RNS INSTITUTE OF TECHNOLOGY

Autonomous Institution Affiliated to VTU, Recognized by GOK, Approved by AICTE
(NAAC 'A+ Grade' Accredited, NBA Accredited (UG - CSE, ECE, ISE, EIE and EEE)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098 Ph:(080)28611880,28611881 URL:

DATA VISUALIZATION
ASSIGNMENT 2

Submitted By

Rachita J

1RN21CD038

Submitted To

Ms. Vinutha S

Assistant Professor

Dept. of CSE(Data Science)

Date: 09-12-2024

Ms. Vinutha S
Assistant Professor

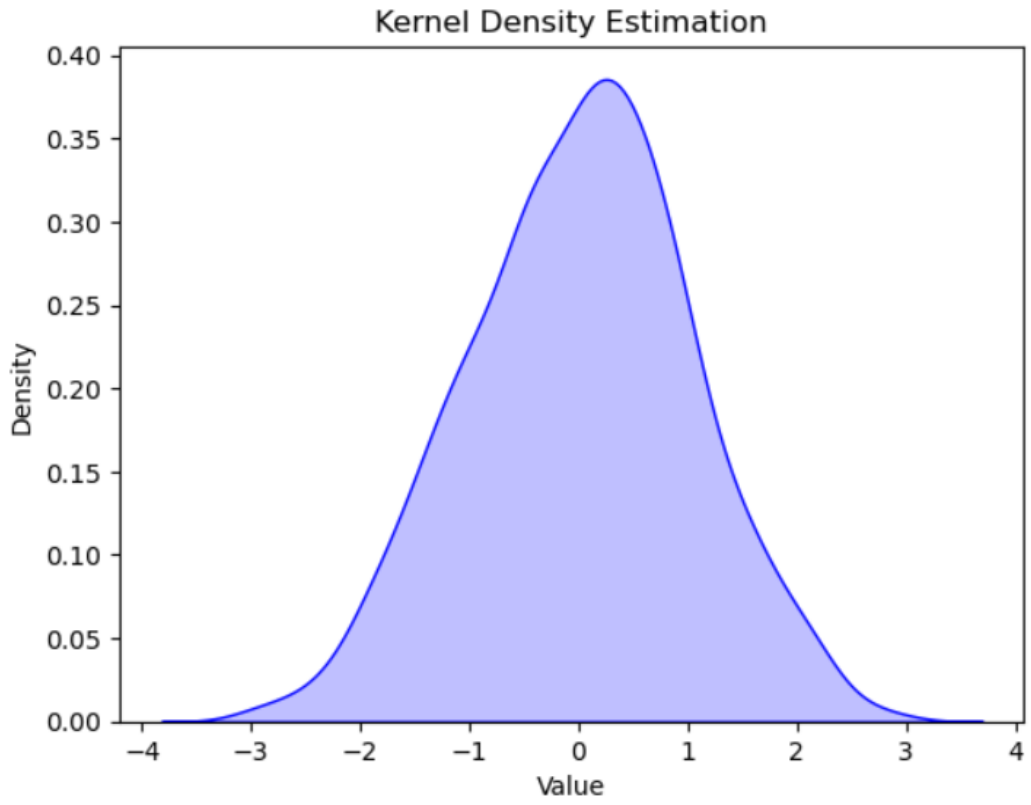
Question 1:

Develop a code to demonstrate Kernel Density Estimation

Solution:

```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

data = np.random.normal(loc=0, scale=1, size=1000)

sns.kdeplot(data, shade=True, color="blue")

plt.title("Kernel Density Estimation")
plt.xlabel("Value")
plt.ylabel("Density")
plt.show()
```

Output:



Question 2:

