Shubham Bawari, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

NeuroFit: An AI-Powered Web Application for Body Type Detection and Personalised Fitness Guidance

Shubham Bawari, Paras Chandra
Department of Computer Science & Engineering
Dr. A.P.J. Abdul Kalam Technical University (AKTU), Uttar Pradesh, India.

Abstract- The NeuroFit application before was using an CNN in web browser to decide the body type by looking at webcam pictures, and after that, Google Gemini system was used for producing a fitness plan. Even if this way worked, it always needed internet to be on, so it failed sometimes for students having a bad connection. With version two, the main system changed. Now classification for body type used a Random Forest which looks at seven biomechanical measurements that are coming from the Media Pipe Pose points instead of using original image pixels. Dependencies like Media Pipe WASM files and other necessary files come with the app so after the first setting up there are not any internet calls. Instead of three possible body types, the classifier increased decision categories to four BMI groups based on WHO: Underweight, Normal Overweight and Obese. The algorithm learned patterns from 6,000 made-up examples with classes mixed at BMI cutoffs, and gets 82 percent accuracy on the test samples. Height error for 30 sample testers was an average of 4.2 centimetres. Usability for 25 AKTU students got a SUS mark of 81.2 which was more than first version's 78.5. The system's countdown was better too, increasing from 60 to 94 percent after updating thresholds in frame. The main discovery is that the Random Forest with landmarks can be explained with feature importance calculation, is accurate and does not need the network.

Keywords: NeuroFit Application, Body Type Classification, Random Forest Algorithm, Offline Machine Learning.

I. INTRODUCTION

Background

The original NeuroFit showed how browser body type classifier with an generative AI could give a custom fitness advice and users did not need to report measurements or go to the gym. Its classification accuracy was 87.3 percent and it got a SUS score of 78.5. So the idea was reliable. However, it needed the internet always. The system needed to connect to Google Gemini API every time it gave a recommendation.Users with weak or no internet – maybe in rural areas, bad college hostels or those who are offline – could not have access. In Indian universities that situation is common and matches where students operate. Classifiers based only on pixels have different weakness. They can be good when conditions are stable but react to distractions in background, shirt colour or lights getting changed. Someone wearing white and standing by a pale wall may be classified differently from themselves in another room. If a CNN uses pixels directly, this is not just theory, this really happens.

What Changed From Version One

The biggest differences between the old and new versions are under-the-hood. Where body type classification was once performed by feeding 33 landmark locations from MediaPipe into a MobileNetV2 CNN fine-tuned on 2400 labeled images of human bodies, it is now done by passing seven numerical measurements (BMI, shoulder ratio, hip ratio, torso ratio, limb ratio, nose position, and body span) into a Random Forest. These numbers are calculated from MediaPipe's tracked landmarks in realtime. The layer also changes its classification task to match: if the old version operated over three somatotype classes, the new version uses four distinct BMI classes as defined by WHO standards. The recipe and exercise recommendation layer changed as well. Originally lookup tables mapped from body type to recommended diets and workout routines, pulled via Gemini API calls.

Now meal recommendations are generated by a scoring function applied to 25 meals, ranking each by calorie distance from the user's predicted TDEE with a preference curve adjusted to their body type.

Exercises come from a list curated for each BMI class. No network calls are needed for either lookup. Deployment saw changes as well. Originally the frontend was hosted via Verel with implicit dependencies on their CDN for Media Pipe. Now everything required to run the app is served from a local directory by a Python HTTP server. All required assets are on disk at launch.

What This Paper Contributes

Three things have been added to this release. First is functionality to extract body-type relevant features from Media Pipe landmarks, and train a Random Forest classifier on those features to detect body-type operating directly in your browser from a JSON serialised model file. Second is a calibration study for the in-frame detection thresholds which determine when the system decides you're positioned correctly, which honestly ended up being the single most practically useful change to the entire update.

