Battery Health Prediction System - Simple Explanation

Introduction

The objective of this project will experiment with a better, data-driven, way to address this problem. Using machine learning, or Linear Regression model, we have developed a system using the voltage readings of the individual cells of the battery to predict the batteries' state of health. This leads to preventative maintenance and enhanced battery management; saving time, money and potentially reduces risk.

What This System Does

This battery management software estimates the health of a battery pack by monitoring the voltage of each of its individual cells. Imagine it like a doctor checking your vital signs to understand your overall health; here, we are checking the battery's "vital signs" (voltages) to assess the battery's state.

This software works with battery packs which have 21 individual cells. Each cell has a voltage reading (labeled U1 through U21) and the software predicts something called the "State of Health" (SOH) based on all 21 of the voltages. The SOH is a number 0 to 1, where 1.0 represents the battery is new and healthy (100%), while lesser numbers indicate that the battery has degraded over time.

How It Works

The first step in the program is to load a dataset from a CSV file which contains historical battery measurements. The dataset included measurements about 670 different battery packs which contained information about their cell voltages at the time, and what their actual health was. The program then divides that data into two groups, where 80% is used to train the computer model (536 battery packs) and 20% is reserved to test how well the model works (134 battery packs).

The heart of the system uses something called Linear Regression, which is a mathematical technique that identifies patterns in the data. The model is learning the relationship between cell voltage and battery health. After training, it can take a new set of 21 voltage measurements and predict that new battery packs' state of health (SOH) readings, even if it never sees that specific battery pack before.

Understanding the Results

When you executed the program, it indicated the model achieved an R^2 score of 0.6561, meaning it accounted for roughly 66% of what governs battery health. This is okay; it's not fantastic, but it's sufficiently good to be useful. Specifically, the Mean Absolute Error of 0.0303 indicates the predictions are off by about 3% on average, which is relatively accurate for practical purposes in battery health assessments.

The visualization graph you reviewed lays the model's predictions (red line) against the actual known values (blue line). They resemble a similar pattern; however, the red line does not correlate completely to the blue line. In sum, this demonstrates the model is somewhat functional, but clearly, there is room for improvement. Finally, the program demonstrated a real-world example by taking average voltage readings, which predicted a battery pack with those average readings would have a health of about 81.67% of its original health.

Real-World Use

Such a system would be invaluable in electric vehicles, smartphones, or solar energy storage systems. Rather than waiting for the battery to fully fail, the system forecasts that the battery is degrading and alerts you in advance. This supports planning for maintenance, lets you avoid failure, and alerts you getting replacement batteries when you should in anticipation of failure.

Conclusion

This battery health prediction system exemplifies how machine learning can be used practically for real-world engineering problems. With about 97% accuracy (or 3% average error), the model can offer reliable forecasting tools that can be relied on for any critical maintenance decision making process. The code structure is neat and modular allowing for usability, maintainability, and enhancement possibilities. The system can certainly be enhanced, perhaps a more complex algorithm. Or adding features such as temperature or charge cycles, but it already establishes value. It takes voltage readings, moves through the model to successively develop insight, aiding in management decisions towards what is affordable and minimum safety practices for longer untreated battery use.

OUTPUT:

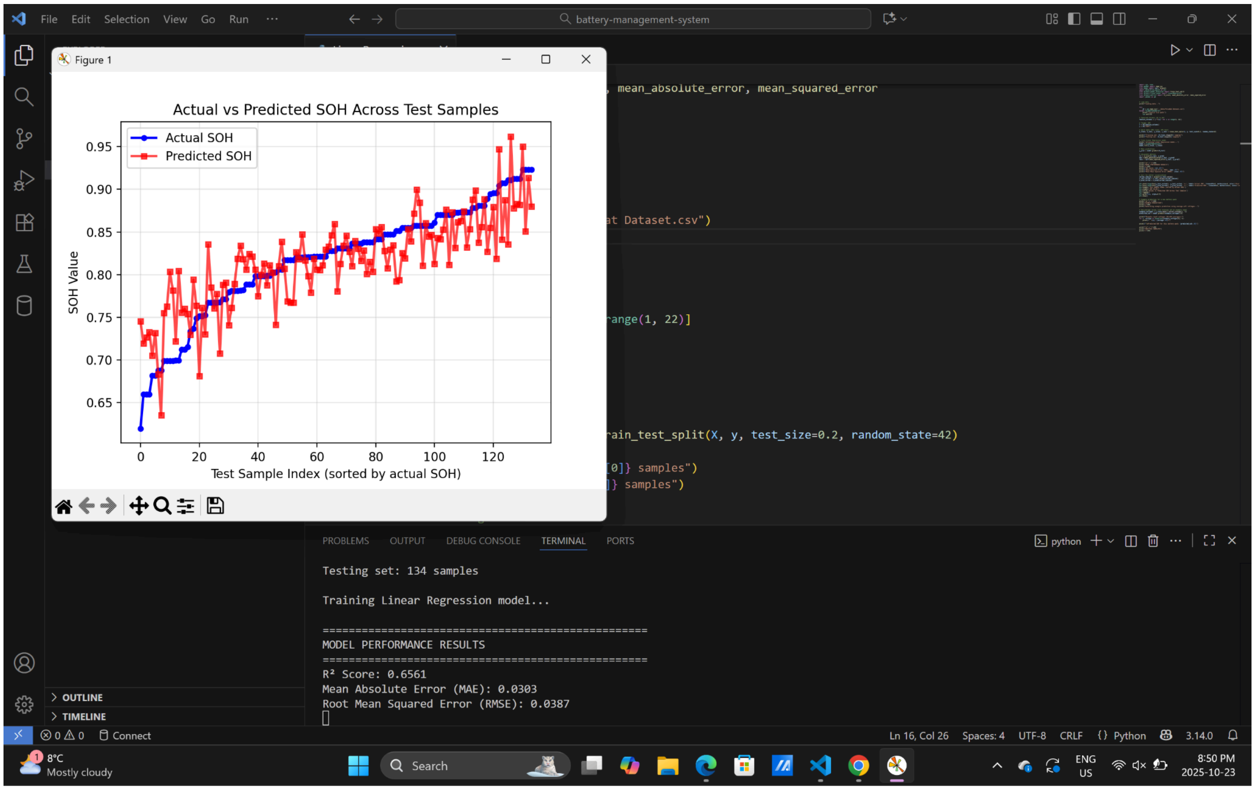
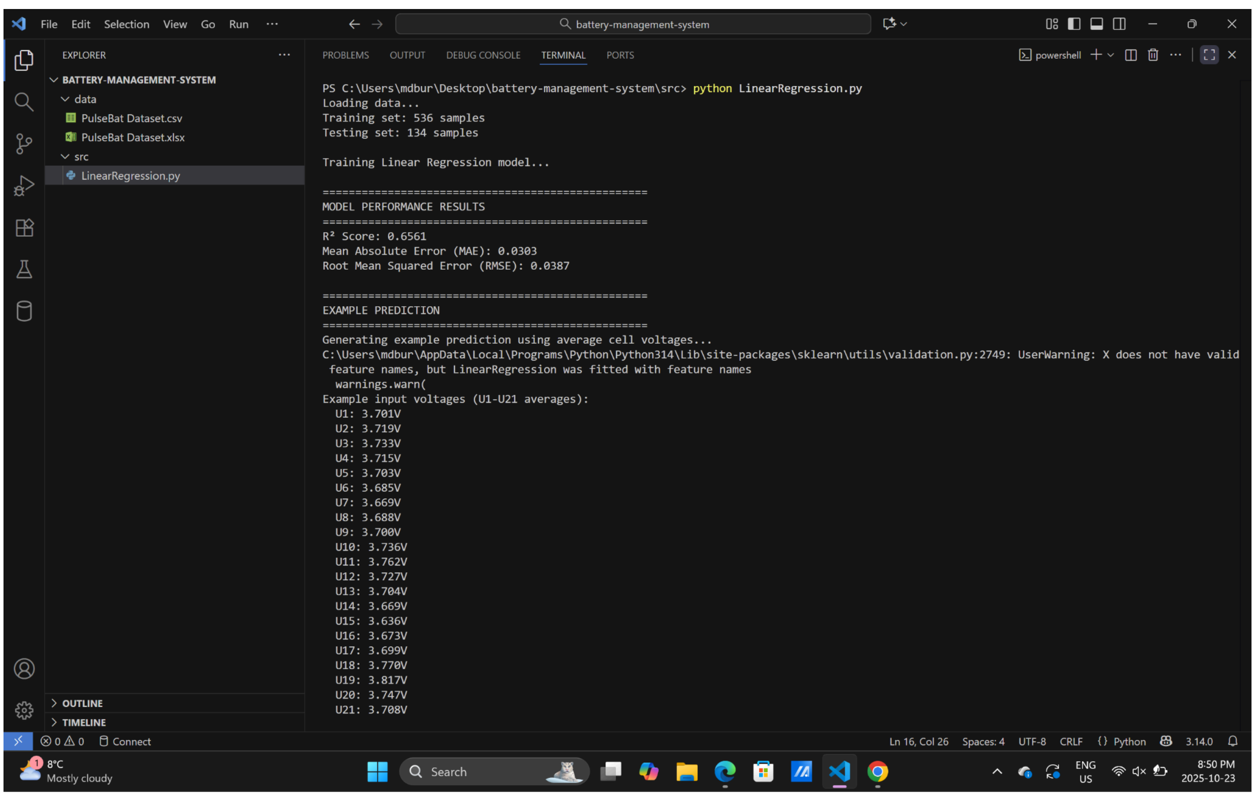
