Available online at sjuoz.uoz.edu.krd

journals.uoz.edu.krd

Science Journal of University of Zakho

Vol. 11, No. 3, pp. 403– 420, July-September, 2023

p-ISSN: 2663-628X
e-ISSN: 2663-6298

INTELLIGENT HOME: EMPOWERING SMART HOME WITH MACHINE LEARNING FOR USER ACTION PREDICTION

Ayad A. Saleem a,*, Masoud M. Hassan b, Ismael A. Ali b

[a] Technical College of Petroleum and Mineral Sciences, Duhok Polytechnic University, Zakho, Kurdistan Region, Iraq - ayad.abdulrahman@dpu.edu.krd

[b] CCNP Research Lab, Department of Computer Science, Faculty of Science, Zakho, Kurdistan Region, Iraq - (masoud.hassan, ismael.ali)@uoz.edu.krd

Received: 28 Mar., 2023 / Accepted: 14 May, 2023 / Published: 15 Aug. 2023 *https://doi.org/10.25271/sjuoz.2023.11.3.1145*

ABSTRACT:

Smart home is an emerging technology that is transforming the way people live and interact with their homes. These homes are equipped with various devices and technologies that allow the homeowner to control, monitor, and automate various aspects of their home. This can include lighting, heating and cooling, security systems, and appliances. However, to enhance the efficiency of these homes, machine learning algorithms can be utilized to analyze the data generated from the home environment and adapt to user behaviors. This paper proposes a smart home system empowered by machine learning algorithms for enhanced user behavior prediction and automation. The proposed system is composed of three modes, including manual, automatic, and intelligent, with the objectives of maximizing security, minimizing human effort, reducing power consumption, and facilitating user interaction. The manual mode offers control and monitoring capabilities through a web-based user interface, accessible from anywhere and at any time. The automatic mode provides security alerts and appliances control to minimize human intervention. Additionally, the intelligent mode employs machine learning classification algorithms, such as decision tree, K-nearest neighbors, and multi-layer perceptron, to track and predict user actions, thereby reducing user intervention and providing additional comfort to homeowners. Experiments conducted employing the three classifiers resulted in accuracies of 97.4%, 97.22%, and 97.36%, respectively. The proposed smart home system can potentially enhance the quality of life for homeowners while reducing energy consumption and increasing security.

KEYWORDS: Smart home, Machine Learning, Raspberry Pi, Decision Tree, K-Nearest Neighbors, Multi-Layer Perceptron, ANN, User Behavior.

1. INTRODUCTION

The remarkable advancement in technology has facilitated the ability to establish a connection between any device and the Internet, thus giving rise to the notion of the Internet of Things (IoT). The IoT refers to a network of internet-connected devices and objects that range from simple household appliances to complex machinery in various settings (Saleem et al., 2022). These devices collect and share data, which enables them to function seamlessly and offer insights into different aspects of human life (Ibrahim et al., 2022; Taiwo et al., 2022). The integration of IoT has led to enhanced efficiency, convenience, and optimization in both residential and commercial settings, resulting in increased revenue and improved customer service for businesses. The application areas for the IoT encompass various fields such as healthcare, smart homes, smart cities, industrial automation, and transportation. Among these, and since the smart home pertains more closely to individuals' daily lives, the smart home has garnered significant interest from both the industrial and academic communities (Jabbar et al., 2018).

The use of smart homes is growing rapidly and is expected to continue to grow in the future as technology continues to advance and become more accessible. The development of smart homes is being driven by the increasing demand for home automation and the growing need for energy efficiency. However, there are challenges associated with the implementation of smart homes, including cost and security concerns (Jabbar et al., 2019). Nevertheless, the potential benefits of smart homes are substantial and are likely to result in significant positive impacts on the quality of life for homeowners. The goal of smart homes is to increase comfort, convenience, safety, and efficiency while reducing energy consumption. The provision of a user interface (UI) for home control and monitoring is crucial, with a preference for web-based applications accessible at any time and from anywhere via the internet. Such an interface should be user-friendly and compatible with all major operating system platforms, including Android, Windows, and iOS, to facilitate the way the user interacts with their smart homes. In addition to enhancing user comfort, home control systems can also promote energy conservation by enabling automatic control. To ensure maximum security, the system should also include real-time notifications and alarms.

The current era of the Fourth Industrial Revolution (Industry 4.0 or 4IR) has resulted in an abundance of digital data, including IoT data, business data, health data, mobile data, social media data, and cybersecurity data, among others. The effective analysis of these data and the creation of related automated and smart applications require a solid understanding of Artificial Intelligence (AI), particularly Machine Learning (ML). Within ML, there are several different algorithms such as supervised, unsupervised, semi-supervised, and reinforcement learning. Of particular significance is deep learning, a subset of ML that possesses the capability to analyze vast amounts of data intelligently (Sarker, 2021a). In order to increase the intelligence of the smart home, ML methods can be utilized. ML algorithms are increasingly being integrated into smart homes to enhance automation and improve the user experience. These algorithms allow for real-time analysis of large amounts of data and personalization based on user behavior and environmental factors. Smart homes powered by ML are a new and rapidly evolving field, combining the benefits of smart homes with the power of AI. ML algorithms can be used to analyze data generated by smart home systems, such as energy usage patterns, occupancy information, and sensor readings, and hence make predictions about future behavior. ML models have proven to be an effective tool in enabling smart home automation to achieve a

multitude of objectives. These include detecting and recognizing objects, human activities, and faces, as well as controlling household appliances intelligently, optimizing energy consumption, monitoring homes, and enhancing safety and security measures (Taiwo et al., 2022). Smart home employs several ML classification algorithms in its applications, including but not limited to K-Nearest Neighbors (KNN), Decision Tree (DT), Support Vector Machine (SVM), Naive Bayes, and Random Forest. In addition, deep learning (DL) models are also utilized in smart home applications (Saleem et al., 2022). In this work, ML classification model is employed to adapt user behavior and learn from performed actions.

This paper aims to design and implement an intelligent home system empowered by ML methods for enhanced user behavior prediction and automation. The proposed system aims to reduce user effort, power consumption, and human intervention, while increasing security. It includes a web-based UI accessible from any operating system platform to control various home appliances such as lights, TV, air conditioners, window, door, curtain, with the capability to be controlled manually by the user through a friendly UI. The proposed system is also capable of performing automatic tasks without user intervention, such as turning on/off lights in order to reduce power consumption. Furthermore, it provides real-time voice and text message alerts through the UI and email for detecting intruders and fire inside the home. Additionally, the system incorporates the power of artificial intelligence, especially ML algorithms, to increase the system intelligence. This work makes use of various sensors and user actions to predict user preferences, particularly regarding controlling the curtain using a classification model. Additionally, this work applies different classifiers to identify the best model to be implemented in the proposed system. The system provides a responsive UI to monitor and control the home environment. The proposed system offers a smart, secure, and user-friendly solution for intelligent home automation.

The rest of this paper is structured as follows. Section 2 presents a review of related work to enable a comparison with the proposed system. Section 3 provides a background on the fundamentals of smart home, ML, and the various tools utilized in this study. Section 4 details the proposed method, including system architecture and design considerations. Section 5 presents the results obtained from applying various ML algorithms to the created dataset, along with a discussion of their performance. Finally, Section 6 provides the conclusion, highlights the contributions of this work, and outlines potential directions for future research.

2. RELATED WORKS

In this section, numerous state-of-the-art smart home systems are reviewed to provide a comparison to the proposed system.

Various smart home systems were proposed for controlling home appliances and monitoring the home environment without making use of the power of ML algorithms. For instance, (Okorie et al., 2020) proposed a smart, cost-effective system for controlling home appliances and monitoring the status of diverse sensors using smart Android phones, in order to help the elderly and people with disabilities to live their lives in the easiest way. The system consists of an Arduino connected to sensors (light and temperature sensors), appliances (TV, fan, light), and a Bluetooth module to provide communication between the Arduino and the smartphone. This system can be controlled by any android-based device using the "arduDroid" application, which is capable of sending controlling commands to the Arduino, such as turning on/off the TV, fan, and light. In addition, it receives information from Arduino related to the value of temperature and light intensity sensors. The system is easy to use and implement, and costly effective. However, this system is controlled only via Android devices.

In another study, home control and monitoring system based on the web application UI was presented (Iqbal et al., 2018). Their system comprises three main functionalities: door control, fan and light control, and water pump control. In the door control functionality, a motion sensor, a camera, and a Light-Emitting Diode (LED) are installed in front of the door. As soon as any motion is detected, the RPi will receive a signal, and then it will turn the light on automatically in case it is night time. In addition, the camera will take a picture and then store it in the database for security purposes. Thus, the user can open the door via a UI. Furthermore, in the fan and light controlling functionality, the user can turn on/off lights and the exhaust fan. A humidity and temperature sensor are installed to read the temperature and humidity status and show them on the web page for the user to decide whenever to turn on the exhaust fan. Finally, in the water pump controlling functionality, two water level sensors are installed on the top and bottom of the water tank, which are connected to the Arduino to send the signals through ZigBee to the RPi to decide when to turn on the water pump automatically. The data from the sensors' and actuators' status is temporarily stored on the MySQL database in RPi and then backed up on the cloud server to be ready for a user history report or even a third-party service provider. As an advantage of the system, an online UI is used to make the system capable of being accessed anywhere and anytime, and the authors took into consideration sensors' status before controlling the actuators. Although the system has multiple features, no ML method is used.

The authors in (Pavithra & Balakrishnan, 2015) proposed a smart home system for the purpose of controlling home appliances and detecting fire. The system is designed to operate through the use of a central controller, which is based on the Raspberry Pi (RPi) platform. Specifically, the system employs relays to switch on and off lights when an infrared (IR) sensor detects an object, and a Passive Infrared (PIR) sensor to switch on the fan when motion is detected. Additionally, the system integrates a fire detection sensor that, when activated, triggers a camera to capture an image of the fire in order to send it along with an alarm message to the user's phone. The phone subsequently makes an automatic call to the closest fire station. The control and monitoring of the system's lights and fans can be managed manually through a web page UI, which is accessible from any operating system platform. It is worth noting that the system's architecture does not involve the use of ML algorithms. Consequently, the system does not have the capability to learn or adapt to changing environmental conditions. While the system's design is commendable in its utilization of a central controller for home automation and fire detection, it does not exhibit intelligent behavior as ML algorithms would provide.

The authors in (Gota et al., 2020) designed and implemented a home automation system by controlling lighting, doors, and windows, as well as monitoring temperature and humidity inside and outside the home. The process of controlling and monitoring was done through a web page to make the system usable on different operating systems. In order to control the windows and doors inside the house, a servo motor was used, which can rotate at an angle of 0 to 180 degrees. Another type of motor called a stepper motor was used to control the opening and closing of the garage door, as this type has the ability to rotate in two directions and at multiple angles. The motors stop when they reach a certain point that is sensed by the magnetic sensor. For manual or automatic ventilation, the temperature inside and outside the house was read along with the humidity reading. Thus, when a certain temperature is reached, the ventilation system is turned on automatically or manually via the web page.

On the other hand, several smart homes systems have been proposed that incorporate ML algorithm for controlling home appliances and enhancing home security. For example, (Abbas & Abdullah, 2021) proposed an innovative approach towards

tracking and predicting user behavior using ML algorithms. The system comprises two essential components, a RPi and a NodeMCU ESP32, which work in tandem to perform the intended function. The television is connected to the ESP32 via a relay, while the pushbutton serves as a switch to activate or deactivate the TV. The RPi records and archives the device's status and user behavior, which are then subjected to a classification process. The decision to turn the television on or off is derived through the use of DT algorithm. This model is adaptable, allowing for updates to be made with newly collected daily data. It is noteworthy that the system does not include a UI and does not utilize any wireless communication technology.

In another study by (Mehmood et al., 2019), the authors devised a deep learning-based system for the detection of individuals. Their system utilized Amazon Web Service to facilitate remote monitoring by the user. The system comprises a camera module, connected to a RPi, which captures live video and employs the Single Shot MultiBox Detector (SSD) algorithm trained with the Microsoft Common Objects in Context (COCO) dataset for video analysis. Upon detection of a person, notifications are sent to the user either through Short Message Service (SMS) or email. Additionally, a control message is transmitted to the actuating device, a Node MicroController Unit (NodeMCU) ESP8266, to turn on or off a LED. The system's accuracy is impacted by decreasing light intensity, increasing distance, and larger frame size, resulting in a reduction from 95-100 to 80-85 percentage.

