



**London
South Bank
University**

Smart Farming using Artificial Intelligence and edge cloud computing

Godwin Omoarebu Idoje

Student ID: 3621400

<https://orcid.org/0000-0002-2730-9898>

A THESIS SUBMITTED TO
LONDON SOUTH BANK UNIVERSITY
FOR THE PARTIAL FULFILMENT OF
DEGREE OF THE DOCTOR OF PHILOSOPHY
THE FACULTY OF ENGINEERING

2023

Abstract

This thesis investigates the application of artificial intelligence to smart farming. The *Amaranthus Viridis* crop has been grown within London South Bank University, and different machine learning models have been used to evaluate the crop dataset.

A comparative analysis of the performance of the machine learning models for the datasets from the Nutrient Film Technique (NFT), Aeroponic (AER), Aggregate (AG), and Floating Hydroponic systems from the Department of Agriculture, University of Peloponnese (UP), Kalamata, Greece to predict the Onion Bulb Diameter (OBD). The dataset has been from four different hydroponics systems, namely Aggregate (AG), Aeroponics (AER), Floating and Nutrient Film Technic (NFT). The onion crop has been grown for 92 days after transplant from the nursery.

Artificial intelligence subsets, such as machine learning and deep neural networks, have been used to evaluate the *Amaranthus Viridis* crop. The centralised smart farm network models evaluate the provided dataset while sharing the data with the server during training. The decentralised network for a smart farm has been considered, a scenario where the raw dataset has yet to be shared with the server during the dataset evaluation. Federated learning models allowed the researcher to train the models and make predictions of the dependent variables.

Smart farming involves applying information and communication technology to the traditional farm system. It implies the use of the Internet of Things (IoT), edge and cloud computing, and centralised and decentralised machine learning models for predictions of the farm produce. A survey of the existing smart farms identifies the various challenges experienced by farmers, which include low technological know-how of smart farm techniques, inadequate infrastructure provision, computational power issues with technological devices, poor internet facilities, high latency within the existing internet

connectivity in the farms, insufficient human, technical skills capacity to manage smart farm operations, security of smart farm data during transmission, data reliability, the communication cost of smart farm network.

The floating hydroponic system involves planting the crop on the water surface, and wool rock holds the crop within the pot, allowing its roots to touch the water beneath as it grows.

The Onion Bulb Diameter (OBD) dataset for four different hydroponic systems, namely Floating, AER, AG, and NFT hydroponic systems, has been analysed using the XGBoost, Linear regression, Deep Neural Network (DNN), and Federated Split Learning (FSL) models for their predictive and interpretive abilities.

To automate the predictions of the OBD, machine learning models have been used to predict the OBD for days not considered manually within the crop life cycle. The model helps the farmers determine the harvest even before the commencement of the new planting season based on the previous planting season dataset.

The developed models have been used to analyse the *Amaranthus Viridis* Leaves image dataset comprehensively. The Convolutional Neural Network (CNN) model has been used to determine the percentage of the predicted *Amaranthus* leaves that match the original images from a hydroponic smart farm. The CNN forecasted a higher accuracy than the K-Nearest Neighbour, Support Vector classifier and Decision Tree model.

The crop growth rate was analysed using machine learning algorithms. The XGBoost models produced higher prediction values for the crop growth rate than the Theil-Sen, Decision Tree, Support Vector, Quantile and K-Neighbours regressors.

The dataset is shared with the server during training in a centralised network. The federated learning models train the dataset from the smart farm network without having access to the raw dataset with the server. The federated averaging model has been used

to investigate the prediction of crop types using climatic parameters as the independent variables. The hyper-tuned federated Learning model predicted the crop chickpea and achieved high accuracy.

The federated Learning smart farm network emulation was used to compare the client, server and centralised node performance. The Server model converged faster than the edge node models, classical centralised models, since it uses all the combined weights from the edge nodes to aggregate the combined models before sending them to the edge node for further training of the raw dataset. The results show that the combined decentralised network model produced a higher accuracy than the classical centralised model.

Acknowledgements

This research would not have been possible without Life. I give all thanks and gratitude to the Giver of Life, The Almighty, the creator of Life.

I am very grateful to my supervisors, Prof. Tasos Dagiuklas and Dr Muddesar Iqbal, for all their academic support and inspiration for this research study. Thank you both for believing in me and allowing me to study at the London South Bank University. All your guidance, encouragement, and support have helped me throughout this research study and will continue to guide me for the rest of my academic journey.

Prof. Tasos, Thank you for all the rounds of valuable reviews and patience with all my submissions. All your effort in my life has produced this thesis as one of the many outputs from this research study.

Secondly, I want to thank George Ubakanma for his inspirational support and motivation, the facilities provided for my research and the gentle words from your lips that ignited the fire to carry on during stressful moments. I want to thank the computer science division staff at London South Bank University, who guided me throughout my studies.

For all my colleagues who stood by me when the codes produced wrong results, you were there to always support me in fixing the bugs. Saptarshi Ghosh and Emeka Uwguanyi, thank you for being friends in my time of despair, pain, and stress and as a source of encouragement. To my wife and son, Mrs Oluwasayo Idoje and Samuel Idoje, Thank you for all the emotional support and patience during my research studies. I want to thank my parents, Mr and Mrs Michael Idoje, and my uncle, Mr Ajayi Idoje, for constantly calling to encourage me throughout my studies. I Thank Emmanuel Ozigi, Mrs Dorcas Sasore, Harrison Oloye, and Christina Okobia for the immense financial support during my Ph.D. studies.

TABLE OF CONTENTS

Abstract.....	ii
Acknowledgement	v
Table of contents.....	vi
List of Figures.....	xi
List of Tables.....	xv
List of Acronyms.....	xvii
Chapter 1 Introduction.....	1
1.1 Overview of Smart Farming.....	1
1.2 Motivation.....	5
1.3 Thesis Aim.....	6
1.3.1 Research Questions.....	6
1.4 Contributions.....	7
1.5 Organisation of Thesis.....	8
1.6 Publications.....	9
1.6.1 Published Papers.....	9
1.6.2 Manuscript Under Review.....	9
Chapter 2 Background.....	10
2.1 A Survey of the gaps identified in a smart farm.....	10
2.2 Smart farming.....	11
2.2.1 Crops production.....	11
2.2.2. Livestock production.....	11
2.2.3. Post-harvesting.....	12
2.2.4 Hydroponic Systems.....	13
2.2.4.1 Floating Hydroponic System.....	13
2.2.4.2 Aeroponic Hydroponic System.....	14

2.2.4.3 Nutrient Film Technique (NFT) Hydroponic System.....	14
2.2.4.4 Aggregate Hydroponic System.....	15
2.2.4.5 Amaranthus Viridis Crop Grown in Suitelab Hydroponic smart farm Testbed.	16
2.2.4.6 Convolutional Neural Network.....	22
2.2.4.7 Federated Learning.....	23
2.2.4.7.1 Federated Learning Algorithm.....	29
2.2.4.8 Gaussian Naïve Base (NB) Classifiers.....	33
2.2.4.9 Modelling of Federated Learning Smart Farm Network	33
2.3 Technologies.....	37
2.3.1. Sensors used in smart farms.....	37
2.3.2. Unmanned Aerial Vehicles (UAV) in smart farming.....	38
2.3.3. Internet of Things.....	39
2.3.4. Artificial intelligence.....	40
2.3.5 Fog computing.....	41
2.3.6 Edge computing	43
2.3.7 Multi-access edge computing (MEC) Cloud computing.....	44
2.3.8 Cloud computing.....	46
2.4. How climate change affects smart farming.....	48
2.5 Challenges and issues of smart farming.....	49
2.6 Cloud-based IoT smart farming.....	50
2.7 Application of machine learning to agriculture.....	52
2.8. Discussion.....	54
Chapter 3 Data and Data Preprocessing.....	56
3.1 Hydroponic dataset.	56
3.2 Amaranthus Viridis dataset.....	57

3.3 Federated Learning Classification of multi-labelled Dataset.....	58
3.4 Dataset for Modelling of Federated Learning Smart farm network.....	59
3.5 Limitations of the Dataset Used.	60
Chapter 4 Analysis of Hydroponic Systems Data Using Machine Learning Algorithms.....	62
4.1.1 Linear regression.....	62
4.1.2 XGBoost.....	62
4.1.3 Deep Neural Network.....	63
4.2 Methodology.....	63
4.3 Results.....	65
4.3.1 Analysis of Onion Bulb Diameter using Deep Neural Network	65
4.3.2 Evaluation of the Onion Bulb Diameter Using Linear Regression Model.....	66
4.3.3 Analysis of Onion Bulb Diameter using XGBoost model	68
4.4.1.1 Classification of the Amaranthus leaves using Convolutional Neural Network.....	71
4.4.1.2 Results from the Amaranthus Leaves Images Evaluation using CNN, KNN and SVM models.	71
4.5. Machine Learning Analysis of Amaranthus Viridis Crop growth rate crop.....	75
4.5.1 Methodology.....	75
4.5.2 Results.....	76
4.5.3 Discussion.....	78
5.0 Chapter 5 Smart Farming Analysis using Federated Averaging Algorithm.....	80
5.1 Analysis of Onion Bulb Diameter using Federated Split Learning.....	80
5.2 Federated Averaging Algorithm Classification of Multi-labelled Smart Farm Dataset.	83
5.2.1 Methodology.....	83

5.2.2 Results and Discussion.....	85
5.3 Modelling of Federated Learning Smart Farm Network.....	91
5.3.1 Methodology.....	91
5.3.2 Time complexity.....	92
5.3.3 Results and Discussion.....	93
Chapter 6 Conclusion and Future Work.....	99
6.1 Summary.....	99
6.2 Research Outcome.....	100
6.2.1 Analysis of Hydroponic Systems Data Using Machine Learning Algorithms.....	101
6.2.2 Machine Learning Analysis of Amaranthus Viridis Image Dataset.....	103
6.2.3 Smart Farming Analysis Using Federated Averaging Algorithm.....	103
6.2.4 Modelling of Federated Learning Smart Farm Network.....	105
6.3 Further work.....	106
Bibliography.....	107
Appendix.....	135

LIST OF FIGURES

Figure 1: The architecture of A smart Farm.	2
Figure. 2. Changes in Farming techniques since 12,000 B.C.....	10
Figure 3: Floating hydroponic system	13
Figure 4: Aeroponic hydroponic system	14
Figure 5: Nutrient Film Technique Hydroponic system.....	15
Figure 6: Aggregate Hydroponic System.....	16
Figure 7: Floating Hydroponic system setup at SuiteLab.....	20
Figure 8: The Amaranthus Viridis crop in the nursery just after two days.....	21
Figure 9: The Amaranthus Viridis crop in the nursery just after five days.....	21
Figure 10: The Amaranthus crop after 14 days in the nursery.....	21
Figure 11: The Amaranthus crop after 21 days of transplant in the hydroponics system with the Raspberry Pi capturing images.	21
Figure 12: The CNN Architecture.....	22
Figure 13: Federated Learning Architecture.....	32
Figure 14: The Federated Learning Sequence.....	32
Figure 15: Architecture of the modelling of the Federated Learning smart farm network.....	36
Figure 16. Network architecture for a smart farm using drones.....	38
Figure 17: Fog computing.....	42
Figure 18: Evolution of cloud computing.....	47
Figure. 19: Cloud-based IoT Architecture for agriculture.	53
Figure.20: Onion crop MSE from the AER DNN model.....	66
Figure. 21 Onion crop MSE from the AG DNN model	66
Figure.22: Onion crop MSE from the Floating DNN model.....	66
Figure.23: Onion crop MSE from the NFT DNN model	66
Figure.24: Onion crop Baseline against Linear Regression diameter predictions for AER system.....	67

Figure.25: Onion crop Baseline against Linear Regression diameter predictions for AG system.....	67
Figure.26: Onion crop Baseline against Linear Regression diameter predictions for Floating system.....	67
Figure.27: Onion crop Baseline against Linear Regression diameter predictions for NFT system.....	67
Figure.28: XGBoost OBD predictions from AER system.....	70
Figure.29: XGBoost OBD predictions from AG system.....	70
Figure.30: XGBoost OBD predictions from Floating system.....	70
Figure.31: XGBoost OBD predictions from NFT system	70
Figure 32: comparison of the R ² for the Linear regressor and XGBoost Models for the Onion Crop.....	70
Figure 33: Decision Tree Confusion Matrix for Amaranthus Viridis.....	72
Figure 34: KNN confusion Matrix for Amaranthus Viridis.....	72
Figure 35: SVM confusion Matrix for Amaranthus Viridis Crop.....	73
Figure 36: CNN Accuracy for Amaranthus Viridis Crop.....	73
Figure 37: CNN Loss for Amaranthus Viridis Crop.....	73
Figure 38: Amaranthus Viridis Decision Tree Accuracy.....	74
Figure 39: Amaranthus Viridis K-Nearest Neighbour Accuracy.....	74
Figure. 40: KNeighbour Crop Growth rate predictions.....	78
Figure. 41: Support vector Regressor Crop Growth rate predictions.....	78
Figure. 42: XGBoost Crop Growth rate predictions.....	78
Figure. 43: Quantile Regressor Crop Growth rate predictions.....	78
Figure. 44: The Decision Tree Crop Growth rate predictions.....	78
Figures. 45: Theil-Sen Regressor Crop Growth rate predictions.....	78
Figure. 46: AER Loss using Adam optimizer, Learning rate=0.0000001.....	80
Figure.47: AER Loss using Adam Optimizer, Learning rate=0.1	80

Figure.48: AER Loss using SGD optimizer, learning=0.0000001.....	81
Figure.49: Loss (Adam optimizer, LR=0.01) for AG system.....	81
Figure.50: Loss (SGD optimizer, LR=0.0000001) for AG system.....	82
Figure.51: Loss (Adam optimizer, LR=0.01) for Float system.....	82
Figure.52: Loss (SGD optimizer, LR=0.000001) for Float system.....	82
Figure.53: Loss (Adam optimizer, LR=0.01) for NFT system.....	82
Figure.54: Loss (SGD optimizer, LR=0.000001) for NFT system	82
Figure 55: Comparison of the Centralised and decentralised Machine Learning Models.....	87
Figure 56: Loss using SGD optimizer, Learning rate=0.001.....	88
Figure 57: Loss using SGD optimizer, Learning rate=0.0	88
Figure 58: Loss using Adam optimizer, Learning rate=0.01.....	90
Figure 59: Loss using optimizer=Adam, Learning rate=0.001.....	90
Figure 60: Pairwise Plot of the Bandwidth Delay Product.....	93
Figure 61: Heat map correlation of the Bandwidth delay duct.....	95
Figure 62: Loss of edge node 1 of BDP network.....	96
Figure 63: Loss of edge node 2 of BDP network.....	96
Figure 64: Loss of edge node 3 of BDP network.....	96
Figure 65: Loss of edge node 4 of BDP network.....	96
Figure 66: Loss of edge node 5 of BDP network.....	96
Figure 67: Loss of edge node 6 of BDP network.....	96
Figure 68: Loss of edge node 7 of BDP network.....	96
Figure 69: Loss of edge node 8 of BDP network.....	96
Figure 70: Loss of edge node 9 of BDP network.....	96
Figure 71: Loss of edge node 10 of BDP network.....	96

Figure 72: Loss of edge node 11 of BDP network.....	96
Figure 73: Loss of edge node 12 of BDP network.....	96
Figure 74: Loss of edge node 13 of BDP network.....	96
Figure 75: Loss of edge node 14 of BDP network.....	96
Figure 76: Loss of edge node 15 of BDP network.....	96
Figure 77: Loss of edge node 16 of BDP network.....	96
Figure 78: Loss of edge node 17 of BDP network.....	96
Figure 79: Loss of edge node 18 of BDP network.....	96
Figure 80: Loss of edge node 19 of BDP network.....	97
Figure 81: Loss of edge node 20 of BDP network.....	97
Figure 82: Loss of edge node 21 of BDP network.....	97
Figure 83: Loss of edge node 22 of BDP network.....	97
Figure 84: Loss of edge node 23 of BDP network.....	97
Figure 85: Loss of edge node 24 of BDP network.....	97
Figure 86: Loss of edge node 25 of BDP network.....	97
Figure 87: Loss of Aggregate model of BDP network.....	97
Figure 88: Loss of classical model of BDP network.....	97
Figure 89: Prediction versus Original Aggregate model of BDP network.....	98
Figure 90: Prediction versus Original Classical model of BDP network.....	98

LIST OF TABLES

Table 1: Comparing Crop, Animal production, & post-harvesting in smart farming

Table 2: Comparing Sensors, UAVs, IoT & Robots in Smart Farms.

Table 3: Comparing IoT challenges in smart Agriculture (Part 1)

Table 4: Comparing IoT challenges in smart Agriculture (Part 2)

Table 5: Comparing the Advantages of IoT Cloud-based Smart Agriculture in Related Works

Table 6: Comparing Federated, Unsupervised & Supervised Machine Learning Techniques in a Smart Farm.

Table 7: Deep Neural Network Models Hydroponic systems results for the Onion crop

Table: 8: The Linear Regression Results for the Hydroponic systems

Table 9: XGBoost models results for Onion crop grown in the hydroponic system.

Table 10: Comparison of the Accuracy of Amaranthus Viridis Images

Table 11: Comparison of the Co-efficient of determinants for the Amaranthus crop growth

Table 12: Comparing the centralised and decentralised Model classification of a multi-labelled dataset.

LIST OF ACRONYMS

Acronyms	Meaning
ABR	Available Bit Rate
AdamW	Adam Window
AER	Aeroponic
AG	Aggregate
AI	Artificial Intelligence
API	Application Programming Interface
ARPANET	Advanced Research Projects Agency Network
AR	Augmented Reality
AVG	Average
AWS	Amazon Web Services
BAM	Bottleneck Attention Module
BDP	Bandwidth delay product
BDP-CoAP	Bandwidth-Delay Product Constrained Application
BQL	Bottleneck Queue Level
CAO	Calcium Oxide
CEO	Chief Executive Officer
CNN	Convolutional Neural Network
CO ₂	Carbon (IV) Oxide
CoAP	Constrained Application Protocol
CoCoA+	Congestion Control Algorithm
CPS	Cyber-Physical System
CPU	Central Processing Unit
CuO	Copper Oxide
°C	Degree Celsius
DAT	Days After Transplant
DDL	Decentralised Deep Learning
DFT	Deep Flow Technique
DNN	Deep Neural Network
DOI	Digital Object Identifier
DODAG	Destination-Oriented Directed Acyclic Graph
EC	Electrical Conductivity
F1 score	Harmonic Mean
FAH	False Alarm per Hour
FD	Federated Distillation
Fe	Iron
FL	Floating
Fig	Figure
FSL	Federated Split Learning
PFALs	Plant Factories With Artificial Lighting
fPAR	Photosynthetic Active Radiation
GNS3	Graphical Network Simulator-3
GPP	Gross Primary Productivity
HFD	Hybrid Federated Distillation
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IT	Information Technology

IoT	Internet of Things
IIoT	Industrial Internet of Things
IoV	Internet of Vehicles
ITS	Intelligent Transportation System
IVR	Interactive Voice Response
K	Potassium
LAI	Leaf Area Index
LED	Light Emitting Diode
LGBM	Light Gradient Boosting Model
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LR	Learning Rate
MAE	Mean Absolute Error
MB	Megabyte
MEC	Multi-Access Edge Computing
ML	Machine Learning
MSE	Mean Square Error
N	No
Na	Sodium
N/A	Not Applicable
NB	Naïve Base
NFT	Nutrient Film Technique
NL	Number of Leaves
Non-IIDD	Non-Independent Identical Distributed
OBD	Onion Bulb Diameter
P	Phosphorus
pH	Potential of Hydrogen
PaaS	Platform as a Service
PMMRA	Profit Maximization Multi-Round Auction
QoE	Quality of Experience
QoS	Quality of Service
R ²	Co-efficient of Determinant
READ	Robust-Oriented Edge Application Deployment
RGBD	Red, Green, Blue plus Depth
RGB_NIR	Red, Green, Blue, and near-infrared
RMSE	Root Mean Squared Error
RMSprop	Root Means Squared propagation
RSUs	Roadside Units
RTT	Round Trip Time
SaaS	Software as a Service
SAaaS	Sensing and Actuators as a Service
SDG	Sustainable Development Goal
SGD	Stochastic Gradient Descent
SGDW	Stochastic Gradient Descent Window
SNN	Spiking Neural Network
SSRN	Social Science Research Network
STDP	Spike Time Dependent Plasticity
SW-MSA	Shifted Window Multi-Head Self-Attention
TCP	Transmission Control Protocol

TDPC	Three-dimensional Point Cloud
VANET	Vehicular Ad-Hoc Network
UAV	Unmanned Aerial vehicles
UBR	Unspecified Bitrate
UP	University of Peloponnese
V2V	Vehicle to Vehicle
Val	Validation
VI	Vegetation Indices
Wi-fi	Wireless fidelity
WPCCNN	Wafer Pattern Counting Convolutional Neural Network
XGBoost	Extreme gradient boost
Y	Yes

Chapter 1 Introduction

1.1 Overview of Smart Farming

According to a report from the Food and Agricultural Organisation (FAO), a United Nations agency monitoring food production across the world, in 2019, about 2.05 billion people did not have access to adequate food on planet Earth and this number increased to 2.37 billion in the year 2020 [1]. United Nations Sustainable Development Goal (SDG) 2 aims to end hunger by achieving food security, improving nutrition, and promoting a sustainable agricultural landscape.

Farmers have used Internet of Things (IoT) devices for collecting climatic data, monitoring irrigation systems, crop growth and farm produce.

Smart farming is applying information technology to crop cultivation, fish and livestock production to increase farm produce's harvest volume and quality by utilising the resources efficiently and reducing the environmental impact. Moreover, it enhances the quantity and quality of crop yield using Information Technology (IT) software, hardware, and Artificial Intelligence models (AI) [4].

Farmers have managed agricultural irrigation systems and fertilizer applications using an Android app [2]. Farming Guru is one of the many innovative applications developed to enhance smart farming [3]. It helps farmers predict weather conditions within their region and assists farmers in understanding the climatic conditions before the commencement of a farming season to avoid loss of crop yield.

Low-power Wide-area Network (LPWAN) has been used to improve the technique of monitoring soil and plant quality by enhancing the wireless communication range within the farm. This approach minimizes the power consumption to as low as 15.36 milliampere-hour (mAh) per day for wireless devices within a smart farm network [5].

Smart farming technologies have been used to monitor soil temperature, moisture, humidity, animal location, rainfall, geospatial location, and light intensity [6]. These

sensors have been used to capture rain and geospatial locations. Using these sensors has boosted profitability and enabled environment-friendly farming.

Sensors are deployed in smart farms to capture rainfall, temperature, humidity, moisture, pH, motion, sound, and electrical conductivity data [7].

Sensors have played a role in the transformation of agriculture, as their utilisation in smart farms has made data collection easy and more reliable. The data collected using sensors from IoT devices can help farmers and researchers in the Agricultural industry to monitor, analyse and make decisions regarding smart farming. The data collected is transferred via Wi-Fi towers over the Internet. Figure 1 shows a smart farm with Unmanned Aerial Vehicles (UAV), sensors, actuators, livestock, and crops. The UAV collects data from the crops and livestock and sends the data to the Mobile edge computers via wireless connectivity.

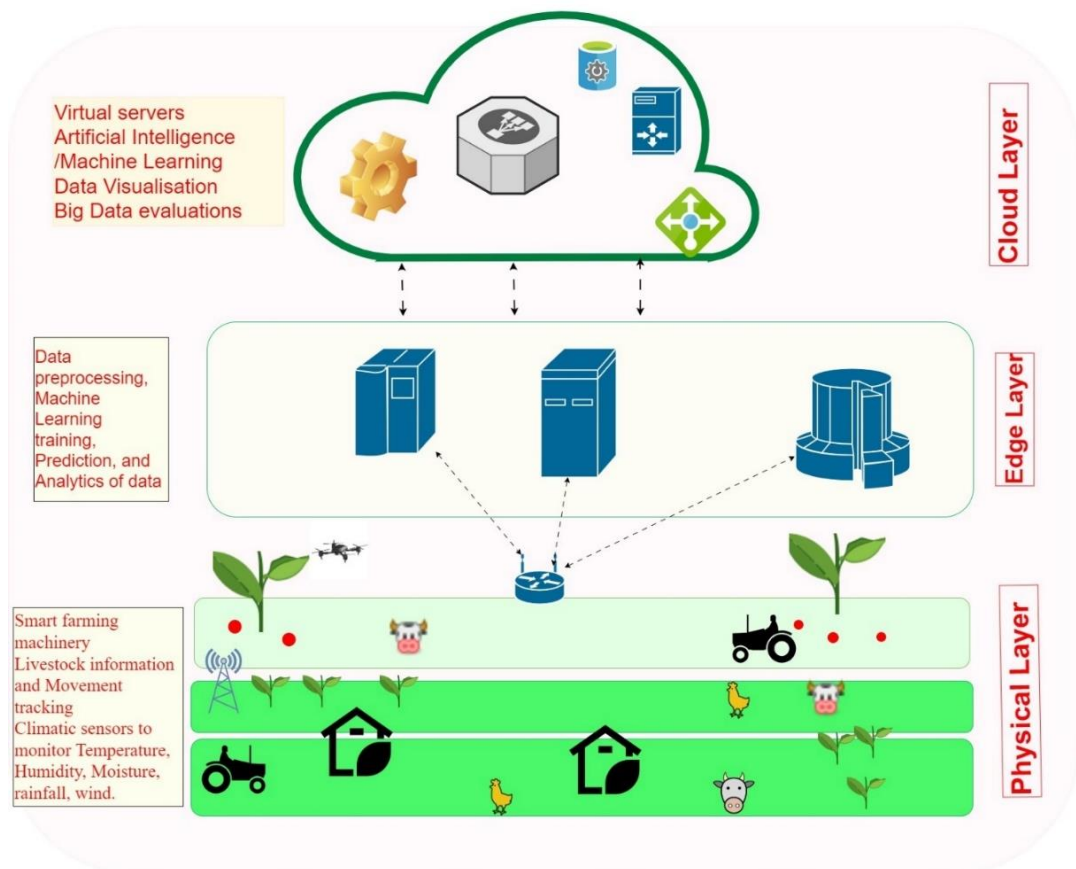


Figure 1: An architecture of a smart Farm

For example, the Cassava crop was planted, and the smart farm's water consumption has been monitored using sensors. The IoT edge devices' captured data was transmitted to the cloud via the Internet [8]. The water pump has been powered using solar panels to provide an uninterrupted power supply. Such supply has been helpful for IoT devices since the smart farms are at a remote location without a national power grid.

Artificial Intelligence (AI) devices such as Raspberry Pi have been used to monitor water levels, humidity, temperature, and pH in water systems in real time [9]. The real-time data from the hydroponic system is transmitted to the cloud and analysed by machine learning models.

A monitoring system has been developed to monitor aquaponic and hydroponic systems' air and water temperature [10]. The collected data from the IoT devices have been analysed. A correlation between the outdoor air and temperature with atmospheric pressure is 1.0 and 0.9, respectively, while the indoor air and water temperatures' correlation with atmospheric pressure is 0.7 and 0.9, respectively. The network architecture is centralised and experiencing high latency, typical of all centralised network systems since the data has been transmitted to the cloud for a thorough analysis.

Tomato plants have been grown using a hydroponic system. The concept was based on the premise that crops can be grown without soil [11]. The hydroponic system is connected to IoT devices such as Raspberry Pi incorporated with sensors that collect data such as water level, humidity, the potential of hydrogen (pH), light intensity, etc. The data analysis has been carried out using machine learning algorithms such as the Bayesian and deep neural networks [11]. However, the accuracy and loss values have not been discussed. This research examines the analysis of our results from the model evaluation of the smart farming dataset.

Due to advancements in the manufacturing of mobile devices and improved AI models, more innovative technologies have been devised for better data analysis and decision-making [12] in crop prediction.

Several challenges have been experienced in the traditional cloud-based technologies used for AI and machine learning applications [13]. These include latency, communication issues and data security. Evaluating data within cloud-based networks gives rise to privacy concerns for data owners. The Agricultural sector is one of the many industries where data privacy is a significant concern. Mobile edge devices can improve privacy by enabling the training of data closer to the data source. Using Federated Learning (FL), these data are trained at their local location, providing data privacy. The data scientists do not access the raw data. The FL concept has provided the data owner security and privacy, and an application of these FL technologies to smart farming has been conducted in this research.

Cloud-based networks have experienced serious latency challenges. The recent innovation of Decentralized Deep Learning (DDL), namely FL and Swarm Learning, has improved privacy concerns about data processing within edge infrastructures [11]. FL and swarm learning may improve the performance of edge computing devices and enhance the communication of the edge devices within the FL network. Smart farming components involve intelligent edge devices. DDL can provide privacy for the agricultural data captured by IoT devices, thereby improving the communication between the edge and the aggregate servers. DDL has many advantages in the inter-network framework, but its handling of resource allocation and deployment within a fully decentralised network is currently under investigation by researchers.

Cross-silo federated learning [16] is a network where all the client nodes of the Federated learning network can be institutions such as hospitals, banks, factories, etc. [16]. The

network edge server receives all the updated weights from each client node and sends them to the server. An example of a federated network, where all the client nodes are hospitals collecting medical data, e.g., scanned images from patients in different hospitals, is evaluated using the federated learning platform. The cross-device FL network constitutes a scenario where all the client nodes are devices [16]. An example is an enterprise network where all the client nodes are mobile devices, e.g., mobile phones, laptops, handheld devices, etc. These client nodes train their local model at the client node location and send their updated weights to the server for aggregation [16]. This thesis explores cross-device Federated Learning for a smart farm network. The modelling of the smart farm Federated Learning platform has been investigated where the client nodes of the smart farm federated learning network are cross-devices, and a comparison of the performance of the client nodes, Server and classical centralised machine learning models has been explored.

1.2 Motivation

The centralised network platform constitutes a platform where the edge nodes send their data to a central server in the cloud, and the data are analysed by the server in the cloud [17]. This process has resulted in high latency within the network. The security and privacy of the data have been a significant concern during the transmission of the data to the server [17]. Resolving the privacy concern in the agricultural sector is one of the motivations for this research. Studies of FL applications to smart farming are minimal, and this research is one of the few offering FL solutions to smart agriculture. More research is needed in the smart farming sector to address the farmers' concerns about their data analysis. Federated Learning is a machine learning technique where the updated weights are sent to the server for aggregation, but the raw data are not sent to the server. The combined model from the aggregate server is sent back to the edge nodes for further

training by the edge nodes using the local models [17]. Due to the latency issues within the centralised network, the computational time within the link is high, resulting in increased bandwidth consumption within the network link. This thesis investigates the modelling of the smart farm network within a Federated Learning platform. Emulators have been used as testbeds for experimentation since simulations cannot represent real-world scenarios within a smart farm.

1.3 Thesis Aim

The research aim is to investigate the application of AI and FL in smart farming. The smart farming network architecture has been illustrated in Figure 1. The *Amaranthus Viridis* crop has been grown within Suitelab at London South Bank University. It has been produced in a hydroponic system. IoT devices (Raspberry Pi) have been used to capture climatic data as well as images of the crop. The images have been evaluated using the Convolutional Neural network, Decision Tree, K-Nearest Neighbours and Support Vector Machines models [116]. Theil-Sen, Support Vector, Quantile, Decision Tree, K-Neighbours and XGBoost regressors [110] have been used to predict the *Amaranthus Viridis* crop growth rate based on the collected dataset from the smart farm within Suitelab. The days after transplant, the length of leaves, the width of the leaves, the number of leaves per day, and crop height have been the independent variables, and the crop growth rate has been the dependent variable.

1.3.1 Research Questions

1. Which Machine Learning Models applied to Hydroponic systems will produce optimal prediction values for the Onion Bulb Diameter?
2. How to measure the accuracy of the smart farm Decentralised and Centralised network?
3. How will machine learning models achieve optimal accuracy and crop growth rate predictions for the *Amaranthus Viridis* crop?

1.4 Contributions

This study investigates the application of Federated Learning models to smart farming. I have developed machine learning models for the Analysis of smart farming datasets from the University of Peloponnese, Greece and developed models that can evaluate smart farming datasets without having access to the raw datasets.

1. Automation of the analysis of the datasets for four hydroponic systems from the University of Peloponnese, Kalamata, Greece.
2. An extensive comparative analysis of several machine learning algorithms for Amaranthus Viridis crop growth rate, the XGBoost model outperformed the other machine Learning models for crop growth rate prediction.
3. Accessibility to smart farming datasets is a challenge. Federated Averaging and Federated split-learning models have been used to evaluate smart farming datasets without having access to the raw dataset.
4. In the comparative analysis of the Amaranthus Viridis image dataset using Machine Learning Algorithms, the CNN model outperformed the other considered models.
5. Modelling of the Federated Learning (FL) smart farm network, the decentralised farm network produced a higher accuracy than the centralised machine learning models.

1.5 Organisation of Thesis

This study identifies some challenges with smart farming and explores the application of Artificial intelligence to smart agriculture, as follows:

Chapter 2: Related Work

Overview of the technologies applied to smart farming. The technologies reviewed are cloud, centralised network platform, decentralised network platform, mobile edge computing, Internet of things, unmanned aerial vehicles, and multi-access mobile computing. Various hydroponic systems have been considered for this research.

Additionally, the use of Federated Learning is presented. The gaps in the application of technologies to smart farming are identified in this chapter.

Chapter 3: Detail description and data preprocessing of the Onion Bulb diameter dataset from the University of Peloponnese, Kalamata, Greece, Amaranthus Viridis Crop numerical and image datasets, and Federated Learning smart farm network modelling datasets used for this research.

Chapter 4: This research has developed Artificial intelligent models that predict the OBD, and these AI models automate the OBD predictions. Amaranthus Viridis has been grown in the Suitelab smart farm. To determine the crop growth rate, AI models have been used to predict the crop growth rate.

Chapter 5: Existing literature discusses centralised models used for crop classification. This research has proposed decentralised models for crop classification. The Modelling of the decentralised smart farm network has also been investigated.

Chapter 6: Conclusion and future works

Conclusions are drawn from the obtained results, and recommendations are proposed.

Future work focuses on further areas of research.

1.6 PUBLICATIONS

The publications listed below are major academic contributions produced while conducting current research.

1.6.1 PUBLISHED PAPERS

1. Idoje Godwin, Tasos Dagiuklas, Muddesar Iqbal, Survey for smart farming technologies: Challenges and issues, *Computers & Electrical Engineering*, Volume 92, 2021, 107104, ISSN 0045-7906, <https://doi.org/10.1016/j.compeleceng.2021.107104>.
(<https://www.sciencedirect.com/science/article/pii/S0045790621001117>)
2. Idoje, Godwin and Mouroutoglou, Christos and Dagiuklas, Tasos and Kotsiras, Anastasios and Muddesar, Iqbal and Alefragkis, Panagiotis, Comparative Analysis of Data Using Machine Learning Algorithms: A Hydroponics System Use Case. Available at SSRN: <https://ssrn.com/abstract=4305401> or <http://dx.doi.org/10.2139/ssrn.4305401>
3. Idoje Godwin, Tasos Dagiuklas, Muddesar Iqbal, (2023), Federated Learning: Crop classification in a smart farm decentralised network, *Smart Agricultural Technology*, Volume 5, 2023, 100277, ISSN 2772-3755, <https://doi.org/10.1016/j.atech.2023.100277>.

1.6.2 THE MANUSCRIPT SUBMITTED AND UNDER PEER REVIEW

1. Godwin Idoje, Tasos Dagiuklas, Muddesar Iqbal, XGBoost Analysis of Amaranthus Viridis Crop Growth Rate, *Smart Agricultural Technology*.
2. Godwin Idoje, Tasos Dagiuklas, Muddesar Iqbal, On the performance of Federated Learning in Smart Farming, *EAI CollaborateCom 2023 - 19th EAI International Conference on Collaborative Computing*.
3. Godwin Idoje, Tasos Dagiuklas, Muddesar Iqbal, Comparative Analysis of Amaranthus Viridis Image Dataset, *Smart Agricultural Technology*.

Chapter 2 Background

2.1 A Survey of the gaps identified in a smart farm

Agriculture production has been enhanced using information Technology, and edge devices have been used to control actuators, manage resources, and improve product quality to reduce production costs and maximize profits [17]. The use of IoT in smart farming has enabled the collection of climatic data, detection of diseases, monitoring of the growth rate of crops, appropriate harvesting techniques to reduce crop wastage, tracking of farm animals' movement within the farm environment, livestock conduct & improvement of the breeding of livestock and crops. It can be inferred from Figure 2 that Agriculture has changed over thousands of years [18], with improved farming techniques, ranging from crop planting to harvesting, storage of farm produce, use of genetically modified seeds and use of AI in farm machinery.

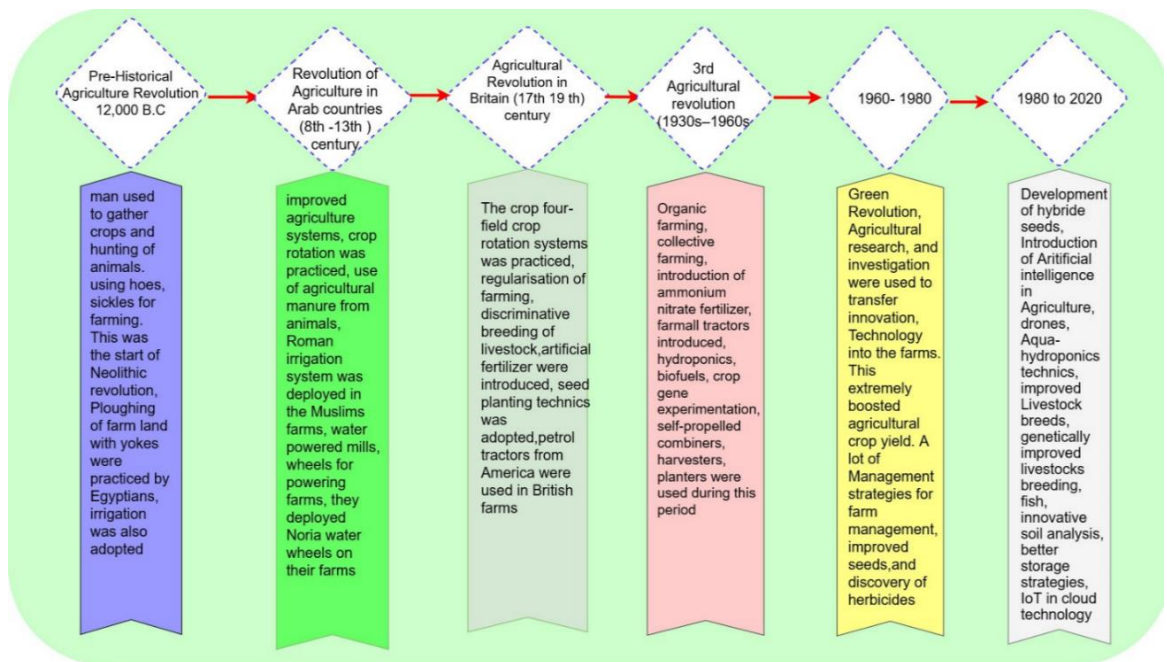


Figure 2. Changes in Farming techniques since 12,000 B.C. [84]

2.2 Smart farming

The introduction of IT devices to farming, such as Artificial intelligence, the Internet of Things, and sensors, is referred to as smart farming [19]. It can be inferred that software and hardware infrastructure have boosted farming produce. Farmers can monitor their farms remotely using technology and control actuators using the Internet of Things within the farm [17].

Water management has been improved using IoT within the farms [19], and climatic parameters prediction, soil management and disease outbreak within the farm are effectively monitored. The researchers discussed that in 2050, 525 million farms would use sensors, and a vast volume of data will be generated and transmitted within these farms. It can be inferred that more advanced software and hardware development IT methodologies will be required to handle these smart farming data.

2.2.1 Crops production

The Authors in [20] discussed that leaves with overlapping background images could be separated using a marked watershed algorithm. Their research achieved a 97% accuracy in strawberry crop disease identification using a minicomputer running a neural network model with a computational time of 1.2 seconds in Cyprus. Still, their developed model cannot resolve the challenge of separation since the crops can display different symptoms at different stages of infestation. The authors need to consider other protocols for communication between the devices used for disease detection for crops and animals.

2.2.2. Livestock production

Farm Animals frequently move within the farm. [21] discussed that monitoring of the animals can be achieved by attaching a neck collar device to the animals for monitoring their movement and collecting body vitals such as temperature and blood pressure. It can

be inferred that the devices will stop transmitting data from the farm animals once the battery-operated device runs out of power. Sea-lice counting and crowd control have been used to solve aquaculture problems in smart fish farming [22]. It can be deduced that IoT devices have been used to monitor animals' movement, behaviour patterns, body vitals, and fish crowd control [21,22]. Still, their research has not addressed the psychological impact of the electronic devices attached to these animals' bodies.

2.2.3. Post-harvesting

The Red Green Blue-D sensors have been used to detect the sweet pepper's colour and shape [23]. Their research enabled them to determine the dimensions of the pepper, but the computation time for the detection was very high. The authors in [24] used the Monte Carlo simulation to determine the harvest age of a coconut crop. It can be deduced from their research that the coconut crop's harvest age impacts the crop's selling price. Their research has yet to consider other factors such as holding cost, inventory, transportation cost, and demand, which also affect the selling price of the crop. The computation time of the model needs to be addressed for faster detection of the sweet pepper dimensions, and factors like demand, transportation cost, holding cost, government regulations, levies and taxes should be considered for their models to determine the coconut harvest age effectively [23,24].

Table 1 depicts gadgets used in smart farming to monitor farm animals' movement and power-related challenges [21,22], and these animals experience psychological problems. The inventory, transportation cost and market demand, computational time, and communication protocol have been issues identified in Table 1 in smart farming [20, 23, 24].

Table 1: Comparing Crop, Animal production, and post-harvesting in smart farming [84]

Properties	Crop production	Animal production	Post harvesting
The computational power of a system	N	N/A	Y[84]
Operated by Batteries	N	Y [84]	N
Psychological effect	N	Y [84]	N/A
Detection speed	Y [8]	N	Y [84]
Demand of market	Y [8,84]	Y [8,84]	N
Inventory	Y [8]	Y [8]	Y [8]
Holding cost	N	N	Y [8]
Transportation cost	N	N	Y [8]

Yes = Y, No = N, N/A = Not Applicable.

2.2.4 Hydroponic Systems

2.2.4.1 Floating Hydroponic System

The floating hydroponic framework is a method in which plants are placed on a floating system, often composed of Styrofoam. The plants float in a nutrient-rich liquid, allowing the roots to absorb the necessary nutrients [161].

This system floats on a nutrient-rich medium, and the plants are placed in gaps in the raft, allowing their roots to dangle freely within the solution. The medium is oxygenated to ensure enough oxygen flow to the roots. The system is simple to set up and maintain, perfect for cultivating vegetables, and inexpensive but unsuitable for growing tap roots crops. Figure 3 depicts the floating hydroponic system with the air pump pumping air into the water and the crops floating on the top.

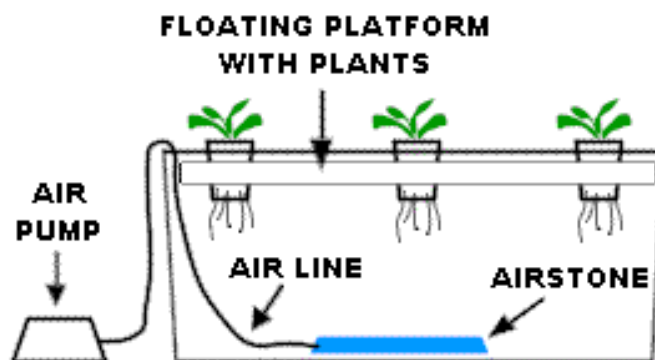


Figure 3: Floating hydroponic system [164]

2.2.4.2 Aeroponic Hydroponic System

[162] discussed that plants cultivated using Aeroponics hydroponic systems have their roots suspended in the air, and the crop roots have been sprayed with the solution. As a result of the suspension of the plant roots, good air circulation is achieved, enabling plant growth. In any case, for the frameworks to work successfully, the root environment must be kept up at 100% relative humidity to anticipate the root's lack of hydration [163].

Although aeroponics frameworks can deliver yields ten times more than crops cultivated using soil, pump glitches or power disruptions can cause drying up and the killing of the crops. Figure 4 shows the Aeroponic Hydroponic system.

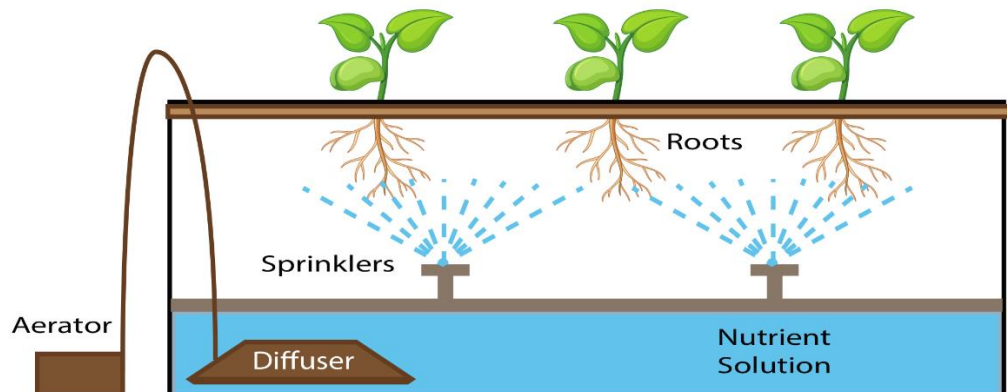


Figure 4: Aeroponic hydroponic system [164]

2.2.4.3 Nutrient Film Technique (NFT) Hydroponic System

The Nutrient Film Technique hydroponic system is a system where the mixed solution flows over the roots, providing them with the necessary nutrients [164]. It requires a smaller volume of nutrients compared with the floating system, but the energy requirements are higher than the floating system. The air within the channels blows over the roots, directly providing oxygen to the crops. The passage is sloppy in shape, which allows the solution to flow by gravity. The crop roots have direct contact with the mixed solution, enabling the crops to absorb the nutrients from the hybrid solution. The mixed

solution is recycled through the system, allowing for reliable use of the solution and preventing wastage. The hydroponic system needs to have a steady, consistent flow of the solution and efficient nutrient concentrations. Still, power and pump failure are some issues associated with the Nutrient Film Technique system [164]. Figure 5 shows the Nutrient Film Technique Hydroponic system.

