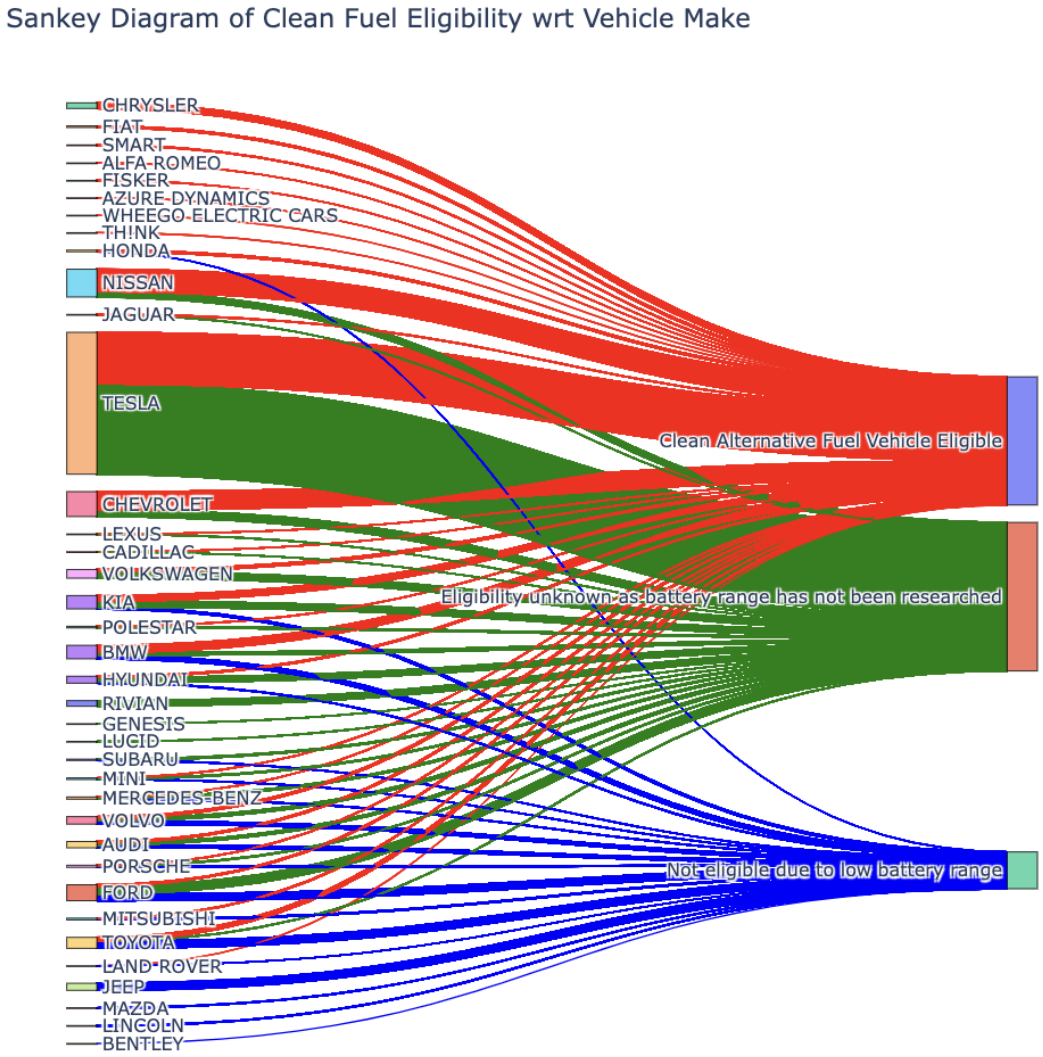
Homework 3 – Information Visualization

Name: Ashutosh Chauhan
UIN: 232009024

1. **Type of Visualization:** Sankey Diagram

   Dataset Description:
   In this plot, I am using Electric vehicle population data stored in file 'Electric_Vehicle_Population_Data_20231102.csv'. I am extracting 'Make' and 'Clean Fuel Eligibility' columns from file and adding them to a graph. This graph I am using to plot connection between 'Make' of vehicle with 'Clean Fuel Eligibility' in form of Sankey Diagram.

