AUTONOMOUS VEHICLES

A PROJECT REPORT (PHASE 2)

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 2 of our autonomous vehicle project embarks on a crucial endeavor dedicated solely to **data wrangling** and **analysis,** fundamental stages in refining the **raw dataset** essential for the development of our **AI-powered navigation system.** This phase constitutes a meticulous exploration of various data manipulation techniques, predominantly using **Python,** to **cleanse, transform,** and **scrutinize** the dataset. Within this context, we envision a scenario where the project aims to enhance the autonomy of vehicles by leveraging personalized content discovery techniques, thereby enriching the driving **experience** for passengers.

Objectives:

- **Data Cleansing:** Our primary objective is to ensure the integrity of the dataset by addressing **inconsistencies, errors,** and **missing values.** Through rigorous data **cleansing procedures,** we aim to enhance the reliability and accuracy of the dataset, laying a robust groundwork for subsequent analysis and modeling tasks.
- **Exploratory Data Analysis (EDA):** We strive to gain comprehensive insights into the dataset's characteristics through extensive **exploratory data analysis.** This involves unraveling intricate **distributions, correlations,** and **patterns** inherent within the dataset, providing valuable insights into the underlying data structure.
- **Feature Engineering:** Our focus extends to engineering relevant features tailored to augment the performance of our **AI-driven navigation system.** By identifying and incorporating pertinent features extracted from the dataset, we aim to optimize the **efficacy** and **adaptability** of the autonomous vehicle's **decision-making capabilities.**
- **Documentation:** Lastly, we prioritize the comprehensive documentation of the data wrangling process to ensure **transparency** and **reproducibility.** Through meticulous documentation, we aim to provide clarity on the methodologies employed during the data preparation phase, facilitating seamless collaboration and knowledge sharing among project stakeholders.

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.

Autonomous vehicles

- **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.

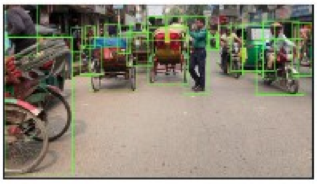
Test

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

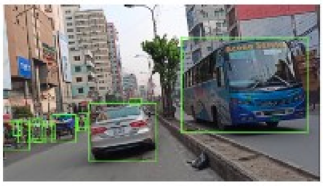
The test set has no provided annotations. Your task is to learn the bounding box from the train set and predict it for the test set images.

Sample Database:

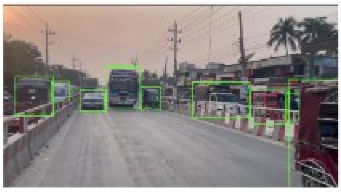
Sylhet



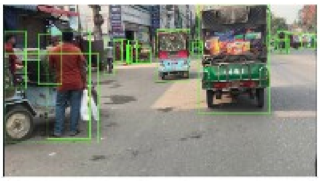
Dhaka



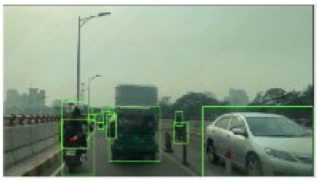
Rajshahi



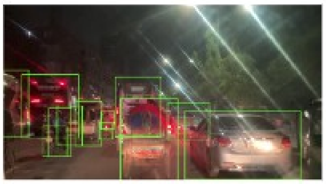
Mymensingh



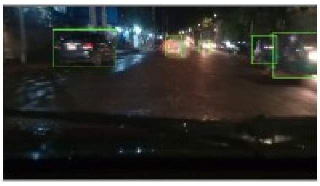
Maowa Expressway



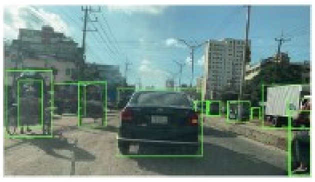
Dhaka Night



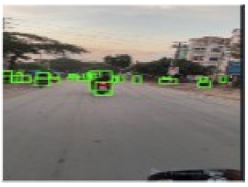
Chittagong Night



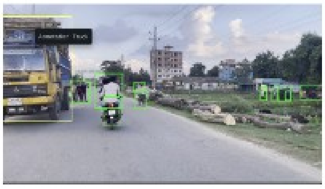
Chittagong Bohoddarhat



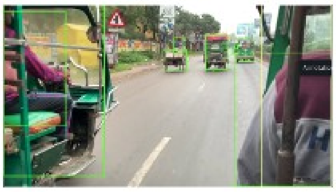
Chuadanga Sirajganj



Sherpur

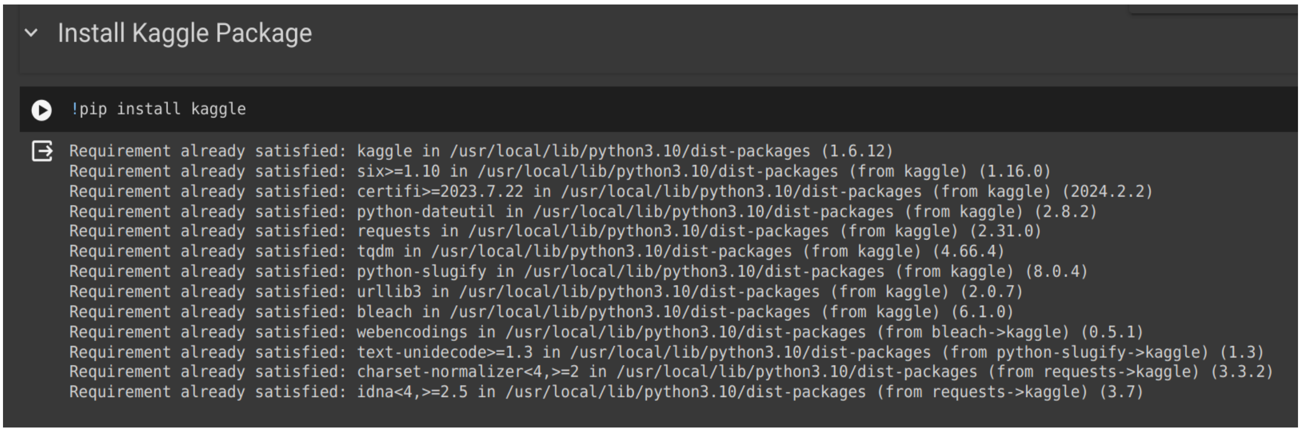


Khulna



Steps to be followed:

Step 1:

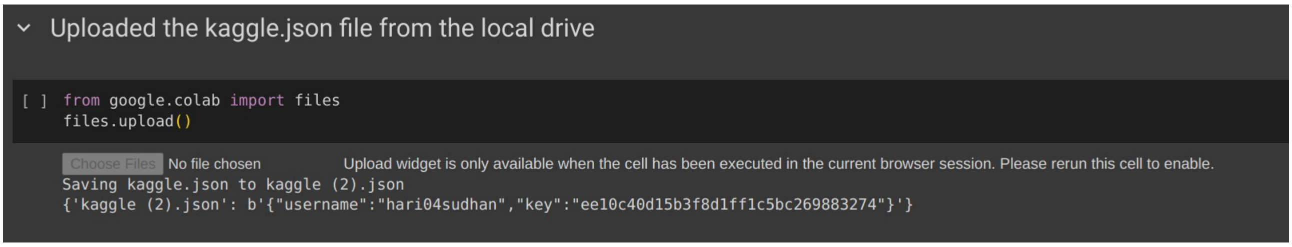


Install Kaggle Package

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.6.12)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.2.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.4)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.7)
```

Step 2:



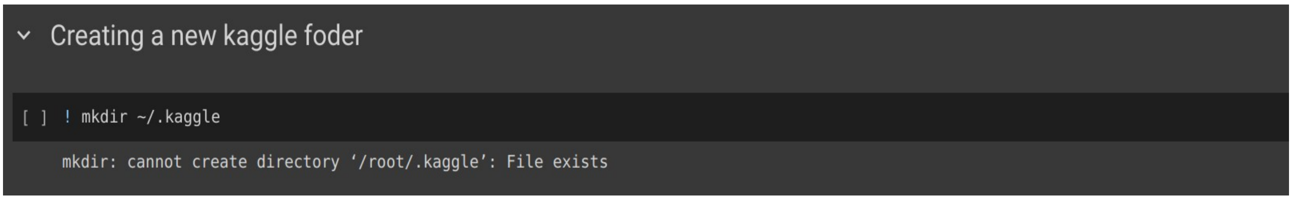
Uploaded the kaggle.json file from the local drive

```
[ ] from google.colab import files
    files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle (2).json
{'kaggle (2).json': b'{"username":"hari04sudhan","key":"ee10c40d15b3f8d1ff1c5bc269883274"}'}
```

Step 3:

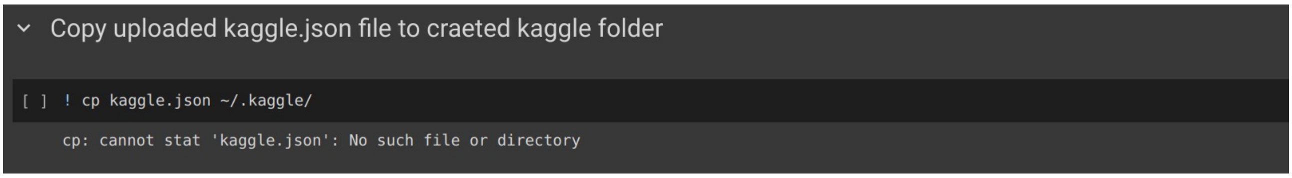


Creating a new kaggle foder

```
[ ] ! mkdir ~/.kaggle
```

```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
```

Step 4:

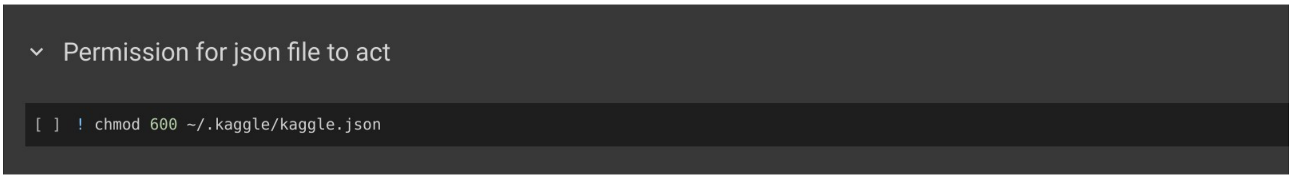


Copy uploaded kaggle.json file to craeted kaggle folder

```
[ ] ! cp kaggle.json ~/.kaggle/
```

```
cp: cannot stat 'kaggle.json': No such file or directory
```

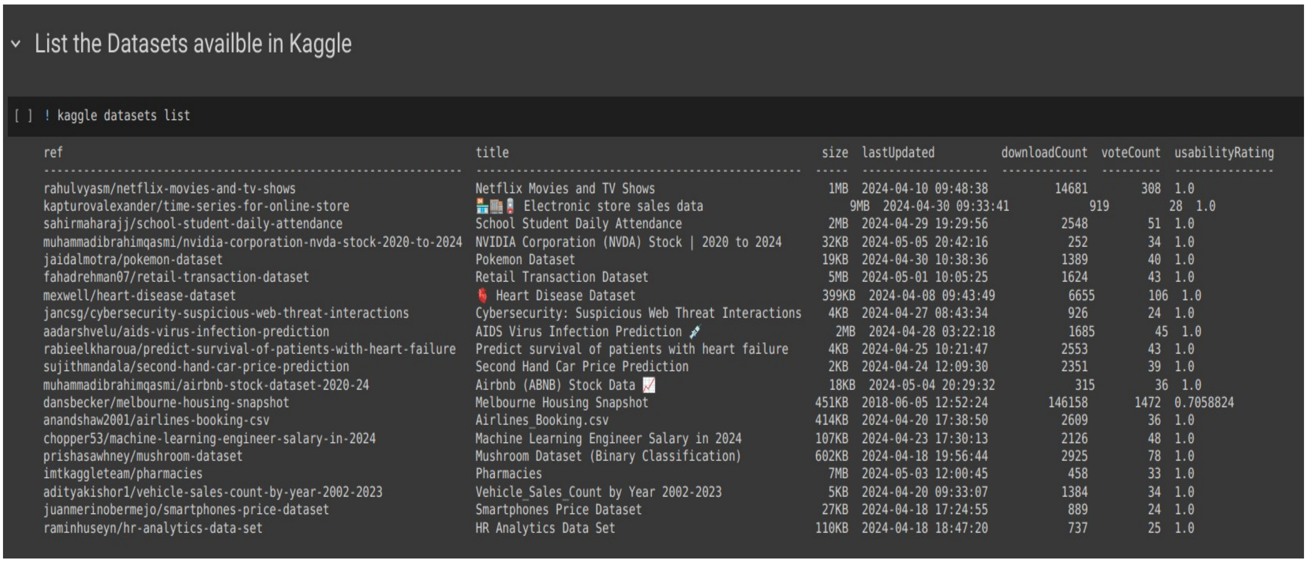
Step 5:



Permission for json file to act

```
[ ] ! chmod 600 ~/.kaggle/kaggle.json
```

Step 6:

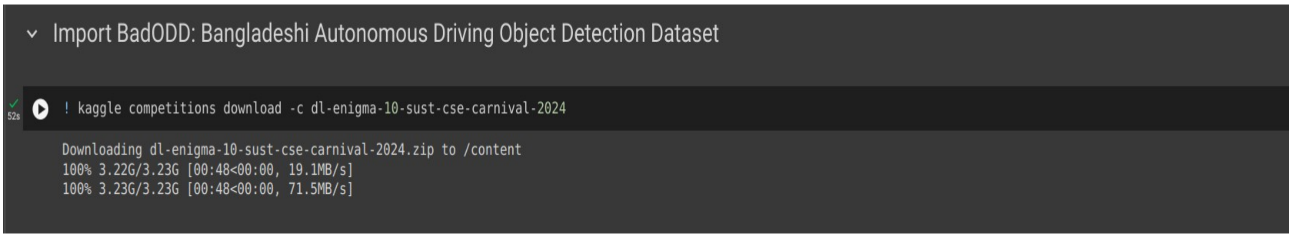


List the Datasets availble in Kaggle

