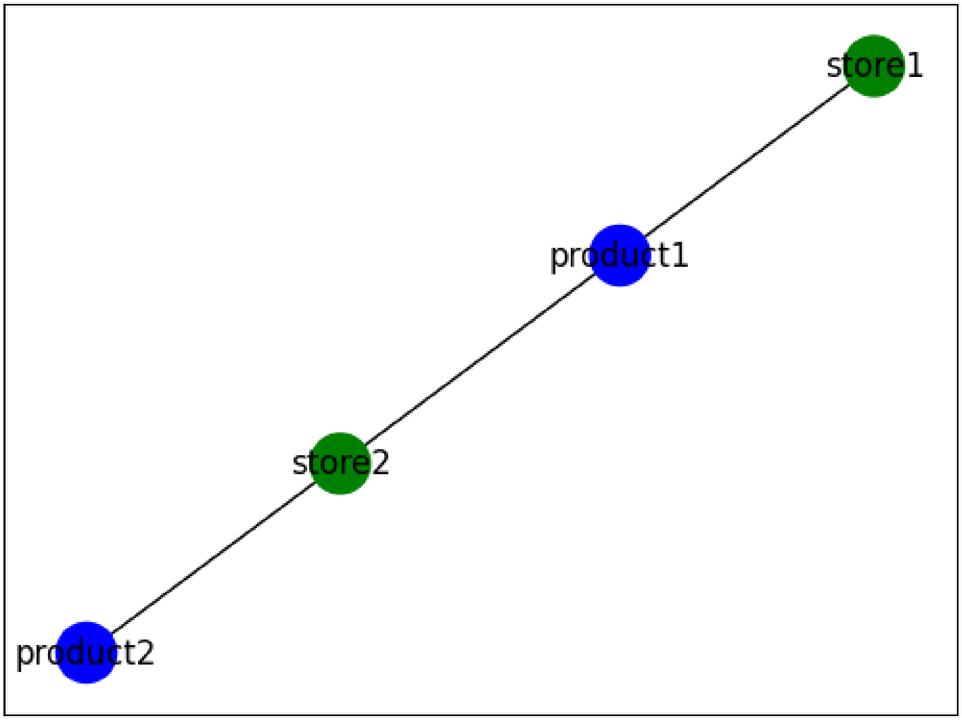
task-6-ex

August 20, 2023

Algorithm 1.Gather Data 2.Choose a Visualization Tool 3.Data Preparation 4.Create the Visualization Create Nodes and Edges Configure Layout Settings Iterative Layout Algorithm Visual Representation

5.Interpretation

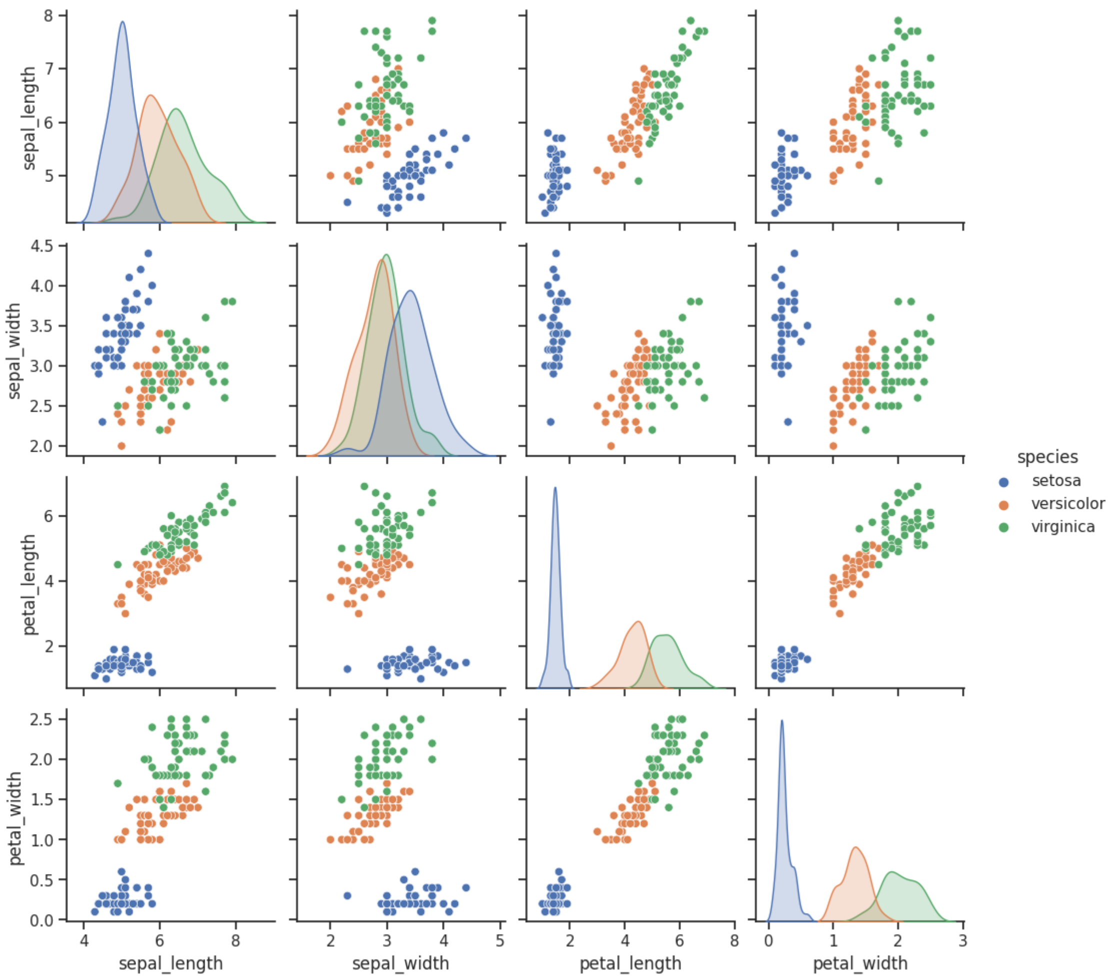
[7]:
```python
import networkx as nx
import matplotlib.pyplot as plt

# Sample data (replace this with your actual data)
nodes = [
    ("product1", {"type": "product"}),
    ("product2", {"type": "product"}),
    ("store1", {"type": "store"}),
    ("store2", {"type": "store"}),
]

edges = [
    ("product1", "store1", {"strength": 0.5}),
    ("product1", "store2", {"strength": 0.3}),
    ("product2", "store2", {"strength": 0.7}),
]

G = nx.Graph()
G.add_nodes_from(nodes)
G.add_edges_from(edges)

pos = nx.spring_layout(G)  # Force-based layout

node_colors = ["blue" if data["type"] == "product" else "green" for _, data in␣
 ↪G.nodes(data=True)]

nx.draw_networkx_nodes(G, pos, node_color=node_colors, node_size=500)
nx.draw_networkx_edges(G, pos)
nx.draw_networkx_labels(G, pos)

plt.show()
```