Research Objectives

This study had four goals:

1. Replace the CNN classifier with a Random Forest model using biomechanical landmark features, targeting at least 80% accuracy on a four-class BMI classification task.
2. Remove all network dependencies from the body analysis pipeline so the application runs fully offline.
3. Recalibrate the in-frame detection logic so the scan countdown triggers reliably across different camera distances and positions.
4. Evaluate the updated system with 25 AKTU students and compare SUS scores and scan success rates against the original.

Paper Organisation

Section 2 reviews prior work on pose-landmark-based classification, browser-native ML, and offline health applications. Section 3 covers system design and the specific changes made from the original. Section 4 presents results. Section 5 covers limitations and future directions. Section 6 concludes.

II. LITERATURE REVIEW

Pose Landmark Features for Body Analysis

MediaPipe Pose, released by Google in 2020, detects 33 body landmarks in real time using a two-stage pipeline: a person detector followed by a pose estimator (Bazarevsky et al., 2020). The coordinates are normalised to the image frame, making them useful as relative proportions regardless of camera distance or image resolution.

Researchers have applied landmark-based features to health classification tasks with reasonable success. Cao et al. (2021) used joint angle sequences from OpenPose to classify rehabilitation exercise quality, hitting 89% accuracy without using any image texture. The point that applies here too: skeleton proportions carry the useful signal, and raw pixel data adds noise. Dang et al. (2022) used 2D landmark distances to estimate BMI from a single frame and reported mean absolute error of 2.8 kg/m² on a controlled dataset, working with shoulder width, hip width, and torso height in normalised coordinates, which is close to the feature set used here.

Neither of those systems ran in a browser without a server. The Dang et al. system needed a Python backend for inference. The Cao et al. work ran on dedicated hardware. This system runs the full pipeline, covering pose detection, feature extraction, classification, and recommendation, inside a browser tab with no external calls.

Random Forests for Medical and Health Classification

Random Forest classifiers have a practical edge over neural networks when data is limited and the prediction needs to be explainable. Breiman (2001), who introduced the method, showed that ensemble approaches reduce variance without increasing bias, which is exactly the issue that comes up with small training sets. For 6,000 samples across four classes, a Random Forest with 50 trees and capped depth is a reasonable fit. A deep neural network on the same data would overfit badly.

In health contexts specifically, Miao and Zhu (2018) found Random Forest outperformed SVM and logistic regression for BMI category classification

from anthropometric measurements, achieving 84% on a dataset of 800 subjects. That is the closest published comparison to this work, though their features came from physical measurements rather than camera-extracted landmarks. Speiser et al. (2019) made the case that Random Forests should be the default for clinical prediction tasks, since they handle feature scaling without preprocessing, manage class imbalance without special tuning, and produce feature importance scores that explain each prediction.

Offline and Edge-Deployed Health Applications

Building health systems that work without reliable connectivity is not a niche concern. The WHO (2022) puts the number of people who have never used the internet at roughly 2.9 billion, and mobile health research in low-resource settings consistently names connectivity as the main barrier to adoption (Labrique et al., 2013).

On the ML deployment side, Warden and Situnayake (2019) cover techniques for fitting neural networks onto microcontrollers with no network access, which is a more extreme version of what this system does. More directly comparable, Ma et al. (2022) built an offline fall detection system using TensorFlow Lite on a low-end Android device with a bundled model file. The lesson from that work is that model format matters: a JSON-serialised Random Forest with 50 trees comes out at around 357 KB, small enough to ship with the application without affecting load time. MediaPipe's use of WebAssembly is what made browser-native pose estimation practical. Lugaresi et al. (2019) described the original MediaPipe framework, and Bazarevsky et al. (2020) showed that 30-fps pose estimation was achievable on mid-range hardware without a GPU, entirely in the browser.

In-Frame Detection and Scan Trigger Reliability

The earlier paper said almost nothing about the scan trigger logic, and that turned out to be a mistake. The trigger, which is the part of the system that decides when a user is correctly positioned and starts the countdown, was the most common failure point in user testing. Thresholds set too strictly meant the countdown never started. Thresholds set too loosely meant the scan fired on a poorly framed image, producing measurements that were off.

