```
clc
clear

filename = 'Liquid-Data_dat.txt';
formatSpec = '%4f%13f%8f%8f%8f%8f%f%[^\n\r]';
fileID = fopen(filename, 'r');
LiquidData = textscan(fileID, formatSpec, 'Delimiter', '', 'WhiteSpace', '',
'ReturnOnError', false);
fclose(fileID);

Data = [LiquidData{1}, LiquidData{2}, LiquidData{3}, LiquidData{4}, LiquidData{5},
LiquidData{6}, LiquidData{7}];
Data = [Data, zeros(numel(Data(:, 1)), 1)];
D = numel(Data(1, 2:end-1));

[rowC1, ~] = find(Data(:, 1) == 1);
[rowC2, ~] = find(Data(:, 1) == 2);
[rowC3, ~] = find(Data(:, 1) == 3);
distance = zeros(max([numel(rowC1), numel(rowC2), numel(rowC3)]), 3);
distance(:) = inf;

nTPC11 = 0;
nFNC1 = 0;
nFPC12 = 0;
nFPC13 = 0;

nTPC22 = 0;
nFNC2 = 0;
nFPC21 = 0;
nFPC23 = 0;

nTPC33 = 0;
nFNC3 = 0;
nFPC31 = 0;
nFPC32 = 0;

class1 = [Data(rowC1, 2), Data(rowC1, 3), Data(rowC1, 4), Data(rowC1, 5), Data(rowC1,
6), Data(rowC1, 7)];
class2 = [Data(rowC2, 2), Data(rowC2, 3), Data(rowC2, 4), Data(rowC2, 5), Data(rowC2,
6), Data(rowC2, 7)];
class3 = [Data(rowC3, 2), Data(rowC3, 3), Data(rowC3, 4), Data(rowC3, 5), Data(rowC3,
6), Data(rowC3, 7)];
N1 = numel(class1(:, 1));
N2 = numel(class2(:, 1));
N3 = numel(class3(:, 1));

for k = 1:3
    for i = min(rowC1):max(rowC1)
        class1_LOO = [Data(setdiff(rowC1, i), 2), Data(setdiff(rowC1, i), 3),
Data(setdiff(rowC1, i), 4), Data(setdiff(rowC1, i), 5), Data(setdiff(rowC1, i), 6),
Data(setdiff(rowC1, i), 7)];
        DataPoint = [Data(i, 2), Data(i, 3), Data(i, 4), Data(i, 5), Data(i, 6),
Data(i, 7)];
```

```
        for j = 1:numel(class1_LOO(:, 1))
            distance(j + 1, 1) = euclideanDistance(DataPoint, class1_LOO(j, :));
        end
        for j = 1:numel(class2(:, 1))
            distance(j, 2) = euclideanDistance(DataPoint, class2(j, :));
        end
        for j = 1:numel(class3(:, 1))
            distance(j, 3) = euclideanDistance(DataPoint, class3(j, :));
        end
        sortedDistance = sort(distance);

        A = sortedDistance(1:k, 1);
        kA = max(numel(find(A(end) == sortedDistance(:, 1))) + numel(find(A(end) ~=
A(:))), k);
        B = sortedDistance(1:k, 2);
        kB = max(numel(find(B(end) == sortedDistance(:, 2))) + numel(find(B(end) ~=
B(:))), k);
        C = sortedDistance(1:k, 3);
        kC = max(numel(find(C(end) == sortedDistance(:, 3))) + numel(find(C(end) ~=
C(:))), k);

        V1 = Vn(D, max(A));
        V2 = Vn(D, max(B));
        V3 = Vn(D, max(C));
        N1_LOO = numel(class1_LOO(:, 1));

        l12 = (kA * N2 * V2) / (kB * N1_LOO * V1);
        l32 = (kC * N2 * V2) / (kB * N3 * V3);
        l13 = (kA * N3 * V3) / (kC * N1_LOO * V1);

        pC1 = N1_LOO / (N1_LOO + N2 + N3);
        pC2 = N2 / (N1_LOO + N2 + N3);
        pC3 = N3 / (N1_LOO + N2 + N3);

        [~, index1] = max([l12, pC2 / pC1]);
        if index1 == 1
            [~, index3] = max([l13, pC3 / pC1]);
            if index3 == 1
                TrueClass = 1;
            else
                TrueClass = 3;
            end
        else
            [~, index2] = max([l32, pC2 / pC3]);
            if index2 == 1
                TrueClass = 3;
            else