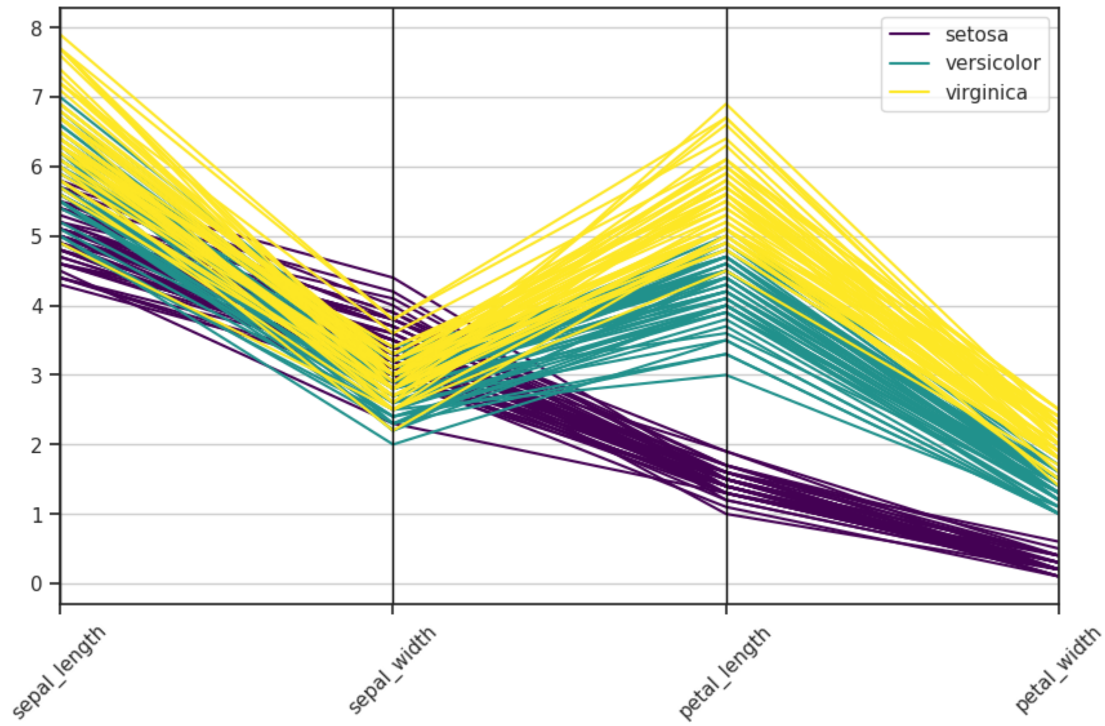


B Alorithm

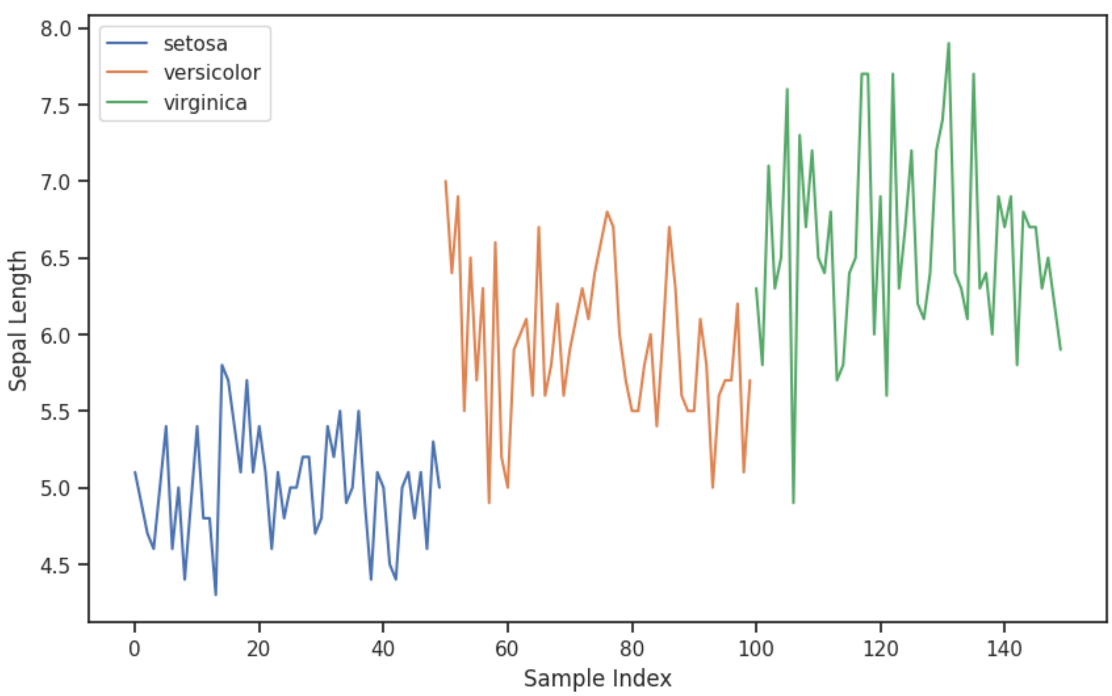
[9]:
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import parallel_coordinates

# Load the Iris dataset
iris = sns.load_dataset("iris")

# Scatterplot Matrix
sns.set(style="ticks")
sns.pairplot(iris, hue="species")
plt.show()

# Parallel Coordinates
plt.figure(figsize=(10, 6))
parallel_coordinates(iris, 'species', colormap='viridis')
plt.xticks(rotation=45)
plt.show()

# Line Graph
species_groups = iris.groupby('species')
plt.figure(figsize=(10, 6))
```

```
for species, group in species_groups:
    plt.plot(group.index, group['sepal_length'], label=species)

