NAAN MUDHALVAN PROJECT
PHASE – II

NAME	:	M.SHOBEYA
DOMAIN	:	ARTIFICIAL INTELLIGENCE
TOPIC	:	DEVELOPMENT-AUTONOMOUS VEHICLES
DEPT.	:	COMPUTER SCIENCE
COLLEGE	:	8201 ARJCET

INTRODUCTION

In 2019, autonomous vehicle technology faced crucial scrutiny through disengagement reports, highlighting instances where human intervention was necessary due to safety concerns or technical limitations. This project aims to analyze these reports using data visualization techniques to uncover trends, patterns, and insights. By visualizing the data, we seek to understand the performance, challenges, and advancements in autonomous driving technology, contributing to the ongoing dialogue on its safety and reliability.

OBJECTIVES

The primary objective of this project is to utilize data visualization techniques to explore and interpret the 2019 autonomous disengagement reports. By visually representing the data, we aim to:

- Identify common causes and patterns of disengagements.
- Analyze the frequency and distribution of disengagements across different operating conditions and environments.
- Assess the performance of various autonomous vehicle companies and their technologies.
- Evaluate the progress and challenges in autonomous vehicle development over time.

DATASET DESCRIPTION

Overview:

The dataset contains records of autonomous vehicle disengagements in 2019, documenting instances where human intervention was necessary during testing due to safety concerns or technical limitations.

Contents:

- Date and time of each disengagement
- Vehicle information (make, model, ID)
- Geospatial data (location of disengagement)
- Driving conditions (weather, road type, traffic density)
- Reason for disengagement
- Duration of disengagement
- Driver interventions
- Company and system information

Purpose:

The dataset facilitates analysis of autonomous vehicle performance, reliability, and safety. It enables identification of trends, patterns, and areas for improvement in self-driving technology.

Potential Analyses:

- ➢ Frequency and distribution analysis
- ➢ Root cause analysis
- ➢ Temporal analysis
- ➢ Comparative analysis
- ➢ Ethical Considerations:
- ➢ Handle the data ethically, considering privacy, proprietary information, and reporting biases. Ensure transparency, fairness, and accountability in analysis and interpretation.

Limitations:

- o May not reflect overall safety and reliability of self-driving technology

- Influenced by testing protocols, operational design domains, and reporting criteria

Availability:

Typically provided by regulatory authorities or autonomous vehicle companies. Access may be subject to privacy agreements, licensing terms, or regulatory requirements.

DATA VISUALIZATION TECHNIQUES

1.UNIVARIATE VISUALIZATION

Histograms:

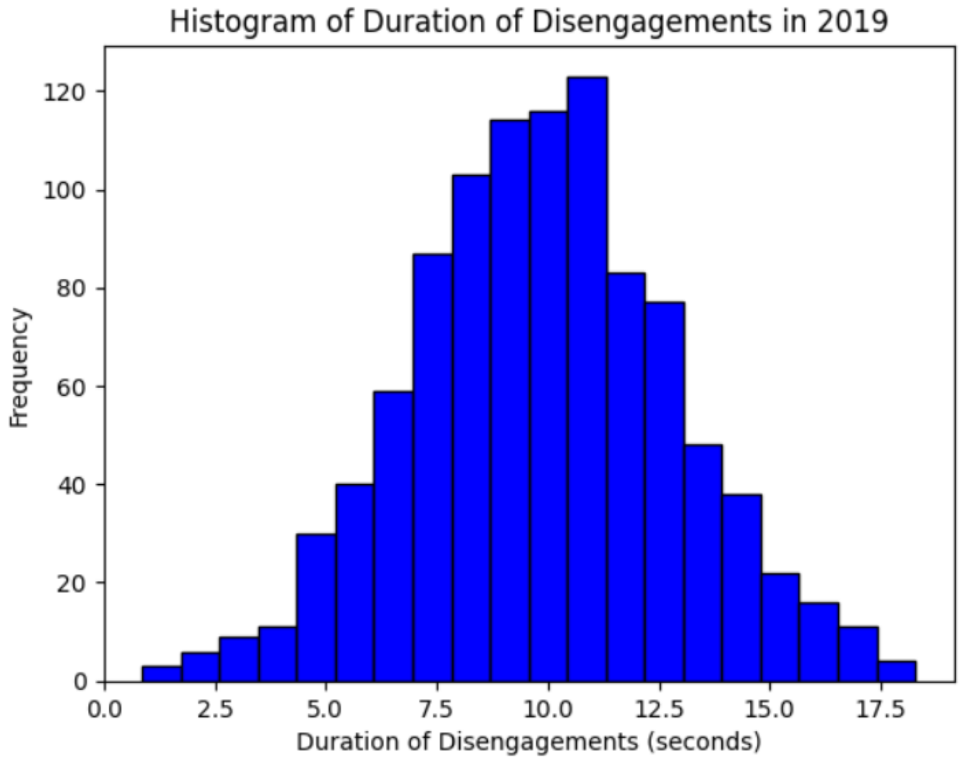
Histograms visually represent the distribution of numerical data by dividing it into intervals (bins) and showing the frequency of observations in each bin through bars of varying heights.

CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(0)
disengagement_data = pd.DataFrame({
    'duration': np.random.normal(loc=10, scale=3, size=1000)  # Sample duration data
})
plt.hist(disengagement_data['duration'], bins=20, color='blue', edgecolor='black')


plt.xlabel('Duration of Disengagements (seconds)')
plt.ylabel('Frequency')
plt.title('Histogram of Duration of Disengagements in 2019')
plt.show()
```

OUTPUT



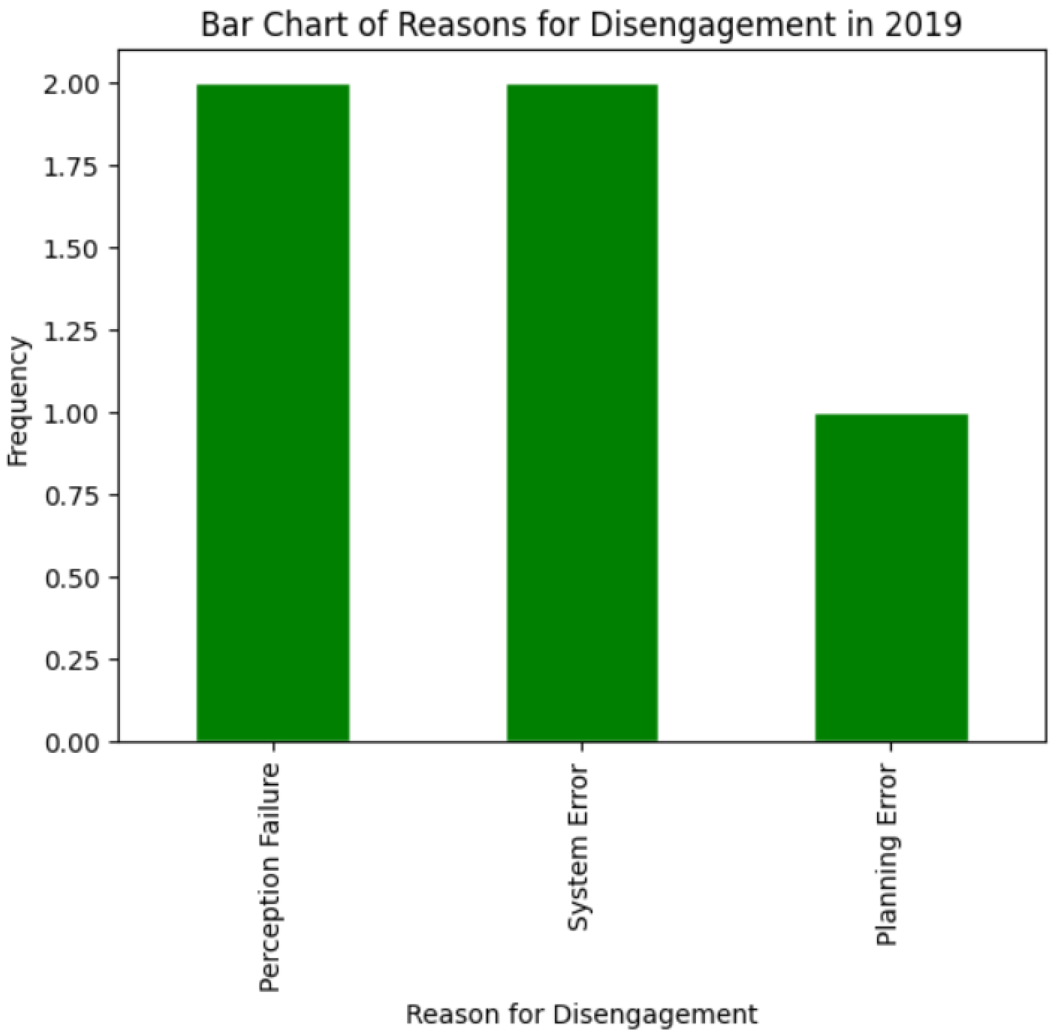
Bar Charts:

Bar charts visually display categorical data by representing each category with a bar, where the length or height of the bar corresponds to the frequency, proportion, or other relevant metric associated with that category.

CODE