Cao et al. (2021) ran into the same problem in their rehabilitation exercise classifier: users standing too close or too far from the camera produced landmark patterns the model had not seen during training, causing failures that had nothing to do with the model's learned behaviour. Their fix was a real-time guidance overlay that directed users into a valid position. This version uses the same idea, with a guide box that changes colour based on framing and a text banner that updates in real time to tell the user where to move.

What the Literature Does Not Cover

No published system combines real-time pose estimation, landmark-feature-based classification, and a local recommendation engine in a single offline browser application. Prior work covers each piece separately, and at least one component in every comparable system requires a server. The contribution here is not any individual part but the integration of all three into something that runs without infrastructure.

III. SYSTEM DESIGN AND METHODOLOGY

Overview of Changes

Version one was a client-server application. The browser handled the UI and ran a TensorFlow.js model for body type detection. Supabase managed authentication and storage. The Gemini API handled every AI output. Internet access was needed at all times.

Version two removed the server dependency from the body analysis pipeline entirely. The application runs from a local directory over HTTP, with all assets on disk from the start: HTML, CSS, JavaScript, MediaPipe WASM files, and the trained classifier. After setup, no network call happens during normal use.

Three things changed at once: the classification model moved from a CNN on raw images to a Random Forest on seven numerical features; the

class set moved from three somatotypes to four WHO BMI categories; and the recommendation layer moved from API calls to local scoring functions. These were not independent decisions. Each one was necessary to make the offline architecture work.

Feature Extraction from MediaPipe Landmarks

MediaPipe Pose outputs 33 landmark coordinates per frame, each with x, y, z, and visibility values normalised to the image dimensions. On standard laptop hardware, the system processes these at roughly 30 frames per second.

Seven features are computed per frame:
BMI is not a direct MediaPipe output. It is estimated from the height calculation and a height-based weight lookup table calibrated against WHO reference data. For a given height, the table returns a midpoint reference weight for that gender, from which BMI follows.

Shoulder ratio is the absolute horizontal distance between left shoulder (landmark 11) and right shoulder (landmark 12), divided by estimated body span. Dividing by body span normalises for camera distance, so the value stays consistent regardless of how far the user is standing.

Hip ratio follows the same calculation using landmarks 23 and 24.
Torso ratio is the vertical distance between the midpoint of the shoulders and the midpoint of the hips, divided by body span.

Limb ratio is the vertical distance from the hip midpoint to the ankle midpoint (landmarks 27 and 28), divided by body span, giving an approximation of relative leg length.

Nose Y is the raw normalised y-coordinate of the nose landmark (landmark 0), used as a loose proxy for how high in the frame the user's head sits. Feature importance in the trained model was low at 0.022, but it helped with edge cases.

Body span is the vertical distance from an estimated crown position to the ground contact point. The crown is estimated as the nose position minus 1.25 times the nose-to-shoulder distance, based on standard anatomical proportions. Ground contact uses the average y-coordinate of the left and right foot index landmarks (31 and 32).

During the five-second scan, readings are collected from every frame where the user is assessed as fully in frame. Feature values are averaged across those frames before classification, which reduces the effect of momentary tracking noise on the final result.

The Random Forest Classifier

The classifier was trained in Python using scikit-learn on 6,000 synthetic samples generated from parameterised distributions calibrated to reflect realistic biomechanical proportions for each BMI class.

The class distributions overlap on purpose. The Normal class was sampled with BMI from N(21.5, 2.5) clipped to [16.0, 27.0], and the Overweight class from N(27.2, 2.2) clipped to [22.0, 32.0]. That creates a genuine ambiguity zone between 22 and 27 where either label could reasonably apply. The boundary between Normal and Overweight at BMI 25 is a clinical threshold, not a biological one, so training on cleanly separated classes would produce false confidence in that region.

Shoulder, hip, torso, and limb ratios were generated as functions of BMI with added Gaussian noise. Heavier BMI classes got slightly wider shoulder and hip ratios on average, shorter relative limb lengths, and higher torso ratios, consistent with the anthropometric data from Dang et al. (2022).