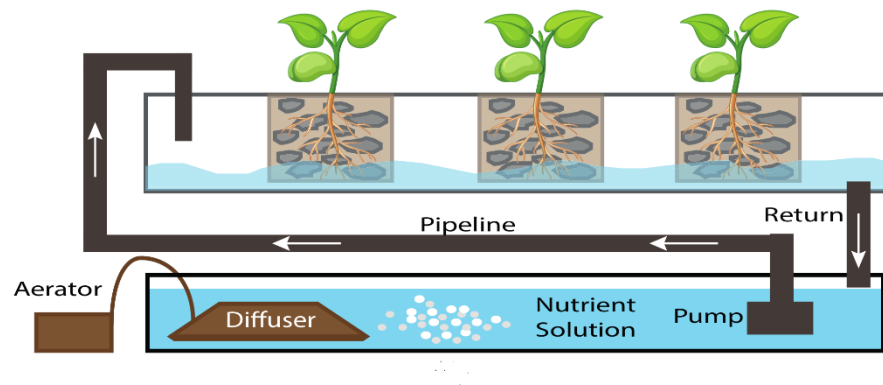


Figure 5: Nutrient Film Technique Hydroponic system [164]

2.2.4.4 Aggregate Hydroponic System

In an Aggregate hydroponic framework, crops are developed in a robust and dormant medium such as perlite, vermiculite, or coconut coir [164]. The medium enables the crops to stand upright. It holds dampness and supplements the crop roots. The system pH and nutrient level should be observed frequently. The solution has been drained to prevent waterlogging. A drip irrigation technique is applied to the system. The system solid medium used for holding the crops upright sometimes dissolves in the solution and contaminates, which causes disease infestation to the crops [164]. Figure 6 shows the Aggregate Hydroponic system.

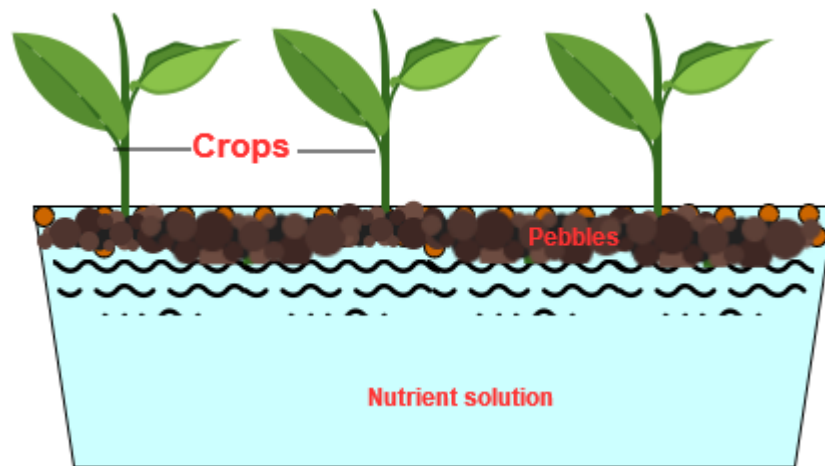


Figure 6: Aggregate Hydroponic System

2.2.4.5 Amaranthus Viridis Crop Grown in Suitelab Hydroponic smart farm Testbed

Smart Farming is the introduction of IoT, AI, cloud computing, and edge computing to agriculture to improve its harvest quality and quantity [83, 84]. Practitioners are now considering using ML. Big data are being used along with sensors to capture data in smart farming, which are used for evaluation to forecast crop yield, manage water usage, and detect infestation within the farm [85]. Some continents have poor weather conditions, which causes them to use hydroponic systems to grow crops [86]. Artificial lighting has been deployed to save energy in hydroponic Farming [87]. To monitor IoT devices deployed in smart farms, Application programming interface (API) models have been developed [88].

Robots are currently deployed to smart farms to cultivate the soil [89]. Raspberry Pi is an edge device used for data capturing via sensors connected to it. These data are trained, and predictions are made from the analysis. The authors in [90] have been able to grow Lettuce crops using the NFT hydroponic system. Still, they have not been able to monitor the evapotranspiration during the cultivation of the lettuce crop.

Tomatoes crop growth rates have been obtained using a machine learning algorithm while growing on soil. They discuss that the crop uptake of nutrients has been higher for Na and K than for other nutrients [92]. Their research did not inform us of the dry weight matter of the crop and the relative crop growth rate. Further research is needed to ascertain the tomatoes' crop growth rate using recycled water.

[93] discuss that hydroponic systems used for crop cultivation have been used to produce good-quality crops. Better quality and quantity of crops have been grown by [94], and it can be inferred from their research that the crops used less water during growth compared with cultivation on a soil medium.

The authors in [96] discuss that highly valued vegetables have been grown using hydroponics systems, and the crops are not infested with soil-prone diseases since they are grown in water mixed with nutrient solutions.

Figure 7 depicts the Floating hydroponic system architecture for cultivating the *Amaranthus Viridis* crop at London South Bank University. It has been observed that the *Amaranthus Viridis* plant gets its light for photosynthesis from artificial light and obtains its nutrients from the water solution mixed with nutrients. The sensors connected to the IoT device (Raspberry Pi) collect the climatic data (Temperature and humidity) every minute and transmit these data from the SQLite3 database to the Mobile edge cloud. The data are further transmitted to the tier 1 cloud. The Application Programming Interface developed using Python coordinates the collection and transmission of the data from collection to the cloud. The edge devices also collected image data of the *Amaranthus* leaves. These images were used to train the dataset using a convolution neural network, support vector machines, and K-Nearest Neighbour models. The submersible pump has been used to refill the crop water container from the reservoir. In contrast, the air pump pumps air into the tank to provide oxygen inside the water, which helps to prevent

bacteria and fungi growth within the water. The crops also need this oxygen during photosynthesis.

[90, 97] narrates the cultivation of lettuce using hydroponic systems. Their considered system monitors the pH of the liquid and climatic parameters in real time. The tomato crop growth rate has been computed using the data obtained from the hydroponic system in which it has been grown [98]. An integrated system that rears fish and grows crops at the same time is known as aquaponics. This system has been used to capture the electrical conductivity (EC), pH, temperature, and carbon IV oxide to evaluate the dataset [99]. The Deep Flow Technique (DFT) is growing crops in a floating raft where the plant roots are aerated with the solution, and the IoT devices connected to the system capture data while the crop grows [100].

Handheld devices have been used to capture data from smart farms remotely. These data can be accessed at any time and from any location for analysis because they are stored in the cloud [101, 102, 103, 104]

Sensors have been used to capture climatic data, light intensity, and carbon (IV) Oxide from a smart farm. Still, the parameters considered for independent and dependent variables have not been iterated [105, 107]. Farms installed with IoT sensors experience data security challenges, and the absence of broadband service has affected such farms' network connectivity [106]. The LoRaWAN platform has been used to collect data within a smart farm, the data can be viewed in graphical format, but farmers will have to pay to use the LoRaWAN platform, and rural farmers cannot use it due to financial constrain [108].

A Floating hydroponic smart farm has been set up within the Suitelab research lab at the computer science of London South Bank University. The *Amaranthus Viridis* crop has been grown for two weeks in a nursery and transplanted to the hydroponics system for another six weeks before it was harvested. The decision to use the *Amaranthus Viridis* crop for this research has been made because of its shallow root system, which is suitable for the hydroponic systems procured for this experiment. The hydroponic system pipe component is 3cm in diameter, as seen in Figure 11, and is more suitable for growing shallow root crops. The *Amaranthus Viridis* crop matures within eight weeks of harvest. The nutrient solution used as fertilizer for the hydroponic systems, which provided the nutrients in the water, contained the following chemical components, namely, Nitrogen, Phosphorus, and Potassium in a ratio of 2:0.9:3.3. The Nutrient solution had Nitrogen (Nitrogen Nitrate, 1.78%, Ammonia Nitrogen, 0.28%), Phosphorus Pentoxide, 0.87%, Potassium oxide, 3.31%.

The Hydroponic nutrient solution, when mixed with water, contains the following percentages of elements: Calcium Oxide, 1.47%; Copper (CU), 0.002%; Iron (Fe), 0.040%; Manganese, 0.010%; Molybdenum, 0.001%; Zinc, 0.0025%. The hydroponic system uses Sunlight or artificial light for photosynthesis. The Lumi 600watts Ballast Grow Light Kit Hydroponics Black 600watts Bulb has been used as a hydroponics system artificial light. The 600-watt light has been put on for 12 hours daily for growing the crop. An IoT device, Raspberry Pi, has been configured to capture images of the crop while it was producing, and the DHT 11 sensor has been connected to a Raspberry Pi. An application program interface (API) has been developed to collect the Temperature and humidity data within the Hydroponic smart farm. The collected data has been stored in the SQLite database repository.

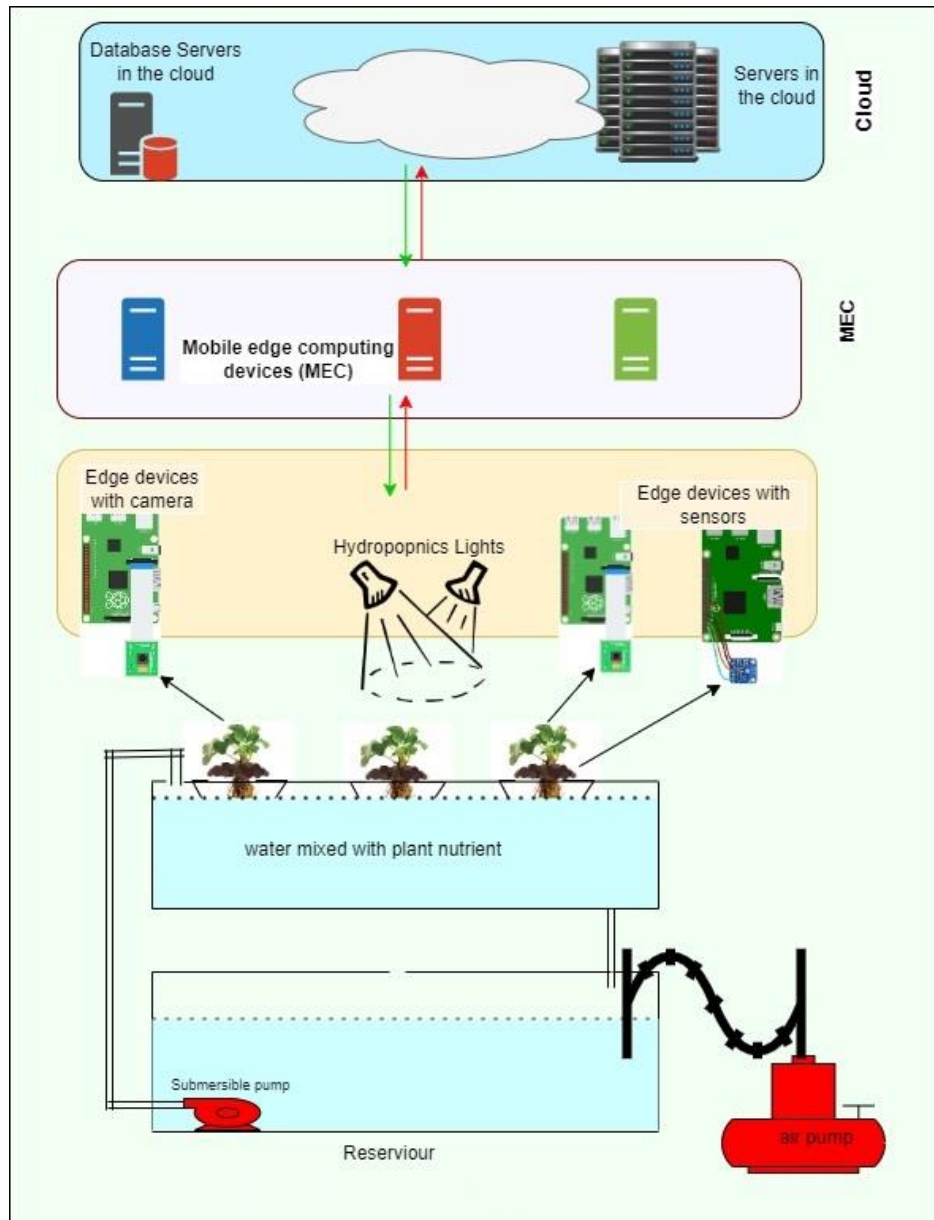


Figure 7: Floating Hydroponic system setup at SuiteLab [158]

Figure 7 shows the Floating Hydroponic system deployed at London South Bank University to cultivate the *Amaranthus Viridis* crop. Figures 8 and 9 show the *Amaranthus Viridis* crop in the nursery just after two days and five days, respectively, of planting. Figure 11 shows the *Amaranthus* crop after 14 days in the nursery. Figure 12 shows the *Amaranthus* crop after 21 days of transplant in the hydroponics system, with the Raspberry Pi capturing images as the crop grows.

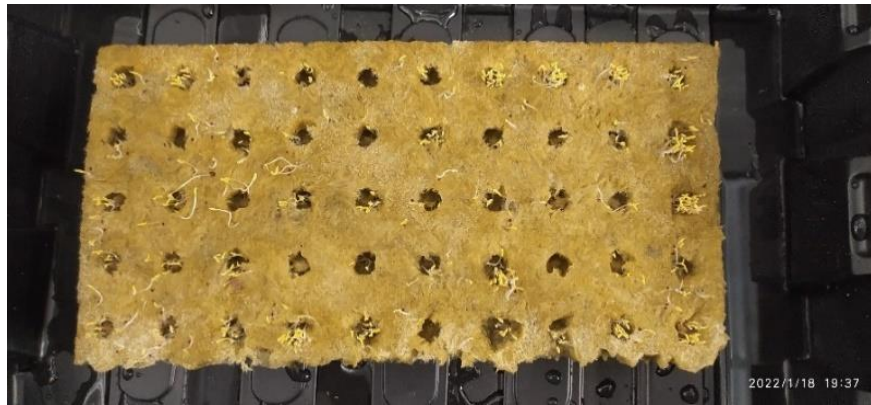


Figure 8: Amaranthus Viridis crop in the nursery just after two days



Figure 9: Amaranthus Viridis crop in the nursery just after five days



Figure 10: Amaranthus crop after 14 days in the nursery



Figure 11: Amaranthus crop after 21 days of transplant in the hydroponics system with the Raspberry Pi capturing images

2.2.4.6 Convolutional Neural Network

The panicle segmentation-based convolution neural network has been used to evaluate wheat ears count, and a harmonic mean of 98.3% has been achieved during one ear classification training [113].

A comparison of the performance of CNN and regional CNN for the evaluation of real-time object detection with an accuracy of 47.7% and 75.6% for CNN and regional CNN, respectively, have been achieved.[114].

The authors in [115] discussed that using their dynamic resource-aware CNN model framework, their model has achieved 97.1% and 94.6% for CNN inference accuracy estimation and CNN model resource consumption, respectively. Figure 12 shows the convolutional neural network architecture using the Amaranthus Viridis in the input layer.

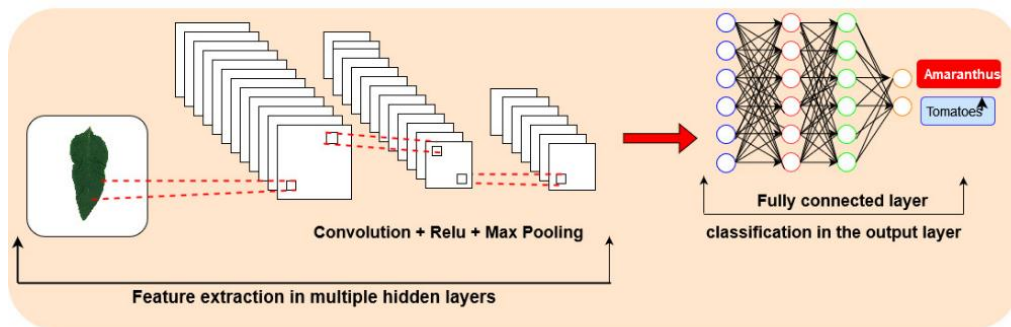


Figure 12: CNN Architecture

The CNN operation formula can be expressed as in equation 1,

$$G_j^l = \sum_i^m G_i^{l-1} \times T_{ij}^l + Q_j^i \dots\dots\dots(1)$$

Where l is the hidden layer has a feature map of $l-1$, the bias is Q_j^i , the feature map weight is T_{ij}^l Using the J^{th} feature map [114].

The Fast-R-CNN algorithm was modified using the bottleneck attention module (BAM) for multi-target detection of transmission lines for image recognition. They used a softer NMS, K-means++ clustering algorithm to optimize the model. The modified Fast-R-CNN

accuracy achieved a higher accuracy value of 90.71% compared with the traditional Fast-R-CNN, which had an accuracy of 86.41% [116].

The Spiking Neural Network (SNN) algorithm uses a low power consumption for its training. The authors use the spike time-dependent plasticity (STDP), reward-modulated STDP learning rules and Spyke Torch library. Upon evaluating the model using the MNIST dataset, an accuracy of 97% has been achieved [117].

The wafer pattern counting convolutional neural network (WPCCNN) uses a deep learning algorithm to identify, evaluate and count wafer patterns with 11 different patterns from over Two hundred wafers. Their proposed model uses an encoder and decoder structure on the CNN algorithm. An accuracy of 99.6% has been obtained from their model evaluation, indicating their model's high performance [118].

2.2.4.7 Federated Learning

The Federated Learning algorithm aggregates several models by a server in a Federated learning platform or network [135]. The Federated Learning server sends its base model to the edge nodes, and these edge nodes use the base model for training its local datasets. The updated weights are sent to the server. The server aggregates all the various updated weights from each edge node and forms a new global model. The updated global model is returned to each edge node for another training of its updated weights. This process continues until convergence is achieved. [159] discussed that one of the challenges experienced in enterprise networks is Data privacy. Federated Learning has been adopted to address this privacy challenge in smart farming.

As discussed in [130], using satellite images has helped to analyze soil and crops in farmlands to determine their condition, which has been used to resolve many challenges in agriculture. These solutions have enhanced farming through the forecast of the crop harvesting time to make decisions to combat poor harvest from the farms. Their research

only considered small state farmlands and has not been widely considered for nationwide farms. According to [131], using the six domain models for designing smart farms has created interconnection between several systems and boosted data utilization. The domain models have enhanced the joint data-sharing ecosystem between the industry players.

This research explores a smart farming use case within the federated learning platform, inferring that smart farms utilize interconnected ICT and IoT devices via the cloud or the internet. Significant data exchange and traffic flow within these farms are indicated in [130, 131]. It will be interesting to investigate federated learning applications to these smart farms, where the data owner retains control over their data and the data scientist can evaluate the smart farming data without direct access to the farm data.

The research of [132] discussed the automation of a smart farm where a mobile app is used to monitor the irrigation system. The App enables farmers to monitor the IoT device's data and the plant's environmental parameters. The limitation of their work is that the automated system does not handle any analysis of the data captured.

Using automation of IoT devices, [133] was able to develop an IoT device that collected climatic data, and these were used to make predictions for frost and pest infestation within their smart farm. Still, the limitation of their work was that the device could not effectively handle the large volume of Big Data generated from the capturing and the farmer's feedback for analysis purposes.

The research conducted by [134] discussed the use of IoT devices to capture climatic data such as humidity, temperature, and moisture content of the soil. Their results inform us that the system retained more water at night due to the low temperature, and more water was lost during the day because of the high temperature within their country. The plants experience high moisture during the day and low humidity at night, as captured by the IoT

device, prompting the researchers to calibrate the automated device to supply more water during the day. The limitation of their work is that they needed to provide an analysis of the data captured by the automated device.

It was observed from [130,131,132,133,134] that IoT devices enabled the automation of the traditional farming system in many countries. Climatic and crop data were captured by the devices deployed by the various researchers in their work. Still, many automated IoT devices performed little or no analysis of the captured data. The synergies of the results, predictions, or analysis have not been achieved due to different platforms and operating systems for global use.

The concept of federated learning is a decentralised technique where the end devices collect data and do not share it with their neighbours [135]. However, they update the model and transmit the updated weights to the server. FL provides privacy for the data. The server then aggregates the updated weights from all the edge devices and sends the new updated global model to the edge or end nodes with information on its new updated global model. This iteration process continues until the model converges.

The authors in [136] discuss that recent research has focused on supervised learning in federated learning platforms and recommend that academics investigate the unsupervised machine learning models within a federated learning platform or the combination of the federated averaging and unsupervised learning models.

As discussed in [137], to preserve the privacy of the data trained in a machine learning system, a shift from the classical machine learning algorithm to a decentralised machine learning platform where the data are not sent to the central server or cloud for training, this equally reduce the latency since the bandwidth consumption within the network is

reduced equally. It can be inferred from [135, 136, 137] that their work has not been applied to an unsupervised learning algorithm, which is a limitation.

As cited in [138], federated learning enables us to establish a cross-domain, cross-data, cross-enterprise biosphere for Big data and unreal cognition. The limitation of their research is that their work did not inform us if the edge nodes considered are homogeneous or heterogeneous.

Homogeneous edge nodes all have the same attributes, such as the same memory, processor, and equal power capacities, while heterogeneous edge nodes have different memory, processing and power capacities. In this research, our edge nodes are homogeneous because all the edge nodes have the same memory, processor and transmitting power.

The authors in [139,140] discussed a server and edge nodes correlation and cross-domain, cross-data transaction between edge nodes and server nodes in a Federated learning network. It was discussed in [139] that little model updates and weight increments within the federated learning network were considered in their research, contributing to minimising the latency in their work.

It can be inferred that two ranks reduced communication costs in a Federated Learning network with variable parameters. Their research considered low bandwidth consumption edge nodes during the rounds. Still, their model has not been tested in a high bandwidth scenario like a video broadcasting network, which is a limitation.

In [140], it is considered that a modest assets scenario where federated learning (FL) network, federated distillation (FD), this is an algorithm that reduces communication overhead better than the Federated averaging algorithm. It interchanges just the aggregated model outputs within the network and the Hybrid federated distillation (HFD)

algorithm. This helps enhance the performance gap between FL and FD by controlling the average probability vector and average input from the dependent variable during the offline phase. It was reported in their paper that FD and HFD yield better results compared with federated learning when the number of uplinks and downlink channels is negligible. Despite the impressive outcome of their work, their research needed to address the use of their model for a wired non-fading channel link and no information was provided on the frequency of the wireless edge nodes used for the experiment.

The work of [139] inspires our architecture, where we set up a server in our test bed for experimentation. We use edge nodes with the same attributes, such as memory capacity processors. We inferred that all our edge nodes are homogeneous since they have the same capacities and features.

In [141], it is observed that using the distance of convex functions enables researchers to pick more nodes compared to other technological technics when the accumulation of mobile edge computing (MEC) devices allows applications to be run close to the service user for a rather demanding mean square error request which was achieved through increment of antennas at a base station in their experiment. The MEC allows cloud computing features and information technology profiles at the edge of any network.

It can be deduced from their paper that the aggregation of more mobile edge nodes in their experiment enhanced the performance of their model. Some limitations were observed in their research, such as not investigating the effect of channel uncertainty in the model accumulation, and their study needed to address the computational complexity of the algorithm used.

This research considers the smart farming variables' performance within the laudable federated learning technique, a subset of machine learning.

The authors in [142] have used a greedy algorithm, a two-magnitude image analytical solution, where the edge nodes are vehicular. It can be deduced that the greedy algorithm helps to achieve model accuracy and aggregation efficiency for a federated learning vehicular network. Their work inspires the accuracy performance of smart farming variables within a federated Learning platform.

The authors in [143] discussed their modified C-fraction Federated Stochastic gradient descent algorithm, which considers the ratio of the online participants to the total number of participants within the federated network. Their modified algorithm gave between 99.65% to 99.85% accuracy from the training using different values of the c-fraction during experimentation. Despite the impressive results from their experiment, it can be observed that the same learning rate was considered for the four different C-fractions. It would have been interesting to get the results for each C-fraction using different learning rates.

Many different learning rates have been considered for this research, unlike the study of [143], to determine the effect of the different learning rates on our accuracy values using other optimizers such as stochastic gradient descent (SGD) and Adam optimizer.

According to the authors in [141], the Adam activation function has been used in a Federated averaging algorithm for crowd-sourcing speech data to study an asset-limited wake word detector instead of using the normal global averaging for its training. Their work achieved a 95% recall per 5 false alarms per hour (FAH) for 100 communication cycles when the crowd-sourced dataset communication cost per participant was 8 Megabytes (MB).

Using the Adam optimisation enabled the network to converge faster. The limitation of their work is that a memory-efficient end-to-end model was not used in their research. [141] used a speech dataset within a federated learning platform. Still, our experimentation is applying Adam and SGD optimizers to a smart farming dataset within a Federated

learning platform, which can be considered an original knowledge contribution. We obtained different accuracy and F1 score values during hyper-tuning the learning rate.

The authors in [142] stated that stochastic gradient descent converges better than other algorithms investigated in their paper. [143] discussed that SGD converges faster, but the step sizes decay quickly, which affects its efficiency during training. However, [144] stated that the Adam optimizer is a robust optimizer that combines two other optimizers, namely Adagrad and RMSProp, and uses less memory for training and converges faster than SGD.

We have considered both the SGD and Adam optimizer in our research for analysis, and our results depict the model's performance using smart farming variables within a Federated Learning network. The results indicate that the Adam optimizer has higher accuracy than the SGD optimizer while using climatic variables for crop type prediction.

It is evident from [137,138, 139,140,141,142,143,144] that federated learning has been implemented in various networks with edge nodes which have reduced edge node queuing, bottleneck traffic, and latency of traffic due to the application of different technique of algorithm schemes to facilitate the communication cost and make the network more efficient.

Related works have shown that researchers have adopted several techniques to reduce the latency and network traffic challenges within a particular network. This research explores options for hyper-tuning the parameters to achieve optimal convergence within the federated learning network while predicting the crop type.

2.2.4.7.1 Federated Learning Algorithm

1. Initialisation of the tasks

The server decides the training task.

The server handles the training process and global model hyperparameters.

The selected participants receive the task and initialise the global model V_p^0

2. Update and train the local model.

The edge nodes use their local data and devices to optimize the local model V_i^t ,

Where t represents the recent iteration index.

The purpose of the edge nodes i in the process t is to determine the best variables V_i^t ,

that will decrease the loss function $L(V_i^t)$

$$V_i^t = \arg \min_{V_i^t} L(V_i^t) \dots\dots\dots(2)$$

The server receives the updated local model parameters.

Global model accumulation and modification.

Local models are aggregated from the edge nodes to the server. The edge nodes receive

the modified global model. V_p^{t+1}

$L(V_p^t)$ is the global loss function, minimised by the server.

$$L(V_p^t) = \frac{1}{N} \sum_i^N L(V_i^t) \dots\dots\dots(3)$$

The global loss function converges after many repetitions of steps 2 to 3 (state which additional iterations do not enhance the model)

Algorithm 1: Federated Averaging Algorithm [11]

The Learning rate is η
The number of local epochs is e
Locally reduced batch (mini-batch) = S
Number of edge nodes in each iteration = c
Global model V_p^o

1. The participants are represented by i
2. Local Training V_i
3. Divide the local dataset G_i to small mini-batches, and place in set $G_i \setminus \setminus$
4. s which is part of a set S_i
5. for every local epoch h , from i to e do
6. for every $s \in S_i$ do where (η = learning rate and δ =gradient of L on S)
7. end for
8. end for
9. [server]
10. set V_p^o
11. for iteration t from 1 to t do
12. arbitrarily select a subset Y_t of C edge nodes from N
13. for each edge node $i \in Y_t$ similarly do
14. $V_i^{t+1} \longleftarrow$ local training (i, V_p^t)
15. end for
16. aggregating $V_p^t = \frac{1}{\sum_{i \in N} D_i} \sum_{i=1}^N D_i V_i^t$
17. end for

The server then aggregates all the updated weights from each edge node to determine its new global model. Subsequently, it sends the updated global model back to the edge devices for the next round of training iterations. This process continues until further training no longer improves the local models. It is important to note that the server never sees the raw data of the edge devices throughout the entire process, which provides data privacy for the data owner for the whole analysis and creates data security.

In this research, as shown in Figure 13, which depicts our architecture, the mobile edge computers receive the data from the IoT devices, the MEC perform the local training of the data and only sends their updated weights to the server upon completion of the aggregation of all the received local weights, the server sends its new updated global model to each MEC, and they also use this new received updated global model to perform the next training, this process continues until convergence is achieved.

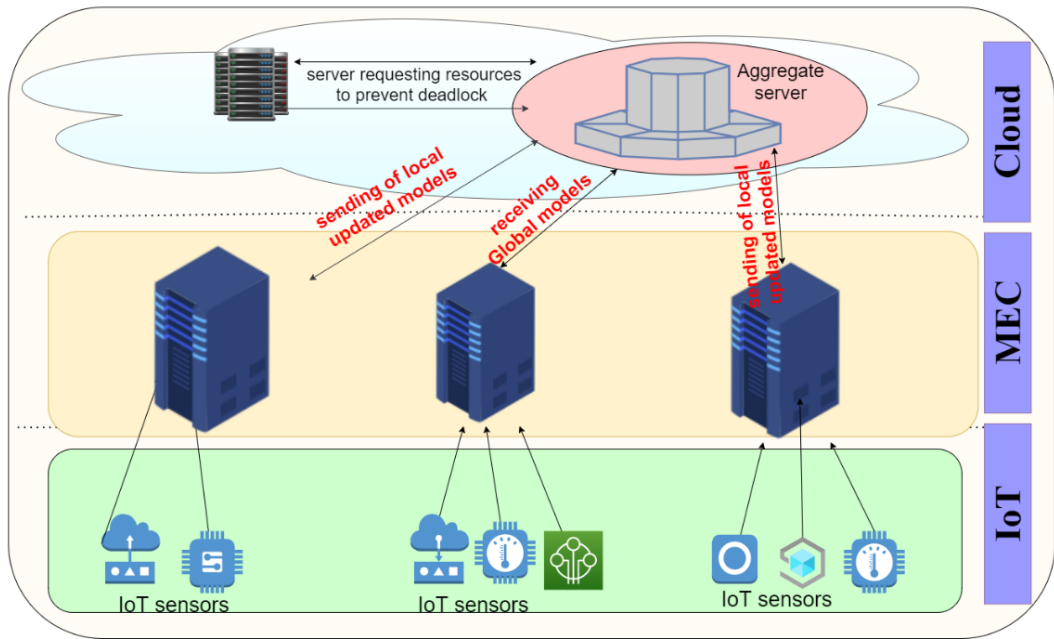


Figure 13: Federated Learning Architecture

Figure 14 shows the Federated Learning network flow sequence from the sensors that capture data and send these data to the edge devices. Unlike classical machine learning, where the data is sent to the cloud for training, Federated learning adopts a different approach. The server sends its initial global models to the edge devices. Since training takes place at the edge nodes where the data is domiciled, the edge devices use the initial global model sent from the server to train its local model, the edge devices then send its updated weights to the server.

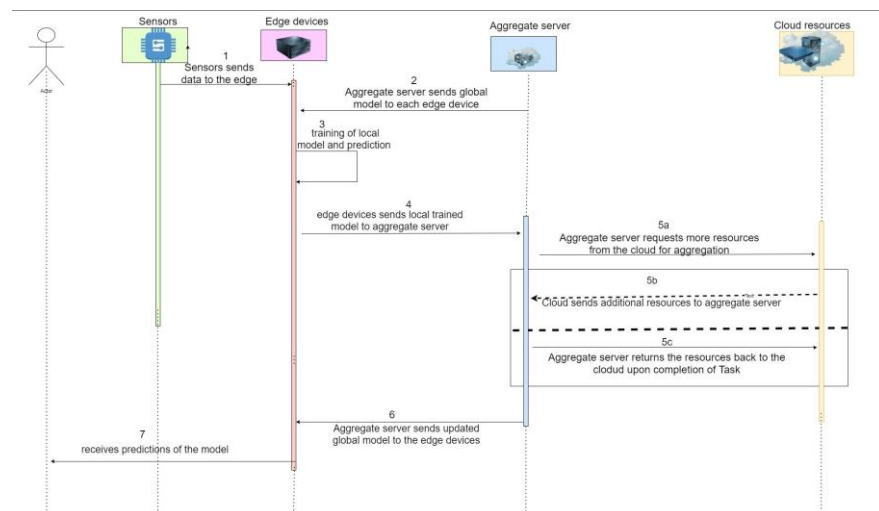


Figure 14: The Federated Learning sequence

2.2.4.8 Gaussian Naïve Base (NB) Classifiers

[146] discussed that Binary relevance breaks down the multi-class dataset into several independent binary variables such that one variable is in one label.

According to [147], the classifier chain Gaussian NB equally disintegrates the multi-class dataset into many independent variables. Still, it recognises the dependent variable correlations, which are an enhancement over the Binary relevance Gaussian NB model. The authors in [147] discuss that the Label powerset Gaussian NB transform the multi-label dataset into many multi-classes single-label classification problem. The Gaussian Naïve Bayes is implemented from the Naive Bayes theorem. Equation 4 depicts the Naive Bayes theorem.

$$P(H|K) = \frac{P(K|H) \times P(H)}{P(K)} \dots\dots\dots(4)$$

H is the hypothesis, and K is the evidence [147].

2.2.4.9 Modelling of Federated Learning Smart Farm Network

According to the authors in [149], by increasing the buffer size for Unspecified BitRate (UBR) or available bit rate (ABR) traffic, the random loss for a high bandwidth-delay product network can be reduced since increasing the buffer size reduces the loss probability for the end-to-end link. This loss is lower than the inverse square of the bandwidth-delay product. The authors in [150] discuss that the buffer regulation algorithm can achieve stability and ensure good performance in a cellular network experiencing bandwidth congestion. They acknowledge that the Prop rate algorithm can achieve a low computational overhead compared with other algorithms such as CUBIX, Verus, and bandwidth-delay product-based approach algorithms in a mobile cellular

network. It can be inferred that the Prop rate algorithm can resolve congested uplink problems because it works with a one-way-delay approach.

According to the authors in [151], Federated Learning is the evaluation of models in a decentralised platform where the communication cost is reduced because only the updated weights of the local models are sent to the server. It can be inferred from their research that the Long short-term memory models considered for the federated Learning network have been five times faster in achieving prediction values than the centralised network.

The authors in [152] discuss that the training of the secure Multi-party Computation (SMPC), FL Homomorphic encryption and FL without encryption indicated that the FL homomorphic encryption scenario has been able to achieve convergence in less than one (1) second while the SMPC converged in fifteen (15) seconds and the FL platform without encryption converged at twenty-four (24) seconds. It can be inferred that applying Homomorphic encryption to an FL network improves its convergence.

The authors in [153] discuss that their efforts to reduce the communication cost of the network considered. They introduced the federated sparse compression (FSC) algorithm, which reduced the communication cost of the FL network. It can be inferred from their research that better generalisation and prediction have been achieved by using the CapsNet to train data in edge devices.

The authors in [154] used the content popularity prediction of privacy-preserving (CPPPP) scheme based on federated learning and Wasserstein generative adversarial network (WGAN) to improve the cache hit ratio and resolve the data leakage during model training in an FL platform. It can be inferred that the transmission time using the FL scheme for caching has been reduced.

The authors in [155] has used FL to analyse chest X-ray images, the federated averaging models have been able to classify the infected lungs and healthy lungs from the image datasets. It can be inferred that their proposed model has reduced the bias in the prediction models because it combines all the updated weights and features of the various edge node models.

The authors in [156] discuss that using Federated averaging + CNN + MobileNet models for the classification of breast cancer images has improved the classification accuracy of breast cancer detection using the federated averaging +CNN model. It can be inferred that their model classification results outperformed other centralised network models.

The authors in [157] discuss that they have resolved the slow convergence of Mobile edge nodes using Federated Learning for heterogeneous nodes. Their research results outperformed the existing centralised network in resource usage, learning accuracy, and convergence speed.

It can be inferred from [148,149,150] that increasing the bandwidth within a link increases the bandwidth-delay product and reduces the congestion.

The research of [151,152,153,154,155,156] discusses that using FL platforms improves model convergence, and the FL model outperforms the centralised network.

Figure 15 shows the architecture of the Modelling of the Federated Smart Farm Network.

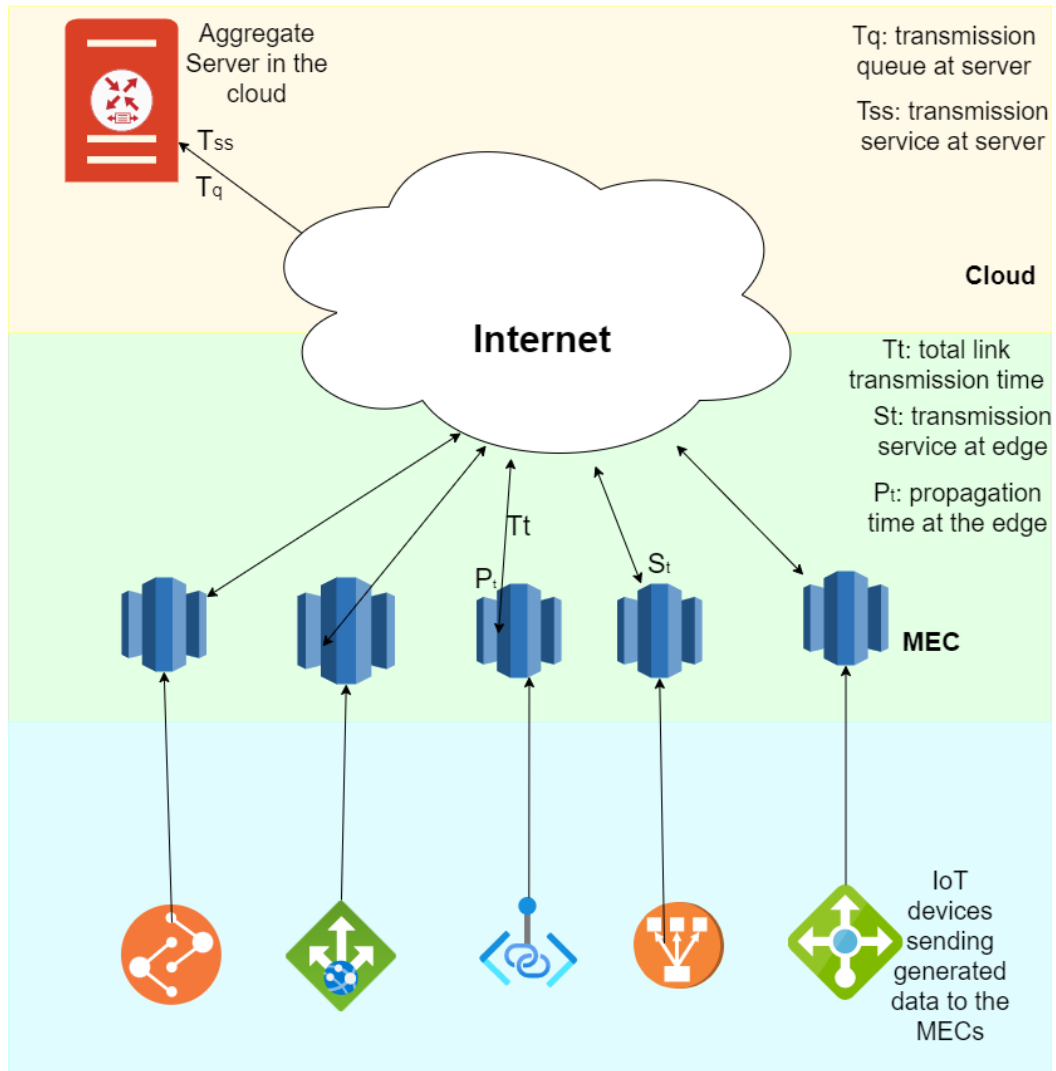


Figure 15: Architecture of the Modelling of Federated Learning smart farm network. The modelling of the federated learning smart farm network is investigated. The server in a federated network receives local model update weights, a network of several queues received at the server ingress. This research evaluates these networks of queues and the impact of the network of queues on the Federated Learning smart farm network. The Bandwidth delay product (BDP) is the product of the bandwidth capacity and the round-trip time taken by the queue within the link. A Testbed has been set up, allowing the researcher to capture several data such as the network's Traffic speed (TS) and round-trip time (RTT). This dataset has been used to model the Smart farm network using the federated Averaging algorithm. The performance metric investigated is the smart farm network's bandwidth-delay product (BDP). The performance of the aggregate, base and

classical machine learning models for the bandwidth-delay product (BDP) has been investigated, and the predictions of the combined and classical models have been analysed.

[148] discuss that the Bandwidth delay product-constrained application protocol (BDP-CoAP) algorithm can handle throughput and minimize the retransmission rate better than the constrained application protocol (CoAP) and congestion control algorithm (CoCoA+) algorithm. It invariably eradicates unresponsive background traffic and ensures stability within a dynamic traffic network.

The testbed setup using the GNS3 emulator was used to evaluate the diverse traffic within the network. The BDP has been analysed using the federated learning edge node models, and the aggregate model and a comparison of the combined model and the classical machine learning model have been considered. It can be inferred that the combined model converges faster than the edge node models and the centralised machine learning network model. The higher the BDP is the faster convergence is achieved. The high BDP has impacted the network positively. From the experimentation, it is shown that the Federated Learning global model converge after eight (8) epochs. The classical machine learning model converges after twenty epochs (20), indicating that the Federated Learning model converges faster than the classical centralised machine learning model. An in-depth comparative analysis of the centralised, decentralised and edge node models is investigated using the BDP as a performance metric for a smart farm network.

2.3 Technologies

2.3.1. Sensors used in smart farms

Water stress levels in crop leaves have been detected using sensors [25], which has allowed researchers to investigate how water affects the health of the crops. The EM4325 UHF chip is one of the many chips installed inside these sensors [25].

According to [26], the smart stick sensor transmits soil moisture parameters from the soil to the IoT devices for analysis. The DHT11 and DS18B20 are some of the sensors used for data collection in smart farming. These sensors have enabled researchers to collect data within the smart farm for various analyses.

2.3.2. Unmanned Aerial Vehicles (UAV) in smart farming

Deep Learning has been used for segmenting and identifying vegetation, crop image classification, detection of disease, and weed using sensors and cameras installed on UAVs [27]. Their research discusses that UAVs have used deep learning to predict crop yield and crop counting. Smart farm data has been captured from imaginable heights using UAVs. UAVs can capture and collect smart farm images and animal voice data, but these data processing challenges require further investigation.

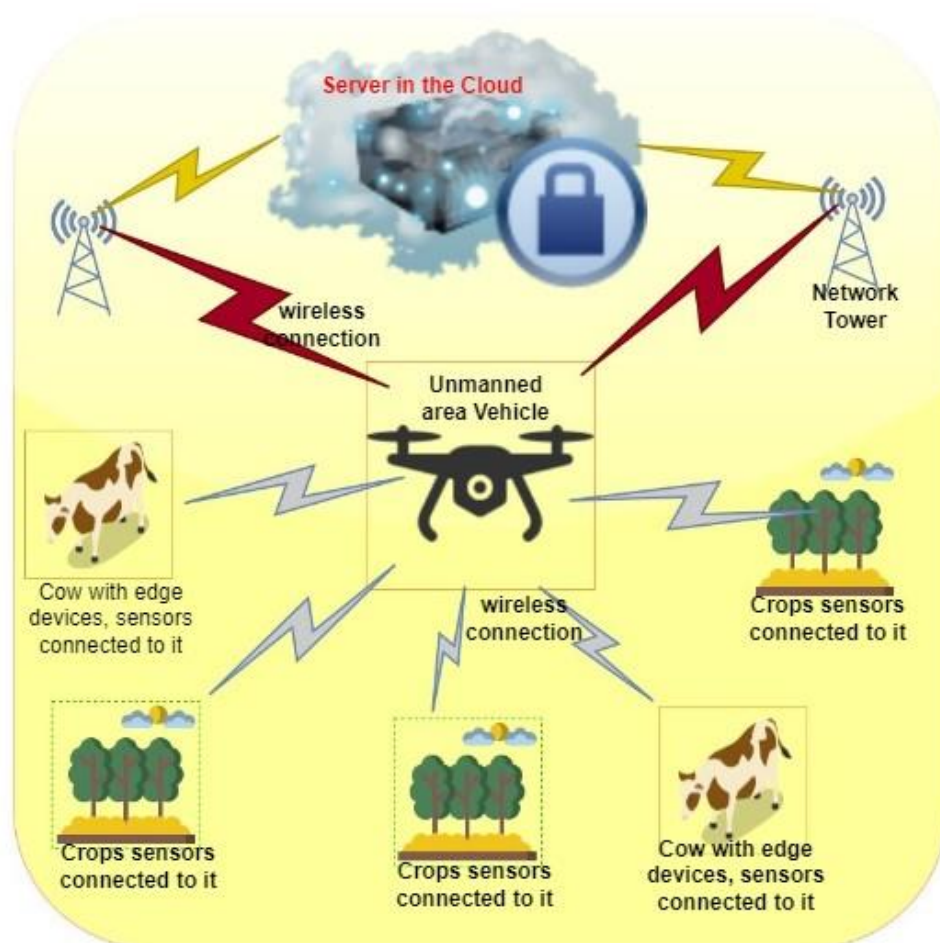


Figure 16. Network architecture for a smart farm using drones.

