Reducing Energy Consumption in Android Devices with User Profile Analysis and AI-based Feedback

Elian Souza, Edwin Monteiro, Raimundo Barreto, Rosiane de Freitas

Institute of Computing – Federal University of Amazonas (UFAM)
CEP 69067-005 – Manaus – AM – Brazil

{elian.souza, edwin, rbarreto, rosiane}@icomp.ufam.edu.br

Abstract. *In recent years, smartphones have greatly expanded in functionality and capability, incorporating advanced features and AI processing. However, battery evolution has lagged, creating challenges for usability and efficiency. Improving battery durability and health remains a critical concern for users. This study tackles mobile device energy consumption by using AI solutions to reduce it. The Tucandeira Data Collector (TDC) app was developed to collect daily data on factors like screen brightness, CPU usage, screen-on time, RAM usage, and unused features like Bluetooth. This data forms a database for analyzing consumption patterns and building a personalized user profile. Machine learning models, including decision trees, random forests, and neural networks, are trained to identify patterns that impact battery life. The Curica Smart Alert (CSA) app uses this profile to provide real-time data collection and personalized feedback, predicting potential battery gains in minutes or hours based on user actions. Accepting these suggestions can extend battery life. The study's findings are promising, with the Random Forest (RF) model achieving high accuracy and the Deep Neural Network (DNN) model performing well. Integrated into CSA, these models offer effective recommendations for optimizing energy use without affecting smartphone performance or inconveniencing users.*

1. Introduction

In recent years, mobile devices, especially smartphones, have significantly advanced in terms of functionalities and capabilities. They incorporate a wide variety of features, from high-resolution cameras and advanced sensors to powerful artificial intelligence processing capabilities. However, despite these technological advances, the evolution of batteries has not improved in the same way. The limited battery capacity remains one of the biggest challenges faced by smartphone users, directly affecting the usability and efficiency of the devices.

The need to improve battery health, both in terms of durability and charge cycles, has led to growing interest in studies focused on optimizing energy consumption. Various approaches have been explored to address this issue, with software-based solutions standing out due to the infeasibility of hardware interventions in modern devices, which generally have non-removable batteries. This scenario highlights the importance of developing effective methods to manage energy consumption through software solutions.

One way to address this issue is to collect and analyze device usage data. The literature points to the common practice of developing mobile applications dedicated to data collection, allowing the construction of databases for energy consumption analysis. Examples include the works of [Barreto Neto et al. 2020] and [Pereira et al. 2021], where data collection is essential for creating predictive models that help understand and optimize the energy consumption of mobile devices.

However, the quality and accessibility of databases are constant challenges. Although the study by [Pereira et al. 2021] makes its databases publicly available, there are significant limitations. The correlation analysis of [Pereira et al. 2021] data, for example, revealed that many attributes were not sufficiently robust or relevant for detailed energy consumption analyses. Additionally, compatibility issues with newer versions of Android and difficulties in modifying existing databases motivated the need for a newer approach.

This study proposes the creation of a new database using an application specifically developed for this purpose, compatible with the latest versions of Android (11 and above). Additionally, an application was developed to provide specialized feedback to users, assisting in the recommendation and optimization stages. The creation of this database alongside with the app aims to allow data modification according to the specific needs of the research, ensuring flexibility and quality in data collection.

The central problem of this study is to develop an efficient software-based approach to reduce energy consumption in mobile devices. This involves not only creating a robust and well-labeled database but also applying advanced machine learning techniques to identify usage patterns that negatively impact battery life. From these patterns, personalized recommendations will be generated to optimize energy consumption using algorithms such as decision trees, Random Forest, and neural networks. The public availability of this new database on Mendeley Data [Monteiro et al. 2024] represents a significant contribution, allowing other research to benefit from the collected data and contribute to this field of study.

This work is organized as follows: Section 2, Theoretical Framework, presents the concepts and AI techniques used. Section 3, Related Work, discusses and compares studies related to energy efficiency in Android devices and user profile identification. Section 4, Proposed Method, describes the architecture of the method, data collection and preprocessing, model definition, and validation. Section 5, Results and Discussions, presents the experimental results and data analysis. Finally, Section 6, Conclusions, summarizes the findings and contributions of the research and describes future steps.

