

Hawk Inspired Optimization For Enhanced Federated Learning

A Project Report submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR.

In Partial Fulfillment of the Requirements for the Award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING(DATA SCIENCE)
BY

Bogireddy Vanaja	20121A3208
Dinnipati Chandana	20121A3217
Nama Yashasree	20121A3240
Nannam Sachin Vardhan	20121A3243

Under the Guidance of

Ms.K.Sandhya
Assistant Professor
Dept of CSE, SVEC



Department of Computer Science and Engineering
SREE VIDYANIKETHAN ENGINEERING COLLEGE
(Affiliated to JNTUA, Anantapuramu)
Sree Sainath Nagar, Tirupathi – 517 102
2020-2024



SREE VIDYANIKETHAN ENGINEERING COLLEGE

(Affiliated to Jawaharlal Nehru Technological University Anantapur)
Sree Sainath Nagar, A. Rangampet, Tirupati – 517 102, Chittoor Dist., A.P.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project Work entitled
Hawk Inspired Optimization For Enhanced Federated Learning

is the bonafide work done by

Bogireddy Vanaja	20121A3208
Dinnipati Chandana	20121A3217
Nama Yashasree	20121A3240
Nannam Sachin Vardan	20121A3243

In the Department of Computer Science and Engineering, Sree Vidyanikethan Engineering College, A. Rangampet. is affiliated to JNTUA, Anantapuramu in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering during 2020-2024.

This work has been carried out under my guidance and supervision.

The results embodied in this Project report have not been submitted in any University or Organization for the award of any degree or diploma.

Internal Guide

Ms.K.Sandhya

Associate Professor
Dept of CSE
Sree Vidyanikethan Engineering College
Tirupathi

Head

Dr. B. Narendra Kumar Rao

Prof & Head
Dept of CSE
Sree Vidyanikethan Engineering College
Tirupathi

INTERNAL EXAMINER

EXTERNAL EXAMINER

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION AND MISSION

VISION

To become a Centre of Excellence in Computer Science and Engineering by imparting high quality education through teaching, training and research.

MISSION

The Department of Computer Science and Engineering is established to provide undergraduate and graduate education in the field of Computer Science and Engineering to students with diverse background in foundations of software and hardware through a broad curriculum and strongly focused on developing advanced knowledge to become future leaders.

Create knowledge of advanced concepts, innovative technologies and develop research aptitude for contributing to the needs of industry and society.

Develop professional and soft skills for improved knowledge and employability of students.

Encourage students to engage in life-long learning to create awareness of the contemporary developments in computer science and engineering to become outstanding professionals.

Develop attitude for ethical and social responsibilities in professional practice at regional, National and International levels.

Program Educational Objectives (PEO's)

1. Pursuing higher studies in Computer Science and Engineering and related disciplines
2. Employed in reputed Computer and I.T organizations and Government or have established startup companies.
3. Able to demonstrate effective communication, engage in team work, exhibit leadership skills, ethical attitude, and achieve professional advancement through continuing education.

Program Specific Outcomes (PSO's)

1. Demonstrate knowledge in Data structures and Algorithms, Operating Systems, Database Systems, Software Engineering, Programming Languages, Digital systems, Theoretical Computer Science, and Computer Networks. (PO1)
2. Analyze complex engineering problems and identify algorithms for providing solutions (PO2)
3. Provide solutions for complex engineering problems by analysis, interpretation of data, and development of algorithms to meet the desired needs of industry and society. (PO3, PO4)
4. Select and Apply appropriate techniques and tools to complex engineering problems in the domain of computer software and computer based systems (PO5)

Program Outcomes (PO's)

1. Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems (**Engineering knowledge**).
2. Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences (**Problem analysis**).
3. Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations (**Design/development of solutions**).
4. Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions (**Conduct investigations of complex problems**).
5. Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations (**Modern tool usage**).
6. Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice (**The engineer and society**).
7. Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development (Environment and sustainability).

8. Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice (**Ethics**).

9. Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings (**Individual and team work**).

10. Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions (**Communication**).

11. Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments (**Project management and finance**).

12. Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change (**Life-long learning**).

Course Outcomes

CO1. Knowledge on the project topic (PO1)

CO2. Analytical ability exercised in the project work.(PO2)

CO3. Design skills applied on the project topic. (PO3)

CO4. Ability to investigate and solve complex engineering problems faced during the project work. (PO4)

CO5. Ability to apply tools and techniques to complex engineering activities with an understanding of limitations in the project work. (PO5)

CO6. Ability to provide solutions as per societal needs with consideration to health, safety, legal and cultural issues considered in the project work. (PO6)

CO7. Understanding of the impact of the professional engineering solutions in environmental context and need for sustainable development experienced during the project work. (PO7)

CO8. Ability to apply ethics and norms of the engineering practice as applied in the project work.(PO8)

CO9. Ability to function effectively as an individual as experienced during the project work. (PO9)

CO10. Ability to present views cogently and precisely on the project work. (PO10)

CO11. Project management skills as applied in the project work. (PO11)

CO12. Ability to engage in life-long learning as experience during the project work. (PO12)

CO-PO Mapping

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3	PSO 4
C01	3												3			
C02		3												3		
C03			3												3	
C04				3											3	
C05					3											3
C06						3										
C07							3									
C08								3								
C09									3							
C010										3						
C011											3					
C012												3				

(Note: 3-High, 2-Medium, 1-Low)

DECLARATION

We hereby declare that this project report titled "**Hawk-Inspired Optimization for Enhanced Federated Learning**" is a genuine project work carried out by us, in **B.Tech (*Computer Science and Engineering*)** degree course of **Jawaharlal Nehru Technological University Anantapur** and has not been submitted to any other course or University for the award of any degree by us.

Signature of the student

- 1.
- 2.
- 3.
- 4.

ACKNOWLEDGEMENT

We are extremely thankful to our beloved Chairman and founder **Dr. M. Mohan Babu** who took keen interest to provide us the infrastructural facilities for carrying out the project work.

We are highly indebted to **Dr. B. M. Satish**, Principal of Sree Vidyanikethan Engineering College for his valuable support and guidance in all academic matters.

We are very much obliged to **Dr. B. Narendra Kumar Rao**, Professor & Head, Department of CSE, for providing us the guidance and encouragement in completion of this project.

We would like to express our indebtedness to the project coordinator, **Dr. G. Sunitha**, Professor, Department of CSE for her valuable guidance during the course of project work.

We would like to express our deep sense of gratitude **Ms. K. Sandhya**, Assistant Professor, Department of CSE, for the constant support and invaluable guidance provided for the successful completion of the project.

We are also thankful to all the faculty members of CSE Department, who have cooperated in carrying out our project. We would like to thank our parents and friends who have extended their help and encouragement either directly or indirectly in completion of our project work.

ABSTRACT

Hawk-Inspired Optimization for Enhanced Federated Learning which can be used to train machine learning models across decentralized devices without exposing raw data, offers a promising paradigm for healthcare applications. This project explores the Hawk-Inspired Optimization (HIO) within the context of federated learning, with a specific focus on healthcare applications utilizing Internet of Medical Things (IoMT) datasets.

Federated Learning (FL) has emerged as a promising paradigm for training machine learning models across decentralized devices without exchanging raw data. However, the inherent challenges of FL, such as communication overhead and non-IID data distribution, necessitate the development of efficient optimization techniques. This project introduces a novel approach inspired by the hunting behavior of hawks, referred to as Hawk-Inspired Optimization (HIO), to enhance the performance of Federated learning scenarios. To address unique challenges within the healthcare domain with predictive healthcare analytics.

Keywords:- FL-Federated Learning, HIO-Hawk-Inspired Optimization, ML-Machine Learning, IOMT- Internet of Medical Things

TABLE OF CONTENTS

Chapter No	Title	Page No.
	Vision and Mission	I
	Program Educational Objectives	II
	Program Specific Outcomes	III
	Program Outcomes	IV
	Course Outcomes	VI
	CO-PO Mapping	VII
	Declaration	VIII
	Acknowledgments	IX
	Abstract	X
1	INTRODUCTION	1 – 6
	1.1 Introduction	
	1.2 Statement of the problem	
	1.3 Objectives	
	1.4 Scope	
	1.5 Application	
	1.6 Limitation	
2	LITERATURE SURVEY	7 – 12
3	ANALYSIS	13 – 15
	3.1 Existing system	
	3.2 Proposed system	
	3.3 Software and Hardware Requirements	
	3.4 Software requirements specifications	
4	DESIGN	16 - 21
	4.1 Use-case diagram	
	4.2 Sequence diagram	
	4.3 ER diagram	
	4.4 Block Diagram	

5	IMPLEMENTATION	22- 27
	5.1 Dataset used	
	5.2 Dataset information	
	5.3 Data preprocessing	
	5.4 Model Building	
6	EXECUTION PROCEDURE & TESTING	28 – 31
	6.1 Execution Procedure	
	6.2 Testing	
	6.3 Types of Testing	
7	RESULT & PERFORMANCE EVALUATION	32 – 36
	7.1 Evaluation Description	
	7.2 Performance Evaluation	
	7.3 Results	
8	CONCLUSION & FUTURE WORK	37
	APPENDIX	38 – 48
	1.Program Source/code	
	2.List of Figures	
	3.List of Tables	
	4.Screen shorts	
	REFERENCES	49 - 50

CHAPTER-1

INTRODUCTION

Federated learning, a completely distributed machine learning paradigm, has recently become more and more commonly used in order to train models which are deployed over the edge of a distributed network of devices.

1.1 Introduction:

In the field of healthcare applications, Federated Learning (FL) has become an innovative approach as it enables various organizations to jointly train machine learning models without exchanging the raw data. This distributed paradigm ensures a lot of privacy and security by leveraging the “swarm intelligence” of the aggregated datasets. But the issue of efficiency and convergence speed for FL algorithms still remains one of the key challenges, especially when deployed in healthcare environments where data is confidential as well as different.

In this regard, federated learning optimization algorithms inspired by the nature have become very prevalent. Out of these, the Hawk Inspired Optimization (HIO) algorithm that takes inspiration from hawks’ amazing hunting tactics in nature proposes a new and effective method to overcome optimization challenges for FL.

Hawk-Inspired Optimization: Hawk-inspired optimization algorithms draw inspiration from the hunting strategies of hawks in nature. Hawks are known for their ability to efficiently locate and capture prey through a combination of keen observation, strategic planning, and decisive action. These optimization algorithms aim to mimic this behavior by dynamically balancing exploration and exploitation in search spaces, similar to how hawks balance between searching for new prey and exploiting known hunting grounds. Some popular hawk-inspired optimization algorithms include Hawk-Dove Optimization (HDO), which models the interactions

between hawks and doves, and Harris Hawks Optimization (HHO), inspired by the cooperative hunting behavior of Harris's hawks.