                TrueClass = 2;
            end
        end
        Data(i, 8) = TrueClass;
        if TrueClass ~= 1
            nFNC1 = nFNC1 + 1;
            switch TrueClass
                case 2
```

```
                    nFPC12 = nFPC12 + 1;
                case 3
                    nFPC13 = nFPC13 + 1;
            end
            display('------------------');
            display(['Row Number: ', num2str(i)]);
            display(['Actual Class: ', num2str(Data(i, 1))]);
            display(['Predicted Class: ', num2str(TrueClass)]);
            display('------------------');
        end
    end
    nTPC11 = numel(rowC1) - nFNC1;

    distance(:) = inf;
    for i = min(rowC2):max(rowC2)
        class2_LOO = [Data(setdiff(rowC2, i), 2), Data(setdiff(rowC2, i), 3),
Data(setdiff(rowC2, i), 4), Data(setdiff(rowC2, i), 5), Data(setdiff(rowC2, i), 6),
Data(setdiff(rowC2, i), 7)];
        DataPoint = [Data(i, 2), Data(i, 3), Data(i, 4), Data(i, 5), Data(i, 6),
Data(i, 7)];
        for j = 1:numel(class1(:, 1))
            distance(j, 1) = euclideanDistance(DataPoint, class1(j, :));
        end
        for j = 1:numel(class2_LOO(:, 1))
            distance(j + 1, 2) = euclideanDistance(DataPoint, class2_LOO(j, :));
        end
        for j = 1:numel(class3(:, 1))
            distance(j, 3) = euclideanDistance(DataPoint, class3(j, :));
        end
        sortedDistance = sort(distance);

        A = sortedDistance(1:k, 1);
        kA = max(numel(find(A(end) == sortedDistance(:, 1))) + numel(find(A(end) ~=
A(:))), k);
        B = sortedDistance(1:k, 2);
        kB = max(numel(find(B(end) == sortedDistance(:, 2))) + numel(find(B(end) ~=
B(:))), k);
        C = sortedDistance(1:k, 3);
        kC = max(numel(find(C(end) == sortedDistance(:, 3))) + numel(find(C(end) ~=
C(:))), k);

        V1 = Vn(D, max(A));
        V2 = Vn(D, max(B));
        V3 = Vn(D, max(C));
        N2_LOO = numel(class2_LOO(:, 1));

        l12 = (kA * N2_LOO * V2) / (kB * N1 * V1);
        l32 = (kC * N2_LOO * V2) / (kB * N3 * V3);
        l13 = (kA * N3 * V3) / (kC *N1 * V1);

        pC1 = N1 / (N1 + N2_LOO + N3);
        pC2 = N2_LOO / (N1 + N2_LOO + N3);
        pC3 = N3 / (N1 + N2_LOO + N3);

        [~, index1] = max([l12, pC2 / pC1]);
```

```
        if index1 == 1
            [~, index3] = max([l13, pC3 / pC1]);
            if index3 == 1
                TrueClass = 1;
            else
                TrueClass = 3;
            end
        else
            [~, index2] = max([l32, pC2 / pC3]);
            if index2 == 1
                TrueClass = 3;
            else
                TrueClass = 2;
            end
        end
        Data(i, 8) = TrueClass;
        if TrueClass ~= 2
            nFNC2 = nFNC2 + 1;
            switch TrueClass
                case 1
                    nFPC21 = nFPC21 + 1;
                case 3
                    nFPC23 = nFPC23 + 1;
            end
            display('------------------');
            display(['Row Number: ', num2str(i - max(rowC1))]);
            display(['Actual Class: ', num2str(Data(i, 1))]);