```
import pandas as pd
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'company': ['Company A', 'Company B', 'Company C', 'Company A', 'Company B'],
    'reason': ['Perception Failure', 'System Error', 'Planning Error', 'Perception Failure', 'System Error']
})
reason_counts = disengagement_data['reason'].value_counts()
reason_counts.plot(kind='bar', color='green', edgecolor='white')
plt.xlabel('Reason for Disengagement')
plt.ylabel('Frequency')
plt.title('Bar Chart of Reasons for Disengagement in 2019')
plt.show()
```

OUTPUT



2. BIVARIATE VISUALIZATION

Scatter Plots

Scatter plots display the relationship between two continuous variables by plotting each data point with one variable on the x-axis and the other on the y-axis. They visually represent patterns, trends, correlations, and outliers in the data.

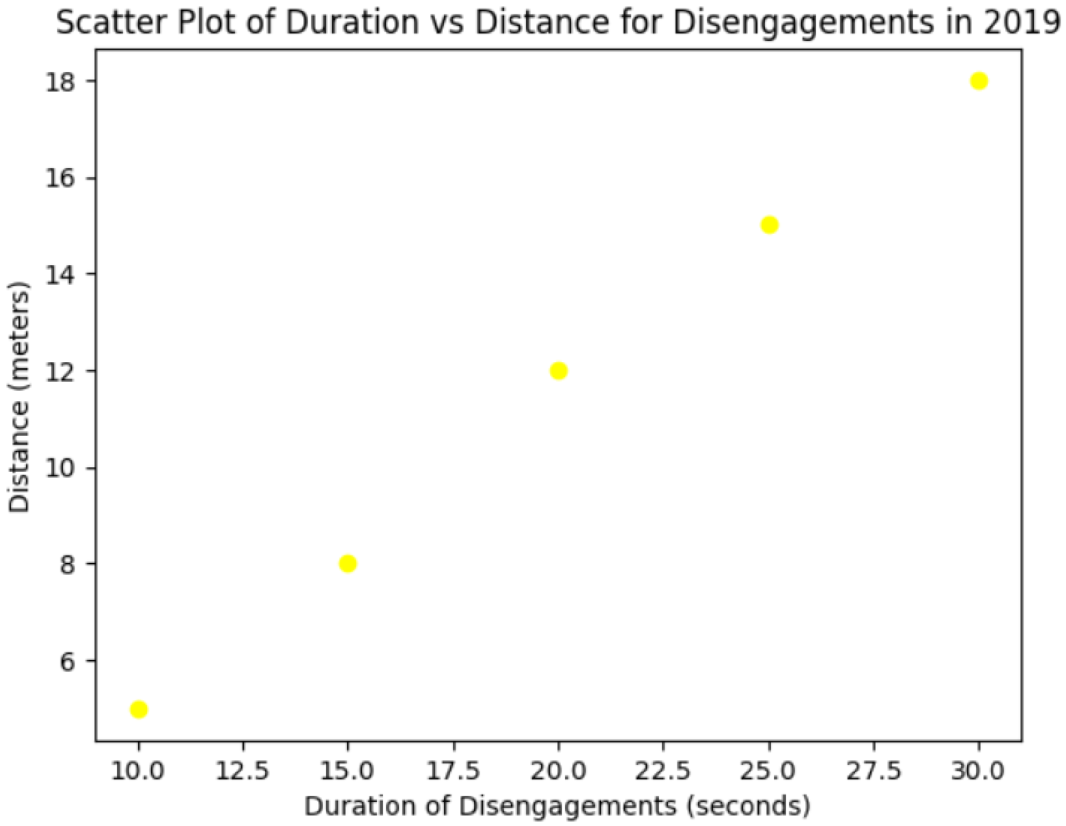
CODE

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

disengagement_data = pd.DataFrame({
    'duration': [10, 15, 20, 25, 30],
    'distance': [5, 8, 12, 15, 18]
})
```

```
plt.scatter(disengagement_data['duration'],
disengagement_data['distance'],
color='yellow')
plt.xlabel('Duration of Disengagements
(seconds)')
plt.ylabel('Distance (meters)')
plt.title('Scatter Plot of Duration vs
Distance for Disengagements in 2019')
plt.show()
```

OUTPUT



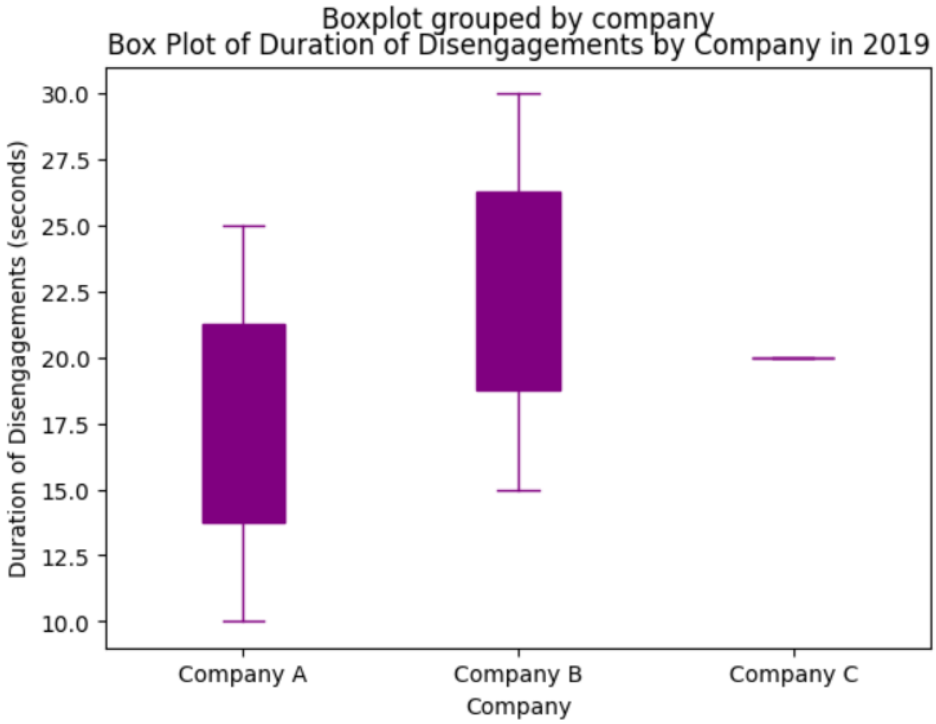
Box Plots

Box plots summarize the distribution of a continuous variable by displaying its median, quartiles, and any outliers. They are useful for comparing the distributions of the same variable across different groups or categories, providing insights into central tendency, spread, and skewness.

CODE

```
import pandas as pd
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'company': ['Company A', 'Company B', 'Company C', 'Company A', 'Company B'],
    'duration': [10, 15, 20, 25, 30]
})
disengagement_data.boxplot(column='duration', by='company', grid=False, patch_artist=True, showfliers=False, color='purple')
plt.xlabel('Company')
plt.ylabel('Duration of Disengagements (seconds)')
plt.title('Box Plot of Duration of Disengagements by Company in 2019')
plt.show()
```

OUTPUT



3.MULTIVARIATE VISUALIZATIONS

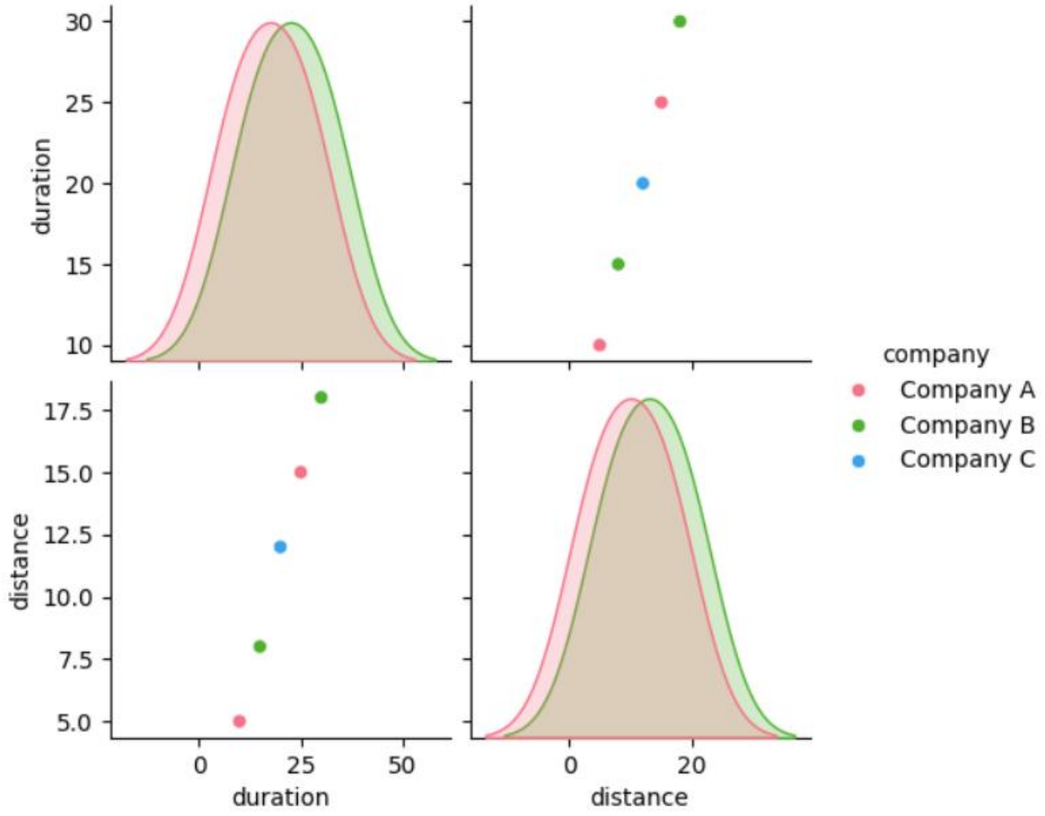
Pair Plot:

Pair plots, also known as scatterplot matrices, are a type of multivariate visualization. They display scatter plots of each variable in a dataset paired with every other variable.

CODE

```
import pandas as pd
import seaborn as sns
disengagement_data = pd.DataFrame({
    'duration': [10, 15, 20, 25, 30],
    'distance': [5, 8, 12, 15, 18],
    'company': ['Company A', 'Company B',
'Company C', 'Company A', 'Company B']
})
sns.pairplot(disengagement_data,
hue='company', palette='husl',
diag_kind='kde')
plt.show()
```

OUTPUT



4.INTERRACTIVE VISUALIZATIONS

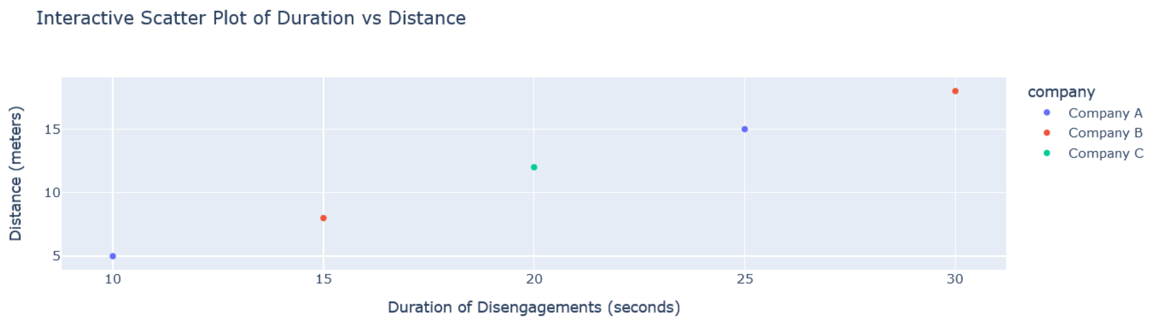
Interactive Scatter Plots:

Dynamic visualizations where users can hover over data points for details and manipulate the view through zooming and panning.