2. Theoretical Foundation

2.1. The Problem of Class Definition in AI

The definition of classes in Artificial Intelligence (AI) problems is crucial, especially when classes are not predefined. In such scenarios, techniques like pseudo-labeling and binning are used to structure the input data so that they can be effectively processed by learning algorithms. Pseudo-labeling is a semi-supervised technique that uses models to assign labels to unlabeled data. An algorithm used for this type of labeling is K-means.

2.2. Dimensionality Reduction

Dimensionality reduction, such as Principal Component Analysis (PCA), is often essential when using K-means on datasets with many variables. PCA reduces the number of features while retaining most relevant information, improving efficiency and reducing overfitting risks. It transforms correlated variables into uncorrelated principal components, ordered by the data variation they capture. K-means then partitions the reduced dataset into k clusters, assigning data points to the nearest centroid and iteratively refining the clusters until the centroids stabilize.

2.3. Energy Consumption Calculation

An important step in analyzing a device's energy consumption is defining how to calculate energy. Based on the temporal distribution of the dataset, [Barreto Neto et al. 2020] describes the calculation of the definite integral to obtain the energy consumption E over time t from the instantaneous power P. To calculate $E(t)$, we must first obtain $P(t)$ as illustrated in Equation 2, where I represents the battery current and V the battery voltage.

$$E(t) = \int_0^t P(t)\,dt \tag{1}$$

$$P(t) = I(t) \times V(t) \tag{2}$$

Thus, $E(t) = P(t) \times t$. If $t = 1s$, then the energy consumption is given by $E(t) = P(t),$ if $t = 1$.

Due to the limitations of requiring hardware modifications or privileged access, system-level tools such as PowerAPI and RAPL are not feasible for use in this testing environment. The experiments are conducted on consumer-end Android devices, where such modifications are impractical. Instead, power consumption is calculated using Android's APIs, which capture voltage and current data. This approach ensures compatibility across a wide range of devices, allowing for broader experimentation without the need for specialized hardware or system-level permissions.

3. Related Work

This section discusses related work that addresses the classification and management of energy consumption in mobile devices, focusing on methodologies, algorithms, and user recommendations. The analyzed works are compared with the approach of this study, which aims to create user profiles and provide personalized recommendations for optimizing energy consumption.

3.1. Objectives and Methodologies

The article by [Mehrotra et al. 2021] aims to classify mobile applications based on energy consumption using multiclass classification techniques. The methodology includes collecting energy consumption data and applying machine learning algorithms to create predictive models. In contrast, the work by [Barreto Neto et al. 2020] seeks to build energy consumption models based on user usage patterns. This work uses data collected from smartphones and applies machine learning techniques to identify patterns and make predictions about energy consumption.

3.2. Data Collection and Databases

Data collection in the paper [Mehrotra et al. 2021] is performed through device logs, while the work by [Barreto Neto et al. 2020] uses similarly collected data but with a specific focus on user usage patterns. Both works use private databases and do not make the data publicly available. In comparison, the present study generated a public database, available at [Monteiro et al. 2024] , collected through the Tucandeira Data Collector app, which records attributes such as screen brightness, Bluetooth status, GPS usage, among others. Tucandeira is compatible with the latest versions of Android (11 and above), representing a significant advantage over previous works that use outdated applications like PowerTutor and Trepn.

3.3. Clustering Algorithms and Techniques

The K-means algorithm is widely used in the analyzed works to cluster users based on their usage patterns. In the article [Duan et al. 2017], K-means is used to classify users into active and inactive, as well as devices into low or high energy consumption. Similarly, the work by [Pereira et al. 2021] uses clustering techniques to group energy consumption data from different devices. This study, however, goes beyond simple clustering by applying decision trees, random forest, and neural networks to extract conditional rules that identify high energy consumption patterns, followed by a regression approach to estimate the potential energy savings.

