IoT Challenging Task V

Suryakumar P 21MIS1146

Consider a sample dataset that have 8 data points representing temperature and humidity readings from environmental sensors. Perform K-means clustering algorithm for the given data set using Absolute Distance. Display the sample output in node red browser as 2d- chart. Consider there are three clusters for the given dataset.

**Aim:**

To perform K-means clustering algorithm for the given data set using Absolute Distance and display the sample output in node red browser as 2d-chart.

**Procedure:**

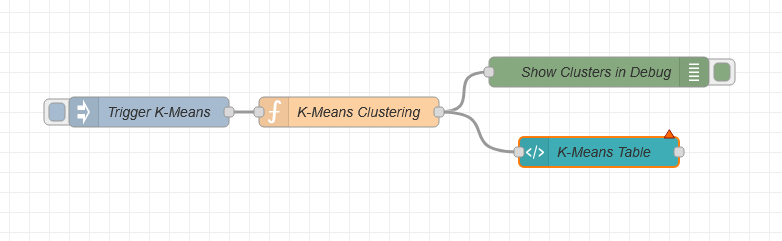
1) Inject Node – Triggers the flow manually or at intervals.

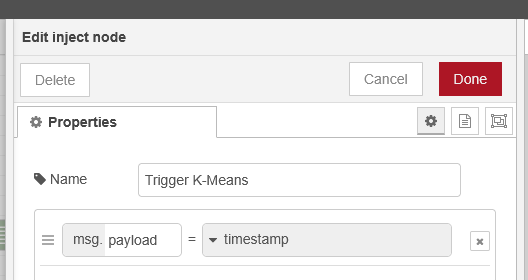
2) Function Node – Add the K-Means clustering logic (from the modified code) inside this node. It will process the temperature and humidity data and assign clusters (Low, Medium, High).

3) Debug Node – Connect it to the Function Node to verify the clustered output in the debug window.

4) Dashboard Chart Node – Add a Dashboard "chart" node (set to scatter plot) to visualize temperature vs. humidity data. Map msg.payload to the chart data.

NodeRED:





**Function Node:**

function manhattanDistance(point1, point2) {

    return Math.abs(point1.temperature - point2.temperature) +

           Math.abs(point1.humidity - point2.humidity);

}

function kMeansClustering(data, k) {

    let centroids = data.slice(0, k);

    let clusters = new Array(data.length).fill(-1);

    let changed = true;

    const labels = ["Low", "Medium", "High"];

    while (changed) {

        changed = false;

        // Assign points to the nearest centroid

        for (let i = 0; i < data.length; i++) {

            let minDist = Infinity, cluster = -1;

            for (let j = 0; j < k; j++) {

                let dist = manhattanDistance(data[i], centroids[j]);

                if (dist < minDist) {

                    minDist = dist;

                    cluster = j;

                }

            }

            if (clusters[i] !== cluster) {

                clusters[i] = cluster;

                changed = true;

            }

        }

        // Compute new centroids

        let newCentroids = Array.from({ length: k }, () => ({ temperature: 0, humidity: 0, count: 0 }));

        for (let i = 0; i < data.length; i++) {

            newCentroids[clusters[i]].temperature += data[i].temperature;

            newCentroids[clusters[i]].humidity += data[i].humidity;

            newCentroids[clusters[i]].count++;

        }

        // Update centroids

        for (let j = 0; j < k; j++) {

            if (newCentroids[j].count > 0) {

                centroids[j] = {

                    temperature: newCentroids[j].temperature / newCentroids[j].count,

                    humidity: newCentroids[j].humidity / newCentroids[j].count

                };

            }

        }

    }

    return data.map((point, index) => ({

        ...point,

        cluster: labels[clusters[index]]  // Replace 0,1,2 with Low, Medium, High

    }));

}

let data = [

    { temperature: 22, humidity: 60 },

    { temperature: 20, humidity: 55 },

    { temperature: 25, humidity: 65 },

    { temperature: 18, humidity: 50 },

    { temperature: 30, humidity: 70 },

    { temperature: 21, humidity: 58 },

    { temperature: 23, humidity: 62 },

    { temperature: 26, humidity: 68 }

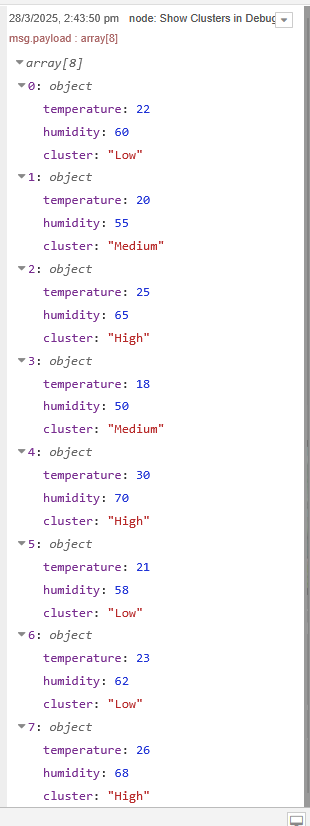
];

let clusteredData = kMeansClustering(data, 3);