Federated Learning: Federated learning is a decentralized machine learning approach where model training occurs locally on distributed data sources, such as mobile devices, edge servers, or IoT devices. Instead of aggregating data into a central server, which raises concerns about data privacy and security, federated learning allows models to be trained directly on the devices where the data resides. This enables users to contribute their data to model training without compromising their privacy, as only model updates, rather than raw data, are shared with a central server or aggregator. Federated learning has gained significant attention in recent years due to its potential applications in scenarios where centralized data collection is impractical or undesirable, such as healthcare, financial services, and edge computing environments.

Healthcare applications need not only high accuracy in model predictions, but also strong privacy protection due to the sensitive nature of medical data. The Hawk-Inspired Optimization aims at improving the communication and cooperation nature of Federated Learning, making it so that medical data is shared over a flock network without compromising individuals privacy. Healthcare is a critical area of research with significant implications for improving the quality of life for individuals and populations. The goal of healthcare is to prevent and treat diseases and injuries while promoting health and well-being. Combining Hawk-Inspired Optimization with Federated Learning: Integrating hawk-inspired optimization techniques into federated learning frameworks offers several potential benefits. Firstly, it can enhance the efficiency and effectiveness of model training by enabling more adaptive exploration and exploitation of the distributed data sources. By leveraging the dynamic search strategies inspired by hawks, algorithms can better navigate the complex and diverse

data landscapes encountered in federated learning settings. Additionally, incorporating hawk-inspired optimization into federated learning can potentially improve convergence speed, robustness to non-stationary data distributions, and scalability to large-scale distributed systems.

Overall, the combination of hawk-inspired optimization and federated learning holds promise for addressing the challenges associated with training machine learning models on decentralized and privacy-sensitive data sources, opening up new opportunities for collaborative learning and knowledge sharing across distributed networks.

This novel method incorporates the ideas of Hawk-Inspired Optimization, a bioinspired algorithm based on hawks' hunting habits into Federated Learning technology in medical environments. Machine learning models can be learned with Federated Learning across scattered devices, protect patient-level data without compromising model collaboration. This research does not only provide solutions to the technical issues that come with Federated Learning implementation in healthcare but also supports medical breakthrough, individualized treatment development and innovations across different public health services while respecting people's privacy rights. The Hawk-Inspired Optimization for Improved Federated Learning serves as an effective solution to moulding the landscape of collaborative and privacy preservation applications in personal healthcare.

The proposed solution for these challenges uses the Hawk-Inspired Optimization approach to boost Federated Learning in healthcare applications. In the field of Federated Learning health care scenarios, inspiration is drawn from attentiveness and collaboration characteristic hawks in nature as an optimization algorithm that improves efficiency and performance.

1.2 Statement of the problem:

In the domain of healthcare applications utilizing Internet of Medical Things (IoMT) datasets, the utilization of Federated Learning (FL) for training machine learning models without exposing raw data shows promise. However, FL faces inherent challenges such as communication overhead and non-IID (non-identically distributed) data across decentralized devices. These challenges demand the development of more efficient optimization techniques. This project aims to investigate and implement Hawk-Inspired Optimization (HIO) within the context of FL specifically tailored for healthcare applications. The goal is to overcome the unique challenges present in healthcare predictive analytics, leveraging HIO to enhance model performance while preserving data privacy and accommodating non-IID data distributions across IoMT devices.

1.3 Objectives:

This study seeks to advance the Federated Learning (FL) in Healthcare utilizing Hawk-Inspired Optimization algorithm by reducing communication load and solving novel challenges specific for medical data. Through the HIO, this study can provide better performance of FL; convergency model, accuracy level and training duration will be higher. Furthermore; there is an amount of adaptation to the features that are inherent dynamically in IoMT datasets it means our point with parameters will have variability within these characteristics Real-world application of this approach in predictive healthcare practice, as with disease prediction for instance or monitoring patients, will show this style has genuine usability. Comparative evaluations between the traditional FL methods and HIO will underline that this latter is very efficient in healthcare scenarios with higher accuracy, convergence speed improvement, communication efficiency ensuring generalization capability of AI algorithm without privacy leaks disclosure.

1.4 Scope:

The field of investigation covers an overview literature review on federated learning and optimization algorithms. Examine the advantages and disadvantages of existing optimization methods for federated learning configurations. The study will involve:

- 1.Introduce Hawk-Inspired Optimization (HIO) in detail, describe its principles, mechanisms and advantages as a nature inspired algorithm. Moreover, emphasize any successful applications within optimization problems.
- 2.Discuss the difficulty of federated learning, including communication load, non-IID data distribution and privacy issues. Discuss the importance of optimization algorithms with respect to addressing these challenges.
- 3.Suggest a strategy for incorporating Hawk-Inspired Optimization into federated learning. Discuss adaptation of HIO to suit federate learning setting consisting of distributed data and real-time communication.
- 4.Propose design experiments to compare the effectiveness of Hawk-Inspired Optimization with other optimization algorithms employed in federated learning. Focus on metrics like convergence rate, communication effectiveness and the accuracy of models.
- 5.Performance Evaluation :Comprehensive testing and evaluation of the developed model in various conditions to determine its accuracy, speed and generalization potential.

1.5 Application:

The Hawk-Inspired Optimization combined with Federated Learning will bring improvements to many fields, including healthcare, finance, telecommunications, and edge computing. The collaborative training in healthcare that is conducted with the help of federated learning protects the privacy of patients and improves the diagnosis of diseases, giving personalized treatment recommendations and discovering drugs. Also, in

finance federated learning helps prevent fraud and risk assessment as it combines data from multiple institutions using Hawk-Inspired Optimization which improves the precision and efficiency of these models. Telecom companies from federated learning for network anomaly detection and QoS improvement, along with edge computing applications for cooperative model training on resource-limited nodes, supported by Hawk-Inspired Optimization for performing optimal inference and decision making. Briefly, this integration allows advancement in innovation and collaboration while data protection and security are being preserved in critical sectors.

1.6 Limitations:

While the integration of Hawk-Oriented Optimization with Federated Learning constitutes the right direction, it is also associated with a number of limitations. Firstly, the number of communications to carry out between the central server and the distributed devices can be greatly increased to be the reason for the snail progress and the consumption of bandwidth. In addition, heterogeneity of data from distributed sources creates problems together with model merging as well as synchronization that, as a result, may impact the quality of the system. Another kind of Security threats such as data poisoning attacks, model inversion attacks, are also common; that is to say, they will affect how private and safe the whole collaborative learning procedure is. Also, computing constraints of edge devices in the context of IoT could hamper deploying sophisticated optimization procedures, thus either reducing the efficiency or the ability for the system to scale. Integrating such limitations into the overall system design should be a critical consideration hence the need to ensure applicability of federated learning for Hawk-Inspired Optimization systems for practical implementation.

CHAPTER-2

LITERATURE SURVEY

The healthcare industry is going through a profound revolution in terms of the deployment of AI. AI was able to make a great change in healthcare because it can provide services of precise diagnosis, treatment and personalized care. Nevertheless, the difficulty of obtaining a variety and sufficient amount of annotated medical data is still one of the issues that complicate the AI model training as desired. Cooperation among few medical institutions to share data while maintaining privacy compliance is another solution which has brought federated learning (FL) to the fore, being an attractive strategy. FL provides a way of making the collaboration of multiple institutions in training models that is safe when it comes to the protection of patient's own personal data. Nevertheless FL shows the problems in realisation like communication effectiveness and convergence speed. In this regard, our study is offering a FL augmentation method, which is based on the particle swarm algorithm.

B. Pandita introduces a novel method that incorporates the Decision Tree and an naive Bayesian neural network in order to classify data effectively. As it has been very successful, problems such as overfitting and sensitivity to redundant features have emerged. **C. Sowmiya and P. Sumithra** , use the Apriori algorithm and Support Vector Machine to purposely synthesise heart disease diagnosis especially focussing on applicability of Models as a classifier .The survey reveals the great number of techniques, Decision Trees; naive Bayesian neural networks SVMs ANN and KNN are used in classification.

On the other hand, IoT integration into FL further promises tremendous prospects in healthcare by delivering patient observations in a timely manner while guaranteeing data privacy. For example, the application of

the meta-heuristic techniques such as better particle swarm optimization could cope with problems of low communication efficiency and slowness of convergence. Using the numerical experiments to elucidate, instead, has revealed that our proposed model, FedImpPSO, performs better in the case of heterogeneous network environments. Overall, the incorporation of FL with stochastic optimization algorithms demonstrate the potential for overcoming difficulties in health AI implementations maintaining the integrity of the privacy and achieving improved performance.

M. Liu and Y.-H. Kim present a classification method based on LSTM, a deep learning scheme targeted at time-series sequences built from ECG signals. Simultaneously, W. Lai and Y. Qiao present a Federated Learning approach to COVID-19 diagnosis in chest CT image using lightweight models that normalize local training via global average vectors of features

H. Ye et al offer a prediction model for the COVID-19 using Fuzzy-KNN techniques influenced by Harris Hawks (HHO). The goal of this model is to classify COVID-19 severity via modification and optimization the Fuzzy KNN's parameters as well as feature subsets with HHO. Paleru Pravallika et al present the Enhanced COVID-19 Detection and Privacy Preserving Using Federated Learning, highlighting patient data privacy while detect Covid using chest Xray images.

To avoid both the fast dispersion of COVID-19 and the perishing of patient privacy, research implements on top of it the use of a decentralized approach referred to as federated learning which allows multiple users to train their own models without any confidential information being shared or disclosed. In the global model, chest X-ray pictures being exploited, was used to develop and evaluate the capability of detecting COVID-19 infection. The peer-to-peer federated learning enables clinics to confidentiality work together in sync, data transmission in a secure way is

allowed. Experimental data draws attention towards the efficiency of the approach which alongside its practicality and relevance is an aid in overcoming Covid-19 challenges.

In today's world, the rate of occurrence of diabetes varies at a swift speed throughout the world. Diabetes is a very serious disease which might result in diabetic retinopathy (DR) if accurately acknowledged and treated in the early stages. Among diabetic retinopathy conditions, the proliferative diabetic retinopathy (PDR) is considered as the main protagonist of blindness by clinicians. DR disease is mainly observed in patients who have suffered from diabetes for a very longer time and so high levels of blood glucose in their bodies.

Nonetheless, the DR is nowadays the first and the most critical danger to adults' sight, and so it demands immediately resolution of its primary cause in order to prevent any possibility of vision deterioration. The process for identifying of DR lesions traditionally using fundus images is quite slow and causes retardation in providing the timely treatment which in turn may lead to the failure of the treatment. Therefore, it becomes necessary to change the systems of screening that will decrease the time of diagnosis and delivering effective treatment using the computer methods. In recent times, researchers have pursued various approaches of AI and ML in DR pattern identification and classification.