3.4. User Recommendations

The works of [Duan et al. 2017] and [Pereira et al. 2021] highlight the importance of providing recommendations to users to optimize energy consumption. They use clustering results to identify anomalous behaviors and suggest adjustments such as turning off unused network interfaces and adjusting screen brightness. This study differentiates itself by offering more detailed and personalized recommendations based on rules extracted from decision trees and validated by regression algorithms, which estimate the potential battery life gain in minutes or hours. The feedback construction and delivery to the user are implemented in the Curica Smart Alert app, which provides personalized feedback to the user with recommendations for actions to reduce energy consumption.

3.5. Comparison with the Present Study

This study stands out for its approach combining clustering techniques, decision trees, random forest, and deep neural network (DNN). Unlike previous works that focus on identifying usage patterns and categorizing users, this study quantifies the impact of recommendations in terms of energy savings, estimating the battery time gain. For data collection and generation, the Tucandeira app was developed, compatible with the latest versions of Android. Additionally, the Curica Smart Alert app was created to build personalized recommendations.

Table 1 presents a comparison between the present work and related works, highlighting the public availability of the database and the unique AI-based feedback approach of this study.

Table 1. Comparison between related works.

Article	Database	Main Algorithm	User Recommendations	Publicly Accessible
[Mehrotra et al. 2021]	Private	Decision Tree	No	No
[Barreto Neto et al. 2020]	Private	Neural Network	Yes	No
[Pereira et al. 2021]	Public (GreenHub)	Clustering	Yes	Yes
[Duan et al. 2017]	Private	K-means	Yes	No
Present Study	Public (Mendeley)	Decision Trees, *Random Forest*, Neural Networks	Yes, with estimated savings and feedback via Curica	Yes

4. Proposed Method

The proposed architecture for reducing energy consumption in smartphones, illustrated in Figure 1, consists of three main modules: Data Collector, AI Model Generator, and Feedback Generator. All modules operate directly on the smartphone. The **Data Collector** module is responsible for capturing user interaction data with the smartphone. The data is collected using the Tucandeira Data Collector app, which monitors attributes related to energy consumption such as screen brightness, CPU usage, screen-on time, current, and voltage. Data is captured every second, daily, and stored in CSV files both locally and in the cloud in a private directory. Due to the absence of predefined labels in the dataset, K-Means clustering was employed to group the data into three clusters representing different energy consumption levels. The elbow method was used to justify the selection of the optimal number of clusters, ensuring a balance between complexity and interpretability. Multiple initializations were applied to guarantee stable results, while 95% of the data variability was preserved to retain significant information from the original dataset.

The **AI Model Generator** module is responsible for training Decision Tree, Random Forest, and Deep Neural Network (DNN) models based on the collected data. This task is executed by the Curica Smart Collector app. Once the data is labeled using K-Means, a Decision Tree classifier is trained to classify user energy consumption profiles into low, medium, or high categories. The decision tree uses the entropy criterion to measure the quality of the splits, focusing on reducing uncertainty in the data. A fixed random seed was used to ensure the model's reproducibility. Decision trees were chosen for their interpretability, allowing the extraction of clear rules that link user behavior to energy consumption patterns.

Finally, the **Feedback Generator** module captures real-time data and compares it with values estimated by the DNN, where the inputs are derived from parameter adjustments based on predefined rules from the analysis of the attributes in the database, associated with minimum values extracted from the Decision Tree. The DNN consists of two hidden layers with 128 and 64 neurons, respectively, both using ReLU activation functions. The network was compiled with the Adam optimizer, chosen for its efficiency in handling large datasets, and trained using the mean squared error as the loss function, which is appropriate for the regression task. The model was trained for 10 epochs, ensuring convergence without overfitting, and is designed to provide real-time predictions suitable for mobile devices. The output is the feedback on what can be altered by the user, along with the estimated time gained if the feedback suggestions are accepted.

4.1. Data Collector Module

In the first module, the Data Collector continuously collects data in the background while monitoring and recording the user's daily smartphone usage. The below items highlight each data collection stage:

- **Data Collection**: Information based on daily smartphone usage is collected by Tucandeira.
- **Local Storage**: The collected data is stored locally on the device in a database.
- **Cloud Upload**: Daily, the data is uploaded to a private repository on Google Drive, ensuring that the information is available for later processing.