UAVs enable farmers to collect data from edge nodes, beaconing, shunting of connected nodes, and localization of cluster heads. Some UAVs have developed 3D models using the images captured from the analysis and weed mapping, yield estimation, disease detection, crop spraying, and crop growth monitoring. Figure 16 depicts the network architecture of a UAV collecting data from sensors installed within the smart farm. The authors in [28] discuss that UAVs can be deployed to a smart farm to resolve transmission connectivity challenges using the LoRa protocol. The UAVs can connect the edge nodes and base station, but their network design has not been implemented for heterogeneous edge nodes. The authors in [29] discuss the relationship between the features of images of the thermal and spectra data from a smart farm. It can be inferred from their work that the UAV's flight height impacts the accuracy of the crop images collected. The Red, Blue, Green and near-infrared (RGB_NIR) bands achieved optimal accuracy. They list the Agdrone, which can fly at an altitude of 400 feet and cover 600-800 acres in 60 minutes, and the DJI matric 100 with double battery capacity with 40 minutes additional flight duration compared with other UAVs. It has GPS incorporated within it. Some UAVs, such as the Agras MG-1-DJI, can carry 10KG of liquid over an area of 4000 – 6000 m², the DJI T600 can capture 4K video images and the EBEE SQ is used for early monitoring of crops to maturity stage, Lancaster 5 precision Hawk is used for capturing climatic parameters, SOLO AGCO is used for precision image capturing. It can be deduced that UAVs have enhanced smart farming by capturing high-precision images, weed spraying and effective monitoring of smart farms.

2.3.3. Internet of Things

The authors in [30] discuss that incorporating IoT in smart farms enables the exchange of information between the sensors and the devices. It can be deduced that intelligent IoT devices reduce crop loss and maximise the farmer's profits annually, improve the quality

of the crops, boost farm harvest, and reduce the risk of pesticide poisoning humans and animals through the crops. Still, they did not mention how IoT can protect farm crops during adverse weather conditions. Their research iterated that IoT deployment in smart farms will scare away wild animals and birds using low-frequency ultrasound devices within the farm.

IoT implementation enables farmers to detect infestation on their crops [31] through image processing techniques, but how their model will handle new disease outbreaks has not been discussed. MapReduce and Hadoop IT framework, big data, and wireless sensor architecture can affect image processing performance within a smart farm network [32]. It can be deduced from [30, 31, 32] that the deployment of IoT within a smart farm can boost crop quality and quantity, but the data security concerns such as data validity, integrity, and trust still exist, recovery, data reliability, access control, data encryption, data attacks and data prediction. This research addresses the data security concerns by building models using Federated learning algorithms which provide data privacy during the training of the models because the algorithm allows for data evaluation without the server having access to the raw data but uses only the updated weights from the edge nodes to aggregate the combined model.

2.3.4. Artificial intelligence

According to [35], the movement and location of farm animals within the farm can be monitored using AI. The data captured helps to understand the pattern of the animal's movement using machine learning models, which enable the farmer to know if an attacker has entered the farm vicinity. The authors in [36] discuss that federated learning (FL) models have been used to evaluate unbalanced, non-Independent Identical distributed (non-IIDD) data using mini batches. This technique has reduced network

communication costs. FL models send their edge nodes' updated weight to the server for aggregation without sharing their raw data with the server.

Deploying IoT, UAVs, and Robots has a high cost to implement in a smart farm, while sensors are cheap to purchase. Some limitations identified in using IoT, UAVs, Robots, and sensors within a smart farm are shown in Table 2

Table 2: Comparing Sensors, UAVs, IoT & Robots in Smart Farms [84].

Properties	[sensors]	[UAVs]	[IoT]	[Robots]
High transmission speed	N	Y [28,84]	N/A	Y [32]
Provide connectivity where no internet is available	N	Y [28]	N	N
Cover a long range of distance for data transmission	N	Y [28]	N/A	N/A
Mobility within the farm	Y [84]	Y [35]	N/A	Y [35]
High processing power	N	N	N/A	Y [32]
Analyze data aggregate	N	N	N/A	Y [32]
High security in the transmission of data	N	Y [8,84]	Y [31]	N/A
Run out of power over time.	Y [8,84]	Y [35]	N/A	N/A
The psychological effect on Livestock	Y [8,84]	Y [8,84]	N/A	N/A
Low cost of deployment	Y [8,84]	N	N	N

Yes = Y, No = N, N/A = Not Applicable.

2.3.5 Fog computing

A distributed computing technique that empowers network devices at various levels of network hierarchy with storage and computational capacities is referred to as Fog. The Fog devices are equipped with intelligence, enabling them to determine the devices that require resources from the cloud to meet networking demands for real-time reduced latency services ranging from smart traffic monitoring, live streaming, e-health applications, and smart farming analytics [37].

The authors in [38] discussed that a Fog computing system with centralised network architecture experiences low latency and improved performance when connected with sensors.

Fog computing has been defined by [39] as the computing technology linking IoT devices and the cloud. It provides computing, inter-networking, management and storage of data on the network nodes. They discussed that it had enabled computing systems to monitor, measure, analyze, process, control, and help IoT devices in speedy decision-making within the network. Figure 17 shows a fog computing network with identified functions such as monitoring, measuring, storage, communication, and industries where it is used, namely, agriculture, oil & gas, construction, finance, etc.

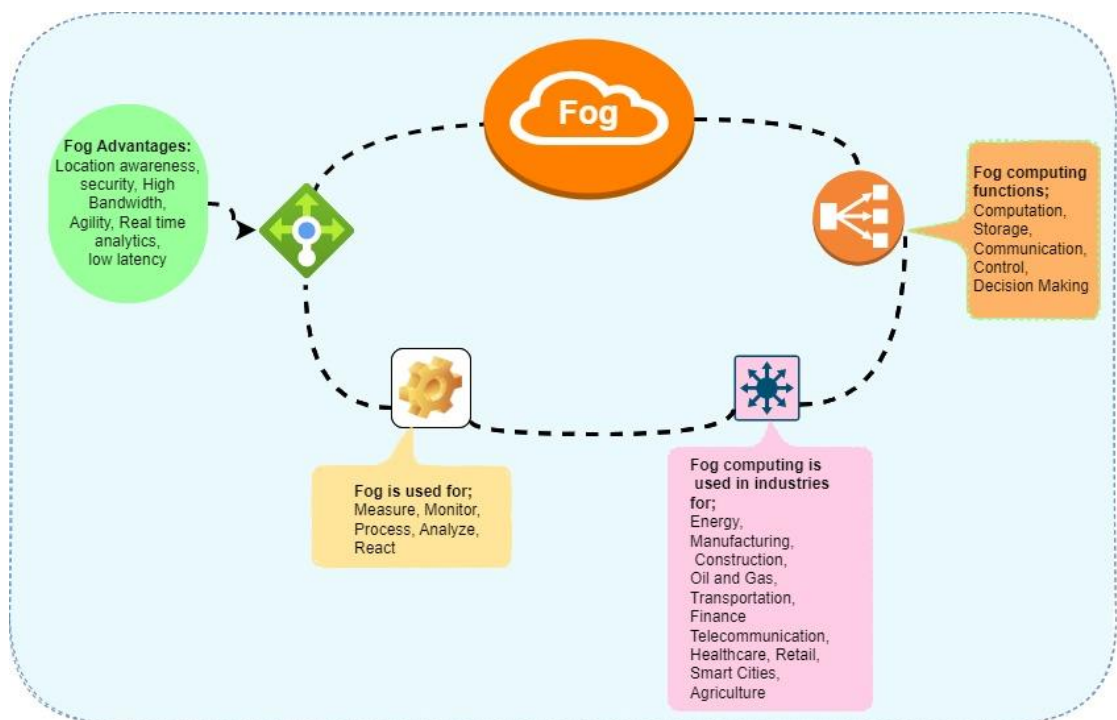


Figure 17: Fog computing

Latency issues can be reduced by replicating network activities such as data analysis and monitoring close to the IoT devices or client [40].

Some devices require a lot of resources to transmit their data to the cloud and experience high energy consumption during the transmission. This energy consumption can be minimised with a fog server [41]. Figure 17 shows a Fog computing architecture.

Many IoT devices used in enterprise networks have challenging security issues. The introduction of fog servers (Mist servers) has been used to monitor neighbouring gadgets' security conditions and minimise security concerns [42].

The Haze server has been used as a fog server to resolve time-sensitive applications and minimise idleness and inertia within the network. It has been used in different sectors, such as smart city deployment, smart Grid systems, and autonomous vehicles. Still, it will be good to apply it to heterogeneous IoT smart farm networks [43].

There have been many developments in computing technologies, such as cloud computing, the Internet of Things, mobile edge computing, and fog computing. However, a recent paradigm of Dew computing synchronises two Dew servers located at different remote sites without an internet connection. Edge devices can connect via the cloud master copy repository within the dew server and browse local online pages. The dew domain scheme and redirection enable remote mapping connectivity between dew servers [44].

The introduction of fog computing to IoT devices within various networks has greatly improved IoT-based applications' efficiency. The fog brings the computing resources close to the IoT devices and reduces the energy requirement of the IoT devices. However, when the computational resources at the fog nodes are limited, the tasks are offloaded to the cloud, and determining the time to offload the task is of great concern [45].

2.3.6 Edge computing

Edge computing is a platform where data computation is executed close to the data source. This minimizes the latency within the network and improves the data processing time due to the proximity of the data source to the data computation. With edge computing, the data are not sent to the cloud for processing. This lowers latency and improves data privacy within a network [46].

It has been discussed in [47] that dataset analysis and evaluation can be executed within proximity to the dataset location using edge computing technologies. Their research has mentioned that bandwidth usage can be lowered drastically through edge technologies where the dataset is not sent to the remote cloud. They further stated that this technology creates additional hops within the network, and edge computing network resources can be unreliable sometimes.

According to the authors in [48], Edge computing minimizes security concerns because the datasets are close to the data source, which enables it to satisfy geographical regulations.

The authors in [49] discussed that edge computing has been used for network computation. They used it to minimize the computing and network resources for migrating data applications from the cloud to the edge in real time.

The authors in [50] discussed that edge computing improves real-time computational processes. Edge computing technology minimizes the network and computing resources needed to move the data to the cloud for analysis.

2.3.7 Multi-access Edge Computing (MEC)

The authors in [51,52] discussed that the storage and computing capacity has been reduced for computational operations due to Mobile edge computing devices, which have limited capacity compared with the cloud. They suggest that a collaborative edge-local-cloud technology should be used to improve users' Quality of experience (QoE) in MEC.

The authors in [53] discussed that mobile edge cloud computing has been used to resolve latency challenges in Audio and visual reality (AR/VR), autonomous driving, and electronic health systems. However, the MEC resources seem less effective than the cloud resources. However, their paper illustrates that the MEC, mobile devices and cloud

collaboration will address the high latency challenges. Still, the heterogeneous architecture of the collaboration affects the performance of the implementation. Clients or users in a network experience delays when they are queued to use or share resources from the cloud or edge server designated to provide resources for their operation. The authors in [54] proposed the stochastic user allocation algorithm to help users or client nodes access these resources for efficient performance by reducing queue delays and latency costs since their proposed optimization algorithm assigns users for provisioning without asking for arrival or departure details.

Resource allocation in edge computing is an area of concern. According to [55], using their proposed destination-oriented directed Acyclic graph (DODAG) algorithm approach, they were able to decide the optimal node for each step and task flow. Their proposed algorithm has been able to address the subsystem's quality-of-service (QoS) requirements. It has been able to predict task completion and task arrival time for datasets across multiple devices.

It has been discussed in [56] that edge computing is an abstract pattern of computing that has moved cloud computing resources to the internetworked edge devices. This has brought network resources close to the end users. This computing paradigm has reduced latency and bandwidth usage within the networks. Privacy and security concerns have been minimized with this technology.

According to the authors in [57], the time delay and energy consumption cost reduction within a MEC network can be achieved by offloading part of the task to the mobile edge device and the other part offloading to the MEC server for processing. They believe prioritizing the high-priority task to the MEC server, which has a high computing capacity compared with the mobile edge device, will reduce the computing delay within the network.

The authors in [58] discussed that a profit maximization multi-round auction (PMMRA) technique has increased the number of users served by a MEC since their suggested algorithm has gotten more capacity for resources within an MEC network.

The Robust-oriented edge Application deployment (READ) has been used to maximize the MEC servers' coverage and equally minimize the latency within the network for a constrained optimization scenario where they proved the NP-hardness, for which the approximation ratio is better than $K=2$, which is a constant, irrespective of the number of servers considered [59].

The decentralised learning-based edge node grouping algorithm has been used to reduce backhaul congestion and latency and improve edge capacity. This is achieved by the multiple edge nodes cooperatively serving one user and giving the user a better experience. Their solution has achieved 96.99% of the Oracle benchmark [60].

2.3.8 Cloud computing

Joseph Carl Robnett Licklider invented the cloud computing paradigm in the 1960s while working on the ARPANET project. Cloud computing terminology grew tremendously in 2006 during an industrial conference where Eric Schmidt, the Google CEO, mentioned the terminology. Subsequently, the AWS S3, a product of Amazon for providing web services, influenced the growth of the awareness of cloud computing in the same year [61].

CompuServe 1983 introduced file storage on disks, and in 1994, AT&T introduced the first online storage for personal and corporate customers into the market. However, users more readily accepted cloud computing in 2007 when Dropbox was introduced via AWS. Other Technological companies released their cloud computing brands, Google released its version in 2008, and Azure, a product of Microsoft, was released in 2010. In 2011, IBM launched its cloud platform. Cloud computing has evolved from file storage to a technology that offers file computation [61].

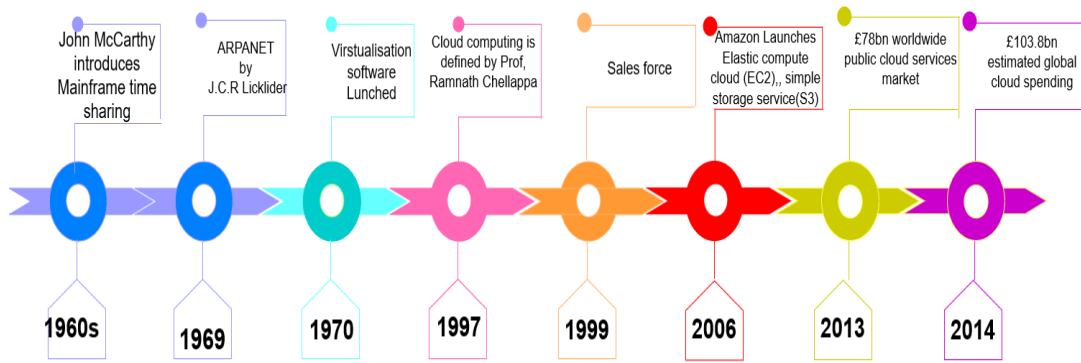


Figure 18: Evolution of the cloud computing

Some of the services offered using cloud technologies are Platform as a service (PaaS), Software as a service (SaaS) and Infrastructure as a service (IaaS) [62]. Figure 18 shows the evolution of cloud computing since the 1960s.

SaaS offers customers all the software they need for their operations. They do not have to buy or install any software or hardware. They do not worry about maintenance or upgrade of the software; they only pay for what they use, and the software comes with the latest upgraded versions. This service gives everyone the affordability to use software that, in the past, could be accessed by large enterprise networks within the industry. The infrastructure as a service provides All the hardware for the customers, such as servers, storage, and networking. However, the developers must install the software they require for their operations, such as database servers, web servers, operating systems, etc.

The PaaS is a cloud service for developers to run, develop and manage their applications in the cloud. They do not require servers, storage, networking, operating systems, middleware, or databases because the service (PaaS) runs on top of the IaaS. Examples are Heroku, Microsoft Azure App Service, Google App engine, etc.

Cloud computing has enabled their customers to offer improved performance services, fulfilment of service level agreements, reduction in the cost of its operations, better

security within their network provisioning, and collocation services within their data centres.

Many research publications exist on cloud computing, cloud adoption [63], cloud application [64], cloud security [65], cloud architecture [66], cloud provisioning [67], and cloud orchestration [62].

A publication by [61] 2010 highlighted the technique of migrating an Information Technology enterprise network to the cloud, such as IaaS. We see further work by [68], which discussed using cloud technologies for video surveillance to improve the surveillance images for better analysis. It can be inferred from the existing publications that using the cloud has reduced the cost of operations and improved real-time data, voice, and video processing.

2.4. How climate change affects smart farms

The climate affects crop colouration [69]. The authors discuss that reduced daily temperature can cause brown patches, indicating the effect of climate change on crops. Their research mentions the smart surface sensing system for monitoring crop vegetation indices and Leaf Area Index (LAI). Still, this solution cannot be used to monitor multiple sites simultaneously.

The Latitude of locations can influence some crops' flowering time [70], as seen in winter wheat growing. They affirmed that higher longitude positively influences the winter wheat crop's flowering. Still, their research did not inform us of the effect of excess water conditions within the farm on the winter wheat crop. It can be inferred that climate affects the evapotranspiration rate within the farm since crops lose more water in hot weather.

2.5 Challenges and Issues of smart farming

The authors in [72] have expressed concerns about the communication protocol used for interaction within the smart farms. These protocols were effective for only short-distance coverage areas.

It can be inferred from [21, 22] that the IoT devices attached to the farm animals were battery-operated, causing the data transmission to last for a few hours.

Smart farm networks currently experience latency, data privacy, reliability, Quality of service and security [17]. Climatic parameter detection is another challenge smart farm networks face [30]. Smart farms network transmit data from the edge nodes to the server and from node to node [36]. A high communication cost is involved in the transmission link, and researchers need to investigate this challenge further. [32] narrates the security concerns about data transmission within a smart farm network. Disease infestation in a farm can adversely affect the harvest. These diseases have been detected early to reduce poor harvest using machine learning models [20].

Recent publications show crops lose water via evapotranspiration due to high temperatures and adverse climatic conditions [74, 75]. Tables 3–4 depict the identified challenges in smart farm networks. These include disease detection, animal behavioural patterns, and leaf water loss. The identified gaps provide opportunities for future research.

Table 3: Comparing IoT challenges in smart Agriculture (Part 1) [84]

Properties	[1]	[14]	[15]	[23]	[24]	[25]
Control actuators	Y	N	N	N	N	N
Detection of weather conditions	N	Y	N	N	N	N
Preventive measures using IoT	Y	N	Y	Y	N	Y
Architecture	N	N	N	N	N	Y
Reduce communication Cost	N	N	N	N	Y	N
Quality of service	N	N	N	Y	Y	Y
Lightweight encryption for IIoT	Y	N	N	N	N	N
Failure detection	Y	N	N	N	N	N
prediction for IIoT	Y	N	N	N	N	N
Data reliability	Y	N	N	N	N	N
Access control in IIoT	Y	N	N	N	N	N
Cyber-attacks in IIoT	Y	N	N	N	N	N
Management of IIoT designs and software	Y	N	N	N	N	N
Trust in IIoT	Y	N	N	N	N	N

Yes = Y, No = N, N/A = Not Applicable.

Table 4: Comparing IoT challenges in smart Agriculture (Part 2) [84]

Properties	[4]	[28]	[5]	[6]	[7]	[8]
noise filtering capacity	Y	N	N	N	N	N
faster detection rate for crop disease	Y	Y	N	N	N	N
Reduced the time of diagnosis of animal illness.	Y	Y	N	N	N	N
Enhanced Data Transmission	N	N	Y	N	N	N
monitor the movement of the animals within and outside the farm	N	N	Y	Y	N	N
determine the animals' attitude and behavioural pattern	N	N	Y	Y	N	N
Monitor health changes among the animals.	N	N	Y	Y	N	N
Colour, Shape from the 3D sensor	N/A	N/A	N/A	N/A	Y	N/A

Yes=Y, No=N, N/A= Not Applicable

2.6 Cloud-based IoT smart farming

AI solutions have been used to improve farming strategies [80]. The GeoFarmer solution has enabled farmers to share their experiences with other stakeholders, and the solution comprises Interactive Voice Response (IVR), which allows the farmers to interact with agricultural partners using smartphones. The solution enables Agricultural experts to

support farmers through the application. Farmers can share information with other farmers, but uneducated farmers cannot use the solution.

Table 5: Comparing the Advantages of IoT Cloud-based Smart Agriculture in Related Works [84]

Properties	[4]	[28]	[5]	[6]	[7]	[8]	[15]	[23]	[24]	[26]
Better segmentation of crop spot edges	Y	N	N	N	N	N	N	N	N	N
identification and separation of plant disease in fast computational time	N	N	N	N	N	N	Y	N	N	N
The algorithm automatically identifies the health and welfare of animals.	N	Y	N	N	N	N	N	N	N	N
A wireless neck collar connected to the farm animals for data transmission	N	N	Y	N	Y	N	N	N	N	N
Sea-lice counting and crowding control in fish farming to enhance fish farming.	N	N	N	Y	N	N	N	N	N	N
RGB-D sensor used for harvesting sweet pepper by cutting the peduncle	N	N	N	N	Y	N	N	N	N	N
Monte Carlo simulation was used to determine the best harvest age of a coconut.	N	N	N	N	N	Y	N	N	N	N
Evaluation of the near-surface air temperature data sets from the ERA-Interim (ERA-I)	N	N	N	N	N	N	N	N	N	Y
AI in edge computing used to monitor the movement and the location of the animals on a farm	N	N	N	N	N	N	N	Y	N	N
FL is used to handle user equipment and edge nodes for unbalanced and non-independent Identical Distributed (non-IDD) data successfully.	N	N	N	N	N	N	N	N	Y	N

Yes=Y, No=N, N/A= Not Applicable

[81] discusses that animals on the farm defecate as they move within the farm, and the animals carry these excretes with their hoof, causing soil compaction and pollution of the water table via the soil, which can cause E. coli disease within the farm vicinity. It can be

inferred that static animal feeders are more suitable for reducing disease outbreaks within the farm.

MapReduce, Machine learning models can be used for smart farm data analysis [82], and the data encryption approach reduces the data size for transmission. They further narrate that challenges exist, such as security, data privacy, hardware heterogeneity, service level agreement among smart farming industry players, network latency, and data reliability. Table 5 shows the advantages of the cloud-based smart farming methodology and IoT. It can be inferred that AI solutions have enabled crop monitoring within the smart farms to take minimal time, and training of smart farm data can be done using decentralised and centralised machine learning models.

2.7 Application of machine learning to Agriculture

Diseases, old agricultural practices, poor farm management strategies, and lack of AI know-how for early crop disease identification are reasons for poor quality crops and harvest [77,78]. AI solutions have been used to determine rice dimensions, according to [79]. Their research has an error rate of one and a half per cent (1.5%) compared to the manual technique. [79] discusses that by using smart IoT mat devices, farmers can monitor the weights of their pigs. These devices allow the farmers to capture the pig's gestation period and monitor the health of the pigs and their piglets within the womb, this solution has been useful for monitoring pregnant pigs to avoid miscarriages, but they have not tested this AI solution on other animals to ascertain its tolerance by other farm animals. The application of ML models has allowed farmers to monitor their farm actuators and data remotely. With the Application Programming Interface (API) deployment within the smart farm network, soil moisture, climatic parameters, and crop growth rate can be captured using the architecture. A comparison of the federated, unsupervised and supervised machine learning methodology is shown in Table 6.

Table 6: Comparing Federated, Unsupervised & Supervised Machine Learning Techniques in a Smart Farm [84].

	Supervised Learning [77]	Unsupervised learning [78]	Federated Learning [138,139]
Disease detection	Y	N	Y
Detection of Ripening of fruits	Y	N	Y
Tiny grain size calculation	Y	Y	Y
Monitoring of animal gestation period, weight, gait	Y	N/A	Y
Usage in edge devices	Y	Y	Y
Training of dataset remotely	N	N	Y
Reduction of network latency	N	N	Y
Data privacy and security	N	N	Y

Yes=Y, No=N, N/A= Not Applicable

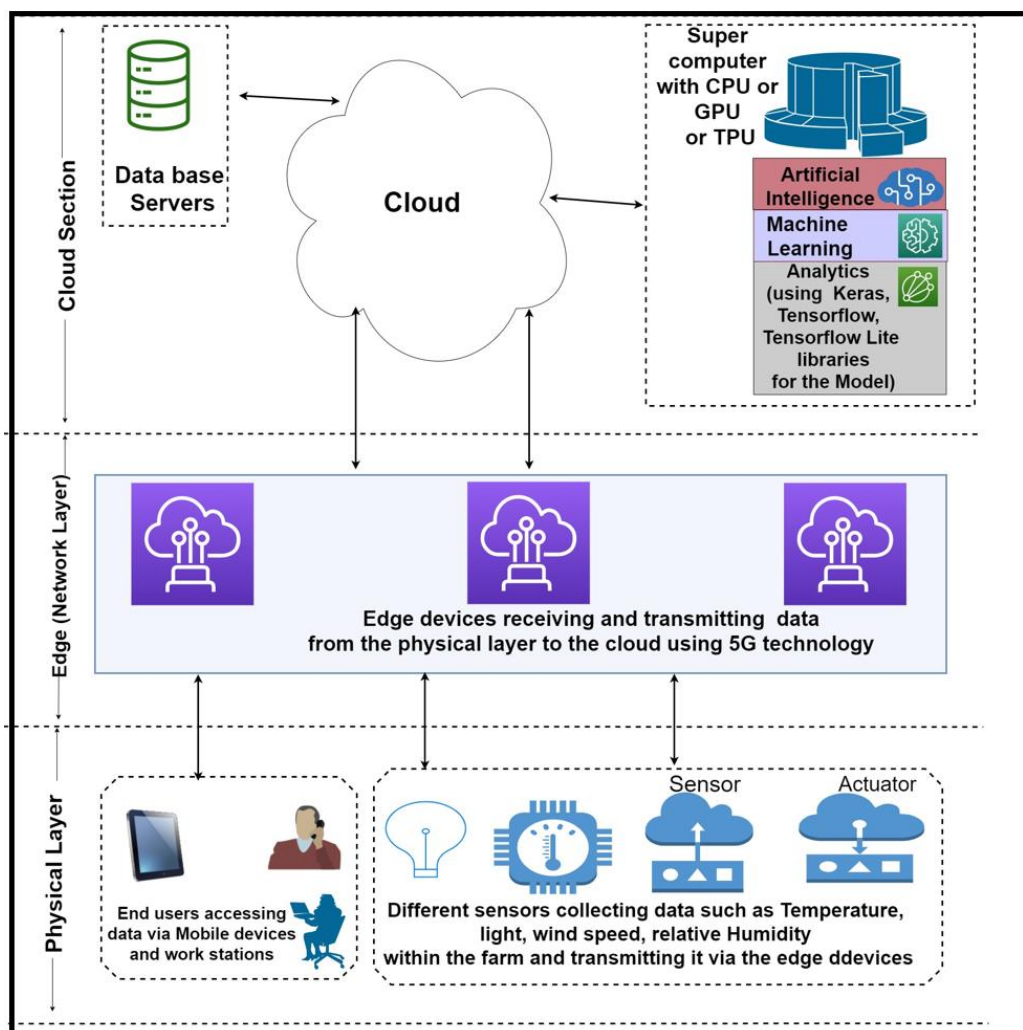


Figure. 19: Cloud-based IoT Architecture for agriculture

Figure 19 shows a cloud-based ML architecture that can be deployed using Keras, TensorFlow, and TensorFlow lite libraries for centralised or decentralised smart farm network data analysis.

2.8. Discussion

AI technologies have been used to boost smart farm production. These include using UAVs, monitoring farm animals' health and movement, and analysing farm images ranging from crops and farmlands to improve irrigation systems within the farm. The application of technology to agriculture has boosted farm produce and empowered the farmers and their stakeholders. Still, these laudable merits have also brought some challenges, such as security and privacy concerns, data governance, and lack of change in culture by the stakeholders to accept the IoT system innovation [8,31, 84].

The technological devices used in smart farms have limited computational power and cannot analyse the volume of data collected within the farm. So farm devices connected to livestock to monitor their movement and body parameters use batteries, which run out of power within hours of usage. Using the devices connected to this livestock psychologically affects these animals [8]. Since Technology in agriculture generates enormous volumes of data, analysing these data is of concern because the edge devices deployed in the farms cannot analyse the Big data generated [32]. Transferring these data to the cloud gives the stakeholders security concerns because intruders can intercept the data and cause a privacy breach within the smart farm network [31]. The farms are located outside the metropolitan environment where internet infrastructure is poor, which causes high transmission and connectivity issues [28,32]. There is a need to encrypt the IoT data in the farms, and the prediction of outputs of farm produce cannot be achieved using machine learning models because the farm devices have low resource

capacities [1]. It can be inferred from [23,24,25] that the quality of service within smart farm networks is still a challenge.

Research is ongoing on improving the speedy detection of animals' health while they are ill and faster disease detection using technology [4,28]. When the livestock is pregnant, there is great concern about their behavioural pattern, which can influence how they are fed and treated within the farm. Current research publications are still addressing these concerns [5,6].

Chapter 3 Data and Data Preprocessing

3.1 Hydroponic Dataset from the University of Peloponnese.

The University of Peloponnese (UP) in Greece studied hydroponic systems, specifically Floating, Aggregate, Aeroponic, and Nutrient Film Techniques (NFT). Data have been collected from these four systems, comprising various features related to the growth of Onion crops [158]. The dataset includes the following numerical features: days after transplant, temperature ($^{\circ}\text{C}$), water consumption (Litres), number of leaves (NL), nitrogen content (mg/g), phosphorus content (mg/g), potassium content (mg/g), calcium content (mg/g), magnesium content (mg/g), sulphur content (mg/g), sodium content (mg/g), and onion bulb diameter (mm). The dataset consists of 64 rows and 12 columns.

The dataset was pre-processed using the scikit-learn library in Python. Missing values have been identified, and normalization techniques have been applied to handle these missing values. In addition, the dataset has been cleaned to address inconsistencies and remove duplicate entries.

The scikit-learn library has been employed to split the dataset into training and testing samples. The training samples accounted for Eighty per cent (80%) of the total population, while the remaining Twenty per cent (20%) was allocated for testing. The developed models have utilised the training samples to learn the patterns and relationships between the dataset's independent and dependent variables. The testing samples have been used to evaluate how well the model generalised to unseen data, assessing its performance in real-world scenarios. The K-Fold cross-validation from the scikit library has been used to evaluate the dataset to prevent overfitting of the model.

The performance of the trained model has been assessed based on its ability to generalise to unseen test samples and forecast its performance on real-world datasets. This

evaluation has provided insights into the model's effectiveness in handling unseen data and making accurate predictions.

3.2 Amaranthus Viridis Dataset

Amaranthus Viridis crops have been grown in the floating hydroponic smart farm at London South Bank University. The dataset collected from this farm includes images of the crop's leaves and numerical variables related to climatic parameters and crop growth rate. The dataset consists of 62 rows and seven columns.

The images of the leaves were captured using a Raspberry Pi camera. These images were pre-processed and classified into two categories: healthy and infected. The classification was based on the colouration of the leaves, with completely green leaves considered healthy and leaves with yellow colouration marked as infected. One Thousand and fifty-three (1053) image dataset was analysed. The Amaranthus Viridis images have been split into Seven Hundred and Twenty-Nine (729) for training of Healthy images, One Hundred and Sixty-Seven (167) for training of infected images, One Hundred and Forty-Five (145) for testing of Healthy images and Twenty (20) for testing of infected images.

For the image data, the scikit-learn library in Python was used to read and process the images. The images were resized to 256x256 pixels to ensure uniformity in size across the dataset. Normalization was applied by dividing the pixel values by 255, which helped improve the model's convergence during training. To enhance training, the images were flattened as a preprocessing technique. Finally, the images were saved in the Joint Photographic Expert Group (JPEG) format.

The dataset also includes numerical data related to the crop's growth rate. This data consists of 62 rows and seven columns. Pre-processing was performed using the scikit-

learn library to handle missing values and check for inconsistencies and duplications within the dataset.

Cross-validation techniques from the scikit-learn library were employed during the data training. Specifically, k-fold cross-validation was utilized, where the dataset was split into K (in this case, K=5) folds. The dataset has been split using the Scikit Learn Python library for training and testing. Eighty per cent (80%) has been used for training, and Twenty per cent (20%) has been used for testing. This process was repeated for different training and testing set combinations to ensure a robust evaluation of the model's performance.

The dataset was effectively prepared for training and testing the models by employing cross-validation and pre-processing techniques, enabling accurate analysis and predictions of the Amaranthus Viridis crop's growth rate.

3.3 Multi-labelled Smart Farm Dataset Used for Federated Learning Classification

The dataset was collected from [145]. It consists of independent variables such as temperature, humidity, the potential of hydrogen (pH), and rainfall, while the dependent variables are rice, maize, and chickpea.

The Python scikit-learn library has been used for the preprocessing of the dataset. Missing values within the dataset were addressed through normalization techniques. Data cleaning procedures were applied to handle duplicates and inconsistencies within the dataset.

One-hot encoding was performed using the Python library for the dependent variables (rice, maize, and chickpea). This encoding assigns numerical values to each category, with rice being assigned a value of 1, maize a value of 2, and chickpea a value of 3.

The dataset was then split into Eighty per cent (80%) for training and Twenty per cent (20%) for testing using the scikit-learn library. This split aims to train the model on most of the dataset and evaluate its performance on the remaining portion.

To further enhance the model's performance and prevent overfitting, cross-validation techniques from the scikit-learn library were applied. The value of K for the multi-labelled smart farming dataset classification model was set to 5. This means that the dataset was divided into five subsets, and the model was trained and tested on different combinations of these subsets. This approach provides insight into how well the model can generalize to new, unseen data.

By utilising these preprocessing techniques, including missing value handling, data cleaning, one-hot encoding, dataset splitting, and cross-validation, the dataset was prepared for training and testing a model that can effectively classify the crops (rice, maize, and chickpea) based on the given independent variables (temperature, humidity, pH, and rainfall).

3.4 Dataset for Modelling of Federated Learning Smart farm network

The dataset for the modelling of the Federated Learning smart farm network has been collected through network emulation using the GNS3 networking tool. The independent variables in the dataset include traffic speed and round-trip time (RTT), while the dependent variable is the Bandwidth Delay Product (BDP). Each edge node has a dataset with 144 rows and two columns for the independent variables, while the dependent variable (BDP) has a single column with 144 rows.

To ensure efficient execution during training, the dataset's matrices of independent variables have been reshaped using a Python function to adjust the cross-product matrix of the variables. The scikit learn library in Python has been utilized to split the dataset into

training and testing samples. The training samples comprise Eighty per cent (80%) of the total population, while the remaining Twenty per cent (20%) is allocated for testing.

The developed models leveraged the training samples to learn the patterns and relationships between the independent and dependent variables within the dataset. The testing samples were used to evaluate how well the model generalized to unseen data, assessing its performance in real-world scenarios.

The dataset has been remotely accessed using the Duet platform, facilitating connectivity between the edge nodes and the server [159]. The Federated Learning evaluation has been conducted without the global model having direct access to the raw dataset located at the edge nodes. Instead, each edge node trained its dataset using a base model and sent the updated weights to the server. The server then aggregated the updated weights from multiple edge nodes and distributed the combined global model back to the edge nodes for further training their local models. This iterative process continued until the model achieved convergence.

The effectiveness of the model's generalisation capabilities has been assessed by evaluating the performance of the trained model on unseen test samples and forecasting its performance on real-world datasets.

3.5 Limitations of the Dataset Used

The datasets from the smart farms are time series datasets, which indicate they have been collected over a time frame. The climatic parameters, the crop height, leaf length, leaf width and the number of days of transplant have been collected daily. The other limitation of the smart farming dataset is that the crops are harvested within a time frame. For instance, the *Amaranthus Viridis* crop has been harvested by week eight (8) of the transplant because the crop has matured for harvesting, leaving the crop beyond eight (8) weeks. It will not be suitable for consumption again. This impacts the volume of

the dataset collected, resulting in the collection of a small volume of the dataset. Some Machine Learning models require a large volume of the dataset for training to determine the pattern within the dataset, so smart farming can generate a small volume of datasets from the cultivation of the crops, especially vegetables, which are harvested within a few months of cultivation.

Chapter 4 Analysis of Hydroponic Systems Data Using Machine Learning Algorithms.

4.1.1 Linear regression

The prediction of a dependent variable from one or several sets of variables can be referred to as Linear regression [109]. Regression calculation depicts the relationship between the labels and independent variables. It can be seen in Equation (5), the outcome and independent variable relationship,

$$H = f(d, c) \dots\dots\dots(5)$$

Where H is the dependent variable, d is the independent variable, and c is an unknown value, as discussed in [109].

Regression can be represented in a single or multivariate format. This is seen in equations (6 & 7), respectively.

$$H = a + Qd + p \dots\dots\dots(6)$$

$$H = a + Q_1d + Q_2d + \dots + Q_nd + p \dots\dots\dots(7)$$

where a and q are coefficients, and p is the error observed in the regression analysis [109, 159]. In this research work, the independent variables include days after transplant (days), Temperature(°C), water consumption (Litres), NL, Nitrogen (mg/g), Phosphorus (mg/g), Potassium (mg/g), Calcium (mg/g), Magnesium (mg/g), Sulphur (mg/g), Sodium (mg/g) while the dependent variable is the Onion Bulb diameter (mm)

4.1.2 XGBoost

The XGBoost algorithm optimises the model's objective function while regularising the model. It is a Gradient boosting algorithm that inserts the negative gradient of the loss function in iterations continuously to achieve optimisation [110].

$$\sum_{i=1}^m L(W_i, B_i) + \frac{1}{2} \sigma \gamma^2 \dots\dots\dots(8)$$

where γ is the output value [110, 158].

The first part is the loss function, and the second part is the regularisation of the tree. The XGBoost Algorithm optimises the model by minimising the loss function to make predictions.

4.1.3 Deep Neural Network

DNN comprises neurons. They have input, hidden and output layers. The outcome of the DNN has been achieved by optimising the bias and weights [111].

$$y_i = \phi(\sum_{t=0}^{n-1} H_t W_{tm} + b) \dots\dots\dots(9)$$

where y_i is the output of the neuron, W_{tm} is the neuron's weights, n is the number of weights, H_t is the neuron's inputs, b is the bias of the neuron, and ϕ (Φ) is the activation function [111, 158].

4.2 Methodology

Nutrient Film Technique (NFT), Aeroponic (AER), Aggregate (AG) and Floating hydroponic systems have been used to grow Onion crops at the Department of Agriculture, University of Peloponnese (UP), Greece. The crops have been grown for Ninety-two (92) days after transplant. The Number of Leaves (NL), water consumption, Temperature, Nitrogen, Phosphorus, Potassium, Calcium, Magnesium, Sulphur and Onion Bulb Diameter (OBD) are the variables obtained from the dataset provided by the university [89].

In the Department of Agriculture, University of Peloponnese (UP), Kalamata, Greece, the onion crop has been planted using the AG, AER, Floating, and NFT hydroponic systems. A

series of raw data have been collected for 92 days after transplant (DAT) from the nursery to the hydroponic systems.

This research aims to use machine learning models to evaluate the hydroponics datasets from the UP, Kalamata, Greece. The model's dataset analysis results will be compared with their baseline results. The independent features are days after transplant (days), Temperature($^{\circ}\text{C}$), water consumption (Litres), Number of Leaves (NL), Nitrogen (mg/g), Phosphorus (mg/g), Potassium (mg/g), Calcium (mg/g), Magnesium (mg/g), Sulphur (mg/g), Sodium (mg/g) while the OBD (mm) is the label.

The temperature has been recorded using Celsius ($^{\circ}\text{C}$), and water intake by the crop has been calculated by taking the dry weight of the Onion and the weight after heating the Onion in the oven. NL was obtained by counting the leaves, and DAT is the recorded days after the crop was transplanted from the nursery.

The data have been trained using Federated split Learning, XGBoost, DNN, and Linear regressor models. Loss during epoch training and Mean Square Error has been considered as performance metrics for each model from the training of the dataset. The dataset has been split into Eighty per cent (80%) for training, and Twenty per cent (20%) for testing the dataset using the Python Scikit learn library. A batch size of 32, an epoch of 100, the relu activation function, and the Adam optimizer have been used to evaluate each hydroponic system using the Deep Neural Network model. The dataset has been normalised using the scikit Learn Standard Scaler library.

4.3 Results

4.3.1 Analysis of Onion Bulb Diameter Using Deep Neural Network

Table 7 depicts the results of evaluating the DNN model on the Floating, NFT, AER, and AG hydroponic systems. Floating, AER, NFT, and AG systems have produced Losses of 0.13, 0.73, 0.13, and 0.34, respectively. It can be inferred that the Floating and NFT hydroponic systems have outperformed the other systems since their Loss values are the lowest at 0.13 for each. This model is learning our dataset very well in the Floating system. Using the DNN model for training, the AG has been the system with the fastest computational time of 25 milliseconds, while the Floating, AER and NFT systems used a computational time of 30 milliseconds, 28 milliseconds, and 27 milliseconds respectively.

Table 7: Deep Neural Network Models Hydroponic systems results for the Onion crop.

	AER	AG	Floating	NFT
Loss	0.73	0.34	0.13	0.13
Mean Squared Error (MSE)	0.73	0.32	0.13	0.13
Test Loss	2.47	4.39	1.77	5.43
Test Mean Squared Error (MSE)	2.47	4.39	1.77	5.43
computational time (milliseconds)	28	25	30	27

The DNN model does not capture the relationship between the actual and predicted OBD.

The Analysis of the four Hydroponic systems datasets from the University of Peloponnese, Kalamata, Greece, which has been Normalized using the Python scikit learn library, generalised very well with the dataset. There have been no overfitting or underfitting indications from the four hydroponic system dataset evaluations, as seen in Figures 20, 21, 22, and 23. The optimal performance has been achieved with the Floating and NFT hydroponic based on the lowest Mean Square error achieved, as shown in Table 7, which should be considered for OBD using the DNN model. It can be inferred that implementing K Fold cross-validation to the DNN model and suitable batch size, using a relu activation function, has enabled the DNN models to generalise properly without learning the noise

& idiosyncrasies of the dataset in learning the data and produced no overfitting when evaluated on the testing dataset which the model has not seen before.

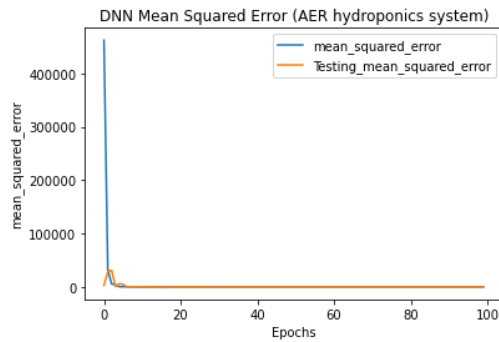


Fig.20: Onion crop MSE from the AER DNN model

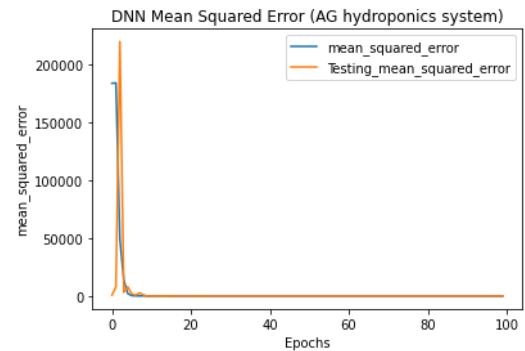


Fig.21: Onion crop MSE from the AG DNN model

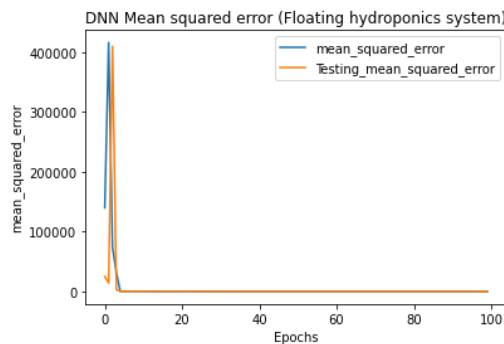


Fig.22: Onion crop MSE from the Floating DNN model

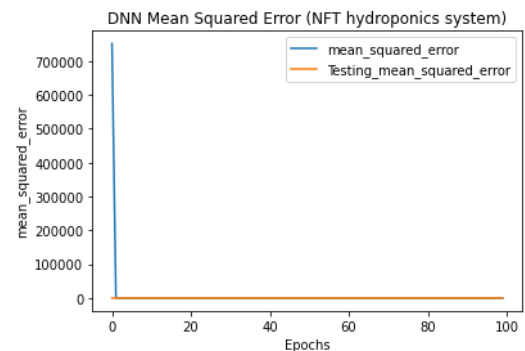


Fig.23: Onion crop MSE from the NFT DNN model

Figures 20, 21, 22 and 23 show the AER, AG, Floating and NFT DNN model Mean Square Error and Testing Means Squared Error, respectively. Implementing the K-Fold cross-validation has enabled the model to generalise very well and reduced overfitting to close to Zero (0) or negligible value, as can be seen from the DNN figures 20, 21, 22 and 23.