Nadzurah Zainal Abidin and Amelia Ritahani Ismail offer a Federated Deep Learning model for automatic detection in the world of diabetic retinopathy. The survey conducts a detailed analysis on federated learning in detecting diabetic retinopathy. Ensuring adequate accessibility and low cost with high accuracy for DR screening of the diabetic populace was considered a challenge in preventing the huge numbers of blindness observed in DR patients. Another side of the proverbial coin is the fact that AI, ML and DL advanced models might ease the hurdles that have been

cited in this review in terms of cost. Therefore, in this research article, a systematic literature review on explored AI, ML and DL based methodologies for DR detection and classification will be conducted. Here, multidimensional studies were carried out on the empirical data of common machine learning methods, fundus image segmentation, DL (Deep Learning) based algorithms, and model-specific key details for the purpose of quality analysis in identifying the accuracy of the models.

FL is an enterprise security measuring model, which tackles the problem of separate data model building based on a single source. The exhausted mode of training and the security-based aggregation procedures can be applied in the wide variety of practical implementations with strong privacy requirements. Yet, there are some difficulties illustrated while FL method is implemented into the actual model training pipeline in practical applications, which influence the success and efficiency of FL systems in real-world applications.

Consequently, more researchers have switched their focus to the barriers of FL, coupled with the effort to come up with new potential research methods for overcoming the challenges. In addition to that, technology wonders of FL are invented to encourage the smart growth of all industries by ensuring the specified tasks. This paper systematically introduces the current researches in FL from five aspects: The basics of FL, data security mechanisms, communication overhead issues and heterogeneity problems are discussed in the second section. In the third section, a literature review is presented on practical applications of FL, as well as state the future research directions which are expected to be discussed in FL.

The cross-silo PPFL (privacy-preserving federated learning) method is a powerful tool for collaboratively training reliable and universal ML models without sharing sensitive data (e.g., healthcare or financial). We introduce APPFLx, a ready-to-use platform that provides as a service privacy-

preserving cross-silo federated learning for an easy and quick adoption of PPFL. APPFLx uses Globus authorization to enable its users to quickly and securely invite trusted collaborators for PPFL, it implements synchronous and asynchronous FL algorithms, it simplifies the process of launching multiple FL experiments, and it provides a tracking and visualization function of the life cycle of FL experiments, which allows domain experts and ML practitioners to easily manage and evaluate cross-silo FL.

It deals with critical attempts that include data quality, adding experts as well as incentives, personalization, and precise models. Management of data quality is of key importance for machine learning models, and that includes "cleaning" the data and enriching it with better information. The ranging of expert knowledge from healthcare professionals is critical to improve the quality of the model performance and trust. Creating efficacious schemes of rewards and motivators should be the primary focus of federated learning developers in order to get valuable data from wearable devices somehow. Personalization in healthcare relating to the customization of healthcare plans and models for patient-specific treatment is undeniably a crucial prerequisite for individual patients' care. Pursuing model accuracy in disease prevention within forthcoming conditions will be among the critical ways to seek.

It focuses on predicting the hospitalization number for cardiac cases with the distributed algorithm through a decentralized optimization framework for sparse SVM classification. cPDS (iterative cluster of primal dual splitting) serves as a mechanism for different data holders to collaborate without disclosing their raw data. Results affirm that cPDS obtains fast convergence and prediction accuracy similar to centralized methods, additionally, with privacy protection of participant's data. Predictive features are distinguished as they improve understanding and help with prevention efforts.

T. S Brisimi et al specify the possible ways of addressing statistical problems, system limitations and privacy issues in biomedical applications focusing on federated learning for healthcare informatics **Shafin Mahmud Jalal et al** propose Horizontal Federated Random Forest for heart disease detection, which improves accuracy with the help of federated central model and decentralized local data clients. This model yields 7.2, to 7.6 accuracy Now we are applying Federated learning optimized with hawk inspired optimization to maximize efficiency and accuracy.

Therefore, in conclusion this survey gives a broad view of different models applied during heart diseases using algorithms from classical to advanced deep learning along with federated learning used for diverse disorders. The surveyed literature clearly shows that there is a constant drive to improve disease detection methodologies, which should optimize models for real-world use in healthcare.

CHAPTER -3

ANALYSIS

System Analysis is the detailed study of the various operations performed by the system and their relationships within and outside the system. The breakdown of something into parts so that the entire system may be understood.

System Analysis is concerned mainly with understanding or being aware of the problem, identifying the relevant variables which are used for decision-making, and analyzing and synthesizing them to obtain optimal solutions. Another view of it is a Problem-Solving technique that breaks down a system into different parts and it studies how those parts will interact to accomplish their purpose.

3.1 Existing System:

In the existing system to predict Heart Disease the following methods are used

- Combination of Decision Tree and naive Bayesian neural network for data categorizing,
- Apriori algorithm and SVM
- Horizontal federated random forest for heart disease detection

In these existing methods, the implementation is a bit complex to build because they fail to maintain data privacy and performance at the same time.

Disadvantages:

- overfitting and sensitivity to redundant features
- Time-consuming.
- Fail to preserve patient data privately.

3.2 Proposed System:

To overcome all disadvantages of the existing systems, we are using machine learning algorithm Which is Federated Learning with Hawk Inspired Optimization.

- By Using these Model It will preserve data privately as below with better performance.
- Proposed federated learning approach with hawk optimization involves a decentralized training process followed by a weighted aggregation method.
- First, each client trains a local machine learning model on their private data and shares only the model weights with the central server.

Next, we apply a hawk optimization technique to update each local model's weights.

- This involves adding an exploration term with random noise to exploring many possible solutions, as well as an exploitation term to use knowledge from the global model to greedily move towards the best solution.
- Finally, the updated local model weights are aggregated at the central server using a weighted average to construct an improved global model.

Advantages:

- Highest accuracy
- Easy to handle
- Data Privacy

3.3 Software & Hardware Requirements:

Hardware Requirements

RAM: 8 GB

Hard disc or SSD: Minimum of 500 MB

Processor: Intel 3rd generation or high or Ryzen with 8 GB Ram

Software Requirements

Operating system: Windows 10 or 11

Software: Python 3.6 or high version

IDE: PyCharm.

Framework: Flask

3.4 Software Requirement Specification:

Functional and non-functional requirements:

Functional Requirements

The system will have the following functional requirements:

- Input Data:

The system will accept input data related to various factors such as age, gender, BMI, lifestyle, family history, and medical history.

- Output:

The system will provide the predicted likelihood of heart disease based on the input data.

Non-Functional Requirements

The system will have the following non-functional requirements:

- Performance:

The system will be able to handle large datasets and provide accurate predictions in a reasonable amount of time. It give accuracy in the range of 87%-90% by training the model with the dataset.

- Scalability:

The system will be scalable to handle increasing amounts of data and users. Because here we use Hawk's Optimization Technique which helps to handle communication overhead while handling the large amount of data.

- Security:

The system will be designed with security by preserving patient raw data. Here it will send only model weights to the central server instead of raw data. So that data will be only accessed in in the local devises which is secured.

- Usability:

The system will be user-friendly and easy to use, with clear instructions and feedback. So we can use it easily just we need to upload the data and get results.

CHAPTER -4

DESIGN

The most creative and challenging part of system development is System Design. The Design of a system can be defined as a process of applying various techniques and principles for the purpose of defining a device, architecture, modules, interfaces and data for the system to satisfy specified requirements. For the creation of a new system, the system design is a solution to "how to" approach.

4.1 Use Case Diagram:

The main purpose of a use case diagram is to show what system functions are performed for which actor. The roles of the actors in the system can be depicted. This system uses a privacy-preserving approach called federated learning for disease detection.

1. Training: First, multiple participants (users) train their own local models on their own data (represented by "Register," "Login").

2. Updating: These models collaborate to update a central model without directly sharing private data (represented by arrows labeled "Send Model Weights to Central Device").

3. Prediction: This improved central model can then analyze new data (represented by "Input Values") to predict disease risk (represented by "Generate Results") while keeping individual patient data confidential ("Logout").

In the below diagram actors and use cases are classified as below

Actors:

- System
- Users

Use Cases:

- Register
- Update Heart Dataset

- Training
- View Results



Fig 4.1 Use-Case Diagram for Predicting Heart Disease

In Fig 4.1 diagram associations that are made between the Actors and use cases are the scenarios that we see in this project.

4.2 Sequence Diagram:

Modeling Language (UML) is a sort of interaction diagram which shows how procedures operate with one another and in what sequence. Sequencediagrams include the following components:

- **Class roles:** These signify Functions that objects can play inside the interaction.
- **Lifelines:** It symbolizes the existence of an item over a time period.
- **Activations:** It signifies the time during which an object is performing the operation
- **Messages:** It symbolizes Communication between items.

This diagram illustrates a disease detection system that leverages federated learning to ensure user privacy. The process begins with users registering and logging in, after which they upload their own heart data sets. Crucially, this data never leaves the user's device, maintaining complete privacy. Each device then trains a local model using its own data, essentially learning patterns from the information. These local models collaborate by sending "learning updates" to a central server, sharing insights without revealing the raw data itself. The central server combines these updates to improve a global model, which becomes more adept at disease detection as it incorporates knowledge from many users. Finally, users can provide new data for analysis without compromising their privacy. The improved global model then analyzes this new information to predict the risk of disease, offering accurate results while safeguarding individual patient data. This system effectively balances the need for accurate disease detection with the critical importance of user privacy.

In Fig 4.2 diagram shows the step wise process of usage of the model for predicting heart disease .