```
[ ] ! kaggle datasets list
```

```
ref                                                           title                                          size  lastUpdated          downloadCount  voteCount  usabilityRating
------------------------------------------------------------  ---------------------------------------------  -----  -------------------  -------------  ---------  ---------------
rahulvyasm/netflix-movies-and-tv-shows                        Netflix Movies and TV Shows                      1MB  2024-04-10 09:48:38          14681        308  1.0
kapturovalexander/time-series-for-online-store                🏪🏬🚥 Electronic store sales data                   9MB  2024-04-30 09:33:41            919         28  1.0
sahirmaharajj/school-student-daily-attendance                 School Student Daily Attendance                  2MB  2024-04-29 19:29:56           2548         51  1.0
muhammadibrahimqasmi/nvidia-corporation-nvda-stock-2020-to-2024  NVIDIA Corporation (NVDA) Stock | 2020 to 2024  32KB  2024-05-05 20:42:16            252         34  1.0
jaidalmotra/pokemon-dataset                                   Pokemon Dataset                                 19KB  2024-04-30 10:38:36           1389         40  1.0
fahadrehman07/retail-transaction-dataset                      Retail Transaction Dataset                       5MB  2024-05-01 10:05:25           1624         43  1.0
mexwell/heart-disease-dataset                                 🫀 Heart Disease Dataset                        399KB  2024-04-08 09:43:49           6655        106  1.0
jancsg/cybersecurity-suspicious-web-threat-interactions       Cybersecurity: Suspicious Web Threat Interactions  4KB  2024-04-27 08:43:34            926         24  1.0
aadarshvelu/aids-virus-infection-prediction                   AIDS Virus Infection Prediction 💉                 2MB  2024-04-28 03:22:18           1685         45  1.0
rabieelkharoua/predict-survival-of-patients-with-heart-failure  Predict survival of patients with heart failure   4KB  2024-04-25 10:21:47           2553         43  1.0
sujithmandala/second-hand-car-price-prediction                Second Hand Car Price Prediction                 2KB  2024-04-24 12:09:30           2351         39  1.0
muhammadibrahimqasmi/airbnb-stock-dataset-2020-24             Airbnb (ABNB) Stock Data 📈                      18KB  2024-05-04 20:29:32            315         36  1.0
dansbecker/melbourne-housing-snapshot                         Melbourne Housing Snapshot                     451KB  2018-06-05 12:52:24         146158       1472  0.7058824
anandshaw2001/airlines-booking-csv                            Airlines_Booking.csv                           414KB  2024-04-20 17:38:50           2609         36  1.0
chopper53/machine-learning-engineer-salary-in-2024            Machine Learning Engineer Salary in 2024       107KB  2024-04-23 17:30:13           2126         48  1.0
prishasawhney/mushroom-dataset                                Mushroom Dataset (Binary Classification)       602KB  2024-04-18 19:56:44           2925         78  1.0
imtkaggleteam/pharmacies                                      Pharmacies                                       7MB  2024-05-03 12:00:45            458         33  1.0
adityakishor1/vehicle-sales-count-by-year-2002-2023           Vehicle_Sales_Count by Year 2002-2023            5KB  2024-04-20 09:33:07           1384         34  1.0
juanmerinobermejo/smartphones-price-dataset                   Smartphones Price Dataset                       27KB  2024-04-18 17:24:55            889         24  1.0
raminhuseyn/hr-analytics-data-set                             HR Analytics Data Set                          110KB  2024-04-18 18:47:20            737         25  1.0
```

Step 7:

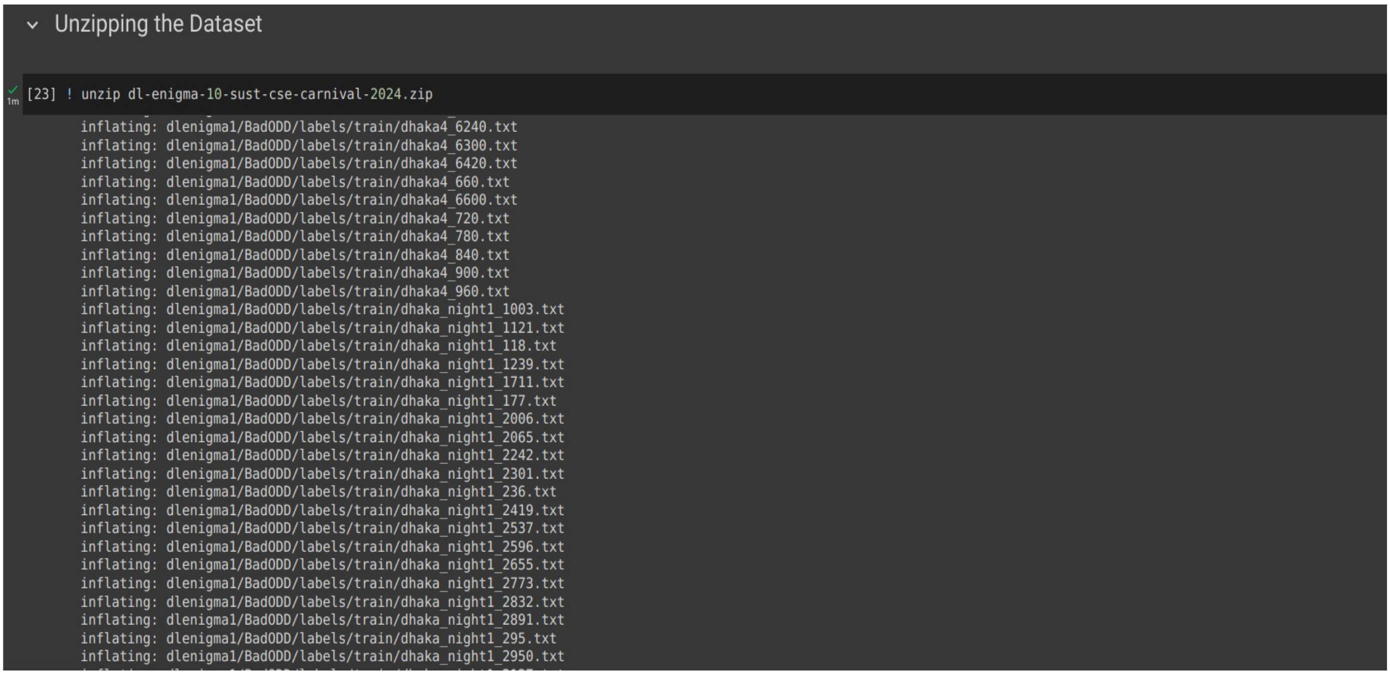


Import BadODD: Bangladeshi Autonomous Driving Object Detection Dataset

