AUTONOMOUS VEHICLES

A PROJECT REPORT (PHASE 3)

submitted by

HARIHARA SUDHAN. R

810022104088

In partial fulfilment of the requirements for

Bachelor of Engineering

in

COMPUTER SCIENCE AND ENGINEERING

Under the course of

ARTIFICIAL INTELLIGENCE

UNIVERSITY COLLEGE OF ENGINEERING,

BIT CAMPUS,

ANNA UNIVERSITY, TIRUCHIRAPPALLI – 620 024

Team Members:

- Arunkumar. V (810022104062)
- Salai sanamrgam. J (810022104072)
- Sathish. S (810022104075)
- Kirthick roshan (810022104116)

Introduction:

Phase 3 of our project marks a significant shift towards **data visualization,** a critical aspect of **data analysis** and **interpretation.** Through the implementation of effective visualization techniques, we aim to visually communicate **insights, trends,** and **patterns** present within the dataset. By doing so, we facilitate stakeholders in making **informed decisions** and understanding complex relationships more intuitively. This phase emphasizes the power of visualization in aiding **comprehension** and fostering actionable insights, ultimately contributing to the success of our project.

Objectives:

- Develop visually **informative** and appealing **visualizations** to explore and effectively communicate key insights derived from the dataset.
- Employ a diverse range of visualization techniques tailored to represent various types of data comprehensively, ensuring **clarity** and **accuracy** in representation.
- Enhance user engagement and understanding through the implementation of interactive visualizations, fostering a **dynamic** and **immersive exploration** of the dataset.
- Document the data visualization process meticulously, ensuring **transparency** and **reproducibility,** thereby facilitating effective collaboration and future reference.

Dataset description:

Dataset Paper Link: https://arxiv.org/abs/2401.10659

- **Dataset Overview:**
  - Covers 9 districts in Bangladesh: Sylhet, Dhaka, Rajshahi, Mymensingh, Maowa, Chittagong, Sirajganj, Sherpur, and Khulna
  - Contains 9825 images.
  - Annotations for 78,943 objects.
  - Includes 13 different classes of objects.
  - Annotated with rectangular bounding boxes.

- **Data Collection:**
  - Images were collected using smartphone cameras.
  - Simulates real-world conditions faced by autonomous vehicles.
  - Absence of online images ensures dataset authenticity.
  - Represents actual driving scenarios in Bangladesh, promoting practical model development.

Main Goal:

- ✓ Develop solutions for detecting objects under diverse road conditions in Bangladesh.

Classes (13 in total):

1. Auto Rickshaw
2. Bicycle
3. Bus
4. Car
5. Cart Vehicle
6. Construction Vehicle
7. Motorbike
8. Person
9. Priority Vehicle
10. Three-Wheeler
11. Train
12. Truck
13. Wheelchair

Dataset Format:

This dataset is in Yolov5 format. The train and test images can be found in in `dlenigma1/BadODD/images/`, and the labels for the train dataset can be found in `dlenigma1/BadODD/labels/`directory.

- Train Image Count: 5896
- Test Image Count: 1964

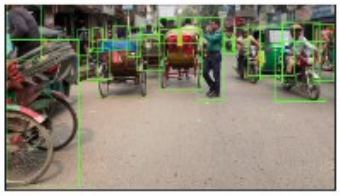
Each training image has a txt file of the same name in the labels directory.

Test
Test set images are in `dlenigma1/BadODD/images/test`.

The test set has no provided annotations. Your task is to learn the bounding box from the train set and predict it for the test set images.

Sample Database:

Sylhet



Dhaka

Rajshahi

Mymensingh

Maowa Expressway

Dhaka Night

Chittagong Night

Chittagong Bohoddarhat

Chuadanga Sirajganj

Sherpur

Khulna

Data Visualizations:

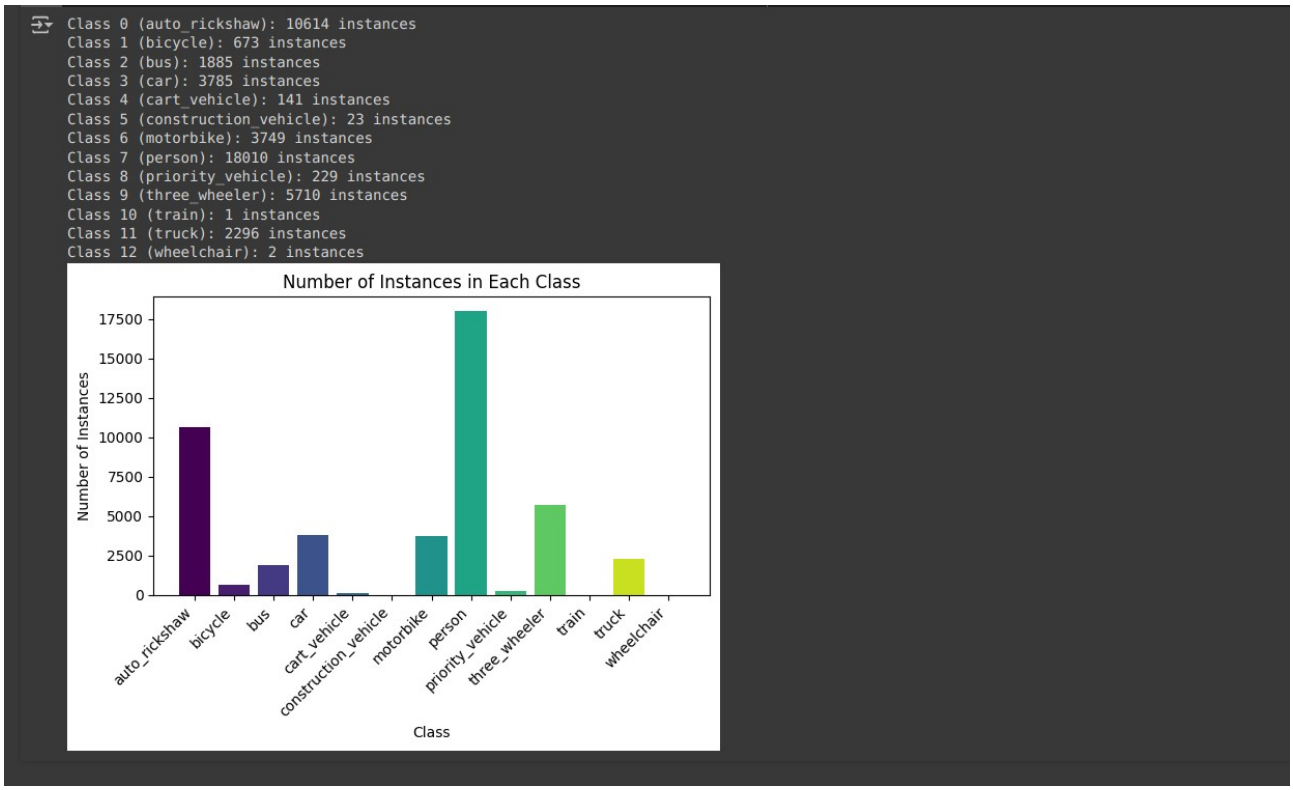
Data visualization is the art of representing **information** and **data** through visual elements like **charts, graphs,** and **maps.** It's a powerful tool to t**ransform raw data** into easily **digestible insights.**