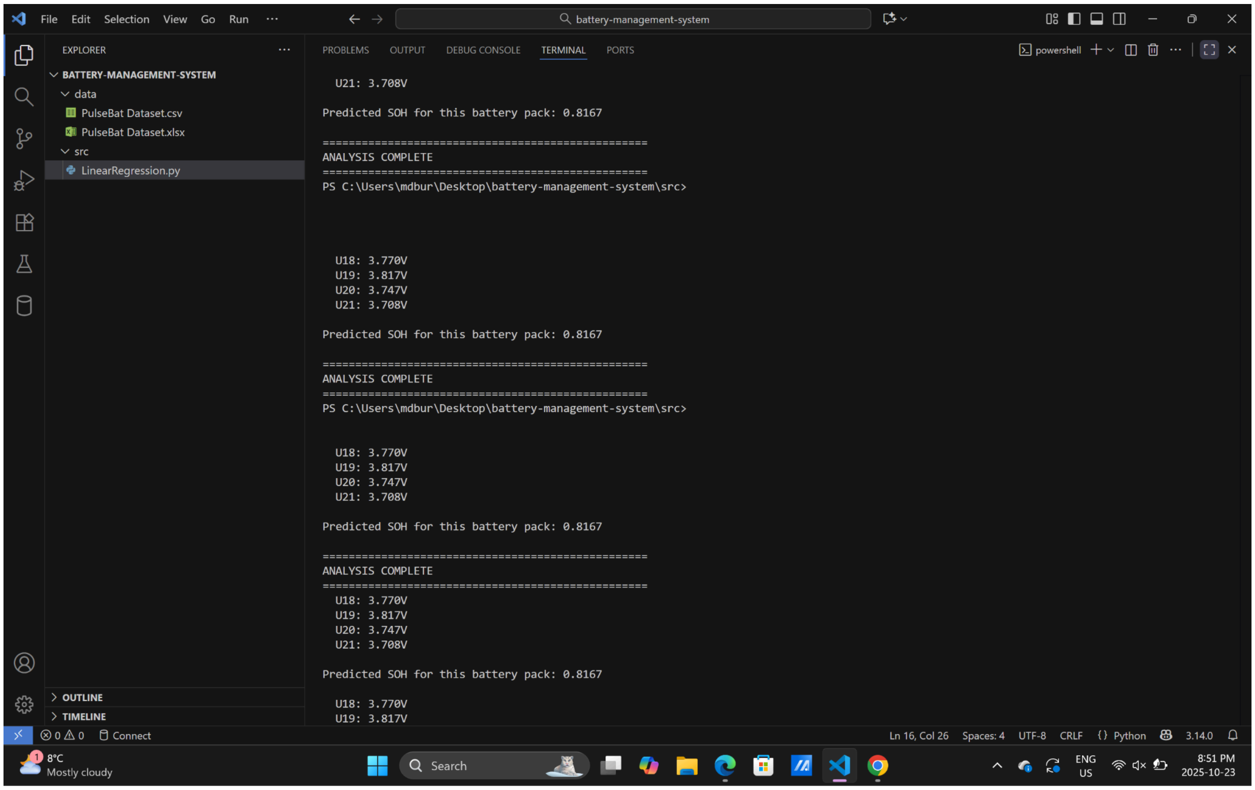
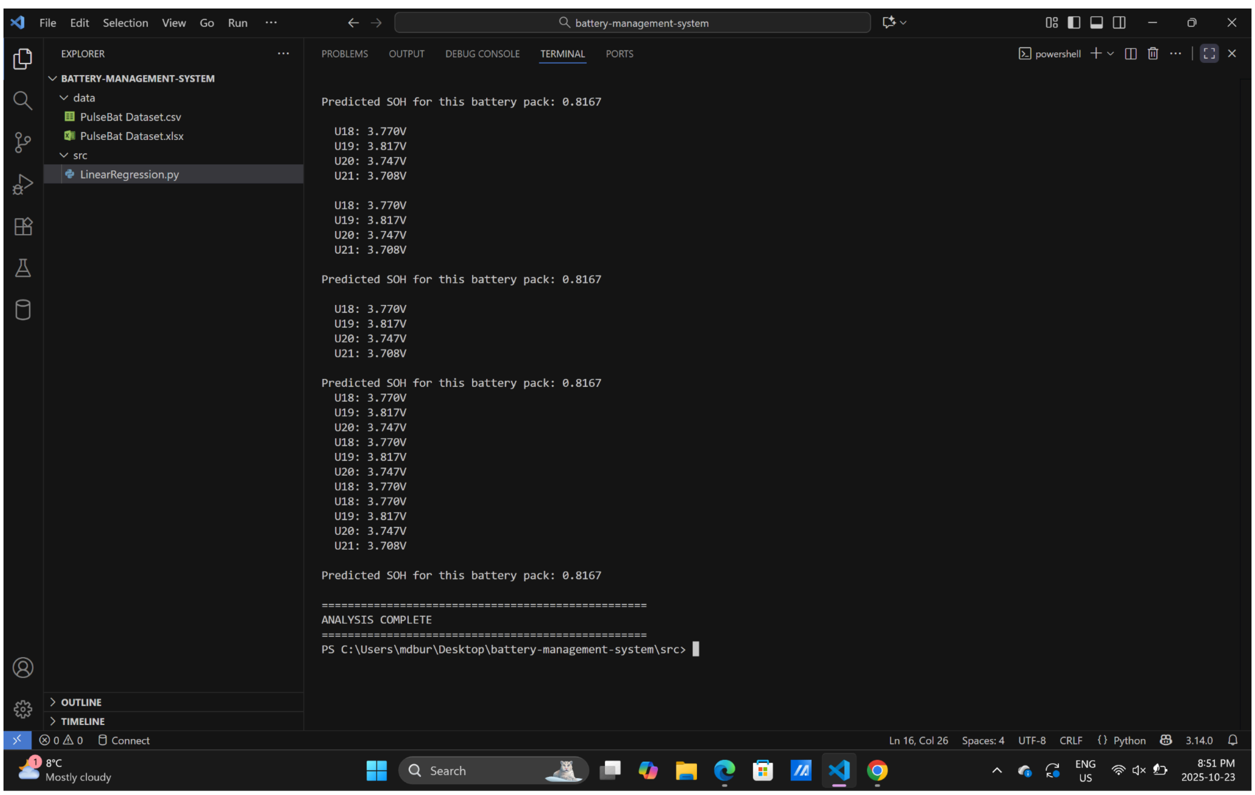


