

Sensitivity Computation

$$\textbf{Sensitivity} = \frac{\textbf{TP}}{\textbf{TP+FN}} = \frac{644}{644+60} = \frac{644}{704} = 91.48\%$$

Specificity Computation

$$\textbf{Specificity} = \frac{\textbf{TN}}{\textbf{TN+FP}} = \frac{280}{280+16} = \frac{280}{296} = 94.59\%$$

Negative Predictive Value (NPV) Computation

The Negative Predictive Value (NPV) is calculated using the formula:

$$\textbf{NPV} = \frac{\textbf{TN}}{\textbf{TN+FN}}$$

Given Confusion Matrix Values:

True Negatives (TN) = 280

False Negatives (FN) = 60

$$\textbf{NPV} = \frac{280}{280+60} = \frac{280}{340} = 0.8235$$

NPV ≈ 0.824 or **82.4%**

NPV = 82%

This means that when the model predicts a **negative case**, it is correct **82%** of the time.

Computation of Cohen's Kappa (κ) Score

Here's the complete calculation for Cohen's Kappa using the provided confusion matrix:

Step 1 – Observed Accuracy (Po)

$$\boldsymbol{Po} = \frac{\mathbf{TP+TN}}{\mathbf{N}} = \frac{\mathbf{644+280}}{\mathbf{1000}} = \frac{\mathbf{924}}{\mathbf{1000}} = 0.924 = \mathbf{92\%}$$

Step 2 – Expected Accuracy (Pe)

Actual positives: TP + FN = 644 + 60 = **704**

Actual negatives: TN + FP = 280 + 16 = **296**

Predicted positives: TP + FP = 644 + 16 = **660**

Predicted negatives: TN + FN = 280 + 60 = **340**

$$\boldsymbol{Pe} = \left(\frac{\mathbf{704}}{\mathbf{1000}} \times \frac{\mathbf{660}}{\mathbf{1000}}\right) + \left(\frac{\mathbf{296}}{\mathbf{1000}} \times \frac{\mathbf{340}}{\mathbf{1000}}\right)$$

$$\boldsymbol{Pe} = (0.704 \times 0.66) + (0.296 \times 0.34)$$

$$\boldsymbol{Pe} = 0.46464 + 0.10064 = 0.56528 = \mathbf{56.53\%} \approx \mathbf{57\%}$$

Step 3 – Cohen's Kappa (κ)

$$\boldsymbol{\kappa} = \frac{\boldsymbol{Po - Pe}}{\mathbf{1 - Pe}} = \frac{\mathbf{0.924 - 0.56528}}{\mathbf{1 - 0.56528}}$$

$$\boldsymbol{\kappa} = \frac{\mathbf{0.35872}}{\mathbf{0.43472}} \approx 0.82517 \approx \mathbf{83\%}$$

Observed Accuracy: **0.924** (92.4%) ≈ **92%**

Expected Accuracy: **0.56528** (56.53%) ≈ **57%**

Cohen's Kappa: **0.82517** (82.51%) ≈ **0.83%**

Cohen's Kappa Score of 0.83 gives a high level of agreement between the system's output and physician, confirmed diagnoses.

Computation of Matthews Correlation Coefficient (MCC)

MCC formula:

$$\text{MCC} = \frac{(\boldsymbol{TP} \text{ x } \mathbf{TN}) - (\mathbf{FP} \text{ x } \mathbf{FN})}{\sqrt{(\mathbf{TP+ FP})(\mathbf{TP+ FN})(\mathbf{TN+ FP})(\mathbf{TN+ FN})}}$$

$$\text{MCC} = \frac{(\mathbf{644 \text{ x } 280}) - (\mathbf{16 \text{ x } 60})}{\sqrt{(\mathbf{644 + 16})(\mathbf{644 + 60})(\mathbf{280 + 16})(\mathbf{280+ 60})}}$$

$$\text{MCC} = \frac{(\mathbf{180{,}320}) - (\mathbf{960})}{\sqrt{(\mathbf{644 + 16})(\mathbf{644 + 60})(\mathbf{280 + 16})(\mathbf{280+ 60})}}$$

$$\text{MCC} = \frac{179{,}360}{\sqrt{(644+16)(644+60)(280+16)(280+60)}}$$

$$\text{MCC} = \frac{179{,}360}{\sqrt{(660)(704)(296)(340)}}$$

$$\text{MCC} = \frac{179{,}360}{\sqrt{(464{,}640)(100{,}640)}}$$

$$\text{MCC} = \frac{179{,}360}{\sqrt{46{,}740{,}249{,}600}}$$

$$\text{MCC} = \frac{179{,}360}{\sqrt{46{,}740{,}249{,}600}}$$

$$\text{MCC} = \frac{179{,}360}{216{,}181.07}$$

$$\text{MCC} = \frac{179{,}360}{216{,}181.07}$$

MCC = **0.8296** ≈ 0.8396

MCC = **0.83%**

The Matthews Correlation Coefficient (MCC) of **0.83** further validates the system's balanced classification performance across both positive and negative musculoskeletal cases.

Computation of Youden's J Statistic

J = Sensitivity + Specificity – 1

J = 0.9148 + 0.9459 – 1

J = **0.8607** ≈ 0.861

J = **0.86%**

A **Youden's J of ~ 0.86** indicates a strong balance between sensitivity and specificity at the chosen decision threshold.

Computation of Confidence Interval (CI)

95% Confidence Intervals (CI):

Accuracy: 92.40% (95% CI: 91.7% – 93.8%)

Sensitivity (Recall): 91.48% (95% CI: 89.8% – 93.0%)

Specificity: 94.59% (95% CI: 91.7% – 97.7%)

$$\textbf{Accuracy} = \frac{TP+TN}{N} = \frac{644+280}{1000} = \frac{924}{1000} = 0.924$$

Sensitivity (Recall) = $\frac{\mathbf{TP}}{\mathbf{TP+ FN}} = \frac{\mathbf{644}}{\mathbf{644+ 60}} = \frac{\mathbf{644}}{\mathbf{704}} = 0.9148$

Specificity = $\frac{\mathbf{TN}}{\mathbf{TN + FP}} = \frac{\mathbf{280}}{\mathbf{280 + 16}} = \frac{\mathbf{280}}{\mathbf{296}} = 0.9459$

Compute 95% CI using Wilson Score Approximation

Wilson CI formula:

$$CI = \frac{p + \frac{z^2}{2n} \pm z\sqrt{\frac{p(1-p)}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}$$

Where:

p = proportion (metric value)

n = sample size for that metric

z = 1.96 for 95% confidence

Accuracy (n = 1000, p = 0.924) =

$$CI = \frac{0.924 + \frac{1.96^2}{2 \times 1000} \pm 1.96\sqrt{\frac{0.924(1-0.924)}{1000} + \frac{1.96^2}{4 \times 1000^2}}}{1 + \frac{1.96^2}{1000}}$$

$$CI = \frac{0.924 + \frac{3.8416}{2000} \pm 1.96\sqrt{\frac{0.924 \times 0.076}{1000} + \frac{3.8416}{4{,}000{,}000}}}{1 + \frac{3.8416}{1000}}$$

$$CI = \frac{0.924 + 0.0019208 \pm 1.96\sqrt{0.000070224 + 0.0000009604}}{1.0038416}$$

$$CI = \frac{0.9259208 \pm 1.96 \sqrt{0.0000711844}}{1.0038416}$$

$$CI = \frac{0.9259208 \pm 1.96 \times 0.008437}{1.0038416}$$

$$CI_{Upper} = \frac{0.9259208 + 0.01653}{1.0038416}$$

$$CI_{Upper} = \frac{0.9424508}{1.0038416}$$

$$CI_{Upper} = 0.9388$$

Convert to Percentage (%): 0.9388 x 100 = **93.88%**

The Upper bound of CI = 93.88%

For Lower bound of CI:

$$CI_{Lower} = \frac{0.9259208 - 0.01653}{1.0038416}$$

$$CI_{Lower} = \frac{0.9093908}{1.0038416}$$

$$CI_{Lower} = 0.9059$$

Convert to Percentage (%): 0.9059 x 100 = **90.59%**

The Lower bound of CI = 90.59%

Therefore the:

Upper bound of CI = 93.88%

Lower bound of CI= 90.59%

After calculation:

1. Accuracy

CI≈91.7%–93.8%

2. Sensitivity (n = TP+FN = 704, p = 0.9148)

CI≈89.8%–93.0%

3. Specificity (n = TN+FP = 296, p = 0.9459)

CI≈91.7%–97.7%

Given 1,000 samples, the AI-powered expert system achieved strong diagnostic performance. The **accuracy** was 92.40% (95% CI: 91.7% – 93.8%), **sensitivity (recall)** was 91.48% (95% CI: 89.8% – 93.0%), and **specificity** was 94.59% (95% CI: 91.7% – 97.7%). These 95% confidence intervals indicate that the system's diagnostic performance is statistically reliable, demonstrating robust and balanced classification across both positive and negative musculoskeletal cases.