```
! kaggle competitions download -c dl-enigma-10-sust-cse-carnival-2024
```

```
Downloading dl-enigma-10-sust-cse-carnival-2024.zip to /content
100% 3.22G/3.23G [00:48<00:00, 19.1MB/s]
100% 3.23G/3.23G [00:48<00:00, 71.5MB/s]
```

Step 8:

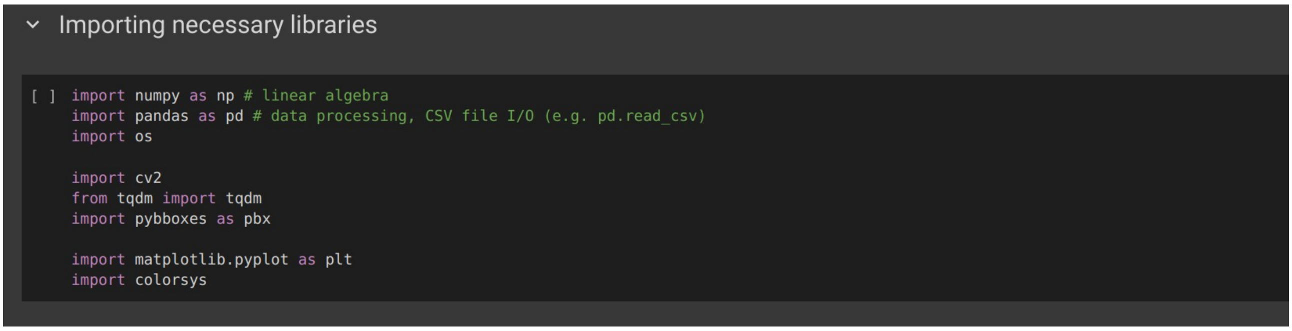


Unzipping the Dataset

```
[23] ! unzip dl-enigma-10-sust-cse-carnival-2024.zip
```

```
  inflating: dlenigma1/BadODD/labels/train/dhaka4_6240.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka4_6300.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka4_6420.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka4_660.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka4_6600.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka4_720.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka4_780.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka4_840.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka4_900.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka4_960.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_1003.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_1121.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_118.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_1239.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_1711.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_177.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2006.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2065.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2242.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2301.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_236.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2419.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2537.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2596.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2655.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2773.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2832.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2891.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_295.txt
  inflating: dlenigma1/BadODD/labels/train/dhaka_night1_2950.txt
```

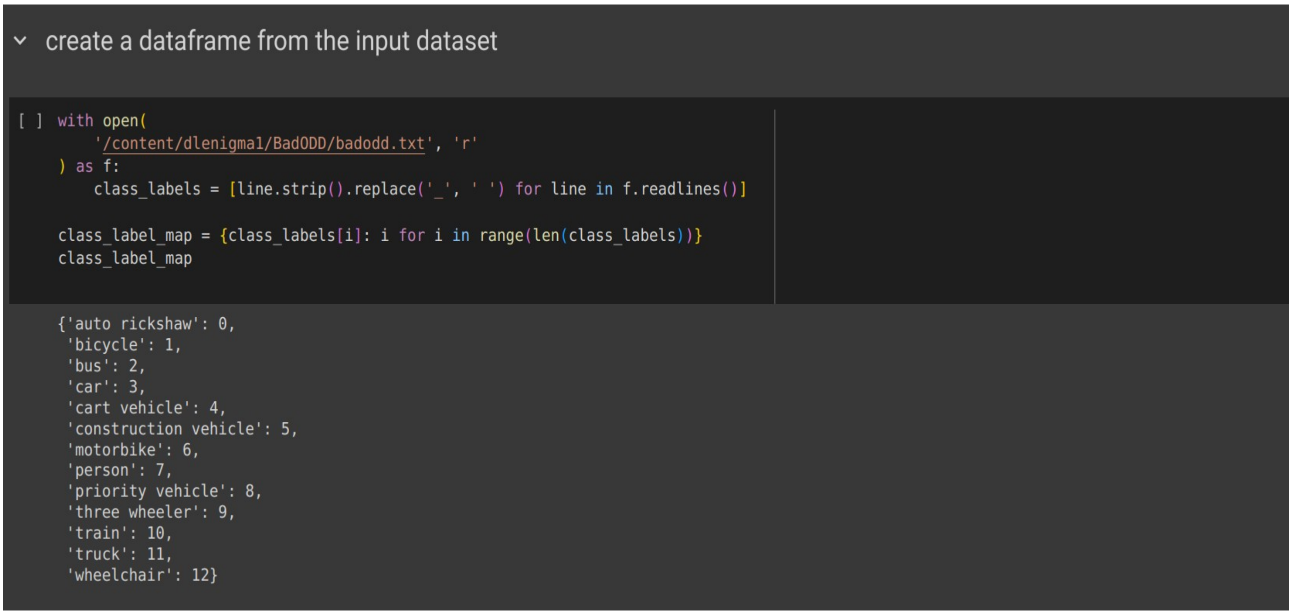
Step 9:



Importing necessary libraries

```
[ ] import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
    import os

    import cv2
    from tqdm import tqdm
    import pybboxes as pbx

    import matplotlib.pyplot as plt
    import colorsys
```

Step 10:



create a dataframe from the input dataset

```
[ ] with open(
        '/content/dlenigma1/BadODD/badodd.txt', 'r'
    ) as f:
        class_labels = [line.strip().replace('_', ' ') for line in f.readlines()]

    class_label_map = {class_labels[i]: i for i in range(len(class_labels))}
    class_label_map
```

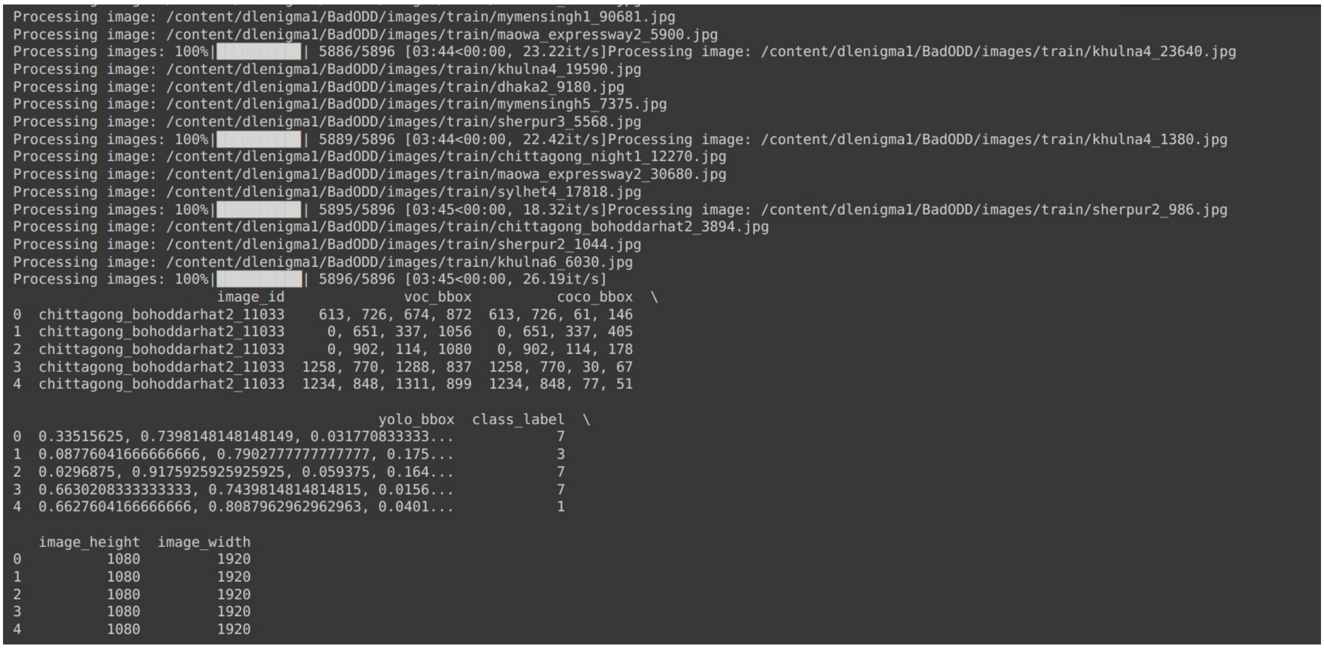
```
{'auto rickshaw': 0,
 'bicycle': 1,
 'bus': 2,
 'car': 3,
 'cart vehicle': 4,
 'construction vehicle': 5,
 'motorbike': 6,
 'person': 7,
 'priority vehicle': 8,
 'three wheeler': 9,
 'train': 10,
 'truck': 11,
 'wheelchair': 12}
```

Step 11: Training the data model

Code:

```
def get_possible_box_format(bbox, input_shape=None):
    if input_shape is None:
        return None, None, None


    voc_bbox = pbx.convert_bbox(
        bbox, from_type="yolo", to_type="voc", image_size=input_shape
    )
    coco_bbox = pbx.convert_bbox(
        bbox, from_type="yolo", to_type="coco", image_size=input_shape
    )
    yolo_bbox = ', '.join(map(str, bbox))
    return ', '.join(map(str, voc_bbox)), ', '.join(map(str, coco_bbox)), yolo_bbox


def prepare_dataframe(image_dir, label_dir):
    data = []
    for image_file in tqdm(os.listdir(image_dir), desc='Processing images'):
        img_path = os.path.join(image_dir, image_file)
        print(f"Processing image: {img_path}")
        img = cv2.imread(img_path)
        if img is None:
            print(f"Failed to read image: {img_path}. Skipping...")
            continue

        img_h, img_w = img.shape[:2]
        image_id = image_file.split('.')[0]
        label_file = os.path.join(label_dir, image_id + '.txt')

        if not os.path.exists(label_file):
            print(f"Label file not found for image: {image_file}. Skipping...")
            continue

        with open(label_file, 'r') as f:
            lines = f.readlines()
            for line in lines:
                class_label, *bbox = map(float, line.strip().split())
                class_label = int(class_label)
                voc_bbox, coco_bbox, yolo_bbox = get_possible_box_format(
                    bbox, input_shape=(img_w, img_h)
                )
                if voc_bbox is None or coco_bbox is None or yolo_bbox is None:
                    print(f"Failed to get box format for image: {image_file}. Skipping...")
                    continue
                data.append({
                    'image_id': image_id,
                    'voc_bbox': voc_bbox,
                    'coco_bbox': coco_bbox,
                    'yolo_bbox': yolo_bbox,
                    'class_label': class_label,
                    'image_height': img_h,
                    'image_width': img_w,
                })

    df = pd.DataFrame(data)
    return df

# Example usage:
image_dir = '/content/dlenigma1/BadODD/images/train'
label_dir = '/content/dlenigma1/BadODD/labels/train'
train_df = prepare_dataframe(image_dir, label_dir)
print(train_df.head())
```

Output:



```
Processing image: /content/dlenigma1/BadODD/images/train/mymensingh1_90681.jpg
Processing image: /content/dlenigma1/BadODD/images/train/maowa_expressway2_5900.jpg
Processing images: 100%|██████████| 5886/5896 [03:44<00:00, 23.22it/s]Processing image: /content/dlenigma1/BadODD/images/train/khulna4_23640.jpg
Processing image: /content/dlenigma1/BadODD/images/train/khulna4_19590.jpg
Processing image: /content/dlenigma1/BadODD/images/train/dhaka2_9180.jpg
Processing image: /content/dlenigma1/BadODD/images/train/mymensingh5_7375.jpg
Processing image: /content/dlenigma1/BadODD/images/train/sherpur3_5568.jpg
Processing images: 100%|██████████| 5889/5896 [03:44<00:00, 22.42it/s]Processing image: /content/dlenigma1/BadODD/images/train/khulna4_1380.jpg
Processing image: /content/dlenigma1/BadODD/images/train/chittagong_night1_12270.jpg
Processing image: /content/dlenigma1/BadODD/images/train/maowa_expressway2_30680.jpg
Processing image: /content/dlenigma1/BadODD/images/train/sylhet4_17818.jpg
Processing images: 100%|██████████| 5895/5896 [03:45<00:00, 18.32it/s]Processing image: /content/dlenigma1/BadODD/images/train/sherpur2_986.jpg
Processing image: /content/dlenigma1/BadODD/images/train/chittagong_bohoddarhat2_3894.jpg
Processing image: /content/dlenigma1/BadODD/images/train/sherpur2_1044.jpg
Processing image: /content/dlenigma1/BadODD/images/train/khulna6_6030.jpg
Processing images: 100%|██████████| 5896/5896 [03:45<00:00, 26.19it/s]
                        image_id             voc_bbox          coco_bbox  \
0  chittagong_bohoddarhat2_11033    613, 726, 674, 872  613, 726, 61, 146
1  chittagong_bohoddarhat2_11033     0, 651, 337, 1056   0, 651, 337, 405
2  chittagong_bohoddarhat2_11033     0, 902, 114, 1080   0, 902, 114, 178
3  chittagong_bohoddarhat2_11033  1258, 770, 1288, 837  1258, 770, 30, 67
4  chittagong_bohoddarhat2_11033  1234, 848, 1311, 899  1234, 848, 77, 51

                                           yolo_bbox  class_label  \
0  0.33515625, 0.7398148148148149, 0.031770833333...            7
1  0.08776041666666666, 0.7902777777777777, 0.175...            3
2  0.0296875, 0.9175925925925925, 0.059375, 0.164...            7
3  0.6630208333333333, 0.7439814814814815, 0.0156...            7
4  0.6627604166666666, 0.8087962962962963, 0.0401...            1