(Crisnapati et al., 2016) proposed a smart home monitoring and control system that utilizes fuzzy logic to decrease energy consumption. The system features an HD camera that is integrated with a motion detection system, a motion sensor, and RFID for security. In this system, the user can manage recorded videos and captured photos, control functions (turning lights on and off), and monitor the home through a user-friendly web interface. Furthermore, the system's artificial intelligence capabilities allow it to automatically control temperature and lighting to promote energy efficiency.

The authors in (Paredes-Valverde et al., 2020) presented a sophisticated system designed for power consumption monitoring and management, called IntelliHome. The system utilizes data obtained from the usage of IoT devices such as electrical appliances, sensors, and smart switches, to provide energy-saving recommendations to the user based on their behavior and preferences. Upon accepting the recommendation, the system implements control over home appliances. The analysis and processing of the data are handled by the Home Energy Consumption Monitor (HECM) module, which utilizes the Holt-Winters-RNN algorithm. Although the system provides a UI, it is only compatible with Android devices.

In (Peng et al., 2019), the authors propose an intelligent home-control system with the utilization of a convolutional neural network (CNN) to recognize and classify human gestures (human point attitude). The system makes use of an Arduino as a central controller for controlling the window, air conditioner, and LED. The gestures are captured via a Kinect sensor that is connected to a personal computer, which processes the obtained data and sends resulting information via Zigbee wireless technology to the Arduino to execute controlling commands. Finally, the obtained results were excellent for all gesture classifications. The system does not have a user interface.

The authors in (Raju et al., 2021) presented a smart home system that enables monitoring of the home, control of appliances such as lights and fans, and prediction of power conservation. This system is built using a Raspberry Pi, various sensors (including motion, temperature, light, and sound), and actuators. Users can control their appliances via a mobile application that sends commands to the Raspberry Pi, which then sends control commands to the relay module to turn the appliances on or off. The system can also automatically control appliances, and users can view the energy consumption of each device via the user interface. To predict energy consumption, the system employs several machine learning algorithms, including decision tree regression, KNN, support vector regression, and random forest regression. The PIR sensor is used to control both the light and the fan, and Bluetooth communication is used to connect the mobile phone with the Raspberry Pi for wireless communication. While the system has a user interface, it cannot be accessed via the internet. Notably, the Decision Tree algorithm was found to provide the highest accuracy among all applied algorithms.

3. BACKGROUND

This section provides a brief overview of smart home systems, and machine learning methods, along with the various tools and devices utilized in this work.

3.1 Smart Homes

As defined by (Marikyan et al., 2019), a smart home “is a residence equipped with smart technologies aimed at providing tailored services for users”. A "smart home" is referred to as a home automation system that is designed with controlling, monitoring, and sensing functionalities such as surveillance, ventilation, lighting, and conditioning systems. These smart systems consist of various essential components, including actuators and sensors connected wired or wirelessly to a central controller (Saleem et al., 2022). This controller receives data from sensors, and sends controlling commands to the actuators. Controlling commands are provided either by the user via a UI or automatically based on some pre-programmed conditions. Furthermore, householders can monitor the houses via a graphical UI using a tablet, computer, or smartphone (Saleem et al., 2022).

Smart home as one of the most common IoT applications (Nikou, 2019), provides access to the components remotely anytime and anywhere via any smart devices (Choi et al., 2021). Smart homes offer diverse services to the homeowner, including:

- Remote monitoring: monitor the environment inside and around the home from anywhere and at any time. For instance, monitoring the devices' status (on/off), and sensors' status (readings).
- Remote controlling: controlling the home appliances anytime and anywhere via the UI.
- Reducing human effort: the user can control home appliances by their phones without any physical movement. For example, switching on the air conditioner, TV, and lights, as well as opening the door.
- Reducing power consumption: for example, turning the lights off during the daytime. In addition, turning the lights on when motion is detected somewhere inside the home would lead to saving energy.
- Security maximizing: integrating a security option in the smart home is essential to prevent a home from being stolen and to provide a real-time alert. For example, installing a surveillant camera in addition to setting up a motion detector to alert the householder by sending an alarming message via email or phone calls, as well as activating buzzers.

3.2 Raspberry Pi

Raspberry Pi (RPi) is a small-sized highly performance single board computer (SBC) that has all standard computer components including processor, RAM, I/O units, and GPU, incorporated in a single board (Jolles, 2021; Pajankar, 2021). In addition, as compared to the available SBCs in the market, the RPi is one of the most common SBCs and the best-selling computers in the world (Pajankar, 2021). Unlike traditional computers, the main disadvantage of the RPi is that the hardware component cannot be upgraded (Pajankar, 2021). Like the other SBCs and due to their suitable size and performance, the RPi is

essentially employed in embedded systems, especially for robotics and IoT applications. The first model of RPi was released in February 2012 and was developed by the RPi foundation in the United Kingdom (Jolles, 2021; Kurniawan, 2019). Among all RPi model, the newest model is the RPi 4 model B which comes with the following specifications: 64-bit quad-core Cortex- A72 1.5GHz Broadcom BCM2711 processor. LPDDR4 3200 SDRAM comes with 1GB, 2GB, 4GB, or 8GB of RAM. Dual-band 2.4GHz and 5GHz wireless networking IEEE 802.11ac, gigabit ethernet, and Bluetooth 5, two USB 3.0 and two USB 2.0 ports. An array of 40-pin headers (described in Figure 2), 28-pins out of these 40 are General Purpose Input/Output (GPIO) used to connect sensors and actuators for controlling and monitoring purposes. Two micro-HDMI ports support up to 4k video streaming. Serial interface port to connect RPi camera, microSD card slot, and it operates with 5V 3A DC input power, (See Figure 1). Although RPi has its official operating system called Raspberry Pi OS (Raspbian OS, previously) (Jolles, 2021), it can run with other OS including android, windows 10 IoT, and ubuntu OS family.

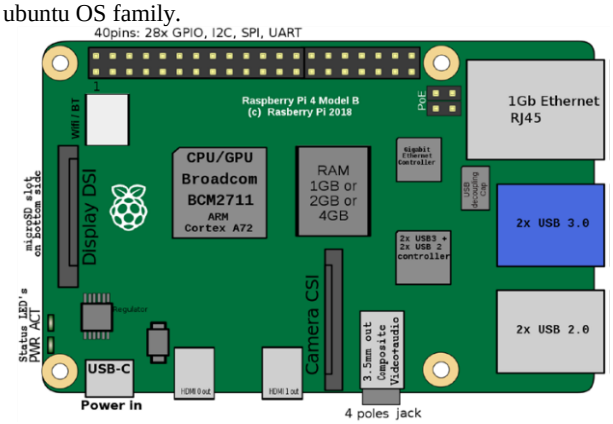


Figure 1: Raspberry Pi 4 model B on board components (Pajankar, 2021).

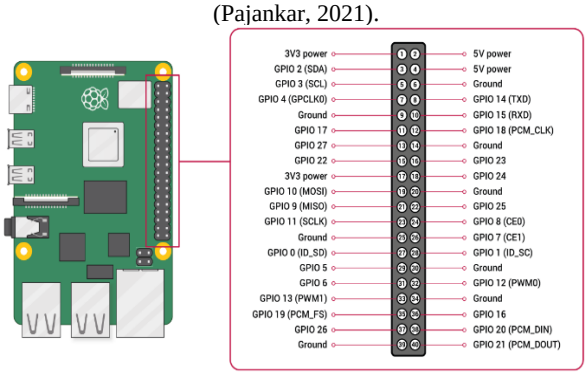


Figure 2: Raspberry Pi 40-pins header description (*Raspberry Pi Documentation*, n.d.).

As illustrated in Figure 2, the RPi consists of 40 pins defined as follows:

- Two power pins provide 5V, and two power pins provide 3.3V.
- 8 unconfigurable ground pins provide 0V.
- The other 28 pins called GPIO pins, which can be used either as output and can be set from 3.3V (high) to 0V (low), or as input which can read also from 3.3V to 0V.

3.3 Machine Learning

Machine Learning (ML) is a branch of AI that consist of a set of algorithms and techniques that enable computer systems to learn and make predictions or decisions to learn from available data based on previous experiences (Bertolini et al., 2021) without being explicitly programmed. ML methods are used in diverse areas including object detection, text and speech interpretation, classification and pattern recognition (Singh et al., 2020). Prior to the implementation of the algorithm on a specific problem, the algorithm is trained on a dataset (available data) to result in the most accurate model. The dataset contains a number of columns called attributes or features, in addition to the output variable (in case of supervised learning). Furthermore, the dataset should be split into two sets, one used for training the model, called the training subset. In contrast, the other called the testing subset utilized for testing the model accuracy (Bertolini et al., 2021). Moreover, the dataset features can be continuous, binary, or categorical (Singh et al., 2020), and the output variable could be continuous or categorical (Bertolini et al., 2021). The types of available data help to choose the best-fit algorithm for the available case.

ML consists of several algorithms, each of which is intended to solve specific kinds of problems, such as classification (in case of categorical output), regression (in case of continuous output), and clustering. There are three main types of ML: supervised learning, unsupervised learning, and reinforcement learning. Classification and regression, are methods of supervised learning, in which the data are labelled (consisting of input and output) (Bertolini et al., 2021; Singh et al., 2020). While the clustering is an unsupervised learning (output variables are missing) (Bertolini et al., 2021; Singh et al., 2020). The classification (predictive learning) comprises various algorithms including Decision Tree (DT), Naive Bayes (NB), Random Forest (RF), Artificial Neural Network (ANN), K-Nearest Neighbors (KNN), Logistic Regression (LR), and many other algorithms that utilized for prediction (Bertolini et al., 2021; Singh et al., 2020). While unsupervised learning, which is also called descriptive learning, analyzes the given dataset and intends not to predict the missing output, but to discover the hidden pattern behind the given data (Bertolini et al., 2021). For example, in clustering, the data are divided into several groups, each of which contains data that are similar to each other, but differ from the other groups (Bertolini et al., 2021). The most common clustering algorithm is K-means. Semi-supervised learning is the combination of both supervised and unsupervised learning which can handle labelled and unlabelled data. It is useful when the dataset contains a small number of labelled data and a large number of unlabelled ones (Sarker, 2021b; van Engelen & Hoos, 2020). Further, it is located between supervised and unsupervised learning. As it aims to enhance the performance of one of the previous techniques by making use of data corresponding to the other. For example, in case of handling classification problems, a large amount of unlabelled samples are utilized to improve the classification process. While in clustering, labelled observations can be used to improve the clustering process as well (van Engelen & Hoos, 2020). A fraud detection and machine translation considered application of semi-supervised method (Sarker, 2021b).

ML has a wide range of applications in various industries, such as finance, healthcare, retail, and many others. It is used to solve a variety of problems, such as predictive analytics, image recognition, natural language processing, robotics, spam filtering, face recognition, handwriting and speech recognition, DNA classification, and computer games (Singh et al., 2020). In the proposed work, supervised ML classification is used in the smart home system to predict the user action regarding to opening and closing the curtain. In this work, several classifiers, including DT, KNN, and MLP, are applied to the Curtain dataset to predict user action. These three classifiers were selected to perform the comparison among the tree-based algorithm, the distance-based algorithm, and the optimization-based algorithm in order to recognize the impact.

3.4 The Decision Tree (DT)

The Decision Tree (DT) is a tree-like graphical representation classifier used for supervised learning in both regression and

classification tasks (Géron, 2019; Mukherjee et al., 2019). It consists of nodes, with decision nodes representing attributes and leaf nodes representing class labels, that are connected via arrows, namely directed edges. Iterative Dichotomiser 3 (ID3) and C4.5 are the most commonly used algorithms for constructing DTs (Mienye et al., 2019). ID3 is used only for categorical data, while C4.5 can be used for both numerical and categorical data (Mienye et al., 2019). To construct a DT, impurity measures (such as Entropy or Gini) and information gain are utilized for each feature, whereas the most informative attribute with the maximum information gain is selected as the root node (Mienye et al., 2019). This process is repeated to determine the best-fit attribute for each node until all attributes are included in the tree. The resulting DT is then translated into rules comprising if-then statements (Mukherjee et al., 2019; Patel & Prajapati, 2018). Decision trees have several advantages, including their interpretability, simplicity, and ability to handle both categorical and numerical features.

