# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

**Assignment 2**
**Report on**

## Data Visualization

**Submitted By**

**OMKAR MURTHY P - 1RN21CD034**

**Under the Guidance of**

**Ms Vinutha S**

**Asst. Professor**

**Department of CSE (Data Science)**

**RN SHETTY TRUST®**

# 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: www.rnsit.ac.in

**2024-2025**

# Table of Contents



**GitHub Link**: **https://github.com/omkar-252003/DATA-VIZUALIZATION.git**

# 1. Introduction

This document presents Python-based implementations for a variety of data analysis and programming tasks across different domains, such as statistical estimation, geospatial visualization, network analysis, and web services. Each code demonstrates the practical use of Python libraries to solve real-world problems effectively. Below is a brief summary of the purpose and functionality of each code:

1. **Kernel Density Estimation (KDE)**
   This code demonstrates the use of statistical methods to estimate the probability density function of a dataset using KDE. It leverages libraries like NumPy and Seaborn to visualize the smoothed distribution of data, making it useful for identifying trends and patterns in univariate data.
2. **Bivariate Distribution Plot**
   This example uses open-source data to explore and visualize relationships between two variables through bivariate distribution plots. Libraries like Seaborn are employed to create kernel density plots, highlighting areas of high and low density in the data.
3. **Interactive Geospatial Data Visualization with Bokeh**
   This script focuses on visualizing geospatial data interactively using the Bokeh library. It integrates map tile providers to overlay data points on geographic maps, making it an excellent tool for exploring spatial distributions and patterns.
4. **Network Plot Using Geospatial Data**
   This code visualizes a network of entities (nodes) and their connections (edges) with geospatial coordinates. Using NetworkX for graph modeling and Matplotlib for plotting, the code helps analyze spatial relationships and connectivity between nodes.
5. **Web Scraping and HTTP-Based Network Programming**
   The script demonstrates how to fetch web content, parse HTML, and extract useful data, such as images, using libraries like requests and BeautifulSoup. It showcases the foundational aspects of web scraping, a vital skill for data collection.
6. **Web Services with XML**
   This code highlights the interaction with web services providing data in XML format. It demonstrates how to retrieve and parse XML data using Python's built-in xml.etree.ElementTree library, illustrating the integration of structured data from external sources.

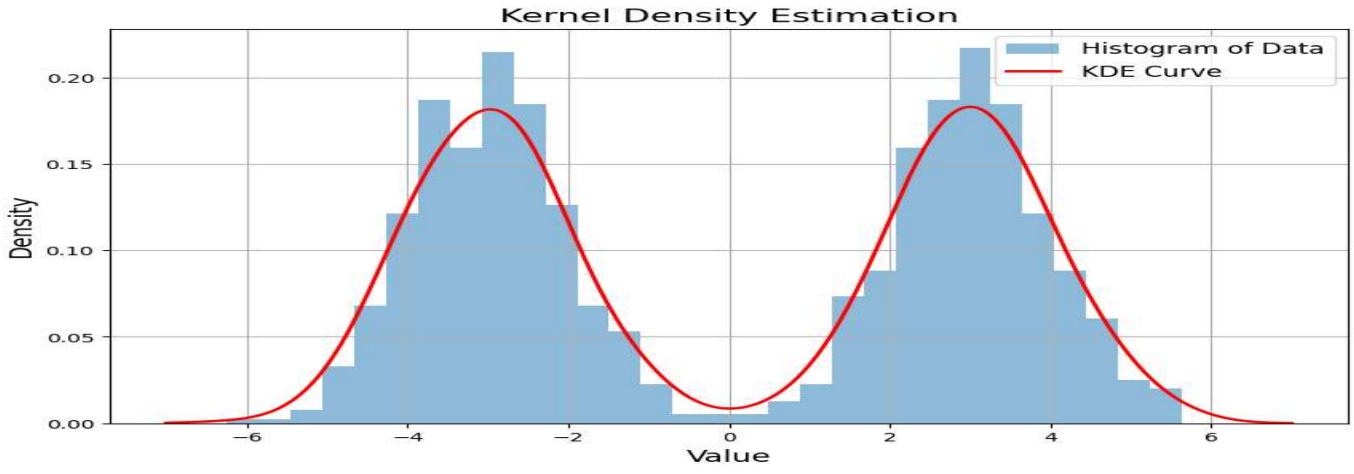
Each script is designed to address specific computational problems, showcasing Python's versatility in diverse application areas.