   image_height  image_width
0          1080         1920
1          1080         1920
2          1080         1920
3          1080         1920
4          1080         1920
```

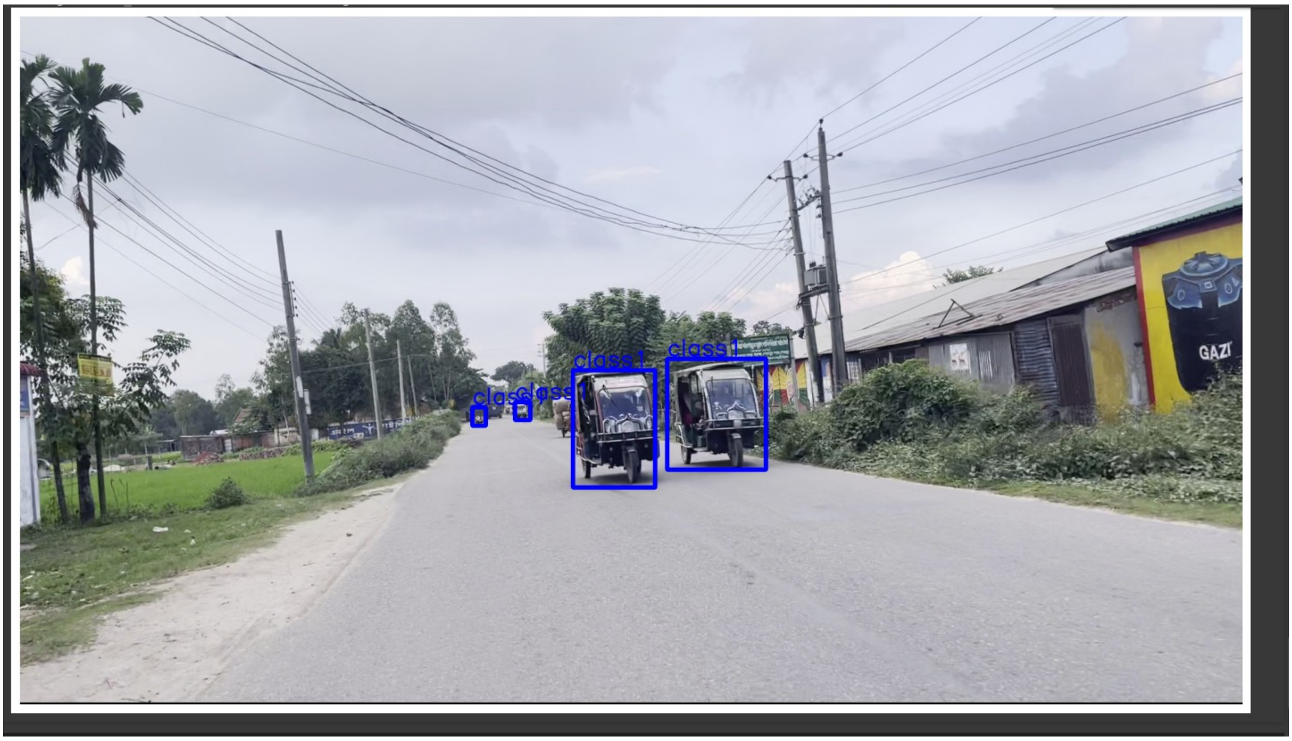
Step 12: Performing Visualizations

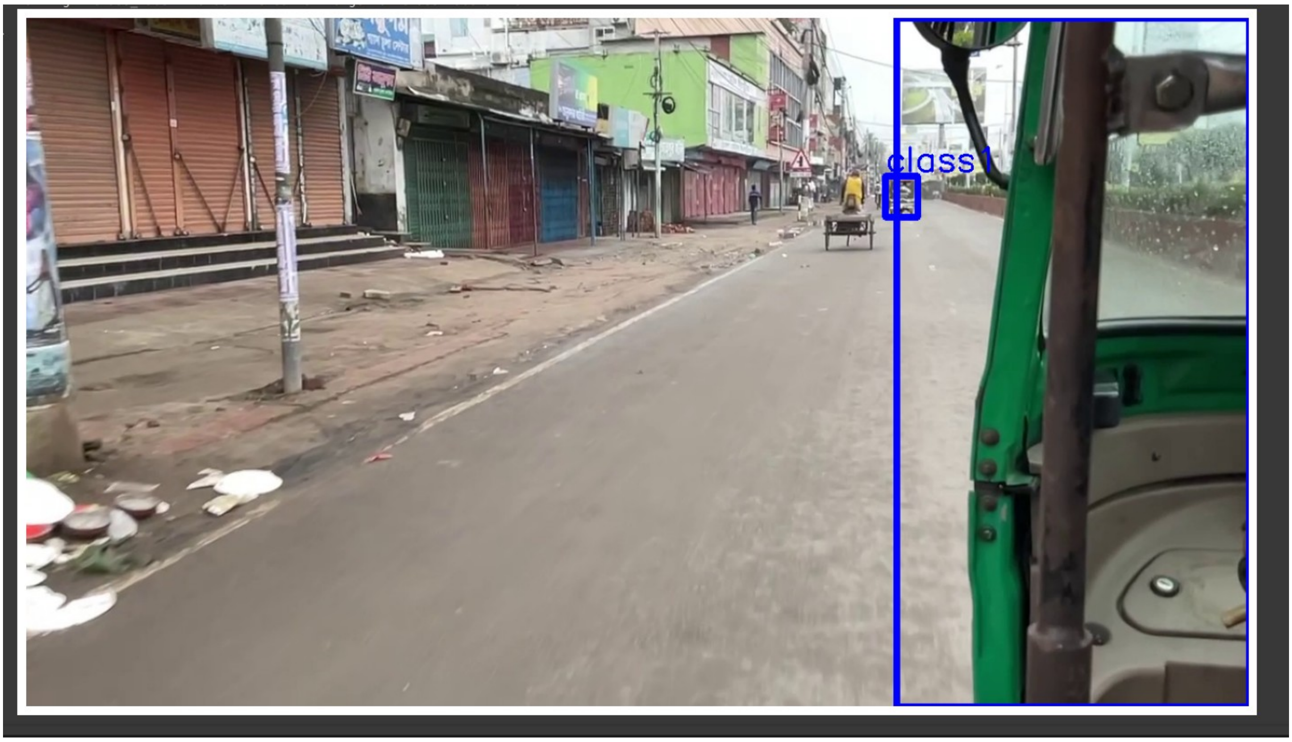
Code Snippet:

```
thickness = 5
font_scale = 1.2
font_thickness = 2

def generate_colors(num_classes):
    hsv_tuples = [(x / num_classes, 1., 1.) for x in range(num_classes)]
    colors = list(map(lambda x: colorsys.hsv_to_rgb(*x), hsv_tuples))
    colors = list(map(lambda x: (int(x[0] * 255), int(x[1] * 255), int(x[2] * 255)), colors))
    return colors

def draw_boxes(image_path, df, class_labels=None):
    image = cv2.imread(image_path)

    if class_labels is None:
        class_labels = ['class1', 'class2']  # Default class labels

    colors = generate_colors(len(class_labels))
    class_color_map = {class_labels[i]: colors[i] for i in range(len(class_labels))}

    if 'class_label' in df.columns:
        for _, row in df.iterrows():
            box = eval(row['voc_bbox'])
            class_label = row['class_label']
            if class_label < len(class_labels):
                color = class_color_map[class_labels[class_label]]
                box = [int(coord) for coord in box]

                cv2.rectangle(image, (box[0], box[1]), (box[2], box[3]), color, thickness)
                cv2.putText(
                    image,
                    class_labels[class_label],
                    (box[0], box[1] - 5),
                    cv2.FONT_HERSHEY_SIMPLEX,
                    font_scale,
                    color,
                    font_thickness
                )
            else:
                print(f"Warning: 'class_label' value {class_label} is out of range for class labels list.")
    else:
        print("Warning: 'class_label' column not found. Using default class labels.")