plt.xlabel('Sample Index')
plt.ylabel('Sepal Length')
plt.legend()
plt.show()
```





8
7
6
5
4
3
2
1
0
setosa
versicolor
virginica
sepal_length
sepal_width
petal_length
petal_width



8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
Sepal Length
setosa
versicolor
virginica
0
20
40
60
80
100
120
140
Sample Index

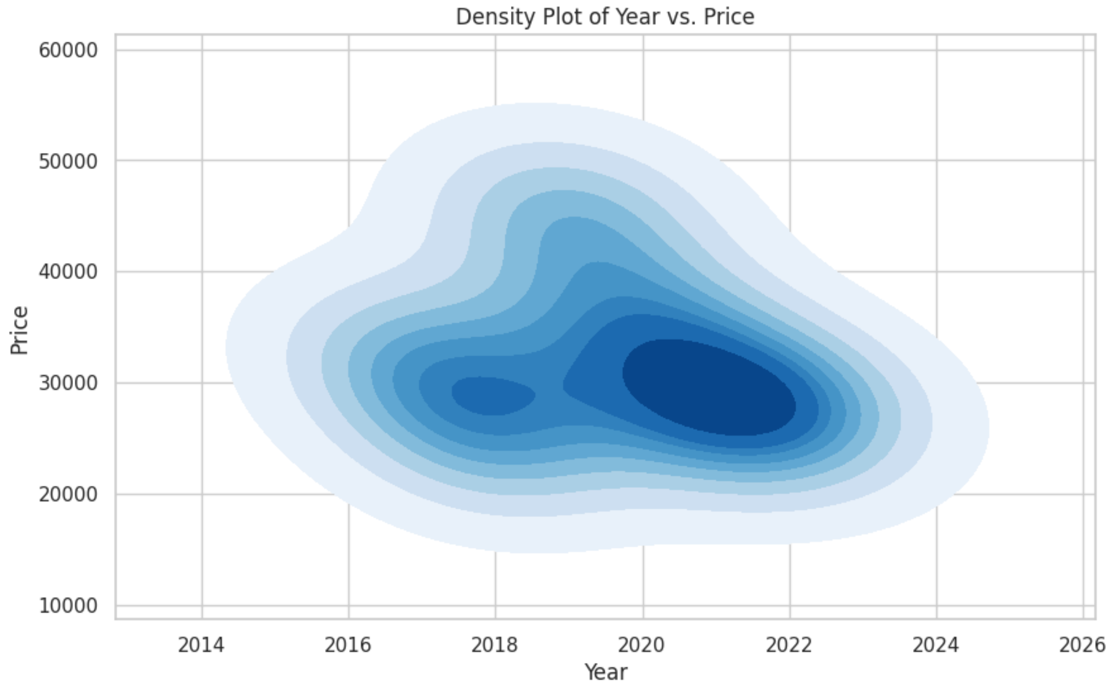
task-7-exe

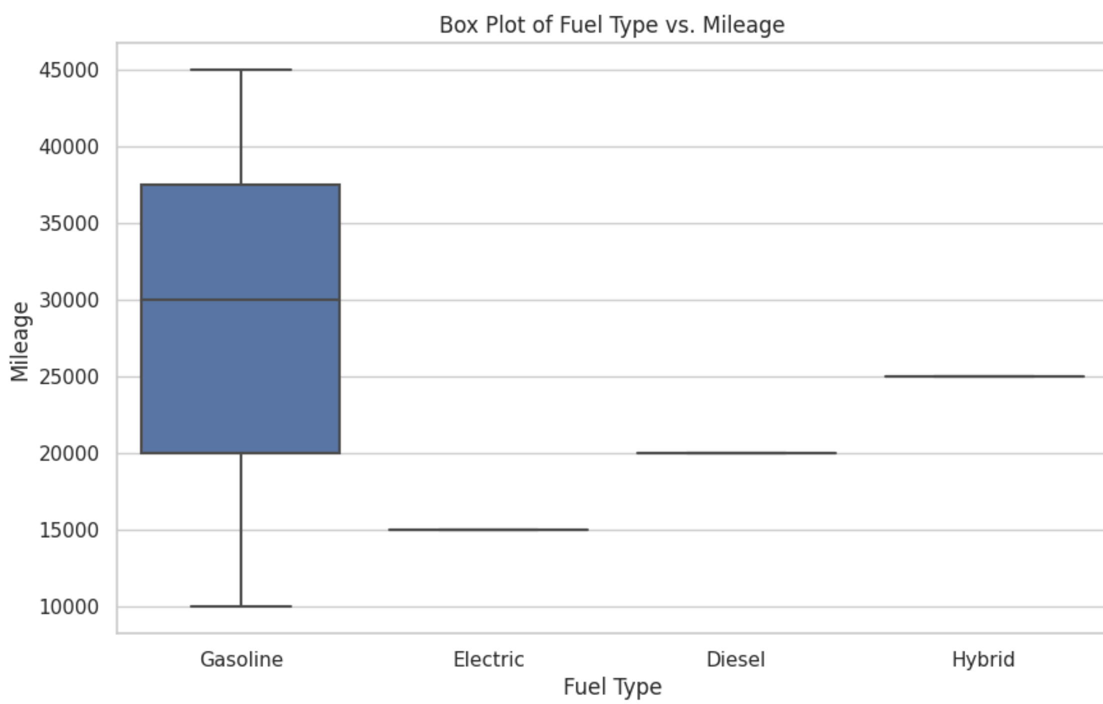
August 20, 2023

Task - 7A create a car dataset and perform Bivariate analysis for continuous and categorical data using Density Plots, Box Plots

[4]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


data = {
    'Car Model': ['Sedan', 'SUV', 'Convertible', 'Hatchback', 'SUV', 'Sedan'],
    'Manufacturer': ['Toyota', 'Ford', 'BMW', 'Honda', 'Ford', 'Toyota'],
    'Year': [2018, 2020, 2019, 2022, 2017, 2021],
    'Price': [25000, 35000, 45000, 28000, 32000, 27000],
    'Mileage': [30000, 15000, 10000, 20000, 45000, 25000],
    'Fuel Type': ['Gasoline', 'Electric', 'Gasoline', 'Diesel', 'Gasoline',␣
 ↪'Hybrid'],
    'Transmission': ['Automatic', 'Automatic', 'Manual', 'Automatic', 'Manual',␣
 ↪'Automatic']
}

car_df = pd.DataFrame(data)

sns.set(style="whitegrid")

# Density Plot for Year vs. Price
plt.figure(figsize=(10, 6))
sns.kdeplot(data=car_df, x='Year', y='Price', cmap='Blues', fill=True)
plt.title('Density Plot of Year vs. Price')
plt.xlabel('Year')
plt.ylabel('Price')
plt.show()

# Box Plot for Fuel Type vs. Mileage
plt.figure(figsize=(10, 6))
sns.boxplot(data=car_df, x='Fuel Type', y='Mileage')
plt.title('Box Plot of Fuel Type vs. Mileage')
plt.xlabel('Fuel Type')
plt.ylabel('Mileage')
```