3.5 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) algorithm is another supervised learning algorithm, which is considered the simplest ML algorithm as it requires only storing the training dataset to build the model (Gao & Li, 2020) KNN is utilized for solving both classification problems for categorical label, as well as regression problems for numerical label (Jung, 2022). The key parameter of the KNN algorithm is the number of neighbors that will be considered (Jung, 2022). In this paper, KNN is employed for classification of user behaviors. In order to predict the class of the new data example, KNN intends to determine the closest data examples out of all the examples in the training dataset (Li et al., 2023). The simplest version is to set K to 1. Thus, it means considering only one nearest neighbor in order to figure out which class the new data point belongs to. In general, for a binary classification, it is preferable to set k greater than two; in this case, voting is utilized to determine the correct label. This implies that the correct label is the class that has the most frequent neighbors to the new data point (Mahmoodzadeh et al., 2020). KNN makes use of distance metrics to calculate the distance between the new point and all other training set points (Kubat & An, 2021). Indeed, the three most commonly utilized distance metrics are Euclidean, Manhattan (also known as city block distance), and Minkowski (Gao & Li, 2020). The Minkowski is a generalization of both Euclidean (when p = 2) and Manhattan distance metric (when p = 1) (Gao & Li, 2020).

3.6 Artificial Neural Network (ANN)

Artificial Neural Network (ANN) is a complex system that operates similarly to the human brain and nervous system which can provide self-learning (Desai & Shah, 2021). ANN is one of the most well-known ML techniques used for the learning process and provides satisfying results for complex problems that cannot be easily interpreted (Desai & Shah, 2021). Furthermore, ANN can solve various problems, including regression, classification, and clustering (Heidari et al., 2020). The learning process can be performed via several iterations called epochs (Desai & Shah, 2021), each iteration consists of two major phases, including the feed-forward and backpropagation processes. ANNs have various forms to represent, such as single-layer perceptron (SLP), MLP, and deep learning (Heidari et al., 2020). SLP consists of only two layers: the input layer and the output layer. It cannot perform well with patterns that are not linearly separable. For this reason and in order to alleviate the drawbacks of SLP, MLP was developed. Unlike SLP, this kind has more than two layers, including an input layer, hidden layers, and an output layer. The input layer comprises a number of neurons representing the number of features in the dataset, while the output layer is composed of only one neuron (Car et al., 2020). The key benefits of MLP are that it performs well with the availability of noise, provides high learning accuracy, and handles non-linear separable data (Heidari et al., 2020). The performance of MLP is highly influenced by various factors, such as wights vector and learning technique.

MLP is one of the most common FeedForward Neural Networks (FFNNs) algorithms that offers a high degree of reliability based on the layered structure of the neural network. The layers in FFNNs consist of nodes called neurons. Each of which is fully connected by connection links to all neurons of the next layer (Desai & Shah, 2021; Heidari et al., 2020). These edges are associated with a real number called weights. Each node can carry out two kinds of functions (Heidari et al., 2020); summation function, which is calculated by taking the summation of the production of input values and weights, with adding bias weight. Thus, the result of the summation function is passed through an activation function to determine whether the next node will be activated or not (Heidari et al., 2020). Several activation functions can be applied, including the most common ones such as Identity (the values stay the same), ReLU (only positive values will be accepted, and the negative ones will be replaced with zero), Sigmoid (the output value will always be in the range 0 to 1), and Tanh (mapping the value of the summation function into the range from -1 to 1). The working mechanism of MLP is first started from the input layer, which is called forward propagation, represented by setting initial weights for the edges (Heidari et al., 2020), the output of the neurons in one layer is considered to be the input of the next layer neurons, after adding weight values to them (Desai & Shah, 2021). This process represents one iteration. For repeating iteration, the process is passed through the backpropagation technique, which operates in reverse from the output layer, ending in the input layer. This technique provides weight updating to enhance accuracy and it is repeated until the desired results are obtained that have the minimum loss (Desai & Shah, 2021; Heidari et al., 2020). This algorithm is used in this study to predict the user action behaviors.

3.7 Cross Validation

Cross-validation is a statistical technique used to evaluate the performance of a ML model by testing it on multiple subsets of the available data. The basic idea behind cross-validation is to split the available data into two sets: a training set, which is used to fit the model, and a testing set, which is used to evaluate its performance. This technique is used to evaluate the model, and to avoid overfitting and underfitting problems. In general, cross-validation can be applied with two common methods, including hold-out and K-Fold (Mahmoodzadeh et al., 2020; Tatsat et al., 2020). The hold-out method divides the original dataset into training and testing samples, while the K-Fold method divides the data into several folds (subsets) based on the value of K, each of which has the same size. One fold is considered as the testing set and the other $K - 1$ folds are the training set (Mahmoodzadeh et al., 2020). Therefore, the testing set is evaluated using evaluation metrics (such as accuracy and F1-score) for each fold independently to gain K results. Consequently, the final result is obtained by calculating the average of the obtained K (Tatsat et al., 2020). Logically, in the hold-out method, the size percentage of the training set must be large as compared with the testing set size. Figure 3 demonstrates the idea behind K-Fold cross-validation (Tatsat et al., 2020).

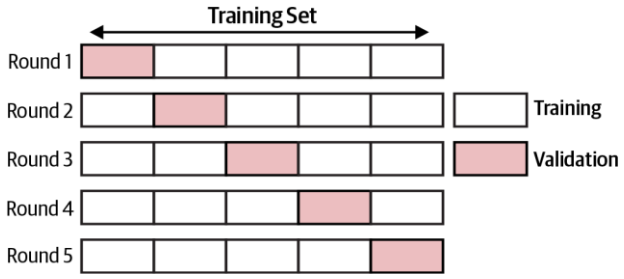


Figure 3: K-Fold cross validation when K=5.

The strength of the K-Fold technique is that each data point in the original dataset is considered to be testing data once, and as training data K-1 times, leading to a decrease in bias. Because of this, it was implemented in this study in the experiments conducted with K-Fold values equal to 3, 5, and 10.

3.8 Evaluation Metrics

In order to evaluate the performance of the classification model, different evaluation metrics are used. Basically, these metrics rely on the confusion matrix, which is a table used to evaluate the performance of a classification algorithm. It is a summary of prediction results on a binary classification problem. The confusion matrix comprises four parts including True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). Thus, this matrix shows the relationship between the actual class and the predicted one (Klok & Nazarathy, 2021). Furthermore, when the classifier predicts the class that matches the actual one, it means it is a TP and TN case. Moreover, as soon as the predicted value does not match the actual value, then it is the case of either FP or FN. Positive classes refer to the class with the values 1, True, or Yes; while the Negative classes represent the values 0, False, or No, and so on. The most common measures obtained from confusion matrix are: Accuracy, Precision, Recall, and F1-score.

The confusion matrix is demonstrated in Table 1, and the evaluation metrics are defined as in the following equations:

$$\text{Accuracy} = \frac{|\text{TP}| + |\text{TN}|}{|\text{TP}| + |\text{TN}| + |\text{FP}| + |\text{FN}|} \quad (1)$$

$$\text{Precision} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|} \quad (2)$$

$$Recall = \frac{|TP|}{|TP| + |FN|} \quad (3)$$

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (4)$$

Another evaluation metric was employed called the Area Under the Curve (AUC) which represents the area that falls under Receiver Operating Characteristics (ROC). AUC is a widely used performance metric in ML, which measures the ability of a binary classification model to distinguish between positive and negative classes by computing the area under the ROC curve. It is a robust and intuitive metric that provides a single scalar value summarizing the overall performance of the model, which is insensitive to the imbalance of class distributions.

Table 1: Confusion matrix for binary classification problem.

		Actual Class	
		Positive (1)	Negative (0)
Predicted Class	Positive (1)	TP	FP
	Negative (0)	FN	TN

4. PROPOSED SMART HOME SYSTEM

This section discusses the structure of the proposed smart home system, including hardware components (appliances, sensor, and devices), and system workflow (with and without ML).

In order to build the proposed system, various devices and sensors were utilized and connected together to collect and exchange data with each other. Figure 5 demonstrates the home architecture of the proposed system and its connected sensors, and appliances. Figure 4 provides a top view of the proposed system prototype. The objectives of the utilized devices and sensors in the proposed system are illustrated clearly in Table 2. In addition, Table 3 shows each sensor and device and their signal type, provided voltage (VCC: Voltage Common Collector), pin mode, and allocated RPi pin number.

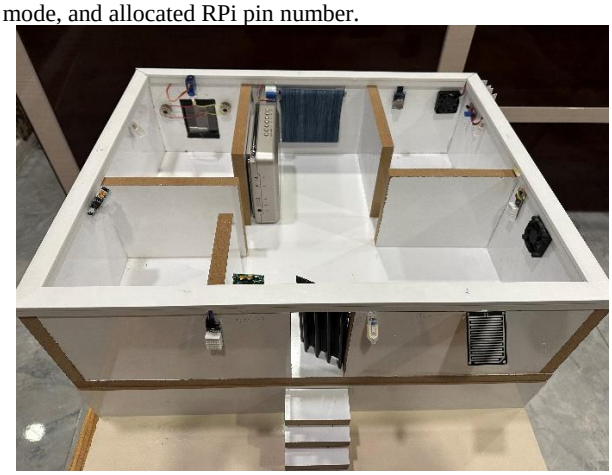


Figure 4: Top view of the prototype.

Table 2. The objective and the position of utilized devices and sensors.

Device	Count	Place	Objective
RPi	1	Controlling box	To connect all devices, send commands, and receive and store data. As well as implementing ML model. Considered local server.
PIR	2	Bathroom	Sensing human existence to turn on the bathroom light
		Living room	Sensing human existence to notify the householders about intruders when "nobody home" mode is on. As well as to turn off the tv after a period of no movement found.
Light Dependent Resistor (LDR)	1	Outside in front of the main door	Installed outside to calculate the real-time light intensity. Useful especially for ML mode
Digital Temperature and Humidity (DHT22)	2	Bedroom	Sensing the temperature and humidity inside the room
		outside	Sensing the temperature and humidity outside
Gas sensor (MQ-2)	1	Kitchen	To measure the intensity of smoke and gas.
Raindrops	1	Outside	Detecting if it is rainy or not.
Flame	1	Kitchen	Detecting if there is a flame inside the kitchen
Analog-to-Digital Converter System (ADS 1115/1015)	1	Controlling box	Converts analog signals received from sensors into digital signals.
Stepper motor	1	Living room	To open and close the living room curtain in both modes: user commands and intelligent (ML) mode

Servo motor	2	Living room	Controlling window
		Main door	Controlling door

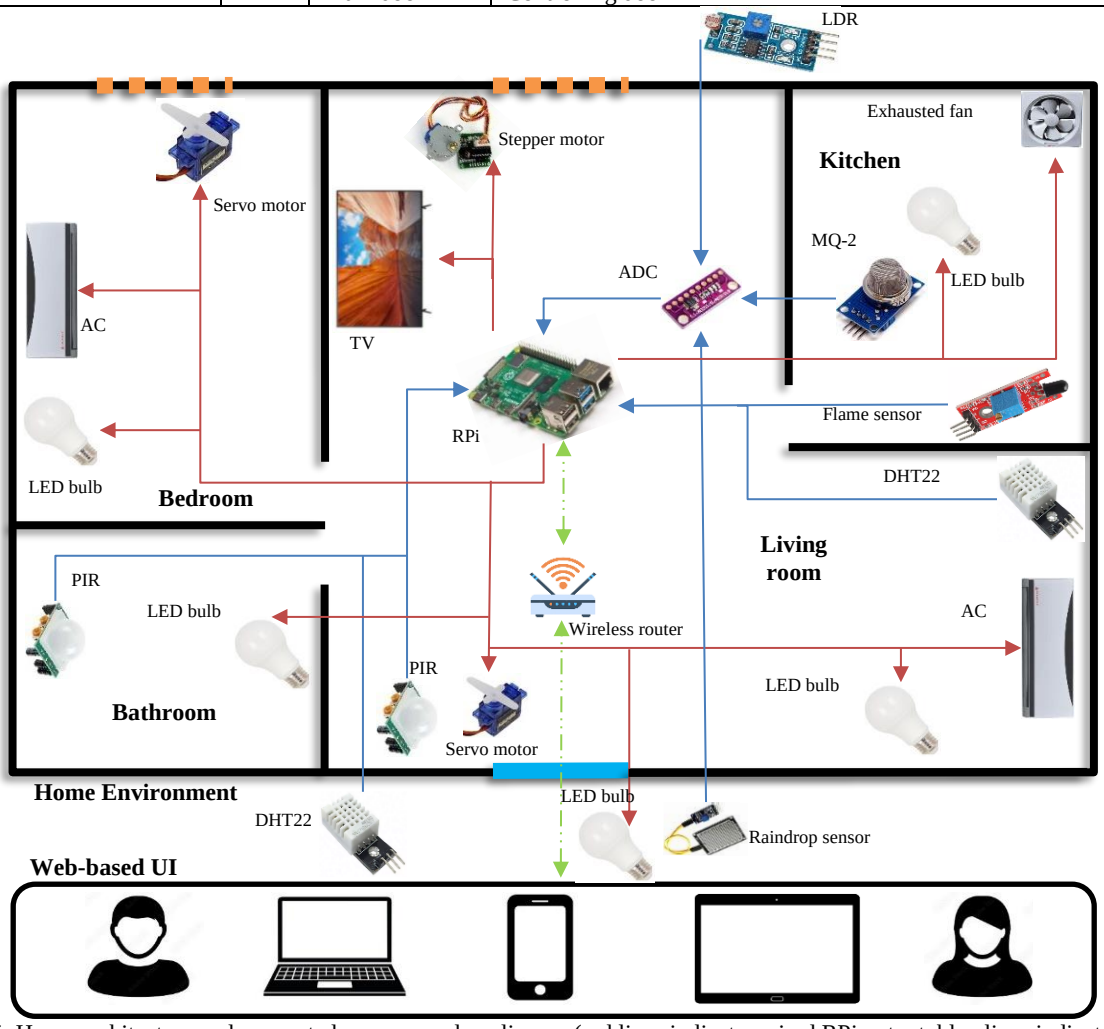


Figure 5: Home architecture and connected sensors, and appliances (red lines indicates wired RPi output, blue lines indicate wired RPi input, and the green lines refers to wireless communication).