        for _, row in df.iterrows():
            box = eval(row['voc_bbox'])
            class_name = row['class_name'] if 'class_name' in df.columns else 'Unknown'
            color = class_color_map[class_name]
            box = [int(coord) for coord in box]

            cv2.rectangle(image, (box[0], box[1]), (box[2], box[3]), color, thickness)
            cv2.putText(
                image,
                class_name,
                (box[0], box[1] - 5),
                cv2.FONT_HERSHEY_SIMPLEX,
                font_scale,
                color,
                font_thickness
            )

    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

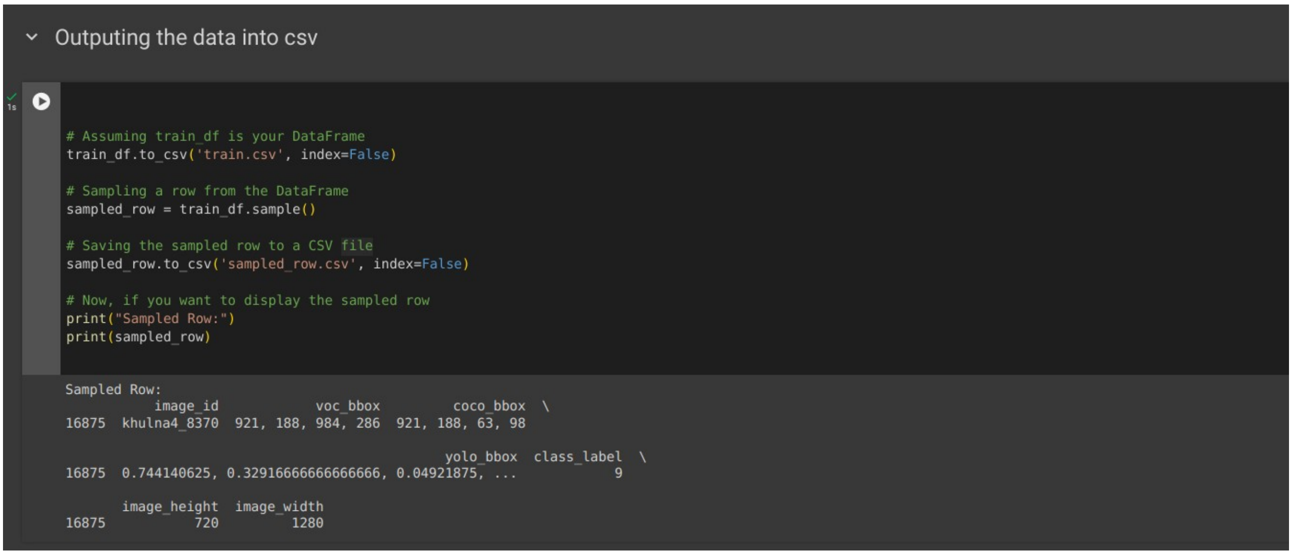
```
    plt.figure(figsize=(20,10))

    plt.imshow(image_rgb)

    plt.axis('off')

    plt.show()



image_id = train_df.sample().reset_index().loc[0, 'image_id']

image_path = '/content/dlenigma1/BadODD/images/train'

image_path = os.path.join(image_path, image_id + '.jpg')

df_image = train_df[train_df['image_id'] == image_id]

class_labels = ['class1', 'class2']

draw_boxes(image_path, df_image, class_labels)
```

Output:



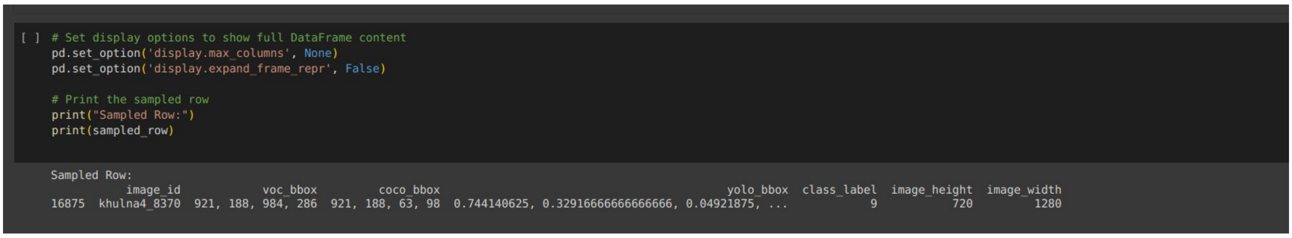


Step 13: Output into CSV


Outputting the data into csv

```
# Assuming train_df is your DataFrame
train_df.to_csv('train.csv', index=False)

# Sampling a row from the DataFrame
sampled_row = train_df.sample()

# Saving the sampled row to a CSV file
sampled_row.to_csv('sampled_row.csv', index=False)

# Now, if you want to display the sampled row
print("Sampled Row:")
print(sampled_row)
```

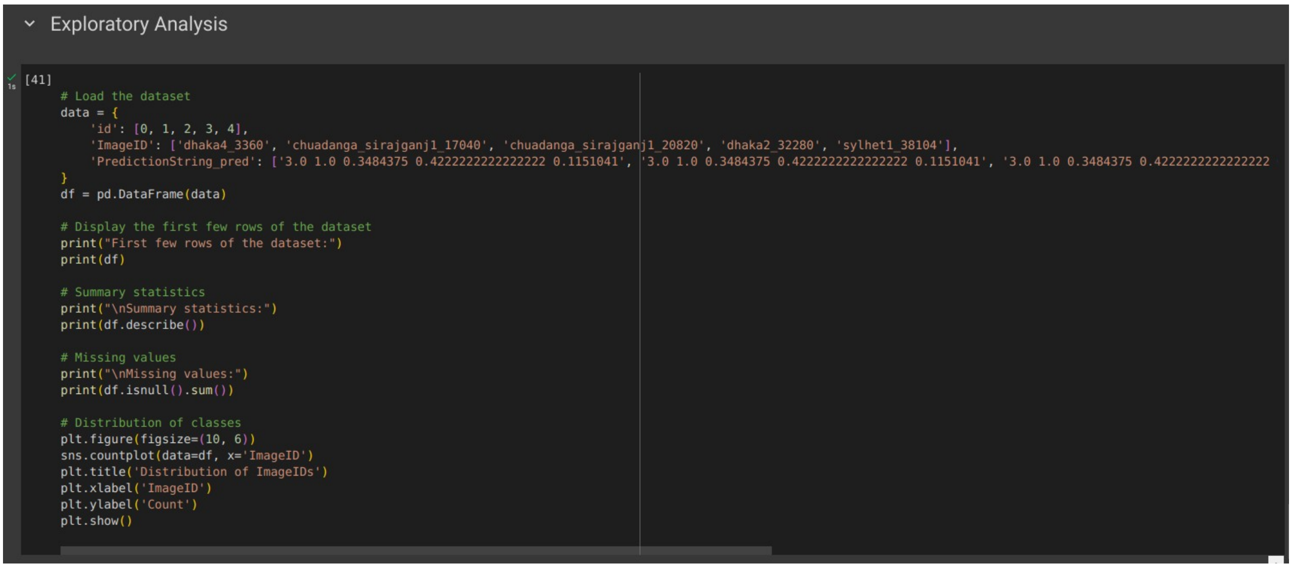
```
Sampled Row:
            image_id            voc_bbox         coco_bbox  \
16875  khulna4_8370  921, 188, 984, 286  921, 188, 63, 98

                                               yolo_bbox  class_label  \
16875  0.744140625, 0.32916666666666666, 0.04921875, ...            9

       image_height  image_width
16875           720         1280
```