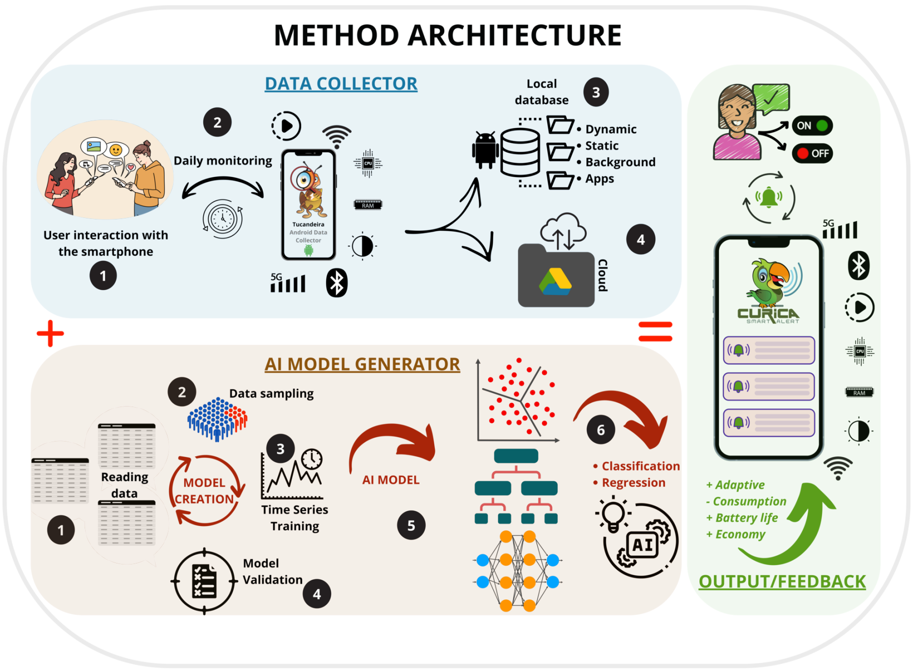


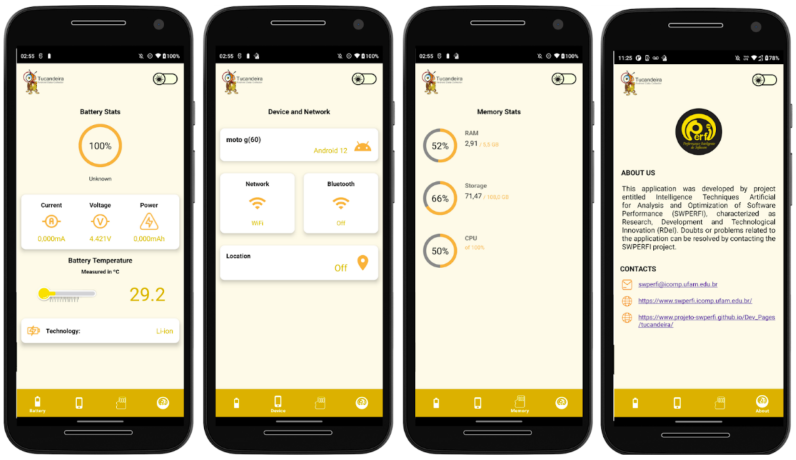
Figure 1. Architecture of the proposed solution: data collection, model training, and feedback generation.

4.1.1. The Tucandeira Data Collection App

Within the project SWPERFI , the Tucandeira Data Collector app was developed, as illustrated in Figure 2, to capture data on Android devices (versions 11 and above). The collected data are stored locally in `./Secondarystorage/Documents/SWPERFI/Tucandeira/Logs/`, with daily copies sent to Google Drive. The data is organized into CSV files divided into four groups:

- **Dynamic Data:** attributes that are constantly changing, such as CPU usage, RAM usage, and screen-on time.
- **Background Data:** data related to applications running in the background, such as codecs and file transfers. Background data is updated throughout the day.
- **Static Data:** data that rarely change, such as manufacturer, battery capacity, and kernel version. These are updated when the Tucandeira app is updated or reinstalled.
- **Application Data:** contains information such as the application identifier (id), package name, and SDK version. These are updated when an application is installed.