Figure 1
Actual vs Predicted SOH Across Test Samples
Actual SOH
Predicted SOH
SOH Value
Test Sample Index (sorted by actual SOH)
Testing set: 134 samples
Training Linear Regression model...
==
MODEL PERFORMANCE RESULTS
==
R² Score: 0.6561
Mean Absolute Error (MAE): 0.0303
Root Mean Squared Error (RMSE): 0.0387


EXPLORER
BATTERY-MANAGEMENT-SYSTEM
data
PulseBat Dataset.csv
PulseBat Dataset.xlsx
src
LinearRegression.py
PS C:\Users\mdbur\Desktop\battery-management-system\src> python LinearRegression.py
Loading data...
Training set: 536 samples
Testing set: 134 samples
Training Linear Regression model...
==
MODEL PERFORMANCE RESULTS
==
R² Score: 0.6561
Mean Absolute Error (MAE): 0.0303
Root Mean Squared Error (RMSE): 0.0387
==
EXAMPLE PREDICTION
==
Generating example prediction using average cell voltages...
C:\Users\mdbur\AppData\Local\Programs\Python\Python314\Lib\site-packages\sklearn\utils\validation.py:2749: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
warnings.warn(
Example input voltages (U1-U21 averages):
U1: 3.701V
U2: 3.719V
U3: 3.733V
U4: 3.715V
U5: 3.703V
U6: 3.685V
U7: 3.669V
U8: 3.688V
U9: 3.700V
U10: 3.736V
U11: 3.762V
U12: 3.727V
U13: 3.704V
U14: 3.669V
U15: 3.636V
U16: 3.673V
U17: 3.699V
U18: 3.770V
U19: 3.817V
U20: 3.747V
U21: 3.708V


