Problem: How to estimate the life cycle of battery using machine learning

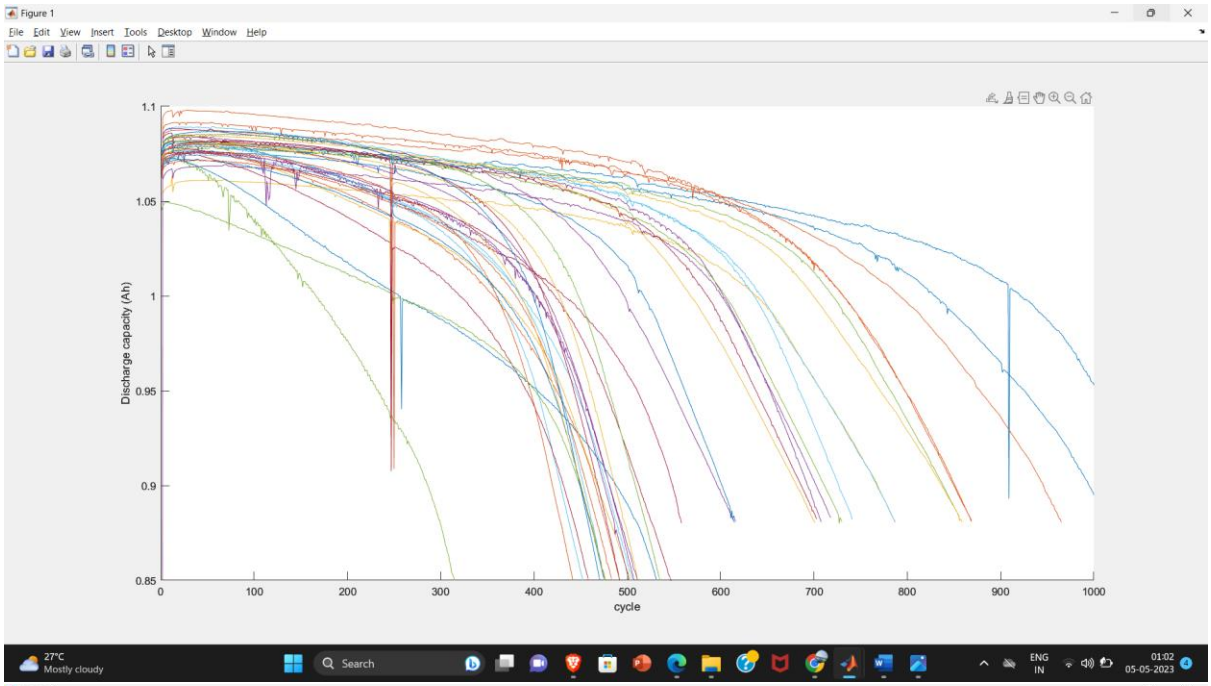
Solution code:

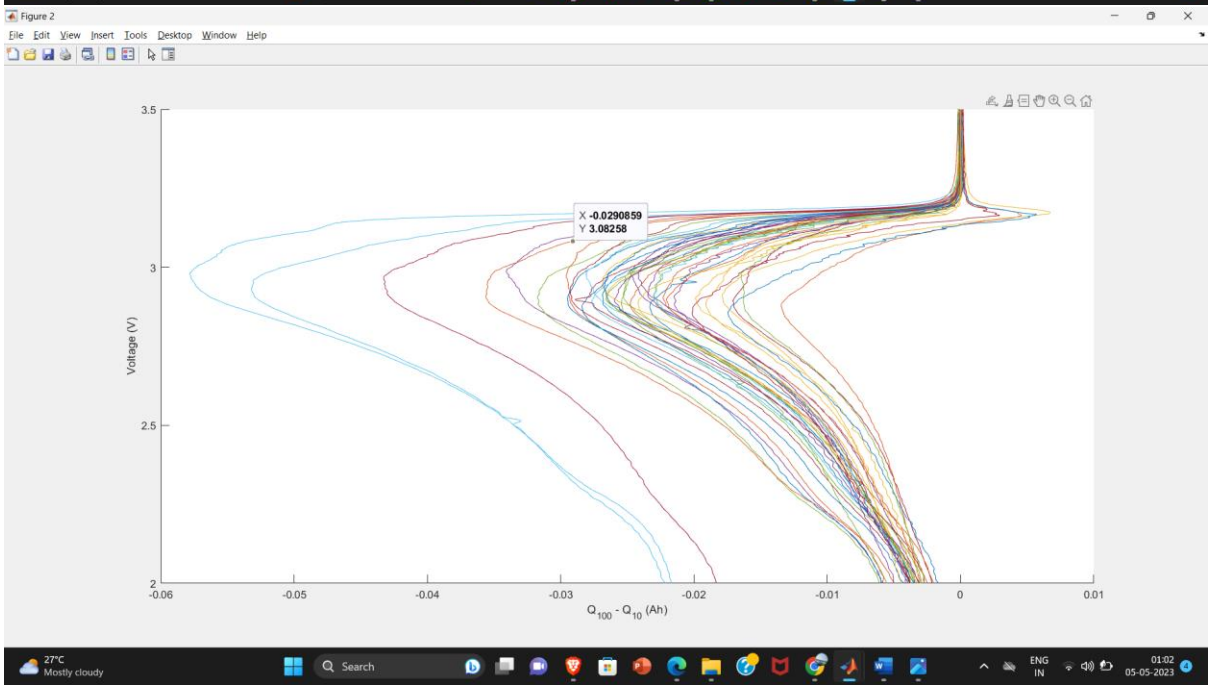
```
figure, hold on;
for i = 1:size(trainData,2)
    if numel(trainData(i).summary.cycle) ==
numel(unique(trainData(i).summary.cycle))
      plot(trainData(i).summary.cycle, trainData(i).summary.QDischarge);
    end
end
ylim([0.85,1.1]),xlim([0,1000]);
ylabel('Discharge capacity (Ah)');
xlabel('cycle');

figure, hold on;
for i = 1:size(testData,2)
    plot((testData(i).cycles(100).Qdlin - testData(i).cycles(10).Qdlin),
testData(i).Vdlin)
end
ylabel('Voltage (V)'); xlabel('Q_{100} - Q_{10} (Ah)');


figure;
trainDataSize = size(trainData,2);
cycleLife = zeros(trainDataSize,1);
DeltaQ_var = zeros(trainDataSize,1);
for i = 1:trainDataSize
    cycleLife(i) = size(trainData(i).cycles,2) + 1;
    DeltaQ_var(i) = var(trainData(i).cycles(100).Qdlin -
trainData(i).cycles(10).Qdlin);
end
loglog(DeltaQ_var,cycleLife, 'o')
ylabel('Cycle life'); xlabel('Var(Q_{100} - Q_{10}) (Ah^2)');


[XTrain,yTrain] = helperGetFeatures(trainData);
head(XTrain)


rng("default")

alphaVec = 0.01:0.1:1;
lambdaVec = 0:0.01:1;
MCReps = 1;
cvFold = 4;

rmseList = zeros(length(alphaVec),1);
minLambdaMSE = zeros(length(alphaVec),1);