## 2. Question 1: Develop a code to demonstrate Kernel Density Estimation

**Code Snippet:**

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KernelDensity

np.random.seed(42)
data = np.concatenate([
    np.random.normal(loc=-3, scale=1, size=500),
    np.random.normal(loc=3, scale=1, size=500)
])

data = data[:, np.newaxis]

kde = KernelDensity(kernel='gaussian', bandwidth=0.5).fit(data)
x_plot = np.linspace(-7, 7, 1000)[:, np.newaxis]
log_density = kde.score_samples(x_plot)
density = np.exp(log_density)
plt.figure(figsize=(10, 6))
plt.hist(data, bins=30, density=True, alpha=0.5, label='Histogram of Data')
plt.plot(x_plot[:, 0], density, label='KDE Curve', color='red', linewidth=2)
plt.title('Kernel Density Estimation', fontsize=16)
plt.xlabel('Value', fontsize=14)
plt.ylabel('Density', fontsize=14)
plt.legend(fontsize=12)
plt.grid()
plt.show()
```

**Output:**

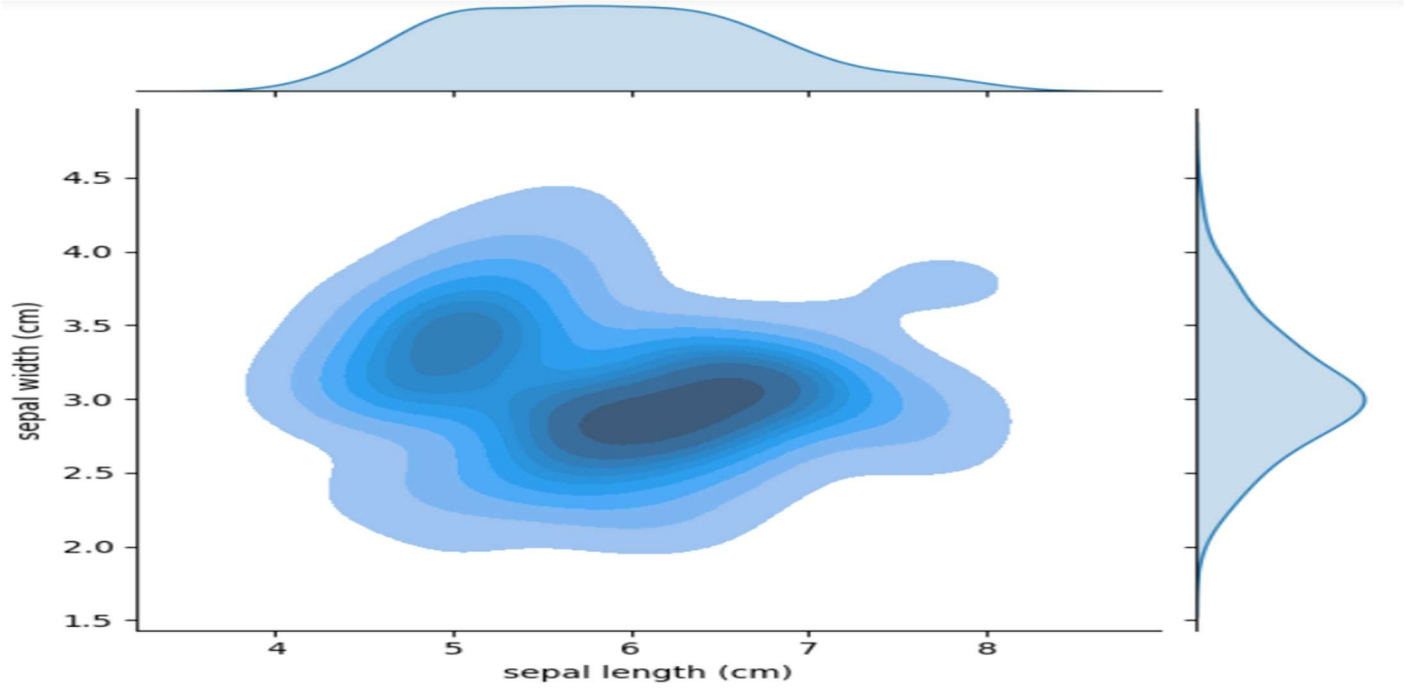


## 3. Question 2: Develop a code to plot bivariate distribution considering a suitable data set available on the open source

**Code Snippet:**

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import pandas as pd

# Load the Iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Plot bivariate distribution
sns.jointplot(data=df, x=df.columns[0], y=df.columns[1], kind="kde", fill=True)
plt.show()
```