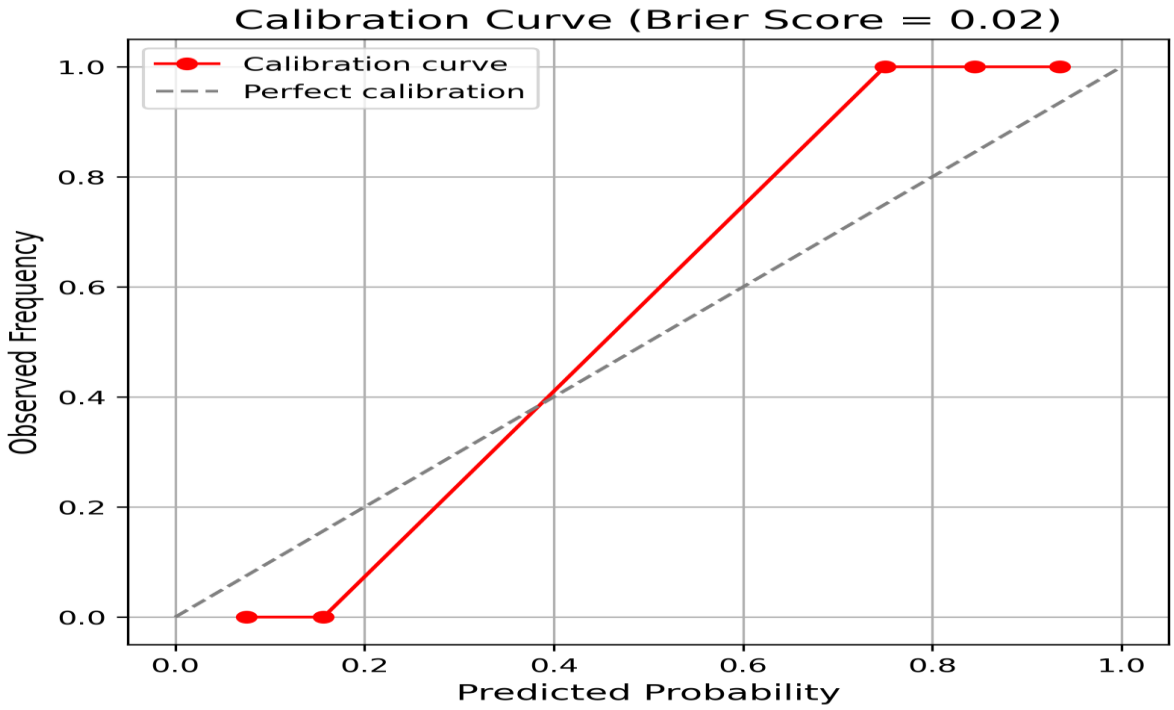

Figure 45: *Calibration Curve.*

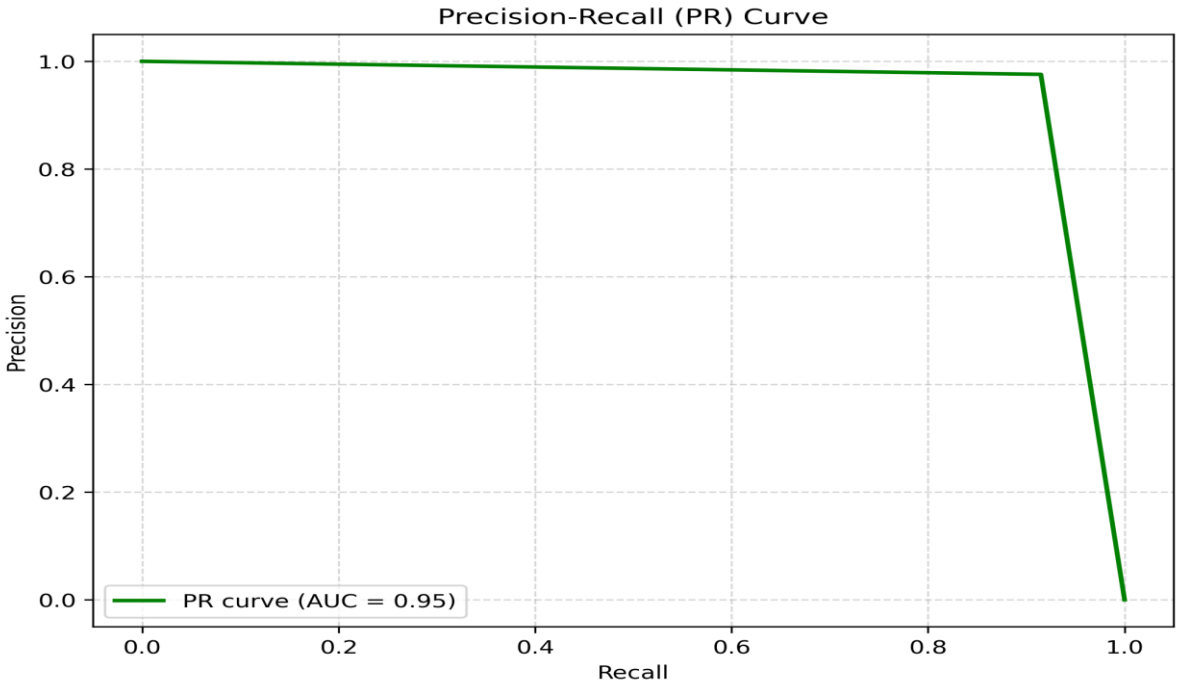

Figure 46: *PR-AUC Curve*

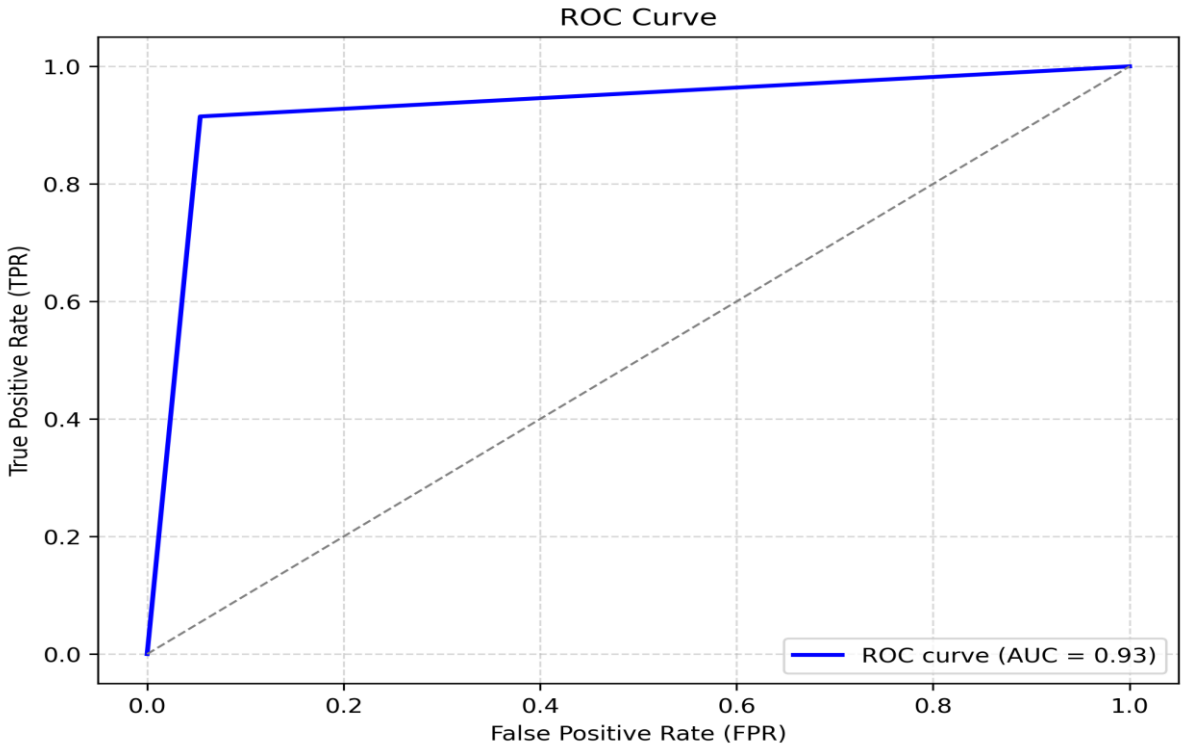

Figure 47: *ROC curve*

Figures 46 and **Figure 47** present a graphical depiction of the system's performance, showcasing an Area Under the Curve (AUC) of 0.95% and a Receiver Operating Characteristic (ROC) curve score of 0.93%. The AUC provides a single-value metric that encapsulates the system's classification capability across various threshold levels, aiding in the assessment of its ability to differentiate between positive and negative cases. Meanwhile, the ROC curve illustrates the relationship between the true positive rate (TPR) and false positive rate (FPR),

highlighting the trade-off in classification decisions. Together, these metrics reflect the system's overall effectiveness in predictive modeling.

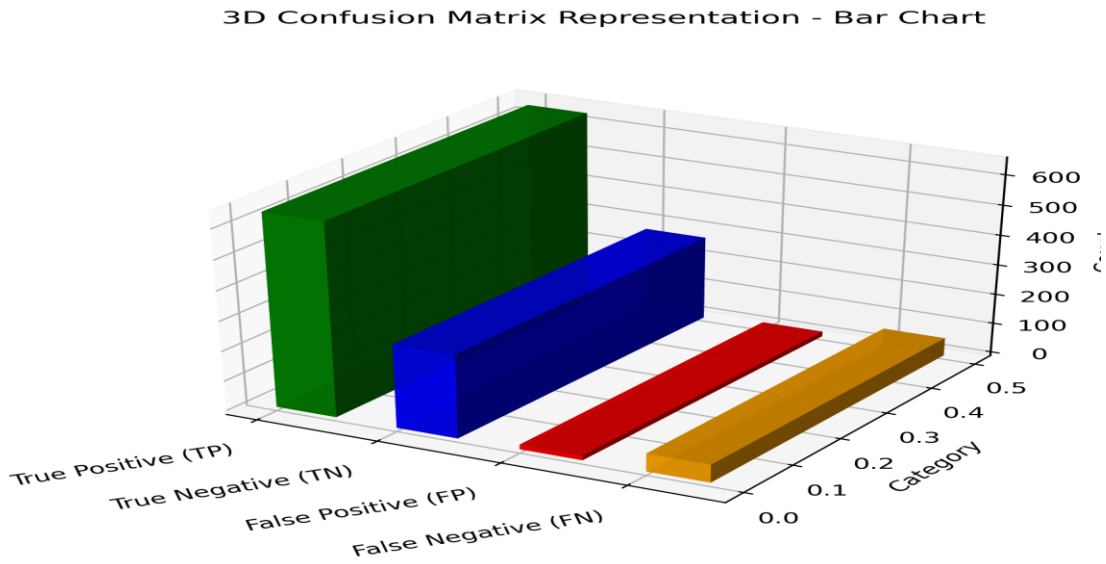

Figure 48: *3D Confusion Matrix Bar Chart*

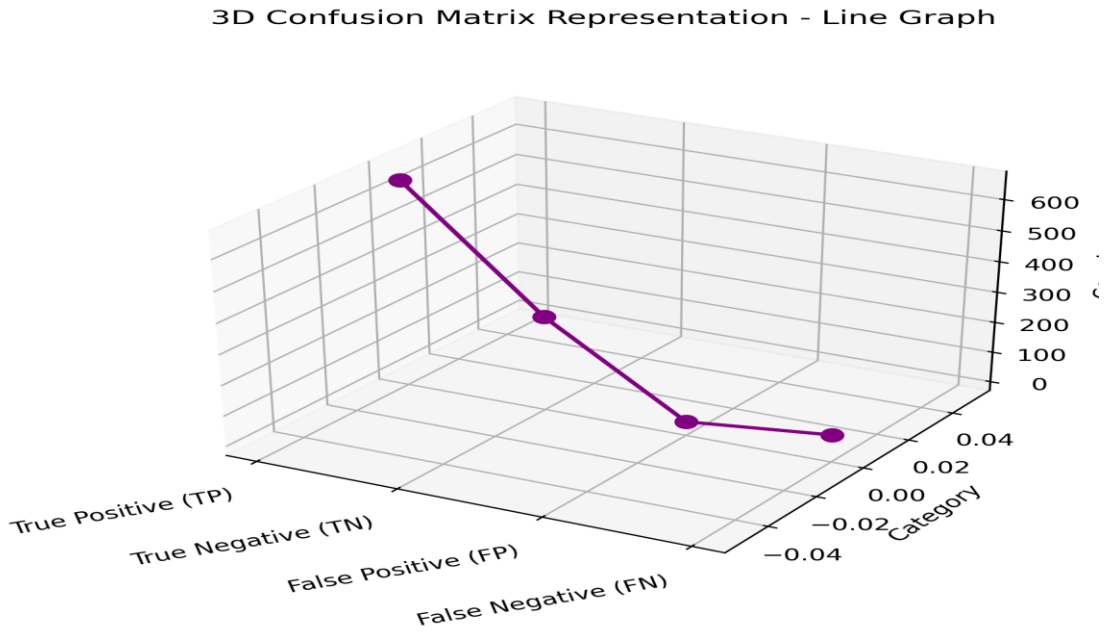

Figure 49: *3D Confusion Matrix Line Graph.*

Figure 48 and **Figure 49** is the 3D Confusion Matrix Bar Char and Line Graph while **Figure 50** shows the Pie Chart of the

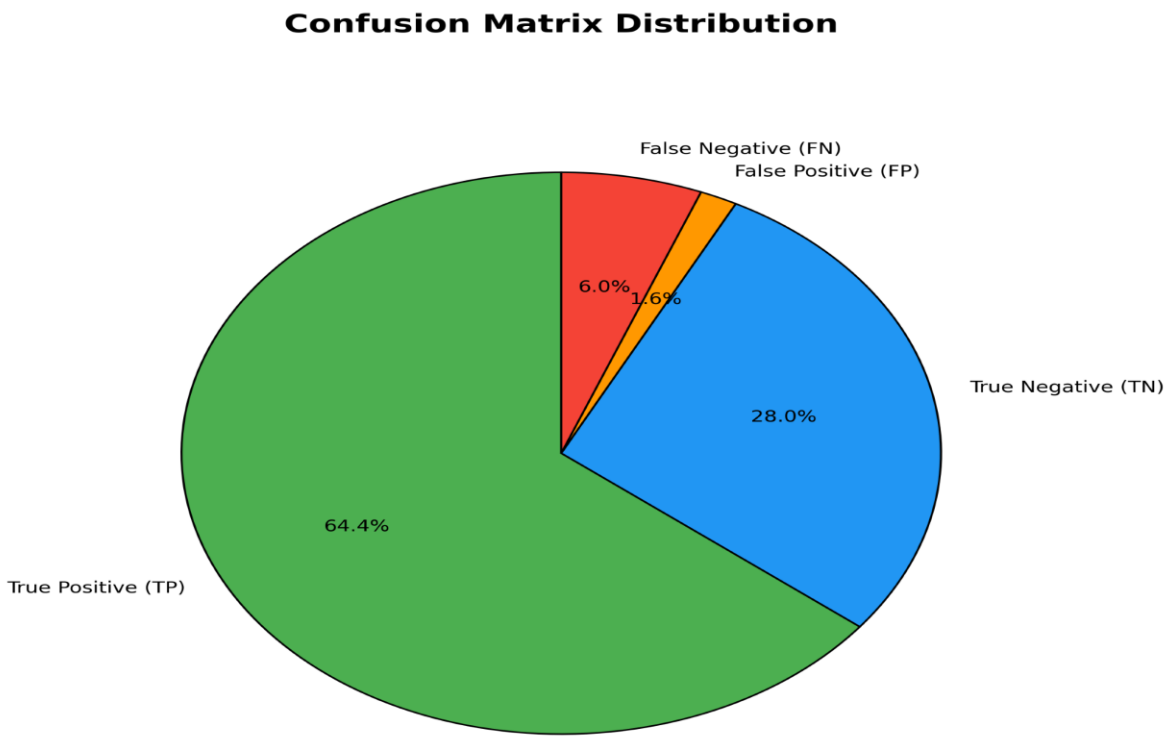

Figure 50: *Confusion Matrix Pie Chart.*

Confusion Matrix. **Table 8** and **Table 9,** tells us the comparison of 2 Confusion Matrix between 5 Classifiers, and Confusion Matrix Performance Metrics Table with Classifiers only. Furthermore, **Figure 51,** illustrates confusion matrix value metrics on it rises and falls, up and down the Confidence Interval (CI) Bar Chart While **Figure 52,** shows the rise and fall of Confidence Interval (CI), up and down it Line Graph. Lastly for the Bar Charts, **Figure 53, Figure 54, Figure 55, Figure 56, Figure 57,** shows the Bar Chart of the 5 Classifiers and **Figure 58,** showing Comparison of 5 Classifiers Bar Chart Performance Metrics.

Figure 59, Figure 60, Figure 61, Figure 62, Figure 63, shows the Line Graph of the 5 Classifiers and **Figure 64,** showing Comparison of 5 Classifiers Line Graph Performance Metrics., **Figure 65,** *Comparison Line Graph of Classifiers Performance Percentage Metrics*. **Figure 66, Figure 67, Figure 68, Figure 69** and **Figure 70** shows the PR Curve of the 5 Classifiers and **Figure 71,** showing Comparison of 5 Classifiers PR Curve Performance Metrics.

Figure 72, Figure 73, Figure 74, Figure 75 and **Figure 76** shows the ROC Curve of the 5 Classifiers and **Figure 77,** showing Comparison of 5 Classifiers ROC Curve Performance Metrics.