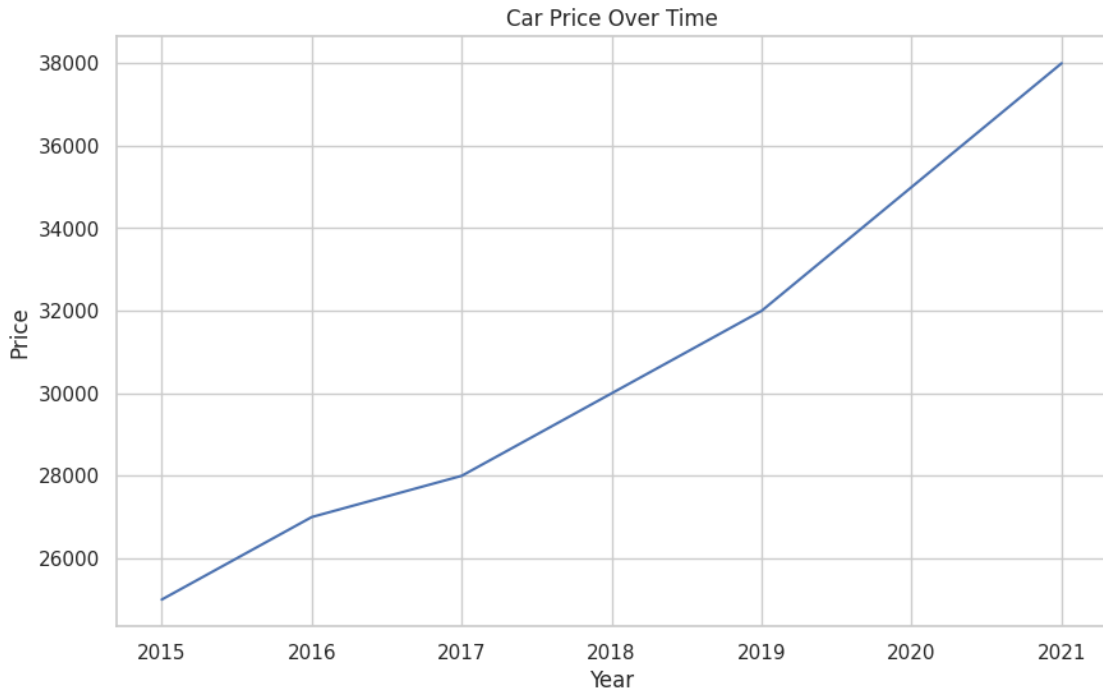
```
plt.show()
```

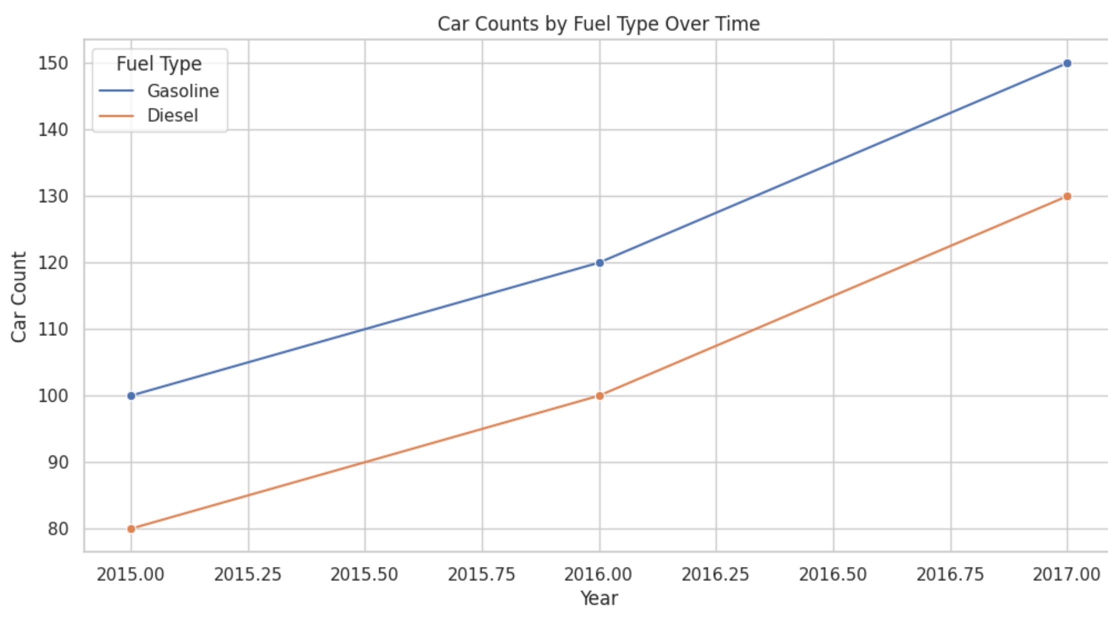

Density Plot of Year vs. Price
60000
50000
40000
30000
20000
10000
Price
2014
2016
2018
2020
2022
2024
2026
Year


Box Plot of Fuel Type vs. Mileage
45000
40000
35000
30000
25000
20000
15000
10000
Mileage
Gasoline
Electric
Diesel
Hybrid
Fuel Type

7 B analyze and visualize Time Oriented Data Analysis to identify systemic patterns in the data that help to form trends, cycles or seasonal variances and to forecast the data. - Line Graph, Trend Lines, Area Chart for the above dataset.

[13]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Create a sample dataset for price over time
price_data = {
    'Year': [2015, 2016, 2017, 2018, 2019, 2020, 2021],
    'Price': [25000, 27000, 28000, 30000, 32000, 35000, 38000]
}

# Create a sample dataset for car counts by fuel type over time
car_counts_data = {
    'Year': [2015, 2015, 2016, 2016, 2017, 2017],
    'Fuel Type': ['Gasoline', 'Diesel', 'Gasoline', 'Diesel', 'Gasoline',␣
↪'Diesel'],
    'Car Count': [100, 80, 120, 100, 150, 130]
}

# Create DataFrames from the sample datasets
price_df = pd.DataFrame(price_data)
car_counts_df = pd.DataFrame(car_counts_data)

# Create a line plot with trend line for price over time
plt.figure(figsize=(10, 6))
sns.set(style="whitegrid")
sns.lineplot(data=price_df, x='Year', y='Price')
plt.title('Car Price Over Time')
plt.xlabel('Year')
plt.ylabel('Price')
plt.show()

# Create an area chart for car counts by fuel type over time
plt.figure(figsize=(12, 6))
sns.set(style="whitegrid")
sns.lineplot(data=car_counts_df, x='Year', y='Car Count', hue='Fuel Type',␣
↪marker='o')
plt.title('Car Counts by Fuel Type Over Time')
plt.xlabel('Year')
plt.ylabel('Car Count')
plt.legend(title='Fuel Type')
plt.show()
```


Car Price Over Time
38000
36000
34000
32000
30000
28000
26000
Price
2015
2016
2017
2018
2019
2020
2021
Year


Car Counts by Fuel Type Over Time
Fuel Type
Gasoline
Diesel
150
140
130
120
110
100
90
80
Car Count
2015.00
2015.25
2015.50
2015.75
2016.00
2016.25
2016.50
2016.75
2017.00
Year

task-8-exe

August 20, 2023

Task 8A A.Use any one of the movie data set Read the movie data file provided and store it in a dataframe movies. • Display the data types of each column using the attribute dtype • Inspect the dataframe for dimensions, null-values, and summary of different numeric columns. • Clean up the dataset to remove columns that are not informative to us for visualization

[ ]:
```
import pandas as pd
import seaborn as sns