            display(['Predicted Class: ', num2str(TrueClass)]);
            display('------------------');
        end
    end
    nTPC22 = numel(rowC2) - nFNC2;

    distance(:) = inf;
    for i = min(rowC3):max(rowC3)
        class3_LOO = [Data(setdiff(rowC3, i), 2), Data(setdiff(rowC3, i), 3),
Data(setdiff(rowC3, i), 4), Data(setdiff(rowC3, i), 5), Data(setdiff(rowC3, i), 6),
Data(setdiff(rowC3, i), 7)];
        DataPoint = [Data(i, 2), Data(i, 3), Data(i, 4), Data(i, 5), Data(i, 6),
Data(i, 7)];
        for j = 1:numel(class1(:, 1))
            distance(j, 1) = euclideanDistance(DataPoint, class1(j, :));
        end
        for j = 1:numel(class2(:, 1))
            distance(j, 2) = euclideanDistance(DataPoint, class2(j, :));
        end
        for j = 1:numel(class3_LOO(:, 1))
            distance(j + 1, 3) = euclideanDistance(DataPoint, class3_LOO(j, :));
        end
        sortedDistance = sort(distance);

        A = sortedDistance(1:k, 1);
        kA = max(numel(find(A(end) == sortedDistance(:, 1))) + numel(find(A(end) ~=
A(:))), k);
        B = sortedDistance(1:k, 2);
```

```
        kB = max(numel(find(B(end) == sortedDistance(:, 2))) + numel(find(B(end) ~=
B(:))), k);
        C = sortedDistance(1:k, 3);
        kC = max(numel(find(C(end) == sortedDistance(:, 3))) + numel(find(C(end) ~=
C(:))), k);

        V1 = Vn(D, max(A));
        V2 = Vn(D, max(B));
        V3 = Vn(D, max(C));
        N3_LOO = numel(class3_LOO(:, 1));

        l12 = (kA * N2 * V2) / (kB * N1 * V1);
        l32 = (kC * N2 * V2) / (kB * N3_LOO * V3);
        l13 = (kA * N3_LOO * V3) / (kC * N1 * V1);

        pC1 = N1 / (N1 + N2 + N3_LOO);
        pC2 = N2 / (N1 + N2 + N3_LOO);
        pC3 = N3_LOO / (N1 + N2 + N3_LOO);

        [~, index1] = max([l12, pC2 / pC1]);
        if index1 == 1
            [~, index3] = max([l13, pC3 / pC1]);
            if index3 == 1
                TrueClass = 1;
            else
                TrueClass = 3;
            end
        else
            [~, index2] = max([l32, pC2 / pC3]);
            if index2 == 1
                TrueClass = 3;
            else
                TrueClass = 2;
            end
        end
        Data(i, 8) = TrueClass;
        if TrueClass ~= 3
            nFNC3 = nFNC3 + 1;
            switch TrueClass
                case 1
                    nFPC31 = nFPC31 + 1;
                case 2
                    nFPC32 = nFPC32 + 1;
            end
            display('------------------');
            display(['Row Number: ', num2str(i - max(rowC2))]);
            display(['Actual Class: ', num2str(Data(i, 1))]);
            display(['Predicted Class: ', num2str(TrueClass)]);
            display('------------------');
        end
    end
    nTPC33 = numel(rowC3) - nFNC3;

    display(['k = ', num2str(k)]);
```

```
    Confusion_matrix{1, 1} = 'P\A';
    Confusion_matrix{2, 1} = 'C1';
    Confusion_matrix{3, 1} = 'C2';
    Confusion_matrix{4, 1} = 'C3';
    Confusion_matrix{1, 2} = 'C1';
    Confusion_matrix{1, 3} = 'C2';
    Confusion_matrix{1, 4} = 'C3';

    Confusion_matrix{2, 2} = nTPC11;
    Confusion_matrix{2, 3} = nFPC21;