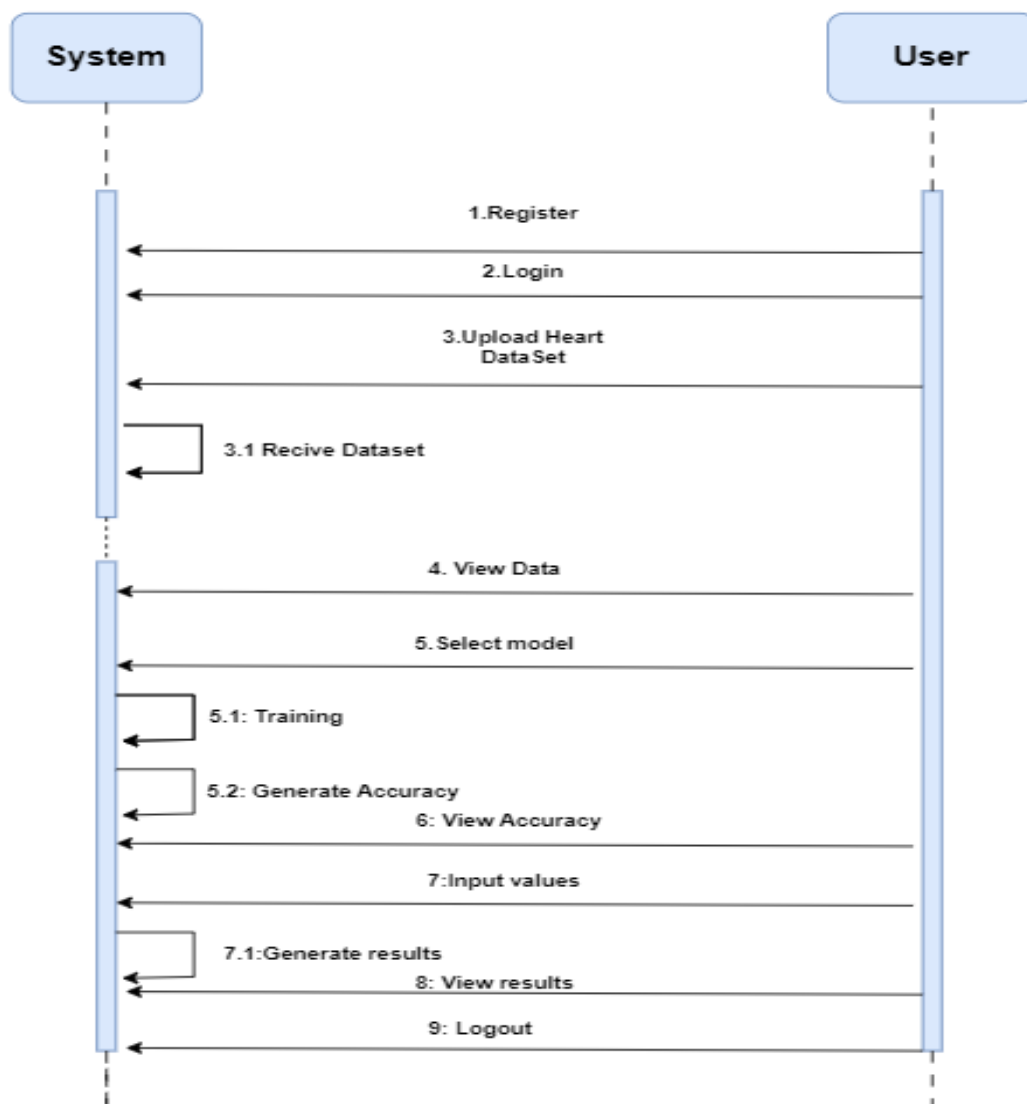


Fig 4.2- Sequence Diagram for Predicting Heart Diseases

4.3 ER Diagram:

An Entity-relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as an Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of the E-R model are the entity set and relationship set.

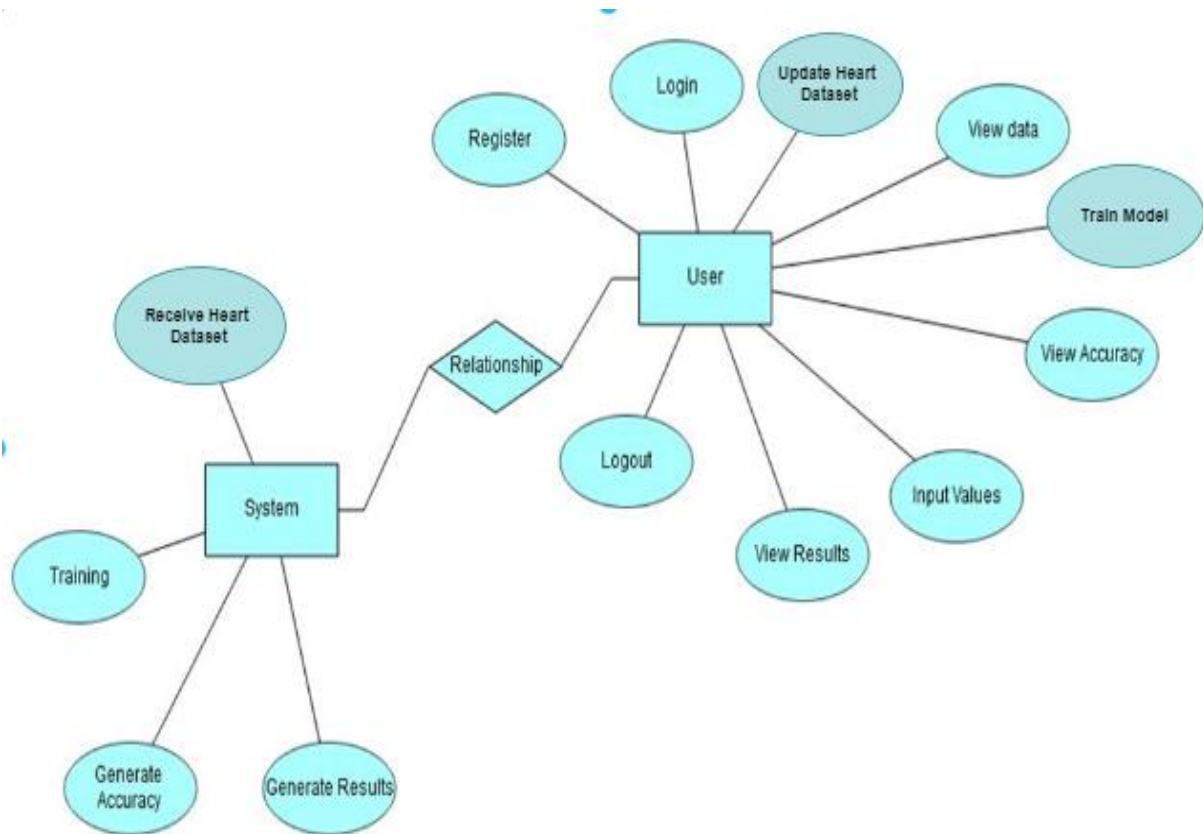


Fig 4.3- ER Diagram for Predicting Heart Disease

This diagram clarifies the division of responsibilities between the user and the system. The system acts as the intelligent core, handling model training and generating accuracy metrics. Users, on the other hand, interact with the system by providing input data and receiving their personalized results. This user-centric design ensures a clear separation of concerns, promoting a smooth and efficient interaction.

This shows depicts a privacy-preserving disease detection system employing federated learning. Users upload their heart data (blood pressure, heart rate, etc.), which remains private on their devices. Each device trains a local model using this data. These models collaborate by sending "learning updates" (not raw data) to a central server, which aggregates them to improve a global model. This enhanced model analyzes new user data (symptoms, readings) to predict disease risk, all while safeguarding individual patient privacy.

4.4 Block Diagram

It below block Diagram describes the basic steps for the identification of chronic kidney illness using a machine-based learning approach .

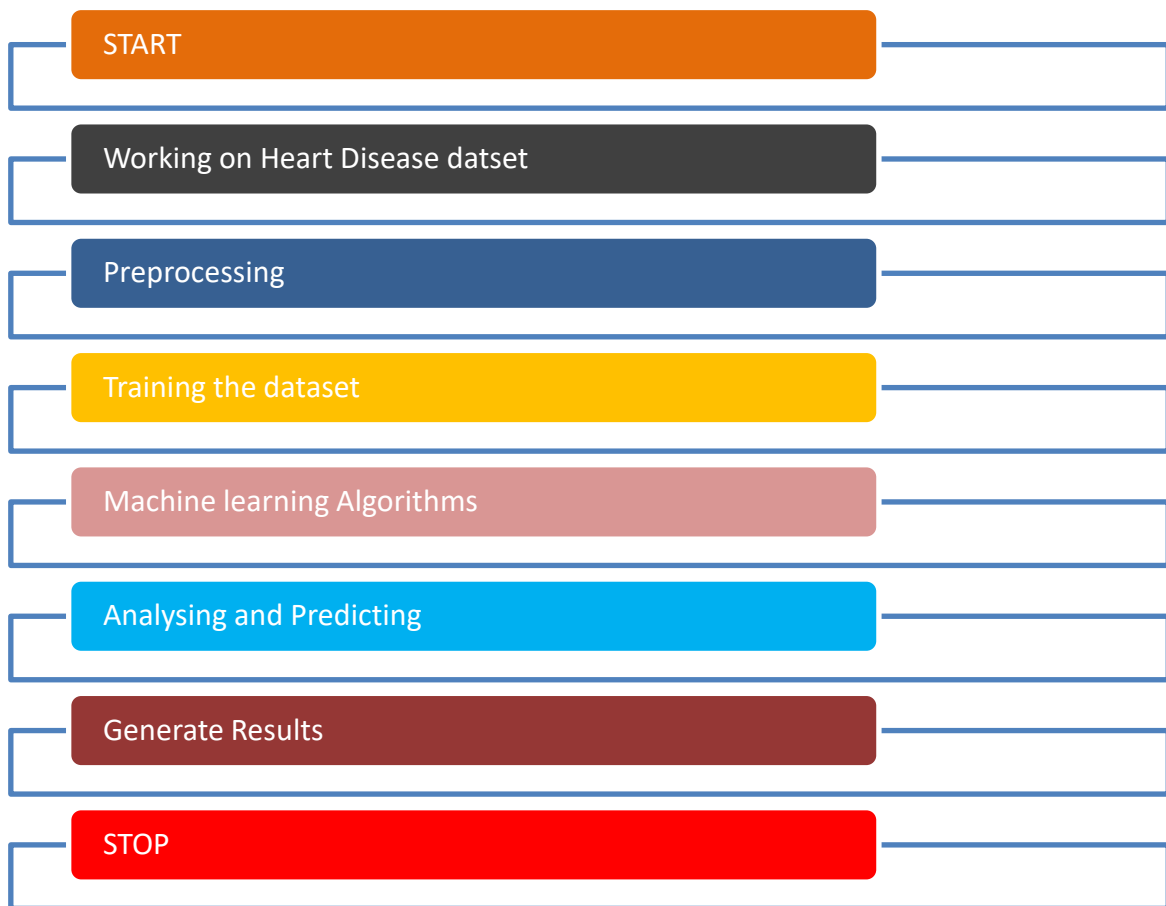


Fig 4.4- Block Diagram for Predicting Heart disease

The diagram depicts a privacy-preserving disease detection model. First, the model is trained on a dataset, learning patterns from the data without ever seeing the raw information itself. This ensures patient privacy is safeguarded. Once trained, the model can then analyze new input data and predict the likelihood of a specific disease, all while keeping the patient's raw data confidential.