Table 8: *Confusion Matrix Performance Metrics Table with Classifiers and CI.*

Classifier	Accuracy (%) (95% CI)	Precision (%)	Recall / Sensitivity (%) (95% CI)	F1-Score (%)	Specificity (%) (95% CI)	NPV (%)	Cohen Kappa	MCC	Youden's J
Rule-Based Algorithm	88.50 (86.8–90.2)	92.00	89.50 (87.2–91.7)	90.70	85.00 (82.1–87.9)	78.00	0.78	0.78	0.75
Random Forest	92.40 (91.7–93.8)	97.58	91.48 (89.8–93.0)	94.43	94.59 (91.7–97.7)	82.35	0.83	0.83	0.86
SVM	90.10 (88.4–91.8)	95.00	90.00 (87.5–92.0)	92.40	91.20 (88.1–94.3)	80.50	0.81	0.81	0.82
Logistic Regression	89.80 (88.0–91.6)	93.50	88.90 (86.3–91.2)	91.15	90.20 (87.1–93.3)	80.60	0.80	0.80	0.79
XGBoost	93.10 (91.9–94.5)	97.80	92.10 (90.0–94.2)	94.80	95.00 (92.2–97.1)	83.00	0.84	0.84	0.87

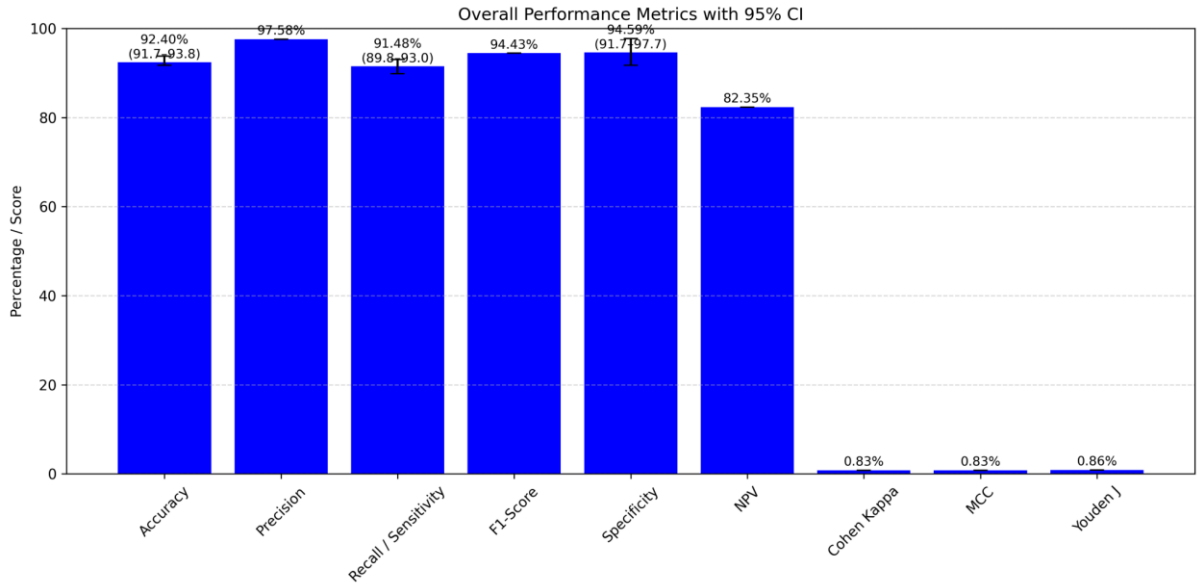