4.3.2 Evaluation of the Onion Bulb Diameter (OBD) Using Linear Regression Model

The OBD has been trained using the simple Linear regressor model on the four considered hydroponic systems. Figures 24, 25, 26 and 27 depict the prediction for AER, AG, FL, and NFT hydroponic systems, respectively. OBD (mm) has been considered as the dependent variable, and Sodium (mg/g), Nitrogen (mg/g), Sulphur (mg/g), Number of Leaves (NL), days after transplant (days), Magnesium (mg/g), water consumption (Litres), Potassium

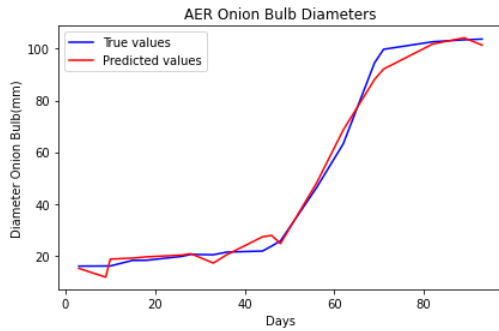


Fig.24: Onion crop Baseline against Linear Regression diameter predictions for AER system

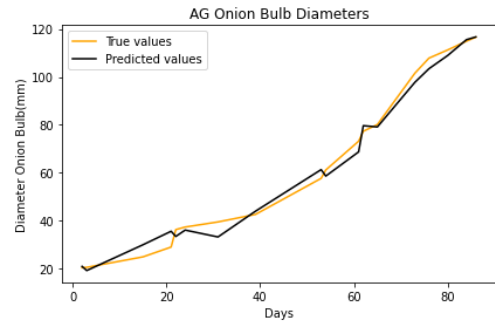


Fig.25: Onion crop Baseline against Linear Regression diameter predictions for AG system

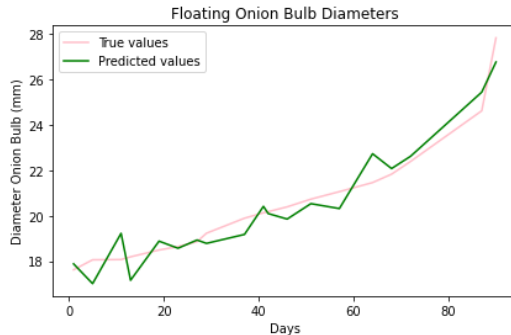


Fig.26: Onion crop Baseline against Linear Regression diameter predictions for the Floating system

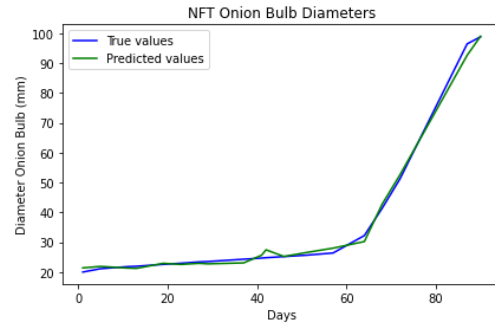


Fig.27: Onion crop Baseline against Linear Regression diameter predictions for NFT system

(mg/g), Temperature($^{\circ}$ C), Calcium (mg/g), and Phosphorus (mg/g) are the independent variables. The farmer can forecast the OBD before planting the crop using the automated system. With this automated system, the farmer can manually predict the OBD for days not recorded by the research team, such as predicting the OBD for the 50th day after the transplant. The predicted OBD using the AER system is 30mm, as shown in Figure 24. Taking the predicted OBD for the 70th day after transplant, as depicted in Figure 24, the OBD value is 90 mm by the automated system. This helps the smart farmer determine his harvest and invariably predict his profit from the cultivation while other conditions remain favourable. The NFT system outperformed the AER, AG, and Floating systems using the linear regression model. The regression model fits very well and generalises the dataset for each system. Still, the NFT is the optimal system using the regressor model

since most of the predicted Onion Bulb diameter values match almost the entire original Onion Bulb diameter values, as shown in Figure 27.

Table: 8: The Linear Regression Results for the Hydroponic systems

	AER	AG	Floating	NFT
Mean Square Error	9.89	14.7	0.47	2.08
R-Square (R²)	0.99	0.98	0.92	0.99

From Table 8, the AER and NFT hydroponic systems have produced the highest Coefficient of the determinant (R²) of 0.99 values, while the floating hydroponic system produced the lowest R² value. The Floating hydroponic system produced the lowest mean square error of 0.47, while the AG system produced the highest MSE values of 14.7.

It has been observed that during the data preprocessing and evaluation of the AER and AG hydroponic dataset from the University of Peloponnese that it has been possible to implement K Fold cross-validation and Scikit Learn standard Scaling for Normalisation of the dataset and to allow the model to generalise very well on the dataset, but a different scenario has been observed for the Floating and NFT hydroponic systems dataset, implementing both K Fold cross-validation and scikit learn standard scaling technique, the dataset Analysis produced negative R-square results and over seven hundred (700) value for the Mean square errors. The Floating and NFT hydroponic system data has been pre-processed and evaluated using only the scikit Learn standard scaler for this analysis, and the MSE results obtained from the analysis are single digit and positive R², this indicates the diversity in the dataset received from the University of Peloponnese, Kalamata, Greece.

4.3.3 Analysis of Onion Bulb Diameter using XGBoost model

The Extreme gradient Boosting model has been used to Evaluate the dataset for each hydroponic system, and the co-efficient of the determinant (R²) has been produced for the AER, Floating, AG, and NFT systems.

Table 9: XGBoost models results for Onion crop grown in the hydroponic system.

	AER	AG	Floating	NFT
MSE	4.43	6.52	1.75	3.13
R² value	0.996	0.994	0.868	0.995
computational time (milliseconds)	750	797	656	844

The dataset has been pre-processed using the scikit Learn standard scaler library and K-Fold cross validation where the K value is Five (K=5) and the Mean Square Error and coefficient of determinant (R^2) values obtained are shown in Table 9. The Floating hydroponic system produced the lowest Mean Square Error of 1.75, while the AG hydroponic system produced the highest Mean Square Error of 6.52. The lowest R^2 of 0.868 has been obtained from the Floating hydroponic system, while the AER hydroponic system produced the highest R^2 of 0.996.

It can be deduced from Table 9 that the AER hydroponic system outperformed the other hydroponic systems using the XGBoost model for analysis, with an optimal R^2 of 0.996 which is higher than the R^2 of NFT, Floating and AG with R^2 values of 0.995, 0.868, 0.994 respectively.

The XGBoost predicted Onion Bulb Diameters are shown in Figures 28, 29, 30 and 31. The NFT predicted values have the highest matches with the original OBD values, as shown in Figure 31. Only three data points did not match the original OBD values. The AER, AG, and Floating hydroponic systems have more original OBD data points, which did not match the predicted OBD values. Smart farming practitioners and academics can consider the NFT hydroponic system based on this research as optimal using the XGBoost algorithm for predicting the OBD. The predictions of the XGBoost models from the hydroponic systems are very close to their original values. These predictions indicate a strong correlation between the independent and dependent variables of OBD. The XGBoost model is a good fit for evaluating the hydroponic system, and the optimal R^2 result has been obtained from the AER hydroponic system.

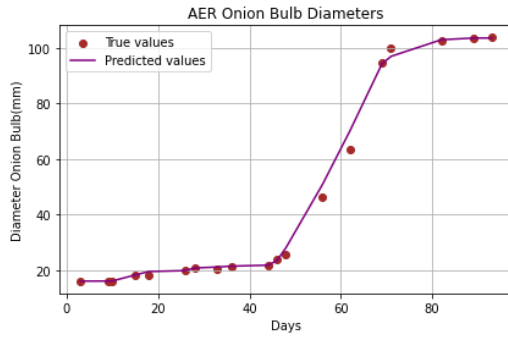


Fig.28: XGBoost OBD predictions from the AER system

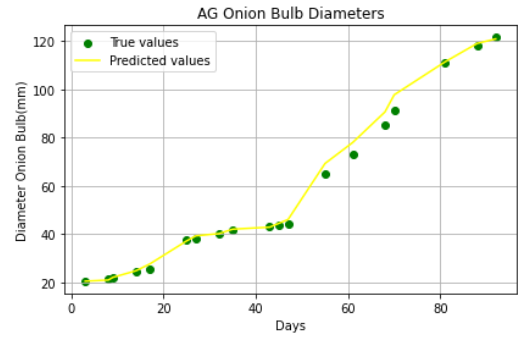


Fig.29: XGBoost OBD predictions from the AG system

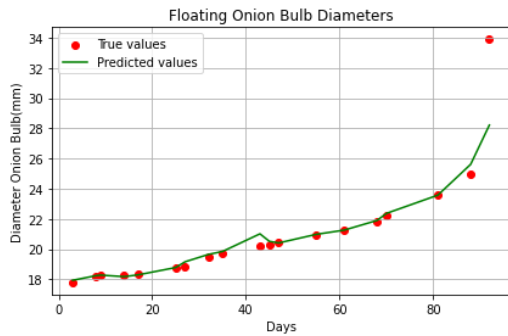


Fig.30: XGBoost OBD predictions from the Floating system

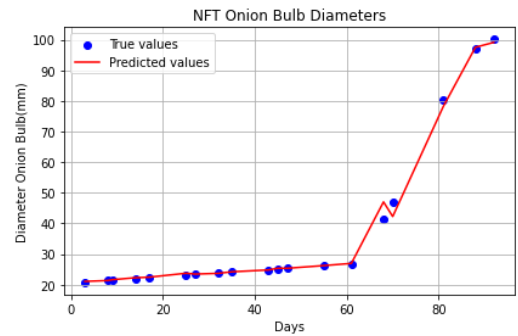


Fig.31: XGBoost OBD predictions from the NFT system

It can be inferred from Figure 32 that the Floating hydroponic system R^2 results using the XGBoost produced the lowest R^2 values of 0.868 when compared with the R^2 values of both the Linear Regressor and XGBoost Models. In contrast, the AER hydroponic system produced the highest results values of 0.996 from comparing the R^2 values of both.

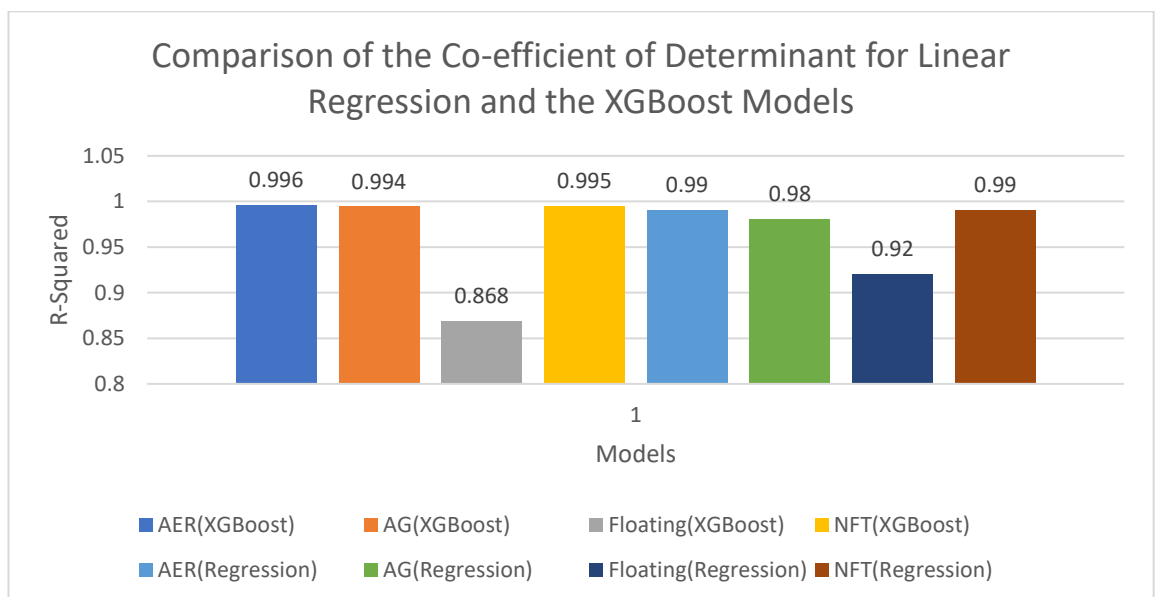


Figure 32: Comparison of the R^2 for the Linear Regressor and XGBoost Models for the Onion Crop

the Linear Regressor and XGBoost Regressor models. At the Department of Agriculture, UP, they have manually compared the four hydroponic systems' co-efficient of determinants. Still, this system has automated the R2 values using the Linear regressor and the XGBoost models.

4.4.1 Classification of the Amaranthus leaves using Convolutional Neural Network

The CNN model trained One thousand and fifty-three (1,053) images. The data have been split into training and testing datasets. The Amaranthus leaves images used for the training have been in two categories: Healthy and infected. The model has been used to classify the Amaranthus image leaves.

One Thousand and fifty-three (1053) image dataset has been analysed. The Amaranthus Viridis images have been split into Seven Hundred and Twenty-Nine (729) for training of Healthy images, Hundred and Sixty-Seven (167) for training of infected images, One Hundred and Forty-Five (145) for testing of Healthy images and Twenty (20) for testing of infected images.

4.4.2 Results from the Amaranthus leaves images Evaluation using CNN, KNN and SVM models.

The Decision tree model has matched only Ninety-two per cent (92.45%) of the original Amaranthus images within the predicted leaves images, which is the lowest accuracy from the CNN, SVM, and K-NN models considered. The CNN model produced the highest accuracy of Ninety-Eight per cent (98.85%), as shown in Table 10.

Table 10: Comparison of the Accuracy of Amaranthus Viridis Images

	CNN	SVM	K-NN	Decision Tree
Accuracy (%)	98.85	95.28	92.92	92.45

It can be inferred from Figure 33 that the confusion matrix has matched Ninety-seven (97%) and Eighty-five (85%) per cent of the Healthy and infected Amaranthus Leaves image datasets, respectively. Fifteen per cent of the Healthy Amaranthus leaves have

been classified as infected, and Three per cent of the infected leaves have been classified as Healthy using the normalised Decision Tree Model.

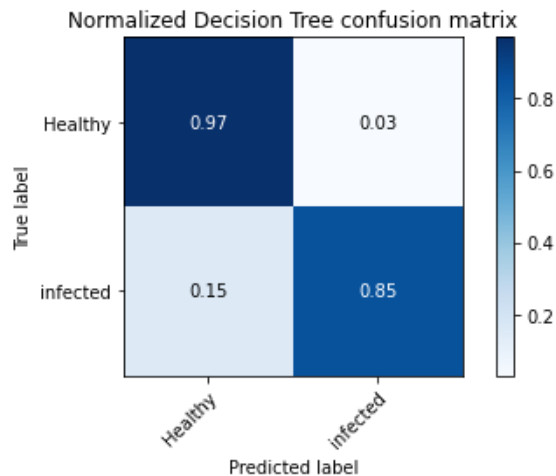


Figure 33: Decision Tree confusion Matrix for Amaranthus Viridis

Using the Normalised K-Nearest Neighbour Models for the confusion matrix as shown in Figure 34, the model has been able to classify One hundred (100) and Sixty-three (63) per cent of the Healthy and infected Amaranthus Viridis image datasets, respectively. No infected Amaranthus Viridis image data has been wrongly classified as Healthy, while Thirty-seven (37) per cent of the Healthy Amaranthus Viridis has been wrongly classified as infected.

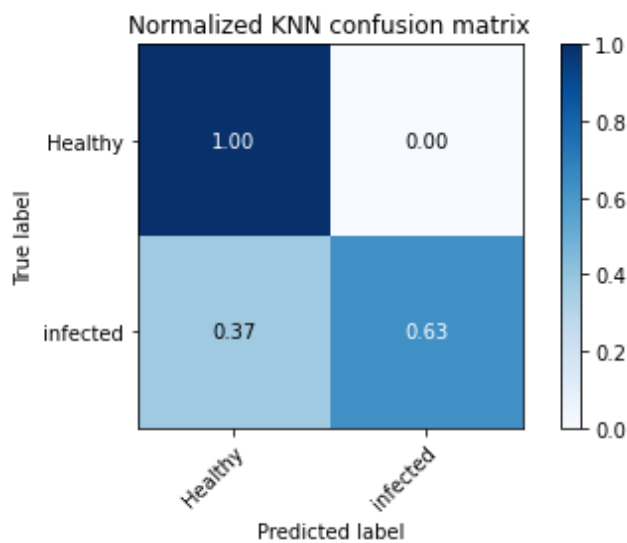


Figure 34: KNN confusion Matrix for Amaranthus Viridis

The Support Vector Machine Model confusion matrix, as shown in Figure 35, has

classified Ninety-Eight (98%) and Eighty-Five (85%) per cent of the Healthy and infected

Amaranthus Viridis image datasets, respectively. Only Two (2%) per cent of the infected Amaranthus Viridis image data has been wrongly classified as Healthy. Fifteen (15%) per cent of the Healthy Amaranthus Viridis has been wrongly classified as infected.

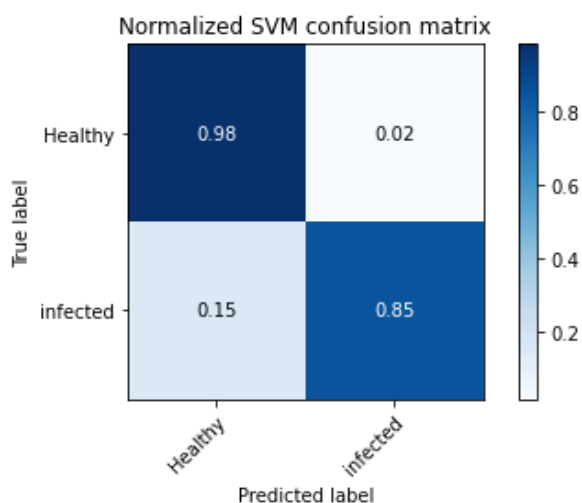


Figure 35: SVM confusion Matrix for Amaranthus Viridis

The Convolutional Neural Network Model has been used to analyze the Amaranthus Viridis image dataset. The model has been able to classify the healthy and infected images by producing an accuracy of over Ninety Eighty per cent (98.85%) for the training dataset, as shown in Figure 36 and after learning the pattern in the training dataset, it produced an accuracy of Ninety-Eight (98%) for the unseen data which is the testing Amaranthus Viridis image dataset. It can be inferred that the model has been able to learn the pattern of the dataset using the training dataset, which shows the model's efficiency.

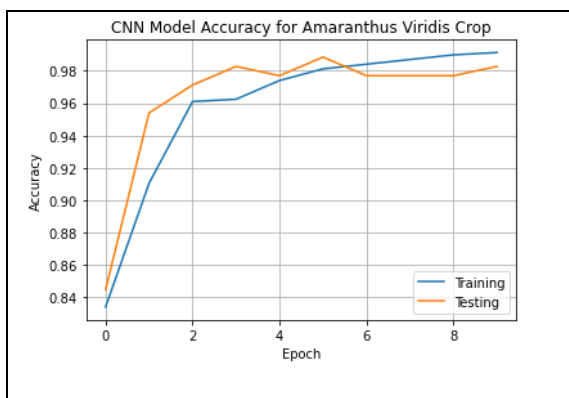


Figure 36: CNN Accuracy for Amaranthus Viridis Crop

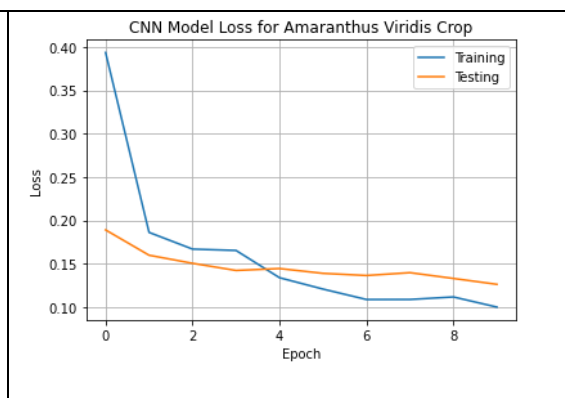


Figure 37: CNN Loss for Amaranthus Viridis Crop

It can be inferred from Figure 37 that the Model produced a Loss of 0.10 while evaluating the training dataset and a Loss of 0.13 during the evaluation of the testing dataset. This implies that the CNN model generalizes very well and has been able to learn the Amaranthus Viridis image dataset and efficiently analyze the unseen image dataset in the testing image dataset.

The Decision Tree Model has been used to evaluate the Amaranthus Viridis image dataset, and Figure 38 shows the model's performance. It achieved an accuracy of about Ninety-four per cent (94%) on unseen Amaranthus Viridis images.

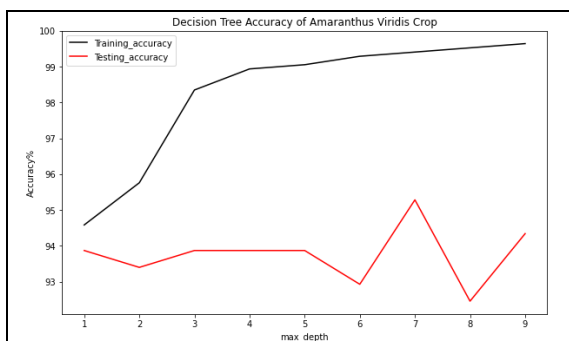


Figure 38: Amaranthus Viridis Decision Tree Accuracy

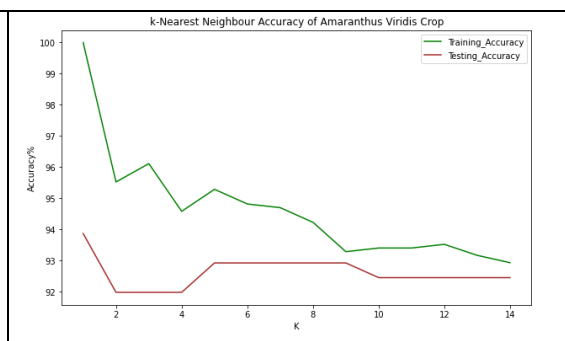


Figure 39: Amaranthus Viridis K-Nearest Neighbour Accuracy

The K-Nearest Neighbour Model has been used to evaluate the Amaranthus Viridis image dataset. The model achieved an accuracy of Ninety-two per cent (92%)

Figures 36, 38 and 39 show the Amaranthus images training and testing Accuracy using the CNN, Decision Tree, and K-Nearest Neighbour models, respectively.

It can be inferred from Table 10 that the CNN model outperformed the SVM, Decision Tree and KNN models. The CNN model generalized very well with no overfitting experienced during its evaluation of the image dataset, as shown in Figure 36, the other models have been pre-processed using the K Fold cross-validation and normalised using the scikit standard scaler library, but their accuracies output has been lower than the CNN model, for this research the CNN produced the optimal performance for the Analysis of the Amaranthus Viridis image dataset.

4.5 Machine Learning Analysis of Amaranthus Viridis Crop growth rate

[127] discuss that crop growth is directly proportional to the change in the weight of the crop and inversely proportional to the change in time and Area of the Land where it is cultivated.

$$\text{Growth rate (g day}^{-1}\text{)} = \frac{\text{change in weight}(w_2 - W_1)}{\text{change in time}(T_2 - T_1)} \text{ [127].....7}$$

Where W_2 is the weight of the crop at Time T_2 , and W_1 is the weight of the crop at time T_1

According to [128], they discuss in their paper that crop spacing can influence the crop growth rate and, invariably, crop yields.

The authors in [129] discuss that climatic parameters can affect the crop yield for winter camelina, a crop grown in the United States of America. Camelina sativa is a member of the Brassicaceae family, a crop used as a source of oil for biofuels, also consumed by humans and animals. The crop produces a high yield during the fall and a very poor yield when planted in September before winter.

From [127, 128, 129], it can be inferred that climatic parameters can influence the crop growth rate.

4.5.1 Methodology

Amaranthus Viridis crop has been planted within the Suitelab smart farm. Amaranthus Viridis has been chosen because the harvest time is within eight weeks, and it has a shallow root system, which is suitable for the floating hydroponic equipment procured by the Suitelab. The XGBoost algorithm has been considered for classifying the Amaranthus Viridis crop growth rate due to the algorithm's ability to classify datasets with high accuracy values. The XGBoost model trains the weak classifiers' data separately and combines all the weak classifiers to form a stronger classifier, which the algorithm uses

for the final prediction values. The XGBoost model does not rely on specific parameters of the dataset before they embark on training the dataset, which makes them suitable for our research. The model is reliable for handling noise and outliers within a dataset. The floating hydroponic system is connected to a Raspberry Pi Internet of Things device, configured to collect climatic and image datasets from the Amaranthus crop growing in the hydroponic system. The following parameters have been collected and recorded from the Amaranthus crop: days after transplant, length of leaves, the width of the leaves, number of leaves per day, and crop height. These were the independent variables, and the crop growth rate was the dependent variable. The dataset has been pre-processed, and the Scikit learn python library has been used to split the dataset into eighty per cent (80%) for training and twenty per cent (20%) for testing. The Theil-Sen, Quantile, Decision Tree, K-Neighbour, XGBoost, and Support Vector regressors have been used to evaluate the crop growth rate of Amaranthus Viridis. The crop growth rate dataset is numerical. It is suitable for regression analysis. The regressor has been used to obtain the determinant of co-efficient (R^2), the mean squared error of the dataset.

4.5.2 Results

From Figure 42, it can be inferred that the XGBoost Boosting algorithm has the highest predicted Amaranthus Viridis Crop growth rate of 4.4 g/day while the decision tree model has produced the lowest crop growth rate of 4.0 g/day, as shown in Figure 44.

From Table 11, it can be inferred that the Lowest Coefficient of the determinant (R^2) has been achieved from the Support Vector Regressor with 0.982. At the same time, the K-Neighbour Regressor produced the highest R^2 of 0.995 value. It can be inferred that the R^2 determines the goodness of fit of the Amaranthus Viridis crop growth rate and the Leaf height, leaf width, days of transplant, and leaf length. The R^2 of the Amaranthus Viridis measures how far the predicted crop growth rates are from the mean.

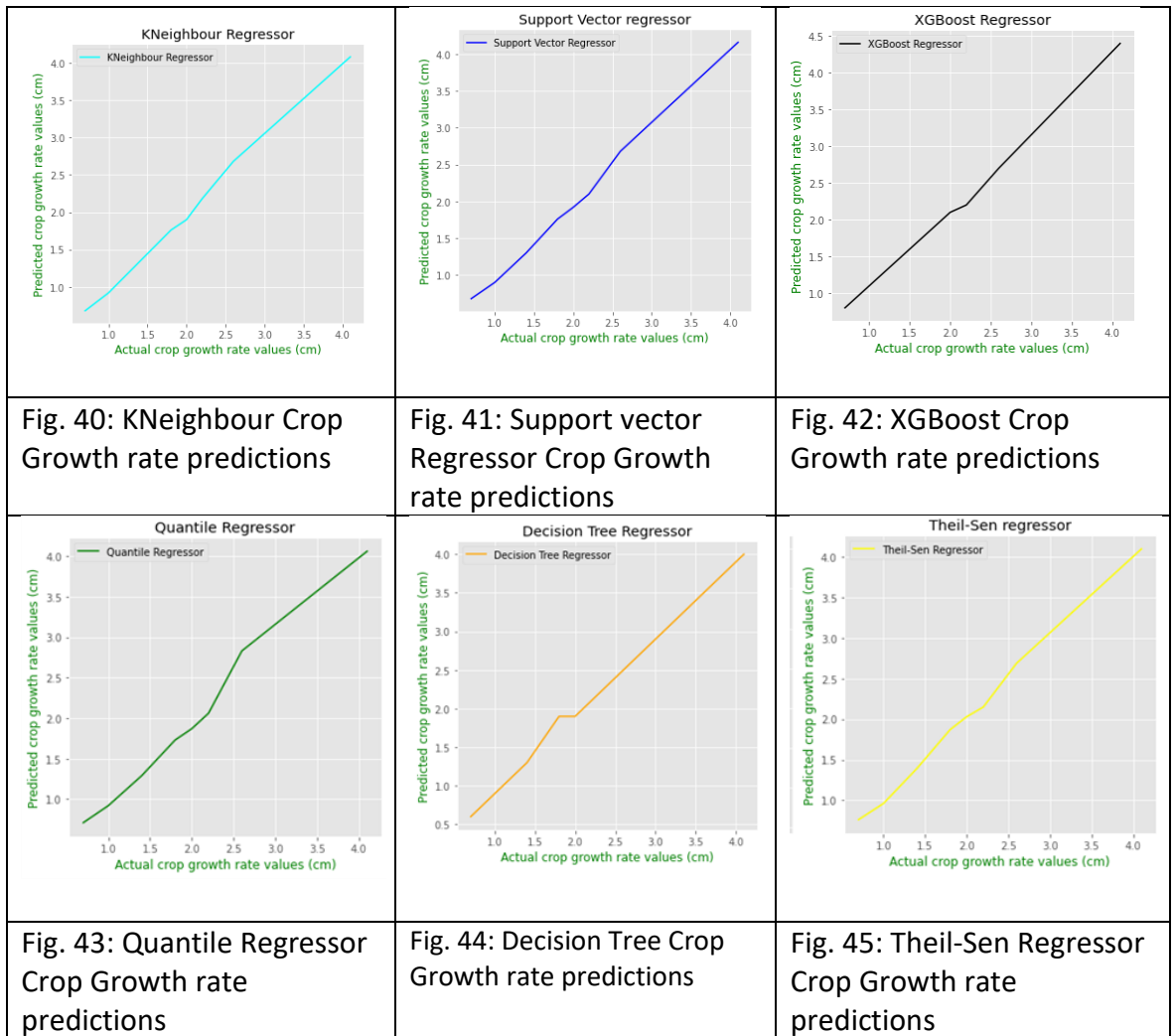
Table 11: Comparison of the Co-efficient of determinants for the Amaranthus crop growth

	XGBoost	Support vector Regressor	Decision Tree regressor	K-Neighbour Regressor	Theil-Sen regressor	Quantile Regressor
Co-efficient of Determinant (R²)	0.987	0.982	0.988	0.995	0.988	0.987
Mean squared error	0.02	0.03	0.018	0.01	0.02	0.019

The XGBoost, Support Vector, Decision Tree, K-Neighbour, Theil-Sen and Quantile

Regressors models all have high R², which indicates the original crop growth rate values are very close to the predicted values, which can be referred to as the line of best fit.

The Mean squared error (MSE) has been used as a performance metric for the Amaranthus Viridis crop growth rate. It is the square of the average difference of the crop growth rate for each model considered. It can be inferred that The K-Neighbour Regressor model produced the lowest MSE of 0.01. In contrast, the Support Vector Regressor produced an MSE of 0.03, resulting in a high R² for the K-Neighbour regressor, showing the minimal variance of the original crop growth rate to the predicted crop growth rate. It can be inferred from Figure 43 that the XGBoost model has the highest predicted Amaranthus Viridis crop growth rate. In contrast, the Decision Tree Regressor model has the lowest predicted Amaranthus Viridis crop growth rate, as shown in Figure 44. It can be inferred that the convergence of the Support vector regressor model for the crop growth rate was very poor, which affected the co-efficient of the determinant, making the predicted values far from the best line of fit or predicted Amaranthus Viridis crop growth rate.



Figures 40, 41, 43, and 45 show the predictions of the K-Neighbour, Support Vector, Quantile and Theil-Sen Regressors crop growth rate, respectively.

4.5.3 Discussion

The DNN model has been used for training the dataset in the four hydroponic systems, Floating and NFT system with an MSE value of 0.13 each outperformed the AG and AER systems with MSE values of 0.34 and 0.73 respectively, indicating that the Floating and NFT system dataset has been well fitted to the DNN model and able to learn the data pattern more than the noise within the dataset.

The optimal R^2 has been achieved using the XGBoost model from the dataset of the AER system with a value of 0.996. It outperformed the NFT, Floating, and AG systems with R^2

of 0.995, 0.868, and 0.994, respectively. The high R^2 indicates the XGBoost has fit very well with the dataset and matches the predicted OBD with the True OBD values.

MSE values of 4.43, 6.52, 1.75, and 3.13 have been obtained for the AER, AG, Floating and NFT, respectively. The Floating outperformed the AER, AG, and NFT systems in fitting the XGBoost models to their dataset.

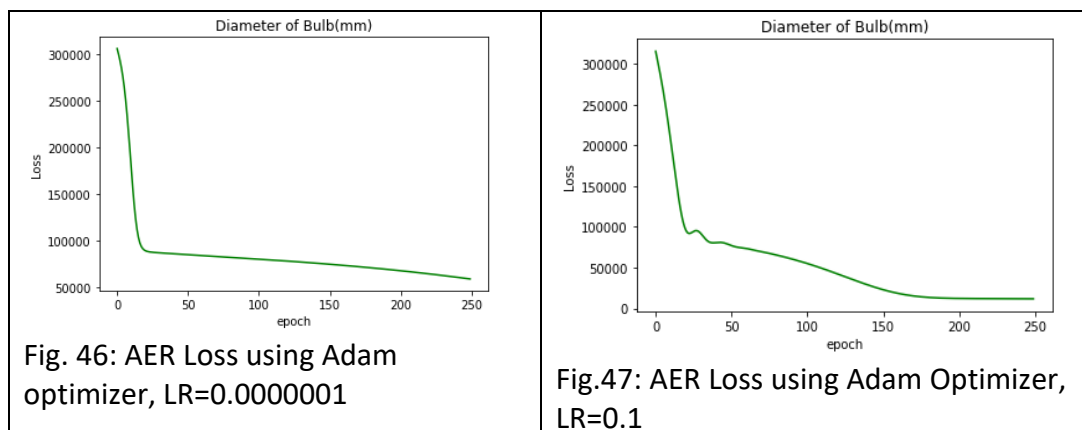
The XGBoost Learned the data pattern very well from each hydroponic system, but optimal performance for the AG data has been achieved.

This research has shown that using the developed models, farmers can predict their Onion Bulb diameter before commencing the planting of their crop and invariably forecast their profits from the harvest. This research has predicted the OBD for daily values, which the Department of Agriculture, UP could not determine manually.

5.0 Chapter 5 Smart Farming Analysis Using Federated Averaging Algorithm

5.1 Analysis of Onion Bulb Diameter using Federated Split Learning

According to the authors in [34], Federated Split Learning (FSL) is a platform where the server aggregates the local updated weights of the edge nodes within the decentralised network. The local weights are split into two, and the last layer or cut layer updated weights are sent to the server for aggregation. The server does not see the raw dataset of the local edge nodes throughout the training. This research used the Federated Split Learning model to evaluate the four hydroponic systems' Onion Bulb diameter (OBD). Splitting the local weights into two parts reduces the computational time of training the local weights since only one part of the updated weights is sent to the server for aggregation. The bandwidth usage on the link is reduced compared with the situation when the total updated local weights are sent to the server for aggregation. This process also reduces the latency of the network. It has been obtained from the training of the FSL models, using the Adam and Stochastic Gradient Descent (SGD) optimizers for hyper tuning while different Learning rates (LR) values range from zero (0) to one (1) has been used. The AER hydroponic system has been hyper-tuned with the SGD and Adam optimizer using 0.01, 0.1, 0.000000 LR and optimal convergence was obtained using the Adam optimizer and LR of 0.1. Attempts made to increase or decrease these LR values caused overfitting of the model and poor convergence.



Figures 46, 47 and 48 show the AER performance using LR of 0.0000001, 0.1 and 0.0000001, respectively. The Adam optimizer has produced optimal convergence since it combines the Adagrad and RMSProp algorithms for handling the sparse gradients for the noisy dataset.

Figures 49 and 50 indicate the performance of the AG hydroponic system when hyper-tuned using the Adam and the SGD optimizers with LR values of 0.01 and 0.0000001. The AG system achieved optimal convergence using the LR value of 0.01 and the Adam optimizer, other LR values were considered, and this caused overfitting of the model for the AG system.

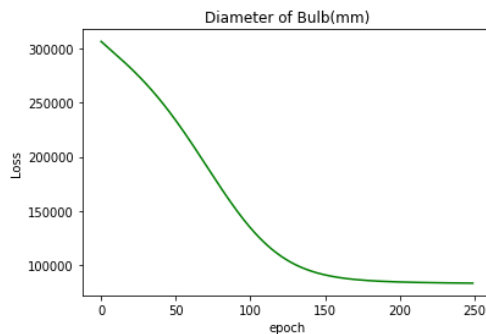


Fig.48: AER Loss using SGD optimizer, LR=0.0000001

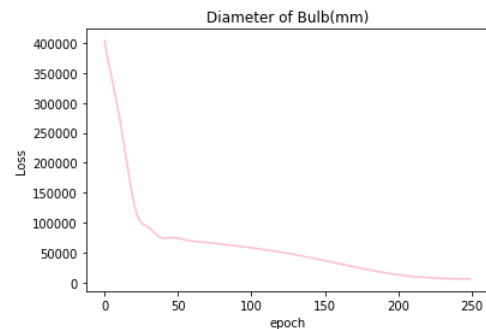


Fig.49: Loss (Adam optimizer, LR=0.01) for the AG system

The dataset from the Floating hydroponic system has been evaluated using the Adam and SGD optimizer with LR values of 0.000001 and 0.01. Figure 51 shows that the optimal convergence has been achieved using the Adam optimizer and LR values of 0.01 with an epoch value of 25 iterations while using the SGD optimizer for the Floating system dataset, convergence has been achieved with an LR value of 0.000001 after 50 epoch iterations as shown in Figure 52, it can be inferred that the Adam optimizer using LR value of 0.01 has converged better than the SGD optimizer since it combines the Adagrad and RMSProp algorithm for achieving its local minima.

Further analysis has been conducted using the dataset from the NFT hydroponic system. LR values of 0.01 & 0.000001 with Adam & SGD optimizer have been used to achieve convergence for the NFT system. Figure 53 shows that the Adam optimizer achieved convergence after 150 epochs using an LR value of 0.01 but an extremely high loss value of 20,000.

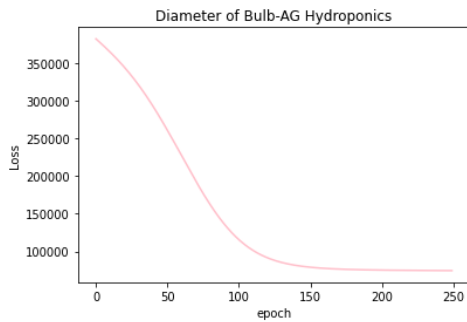


Fig.50: Loss (SGD optimizer, LR=0.000001) for AG system

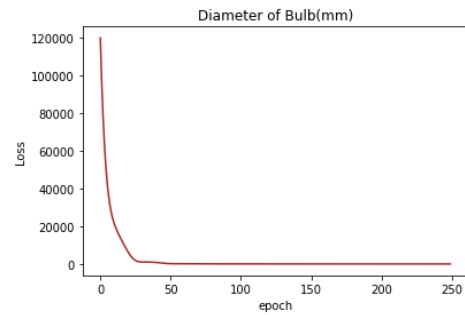


Fig.51: Loss (Adam optimizer, LR=0.01) for the Floating system

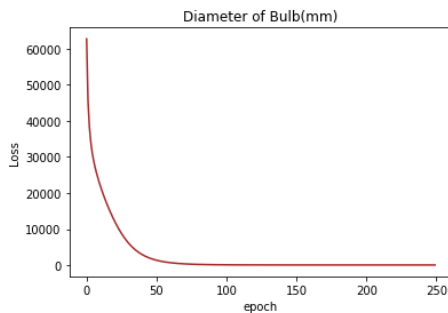


Fig.52: Loss (SGD optimizer, LR=0.000001) for Floating system

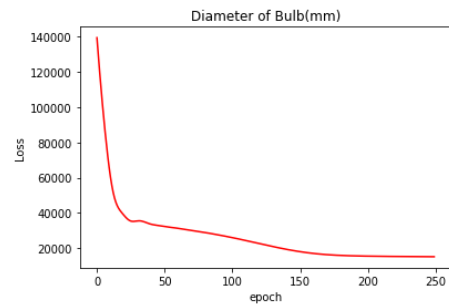


Fig.53: Loss (Adam optimizer, LR=0.01) for NFT system

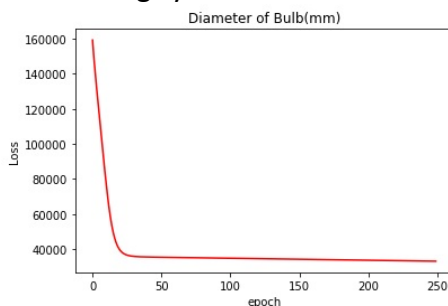


Fig.54: Loss (SGD optimizer, LR=0.000001) for NFT system

Figure 54 achieved convergence after 50 epochs with LR values of 0.000001, and a Loss value of 40,000 has been achieved. It can be deduced that the NFT dataset did not fit the FSL model very well, and this research affirms that the FSL should not be used for evaluating the NFT system dataset from the University of Peloponnese (UP). These high Loss values indicate overfitting. The model has been hyper-tuned with several LR

parameters. Still, the experimentation indicates the NFT dataset is noisy, which converged poorly since hyper-tuning and changing the optimizer did not improve the convergence. The model has been learning more about the noise within the dataset.

The Floating hydroponic system has outperformed the AER, AG, and NFT when evaluated using the FSL model. It achieved the best optimal convergence from the four hydroponic systems considered using the Adam optimizer, with LR values 0.01 within 25 epochs of training.

5.2 Federated Averaging Algorithm Classification

5.2.1 Methodology

This section applies Federated learning models to the smart farming dataset. The federated learning model provides privacy for the data owner since the dataset is evaluated without the data leaving the data owner's location. This provides privacy, and the network latency is reduced. The crop type is predicted from the dataset, which comprises climatic data as independent variables and uses three classes of crops as the dependent variables. The climatic features of the data that have been considered are temperature, humidity, the potential of hydrogen (pH), and rainfall. At the same time, the following crops are the dependent variables: rice, maize and chickpea. A testbed using PySyft, Pytorch and Syft libraries has been used for the emulation.

The use of categorical variables has been explored in the smart farming dataset, and the evaluation of the developed model within a federated learning network to produce predictions of the type of crops as labels while using climatic parameters as independent variables.

In this research, we analyse the climatic data and crop data. These climatic parameters have been considered independent variables, while the crop type is the label.

We will use the federated learning platform to predict the crop types from the climatic parameters in the federated learning platform. This research contributes to knowledge by applying the federated Learning algorithm to smart farming where crop types are predicted from climatic variables, which are used as independent variables and the crop types as labels with all evaluations performed without having access to the raw data.

Edge devices have been used to capture data processed or transmitted to the cloud or server for analytics to ascertain decisions in various sectors.

The data used for this research include climatic features, namely temperature, humidity, the potential of hydrogen(pH), and rainfall, which are the independent variables, and the labels are rice, maize, and chickpea. The classes in the dataset, namely chickpea, rice, and maize, are equally distributed. This implies that the dataset is balanced. The dataset has been split into 80% for training and 20% for testing using the sci-kit learn library [110]. Several Learning Rates (LR) values ranging from zero (0) to One (1) have been used for the evaluation of the dataset. It was observed that the LR values ranging from 0.01 to 0.09, 0.001 to 0.009, 0.0001 to 0.0009, 0.00001 to 0.00009, 0.000001 to 0.000009 produced the same convergence pattern, LR values from 0.00000001 to 0.00000009 has been used in the experimentation. Still, the model did not converge, and these LR values have been dropped for consideration in the experiment. Each federated node has the same labels and attributes since we are exploring homogeneous edge nodes where all the edge nodes have the same attributes and features.

The research experiment aims to investigate the prediction of a particular crop from a class of crops using climatic data as the independent features from the dataset. In contrast, the crop types are the labels from the dataset. This has been achieved using a hyper tuned federated averaging algorithm model. The Duet is a decentralised platform where the edge nodes' data reside at the edge nodes, and the data scientist remotely trains the dataset

without seeing the data. It allows the edge nodes to control what manipulation can be executed on its dataset. This research duet platform is running on our testbed using the PySyft library.

The Testbed has been set up using a Linux machine, using Jupyter Notebook. The data scientist and the data owner could interact via the duet platform. The Data owner is the custodian of the data, and these scenarios can be scaled up for production. First, the data owner establishes the connection using the duet server and waits for the Data scientist to connect to the data owner via the duet server. Once a connection is established, the data owner (edge device) then proceeds to train its dataset and sends its local updated weights to the server or data scientist, the updated global model is then sent back to the edge devices for a repeat iteration, and this process continues until the model converges.

An emulation of the network has been set up using the GNS3 tool to test the Federated Learning model for a smart farming dataset. Climatic data with independent variables such as temperature, humidity, pH, and rainfall have been used as the independent variables. At the same time, three crops, namely rice, maize, and chickpea, were considered the dependent variable; the results are shown in Table 12.

5.2.2 Results and Discussion

The dataset with independent variables of temperature, humidity, pH, and rainfall and dependent variables of rice, maize, and chickpea has been evaluated using the Binary Relevance (Gaussian NB), Classifier chain (Gaussian NB) and Label Power (Gaussian NB) model in the test bed setup within the Jupyter Notebook and the following results have been obtained for each in Figures 55.

The Binary Relevance Gaussian NB, classifier chain (Gaussian NB) and Label power (Gaussian NB) produced an accuracy of 80%, 75%, and 80%, respectively, as seen in Figure 55.

The Binary Relevance and classifier chain Gaussian naïve Bayes model has been used to evaluate the multi-labelled dataset. Table 12 shows the result, the binary relevance Gaussian NB and label powerset Gaussian NB model both produced a harmonic mean of 0.97 and an Accuracy of 0.80. The Binary relevance Gaussian NB and Label powerset Gaussian NB has been able to use the sample averages of each instance of the multi-labelled dataset to produce a harmonic mean of 0.97, and both models could only match 80% of the predicted multi-labelled variables to the original labels of the dataset.

The Classifier Chain Gaussian NB model produced an accuracy of 75%, as seen in Table 12, which indicated it has been able to match only 75% of the predicted labels with the original dependent variables of the dataset. The F1-score of 0.76 has been achieved by the model, showing that the ratio of the product of the precision and recall to the sum of the precision and the recall values from the model during evaluation is 0.76 only. The model considers the sample average since the dataset considered is a multi-label, and each of the sample averages for each instance is used during evaluation to produce the harmonic mean of the model.