wModelList = cell(1,length(alphaVec));
betaVec = cell(1,length(alphaVec));


for i=1:length(alphaVec)
    [coefficients,fitInfo] = ...
```

```
lasso(XTrain.Variables,yTrain,'Alpha',alphaVec(i),'CV',cvFold,'MCReps',MCReps,'Lam
bda',lambdaVec);
     betaVec{i} = fitInfo.Intercept(fitInfo.IndexMinMSE);
    indexMinMSE = fitInfo.IndexMinMSE;
    rmseList(i) = sqrt(fitInfo.MSE(indexMinMSE));
    minLambdaMSE(i) = fitInfo.LambdaMinMSE;
end
numVal = 4;
[out,idx] = sort(rmseList);
val = out(1:numVal);
index = idx(1:numVal);
alpha = alphaVec(index);
lambda = minLambdaMSE(index);
wModel = wModelList(index);
beta = betaVec(index);



[XVal,yVal] = helperGetFeatures(valData);
rmseValModel = zeros(numVal);


for valList = 1:numVal
   yPredVal = (XVal.Variables*wModel{valList} + beta{valList});
   rmseValModel(valList) = sqrt(mean((yPredVal-yVal).^2));
end
[rmseMinVal,idx] = min(rmseValModel);
wModelFinal = wModel{idx};
betaFinal = beta{idx};


[XTest,yTest] = helperGetFeatures(testData);
yPredTest = (XTest.Variables*wModelFinal + betaFinal);

figure;
scatter(yTest,yPredTest)
hold on;
refline(1, 0);
title('Predicted vs Actual Cycle Life')
ylabel('Predicted cycle life');xlabel('Actual cycle life');



errTest = (yPredTest-yTest);
rmseTestModel = sqrt(mean(errTest.^2))
n = numel(yTest);