# Load the Titanic dataset from Seaborn
titanic_data = sns.load_dataset("titanic")

# Display the data types of each column
print("Data Types of Each Column:")
print(titanic_data.dtypes)

# Inspect the dataframe dimensions
print("\nDimensions of the DataFrame:")
print(titanic_data.shape)

# Check for null values
print("\nNull Values in the DataFrame:")
print(titanic_data.isnull().sum())

# Summary of numeric columns
print("\nSummary of Numeric Columns:")
print(titanic_data.describe())

# List of columns not informative for visualization
columns_to_remove = ['deck', 'embark_town', 'alive', 'adult_male']
titanic_cleaned = titanic_data.drop(columns=columns_to_remove)

# Display cleaned DataFrame
print("\nCleaned DataFrame:")
print(titanic_cleaned.head())
```

```
Data Types of Each Column:
survived          int64
pclass            int64
```

```
sex               object
age              float64
sibsp              int64
parch              int64
fare             float64
embarked          object
class           category
who               object
adult_male          bool
deck            category
embark_town       object
alive             object
alone               bool
dtype: object

Dimensions of the DataFrame:
(891, 15)

Null Values in the DataFrame:
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64

Summary of Numeric Columns:
         survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200

Cleaned DataFrame:
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  alone
0    man  False
1  woman  False
2  woman   True
3  woman  False
4    man   True
```

Task 8B Construct a sunburst display for movie with high ratings

[ ]:
```python
import pandas as pd
import plotly.express as px

# Create a sample dataset (replace with your actual dataset)
data = {
    'Movie': ['Movie A', 'Movie B', 'Movie C', 'Movie D', 'Movie E'],
    'Genre': ['Action/Adventure', 'Comedy', 'Drama', 'Action/Adventure',␣
 ↪'Comedy'],
    'Subgenre': ['Action', 'Romantic Comedy', 'Thriller', 'Sci-Fi', 'Satire'],
    'Rating': [8.2, 7.5, 6.8, 9.0, 7.2]
}

# Create a DataFrame
df = pd.DataFrame(data)

# Filter movies with high ratings (example threshold)
high_rating_threshold = 7.5
high_rated_movies = df[df['Rating'] >= high_rating_threshold]

# Create a sunburst chart
fig = px.sunburst(
    high_rated_movies,
    path=['Genre', 'Subgenre', 'Movie'],
    values='Rating',
    color='Rating',
    color_continuous_scale='Blues',
    hover_data=['Rating']
)

# Update layout for better appearance
fig.update_layout(
    title="Sunburst Display for Movies with High Ratings",
```

```
    margin=dict(l=0, r=0, b=0, t=30)
)

# Show the plot
fig.show()
```

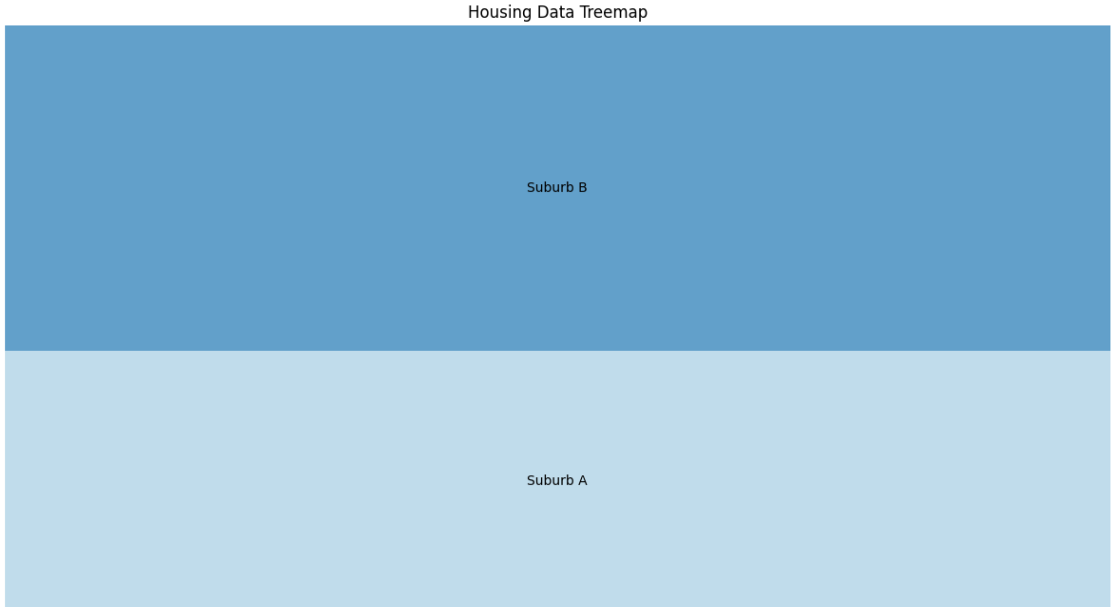
untitled9

August 20, 2023

Task 9A Consider a Housing dataset based housing id,size,room,price,location and construct a tree map and also sunburst display for the given dataset.

[ ]:
```
pip install matplotlib squarify plotly
```

[7]:
```
import matplotlib.pyplot as plt
import squarify
import plotly.express as px
import pandas as pd

# Sample data
data = [
    {"housing_id": 1, "size": 1200, "room": 3, "price": 250000, "location":␣
 ↪"Suburb A"},
    {"housing_id": 2, "size": 1500, "room": 4, "price": 320000, "location":␣
 ↪"Suburb B"},
    # ... more data ...
]

# Create DataFrame
df = pd.DataFrame(data)

# Treemap using matplotlib and squarify
plt.figure(figsize=(15, 8))
sizes = df['size']
labels = df['location']
colors = plt.cm.Paired(range(len(sizes)))
squarify.plot(sizes=sizes, label=labels, color=colors, alpha=0.7)
plt.axis('off')
plt.title('Housing Data Treemap')
plt.show()

# Sunburst using plotly
fig = px.sunburst(df, path=['location', 'room'], values='price')
fig.update_layout(title='Housing Data Sunburst')
fig.show()
```

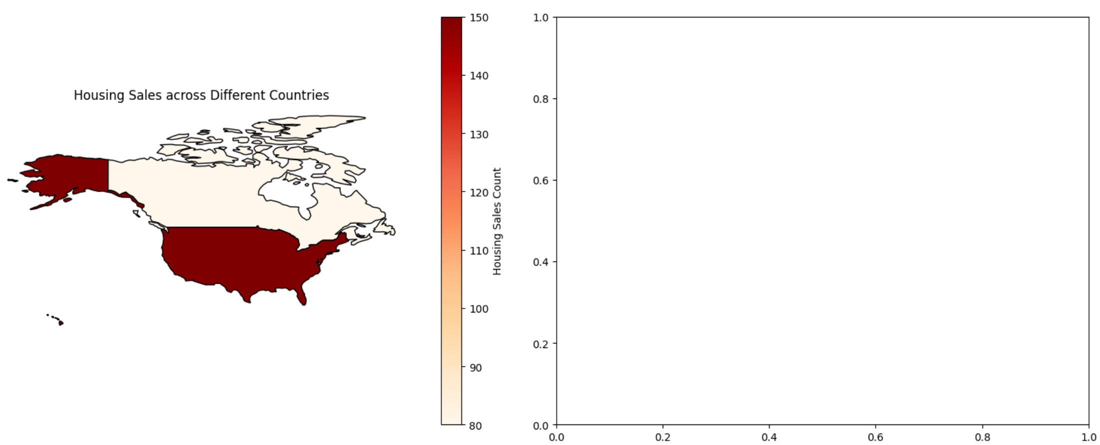


TAsk - 9B Visualize the Geospatial data of housing sales across different countries.

[10]:
```
import geopandas as gpd
import matplotlib.pyplot as plt
import pandas as pd
import plotly.express as px