The structuring of the collected data is based on the works of [Barreto Neto et al. 2020, Pereira et al. 2021] regarding the definition of relevant attributes to compose the database, such as device identification, screen brightness, battery level, mobile data networks, and Wi-Fi. Additionally, attributes such as CPU usage, the identifier of the application running in the foreground and background, energy consumption in milliampere-hours (mAh), screen-on time, and timestamp are added to the selected set. Another addition to the data collection configuration is the time interval in which new data is obtained, set to 1 second, facilitating the calculation of energy consumption by multiplying voltage by current, as described in Section 2.3. This factor is important as this paper addresses the problem of using a software approach through a mobile application, given that new smartphones do not allow battery removal, making hardware instrumentation difficult. Table 2 presents a sample of the data collected by the device identified as "70a09b5174d07fff". This sample includes attributes such as device identifier (*device_id*), timestamp, screen status (*screen_status*), brightness level (*bright_level*), brightness mode (*bright_mode*), screen-on time (*screen_on_time*), Bluetooth status (*bluetooth*), and GPS status (*gps_status*). A detailed description of the attributes is addressed in [SWPERFI 2024].



This application was developed by project entitled Intelligence Techniques Artificial for Analysis and Optimization of Software Performance (SWPERFI), characterized as Research, Development and Technological Innovation (RDeI). Doubts or problems related to the application can be resolved by contacting the SWPERFI project.

swperfi@icomp.ufam.edu.br
https://www.swperfi.icomp.ufam.edu.br/
https://www.projeto-swperfi.github.io/Dev_Pages/tucandeira/

Figure 2. Different screens of the Tucandeira interface.

Table 2. Sample data collected by the device "70a09b5174d07fff".

device_id	timestamp	screen_status	bright_level	screen_on_time	bluetooth
70a09b5174d07fff	2023-02-26 00:03:09	1	107	0	4
70a09b5174d07fff	2023-02-26 00:03:10	1	107	0	4
70a09b5174d07fff	2023-02-26 00:03:11	1	107	0	4
70a09b5174d07fff	2023-02-23 02:36:37	2	6	16969	4
70a09b5174d07fff	2023-02-23 02:36:38	2	6	16970	4
70a09b5174d07fff	2023-02-23 02:36:39	2	6	16972	4
70a09b5174d07fff	2023-02-23 02:36:40	2	6	16974	4
70a09b5174d07fff	2023-02-23 02:36:41	2	6	16976	4
70a09b5174d07fff	2023-02-23 02:36:42	2	6	16978	4
70a09b5174d07fff	2023-02-23 02:36:43	2	6	16980	4

4.2. AI Model Generator Module

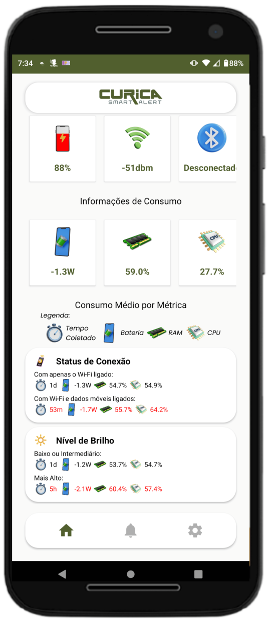
The second module, the AI Model Generator, processes the collected data to train AI models that will be used to predict energy consumption. The entire process of training, classification, and regression is performed on the device using the Curica Smart Alert app.

- **Data Reading**: The data stored in the cloud by the Tucandeira app is read and prepared for analysis.
- **Preprocessing**: This includes steps such as data cleaning, relevant attribute selection, sampling, and temporal organization of the data. The processed data is then structured for subsequent machine learning tasks, ensuring that the attributes are ready for both classification and regression models.

- **Model Training**: Various AI models are trained with the processed data directly in the Curica Smart Alert app, seeking to identify the one that best fits the data. This process includes validating the models to ensure their accuracy and reliability.
- **Classification and Regression**: The generated AI model is used to classify the user's energy consumption profile (high, medium, or low). Additionally, regression models are used to estimate energy consumption based on current data.