Code snippet (Data Visualization):

```
import os

import matplotlib.pyplot as plt

import numpy as np

class_id_to_name = {
0: 'auto_rickshaw',
1: 'bicycle',
2: 'bus',
3: 'car',
4: 'cart_vehicle',
5: 'construction_vehicle',
6: 'motorbike',
7: 'person',
8: 'priority_vehicle',
9: 'three_wheeler',
10: 'train',
11: 'truck',
12: 'wheelchair'
}

# Define the label folder path
label_folder = "/content/dlenigma1/BadODD/labels/train"
# Initialize a dictionary to store class counts
class_counts = {class_id: 0 for class_id in class_id_to_name}

# Iterate through each text file in the label folder
for filename in os.listdir(label_folder):
if filename.endswith(".txt"):
file_path = os.path.join(label_folder, filename)

# Read the content of the label file and count class occurrences
with open(file_path, 'r') as file:
lines = file.readlines()

for line in lines:
parts = line.strip().split()
if parts: # Check if the line is not empty
class_id = int(parts[0])
if class_id in class_counts:
class_counts[class_id] += 1

# Print the class counts
for class_id, count in class_counts.items():
class_name = class_id_to_name[class_id]
print(f'Class {class_id} ({class_name}): {count} instances')

# Create a bar plot
class_ids = np.array(list(class_counts.keys()))
class_names = [class_id_to_name[class_id] for class_id in class_ids]
```

```
counts = np.array(list(class_counts.values()))
colors = plt.cm.viridis(np.linspace(0, 1, len(class_ids)))

plt.bar(class_names, counts, color=colors)
plt.xlabel('Class')
plt.ylabel('Number of Instances')
plt.title('Number of Instances in Each Class')
plt.xticks(rotation=45, ha="right") # Rotate x-axis labels for better visibility
plt.tight_layout()
plt.show()
```

Output:



➢ **Univariate Data Visualization:**

Univariate visualization is a way of using graphs and charts to understand the distribution of a **single variable** in your data set. This variable can be either categorical (like hair color or shoe size) or continuous (like weight or height).

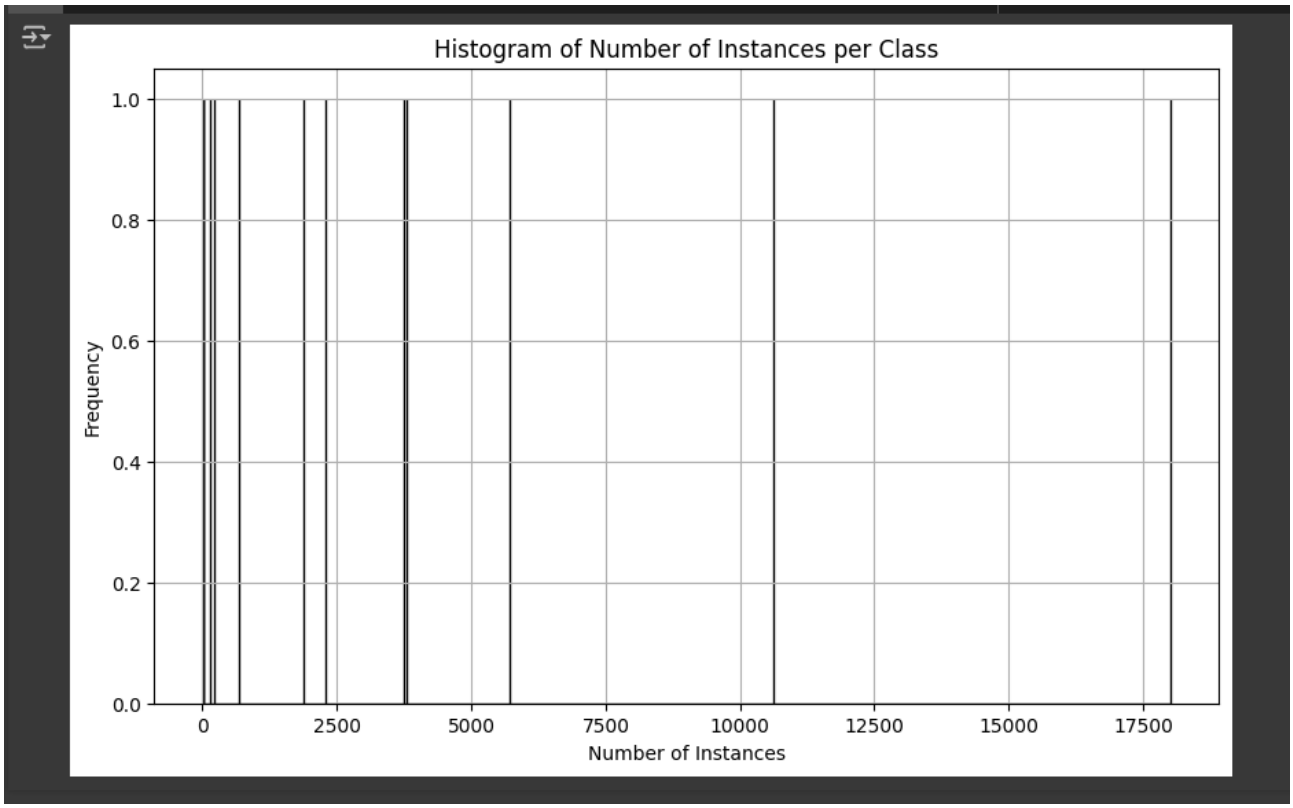
Code snippet for Univariate Data Visualization:

Histogram:

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

class_id_to_name = {
0: 'auto_rickshaw',
1: 'bicycle',
```

```
2: 'bus',
3: 'car',
4: 'cart_vehicle',
5: 'construction_vehicle',
6: 'motorbike',
7: 'person',
8: 'priority_vehicle',
9: 'three_wheeler',
10: 'train',
11: 'truck',
12: 'wheelchair'
}

# Define the label folder path
label_folder = "/content/dlenigma1/BadODD/labels/train"

# Initialize a dictionary to store class counts
class_counts = {class_id: 0 for class_id in class_id_to_name}

# Iterate through each text file in the label folder
for filename in os.listdir(label_folder):
if filename.endswith(".txt"):
file_path = os.path.join(label_folder, filename)

# Read the content of the label file and count class occurrences
with open(file_path, 'r') as file:
lines = file.readlines()

for line in lines:
parts = line.strip().split()
if parts: # Check if the line is not empty
class_id = int(parts[0])
if class_id in class_counts:
class_counts[class_id] += 1

# Univariate analysis
class_ids = np.array(list(class_counts.keys()))
counts = np.array(list(class_counts.values()))

# Plotting histogram
plt.figure(figsize=(10, 6))
plt.hist(counts, bins=range(min(counts), max(counts) + 1), edgecolor='black')
plt.xlabel('Number of Instances')
plt.ylabel('Frequency')
plt.title('Histogram of Number of Instances per Class')
plt.grid(True)
plt.show()
```

Output:

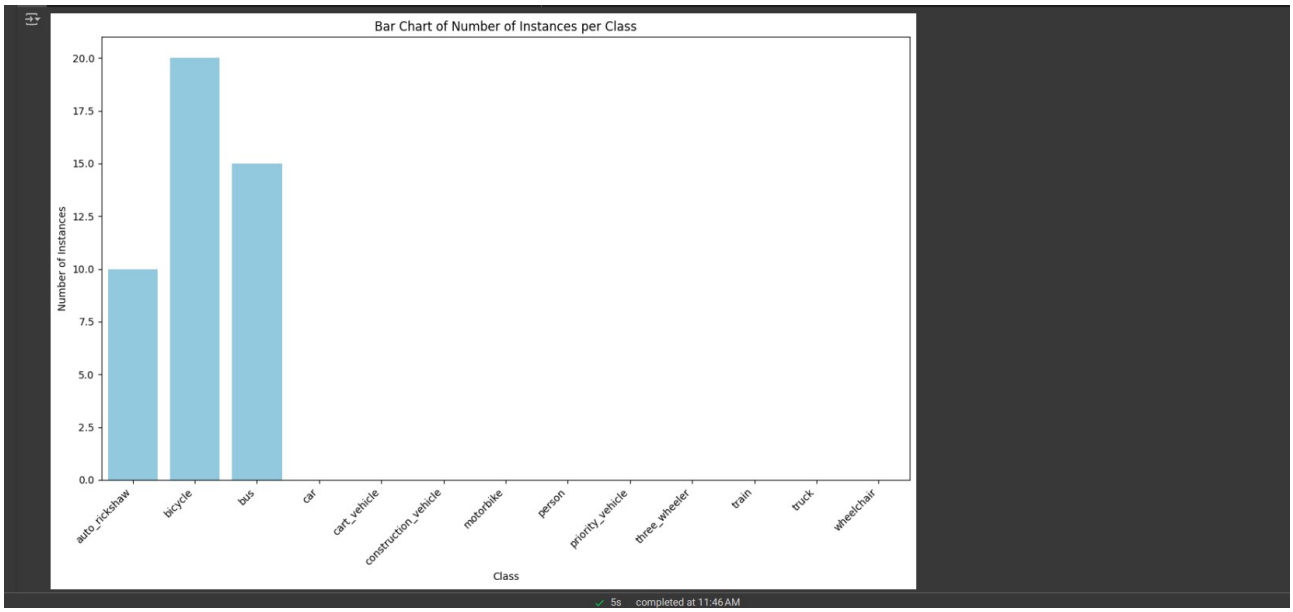


Barchart using Seaborn for Univariate visualizations:

code:

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

# Define class names and counts
class_names = list(class_id_to_name.values())
counts = [class_counts.get(class_id, 0) for class_id in class_id_to_name.keys()]

# Create a bar plot using Seaborn
plt.figure(figsize=(12, 8))
sns.barplot(x=class_names, y=counts, color='skyblue')
plt.xlabel('Class')
plt.ylabel('Number of Instances')
plt.title('Bar Chart of Number of Instances per Class')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Output:



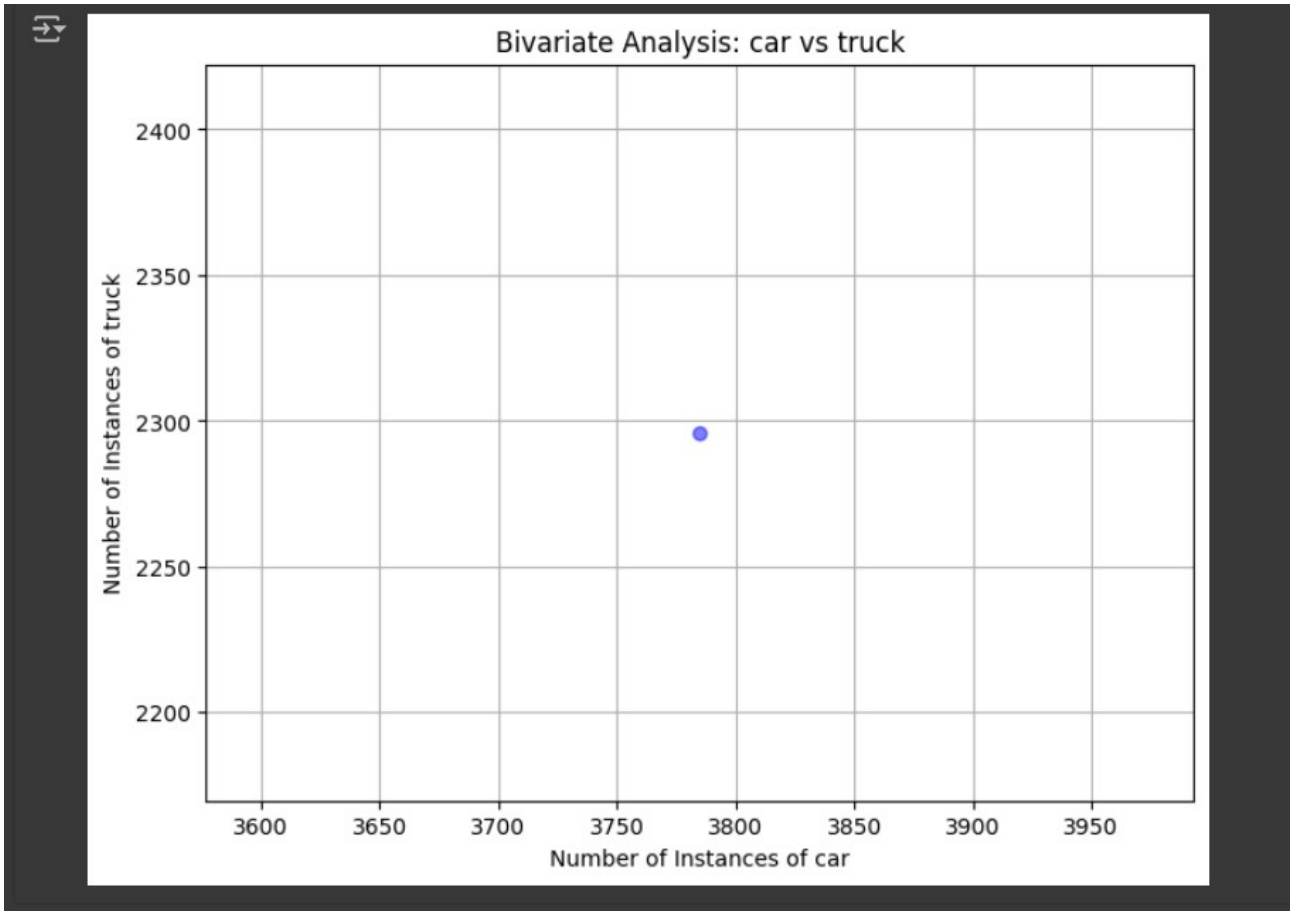
- **Bivariate Data Visualization:**
  - Bivariate visualization is a type of data visualization specifically designed to explore the relationship between **two variables** in your data set. These variables can be either **categorical** (like occupation and income level) or **continuous** (like hours studied and exam score).

Code snippet for Bivariate Data Visualization:

Histogram:

```
import matplotlib.pyplot as plt

# Define the classes you want to compare
class_id_1 = 3 # car
class_id_2 = 11 # truck

# Get the counts for the two classes
count_class_1 = class_counts[class_id_1]
count_class_2 = class_counts[class_id_2]

# Create a scatter plot
plt.figure(figsize=(8, 6))
plt.scatter(count_class_1, count_class_2, color='blue', alpha=0.5)
plt.title(f'Bivariate Analysis: {class_id_to_name[class_id_1]} vs {class_id_to_name[class_id_2]}')
plt.xlabel(f'Number of Instances of {class_id_to_name[class_id_1]}')
plt.ylabel(f'Number of Instances of {class_id_to_name[class_id_2]}')
plt.grid(True)
plt.show()
```

Output:



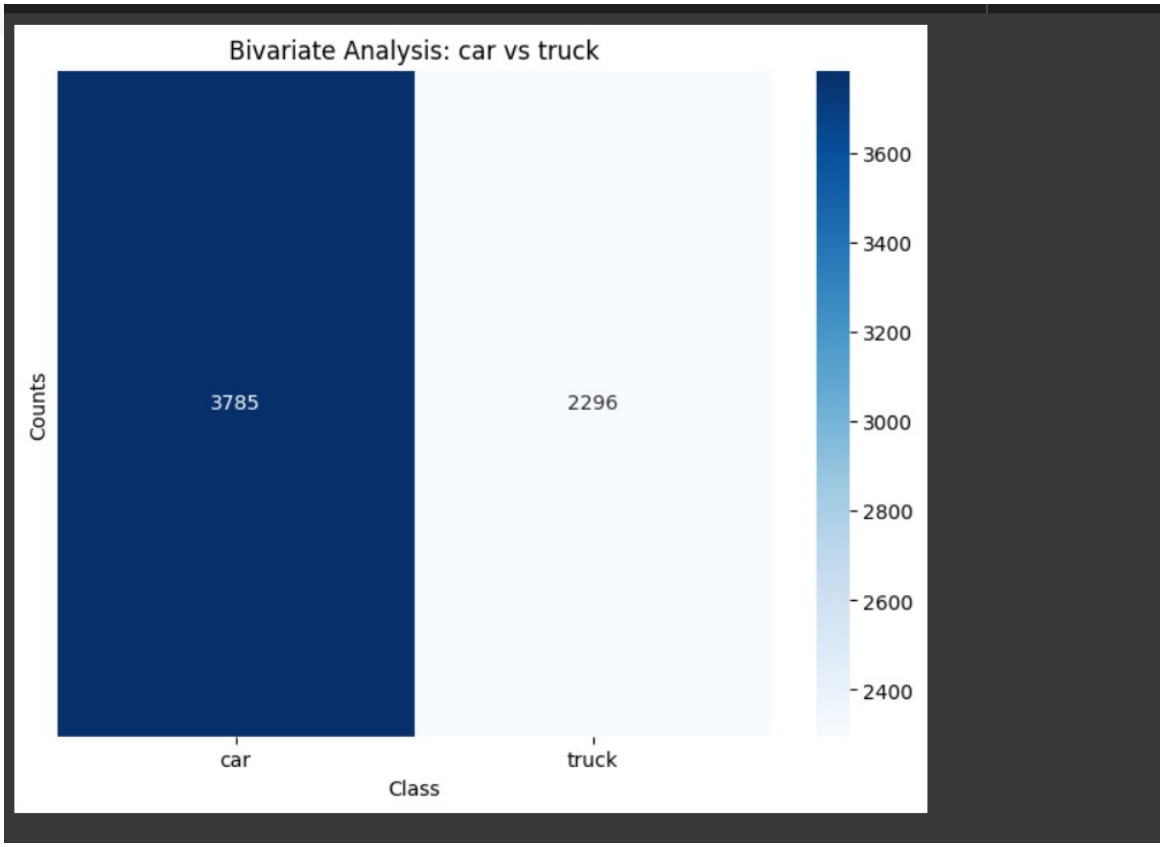
Code for Bivariate Data Visualization:

Heat map_visualization:

```
import seaborn as sns

# Define the classes you want to compare
class_id_1 = 3 # car
class_id_2 = 11 # truck

# Get the counts for the two classes
count_class_1 = class_counts[class_id_1]