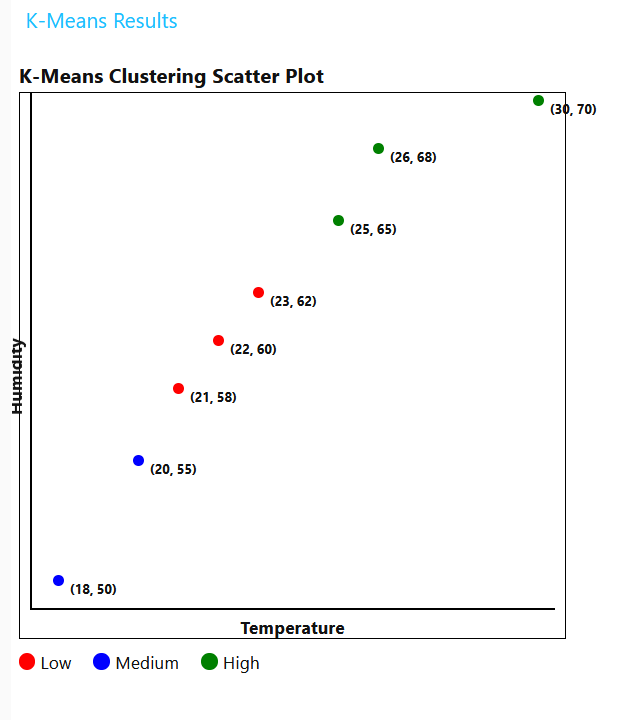
msg.payload = clusteredData;

return msg;

**Output:**

**Debug Node:**  


**NodeRed Dashboard:**



**JSON Code:**

[

{

"id": "8715f4448cc4fc99",

"type": "tab",

"label": "Flow 2",

"disabled": false,

"info": "",

"env": []

},

{

"id": "5d996d53fcae5d63",

"type": "inject",

"z": "8715f4448cc4fc99",

"name": "Trigger K-Means",

"props": [

{ "p": "payload"}

],

"repeat": "",

"crontab": "",

"once": false,

"onceDelay": 0.1,

"topic": "",

"payload": "",

"payloadType": "date",

"x": 360,

"y": 220,

"wires": [

[

"1168491c19a76d35"

]

]

},

{

"id": "1168491c19a76d35",

"type": "function",

"z": "8715f4448cc4fc99",

"name": "K-Means Clustering",

"func": "function manhattanDistance(point1, point2) {\n return Math.abs(point1.temperature - point2.temperature) +\n Math.abs(point1.humidity - point2.humidity);\n}\n\nfunction kMeansClustering(data, k) {\n let centroids = data.slice(0, k);\n let clusters = new Array(data.length).fill(-1);\n let changed = true;\n const labels = [\"Low\", \"Medium\", \"High\"];\n\n while (changed) {\n changed = false;\n\n // Assign points to the nearest centroid\n for (let i = 0; i < data.length; i++) {\n let minDist = Infinity, cluster = -1;\n for (let j = 0; j < k; j++) {\n let dist = manhattanDistance(data[i], centroids[j]);\n if (dist < minDist) {\n minDist = dist;\n cluster = j;\n }\n }\n if (clusters[i] !== cluster) {\n clusters[i] = cluster;\n changed = true;\n }\n }\n \n // Compute new centroids\n let newCentroids = Array.from({ length: k }, () => ({ temperature: 0, humidity: 0, count: 0 }));\n for (let i = 0; i < data.length; i++) {\n newCentroids[clusters[i]].temperature += data[i].temperature;\n newCentroids[clusters[i]].humidity += data[i].humidity;\n newCentroids[clusters[i]].count++;\n }\n \n // Update centroids\n for (let j = 0; j < k; j++) {\n if (newCentroids[j].count > 0) {\n centroids[j] = {\n temperature: newCentroids[j].temperature / newCentroids[j].count,\n humidity: newCentroids[j].humidity / newCentroids[j].count\n };\n }\n }\n }\n\n return data.map((point, index) => ({ \n ...point, \n cluster: labels[clusters[index]] // Replace 0,1,2 with Low, Medium, High\n }));\n}\n\nlet data = [\n { temperature: 22, humidity: 60 },\n { temperature: 20, humidity: 55 },\n { temperature: 25, humidity: 65 },\n { temperature: 18, humidity: 50 },\n { temperature: 30, humidity: 70 },\n { temperature: 21, humidity: 58 },\n { temperature: 23, humidity: 62 },\n { temperature: 26, humidity: 68 }\n];\n\nlet clusteredData = kMeansClustering(data, 3);\nmsg.payload = clusteredData;\nreturn msg;\n",

"outputs": 1,

"timeout": "",

"noerr": 0,

"initialize": "",

"finalize": "",

"libs": [],

"x": 560,

"y": 220,

"wires": [

[

"39f0dd698cfba47e",

"dce28dea698f016e"

]

]

