IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151154

Laptop Battery Report Using Random Forest Regression Algorithm, Machine Learning

Dr. Beenarani Manoj[1], Hariharan N[2], Jeron Sam Varghese[3], Akash Kumar[4]

Department Of Computer Science and Engineering, Faculty of Engineering and Technology,

SRM Institute of Science and Technology, Vadapalani, Chennai, Tamil Nadu, India[1-4]

Abstract: Everyone who uses a laptop heavily has been caught out by a battery that ran flat sooner than expected. The percentage on screen only tells you what is left right now. It says nothing about how quickly that number is falling or where it will be in the next half hour. We wanted to close that gap, so we built a Machine Learning-Based Laptop Battery Drain Forecasting and Early Warning System that looks at how a laptop is being used and predicts its battery level ahead of time.

The system works in a few steps. A Python script keeps logging battery percentage, CPU usage, RAM usage, charging status and elapsed time into a CSV file. We tidied that data with Pandas and used it to train a Random Forest Regressor, which we saved with Joblib so it could be reused without retraining. A Streamlit dashboard then loads the model and puts the current battery status beside the predicted percentage. If the prediction falls below a safe level, a warning appears, giving the user a chance to plug in or save their work before the laptop shuts down. Put together, the project shows how simple data logging and a basic machine learning model can turn an ordinary battery indicator into something that helps people plan their work.

Keywords: Laptop Battery Monitoring, Battery Drain Forecasting, Machine Learning, Random Forest Regressor, Predictive Analytics, Python, Streamlit, Early Warning System

1. INTRODUCTION

I'm rarely without my laptop, and I'd guess the same is true for most students. Coding, online classes, assignments, it all happens on one machine. And nearly everyone has had the battery die at the worst possible moment. The percentage on screen tells you what's left right now, but not how quickly it's draining. That speed keeps changing with what the laptop is doing: how hard the CPU is working, how much RAM is being used, how many apps are open, how bright the screen is.

Built-in battery tools do show a time estimate, but it's easy to see how shaky it is. Open a few heavy programs and the "3 hours left" from a minute ago drops fast. We felt a better approach was to look at how the battery behaved in the past and use that to say where it's heading. That thought became our project.

We built a Machine Learning-Based Laptop Battery Drain Forecasting and Early Warning System. A script logs battery percentage, CPU usage, RAM usage, charging status and elapsed time into a CSV file. We trained a Random Forest Regressor on that log so it can estimate the battery percentage a little way into the future. The model feeds a Streamlit dashboard showing current battery status, live system usage and the predicted level. When the prediction gets low, the dashboard raises a warning, which gives the user time to find a charger or save their work. In short, we wanted a battery monitor that helps you plan ahead instead of just reporting the present.

2. LITERATURE REVIEW

We check the battery icon so often that we rarely stop to ask what it actually tells us. It shows how much charge is left and whether the charger is plugged in, and nothing more. Where the battery is heading is left for us to guess. Past battery data can fill that gap, because machine learning can pick out patterns in it and use them to estimate future charge levels.

Traditional Battery Monitoring: Operating systems already track the basics. Windows, for example, offers system functions that let programs read battery and power information [11]. That is enough to show the current level or the charging state. It was never meant to look ahead, so a user still has no idea how much battery will remain in the next hour.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151154

Machine Learning in Predictive Analytics: Learning from earlier observations in order to predict later ones is the core of predictive machine learning. James et al. [7] and Hastie et al. [8] present a wide range of techniques for doing this. In our case, readings such as battery percentage, CPU usage and RAM usage hint at how fast power is being used up. This raw data needs some preparation first, and Python with Pandas handles the loading, cleaning and organizing [2], [5].

Regression Models and Decision Trees: Our target is a number, a battery percentage, so this is a regression problem rather than a classification one. Decision trees are a natural fit. They ask a series of simple questions about the data and follow the answers to a prediction, and they work for both problem types. Breiman, Friedman, Olshen and Stone [10] explain this approach in detail. For building and testing models like these, we rely on Scikit-learn [9].

Ensemble Learning and Random Forest: One tree on its own can easily be thrown off by quirks in the data. Random Forest avoids this by growing many trees, each from a different sample and a different selection of features, and then combining their outputs into one answer. Breiman [6] introduced the method. We apply it through Scikit-learn's RandomForestRegressor [1] to forecast future battery percentage. Once the model is trained, Joblib stores it [4], so it can be reloaded later without training again.

Dashboard Development and Model Evaluation: A prediction does little good if the user can't see it clearly. Streamlit lets us put the battery details and the forecast on a simple interactive page [3]. Evaluation needs care as well. Our readings are taken over time, so the order of the rows carries meaning, and randomly shuffling them would give results that look better than they really are. TimeSeriesSplit from Scikit-learn avoids this by testing the model in chronological order [12].