4.2.1. The Curica Smart Alert

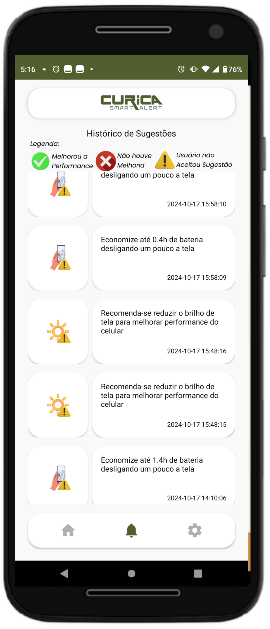
The Curica app, see Figure 3, is structured into three main screens, each designed to provide detailed information and suggestions for optimizing energy consumption on Android devices.



Status de Conexão
Com apenas o Wi-Fi ligado:
1d -1.3W 54.7% 54.9%
Com Wi-Fi e dados móveis ligados:
53m -1.7W 55.7% 64.2%

Nível de Brilho
Baixo ou Intermediário:
1d -1.2W 53.7% 54.7%
Mais Alto:
5h -2.1W 60.4% 57.4%

(a) Main screen



desligando um pouco a tela
2024-10-17 15:58:10

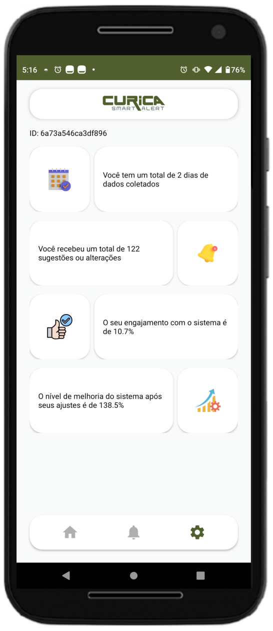
Economize até 0.4h de bateria desligando um pouco a tela
2024-10-17 15:58:09

Recomenda-se reduzir o brilho de tela para melhorar performance do celular
2024-10-17 15:48:16

Recomenda-se reduzir o brilho de tela para melhorar performance do celular
2024-10-17 15:48:15

Economize até 1.4h de bateria desligando um pouco a tela
2024-10-17 14:10:06

(b) Feedback list



ID: 6a73a546ca3df896

Você tem um total de 2 dias de dados coletados

Você recebeu um total de 122 sugestões ou alterações

O seu engajamento com o sistema é de 10.7%

O nível de melhoria do sistema após seus ajustes é de 138.5%

(c) Stats summary

Figure 3. The three main screens of Curica.

The first screen, Figure 3(a), displays general device information, such as battery level, Wi-Fi signal strength, Bluetooth and mobile network status, screen brightness level, and screen-on time. Additionally, it shows energy consumption data, including RAM and CPU usage percentages. This screen offers a comprehensive overview of the system's status and key energy consumption indicators.

The second screen, Figure 3(b), is dedicated to notifications. For each attribute monitored by the app, there is an associated legend indicating whether the suggestion for that attribute was accepted and resulted in performance improvement, if there was no change, or if the suggestion was not accepted. As the app detects the need for adjustments in any attribute, new notifications are generated, providing detailed suggestions on how to optimize energy consumption.

The third screen, Figure 3(c), provides a general summary of the user's interactions with the app. This screen displays the device identifier, the total number of days of data collection, the total number of suggestions provided by the app, and the user's engagement with the recommendations, including feedback generated. This summary allows the user to track usage history and the effectiveness of the suggestions offered by Curica.

4.3. Feedback Generator Module

The third module, called Feedback Generator, applies the trained model in real time to provide user recommendations aimed at optimizing energy consumption.

- **Model Integration into the Curica App**: The AI model is integrated into the Curica Smart Alert app, which classifies energy consumption in real time.
- **Analysis and Adjustments**: If energy consumption is classified as high, the Curica app analyzes the current rules and settings of the smartphone's attributes. It then provides recommendations for adjustments, such as reducing screen brightness, closing background apps, or restarting the device to free up memory.
- **Energy Consumption and Time Gain Estimation**: Along with the adjustments, a regression process estimates the energy consumption. This value is converted into additional battery life, allowing Curica to inform the user how much time can be gained by following the provided recommendations.