},

{

"id": "39f0dd698cfba47e",

"type": "debug",

"z": "8715f4448cc4fc99",

"name": "Show Clusters in Debug",

"active": true,

"tosidebar": true,

"console": false,

"tostatus": false,

"complete": "payload",

"targetType": "msg",

"statusVal": "",

"statusType": "auto",

"x": 810,

"y": 180,

"wires": []

},

{

"id": "dce28dea698f016e",

"type": "ui\_template",

"z": "8715f4448cc4fc99",

"group": "8fb3b77c5d35bb88",

"name": "K-Means Table",

"order": 0,

"width": "40",

"height": "40",

"format": "<!DOCTYPE html>\n<html lang=\"en\">\n\n<head>\n <meta charset=\"UTF-8\">\n <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\n <title>K-Means Clustering Scatter Plot</title>\n <style>\n .chart-container {\n width: 500px;\n height: 500px;\n border: 1px solid black;\n position: relative;\n display: flex;\n flex-direction: column;\n align-items: center;\n }\n\n .chart {\n width: 480px;\n height: 480px;\n position: relative;\n border-left: 2px solid black;\n border-bottom: 2px solid black;\n }\n\n .point {\n position: absolute;\n width: 10px;\n height: 10px;\n border-radius: 50%;\n cursor: pointer;\n }\n\n .point-label {\n position: absolute;\n font-size: 12px;\n font-weight: bold;\n white-space: nowrap;\n }\n\n .x-axis-label {\n text-align: center;\n margin-top: 5px;\n font-weight: bold;\n }\n\n .y-axis-label {\n position: absolute;\n left: -40px;\n top: 50%;\n transform: rotate(-90deg);\n font-weight: bold;\n }\n\n .legend {\n margin-top: 10px;\n display: flex;\n gap: 20px;\n }\n\n .legend-item {\n display: flex;\n align-items: center;\n gap: 5px;\n }\n\n .legend-color {\n width: 15px;\n height: 15px;\n border-radius: 50%;\n }\n </style>\n</head>\n\n<body>\n <h3>K-Means Clustering Scatter Plot</h3>\n <div class=\"chart-container\">\n <div class=\"y-axis-label\">Humidity</div>\n <div class=\"chart\" id=\"chart\"></div>\n <div class=\"x-axis-label\">Temperature</div>\n </div>\n\n <div class=\"legend\">\n <div class=\"legend-item\">\n <div class=\"legend-color\" style=\"background-color: red;\"></div> Low\n </div>\n <div class=\"legend-item\">\n <div class=\"legend-color\" style=\"background-color: blue;\"></div> Medium\n </div>\n <div class=\"legend-item\">\n <div class=\"legend-color\" style=\"background-color: green;\"></div> High\n </div>\n </div>\n\n <script>\n (function(scope) {\n scope.$watch('msg.payload', function(data) {\n if (!data || !Array.isArray(data)) return;\n\n let chart = document.getElementById(\"chart\");\n chart.innerHTML = \"\"; // Clear previous points\n\n let clusterColors = { \"Low\": \"red\", \"Medium\": \"blue\", \"High\": \"green\" };\n \n let minX = Math.min(...data.map(d => d.temperature));\n let maxX = Math.max(...data.map(d => d.temperature));\n let minY = Math.min(...data.map(d => d.humidity));\n let maxY = Math.max(...data.map(d => d.humidity));\n\n let padding = 20; // Add padding to prevent edge overlap\n\n function scale(value, min, max, range) {\n return padding + ((value - min) / (max - min)) \* (range - 2 \* padding);\n }\n\n data.forEach(point => {\n let dot = document.createElement(\"div\");\n dot.className = \"point\";\n dot.style.backgroundColor = clusterColors[point.cluster];\n\n // Positioning within bounds\n dot.style.left = scale(point.temperature, minX, maxX, 480) + \"px\";\n dot.style.bottom = scale(point.humidity, minY, maxY, 480) + \"px\";\n\n let label = document.createElement(\"div\");\n label.className = \"point-label\";\n label.innerText = `(${point.temperature}, ${point.humidity})`;\n \n // Offset to prevent overlap\n label.style.left = scale(point.temperature, minX, maxX, 480) + 15 + \"px\";\n label.style.bottom = scale(point.humidity, minY, maxY, 480) - 10 + \"px\";\n\n chart.appendChild(dot);\n chart.appendChild(label);\n });\n });\n })(scope);\n </script>\n</body>\n\n</html>",

"storeOutMessages": true,

"fwdInMessages": true,

"resendOnRefresh": true,

"templateScope": "local",

"className": "",

"x": 810,

"y": 260,

"wires": [

[]

]

},

{

"id": "8fb3b77c5d35bb88",

"type": "ui\_group",

"name": "K-Means Results",

"tab": "b2ca583139859455",

"order": 1,

"disp": true,

"width": "20",

"collapse": false,

"className": ""

},

{

"id": "b2ca583139859455",

"type": "ui\_tab",

"name": "Dashboard",

"icon": "dashboard",

"order": 1

}

]