nr = abs(yTest - yPredTest);
errVal = (1/n)*sum(nr./yTest)*100
function [xTable, y] = helperGetFeatures(batch)



N = size(batch,2);
y = zeros(N, 1);
DeltaQ_var = zeros(N,1);
DeltaQ_min = zeros(N,1);
CapFadeCycle2Slope = zeros(N,1);
```

```
CapFadeCycle2Intercept = zeros(N,1);
Qd2 = zeros(N,1);
AvgChargeTime = zeros(N,1);
IntegralTemp = zeros(N,1);
MinIR = zeros(N,1);
IRDiff2And100 = zeros(N,1);

for i = 1:N

    y(i,1) = size(batch(i).cycles,2) + 1;

    %
    numCycles = size(batch(i).cycles, 2);
    timeGapCycleIdx = [];
    jBadCycle = 0;
    for jCycle = 2: numCycles
        dt = diff(batch(i).cycles(jCycle).t);
        if max(dt) > 5*mean(dt)
            jBadCycle = jBadCycle + 1;
            timeGapCycleIdx(jBadCycle) = jCycle;
        end
    end
    batch(i).cycles(timeGapCycleIdx) = [];
    batch(i).summary.QDischarge(timeGapCycleIdx) = [];
    batch(i).summary.IR(timeGapCycleIdx) = [];
    batch(i).summary.chargetime(timeGapCycleIdx) = [];


    DeltaQ = batch(i).cycles(100).Qdlin - batch(i).cycles(10).Qdlin;
    DeltaQ_var(i) = log10(abs(var(DeltaQ)));
    DeltaQ_min(i) = log10(abs(min(DeltaQ)));



    coeff2 = polyfit(batch(i).summary.cycle(2:100),
batch(i).summary.QDischarge(2:100),1);
    CapFadeCycle2Slope(i) = coeff2(1);
    CapFadeCycle2Intercept(i) = coeff2(2);