CODE

```
import pandas as pd
import plotly.express as px
disengagement_data = pd.DataFrame({
    'duration': [10, 15, 20, 25, 30],
    'distance': [5, 8, 12, 15, 18],
    'company': ['Company A', 'Company B', 'Company C', 'Company A', 'Company B']
})
fig = px.scatter(disengagement_data, x='duration', y='distance', color='company',
                 title='Interactive Scatter Plot of Duration vs Distance',
                 labels={'duration': 'Duration of Disengagements (seconds)', 'distance': 'Distance (meters)'},
                 hover_name='company')
fig.show()
```

OUTPUT



Interactive Dashboards:

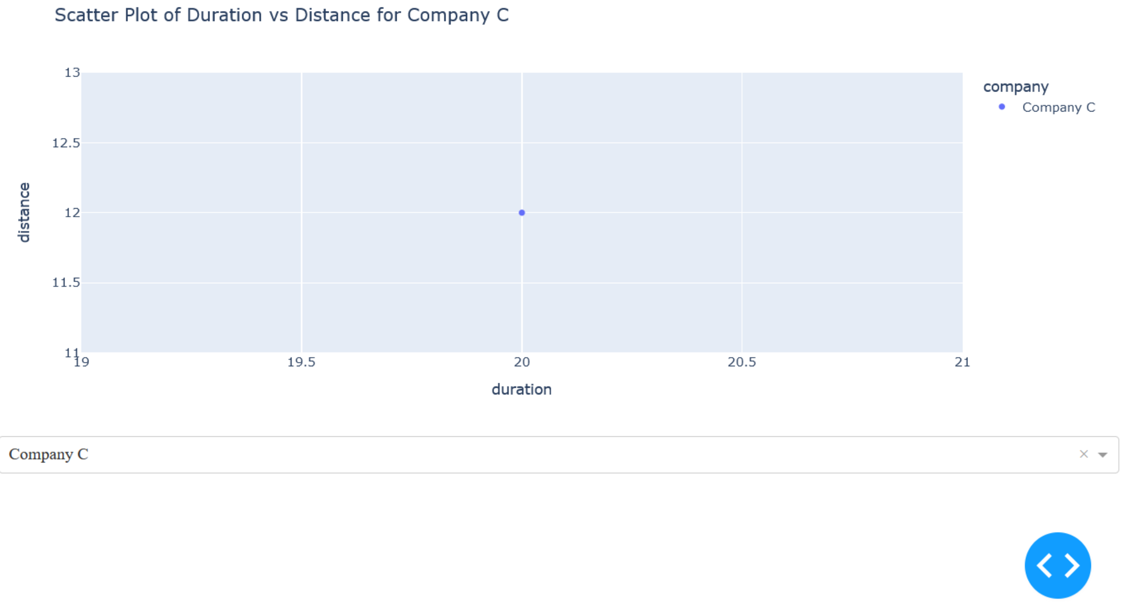
User interfaces that integrate multiple visualizations, allowing users to interactively explore data and gain insights through filtering and selection.

CODE

```
import pandas as pd
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.express as px
disengagement_data = pd.DataFrame({
    'duration': [10, 15, 20, 25, 30],  # Sample duration data
    'distance': [5, 8, 12, 15, 18],     # Sample distance data
    'company': ['Company A', 'Company B', 'Company C', 'Company A', 'Company B']
})
app = dash.Dash(__name__)
app.layout = html.Div([
    dcc.Graph(id='scatter-plot'),
    dcc.Dropdown(
        id='company-dropdown',
        options=[{'label': company, 'value':
company} for company in
disengagement_data['company'].unique()],

value=disengagement_data['company'].uni
que()[0]
    )
])
@app.callback(
    Output('scatter-plot', 'figure'),
    [Input('company-dropdown', 'value')]
)
def
update_scatter_plot(selected_company):
    filtered_data =
disengagement_data[disengagement_data['
company'] == selected_company]
    fig = px.scatter(filtered_data,
x='duration', y='distance', color='company',
                    title=f'Scatter Plot of
Duration vs Distance for
{selected_company}')
    return fig
if __name__ == '__main__':
    app.run_server(debug=True)
```

OUTPUT



Company C

SOME OTHER VISUALIZATION TECHNIQUES

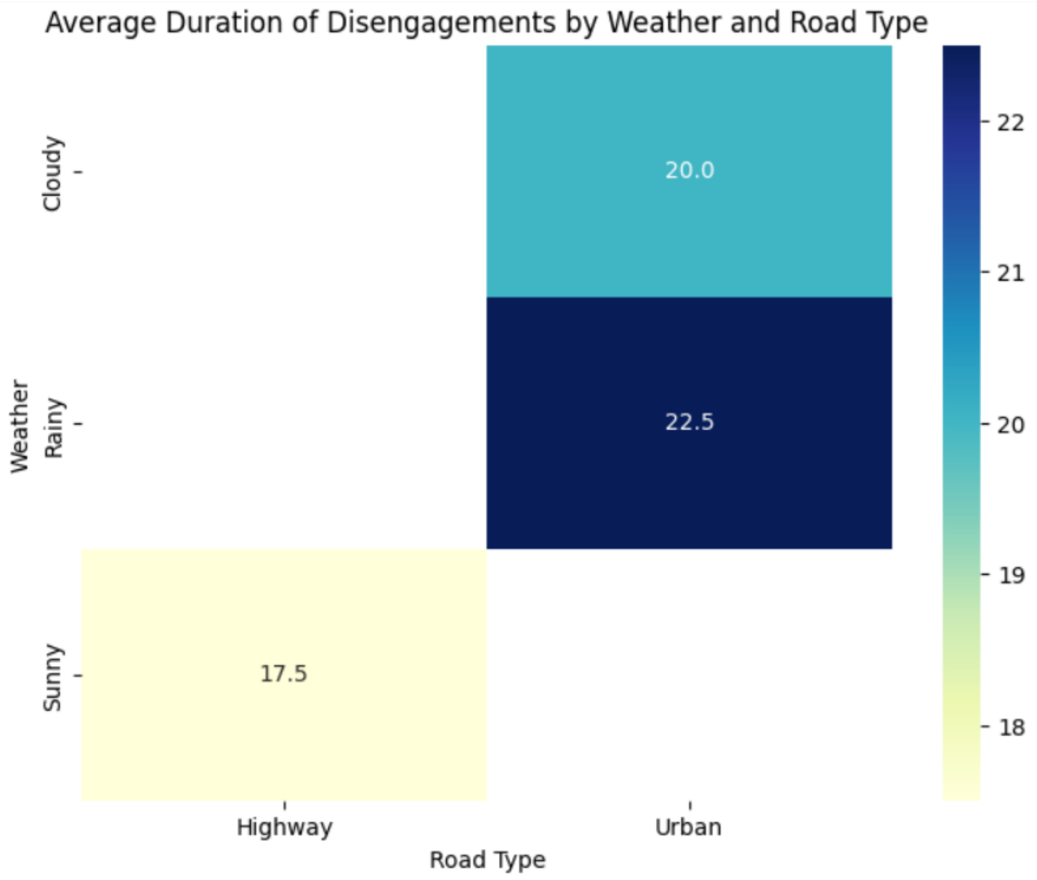
Heat Map:

A graphical representation of data where values in a matrix are represented as colors. Heat maps are commonly used to visualize the intensity of data points within a two-dimensional grid, with colors ranging from low to high values.

CODE

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'weather': ['Sunny', 'Rainy', 'Cloudy',
'Sunny', 'Rainy'],
    'road_type': ['Highway', 'Urban', 'Urban',
'Highway', 'Urban'],
    'duration': [10, 15, 20, 25, 30]
})
heatmap_data = disengagement_data.pivot_table(index='weather', columns='road_type', values='duration', aggfunc='mean')
plt.figure(figsize=(8, 6))
sns.heatmap(heatmap_data, annot=True, cmap='YlGnBu', fmt='.1f')
plt.xlabel('Road Type')
plt.ylabel('Weather')
plt.title('Average Duration of Disengagements by Weather and Road Type')
plt.show()
```

OUTPUT



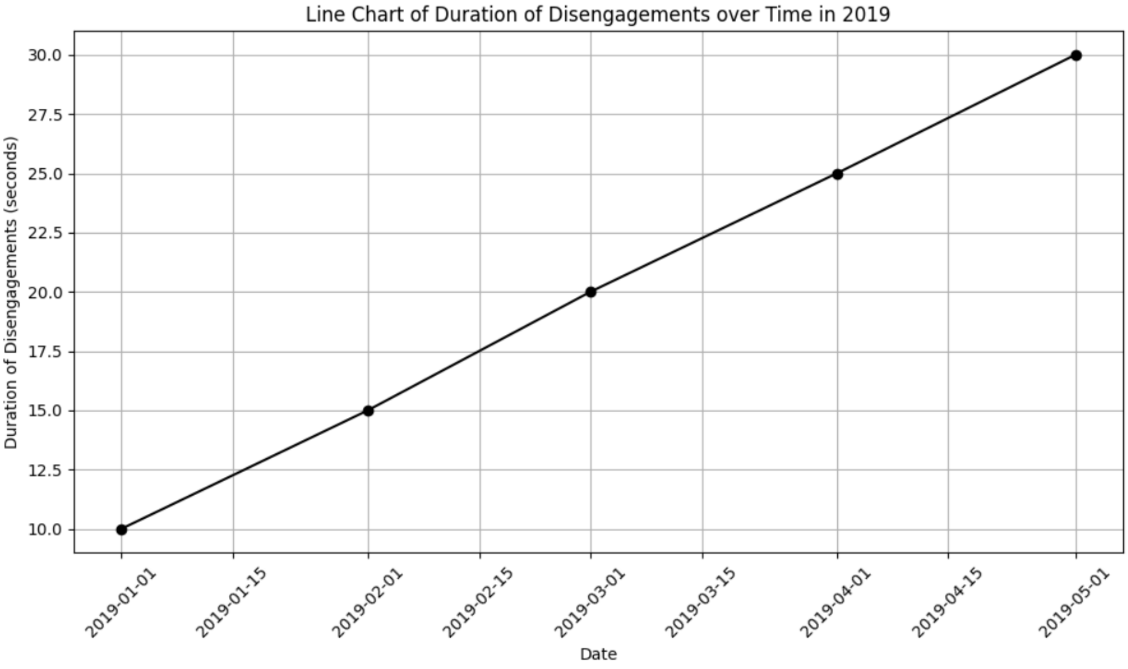
Line Chart:

A graphical representation of data where data points are connected by straight lines. Line charts are commonly used to show trends and changes in data over time or to display continuous data series.

CODE