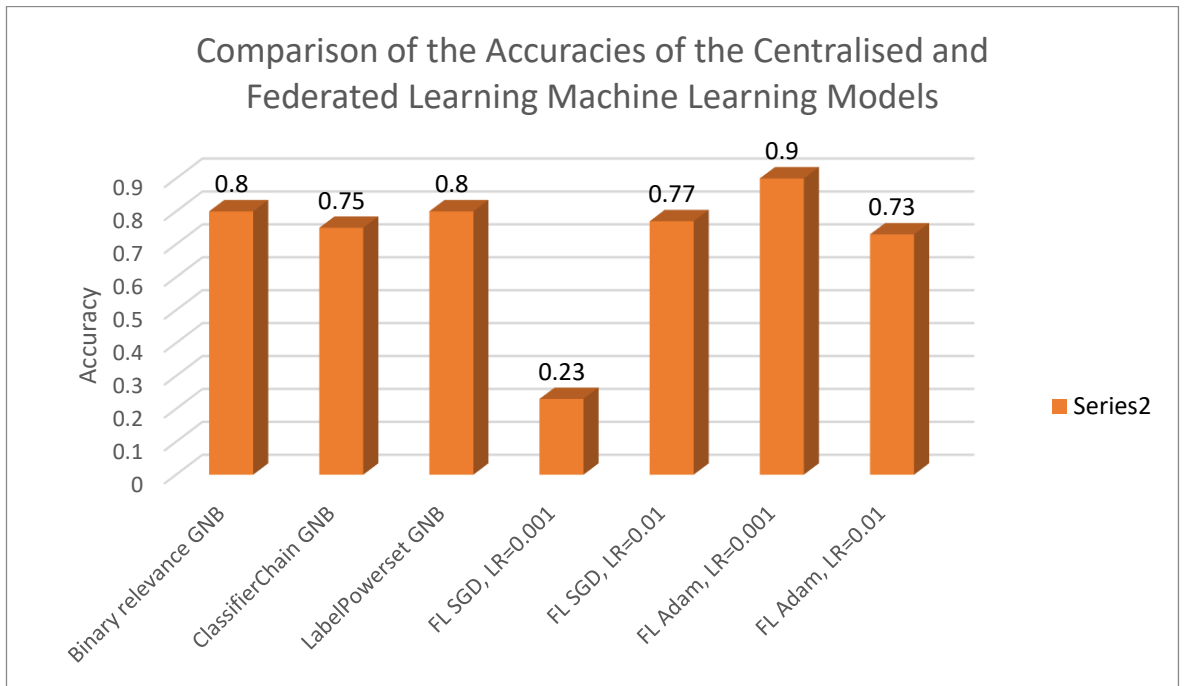


Figure 55: Comparison of the Centralised and Decentralised Machine Learning Models

Table 12: Comparing a multi-labelled dataset's centralised and decentralised Model classification.

	Binary relevance (Gaussian NB)	Classifier Chain (Gaussian NB)	Label powerset (Gaussian NB)	Federated Averaging Model			
	f1-score	f1-score	f1-score	f1-score LR=0.001, SGD optimizer	F1-score LR=0.01, SGD optimizer	F1-score LR=0.001, Adam optimizer	F1-score LR=0.01, Adam optimizer
0	0.97	0.76	0.97	0.48	0.77	0.91	0.91
1	0.97	1.00	0.97	0.80	0.46	0.82	0.50
2	0.86	0.89	0.95	0.00	0.95	0.95	0.73
Accuracy	0.80	0.75	0.80	0.23	0.77	0.90	0.73
macro avg	0.96	0.93	0.97	0.19	0.73	0.90	0.73
weighted avg	0.96	0.94	0.97	0.19	0.73	0.90	0.73

Table 12 compares the results obtained from the research using the federated learning and centralised models to predict the crop type using climatic parameters as independent variables and crops as labels.

The model hyperparameters have been tuned to obtain various results. The learning rate hyperparameters range from zero (0) to One (1), and different values of learning rates between zero (0) and one (1) have been considered for hyper tuning of the models, more so different optimizers such as SGD and Adam have been considered based on previous research by [143]. Using an SGD optimizer, a learning rate of 0.001 and a Computational time of 0.00013 seconds were obtained during the model's training. An Accuracy of 23% has been obtained while the predicted crop was rice, implying the model has been learning more of the noise within the dataset, resulting in high errors since its loss values are also high, as can be seen in Figure 56. It can be inferred that using the SGD optimizer and a learning rate of 0.001, only 23% of the predicted labels were matched with the original labels in the dataset after the training, which indicates the SGD optimizer at this learning rate produced a poor accuracy and failed to match the predicted classes with the original labels. Figures 56 and 57 show the loss value decreasing during the model's training using stochastic gradient descent (SGD) optimizer, with a learning rate (LR) of 0.001 and 0.01, respectively.



Figure 56: Loss using SGD optimizer, Learning rate=0.001

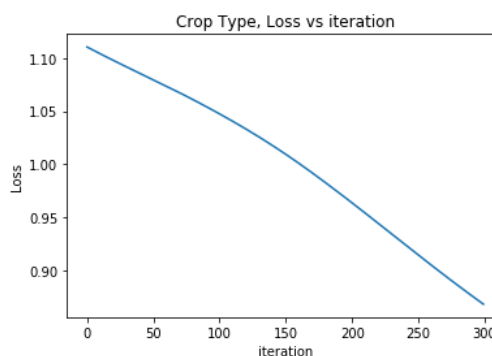


Figure 57: Loss using SGD optimizer, Learning rate=0.01

From the results obtained, as shown in Figure 56, a minimum Loss of 1.096 has been obtained from the evaluation of the model, and Figure 57 has produced a minimum loss value of 0.7. The federated learning model produced an F1 score of 0.48, which is the harmonic mean, that's the reciprocal of the arithmetic means using the SGD optimizer with

an LR of 0.001, as shown in Table 12. It can be inferred that the SGD optimizer function, when a learning rate of 0.001, converged poorly to its local minima. Further hyper-tuning of the model parameters has been conducted with the SGD optimizer but with a different learning rate value of 0.01. The results in Table 12 indicate that only 77% of the predicted labels matched the original labels of the classes of chickpea, which is the predicted crop, using this optimizer and LR value. A harmonic mean (F1-score) of 0.77 has been obtained using the SGD optimizer and LR value of 0.01, which performs better than the initial learning rate considered earlier. It can be inferred that the federated learning model has converged to its local minima much faster, which improved its performance when compared with the other optimizers and Learning rates used.

A different optimizer function, namely the Adam optimizer, is considered for the hyper-tuning of the model using an LR value of 0.001. The predicted class has matched the original values with a percentage of 90%, which indicates a good performance of the accuracy metric, and a harmonic mean (f1-score) of 0.91 has been achieved, as shown in Table 12. It can be inferred that the model has converged very fast, which enabled it to reach its local minima, thereby improving its performance with a 0.91 harmonic mean (F1-score) value and accuracy of Ninety per cent (90%).

Further analysis using the Adam optimizer with a learning rate of 0.01, the hyper-tuning of the model, the predicted class has a match with the original values with a percentage of 73% and harmonic mean (F1-score) of 0.91, which indicates good performance of accuracy metric. It can be inferred that the model dropped its accuracy metric from the previous value using the 0.001 learning rate when a learning rate of 0.01 is considered but has maintained the F1 score. It can be inferred that the model using the Adam optimizer and a learning rate of 0.01 achieved a lower accuracy of 0.73 compared with the accuracy of 0.90 when a LR of 0.001 has been used because the model began to learn more noise within the

dataset while converging to its local minima as shown in table 12. The dataset contained three (3) classes in the dependent variables. During each hyper-tuning with different optimizer functions and learning rate parameters, it has been observed that chickpea was the predicted crop, indicating the federated learning model, without seeing the raw dataset, has been able to match a higher percentage of the crop with its original values.

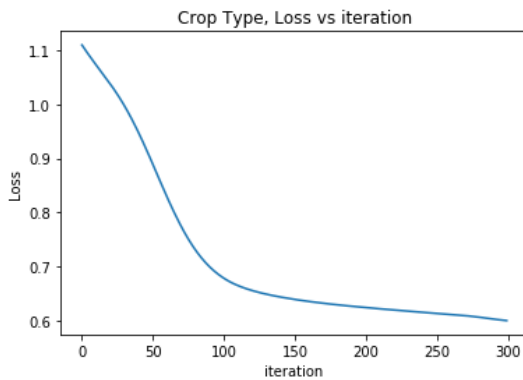


Figure 58: Loss using Adam optimizer, Learning rate=0.01

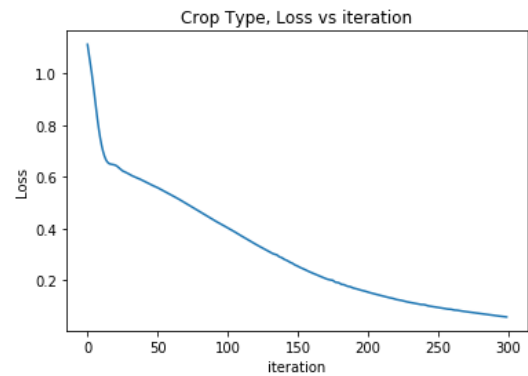


Figure 59: Loss using optimizer=Adam, Learning rate=0.001

Figure 58 depicts a minimum loss value of 0.6, and the loss started to converge appreciably after 100 iterations. However, from Figure 59, the loss has started to converge appreciably after 20 iterations and eventually converged at 0.1, which is a better improvement than the other initial learning rate of 0.001, 0.01 for SGD optimizer and a learning rate of 0.001 for the Adam optimizer. It can be inferred that with the learning rate of 0.001 using the Adam optimizer, the federate learning model has reached its local minima, although its training time at this learning rate has been increased, as shown in Figure 58. The initial training iteration converges after 100 iterations. However, in Figure 59, its training iteration is over 200. This implies the model has begun to learn the noise in the dataset, and it causes overfitting and generalising poorly. This research results confirm the efficiency of the Adam optimizer from the hyper-tuning of the model's parameters. It can be inferred that the Adam optimizer converges better than the SGD optimizer. This confirms that federated learning models reach their local minima at low learning rates and use high training time to converge. The dataset used for this experiment was obtained from [145].

5.3 Modelling of Federated Learning Smart Farm Network

5.3.1 Methodology

The GNS3 tool has been used to emulate the network, shown in Figure 15. The architecture has 3 tier layers. The generated data have been sent to the second-tier layer, which contains the Mobile Edge Computers (MEC). The data have been trained at the MEC, and the local updates have been sent to the server. The server aggregates all the various models from the respective edge nodes, creating a global model that the server sends to the various edge nodes to update its local model with the new global weights for its next iteration. The traffic has been generated using the *ostinato* installed in the GNS3 emulator tool. The *ostinato* generates packets with different capacities, such as one (1) packet per second, five (5) packets per second, or Ten (10) packets per second. These packets have been sent to the server node via the Fast Ethernet adapter interface of the edge nodes.

Some Assumptions considered in the Modelling of the Federated Learning Smart Network.

1. The Arrival of the local model from the edge nodes is independent of each other and discrete events.
2. The Arrival rate of the local models follows the Poisson distribution.
3. The inter-arrival rate time is independent, and we assume the service rate is said to be exponentially distributed.
4. Let b_i represent the effective bandwidth of the i^{th} node.
5. Let d_i represent the effective delay of the i^{th} node.
6. Let BW_{eff} represent the effective bandwidth at the aggregate server.
7. Let DL_{eff} represent the effective delay at the aggregate server.

$$\sum_{j=1}^k b_i^j = b_i \left| \min_j(d_i^j) = d_i \right. \dots\dots\dots(9)$$

$$(B_i, D_i) = \left(\sum_j b_i^j, \min_j(d_i^j) \right) \dots\dots\dots(10)$$

$$BW_{eff} = \min(B_s, B_i)$$

$$DL_{eff} = (D_s, D_i)$$

The Bandwidth delay product (BDP) can be calculated as follows;

$$\begin{aligned} BW_{eff} \times DL_{eff} &= \sum_{i=1}^n \min(B_s, B_i) \times D_s \sum_{i=1}^n D_i \\ &= \left(\sum_{i=1}^n \min(B_s, \sum_{j=1}^k b_i^j) \times D_s \sum_{i=1}^n \min_j(d_i^j) \right) \dots\dots\dots(11) \end{aligned}$$

5.3.2 Time complexity

The Big O notation is used to determine the time complexity of the Model.

$$\text{The iteration } H = n \div 2^H \dots\dots\dots(12)$$

Taking the iteration to equal 1,

$$1 = n \div 2^H$$

Therefore, $n = 2^H$

taking the log of both sides of the equation

$$\log n = \log 2^H$$

$$\text{Log } n = H \times \log_2 2, \text{ where } \log_2 2 = 1$$

$$\text{Log } n = H$$

From the model developed, the Big O notation for the modified FedAve algorithm is O(n).

it can be inferred that the time complexity for the modified model is in order of n, O(n).

5.3.3 Results and Discussion

From the GNS3 emulated network architecture, the following results have been obtained.

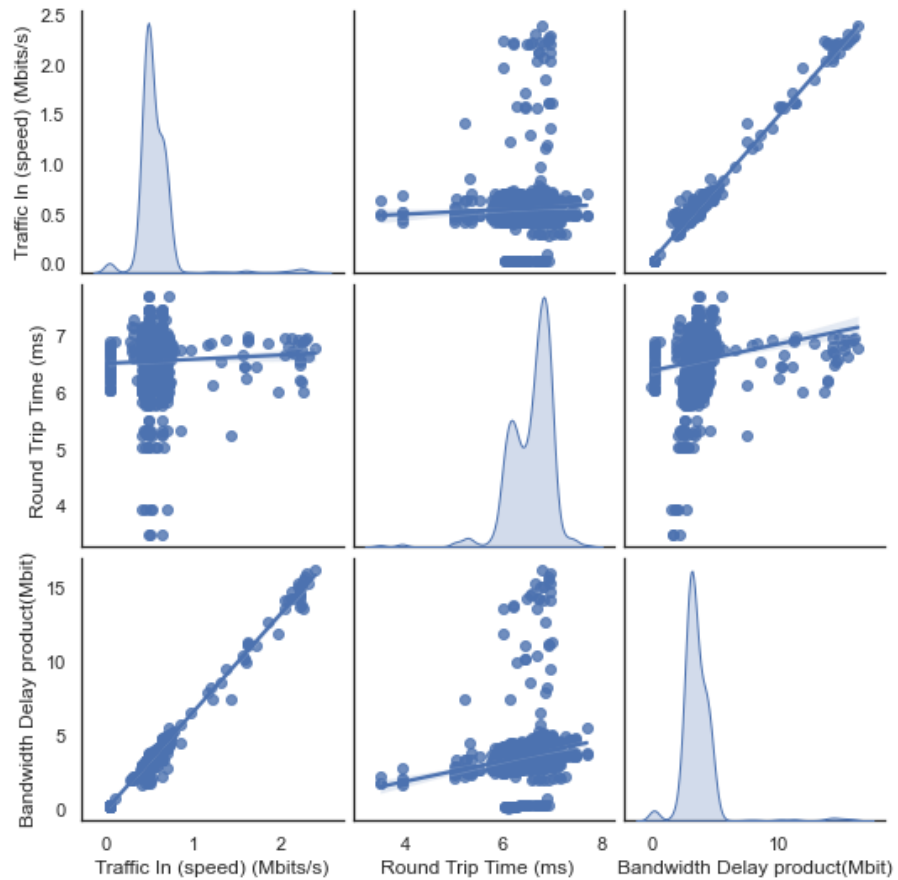


Figure 60: Pairwise Plot of the Bandwidth Delay Product.

It can be inferred from Figure 60, which shows the univariate distribution of the TS, RTT and BDP. It can be inferred that there is a positive correlation between traffic speed and the BDP. It can be inferred that the chart depicts the density probability function of the variables. The RTT shows a bi-modal distribution indicating two variations of the RTT having peak values, and the RTT is skewed to the right, implying the RTT mean values are more than the median values in the dataset. The BDP and TS both show they have a single mode, indicating that a single sample of the dataset population represents the mean, and the probability density for the TS and BDP are skewed to the left, indicating the mean for TS and BDP are less than the median of the dataset.

From Figure 60, it can be deduced that the correlation between the BDP and round-trip traffic speed & round-trip time is very poor, and a high volume of noise is shown in their correlation. The two regression charts in Figure 60 indicate that the correlation between BDP and round trip time, round trip time and BDP are very weak. They both produce different regression charts that have not matched points between the originals and the prediction in the regression charts within Figure 60. This indicates that round-trip time has a minimal influence on the BDP, and traffic speed also has a minimal influence on round-trip time.

The kernel density estimate indicates that the round-trip time is bi-modal while the traffic speed and BDP are unimodal. This implies the round trip time has two peak values with the greatest distribution. In contrast, the BDP and traffic speed have single values with the greatest distribution, representing the peak value within the BDP dataset. It is very important to note that Figure 60 shows the pairwise plot to express the correlation between the variables, and it cannot be conclusive evidence of the entire relationship between the traffic speed, round trip time and BDP within the dataset. Further analysis is carried out on the BDP dataset using the federated and classical machine learning models.

To further substantiate our analysis of the BDP dataset and support our analysis, a heatmap has been developed, as shown in Figure 61. The heatmap gives numerical values to the strength of the correlation between the variables in the BDP product dataset. The highest value is one, indicating a strong correlation between the variables, and zero (0) indicates a weak correlation between the variables. It can be inferred from Figure 61 that the correlation value between Traffic speed and round-trip time has a value of 0.042, substantiating which confirms that the correlation indicated in Figure 60 between the two variables is very poor. The correlation value between round-trip time and the BDP has a value of 0.18, which is higher than the correlation value of round-trip time and traffic speed

but also shows a poor correlation between the two variables. The correlation values between the BDP and traffic speed have a correlation value of 0.99, which is extremely high, indicating the Traffic speed within a smart farm federated learning network can influence the BDP values within the smart farm network. The diagonal values show the correlation of each to itself, which all give the value of one (1).

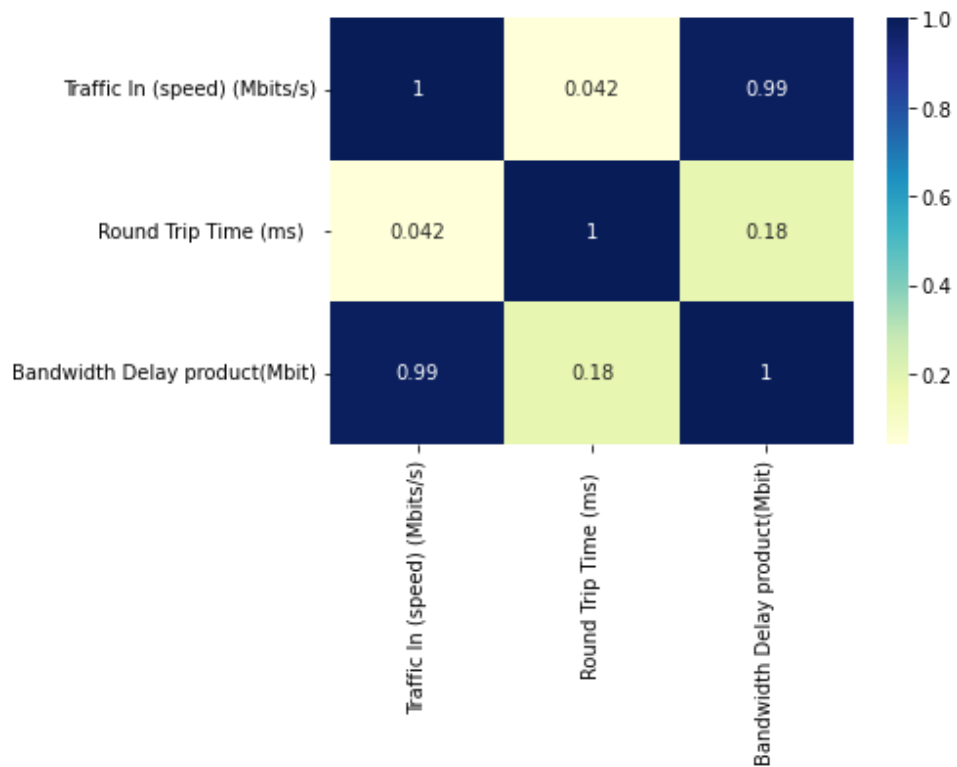
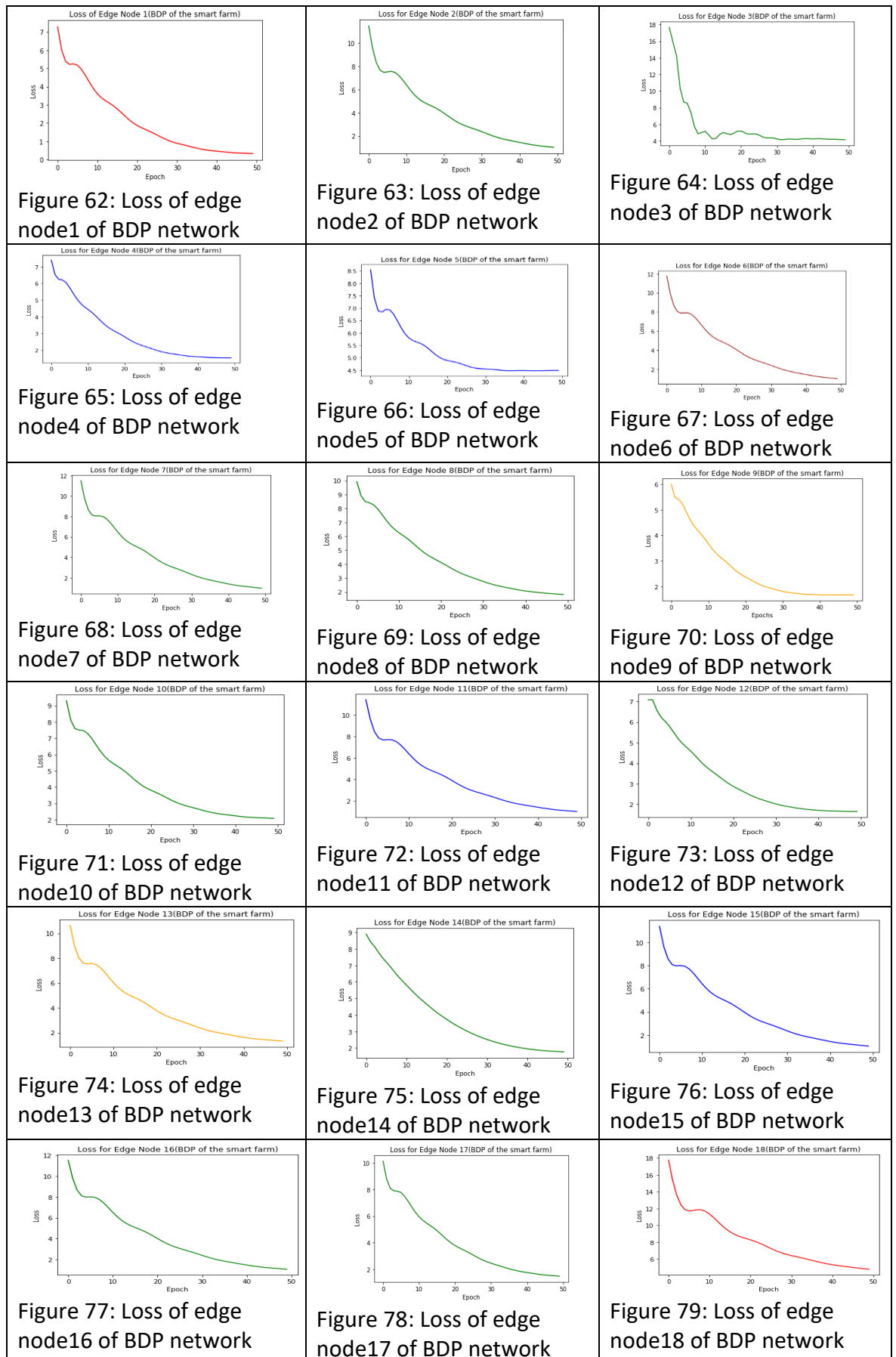


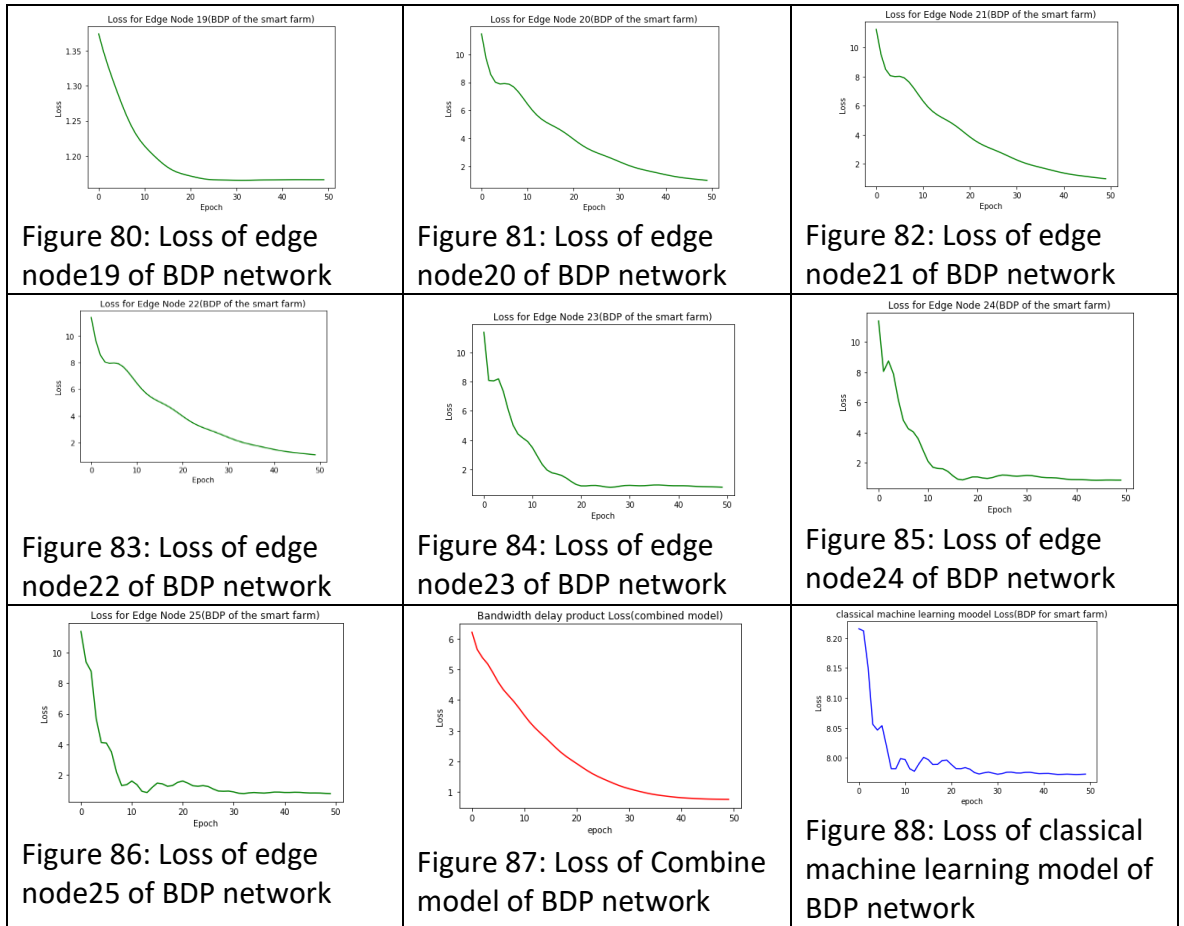
Figure 61: Heat map correlation of the Bandwidth-delay Product

The evaluation of the Federated Learning model for the edge nodes, aggregated model and classical linear model all converge at different epoch values.

From Figures 62 to 86, it can be observed that the edge nodes converged at different loss values, as can be seen in the diagrams. As seen in Figure 87, the classical machine model converged after 20 epochs, indicating the classical machine learning model converged at a slow rate compared with the combined federated learning model. The Aggregated Federated Learning model chart, as seen in Figure 87, converged after Thirty (30) epochs and the combined model did not fluctuate or indicate any overfitting during the

aggregation of the local weight updates, which is reflected in its ability to match a high proportion of the predicted BDP values with the original BDP values as shown in Figure 88.





It can be observed that a high portion of the predictions is very accurate with the original BDP values. This indicates that the aggregated model can combine the individual edge node models and produce an acceptable prediction of the BDP. This is shown in Figure 89, where a plot of the predictions against the original BDP values is shown. It can be inferred that the Federated Learning model can predict a high proportion of the BDP original values. The research investigates the predictions of the classical model, as shown in Figure 90, where a low proportion of the original values accurately match the original BDP values. It can be observed from Figures 89 & 90, that the aggregated model of the Federated Learning platform has been able to predict a high proportion of the original values of the original BDP values, and most of the predicted BDP values are an accurate match of the original BDP values.

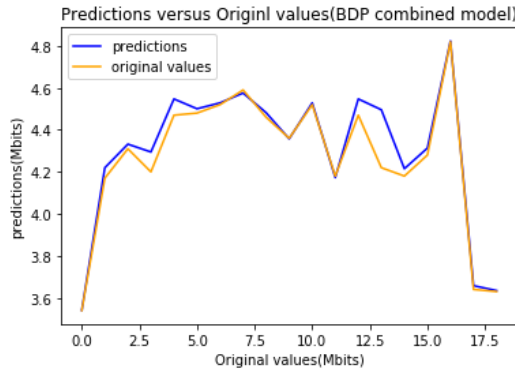


Figure 89: Prediction versus Original Aggregate model of BDP network

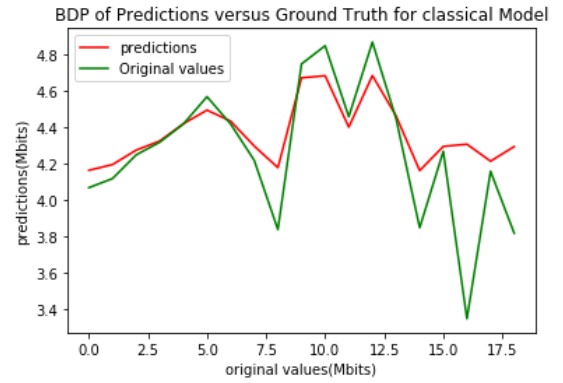


Figure 90: Prediction versus Original Classical model of BDP network

Figures 89 and 90 show that the Aggregated model produce a better performance of the predictions of the BDP model compared to the original BDP values. It can be inferred that high BDP causes faster convergence, when the bandwidth capacity is high within a smart farm link. For rural farms where internet connectivity is challenging, this can reduce the financial expenditure on internet provision for a metered link. The faster convergence indicates the congestion within the link has been overcome due to the high BDP.

6.0 Chapter 6 Conclusion and Future Work

6.1 Summary

Amaranthus Viridis is a vegetable crop produced by many countries, grown and harvested within eight weeks. The crop has been grown on a floating hydroponic system in the SuiteLab of the Computer Science and Informatics Division of London South Bank University. Six hundred watts of sodium light has been used for the crop and a compound fluid fertilizer containing Nitrogen, Phosphorus, Potassium, and other trace elements. The Amaranthus crop has been planted in a nursery for two weeks using a wool rock to provide a solid platform for germinating seeds. After that, it was transferred to the floating hydroponic systems for another six weeks before harvesting the crop.

The IoT device connected with the DHT11 sensors collect climatic data within the hydroponic farm. The images of the Amaranthus Viridis leaves have been captured with the Raspberry Pi while the crop grows. The crop height, leaves length and width parameters have been collected while the crop was growing. The images dataset collected has been analysed using the CNN, SVM, Decision Tree, and KNN algorithm models. The CNN has outperformed the other algorithms for analysing the Amaranthus Viridis image dataset. Other algorithms considered for the Amaranthus Viridis Crop growth rate analysis are Theil-Sen Regressor, Decision Tree Regressor, K-Neighbours Regressor, Support Vector Regressor, Quantile Regressor, and XGBoost Regressor. The Classifier chain Gaussian Naïve Base, Label powerset Gaussian Naïve Base, Binary relevance Gaussian Naïve Base and the Federated Averaging algorithm have been used to analyse the Multi-class smart farming dataset. An in-depth comparative analysis has been done using Deep Neural Network, Linear regression, XGBoost regressor, and Federated Split Learning for the Smart farming dataset from the University of Peloponnese, Kalamata, Greece.

6.2 Research Outcome.

This section summarises the research outcome for this thesis.

A detailed survey of the recent publications on smart farming to identify existing challenges. Review of the crop and Livestock production and post-harvesting has been carried out. The following parameters, the namely computational power of systems, counting and crowd control, devices operated by batteries, holding cost, transport cost, the demand of the market, and the psychological effect, have been considered to ascertain their influence on smart farming [84]. Various sensors have been used in smart farms, and their contributions have enabled farmers to collect climatic parameters within the farm. Unmanned aerial vehicles have been explored, and the valuable contributions they bring to the farming industry through spraying fertilizers, herbicides, and monitoring of farmlands, to mention a few.

IoT, Artificial intelligence, and machine learning concepts have been investigated, as well as their impact on smart farming. Their introduction into agriculture has revolutionised how farmers monitor their animals, grow their crops, and process their produce faster and in large volume. These techniques have boosted food production [84]. Machine learning models have been used to evaluate centralised and decentralised smart farm networks. This enabled the farmers to forecast crops and animal production for the future based on existing farm capacity, climatic conditions, and other infrastructural facilities. Despite the laudable contributions of the use of technologies in the farming industry, many challenges have been identified that impede the progression of farmers, such as data privacy, governance, security, and refusal of acceptance of Information Technology by farmers to automate their farms. Other challenges include low-powered edge devices and improved ML models to address new disease detection in the farms [84].

All over the world, farmers experience a high volume of wastage of farm products or meager harvests. This is due to poor storage facilities, crop disease outbreaks within the farms, poor seed quality used for farming, unsuitable land for planting crops, and lack of technological facilities for farming, especially in developing countries [84]. Using professional farming practices, Intelligent IoT devices, and alternative farming mediums such as hydroponic systems, the farmers will be able to meet the food demand by the populace and industry and furnish academic and non-academic practitioners with reliable information for their research.

6.2.1 Analysis of Hydroponic Systems Data Using Machine Learning Algorithms

The Deep Neural Network (DNN), Linear regression, XGBoost, and Federated Split Learning models have been used to evaluate the Onion Bulb Diameter (OBD) dataset collected at the University of Peloponnese, Kalamata, Greece. The dataset contained the following independent variables Temperature($^{\circ}\text{C}$), water consumption (Litres), Number of Leaves (NL), Nitrogen (mg/g), Phosphorus (mg/g), Potassium (mg/g), Calcium (mg/g), Magnesium (mg/g), Sulphur (mg/g), sodium (mg/g) are the independent variable. At the same time, the OBD has been considered the dependent variable. The results indicated that the Aeroponic (AER) hydroponic system predicted OBDs were very close in values to the original OBDs, inferred from the high coefficient of determinant values produced. In contrast, the Floating hydroponic system predicted OBDs have been far from the original OBD, as seen in the value of the coefficient of determinant value produced from the XGBoost model. This is indicated in Figure 31 [158].

Linear regression, Deep Neural Network, XGBoost, and Federated Split Learning algorithms have been employed to analyse the Onion Bulb Diameter (OBD) dataset provided by the University of Peloponnese, Kalamata, Greece.

Dataset from four different hydroponics systems: Aggregate (AG), Aeroponics (AER), Floating and Nutrient Film Technic (NFT). Evaluating the four hydroponics systems using the XGBoost algorithm. It has been inferred that the Floating hydroponic system achieved the lowest Mean Square Error (MSE) of 1.75 to obtain convergence, while the NFT, AG, and AER have an MSE of 3.13, 6.52 and 4.43, respectively. The DNN algorithm has also been considered for evaluating the AER, AG, Floating, and NFT. The Floating and NFT hydroponic system converged with the lowest MSE of 0.13 each. The AG and AER hydroponic systems converged using the DNN model with MSE of 0.34 and 0.74, respectively.

The Floating hydroponic system produced the lowest MSE of 0.47, while the AER, AG, and NFT produced MSE of 9.89, 14.7 and 2.08, respectively, using the Linear Regression algorithm.

The Linear regression algorithm has been used to predict the OBD for the AG, AER, Floating and NFT hydroponic systems. The NFT system predicted OBDs are the closest to the original OBD, as shown in Figure 27, when compared with the predicted values of the AER, AG, and Floating hydroponic systems [158].

A comparison of the R^2 of the AG, AER, Floating and NFT hydroponic systems has been carried out, and the AER hydroponic system using the XGBoost model produced the highest R^2 , as shown in Figure 32.

The computational time of the AG, AER, Floating and NFT hydroponic systems have been investigated using the XGBoost model, and it can be observed that the Floating hydroponic system converged with the lowest computation time of 656ms. In contrast, the AER, AG and NFT converged with 750ms, 797ms, and 844ms, respectively.

6.2.2 Machine Learning Analysis of Amaranthus Viridis Image Dataset

The images of Amaranthus Viridis leaves have been evaluated using the Convolutional Neural Network (CNN), Decision Tree classifier, Support Vector classifier (SVC) and K-Nearest Neighbours (KNN) algorithm.

The CNN, Decision Tree classifier, Support Vector classifier (SVC) and K-Nearest Neighbours (KNN) evaluation of the Amaranthus Viridis leaves images achieved the following accuracies of 98.85%, 92.45%, 95.28% and 92.92% respectively. The accuracy is the percentage of the predicted images that match the original images.

The Amaranthus Viridis crop growth rate has been trained using Support Vector regressor, Theil-Sen, Quantile, K-Neighbours, Decision Tree, and XGBoost Regressors. It can be inferred from the experimentation that the K-Neighbours model has produced the highest coefficient of determinant with a value of 0.995. The predicted Amaranthus crop growth rate from the K-Neighbours model is very close to the original crop growth rate values as expressed from the coefficient of the determinant of the K-Neighbours model. In contrast, the predicted crop growth rate from the Support Vector Regressor model with a value of 0.982 has been the lowest R^2 obtained from the analysis.

6.2.3 Smart Farming analysis using Federated Averaging Algorithm

The Federated Split Learning (FSL) model has been used to evaluate the AG, AER, Floating and NFT hydroponic systems. Hyper tuning the parameters of the Federated Split Learning models using different optimizers, such as Stochastic Gradient Descent (SGD) and Adam optimizers, with different learning rate values ranging from 0.0000001 to 0.1. It may be inferred that the Floating hydroponic system obtained optimal convergence with a learning rate of 0.01 using the Adam optimizer, as shown in Figure 51. It implies that the Floating Hydroponic system FSL model was not learning the noise in the dataset and

achieved convergence within a very short time without overfitting during the dataset training [158].

The Label powerset Gaussian Naïve Base, Binary Relevant Gaussian Naïve Base, Classifier Chain Gaussian Naïve Base algorithm has been used to evaluate the climatic parameters to predict the crop type for a smart farm within a centralized network. The Label powerset Gaussian Naïve Base, Binary Relevant Gaussian Naïve Base, and Classifier Chain Gaussian Naïve Base algorithm have been able to match the predicted crop with the original crop with an accuracy of 80%, 80%, and 75%, respectively. The ratio of the precision and recall of the product to the sum of the recall and precision for the Label powerset Gaussian Naïve Base, Binary Relevant Gaussian Naïve Base, and Classifier Chain Gaussian Naïve Base obtained are 0.97, 0.97, and 0.76, respectively. It can be inferred that the Binary Relevant Gaussian Naïve Base and Label powerset Gaussian Naïve Base models produced a higher Harmonic Mean from the evaluation of the centralized network.

The Federated learning Averaging model has been used to evaluate the Multi-labelled smart Farming dataset. The model has been hyper-tuned using different optimizers such as SGD and Adam, and learning rates ranging from Zero (0) to One (1) value. The Federated averaging model using the SGD optimizer and learning rate of 0.001 matched the least amount of the predicted crop type to the original crop type values with a value of Twenty-three per cent (23%). In comparison, the optimal matching of the predicted crop types with the original crop type values has been achieved using the Adam optimizer with a learning rate of 0.001 with a value of 90%.

A harmonic mean is the consideration of the recall and precision product ratio to the sum of the recall and precision. It can be inferred that the optimal harmonic mean has been obtained from the hyper-tuned model using the Adam optimizer and a Learning rate of

0.001 with a value of 0.91. In contrast, the least harmonic mean of 0.48 has been produced using the SGD optimizer and LR value of 0.001. From the centralised and decentralised network models, it can be inferred that the decentralised network model using the federated averaging model has produced a better convergence without having access to the raw dataset than the centralised network model [160].

6.2.4 Modelling of Federated Learning Smart Farm

The Modelling of the smart farm federated learning network has been investigated using the GNS3 tool for experimentation. It has been observed that the decentralised network using the federated Learning model converged much faster than the centralised classic machine learning model using the bandwidth-delay product as performance metrics. This result indicates that the Federated learning model, which trains its dataset without access to the raw dataset, has converged much faster than the classical machine learning model.

The BDP can be concluded as having a higher correlation with the traffic speed than the round-trip time within a smart farm federated learning network. This confirms that a smart farm federated learning network will converge faster when the traffic speed within the network is very high from this research experimentation.

A Model has been developed for the Federated Learning Smart Farm Network. The Bandwidth Capacity and round-trip time (RTT) for the smart farm network have been considered, and it can be observed that there exists a correlation between the Bandwidth delay product (BDP) and the traffic speed (TS) of the link within a smart farm with a correlation value of 0.99 as shown in Figure 61. A comparison of the convergence performance for the local, classical, and aggregate models has been carried out for the smart farm network. It can be inferred that the local models converge poorly, as seen in Figures 62 - 86.

The combined model has matched a higher portion of the predicted BDP with the original BDP values. This is seen in Figure 89, while Figure 90 shows the poor matching of the predicted BDP values with the original BDP values for a classical model. It can be inferred that the decentralised, federated learning model for a smart farm network has outperformed the classical centralised model in matching the predicted values to the original values of the BDP [160].

6.3 Further work

The effect of parallel learning algorithm models to AER, AG, NFT, and Floating hydroponic systems should be investigated. Academic researchers can use this work as an insight into findings on hydroponic systems. Still, other features may be considered to see the impact of features such as Artificial light, the micro-nutrients ($\mu\text{g/g}$ dry weight), and some climatic parameters on the Onion crop Bulb diameter within the chosen hydroponic systems. A comparison of the performance of the Federated Split Learning model, Federated Averaging model and classical machine learning models for the AER, AG, Floating and NFT hydroponic systems can be investigated to predict the OBD for each of the hydroponic systems.

Further investigation can be explored to determine the prediction of OBD for the AER, AG, Floating, and NFT hydroponic systems using the Huber and Transformed Target regressor algorithm.

The vision Transformer algorithm can be considered for future evaluation of the Amaranthus leaves images while considering the CNN as benchmarks from the results of this research. An in-depth comparative analysis of the Random Forest, CatBoost, and XGBoost algorithms of the Amaranthus Crop growth rate should be explored.

The application of a software-defined network within a federated Learning platform to reduce the Latency of a smart farm network should be considered for further analysis.