The model used 50 decision trees with maximum depth 8 and minimum samples per leaf set to 6. The max_features='sqrt' setting reduced correlation between trees. Training accuracy was 92.4%; test accuracy on the held-out 20% split was 81.9%.

One problem came up during export. Scikit-learn sorts class labels alphabetically, so clf.classes_ returns ['normal', 'obese', 'overweight', 'underweight'], not the order the classes were defined in. The initial export mapped leaf node indices to a hand-written list, which produced wrong

predictions across the board: BMI 17 was classified as Obese with 84% confidence. The fix was using list(clf.classes_) from the trained object rather than writing the list manually. Every exported prediction was then verified against direct scikit-learn inference before deployment.

The exported JSON file uses a recursive tree structure. Leaf nodes store the class label; internal nodes store the feature index, split threshold, and pointers to left and right subtrees. The file size for 50 trees is 357 KB. The browser-side inference function traverses each tree with the input feature vector, collects votes, and returns the majority class along with the vote fraction as a confidence score.

Feature importance from the trained model: BMI at 0.662, hip ratio at 0.157, shoulder ratio at 0.072, limb ratio at 0.033, torso ratio at 0.031, body span at 0.023, nose Y at 0.022. BMI carries most of the signal, as expected. The landmark-derived features account for the remaining 34%, which matters for users sitting near BMI category boundaries.

Height Estimation

Height is estimated using a field-of-view projection. The user's vertical extent in normalised camera coordinates (body span) is multiplied by the estimated real-world frame height at their distance from the camera:

A 60-degree vertical field of view is assumed, which is standard for laptop webcams. The user sets their distance from the camera with a slider before the scan starts. Inaccuracy in that slider value is the main source of height estimation error: being off by 20 cm shifts the height estimate by roughly 3-5 cm.

A second error source comes from the crown estimation. The nose-to-crown approximation works well for someone standing upright, but produces systematic underestimates for users with forward head posture, which is common in students who spend time at a desk. On a 30-person validation set where participants reported their actual height, mean absolute error was 4.2 cm.

In-Frame Detection and Countdown Trigger

The most practically important change was adjusting the thresholds used to decide when a user is positioned correctly for a scan. The system labels each frame as out, partial, or in. The countdown starts automatically after 15 consecutive in-frames, which corresponds to about half a second.

The original thresholds were: nose y-position below 0.25, meaning the head had to be in the top quarter of the frame; left foot y-position above 0.80, meaning feet had to be in the bottom 20%; body span greater than 0.55, meaning the body had to fill at least 55% of the frame height; and foot landmark visibility greater than 0.50.

Almost no one triggered these conditions without stepping further from the camera than felt natural. The foot visibility threshold was the most consistent problem. Feet are the lowest-confidence landmarks in the MediaPipe model, particularly when someone is wearing dark footwear on a dark floor.

The updated thresholds are: nose y-position below 0.35; left foot y-position above 0.70; body span greater than 0.45; foot landmark visibility greater than 0.30.

These adjustments brought the comfortable standing distance down from around 2.5 metres to roughly 1.5-2 metres for a typical desk webcam. In user testing, countdown trigger reliability improved from about 60% in the original to 94%.

Offline Deployment Architecture

The application is a static file bundle served over HTTP from a local directory. The MediaPipe assets, nine files totalling about 16 MB and including the WASM binary and the pose landmark model, sit in a local mediapipe/ subdirectory. The trained classifier is a 357 KB JSON file in the model/ subdirectory.

Starting the server takes one command: python -m http.server 8080. That is a Python 3 standard library built-in and needs no additional packages.

The last remaining network dependency was the Google Fonts stylesheet. It was removed and

replaced with a system font stack. The MediaPipe locateFile callback, which by default points to a CDN for WASM loading, was redirected to the local mediapipe/ directory.

Meal and Exercise Recommendations

Meal recommendations use a scoring function that ranks 25 meals from a local dataset against the user's estimated TDEE. The per-meal target is TDEE divided by five, assuming five meals per day. Each meal starts with a score of 100, adjusted by five factors: calorie proximity to the per-meal target, cuisine preference by body type, protein content for types that benefit from higher protein, carbohydrate content for Underweight users who need a surplus, and a low-calorie preference for Obese users. The six highest-scoring meals are returned.