# Sample housing sales data with country codes and sales count
housing_sales_data = [
    {"country_code": "USA", "sales_count": 150},
    {"country_code": "CAN", "sales_count": 80},
    # ... more data ...
]

# Create a DataFrame
df = pd.DataFrame(housing_sales_data)

# Load a world map shapefile (you can download one online)
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

# Merge housing sales data with the world map data
merged = world.set_index('iso_a3').join(df.set_index('country_code'))

# Plot the map using Matplotlib
fig, ax = plt.subplots(1, 2, figsize=(15, 6))

# Matplotlib Geospatial Visualization
```

```
merged.plot(column='sales_count', ax=ax[0], legend=True,
            legend_kwds={'label': "Housing Sales Count"},
            cmap='OrRd', edgecolor='black')
ax[0].set_title('Housing Sales across Different Countries')
ax[0].set_axis_off()

# Plotly Express Geospatial Visualization
geojson_file = 'path_to_geojson_file.json'
fig = px.choropleth(df,
                    geojson=geojson_file,
                    locations='country_code',
                    color='sales_count',
                    color_continuous_scale='OrRd',
                    range_color=(0, df['sales_count'].max()),
                    title='Housing Sales across Different Countries')
fig.update_geos(fitbounds="locations", visible=False)
fig.update_layout(margin={"r":0,"t":30,"l":0,"b":0})

plt.tight_layout()
plt.show()
```

```
<ipython-input-10-cc7ab5b0c599>:17: FutureWarning:

The geopandas.dataset module is deprecated and will be removed in GeoPandas 1.0.
You can get the original 'naturalearth_lowres' data from
https://www.naturalearthdata.com/downloads/110m-cultural-vectors/.
```



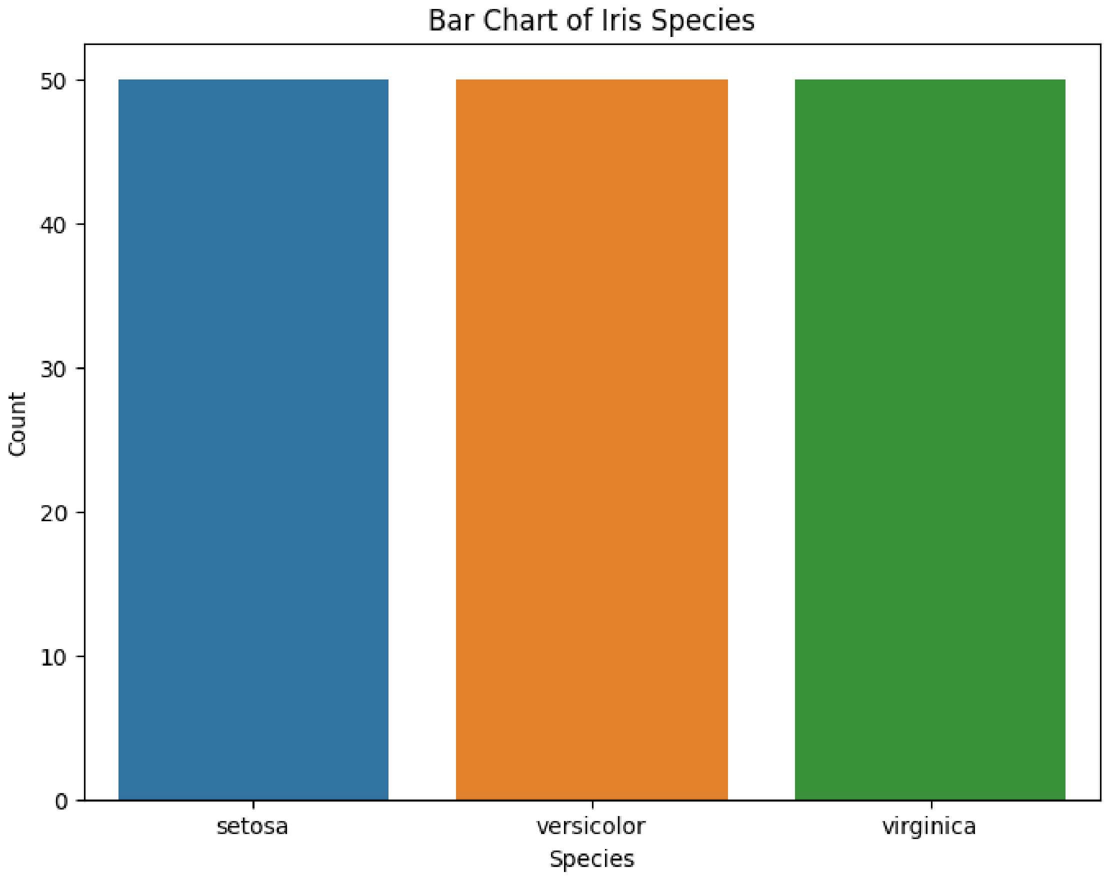
untitled10

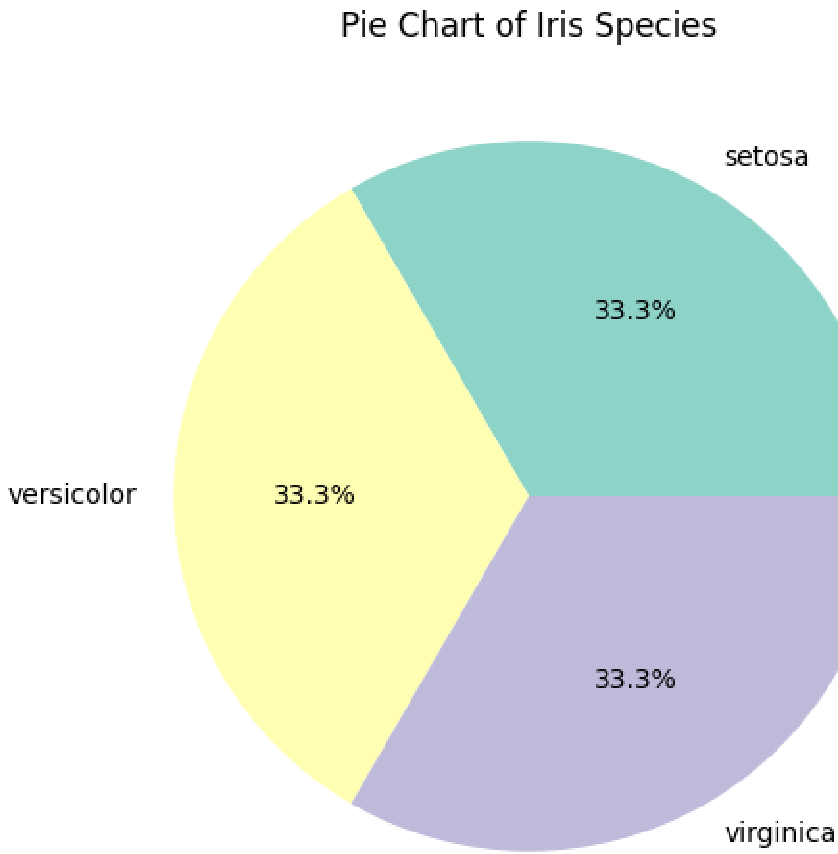
August 20, 2023

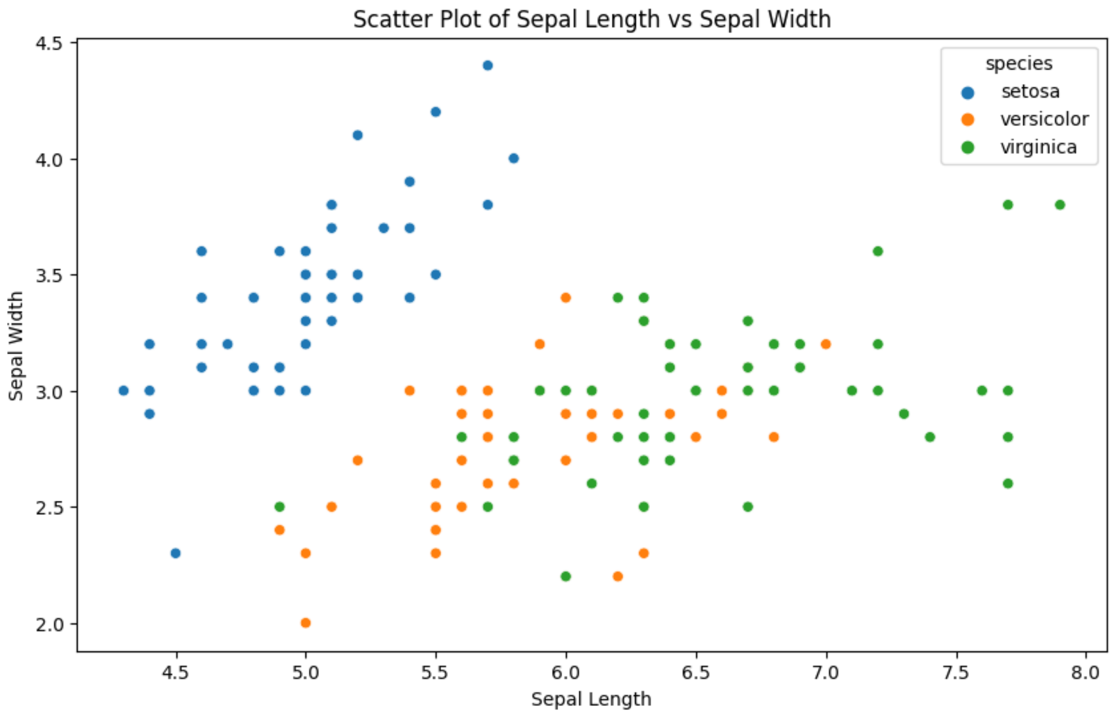
Task 10 A Consider a iris dataset.The iris data set contains 3 classes .where each class refers to a type of iris plant. The different variables involved in the data set are Sepal Length, Sepal Width, Petal Length, Petal width which is continuous and Variety which is a categorical variable 1. Import the necessary modules,and data set. 2. Design a Bar chart, Pie Chart using Univariate analysis of Categorical Data given in above data set. 3. Design a Scatterplot, Line Plot, Strip Plot, Swarm Plot using Univariate analysis of continuous data given in above data set.

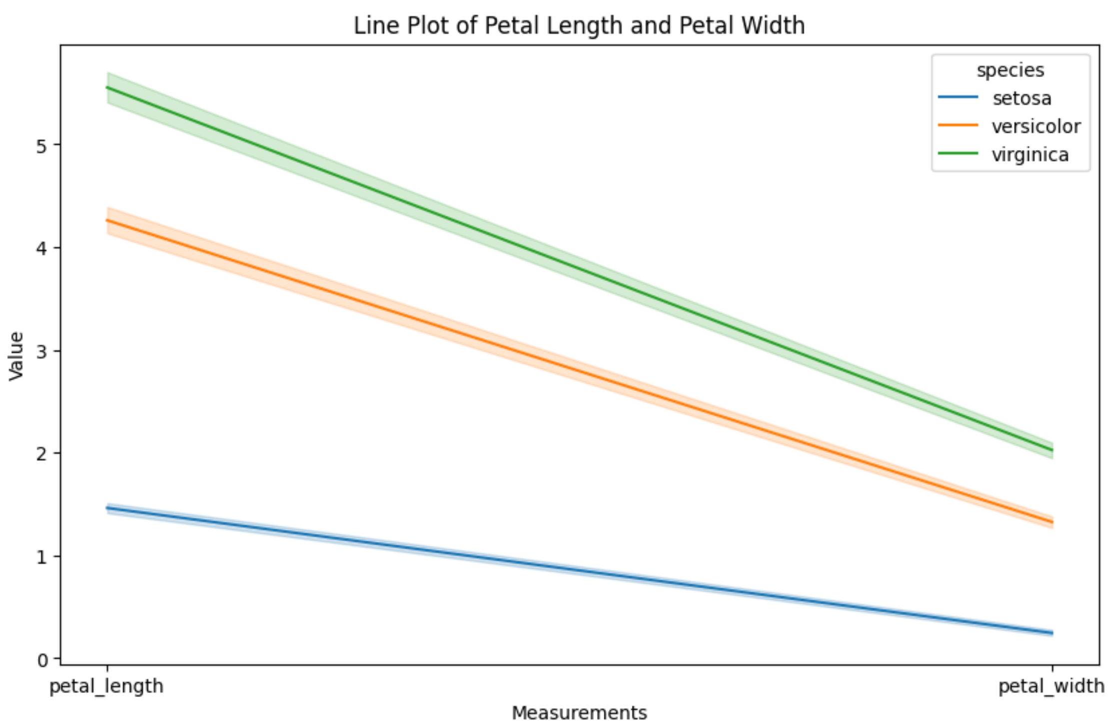
[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the iris dataset
iris = sns.load_dataset("iris")

# Univariate analysis of categorical data
# Bar chart
plt.figure(figsize=(8, 6))
sns.countplot(data=iris, x="species")
plt.title("Bar Chart of Iris Species")
plt.xlabel("Species")
plt.ylabel("Count")
plt.show()

# Pie chart
plt.figure(figsize=(8, 6))
iris["species"].value_counts().plot.pie(autopct="%1.1f%%", colors=sns.
 ↪color_palette("Set3"))
plt.title("Pie Chart of Iris Species")
plt.ylabel("")
plt.show()

# Univariate analysis of continuous data
# Scatter plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=iris, x="sepal_length", y="sepal_width", hue="species")
plt.title("Scatter Plot of Sepal Length vs Sepal Width")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
```

```
plt.show()

# Line plot (for continuous variables, we need to summarize data first)
plt.figure(figsize=(10, 6))
sns.lineplot(data=iris.melt(id_vars="species", value_vars=["petal_length",␣
↪"petal_width"]),
             x="variable", y="value", hue="species")
plt.title("Line Plot of Petal Length and Petal Width")
plt.xlabel("Measurements")
plt.ylabel("Value")
plt.show()

# Strip plot
plt.figure(figsize=(10, 6))
sns.stripplot(data=iris.melt(id_vars="species", value_vars=["sepal_length",␣
↪"sepal_width"]),
              x="variable", y="value", hue="species", dodge=True, jitter=True)
plt.title("Strip Plot of Sepal Length and Sepal Width")
plt.xlabel("Measurements")
plt.ylabel("Value")
plt.show()

# Swarm plot
plt.figure(figsize=(10, 6))
sns.swarmplot(data=iris.melt(id_vars="species", value_vars=["petal_length",␣
↪"petal_width"]),
              x="variable", y="value", hue="species", dodge=True)
plt.title("Swarm Plot of Petal Length and Petal Width")
plt.xlabel("Measurements")
plt.ylabel("Value")
plt.show()
```

