Name: Omkar Shirwadkar
UID: 2021600062
Batch: D
Class: CSE(AI & ML)

EXPERIMENT NO: 8
Designing Interactive Dashboards and Storytelling using D3.js on Environment/Forest Cover Dataset

Aim:

To design interactive dashboards and create visual storytelling using D3.js on a dataset related to Environment/Forest cover, covering basic and advanced charts.

Objectives:

1. **To understand how to use D3.js for data visualization.**
2. **To implement basic charts like Bar chart, Pie chart, Histogram, Timeline chart, Scatter plot, and Bubble plot.**
3. **To implement advanced charts like Word chart, Box and whisker plot, Violin plot, Regression plot (linear and nonlinear), 3D chart, and Jitter.**
4. **To draw observations and insights from each chart.**
5. **To create an interactive storytelling dashboard using the above visualizations.**

Expected Outcomes:

1. **Ability to create various types of visualizations using D3.js.**
2. **Interactive dashboards demonstrating different types of charts.**
3. **Insights from the Environment/Forest cover the dataset through visual storytelling.**

Dataset:

https://www.kaggle.com/datasets/arjunprasadsarkhel/forest-cover-in-india/data

4. **Program and Code:**
   index.html:

```
To remove the line numbers from the left side in your
HTML code, simply delete them from the beginning of each
line. Below is the corrected version of your code without
the line numbers:
```

~~~~
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Grouped Bar Chart and Interactive Pie Chart -
Forest Cover</title>
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <style>
    .bar {
      fill-opacity: 0.7;
    }
    .axis-label {
      font-size: 12px;
      font-weight: bold;
    }
    .legend text {
      font-size: 12px;
    }
    .pie-label {
      font-size: 12px;
      text-anchor: middle;
      fill: white;
    }
    .info-box {
      font-size: 16px;
      font-weight: bold;
      text-align: center;
      margin-top: 10px;
~~~~

```
    }
    .scatter-dot {
      fill: steelblue;
      stroke: #333;
      stroke-width: 1px;
      fill-opacity: 0.7;
    }
    .line {
      fill: none;
      stroke-width: 2;
    }
    .tooltip {
      position: absolute;
      text-align: center;
      padding: 5px;
      font: 12px sans-serif;
      background: lightsteelblue;
      border: 1px solid gray;
      border-radius: 5px;
      pointer-events: none;
      opacity: 0;
    }
  </style>
</head>

<body>
  <h2>Grouped Bar Chart of Forest Cover by State/UT</h2>
  <div id="bar-chart"></div>
  <div class="info-box" id="info-box">Click on a bar to
see details</div>
```

```
  <h2>Interactive Pie Chart of Total Forest Area
Distribution</h2>
  <div id="pie-chart"></div>
  <div class="info-box" id="info-box2">Click on a slice
to see details</div>

  <h2>Scatter Plot: Geographical Area vs Total
Forest</h2>
  <div id="scatter-plot"></div>
  <div class="info-box" id="scatter-info">Hover over a
point to see details</div>

  <h2>Interactive Timeline Chart: Forest Cover Over
Years</h2>
  <div id="timeline-chart"></div>
  <div class="tooltip" id="tooltip"></div>

  <script>
    const margin = { top: 40, right: 30, bottom: 70,
left: 60 };
    const width = 800 - margin.left - margin.right;
    const height = 500 - margin.top - margin.bottom;

    const pieWidth = 400;
    const pieHeight = 400;
    const radius = Math.min(pieWidth, pieHeight) / 2;

    const svgBar = d3.select("#bar-chart")
      .append("svg")
      .attr("width", width + margin.left + margin.right)
      .attr("height", height + margin.top +
margin.bottom)
```

```
      .append("g")
      .attr("transform",
`translate(${margin.left},${margin.top})`);

    const svgPie = d3.select("#pie-chart")
      .append("svg")
      .attr("width", pieWidth)
      .attr("height", pieHeight)
      .append("g")
      .attr("transform", `translate(${pieWidth /
2},${pieHeight / 2})`);

    const infoBox = d3.select("#info-box");
    const infoBox2 = d3.select("#info-box2");

    const scatterMargin = { top: 40, right: 30, bottom:
70, left: 90 };
    const scatterWidth = 800 - scatterMargin.left -
scatterMargin.right;
    const scatterHeight = 500 - scatterMargin.top -
scatterMargin.bottom;

    const svgScatter = d3.select("#scatter-plot")
      .append("svg")
      .attr("width", scatterWidth + scatterMargin.left +
scatterMargin.right)
      .attr("height", scatterHeight + scatterMargin.top +
scatterMargin.bottom)
      .append("g")
      .attr("transform",
`translate(${scatterMargin.left},${scatterMargin.top})`);
```

```
    const scatterInfoBox = d3.select("#scatter-info");

    // Read the data
    d3.csv("Forest.csv").then(function (data) {
      const subgroups = ['Very dense forest', 'Mod. dense
forest', 'Open forest'];
      const groups = data.map(d => d['State/UTs']);

      const x = d3.scaleBand()
        .domain(groups)
        .range([0, width])
        .padding([0.2]);

      svgBar.append("g")
        .attr("transform", `translate(0,${height})`)
        .call(d3.axisBottom(x))
        .selectAll("text")
        .attr("transform", "translate(-10,0)rotate(-45)")
        .style("text-anchor", "end");

      const y = d3.scaleLinear()
        .domain([0, d3.max(data, d => +d['Very dense
forest'])])
        .range([height, 0]);

      svgBar.append("g")
        .call(d3.axisLeft(y));

      const xSubgroup = d3.scaleBand()
        .domain(subgroups)
        .range([0, x.bandwidth()])
        .padding([0.05]);
      const color = d3.scaleOrdinal()
        .domain(subgroups)
        .range(['#4daf4a', '#377eb8', '#ff7f00']);

      svgBar.append("g")
        .selectAll("g")
        .data(data)
        .join("g")
        .attr("transform", d =>
`translate(${x(d['State/UTs'])}, 0)`)
        .selectAll("rect")
        .data(d => subgroups.map(key => ({ key: key,
value: d[key], state: d['State/UTs'] })))
        .join("rect")
        .attr("x", d => xSubgroup(d.key))
        .attr("y", d => y(d.value))
        .attr("width", xSubgroup.bandwidth())
        .attr("height", d => height - y(d.value))
        .attr("fill", d => color(d.key))
        .attr("class", "bar")
        .on("click", function (event, d) {
          infoBox.text(`State/UT: ${d.state}, ${d.key}:
${d.value} sq km`);
        });

      const legendBar = svgBar.append("g")
        .attr("transform", `translate(${width - 120},
-10)`);

      subgroups.forEach((subgroup, i) => {
        legendBar.append("rect")
```

```
    .attr("x", 0)
    .attr("y", i * 20)
    .attr("width", 18)
    .attr("height", 18)
    .style("fill", color(subgroup));

  legendBar.append("text")
    .attr("x", 24)
    .attr("y", i * 20 + 9)
    .attr("dy", "0.35em")
    .text(subgroup);
});

const totalForestArea = data.map(d => ({
  state: d['State/UTs'],
  value: +d['Total forest']
}));

const pie = d3.pie()
  .value(d => d.value);

const arc = d3.arc()
  .innerRadius(0)
  .outerRadius(radius);

const pieColor = d3.scaleOrdinal()
  .domain(totalForestArea.map(d => d.state))
  .range(d3.schemeSet3);

svgPie.selectAll('path')
  .data(pie(totalForestArea))
  .enter()
```

```
        .append('path')
        .attr('d', arc)
        .attr('fill', d => pieColor(d.data.state))
        .attr("stroke", "white")
        .style("stroke-width", "2px")
        .on("click", function (event, d) {
          infoBox2.text(`State/UT: ${d.data.state}, Total
Forest Area: ${d.data.value} sq km`);
        });

      const xScale = d3.scaleLinear()
        .domain([0, d3.max(data, d => +d['Geographical
area'])])
        .range([0, scatterWidth]);

      const yScale = d3.scaleLinear()
        .domain([0, d3.max(data, d => +d['Total
forest'])])
        .range([scatterHeight, 0]);

      svgScatter.append("g")
        .attr("transform",
`translate(0,${scatterHeight})`)
        .call(d3.axisBottom(xScale))
        .append("text")
        .attr("x", scatterWidth / 2)
        .attr("y", 50)
        .attr("fill", "black")
        .attr("class", "axis-label")
        .text("Geographical Area (sq km)");

      svgScatter.append("g")
```

