Integrating Machine Learning Based Wake Time Prediction into Smart Home Device Automation

Mrudul A Thakadiyel
Department of Computer Applications
Amal Jyothi College of Engineering
Kanjirappally, Kottayam, Kerala, India
mrudulathakadiyel@mca.ajce.in

Jinson Devis
Assistant Professor, Department of Computer Applications
Amal Jyothi College of Engineering
Kanjirappally, Kottayam, Kerala, India
jinsondevis@amaljyothi.ac.in

Abstract—A smart home's entire purpose is to make life at home a little bit easier. This feature has traditionally been enabled by users manually establishing connections between devices and actions via automation rules. As a result, humans have completely or mostly predetermined the intelligence of these systems. We can use Artificial Intelligence techniques in this field to try to improve the smartness of Smart Homes to address this challenge. Automating your daily routine is a great place to start with smart home technology. This concept is centered on applying machine learning techniques to automate smart home devices based on waking time prediction.

Keywords—Machine learning, linear regression, home assistant, wake time prediction

I. INTRODUCTION

Various morning routine tasks can be automated. Turning on the light, gradually increasing the temperature, making coffee automatically, receiving the morning news, and so on are only a few examples.

Despite the fact that home automation systems exist today, the great majority of them are unable to adapt to human sleep habits. This research offers a system that forecasts a user's awake time using machine learning techniques such as linear regression and data acquired from smart wearable devices. The information is then utilized to train a model that predicts future values. The smart home devices are then triggered based on this predicted value. Home Assistant, a free open-source tool, is utilized for this.

The study is divided into two parts: the first estimates the wake time using the provided data, and the second uses the predicted value to start smart home device automation.

II. WAKE TIME PREDICTION

For this study Multiple Linear Regression (MLR) model is being used for forecasting Wake time. MLR is a research method that examines the link between one or more independent variables and a single dependent variable.

A. Data Collection and Preparation

Dataset for the analysis can be acquired from smart wearable devices that supports sleep monitoring features. Before using the data to train a model, it had to be prepared.

TABLE I. UNPREPARED DATASET

	DATE	DAYNO	WAKETIME
0	01-11-2021	1	8.15
1	02-11-2021	2	8.22
2	03-11-2021	3	8.26
3	04-11-2021	4	8.12
4	05-11-2021	5	8.29
5	06-11-2021	6	9.22
6	07-11-2021	7	9.19
7	08-11-2021	1	8.15
8	09-11-2021	2	8.22
9	10-11-2021	3	8.26
10	11-11-2021	4	8.18
11	12-11-2021	5	8.19
12	13-11-2021	6	9.24
13	14-11-2021	7	9.33
14	15-11-2021	1	8.15
15	16-11-2021	2	8.17
16	17-11-2021	3	8.26
17	18-11-2021	4	8.12
18	19-11-2021	5	8.29
19	20-11-2021	6	9.32
20	21-11-2021	7	9.27
21	22-11-2021	1	8.25
22	23-11-2021	2	8.16
23	24-11-2021	3	8.22
24	25-11-2021	4	8.19
25	26-11-2021	5	8.21
26	27-11-2021	6	9.22
27	28-11-2021	7	9.12
28	29-11-2021	1	NaN
29	30-11-2021	2	NaN
30	01-12-2021	3	NaN

DOI: 10.5281/zenodo.6180322
ISBN: 978-93-5607-317-3 @2022 MCA, Amal Jyothi College of Engineering Kanjirappally, Kottayam

31	02-12-2021	4	NaN
32	03-12-2021	5	NaN
33	04-12-2021	6	NaN
34	05-12-2021	7	NaN

	WakeTime	t_dayno	t_dayno2	t_dayno3	day_2	day_3	day_4	day_5	day_6	day_7
0	8.15	1	1	1	0	0	0	0	0	0
1	8.22	2	4	8	1	0	0	0	0	0
2	8.26	3	9	27	0	1	0	0	0	0
3	8.12	4	16	64	0	0	1	0	0	0
4	8.29	5	25	125	0	0	0	1	0	0

In the Multiple Regression Model, we use a lot of categorical data. Categorical Variables can be used to include non-numeric data in the relevant Regression Model. Categorical data refers to data values that identify categories. Dummy Variables can be used to represent these values in the regression model.

Dayno's default data type will be numeric in this case. We must set the model type to nominal over here, else the dummy variables will not be generated automatically. So, we add the dummy variables t_dayno, t_dayno2, and t_dayno3, where t_dayno is the linear time, and then we square and cube these values to get time squared(t_dayno2) and time cube(t_dayno3).

We'll remove the Date and Dayno columns because we've already extracted all of the information we need. Following the preparation of the data, the dataset will look like TABLE II.