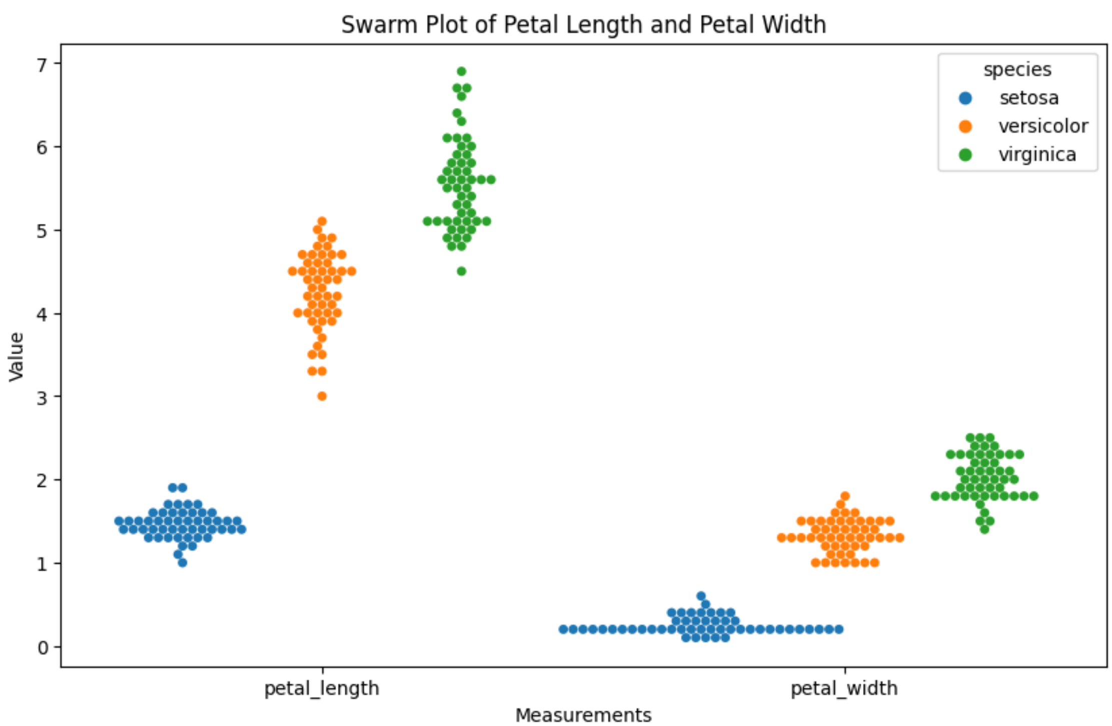

Bar Chart of Iris Species
Count
Species
setosa
versicolor
virginica
0
10
20
30
40
50


Pie Chart of Iris Species
setosa
33.3%
versicolor
33.3%
33.3%
virginica


Scatter Plot of Sepal Length vs Sepal Width
species
setosa
versicolor
virginica
Sepal Width
Sepal Length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Line Plot of Petal Length and Petal Width
species
setosa
versicolor
virginica
Value
Measurements
petal_length
petal_width
0
1
2
3
4
5


Strip Plot of Sepal Length and Sepal Width
species
setosa
versicolor
virginica
Value
8
7
6
5
4
3
2
sepal_length
sepal_width
Measurements


Swarm Plot of Petal Length and Petal Width
species
setosa
versicolor
virginica
Value
7
6
5
4
3
2
1
0
petal_length
petal_width
Measurements

TAsk 10 c read entire details of the dataframe.

[2]:
```
import pandas as pd

# Load the iris dataset
iris = pd.read_csv("https://raw.githubusercontent.com/uiuc-cse/data-fa14/
↪gh-pages/data/iris.csv")

# Display basic information about the dataframe
print("DataFrame Info:")
print(iris.info())

# Display statistical summary of the dataframe
print("\nDataFrame Summary Statistics:")
print(iris.describe(include="all"))
```

```
DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None

DataFrame Summary Statistics:
        sepal_length  sepal_width  petal_length  petal_width species
count     150.000000   150.000000    150.000000   150.000000     150
unique           NaN          NaN           NaN          NaN       3
top              NaN          NaN           NaN          NaN  setosa
freq             NaN          NaN           NaN          NaN      50
mean        5.843333     3.054000      3.758667     1.198667     NaN
std         0.828066     0.433594      1.764420     0.763161     NaN
min         4.300000     2.000000      1.000000     0.100000     NaN
25%         5.100000     2.800000      1.600000     0.300000     NaN
50%         5.800000     3.000000      4.350000     1.300000     NaN
75%         6.400000     3.300000      5.100000     1.800000     NaN
max         7.900000     4.400000      6.900000     2.500000     NaN
```

Task 10 D Display the data types of each column using the attribute dtype

[3]:

```
import pandas as pd

# Load the iris dataset
iris = pd.read_csv("https://raw.githubusercontent.com/uiuc-cse/data-fa14/
↪gh-pages/data/iris.csv")

# Display data types of each column
print("Data Types of Each Column:")
print(iris.dtypes)
```

```
Data Types of Each Column:
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```