   Visualization Description:
   For the above dataset we have 1 independent variable (Make) and 1 dependent variable (Clean Fuel Eligibility). We want to know what percentage of Clean Fuel vehicles are manufactured by different vehicle manufacturers. We used sankey diagrams because these are excellent for visualizing the flow of data, resources, or quantities between multiple categories or stages. They provide a clear representation of how inputs, outputs, or transitions are distributed and connected. We can observe from the below plot easily that a major portion of clean fuel vehicles come from Tesla.




```
import plotly.graph_objects as graph
ev_df = pd.read_csv("/content/Electric_Vehicle_Population_Data_20231102.csv")

node_labels = ev_df['Clean Alternative Fuel Vehicle (CAFV) Eligibility'].append(ev_df['Make']).unique()
node_dict = {label: idx for idx, label in enumerate(node_labels)}

first_colm = ev_df['Make'].map(node_dict)
sec_colm = ev_df['Clean Alternative Fuel Vehicle (CAFV) Eligibility'].map(node_dict)

link_values = [1] * len(ev_df)

conn_colors = []
for alt_type in ev_df['Clean Alternative Fuel Vehicle (CAFV) Eligibility']:
  if alt_type == 'Clean Alternative Fuel Vehicle Eligible':
    color = 'red'
  elif alt_type == 'Eligibility unknown as battery range has not been researched':
    color = 'green'
  else:
    color = 'blue'
  conn_colors.append(color)

fig = graph.Figure(graph.Sankey(
    node=dict(pad=15, thickness=20, line=dict(color="black", width=0.5), label=node_labels),
    link=dict(source= first_colm, target=sec_colm, value=link_values, color = conn_colors
    )
))

fig.update_layout(title_text="Sankey Diagram of Clean Fuel Eligibility wrt Vehicle Make", width=800, height=800 )
fig.show()
```

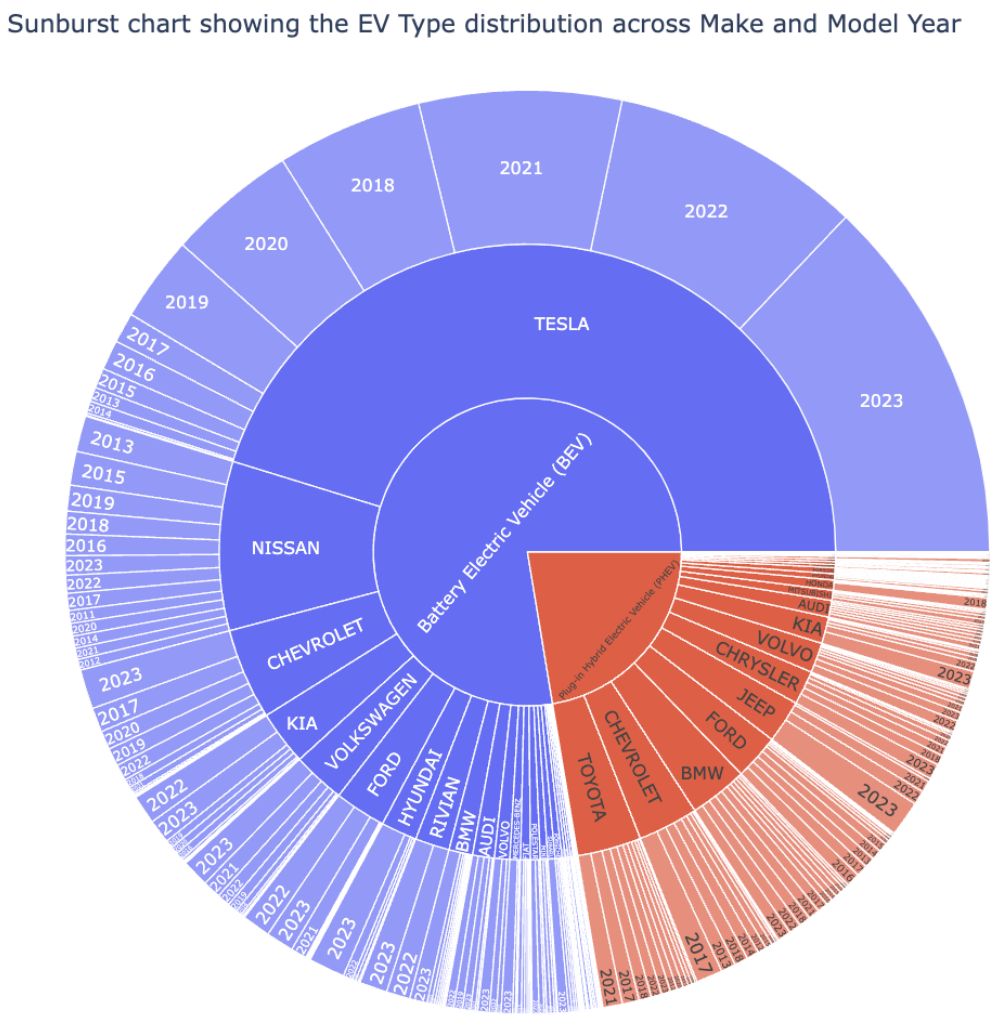
2. **Type of Visualization:** Sunburst Chart (Radial Icicle Tree)