```
import pandas as pd
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'date': ['2019-01-01', '2019-02-01', '2019-03-01', '2019-04-01', '2019-05-01'],
    'duration': [10, 15, 20, 25, 30]
})
disengagement_data['date'] = pd.to_datetime(disengagement_data['date'])
disengagement_data = disengagement_data.sort_values(by='date')
plt.figure(figsize=(10, 6))
plt.plot(disengagement_data['date'], disengagement_data['duration'], marker='o', color='black', linestyle='-')
plt.xlabel('Date')
plt.ylabel('Duration of Disengagements (seconds)')
plt.title('Line Chart of Duration of Disengagements over Time in 2019')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

OUTPUT



Pie Chart:

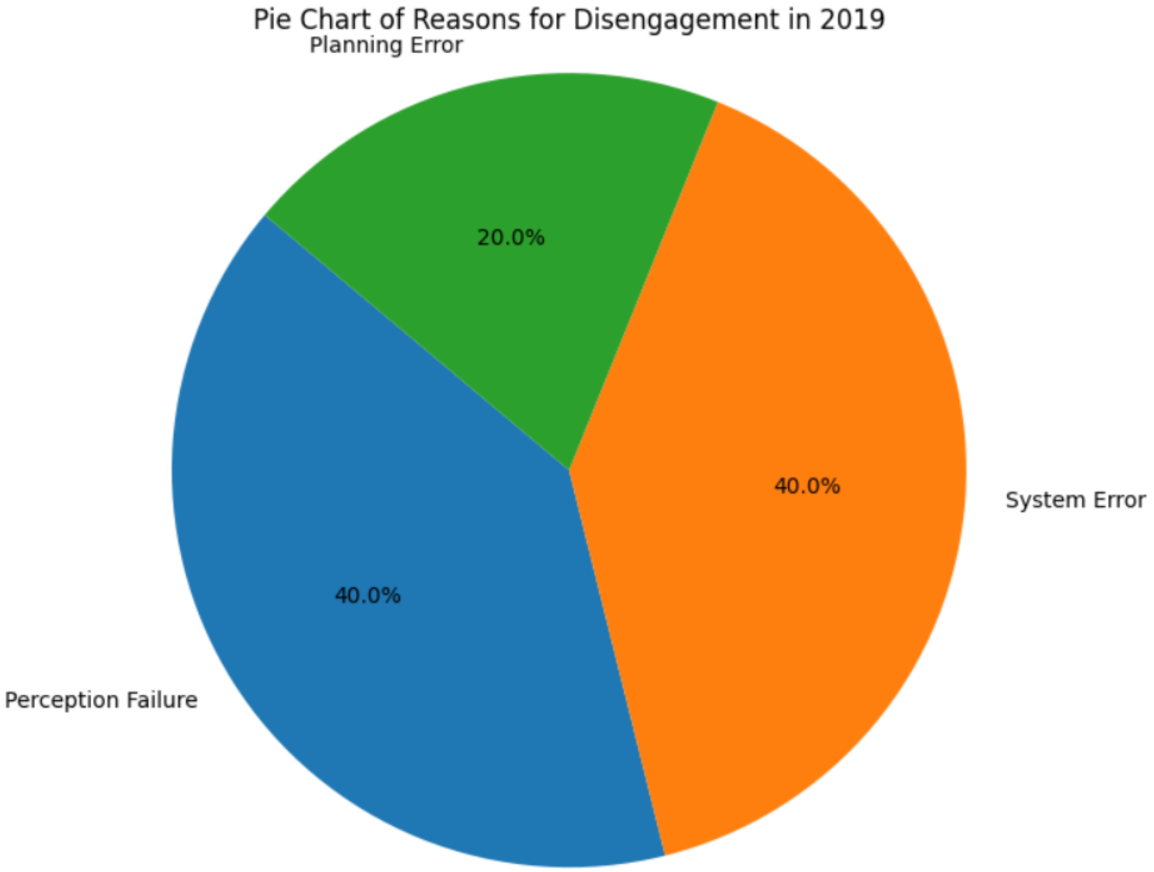
A circular statistical graphic that is divided into slices to illustrate numerical proportions. Each slice represents a proportionate part of the whole data set, making it easy to visualize the relative sizes of different categories.

CODE

```
import pandas as pd
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'reason': ['Perception Failure', 'System Error', 'Planning Error', 'Perception Failure', 'System Error']
})
reason_counts = disengagement_data['reason'].value_counts()
```

```
plt.figure(figsize=(8, 6))
plt.pie(reason_counts,
labels=reason_counts.index,
autopct='%1.1f%%',
colors=plt.cm.tab10.colors,
startangle=140)
plt.title('Pie Chart of Reasons for
Disengagement in 2019')
plt.axis('equal')
plt.tight_layout()
plt.show()
```

OUTPUT



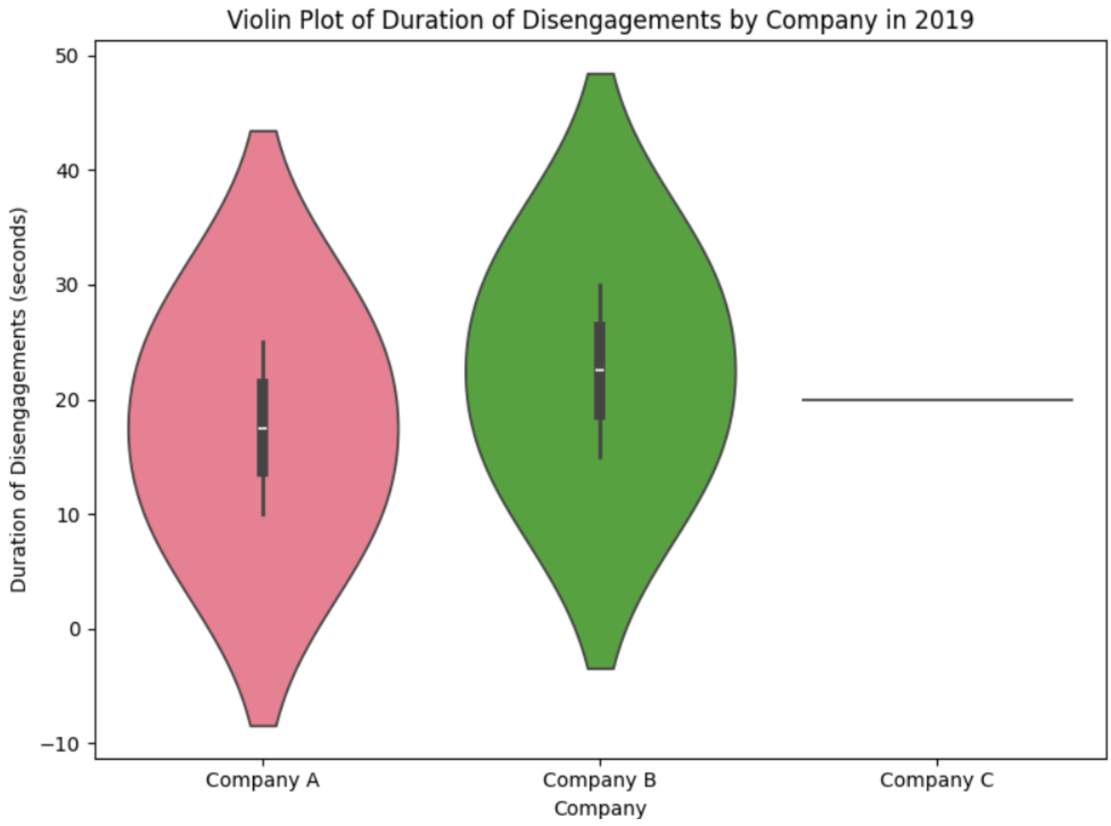
Violin Plot:

A method of plotting numeric data and probability density functions of different categories. Violin plots display the distribution of data across different levels of a categorical variable, similar to a box plot, but also show the probability density of the data at different values.

