IoT LAB

CHALLENGING TASK – 5

Name: PANDUGA VENKATA JAYA SRIKANTH REDDY
Reg No: 21MIS1095

K-MEANS CLUSTERING

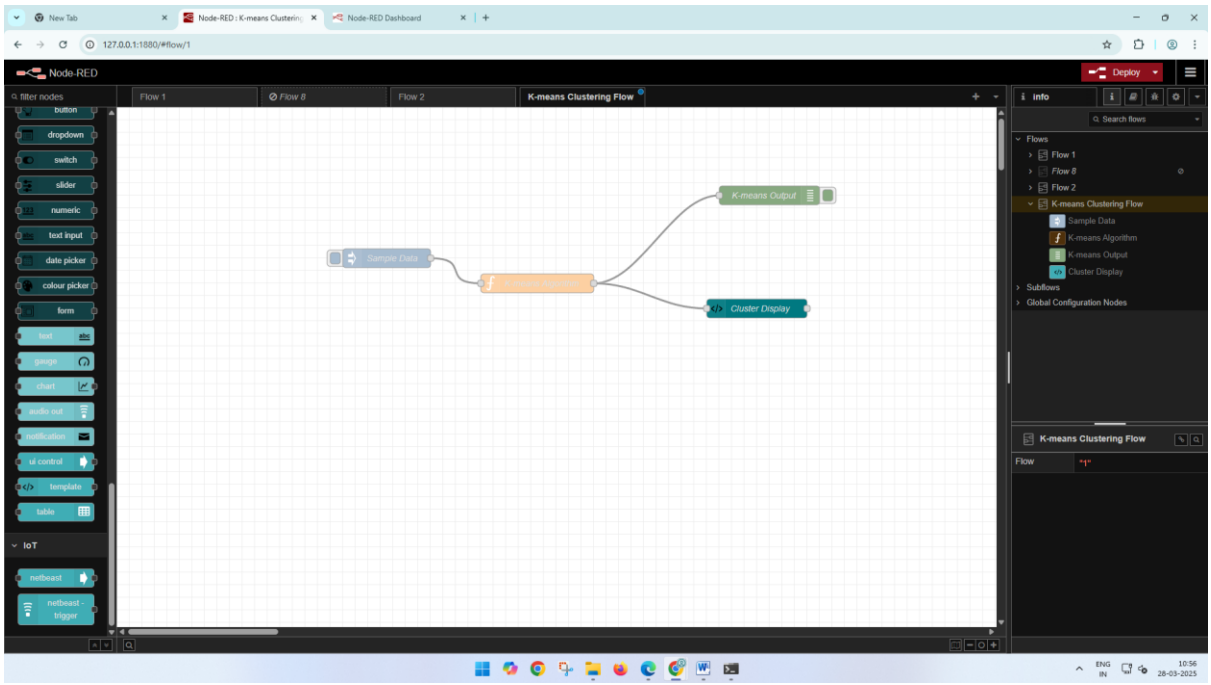
AIM:

To perform K-Means Clustering and Divide the given dataset into clusters

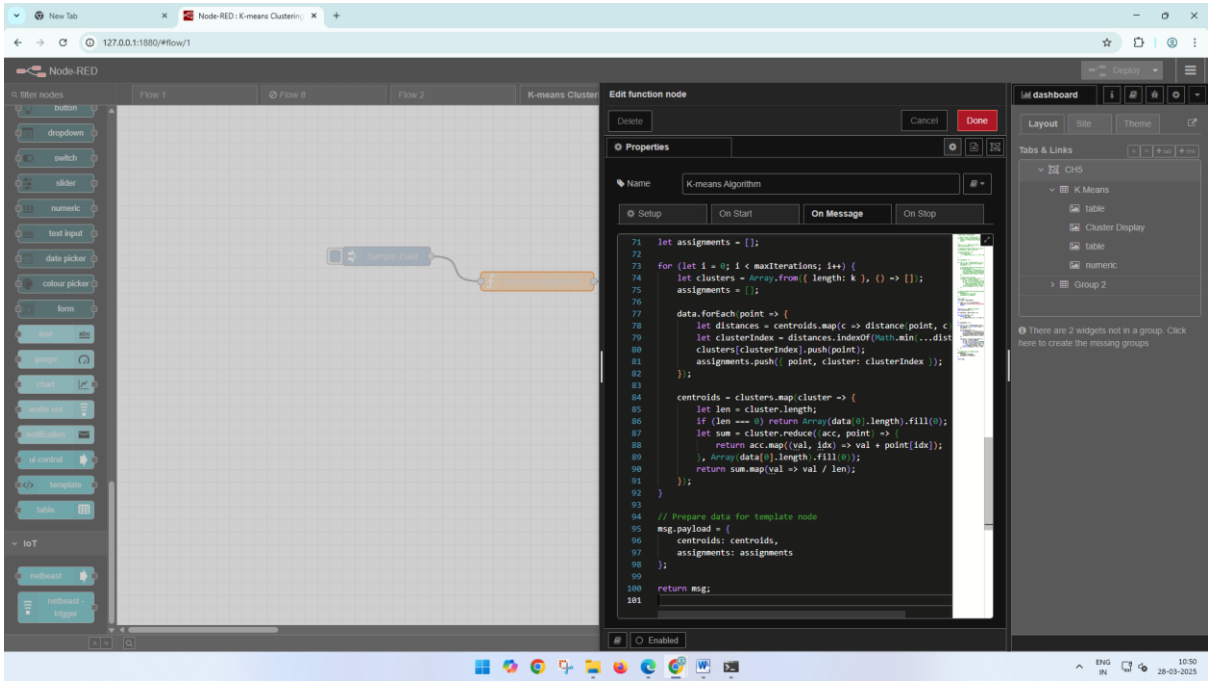
PROCEDURE:

- Initialize the clusters and number of iterations
- Finding the distances
- For each iteration starting from 0
- Calculate the distance of each point with cluster heads
- Find the min distance cluster
- Assign the point into that cluster
- Assign the cluster with cluster index
- Recalculate the new centroids
- Generate cluster groups and assignment groups for points

Node Red Flow



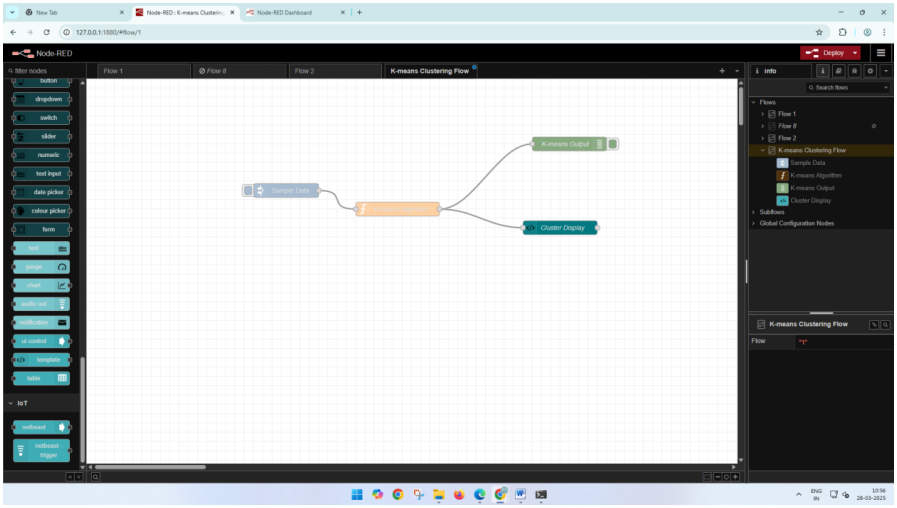
Function Node:

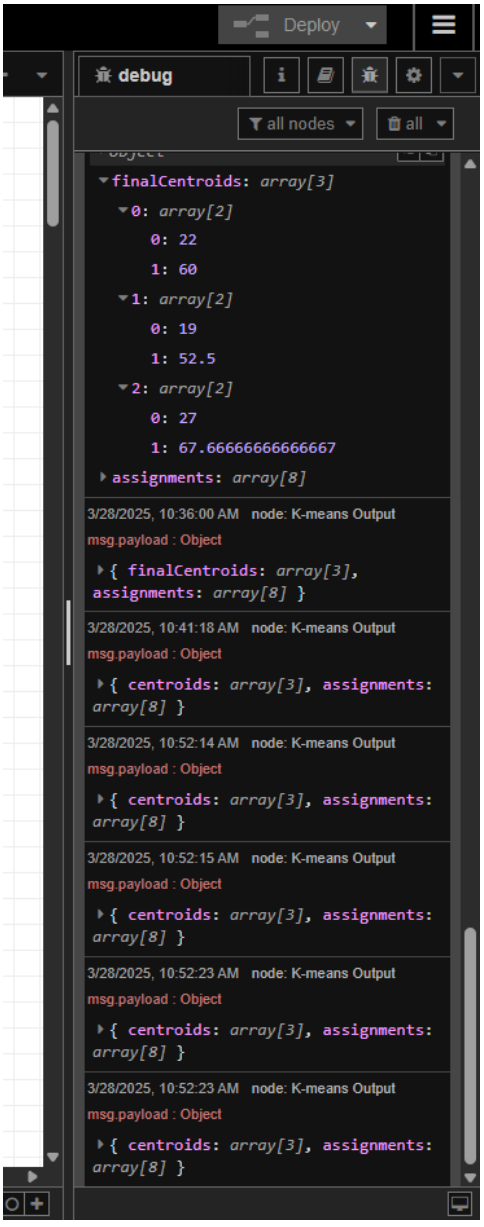


Function Node Code:

```
const data = msg.payload;

if (!data || !Array.isArray(data)) {
    node.error("Invalid data: Payload must be an array of points.");
    return;
}

const k = 3; // Number of clusters
const maxIterations = 10;

let centroids = data.slice(0, k);

function distance(a, b) {
    return Math.sqrt(
        a.reduce((sum, val, i) => sum + Math.pow(val - b[i], 2), 0)
    );
}

let assignments = [];

for (let i = 0; i < maxIterations; i++) {
    let clusters = Array.from({ length: k }, () => []);
    assignments = [];

    data.forEach(point => {
        let distances = centroids.map(c => distance(point, c));
        let clusterIndex = distances.indexOf(Math.min(...distances));
        clusters[clusterIndex].push(point);
        assignments.push({ point, cluster: clusterIndex });
    });

    centroids = clusters.map(cluster => {
        let len = cluster.length;
        if (len === 0) return Array(data[0].length).fill(0);
        let sum = cluster.reduce((acc, point) => {
            return acc.map((val, idx) => val + point[idx]);
        }, Array(data[0].length).fill(0));
        return sum.map(val => val / len);
    });
}
msg.payload = {
    centroids: centroids,
    assignments: assignments
};

return msg;
```

Output:


Node-RED
Deploy
K-means Clustering Flow
K-means Output
Cluster Display


Deploy
debug
all nodes
all
finalCentroids: array[3]
0: array[2]
0: 22
1: 60
1: array[2]
0: 19
1: 52.5
2: array[2]
0: 27
1: 67.66666666666667
assignments: array[8]
3/28/2025, 10:36:00 AM node: K-means Output
msg.payload : Object
{ finalCentroids: array[3], assignments: array[8] }
3/28/2025, 10:41:18 AM node: K-means Output
msg.payload : Object
{ centroids: array[3], assignments: array[8] }
3/28/2025, 10:52:14 AM node: K-means Output
msg.payload : Object
{ centroids: array[3], assignments: array[8] }
3/28/2025, 10:52:15 AM node: K-means Output
msg.payload : Object
{ centroids: array[3], assignments: array[8] }
3/28/2025, 10:52:23 AM node: K-means Output
msg.payload : Object
{ centroids: array[3], assignments: array[8] }
3/28/2025, 10:52:23 AM node: K-means Output
msg.payload : Object
{ centroids: array[3], assignments: array[8] }