CHAPTER-5

IMPLEMENTATION

Enhanced Federated Learning by Hawk-Inspired Optimization has potential benefits which increases accuracy by reducing communication overhead and preserving patient information privately make this methodology efficient. Here we consider the Heart disease detection dataset for testing the model.

In this client-server architecture, each individual in the federation trains his or her model locally using data from that locality. After training these trained models are accumulated in a central server to create a global model through averaging their weights. In this case, each client has global model weights, and it uses them as an input together with its model weights along with a little bit of random noise for updating its local model using optimization function. However, the main purpose of this function is to unite the knowledge received from a global model with those which were obtained by a client itself and provided at least random guidelines for their further exploration. The client retrain the weights that developed the updated processing model. This loop of local training, aggregation, and Hawk-advanced refinement continues for a few copies enlivening the overall model to step by step advance with the entirety of its information decentralized on the customers.

5.1 Dataset Used:

We utilize the Heart disease dataset for implementing the machine-learning model to predict heart disease. This dataset is taken from the Online source which Kaggle. There are 270 occurrences in this dataset with 14 characteristics, including 1 objective attribute. The best property has 2 classes labeled to indicate presence or absence of Heart disease. The table includes descriptions for all 14 qualities.

Characteristics of Dataset: Multivariate

Total Number of Instances: 270

Attribute Characteristics: Real

Total Attributes: 15

Missing Values- No

5.2 Data Set Information:

We use the following representation to collect the dataset: ID, AGE, BP, Chest pain type, Cholesterol, FBS over 120, EKG results, Max HR, Exercise angina, ST depression, Slope of ST, Number of vessels fluorid, Thallium, Heart Disease.

Table 5.1 – Preview of Heart disease dataset

id	age	bp	Chest pain type	Cholesterol	FBS over 120	EKG results	Max HR	Heart disease
0	70	130	4	322	0	2	109	Presence
1	67	115	3	212	0	2	160	Absence
2	57	124	2	241	0	0	141	Presence
3	64	128	4	263	0	0	105	Absence

Target variable: The target variable is "presence or absence of heart disease".

Features: The dataset includes a variety of features that may be relevant to heart disease prediction, including:

- Demographics: Age and sex
- Medical history: Chest pain type, blood pressure, cholesterol, fasting blood sugar, EKG results, maximum heart rate, exercise angina, ST depression, slope of ST segment, and number of vessels.
- Imaging/test results: Thallium stress test results

5.3 Preprocessing:

Data preprocessing is a technique that may be used to transform unclear information into a tidy dataset. Each method for a machine learning classifier requires training as a fundamental step. This method completes tasks including managing unclear values, downscaling the data, creating binary data from it, and standardizing the data frame. When upgrading the

dataset, a stronger tendency is utilized when the collection of characteristics had scales that varied. The value has been transformed into 0 and 1 using the binary transformation. Every attribute's value is regarded as either 1 for values above the threshold or 0 for values below the threshold. Last column indicates presence or absence of the heart disease in Boolean format. This dataset is used for supervised learning of the model.

Table 5.2 - Familiarizing with data

#	Column	Non-Null Count	Dtype
0	id	270 non-null	int64
1	age	270 non-null	int64
2	sex	270 non-null	int64
3	cpt	270 non-null	int64
4	chl	270 non-null	int64
5	FBS	270 non-null	int64
6	ECG	270 non-null	int64
7	MAX HR	270 non-null	int64
8	EA	270 non-null	int64
9	STD	270 non-null	float64
10	Slop of ST	270 non-null	int64
11	NVF	270 non-null	int64
12	Thallium	270 non-null	int64
13	Heart Disease	270 non-null	int64

5.4 Model Building:

First we defines the architecture of an individual neural network model for the heart disease prediction task .It is based on the TensorFlow Keras module's Sequential API. This architecture includes a single Flatten layer that flattens the input data and then a Dense layer with 128 neurons and activation function Rectified Linear Unit. The last layer is a second Dense one with 10 neurons and SoftMax activation, appropriate for multi-class classification because it yields the probability distributions over the possible classes.

Then we perform aggregation where it accepts a list of client models as input which has been trained using the local data independently. The

operation combines the models by averaging the layer weights across all client-models. This weighted averaging is carried out to update the parameters of the global model.

Average weights are calculated as below:-

Equation:

$$average_weights[i] = (1/N) \sum_j (1/N) weights[j][i] \quad (1)$$

where:

- N is no. of client models,
- $weights[j][i]$ is the weights of layer 'i' in the 'j'th client model,
- $average_weights[i]$ represent to the averaged weights for layer 'i' in the global model.

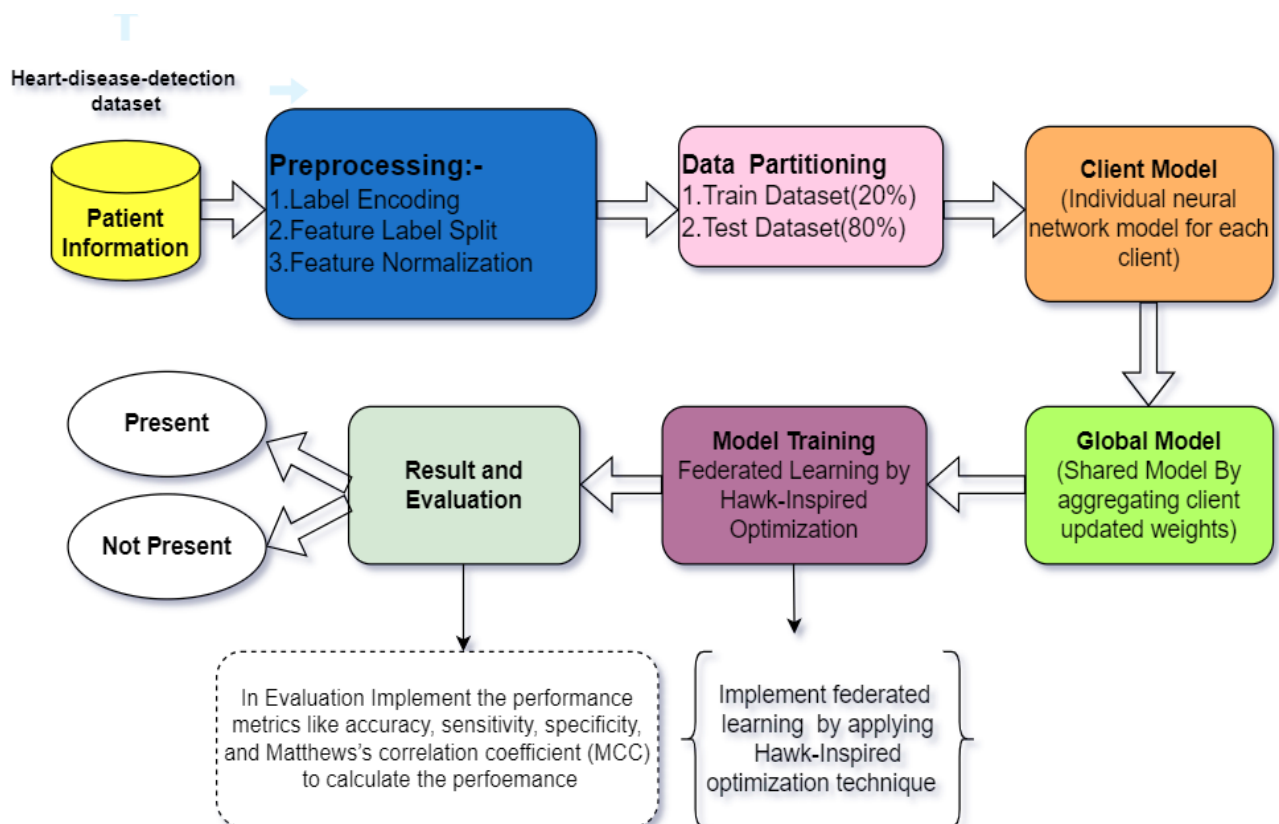


Fig-5.1 Architecture of Heart disease detection using ML

Fig-5.1 illustrates a privacy-preserving heart disease detection system using federated learning. Patients register and upload their heart data (blood pressure, heart rate, etc.), which remains private on their devices.

Each device trains a local model on this data, then collaborates by sending "learning updates" (not raw data) to a central server. This server aggregates these updates to improve a global model, which analyzes new patient data (symptoms, readings) to predict disease risk while safeguarding individual patient privacy.

Here we implement federated learning along with Hawk-Inspired optimization. This Model applies a combination of many terms for exploration and exploitation to the current weights for each layer in the given model. The exploration term injects a stochastic disturbance to foster the exploration, while the exploitation term reflects the estimated difference gap between local and global model weights enforcing teamwork and coordination. A training loop for federated learning cycle involves epoch and model weight updates using the Hawk optimization. For each client, the final training accuracies are being saved.

Lastly, the test set is commonly used to evaluate the global model and then print its accuracy. The code also contains a graphical view of the training accuracy against epochs for each client, which illustrates that federated learning is very cooperative. Federated Learning environment that involves many clients who contribute to the training of the global model via collaborative weight updates for enabling privacy-enhanced machine learning across distributed data repositories.

In the model we calculate Exploration, Exploitation to calculate updated weights for that we use formulas :

Exploration Equation :

$$\text{exploration_term} = a \times \text{epsilon} \times (\text{current_weights} - \text{global_weights})$$

where:

- a is exploration parameter
- epsilon is the random noise

Exploitation Equation:

$$\text{Exploitation_term} = \beta \times (\text{current_weights} - \text{global_weights})$$

where:

- β is Exploitation parameter

Overall Update:

$updated_weights = current_weights - learning_rate \times (exploration_term + exploitation_term)$

In the federated learning, this updating process is applied to each layer of the neural network at all clients where a local model update for each client takes place using Hawk optimization algorithm and contributes towards collective training of global models.

CHAPTER-6

EXECUTION PROCEDURE AND TESTING

The execution procedure for building a heart disease prediction model using Hawk-inspired optimization (HIO) combined with enhanced federated learning involves data collection and preprocessing, feature selection, splitting data into training and testing sets, training machine learning models with HIO, evaluating model performance, selecting the best model, and deploying it for predictions. Testing, an integral part of model development, ensures the model's accuracy and reliability by evaluating its performance on unseen data using metrics such as accuracy, precision, recall, and F1-score. Common types of testing include unit testing to verify individual components, integration testing to ensure seamless functionality across all phases, performance testing to assess the model's efficiency under various conditions, user acceptance testing to gauge its suitability for implementation, cross-validation testing to estimate its generalization ability, and robustness testing to evaluate its resilience to diverse inputs, enhancing reliability in real-world scenarios.