Bibliography

1. <https://www.fao.org/sustainable-development-goals/indicators/212/en/> [Access: January 2022]
2. P. Rekha; Venkata Prasanna Rangan; Maneesha V. Ramesh; K. V. Nibi (2017), High yield groundnut agronomy: An IoT based precision farming framework, 2017 IEEE Global Humanitarian Technology Conference (GHTC), DOI: 10.1109/GHTC.2017.8239287
3. Nita Jaybhaye, Purva Tatiya, Avdut.Joshi, Sakshi Kothari, and JyotiTapkir (2022), Farming Guru: - Machine Learning Based Innovation for Smart Farming, Proceedings of the Fourth International Conference on Smart Systems and Inventive Technology (ICSSIT-2022), 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT) | 978-1-6654-0118-0/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/ICSSIT53264.2022.9716287
4. Hyun Yeo (2022), Smart Farming Technology Review, 2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA) | 978-1-6654-8350-6/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/SERA54885.2022.9806473
5. Nahina Islam, Biplob Ray, Faezeh Pasandideh IoT Based Smart Farming: Are the LPWAN Technologies Suitable for Remote Communication? 2020 IEEE International Conference on Smart Internet of Things (SmartIoT), 978-1-7281-6514-1/20/\\$31.00 ©2020 IEEE, DOI:10.1109/SmartIoT49966.2020.00048
6. Sushitha S, Harsha Hegde (2022), Smart Farming-a key to Sustainable Agriculture Development in India” -A Study, 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT) | 978-1-6654-0118-0/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/ICSSIT53264.2022.9716494

7. Dr M.Pyingkodi¹, Dr K.Thenmozhi², K.Nanthini¹, M.Karthikeyan¹, Dr Suresh Palarimath³, V.Erajavignesh⁴, G.Bala Ajith Kumar (2022), Sensor Based Smart Agriculture with IoT Technologies: A Review, 2022 International Conference on Computer Communication and Informatics (ICCCI), Jan. 25 – 27, 2022, Coimbatore, INDIA, 978-1-6654-8035-2/22/\\$31.00 ©2022 IEEE, 2022 International Conference on Computer Communication and Informatics (ICCCI) | 978-1-6654-8035-2/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/ICCCI54379.2022.974100
8. Peerawoot Rattanawichai, Thipwan Fangsuwannarak, Supanut Laohawiroj (2021), Monitoring System of Smart Cassava Farm with Solar Energy by Using Internet of Things, 2021 International Conference on Power, Energy and Innovations (ICPEI 2021) October 20-22, 2021, Nakhon Ratchasima, Thailand
978-1-6654-0216-3/21/\\$31.00 ©2021 IEEE, 2021 International Conference on Power, Energy and Innovations (ICPEI) | 978-1-6654-0216-3/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/ICPEI52436.2021.9690659
9. C.H. Vanipriya, Maruyi, Subhash Malladi, Gaurav Gupta (2021), Artificial intelligence enabled plant emotion xpresser in the development hydroponics system, Materials Today: Proceedings 45 (2021) 5034–5040,
<https://doi.org/10.1016/j.matpr.2021.01.512>, 2214-7853/\Ó 2021
10. 10 Shanhong Zhang, Yu Guo, Shuai Li, Zhixin Ke, Huajian Zhao, Jinqi Yang, Yang Wang, Daoliang Li, Liang Wang, Wenhua Yang, Zhili Zhang (2021), Investigation on an environment monitoring system for a combination of hydroponics and aquaculture in the greenhouse, <https://doi.org/10.1016/j.inpa.2021.06.006> 2214-3173.
11. Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao (2020), IEEE communications

- surveys & Tutorials, vol. 22, no. 3, third quarter 2020, Federated Learning in Mobile Edge Networks: A Comprehensive Survey, IEEE communications surveys & tutorials, VOL. 22, NO. 3, Third Quarter 2020, Digital Object Identifier 10.1109/COMST.2020.2986024
12. Yuwei Sun, Member, IEEE, Hideya Ochiai, Member, IEEE, and Hiroshi Esaki (2022), Decentralized Deep Learning for Multi-Access Edge Computing: A Survey on Communication Efficiency and Trustworthiness, IEEE Transactions on Artificial Intelligence, Vol. 3, No. 6, December 2022 963
13. Manav Mehra, Sameer Saxena, Suresh Sankaranarayanan, Rijo Jackson Tom, M. Veeramanikandan (2018), IoT based hydroponics system using Deep Neural Networks, Computers and Electronics in Agriculture 155 (2018) 473–486, <https://doi.org/10.1016/j.compag.2018.10.015>
14. Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan, (2019), Adaptive Federated Learning in Resource-Constrained Edge Computing Systems, IEEE Journal on Selected Areas in Communications, Vol. 37, NO. 6, JUNE 2019, Digital Object Identifier 10.1109/JSAC.2019.2904348
15. B. Custers, A. Sears, F. Dechesne, I. Georgieva, T. Tani, and S. van der Hof, EU Personal Data Protection in Policy and Practice. Springer, 2019.
16. M. G. Poirot, P. Vepakomma, K. Chang, J. Kalpathy-Cramer, R. Gupta, and R. Raskar, “Split learning for collaborative deep learning in healthcare,” 2019, arXiv:1912.12115
17. Antonis Tzounis, Nikolaos Katsoulas, Thomas Bartzanas, Constantinos Kittas (2017), Internet of Things in agriculture, recent advances and future challenges, Article in Biosystems Engineering, December 2017, DOI: 10.1016/j.biosystemseng.2017.09.007, <https://www.researchgate.net/publication/321331354.584>

18. Siow Eugene, Thanassis Tiropanis, Wendy Hall (2018), Analytics for the Internet of Things: A Survey, ACM Computing Surveys, Vol. 1, No. 1, Article 1. Publication date: January 2018.<https://www.researchgate.net/publication/326171965.587>
19. Anna Triantafyllou, Dimosthenis C. Tsouros, Panagiotis G. Sarigiannidis, Stamatia Bibi (2019), An Architecture model for Smart Farming, Conference: 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), DOI: 10.1109/DCOSS.2019.00081, <https://www.researchgate.net/publication/335362251592>
20. Bai Xuebing, Xinxing Li, Zetian Fu, Xiongjie Lv, Lingxian Zhang (2017), A fuzzy clustering segmentation method based on neighbourhood grayscale information for defining cucumber leaf spot disease images, Computers and Electronics in Agriculture, Volume 136, 15 April 2017, Pages 157-165, <https://doi.org/10.1016/j.compag.2017.03.004>.
21. Andonovic Ivan, Craig Michie, Philippe Cousin, Ahmed Janati, Congduc Pham, Mamour Diop (2018), Precision Livestock Farming Technologies, 2018 Global Internet of Things Summit (GloTS), /18/c 2018 IEEE
22. Fore Martin, Kevin Frank, Tomas Norton, Eirik Svendsen, Jo Arve Alfredsen, Tim Dempster, Harkaitz Eguraun, Win Watson, Annette Stahl, Leif Magne Sunde, Christian Schellewald, Kristoffer R. Skien, Morten O. Alver, Daniel Berckmans (2018), Precision fish farming: A new framework to improve production in aquaculture, biosystems engineering 173(2018) 176e193, <https://doi.org/10.1016/j.biosystemseng.2017.10.014>, <http://creativecommons.org/licenses/by/4.0/>).
23. Inkyu Sa, Chris Lehnert, Andrew English, Chris McCool, Feras Dayoub, Ben Upcroft, and Tristan Perez (2017), Peduncle Detection of Sweet Pepper for Autonomous Crop

- Harvesting Combined Color and 3-D Information, IEEE Robotics and Automation Letters, Vol. 2, No. 2, April 2017 765, 2377-3766 c 2017 IEEE.
24. Deepradit Siraprapha, Roongrat Pisuchpen, Pornthipa Ongkunaruk (2017), The Harvest Planning of Aromatic Coconut by Using Monte Carlo Simulation, 2017 4th International Conference on industrial engineering and Applications, 978-1-5090-6775-6/17/c 2017 IEEE.
 25. V. Palazzi, F. Gelati, U. Vaglioni, F. Alimenti, P. Mezzanotte, L. Roselli (2019), Leaf-Compatible Autonomous RFID-based Wireless Temperature Sensors for Precision Agriculture, 2019 IEEE Topical Conference on Wireless Sensors and Sensor Networks (WiSNet), 978-1-5386-5953-3/19/ 2019 IEEE
 26. Nayyar, A. & Puri, V. (2016). Smart farming: IoT based smart sensors agriculture stick for live temperature and moisture monitoring using Arduino, cloud computing and solar technology. In Proc. of The International Conference on Communication and Computing Systems (ICCCS-2016) (pp. 9781315364094-121).
 27. Mahdi Maktabdar Oghaz, Manzoor Razaak, Hamideh Kerdegari, Vasileios Argyriou, Paolo Remagnino (2019), Scene and Environment Monitoring Using Aerial Imagery and Deep Learning, 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), IEEE Computer Society, 2325-2944/19/2019 IEEE, DOI 10.1109/DCOSS.2019.00078
 28. Dimitrios Zorbas and Brendan O'Flynn (2019), A Network Architecture for High Volume Data Collection in Agricultural Applications, 2019 15th international conference on distributed computing sensor systems (DCSS), 2325-2944/19/\$31.00 @2019 IEEE DOI:10.1109/DCOSS.2019.00107
 29. Haoyu Niu, Tiebiao Zhao, Dong Wang and YangQuan Chen (2019), A UAV Resolution and Waveband Aware Path Planning for Onion Irrigation Treatments Inference, 2019

International Conference on Unmanned Aircraft Systems (ICUAS) Atlanta, GA, USA,
June 11-14, 2019, 978-1-7281-0332-7/19/\$31.00 2019 IEEE

30. Kamilaris Andreas, Feng Gaoy, Francesc X. Prenafeta-Bolduand Muhammad IntizarAliy (2016), Agri-IoT: A Semantic Framework for Internet of Things enabled Smart Farming Applications, DOI: 10.1109/WFIoT. 2016.7845467, <https://www.researchgate.net/publication/309557641>.
31. Bhangе Manisha, H.A.Hingoliwala (2015), Smart Farming: Pomegranate Disease Detection Using Image Processing, ScienceDirect, Procedia Computer Science 58(2015) 280 288 1877-0509 c 2015, The Authors. Published by Elsevier B.V., This is an open-access article under the CCBY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).Peer review under the responsibility of the organising committee of the Second International Symposium on Computer Vision and the Internet (VisionNet15), doi: 10.1016/j.procs.2015.08.022, Available online at www.sciencedirect.com. <https://www.researchgate.net/publication/283184986> [Accessed: July 2, 2017].
32. Choo Kim-Kwang Raymond, Stefanos Gritzalis, and Jong Hyuk Park (2018), Cryptographic Solutions for Industrial Internet-of-Things: Research Challenges and Opportunities, IEEE Transactions on industrial informatics, VOL. 14, No. 8, August 2018 3567.
33. Chi Yang, Deepak Puthal, Saraju P. Mohanty, and Elias Kougianos (2017), Big Sensing-Data Curation for the Cloud Is Coming, A promise of scalable cloud-data-centre mitigation for next-generation IoT and wireless sensor networks, Digital Object Identifier 10.1109/MCE.2017.2714695
34. Yansong Gao, Minki Kim, Sharif Abuadbba, Yeonjae Kim, Chandra Thapa, Kyuyeon Kim, Seyit A. Camtepe, Hyounghick Kim, and Surya Nepal (2020), End-to-End

35. Chi Yang, Deepak Puthal, Saraju P. Mohanty, and Elias Kougiianos (2017), Big Sensing-Data Curation for the Cloud Is Coming, A promise of scalable cloud-data-centre mitigation for next-generation IoT and wireless sensor networks, Digital Object Identifier 10.1109/MCE.2017.2714695
36. Muhammad Asad Saleem, Khalid Mahmood, and Saru Kumari (2020), Comments on AKM-IoV: Authenticated Key Management Protocol in Fog Computing-Based Internet of Vehicles Deployment, IEEE Internet of Things Journal, Vol. 7, No. 5, May 2020
37. Subhadeep Sarkar, Subarna Chatterjee, and Sudip Misra (2018), Assessment of the Suitability of Fog Computing in the Context of Internet of Things, IEEE Transactions on cloud computing, VOL. 6, NO. 1, January-March 2018, 2168-7161 © 2015 IEEE, Digital Object Identifier no. 10.1109/TCC.2015.2485206.
38. S. Dalmini and M. Ventura, "Resource management in fog computing: Review," in Proc. IEEE Int. Conf. Adv. Big Data Comput. Data Commun. Syst. (icABCD), 2019, pp. 1–7.
39. Ashkan Yousefpour, Fatemeh Jalali, Caleb Fung, Amirreza Niakanlahiji, Tam Nguyen, Jian Kong, Krishna Kadiyala, Jason P. Jue (2019), All One Needs to Know about Fog Computing and Related Edge Computing Paradigms, A Complete Survey, arXiv:1808.05283v3 [cs.NI] 13 Feb 2019.
40. Davydow, A., Chuprikov, P., Nikolenko, S. I., & Kogan, K. (2021). Competitive buffer management for packets with latency constraints. *Computer Networks*, 189, 107942.
41. Nugur, A., Pipattanasomporn, M., Kuzlu, M., & Rahman, S. (2018). Design and development of an IoT gateway for smart building applications. *IEEE Internet of Things Journal*.

42. Mohanty, J., Mishra, S., Patra, S., Pati, B., & Panigrahi, C. R. (2021). IoT Security, Challenges, and Solutions: A Review. *Progress in Advanced Computing and Intelligent Engineering*, 493-504.
43. Waheed Javed, Syeda um e Rubab, Gulnaz Parveen, Sidra Ikram, Khawaja Ubaid UR Rehman, Muhammad Danish (2021), A Review on Fog Computing for the Internet of Things, 2021 International Conference on Innovative Computing (ICIC) | 978-1-6654-0091-6/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/ICIC53490.2021.9692966
44. Zainab Salih Ageed, Rowaida Khalil Ibrahim, Subhi R. M. Zeebaree, Hanan M. Shukur, Mohammed A. M.Sadeeq, Ahmed Alkhayyat (2021), Comprehensive Study of Moving from Grid and Cloud Computing Through Fog and Edge Computing towards Dew Computing, 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA) | 978-1-6654-9461-8/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/IICETA51758.2021.9717894.
45. Rajni Jindal, Neetesh Kumar, Hitesh Nirwan (2020), MTFCT: A task offloading approach for fog computing and cloud computing, 978-1-7281-2791-0/20/\\$31.00_c 2020 IEEE.
46. Edge computing vs fog computing: Definitions and enterprise uses, *Cisco*.
<https://www.cisco.com/c/en/us/solutions/enterprise-networks/edge-computing.html> [Accessed Oct. 20, 2022].
47. Jaeun Cho, Younghan Kim, (2021), A Design of Serverless Computing Service for Edge Clouds, 2021 International Conference on Information and Communication Technology Convergence (ICTC) | 978-1-6654-2383-0/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/ICTC52510.2021.9621162
48. Jungae Park, Unho Choi, Seungwoo Kum, Jaewon Moon, Kyungyong Lee (2021), Accelerator-Aware Kubernetes Scheduler for DNN Tasks on Edge Computing

- Environment, 2021 ACM/IEEE 6th Symposium on Edge Computing (SEC), ACM ISBN 978-1-4503-8390-5/21/12. <https://doi.org/10.1145/3453142.3491411>
49. Marco Savi, Daniele Santoro, Katarzyna Di Meo, Daniele Pizzolli, Miguel Pincheira, Raffaele Giaffreda, Silvio Cretti, Seung-woo Kum, and Domenico Siracusa. 2020. A Blockchain-based Brokerage Platform for Fog Computing Resource Federation. In *2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. 147–149. <https://doi.org/10.1109/ICIN48450.2020.9059337>.
50. Francesc Guim, Thijs Metsch, Hassnaa Moustafa, Timothy Verrall, David Carrera, Nicola Cadenelli, Jiang Chen, David Doria, Chadie Ghadie, and Raül González Prats, (2022), Autonomous Lifecycle Management for Resource-Efficient Workload Orchestration for Green Edge Computing, *IEEE Transactions on Green communications and networking*, VOL. 6, NO. 1, March 2022, Digital Object Identifier 10.1109/TGCN.2021.3127531
51. Hongzhi Guo, Jiajia Liu, Huiling Qin, and Haibin Zhang. 2017. Collaborative Computation Offloading for Mobile-Edge Computing over Fiber-Wireless Networks. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. 1–6. <https://doi.org/10.1109/GLOCOM.2017.8254982>
52. Md Delowar Hossain, Luan NT Huynh, Tangina Sultana, Tri DT Nguyen, Jae Ho Park, Choong Seon Hong, and Eui-Nam Huh. 2020. Collaborative task offloading For overloaded mobile edge computing in small-cell networks. In *2020 International Conference on Information Networking (ICOIN)*. IEEE, 717–722.
53. Anousheh Gholami, John S. Baras, (2021), Collaborative Cloud-Edge-Local Computation Offloading for Multi-Component Applications, 2021 ACM/IEEE 6th Symposium on Edge Computing (SEC), ACM ISBN 978-1-4503-8390-5/21/12\\$.15.00 <https://doi.org/10.1145/3453142.3493515>

54. Phu Lai, Qiang He, Xiaoyu Xia, Feifei Chen, Mohamed Abdelrazek, John Grundy, John Hosking, Yun Yang (2022), Dynamic User Allocation in Stochastic Mobile Edge Computing Systems, 2022 IEEE World Congress on Services (SERVICES) | 978-1-6654-8131-1/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/SERVICES55459.2022.00030
55. Arpan Majumder, S. Mohan Kumar, D. V. Ashoka and A.Shajin Nargunam (2021), Resource Allocation Techniques in Edge/Fog Computing, 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT) | 978-1-7281-5791-7/20/\\$31.00 ©2021 IEEE | DOI: 10.1109/ICAECT49130.2021.9392422
56. Salmane Douch, Mohamed Riduan Abid, Khalid Zine-Dine, Driss Bouzidi, and Driss Benhaddou, (2022), Edge Computing Technology Enablers: A Systematic Lecture Study, Digital Object Identifier 10.1109/Access.2022.3183634
57. Meini Pan, Zhihua Li, (2021) Multi-user Computation Offloading Algorithm for Mobile Edge Computing, 2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT) | 978-1-6654-3757-8/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/CECIT53797.2021.00140
58. Quyuan Wang, Songtao Guo, Senior Member, IEEE, Jiadi Liu, Chengsheng Pan, and Li Yang (2022), Profit Maximization Incentive Mechanism for Resource Providers in Mobile Edge Computing IEEE Transactions on services computing, Vol. 15, No. 1, January/February 2022, Digital Object Identifier no. 10.1109/TSC.2019.2924002
59. Bo Li, Qiang He, Guangming Cui, Xiaoyu Xia, Feifei Chen, Hai Jin, and Yun Yang, (2022), READ: Robustness-Oriented Edge Application Deployment in Edge Computing Environment, IEEE Transactions on services computing, VOL. 15, NO. 3, May/June 2022, Digital Object Identifier no. 10.1109/TSC.2020.3015316

60. Qing Li, Xiao Ma, Ao Zhou, Xiapu Luo, Fangchun Yang, and Shangguang Wang (2021), User-Oriented Edge Node Grouping in Mobile Edge Computing, IEEE Transactions on mobile computing, 2021, DOI 10.1109/TMC.2021.3139362.
61. A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud Migration: A Case Study of Migrating an Enterprise IT System to IaaS," in 2010 IEEE 3rd International Conference on Cloud Computing, Jul. 2010, pp. 450–457. doi 10.1109/CLOUD.2010.37.
62. O. Min, C. Park, J. Lee, J. Cho, and H. Kim, "Issues on supporting public cloud virtual machine provisioning and orchestration," in 13th International Conference on Advanced Communication Technology (ICACT2011), Feb. 2011, pp. 270–273.
63. B. M. R. Wilson, B. Khazaei, and L. Hirsch, "Enablers and Barriers of Cloud Adoption among Small and Medium Enterprises in Tamil Nadu," in 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), Nov. 2015, pp. 140–145. doi: 10.1109/CCEM.2015.21.
64. M. Bahrami, "Cloud Computing for Emerging Mobile Cloud Apps," in 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, Mar. 2015, pp. 4–5. doi 10.1109/MobileCloud.2015.40.
65. A. Hendre and K. P. Joshi, "A Semantic Approach to Cloud Security and Compliance," in 2015 IEEE 8th International Conference on Cloud Computing, Jun. 2015, pp. 1081–1084. doi 10.1109/CLOUD.2015.157.
66. M. Aazam and E. N. Huh, "Inter-cloud Media Storage and Media Cloud Architecture for Inter-cloud Communication," in 2014 IEEE 7th International Conference on Cloud Computing, Jun. 2014, pp. 982–985. doi:10.1109/CLOUD.2014.151.
67. Chard, K. Chard, K. Bubendorfer, L. Lacinski, R. Madduri, and I. Foster, "Cost-Aware Elastic Cloud Provisioning for Scientific Workloads," in 2015 IEEE 8th International

- Conference on Cloud Computing, Jun. 2015, pp. 971–974. Doi: 10.1109/CLOUD.2015.130.
68. D. A. Rodríguez-Silva, L. Adkinson-Orellana, F. J. González-Castaño, I. Armiño-Franco, and D. González-Martínez, “Video Surveillance Based on Cloud Storage,” in 2012 IEEE Fifth International Conference on Cloud Computing, Jun. 2012, pp. 991–992, Doi 10.1109/CLOUD.2012.44.
69. Xiaofei Wang, Yiwen Han, Chenyang Wang, Qiyang Zhao, Xu Chen, Min Chen (2019), In-Edge AI: Intelligentizing Mobile Edge Computing, Caching, IEEE 2019, <https://arxiv.org/abs/1809.07857> [Accessed: May 20, 2019]
70. M.D. Muzakkir Hussain and M.M. S. Beg (2019), Using Vehicles as Fog Infrastructures for Transportation Cyber-Physical Systems (T-CPS): Fog Computing for Vehicular Networks, International Journal of Software Science and Computational Intelligence, Volume 11 Issue 1 January-March 2019, <https://orcid.org/0000-0002-6371-2545>
70. Mudhakar Srivatsa (2018), AI @ Edge, IEEE CIC 2018 Tutorial, [www.sis.pitt.edu/lersais/cic/2018/resources/AI @ edge.cic2018.pdf](http://www.sis.pitt.edu/lersais/cic/2018/resources/AI%20at%20edge.cic2018.pdf) [Accessed: May 20, 2019]
71. Alzubi, J., Nayyar, A., & Kumar, A. (2018, November). Machine learning from theory to algorithms: an overview. In Journal of Physics: conference series (Vol. 1142, No. 1, p. 012012). IOP Publishing.
72. Kim Jongmin, Youngryel Ryu, Chongya Jiang, Yorum Hwang (2019), Continuous observation of vegetation canopy dynamics using an integrated low-cost, near-surface remote sensing system, Agricultural and Forest Meteorology 264 (2019) 164177, <https://doi.org/10.1016/j.agrformet.2018.09.014>.
73. Franch Belen, Andres E. Santamara-Artigas, Pierre Guillevic, Jean-Claude Roger, Eric F. Vermote, and Sergii Skakun (2019), Evaluation of Near-Surface Air Temperature From

Reanalysis Over the United States and Ukraine: Application to Winter Wheat Yield Forecasting, IEEE Journal of selected topics in applied earth observations and remote sensing, DOI:10.1109/JSTARS.2019.2902479.

74. Jang Seung-Hwan, Chang Ho Yu (2017), A Study on Internet of Things (IoT): Users Reuse Intention Using Technology Acceptance Model in Korea, International Journal of Business and Management Science, Published by the Society for Alliance, Fidelity and Advancement (SAFA), ISSN 1985-692X.
75. Tullo Emmanuela, Ilaria Fontana, Alessia Diana, Tomas Norton, Daniel Berckmans, Marcella Guarino (2017), Application note: Labelling, a methodology to develop a reliable algorithm in PLF, Computers and Electronics in Agriculture 142 (2017) 424428, <http://dx.doi.org/10.1016/j.compag.2017.09.030>
76. Jayme Garcia Arnal Barbedo (2017), A review on the main challenges in automatic plant disease identification based on visible range images, Biosystems Engineering Volume 144, April 2016, Pages 52-60, <https://doi.org/10.1016/j.biosystemseng.2016.01.017>.
77. Eitzinger Anton, James Cock, Karl Atzmanstorfer, Claudia R. Binder, Peter Laderach, Osana Bonilla-Findji, Mona Bartling, Caroline Mwongera, Leo Zurita, Andy Jarvis (2019), GeoFarmer: A monitoring and feedback system for agricultural development projects, Computers and Electronics in Agriculture 158 (2019)109-121, <https://doi.org/10.1016/j.compag.2019.01.049>.
78. McGechan. M.B, A. Barnes, R. Fychan, C.L. Marley (2017), A simulation modelling study of water pollution caused by outwintering of Cows, Computers and Electronics in Agriculture 142 (2017) 397405, <http://dx.doi.org/10.1016/j.compag.2017.09.022>, 0168-1699/ c 2017 Elsevier B.V.

79. Augusto J. V. Neto, Zhongliang Zhao, Joel J. P. C. Rodrigues, Hugo Barros Camboim, and Torsten Braun (2018), Fog-based crime-assistance in smart IoT transportation system, a special section on cyber-physical-social computing and networking, digital object identifier 10.1109/access.2018.2803439
80. Behroozi-Khazaei Nasser, Mohammad Reza Maleki (2017), A robust algorithm based on colour features for grape cluster Segmentation, Computers and Electronics in Agriculture 142 (2017) 4149, <http://dx.doi.org/10.1016/j.compag.2017.08.025>, 0168-1699/ 2017 Elsevier B.V.
81. Li Hua, Yan Qian, Peng Cao, Wenqing Yin, Fang Dai, Fei Hu, ZhijunYan(2017), the Calculation method of surface shape feature of rice seed based on point Cloud, Computers and Electronics in Agriculture 142 (2017) 416423, <https://doi.org/10.1016/j.compag.2017.09.009>, 0168-1699/ 2017 Elsevier B.V.
82. Vaughan John, Peter M Green, Michael Salter¹, Bruce Grieve and Krikor B Ozanyan (2017), Floor Sensors of Animal Weight and Gait for Precision Livestock Farming, DOI:10.1109/ICSENS.2017.8234202, 2017 IEEE Sensors, Electronic ISBN: 978-1-5090-1012-7 c 2017 IEEE.
83. Dhanaraju, M, Chenniappan . P, Ramalingam. K, Pazhanivelan .S, Kaliaperumal . R, (2022), Smart Farming: Internet of Things (IoT)-Based Sustainable Agriculture. Agriculture 2022, 12, 1745. <https://doi.org/10.3390/agriculture12101745>
84. Godwin Idoje, Tasos Dagiuklas, Muddesar Iqbal, (2021), Survey for smart farming technologies: Challenges and issues, <https://doi.org/10.1016/j.compeleceng.2021.107104>
85. Liakos, K.G.; Busato, P.; Moshou, D.; Pearson, S.; Bochtis, D. (2018), Machine Learning in Agriculture: A Review. Sensors 2018, 18, 2674. <https://doi.org/10.3390/s18082674>

86. Savvas, D., Gruda, N., (2018), Application of soilless culture technologies in the modern greenhouse industry - A review. *Eur. J. Hortic. Sci.* 83, 280–293.
87. Kozai, T. (2018), Current Status of Plant Factories with Artificial Lighting (PFALs) and Smart PFALs. In: Kozai, T. (eds) *Smart Plant Factory*. Springer, Singapore. https://doi.org/10.1007/978-981-13-1065-2_1
88. Marques, G., Aleixo, D., Pitarma, R. (2019), Enhanced Hydroponic Agriculture Environmental Monitoring: An Internet of Things Approach. In, et al. *Computational Science – ICCS 2019*. ICCS 2019. *Lecture Notes in Computer Science* (), vol 11538. Springer, Cham. https://doi.org/10.1007/978-3-030-22744-9_51
89. Mouroutoglou, C.; Kotsiras, A.; Ntatsi, G.; Savvas, D.(2021), Impact of the Hydroponic Cropping System on Growth, Yield, and Nutrition of a Greek Sweet Onion (*Allium cepa* L.) Landrace. *Horticulturae* 2021, 7, 432. <https://doi.org/10.3390/horticulturae7110432>
90. Angelika Lindberg, Rachel Logan, Hannah Marron, Bethany Brinkman, Dr Michelle Gervasio, Dr BryanA (2021), Comprehensive Guide to Sweet Briar College's Greenhouse Hydroponics System (2021), *Systems and Information Engineering Design Symposium (SIEDS)* | 978-1-6654-1250-6/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/SIEDS52267.2021.9483761
91. Dong-Hee Noh, Tae-Hwan Ko, Ahhyeon Hong, Kyeong-Hun Kim, Seok-Bong Noh (2021), Faulty Node Detection Method in Wireless Sensor Network in Seedling for Hydroponics, 2021 Twelfth International Conference on Ubiquitous and Future Networks (ICUFN) | 978-1-7281-6476-2/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/ICUFN49451,2021, 9528646
92. Ms Swapnil Verma and Dr Sushopti D.Gawade (2021), A machine learning approach for prediction system and analysis of nutrients uptake for better crop growth in the

- Hydroponics system, 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) | 978-1-7281-9537-7/20/\\$31.00 ©2021 IEEE | DOI: 10.1109/ICAIS50930.2021.9395956
93. Geetali Saha (2021), Technological Influences on Monitoring and Automation of the Hydroponics System, 2021 Innovations in Power and Advanced Computing Technologies (i-PACT) | 978-1-6654-2691-6/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/i-PACT52855.2021.9696519
94. R.Vidhya, Dr K.Valarmathi (2018), Survey on Automatic Monitoring of Hydroponics Farms Using IoT, Proceedings of the International Conference on Communication and Electronics Systems (ICCES 2018), IEEE Xplore Part Number: CFP18AWO-ART; ISBN:978-1-5386-4765-3
95. Srinidhi H. K, Shreenidhi H. S, Vishnu G. S (2020), Smart Hydroponics system integrating with IoT and Machine learning algorithm, 2020 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT) | 978-1-7281-9772-2/20/\\$31.00 ©2020 IEEE | DOI: 10.1109/RTEICT49044.2020.9315549
96. Saraswathi, D., Manibharathy, P., Gokulnath, R., Sureshkumar, E.M., & Karthikeyan, K. (2018). Automation of Hydroponics Green House Farming using IOT. 2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA), 1-4.
97. Lakshmanan R, Djama M., Selvaperumal S. K., Abdulla R. (2020), Automated smart hydroponics system using internet of things. International Journal of Electrical and Computer Engineering (IJECE), Vol. 10, No. 6, December 2020, pp. 6389~6398 ISSN: 2088-8708, DOI: 10.11591/ijece.v10i6. pp 6389-6398

98. Ms Swapnil Verma and Dr Sushopti D.Gawade (2021), A machine learning approach for prediction system and analysis of nutrients uptake for better crop growth in the Hydroponics system, 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) — 978-1-7281-9537-7/20/\\$31.00 ©2021 IEEE — DOI: 10.1109/ICAIS50930.2021.9395956
99. Taha, M.F.; ElMasry, G.; Gouda, M.; Zhou, L.; Liang, N.; Abdalla, A.; Rousseau, D.; Qiu, Z. (2022), Recent Advances of Smart Systems and Internet of Things (IoT) for Aquaponics Automation: A Comprehensive Overview. *Chemosensors* 2022, 10, 303. <https://doi.org/10.3390/chemosensors10080303>
100. Usman, N.; Arief, P.; Gilang, L.; Erfan, R.; Hendra, P. (2018), Implementation IoT in System Monitoring Hydroponic Plant Water Circulation and Control. *Int. J. Eng. Technol.* 2018, 7, 122–126.
101. Jialong Li, Zhenyu Mao, Zhen Cao, Kenji Tei, Shinichi Honiden (2021), Self-adaptive Hydroponics Care System for Human-hydroponics Coexistence, 2021 IEEE 3rd Global Conference on Life Sciences and Technologies (LifeTech) | 978-1-6654-1875-1/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/LIFETECH52111.2021.9391909.
102. Nita Jaybhaye, Purva Tatiya, Avdut.Joshi, Sakshi. Kothari Jyoti.Tapkir (2022), 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT) | 978-1-6654-0118-0/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/ICSSIT53264.2022.9716287
103. Harshkumar Prakashbhai Thakor, Sailesh Iyer (2022), Development and Analysis of Smart Digi-farming Robust Model for Production Optimization in Agriculture, 6th International Conference on Computing for Sustainable Global Development (INDIACom), 978-93-80544-34-2\\$31.00 c 2019 IEEE.

104. Tharindu Madushan Bandara, Wanninayaka Mudiyansele, Mansoor RAZA (2020), Smart farm and monitoring system for measuring the Environmental condition using wireless sensor network - IOT Technology in farming, 2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA) | 978-1-7281-9437-0/20/\\$31.00 ©2020 IEEE | DOI: 10.1109/CITISIA50690.2020.9371830
105. Jaekuk Choi, Dongsun Lim, Sangwon Choi, Jeonghyeon Kim and Jonghoek Kim (2020), Light Control Smart Farm Monitoring System with Reflector Control, 20th International Conference on Control, Automation and Systems (ICCAS 2020) Oct. 13—16, 2020; BEXCO, Busan, Korea. 978-89-93215-19-9/20/\\$31.00 ©ICROS.
106. M.Pyngkodi, K.Thenmozhi, K.Nanthini, M.Karthikeyan, Dr Suresh Palarimath, V.Erajavignesh, G.Bala Ajith Kumar, (2022), Sensor-Based Smart Agriculture with IoT Technologies: A review, 2022 International Conference on Computer Communication and Informatics (ICCCI), Jan. 25 – 27, 2022, Coimbatore, INDIA, 978-1-6654-8035-2/22/\\$31.00 ©2022, IEEE, | DOI: 10.1109/ICCCI54379.2022.9741001
107. Vijaya Saraswathi R, Nikhil K, Sridharani J, Sri Harshitha M, Saranya Chowdary P, Mahanth Sai K(2022), Smart Farming: The IoT based Future Agriculture, 4th International Conference on Smart Systems and Inventive Technology (ICSSIT) | 978-1-6654-0118-0/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/ICSSIT53264.2022.9716331, IEEE explore Part Number: CFP22P17-ART; ISBN: 978-1-6654-0118-0.
108. Soumyajit Das, Neha Gupta, Sakshi Verma, Tanishka Kaushik (2022), LoRaWAN IoT based Precision Smart Farming Architecture with Data Visualization, 2022 International Conference for Advancement in Technology (ICONAT) | 978-1-6654-2577-3/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/ICONAT53423.2022.9725941.

109. Kavitha S, Varuna S, Ramya R (2016), A Comparative Analysis on Linear Regression and Support Vector Regression, 2016 Online International Conference on Green Engineering and Technologies (IC-GET), 978-1-5090-4556-3/16/\\$31.00 ©2016 IEEE
110. Liao Xiaoqun, Cao Nanlan, Ma Li, Kang Xiaofan (2019), Research on Short-term Load Forecasting Using XGBoost Based on Similar Days, 2019 International Conference on Intelligent Transportation, Big Data \& Smart City (ICITBS), 978-1-7281-1307-4/19/\\$31.00 ©2019 IEEE DOI: 10.1109/ICITBS.2019.00167
111. Giorgos Armeniakos, Georgios Zervakis, Dimitrios Soudris, Jörg Henkel, (2022), Hardware Approximate Techniques for Deep Neural Network Accelerators: a survey, ACM computing surveys, 2022, <https://doi.org/10.1145/3527156>
112. Richard Bray (2018), Hydroponics, How to pick the best hydroponic system and crops for homegrown food year-round, Independently published (12 Sept. 2018), pp. 37-47
113. Deepak Kumar, Vinay Kukreja (2022), A Symbiosis with Panicle-SEG Based CNN for Count the Number of Wheat Ears, 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO) Amity University, Noida, India. Oct 13-14, 2022, | 978-1-6654-7433-7/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/ICRITO56286.2022.9964954
114. G. Vinod, Dr G. Padmapriya (2022), An Adaptable Real-Time Object Detection for Traffic Surveillance using R-CNN over CNN with Improved Accuracy, 2022 International Conference on Business Analytics for Technology and Security (ICBATS) | 978-1-6654-0920-9/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/ICBATS54253.2022.9759030
115. Zirui Xu, Fuxun Yu, Zhuwei Qin, Chenchen Liu, and Xiang Chen (2021), DirectX: Dynamic Resource-Aware CNN Reconfiguration Framework for Real-Time Mobile Applications, IEEE Transactions on computer-aided design of integrated

circuits and systems, vol. 40, no. 2, February 2021, Digital Object Identifier

10.1109/TCAD.2020.2995813

116. Yuan Wang, Ying Li (2021), Research on Digital Media Image Data Tampering Forensics Technology Based on Improved CNN Algorithm, 2021 5th Asian Conference on Artificial Intelligence Technology (ACAIT) | 978-1-6654-2630-5/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/ACAIT53529.2021.9731182
117. Ke ZHANG (2021), Faster R-CNN Transmission Line Multi-target Detection Based on BAM, 2022 4th International Conference on Intelligent Control, Measurement and Signal Processing (ICMSP), 2022 4th International Conference on Intelligent Control, Measurement and Signal Processing (ICMSP) | 978-1-6654-8658-3/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/ICMSP55950.2022.9858980.
118. Mandar Kalbande, Punitkumar Bhavsar (2022), Performance Comparison of Deep Spiking CNN with Artificial Deep CNN for Image Classification Tasks, 2022 IEEE Region 10 Symposium (TENSYP) | 978-1-6654-6658-5/22/\\$31.00 © 2022 | DOI: 10.1109/TENSYP54529.2022.986455
119. Yu Lin (2022), Wafer Pattern Counting, Detection and Classification Based on Encoder-Decoder CNN Structure, 2022 IEEE International Conference on Electro Information Technology (EIT) | 978-1-6654-8009-3/22/\\$31.00 ©2022 IEEE | DOI: 10.1109/eIT53891.2022.9813870
120. Fatema-E- Jannat, Andrew R. Willis (2022), Improving Classification of Remotely Sensed Images with the Swin Transformer, Southeast Con 2022 | 978-1-6654-0652-9/22/\\$31.00 © 2022 IEEE | DOI: 10.1109/SoutheastCon48659.2022.9764016
121. Ibraheem Alhashim, Peter Wonka (2019), High-Quality Monocular Depth Estimation via Transfer Learning, arXiv:1812.11941 [cs.CV],
<https://doi.org/10.48550/arXiv.1812.11941>

122. H. Po-Han, M. Kevin, K. Johannes, A. Narendra, H. Jia-Bin, "Deepmvs: Learning multi-view stereopsis," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Pp. 2821-2830, 2018.
123. Mengnan Chen, Jiatao Liu, Yaping Zhang, Qiaosheng Feng (2022), RA-Swin: A RefineNet Based Adaptive Model Using Swin Transformer for Monocular Depth Estimation, 2022 8th International Conference on Virtual Reality (ICVR) | 978-1-6654-7911-0/22/\$31.00 ©2022 IEEE | DOI: 10.1109/ICVR55215.2022.9847988.
124. Wenzhu Xing, Karen Egiazarian (2022), Residual Swin Transformer Channel Attention Network for Image Demosaicing, 2022 10th European Workshop on Visual Information Processing (EUVIP) | 978-1-6654-6623-3/22/\$31.00 ©2022 IEEE | DOI: 10.1109/EUVIP53989.2022.9922679.
125. Ali Jamali, Fariba Mohammadimanesh, and Masoud Mahdianpari (2022), Wetland classification with Swin transformer using sentinel-1 and sentinel-2 data, IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium | 978-1-6654-2792-0/22/\$31.00 ©2022 IEEE | DOI: 10.1109/IGARSS46834.2022.9884602
IEEE Transactions on Instrumentation and Measurement, VOL. 71, 2022 4005615
126. Mushtaq Ahmad Khan, Muhammad Adnan, Abdul Basir, Shah Fahad, Aqsa Hafeez, Muhammad Hamzah Saleem, Manzoor ahmad5, Farhana Gul, Durrishahwar, Fazli Subhan, Saad Alamri, Mohamed Hashem, and Inayat Ur Rahman (2023), Impact of Tillage and potassium levels and sources on growth, yield and yield attributes of wheat, Pak. J. Bot., 55(1): 321-326, 2023. DOI: [http://dx.doi.org/10.30848/PJB2023-1\(30\)](http://dx.doi.org/10.30848/PJB2023-1(30)).
127. Gopichand, R.D. Singh, R.L. Meena, M.K. Singh, V.K. Kaul, Brij Lal, Ruchi Acharya, Ramdeen Prasad (2006), Effect of manure and plant spacing on crop growth, yield and oil-quality of *Curcuma aromatica* Salisb. mid hill of Western Himalaya, Industrial Crops

- and Products, Volume 24, Issue 2, September 2006, Pages 105-112,
<https://doi.org/10.1016/j.indcrop.2005.06.006>
128. Alex Wittenberg, James V. Anderson, Marisol T. Berti (2020), Crop growth and productivity of winter camelina in response to sowing date in the Northwestern Corn Belt of the USA, *Industrial Crops and Products*, Volume 158, 15 December 2020, 113036, <https://doi.org/10.1016/j.indcrop.2020.113036>
126. Ailing Lin, Bingzhi Chen, Jiayu Xu, Zheng Zhang, Guangming Lu, and David Zhang (2022), DS-TransUNet: Dual Swin Transformer U-Net for Medical Image Segmentation, *IEEE Transactions on instrumentation and measurement*, VOL. 71, 2022 4005615, DOI: 10.1109/TIM.2022.3178991.
127. Javeed, S. Narayan, A. A. Malik, A. Kumar, R. Rahman, S. Nisar, A. Akhter, S. A. Indrabi, and A. Sultan (2020), Role of information and communication technology in agriculture, || *Int. J. Curr. Microbiol. App. Sci*, ISSN: 2319- 7706, Special Issue 11, pp. 2028–2037, 2020.
128. Wang Xu, Zhou Kaili, Wang Tianlei (2021), Smart Farm Based on Six-Domain Model, 2021 IEEE 4th International Conference on Electronics Technology (ICET) | 978-1-7281-7673-4/20/\$31.00 ©2021 IEEE | DOI: 0.1109/ICET51757.2021.9451003
129. Wang Xu, Zhou Kaili, Hong Zhiyong (2021), Open Field Smart Planting System of Family Farm, 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS) | 978-1-6654-1256-8/21/\$31.00 ©2021 IEEE | DOI: 10.1109/ICCCS52626.2021.9449242
130. Juyoung Park, Aekyung Moon, Eunryung Lee (2021), Understanding IoT climate Data based Predictive Model for Outdoor Smart Farm, 2021 International Conference on Information and Communication Technology Convergence (ICTC) | 978-1-6654-2383-0/21/\$31.00 ©2021 IEEE | DOI: 10.1109/ICTC52510.2021.962097

131. Md Toufiqur Rahman, Sakib Mahmud, Yue Li, Md Abdur Rahman (2021), IoT based smart farming system to reduce manpower, wastage of time & natural resources in both traditional & urban mega farming, 2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE) | 978-1-6654-1596-5/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/AEMCSE51986.2021.00241
132. Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, Bingsheng He (2021), Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection, arXiv:1907.09693v6 [cs.LG], 1 Jul 2021
133. Tian Li, Anit Kumar Sahu, Ameet Talwalkar, Virginia Smith (2019), Federated Learning: Challenges, Methods, and Future Directions, arXiv 1908.07873v1 [cs.LG] 21 Aug 2019 28
134. Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, Kevin Chan (2019), Adaptive Federated Learning in Resource-Constrained Edge Computing Systems, arXiv:1804.05271v3 [cs.DC] 17 Feb 2019
135. Shaoxiong Ji, Shirui Pany, Guodong Longz, Xue Li, Jing Jiangz, Zi Huang (2019), Learning Private Neural Language Modeling with Attentive Aggregation, IJCNN 2019. International Joint Conference on Neural Networks. Budapest, Hungary. 14-19 July 2019
136. Jakub Konecny, H. Brendan McMahan, Felix X. Yu, Ananda Theertha Suresh Dave Bacon, Peter Richtarik (2017), Federated Learning Strategies for improving communication efficiency, arXiv 1610.05492v2 [cs.LG], 3, Oct 2017
137. Jin-Hyun Ahn, Osvaldo Simeone, and Joonhyuk Kang (2020), Cooperative learning via federated distillation over fading channels, 978-1-5090-6631-5/20/\\$31.00 c 2020 IEEE

- 138.Kai Yang, Tao Jiang, Yuanming Shi, and Zhi Ding (2020), Federated Learning via Over-the-Air Computation, IEEE Transactions on wireless communications, Vol. 19, No. 3, March 2020
- 139.Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Merouane Debbah (2018), Federated Learning for Ultra-Reliable Low-LatencyV2V Communications, 978-1-5386-4727-1/18/\\$31.00 c 2018 IEEE
- 140.Guan-Ying Huang and Ching-Hung Lee (2021), Federated Learning Architecture for Bearing Fault Diagnosis, 2021 International Conference on System Science and Engineering (ICSSE) | 978-1-6654-4848-2/21/\\$31.00 ©2021 IEEE | DOI:10.1109/ICSSE52999.2021.9538492
- 141.David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht and Joseph Dureau (2019), Federated Learning for keyword spotting, 978-1-5386-4658-8/18/\\$31.00 c 2019 IEEE, ICASSP 2019
- 142.Dequan Li, Yuheng Zhang, Yuejin Zhou (2021), Fast Distributed Stochastic Nesterov Gradient Descent Algorithm for Image Classification, 2021 China Automation Congress (CAC) | 978-1-6654-2647-3/21/\\$31.00 ©2021 IEEE | DOI: 10.1109/CAC53003.2021.9727635
- 143.Tao Sun, Linbo Qiao, Qing Liao, and Dongsheng Li (2021), Novel Convergence Results of Adaptive Stochastic Gradient Descents, IEEE Transactions on image processing, Vol. 30, 2021
- 144.Diederik P. Kingma and Jimmy Lei Ba (2017), Adam: A method for stochastic optimization, arXiv:1412.6980v9 [cs.LG], 30 Jan 2017
- 145.Arthava Ingle, (2020), <https://www.kaggle.com/atharvaingle/crop-recommendation-dataset>

146. Ming Qiu, Yiru Zhang, Tianqi Ma, Qingfeng Wu, and Fanzhu Jin (2020), Convolutional-neural-network-based Multilabel Text Classification for Automatic Discrimination of Legal Documents Sensors and Materials, Vol. 32, No. 8 (2020) 2659–2672 MYU Tokyo, <https://doi.org/10.18494/SAM.2020.2794>, ISSN 0914-4935 © MYU K.K.A.N.M. JuBaer.
147. Abu Sayem and Md. Ashikur Rahman (2019), Bangla Toxic Comment Classification (Machine Learning and Deep Learning Approach), Proceedings of the SMART–2019, IEEE Conference ID: 46866, 8th International Conference on System Modelling & Advancement in Research Trends, 22nd–23rd November 2019, College of Computing Sciences & Information Technology, Teerthanker Mahaveer University, Moradabad, India.
148. Emilio Ancillotti, Raffaele Bruno (2019), BDP-CoAP: Leveraging Bandwidth-Delay Product for Congestion Control in CoAP, 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), 978-1-5386-4980-0/19/\$31.00 ©2019 IEEE
149. T. V. Lakshman, and Upamanyu Madhow (1997), The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss IEEE/ACM Transactions on Networking, Vol. 5, NO. 3, June 1997, 1063–6692/97\$10.00 © 1997 IEEE
150. Wai Kay Leong, Zixiao Wang, Ben Leong (2017), TCP Congestion Control Beyond Bandwidth-Delay Product for Mobile Cellular Networks, ACM ISBN 978-1-4503-5422-6/17/12. . . \$15.00, <https://doi.org/10.1145/3143361.3143378>
151. Leonardo F. da Costa, Lia S. Furtado, Paulo H. G. Rocha, Paulo A. L. Rego, Fernando A. M. Trinta (2023), Time series prediction in IoT: a comparative study of federated versus centralised learning, 2023 IEEE 20th Consumer Communications & Networking

- Conference (CCNC) | 978-1-6654-9734-3/23/\\$31.00 ©2023 IEEE | DOI:
10.1109/CCNC51644.2023.10060467
152. Sogo Pierre Sanon, Rekha Reddy, Christoph Lipps and Hans Dieter Schotten (2023),
Secure Federated Learning: An Evaluation of Homomorphic Encrypted Network Traffic
Prediction, 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)
| 978-1-6654-9734-3/23/\\$31.00 ©2023 IEEE | DOI:
10.1109/CCNC51644.2023.10060116
153. Aili Wang, Yinghui Zhao, Linlin Yang, Haibin Wu, and Yuji Iwahori (2023),
Heterogeneous Defect Prediction Algorithm Combined with Federated Sparse
Compression, IEEE Access, Volume 11, 2023, Digital Object Identifier
10.1109/Access.2023.3253765
154. K. Wang, N. Deng and X. Li, "An Efficient Content Popularity Prediction of Privacy
Preserving Based on Federated Learning and Wasserstein GAN," in IEEE Internet of
Things Journal, vol. 10, no. 5, pp. 3786-3798, 1 March 1, 2023, doi:
10.1109/JIOT.2022.3176360.
155. Kristen Pereira, Aman Parikh, Pranav Kumar, Kailas Devadkar (2023), Healthcare
Diagnostics Service Using Federated Learning 2023 International Conference for
Advancement in Technology (ICONAT) | 978-1-6654-7517-4/23/\\$31.00 ©2023 IEEE |
DOI: 10.1109/ICONAT57137.2023.10080053
156. Y. Nguyen Tan, Vo Phuc Tinh, Pham Duc Lam, Nguyen Hoang Nam, and Tran Anh
Khoa (2023), A Transfer Learning Approach to Breast Cancer Classification in a Federated
Learning Framework, IEEE Access Volume 11, 2023, Digital Object Identifier
10.1109/ACCESS.2023.3257562

157. W. Sun, Y. Zhao, W. Ma, B. Guo, L. Xu and T. Q. Duong (2023), "Accelerating Convergence of Federated Learning in MEC with Dynamic Community," in IEEE Transactions on Mobile Computing, doi: 10.1109/TMC.2023.3241770.
158. Godwin Idoje, Christos Mouroutoglou, Tasos Dagiuklas, Anastasios Kotsiras, Iqbal Muddesar, Panagiotis Alefragkis, Comparative analysis of data using machine learning algorithms: A hydroponics system use case, Smart Agricultural Technology, Volume 4, 2023, 100207, ISSN 2772-3755, <https://doi.org/10.1016/j.atech.2023.100207>.
(<https://www.sciencedirect.com/science/article/pii/S2772375523000370>)
159. Adam James Hall, MadhavaJay, Tudor Cebere, Bogdan Cebere, Koen Lennart van der Veen, George Muraru, Tongye Xu, Patrick Cason, William Abramson, Ayoub Benaissa, Chnimay Shah, AlanAboudib, Th´eoRyffel, Kritika Prakash, Tom Titcombe, Varun Kumar Khare, Maddie Shang, Ionesio Junior, Animesh Gupta, Jason Paumier, Nahua Kang, Vova Manannikov, and AndrewTrask, SYFT0.5: A platform for universally deployable structured transparency, ICLR2021-Workshop on Distributed and Private Machine Learning (DPML), arXiv:2104.12385v2 [cs.LG] 27 Apr 2021
160. Godwin Idoje, Tasos Dagiuklas, Muddesar Iqbal, (2023), Federated Learning: Crop classification in a smart farm decentralised network, Smart Agricultural Technology, Volume 5, 2023, 100277, ISSN 2772-3755, <https://doi.org/10.1016/j.atech.2023.100277>.
161. Kutluk Bilge Bostanci, Salih Ulger (2022), comparison of spinach cultivation in floating hydroponic system and soil in the glasshouse and open field conditions, Mediterranean agricultural sciences, (2022) 35(1): 7-14, Doi: 10.29136/mediterranean.1061475.
162. Wimmerova L, Keken Z., Solcova O, Bartos L, Spacilova M. (2022), A Comparative LCA of Aeroponic Hydroponic, and Soil Cultivations of Bioactive Substance Producing Plants, <https://doi.org/10.3390/su14042421>.