    Confusion_matrix{2, 4} = nFPC31;

    Confusion_matrix{3, 2} = nFPC12;
    Confusion_matrix{3, 3} = nTPC22;
    Confusion_matrix{3, 4} = nFPC32;

    Confusion_matrix{4, 2} = nFPC13;
    Confusion_matrix{4, 3} = nFPC23;
    Confusion_matrix{4, 4} = nTPC33;

    display(Confusion_matrix);

    nTPC11 = 0;
    nFNC1 = 0;
    nFPC12 = 0;
    nFPC13 = 0;

    nTPC22 = 0;
    nFNC2 = 0;
    nFPC21 = 0;
    nFPC23 = 0;

    nTPC33 = 0;
    nFNC3 = 0;
    nFPC31 = 0;
    nFPC32 = 0;
end
------------------
Row Number: 3
Actual Class: 2
Predicted Class: 3
------------------
------------------
Row Number: 4
Actual Class: 2
Predicted Class: 3
------------------
------------------
Row Number: 15
Actual Class: 2
Predicted Class: 1
------------------
------------------
Row Number: 23
Actual Class: 2
```

```
Predicted Class: 1
------------------
------------------
Row Number: 25
Actual Class: 2
Predicted Class: 3
------------------
------------------
Row Number: 7
Actual Class: 3
Predicted Class: 2
------------------
------------------
Row Number: 16
Actual Class: 3
Predicted Class: 2
------------------
k = 1

Confusion_matrix =

  4×4 cell array

    {'P\A'}    {'C1'}    {'C2'}    {'C3'}
    {'C1' }    {[59]}    {[ 2]}    {[ 0]}
    {'C2' }    {[ 0]}    {[66]}    {[ 2]}
    {'C3' }    {[ 0]}    {[ 3]}    {[46]}

------------------
Row Number: 44
Actual Class: 1
Predicted Class: 2
------------------
------------------
Row Number: 3
Actual Class: 2
Predicted Class: 3
------------------
------------------
Row Number: 4
Actual Class: 2
Predicted Class: 3
------------------
------------------
Row Number: 7
Actual Class: 2
Predicted Class: 1
------------------
------------------
Row Number: 15
Actual Class: 2
Predicted Class: 1
------------------
------------------
Row Number: 23
```

```
Actual Class: 2
Predicted Class: 1
------------------
------------------
Row Number: 25
Actual Class: 2
Predicted Class: 3
------------------
------------------
Row Number: 16
Actual Class: 3
Predicted Class: 2
------------------
k = 2

Confusion_matrix =

  4×4 cell array

    {'P\A'}    {'C1'}    {'C2'}    {'C3'}
    {'C1' }    {[58]}    {[ 3]}    {[ 0]}
    {'C2' }    {[ 1]}    {[65]}    {[ 1]}
    {'C3' }    {[ 0]}    {[ 3]}    {[47]}

------------------
Row Number: 44
Actual Class: 1
Predicted Class: 2
------------------
------------------
Row Number: 3
Actual Class: 2
Predicted Class: 3
------------------
------------------
Row Number: 7
Actual Class: 2
Predicted Class: 1
------------------
------------------
Row Number: 15
Actual Class: 2
Predicted Class: 1
------------------
------------------
Row Number: 23
Actual Class: 2
Predicted Class: 1
------------------
------------------
Row Number: 25
Actual Class: 2
Predicted Class: 3
------------------
------------------
```

```
Row Number: 16
Actual Class: 3
Predicted Class: 2
------------------
k = 3

Confusion_matrix =

  4×4 cell array

    {'P\A'}    {'C1'}    {'C2'}    {'C3'}
    {'C1' }    {[58]}    {[ 3]}    {[ 0]}
    {'C2' }    {[ 1]}    {[66]}    {[ 1]}
    {'C3' }    {[ 0]}    {[ 2]}    {[47]}
```