6.1 Execution Procedure:

The execution procedure for building a predictive model for heart disease prediction using Hawk-inspired optimization (HIO) combined with enhanced federated learning can be broken down into the following steps:

❖ Data Collection and Preprocessing:

Collect a dataset including features of age, sex, cholesterol and blood pressure along with other medical history which may predict heart disease. Pre-process the data by getting rid of outdated records and inconsistencies, addressing for missing values, and normalizing features

❖ Feature Selection:

Pick the set of features most closely related to the features of heart disease detecting. With tools such as correlation analysis, feature importance ranking and others, domain knowledge can be applied to confine the search. Pick the features that are proper in terms of federated learning and would be able to be transmitted securely among the different distributed nodes.

❖ Split Data into Training and Testing Sets:

Split up the dataset into training and evaluate the subsets correctly considering the situation of federated learning is highly distributed. Each node involved may have its own information locality of a sub section of the data, for the sake of privacy.

❖ Train Machine Learning Models with Hawk-Inspired Optimization:

Train Machine Learning Models with Hawk-Inspired Optimization: Facilitate hawk-inspired optimization (HIO) algorithms application that show the same collaborative conduct of hawks in nature for federated learning process optimization in a distributed environment. Develop a medical data lake based on implementing federated learning algorithms with HIO built in to train machine learning algorithms while keeping data privacy and security.

❖ Evaluate Model Performance:

Hopefully, consider the adequacy of the train models by the metrics that are suitable for the heart disease diagnosis, for example, accuracy, the precision, recall, F1-score. Guarantee that the rating procedure is designed with a decentralized learning setup at the centre of it, and collect performance metrics from all the units involved.

❖ Select the Best Model:

Compare the performance of each model and select the one with the highest performance.

❖ Deploy the Model:

Once the best model has been selected, it can be deployed and used to make predictions on new data.

Hawk-inspired optimization in federated learning can help you improve the efficiency and effectiveness of the heart disease prediction model in the entity of distributed health while maintaining privacy and security matters in the healthcare community.

6.2 Testing:

Testing is an important part of model development and involves evaluating the performance of the trained models on a previously unseen dataset. This is done to ensure that the models have not to overfit the training data and can generalize well to new data. The testing dataset is used to evaluate the performance of the models by comparing the predicted values to the actual values. Metrics such as accuracy, precision, recall, and F1-score can be used to evaluate the performance of the models. The testing dataset should be representative of the population that the model will be used on, and should not be used for training the model. It is important to repeat the testing process several times using different subsets of the data to ensure that the results are consistent and reliable.

Repeated testing with different subsets of the data helps ensure the Consistency and reliability of results, allowing for the detection and removal of errors from the predictive models, thus enhancing their accuracy and reliability in real-world applications.

Testing is the process of executing a program with the aim of finding errors. To make our software perform well it should be error-free. If testing is done successfully, it will remove all the errors from the software. It is the process of ensuring that the software meets the requirements and functions successfully on the user front.

6.3 Types of Testing

Here are the common types of testing that can be conducted for the predictive model developed using Hawk-inspired Optimization (HIO) combined with Enhanced Federated Learning for heart disease prediction :

- **Unit Testing:** Unit testing refers to the testing of each component of the predictive model in order to ensure they are properly working separately. This involves trying questions that verify the proper doing and functionality of such algorithms as random forest, SVM, KNN, and neural networks , to make sure they were implemented and are working.
- **Integration Testing:** Integration testing ensures that all the predictive model details work in a single unit. In that regard, there should be a testing of the integration between preprocessing, the feature selection and the model training phases in order to make sure that they work together well, which will bring accuracy.
- **Performance Testing:** Performance testing in this case basically draws a performance chart of the model depending different conditions. The model should, however, be tested as some tasks will require a model to be implement in different scenarios which include testing the model on many datasets or measuring its performance under high traffic or limited computational resources. This has been implemented using some metrics which includes confusion matrix, Accuracy were we are getting an average of 87%.
- **User Acceptance Testing:** User acceptance testing is important to make sure that predictive algorithms are suitable to be implemented. This implies testing the model's efficiency in predicting heart disease cases and ensuring that its output improves the model's accuracy, reliability, and the users' judgment. This done by giving an input after training the model and predicting accurate results.

CHAPTER-7

RESULT AND PERFORMANCE EVALUATION

7.1 Evaluation Description:

CONFUSION MATRIX DESCRIPTION:

True Positive (TP) refers to an output that is positive in the sense that the result is accurately classified.

True Negative (TN) refers to negative output for the result to be correctly classified.

False Positive (FP) refers to an outcome that is positive but not as expected.

False Negative (FN) refers to output that is negative so that the projected result is misclassified.

Accuracy Obtained:

The accuracy of the classification shows the likelihood that forecasts will come true. To calculate it, utilize the confusion matrix.

$$\text{Accuracy} = \frac{TN+TP}{TN+TP+FN+FP} * 100$$

Recall:

Recall plays a significant role in the evaluation of the model's performance. It is the percentage of linked instances about all retrieved instances.

$$\text{Recall} = \frac{TP}{TP+FN} * 100$$

Precision:

The model performance evaluation matrix includes precision as a key factor. Instances that are connected as a percentage of all retrieved instances make up this percentage. It has a projected value that is favorable.

$$\text{Precision} = \frac{TP}{TP+FP} * 100$$

F-Measure:

It also goes by the name F Score. To gauge test accuracy, the F-measure is calculated.

$$\text{F-Measure} = 2 * \frac{\text{Precision} * \text{recall}}{\text{Precision} + \text{recall}}$$

7.2 Performance Evaluation:

To evaluate the performance of machine learning algorithms for predicting heart disease, several metrics can be used, such as accuracy, precision, recall and F1-score.

Here's a brief overview of the performance evaluation results for the mentioned algorithms:

Support Vector Machine (SVM):

SVM is a supervised learning method used for solving classification and prediction tasks. It works out by the determination of the highest plane that differentiates amongst different classes in the feature space. SVM when there is well-formed boundaries between classes shows its efficacy even in more-dimensional spaces.

The performance of the SVM algorithm was evaluated using 10-fold cross-validation. The algorithm achieved an accuracy of 88.89%, a precision of 81.00%, a recall of 81.00%, and an F1-score of 81.00.

KNN:

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm that uses a distance-based approach to classify new instances. The KNN algorithm is a method that is deterministic, and is widely used for cases both classification and regression.

It tells a similar feature space which nearest class neighbors belongs to the relative majority class this new data point shares the classification with. The letter "k" in KNN signifies the number of nearest neighbors computed per choice. The performance of the KNN algorithm was evaluated using 10-fold cross-validation. The algorithm achieved an accuracy of 81.48%, a precision of 82.00%, a recall of 81.00%, and an F1-score of 81.00%.

Decision Tree:

Decision Tree is referred as one of the families of supervised machine learning algorithms used for the Assignment of classes and regression behaviors. It forms a decision tree where each node represents a tree node and are split based on a condition, and each leaf node represents a class

label or a real value. Decision trees are interpretable because both numerical and categorical inputs can be accepted.

The performance of the Decision Tree algorithm was evaluated using 10-fold cross-validation. The algorithm achieved an accuracy of 75.93%, a precision of 72.00%, a recall of 76.00%, and an F1-score of 76.00%.

XGBoost:

XGBoost is a machine learning technique which is a combination of algorithms that are used for classification and regression problem solving. In it, a sequence of decision trees is progressively built, whereby the successive ones are made from the correction of errors of the previous ones. In XGBoost, high performance is associated with its scalability and its ability to handle multiple variables in a depending dataset.

The performance of the XGBoost algorithm was evaluated using 10-fold cross-validation. The algorithm achieved an accuracy of 81.48%, a precision of 81.00%, a recall of 81.00%, and an F1-score of 81.00%.

Neural Network (NN):

A neural Network is a deep learning algorithm that uses multiple layers of artificial neurons to learn complex patterns in the data. The algorithm was trained and evaluated using the same dataset as the KNN algorithms.

The performance of the NN algorithm was evaluated using 10-fold cross-validation. The algorithm achieved an accuracy of 75.93%, a precision of 82.00%, a recall of 74.00%, and an F1-score of 74.00%.

Federated Learning:

Federated learning is an individual training method of machine learning where there is no need to share the original data as the nodes involved train a single shared model. Every node trains the model with its own local data and sends only the model update to a central server among which the server aggregates them to produce an updated global model. The federated learning mechanism includes the protection of privacy and data security but also allows the training of the model on distributed computing bases.

The performance of the Federated algorithm was evaluated using 10-fold cross-validation. The algorithm achieved an accuracy of 88.89%, a precision of 81.00%, a recall of 81.00%, and an F1-score of 81.00%.

Table 7.1 - Evaluated Results of Various Algorithms for the Prediction of Heart Disease

S. No	Name of the Algorithm	Accuracy	Precision	Recall	F-Measure
1	SVM	88.89	81.00	81.00	81.00
2	KNN	81.48	82.00	81.00	81.00
3	Decision Tree	75.93	77.00	76.00	76.00
4	XGBoost	81.48	81.00	81.00	81.00
5	Neural Network	75.93	82.00	74.00	74.00
6	Federated Learning	88.89	81.00	81.00	81.00

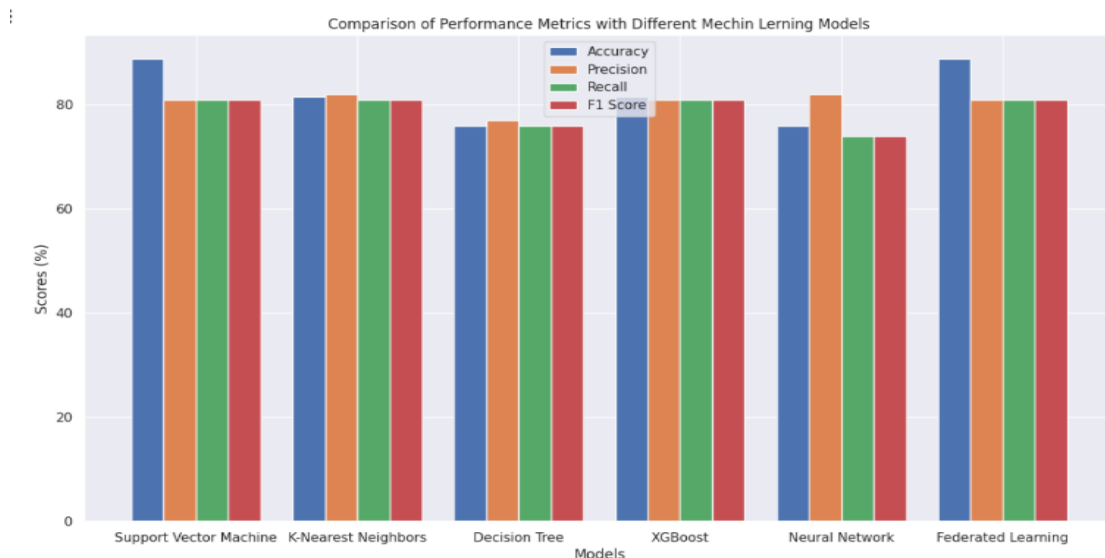


Figure 7.1 - Evaluated Results of various algorithms

The graph shows that Support Vector Machines (SVM) and Federated Learning achieve comparable performance on the given dataset. This

shows that the two models have same capabilities in building effective model for the dataset.