163. Roberto S. Velazquez-Gonzalez, Adrian L. Garcia-Garcia, Elsa Ventura-Zapata, Jose Dolores Oscar Barceinas-Sanchez and Julio C. Sosa-Savedra, *Agriculture* 2022, 12(5), 646; <https://doi.org/10.3390/agriculture12050646>.
164. Ampim, P.A.Y., Obeng, E., Olvera-Gonzalez, E. (2022), Indoor Vegetable Production: An Alternative Approach to Increasing Cultivation. *Plants* 2022, 11, 2843. <https://doi.org/10.3390/plants/11212843>.
165. Marina Galdez de Castro Silva, Cristina Moll Hüther, Bruno Bernardo Ramos, Patrícia da Silva Araújo, Leonardo da Silva Hamacher, Carlos Rodrigues Pereira (2022), A Global overview of hydroponics: Nutrient Film Technique, *Revista Engenharia na Agricultura*, Viçosa, MG, DEA/UFV - DOI: 10.13083/reveng. v29i1.11679.

APPENDIX SECTION

Onion Bulb Diameter DNN code

main ▾ Onion-Bulb-predictions-using-DNN / FL_PTC_22_11_DNN.ipynb

Go to file ...

igoneal Add files via upload

Latest commit 9f012aa 5 minutes ago History

1 contributor

484 lines (484 sloc) 66.2 KB

<> 📄 Raw Blame ✎ 📄 🗑

```
In [1]: import math
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import matplotlib.pyplot as plt
from tensorflow.keras import Model
from tensorflow.keras import Sequential
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from tensorflow.keras.losses import MeanSquaredLogarithmicError
from tensorflow.keras.losses import MeanSquaredError, MeanAbsoluteError
from sklearn.metrics import accuracy_score
TRAIN_DATA_PATH = 'FL_G_dataset-PTC_training.csv'
TEST_DATA_PATH = 'FL_G_dataset-PTC_test.csv'
TARGET_NAME = 'Diameter of Bulb(mm)'
```

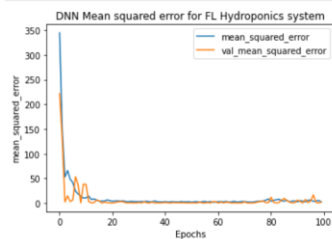
```
In [2]: # x_train = features, y_train = target
train_data = pd.read_csv(TRAIN_DATA_PATH)
test_data = pd.read_csv(TEST_DATA_PATH)
x_train, y_train = train_data.drop(TARGET_NAME, axis=1), train_data[TARGET_NAME]
x_test, y_test = test_data.drop(TARGET_NAME, axis=1), test_data[TARGET_NAME]
```

```
In [3]: """
Standard Scale test and train data
Z - Score normalization
"""
def scale_datasets(x_train, x_test):
    standard_scaler = StandardScaler()
    x_train_scaled = pd.DataFrame(standard_scaler.fit_transform(x_train), columns=x_train.columns)
    x_test_scaled = pd.DataFrame(standard_scaler.transform(x_test), columns=x_test.columns)
    return x_train_scaled, x_test_scaled
x_train_scaled, x_test_scaled = scale_datasets(x_train, x_test)
```

```
In [4]: hidden_units1 = 160
hidden_units2 = 320
hidden_units3 = 256
learning_rate = 0.01
# Creating model using the Sequential in tensorflow
def build_model_using_sequential():
    model = Sequential([
        Dense(hidden_units1, kernel_initializer='normal', activation='relu'),
        Dropout(0.2),
        Dense(hidden_units2, kernel_initializer='normal', activation='relu'),
        Dropout(0.2),
        Dense(hidden_units3, kernel_initializer='normal', activation='relu'),
        Dense(1, kernel_initializer='normal', activation='linear')
    ])
    return model
# build the model
model = build_model_using_sequential()
```

```
In [5]: %time
# Loss function
#mse = MeanSquaredLogarithmicError()
mse = MeanSquaredError()
model.compile(
    loss=mse,
    optimizer=Adam(learning_rate=learning_rate),
    metrics=[mse]
)
# train the model
history = model.fit(
    x_train_scaled.values,
    y_train.values,
    epochs=100,
    batch_size=32,
    validation_split=0.2
)
```

```
In [6]: def plot_history(history, key):
plt.plot(history.history[key])
plt.plot(history.history['val_'+key])
plt.title("DNN Mean squared error for FL Hydroponics system")
plt.xlabel("Epochs")
plt.ylabel(key)
plt.legend([key, 'val_'+key])
plt.show()
# Plot the history
#plot_history(history, 'mean_squared_logarithmic_error')
plot_history(history, 'mean_squared_error')
```



Onion Bulb Diameter XGBoost code

main - xgboost-analysis-of-Onion-Bulb-diameter / Diam_Bulb_xgboost_AER-PTC.ipynb Go to file ...

igoneal Add files via upload Latest commit aaf6dc6 on Jun 15, 2022 History

1 contributor

460 lines (460 sloc) | 36.8 KB <> Raw Blame 📄 🔗 🗑️

```
In [1]: !pip install xgboost

Requirement already satisfied: xgboost in c:\users\igone\anaconda3\lib\site-packages (1.6.1)
Requirement already satisfied: numpy in c:\users\igone\anaconda3\lib\site-packages (from xgboost) (1.20.3)
Requirement already satisfied: scipy in c:\users\igone\anaconda3\lib\site-packages (from xgboost) (1.7.1)

In [2]: import xgboost as xgb
#from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score, KFold
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

In [3]: raw_dataset = pd.read_csv('AER_G_dataset-PTC.csv')
x = raw_dataset.iloc[:, :-1].values
y = raw_dataset.iloc[:, 11].values
raw_dataset

Out[3]:
```

	DAT(days)	Temp	WC	NL	N(PTC) (mg/g)	P(PTC) (mg/g)	K(PTC) (mg/g)	Ca(PTC) (mg/g)	Mg(PTC) (mg/g)	S(PTC) (mg/g)	Na(PTC) (mg/g)	Diameter of Bulb(mm)
0	1	18.3	0.11	5.75	33.56	4.90	37.29	7.25	2.13	1.93	0.77	16.03
1	2	19.8	0.22	5.79	34.38	4.91	38.62	7.46	2.14	1.95	0.78	16.04
2	3	18.3	0.31	5.83	34.60	5.01	39.81	7.69	2.15	1.97	0.79	16.05
3	4	16.9	0.42	5.87	35.61	5.02	40.65	7.82	2.16	1.99	0.80	16.06
4	6	18.9	0.56	5.91	36.62	5.03	42.15	7.01	2.18	2.00	0.81	16.07
...
86	89	29.5	15.18	10.23	43.02	7.56	50.64	14.06	1.83	3.72	1.99	103.39
87	90	27.9	15.21	10.34	43.00	7.64	50.43	14.07	1.84	3.73	2.00	103.47
88	91	27.1	15.27	10.48	42.82	7.68	50.27	14.08	1.85	3.74	2.01	103.49
89	92	26.1	15.30	10.51	42.61	7.72	49.94	14.09	1.86	3.75	2.02	103.52
90	93	26.9	15.37	10.59	42.56	7.74	49.38	14.10	1.87	3.76	2.03	103.66

91 rows x 12 columns

```
In [4]: xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.15)

In [9]: %%time
xgbr = xgb.XGBRegressor(verbosity=0)
#print(xgbr)

xgbr.fit(xtrain, ytrain)

score = xgbr.score(xtrain, ytrain)
print("Training score: ", score)

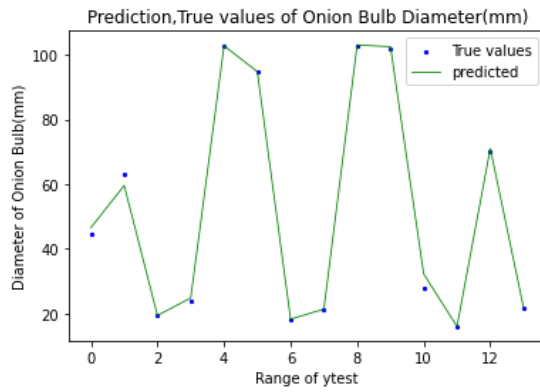
# - cross validation
scores = cross_val_score(xgbr, xtrain, ytrain, cv=5)
print("Mean cross-validation score: %.2f" % scores.mean())

kfold = KFold(n_splits=10, shuffle=True)
kf_cv_scores = cross_val_score(xgbr, xtrain, ytrain, cv=kfold)
print("K-fold CV average score: %.2f" % kf_cv_scores.mean())

ypred = xgbr.predict(xtest)
r2 = 1 - sum((ytest - ypred) ** 2) / sum((ytest - np.mean(ytest)) ** 2)
#r2 = r2_score(ytest, ypred)
mse = mean_squared_error(ytest, ypred)
print("MSE: %.2f" % mse)
print("RMSE: %.2f" % (mse**(1/2.0)))

Training score: 0.999999983677004
Mean cross-validation score: 0.99
K-fold CV average score: 0.99
MSE: 5.98
RMSE: 2.45
Wall time: 481 ms

In [6]: x_ax = range(len(ytest))
plt.title('Prediction, True values of Onion Bulb Diameter(mm)')
plt.scatter(x_ax, ytest, s=5, color="blue", label="True values")
plt.plot(x_ax, ypred, lw=0.8, color="green", label="predicted")
plt.legend()
plt.show()
```



```
In [7]: print('ypred =',ypred)

ypred = [ 21.528963 102.758156 21.689304 21.757694 16.08015 85.971634
 23.793789 57.49505 92.196365 103.29828 20.32324 21.799212
 100.547134 99.33185 ]
```

```
In [8]: print('r2_score = ',r2)

r2_score = 0.9957363169197625
```

main xgboost-analysis-of-Onion-Bulb-diameter / bar_charts_comparing R^2.ipynb

Go to file

igoneal Add files via upload

Latest commit aafdc66 on Jun 15, 2022 History

1 contributor

87 lines (87 sloc) 23.5 KB

Raw Blame

```
In [4]: import matplotlib.pyplot as plt
import numpy as np

# set width of bar
barWidth = 0.25
fig = plt.subplots(figsize=(15, 8))

# set height of bar
True_R = [0.998, 0.995, 0.997, 0.997]
Predicted_R = [0.995, 0.996, 0.981, 0.988]

# Set position of bar on X axis
br1 = np.arange(len(True_R))
br2 = [x + barWidth for x in br1]

# Make the plot
plt.bar(br1, True_R, color='r', width=barWidth,
        edgcolor='grey', label='Baseline_R^2')
plt.bar(br2, Predicted_R, color='g', width=barWidth,
        edgcolor='grey', label='Predicted_R^2')

# Adding Xticks
plt.xlabel('Hydroponic systems', fontweight='bold', fontsize=15)
plt.ylabel('R^2', fontweight='bold', fontsize=15)
plt.xticks([r + barWidth for r in range(len(True_R))],
           ['AER', 'AG', 'Floating', 'NFT'])
plt.title('comparison of Hydroponic systems R square values from the xgboost model')
plt.legend(loc='lower right')
plt.show()
```

Onion Bulb Diameter Split Learning model code for Data scientist

main - Onion-Bulb-Diameter-using-split-Learning / split_Learning_Onion_DS-AER_Adam-PTC.ipynb Go to file ...

igoneal Add files via upload Latest commit bf52bb9 on Jun 8, 2022 History

1 contributor

5063 lines (5063 sloc) | 742 KB <> Raw Blame ...

```
In [ ]: |pip install matplotlib

In [1]: import syft as sy
        %matplotlib inline
        duet = sy.join_duet(loopback=True)

        >>> Joining Duet

        >>> DISCLAIMER: Duet is an experimental feature currently in beta.
        >>> Use at your own risk.

        > ❤️ Love Duet? Please consider supporting our community!
        > https://github.com/sponsors/OpenMined

        >>> Punching through firewall to OpenGrid Network Node at:
        >>> http://ec2-18-218-7-180.us-east-2.compute.amazonaws.com:5000
        >>>
        >>> ...waiting for response from OpenGrid Network...
        >>> DONE!

/home/user-010/anaconda3/envs/pysyft/lib/python3.7/site-packages/aiortc/rtpcdtlstransport.py:211: CryptographyDeprecationWarning: This version of cryptography contains a temporary pyOpenSSL fallback path. Upgrade pyOpenSSL now.
  _openssl_assert(lib.SSL_CTX_use_certificate(ctx, self.cert_x509) == 1) # type: ignore
/home/user-010/anaconda3/envs/pysyft/lib/python3.7/site-packages/aiortc/rtpcdtlstransport.py:186: CryptographyDeprecationWarning: This version of cryptography contains a temporary pyOpenSSL fallback path. Upgrade pyOpenSSL now.
  value=certificate_digest(self.cert_x509), # type: ignore
        >>> CONNECTED!
```

```
In [2]: class SyNet(sy.Module):
        def __init__(self, torch_ref):
            super(SyNet, self).__init__(torch_ref=torch_ref)

            self.lin = self.torch_ref.nn.Linear(11,8)
        def forward(self, x):
            x = self.lin(x)
            return x

In [3]: import torch
        model1 = SyNet(torch)
        model1_ptr = model1.send(duet)

In [4]: opt1 = duet.torch.optim.SGD(params=model1_ptr.parameters(), lr=0.0000001)

In [5]: duet.store.pandas

Out[5]:
```

ID	Tags	Description	object_type
0	<UID: 44a3cc39ff664e4b61e3facbb6e547b> [dataset]	Aeroponics(AER) Independent features Temp,WC,N...	<class 'torch.Tensor'>

```
In [6]: data_pointer = duet.store[0]

In [7]: import numpy as np

In [8]: model2 = torch.nn.Linear(8,11)
        opt2 = torch.optim.SGD(params=model2.parameters(), lr=0.0000001)

        target = torch.tensor([[16.03, 6.04, 16.05, 16.06, 16.07, 16.08, 16.09, 16.1, 16.11, 16.11, 18.22],
                                [18.25, 18.26, 18.27, 18.28, 18.29, 18.3, 19.26, 19.32, 19.33, 19.43, 19.51],
                                [19.62, 19.73, 19.81, 19.95, 20.6, 20.61, 20.62, 20.43, 20.44, 21.23, 21.41],
                                [21.45, 21.53, 21.64, 21.67, 21.72, 21.75, 21.84, 21.86, 21.88, 21.92, 23.77],
                                [24.85, 25.79, 28.07, 32.47, 35.49, 35.85, 40.84, 44.59, 44.77, 46.21, 50.8],
                                [52.12, 53.58, 88.60, 81.63, 25.69, 98.70, 96.79, 32.84, 19.84, 91.85, 77],
                                [94.55, 94.12, 99.72, 99.74, 99.81, 100.53, 101.85, 101.92, 102.26, 102.38, 102.41],
                                [102.44, 102.59, 102.63, 102.71, 102.79, 102.81, 102.85, 103.31, 103.36, 103.39, 103.47]])

        target.shape
        #target.type

Out[8]: torch.Size([8, 11])
```

```
In [9]: import matplotlib.pyplot as plt
#epoch = 200
losses = []
predictions = []
for iter in range(250):
    opt1.zero_grad()
    opt2.zero_grad()

    activation_ptr = model1_ptr(data_pointer)
    activation = activation_ptr.clone().get(request_block=True)

    pred = model2(activation)

    #target = target.detach().numpy()
    loss = ((pred - target)**2).sum()

    loss.backward()

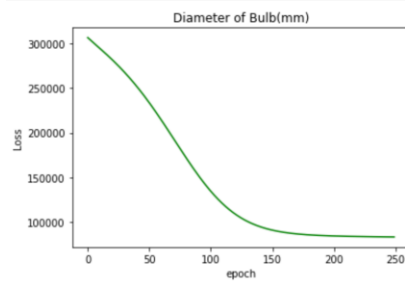
    grad_ptr = activation.grad.clone().send(duet)
    activation_ptr.backward(grad_ptr)

    opt1.step()
    opt2.step()
    losses.append(loss.tolist())
    predictions.append(pred.tolist())

print("losses: ",losses)
print('Predictions =',pred)

losses: [306680.1875]
Predictions = tensor([[ -0.5402, -0.2426,  0.3386, -4.9986,  0.5270,  4.4333,  2.4579, -1.7193,
  2.4888,  3.7654, -1.6816],
 [-0.5958, -0.2952,  0.5184, -5.2822,  0.4523,  4.6305,  2.5941, -1.9009,
  2.4389,  4.1201, -1.7130],
 [-0.5894,  0.0650,  0.7918, -5.2615,  0.6059,  4.9219,  2.8352, -1.5286,
  2.5999,  3.8446, -1.9398],
 [-0.6300,  0.4518,  1.0296, -5.3056,  0.8526,  5.3910,  3.1391, -1.2585,
  2.7746,  3.5706, -2.2287],
 [-0.7305,  0.4583,  1.4392, -5.7617,  0.7073,  5.8390,  3.4527, -1.6419,
  2.4776,  4.1201, -2.2826],
 [-0.7157,  1.0660,  1.7247, -5.9363,  0.9133,  6.3025,  3.9560, -0.9601,
  2.8739,  3.7407, -2.7292],
 [-0.7601,  1.2026,  1.9436, -6.2004,  0.9351,  6.6388,  4.2212, -0.9608,
  2.8982,  3.8926, -2.8874],
 [-0.7277,  1.3402,  2.2170, -6.3775,  0.7560,  6.6398,  4.4032, -0.7642,
  2.8407,  4.0393, -2.9377]], grad_fn=<AddmmBackward>)
losses: [306680.1875, 305454.6875]
```

```
In [10]: import matplotlib.pyplot as plt
#%matplotlib inline
#pred = pred.detach().numpy()
#pred = pred.numpy()
#Loss = loss.detach().numpy()
# print("Loss", Losses)
# print("List", Losses[0].tolist())
# print("pred", predictions[0].tolist())
# print("pred", predictions[0].ravel().tolist())
#model2 = model2.detach().numpy()
#target = target.detach().numpy()
plt.title('Diameter of Bulb(mm)')
plt.plot(losses, color='green')
plt.ylabel("Loss")
plt.xlabel("epoch")
plt.show()
```



Onion Bulb Diameter Split Learning model code for Data Owner

main [Onion-Bulb-Diameter-using-split-Learning / split_learning_Onion_DO-AER_Adam-PTC.ipynb](#) Go to file ...

igoneal Add files via upload Latest commit bf52bb9 on Jun 8, 2022 History

1 contributor

209 lines (209 sloc) | 6.87 KB <> Raw Blame ...

```
In [1]: import syft as sy
duet = sy.duet(loopback=True)

>>> Starting Duet

>>> DISCLAIMER: Duet is an experimental feature currently in beta.
>>> Use at your own risk.

>>> Love Duet? Please consider supporting our community!
>>> https://github.com/sponsors/OpenMined

>>> Punching through firewall to OpenGrid Network Node at:
>>> http://ec2-18-218-7-180.us-east-2.compute.amazonaws.com:5000
>>> ...waiting for response from OpenGrid Network...
>>> DONE!

>>> STEP 1: Send the following code to your Duet Partner!

import syft as sy
duet = sy.join_duet(loopback=True)

>>> Connecting...
/home/user-010/anaconda3/envs/pysyft/lib/python3.7/site-packages/aiortc/rtdctlsrtransport.py:211: CryptographyDeprecationWarning: This version of cryptography contains a temporary pyOpenSSL fallback path. Upgrade pyOpenSSL now.
value=certificate_digest(self._cert_x509), # type: ignore
>>> CONNECTED!

>>> DUET LIVE STATUS - Objects: 255 Requests: 0 Messages: 3495 Request Handlers: 1

In [2]: import torch
import numpy as np
#Define and send our dummy input data

data = torch.tensor([[1,18.3,0.11,5.75,33.56,4.9,37.29,7.25,2.13,1.93,0.77],
[2,19.8,0.22,5.79,34.38,4.91,38.62,7.46,2.14,1.95,0.78],
[3,18.3,0.31,5.83,34.6,5.01,39.81,7.69,2.15,1.97,0.79],
[4,16.9,0.42,5.87,35.61,5.02,40.65,7.82,2.16,1.99,0.8],
[6,18.9,0.56,5.91,36.62,5.03,42.15,7.01,2.18,2.0,0.81],
[7,17.0,0.63,5.95,37.64,5.08,46.37,8.16,2.2,2.02,0.82],
[8,17.5,0.75,5.99,38.65,5.1,48.25,8.23,2.2,2.04,0.83],
[9,17.8,0.84,6.03,38.5,5.12,50.81,8.46,2.24,2.07,0.84]])

data.send(duet, tags=["dataset"], description="Aeroponics(AER) Independent features Temp,WC,NL,N(UC),Fe(UC),B(UC)", pointable=True)
data.shape

Out[2]: torch.Size([8, 11])

In [3]: duet.requests.add_handler(action="accept")

In [4]: duet.store.pandas

Out[4]:
```

ID	Tags	Description	object type
0 <UID: 44a3cc39ff664fe4b61e3facbb6e547b>	[dataset]	Aeroponics(AER) Independent features Temp,WC,NL,N(UC),Fe(UC),B(UC)	<class 'torch.Tensor'>

Smart Farming Analysis using Federated Averaging Algorithm

main - Smart-Farming-analysis-using-Federated-averaging-algorithm / DO_crop.ipynb Go to file ...

 igoneal Add files via upload Latest commit 7eeb0dc on May 6, 2022 History

1 contributor

589 lines (589 sloc) | 24.6 KB <> Raw Blame Copy Share

```
In [4]: #!pip install xgboost
        !pip install syft==0.5.0
```



```

In [1]: import syft as sy

In [2]: duet = sy.launch_duet(loopback=True)

>>> Starting Duet

#### > DISCLAIMER: Duet is an experimental feature currently in beta.
#### > Use at your own risk.

> ❤️ Love Duet? Please consider supporting our community!
> https://github.com/sponsors/OpenMined

#### > Punching through firewall to OpenGrid Network Node at:
#### > http://ec2-18-218-7-180.us-east-2.compute.amazonaws.com:5000
#### >
#### > ...waiting for response from OpenGrid Network...
#### > DONE!

#### > STEP 1: Send the following code to your Duet Partner!

import syft as sy
duet = sy.join_duet(loopback=True)

#### > Connecting...
/home/user-010/anaconda3/envs/pysyft/lib/python3.7/site-packages/aiortc/rtpcdtlstransport.py:211: CryptographyDeprecationWarning: This version of cryptography contains a temporary pyOpenSSL fallback path. Upgrade pyOpenSSL now.
  _openssl_assert(lib.SSL_CTX_use_certificate(ctx, self._cert._x509) == 1) # type: ignore
/home/user-010/anaconda3/envs/pysyft/lib/python3.7/site-packages/aiortc/rtpcdtlstransport.py:186: CryptographyDeprecationWarning: This version of cryptography contains a temporary pyOpenSSL fallback path. Upgrade pyOpenSSL now.
  value=certificate_digest(self._cert._x509), # type: ignore
#### > CONNECTED!

#### > DUET LIVE STATUS - Objects: 8 Requests: 0 Messages: 8819 Request Handlers: 1

In [3]: from sklearn import datasets
import torch as th
import pandas as pd
import numpy as np
import torch.nn as nn
import time
#import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

#### > DUET LIVE STATUS - Objects: 0 Requests: 0 Messages: 0 Request Handlers: 0

In [4]: #dataset = pd.read_csv('Crop_Artharva.csv')
dat = pd.read_csv('Crop_data_Artharva12.csv')
dat1 = pd.read_csv('Crop_target_Artharva12.csv')
X = dat.iloc[:,0:5].values
y = dat1.iloc[:,0].values
print(X[0:5])
print(X)

```


1 contributor

1432 lines (1432 sloc) 87.9 KB

<> Raw Blame

In [1]: `import syft as sy`

Part 1: Join the Duet Server the Data Owner connected to

In [2]: `duet = sy.join_duet(loopback=True)`

```

>>> Joining Duet
>>> DISCLAIMER: Duet is an experimental feature currently in beta.
>>> Use at your own risk.

> Love Duet? Please consider supporting our community!
> https://github.com/sponsors/OpenMined

> Punching through firewall to OpenGrid Network Node at:
> http://ec2-18-218-7-180.us-east-2.compute.amazonaws.com:5000
>
> ...waiting for response from OpenGrid Network...
> DONE!

/home/user-010/anaconda3/envs/pysyft/lib/python3.7/site-packages/aiortc/rtcdtlstransport.py:211: CryptographyDeprecationWarning: This version of cryptography contains a temporary pyOpenSSL fallback path. Upgrade pyOpenSSL now.
  _openssl_assert(lib.SSL_CTX_use_certificate(ctx, self._cert_x509) == 1) # type: ignore
/home/user-010/anaconda3/envs/pysyft/lib/python3.7/site-packages/aiortc/rtcdtlstransport.py:186: CryptographyDeprecationWarning: This version of cryptography contains a temporary pyOpenSSL fallback path. Upgrade pyOpenSSL now.
  value=certificate_digest(self._cert_x509), # type: ignore
>>> CONNECTED!
    
```



Checkpoint 0 : Now STOP and run the Data Owner notebook until Checkpoint 1.

Part 2: Search for Available Data

In [3]: `# The data scientist can check the List of pointable data in Data Owner's duet store`
`duet.store.pandas`

	ID	Tags	Description	object_type
0	<UID: 0e3419be89044758b740b812bcd507df>	[Crops-data]	This is a dataset for Crops classification tem...	<class 'torch.Tensor'>
1	<UID: fed555467b904e7e85f501973e02ab65>	[Crops-target]	Labels for Crops: rice, maize, chickpea	<class 'torch.Tensor'>

In [4]: `# Data Scientist wants to get the crop dataset. (S)He needs a pointer to the data and`
`# a pointer to the target for prediction.`
`data_ptr = duet.store[0]`
`target_ptr = duet.store[1]`
`# data_ptr.requires_grad = True`
`# target_ptr.requires_grad = True`
`# data_ptr is a reference to the crop dataset remotely available on data owner's server`
`# target_ptr is a reference to the crop dataset LABELS`
`# remotely available on data owner's server`
`print(data_ptr)`
`print(target_ptr)`

<syft.proxy.torch.TensorPointer object at 0x7f3ec5fe1c50>
 <syft.proxy.torch.TensorPointer object at 0x7f3ec6208710>

Part 3: Perform Logistic Regression on Crop dataset

Now the data scientist can perform machine learning on the data that is in the Data Owner's duet server, without the owner having to share his/her data.

Basic analysis

First the data scientist needs to know some basic information about the dataset.

1. The length of the dataset
2. The input dimension
3. The output dimension

These information have to explicitly shared by the Data Owner. Let's try to find them in the data description.

In [5]: `print(duet.store.pandas["Description"][0])`
`print(duet.store.pandas["Description"][1])`

This is a dataset for Crops classification temperature, humidity, Ph level, Rainfall
 Labels for Crops: rice, maize, chickpea

Train model

```
In [6]: import torch
import xgboost as xgb
import time
from sklearn.metrics import mean_squared_error
```

```
In [7]: in_dim = 4
out_dim = 3
n_samples = 150
```

```
In [8]: class SyNet(sy.Module):
def __init__(self, torch_ref):
super(SyNet, self).__init__(torch_ref=torch_ref)
self.layer1 = self.torch_ref.nn.Linear(in_dim, 20)
self.layer2 = self.torch_ref.nn.Linear(20, 30)
self.out = self.torch_ref.nn.Linear(30, out_dim)
#self.flatten = self.torch_ref.nn.Flatten()
def forward(self, X111):
X111 = self.torch_ref.nn.functional.relu(self.layer1(X111))
X111 = self.torch_ref.nn.functional.relu(self.layer2(X111))
output = self.torch_ref.nn.functional.log_softmax(self.out(X111), dim=1)

#self.torch_ref.nn.Flatten(x)
return output

local_model = SyNet(torch)
```

```
In [9]: remote_model = local_model.send(duet)
```

Let's create an alias for our partner's torch called remote_torch so we can refer to the local torch as torch and any operation we want to do remotely as remote_torch. Remember, the return values from remote_torch are Pointers, not the real objects. They mostly act the same when using them with other Pointers but you can't mix them with local torch objects.

```
In [10]: remote_torch = duet.torch
```

```
In [11]: #remote_torch
```

```
In [12]: params = remote_model.parameters()
optim = remote_torch.optim.Adam(params=params, lr=0.01)
```

```
In [13]: print(remote_model.parameters())
```

```
<syft.proxy.syft.lib.python.ListPointer object at 0x7f3ec62080d0>
```

```
In [14]: print(optim)
```

```
<syft.proxy.torch.optim.AdamPointer object at 0x7f3eb58d7550>
```

```
In [15]: start_time = time.time()
def train(iterations, model, torch_ref, optim, data_ptr, target_ptr):
    losses = []
    for i in range(iterations):
        optim.zero_grad()
        output = model(data_ptr)
        loss = torch_ref.nn.functional.nll_loss(output, target_ptr.long())
        loss_item = loss.item()
        loss_value = loss_item.get(
            reason="To evaluate training progress", request_block=True, timeout_secs=5
        )
        if i % 10 == 0:
            print("Epoch", i, "loss", loss_value)
        losses.append(loss_value)
        loss.backward()
        optim.step()
    return losses
print("Computational time :- %s seconds" % (time.time() - start_time))
```

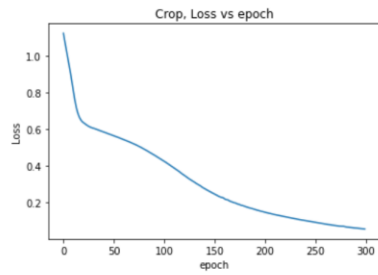
```
Computational time :- 0.00028443336486816406 seconds
```

```
In [16]: iteration = 300
losses = train(iteration, remote_model, remote_torch, optim, data_ptr, target_ptr)
```

```
In [17]: import matplotlib.pyplot as plt
```

```
In [18]: plt.title('Crop, Loss vs epoch')
plt.plot(range(iteration), losses)
plt.ylabel("Loss")
plt.xlabel("epoch")
```

```
Out[18]: Text(0.5, 0, 'epoch')
```



Download model

```
In [19]: def get_local_model(model):
    if not model.is_local:
        local_model = model.get(
            request_block=True,
            reason="To run test and inference locally",
            timeout_secs=5,
        )
    else:
        local_model = model

    return local_model

local_model = get_local_model(remote_model)
```

```
In [20]: local_model
```

```
Out[20]: <__main__.SynNet at 0x7f3eb1479c90>
```

```
In [ ]:
```

Test on local data

```
In [21]: import torch
import pandas as pd
import numpy as np
from sklearn.metrics import accuracy_score
```

```
In [22]: crop_test = pd.read_csv(f"Crop_test_data_Artharva1.csv")
```

```
In [23]: from sklearn.preprocessing import StandardScaler
df2 = StandardScaler().fit_transform(crop_test.iloc[:, 0:5])
```

```
In [24]: df = pd.DataFrame(df2)
```

```
In [25]: XX_test = df.loc[:, crop_test.columns != "label"]
yy_test = crop_test["label"]
```

```
In [26]: XX_test
```

```

In [27]: yy_test
Out[27]: 0 0
1 0
2 1
3 2
4 0
5 0
6 1
7 1
8 2
9 2
10 0
11 0
12 0
13 1
14 1
15 1
16 0
17 0
18 0
19 2
20 2
21 2
22 1
23 1
24 1
25 1
26 2
27 2
28 2
29 2
Name: label, dtype: int64

In [28]: XX1_test = torch.FloatTensor(np.array(XX_test))
yy1_test = torch.LongTensor(np.array(yy_test))

In [29]: preds = []
with torch.no_grad():
    #Local_model.eval()
    for i in range(len(XX1_test)):
        sample = XX1_test[i]

        y_hat = local_model(sample.unsqueeze(0))
        pred = y_hat.argmax().item()
        print(f"Prediction: {pred} Ground Truth: {yy1_test[i]}")
        preds.append(pred)

In [30]: #preds = []
#with torch.no_grad():
#    #Local_model.eval()
#    # for i in range(len(XX1_test)):
#    #     sample = XX1_test[i]

#    # y_hat = Local_model(sample.unsqueeze(0))
#    # pred = y_hat.argmax().item()
#    # print(f"Prediction: {pred} Ground Truth: {yy1_test[i]}")
#    #preds.append(pred)
#preds = [sample.squeeze().tolist() for sample in preds]

In [31]: sample
Out[31]: tensor([-1.2161, -1.3208, 2.9069, -0.7050])

In [32]: y_hat
Out[32]: tensor([-1.6114e+01, -9.6993e+01, -1.1921e-07])

In [33]: acc = accuracy_score(yy1_test, preds)
print("Overall test accuracy", acc * 100)

Overall test accuracy 73.33333333333333

In [34]: plt.title('Crop Type Predictions versus Ground Truth')

plt.plot(preds, label='pred')
plt.plot(yy1_test, label='actual')
plt.legend()
plt.xlabel('crop type')
plt.ylabel('predictions')
plt.show()

```

```
In [35]: #plt.title('Crop Type Predictions versus Ground Truth')

#plt.scatter(preds, range(len(preds)), label='pred')
#plt.scatter(yy1_test, range(len(yy1_test)), label='actual')
#plt.legend()
#plt.xlabel('crop type')
#plt.ylabel('predictions')
#plt.show()
```

```
In [36]: #print("Base Model parameters:")
print(remote_model)
print()

#print("Remote model1 parameters:")
#print(remote_torch)
print()

#print("Remote model2 parameters:")
#print(param2)

<__main__.SyNet object at 0x7f3ec61b5c90>
```

```
In [37]: #y_pred_list = []
#model.eval()
#with torch.no_grad():
#    for X_batch in test_loader:
#        X_batch = X_batch.to(device)
#        y_test_pred = model(X_batch)
#        y_test_pred = torch.sigmoid(y_test_pred)
#        y_pred_tag = torch.round(y_test_pred)
#        y_pred_list.append(y_pred_tag.cpu().numpy())

#y_pred_list = [a.squeeze().tolist() for a in y_pred_list]
```

```
In [38]: from sklearn.metrics import classification_report

print(classification_report(yy1_test, preds))
```


	precision	recall	f1-score	support
0	0.69	0.90	0.78	10
1	0.80	0.40	0.53	10
2	0.75	0.90	0.82	10
accuracy			0.73	30
macro avg	0.75	0.73	0.71	30
weighted avg	0.75	0.73	0.71	30

Bandwidth delay product for smart farm

Jupyter Duet_FL_Data_Scientist-BDP_26_1 Last Checkpoint: 06/02/2023 (autosaved) Python 3 (ipykernel)

```
In [ ]: ## Join the Duet Server the Data Owner 1 connected to
```

```
In [1]: import syft as sy
duet1 = sy.join_duet(loopback=True)
import matplotlib.pyplot as plt
import torch
```



Duet

Joining Duet


DISCLAIMER: Duet is an experimental feature currently in beta.
Use at your own risk.

Love Duet? Please consider supporting our community!
<https://github.com/sponsors/OpenMined>

Punching through firewall to OpenGrid Network Node at:
<http://ec2-18-218-7-180.us-east-2.compute.amazonaws.com:5000>
...waiting for response from OpenGrid Network...

Join the Duet Server the Data Owner 2 connected to

```
In [2]: duet2 = sy.join_duet(loopback=True)
```



Duet

Joining Duet

DISCLAIMER: Duet is an experimental feature currently in beta.
Use at your own risk.

Love Duet? Please consider supporting our community!
<https://github.com/sponsors/OpenMined>

Punching through firewall to OpenGrid Network Node at:
<http://ec2-18-218-7-180.us-east-2.compute.amazonaws.com:5000>
...waiting for response from OpenGrid Network...


```
In [3]: duet1.store.pandas
```

```
Out[3]:
```

	ID	Tags	Description	object_type
0	<UID: 1da8e71b99484fc3aac41c807f13623e>	[data]	dataset of speed and round trip time used for ...	<class 'torch.Tensor'>

```
In [4]: duet2.store.pandas
```

```
Out[4]:
```

	ID	Tags	Description	object_type
0	<UID: cdf19bb1865641e28fe35f601e594cfd>	[data]	dataset of speed and round trip time used for ...	<class 'torch.Tensor'>

Linear regression

```
In [5]: data1_ptr = duet1.store[0]
data2_ptr = duet2.store[0]

print(data1_ptr)
print(data2_ptr)

<syft.proxy.torch.TensorPointer object at 0x7ff92c7a2ad0>
<syft.proxy.torch.TensorPointer object at 0x7ff92c7a2990>
```

Create Base Model

```
In [ ]:
```

```
In [6]: in_dim = 1
out_dim = 1
```

```
In [7]: class SyNet(sy.Module):
def __init__(self, torch_ref):
    super(SyNet, self).__init__(torch_ref=torch_ref)
    self.linear = self.torch_ref.nn.Linear(in_dim, out_dim)

def forward(self, x):
    x = self.linear(x)
    return x
```

Training

```
In [8]: def train(iterations, model, torch_ref, optim, data1_ptr, target1_ptr):

    losses = []

    for i in range(iterations):
        optim.zero_grad()

        output = model(data1_ptr)

        loss = torch_ref.nn.functional.mse_loss(output, target1_ptr)

        loss_item = loss.item()

        loss_value = loss_item.get(
            reason="To evaluate training progress",
            request_block=True,
            timeout_secs=5,
        )

        print("Epoch", i, "loss", loss_value)

        losses.append(loss_value)

        loss.backward()

        optim.step()

    return losses
```

Send one copy of the model to each data owner or client and train them remotely one by one

```
In [9]: import torch as th
import numpy as np
```

```
In [10]: base_model = SyNet(torch)

Training of Data Owner 1 data
```

```
In [11]: remote_model1 = base_model.send(duet1)
```

```
In [12]: remote_torch1 = duet1.torch
params = remote_model1.parameters()
optim1 = remote_torch1.optim.Adam(params=params, lr=0.01)
```

```
In [13]: target1_ptr = th.FloatTensor(np.array([
3.18,3.48,3.01,3.45,3.36,4.07,3.81,3.64,3.74,3.96,4.01,3.76,3.88,3.62,3.94,3.84,4.02,
4.07,3.64,3.83,3.65,3.89,3.58,3.75,3.64,3.55,4.06,2.86,0.69,3.92,5.25,3.33,3.23,3.32,
3.36,3.37,3.42,3.28,3.20,3.00,3.34,3.09,3.37,2.98,3.25,3.09,2.53,3.36,2.966,2.77,
3.22,3.56,2.89,3.23,3.28,3.17,3.11,3.07,3.2,2.92,3.23,3.19,3.33,3.03,2.65,3.14,
3.18,3.15,3.4,3.18,3.59,3.25,3.27,3.40,3.77,3.09,3.21,3.22,3.11,3.25,3.22,3.35,
3.38,3.23,3.29,3.44,3.61,3.31,3.44,3.54,3.58,3.17,3.45,3.38,3.66,3.34,1.75,3.39,3.81,3.58,
3.80,3.44,3.54,3.13,3.42,3.4,3.46,3.45,3.63,3.3,3.44,3.60,3.61,3.45,3.64,3.55,3.61,
4.57,4.27,4.22,4.58,4.41,4.29,4.44,4.12,4.85,4.52,4.52,4.7,4.06,4.54,4.55,4.53,4.47,
4.43,4.44,4.50,4.4,3.55,4.49,4.42,4.75,4.17,4.28])).reshape(-1, 1))
target1_ptr.shape
```

```
Out[13]: torch.Size([144, 1])
```

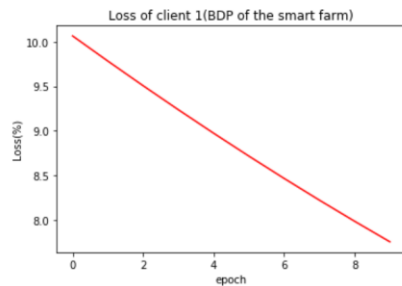
```
In [14]: target1_ptr
```

```
Out[14]: tensor([[3.1800],
[3.4800],
[3.0100],
[3.4500],
[3.3600],
[4.0700],
[3.8100],
[3.6400],
[3.7400],
[3.9600],
[4.0100],
[3.7600],
[3.8800],
[3.6200],
[3.9400],
[3.8400],
[4.0200],
[4.0700],
[3.6400],
[3.8300],
[3.6500],
[3.8900],
[3.5800],
[3.7500],
[3.6400],
[3.5500],
[4.0600],
[2.8600],
[0.6900],
[3.9200],
[5.2500],
[3.3300],
[3.2300],
[3.3200],
[3.3600],
[3.3700],
[3.4200],
[3.2800],
[3.2000],
[3.3400],
[3.0900],
[3.3700],
[2.9800],
[3.2500],
[3.0900],
[2.5300],
[3.3600],
[2.9660],
[2.7700],
[3.2200],
[3.5600],
[2.8900],
[3.2300],
[3.2800],
[3.1700],
[3.1100],
[3.0700],
[3.2000],
[2.9200],
[3.2300],
[3.1900],
[3.3300],
[3.0300],
[2.6500],
[3.1400],
[3.1800],
[3.1500],
[3.4000],
[3.1800],
[3.5900],
[3.2500],
[3.2700],
[3.4000],
[3.7700],
[3.0900],
[3.2100],
[3.2200],
[3.1100],
[3.2500],
[3.2200],
[3.3500],
[3.3800],
[3.2300],
[3.2900],
[3.4400],
[3.6100],
[3.3100],
[3.4400],
[3.5400],
[3.5800],
[3.1700],
[3.4500],
[3.3800],
[3.6600],
[3.3400],
[1.7500],
[3.3900],
[3.8100],
[3.5800],
[3.8000],
[3.4400],
[3.5400],
[3.1300],
[3.4200],
[3.4000],
[3.4600],
[3.4500],
[3.6300],
[3.3000],
[3.4400],
[3.6000],
[3.6100],
[3.4500],
[3.6400],
[3.5500],
[3.6100],
[4.5700],
[4.2700],
[4.2200],
[4.5800],
[4.4100],
[4.2900],
[4.4400],
[4.1200],
[4.8500],
[4.5200],
[4.5200],
[4.7000],
[4.0600],
[4.5400],
[4.5500],
[4.5300],
[4.4700],
[4.4300],
[4.4400],
[4.5000],
[4.4000],
[3.5500],
[4.4900],
[4.4200],
[4.7500],
[4.1700],
[4.2800]])
```

```
In [15]: iteration = 10
losses = train(iteration, remote_model1, remote_torch1, optim1, data1_ptr, target1_ptr)
```

```
Epoch 0 loss 10.06574535369873
Epoch 1 loss 9.783493041992188
Epoch 2 loss 9.507425308227539
Epoch 3 loss 9.23764419555664
Epoch 4 loss 8.974245071411133
Epoch 5 loss 8.717314720153809
Epoch 6 loss 8.46693229675293
Epoch 7 loss 8.223170280456543
Epoch 8 loss 7.986085891723633
Epoch 9 loss 7.755733013153076
```

```
In [16]: plt.title('Loss of client 1(BDP of the smart farm)')
plt.plot(losses, color='red')
plt.ylabel("Loss(%)")
plt.xlabel("epoch")
plt.show()
```



Training of Data Owner 2 data

```
In [17]: remote_model2 = base_model.send(duet2)
```

```
In [18]: remote_torch2 = duet2.torch
params = remote_model2.parameters()
optim2 = remote_torch2.optim.Adam(params=params, lr=0.1)
```

```
In [19]: target2_ptr = th.FloatTensor(np.array([3.22,3.38,3.28,3.20,3.26,3.32,3.25,3.41,3.12,3.56,3.23,3.33,3.51,3.32,3.34,3.32,3.29,2.51,
3.14,3.24,3.33,3.26,3.28,3.74,4.00,4.85,4.41,4.53,5.03,4.27,4.42,4.98,4.36,4.79,4.54,3.28,4.89,4.00,4.67,4.52,4.61,
3.96,5.01,4.58,4.42,4.43,7.48,6.19,6.53,6.55,7.93,5.54,4.18,9.44,4.92,3.78,2.72,4.40,4.42,4.67,4.35,4.73,3.86,4.35,
4.62,4.44,4.19,4.59,4.07,4.83,4.31,4.29,4.60,4.37,4.21,4.64,4.28,4.20,4.44,4.22,4.81,4.11,4.12,4.67,4.19,4.69,3.96,
4.31,4.75,4.30,4.52,4.60,4.45,4.76,4.32,3.98,4.35,4.24,4.70,4.41,4.44,4.78,4.35,4.43,4.73,4.44,4.44,3.87,4.15,4.85,
4.39,3.88,4.82,3.78,4.82,4.34,4.29,4.39,4.33,4.50,4.72,4.24,4.57,4.32,4.38,4.83,4.43,4.52,4.03,4.01,4.25,4.01,3.95,
4.26,3.91,4.22,4.01,3.91,4.56])).reshape(-1, 1))
target2_ptr
```

```

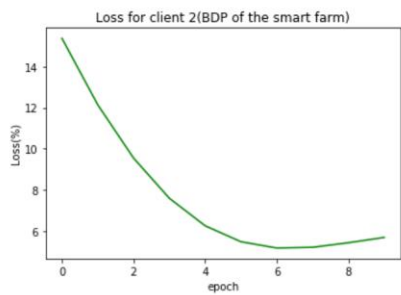
Out[19]: tensor([[3.2200],
                [3.3800],
                [3.2800],
                [3.2000],
                [3.2600],
                [3.3200],
                [3.2500],
                [3.4100],
                [3.1200],
                [3.5600],
                [3.2300],
                [3.3300],
                [3.5100],
                [3.3200],
                [3.3400],
                [3.3200],
                [3.2900],
                [2.5100],
                [3.1100],
                [3.4400]])

In [20]: iteration = 10
losses = train(iteration, remote_model2, remote_torch2, optim2, data2_ptr, target2_ptr)

Epoch 0 loss 15.36158275604248
Epoch 1 loss 12.134000778198242
Epoch 2 loss 9.545798301696777
Epoch 3 loss 7.59517765045166
Epoch 4 loss 6.259440898895264
Epoch 5 loss 5.485530853271484
Epoch 6 loss 5.1818084716796875
Epoch 7 loss 5.2177300453186035
Epoch 8 loss 5.4387407302856445
Epoch 9 loss 5.695688247680664

In [21]: plt.title('Loss for client 2(BDP of the smart farm)')
plt.plot(losses, color='green')
plt.ylabel("Loss(%)")
plt.xlabel("epoch")
plt.show()

```



Averaging Model Updates

The Aggregate server combines all the models and get the model updates from different clients and make an aggregation. For this test bed, we use the Data Scientist server as the coordinator.