B. Partitioning the dataset

The dataset is divided into four sections. Training, validation, validation with training, and forecasting. Both the input and output variables are included in this split.

The training partition will be records 1-21 (index 0-20), the validation partition will be records 22-28 (index 21-27), we will use records 1-28 (index 0-27) for both training and validation, so it will be a training plus validation partition, and the forecasted values will be records 29 to 35 (index 28-34)

C. Separate the outcome variable from the input Data Frame

At this point, we have partitions that include the outcome variable and the input variables. Now we separate the x variables from the y variable. We remove the y-variable from other Data Frames that contains the input variables. The y-variable becomes a series. This is done for each partition. Because the forecast partition contains no y values, we don't need to separate the y value from it.

D. Discover which combination of input variables create the best model

Model I – with all input variables:

Now, using the training data, fit (train) the model. The training data for this model came from a Data Frame that included all time variables and dummy variables.

Then, to ensure that our model produces an estimate of the regression coefficient for the intercept, we must add a constant column to the training data.

We can now use the trained model to predict the y values for the validation dataset records. TABLE III shows the result of the actual vs the predicted values side-by-side.

TABLE III. ACTUAL VS PREDICTED VALUES GENERATED BY MODEL I

	ACTUAL	PREDICTED
21	8.25	8.188483
22	8.16	8.248846
23	8.22	8.312467
24	8.19	8.199349
25	8.21	8.322823
26	9.22	9.332890
27	9.12	9.342883

Model II – without t_dayno3 column:

TABLE IV. ACTUAL VS PREDICTED VALUES GENERATED BY MODEL II

	ACTUAL	PREDICTED
21	8.25	8.189185
22	8.16	8.249794
23	8.22	8.313736
24	8.19	8.201011
25	8.21	8.324953
26	9.22	9.335562
27	9.12	9.346171

Model III – without t_dayno2 column:

TABLE V. ACTUAL VS PREDICTED VALUES GENERATED BY MODEL III

	ACTUAL	PREDICTED
21	8.25	8.194889
22	8.16	8.257498
23	8.22	8.324088
24	8.19	8.214658
25	8.21	8.342541
26	9.22	9.357739
27	9.12	9.373584

Model IV – without t_dayno2 and t_dayno3 columns:

TABLE V. ACTUAL VS PREDICTED VALUES GENERATED BY MODEL IV

	ACTUAL	PREDICTED
21	8.25	8.166786
22	8.16	8.184286
23	8.22	8.241786
24	8.19	8.144286
25	8.21	8.236786
26	9.22	9.241786
27	9.12	9.219286

E. Evaluating models

To check the quality of our regression, use model quality metrics such as Coefficient of Determination (R2), Mean Absolute Error (MAE), Root Mean Squared Error on Prediction (RMSE). These metrics are calculated by comparing the y_actual with y_predicted.

TABLE VI. COMPARISON OF MODELS WITH RESPECT TO R2, MAE, RMSE

	MODEL I:	MODEL II	MODEL III	MODEL IV
R2	0.928602	0.926414	0.906641	0.963517
MAE	0.1	0.1	0.1	0.1
RMSE	0.1	0.1	0.1	0.1

As a result, we discovered that model 4 provides greater forecast accuracy.

F. *Predict (forecast) future values using the training model*

We use modal IV to forecast future values since it produces the best results.

TABLE VI. PREDICTED WAKE TIME

	Day	Predicted
0	1	8.166786
1	2	8.184286
2	3	8.241786
3	4	8.144286
4	5	8.236786
5	6	9.241786
6	7	9.219286

III. SMART DEVICE AUTOMATION USING PREDICTED DATA

Before getting started with using predicted data to trigger automation, we need to create automation first. For this, we use Home Assistant, an open-source platform that can be used to manage any smart home devices as well as automate them.

Home Assistant is a Python-based home-automation server that prioritizes privacy and local control. It can be installed on Raspberry Pi's or your local system to connect to a variety of Internet of Things (IoT) devices and cloud services through Wi-Fi.

A. Setting up Home Assistant

When you initially start the Home Assistant server, it will prompt to create an account.

After creating an account, we need to connect to your device

One of two methods exists for integrating Home Assistant with HomeKit-enabled devices:

- **HomeKit Controller**: connects accessories that have the "Works with HomeKit" badge.
- **HomeKit Bridge**: allows you to use HomeKit to operate Home Assistant-compatible devices.

In a nutshell, HomeKit standardizes all of the different IoT devices that manufacturers create, and then Home Assistant allows you to use those devices.