Dataset Description:

In this plot, I am using Electric vehicle population data stored in file 'Electric_Vehicle_Population_Data_20231102.csv'. I am extracting 'Electric Vehicle Type', 'Make' and 'Model year columns from file and adding them to a dataframe. This dataframe I am using to plot hierarchy in 'Electric Vehicle Type', 'Make' of vehicle and 'Model year' in form of Sunburst Chart (Radial Icicle Tree).

Visualization Description:

For the above dataset we have 3 independent variables ('Electric Vehicle Type', 'Make' and 'Model year). We want to know the hierarchy between Electric vehicle type, make and model year. We used sunburst charts because these are excellent for representing hierarchical data structures. They provide a clear and intuitive way to display parent-child relationships, making it easy to see how data is organized and structured. We can very easily see the composition of Electric vehicle type along with further hierarchical information.



```
import plotly.express as px
ev_df = pd.read_csv("/content/Electric_Vehicle_Population_Data_20231102.csv")

sunburst_data = ev_df.groupby(['Electric Vehicle Type','Make', 'Model Year']).size().reset_index(name='count')
# sunburst_data.head(40)
fig = px.sunburst(sunburst_data, path=['Electric Vehicle Type','Make', 'Model Year'], values='count')
fig.update_layout( title_text = "Sunburst chart showing the EV Type distribution across Make and Model Year", width=800, height=800)
fig.show()
```

3. **Type of Visualization**: Tree Maps

Dataset Description:

In this plot, I am using vehicle registration data stored in file 'Vehicle_Registration_Summary_20231102.csv'. I am extracting all the rows from dataset corresponding to Washington state and calculating the vehicle count of various 'Electrification level'. This dataframe I am using to plot containment of 'Electrification level' in the registered vehicles in Washington state using Treemap.

Visualization Description:

For the above dataset we have 1 independent variable (Electrification level). We are interested to know the Electrification level composition of vehicles registered in Washington between 2017 and 2023. We used Treemaps because these are particularly useful for displaying hierarchical or nested data structures. They allow you to show the structure of your data and how larger categories break down into smaller ones. They enable quick and intuitive comparisons between categories. Users can see the relative sizes of different data points within the same treemap. We can even draw observations easily from below plot related to the categorization level. We can easily know that the maximum number of vehicles registered are of ICE type.


```
import plotly.express as px
reg_df = pd.read_csv("/content/Vehicle_Registration_Summary_20231102.csv")

reg_df = reg_df[reg_df['State'].isin(['WA'])]
elec_level_count = reg_df.groupby('Electrification Level').size().reset_index(name='count')

fig = px.treemap(elec_level_count, path=['Electrification Level'], values='count', color='count', hover_data=['Electrification Level','count'],color_continuous_scale='rainbow')
fig.update_traces(textinfo="label+percent entry")
fig.update_layout(title="Treemap of Electrification level in  Washington")

fig.show()
```