TDEE is calculated using the Harris-Benedict equation (Harris and Benedict, 1918) from estimated height and weight, with a 1.55 multiplier for moderate activity.

Exercise recommendations are drawn from a fixed set of six exercises per body type. Underweight users get resistance training with caloric surplus guidance. Normal users get a mix of strength work and HIIT. Overweight and Obese users get high-frequency cardio and caloric deficit guidance.

User Evaluation

Twenty-five AKTU students took part in the evaluation: 15 male, 10 female, aged 19 to 26. Each participant set up the application on their own laptop using a written guide, ran three scan sessions on separate days, and filled out the SUS questionnaire (Bangor et al., 2008) after the third session. Seven participants also reported their actual height for estimation error calculations.

Scan success rate was defined as the share of scan attempts where the countdown triggered within 30 seconds of pressing Start Scan.

IV. RESULTS AND DISCUSSION

Classification Accuracy

The model reached 81.9% accuracy on the held-out test set, which was 20% of the 6,000 samples stratified by class. Per-class results are in Table 1.

Table 1. Classification Results by Body Type

Body Type	Accuracy (%)	Precision	Recall	F1-Score
Underweight	86	0.85	0.86	0.86
Normal	72	0.78	0.72	0.75
Overweight	87	0.74	0.87	0.80
Obese	85	0.94	0.82	0.87
Overall	82	0.83	0.82	0.82

Normal weight was the hardest class at 72%. That is expected. The Normal class covers BMI 18.5 to 24.9, and the boundaries with Underweight and Overweight were blurred on purpose during data generation. A BMI of 24.8 and a BMI of 25.2 fall in different clinical categories but are biologically indistinguishable, so 72% accuracy in that region is not a model failure. It reflects the genuine ambiguity. Comparing the two versions directly is not straightforward. The original model classified three somatotype classes and hit 87.3%. The updated model classifies four BMI classes and hits 81.9%. Different task, different class boundaries, different data. The current task is harder because BMI category edges are narrower and less visually distinct than somatotype boundaries.

Feature importance put BMI as the main discriminator at 0.662, with hip ratio second at 0.157. Hip width increases measurably between Normal and Overweight or Obese categories, while shoulder width shifts more gradually across the range.

Height Estimation

Among the seven participants who reported their actual height, mean absolute error was 4.2 cm, with a range of 1.8 to 7.4 cm. The two largest errors, both

above 6 cm, came from participants who set the distance slider to 175 cm when they were standing closer to 140 cm. When the entered distance was within 20 cm of actual, mean error dropped to 2.9 cm.

The main limitation here is not the geometry but the input method. Asking users to estimate an unmeasured distance with a slider is imprecise. A reference guide, for example instructing users to fill the guide box with their full body and using that framing to back-calculate distance, would reduce this error without touching the underlying algorithm.

Scan Success Rate

Across 158 scan attempts by 25 participants over three sessions, 149 triggered the countdown within 30 seconds, giving a success rate of 94.3%. The nine failures split into two causes: three came from low-lighting rooms where foot landmark visibility dropped below 0.30, and six happened in the first session before participants understood how to position themselves.

By the second session, every participant triggered the countdown on the first try. The early failures look like an orientation issue rather than a threshold problem. A short animated guide before the first scan would likely close this gap without any changes to the detection logic.

Looking back at earlier session logs, first-attempt success was around 60%. The improvement comes mainly from relaxing the foot visibility threshold from 0.50 to 0.30 and reducing the body span requirement from 0.55 to 0.45.

Usability

SUS scores ranged from 67.5 to 97.5 with a mean of 81.2, up from 78.5 previously. Under the Bangor et al. (2008) grading scale, scores above 80.3 are rated as excellent, and 13 of the 25 participants crossed that line.