CODE

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'company': ['Company A', 'Company B', 'Company C', 'Company A', 'Company B'],
    'duration': [10, 15, 20, 25, 30]  # Sample duration data
})
plt.figure(figsize=(8, 6))
sns.violinplot(x='company', y='duration', data=disengagement_data, palette='husl')
plt.xlabel('Company')
plt.ylabel('Duration of Disengagements (seconds)')
plt.title('Violin Plot of Duration of
Disengagements by Company in 2019')
plt.tight_layout()
plt.show()
```

OUTPUT



Area Chart:

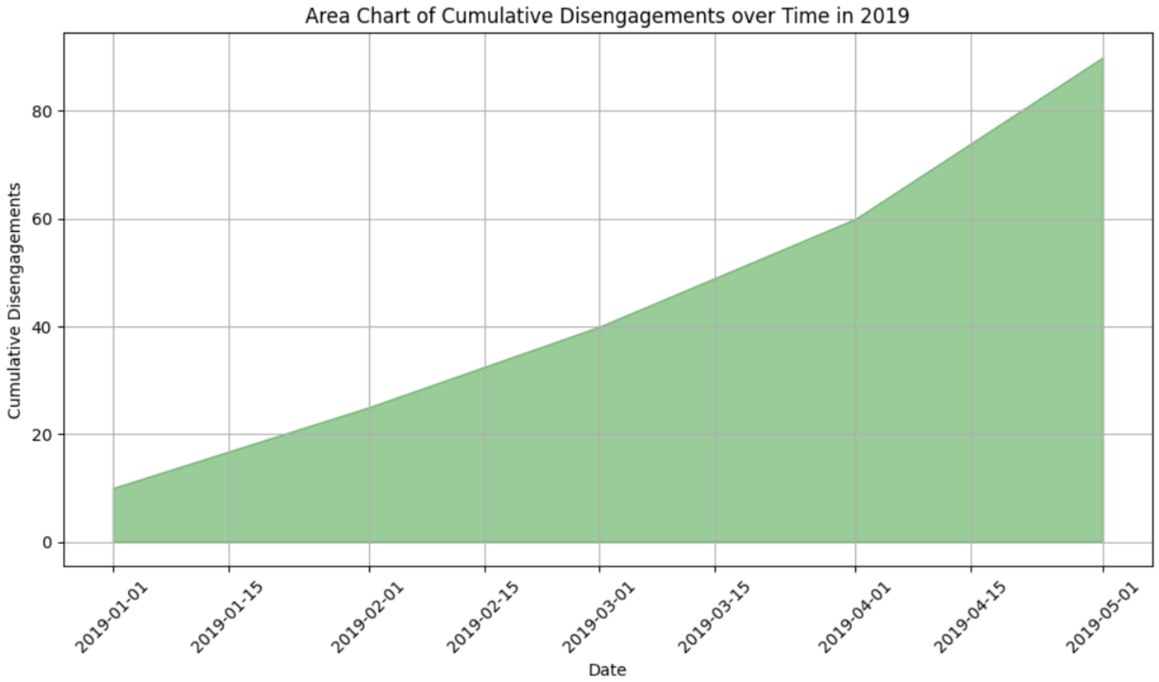
A graphical representation of data where the area under the curve is filled with color. Area charts are commonly used to represent cumulative totals over time or to show the composition of a whole over time.

CODE

```
import pandas as pd
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'date': ['2019-01-01', '2019-02-01',
'2019-03-01', '2019-04-01', '2019-05-01'],
    'cumulative_disengagements': [10, 25,
40, 60, 90]
})
disengagement_data['date'] =
pd.to_datetime(disengagement_data['date']
)
disengagement_data =
disengagement_data.sort_values(by='date')
```

```
plt.figure(figsize=(10, 6))
plt.fill_between(disengagement_data['date'
],
disengagement_data['cumulative_disengag
ements'], color='green', alpha=0.4)
plt.xlabel('Date')
plt.ylabel('Cumulative Disengagements')
plt.title('Area Chart of Cumulative
Disengagements over Time in 2019')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

OUTPUT



Stacked Bar Chart:

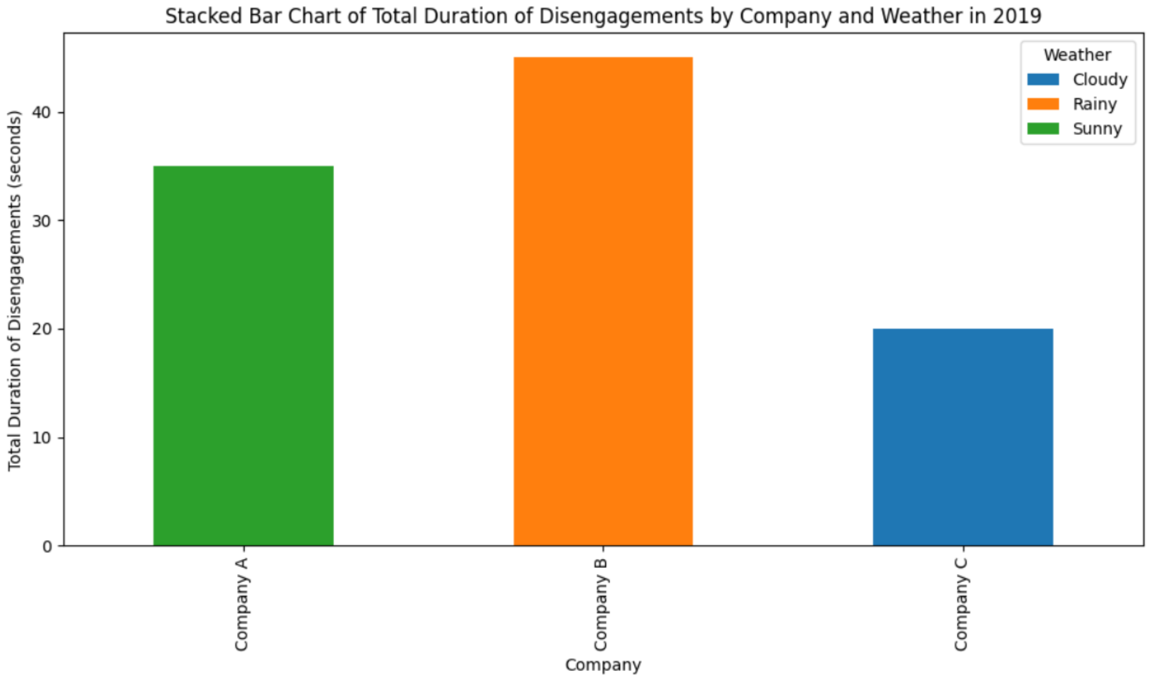
A type of bar chart where multiple data series are stacked on top of each other. Each bar represents a categorical variable, and the height of the bar represents the total value. Stacked bar charts are useful for comparing the total and individual contributions of different categories.

CODE

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


disengagement_data = pd.DataFrame({
    'company': ['Company A', 'Company B', 'Company C', 'Company A', 'Company B'],
    'weather': ['Sunny', 'Rainy', 'Cloudy', 'Sunny', 'Rainy'],
    'duration': [10, 15, 20, 25, 30]  # Sample duration data
})
grouped_data = disengagement_data.groupby(['company', 'weather']).agg(total_duration=('duration', 'sum')).reset_index()
pivot_data = grouped_data.pivot(index='company',
columns='weather', values='total_duration')
pivot_data.plot(kind='bar', stacked=True, figsize=(10, 6))
plt.xlabel('Company')
plt.ylabel('Total Duration of Disengagements (seconds)')
plt.title('Stacked Bar Chart of Total Duration of Disengagements by Company and Weather in 2019')
plt.legend(title='Weather')
plt.tight_layout()
plt.show()
```

OUTPUT



Network Graph:

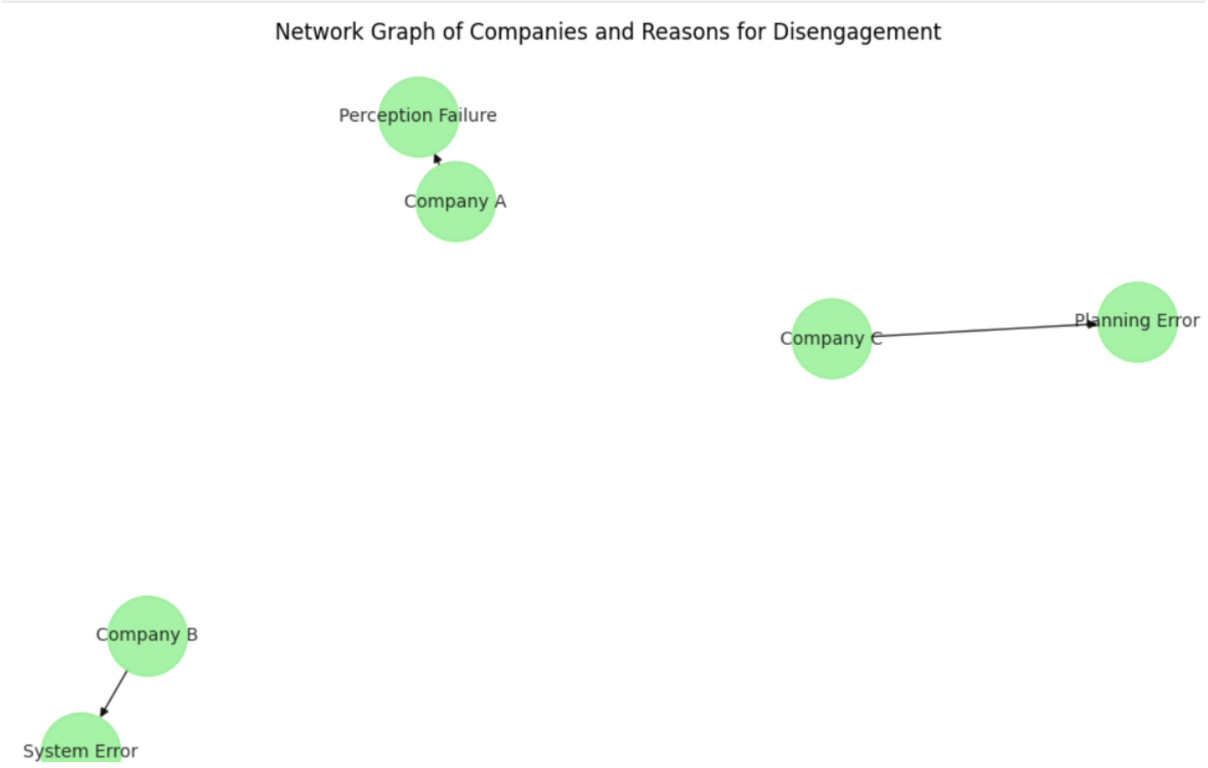
A graphical representation of a network or interconnected data. Network graphs consist of nodes (representing entities) and edges (representing connections between entities). They are commonly used in fields like social network analysis, transportation networks, and biological networks.

CODE