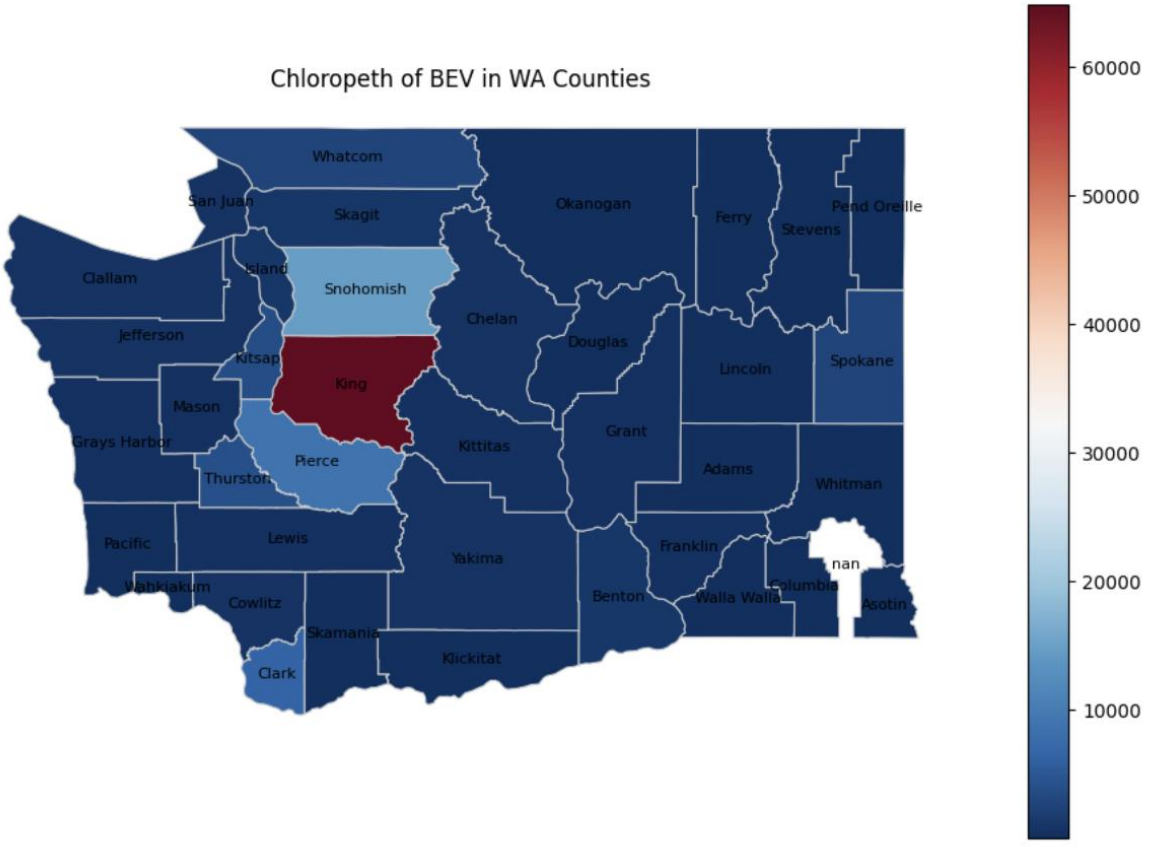
4. **Type of Visualization:** Choropleth Map

Dataset Description:

In this plot, I am using Electric vehicle population data stored in file 'Electric_Vehicle_Population_Data_20231102.csv'. I am extracting rows from dataset that are corresponding to 'Battery Electric Vehicle (BEV)' electric vehicle type in Washington. This dataframe I am using to plot cholopleth map showing BEV distribution in various counties of Washington state.

Visualization Description:

For the above dataset we have 1 dependent variable (number of BEV vehicles) and 1 independent variable (county). Here we are interested to know the distribution of BEV's in various counties of Washington state. We used choropleth maps because these provide a clear and intuitive way to interpret data by representing it geographically. The use of colors and shades allows for quick comprehension of spatial patterns. Choropleth maps allow for easy comparison of data across regions. By using a consistent color scale, we can quickly compare values in different areas. We can easily make out from below plot that 'king' county has maximum number BEV's in Washington state.



```
import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd

ev_data = pd.read_csv("/content/Electric_Vehicle_Population_Data_20231102.csv")
g_df = gpd.read_file('/content/WA_County_Boundaries.geojson')

ev_df = ev_data[ev_data['State']=='WA']
ev_df = ev_df[ev_df['Electric Vehicle Type'] == 'Battery Electric Vehicle (BEV)']
result = ev_df.groupby('County').size().reset_index(name = 'Count')

data = g_df.merge(result, left_on='JURISDICT_LABEL_NM', right_on='County', how='left')
fig, ax = plt.subplots(1, 1, figsize=(12, 8))
data.plot(column='Count', cmap='RdBu_r', linewidth=0.8, ax=ax, edgecolor='0.8', legend=True)

for x, y, label in zip(data.geometry.centroid.x, data.geometry.centroid.y, data.County):
    ax.text(x, y, label, fontsize=8, ha='center', va='center', color='black')

ax.set_title('Chloropeth of BEV in WA Counties')
ax.set_axis_off()

plt.show()
```