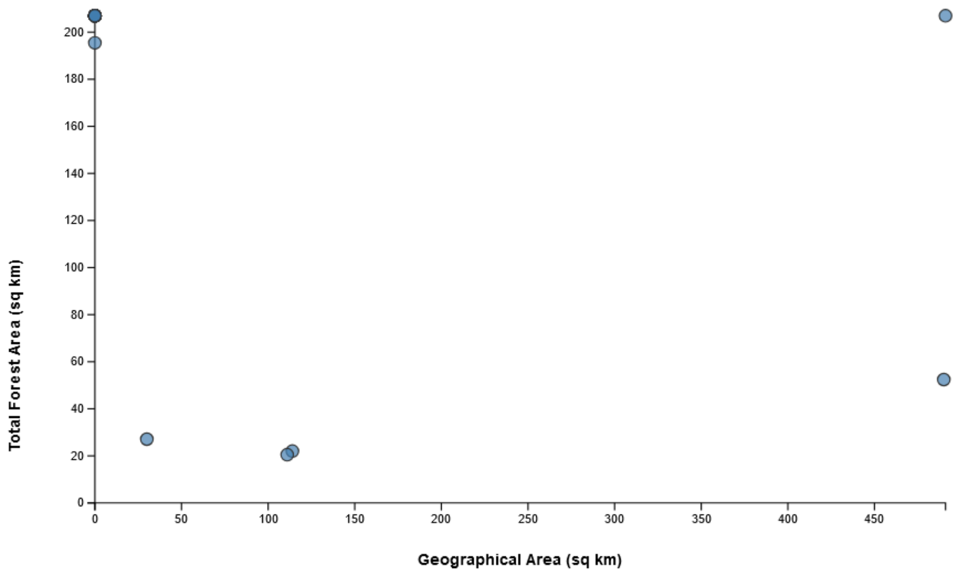
```
        .call(d3.axisLeft(yScale))
        .append("text")
        .attr("x", -scatterHeight / 2)
        .attr("y", -60)
        .attr("fill", "black")
        .attr("transform", "rotate(-90)")
        .attr("class", "axis-label")
        .text("Total Forest Area (sq km)");

      svgScatter.selectAll(".scatter-dot")
        .data(data)
        .enter()
        .append("circle")
        .attr("class", "scatter-dot")
        .attr("cx", d => xScale(d['Geographical area']))
        .attr("cy", d => yScale(d['Total forest']))
        .attr("r", 5)
        .on("mouseover", function (event, d) {
          scatterInfoBox.text(`State/UT:
${d['State/UTs']}, Geographical Area: ${d['Geographical
area']} sq km, Total Forest: ${d['Total forest']} sq
km`);
        })
        .on("mouseout", function () {
          scatterInfoBox.text("");
        });
    });
  </script>
</body>
</html>
```

5. Visualization and Observations:

a. Scatter Plot:

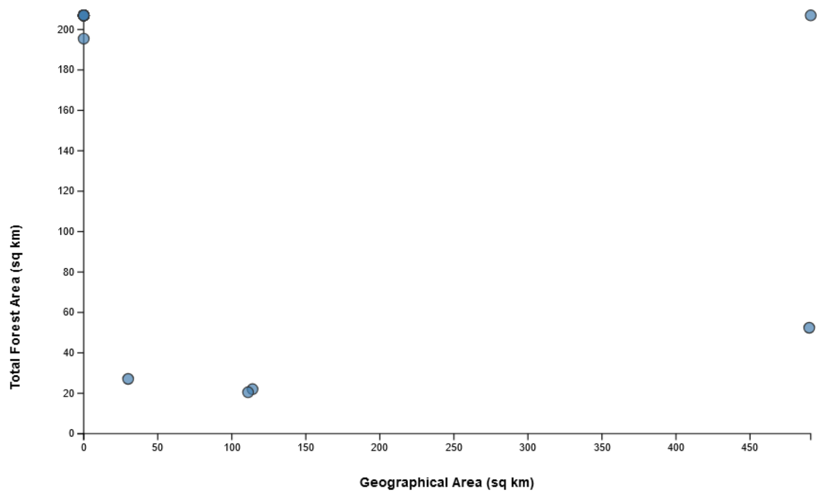
Scatter Plot: Geographical Area vs Total Forest



Hover over a point to see details

The above scatter plot shows the relationship between the geographical area occupied by an union territory and the Total forest area in that state.

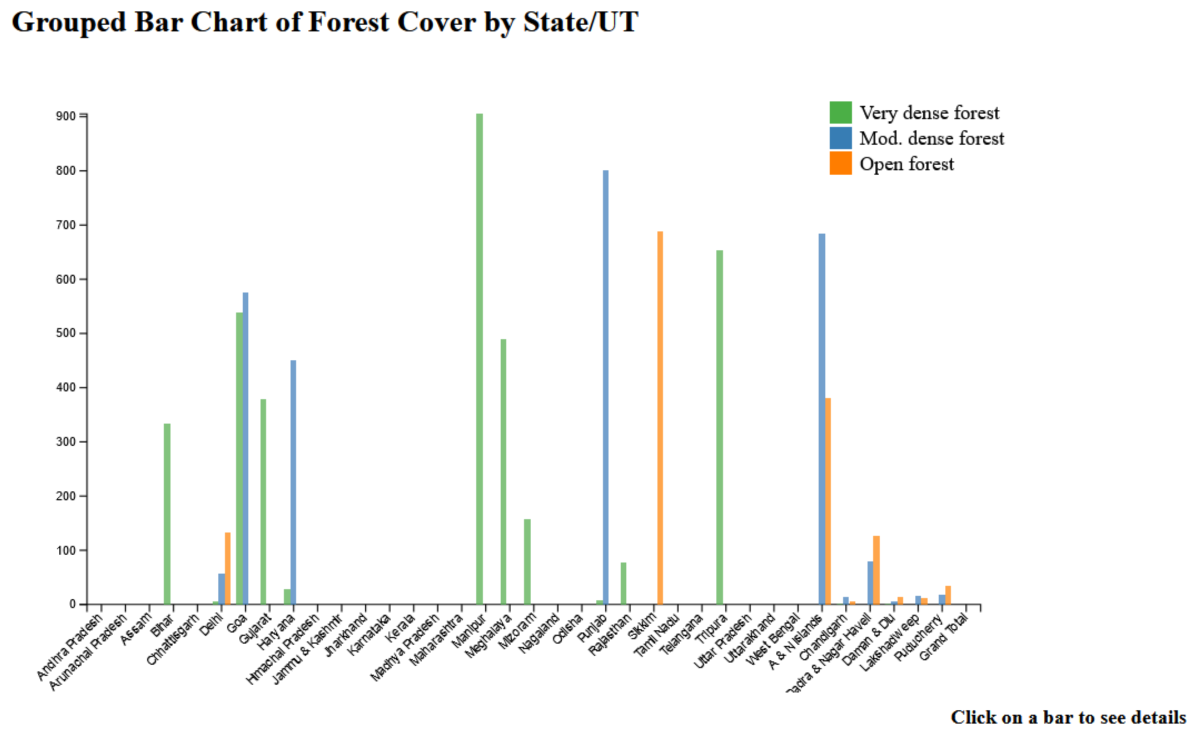
Scatter Plot: Geographical Area vs Total Forest