Table 3. Home overall utilized sensors, actuators, and appliances associated with allocated pin number and provided voltage.

Sensers / Appliances	Place	Allocated pin number	Pin mode	Signal type	VCC
PIR	Bathroom	6	Input	Digital	5v
	Living room	19	Input	Digital	5v
DHT22	Living room	26	Input	Analog	3.3v
	Outside	23	Input	Analog	5v
ADS 1115/1015	Between analog sensors and RPi GPIO	I2C protocol (SCL, SDA)	Input	Analog to Digital converter (ADC) A0, A1, A2, A3	3.3v
LDR	Outside	A0	Input	Analog	5v
MQ-2	Kitchen	A1	Input	Analog	5v
Raindrops HL-83	Outside	A2	Input	Analog	3.3v
Flame	Kitchen	24	Input	Digital	5v
Stepper	Living room curtain	10, 16, 20, 21	Output	Digital	5v
Servo	Bedroom window	13	Output	Analog	5v
	Main-door (outdoor)	12	Output	Analog	5v
Exhausted fan	Kitchen	25	Output	Digital	12v external adapter
LED bulb	Kitchen	7	Output	Digital	12v external adapter
	Bedroom	4	Output	Digital	12v external adapter
	Bathroom	17	Output	Digital	12v external adapter
	Outside	8	Output	Digital	12v external adapter

	Living room	18	Output	Digital	12v external adapter
TV	Living room	27	Output	Digital	12v external adapter
Air conditioner (AC)	Living room	22	Output	Digital	12v external adapter
	Bedroom	5	Output	Digital	12v external adapter

4.1 Dataset Description and Design

The proposed work intends to create a dataset, referred to as "Curtain dataset" that captures user behavior regarding the opening and closing of curtain inside the house. Although the dataset is intended to store real-time data received from the proposed smart home system, the first version of the Curtain dataset was generated using logical simulation data representing as much as possible real data. The dataset represents a classification-based problem that records the data based on specific times and includes six input features and a binary classification output (user behavior). It comprises 8030 samples collected over one year, with some features were encoded, such as in "time", "weekday", and "out_light", while other features remain in raw data format.

Table 4 provides a detailed description of the Curtain dataset, including the encoding methodology used for each feature. Indeed, before generating the semi-real simulation data, experiments were conducted on the sensors utilized to identify the range and factors that influence their values. For example, the "out_light" feature, which represents the value of the LDR sensor, is influenced by the date (month) and time, particularly sunrise and sunset times (noted in (*Sunrise and Sunset Zakho Dahuk Iraq*, n.d.)) for each month.

Figure 7 demonstrates the correlation between the dataset features and the output label, while Figure 6 displays the frequency distribution of the available classes. Some features in the dataset were generated randomly, whereas this randomness was controlled based on the established range and experiments performed. The Curtain dataset is updated daily every 30 minutes with actual real-time sensors data to provide accurate prediction for the model. Table 5 provides a sample of the Curtain dataset, consisting of a few observations.

Table 4: Curtain dataset description.

Input/output	Column name	Data type	Description and range
Input feature	date	Integer	Represent the month from 1 to 12
	time	Integer	Represents the time of 24 hours, represented as follows: 20-4:29=0, 4:30-4:59=1, 5:00-5:29=2, 5:30-5:59=3, 6:00-6:29=4, 6:30-6:59=5, 7:00-7:29=6, 7:30-7:59=7, 8:00-8:29=8, 8:30-8:59=9, 9:00-9:29=10, 9:30-9:59=11, 10:00-10:29=12, 10:30-10:59=13, 11:00-16:29=14, 16:30-16:59=15, 17:00-17:29=16, 17:30-17:59=17, 18:00-18:29=18, 18:30-18:59=19, 19:00-19:29=20, 19:30-19:59=21.
	weekday	Binary	Represents either 0 (weekend: Friday and Saturday) or 1 (working days: Sunday to Thursday)
	motion	Binary	Generated randomly. Represents the current value of the motion (PIR) sensor 0 = no motion, 1 = motion detected.
	out_light	Integer	Generated randomly taking into account the time and date feature. Represents the outside light intensity which is calculated based on the value that passes from the resistance (LDR sensor) to the RPi GPIO pin. The lower the value received from RPi, the higher the light intensity. The values are converted to percentages to make them easy to understand.
	smoke	Integer	Generated randomly based on the tests conducted. Represents the data of the gas/smoke sensor. The normal rang when no gas or smoke detected is from 350 to 390. The cases of greater than 400 means smoke or gas is detected.
Classification output	user action	Binary	Represents the user behavior. 0 = closing the curtain, 1= opening the curtain.

Table 5: Curtain Dataset Sample.

date	time	weekday	motion	out_light	smoke	user action
7	6	0	0	82	362	0
7	7	0	0	83	382	0
7	8	0	0	84	352	0
7	9	0	1	84	359	0
7	10	0	0	85	373	0
7	11	0	0	86	355	1
7	12	0	0	87	354	1
7	13	0	1	88	367	1
7	14	0	0	89	386	1
7	15	0	1	88	361	1
7	16	0	0	84	364	1
7	17	0	1	79	366	1
7	18	0	1	65	358	1
7	19	0	0	60	387	1
7	20	0	0	40	387	0
7	21	0	1	0	380	0
8	0	1	1	0	373	0
8	1	1	0	0	374	0
8	2	1	0	38	389	0
8	3	1	1	70	355	0
8	4	1	1	78	351	0
8	5	1	1	82	364	0
8	6	1	1	82	385	1
8	7	1	0	83	387	1
8	8	1	0	84	372	1

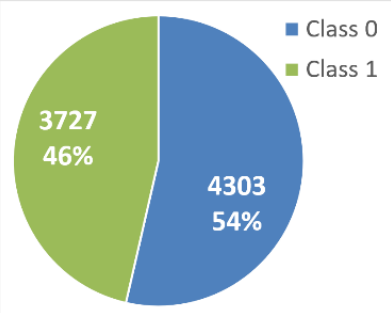


Figure 6: Curtain dataset class distribution.

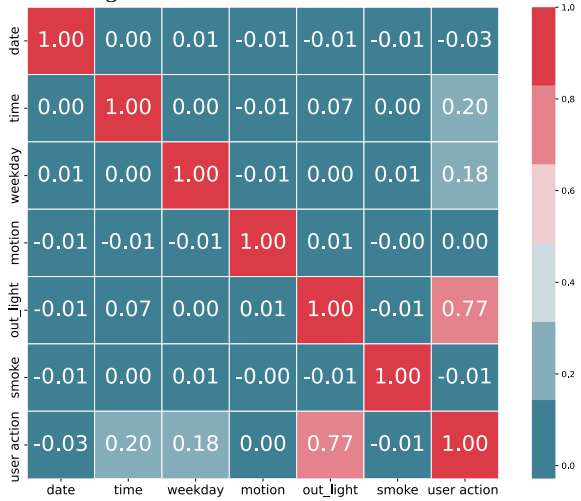


Figure 7: Curtain dataset column correlations.

4.2 System Workflow

The proposed system is designed to provide various functionalities that enable the creation of a fully automated home system with enhanced security and comfort, while also reducing power consumption and human effort. For this, the system consists of three modes, namely manual, automatic, and intelligent (ML) modes. In the manual mode, the user has full control over the appliances and can operate them through user intervention. In constant, the automatic mode allows the system to control the appliances based on predefined conditions using sensor values. Finally, in the intelligent mode, the system employs ML algorithms to perform actions based on the output of the trained model.

By integrating these three modes, the system provides a robust and flexible solution that can adapt to the user's preferences and environment conditions. This approach allows for the creation of a more efficient and effective home system that reduces energy consumption and increases comfort and security. In the following subsections, each mode is described in detail to provide a better understanding of the proposed system's capabilities and functionalities.

4.3 Manual Mode:

In this mode, the user sends control commands to the actuators using a web-based UI to control the following appliances:

- Opening and closing the living room curtain using a stepper motor.
- Turning on/off lights in the bedroom, living room, and kitchen via the relay module.
- Switching on/off the living room and bedroom air conditioners (represented as a fan in the prototype) via relay module.
- Switching on/off the TV using relay module.
- Turning on/off the kitchen exhaust fan using relay module.
- Controlling the main door and bedroom window by utilizing servo motors.

4.4 Automatic Mode:

This mode provides automatic control based on predefined conditions that rely on sensors' values. The following actions demonstrate the mode:

- Based on the outside darkness obtained from LDR, turning on or off outside light.
- Based on human existence, turning on and off the bathroom light. Thus, the light will turn on as soon as motion is detected via the PIR sensor, and it will turn off after 2 minutes, starting from the time when no motion is detected.
- Turning on the exhaust fan when abnormal smoke/gas is detected. In addition to sending an email, and viewing text and voice messages on the UI.
- For security reasons, the system contains an option called "Nobody at home". The home residents can turn on this option when they leave home to tell the system to send an alert via email and also show a text and voice notification on the GUI when the PIR (installed in front of the main door) senses any motion. Thus, detecting the intruder.
- Automatically close windows when rain is detected via raindrop sensor.
- Turning off the TV automatically after 10 minutes starting from the moment when no motion is detected.
- Sending email and viewing text and voice alerts as soon as a fire is detected.
- The system automatically stores the values of the mentioned sensors in the Curtain dataset every 30 minutes based on the specific pattern.
- Show text and voice alert if the outside temperature is very hot or very cold.
- Show text and voice alerts if the main-door stays open for a while.

4.5 Intelligent Mode:

The proposed system is incorporated with an intelligent mode which relies on using ML technique.

This mode can be activated by the user via the UI, allowing the system to predict user preferences for automatically opening or shutting the curtain. In this mode, the prediction is performed using a DT classifier. The reason behind using this predictor is discussed in the next section. The process of storing data into Curtain dataset was presented in section 1.1. Incorporating this part into the proposed work resulted in reduced user effort and energy consumption, as the user will no longer need to turn on the light when the outside sunlight enters the home after opening the curtain. This also demonstrates the potential of ML to make home environments smarter, more comfortable, and more feasible.

Figure 8 depicts the workflow of the proposed system, highlighting the use of ML in predicting user preferences and automating the opening and closing of the curtain. The decision tree classifier was chosen for its ability to handle both numerical and categorical data, making it suitable for the data collected in the "Curtain" dataset.

Figure 8 illustrates the functioning of the intelligent mode in detail. This mode is designed to leverage the RPi to collect and store data from sensors it in the Curtain dataset, along with the corresponding date (month, and weekday) and time after encoding some data and converting some others into a more comprehensible format. Prior to fitting classification models, the essential process of feature selection and the classification label determination is carried out. Moreover, feature scaling techniques such as MinMax Scaler (for the KNN classifier) and Standard Scaler (for the MLP classifier) are applied, with the exception of the DT classifier, since its performance not affected by feature scaling. The dataset is subsequently split into training and testing sets utilizing the K-Fold cross-validation method. The

training sets are used for the fitting process to obtain the trained model, which is then used to predict the output of the testing set comprising previously unseen samples. Several models are trained to determine the optimal model and classifier for the intelligent mode. The selected model is then utilized by the RPi to periodically check the sensors' data, along with the date and time, every 10 minutes to predict the final decision of opening or closing the curtain.