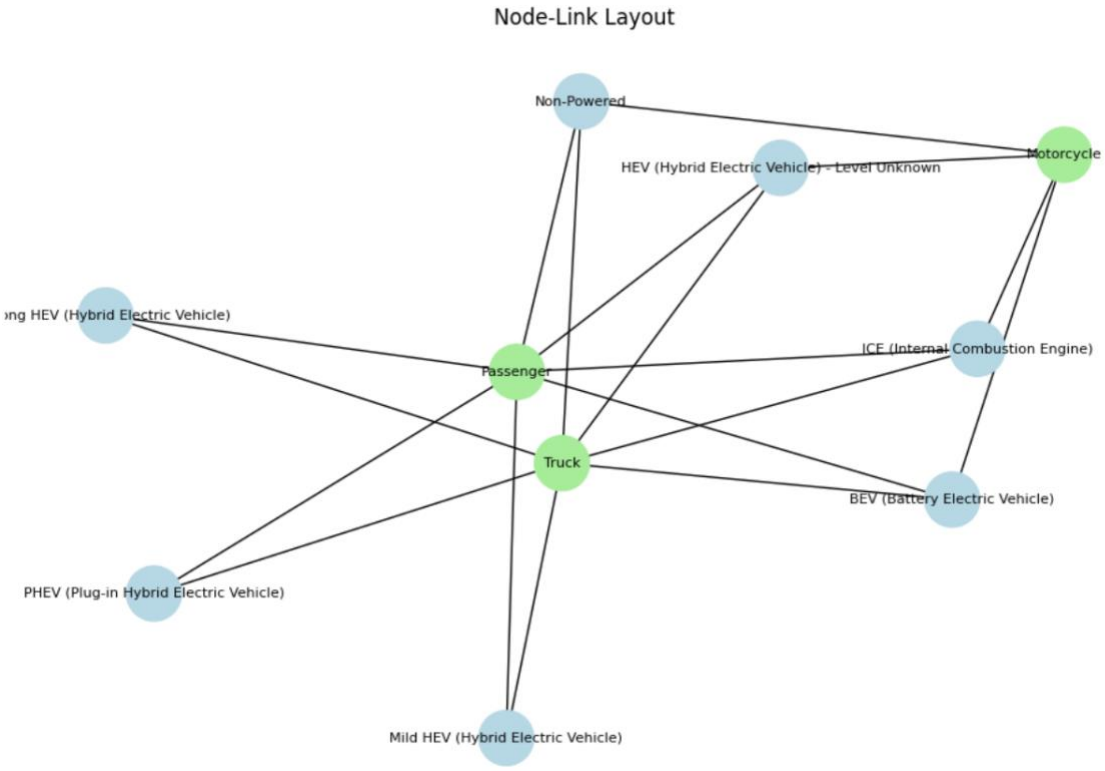
5. **Type of Visualization**: Node-Link Layout

Dataset Description:

In this plot, I am using vehicle registration data stored in file 'Vehicle_Registration_Summary_20231102.csv'. I am extracting 'Vehicle Primary use' and 'Electrification Level' data from file and adding them to a graph. This graph I am using to plot connection between both using Node-Link Layout.

Visualization Description:

For the above dataset we have 2 independent variables ('Vehicle primary use', 'Electrification Level'). We want to know the connection and links between vehicle primary use and various electrification levels. We used node-link layouts because these provide an intuitive way to visualize complex networks by representing nodes as points and edges as lines connecting these points. This approach makes it easy to grasp the structure and relationships in the network. We can observe this in below plot as well.




```python
import networkx as nx
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt


reg_df = pd.read_csv('/content/Vehicle_Registration_Summary_20231102.csv')
reg_df = reg_df[reg_df['Vehicle Type'] != 'INCOMPLETE VEHICLE']

G = nx.Graph()
G.add_nodes_from(reg_df['Electrification Level'], type='Electrification Level')
G.add_nodes_from(reg_df['Vehicle Primary Use'], type='Vehicle Primary Use')

for index, row in reg_df.iterrows():
    G.add_edge(row['Electrification Level'], row['Vehicle Primary Use'])

node_colors = {
    node: '#add8e6' if 'Electrification Level' in G.nodes[node]['type'] else '#90ee90'
    for node in G.nodes
}

plt.figure(figsize=(12, 8))
pos = nx.spring_layout(G, k=0.2)

node_size = 1000
# nx.draw_networkx_nodes(G, pos, node_size=node_size, node_color='pink')
nx.draw_networkx_nodes(G, pos, node_size=node_size, node_color=[node_colors[node] for node in G.nodes])

node_labels = {node: node for node in G.nodes}
nx.draw_networkx_labels(G, pos, labels=node_labels, font_size=8, font_color='black')

nx.draw_networkx_edges(G, pos)

plt.title('Node-Link Layout')
plt.axis('off')
plt.show()
```

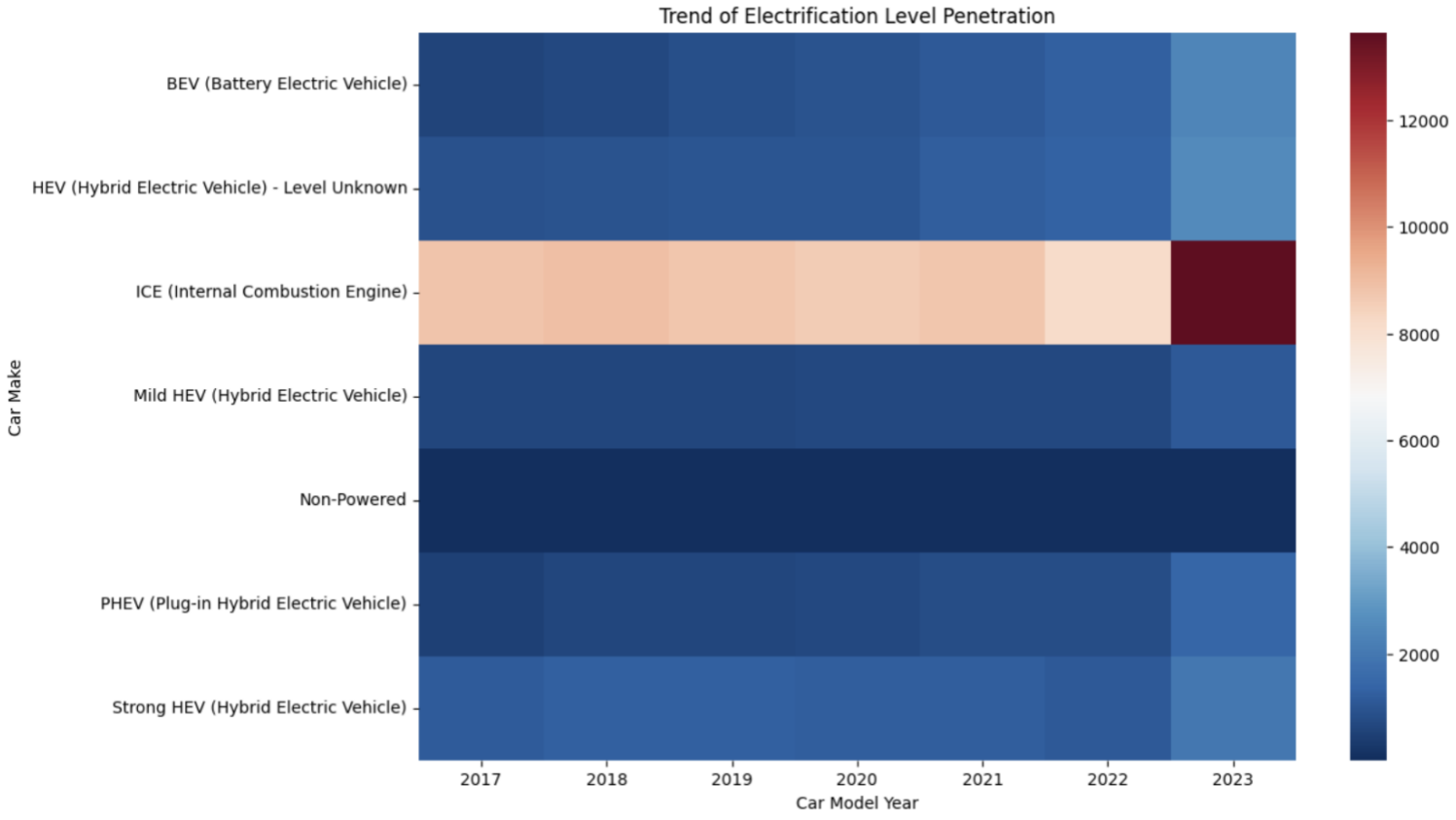
6. **Type of Visualization:** Heat Map of Time Series