5. Results and Discussions

Experiments were conducted to generate feedback on energy consumption in mobile devices using the proposed method, structured in two distinct phases: Rule Generation and Feedback Generation. The first phase was based on a classification approach, where a decision tree was trained to identify the user's profile from daily collected data. Subsequently, the second phase used a regression approach to estimate energy consumption, adjusting the values of the attributes collected in real time to optimize energy usage. The experimentation was conducted on a Motorola Moto G60 smartphone with Android 12. It involved the use of the Tucandeira Data Collector app for data collection and the Curica Smart Alert app to execute the phases. The data was collected from multiple devices and users to capture diverse usage patterns. However, the model is trained per user to reflect individual consumption profiles and provide personalized recommendations. While this supports generalization, future work could expand the dataset to include more devices and scenarios for finer personalization.

5.1. Rule Generation

Based on the methodology described in [Mehrotra et al. 2021], the data was preprocessed and treated using feature selection. The data was also subjected to the PCA algorithm for dimensionality reduction and subsequent application of the K-Means algorithm, which defines the clusters. This cluster definition is essential for rule generation as it allows the database to be labeled in terms of energy consumption levels: low, medium, or high. Therefore, PCA was configured to identify the principal components with $K = 3$, considering this value ideal for clusters based on the elbow method estimation. The labeling is used as the target attribute to train the decision tree.

It is worth noting that, unlike [Mehrotra et al. 2021], the definition of consumption labels in this study is done automatically. The automatic definition of each cluster's energy consumption level is based on the collected data's statistics. Initially, dictionaries are created to store the scores of each relevant attribute for each cluster. Then, for each cluster and each feature, the median value is calculated. Specifically, for the features *battery_power* and *battery_current*, the values are inverted (considering the device's discharge, resulting in negative values) to align the scoring, and these scores are then stored. The scores of each attribute are then normalized concerning the maximum and minimum values found for that attribute across all clusters, ensuring that the scores are within a common scale (0 to 1), facilitating comparison between clusters. After normalization, the normalized scores of each attribute are aggregated to calculate a total score for each cluster, representing the sum of the normalized scores of all relevant attributes. The clusters are then ordered based on the total scores in descending order, and the labels "High", "Medium", and "Low" are assigned to the clusters.

After labeling the data, the user's profile was identified using a decision tree. This tree was used to predict the energy consumption class and extract conditional rules that allow optimizing the collected parameters if energy consumption is high, using values from low or medium consumption classes. For example, if consumption is high, then low or medium consumption values are extracted from the tree to replace the high consumption, to predict later if there is any gain with this change. It is worth noting that a completely different tree can be generated for data collected on different days from the same user.

Some of the rules extracted from the decision tree include:

- If the brightness level (*bright_level*) is greater than 80, the brightness level is reduced to 60. Rule extracted from the tree generated for the smartphone during experimentation.
- If Bluetooth (*bluetooth*) is in one of the states 1 (connecting), 3 (disconnecting), or 4 (disconnected) without change, it is suggested to turn it off (*bluetooth = 0*). Example of feedback: *"It is advisable to turn off Bluetooth when not in use to save energy."*
- If the network mode (*network_mode*), Wi-Fi status (*wifi_status*), and Wi-Fi intensity (*wifi_intensity*) indicate a weak connection, the Wi-Fi value is set to 0, and it is suggested to activate mobile data.
- In low battery scenarios, if the battery level (*battery_level*) is less than or equal to 20, the power-saving mode attribute is set to 1 (enabled), and the screen brightness attribute is reduced (the value is extracted from the decision tree).
- If RAM usage (*ram_usage*) is greater than 4 GB, it is suggested to close background applications or even restart the smartphone.
- Specific rules for GPS include turning off GPS if the GPS activity (*gps_activity*) is 0 or turning off GPS if the activity is 1, but the battery level is less than or equal to 20%.