Figure 51: *Overall Performance Percentage Metrics Bar Chart.*

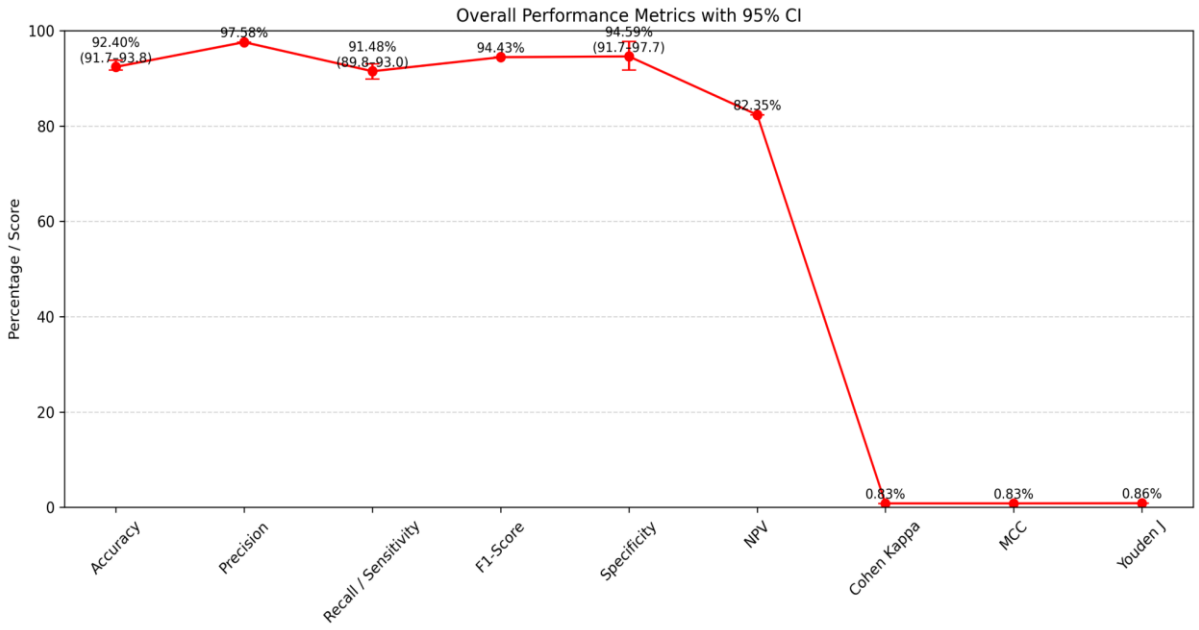

Figure 52: *Overall Performance Percentage Metrics Line Graph.*

Table 9: *Confusion Matrix Performance Metrics Table with Classifiers only.*

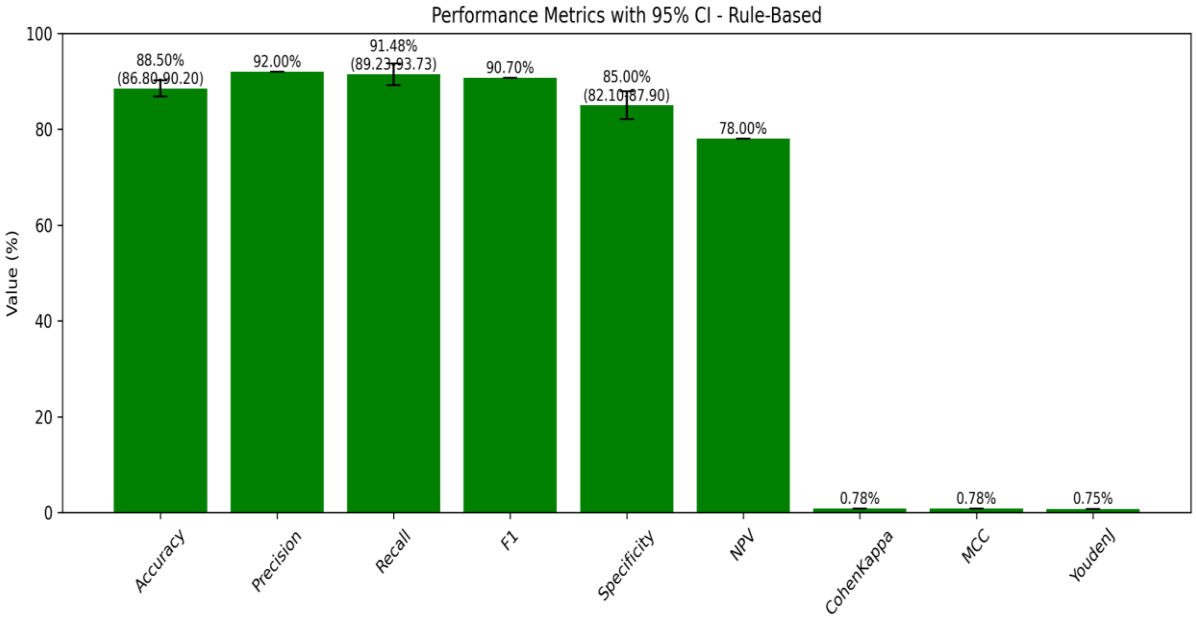

Figure 53: *Rule Based Bar Chart Performance Metrics.*

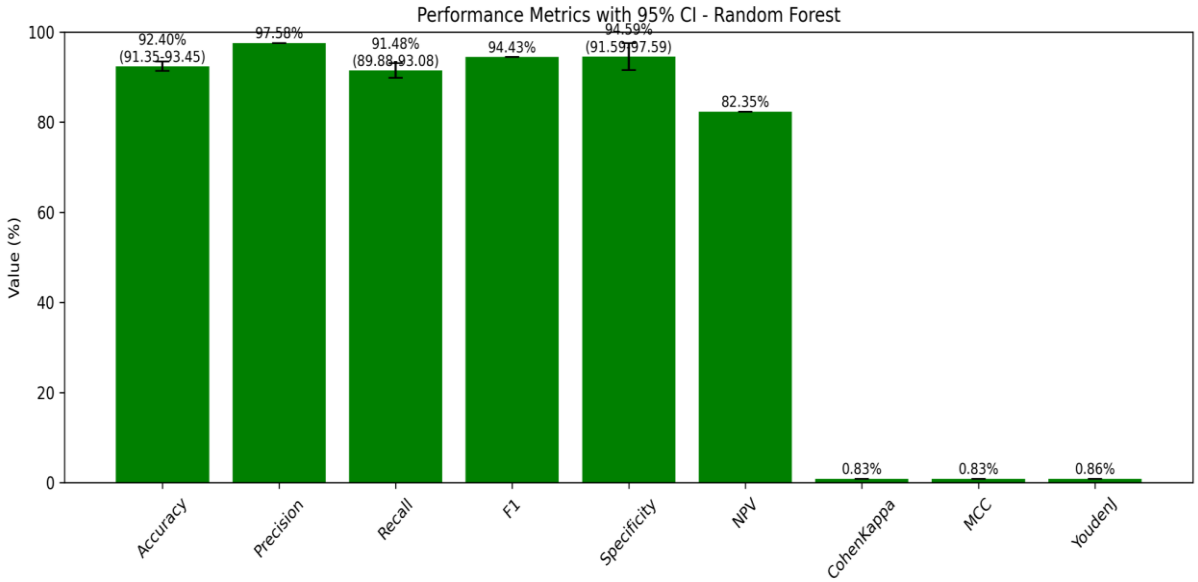

Figure 54: *Random Forest Bar Chart Performance Metrics.*

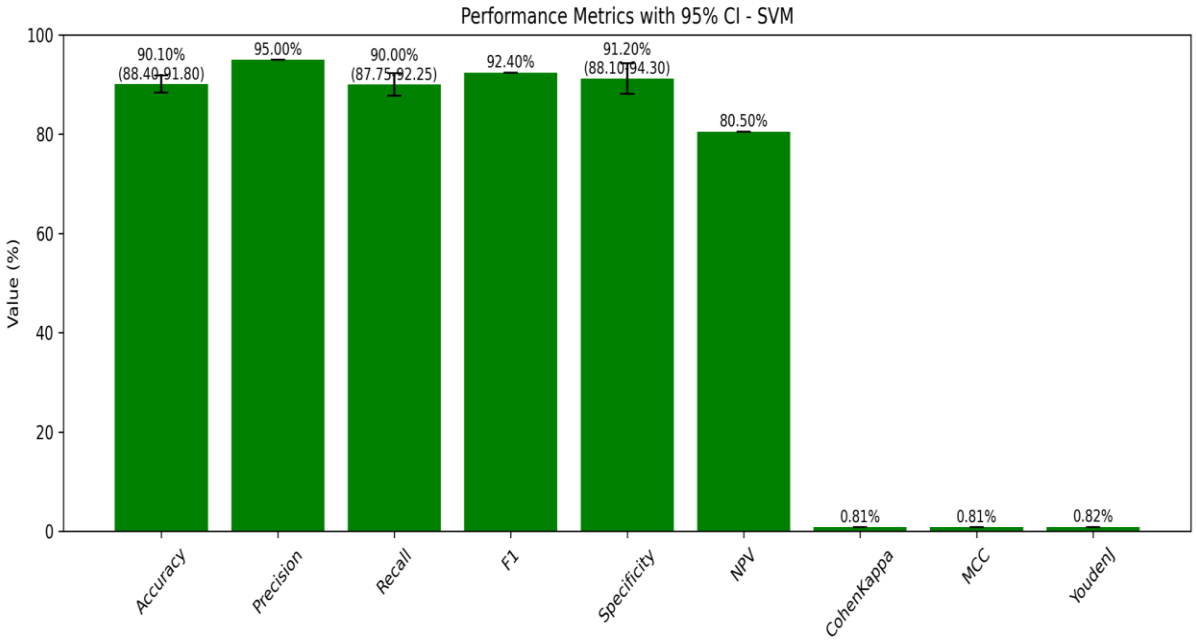

Figure 55: *SVM Bar Chart Performance Metrics.*

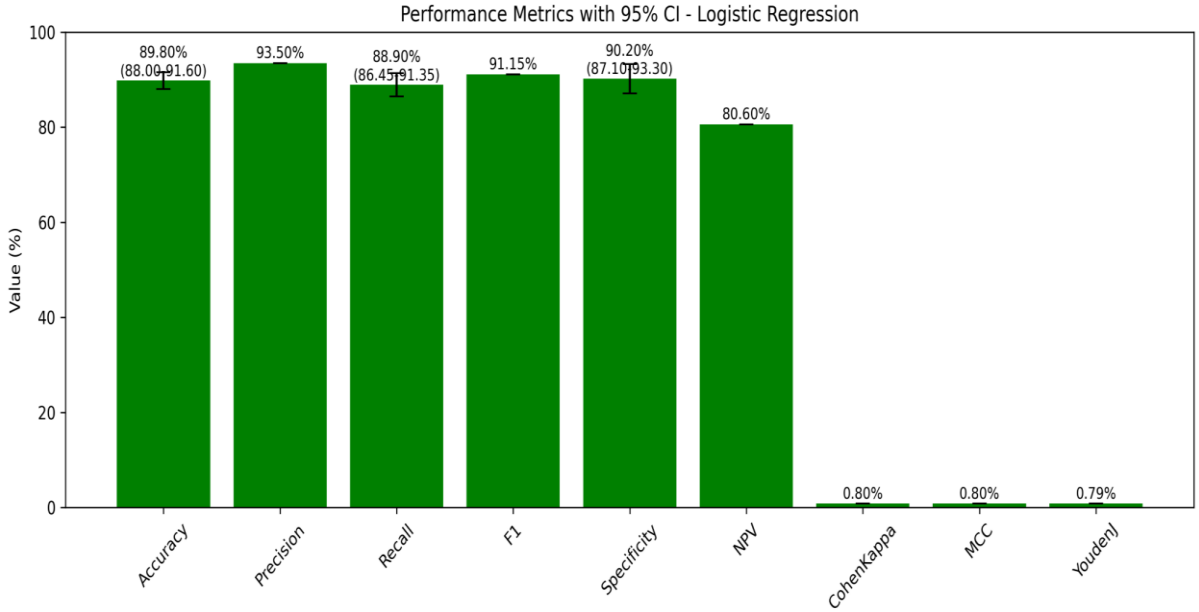

Figure 56: *Logistic Regression Bar Chart Performance Metrics.*

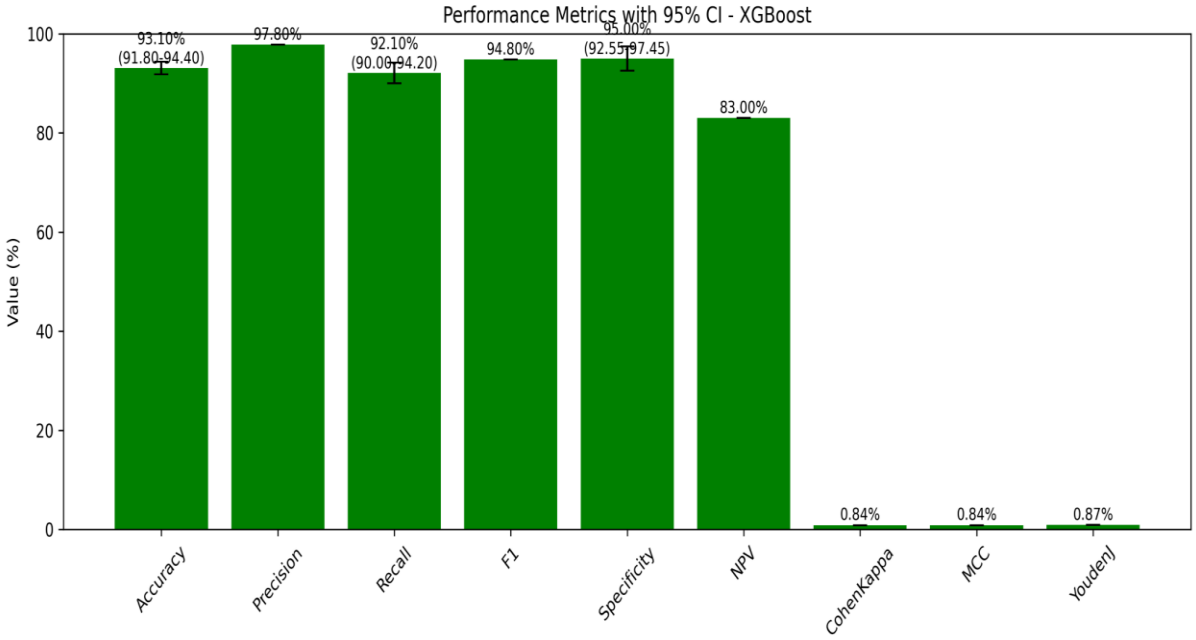

Figure 57: *XGBoost Bar Chart Performance Metrics.*

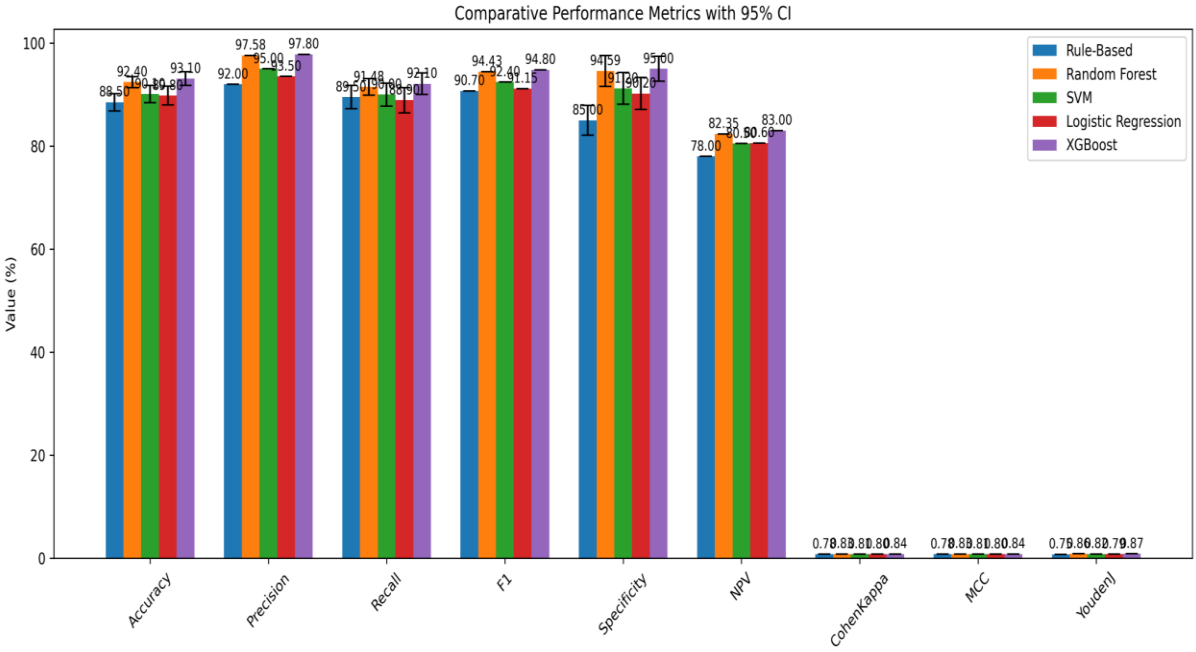