Dataset Description:

In this plot, I am using vehicle registration data stored in file 'Vehicle_Registration_Summary_20231102.csv'. I am extracting 'Electrification Level' data for every year between 2017 and 2023 and creating a pivot table. This table I am using to show trends in number of vehicles of each electrification level with year using heat map of time series.

Visualization Description:

For the above dataset, we have 1 dependent variables (number of vehicle) and 2 independent variables (Electrification Level and year). We want to know how number of vehicles in each electrification level varies with time. We used heat maps because these help in recognizing patterns and trends in large datasets, which might be difficult to identify in raw numerical data or traditional tables. Using below heat map we can directly extract the insights like the increase in ICE type vehicle is maximum between 2022 to 2023.



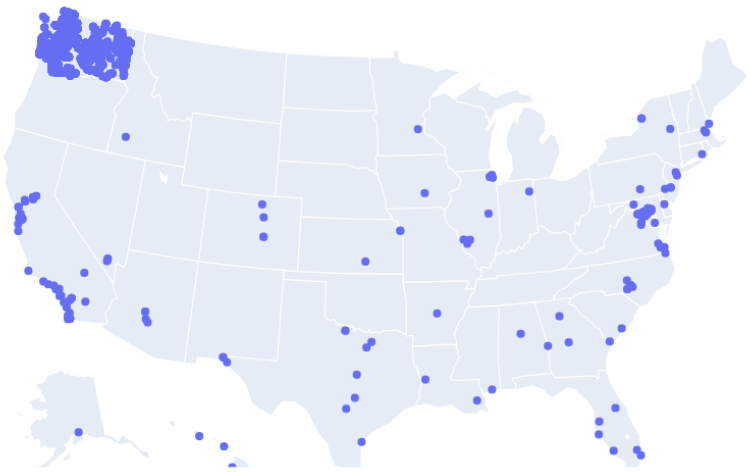
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

reg_df = pd.read_csv("/content/Vehicle_Registration_Summary_20231102.csv")
table = reg_df.pivot_table(index="Electrification Level", columns="Calendar Year", aggfunc="size", fill_value=0)

plt.figure(figsize=(12, 8))
sns.heatmap(table, annot=False, cmap="RdBu_r")

# Set labels and title
plt.xlabel("Car Model Year")
plt.ylabel("Car Make")
plt.title("Trend of Electrification Level Penetration")

# Show the heatmap
plt.show()
```

7. **Type of Visualization**: Dot Map

   Dataset Description:
   In this plot, I am using Electric vehicle population data stored in file 'Electric_Vehicle_Population_Data_20231102.csv'. I am extracting rows from dataset that are corresponding to manufacturer 'TESLA' in a dataframe. This dataframe I am using to plot dot map showing Tesla distribution throughout US.

   Visualization Description:
   For the above dataset we have 1 dependent variable (count). We want to know the distribution of Tesla vehicle in US. We used dot maps because these excel in showing the spatial distribution of data, making it easy to understand how a variable or phenomenon is distributed across a geographic area. This can be particularly valuable for geographic analysis and pattern recognition. We can see this in below plot as well how we can easily make out that most number of tesla vehicles sold were in Washington state.

Penetration of Tesla in US