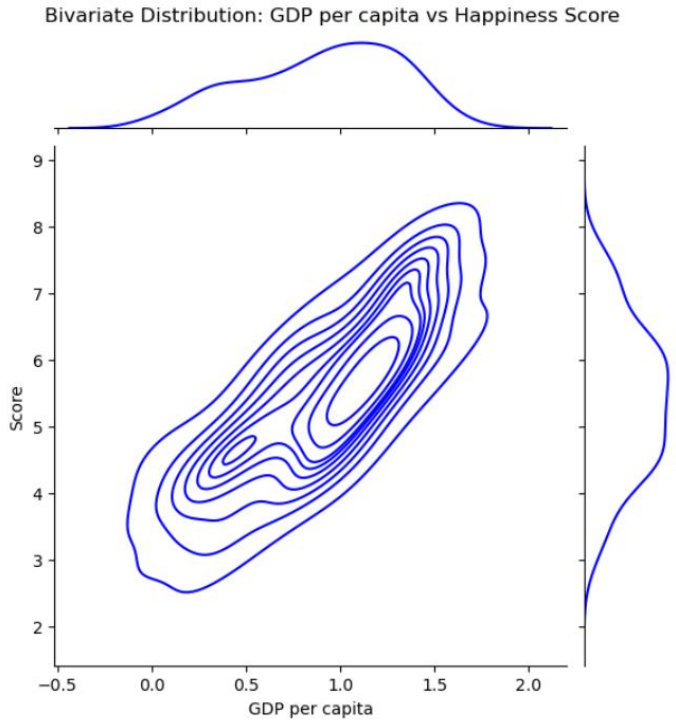
Develop a code to plot bivariate distribution considering a suitable data set available on the open source.

Solution:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

data = pd.read_csv("World_Happiness_Report_2019.csv")

print(data.columns)

# Plotting Bivariate Distribution for GDP per capita vs Happiness Score
sns.jointplot(
    x="GDP per capita",
    y="Score",
    data=data,
    kind="kde",
    color="blue"
)

plt.suptitle("Bivariate Distribution: GDP per capita vs Happiness Score", y=1.02)
plt.show()
```

Output:

```
Index(['Overall rank', 'Country or region', 'Score', 'GDP per capita',
       'Social support', 'Healthy life expectancy',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption'],
      dtype='object')
```



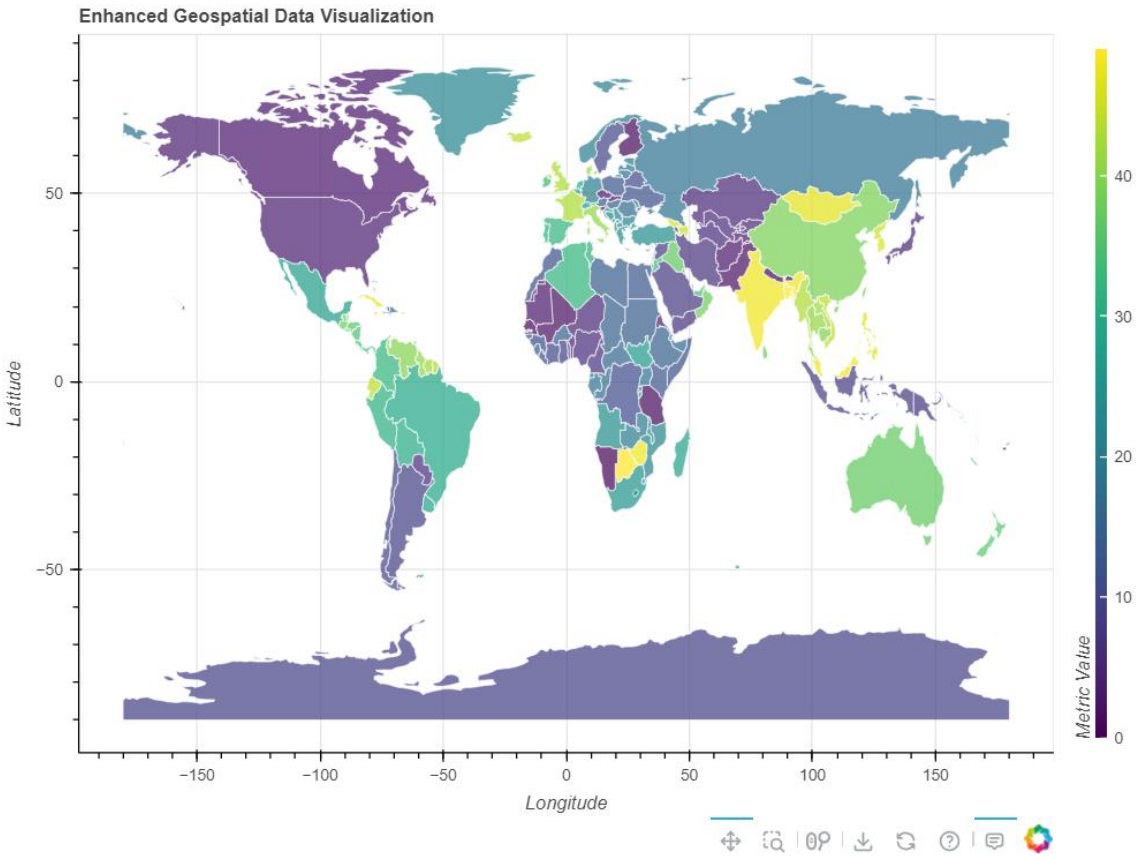
Question 3:

Develop a code to showcase various Geospatial data also make use of Bokeh to make it more interactive.

Solution:

```
from bokeh.io import output_file, show
from bokeh.plotting import figure
from bokeh.models import GeoJSONDataSource, HoverTool, ColorBar
from bokeh.palettes import Viridis256
from bokeh.transform import linear_cmap
import json
import geopandas as gpd

output_file("enhanced_geospatial_data.html")
world = gpd.read_file("ne_110m_admin_0_countries.shp")
world['dummy_metric'] = world.index % 50

# Converting GeoPandas DataFrame to GeoJSON format
geojson_data = json.loads(world.to_json())
geojson_source = GeoJSONDataSource(geojson=json.dumps(geojson_data))

# Color mapper
color_mapper = linear_cmap(
    field_name="dummy_metric",
    palette=Viridis256,
    low=min(world["dummy_metric"]),
    high=max(world["dummy_metric"])
)

# Along with color mapping ,also adding patches
p.patches(
    "xs",
    "ys",
    source=geojson_source,
    fill_color=color_mapper,
    fill_alpha=0.7,
    line_color="white",
    line_width=0.5
)

# Adding interactivity using HoverTool
hover = HoverTool(
    tooltips=[
        ("Country", "@name"),
        ("Metric Value", "@dummy_metric")
    ]
)
p.add_tools(hover)

# Adding color bar to the metrics
color_bar = ColorBar(
    color_mapper=color_mapper['transform'],
    width=8,
    location=(0, 0),
    title="Metric Value"
)
p.add_layout(color_bar, 'right')

show(p)
```

Output:


Enhanced Geospatial Data Visualization
Latitude
50
0
−50
−150
−100
−50
0
50
100
150
Longitude
Metric Value
40
30
20
10
0

Question 4:

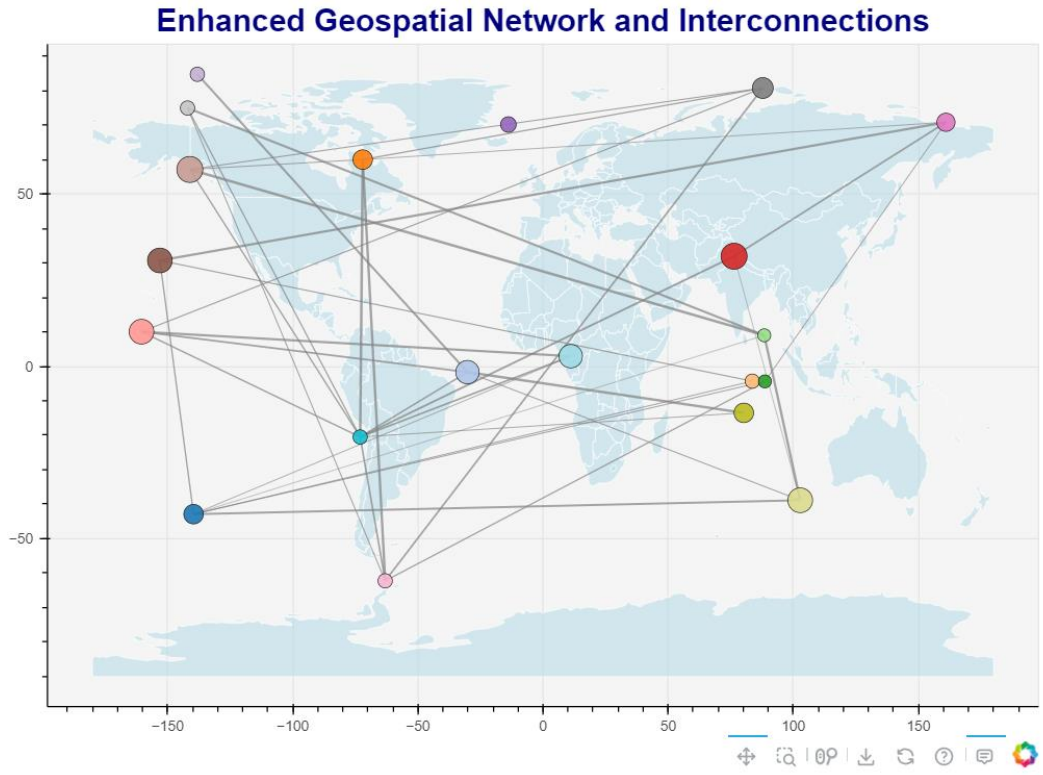
Develop a code to plot network and interconnection using geospatial data.

Solution:

```
import random
from bokeh.io import output_file, show
from bokeh.plotting import figure
from bokeh.models import GeoJSONDataSource, HoverTool, ColumnDataSource
from bokeh.palettes import Category20
import geopandas as gpd
import json
import networkx as nx
output_file("enhanced_geospatial_network.html")
world = gpd.read_file("ne_110m_admin_0_countries.shp")
world['dummy_name'] = [f"Country {i}" for i in range(len(world))]
# Creating a random network graph
countries = world['dummy_name'][:20]
G = nx.Graph()
for country in countries:
    G.add_node(country, x=random.uniform(-180, 180), y=random.uniform(-90, 90))
# Adding random edges with weights
for _ in range(40):
    node1, node2 = random.sample(list(G.nodes), 2)
    weight = random.uniform(0.5, 2.0)
    G.add_edge(node1, node2, weight=weight)
# Extracting the node positions
node_positions = {node: (G.nodes[node]['x'], G.nodes[node]['y']) for node in G.nodes}
edges = list(G.edges)
# Edge coordinates for Bokeh
edge_xs = []
edge_ys = []
edge_weights = []
for edge in edges:
    x0, y0 = node_positions[edge[0]]
    x1, y1 = node_positions[edge[1]]
    edge_xs.append([x0, x1])
    edge_ys.append([y0, y1])
    edge_weights.append(G.edges[edge]["weight"])
# Bokeh plot
p = figure(
    title="Enhanced Geospatial Network and Interconnections",
    width=800,
    height=600,
    background_fill_color="#f5f5f5",
    toolbar_location="below"
)
p.title.text_color = "navy"
p.title.text_font_size = "18pt"
p.title.align = "center"
```

```
# Map
geojson_data = json.loads(world.to_json())
geojson_source = GeoJSONDataSource(geojson=json.dumps(geojson_data))
p.patches(
    "xs",
    "ys",
    source=geojson_source,
    fill_alpha=0.5,
    fill_color="lightblue",
    line_color="white",
    line_width=0.5
)
# Adding edges with variable thickness
p.multi_line(
    xs=edge_xs,
    ys=edge_ys,
    line_color="gray",
    line_width=edge_weights,
    alpha=0.7
)
# Adding colorful nodes
node_colors = [Category20[20][i % 20] for i in range(len(node_positions))]
node_sizes = [random.randint(10, 20) for _ in range(len(node_positions))]
node_source = ColumnDataSource(data=dict(
    x=[pos[0] for pos in node_positions.values()],
    y=[pos[1] for pos in node_positions.values()],
    name=list(node_positions.keys()),
    color=node_colors,
    size=node_sizes
))
p.circle(
    "x",
    "y",
    source=node_source,
    size="size",
    fill_color="color",
    line_color="black",
    line_width=0.5,
    alpha=0.9
)
hover = HoverTool(tooltips=[("Country", "@name")])
p.add_tools(hover)
show(p)
```

Output:



Question 5:

Develop a code showcase networked program including retrieving image over HTTP, parsing HTML and scraping the web.

Solution:

```
import os
import requests
from bs4 import BeautifulSoup
from PIL import Image
from io import BytesIO

def download_image(image_url, folder_path):
    try:
        # Retrieving images over HTTP
        response = requests.get(image_url)
        response.raise_for_status()
        image_name = image_url.split("/")[-1]
        image_path = os.path.join(folder_path, image_name)
        image = Image.open(BytesIO(response.content))
        image.save(image_path)
        print(f"Image saved at {image_path}")
    except requests.exceptions.RequestException as e:
        print(f"Error downloading image: {e}")
    except Exception as e:
        print(f"Error processing image: {e}")

def scrape_images(url, folder_path):
    try:
        response = requests.get(url)
        response.raise_for_status()
        # Parsing HTML
        soup = BeautifulSoup(response.content, "html.parser")
        img_tags = soup.find_all("img")
        img_urls = [img['src'] for img in img_tags if 'src' in img.attrs]
        if not os.path.exists(folder_path):
            os.makedirs(folder_path)
        for img_url in img_urls:
            # Web Scraping
            if not img_url.startswith("http"):
                img_url = f"http://books.toscrape.com/{img_url}"
            download_image(img_url, folder_path)
    except requests.exceptions.RequestException as e:
        print(f"Error fetching webpage: {e}")
    except Exception as e:
        print(f"Error parsing webpage: {e}")

url = "http://books.toscrape.com/catalogue/page-1.html"
folder_path = "scraped_books_images"

scrape_images(url, folder_path)
```

Output:

```
Image saved at scraped_books_images\2cdad67c44b002e7ead0cc35693c0e8b.jpg
Image saved at scraped_books_images\260c6ae16bce31c8f8c95daddd9f4a1c.jpg
Image saved at scraped_books_images\3eef99c9d9adef34639f510662022830.jpg
Image saved at scraped_books_images\3251cf3a3412f53f339e42cac2134093.jpg
Image saved at scraped_books_images\bea5697f2534a2f86a3ef27b5a8c12a6.jpg
Image saved at scraped_books_images\68339b4c9bc034267e1da611ab3b34f8.jpg
Image saved at scraped_books_images\92274a95b7c251fea59a2b8a78275ab4.jpg
Image saved at scraped_books_images\3d54940e57e662c4dd1f3ff00c78cc64.jpg
Image saved at scraped_books_images\66883b91f6804b2323c8369331cb7dd1.jpg
Image saved at scraped_books_images\5846057e28022268153beff6d352b06c.jpg
Image saved at scraped_books_images\bef44da28c98f905a3ebec0b87be8530.jpg
Image saved at scraped_books_images\1048f63d3b5061cd2f424d20b3f9b666.jpg
Image saved at scraped_books_images\5b88c52633f53cacf162c15f4f823153.jpg
Image saved at scraped_books_images\94b1b8b244bce9677c2f29ccc890d4d2.jpg
Image saved at scraped_books_images\81c4a973364e17d01f217e1188253d5e.jpg
Image saved at scraped_books_images\54607fe8945897cdcced0044103b10b6.jpg
Image saved at scraped_books_images\553310a7162dfbc2c6d19a84da0df9e1.jpg
Image saved at scraped_books_images\09a3aef48557576e1a85ba7efea8ecb7.jpg
Image saved at scraped_books_images\0bbcd0a6f4bcd81ccb1049a52736406e.jpg
Image saved at scraped_books_images\27a53d0bb95bdd88288eaf66c9230d7e.jpg
```

Question 6:

Develop a code to showcase the web services including eXtensible Markup Language.

Solution:

```
import xml.etree.ElementTree as ET

# Predefined XML string
mock_xml = """
<countries>
    <country>
        <name>India</name>
        <capital>New Delhi</capital>
        <population>1393409038</population>
        <region>Asia</region>
    </country>
    <country>
        <name>Germany</name>
        <capital>Berlin</capital>
        <population>83166711</population>
        <region>Europe</region>
    </country>
</countries>
"""

def parse_mock_xml(xml_data):
    try:
        root = ET.fromstring(xml_data)

        print("Country Information:")
        for country in root.findall("country"):
            name = country.find("name").text
            capital = country.find("capital").text
            population = country.find("population").text
            region = country.find("region").text

            print(f"Name: {name}")
            print(f"Capital: {capital}")
            print(f"Population: {population}")
            print(f"Region: {region}")
            print("-" * 40)
    except ET.ParseError as e:
        print(f"Error parsing XML: {e}")
    except Exception as e:
        print(f"Unexpected error: {e}")

parse_mock_xml(mock_xml)
```

Output:

```
Country Information:
Name: India
Capital: New Delhi
Population: 1393409038
Region: Asia
----------------------------------------
Name: Germany
Capital: Berlin
Population: 83166711
Region: Europe
----------------------------------------
```

Github Repository:

https://github.com/thisisrach/Data-Visualisation-Assignment