5. RESULTS AND DISCUSSION

This section represents the results of various experiments carried out the "Curtain" dataset using K-Fold cross-validation (with a specified random state parameter of 42) to predict user behavior regarding the opening and closing the curtain. The experiments involved applying three different classification algorithms: decision tree (DT), K-Nearest Neighbor (KNN), and Multilayer Perceptron (MLP) artificial neural network (ANN). The objective was to evaluate the performance of each algorithm and determine the best model to be implemented in the proposed system.

Initially, the experiments were conducted independently for each algorithm, with diverse results obtained to assess the accuracy of the classifiers. The results of each experiment were compared to determine the optimal parameter values that would enhance the accuracy of the classifier. Thereafter, the highest accuracies for each algorithm were compared against each other to select the best overall model for the proposed system.

All experiments were carried out using the same computer, which was equipped with Windows 10 64-bit operating system, x64-based processor, Intel(R) Core (TM) i7-4700MQ CPU @2.40GHz, and 8.00 GB of RAM. To facilitate the implementation of the experiments, popular Python programming language packages such as Scikit-Learn, Pandas, and NumPy were used. Furthermore, Table 6 outlines the parameters and their respective values employed for the DT, KNN, and ANN classifiers in each conducted experiment.

Table 6: Classification algorithms and their utilized parameters.

Classifier	Parameter	Utilized values	Description
DT	criterion	'gini', and 'entropy'	Measurement function to calculate the quality of the feature.
	splitter	'best', and 'random'	A strategy used to split the tree in each node.
	max_depth	Integer number from 3 to 12	The depth of the tree. Represents the number of levels in the tree.
KNN	n_neibors	Integer number from 1 to 10	The number of nearest neighbors helps in deciding which class the new data point belongs to.
	p	1, 2, and 4	Representing distance metric. It is the power value for the Minkowski distance metric that determines the utilized metrics among Manhattan, Euclidean, or Minkowski.
MLP	hidden_layer_sizes	Integer number from 6 to 10	Number of hidden layers (Only one hidden layer utilized in this study) with the number of neurons in each layer.
	activation	'identity', 'logistic', 'tanh', and 'relu'	It is the activation function which calculate the importance of the neurons to decide if the neuron is active or not.
	solver	'sgd,' and 'adam'	Optimizer to use for weights update.
	learning_rate	'Constant'	This parameter is uset to determine if the initial learning rate will be changed or be constant. in this study, 'constant' is used. used only when solver='sgd'.
	learning_rate_init	0.1, 0.01, 0.001	This is the actual learning rate to use as the initial value of the learning rate for weight updates. It controls the step-size in updating the weights. Only used when solver='adam' or 'sgd'.
	max_iter	700	Representing epoch, the max number of times that weights will be updated for each data instance.

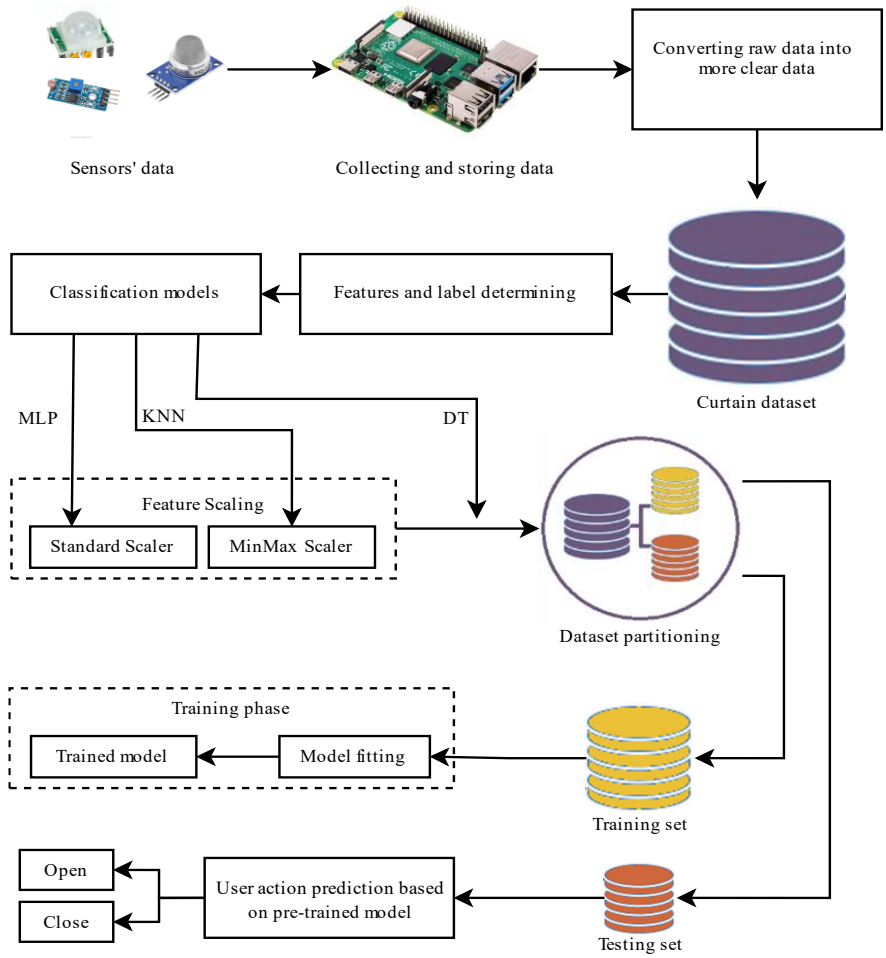


Figure 8: Workflow of the proposed system incorporating with ML.

5.1 Experiment #1: DT Classifier Performance

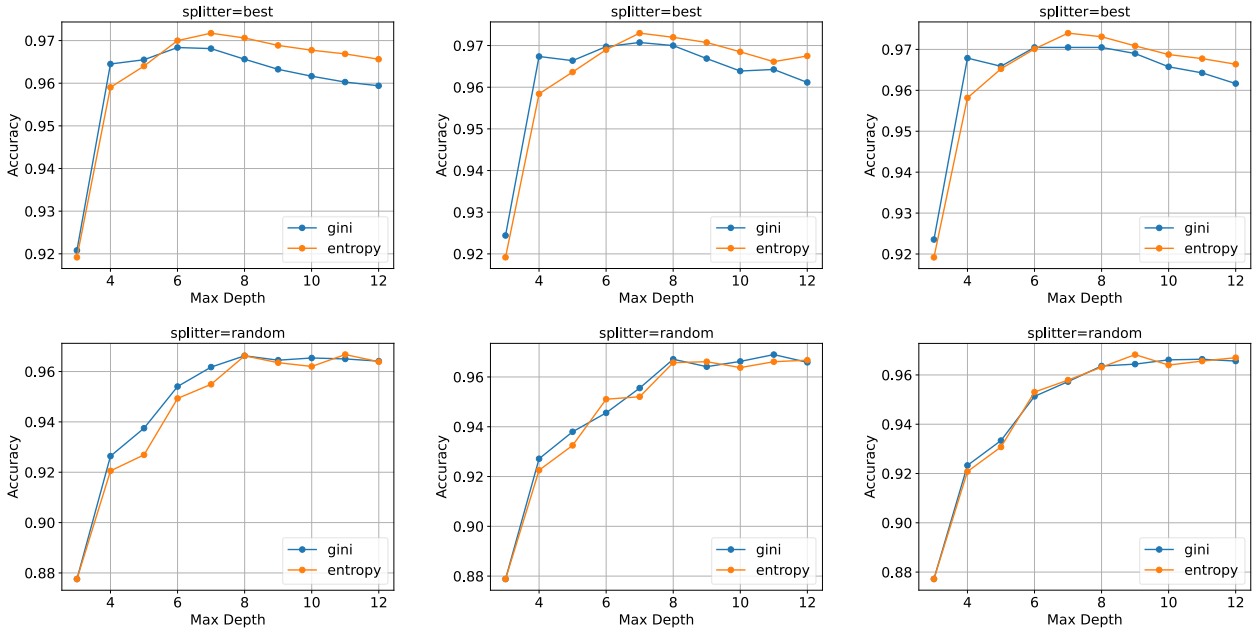
In the context of the experiments conducted in this study, the DT classifier was utilized to determine the optimal model with the best parameter values. Several tests were performed on different models, and the accuracy scores obtained from the DT models were demonstrated using K-Fold cross-validation with 3, 5, and 10 folds. The results were summarized in Figure 9. The best results of the three K-Fold values were compared and presented in Table 7 and Figure 10. The findings indicate that the highest accuracy was achieved when the best splitter was performed on the tree. Moreover, the other parameters, including criterion and max_depth, were set to entropy and 7, respectively. The results demonstrated that the model with 10 folds provided the best accuracy score of 97.4% and F1-score of 97.16%, outperforming the other values of 3 and 5 folds. Therefore, the accuracy was found to increase with the increase in the number of folds used in the experiment. It should be noted that no normalization techniques were applied to the dataset during the pre-processing phase. This decision was made since DT is not a distance-based algorithm, and therefore, normalization does not significantly impact its performance.

The experiments performed in this study involved testing the DT classifier with different values for several parameters, as illustrated in Figure 9. The findings indicate that when the splitter parameter was set to "best", the criterion "gini" produced higher accuracy scores than the criterion "entropy" when the maximum depth of the tree was less than or equal to six. However, when the maximum depth was greater than six, the criterion "entropy" performed better. On the other hand, when the random splitter was employed, the results of both criteria ("entropy" and "gini") exhibited an irregular pattern and did not follow a clear trend. These findings highlight the importance of selecting appropriate parameter values to achieve the best performance of the DT classifier.

Table 7: DT results for the best models selected based on different K-Fold value (3, 5, and 10).

K-Fold	Criterion	Max_depth	Splitter	Accuracy	F1-score
3	Entropy	7	Best	97.17	96.93
5	Entropy	7	Best	97.30	97.05
10	Entropy	7	Best	97.40	97.16

To validate the performance of the best models obtained from each K-Fold value (3, 5, and 10), the ROC curves were plotted and AUCs were calculated, as shown in Figure 11. The AUC values presented in each figure correspond to the AUC result for each independent test split (fold). For example, the graph illustrating the results of the selected best model for K-Fold when assigned to 3 displays three AUC values. Similarly, the graph with five AUC values represents the model for K-Fold = 5, and so on. Furthermore, to ensure the reliability of obtained results, the training accuracy against the testing accuracy for each fold were compared, as shown in Figure 12. These analyses provide insights into the generalization performance of the models and demonstrate how well they can perform on new, unseen data. The results suggest that the models trained with the best parameters achieve high accuracy scores and can effectively predict user behavior regarding the opening and closing of curtains.

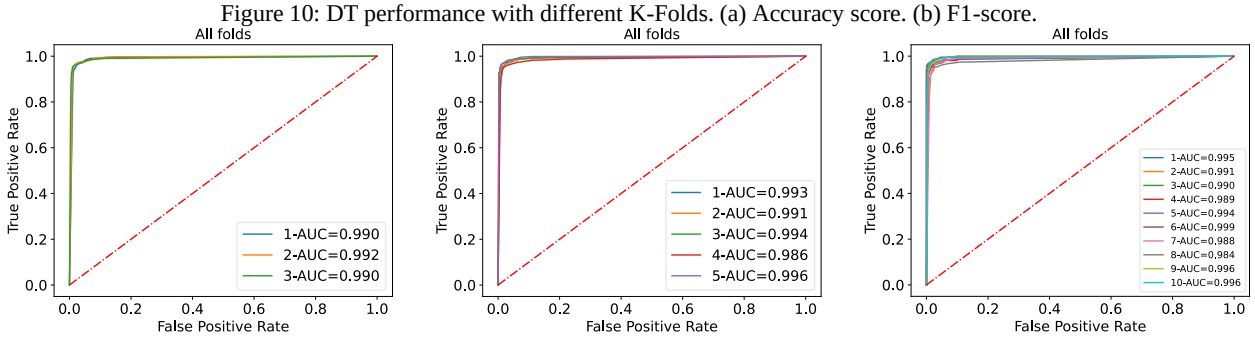


(a) (b) (c)

Figure 9: Results of DT classifier when: (a) K-Fold=3. (b) K-Fold=5. (c) K-Fold=10.