```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'company': ['Company A', 'Company B', 'Company C', 'Company A', 'Company B'],
    'reason': ['Perception Failure', 'System Error', 'Planning Error', 'Perception Failure', 'System Error']
})
G = nx.DiGraph()
for _, row in disengagement_data.iterrows():
    G.add_edge(row['company'], row['reason'])
plt.figure(figsize=(10, 6))
pos = nx.spring_layout(G)
```

```
nx.draw(G, pos, with_labels=True,
node_size=2000, font_size=10,
node_color='lightgreen',
font_color='black', alpha=0.8)
plt.title('Network Graph of Companies and
Reasons for Disengagement')
plt.show()
```

OUTPUT



Word Graph:

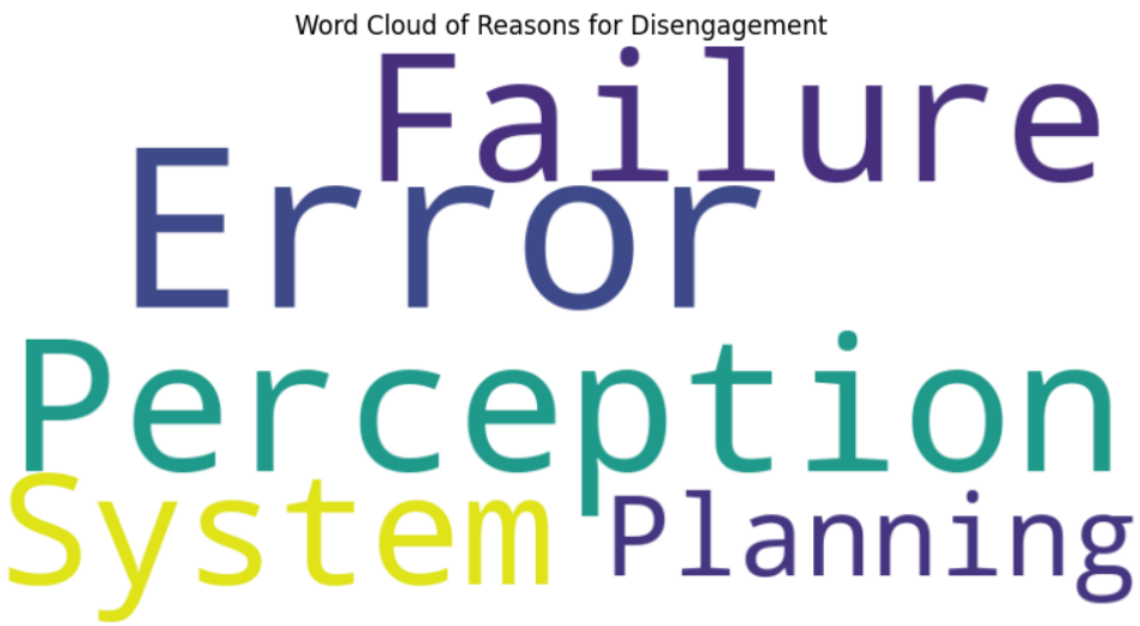
A graphical representation of text data where words are nodes connected by edges based on their co-occurrence or semantic relationships. Word graphs can be used to visualize the structure and connections between words in a text corpus.

CODE

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'reason': ['Perception Failure', 'System Error', 'Planning Error', 'Perception Failure', 'System Error']
})
reason_text = ' '.join(disengagement_data['reason'])
```

```
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(reason_text)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Word Cloud of Reasons for Disengagement')
plt.axis('off')
plt.show()
```

OUTPUT


Word Cloud of Reasons for Disengagement

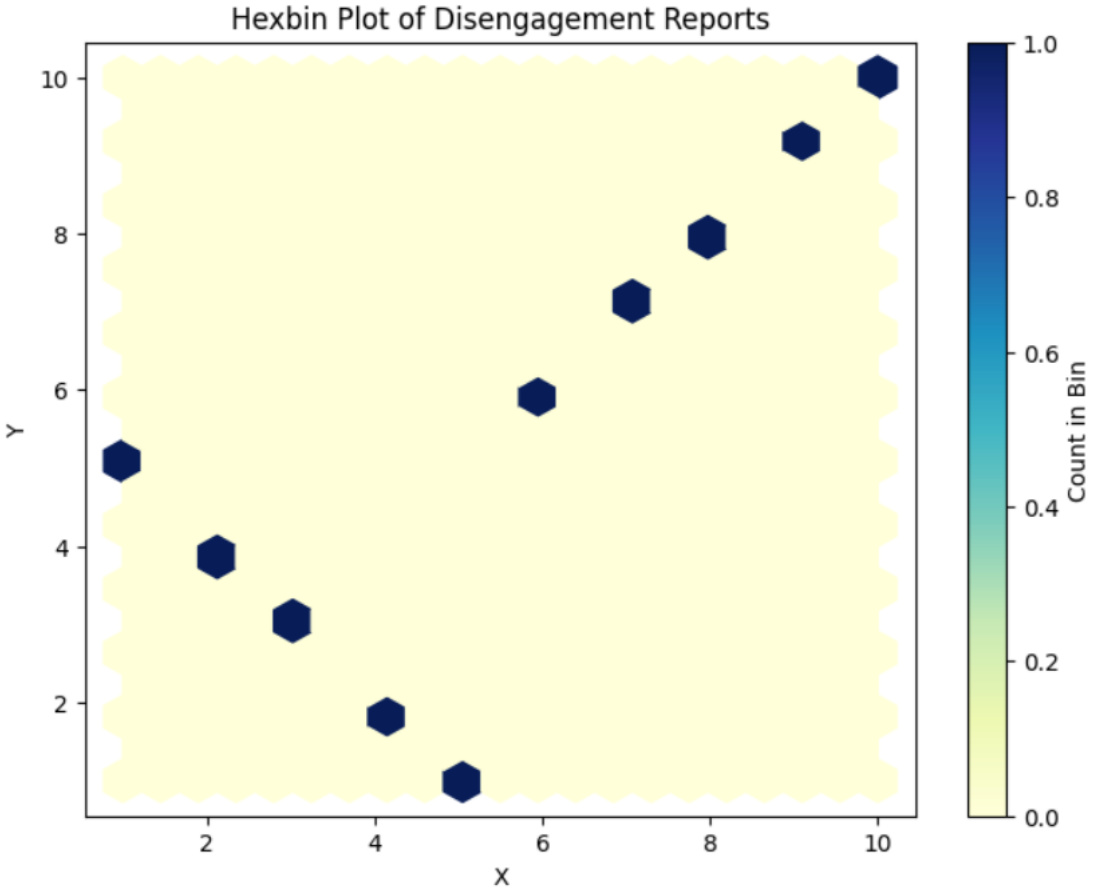
Hexbin Plots:

A two-dimensional binning technique for visualizing the distribution of data points in a scatter plot. Hexbin plots divide the plot area into hexagonal bins and count the number of data points within each bin, representing this count with color intensity. They are useful for visualizing dense regions of data and identifying patterns in scatter plots.

CODE

```
import pandas as pd
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'x': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'y': [5, 4, 3, 2, 1, 6, 7, 8, 9, 10]
})
plt.figure(figsize=(8, 6))
plt.hexbin(disengagement_data['x'], disengagement_data['y'], gridsize=20, cmap='YlGnBu')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Hexbin Plot of Disengagement Reports')
plt.colorbar(label='Count in Bin')
plt.show()
```

OUTPUT



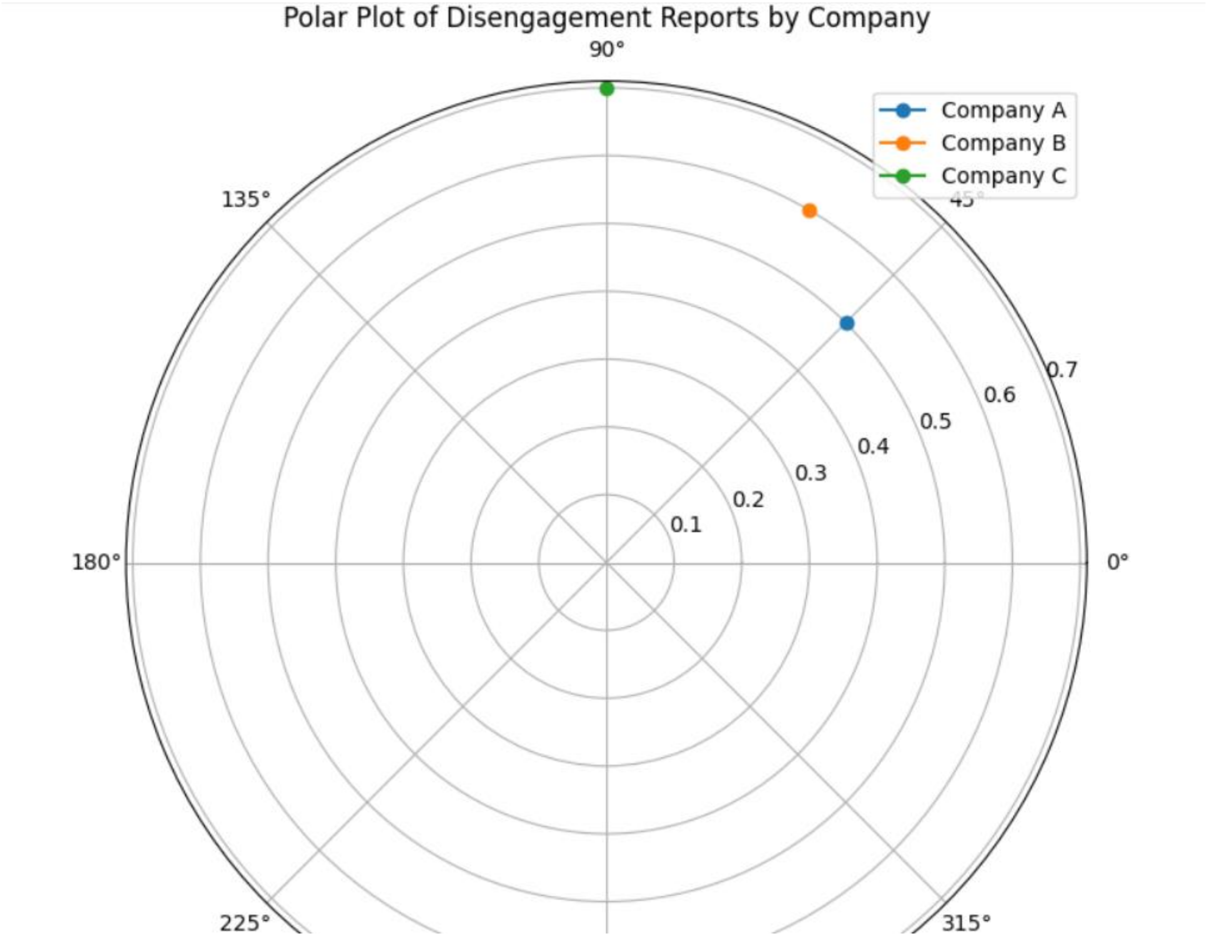
Polar Plot:

A graphical representation of data in polar coordinates, where data points are plotted radially and angularly. Polar plots are commonly used to visualize cyclic patterns, such as seasonal variations, or directional data, such as wind direction.

CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'company': ['Company A', 'Company B', 'Company C'],
    'distance_from_center': [0.5, 0.6, 0.7],
    'angle': [np.pi/4, np.pi/3, np.pi/2]
})
fig, ax = plt.subplots(subplot_kw={'projection': 'polar'}, figsize=(8, 8))
for _, row in disengagement_data.iterrows():
    ax.plot(row['angle'], row['distance_from_center'], marker='o', label=row['company'])
ax.set_title('Polar Plot of Disengagement Reports by Company')
ax.legend(loc='upper right')
plt.show()
```

OUTPUT



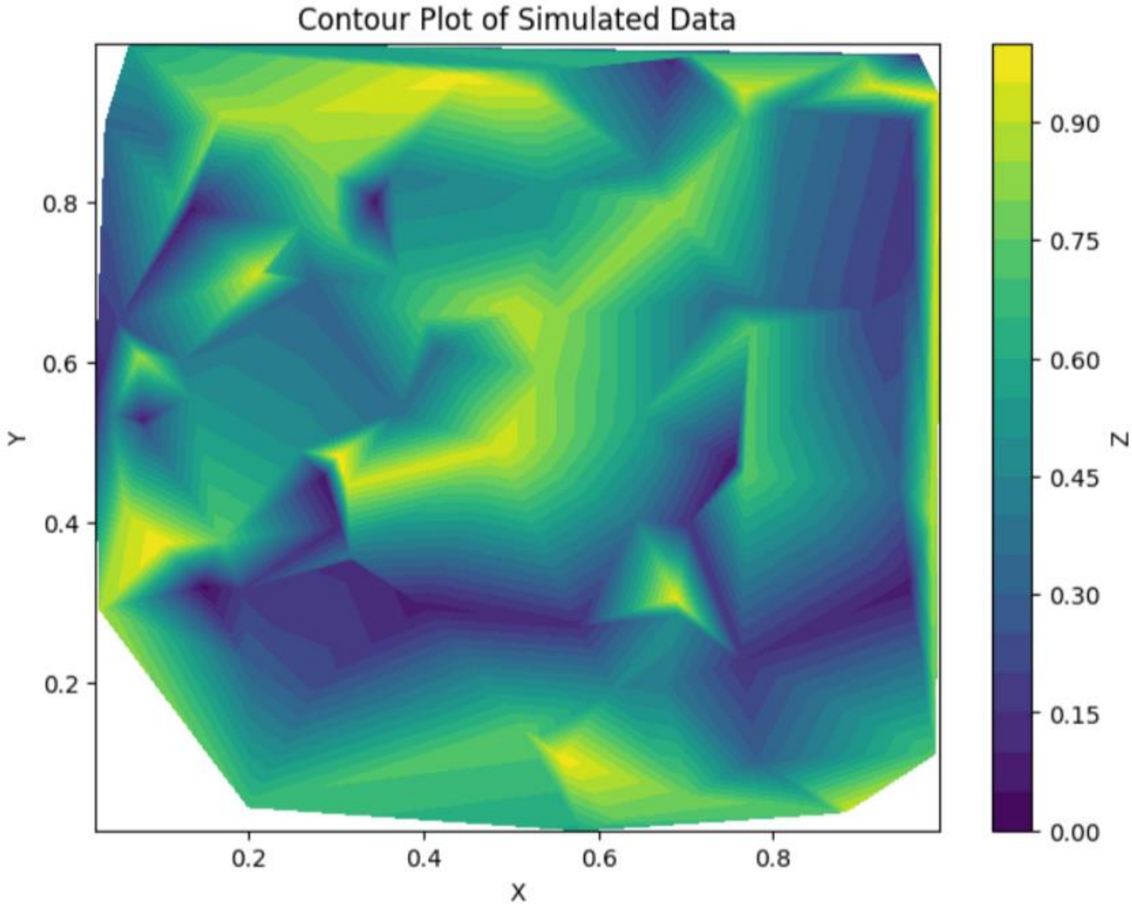
Contour Plot:

A graphical representation of three-dimensional data on a two-dimensional plane, where contours (lines of equal value) are used to represent data values. Contour plots are useful for visualizing the variation of a continuous variable across a surface or landscape.

CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
x = np.random.rand(100)
y = np.random.rand(100)
z = np.random.rand(100)
plt.figure(figsize=(8, 6))
plt.tricontourf(x, y, z, levels=20,
cmap='viridis')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Contour Plot of Simulated Data')
plt.colorbar(label='Z')
plt.show()
```

OUTPUT



Bubble Chart:

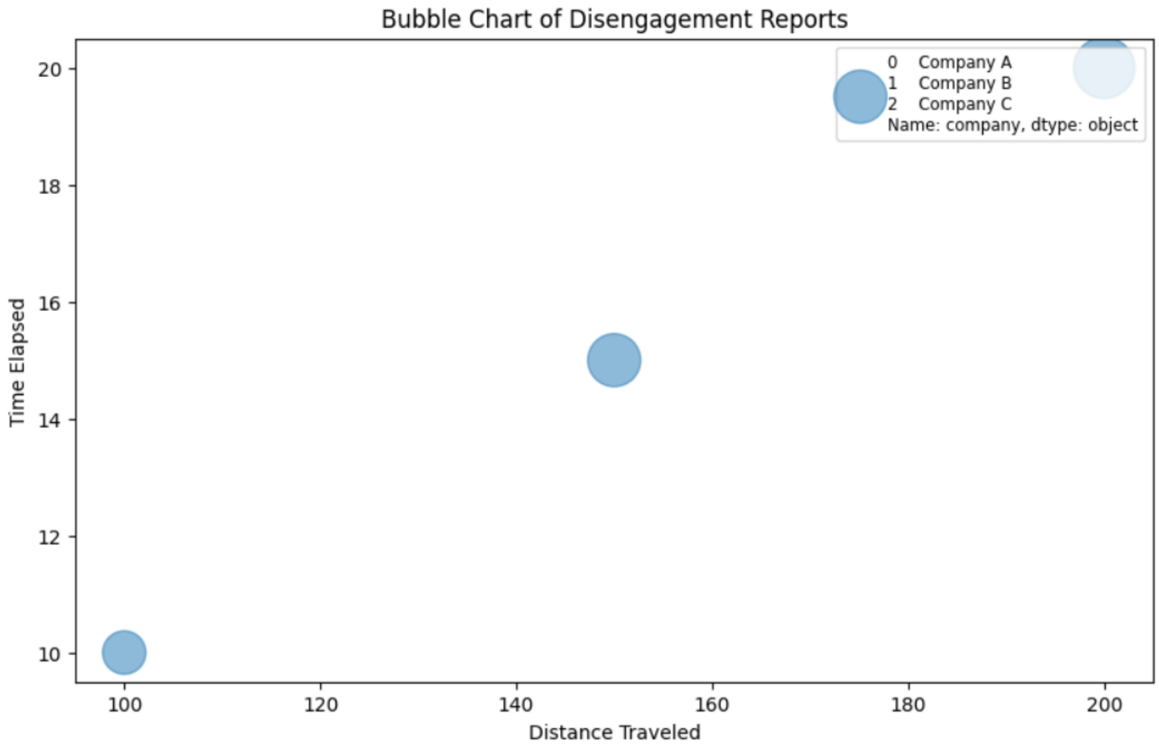
A type of scatter plot where data points are represented as bubbles, with the size of each bubble encoding a third variable. Bubble charts are effective for visualizing relationships between three variables, particularly when comparing multiple categories or groups.

CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'company': ['Company A', 'Company B',
'Company C'],
    'distance_traveled': [100, 150, 200],
    'time_elapsed': [10, 15, 20],
    'disengagement_count': [10, 15, 20]
})
plt.figure(figsize=(10, 6))
plt.scatter(disengagement_data['distance_tr
aveled'],
disengagement_data['time_elapsed'],