However, Enhanced Federated Learning offers many advantages like avoiding communication overhead, privacy preservice and also non-IID data handling when compared to SVM which uses central repository for data collection.

7.3 Result:

As our project is about the prediction of Heart disease using machine learning we used different algorithms like SVM, KNN, Decision Tree, XGBoost, Neural Networks, Federated Learning by using the Heart disease dataset in that Federated learning and SVM gives us the same and best for predicting the heart disease.

However, Enhanced Federated Learning offers many advantages like avoiding communication overhead, privacy preservice and also non-IID data handling when compared to SVM which uses central repository for data collection.

CHAPETR-8

Conclusion And Future Work

The project of predicting Heart disease using machine learning algorithm which is Federated Learning using Hawk inspired Optimization is a promising area of research. The findings of this project can help healthcare professionals and researchers in improving the treatment of Heart disease. Based on the results of this project, it can be concluded that all the machine learning algorithms used in this study have their own strengths and weaknesses in predicting Heart disease. Among these algorithms, SVM and Federated Learning had the highest accuracy in predicting Heart disease, but the difference is by using federated learning we can achieve data privacy which is not possible by using SVM.

Future work in this area can focus on improving the performance of the machine learning algorithm by using different optimizing and advanced techniques . The dataset used in this study can also be expanded by adding more features and including data from different sources. Moreover, the model developed in this study can be validated on larger and more diverse datasets to evaluate their robustness and generalizability. Finally, the implementation of the model can be integrated into clinical practice to assist healthcare professionals in diagnosing and treating Heart disease.

APPENDIX

Program Source/Code:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import pandas as pd
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Sequential

# Load the Heart Disease dataset
data = pd.read_csv('Heart_Disease_Prediction.csv')
label_encoder = LabelEncoder()
data['Heart Disease'] = label_encoder.fit_transform(data['Heart
Disease'])

# Split the data into features (X) and labels (y)
X = data.drop('Heart Disease', axis=1)
y = data['Heart Disease']

# Normalize the features
scaler = StandardScaler()
X = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
def create_model(input_shape=(13,)): # Adjust the default input shape
    model = Sequential()
    model.add(Flatten(input_shape=input_shape)) # Use the provided
input shape
    model.add(Dense(128, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    return model
```

```
# Assuming that 'client_models' is a list of models trained by clients
client_models = [create_model() for _ in range(10)]
def aggregate_models(models):
    total_layers = len(models[0].layers)
    weights = [[] for _ in range(total_layers)]

    for model in models:
        for i, layer in enumerate(model.layers):
            layer_weights = layer.get_weights()

            for j, weight_array in enumerate(layer_weights):
                if len(weights[i]) <= j:
                    weights[i].append([])
                weights[i][j].append(weight_array)

    average_weights = [
        [np.mean(np.array(weight_list), axis=0) for weight_list in
layer_weights]
        for layer_weights in weights
    ]

    for i, (layer, layer_weights) in enumerate(zip(models[0].layers,
average_weights)):
        layer.set_weights(layer_weights)

    return models[0]
# # Assuming that 'clients_models' is a list of models trained by clients
global_model = aggregate_models(client_models)

def hawk_optimization(current_weights, global_weights, learning_rate,
alpha=0.5, beta=0.5):
```

```
    epsilon = np.random.rand(*current_weights.shape) # Random noise
for exploration
    exploration_term = alpha * epsilon * (current_weights -
global_weights)
    exploitation_term = beta * (current_weights - global_weights)

    updated_weights = current_weights - learning_rate *
(exploration_term + exploitation_term)

    return updated_weights
def train_model_with_hawk_optimization(model, X, y, global_model,
epochs=5, batch_size=32, learning_rate=0.01):
    model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])

    training_accuracies = [] # To store training accuracies for each epoch

    for epoch in range(epochs):
        for layer_index, (layer, global_layer) in
enumerate(zip(model.layers, global_model.layers)):
            current_weights = layer.get_weights()
            global_weights = global_layer.get_weights()

            # Apply Hawk-inspired optimization to update the layer weights
            updated_weights = []

            for cw, gw in zip(current_weights, global_weights):
                if cw is not None and gw is not None:
                    updated_weight = hawk_optimization(cw, gw,
learning_rate)
                    updated_weights.append(updated_weight)
                else:
```

```
        updated_weights.append(None)

    # Set the updated weights to the layer
    layer.set_weights(updated_weights)

    # Fit the model with the updated weights
    history = model.fit(X, y, epochs=1, batch_size=batch_size,
verbose=0)

    # Track training accuracy for each epoch
    training_accuracies.append(history.history['accuracy'][0])

    return training_accuracies
def evaluate_model(model, X, y):
    loss, accuracy = model.evaluate(X, y)
    return loss, accuracy
# Train models using Hawk-inspired optimization
training_accuracies = []
for model in client_models:
    acc = train_model_with_hawk_optimization(model, X_train, y_train,
global_model)
    training_accuracies.append(acc)

# Evaluate the federated learning model
loss, accuracy = evaluate_model(global_model, X_test, y_test)
accuracy_fl=round(accuracy*100,2)
print(f"Test accuracy: {accuracy_fl}%")
# Assuming 'global_model' is the trained federated learning model
# 'new_data' is the new input data that you want to predict on

# Load your new data
```

```
new_data = pd.DataFrame({
    'Age': [70],
    'Sex': [1],
    'Chest pain type': [4],
    'BP': [130], 'Cholesterol': [322],
    'FBS over 120': [0],
    'EKG results': [2],
    'Max HR': [109],
    'Exercise angina': [0],
    'ST depression': [2.4],
    'Slope of ST': [2],
    'Number of vessels fluro': [3],
    'Thallium': [3]
})

# Normalize the new data using the same scaler used for training
new_data_scaled = scaler.transform(new_data)

# Predict using the global model
predictions = global_model.predict(new_data_scaled)

# Assuming the predictions are probabilities for each class
# You can convert these probabilities to class labels if needed
predicted_classes = np.argmax(predictions, axis=1)

# Print the predicted classes
for i in predicted_classes:
    if(i==1):
        print("Heart disease present")
    else:
        print("Heart disease absent")
```


LIST OF FIGURES

Fig No.	Title	Page No.
4.1	Use-case diagram for predicting Heart Disease	17
4.2	Sequence diagram for predicting Heart Disease	19
4.3	ER diagram for predicting Heart Disease	20
4.4	Block diagram for predicting Heart Disease	21
5.1	The architecture of Heart Disease using ML	25
7.1	Evaluated results of various algorithms	35

LIST OF TABLES

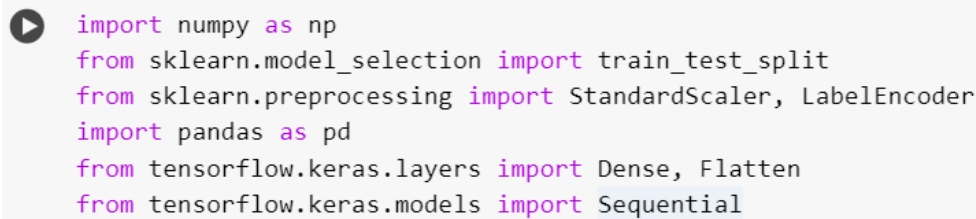
Table No.	Title	Page No.
5.1	Preview of Heart Disease Dataset	23
5.2	Familiarizing with data	24
7.1	Evaluated results of various algorithms for predicting Heart Disease	35

Screen Shorts

Our project is about the prediction of Heart disease using machine learning algorithm for this project execution we used google Collab.

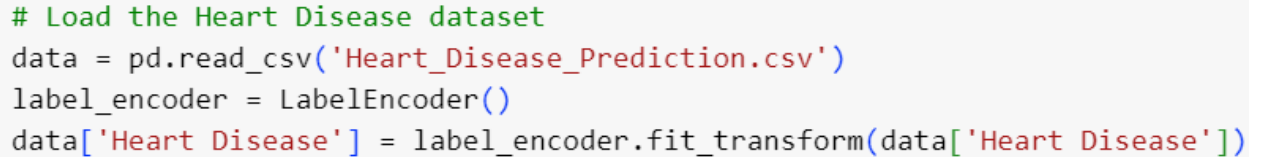
Here I will display the screen short of Google collab:

1.Import Data



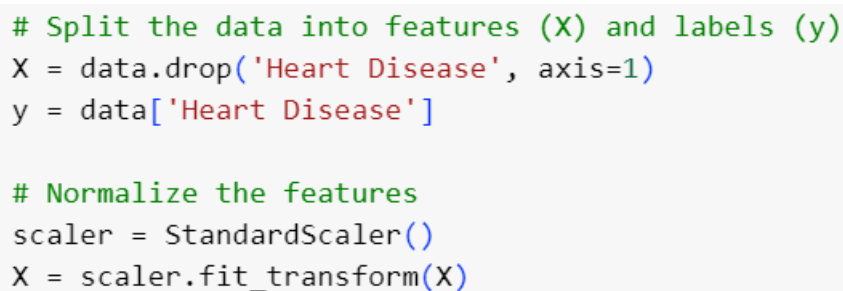
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import pandas as pd
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Sequential
```

Fig A.4.1 Import the required libraries



```
# Load the Heart Disease dataset
data = pd.read_csv('Heart_Disease_Prediction.csv')
label_encoder = LabelEncoder()
data['Heart Disease'] = label_encoder.fit_transform(data['Heart Disease'])
```

Fig A.4.2 Uploading the dataset



```
# Split the data into features (X) and labels (y)
X = data.drop('Heart Disease', axis=1)
y = data['Heart Disease']

# Normalize the features
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

Fig A.4.3 Dataset Preprocessing

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	\
0	70	1	4	130	322	0	2	
1	67	0	3	115	564	0	2	
2	57	1	2	124	261	0	0	
3	64	1	4	128	263	0	0	
4	74	0	2	120	269	0	2	

	Max HR	Exercise angina	ST depression	Slope of ST	\
0	109	0	2.4	2	
1	160	0	1.6	2	
2	141	0	0.3	1	
3	105	1	0.2	2	
4	121	1	0.2	1	

	Number of vessels fluro	Thallium	Heart Disease
0	3	3	1
1	0	7	0
2	0	7	1
3	1	7	0
4	1	3	0

Fig A.4.4 Dataset after preprocessing