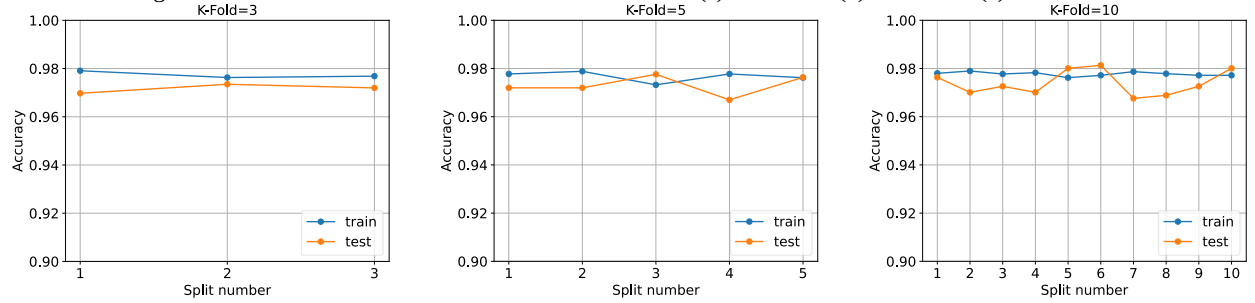
(a) (b)

Figure 10: DT performance with different K-Folds. (a) Accuracy score. (b) F1-score.



(a) (b) (c)

Figure 11: ROC curve with AUC results for DT classifier. (a) K-Fold=3. (b) K-Fold=5. (c) K-Fold=10.



(a) (b) (c)

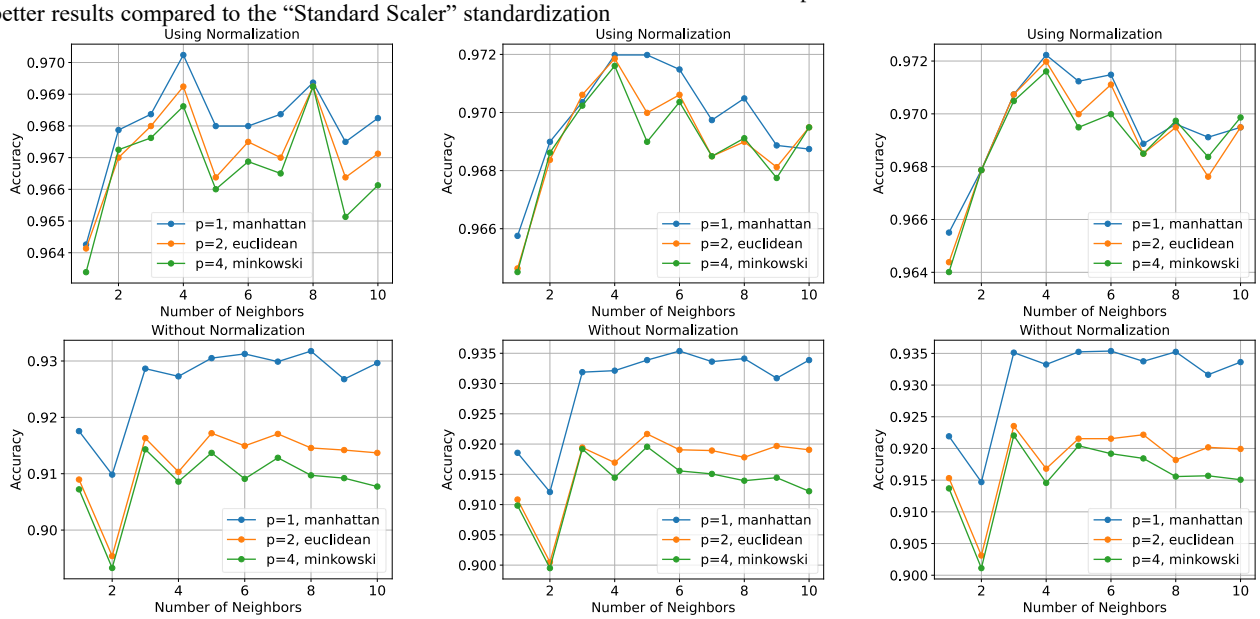
Figure 12: Model performance for DT classifier (Train vs Test accuracy). (a) K-Fold=3. (b) K-Fold=5. (c) K-Fold=10.

The trained models exhibit high reliability, as evidenced by their ability to achieve a high true positive rate of over 98%, as shown in Figure 11. Furthermore, Figure 12 indicates that the train and test accuracies were consistently high and that their values were very similar for each split, implying that there was no evidence of overfitting or underfitting in the trained models. This indicates

that the models can accurately generalize to new, unseen data, which is essential for their practical implementation in real-world scenarios. Overall, these findings demonstrate the efficacy of the selected classifiers and parameter values in predicting user behavior regarding the opening and closing of curtains.

5.2 Experiment #2: KNN Classifier Performance

This experiment investigates the performance of the KNN algorithm as a classifier to predict the user's action. Given that KNN is a distance-based algorithm, feature scaling techniques, particularly Normalization, can have a significant influence on its performance. To address this issue, the “MinMax Scaler” normalization method was employed, which has shown to yield better results compared to the “Standard Scaler” standardization method. To ensure a fair comparison among the utilized classifiers, the K-Fold cross-validation was used with the same values of 3, 5, and 10 folds, as previously done in Experiment #1. For each of these three K-Fold values, this experiment conducted tests with the KNN classifier and reported the parameter values and results obtained for each model with and without normalization. The results, as shown in Figure 13, indicate that the performance of the KNN classifier significantly improves with the use of normalization. This improvement is particularly evident in cases where the number of folds is higher (i.e., 5 and 10 folds). Therefore, the use of the "MinMax Scaler" normalization technique was recommended when applying KNN for user action prediction.



(a) (b) (c)

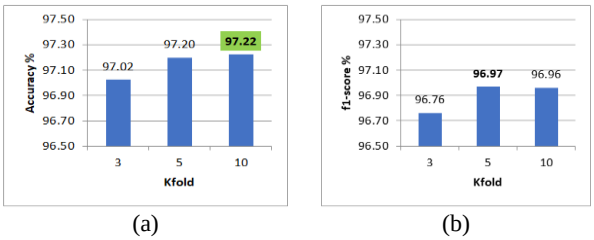
Figure 13: KNN performance results when: (a) K-Fold=3. (b) K-Fold=5. (c) K-Fold=10.

Figure 13 provides a visual representation of the parameters used in these experiments and their impact on the results. these findings suggest that a value of "p" equal to one, which corresponds to the Manhattan distance metric, yields the highest accuracies in most cases, irrespective of the number of neighbors used. Additionally, the obtained results indicate that the accuracy of the KNN classifier improves significantly when using normalized data compared to raw data. Moreover, the Euclidean distance metric outperforms the Minkowski metric in the conducted experiments. These findings highlight the importance of carefully selecting the appropriate distance metric for achieving optimal performance when using KNN as a classifier.

The experimental results presented in the three subfigures of Figure 13 are summarized in Table 8 and visualized in Figure 14, where the best model that achieved the highest accuracy among all tests is selected. According to Table 8, the highest accuracy is achieved using a 10-fold test with four neighbors and the Manhattan distance metric, where the value of "p" is set to one. On the other hand, the best model in terms of the F1-score evaluation metric is obtained using five neighbors, the Manhattan distance metric, and a five-fold test. To further validate the selected best models, this experiment presents the ROC curves in Figure 15 and plot the training accuracy against the testing accuracy in Figure 16, using the same procedures as in Experiment #1. These analyses demonstrate that the selected models provide high accuracy and robustness, thus confirming their suitability for user action prediction using the KNN classifier.

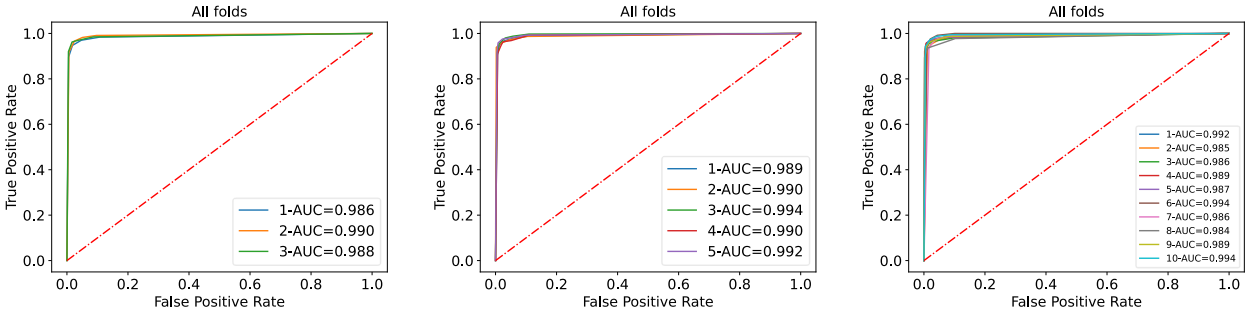
Table 8: KNN results for the best models selected based on different K-Fold values (3, 5, and 10).

K-Fold	n_neighbors	P (Distance metric)	Accuracy	F1-score
3	4	1	97.02	96.76
5	5	1	97.20	96.97
10	4	1	97.22	96.96



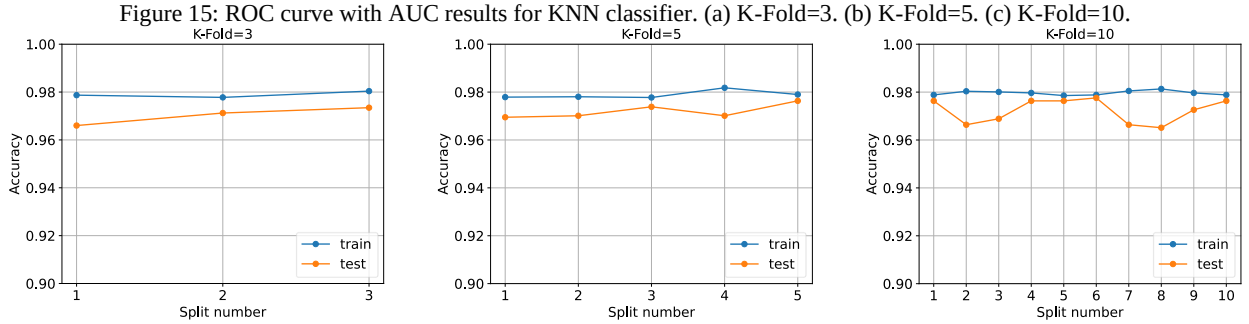
(a) (b)

Figure 14: KNN performance with different K-Folds. (a) Accuracy score (b) F1-score.



(a) (b) (c)

Figure 15: ROC curve with AUC results for KNN classifier. (a) K-Fold=3. (b) K-Fold=5. (c) K-Fold=10.



(a) (b) (c)

Figure 16: Model performance for KNN classifier (Train vs Test accuracy). (a) K-Fold=3. (b) K-Fold=5. (c) K-Fold=10.

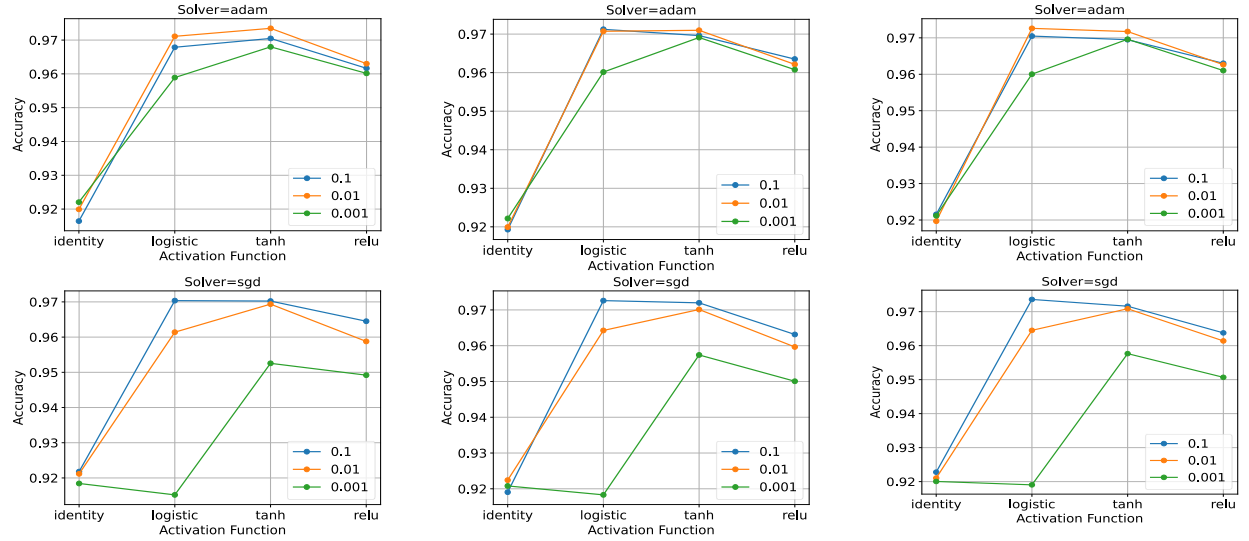
The ROC curves displayed in Figure 15 indicate that all models achieved a high percentage of the true positive rate, which suggests that they are highly reliable for user action prediction. The high AUC rates observed in the ROC curves further confirm the robustness and accuracy of the selected models. Furthermore, Figure 16 presents the plot of training accuracy against testing accuracy for each split, which provides insights into the performance of the selected models in terms of overfitting and underfitting. It can be observed that the models were well-trained, as they do not exhibit significant overfitting or underfitting, and the train and test accuracies in each split are very close to each other. This suggests that the selected models generalize well to unseen data, which is an essential property for any machine learning model to be effective in real-world applications.