s=disengagement_data['disengagement_co
unt'] * 50, alpha=0.5,
label=disengagement_data['company'])
plt.xlabel('Distance Traveled')
plt.ylabel('Time Elapsed')
plt.title('Bubble Chart of Disengagement
Reports')
plt.legend(loc='upper right',
fontsize='small')
plt.show()
```

OUTPUT



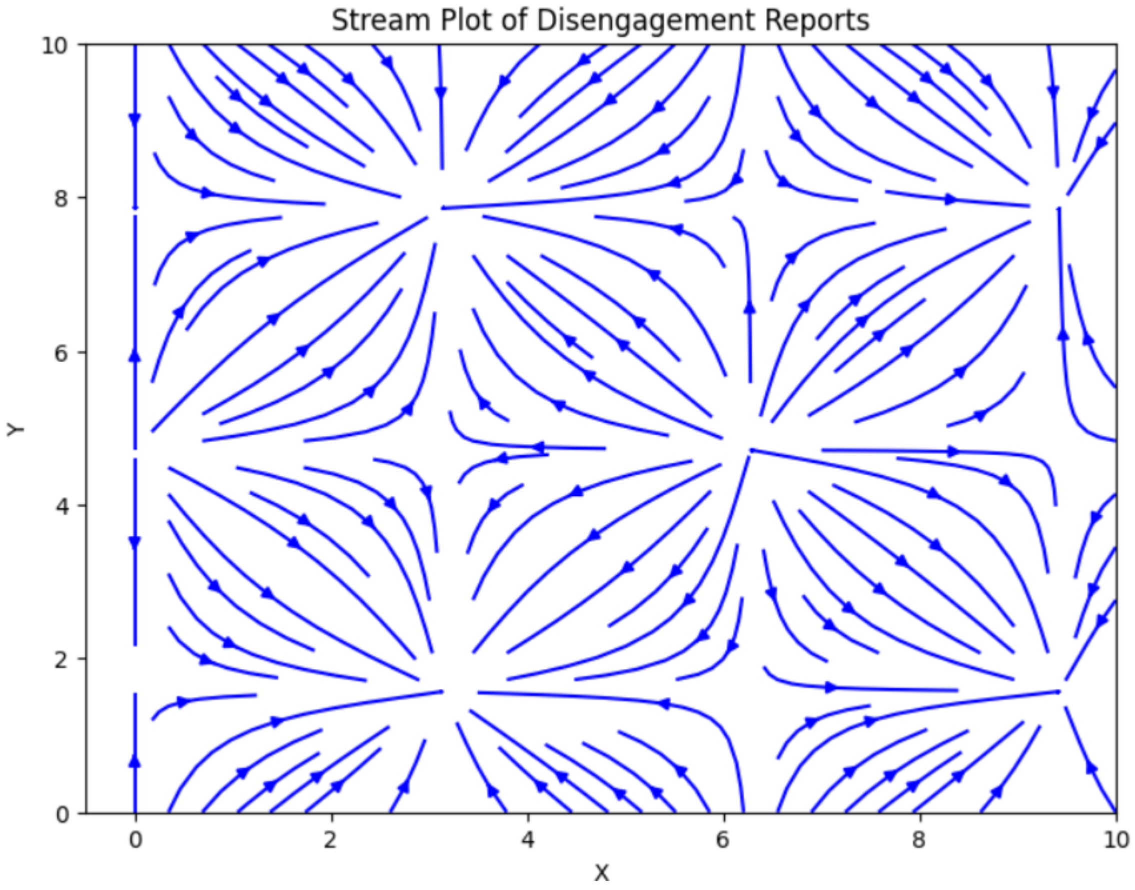
Stream Plot:

A type of vector field visualization that represents the flow of a vector field using streamlines. Stream plots are commonly used in fluid dynamics and meteorology to visualize the direction and magnitude of fluid flow or wind patterns.

CODE

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 10, 20)
y = np.linspace(0, 10, 20)
X, Y = np.meshgrid(x, y)
U = np.sin(X)
V = np.cos(Y)
plt.figure(figsize=(8, 6))
plt.streamplot(X, Y, U, V, color='b')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Stream Plot of Disengagement Reports')
plt.show()
```

OUTPUT



Quiver Plot:

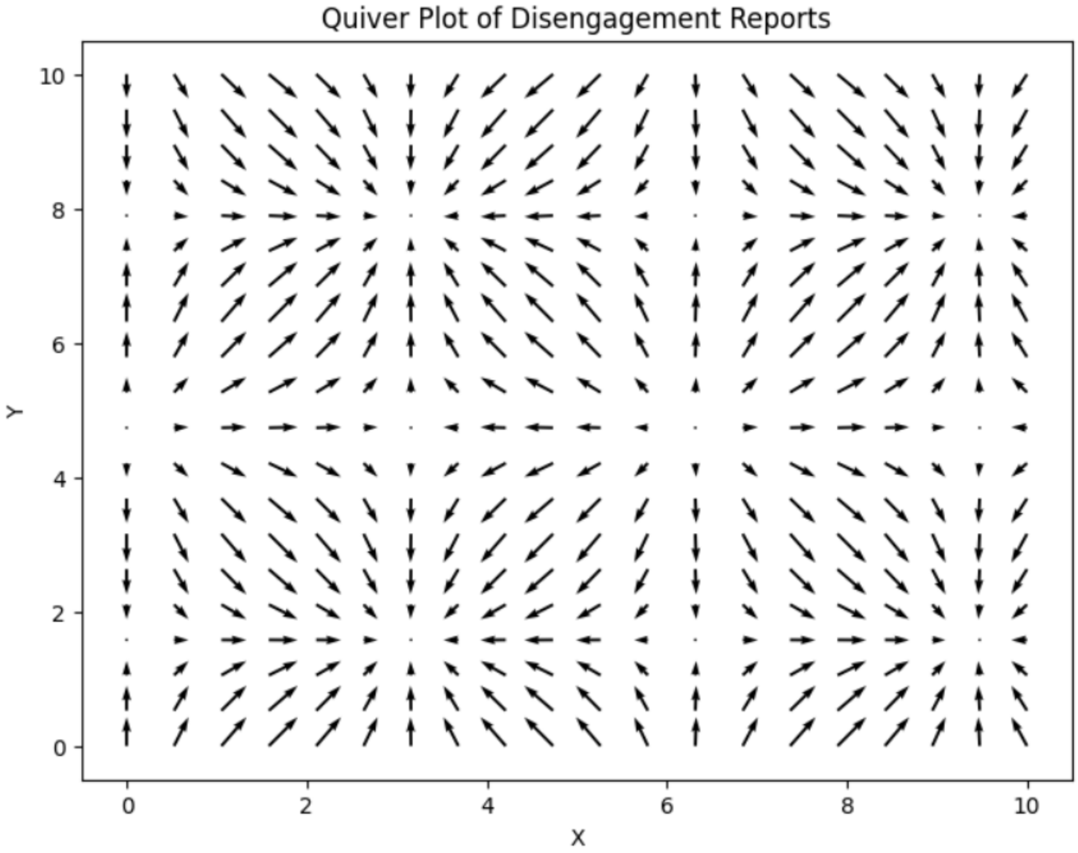
A type of vector field visualization similar to stream plots, where arrows (quivers) are used to represent the direction and magnitude of vectors at different points in space. Quiver plots are useful for visualizing vector fields in physics, engineering, and geosciences.

CODE

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 10, 20)
y = np.linspace(0, 10, 20)
X, Y = np.meshgrid(x, y)
U = np.sin(X)
V = np.cos(Y)
plt.figure(figsize=(8, 6))
plt.quiver(X, Y, U, V)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Quiver Plot of Disengagement
Reports')
```

```
plt.show()
```

OUTPUT



Violin Swarm Plot:

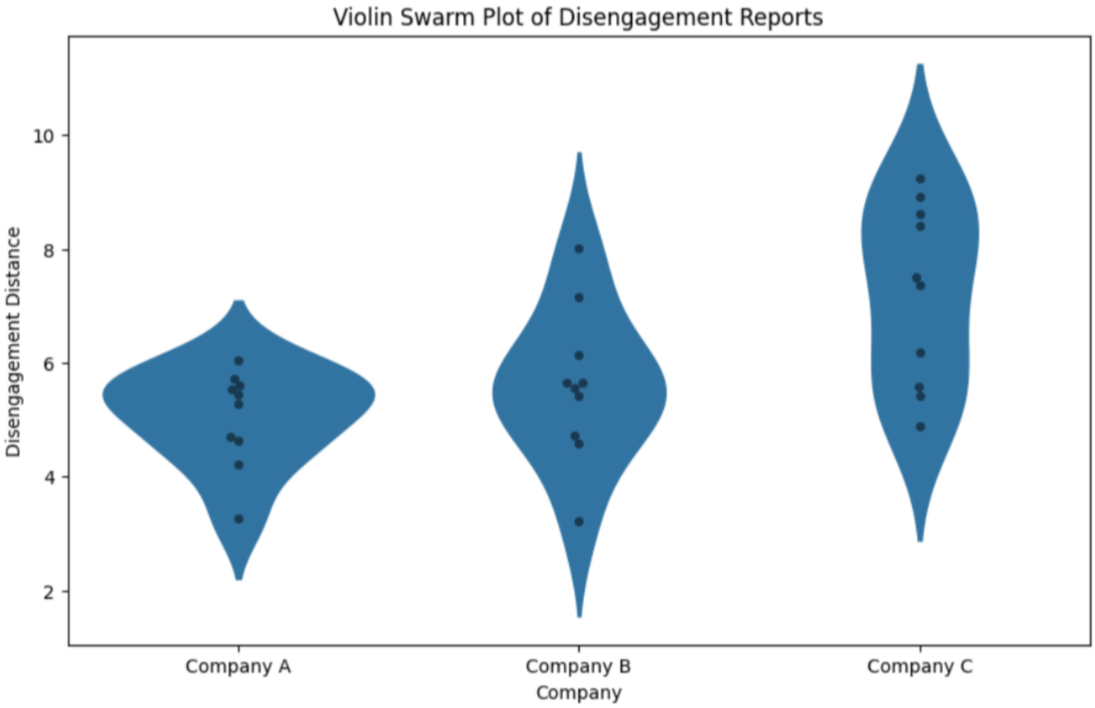
A combination of violin plots and swarm plots, where violin plots display the distribution of data within each category, and swarm plots jitter individual data points to prevent overlap and show their distribution more clearly. Violin swarm plots are useful for visualizing both the overall distribution of data and the individual data points within each category.

CODE

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
disengagement_data = pd.DataFrame({
    'company': ['Company A'] * 10 + ['Company B'] * 10 + ['Company C'] * 10,
    'disengagement_distance': np.concatenate([np.random.normal(5, 1, 10), np.random.normal(6, 1, 10), np.random.normal(7, 1, 10)])
})
```

```
plt.figure(figsize=(10, 6))
sns.violinplot(x='company',
y='disengagement_distance',
data=disengagement_data, inner=None,
linewidth=0)
sns.swarmplot(x='company',
y='disengagement_distance',
data=disengagement_data, color='k',
alpha=0.5)
plt.xlabel('Company')
plt.ylabel('Disengagement Distance')
plt.title('Violin Swarm Plot of
Disengagement Reports')
```

OUTPUT


Violin Swarm Plot of Disengagement Reports
Disengagement Distance
10
8
6
4
2
Company A
Company B
Company C
Company