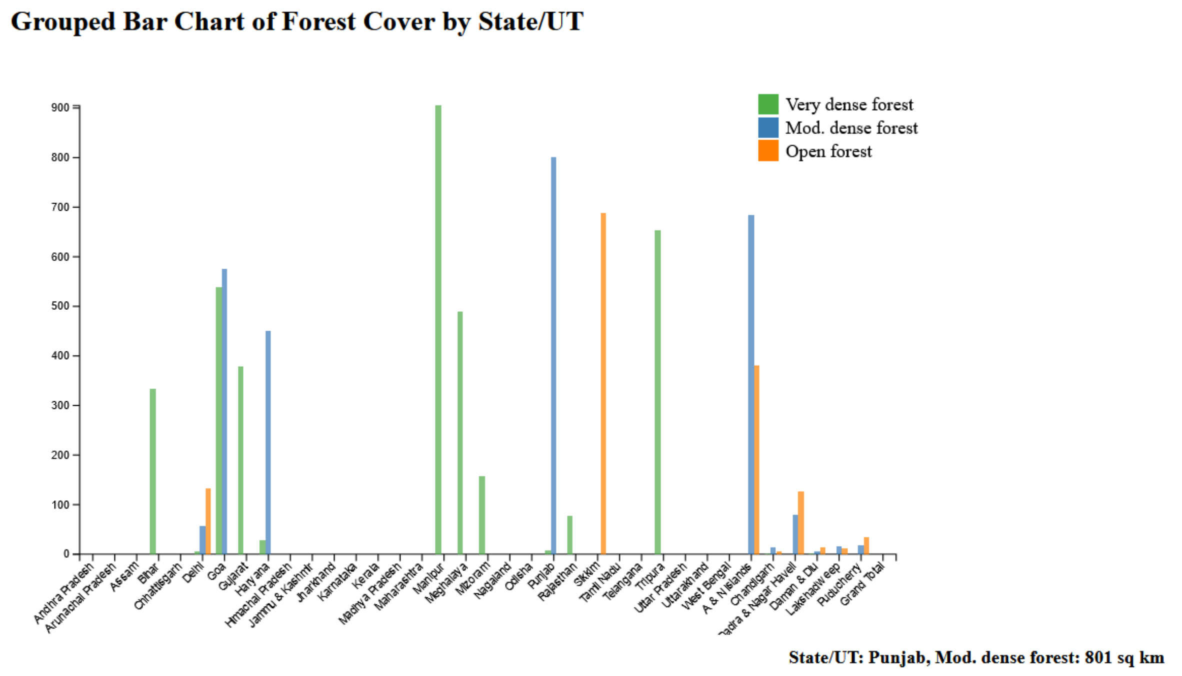
State/UT: Dadra & Nagar Haveli, Geographical Area: 491 sq km, Total Forest: 207 sq km

The above stats are for Dadra and Nagar Haveli

b. Bar Chart:



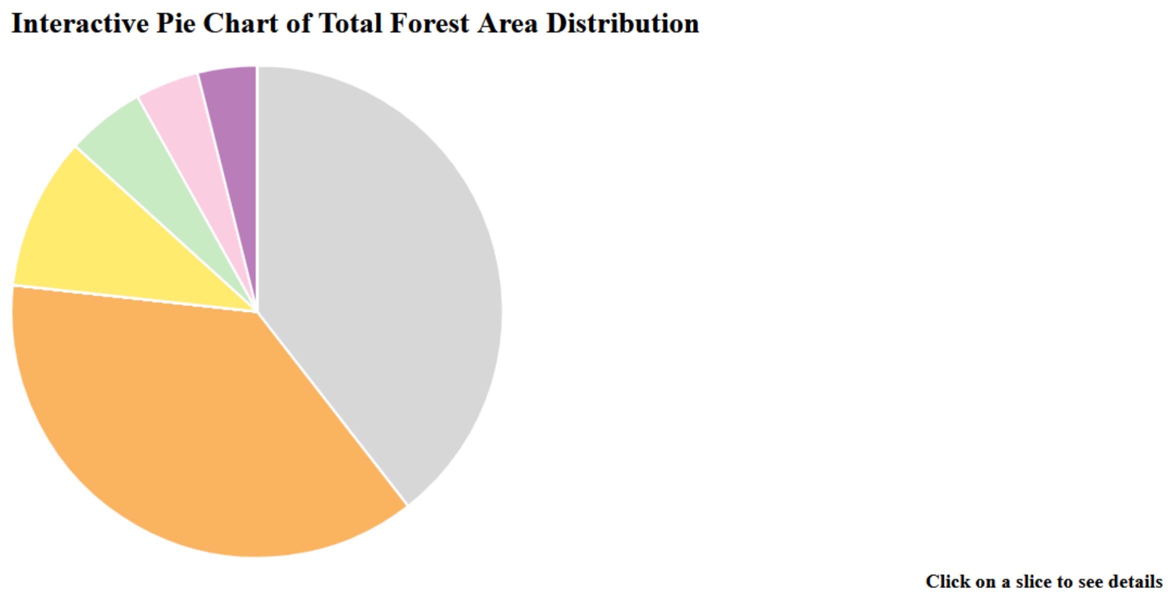
For every state the count of Very dense, medium dense and open forest count is shown in the above grouped bar chart.



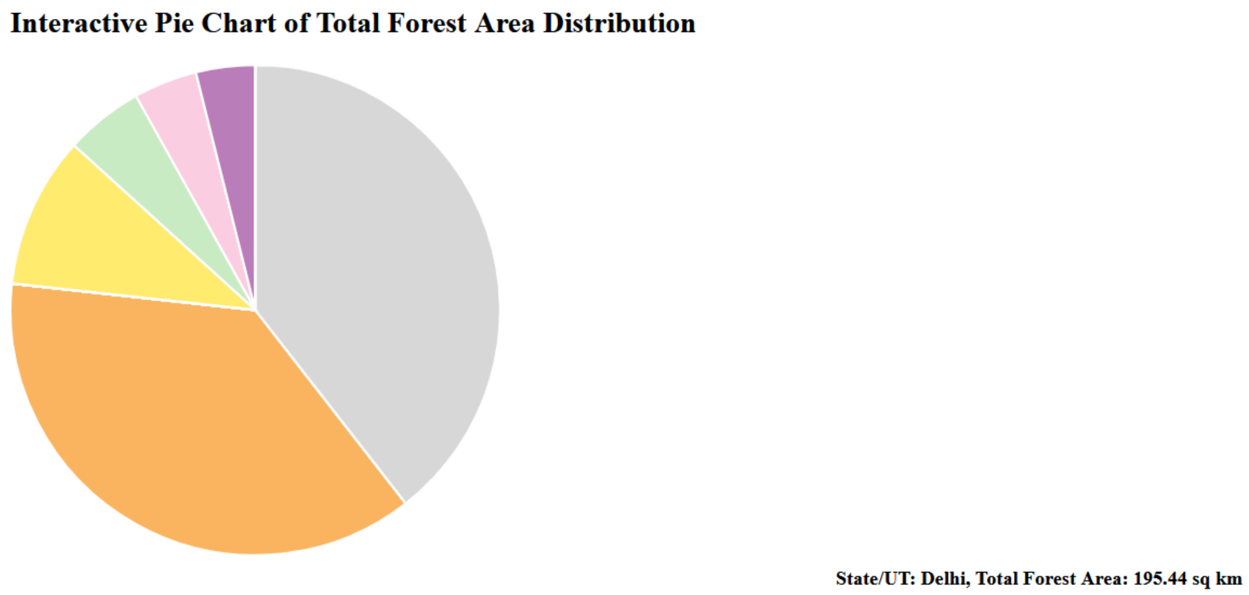
State/UT: Punjab, Mod. dense forest: 801 sq km

After clicking the tallest blue bar we see the above statement - State punjab, mod dense forest and 801 sq km

c. Pie Chart:



The above Pie chart shows the total forest area distribution among all union territories.



After clicking the pie chart containing the orange color we see the above output.