5.3 Experiment #3: MLP Classifier Performance

Another experiment employing MLP classification algorithm was conducted over “Curtain” dataset, which yielded obtaining high accuracies. Similar to the previous experiments, K-Fold cross-validation was used to ensure a fair comparison among the different models. Therefore, different results were obtained from several models represented in Figure 17. Unlike the KNN which employed MinMax Scaler as the feature scaler, the MLP classifier makes use of Standard Scaler standardization. Thus, this scaler led to better results and helped the model to converge faster. Indeed, it can be observed that the MLP provided very low results when the feature scaling technique was not applied.

The results indicate that the best-performing MLP model achieved an accuracy score of 97.36% and an F1-score of 97.13% when K-Fold was set to 10, and the following parameters were assigned: "logistic" activation function, one hidden layer with six nodes, a learning rate of 0.1, and "sgd" solver (optimizer). Additionally, the max_iter parameter, which determines the number of epochs necessary to train the model, was set to 700. The model was observed to converge entirely at this value after testing various max_iter values, ranging from 100 to 700 in increments of 100. The loss function used was "cross_entropy," which is the only option available in the MLP classifier in the Sklearn package.

These findings are summarized in Table 9, which highlights the optimal model for each K-Fold value. Additionally, Figure 18 illustrates the performance of the best model configurations across all K-Fold values.



(a) (b) (c)

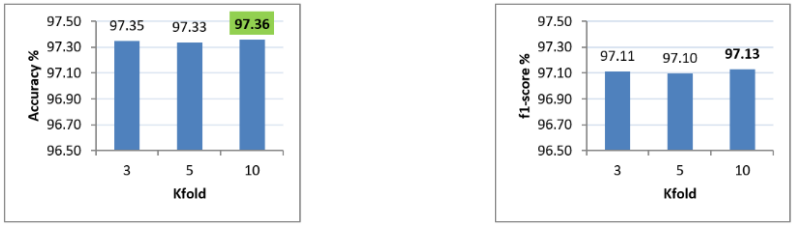
Figure 17: MLP performance with different K-Folds. (a) K-Fold=3. (b) K-Fold=5. (c) K-Fold=10.

Figure 17 (a) demonstrates that when the solver=”adam” the best results obtained are when the value of the learning rate equals to 0.01, and “tanh” as an activation function. While for the solver “sgd”, the highest results obtained with the learning rate 0.1 and the activation function is assigned to “logistic”. From this figure, it can be observed that the solver “adam” provides higher results than the “sgd”.

It can be seen in Figure 17 (b) that the best results obtained were with activation function “logistic” as well as when the value of the learning rate assigned to 0.1. Furthermore, the learning rate with the value 0.001 performs the lowest results in the most cases as compared with the other learning rate values (0.1, and 0.01). As visualized in Figure 17 (c), the activation function “logistic” performs the highest results among all other activation function. Moreover, the learning rate 0.01 outperform the other learning rate when the “adam” optimizer is used. While the learning rate of 0.1 outperforms the other learning rate utilized in this experiment when associated with “sgd” solver. It can be observed that from the recently three mentioned figures, the lowest results gained when the “identity” activation function is utilized.

Table 9: MLP results for the best models selected based on different K-Fold values (3, 5, and 10)

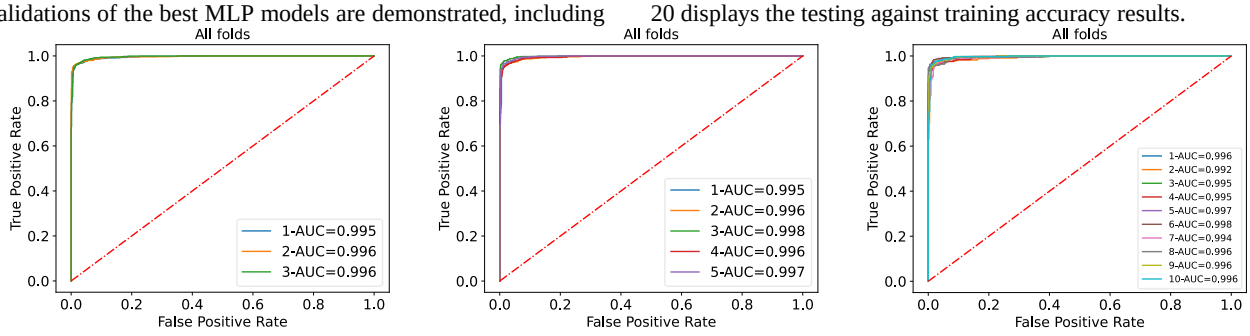
K-Fold	Activation function	Hidden layer sizes	Learning rate	Solver	Accur-acy	F1-score
3	tanh	6	0.01	adam	97.35	97.11
5	tanh	9	0.1	sgd	97.33	97.10
10	logistic	6	0.1	sgd	97.36	97.13



(a) (b)

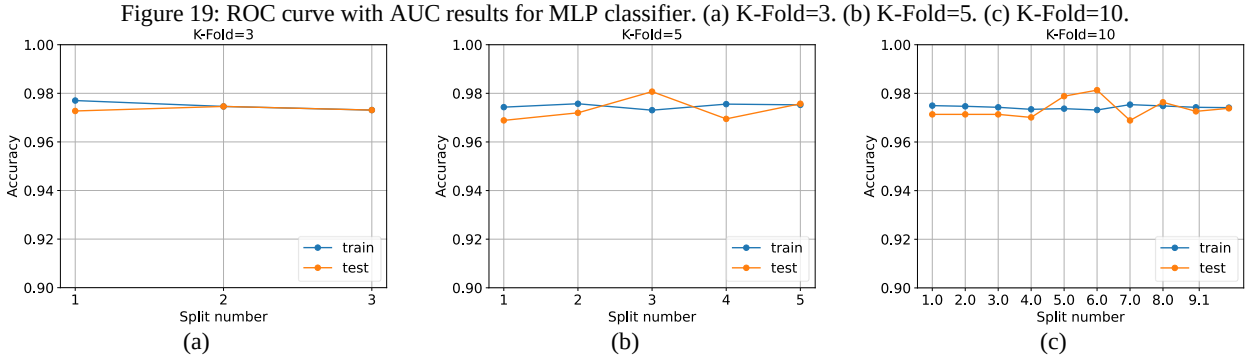
Figure 18: Performance with different K-Folds. (a) Accuracy score. (b) F1-score.

Figure 18, illustrates that the highest results obtained when the number of folds assigned to 10. In the next figures, the validations of the best MLP models are demonstrated, including those with 3, 5, and 10 as K-Fold values. In Figure 19, the ROC curves associated with AUC rates were presented, while Figure 20 displays the testing against training accuracy results.



(a) (b) (c)

Figure 19: ROC curve with AUC results for MLP classifier. (a) K-Fold=3. (b) K-Fold=5. (c) K-Fold=10.



(a) (b) (c)

Figure 20: Model performance for MLP classifier (Train vs Test accuracy). (a) K-Fold=3. (b) K-Fold=5. (c) K-Fold=10.

The performance of the models was assessed by analyzing true positive rates across different test splits, as shown in Figure 19. The results indicate that the models were highly reliable, achieving true positive rates of more than 99% for each test split. Furthermore, Figure 20 illustrates that the models were appropriately trained, with neither overfitting nor underfitting observed. The train and test accuracies were found to be extremely close to one another across all splits, providing further evidence of the models' robustness and generalizability. These findings suggest that the models developed in this study can be considered highly reliable and adequately trained for use in real-world scenarios.

Table 10: A comparison among the best results for each classifier with their training time.

K-Fold	Classifier	Accuracy	F1-score	Average training time in milliseconds
10	DT	97.40	97.16	5.45
10	KNN	97.22	96.96	11.25
10	MLP	97.36	97.13	3157.6 (4.51 per epoch)

Upon analyzing the results of the three experiments conducted on the "Curtain" dataset, it can be concluded that the DT classifier demonstrated superior performance compared to the other classifiers in terms of accuracy and training time (as illustrated in Table 10). This can be attributed to the fact that the "Curtain" dataset is composed solely of numerical values, which are well-suited for decision tree-based algorithms.

To obtain the training times for the classifiers' best models, the average time in milliseconds was calculated by averaging the results of four different executions of the same model. As a result, the average training times for the DT, KNN, and MLP classifiers were found to be 5.45, 11.25, and 3157.6 milliseconds (4.51 per epoch), respectively. Based on these findings, it is recommended that the DT model be implemented in the system due to its superior performance in terms of accuracy and training time.

5.4 Comparison of the proposed system with the state-of-the-art systems

Table 11 presents a comparative analysis of the proposed system against several reviewed systems in the literature.

Table 11: Summary of proposed system versus reviewed works, where MLu: Machine Learning is utilized, RPi: Raspberry Pi used, oRPi: only the RPi utilized as a central controller, DPS: Dataset Provided by the system, Adp: the system is adaptive to the user behavior, SP: security purpose in the system, AM: does any appliances controlled automatically, MM: does any appliances controlled manually, MF: Muli-Featured, WUI: Web-based User Interface.

Ref.	MLu	RPi	oRPi	DPS	Adp	SP	AM	MM	MF	WUI
(Okorie et al., 2020)	×	×	×	×	×	×	×	✓	✓	×
(Iqbal et al., 2018)	×	✓	×	×	×	✓	✓	✓	✓	✓
(Pavithra & Balakrishnan, 2015)	×	✓	✓	×	×	✓	✓	✓	✓	✓
(Gota et al., 2020)	×	×	×	×	×	×	✓	✓	✓	✓
(Abbas & Abdullah, 2021)	✓	✓	×	✓	✓	×	✓	✓	×	×
(Mehmood et al., 2019)	✓	✓	×	×	×	✓	✓	×	×	✓
(Crisnapati et al., 2016)	✓	✓	×	×	×	✓	✓	✓	✓	✓
(Paredes-Valverde et al., 2020)	✓	×	×	✓	×	×	×	✓	✓	×
(Peng et al., 2019)	✓	×	×	×	×	×	×	✓	×	×
(Raju et al., 2021)	✓	✓	✓	×	×	×	✓	✓	✓	×
Proposed system	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

The proposed system outperforms the reviewed systems in various aspects. Firstly, the proposed system excels having a web-based user interface (UI) that enables local and internet-based system access surpassing the systems reviewed in (Abbas & Abdullah, 2021; Okorie et al., 2020; Paredes-Valverde et al., 2020; Peng et al., 2019; Raju et al., 2021). Secondly, the proposed system stands out for its ability to use only a RPi as a central controller, outperforming all reviewed systems except for the ones in (Pavithra & Balakrishnan, 2015; Peng et al., 2019). Thirdly, this system has a unique ability to adapt to user behavior by leveraging the dataset it creates, a feature missing in all reviewed systems except for the one in (Abbas & Abdullah, 2021) which does not take into consideration the environmental conditions. Additionally, the proposed system outperforms the reviewed system in terms of using ML techniques to enhance home intelligence, surpassing the systems reviewed in (Gota et al., 2020; Iqbal et al., 2018; Okorie et al., 2020; Pavithra & Balakrishnan, 2015). Moreover, this system provides a variety of ways for users to interact with the system, including features related to security, energy-saving, and more, outperforming all reviewed works. Furthermore, this system provides real-time voice and text alerts to users via the UI and email to improve home security and safety. Finally, the proposed system provides a fully automated home, utilizing several sensors and appliances, and enhances home intelligence by applying automatic decision-making on behalf of the user.