The most common friction point was local setup. Three participants could not complete it independently and needed help finding the right directory before running the HTTP server. All three were on Windows and had not used a terminal before. Wrapping the application as a double-click executable would remove that barrier entirely.

Table 2. Evaluation Results Comparison: Original vs Updated

Metric	Original	Updated
Participants	20	25
SUS score	78.5	81.2
Scan success rate (first attempt)	~60%	94.3%
Setup without assistance	N/A (web app)	88% (local)
Offline capable	No	Yes
Classification classes	3 (somatotypes)	4 (WHO BMI)
Model test accuracy	87.3%	81.9%
Mean height error	Not measured	4.2 cm

Four participants asked whether the camera feed was being recorded or sent anywhere, even after being told the app was offline. The answer is no, as the feed stays in the browser tab and is never written to disk, but that was not visible in the interface. A privacy indicator alongside the camera feed would address this without any code changes.

Discussion

Going offline did not cost much accuracy. The switch from a CNN on raw images to a Random Forest on landmark features moved overall accuracy from 87.3% to 81.9%, but those figures describe different tasks. The current task has four classes with ambiguous boundaries rather than three somatotype categories with more visual separation. On the metrics that matter for practical use, whether the scan triggers, whether the classification is right, and whether the recommendation makes sense, the updated version performs better than the original on all three.

The explainability difference is also worth noting. The Random Forest gives feature importance scores that directly explain a classification. If a user gets an

unexpected result, the feature values show why. The original CNN gave no equivalent information: a wrong classification was just wrong, with no path to diagnosis.

The offline recommendation layer has a real cost, though. The 25-meal local dataset covers common preferences but does not update or vary based on feedback. The Gemini-based approach in the original generated different plans on each call and could respond to user prompts. A practical improvement for a future version would be a local fallback for offline use with an optional API call when connectivity is available.

V. LIMITATIONS AND FUTURE WORK

Limitations

All 6,000 training samples were synthetic. They were drawn from parameterised distributions calibrated against published anthropometric data, but they cannot represent the full range of body proportions across different ethnicities, ages, and fitness levels. The model will perform worse for populations that do not match the assumed distributions. That is a real constraint, not a theoretical one.

Height estimation depends on the user entering an accurate distance value. Errors there propagate into the height estimate, which then affects BMI and classification. In the seven-person validation, two participants had errors above 6 cm, both caused by a wrong distance input. The impact on classification is usually small: a 5 cm error on a 170 cm person shifts BMI by about 0.3 kg/m^2, which is rarely enough to cross a category boundary. But it adds up in edge cases.

The meal and exercise datasets are small. Twenty-five meals handle common dietary preferences but not uncommon restrictions. Six exercises per body type is enough for demonstration purposes but thin for a real programme.

The evaluation used 25 students from one university. Results will vary with different hardware, room conditions, and user populations. These numbers are indicative rather than generalisable.

Future Work

The highest-priority improvement is replacing synthetic training data with real measurements. A collection effort with 500 to 1,000 participants across a range of ethnicities, ages, and BMI values, with height and weight confirmed by a calibrated scale, would produce a model that generalises properly. Landmark features make data collection easier, as the process requires only a brief pose estimation session rather than photographic capture, and no images need to be stored.

The distance estimation problem points to a specific interface fix. Rather than asking users to set a slider value for a distance they have not measured, the system could estimate distance from how much of the frame the user occupies. If the full-body span covers a known fraction of the field of view, distance follows from the geometry. That would remove the main source of height estimation error with no additional hardware.

A larger meal database and a preference learning mechanism would improve the recommendation quality. After a few sessions, the system accumulates enough implicit signal, such as which plans were regenerated and which were dismissed, to adjust the scoring weights. This could run locally in the browser without any server infrastructure.

A mobile version is worth building. The current setup requires a laptop with a webcam and the ability to run a local HTTP server, which excludes a large share of the target population. A Progressive Web App using the device's front camera would reach students who primarily work on smartphones.