    %  Discharge capacity at cycle 2
    Qd2(i) = batch(i).summary.QDischarge(2);

    % Avg charge time, first 5 cycles (2 to 6)
    AvgChargeTime(i) = mean(batch(i).summary.chargetime(2:6));

    % Integral of temperature from cycles 2 to 100
    tempIntT = 0;
    for jCycle = 2:100
        tempIntT = tempIntT + trapz(batch(i).cycles(jCycle).t,
batch(i).cycles(jCycle).T);
    end
    IntegralTemp(i) = tempIntT;

    % Minimum internal resistance, cycles 2 to 100
    temp = batch(i).summary.IR(2:100);
    MinIR(i) = min(temp(temp~=0));
    IRDiff2And100(i) = batch(i).summary.IR(100) - batch(i).summary.IR(2);
end
```

```
xTable = table(DeltaQ_var, DeltaQ_min, ...
    CapFadeCycle2Slope, CapFadeCycle2Intercept,...
    Qd2, AvgChargeTime, MinIR, IRDiff2And100);

end
```





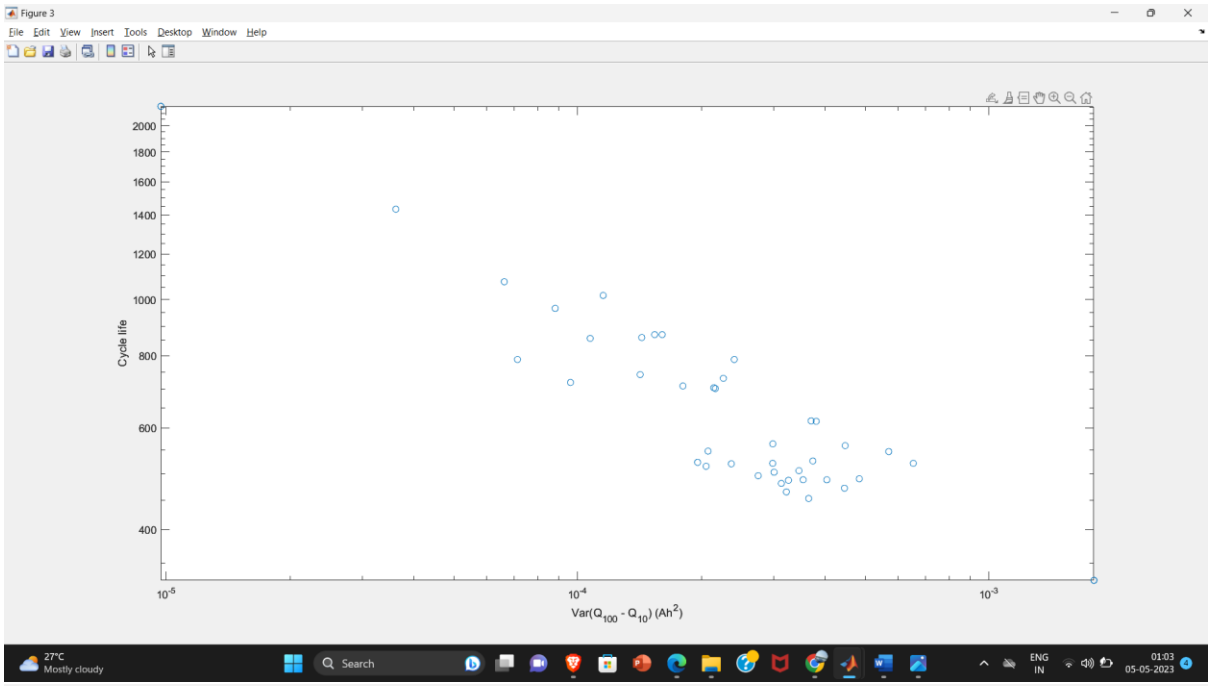


Figure 3
File Edit View Insert Tools Desktop Window Help
2000
1800
1600
1400
1200
1000
800
600
400
Cycle life
10^{-5}
10^{-4}
10^{-3}
$Var(Q_{100} - Q_{10})$ (Ah^2)
27°C
Mostly cloudy
Search
ENG
IN
01:03
05-05-2023

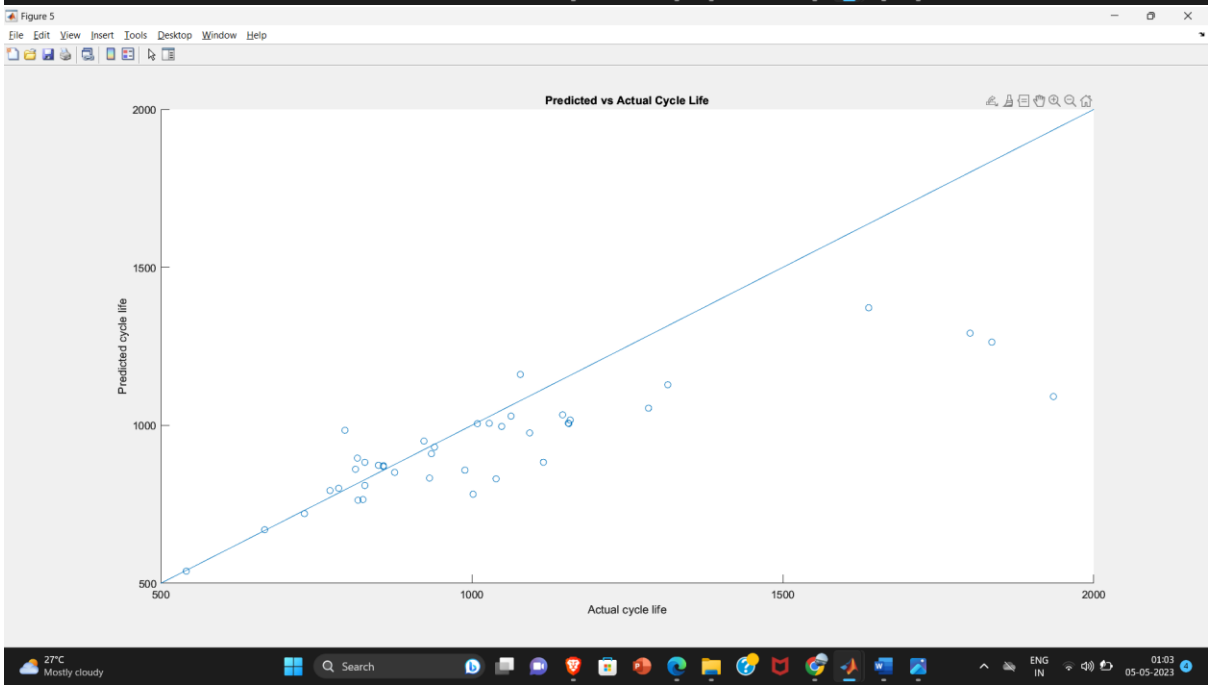


Figure 5
File Edit View Insert Tools Desktop Window Help
Predicted vs Actual Cycle Life
2000
1500
1000
500
Predicted cycle life
500
1000
1500
2000
Actual cycle life
27°C
Mostly cloudy
Search
ENG
IN
01:03
05-05-2023