count_class_2 = class_counts[class_id_2]

# Create a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap([[count_class_1, count_class_2]], annot=True, fmt='d', cmap='Blues',
xticklabels=[class_id_to_name[class_id_1], class_id_to_name[class_id_2]],
yticklabels=False)
plt.title(f'Bivariate Analysis: {class_id_to_name[class_id_1]} vs {class_id_to_name[class_id_2]}')
plt.xlabel('Class')
```

```
plt.ylabel('Counts')
plt.show()
```

Output:



- **Multivariate Data Visualization:**
  - Multivariate variate data visualization involves visualizing **more than one data value in a single renderer.** This is done for many reasons, including to: View the relationship between two or more variables. Compare or contrast the difference between **two variables**.

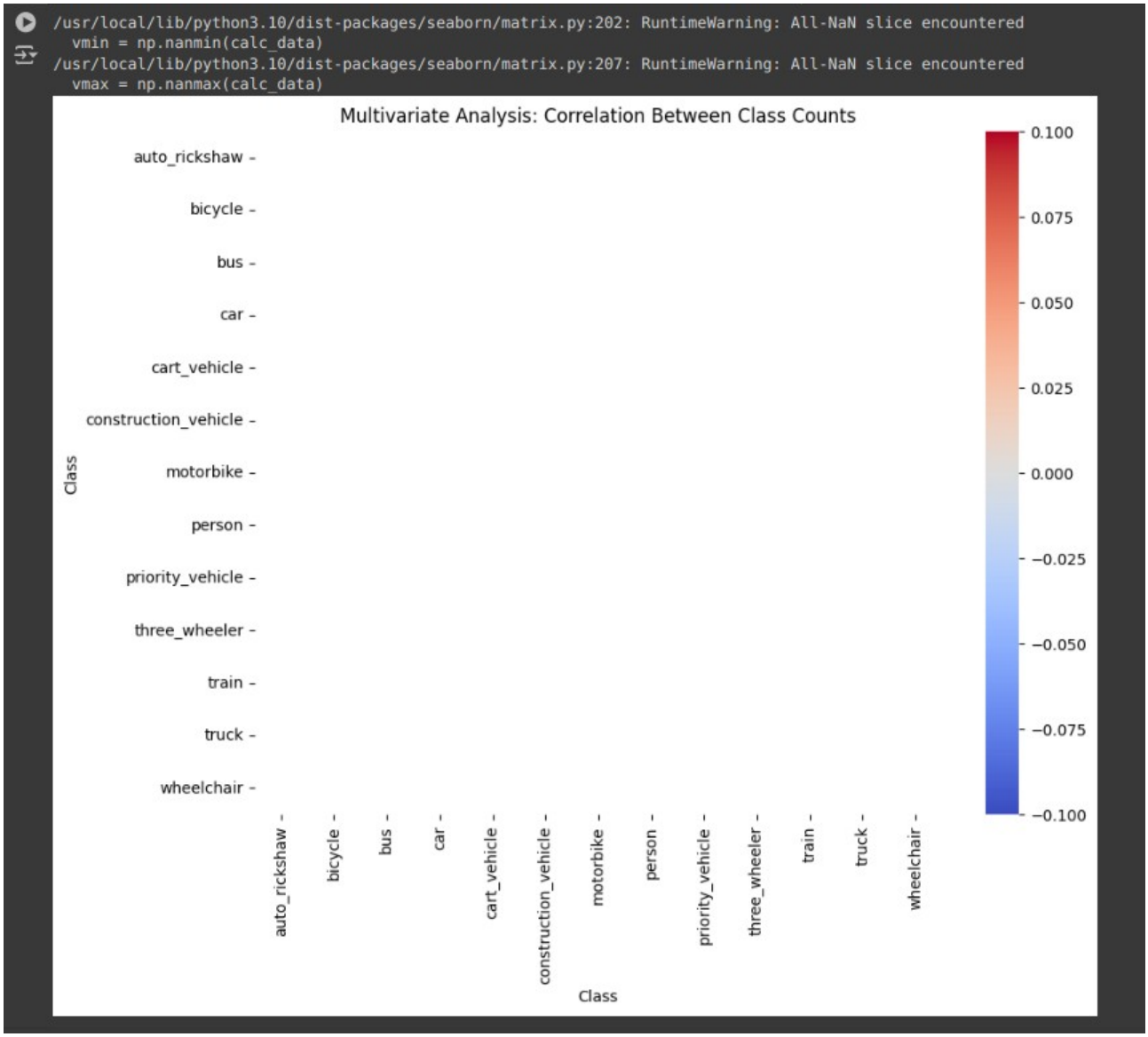
Code snippet for Multivariate Data visualization:

Code for Heat_map_visualization:

```
import pandas as pd
import seaborn as sns

# Create a DataFrame with class counts
counts_df = pd.DataFrame([class_counts.values()], columns=class_id_to_name.values())

# Calculate the correlation matrix
correlation_matrix = counts_df.corr()

# Create a heatmap
```

```
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
xticklabels=correlation_matrix.columns, yticklabels=correlation_matrix.columns)
plt.title('Multivariate Analysis: Correlation Between Class Counts')
plt.xlabel('Class')
plt.ylabel('Class')
plt.show()
```

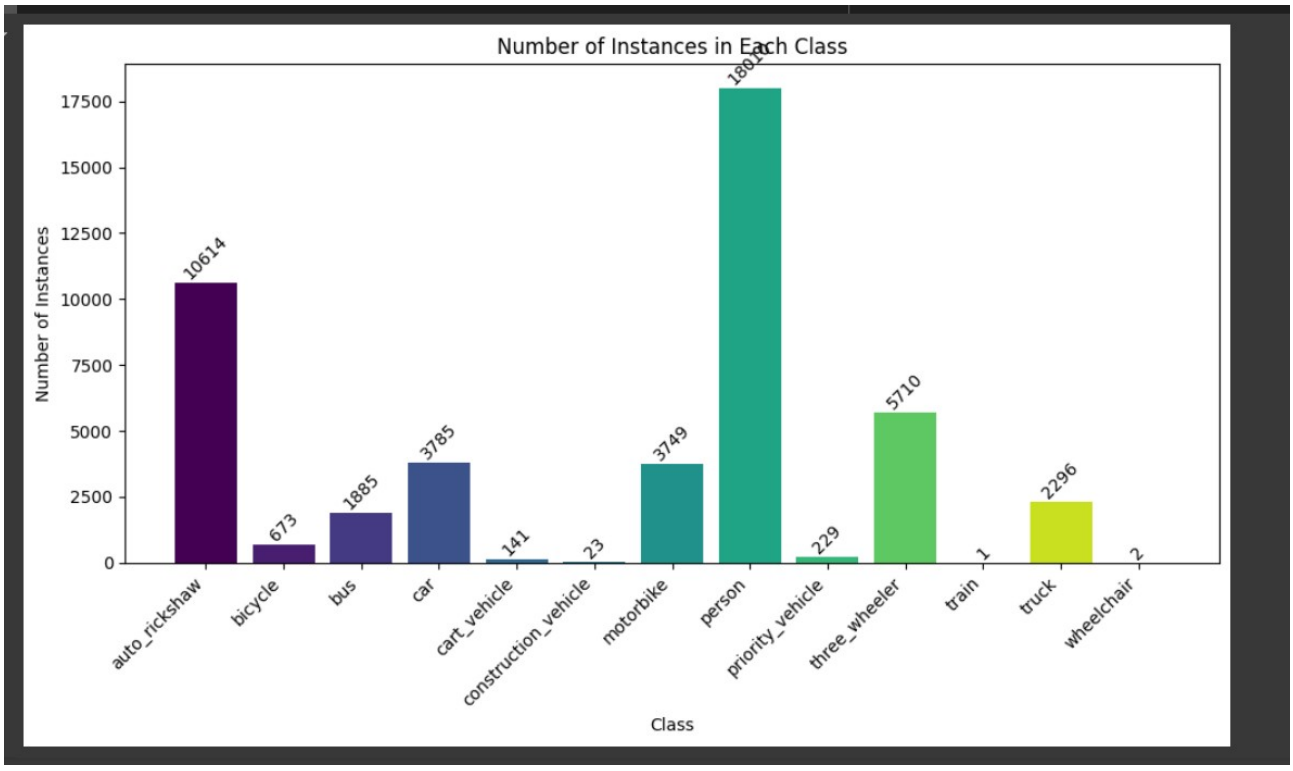
Output:



Code for Histogram (Multivariate):

```
# Create a bar plot
plt.figure(figsize=(10, 6))
bars = plt.bar(class_names, counts, color=colors)