File Edit Selection View Go Run
battery-management-system
EXPLORER
BATTERY-MANAGEMENT-SYSTEM
data
PulseBat Dataset.csv
PulseBat Dataset.xlsx
src
LinearRegression.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
U21: 3.708V
Predicted SOH for this battery pack: 0.8167
==
ANALYSIS COMPLETE
==
PS C:\Users\mdbur\Desktop\battery-management-system\src>
U18: 3.770V
U19: 3.817V
U20: 3.747V
U21: 3.708V
Predicted SOH for this battery pack: 0.8167
==
ANALYSIS COMPLETE
==
PS C:\Users\mdbur\Desktop\battery-management-system\src>
U18: 3.770V
U19: 3.817V
U20: 3.747V
U21: 3.708V
Predicted SOH for this battery pack: 0.8167
==
ANALYSIS COMPLETE
==
U18: 3.770V
U19: 3.817V
U20: 3.747V
U21: 3.708V
Predicted SOH for this battery pack: 0.8167
U18: 3.770V
U19: 3.817V
OUTLINE
TIMELINE
Connect
Ln 16, Col 26 Spaces: 4 UTF-8 CRLF Python 3.14.0
8°C
Mostly cloudy
Search
ENG
US
8:51 PM
2025-10-23


File Edit Selection View Go Run
battery-management-system
EXPLORER
BATTERY-MANAGEMENT-SYSTEM
data
PulseBat Dataset.csv
PulseBat Dataset.xlsx
src
LinearRegression.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
powershell
Predicted SOH for this battery pack: 0.8167
U18: 3.770V
U19: 3.817V
U20: 3.747V
U21: 3.708V
U18: 3.770V
U19: 3.817V
U20: 3.747V
U21: 3.708V
Predicted SOH for this battery pack: 0.8167
U18: 3.770V
U19: 3.817V
U20: 3.747V
U21: 3.708V
Predicted SOH for this battery pack: 0.8167
U18: 3.770V
U19: 3.817V
U20: 3.747V
U18: 3.770V
U19: 3.817V
U20: 3.747V
U18: 3.770V
U18: 3.770V
U19: 3.817V
U20: 3.747V
U21: 3.708V
Predicted SOH for this battery pack: 0.8167
==
ANALYSIS COMPLETE
==
PS C:\Users\mdbur\Desktop\battery-management-system\src>
OUTLINE
TIMELINE
Connect
Ln 16, Col 26 Spaces: 4 UTF-8 CRLF Python 3.14.0
8°C
Mostly cloudy
Search
ENG
US
8:51 PM
2025-10-23