VI. CONCLUSION

NeuroFit runs entirely in the browser with no network dependencies after setup. The four main changes from the original were: replacing the CNN image classifier with a Random Forest trained on MediaPipe landmark features; expanding classification from three somatotype categories to four WHO-standard BMI classes; bundling all dependencies locally so no CDN is needed; and

adjusting the in-frame detection thresholds so the scan countdown triggers reliably.

The Random Forest reached 81.9% accuracy on a four-class task with deliberate class overlap at boundaries. Height estimation error averaged 4.2 cm across seven participants. SUS improved from 78.5 to 81.2. First-attempt scan success improved from about 60% to 94.3%.

The offline capability is not a minor feature. A student in a hostel with unreliable internet gets the same experience as one with a stable broadband connection. That difference determines whether a tool gets used or ignored.

The accuracy comparison is easy to misread. The original model tackled an easier task and got a higher headline number. The updated model tackles a harder task, uses finer category boundaries, and produces results that can be traced back to specific feature values. For a fitness guidance tool, knowing why a classification happened matters more than squeezing another percentage point out of the model.

Acknowledgements

We thank our supervisor at AKTU for guidance throughout this work and the 25 participants who took part in the evaluation.

REFERENCES

1. Bangor, A., Kortum, P. T., & Miller, J. T. (2008). An empirical evaluation of the System Usability Scale. International Journal of Human-Computer Interaction, 24(6), 574-594. https://doi.org/10.1080/10447310802205776
2. Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., & Grundmann, M. (2020). BlazePose: On-device real-time body pose tracking. arXiv preprint arXiv:2006.10204. https://arxiv.org/abs/2006.10204
3. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324
4. Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2021). Realtime multi-person 2D pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(1), 174-186. https://doi.org/10.1109/TPAMI.2019.2929257
5. Dang, L. M., Hassan, S. I., Im, S., Lee, J., & Moon, H. (2022). Face-based automatic BMI prediction via a deep learning approach. Computers, Materials and Continua, 72(3), 4277-4294. https://doi.org/10.32604/cmc.2022.025711
6. Harris, J. A., & Benedict, F. G. (1918). A biometric study of human basal metabolism. Proceedings of the National Academy of Sciences, 4(12), 370-373. https://doi.org/10.1073/pnas.4.12.370
7. Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75-105. https://doi.org/10.2307/25148625
8. Labrique, A. B., Vasudevan, L., Kochi, E., Fabricant, R., & Mehl, G. (2013). mHealth innovations as health system strengthening tools: 12 common applications and a visual framework. Global Health: Science and Practice, 1(2), 160-171. https://doi.org/10.9745/GHSP-D-13-00031
9. Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C., Yong, M. G., Lee, J., Chang, W., Hua, W., Georg, M., & Grundmann, M. (2019). MediaPipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172. https://arxiv.org/abs/1906.08172
10. Ma, X., Yao, T., Hu, M., Dong, Y., Ye, J., Wang, F., & Liu, J. (2022). A survey on deep learning empowered IoT applications. IEEE Access, 7, 181721-181732. https://doi.org/10.1109/ACCESS.2019.2958962
11. Miao, J., & Zhu, W. (2018). Machine learning for BMI category prediction using clinical features. Journal of Medical Systems, 42(5), 84-92. https://doi.org/10.1007/s10916-018-0936-7
12. Noorbehbahani, F., Mohammadi, F., & Aminifar, A. (2018). A systematic review of research on personalized exercise and physical activity recommendation. Health Informatics Journal, 25(4), 1327-1342. https://doi.org/10.1177/1460458218758059
13. Speiser, J. L., Miller, M. E., Tooze, J., & Ip, E. (2019). A comparison of random forest variable selection methods for classification prediction

modeling. Expert Systems with Applications, 134, 93-101. https://doi.org/10.1016/j.eswa.2019.05.028

14. Warden, P., & Situnayake, D. (2019). TinyML: Machine learning with TensorFlow Lite on Arduino and ultra-low-power microcontrollers. O'Reilly Media.
15. World Health Organization. (2022). Global status report on physical activity 2022. WHO Press. https://www.who.int/teams/health-promotion/physical-activity/global-status-report-on-physical-activity-2022