**Output:**

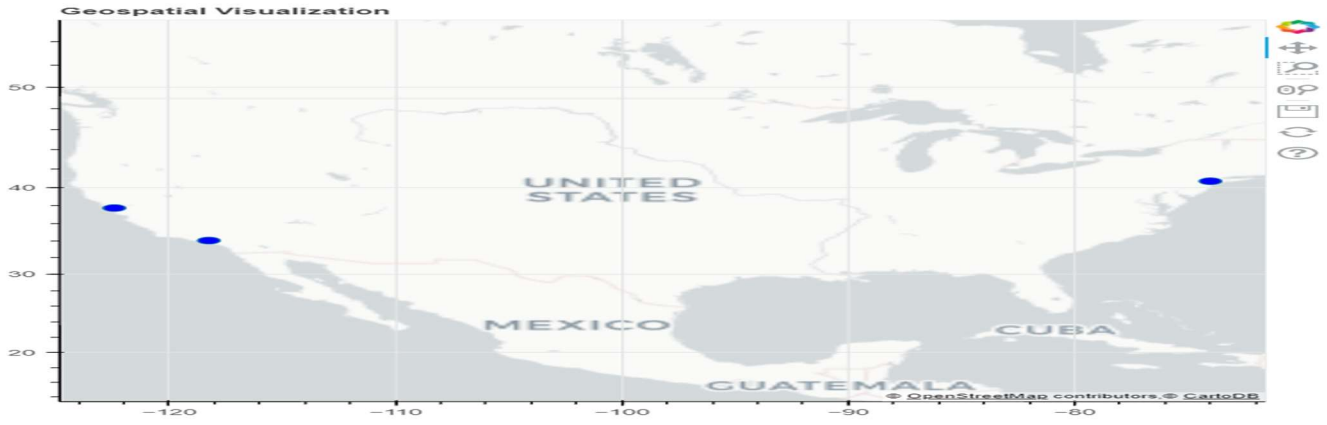


## 4. Question 3: Develop a code to showcase various Geospatial data also make use of Bokeh to make it more interactive

Code Snippet:

```
from bokeh.plotting import figure, show
from bokeh.tile_providers import get_provider, Vendors
from bokeh.models import ColumnDataSource
import pandas as pd

# Sample geospatial data
data = {'lat': [37.7749, 40.7128, 34.0522], 'lon': [-122.4194, -74.0060, -118.2437]}
df = pd.DataFrame(data)

# Convert latitude and longitude to web mercator coordinates
df['x'] = df['lon'] * (20037508.34 / 180)
df['y'] = pd.np.log(pd.np.tan((90 + df['lat']) * pd.np.pi / 360.0)) * (20037508.34 / pd.np.pi)

# Create Bokeh plot
tile_provider = get_provider(Vendors.CARTODBPOSITRON)
p = figure(x_axis_type="mercator", y_axis_type="mercator", title="Geospatial Visualization")
p.add_tile(tile_provider)

source = ColumnDataSource(df)
p.circle(x='x', y='y', size=10, fill_color="blue", source=source)

show(p)
```