3. PROPOSED SYSTEM ARCHITECTURE

We split the project into four stages, and each stage hands its output to the next. First, battery and usage readings are collected from the laptop. Next, those readings are cleaned and shaped into a training dataset. Then a model learns from that dataset and predicts the battery level ahead of time. Last, the results are put on a dashboard that anyone can read at a glance.

3.1 System Features

Module 1 (Battery Data Collection): Everything the project learns from begins here. The module records the battery percentage, charging status, CPU usage, RAM usage and the time at which each reading was taken, and saves all of it in a CSV file. Over time, this file grows into the dataset that the model is trained on.

Module 2 (Data Processing): A raw log isn't ready for training, so this module reads the CSV file, keeps the fields the model needs and arranges them properly. It also builds the value we want to predict: the battery percentage at a later point in time. Without this step the model would have nothing to aim for.

Module 3 (Battery Prediction): The learning takes place in this module. A Random Forest Regressor goes through the prepared data and picks up how battery drain relates to things like CPU and RAM usage. From these patterns it estimates how much charge will be left after a set period. Once trained, the model is stored with Joblib, so it can be loaded straight away next time instead of being trained again.

Module 4 (Dashboard and Early Warning): This is the only module the user directly interacts with. Built with Streamlit, the dashboard shows the current battery percentage, charging status, CPU usage, RAM usage and the predicted battery level. It also checks the prediction against a fixed limit. If the predicted level falls under that limit, a warning is displayed, which leaves the user enough time to plug

4. METHODOLOGY

4.1 Dataset and Features

Predicting how a battery will behave requires data showing how the laptop spends its power while it runs. For that reason, our system keeps a record of several readings in a CSV file, and this file later becomes the training data for the model. Each record includes:
Battery Percentage: the amount of charge the battery holds at that moment.
CPU Usage: the share of processor capacity being used.
RAM Usage: the amount of system memory in use.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151154

Charging Status: whether the charger is connected or the laptop is running on battery.
Elapsed Time: the time that has passed at each reading, which lets us follow how the battery level changes.

4.2 Data Collection and Preprocessing
Data Collection:
We start by recording readings while the laptop is being used. Battery percentage, CPU usage, RAM usage and charging status are written to the CSV file, and each new reading is appended to the existing records. This lets us follow how the battery behaves under different usage conditions.
Data Preparation: The collected readings can't be given to the model as they are. We first pick out the features that are needed and arrange the data for training. We also link each current reading to the correct future battery value, so the model is taught exactly what it needs to predict.

4.3 Target Preparation
Because we want the future battery level, the current percentage alone is not enough as an output. The model needs a later value to learn from. To provide it, we apply a six-row shift to the dataset: the battery percentage recorded six rows after a given reading becomes the target for that reading. Through these pairs, the model learns how the battery level changes as time goes on. The real time covered by six rows depends on how often the readings are logged.

4.4 Model Configuration and Training
Once the dataset is ready, we train a Random Forest Regressor on it. The model examines the battery and system usage details and learns how they connect to battery consumption. It grows several decision trees and combines their results into one prediction. When training finishes, we save the model using Joblib, so it can be loaded whenever the application opens and doesn't need to be trained from scratch. To judge how closely the predictions match the real battery readings, we use Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and the R^2 score.

4.3 Target Preparation
Since our project is mainly focused on predicting future battery levels, we cannot use only the current battery percentage as the output. We need to give the model a future battery value to learn from. For this, we use a six-row shift in the collected dataset. In simple terms, the battery percentage recorded six rows later is used as the target value for the current reading. This helps the model learn how the battery level changes over time. The actual time covered depends on how frequently the readings are collected.

4.4 Model Configuration and Training
After preparing the dataset, we use a Random Forest Regressor to train the prediction model. The model looks at the different battery and system usage details and tries to understand how they are related to battery consumption. It uses multiple decision trees and combines their results to make a prediction. Once the training is completed, the model is saved using Joblib. This makes it easier to use the trained model again whenever the application is opened, without having to train it from the beginning. We also use evaluation measures such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R^2 score to understand how closely the predictions.

5. EXPERIMENTAL RESULTS

5.1 Battery Monitoring Results
After starting the application, we let it collect information on the laptop's battery and system performance. The readings were kept in a CSV file, so we could open it afterwards and trace how the battery level behaved while the laptop was in use. The following details are logged:

Battery Percentage: how much charge the laptop has left.
CPU Usage: how much of the processor is being used.
RAM Usage: how much memory the laptop is using.
Charging Status: whether the laptop is on the charger or running on its battery.
Time: the moment each reading was recorded, so changes can be examined later.

5.2 Battery and System Usage Graphs
A table full of numbers doesn't tell you much at a glance, so we turned the readings into graphs of battery level and CPU usage against time. The battery graph shows how the charge percentage changed, and the CPU graph shows how processor usage varied during the same monitoring session. Seeing the data plotted makes it far easier to follow than reading raw values.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151154

5.3 Laptop Battery Forecast Dashboard

We built a Streamlit dashboard so that the key battery details sit together on a single screen. It displays the current battery percentage, charging condition, CPU usage and memory usage. Beside these, it shows the estimated battery drain rate and the predicted battery percentage after a future time interval. This way, users can see how the battery stands now as well as what the model expects next.