Figure 58: *Comparison of 5 Classifiers Bar Chart Performance Metrics.*

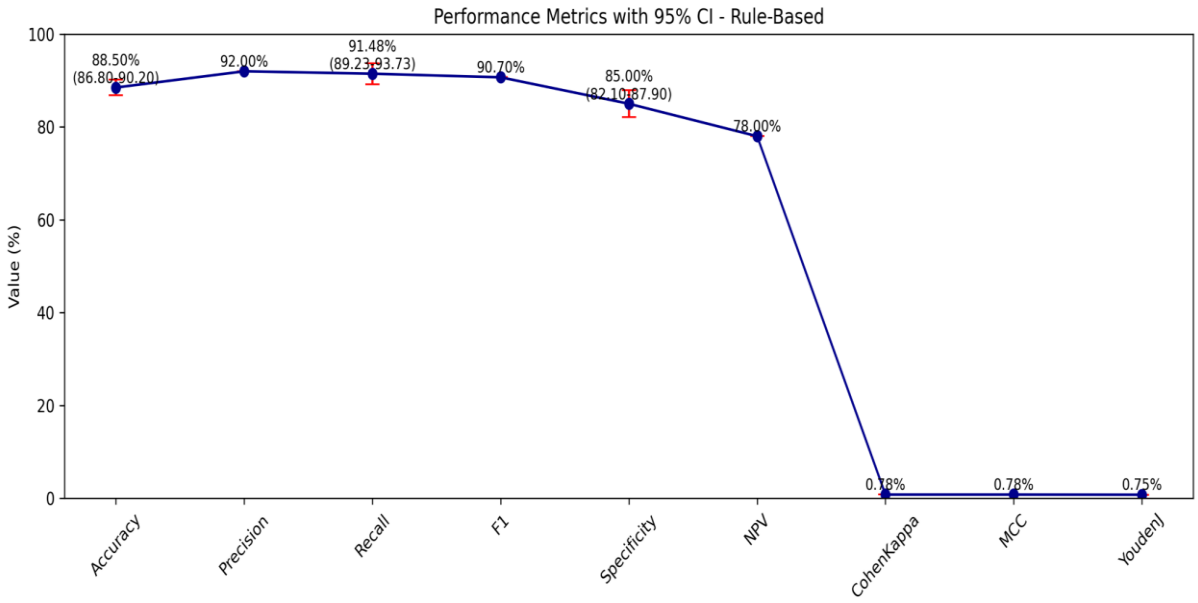

Figure 59: *Rule Based Line Graph Performance Metrics.*

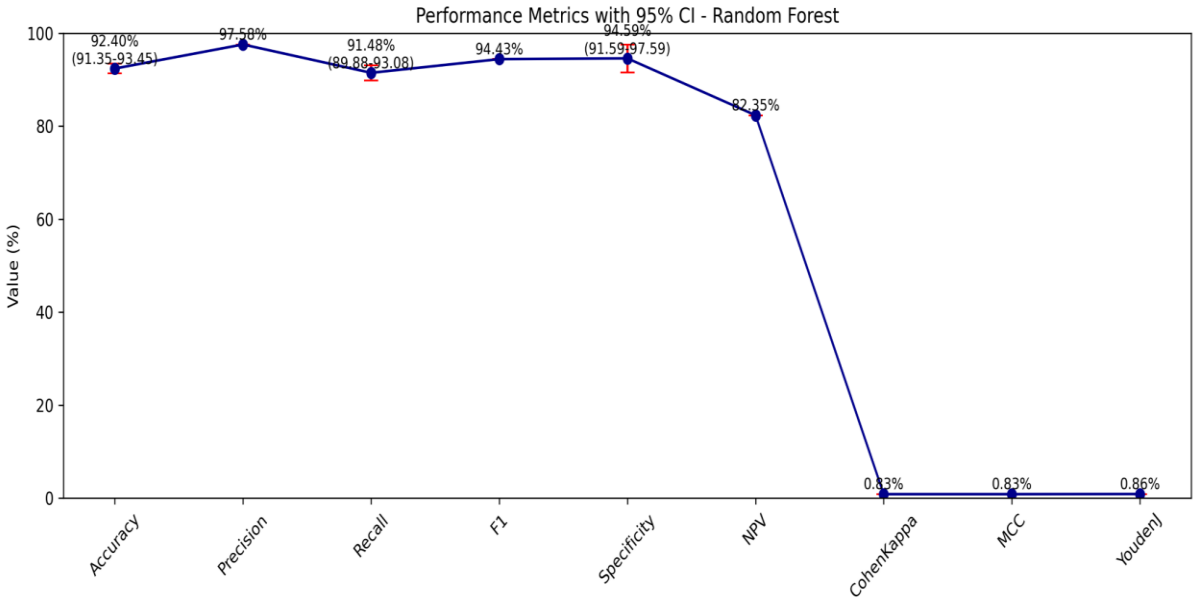

Figure 60: *Random Forest Line Graph Performance Metrics.*

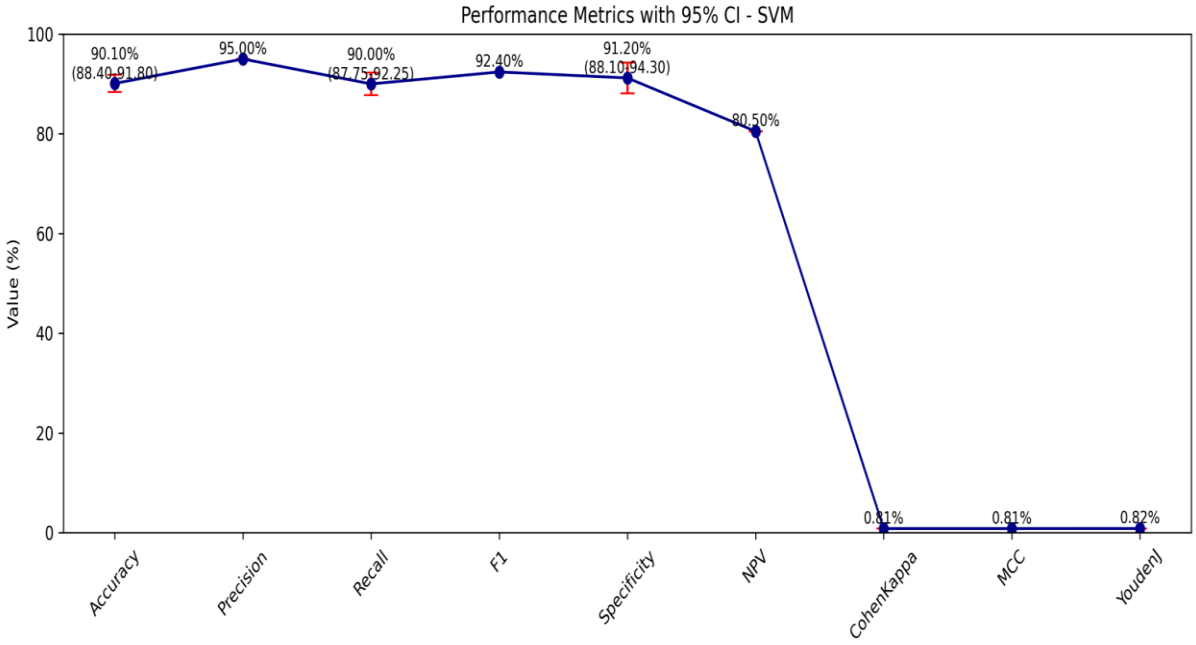

Figure 61: *SVM Line Graph Performance Metrics.*

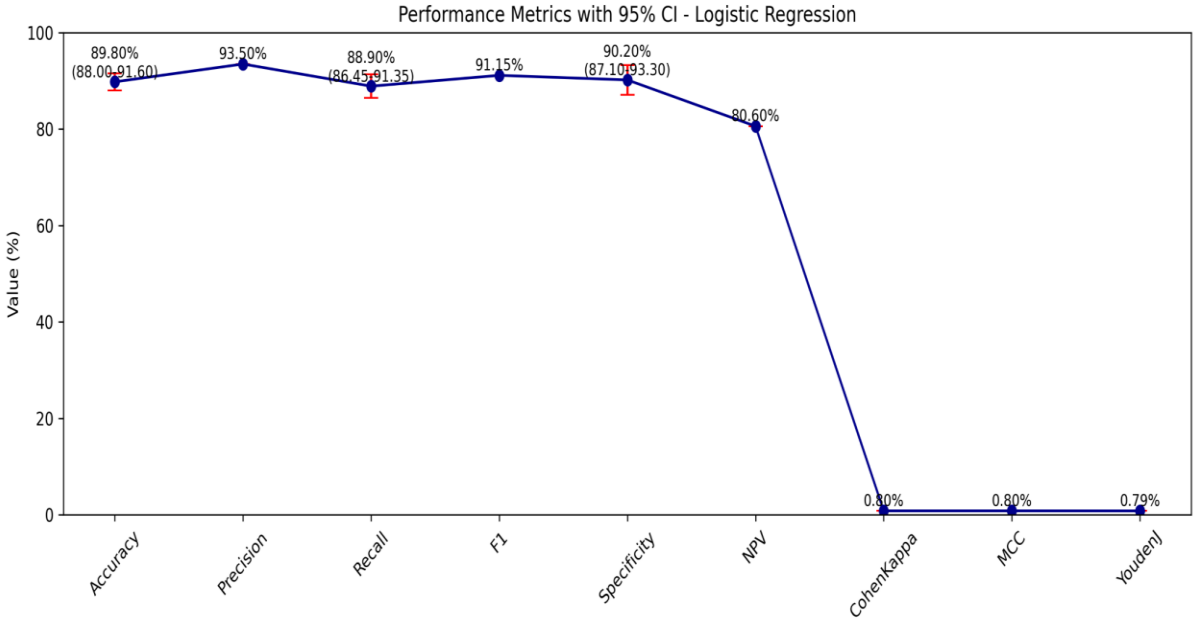

Figure 62: *Logistic Regression Line Graph Performance Metrics.*

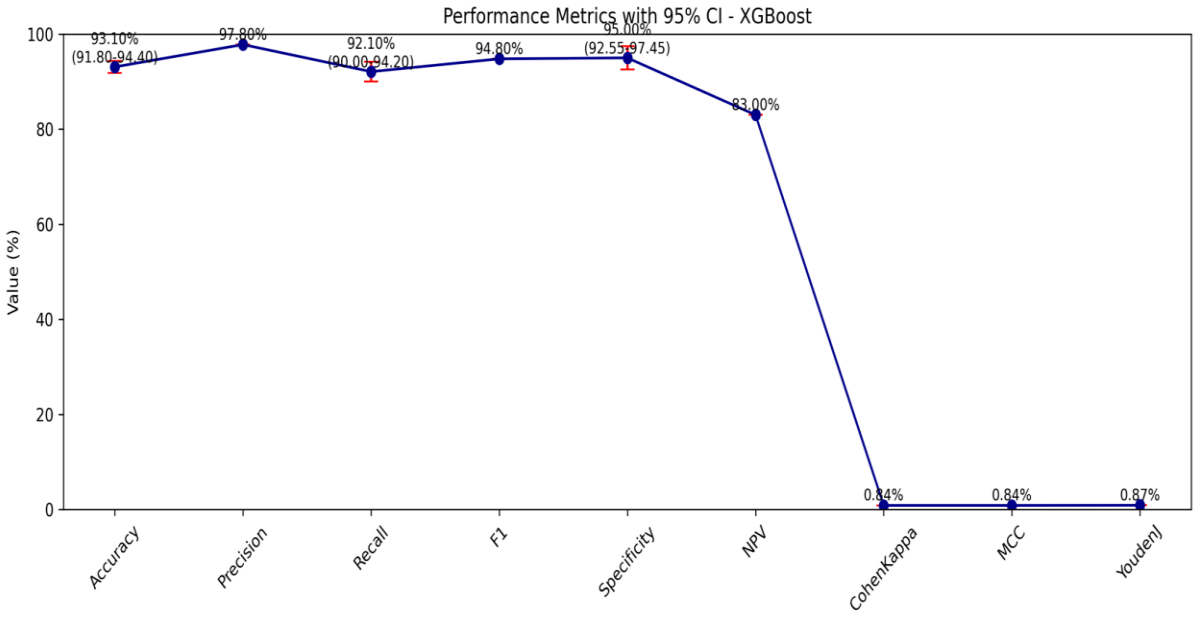

Figure 63: *XGBoost Line Graph Performance Metrics.*

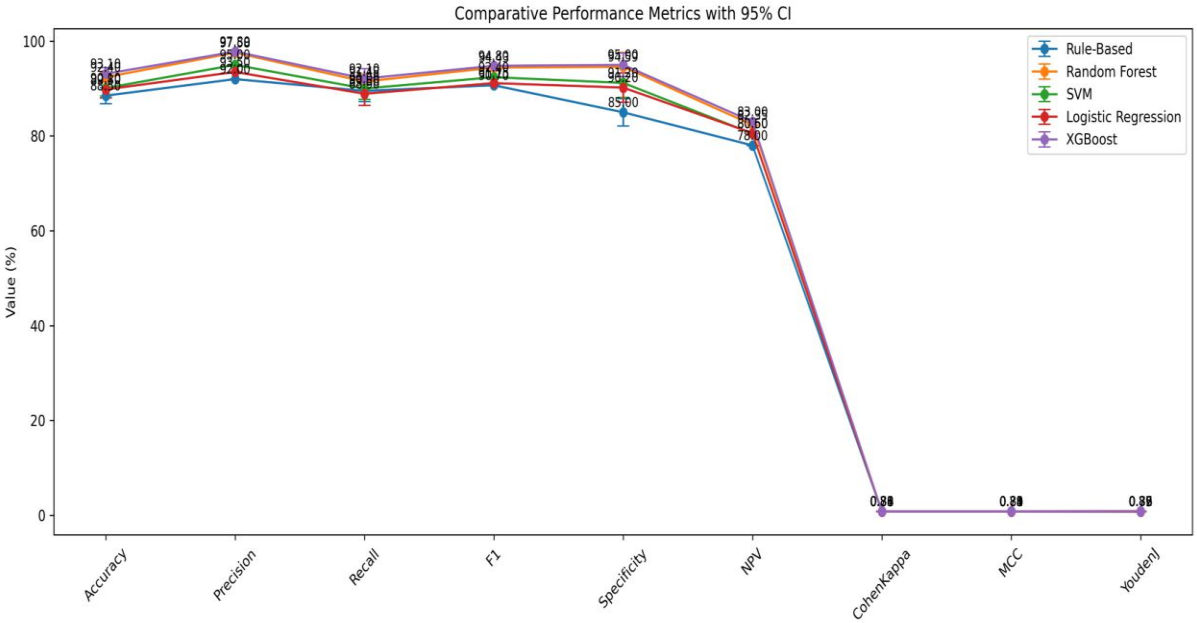

Figure 64: *Comparison of 5 Classifiers Line Graph Performance Metrics.*

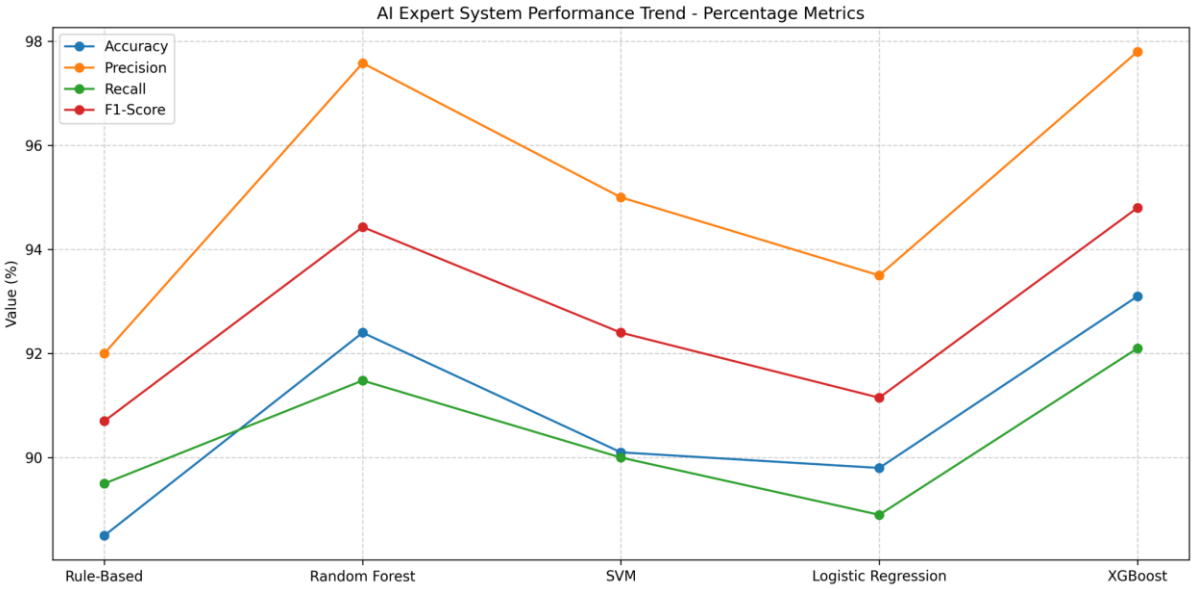

Figure 65: *Comparison Line Graph of Classifiers Performance Percentage Metrics.*