**Output:**

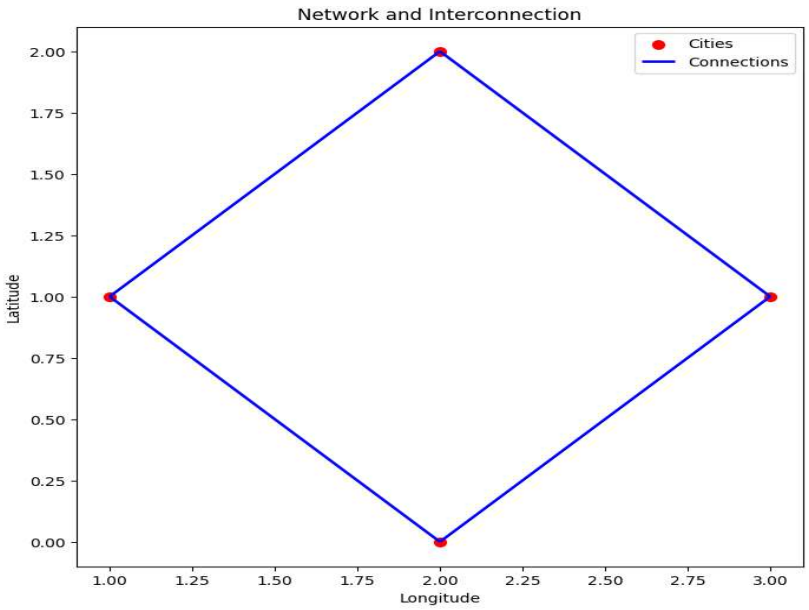


## 5. Question 4 : Develop a code to plot network and interconnection using geospatial data

**Code Snippet:**

```
import geopandas as gpd
from shapely.geometry import Point, LineString
import matplotlib.pyplot as plt

cities = gpd.GeoDataFrame({
    'city': ['City A', 'City B', 'City C', 'City D'],
    'geometry': [Point(1, 1), Point(2, 2), Point(3, 1), Point(2, 0)],
})

connections = [
    ('City A', 'City B'),
    ('City B', 'City C'),
    ('City C', 'City D'),
    ('City D', 'City A'),
]
lines = []

for city1, city2 in connections:
    point1 = cities[cities['city'] == city1].geometry.iloc[0]
    point2 = cities[cities['city'] == city2].geometry.iloc[0]
    line = LineString([point1, point2])
    lines.append(line)
connections_gdf = gpd.GeoDataFrame({'geometry': lines})

fig, ax = plt.subplots(figsize=(8, 8))
cities.plot(ax=ax, marker='o', color='red', markersize=50, label='Cities')
connections_gdf.plot(ax=ax, color='blue', linewidth=2, label='Connections')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('Network and Interconnection')
ax.legend()
plt.show()
```

**Output:**




## 6. Question 5: Develop a code showcase networked program including retrieving image over HTTP, parsing HTML and scraping the web

## Code Snippet:

```
# !pip install requests beautifulsoup4
import requests
from bs4 import BeautifulSoup
import os
import shutil
url = "https://en.wikipedia.org/wiki/Main_Page"  # Wikipedia main page
response = requests.get(url)
if response.status_code == 200:
    print("Website fetched successfully.")
else:
    print(f"Failed to fetch the website. Status code: {response.status_code}")
    exit()
soup = BeautifulSoup(response.content, 'html.parser')
image_tag = soup.find('img')
if image_tag and 'src' in image_tag.attrs:
    image_url = image_tag['src']
    if not image_url.startswith('http'):  # Handle relative URLs
        image_url = requests.compat.urljoin(url, image_url)
    print(f"Image URL: {image_url}")
else:
```

```
print("No image found on the page.")
        exit()
    image_response = requests.get(image_url, stream=True)
    if image_response.status_code == 200:
        print("Image retrieved successfully.")
        # Step 6: Save the image locally
        image_filename = os.path.basename(image_url)
        with open(image_filename, 'wb') as f:
            shutil.copyfileobj(image_response.raw, f)
        print(f"Image saved as {image_filename}.")
    else:
        print(f"Failed to fetch the image. Status code: {image_response.status_code}")

    print("\nExtracting all hyperlinks from the page:")
    for link in soup.find_all('a', href=True):
        href = link['href']
        full_url = requests.compat.urljoin(url, href)  # Handle relative URLs
        print(full_url)
```