To check if the remote models are different from the base model

```

In [22]: param1 = remote_model1.parameters().get(request_block=True)
param2 = remote_model2.parameters().get(request_block=True)

print("Base Model parameters:")
print(base_model.parameters())
print()

print("Remote model1 parameters:")
print(param1)
print()

print("Remote model2 parameters:")
print(param2)

Base Model parameters:
[Parameter containing:
 tensor([[0.0730]], requires_grad=True), Parameter containing:
 tensor([0.1908], requires_grad=True)]

Remote model1 parameters:
[Parameter containing:
 tensor([[0.1721]], requires_grad=True), Parameter containing:
 tensor([0.2903], requires_grad=True)]

```

```
Remote model2 parameters:
[Parameter containing:
tensor([[0.8013]], requires_grad=True), Parameter containing:
tensor([1.0894], requires_grad=True)]
```

As you can see, the remote model parameter values are different from the base model parameter values. That means the remote copies of our base model got trained and updated.

```
In [23]: remote_model1_updates = remote_model1.get(
         request_block=True
         ).state_dict()
print(remote_model1_updates)
OrderedDict([('linear.weight', tensor([[0.1721]])), ('linear.bias', tensor([0.2903]))])
```

```
In [24]: remote_model2_updates = remote_model2.get(
         request_block=True
         ).state_dict()
print(remote_model2_updates)
OrderedDict([('linear.weight', tensor([[0.8013]])), ('linear.bias', tensor([1.0894]))])
```

```
In [25]: from collections import OrderedDict
```

The aggregation of the weights is computed for 2 clients nodes

```
In [26]: avg_updates = OrderedDict()
avg_updates["linear.weight"] = (
    remote_model1_updates["linear.weight"] + remote_model2_updates["linear.weight"]) / 2
avg_updates["linear.bias"] = (
    remote_model1_updates["linear.bias"] + remote_model2_updates["linear.bias"]) / 2
print(avg_updates)
OrderedDict([('linear.weight', tensor([[0.4867]])), ('linear.bias', tensor([0.6898]))])
```

Load aggregated weights

```
In [27]: combined_model = SyNet(torch)
```

```
In [28]: combined_model.load_state_dict(avg_updates)
```

```
In [29]: del avg_updates, remote_model1_updates, remote_model2_updates
```

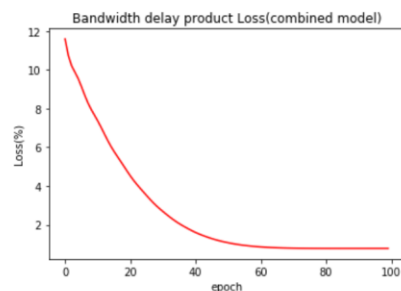
```
In [103]: test_data = th.FloatTensor(np.array([[5.73,7.18,7.42,7.34,7.88,7.78,7.84,7.94,7.74,7.474,7.84,7.08,7.88,
        7.77,7.17,7.38,8.47,5.98,5.93]]).reshape(-1,1))
```

```
In [129]: test_target = th.FloatTensor(np.array([[3.41,4.17,4.31,4.2,4.47,4.48,4.42,4.59,4.46,4.32,4.25,4.26,4.47,4.22,4.18,
        4.28,4.82,3.64,3.52]]).reshape(-1,1))
```

```
In [130]: preds = []
with torch.no_grad():
    for i in range(len(test_data)):
        sample = test_data[i]
        y_hat = combined_model(sample)
        print(f"Prediction: {y_hat.item()} Ground Truth: {test_target[i].item()}")
        preds.append(y_hat)
```

```
Prediction: 3.4786853790283203 Ground Truth: 3.4100000858306885
Prediction: 4.184420585632324 Ground Truth: 4.170000076293945
Prediction: 4.30123233795166 Ground Truth: 4.309999942779541
Prediction: 4.262295246124268 Ground Truth: 4.199999809265137
Prediction: 4.525120735168457 Ground Truth: 4.46999979019165
Prediction: 4.476449489593506 Ground Truth: 4.48000019073486
Prediction: 4.505651950836182 Ground Truth: 4.420000076293945
Prediction: 4.554323673248291 Ground Truth: 4.590000152587891
Prediction: 4.4569807052612305 Ground Truth: 4.460000038146973
Prediction: 4.3275146484375 Ground Truth: 4.320000171661377
Prediction: 4.505651950836182 Ground Truth: 4.25
Prediction: 4.135749340057373 Ground Truth: 4.26000228881836
Prediction: 4.525120735168457 Ground Truth: 4.46999979019165
Prediction: 4.471581935882568 Ground Truth: 4.21999979019165
Prediction: 4.179553508758545 Ground Truth: 4.179999828338623
Prediction: 4.281763553619385 Ground Truth: 4.2800020980835
Prediction: 4.812282085418701 Ground Truth: 4.82000171661377
Prediction: 3.6003637313842773 Ground Truth: 3.640000104904175
Prediction: 3.576028347015381 Ground Truth: 3.5199999809265137
```

```
In [131]: plt.title('Bandwidth delay product Loss(combined model)')
plt.plot(losses, color='red')
plt.ylabel("Loss(%)")
plt.xlabel("epoch")
plt.show()
```

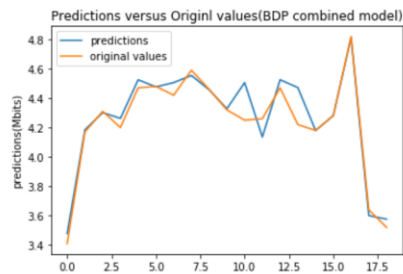


```
In [132]: preds
```

```
Out[132]: [tensor([3.4787]),
           tensor([4.1844]),
           tensor([4.3012]),
           tensor([4.2623]),
           tensor([4.5251]),
           tensor([4.4764]),
           tensor([4.5057]),
           tensor([4.5543]),
           tensor([4.4570]),
           tensor([4.3275]),
           tensor([4.5057]),
           tensor([4.1357]),
           tensor([4.5251]),
           tensor([4.4716]),
           tensor([4.1796]),
           tensor([4.2818]),
           tensor([4.8123]),
           tensor([3.6004]),
           tensor([3.5760])]
```

```
In [133]: plt.title('Predictions versus Originl values(BDP combined model)')
plt.plot(preds, label='predictions')
plt.plot(test_target, label='original values')
plt.legend()
plt.xlabel('Original values(Mbits)')
plt.ylabel('predictions(Mbits)')
plt.show()
```

```
/home/user_iot/anaconda3/lib/python3.7/site-packages/numpy/core/shape_base.py:65: FutureWarning: The input object of type 'Tensor
or' is an array-like implementing one of the corresponding protocols ('__array__', '__array_interface__' or '__array_struct__
'); but not a sequence (or 0-D). In the future, this object will be coerced as if it was first converted using 'np.array(obj)'.
To retain the old behaviour, you have to either modify the type 'Tensor', or assign to an empty array created with 'np.empty(co
rrect_shape, dtype=object)'.
  ary = asanyarray(ary)
/home/user_iot/anaconda3/lib/python3.7/site-packages/numpy/core/shape_base.py:65: VisibleDeprecationWarning: Creating an ndarra
y from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is de
precated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
  ary = asanyarray(ary)
```



Comparison of the federated combine models to classical linear regression on centralised data

```
In [36]: import torch
import numpy as np

in_dim = 1
out_dim = 1
class ClassicalLR(torch.nn.Module):
    def __init__(self, torch):
        super(ClassicalLR, self).__init__()
        self.linear = torch.nn.Linear(in_dim, out_dim)
    def forward(self, x):
        x = self.linear(x)
        return x
classical_model = ClassicalLR(torch)
```

```
In [37]: data = torch.FloatTensor(np.array([[0.48,0.48,0.49,0.49,0.49,0.49,0.49,0.49,0.49,0.49,0.49,0.48,0.48,0.49,0.5,0.49,0.49,0.5,0.53,0.52,
0.51,0.52,0.5,0.5,0.5,0.52,0.51,0.52,0.56,0.51,0.52,0.53,0.5,0.5,0.5,0.51,0.53,0.51,0.53,0.52,0.53,0.52,0.53,0.53,
0.53,0.53,0.51,0.52,0.51,0.51,0.51,0.51,0.51,0.52,0.51,0.51,0.53,0.5,0.52,0.51,0.53,0.53,0.51,0.51,0.53,0.52,0.53,
0.53,0.53,0.51,0.48,0.49],[6.79,6.63,7.47,6.62,6.67,6.94,7.7,6.31,6.54,6.56,6.36,6.77,6.71,6.83,6.76,6.60,6.72,6.89,6.81,
6.35,6.76,6.5,7.32,6.68,3.49,6.53,7.46,6.89,6.79,6.74,6.81,5.91,6.83,6.8,6.91,6.76,6.85,6.47,6.49,6.93,6.81,6.64,
6.87,6.69,6.80,6.74,6.41,6.18,7.03,6.68,6.46,6.75,6.11,7.40,6.86,6.91,6.97,6.11,6.82,6.97,6.67,6.55,6.72,6.74,6.61,
6.53,6.70,6.58,6.44,7.35,6.61,6.67])).reshape(-1, 1))

target = torch.FloatTensor(np.array([
3.22,3.38,3.28,3.20,3.26,3.32,3.25,3.41,3.12,3.56,3.23,3.33,3.51,3.32,3.34,3.32,3.29,2.51,3.11,3.41,2.56,3.03,3.09,
3.14,3.24,3.33,3.26,3.28,3.74,4.00,4.85,4.41,4.53,5.03,4.27,4.42,4.98,4.36,4.79,4.54,3.28,4.89,4.00,4.67,4.52,4.61,
3.96,5.01,4.58,4.42,4.43,7.48,6.19,6.53,6.55,7.93,5.54,4.18,9.44,4.92,3.78,2.72,4.40,4.42,4.67,4.35,4.73,3.86,4.35,
4.62,4.44,4.19,4.59,4.07,4.83,4.31,4.29,4.60,4.37,4.21,4.64,4.28,4.20,4.44,4.22,4.81,4.11,4.12,4.67,4.19,4.69,3.96,
4.31,4.75,4.30,4.52,4.60,4.45,4.76,4.32,3.98,4.35,4.24,4.70,4.41,4.44,4.78,4.35,4.43,4.73,4.44,4.44,3.87,4.15,4.85,
4.39,3.88,4.82,3.78,4.82,4.34,4.29,4.39,4.33,4.50,4.72,4.24,4.57,4.32,4.38,4.83,4.43,4.52,4.03,4.01,4.25,4.01,3.95,
4.26,3.91,4.22,4.01,3.91,4.56
])).reshape(-1, 1))
```

```
In [38]: def classic_train(iterations, model, torch, optim, data, target, criterion):
```

```
    losses = []

    for i in range(iterations):

        optim.zero_grad()

        output = model(data)

        loss = criterion(output, target)

        loss_item = loss.item()

        if i % 10 == 0:
            print("Epoch", i, "loss", loss_item)

        losses.append(loss_item)

        loss.backward()

        optim.step()

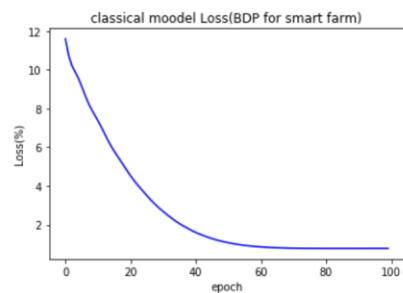
    return losses
```

```
In [39]: params = classical_model.parameters()
optim = torch.optim.Adam(params=params, lr=0.1)
criterion = torch.nn.MSELoss()
```

```
In [40]: iteration = 100
losses = classic_train(iteration, classical_model, torch, optim, data, target, criterion)
```

```
Epoch 0 loss 11.59276294708252
Epoch 10 loss 7.363425254821777
Epoch 20 loss 4.520441055297852
Epoch 30 loss 2.6808900833129883
Epoch 40 loss 1.5984816551208496
Epoch 50 loss 1.0677342414855957
Epoch 60 loss 0.8526885509490967
Epoch 70 loss 0.7874858379364014
Epoch 80 loss 0.775553286075592
Epoch 90 loss 0.7755898237228394
```

```
In [41]: plt.title('classical model Loss(BDP for smart farm)')
plt.plot(losses, color='blue')
plt.ylabel("Loss(%)")
plt.xlabel("epoch")
plt.show()
```



```
In [42]: XX_test = th.FloatTensor(np.array([
0.17,1.18,3.72,5.34,8.38,10.78,8.84,4.44,0.64,16.474,16.84,7.78,16.88,9.77,0.12,
4.38,4.77,1.78,4.35
])).reshape(-1, 1))
yy_test = th.FloatTensor(np.array([4.07,4.12,4.25,4.32,4.42,4.57,4.42,4.22,3.84,4.75,4.85,4.46,4.87,4.45,3.85,
4.27,3.35,4.16,3.82])).reshape(-1, 1))
```

```
In [43]: XX1_test = torch.FloatTensor(np.array(XX_test))
yy1_test = torch.FloatTensor(np.array(yy_test))
```

```
In [44]: preds = []
with torch.no_grad():
    for i in range(len(XX1_test)):
        sample = XX1_test[i]
        y_hat = classical_model(sample)

        print(f"Prediction: {y_hat.item()} Ground Truth: {yy1_test[i].item()}")
        preds.append(y_hat)
```

```
Prediction: 4.2258405685424805 Ground Truth: 4.070000171661377
Prediction: 4.247415065765381 Ground Truth: 4.119999885559082
Prediction: 4.301671504974365 Ground Truth: 4.25
Prediction: 4.336275577545166 Ground Truth: 4.320000171661377
Prediction: 4.401212215423584 Ground Truth: 4.420000076293945
Prediction: 4.452477931976318 Ground Truth: 4.570000171661377
Prediction: 4.411038398742676 Ground Truth: 4.420000076293945
Prediction: 4.317050933837891 Ground Truth: 4.21999979019165
Prediction: 4.235880374908447 Ground Truth: 3.8399999141693115
Prediction: 4.574106216430664 Ground Truth: 4.75
Prediction: 4.581923961639404 Ground Truth: 4.849999904632568
Prediction: 4.3883957862854 Ground Truth: 4.460000038146973
Prediction: 4.582778453826904 Ground Truth: 4.869999885559082
Prediction: 4.430903911590576 Ground Truth: 4.449999809265137
Prediction: 4.224772930145264 Ground Truth: 3.8499999046325684
Prediction: 4.315769195556641 Ground Truth: 4.269999980926514
```

```

Prediction: 4.324100017547607 Ground Truth: 3.3499999046325684
Prediction: 4.2602314949035645 Ground Truth: 4.159999847412109
Prediction: 4.315128803253174 Ground Truth: 3.819999933242798

```

```

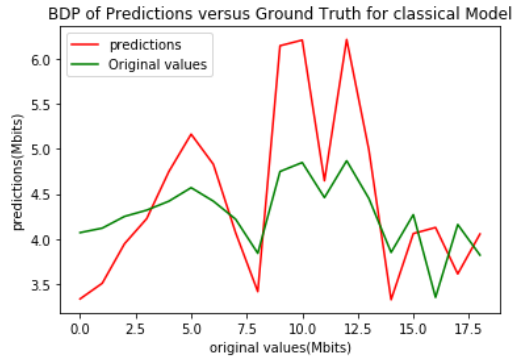
In [45]: plt.title('BDP of Predictions versus Ground Truth for classical Model')
plt.plot(preds, label='predictions',color="red")
plt.plot(yy1_test, label='Original values',color="green")
plt.legend()
plt.xlabel('original values(Mbits)')
plt.ylabel('predictions(Mbits)')
plt.show()

```

```

/home/user_iot/anaconda3/lib/python3.7/site-packages/numpy/core/shape_base.py:65: FutureWarning: The input object of type 'Tensor' is an array-like implementing one of the corresponding protocols ('__array__', '__array_interface__' or '__array_struct__'); but not a sequence (or 0-D). In the future, this object will be coerced as if it was first converted using 'np.array(obj)'. To retain the old behaviour, you have to either modify the type 'Tensor', or assign to an empty array created with 'np.empty(correct_shape, dtype=object)'.
  ary = asanyarray(ary)
/home/user_iot/anaconda3/lib/python3.7/site-packages/numpy/core/shape_base.py:65: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray.
  ary = asanyarray(ary)

```



Bandwidth Delay Product edge node 1 code

jupyter
Duet_FL_Data_Owner_1-BDP_26_1-Copy1
Last Checkpoint: 02/05/2023 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)

+
%
⌂
📄
⬆️
⬇️
▶️ Run
🔄
▶️▶️
Markdown
🗨️

Part 1: Launch a Duet Server and upload data

```

In [1]: import syft as sy
duet1 = sy.launch_duet(loopback=True)

```

Duet

```

>>> Starting Duet
> DISCLAIMER: Duet is an experimental feature currently in beta.
> Use at your own risk.

> ❤️ Love Duet? Please consider supporting our community!
> https://github.com/sponsors/OpenMined

> Punching through firewall to OpenGrid Network Node at:
> http://ec2-18-218-7-180.us-east-2.compute.amazonaws.com:5000
> ...waiting for response from OpenGrid Network...
> DONE!

> STEP 1: Send the following code to your Duet Partner!

import syft as sy
duet = sy.join_duet(loopback=True)

> Connecting...
> CONNECTED!

> DUET LIVE STATUS - Objects: 4 Requests: 0 Messages: 227 Request Handlers: 1

```


```

In [2]: import torch
import numpy as np

```


Part 1: Launch a Duet Server and upload data

```
In [1]: import syft as sy
duet2 = sy.duet(loopback=True)
```



Starting Duet

DISCLAIMER: Duet is an experimental feature currently in beta. Use at your own risk.

Love Duet? Please consider supporting our community!
<https://github.com/sponsors/OpenMined>

Punching through firewall to OpenGrid Network Node at:
<http://ec2-18-218-7-180.us-east-2.compute.amazonaws.com:5000>

...waiting for response from OpenGrid Network...

```
In [2]: import torch
import numpy as np
```

```
In [3]: data2 = torch.FloatTensor(np.array([[
0.48,0.49,0.48,0.49,0.49,0.48,0.49,0.48,0.49,0.48,0.49,0.48,0.48,0.49,0.48,0.48,0.49,0.48,0.49,0.48,0.48,0.49,0.48,0.49,0.48,0.48,0.49,0.48,0.49,0.48,0.48,0.49,0.48,0.49,0.48,0.48,0.49,0.48,0.49,0.48,0.48,0.49,0.48,0.48,0.48,0.56,0.63,0.72,0.65,0.65,0.72,0.65,0.66,0.73,0.65,0.73,0.65,0.65,0.73,0.65,0.67,0.73,0.66,0.66,0.73,0.66,0.72,0.65,1.22,2.39,2.25,0.97,1.16,2.25,2.08,1.36,0.71,0.63,1.85,0.64,0.64,0.70,0.63,0.71,0.63,0.64,0.69,0.64,0.75,0.63],
[
6.58,7.04,6.70,6.66,6.65,6.78,6.77,6.97,6.51,7.27,6.60,6.94,7.16,6.92,6.96,6.77,6.85,5.23,6.48,6.96,5.32,6.19,6.30,6.55,6.62,6.94,6.79,6.83,6.68,6.35,6.74,6.79,6.98,6.99,6.58,6.69,6.82,6.71,6.57,6.99,5.05,6.70,6.16,6.96,6.19,6.99,6.01,6.86,6.93,6.13,6.81,6.13,6.78,6.01,6.75,6.84,6.91,6.82,6.94,6.93,6.00,6.87,6.88,6.91,6.66,6.90,6.76,6.87,6.20,6.44,6.77,6.82]]).reshape(-1,1))
data2.send(duet2, tags=["data"], description="Bandwidth delay product, speed and round trip time", pointable=True)
data2.shape
```

```
Out[3]: torch.Size([144, 1])
```

```
In [4]: duet2.store.pandas
```

```
Out[4]:
```

	ID	Tags	Description	object_type
0	<UID: 829618a081ef4aa485f271be8f413d08>	[data]	Bandwidth delay product, speed and round trip ...	<class 'torch.Tensor'>

```
In [5]: data2
```

```
[0.7300],
[0.6500],
[0.7300],
[0.6500],
[0.6500],
[0.7300],
[0.6500],
[0.6700],
[0.7300],
[0.6600],
[0.6600],
[0.7300],
[0.6600],
[0.7200],
[0.6500],
[1.2200],
[2.3900],
[2.2500],
[0.9700],
[1.1600]
```

```
In [6]: duet2.requests.add_handler(action="accept", print_local=True)
```

AMARANTHUS VIRIDIS CLASSIFICATION

```
import numpy as np
import cv2
import glob
import os
import matplotlib.pyplot as plt
import string
from mlxtend.plotting import plot_decision_regions
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.utils.multiclass import unique_labels
from sklearn import metrics
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score, KFold

#print(os.listdir("C:/Users/igone/Downloads/Amara/"))
dim = 100
def getYourAmaranthus(Amaranthus, data_type, print_n=False, k_fold=False):
    images = []
    labels = []
    val = ['Train', 'Test']
    if not k_fold:
        path = "C:/Users/igone/Downloads/Amara/" + data_type + "/"
        for i,f in enumerate(Amaranthus):
            p = path + f
            j=0
            for image_path in glob.glob(os.path.join(p, "*.jpg")):
                image = cv2.imread(image_path, cv2.IMREAD_COLOR)
                image = cv2.resize(image, (dim, dim))
                image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
                images.append(image)
                labels.append(i)
                j+=1
            if(print_n):
                print("There are " , j , " " , data_type.upper(), " images of " ,
Amaranthus[i].upper())
            images = np.array(images)
            labels = np.array(labels)
            return images, labels
    else:
        for v in val:
            path = "C:/Users/igone/Downloads/Amara/" + v + "/"
            for i,f in enumerate(Amaranthus):
                p = path + f
                j=0
```

```

        for image_path in glob.glob(os.path.join(p, "*.jpg")):
            image = cv2.imread(image_path, cv2.IMREAD_COLOR)
            image = cv2.resize(image, (dim, dim))
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
            images.append(image)
            labels.append(i)
            j+=1
    images = np.array(images)
    labels = np.array(labels)
    return images, labels

def getAllAmaranthus():
    Amaranthus = []
    for Amaranthus_path in glob.glob("C:/Users/igone/Downloads/Amara/train/*"):
        Amaranthus = Amaranthus_path.split("/")[-1]
        Amaranthus.append(Amaranthus)
    return Amaranthus

Amaranthus = ['Healthy' , 'infected'] #Binary classification

#Get Images and Labels
X_t, y_train = getYourAmaranthus(Amaranthus, 'Train', print_n=True, k_fold=False)
X_test, y_test = getYourAmaranthus(Amaranthus, 'Test', print_n=True, k_fold=False)

#Get data for k-fold
X,y = getYourAmaranthus(Amaranthus, "", print_n=True, k_fold=True)

#Scale Data Images
scaler = StandardScaler()
X_train = scaler.fit_transform([i.flatten() for i in X_t])
X_test = scaler.fit_transform([i.flatten() for i in X_test])
X = scaler.fit_transform([i.flatten() for i in X])
There are 729 TRAIN images of HEALTHY
There are 167 TRAIN images of INFECTED
There are 145 TEST images of HEALTHY
There are 20 TEST images of INFECTED

# Perform k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Fit the model on the training data
    #classifier.fit(X_train, y_train)

#SVM
model = SVC(gamma='auto', kernel='linear')

```

```

model.fit(X_train, y_train)
y_pred = model.predict(X_test)
precision = metrics.accuracy_score(y_pred, y_test) * 100
print("Accuracy with SVM: {0:.2f}%".format(precision))

#K-NN
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
precision = metrics.accuracy_score(y_pred, y_test) * 100
print("Accuracy with K-NN: {0:.2f}%".format(precision))

#DECISION TREE
model = DecisionTreeClassifier()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
precision = metrics.accuracy_score(y_pred, y_test) * 100
print("Accuracy with Decision Tree: {0:.2f}%".format(precision))
Accuracy with SVM: 95.28%
Accuracy with K-NN: 92.92%
Accuracy with Decision Tree: 94.81%

```

```

score_train=[]
score_test=[]

```

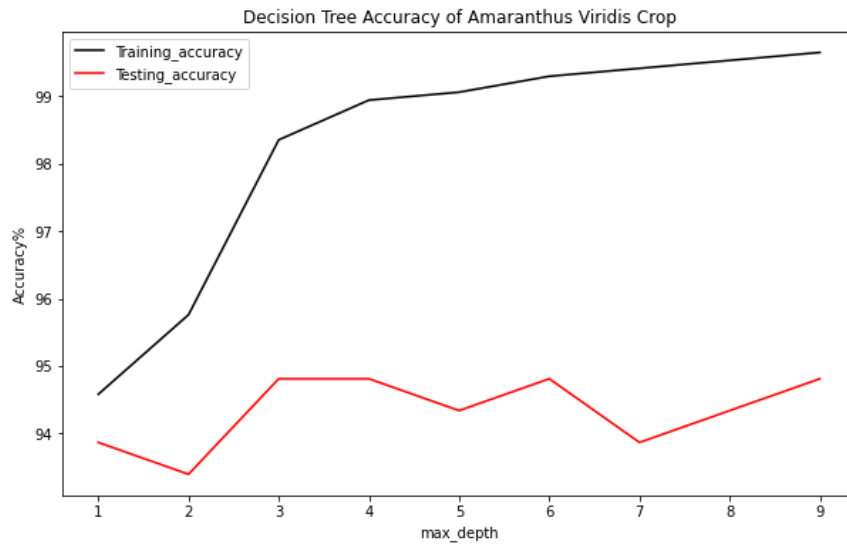
```

for i in range(1,10):
    dtree_md = DecisionTreeClassifier(max_depth=i)
    dtree_md.fit(X_train,y_train)

    score_train.append(dtree_md.score(X_train,y_train)*100)
    score_test.append(dtree_md.score(X_test,y_test)*100)

score_train_array=np.asarray(score_train)
score_test_array=np.asarray(score_test)
plt.figure(figsize=(10,6))
plt.plot(range(1,10),score_train_array,color='black', label="Training_accuracy")
plt.plot(range(1,10),score_test_array,color='red',label="Testing_accuracy")
plt.title("Decision Tree Accuracy of Amaranthus Viridis Crop")
plt.legend()
plt.xlabel('max_depth')
plt.ylabel('Accuracy%')
plt.show()

```



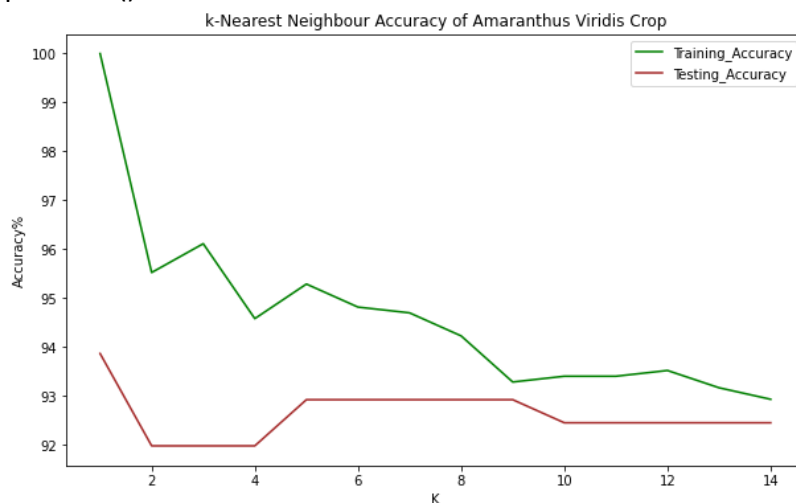
```
accuracy_train = []
accuracy_test = []
```

```
for i in range(1,15): #check all possible values for 1 to 15
    k_nn = KNeighborsClassifier(n_neighbors=i)
    k_nn.fit(X_train,y_train)
    pred_i = k_nn.predict(X_test)
    accuracy_train.append(k_nn.score(X_train,y_train)*100)
    accuracy_test.append(k_nn.score(X_test,y_test)*100)
```

```
accuracy_train_array=np.asarray(accuracy_train)
accuracy_test_array=np.asarray(accuracy_test)
```

```
plt.figure(figsize=(10,6))
plt.plot(range(1,15),accuracy_train_array, label='Training_Accuracy', color='green')
plt.plot(range(1,15),accuracy_test_array, label='Testing_Accuracy', color='brown')
plt.legend()
plt.title("k-Nearest Neighbour Accuracy of Amaranthus Viridis Crop")
plt.xlabel('K')
plt.ylabel('Accuracy%')
```

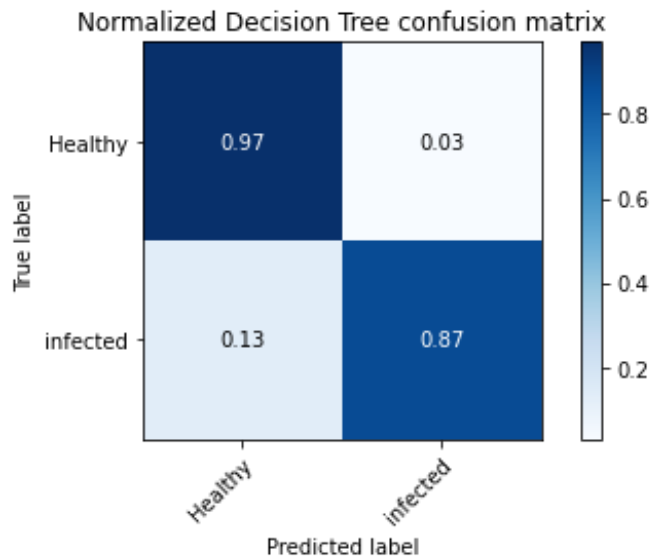
```
plt.show()
```



```
tree = DecisionTreeClassifier()
tree = tree.fit(X_train,y_train)
y_pred = tree.predict(X_test)
```

#Evaluation

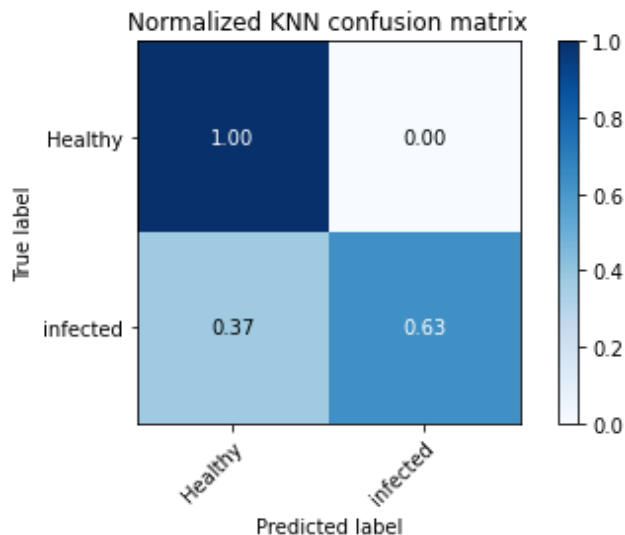
```
precision = metrics.accuracy_score(y_pred, y_test) * 100
print("Accuracy with Decision Tree: {0:.2f}%".format(precision))
cm , _ = plot_confusion_matrix(y_test, y_pred, classes=y_train, normalize=True,
title='Normalized Decision Tree confusion matrix')
plt.show()
Accuracy with Decision Tree: 94.81%
```



```
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

#Evaluation

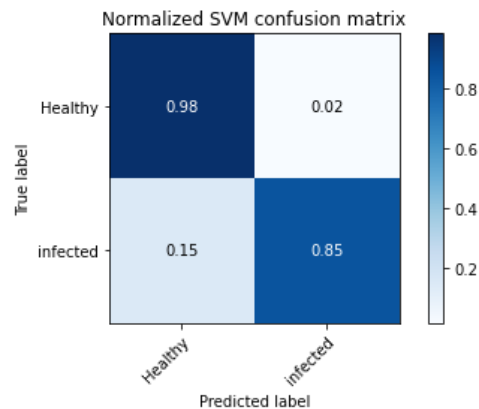
```
precision = metrics.accuracy_score(y_pred, y_test) * 100
print("Accuracy with K-NN: {0:.2f}%".format(precision))
cm , _ = plot_confusion_matrix(y_test, y_pred, classes=y_train, normalize=True,
title='Normalized KNN confusion matrix')
plt.show()
Accuracy with K-NN: 91.98%
```



```
svm = SVC(gamma='auto', kernel='linear', probability=True)
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
```

#Evaluation

```
precision = metrics.accuracy_score(y_pred, y_test) * 100
print("Accuracy with SVM: {0:.2f}%".format(precision))
cm, _ = plot_confusion_matrix(y_test, y_pred, classes=y_train, normalize=True,
title='Normalized SVM confusion matrix')
plt.show()
Accuracy with SVM: 95.28%
```



Amaranthus Viridis Crop growth Rate

```
import catboost as ctb
import lightgbm as lgb
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.datasets import make_regression
from sklearn.ensemble import AdaBoostRegressor
from lightgbm import LGBMRegressor
from catboost import CatBoostRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.metrics import accuracy_score, mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn import datasets
from sklearn import metrics
```

```
plt.style.use("ggplot")
```

```
#Load the Amaranthus dataset collected from the Suitelab Smart Farm
dataset = pd.read_csv("Amaranthus_growth_rate_A.csv")
```

```

X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 5].values

#splitting of the Dataset using Sklearn library in ratio
#of 20% for test and 80% for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

#KNeighbors Regressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# Generate some random data for demonstration
#np.random.seed(42)
#X = np.random.rand(100, 1) * 10
#y = 2 * X[:, 0] + np.random.randn(100)
#Load the Amaranthus dataset collected from the Suitelab Smart Farm
dataset = pd.read_csv("Amaranthus_growth_rate_A.csv")
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 5].values

#splitting of the Dataset using Sklearn library in ratio
#of 20% for test and 80% for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
# Create a k-nearest neighbors regressor object
knn = KNeighborsRegressor(n_neighbors=5)

# Perform k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
mse_scores = []
r2_scores = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Fit the model on the training data
    knn.fit(X_train, y_train)

    # Predict on the test data
    y_predd1 = knn.predict(X_test)

    # Calculate evaluation metrics
    mse = mean_squared_error(y_test, y_predd1)
    r2 = r2_score(y_test, y_predd1)

    # Append scores to the lists
    mse_scores.append(mse)
    r2_scores.append(r2)

```



```

# Calculate the average scores
avg_mse = np.mean(mse_scores)
avg_r2 = np.mean(r2_scores)

# Print the evaluation metrics
print("Average Mean Squared Error:", avg_mse)
print("Average R-squared:", avg_r2)
print("Predictions:", y_predd1)
Average Mean Squared Error: 0.030173333333333347
Average R-squared: 0.9854849186801763
Predictions: [0.68 0.92 1.34 1.76 1.9 2.18 2.68 4.08]

#Support Vector regressor evaluation
from sklearn.svm import SVR
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

#Load the Amaranthus dataset collected from the Suitelab Smart Farm
dataset = pd.read_csv("Amaranthus_growth_rate_A.csv")
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 5].values
#splitting of the Dataset using Sklearn library in ratio
#of 20% for test and 80% for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
# Create a support vector regression object
svr = SVR(kernel='rbf')

# Perform k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
mse_scores = []
r2_scores = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Fit the model on the training data
    svr.fit(X_train, y_train)

    # Predict on the test data
    y_predd2 = np.round(svr.predict(X_test), decimals=2)

    # Calculate evaluation metrics
    mse = mean_squared_error(y_test, y_predd2)
    r2 = r2_score(y_test, y_predd2)

    # Append scores to the lists
    mse_scores.append(mse)

```

```

r2_scores.append(r2)

# Calculate the average scores
avg_mse = np.mean(mse_scores)
avg_r2 = np.mean(r2_scores)

# Print the evaluation metrics
print("Average Mean Squared Error:", avg_mse)
print("Average R-squared:", avg_r2)
print("Predictions:", y_predd2)

Average Mean Squared Error: 0.0305330555555555542
Average R-squared: 0.9854818451456568
Predictions: [0.68 0.9 1.3 1.76 1.92 2.1 2.68 4.16]

#XGBoost regressor evaluation
import xgboost as xgb
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import pandas as pd

#Load the Amaranthus dataset collected from the Suitelab Smart Farm
dataset = pd.read_csv("Amaranthus_growth_rate_A.csv")
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 5].values
#splitting of the Dataset using Sklearn library in ratio
#of 20% for test and 80% for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
# Create XGBoost regression object
xgb_reg = xgb.XGBRegressor(objective='reg:squarederror')

# Perform k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
mse_scores = []
r2_scores = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Fit the model on the training data
    xgb_reg.fit(X_train, y_train)

    # Predict on the test data
    y_predd3 = np.round(xgb_reg.predict(X_test), decimals=2)

    # Calculate evaluation metrics
    mse = mean_squared_error(y_test, y_predd3)
    r2 = r2_score(y_test, y_predd3)

```

```

    # Append scores to the lists
    mse_scores.append(mse)
    r2_scores.append(r2)

# Calculate the average scores
avg_mse = np.mean(mse_scores)
avg_r2 = np.mean(r2_scores)

# Print the evaluation metrics
print("Average Mean Squared Error:", avg_mse)
print("Average R-squared:", avg_r2)
print("Predictions:", y_predd3)

Average Mean Squared Error: 0.019916668936614073
Average R-squared: 0.9864553845971022
Predictions: [0.8 1.1 1.5 1.9 2.1 2.2 2.7 4.4]

from sklearn.linear_model import QuantileRegressor
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

#Load the Amaranthus dataset collected from the Suitelab Smart Farm
dataset = pd.read_csv("Amaranthus_growth_rate_A.csv")
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 5].values
#splitting of the Dataset using Sklearn library in ratio
#of 20% for test and 80% for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
# Create a quantile regression object
quant_reg = QuantileRegressor()

# Perform k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
mse_scores = []
r2_scores = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Fit the model on the training data
    quant_reg.fit(X_train, y_train)

    # Predict on the test data
    y_predd4 = np.round(quant_reg.predict(X_test), decimals=2)

    # Calculate evaluation metrics
    mse = mean_squared_error(y_test, y_predd4)
    r2 = r2_score(y_test, y_predd4)

```

```

    # Append scores to the lists
    mse_scores.append(mse)
    r2_scores.append(r2)

# Calculate the average scores
avg_mse = np.mean(mse_scores)
avg_r2 = np.mean(r2_scores)

# Print the evaluation metrics
print("Average Mean Squared Error:", avg_mse)
print("Average R-squared:", avg_r2)
print("Predictions:", y_predd4)

Average Mean Squared Error: 0.019808333333333333
Average R-squared: 0.9875787354776486
Predictions: [0.71 0.92 1.29 1.73 1.87 2.06 2.83 4.06]

from sklearn.kernel_ridge import KernelRidge
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

dataset = pd.read_csv("Amaranthus_growth_rate_A.csv")
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 5].values

#splitting of the Dataset using Sklearn library in ratio
#of 20% for test and 80% for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
# Create a Decision Tree Regression object
tree_reg = DecisionTreeRegressor()

# Perform k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
mse_scores = []
r2_scores = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Fit the model on the training data
    tree_reg.fit(X_train, y_train)

    # Predict on the test data
    y_predd5 = np.round(tree_reg.predict(X_test), decimals=2)

```

```

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_predd5)
r2 = r2_score(y_test, y_predd5)

# Append scores to the lists
mse_scores.append(mse)
r2_scores.append(r2)

# Calculate the average scores
avg_mse = np.mean(mse_scores)
avg_r2 = np.mean(r2_scores)

# Print the evaluation metrics
print("Average Mean Squared Error:", avg_mse)
print("Average R-squared:", avg_r2)
print("Predictions:", y_predd5)

Average Mean Squared Error: 0.0188055555555555565
Average R-squared: 0.9876609164265139
Predictions: [0.6 0.9 1.3 1.9 1.9 2.1 2.5 4. ]

from sklearn.linear_model import TheilSenRegressor
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

dataset = pd.read_csv("Amaranthus_growth_rate_A.csv")
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 5].values

#splitting of the Dataset using Sklearn library in ratio
#of 20% for test and 80% for training
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
# Create a Theil-Sen Regression object
theil_sen_reg = TheilSenRegressor()

# Perform k-fold cross-validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
mse_scores = []
r2_scores = []

for train_index, test_index in kf.split(X):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

# Fit the model on the training data
theil_sen_reg.fit(X_train, y_train)

# Predict on the test data
y_predd6 = np.round(theil_sen_reg.predict(X_test), decimals=2)

```

```

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_predd6)
r2 = r2_score(y_test, y_predd6)

# Append scores to the lists
mse_scores.append(mse)
r2_scores.append(r2)

# Calculate the average scores
avg_mse = np.mean(mse_scores)
avg_r2 = np.mean(r2_scores)

# Print the evaluation metrics
print("Average Mean Squared Error:", avg_mse)
print("Average R-squared:", avg_r2)

# Print the evaluation metrics
print("Average Mean Squared Error:", avg_mse)
print("Average R-squared:", avg_r2)
print("Predictions:", y_predd6)
Average Mean Squared Error: 0.013824166666666666
Average R-squared: 0.9925642221775035
Average Mean Squared Error: 0.013824166666666666
Average R-squared: 0.9925642221775035
Predictions: [0.76 0.96 1.39 1.87 2.03 2.15 2.69 4.1 ]

fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# First subplot
ax = sns.lineplot(x=y_test, y=y_predd1, label='KNeighbour Regressor', color='cyan',
ax=axes[0, 0])
ax.set_title('KNeighbour Regressor')
ax.set_xlabel('Actual crop growth rate values (cm)', color='g')
ax.set_ylabel('Predicted crop growth rate values (cm)', color='g')

# Second subplot
ax1 = sns.lineplot(x=y_test, y=y_predd2, label='Support Vector Regressor', color='blue',
ax=axes[0, 1])
ax1.set_title('Support Vector regressor')
ax1.set_xlabel('Actual crop growth rate values (cm)', color='g')
ax1.set_ylabel('Predicted crop growth rate values (cm)', color='g')

# Third subplot
ax2 = sns.lineplot(x=y_test, y=y_predd3, label='XGBoost Regressor', color='black',
ax=axes[0, 2])
ax2.set_title('XGBoost Regressor')
ax2.set_xlabel('Actual crop growth rate values (cm)', color='g')
ax2.set_ylabel('Predicted crop growth rate values (cm)', color='g')

```

Fourth subplot

```
ax3 = sns.lineplot(x=y_test, y=y_predd4, label='Quantile Regressor', color='green',  
ax=axes[1, 0])  
ax3.set_title('Quantile Regressor')  
ax3.set_xlabel('Actual crop growth rate values (cm)', color='green')  
ax3.set_ylabel('Predicted crop growth rate values (cm)', color='green')
```

Fifth subplot

```
ax4 = sns.lineplot(x=y_test, y=y_predd5, label='Decision Tree Regressor',  
color='orange',ax=axes[1, 1])  
ax4.set_title('Decision Tree Regressor')  
ax4.set_xlabel('Actual crop growth rate values (cm)', color='g')  
ax4.set_ylabel('Predicted crop growth rate values (cm)', color='g')
```

Sixth subplot

```
ax5 = sns.lineplot(x=y_test, y=y_predd6, label='Theil-Sen Regressor', color='yellow',  
ax=axes[1, 2])  
ax5.set_title('Theil-Sen regressor')  
ax5.set_xlabel('Actual crop growth rate values (cm)', color='g')  
ax5.set_ylabel('Predicted crop growth rate values (cm)', color='g')
```

Adjust spacing between subplots

```
fig.tight_layout()
```

Display the figure

```
plt.show()
```