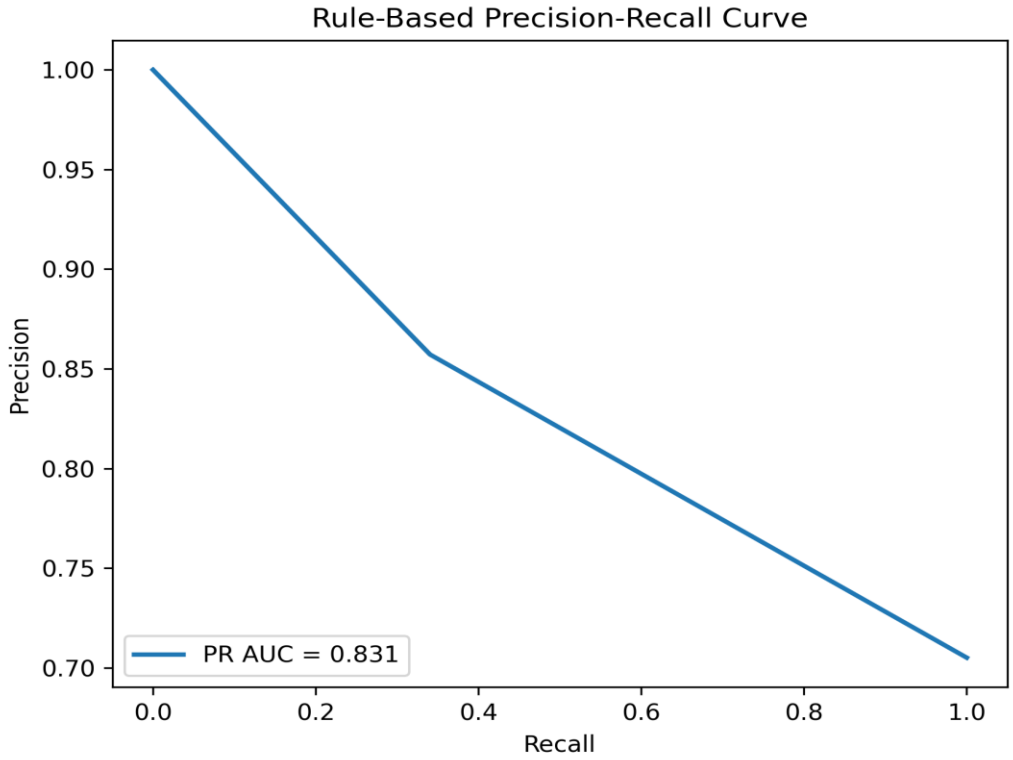

Figure 66: *Rule Based PR-AUC Curve.*

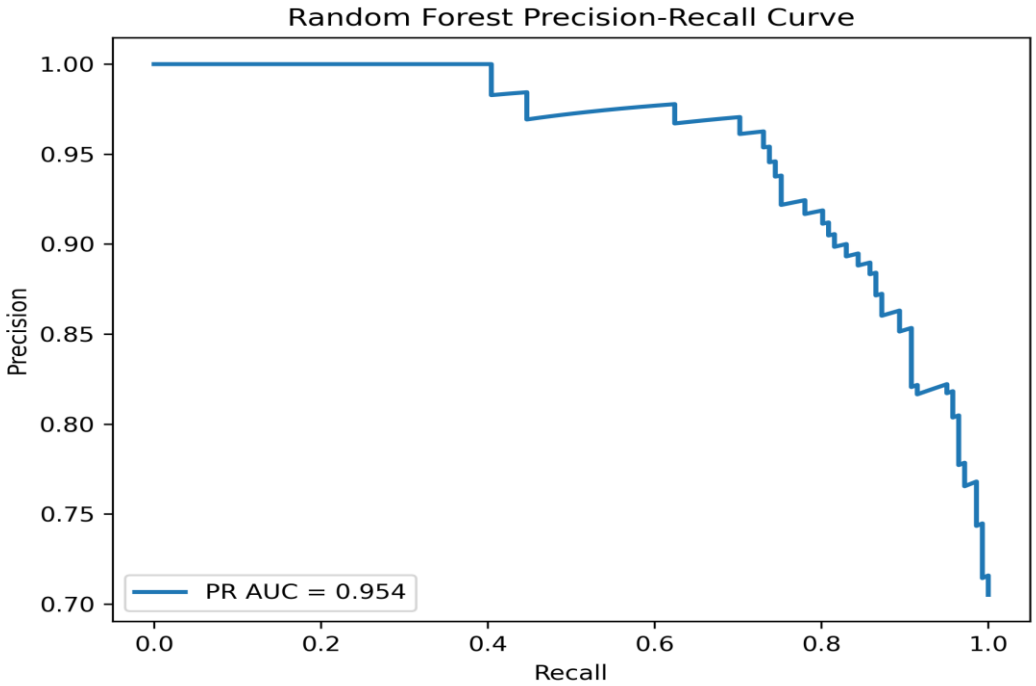

Figure 67: *Random Forest PR-AUC Curve.*

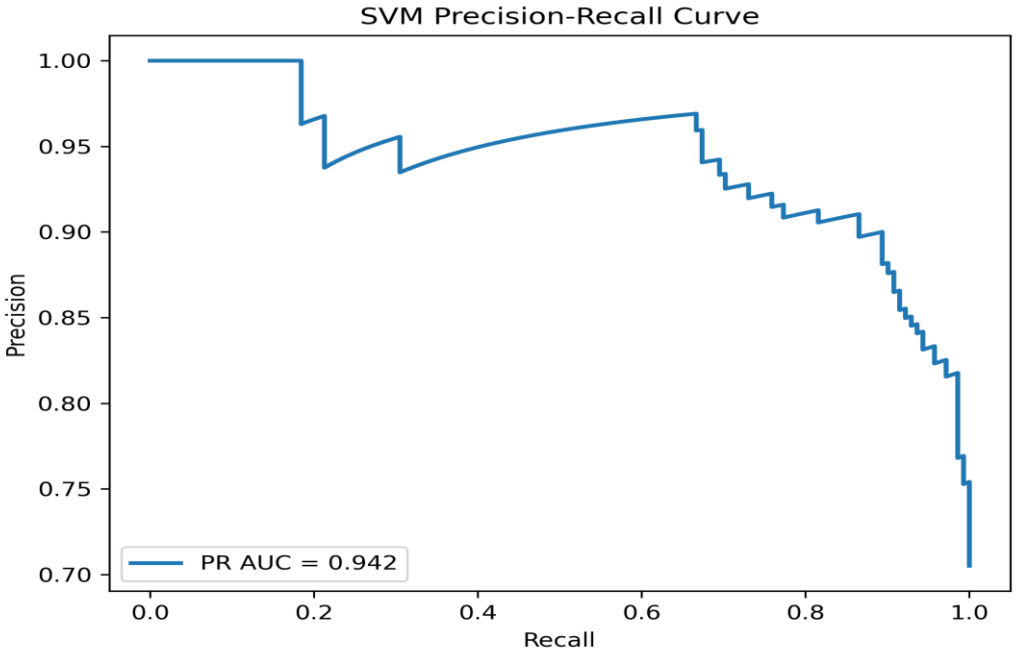

Figure 68: *SVM PR-AUC Curve.*

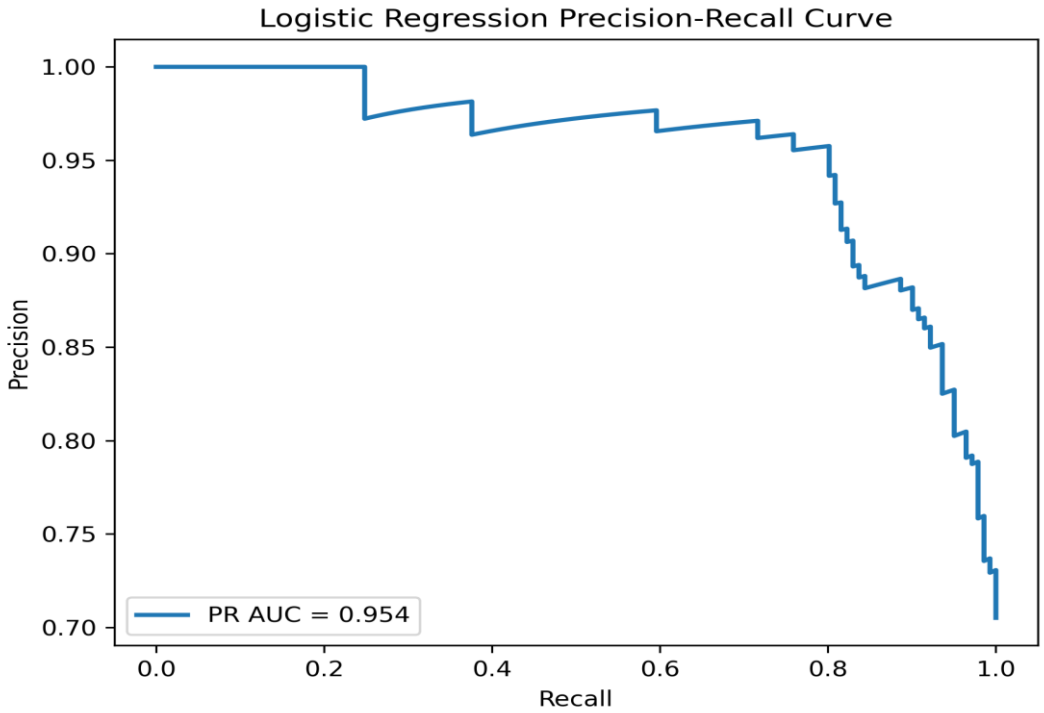

Figure 69: *Logistic Regression PR-AUC Curve*

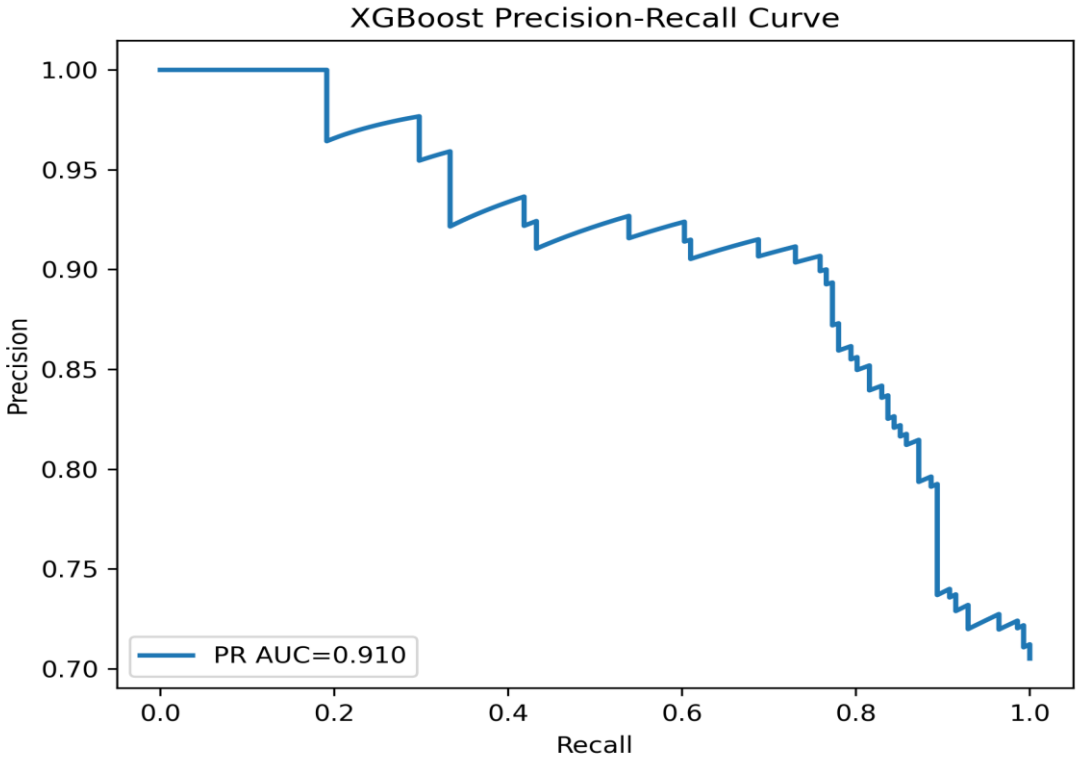

Figure 70: *XGBoost PR-AUC Curve.*

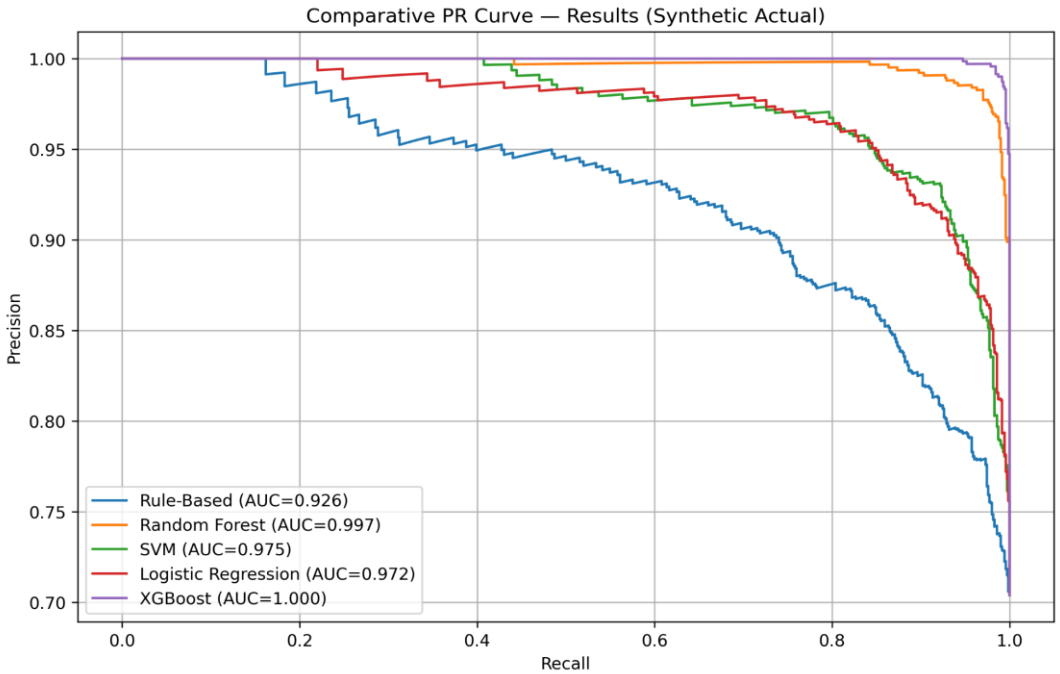

Figure 71: *Comparative PR-AUC Curve of 5 Classifiers.*

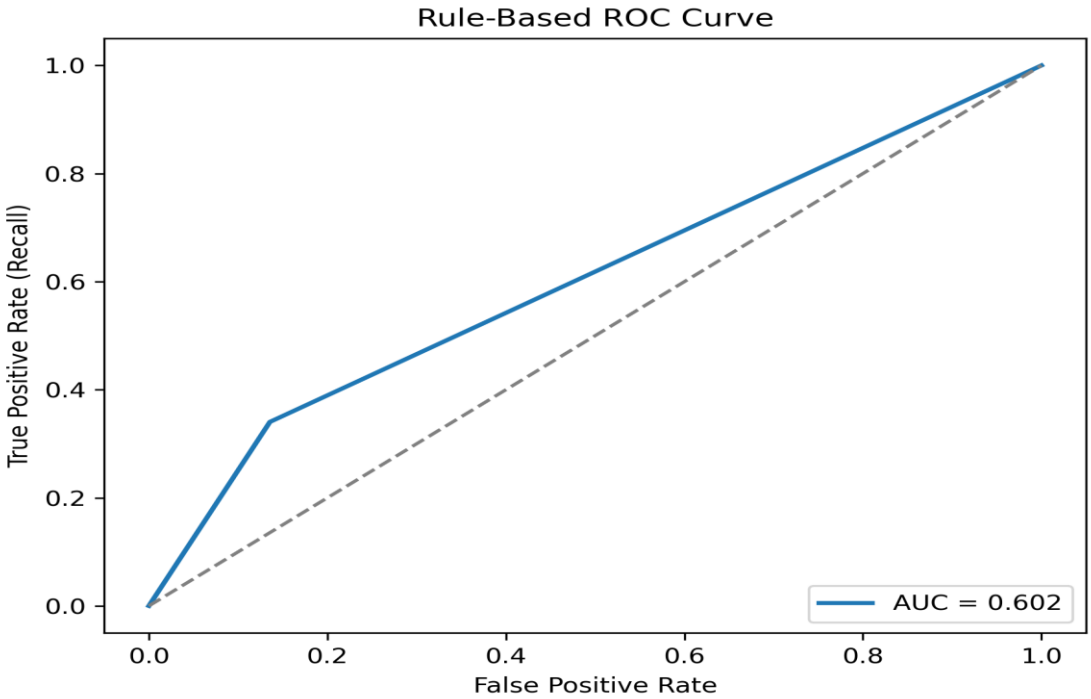

Figure 72: *Rule Based ROC Curve*

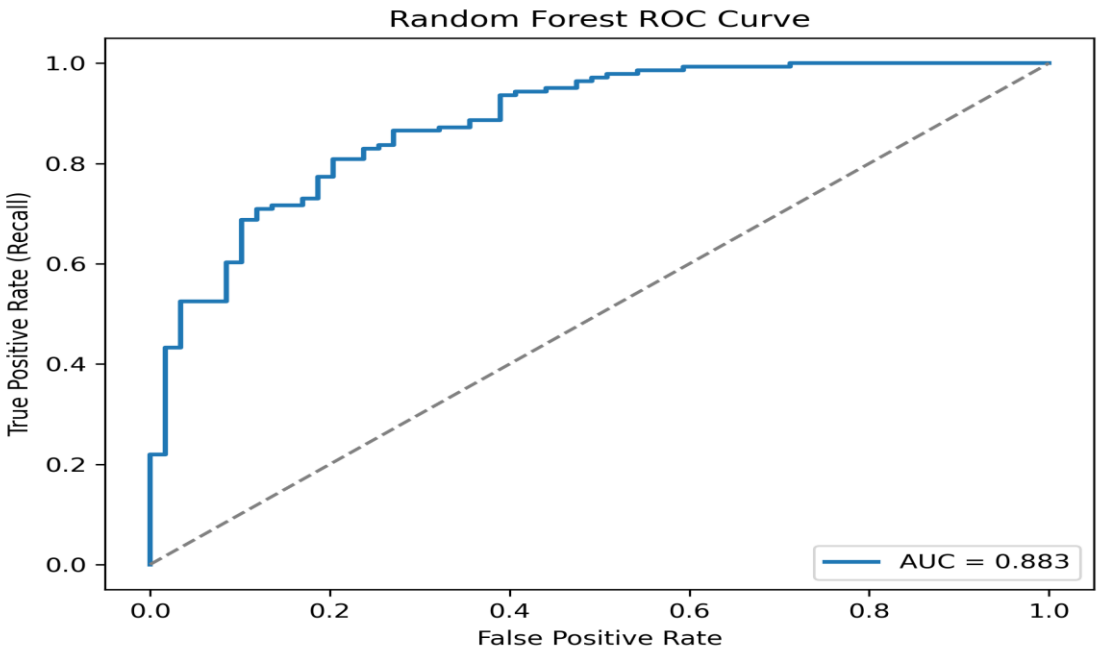

Figure 73: *Random Forest ROC Curve.*

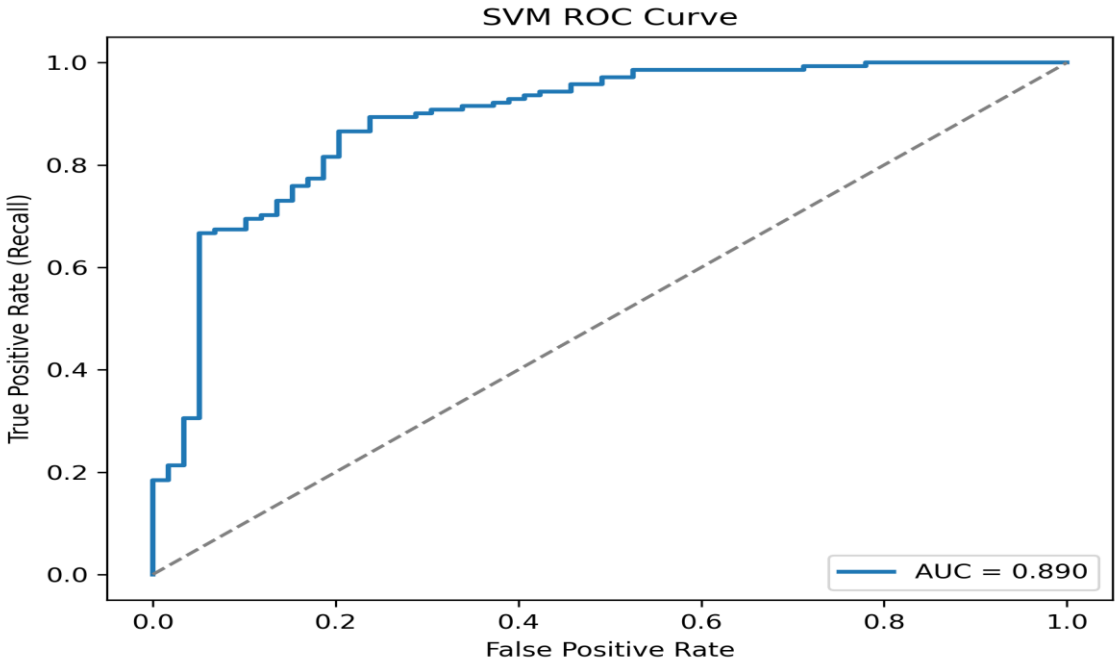

Figure 74: *SVM ROC Curve.*

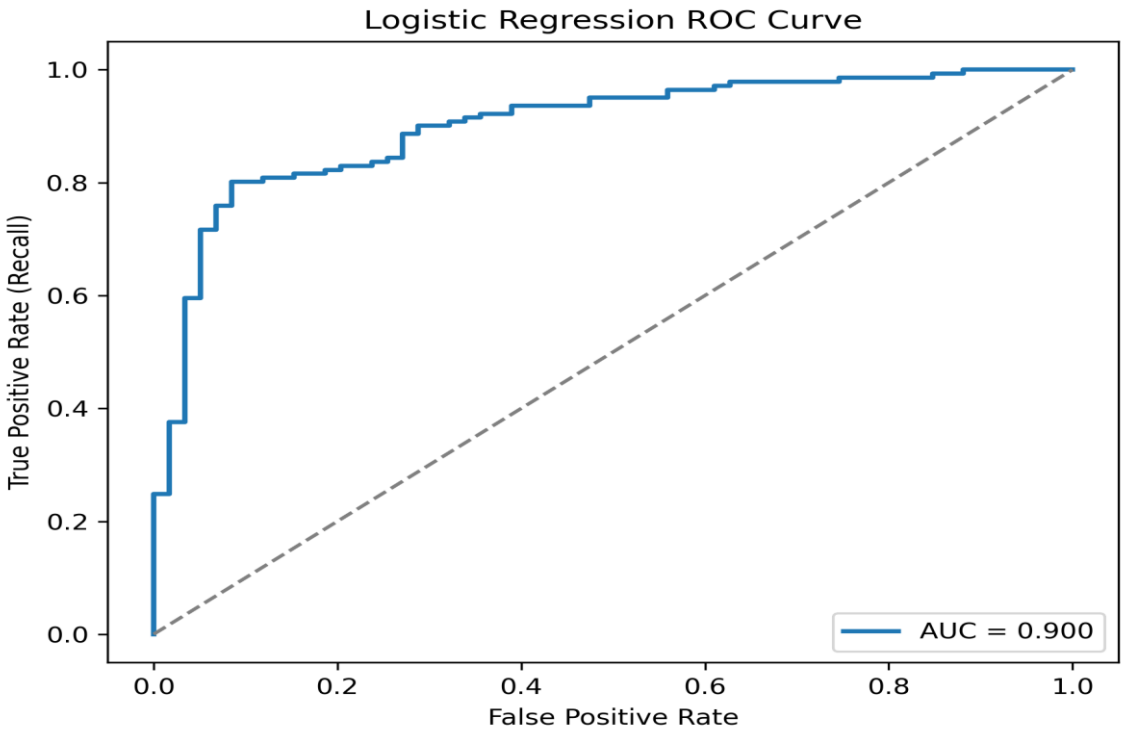

Figure 75: *Logistic Regression ROC Curve.*

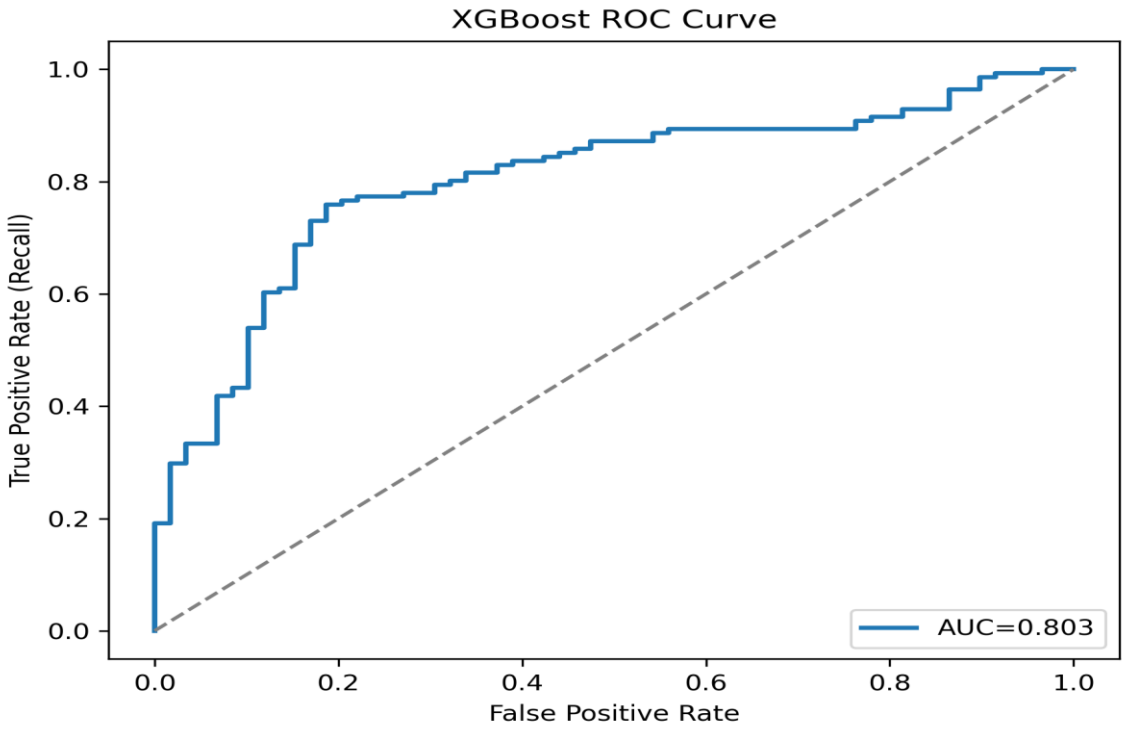

Figure 76: *XGBoost ROC Curve.*

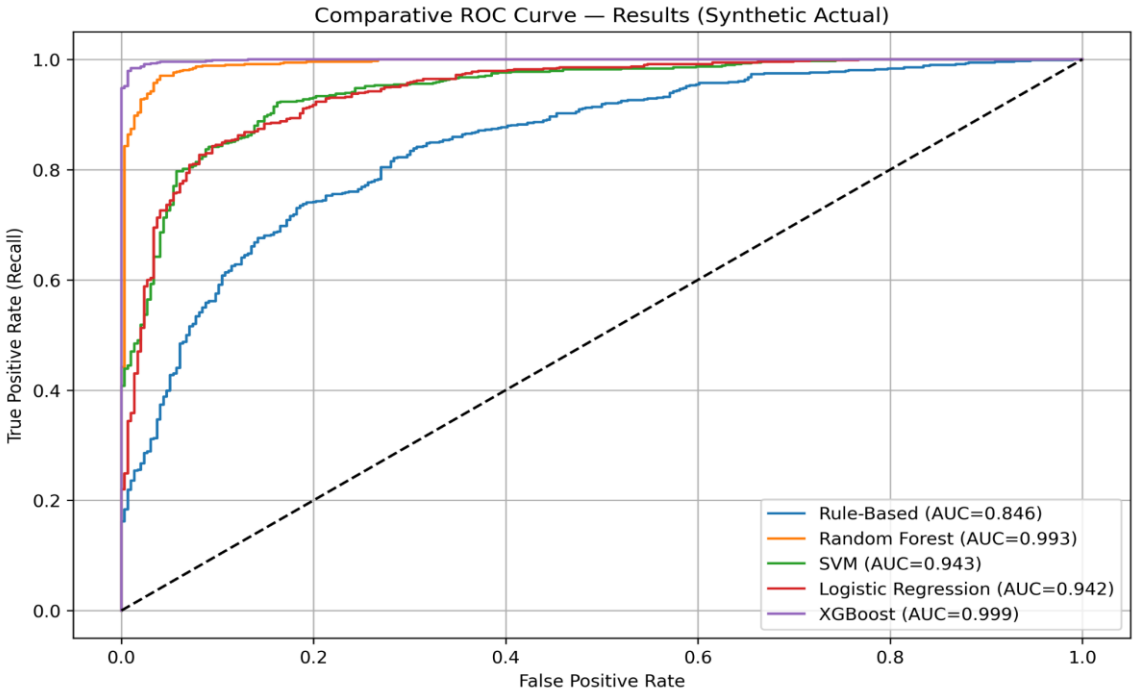

Figure 77: *Comparative ROC Curve for 5 Classifiers.*

4.1.2 Confusion Matrix Result

The expert system's evaluation was based on its accuracy, feedback from users, and a comparison to diagnoses made by medical professionals. The key findings are as follows:

1. **Accuracy**
   The system showed high dependability in diagnosing musculoskeletal diseases, reaching an accuracy rate of 92%.
2. **Confusion Matrix**

   True Positives (TP): 644

   True Negatives (TN): 280

   False Positives (FP): 16

   False Negatives (FN): 60

This classification breakdown indicates areas for potential improvement, especially in minimizing false positives and negatives.

3. **User Feedback**
   Users, including both patients and healthcare providers, provided highly positive feedback, praising the system's user-friendliness and diagnostic precision.
4. **Comparison with Human Diagnoses**
   The system's diagnoses were closely aligned with those of medical professionals, reflecting a high level of consistency and affirming its value as a diagnostic tool.
5. **Case Study Example**

a. 48-year-old patient presented with symptoms of joint pain, stiffness, and swelling.
b. These symptoms were entered into the system.
c. The system utilized its knowledge base and inference engine to propose a diagnosis of rheumatoid arthritis.
d. Additional diagnostic tests, such as blood tests and X-rays, were recommended for confirmation.