**Output:**

```
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.32.3)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (4.12.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2024.8.30)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4) (2.6)
Website fetched successfully.
Image URL: https://en.wikipedia.org/static/images/icons/wikipedia.png
Image retrieved successfully.
Image saved as wikipedia.png.

Extracting all hyperlinks from the page:
https://en.wikipedia.org/wiki/Main_Page#bodyContent
https://en.wikipedia.org/wiki/Main_Page
https://en.wikipedia.org/wiki/Wikipedia:Contents
https://en.wikipedia.org/wiki/Portal:Current_events
https://en.wikipedia.org/wiki/Special:Random
https://en.wikipedia.org/wiki/Wikipedia:About
https://en.wikipedia.org/wiki/Wikipedia:Contact_us
https://en.wikipedia.org/wiki/Help:Contents
https://en.wikipedia.org/wiki/Help:Introduction
https://en.wikipedia.org/wiki/Wikipedia:Community_portal
https://en.wikipedia.org/wiki/Special:RecentChanges
https://en.wikipedia.org/wiki/Wikipedia:File_upload_wizard
https://en.wikipedia.org/wiki/Main_Page
https://en.wikipedia.org/wiki/Special:Search
https://donate.wikimedia.org/?wmf_source=donate&wmf_medium=sidebar&wmf_campaign=en.wikipedia.org&uselang=en
https://en.wikipedia.org/w/index.php?title=Special:CreateAccount&returnto=Main+Page
https://en.wikipedia.org/w/index.php?title=Special:UserLogin&returnto=Main+Page
https://donate.wikimedia.org/?wmf_source=donate&wmf_medium=sidebar&wmf_campaign=en.wikipedia.org&uselang=en
https://en.wikipedia.org/w/index.php?title=Special:CreateAccount&returnto=Main+Page
https://en.wikipedia.org/w/index.php?title=Special:UserLogin&returnto=Main+Page
https://en.wikipedia.org/wiki/Help:Introduction
https://en.wikipedia.org/wiki/Special:MyContributions
https://en.wikipedia.org/wiki/Special:MyTalk
https://en.wikipedia.org/wiki/Main_Page
https://en.wikipedia.org/wiki/Talk:Main_Page
```

## 7. Question 6: Develop a code to showcase the web services including eXtensible Markup Language

**Code Snippet:**

```
import requests
import xml.etree.ElementTree as ET

# Sample XML-based web service
url = "https://www.w3schools.com/xml/simple.xml"
response = requests.get(url)

# Parse the XML response
root = ET.fromstring(response.content)

# Print XML data
for food in root.findall('food'):
    name = food.find('name').text
    price = food.find('price').text
    print(f"{name}: {price}")
```

**Output:**

```
Belgian Waffles: $5.95
Strawberry Belgian Waffles: $7.95
Berry-Berry Belgian Waffles: $8.95
French Toast: $4.50
Homestyle Breakfast: $6.95
```

## 8. Conclusion

This report demonstrates various data analysis and visualization techniques using Python libraries such as Numpy, Pandas, Matplotlib, and Seaborn, Bokeh and web scraping  Each question addresses a specific aspect of data analysis and visualization, showcasing the capabilities of these libraries.

## 9. References


- Pandas Documentation
- Numpy Documentation
- Matplotlib Documentation
- Seaborn Documentation
- Bokeh Documentation


**GitHub Repo Link**: **https://github.com/omkar-252003/DATA-VIZUALIZATION.git**