5.4 Overall Results

The project brings battery monitoring and machine learning forecasting together in a single application. It logs laptop usage data, feeds it to the trained Random Forest model to estimate upcoming battery levels, and presents the output on the dashboard. The early warning feature goes a step further by notifying the user when the predicted battery level is low.

All in all, the application offers a simple way to follow battery usage and get an idea of how much charge is likely to remain in the coming period.levels and displays the results through the dashboard. The early warning feature adds another useful part to the application by informing users when the predicted battery level becomes low.

Overall, the project provides a simple way to keep track of battery usage and get an idea of how much charge may be left in the coming period.

6. DISCUSSION

Battery Monitoring and Prediction:

We didn't want the project to simply repeat the battery percentage that every laptop already shows. What interested us was how the battery level moves during normal use, and whether that pattern could help us estimate what comes next. The application logs battery and system details as time passes, which gives users a better sense of where their laptop's power is going. It also makes it easier to see when the battery might run low.

Ease of Use:

We also wanted anyone to be able to use the application, even someone with little technical knowledge. So, we gathered the important details on one Streamlet dashboard. The battery percentage, charging status, CPU usage, RAM usage and the predicted battery level are all visible together, so nobody must search through different settings. The warning feature is a helpful extra. Once the predicted level turns low, users get an opportunity to save their work or connect the charger.

Observed Limitations:

While building the project, we learned that forecasting battery use is not always straightforward. The rate of drain depends on what the user is doing, which applications are running and how heavily the laptop is being used. As a result, the predicted level can occasionally be different from the real reading.

How good the prediction also depends on the data the model was trained on. Our model uses a six-row shift to look ahead, so the interval between logged readings changes how far ahead the prediction goes.

What the system gives is an estimate of the future battery percentage, and it shouldn't be taken as an exact measure of how much battery time is left.

7. CONCLUSION AND FUTURE WORK

The result of this project is an application that lets users monitor their laptop battery and get an idea of how much charge could be left after some time. It started as a basic plan to record battery information, and step by step we added a machine learning model so that the application could make future predictions. The Random Forest Regressor forms the prediction part, and Streamlit brings all the information onto one dashboard. A warning feature is included as well, so users are told when the predicted battery level gets low. The project showed us how machine learning can be used in a simple, everyday situation, and how useful it can be when it is combined with a practical application.

Future Scope:

Collecting More Battery Data:

One improvement we would like to make is to train the model on more data. Readings from different laptops, taken during activities such as browsing, gaming, programming and watching videos, could help the model understand battery usage better.

Making Predictions More Accurate:

At the moment, the future battery value is prepared with a six-row shift. We would like to replace this with the actual

DOI: 10.17148/IJARCCE.2026.151154

time stored for each reading. That way, the system could predict the battery level after a fixed period, even if the readings are taken at slightly different intervals.

Adding More Features:
A few more details about laptop usage could be added to the data, for example screen brightness, running applications and power mode. These may help the model understand why the battery drains faster during certain activities.

Making the Dashboard More Useful:
We would also like to extend the dashboard with additional battery usage graphs, records of previous predictions and better warning options. Together, these would show users how their battery has been performing and make it more comfortable for them to plan their laptop usage.

REFERENCES

[1]. Scikit-learn Developers. (2007–present). "RandomForestRegressor." Scikit-learn Official Documentation. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
[2]. Pandas Development Team. (2008–present). "pandas: Python Data Analysis Library." Pandas Official Documentation. https://pandas.pydata.org/docs/
[3]. Streamlit, Inc. (2019–present). "Streamlit Documentation." Streamlit Official Documentation. https://docs.streamlit.io/
[4]. Joblib Developers. (2008–present). "Persistence Utilities." Joblib Official Documentation. https://joblib.readthedocs.io/
[5]. Python Software Foundation. (1991–present). "Python Documentation." Python Official Documentation. https://docs.python.org/3/
[6]. Breiman, L. (2001). "Random Forests." Machine Learning, Volume 45, pages 5–32. Springer, USA.
[7]. James, G., Witten, D., Hastie, T., and Tibshirani, R. (2021). "An Introduction to Statistical Learning." 2nd Edition. Springer, New York, USA.
[8]. Hastie, T., Tibshirani, R., and Friedman, J. (2009). "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." 2nd Edition. Springer, New York, USA.
[9]. Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, Volume 12, pages 2825–2830.
[10]. Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). "Classification and Regression Trees." Wadsworth International Group, Belmont, California, USA.
[11]. Microsoft Corporation. (n.d.). "GetSystemPowerStatus Function." Microsoft Learn, Redmond, Washington, USA. https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-getsystempowerstatus
[12]. Scikit-learn Developers. (n.d.). "TimeSeriesSplit – Cross-Validation." Scikit-learn Official Documentation. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

 This work is licensed under a Creative Commons Attribution 4.0 International License