# Add counts as annotations above each bar
for bar, count in zip(bars, counts):
plt.text(bar.get_x() + bar.get_width() / 2,
bar.get_height() + 5, # Adjust the position of the annotation
str(count),
ha='center', va='bottom', rotation=45)

plt.xlabel('Class')
plt.ylabel('Number of Instances')
plt.title('Number of Instances in Each Class')
plt.xticks(rotation=45, ha="right") # Rotate x-axis labels for better visibility
plt.tight_layout()
plt.show()
```

Output:



- **Interactive Visualizations:**

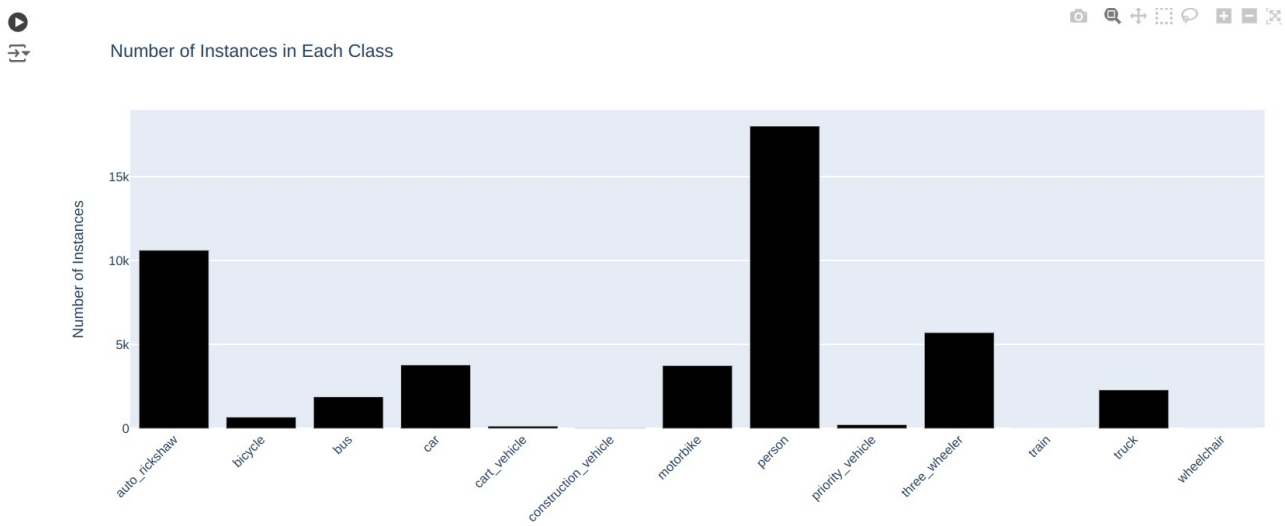
  - Interactive data visualization is the use of tools and processes to produce **a visual representation of data** which can be explored and analyzed directly within the visualization itself. This interaction can help **uncover insights** which lead to better, data-driven decisions.

Code snippet for Interactive Data Visualization:

Code:

```
import plotly.graph_objects as go

# Create a bar chart figure
fig = go.Figure(data=[go.Bar(
x=class_names,
y=counts,
marker=dict(color=colors)
)])

# Update the layout of the figure
fig.update_layout(
title='Number of Instances in Each Class',
xaxis=dict(title='Class'),
yaxis=dict(title='Number of Instances'),
xaxis_tickangle=-45 # Rotate x-axis labels for better visibility
)

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

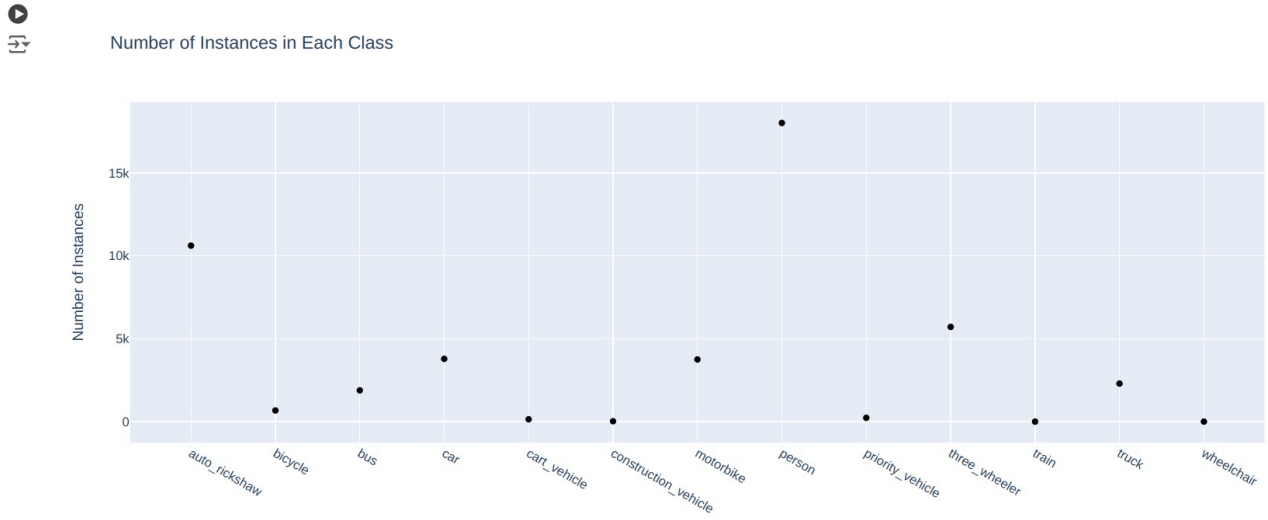
Output:



Code for Scatter plot:

```
import plotly.graph_objects as go

# Create a scatter plot figure
fig = go.Figure(data=go.Scatter(
x=class_names,
y=counts,
mode='markers',
marker=dict(color=colors)
))

# Update the layout of the figure
fig.update_layout(
title='Number of Instances in Each Class',
xaxis=dict(title='Class'),
yaxis=dict(title='Number of Instances')
)

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

Output:

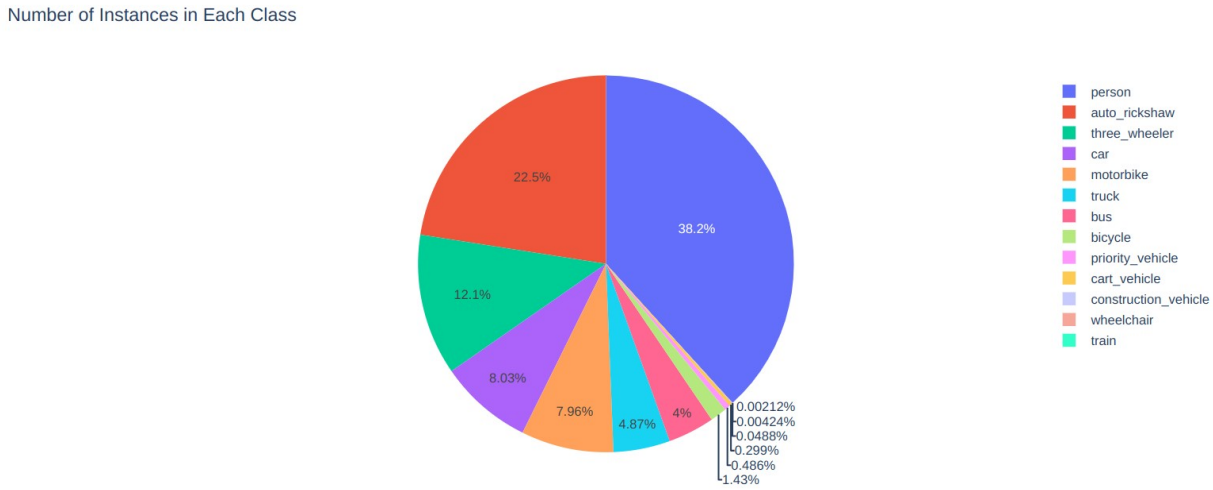


Code for Pie chart:

```
import plotly.graph_objects as go

# Create a pie chart figure
fig = go.Figure(data=[go.Pie(labels=class_names, values=counts)])

# Update the layout of the figure
fig.update_layout(title='Number of Instances in Each Class')
```

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

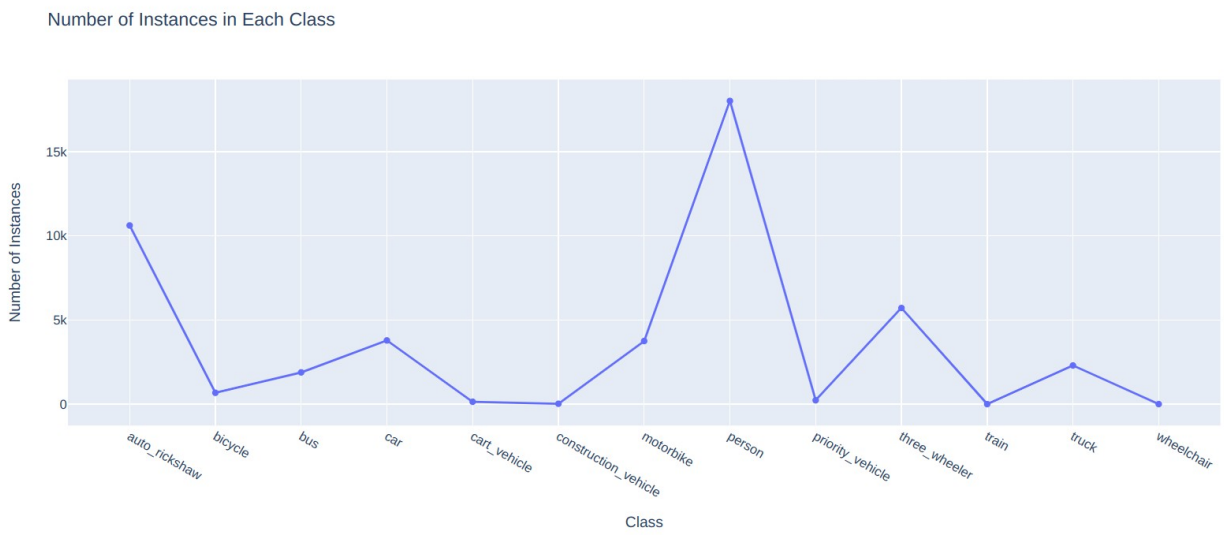
Output:



Code for line plot:

```
import plotly.graph_objects as go

# Create a line plot figure
fig = go.Figure(data=go.Scatter(
x=class_names,
y=counts,
mode='lines+markers'
))

# Update the layout of the figure
fig.update_layout(
title='Number of Instances in Each Class',
xaxis=dict(title='Class'),
yaxis=dict(title='Number of Instances')
)

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

Output:



Assumed Scenario:

- ➔ **Scenario:** The project focuses on **analyzing** and **visualizing** a dataset containing images of various road conditions in Bangladesh, annotated with **object bounding boxes.**
- ➔ **Objective:** The objective is to develop a **robust solution** for object detection in diverse road environments, **leveraging machine learning techniques.**
- ➔ **Target Audience:** The project is intended for **stakeholders** involved in road safety initiatives, **autonomous vehicle development,** and **traffic management,** seeking insights into road infrastructure and potential hazards.

Conclusion:

In the 3rd phase of the project, emphasis will be placed on employing **advanced data visualization techniques** to uncover meaningful insights and patterns within the dataset. Through the utilization of diverse visualization methods and assuming a scenario geared towards providing stakeholders with **interactive visualizations,** the goal is to facilitate **enhanced decision-making** and **comprehension of road conditions.** By presenting the data in intuitive and informative visual formats, the project aims to empower stakeholders with the tools necessary to make **informed decisions** and implement **effective strategies** for **road safety** and **traffic management** in Bangladesh.

LINKS:

Google colab:

https://colab.research.google.com/drive/1ByzXvGlofaDR4RgmKq2gQZ3ypwBSEQeN?usp=sharing

Github link:

https://github.com/Harihara04sudhan/naan-mudhalvan