```
import plotly.graph_objects as go
import pandas as pd

ev_df = pd.read_csv('/content/Electric_Vehicle_Population_Data_20231102.csv')
ev_df = ev_df[ev_df['Make'] == 'TESLA']
ev_df['coordinates'] = ev_df['Vehicle Location'].str.extract(r'POINT \((.*?)\)')
ev_df[['long', 'lat']] = ev_df['coordinates'].str.split(" ", expand=True).astype(float)

fig = go.Figure(data=go.Scattergeo(
        lon = ev_df['long'],
        lat = ev_df['lat'],
        mode = 'markers',
        ))

fig.update_layout(
        title = 'Penetration of Tesla in US',
        geo_scope='usa',
    )
fig.show()
```

8. **Type of Visualization**: Circle Mapping

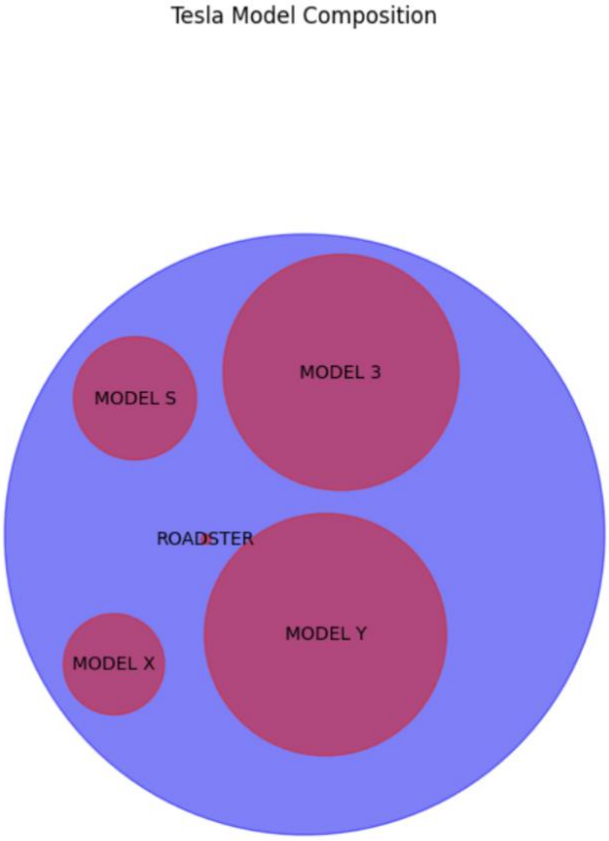
Dataset Description:

In this plot, I am using Electric vehicle population data stored in file 'Electric_Vehicle_Population_Data_20231102.csv'. I am extracting rows from dataset that are corresponding to manufacturer 'TESLA' in a dataframe. I am counting the number of tesla models of each type and ploting it using circle mapping.

Visualization Description:

For the above dataset we have 1 dependent variable (count) and 1 independent variable (Model type). We want to know the composition of tesla model types. We are using circle mapping because these plots excel at displaying hierarchical structures and the nesting of data. The size of each circle accurately represents the quantity or proportion of data it represents. This allows viewers to quickly grasp the relative sizes of data elements. We can observe from below plot that least sold model of tesla is roadster.

	Model
MODEL Y	29209
MODEL 3	27740
MODEL S	7533
MODEL X	5071
ROADSTER	48



```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np


df = pd.read_csv("/content/Electric_Vehicle_Population_Data_20231102.csv")
model_counts = df[df['Make'] == 'TESLA']['Model'].value_counts()
model_counts = pd.DataFrame(model_counts)
fig, ax = plt.subplots(figsize=(10, 10))


tesla_circle = plt.Circle((0.5, 0.5), 0.3, color='blue', alpha=0.5)
ax.add_artist(tesla_circle)

model_types = model_counts.index
model_sizes = model_counts.values
print(model_counts)
max_radius = 0.1

theta = np.linspace(0, 2 * np.pi, len(model_types), endpoint=False)
center= [[0.4,0.4],[0.5,0.55],[0.38,0.6],[0.35,0.4],[0.4,0.5]]
for i, model_type in enumerate(model_types):
    radius = max_radius * np.sqrt(model_sizes[i] / 20000)
    x = center[i][0] + radius * np.cos(theta[i])
    y = center[i][1] + radius * np.sin(theta[i])
    model_circle = plt.Circle((x, y), radius, color='red', alpha=0.5)
    ax.add_artist(model_circle)
    ax.text(x, y, model_type, ha='center', va='center')

ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
ax.set_aspect('equal', adjustable='box')

ax.set_title("Tesla Model Composition")
plt.axis('off')
plt.show()
```