JSON CODE:


```
[
  {
    "id": "1",
    "type": "tab",
    "label": "K-means Clustering Flow",
    "disabled": false,
    "info": ""
  },
  {
    "id": "2",
    "type": "inject",
    "z": "1",
    "name": "Sample Data",
    "props": [
      {
        "p": "payload"
      }
    ],
    "repeat": "",
    "crontab": "",
    "once": false,
    "onceDelay": 0.1,
    "topic": "",
    "payload": "[[22, 60], [20, 55], [25, 65], [18, 50], [30, 70], [21, 58], [23, 62], [26, 68]]",
    "payloadType": "json",
    "x": 430,
    "y": 240,
    "wires": [
      [
        "3"
      ]
    ]
  },
  {
    "id": "3",
    "type": "function",
    "z": "1",
    "name": "K-means Algorithm",
    "func": "/*const data = msg.payload;\n\n// Check if data is defined and is an array\nif (!data || !Array.isArray(data)) {\n    node.error(\"Invalid data: Payload must be an array of points.\");\n    return;\n}\n\nconst k = 3; // Number of clusters\nconst maxIterations = 10;\n\n// Randomly initialize centroids (first k points)\nlet centroids = data.slice(0, k);\n\nfunction distance(a, b) {\n    return Math.sqrt(\n        a.reduce((sum, val, i) => sum + Math.pow(val - b[i], 2), 0)\n    );\n}\n\nlet assignments = [];\n\nfor (let i = 0; i < maxIterations; i++) {\n    let clusters = Array.from({ length: k }, () => []);\n    assignments = [];\n\n    // Assign each point to the nearest centroid\n    data.forEach(point => {\n        let distances = centroids.map(c => distance(point, c));\n        let clusterIndex = distances.indexOf(Math.min(...distances));\n        clusters[clusterIndex].push(point);\n        assignments.push({ point, cluster: clusterIndex });\n    });\n\n    // Recalculate centroids\n    centroids = clusters.map(cluster => {\n        let len = cluster.length;\n        if (len === 0) return Array(data[0].length).fill(0); // Prevent NaN\n        let sum = cluster.reduce((acc, point) => {\n            return acc.map((val, idx) => val + point[idx]);\n        }, Array(data[0].length).fill(0));\n        return sum.map(val => val / len);\n    });\n}\n\n// Output includes final centroids and point-cluster mapping\nmsg.payload = {\n    finalCentroids: centroids,\n    assignments: assignments\n};\n\nreturn msg;*/\nconst data = msg.payload;\n\nif (!data || !Array.isArray(data)) {\n    node.error(\"Invalid data: Payload must be an array of points.\");\n    return;\n}\n\nconst k = 3; // Number of clusters\nconst maxIterations = 10;\n\nlet centroids = data.slice(0, k);\n\nfunction distance(a, b) {\n    return Math.sqrt(\n
```

```
a.reduce((sum, val, i) => sum + Math.pow(val - b[i], 2), 0)\n    );\n}\n\nlet assignments = [];\n\nfor (let i = 0; i < maxIterations; i++) {\n    let clusters = Array.from({ length: k }, () => []);\n    assignments = [];\n\n    data.forEach(point => {\n        let distances = centroids.map(c => distance(point, c));\n        let clusterIndex = distances.indexOf(Math.min(...distances));\n        clusters[clusterIndex].push(point);\n        assignments.push({ point, cluster: clusterIndex });\n    });\n\n    centroids = clusters.map(cluster => {\n        let len = cluster.length;\n        if (len === 0) return Array(data[0].length).fill(0); \n        let sum = cluster.reduce((acc, point) => {\n            return acc.map((val, idx) => val + point[idx]);\n        }, Array(data[0].length).fill(0));\n        return sum.map(val => val / len);\n    });\n}\n\n// Prepare data for template node\nmsg.payload = {\n    centroids: centroids,\n    assignments: assignments\n};\n\nreturn msg;\n",
    "outputs": 1,
    "timeout": "",
    "noerr": 0,
    "initialize": "",
    "finalize": "",
    "libs": [],
    "x": 670,
    "y": 280,
    "wires": [
      [
        "4",
        "5"
      ]
    ]
  },
  {
    "id": "4",
    "type": "debug",
    "z": "1",
    "name": "K-means Output",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "false",
    "statusVal": "",
    "statusType": "auto",
    "x": 1040,
    "y": 140,
    "wires": []
  },
  {
    "id": "5",
    "type": "ui_template",
    "z": "1",
    "group": "a35d741d84676d55",
    "name": "Cluster Display",
    "order": 2,
    "width": 0,
    "height": 0,
    "format": "<!DOCTYPE html>\n<html>\n\n<head>\n    <style>\n        table {\n            font-family: Arial, sans-serif;\n            border-collapse: collapse;\n            width: 100%;\n        }\n\n        th,\n        td {\n            border: 1px solid #dddddd;\n            text-align: left;\n            padding: 8px;\n        }\n\n        th {\n            background-color: #f2f2f2;\n        }\n\n        tr:nth-child(even) {\n            background-color: #f9f9f9;\n        }\n    </style>\n</head>\n\n<body>\n\n    <h2>K-means Clustering Results</h2>\n\n    <!-- Display Centroids -->\n    <h3>Centroids</h3>\n    <table>\n        <tr>\n            <th>Cluster</th>\n            <th>Centroid Coordinates</th>\n        </tr>\n        {{#payload.centroids}}\n        <tr>\n            <td>{{@index}}</td>\n            <td>{{this}}</td>\n        </tr>\n        {{/payload.centroids}}\n    </table>\n\n    <!-- Display Points and Their Clusters -->\n
```

<h3>Assignments</h3>\n <table>\n <tr>\n <th>Point</th>\n <th>Cluster</th>\n </tr>\n
{{#payload.assignments}}\n <tr>\n <td>{{this.point}}</td>\n <td>{{this.cluster}}</td>\n </tr>\n
{{/payload.assignments}}\n </table>\n\n</body>\n\n</html>",
    "storeOutMessages": true,
    "fwdInMessages": true,
    "resendOnRefresh": true,
    "templateScope": "local",
    "className": "",
    "x": 1020,
    "y": 320,
    "wires": [
      []
    ]
  },
  {
    "id": "a35d741d84676d55",
    "type": "ui_group",
    "name": "Group 2",
    "tab": "5a82fa95b356b0e0",
    "order": 2,
    "disp": true,
    "width": "6",
    "collapse": false,
    "className": ""
  },
  {
    "id": "5a82fa95b356b0e0",
    "type": "ui_tab",
    "name": "CH5",
    "icon": "dashboard",
    "disabled": false,
    "hidden": false
  }
]