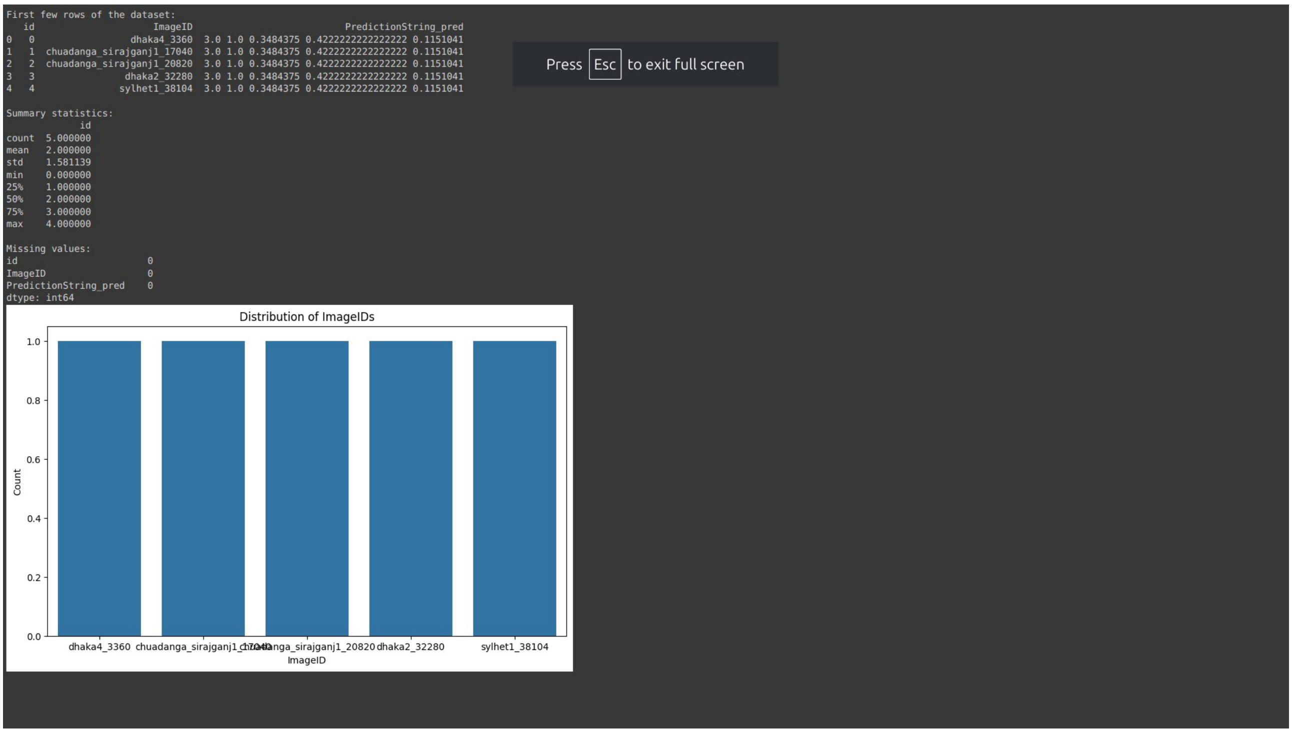
```
[ ] # Set display options to show full DataFrame content
    pd.set_option('display.max_columns', None)
    pd.set_option('display.expand_frame_repr', False)

    # Print the sampled row
    print("Sampled Row:")
    print(sampled_row)
```

```
Sampled Row:
            image_id            voc_bbox         coco_bbox                                          yolo_bbox  class_label  image_height  image_width
16875  khulna4_8370  921, 188, 984, 286  921, 188, 63, 98  0.744140625, 0.32916666666666666, 0.04921875, ...            9           720         1280
```

Step 14:



```
# Load the dataset
data = {
    'id': [0, 1, 2, 3, 4],
    'ImageID': ['dhaka4_3360', 'chuadanga_sirajganj1_17040', 'chuadanga_sirajganj1_20820', 'dhaka2_32280', 'sylhet1_38104'],
    'PredictionString_pred': ['3.0 1.0 0.3484375 0.4222222222222222 0.1151041', '3.0 1.0 0.3484375 0.4222222222222222 0.1151041', '3.0 1.0 0.3484375 0.4222222222222222
}
df = pd.DataFrame(data)

# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(df)

# Summary statistics
print("\nSummary statistics:")
print(df.describe())

# Missing values
print("\nMissing values:")
print(df.isnull().sum())

# Distribution of classes
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='ImageID')
plt.title('Distribution of ImageIDs')
plt.xlabel('ImageID')
plt.ylabel('Count')
plt.show()
```



```
First few rows of the dataset:
   id                     ImageID                             PredictionString_pred
0   0                 dhaka4_3360  3.0 1.0 0.3484375 0.4222222222222222 0.1151041
1   1  chuadanga_sirajganj1_17040  3.0 1.0 0.3484375 0.4222222222222222 0.1151041
2   2  chuadanga_sirajganj1_20820  3.0 1.0 0.3484375 0.4222222222222222 0.1151041
3   3                dhaka2_32280  3.0 1.0 0.3484375 0.4222222222222222 0.1151041
4   4               sylhet1_38104  3.0 1.0 0.3484375 0.4222222222222222 0.1151041

Summary statistics:
             id
count  5.000000
mean   2.000000
std    1.581139
min    0.000000
25%    1.000000
50%    2.000000
75%    3.000000
max    4.000000

Missing values:
id                       0
ImageID                  0
PredictionString_pred    0
dtype: int64
```

Conclusion:

The journey into object detection across varied road conditions in **Bangladesh** commenced with a **meticulous exploration** of a dataset spanning nine districts, comprising 9,825 images meticulously annotated with rectangular bounding boxes. Representing a diverse array of road landscapes, from bustling towns to tranquil village roads, the dataset encapsulates **78,943 objects** across 1**3 distinct classes**, including vehicles such as **auto-rickshaws, buses, and bicycles,** as well as non-vehicle entities like **people and wheelchairs.** Ensuring authenticity, images were captured using smartphone cameras, replicating real-world driving scenarios and fostering the development of models with practical applicability. Implementation involved hands-on code development for drawing **bounding boxes** around **detected objects, navigating DataFrame** operations with **finesse,** and addressing common errors. This endeavor underscores the significance of realistic datasets and effective model implementations in advancing **computer vision,** particularly for autonomous driving systems, with the potential to enhance **road safety** and navigation not only in Bangladesh but also beyond.

LINKS:

Google colab:

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

Github link:

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