Confirm that the device and the system on which Home Assistant is installed are on the same network. Otherwise, the IoT device will not be detected by Home Assistant.

B. Creating automation

Python scripts can be used to create "automations" for IoT devices. A trigger, a condition, and an action comprise each automation. Home Assistant comes with a nice selection of pre-programmed triggers that serve as the foundation for automation. The following are some examples:

- **Sun Trigger**: triggers when the sun sets or rises in the geolocation you've specified.
- **Time Trigger**: This feature triggers at a predetermined time.

- **Home Assistant Trigger**: When the Home Assistant starts or stops, this trigger is activated. This can be used to make all of your IoT devices default to a certain state.
- **Webhook Trigger**: When you send a POST request to the defined webhook route, it becomes active.

In this approach, the webhook trigger is used to activate our device.

- On the sidebar, choose the "Configuration" button, then the "Automations" option on the right.
- Select the "Webhook Trigger" and set it up. (Fig. 1)
- The path of your webhook is defined by its ID. "/api/webhook/test" will be the path in this case. A POST request to this path will initiate the next step's activity.
- Select "Device" and select desired action from the Actions menu. (Fig. 2)
- Click Save. The Home Assistant takes care of all the Python programming for you and writes the script for you.

Fig. 1: Adding a trigger

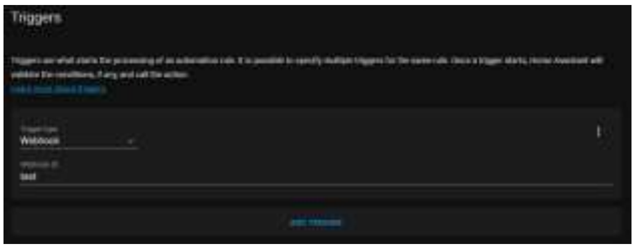
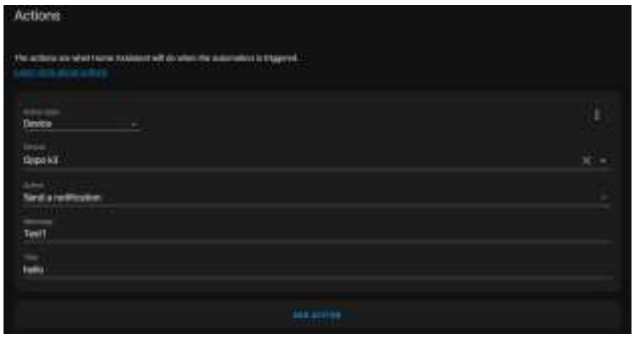

Triggers
[illegible]
Webhook
test

Fig. 2: Adding an action


Actions
[illegible]
Device
Oppo k3
Send a notification
Test1
hello

C. *Triggering the automation using predicted data*

Sending a post request to a preset webhook id can initiate this automation.

We'll use basic python code to check if the day and predicted waking time match the current day and time, and then send a post request to the webhook id.

IV. CONCLUSION

Our lives have become very time-constrained as a result of our demanding schedules and hectic lifestyles. A smart home automation solution helps you to simplify your daily life, boost home security, and save time without having to worry about the small jobs.

The concept of a smart system has evolved beyond simply asking Siri or Alexa for updates or the weather forecast. You can now use smart gadgets to clean floors or turn off the lights in the kitchen while watching a movie in the living room, thanks to technology infiltrating every aspect of life.

In reality, home automation is a concept spawned by the concept of smart homes. It provides you complete control over your electronic gadgets in your home from anywhere in the world using a smartphone or mobile device.

Adding machine learning-based prediction to these automation systems will take them to a whole new level. It improves the "smartness" of smart home devices and eliminates the need for rush in our everyday routine.

V. ACKNOWLEDGMENT

This work was supported in part by Department of Computer Applications, Amal Jyothi College of Engineering. This paper and the research behind it would not have been possible without the exceptional support of my guide, Mr. Jinson Devis. His enthusiasm, intelligence, and strong attention to detail have inspired me and helped me stay on track with my work.

VI. REFERENCES

[1] Forecasting Behavior in Smart Homes Based on Sleep and Wake Patterns by Jennifer A. Williams and Diane J. Cook, Technol Health Care. Author manuscript; available in PMC 2017 Jun 7.
[2] Multiple Linear Regression Using Python and Scikit-learn [Analytics Vidhya].
[3] Multiple Linear Regression using Python [GeekforGeeks].
[4] Home Automation with Python: Beginner-Friendly Python Project [ActiveState].
[5] Ways to Evaluate Regression Models [Towards Data Science]
[6] Linear Regression for Machine Learning [Machine Learning Mastery]
[7] How To Model Time Series Data with Linear Regression [Towards Data Science]