```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Age                                    270 non-null    int64
1   Sex                                    270 non-null    int64
2   Chest pain type                       270 non-null    int64
3   BP                                     270 non-null    int64
4   Cholesterol                           270 non-null    int64
5   FBS over 120                          270 non-null    int64
6   EKG results                           270 non-null    int64
7   Max HR                                270 non-null    int64
8   Exercise angina                       270 non-null    int64
9   ST depression                         270 non-null    float64
10  Slope of ST                           270 non-null    int64
11  Number of vessels fluro               270 non-null    int64
12  Thallium                              270 non-null    int64
13  Heart Disease                         270 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 29.7 KB
None
```

Fig A.4.5 Info of dataset

2.Split Data for training and testing

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig A.4.6 Splitting of dataset

3.Create Client Models

```
[ ] def create_model(input_shape=(13,)): # Adjust the default input shape
    model = Sequential()
    model.add(Flatten(input_shape=input_shape)) # Use the provided input shape
    model.add(Dense(128, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    return model

# Assuming that 'client_models' is a list of models trained by clients
client_models = [create_model() for _ in range(10)]
```

Fig A.4.7 create client model

4.Aggregate Client models and Create Global Model

```
def aggregate_models(models):
    total_layers = len(models[0].layers)
    weights = [[] for _ in range(total_layers)]

    for model in models:
        for i, layer in enumerate(model.layers):
            layer_weights = layer.get_weights()

            for j, weight_array in enumerate(layer_weights):
                if len(weights[i]) <= j:
                    weights[i].append([])
                weights[i][j].append(weight_array)

    average_weights = [
        np.mean(np.array(weight_list), axis=0) for weight_list in layer_weights
        for layer_weights in weights
    ]

    for i, (layer, layer_weights) in enumerate(zip(models[0].layers, average_weights)):
        layer.set_weights(layer_weights)

    return models[0]

# Assuming that 'clients_models' is a list of models trained by clients
global_model = aggregate_models(client_models)
```

Fig A.4.8 Aggregate client model

5.Hawk Function

```
[ ] def hawk_optimization(current_weights, global_weights, learning_rate, alpha=0.5, beta=0.5):
    epsilon = np.random.rand(*current_weights.shape) # Random noise for exploration
    exploration_term = alpha * epsilon * (current_weights - global_weights)
    exploitation_term = beta * (current_weights - global_weights)

    updated_weights = current_weights - learning_rate * (exploration_term + exploitation_term)

    return updated_weights
```

Fig A.4.9 Hawk Function

6.Train Federated Learning with Hawk Optimization

```
def train_model_with_hawk_optimization(model, X, y, global_model, epochs=5, batch_size=32, learning_rate=0.01):
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

    training_accuracies = [] # To store training accuracies for each epoch

    for epoch in range(epochs):
        for layer_index, (layer, global_layer) in enumerate(zip(model.layers, global_model.layers)):
            current_weights = layer.get_weights()
            global_weights = global_layer.get_weights()

            # Apply Hawk-inspired optimization to update the layer weights
            updated_weights = []

            for cw, gw in zip(current_weights, global_weights):
                if cw is not None and gw is not None:
                    updated_weight = hawk_optimization(cw, gw, learning_rate)
                    updated_weights.append(updated_weight)
                else:
                    updated_weights.append(None)

            # Set the updated weights to the layer
            layer.set_weights(updated_weights)

            # Fit the model with the updated weights
            history = model.fit(X, y, epochs=1, batch_size=batch_size, verbose=0)

            # Track training accuracy for each epoch
            training_accuracies.append(history.history['accuracy'][0])

    return training_accuracies
```

Fig A.4.10 Training

7.Evaluation

```
[ ] def evaluate_model(model, X, y):
    loss, accuracy = model.evaluate(X, y)
    return loss, accuracy

# Train models using Hawk-inspired optimization
training_accuracies = []
for model in client_models:
    acc = train_model_with_hawk_optimization(model, X_train, y_train, global_model)
    training_accuracies.append(acc)

# Evaluate the federated learning model
loss, accuracy = evaluate_model(global_model, X_test, y_test)
accuracy_fl=round(accuracy*100,2)
print(f"Test accuracy: {accuracy_fl}%")
```

WARNING:tensorflow:5 out of the last 9 calls to <function Model.make_test_function.<locals>.t
2/2 [=====] - 0s 11ms/step - loss: 1.4743 - accuracy: 0.8519
Test accuracy: 85.19%

Fig A.4.11 Evaluation

```
# Assuming 'global_model' is the trained federated learning model
# 'new_data' is the new input data that you want to predict on

# Load your new data
new_data = pd.DataFrame({
    'Age': [70],
    'Sex': [1],
    'Chest pain type': [4],
    'BP': [130], 'Cholesterol': [322],
    'FBS over 120': [0],
    'EKG results': [2],
    'Max HR': [109],
    'Exercise angina': [0],
    'ST depression': [2.4],
    'Slope of ST': [2],
    'Number of vessels fluro': [3],
    'Thallium': [3]
})

# Normalize the new data using the same scaler used for training
new_data_scaled = scaler.transform(new_data)

# Predict using the global model
predictions = global_model.predict(new_data_scaled)

# Assuming the predictions are probabilities for each class
# You can convert these probabilities to class labels if needed
predicted_classes = np.argmax(predictions, axis=1)

# Print the predicted classes
for i in predicted_classes:
    if(i==1):
        print("Heart disease present")
    else:
        print("Heart disease absent")
```

```
1/1 [=====] - 0s 50ms/step
Heart disease present
```

Fig A.4.12 Predicting the Disease for new Data

REFERENCES

- [1] A. Pandita, "Prediction of Heart Disease using Machine Learning Algorithms," International Journal for Research in Applied Science and Engineering Technology, vol. 9, no. VI, pp. 2422–2429, Jun. 2021
- [2] C. Sowmiya and P. Sumithra, "Analytical study of heart disease diagnosis using classification techniques," IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS), 2017
- [3] M. Liu and Y.-H. Kim, "Classification of Heart Diseases Based On ECG Signals Using Long Short-Term Memory," PubMed, Jul. 2018, doi:<https://doi.org/10.1109/embc.2018.8512761>.
- [4] W. Lai and Y. Qiao, "Federated Learning for Detecting COVID-19 in Chest CT Images: A Lightweight Federated Learning Approach," Dec. 2022, doi:<https://doi.org/10.1109/icftic57696.2022.1007516>
- [5] H. Ye et al., "Diagnosing Coronavirus Disease2019 (COVID-19): Efficient Harris HawksInspired Fuzzy K-Nearest Neighbor Prediction Methods," IEEE Access, vol. 9, pp. 17787–17802, 2021, doi:<https://doi.org/10.1109/access.2021.3052835>.
- [6] Paleru Pravallika, Vemireddy Gnanasri, Makineni Gireesh, Avuthu Avinash Reddy, V. Kalpana, and Jasthi Sumitha Chowdary, "EnhancedCOVID-19 Detection and Privacy Preserving Using Federated Learning," Aug. 2023, doi:<https://doi.org/10.1109/icirca57980.2023.1022077>.
- [7] Nadzurah Zainal Abidin and Amelia Ritahani Ismail, "Federated Deep Learning for Automated Detection of Diabetic Retinopathy," Jul. 2022, doi:<https://doi.org/10.1109/icced56140.2022.1001063>.
- [8] W. Y. B. Lim et al., "Federated Learning in Mobile Edge Networks: a Comprehensive Survey," IEEE Communications Surveys & Tutorials, vol. 22, no. 3, pp. 1–1, 2020, doi:<https://doi.org/10.1109/comst.2020.2986024>
- [9] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. Ch. Paschalidis, and W. Shi, "Federated learning of predictive models from federated Electronic Health Records," International Journal of Medical Informatics, vol. 112, pp. 59–67, Apr. 2018, doi:<https://doi.org/10.1016/j.ijmedinf.2018.01.007>.
- [10] Shafin Mahmud Jalal, Md. Rezuwan Hasan, Md. Ashfaquul Haque, and R. Alam, "A Horizontal Federated Random Forest for Heart Disease Detectionfrom Decentralized Local Data," Sep. 2022, doi:<https://doi.org/10.1109/r10-htc54060.2022.9929490>
- [11] "Heart Disease Prediction using Supervised Machine Learning Algorithms," International

Journal of Innovative Technology and Exploring Engineering, vol. 9, no. 4, pp. 148–151, Feb. 2020, doi: <https://doi.org/10.35940/ijitee.d1381.029420>.

[12]R. Goel, “Heart Disease Prediction Using Various Algorithms of Machine Learning,” SSRN Electronic Journal, 2021, doi:<https://doi.org/10.2139/ssrn.3884968>.

[13]A. Agrahary, “Heart Disease Prediction Using Machine Learning Algorithms,” International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp. 137–149, Jul. 2020, doi:<https://doi.org/10.32628/cseit206421>.

[14]F. R. Wani and Er. H. Kaur, “Analytical Study of Diagnosis for Angioplasty and Stents Patients using Improved Classification Technique,” International Journal of Trend in Scientific Research and Development, vol. Volume-3, no. Issue-1, pp. 1170–1173, Dec. 2018

[15]Z. M. Elgamal, N. B. M. Yasin, M. Tubishat, M. Alswaitti, and S. Mirjalili, “An Improved Harris Hawks Optimization Algorithm With Simulated Annealing for Feature Selection in the Medical Field,” IEEE Access, vol. 8, pp. 186638–186652, 2020, doi: <https://doi.org/10.1109/access.2020.3029728>.

[16]Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated Machine Learning,” ACM Transactions on Intelligent Systems and Technology, vol. 10, no. 2, pp. 1–19, Feb. 2019, doi: <https://doi.org/10.1145/3298981>

[17]Q. Dou et al., “Federated deep learning for detecting COVID-19 lung abnormalities in CT: a privacy-preserving multinational validation study,” npj Digital Medicine, vol. 4, no. 1, pp. 1–11, Mar. 2021, doi: <https://doi.org/10.1038/s41746-021-00431-6>

[18]N Jagan Mohan, R. Murugan, T. Goel, and P. Roy, “DRFL: Federated Learning in Diabetic Retinopathy Grading Using Fundus Images,” IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 6, pp. 1789–1801, Jun. 2023, doi: <https://doi.org/10.1109/tpds.2023.3264473>.

[19] P. Hurlen, K. Skifjeld, and E. P. Andersen, “The basic principles of the synapses federated healthcare record.

[20]Z. Li et al., “APPFLx: Providing Privacy- Preserving Cross-Silo Federated Learning as a Service,” Oct. 2023, doi: <https://doi.org/10.1109/e-science58273.2023.1025484>