6. CONCLUSION AND FUTURE WORKS

Smart homes using Machine Learning (ML) offer significant benefits over traditional smart homes by incorporating the power of artificial intelligence. By leveraging ML algorithms, smart homes can automatically adjust settings and operations based on learned patterns and preferences, providing a more personalized and efficient experience for residents. One of the key applications of ML in smart homes is in the area of home automation. For example, ML algorithms can be used to control lighting, heating, cooling, and security systems based on the presence and behavior of residents.

This study proposes a smart home system that aims to enhance security, comfort, energy efficiency, and intelligence. The system comprises three primary modes, including the manual mode, automatic mode, and intelligent mode. The manual mode allows the user to control different appliances and monitor the home environment through a web-based application. The automatic mode, on the other hand, reduces human intervention and maximizes security. This mode provides real-time alerts through the UI in both text and voice form and also sends text alerts via email in the event of an intruder or fire detection. Additionally, this mode controls appliances automatically based on sensor readings to minimize energy consumption. The intelligent mode employs ML algorithms to predict and track user behavior patterns related to opening and closing curtains. Several experiments were conducted to compare the performance of various classifiers (DT, KNN, and MLP) applied to the "Curtain" dataset. The results revealed that DT provided the highest accuracy (97.4%) with the shortest average training time (5.45 milliseconds) among all the implemented classifiers, making it the best choice for implementation in the proposed smart home system.

For the future goal it has been aimed to enable the user to interact with the system using voice commands. Additionally, it has been planned to incorporate image processing capabilities into the system to automatically control the main door based on the image of the authorized person. Furthermore, for the intelligent mode, it has been aimed to create another dataset for user action for predicting user preferences using regression-based models. In addition, applying various feature selection techniques on the "Curtain" dataset in order to evaluate their impact on the performance of the utilized models.

REFERENCES

Abbas, A. F., & Abdullah, M. Z. (2021). Design and Implementation of Tracking a user's Behavior in a Smart Home. *IOP Conference Series: Materials Science and Engineering*, *1094*(1), 012008. https://doi.org/10.1088/1757-899x/1094/1/012008

Bertolini, M., Mezzogori, D., Neroni, M., & Zammori, F. (2021). Machine Learning for industrial applications: A comprehensive literature review. *Expert Systems with Applications*, *175*(February), 114820. https://doi.org/10.1016/j.eswa.2021.114820

Car, Z., Baressi Šegota, S., Anđelić, N., Lorencin, I., & Mrzljak, V. (2020). Modeling the Spread of COVID-19 Infection Using a Multilayer Perceptron. *Computational and Mathematical Methods in Medicine*, *2020*. https://doi.org/10.1155/2020/5714714

Choi, W., Kim, J., Lee, S. E., & Park, E. (2021). Smart home and internet of things: A bibliometic study. *Journal of Cleaner Production*, *301*, 126908. https://doi.org/10.1016/j.jclepro.2021.126908

Crisnapati, P. N., Wardana, I. N. K., & Aryanto, I. K. A. A. (2016). Rudas: Energy and sensor devices management system in home automation. *Proceedings - 2016 IEEE Region 10 Symposium, TENSYMP 2016*, 184–187. https://doi.org/10.1109/TENCONSpring.2016.7519401

Desai, M., & Shah, M. (2021). An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (MLP) and Convolutional neural network (CNN). *Clinical EHealth*, *4*, 1–11. https://doi.org/10.1016/j.ceh.2020.11.002

Gao, X., & Li, G. (2020). A KNN Model Based on Manhattan Distance to Identify the SNARE Proteins. *IEEE Access*, *8*, 112922–112931. https://doi.org/10.1109/ACCESS.2020.3003086

Géron, A. (2019). Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. In *O'Reilly Media, Inc* (Second Edi). O'Reilly Media, Inc.

Gota, D. I., Puscasiu, A., Fanca, A., Miclea, L., & Valean, H. (2020). Smart home automation system using Arduino microcontrollers. *2020 22nd IEEE International Conference on Automation, Quality and Testing, Robotics - THETA, AQTR 2020 - Proceedings*. https://doi.org/10.1109/AQTR49680.2020.9129989

Heidari, A. A., Faris, H., Mirjalili, S., Aljarah, I., & Mafarja, M. (2020). Ant lion optimizer: Theory, literature review, and application in multi-layer perceptron neural networks. *Studies in Computational Intelligence*, *811*, 23–46. https://doi.org/10.1007/978-3-030-12127-3_3

Ibrahim, A. K., Hassan, M. M., & Ali, I. A. (2022). Smart Homes for Disabled People: A Review Study. *Science Journal of University of Zakho*, *10*(4), 213–221. https://doi.org/10.25271/sjuoz.2022.10.4.1038

Iqbal, A., Ullah, F., Anwar, H., Kwak, K. S., Imran, M., Jamal, W., & Rahman, A. ur. (2018). Interoperable Internet-of-Things platform for smart home system using Web-of-Objects and cloud. *Sustainable Cities and Society*, *38*, 636–646. https://doi.org/10.1016/j.scs.2018.01.044

Jabbar, W. A., Alsibai, M. H., Amran, N. S. S., & Mahayadin, S. K. (2018). Design and Implementation of IoT-Based Automation System for Smart Home. *2018 International Symposium on Networks, Computers and Communications, ISNCC 2018, November 2018*, 1–6. https://doi.org/10.1109/ISNCC.2018.8531006

Jabbar, W. A., Kian, T. K., Ramli, R. M., Zubir, S. N., Zamrizaman, N. S. M., Balfaqih, M., Shepelev, V., & Alharbi, S. (2019). Design and Fabrication of Smart Home with Internet of Things Enabled Automation System. *IEEE Access*, *7*, 144059–144074. https://doi.org/10.1109/ACCESS.2019.2942846

Jolles, J. W. (2021). Broad-scale applications of the Raspberry Pi: A review and guide for biologists. *Methods in Ecology and Evolution*, *12*(9), 1562–1579. https://doi.org/10.1111/2041-210X.13652

Jung, A. (2022). *Machine Learning: Foundations, Methodologies, and Applications*. Springer Nature.

Klok, H., & Nazarathy, Y. (2021). *Statistics with Julia: Fundamentals for Data Science, Machine Learning and Artificial Intelligence*. Springer Nature. https://statisticswithjulia.org

Kubat, M., & An. (2021). An Introduction to Machine Learning. In *Springer Nature* (Third Edit). Springer Nature. https://doi.org/10.1002/9781119720492.ch7

Kurniawan, A. (2019). Introduction to Raspberry Pi. In *Raspbian OS Programming with the Raspberry Pi* (pp. 1–25). Apress, Berkeley, CA. https://doi.org/https://doi.org/10.1007/978-1-4842-4212-4_1

Li, J., Gao, F., Lin, S., Guo, M., Li, Y., Liu, H., Qin, S., & Wen, Q. (2023). Quantum K-fold cross-validation for nearest neighbor classification algorithm. *Physica A: Statistical Mechanics and Its Applications*, *611*, 128435. https://doi.org/10.1016/j.physa.2022.128435

Mahmoodzadeh, A., Mohammadi, M., Daraei, A., Farid Hama Ali, H., Kameran Al-Salihi, N., & Mohammed Dler Omer, R. (2020). Forecasting maximum surface settlement caused by urban tunneling. *Automation in Construction*, *120*(July), 103375. https://doi.org/10.1016/j.autcon.2020.103375

Marikyan, D., Papagiannidis, S., & Alamanos, E. (2019). A systematic review of the smart home literature: A user perspective. *Technological Forecasting and Social Change*, *138*(November 2017), 139–154. https://doi.org/10.1016/j.techfore.2018.08.015

Mehmood, F., Ullah, I., Ahmad, S., & Kim, D. H. (2019). Object detection mechanism based on deep learning algorithm using embedded IoT devices for smart home appliances control in CoT. *Journal of Ambient Intelligence and Humanized Computing*, *0123456789*. https://doi.org/10.1007/s12652-019-01272-8

Mienye, I. D., Sun, Y., & Wang, Z. (2019). Prediction performance of improved decision tree-based algorithms: A review. *Procedia Manufacturing*, *35*, 698–703. https://doi.org/10.1016/j.promfg.2019.06.011

Mukherjee, A., Mondal, S., Chaki, N., & Khatua, S. (2019). Naive bayes and decision tree classifier for streaming data using hbase. In *Advances in Intelligent Systems and Computing* (Vol. 897). Springer Singapore. https://doi.org/10.1007/978-981-13-3250-0_8

Nikou, S. (2019). Factors driving the adoption of smart home technology: An empirical assessment. *Telematics and Informatics*, *45*(September), 101283. https://doi.org/10.1016/j.tele.2019.101283

Okorie, P. U., Ibraim, A. A., & Auwal, D. (2020). Design and Implementation of an Arduino Based Smart Home. *HORA 2020 - 2nd International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Proceedings, October 2012*. https://doi.org/10.1109/HORA49412.2020.9152922

Pajankar, A. (2021). Introduction to Raspberry Pi. In *Practical Linux with Raspberry Pi OS*. Apress, Berkeley, CA. https://doi.org/https://doi.org/10.1007/978-1-4842-6510-9_1

Paredes-Valverde, M. A., Alor-Hernández, G., García-Alcaráz, J. L., Salas-Zárate, M. del P., Colombo-Mendoza, L. O., & Sánchez-Cervantes, J. L. (2020). IntelliHome: An internet of things-based system for electrical energy

saving in smart home environment. *Computational Intelligence*, *36*(1), 203–224. https://doi.org/10.1111/coin.12252

Patel, H. H., & Prajapati, P. (2018). Study and Analysis of Decision Tree Based Classification Algorithms. *International Journal of Computer Sciences and Engineering*.

Pavithra, D., & Balakrishnan, R. (2015). IoT based monitoring and control system for home automation. *2015 Global Conference on Communication Technologies (GCCT)*. https://doi.org/10.1109/GCCT.2015.7342646

Peng, Y., Peng, J., Li, J., & Yu, L. (2019). Smart Home System Based on Deep Learning Algorithm. *Journal of Physics: Conference Series*, *1187*(3). https://doi.org/10.1088/1742-6596/1187/3/032086

Raju, L., Sowmya, G., Srividhya, S., Surabhi, S., Retika, M. K., & Reshmika Janani, M. (2021). Advanced home automation using raspberry pi and machine learning. *Proceedings of the 7th International Conference on Electrical Energy Systems, ICEES 2021*, 600–605. https://doi.org/10.1109/ICEES51510.2021.9383738

Raspberry Pi Documentation. (n.d.). Retrieved December 14, 2022, from https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#gpio-and-the-40-pin-header

Saleem, A. A., Hassan, M. M., & Ali, I. A. (2022). Smart Homes Powered by Machine Learning: A Review. *Proceedings of the 2nd 2022 International Conference on Computer Science and Software Engineering, CSASE 2022*, 355–361. https://doi.org/10.1109/CSASE51777.2022.9759682

Sarker, I. H. (2021a). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, *2*(3). https://doi.org/10.1007/s42979-021-00592-x

Sarker, I. H. (2021b). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, *2*(3), 1–21. https://doi.org/10.1007/s42979-021-00592-x

Singh, P. K., Kar, A. K., Singh, Y., Kolekar, M. H., & Tanwar, S. (2020). Proceedings of ICRIC 2019: Recent Innovations in Computing. In P. K. Singh, A. K. Kar, Y. Singh, M. H. Kolekar, & S. Tanwar (Eds.), *Lecture Notes in Electrical Engineering* (First Edit). Springer Nature. https://doi.org/https://doi.org/10.1007/978-3-030-29407-6

Sunrise and sunset Zakho Dahuk Iraq. (n.d.). Retrieved January 7, 2022, from https://www.weatheravenue.com/en/asia/iq/dahuk/zakho-sunrise.html

Taiwo, O., Ezugwu, A. E., Oyelade, O. N., & Almutairi, M. S. (2022). Enhanced Intelligent Smart Home Control and Security System Based on Deep Learning Model. *Wireless Communications and Mobile Computing*, *2022*. https://doi.org/10.1155/2022/9307961

Tatsat, H., Puri, S., & Lookabaugh, B. (2020). Machine Learning and Data Science Blueprints for Finance. In *O'Reilly Media* (First Edit). O'Reilly Media, Inc.

van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*(2), 373–440. https://doi.org/10.1007/s10994-019-05855-6