5.2. Feedback Generation

Proceeding with the feedback generation phase for the user, regression is used to estimate energy consumption based on the rules established in the previous phase. The results of the estimates are used to calculate the battery life that can be saved. The feedback phase aims to calculate and communicate the estimated battery life gain resulting from the applied optimizations. The steps performed in this phase are described below:

Training the Regressor Model First, a real-time collection of attributes related to energy consumption is performed, such as Bluetooth status, screen brightness level, screen-on time, and Wi-Fi transmission rate, among others, as described in [Monteiro et al. 2024]. This phase uses a regression approach, where the target is the energy consumption, calculated as the product of current and voltage. Data collection is performed by the Curica Smart Alert app using two models: Random Forest (RF) and Deep Neural Networks (DNN). While Random Forest initially showed strong performance, the DNN was preferred for handling complex data and faster predictions. Both models require retraining for new data as they do not support incremental learning. Once energy consumption is predicted, it is compared with real-time data to determine battery life gain, and rules are generated to guide the user in achieving energy savings.
Calculation of Time Gain in Minutes For each model, Curica calculates the estimated battery life gain in minutes. This gain is obtained by subtracting the current battery life (before optimizations) from the optimized battery life (after optimizations).
Conversion of Minutes to Hours (if applicable) If the estimated battery life gain is greater than 60 minutes, Curica converts this value to hours. This conversion facilitates communication of the time gain, making the information more understandable to the user.
Feedback Generation Based on the calculated gains, Curica Smart Alert generates feedback messages for the user. These messages are formatted to indicate the increase in battery life provided by the optimizations, specifying whether the gain is in minutes or hours. For example, if the estimated gain for the model is more than 60 minutes, the feedback message will indicate an increase in hours. Otherwise, the increase will be indicated in minutes.
Accumulation of Feedback Messages Curica accumulates feedback messages in a list, which is subsequently used to notify the user about improvements in energy consumption and the corresponding extension in battery life. The Curica Smart Alert app's feedback method offers users suggestions, see Figure 3(b) for adjusting settings. It then calculates the potential energy savings if the recommendation is followed and provides an estimate of the increase in battery life, displayed in minutes or hours, to guide the user in making informed decisions.

6. Conclusions

The construction of a database was a crucial challenge, overcome with the development of the Tucandeira app for data collection on Android devices. After collection, data analysis and preprocessing were performed using techniques such as PCA and K-means to identify consumption clusters and label consumption levels. Several models were built and evaluated, with decision trees standing out for representing the user profile and extracting consumption rules. Random Forest and Deep Neural Network (DNN) models were used to estimate the battery time gain with adjustments according to the extracted rules. The DNN, executed on TensorFlow Lite, enabled real-time estimation of battery gain. The results include a public database on Mendeley Data, and the Tucandeira and Curica apps, developed for data collection and energy consumption optimization notifications, respectively. Future steps include using incremental machine learning algorithms, allowing models to adapt in real time, and investigating explainable learning techniques such as the SHAP model. Additionally, Long Short-Term Memory (LSTM) networks will be explored to capture temporal dependencies and reinforcement learning techniques to optimize energy consumption based on user behavior.

References

Barreto Neto, A. C. S., Farias, F., Mialaret, M. A. T., Cartaxo, B., Lima, P. A., e Maciel, P. R. M. (2020). Building energy consumption models based on smartphone user's usage patterns. *CoRR*, abs/2012.10246. `https://arxiv.org/pdf/2012.10246`.

Duan, L.-T., Lawo, M., Rügge, I., e Yu, X. (2017). Power management of smartphones based on device usage patterns. In *Dynamics in Logistics: Proceedings of the 5th International Conference LDIC, 2016 Bremen, Germany*, pages 197–207. Springer.

Mehrotra, D., Srivastava, R., Nagpal, R., e Nagpal, D. (2021). Multiclass classification of mobile applications as per energy consumption. *Journal of King Saud University - Computer and Information Sciences*, 33(6):719–727.

Monteiro, E., Souza, E., José, R., Balico, L., Barreto, R., e de Freitas, R. (2024). A dataset from the daily use of features in android devices. Mendeley Data.

Pereira, R., Matalonga, H., Couto, M., Castor, F., Cabral, B., Carvalho, P., Sousa, S., e Fernandes, J. (2021). Greenhub: a large-scale collaborative dataset to battery consumption analysis of android devices. *Empirical Software Engineering*, 26.

SWPERFI (2024). Tucandeira data collector app. `https://swperfi-project.github.io/Pages-dev/TucdAndroidDataCollector-app/`. Accessed: 2024-06-30.