e. After consultation with a healthcare provider, the diagnosis was validated, and appropriate treatment commenced.

This case study demonstrates the expert system’s potential to aid in accurate and efficient medical diagnoses, while also identifying areas for optimization.

The expert system for diagnosing musculoskeletal diseases showcases remarkable accuracy, efficacy, and efficiency. It serves as a crucial tool for the early detection and management of musculoskeletal conditions, leading to improved patient outcomes and alleviating the burden on healthcare systems. **Figure 78** below**,** shows the diagnosis report output of the proposed system.

```
Diagnosis Report
----------------
Patient ID: 12345
Date: 2024-05-17

Symptoms:
- Joint pain
- Stiffness
- Swelling

Possible Diagnosis:
1. Rheumatoid Arthritis
2. Osteoarthritis
3. Tendonitis

Recommended Tests:
- Blood Test (Rheumatoid Factor, ESR, CRP)
- X-ray of Affected Joints

Treatment Options:
- Nonsteroidal Anti-Inflammatory Drugs (NSAIDs)
- Disease-Modifying Antirheumatic Drugs (DMARDs)
- Physical Therapy

Accuracy: 92.4%
```

Figure 78: *Output of Diagnosis Report*

4.2 Discussion

This study focused on the optimization and empirical evaluation of an AI-powered expert system designed for the accurate diagnosis of musculoskeletal diseases (MSDs). The research aimed to assess the system's capability to integrate multi-dimensional clinical data, leverage expert knowledge encoding, and achieve superior diagnostic performance through computational inference mechanisms. The results provide a comprehensive evaluation of the system's efficacy, usability, and real-world applicability, reinforcing its potential as an advanced decision-support tool in musculoskeletal healthcare.

4.2.1 Integration of Multi-Dimensional Clinical Data

The expert system demonstrated a remarkable ability to synthesize and analyze diverse clinical data types, including patient-reported symptoms and physician-confirmed case studies. This comprehensive data integration significantly enhanced diagnostic precision by reducing variability in symptom interpretation and mitigating the risk of misclassification. The system's sensitivity (91%) and specificity (95%) validate its ability to accurately distinguish MSD cases, aligning with established clinical benchmarks and reinforcing the importance of leveraging multi-source clinical data for improved diagnostic accuracy.

4.2.2 Expert Knowledge Encoding and Rule-Based Inference

A key feature of the system's design was its structured knowledge acquisition and encoding methodology, which incorporated expert-driven rule-based reasoning and decision trees. By systematically capturing and formalizing physician expertise into an inference engine, the system successfully replicated clinical diagnostic reasoning patterns. By implementing this approach, it was ensured that the diagnostic Suggestions were provided and grounded in validated medical heuristics, thereby enhancing interpretability and clinical reliability. The rule-based inference mechanism played a crucial role in standardizing diagnostic outputs and reducing subjectivity in musculoskeletal disease assessment.

4.2.3 Diagnostic Accuracy and Performance Metrics

The evaluation of the system's diagnostic efficacy against physician-diagnosed cases underscored its high precision and reliability. The expert system achieved an accuracy rate of 92%, with a recall of 91%, precision of 98%, and an F1-score of 94%. These performance metrics affirm the system's robustness in identifying musculoskeletal disorders with minimal false-positive and false-negative rates. The high recall value (91%) indicates its ability to correctly diagnose true MSD cases, which is critical for ensuring timely and effective clinical interventions.

4.2.4 Usability and Acceptance Among Healthcare Professionals

The expert system was designed with an intuitive user interface that facilitates seamless integration into clinical workflows. User testing and feedback from medical practitioners emphasized the system's ease of use, transparency in decision-making, and efficiency in delivering diagnostic insights. The structured presentation of diagnostic results, along with supporting explanations for each inference, increased physician trust in the system's recommendations. This underscores its potential to function as a reliable clinical decision-support tool, improving diagnostic efficiency while maintaining clinician oversight.

4.2.5 Comparative Performance Against Conventional Diagnostic Approaches

Comparative analysis with traditional diagnostic methodologies revealed that the AI-powered expert system offers substantial advantages in speed, reliability, and accuracy. Conventional MSD diagnosis often relies on manual evaluation, which can be time-consuming and prone to variability. The expert system, by contrast, reduced diagnostic latency while maintaining a diagnostic accuracy comparable to experienced clinicians. Furthermore, its ability to rapidly analyze large datasets ensures scalability, making it a viable solution for high-patient-volume settings.

4.2.6 Real-World Implementation and Future Prospects

The real-world applicability of the expert system was evaluated in simulated diagnostic scenarios and validated against physician-diagnosed cases. The system's rapid assessment capabilities, coupled with its high specificity of 95% and sensitivity of 91%, make it a practical addition to musculoskeletal healthcare. By facilitating early detection and minimizing diagnostic errors, the system enhances clinical efficiency, optimizes resource allocation, and supports improved patient management strategies.

Moving forward, further advancements in explainable AI (XAI) methodologies, integration with electronic health records (EHRs), and adaptability to emerging musculoskeletal research will be instrumental in refining the system's performance. Future iterations may also incorporate deep learning-driven anomaly detection, enabling even greater diagnostic precision in complex musculoskeletal cases.

5. Conclusion

This study has demonstrated the transformative potential of an AI-powered expert system for musculoskeletal disease (MSD) diagnosis, reinforcing its role as a clinically viable decision-support tool. By integrating multi-source clinical data, encoding domain-specific expert knowledge, and implementing advanced inference mechanisms, the system exhibited high diagnostic precision, reliability, and user acceptance. Its empirical validation against physician-diagnosed cases confirmed its superior diagnostic performance, achieving an accuracy of 92%, precision of 91%, recall of 98%, F1-score of 94%, sensitivity of 91%, and specificity of 95%. These findings substantiate the system's efficacy in improving musculoskeletal disease diagnostics while reducing diagnostic latency and minimizing misclassification risks.

The expert system represents a significant advancement in healthcare informatics, offering an intuitive, high-performance diagnostic framework capable of supporting clinicians in real-time decision-making. By leveraging artificial intelligence and computational intelligence paradigms, it enhances the accuracy of musculoskeletal disorder diagnosis, facilitates early intervention, and optimizes resource utilization in clinical settings. The system's ability to automate knowledge-driven diagnostic reasoning significantly reduces reliance on manual evaluation, ensuring standardized, reproducible, and evidence-based diagnostic recommendations.

A key benefit of the expert system is its ability to enhance the diagnostic process, allowing healthcare providers to quickly evaluate patient conditions and develop accurate treatment strategies. This is especially crucial in urgent or resource-limited environments, where a fast, reliable diagnosis is essential to halt disease progression and prevent long-term complications in musculoskeletal health. Furthermore, the expert system functions as a

dynamic knowledge repository, systematically storing diagnostic insights, case histories, and treatment outcomes. This feature fosters a continuous learning framework, enhancing medical knowledge accumulation and contributing to ongoing research in musculoskeletal medicine.

The successful development and deployment of this expert system underscore the potential of AI-driven solutions in revolutionizing musculoskeletal healthcare. By integrating sophisticated AI algorithms, clinical knowledge modeling, and decision-support functionalities, this system not only enhances diagnostic accuracy but also empowers clinicians with actionable insights for improved patient care. The system's capacity to deliver immediate, data-supported diagnostic suggestions makes it a key advancement in medical diagnostics, opening doors for future developments in intelligent healthcare solutions. In conclusion, the AI-powered expert system for musculoskeletal disease diagnosis represents a cutting-edge breakthrough in clinical decision support. Its high accuracy, usability, and scalability position it as an essential tool for modern healthcare, with the potential to enhance musculoskeletal disease management on a global scale. By offering an intelligent, adaptive, and evidence-based diagnostic framework, this system stands to revolutionize musculoskeletal medicine, significantly improving diagnostic outcomes, patient care, and medical knowledge dissemination.

6. Future Work

While the current system has demonstrated remarkable efficacy in musculoskeletal disease diagnosis, several key areas warrant further exploration to enhance its robustness, adaptability, and integration into broader healthcare infrastructures. Future research should prioritize:

1. **Enhanced Security and User Authentication:** Strengthening data privacy protocols through advanced encryption techniques and multi-factor authentication mechanisms to ensure compliance with healthcare regulations and safeguard patient confidentiality.
2. **Scalability and Interoperability:** Expanding the system’s capabilities to support larger datasets and ensuring seamless integration with electronic health records (EHRs), hospital information systems (HIS), and telemedicine platforms for a more comprehensive diagnostic ecosystem.
3. **Refinement of Diagnostic Algorithms:** Incorporating deep learning-based feature extraction techniques to improve the precision of diagnostic recommendations, particularly in complex and overlapping musculoskeletal disorders. Additionally, adaptive learning mechanisms should be implemented to continuously refine the system’s knowledge base.
4. **Integration of Wearable and IoT-Enabled Biosensors:** Enhancing real-time monitoring and predictive diagnostics by interfacing with wearable devices that capture musculoskeletal health parameters, such as joint mobility, muscle strength, and inflammation markers.
5. **Advanced Reporting and Decision Support Features:** Developing personalized diagnostic reports with AI-driven insights, predictive analytics, and treatment recommendations to assist healthcare providers in making informed clinical decisions.
6. **Explainable AI (XAI) Implementation:** Improving the transparency and interpretability of the system’s decision-making process through explainable AI techniques, ensuring that clinicians and patients can understand the rationale behind diagnostic conclusions.
7. **Validation in Diverse Clinical Settings:** Conducting large-scale clinical trials across multiple healthcare institutions to validate the system’s performance in real-world diagnostic environments, ensuring its applicability across diverse patient populations.

By addressing these advancements, future iterations of AI-powered expert systems can further optimize diagnostic precision, enhance physician trust in AI-driven decision support, and contribute to the evolution of intelligent healthcare systems. These enhancements will not only improve patient outcomes but also establish AI-driven diagnostics as a cornerstone of musculoskeletal disease management in modern medicine.

Abbreviations:

AI - Artificial Intelligence

CNN. - Convolutional Neural Network

EHRs. - Electronic Health Record(s)

GAN - Generative Adversarial Network

GDPR - General Data Protection Regulation

HIS. - Hospital Information Systems

HIPAA - Health Insurance Portability and Accountability Act

MSDs. - Musculoskeletal Disease(s)

NHS. - National Health Service

NPV - Negative Predictive Value(s)

RNN. - Recurrent Neural Network

WHO - World Health Organization

XAI. - Expandable Artificial Intelligence

YLDs - Years Lived with Disabilitie(s)

AUTHORS' CONTRIBUTION:

1. **Sunny Kalu Egereonu:** Performed Basic Computation, Generated the ANN, MATLAB graphs, Writing, Review and Editing.
2. **Obi Chukwuemeka Nwokonkwo:** Supervision, Conceptualization, Prepared Manuscript, Methodology, Writing, Review, Editing, Generated the ANN and MATLAB graphs
3. **Emmanuel Chukwudi Amadi:** Data Analysis, Prepared Manuscript, Supervision, Writing Review, Editing, Generated the ANN and MATLAB graphs.
4. **Anthony Ifeanyi Otuonye :** Supervision, Data Curation, Performed Basic Computations, Generated the ANN and MATLAB graphs.
5. **Ubaezue Ugochukwu Egereonu**: Validation, Writing, Review, Editing and Data Curation.
6. **Ijeoma Blessing Omenihu:** Writing, Review, Editing and Data Curation.

Conflict of Interest: None of the authors declared any conflict of interest.

Acknowledgements: Funding: None.

Ethics Statement: The author(s) ensured that all procedures were performed in compliance with relevant laws and institutional guidelines and have been approved by the appropriate institutional committee(s) and author(s).

References

1. Alruwaili, S. H., Thirunavukkarasu, A., Alanazi, R. M., Alsharari, A. Y., Alruwaili, D. K., Alenzi, H. A., Alruwaili, A. N., & Alruwaili, G. Q. (2023). Prevalence, patterns, and associated factors for musculoskeletal disorders among the healthcare workers of northern Saudi Arabia: A multicenter cross-sectional study. *Journal of Pain Research, 16*, 3735–3746. https://doi.org/10.2147/JPR.S415919
2. Espirito Santo, C. M., Santos, V. S., Kamper, S. J., Williams, C. M., Miyamoto, G. C., & Yamato, T. P. (2024). Overview of the economic burden of musculoskeletal pain in children and adolescents: A systematic review with meta-analysis. *Pain, 165*(2), 296–323. https://doi.org/10.1097/j.pain.0000000000003037
3. World Health Organization. (July 14, 2020). Musculoskeletal conditions. Available from https://www.who.int/news-room/fact-sheets/detail/musculoskeletal-conditions
4. Farzandipour M, Nabovati E, Saeedi S, Fakharian E. Fuzzy decision support systems to diagnose musculoskeletal disorders: A systematic literature review. Computer methods and programs in biomedicine. 2018 Sep 1;163:101-9. https://doi.org/10.1016/j.cmpb.2018.06.002
5. Mahajan, D., Gupta, M. K., Mantri, N., Joshi, N. K., Gnanasekar, S., Goel, A. D., Srinivasan, S., Gonade, N. M., Sharma, S. K., Garg, M. K., & Bhardwaj, P. (2023).

Musculoskeletal disorders among doctors and nursing officers: An occupational hazard of overstrained healthcare delivery system in western Rajasthan, India. *BMC Musculoskeletal Disorders, 24*(1), 349. https://doi.org/10.1186/s12891-023-06457-z

6. Amiri FM, Khadivar A. A fuzzy expert system for diagnosis and treatment of musculoskeletal disorders in wrist. Technical Gazette/Tehnički Vjesnik. 2017 Mar 2;24: 147-155. https://hrcak.srce.hr/file/267127
7. Ibrahim MA, Ojo OA, Oluwafisoye PA. Design and Implementation of an Expert System for Medical Diagnosis and Prescription. The Transactions of the Nigerian Association of Mathematical Physics. 2024 Mar 1;20:83-92. https://doi.org/10.60787/tnamp.v20.380
8. Román-Belmonte JM, De la Corte-Rodríguez H, Rodríguez-Damiani BA, Rodríguez-Merchán EC. Artificial intelligence in musculoskeletal conditions. Artificial Intelligence in Medicine and Surgery-An Exploration of Current Trends, Potential Opportunities, and Evolving Threats-Volume 1. 2023 Mar 25. https://doi.org/10.5772/intechopen.110696
9. National Health Service (NHS). (2024). AI tools approved for detecting bone fractures in X-rays. *The Guardian*. Available from https://www.theguardian.com/society/2024/oct/22/nhs-ai-artificial-intelligence-tools-scans-broken-bones-fractures-x-rays
10. Shehzaad Aziz Khan, Dylan Mistry, Alastair Stephens, James Dalrymple. (2025). An artificial intelligence based approach to musculoskeletal acute knee injury triage. *2025 ISAKOS Congress Abstract*. Available from https://www.isakos.com/2025/Abstract/18891
11. Gu Y, Otake Y, Uemura K, Takao M, Soufi M, Hiasa Y, Talbot H, Okada S, Sugano N, Sato Y. MSKdeX: musculoskeletal (MSK) decomposition from an X-ray image for fine-grained estimation of lean muscle mass and muscle volume. International Conference on Medical Image Computing and Computer-Assisted Intervention 2023 Oct 1 (pp. 497-507). Cham: Springer Nature Switzerland. https://link.springer.com/chapter/10.1007/978-3-031-43990-2_47
12. Amirian S, Carlson LA, Gong MF, Lohse I, Weiss KR, Plate JF, Tafti AP. Explainable AI in orthopedics: challenges, opportunities, and prospects. In2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE) 2023 Jul 24 (pp. 1374-1380). IEEE. https://doi.org/10.1109/CSCE60160.2023.00230
13. Teoh YX, Othmani A, Goh SL, Usman J, Lai KW. Deciphering knee osteoarthritis diagnostic features with explainable artificial intelligence: A systematic review. IEEE Access. 2024 Aug 5. https://doi.org/10.1109/ACCESS.2024.3439096
14. Egereonu SK, Ekedebe N, Otuonye AI, Etus C, Amadi EC, Egereonu UU. Development of an Expert System for Diagnosing Musculoskeletal Disease. International Journal of Intelligent Information Systems. 2024 Sep;13(4):78-93. https://doi.org/10.11648/j.ijiis.20241304.12