Problem statement 1:

Autonomous vehicles (AV) and intelligent transport systems (ITS) are the future of road transport. Automatic detection of vehicles on the road in real-time helps AV technology and makes ITS more intelligent in terms of vehicle tracking, vehicle counting, and road incident response.

Objective part 1:

As the first part of this project, you need to develop an AI model using a deep learning framework that predicts the type of vehicle present in an image as well as localizes the vehicle by rectangular bounding box.

In [ ]:
```
#Import the packages
from ultralytics import YOLO
import pandas as pd
import os
import shutil
from PIL import Image
```

Object detection

1. Create a parent folder for custom model training and child folders to store data
2. Prepare the dataset for model training and keep the following points in mind while preparing it • This dataset contains many images, and depending on the compute power of the VM, it might take a very long time to unzip this huge amount of data

In [ ]:
```
# Define paths
from ultralytics import YOLO
import pandas as pd
import os
import shutil
from PIL import Image



csv_path = "./sample_data/Part1/labels.csv"
img_folder = "./sample_data/Part1/images/"
yolo_dataset_path = "./sample_data/yolo_dataset/"
yolo_images_dir = os.path.join(yolo_dataset_path, "images")
yolo_labels_dir = os.path.join(yolo_dataset_path, "labels")

# Create necessary folders
for split in ['train', 'val']:
    os.makedirs(os.path.join(yolo_images_dir, split), exist_ok=True)
    os.makedirs(os.path.join(yolo_labels_dir, split), exist_ok=True)

# Load CSV
df = pd.read_csv(csv_path, header=None)
df.columns = ['image_id', 'class_name', 'x_min', 'y_min', 'x_max', 'y_max']

# Class name to ID mapping
class_map = {name: idx for idx, name in enumerate(df['class_name'].unique())}

# Group by image
grouped = df.groupby('image_id')
print(grouped.head())
# Simple train/val split
image_ids = df['image_id'].unique()
threshold = 5657
filtered_img = [s for s in image_ids if int(s) <= threshold]
split_idx = int(len(filtered_img) * 0.8)
train_ids = set(filtered_img[:split_idx])

for image_id, group in grouped:
    file_name = f"{str(image_id).zfill(8)}.jpg"
    orig_path = os.path.join(img_folder, file_name)

    if not os.path.exists(orig_path):
        continue  # skip if image is missing

    # Determine split
    split = 'train' if image_id in train_ids else 'val'

    # Copy image
    dst_img_path = os.path.join(yolo_images_dir, split, file_name)
    shutil.copy(orig_path, dst_img_path)

    # Get image size
    with Image.open(orig_path) as img:
        width, height = img.size
```

```python
    # Convert annotations
    yolo_lines = []
    for _, row in group.iterrows():
        class_id = class_map[row['class_name']]
        x_center = (row['x_min'] + row['x_max']) / 2 / width
        y_center = (row['y_min'] + row['y_max']) / 2 / height
        bbox_width = (row['x_max'] - row['x_min']) / width
        bbox_height = (row['y_max'] - row['y_min']) / height
        yolo_lines.append(f"{class_id} {x_center} {y_center} {bbox_width} {bbox_height}")

    # Save label file
    label_path = os.path.join(yolo_labels_dir, split, file_name.replace(".jpg", ".txt"))
    with open(label_path, "w") as f:
        f.write("\n".join(yolo_lines))
```

```
        image_id             class_name  x_min  y_min  x_max  y_max
0              0           pickup_truck    213     34    255     50
1              0                    car    194     78    273    122
2              0                    car    155     27    183     35
3              0      articulated_truck     43     25    109     55
4              0                    car    106     32    124     45
...          ...                    ...    ...    ...    ...    ...
351544    110590                    car     18     57     97     98
351545    110591      articulated_truck      2     71    690    351
351546    110592           pickup_truck      3    240    214    378
351547    110592                    car    465    111    507    135
351548    110592  non-motorized_vehicle    197    187    318    269

[293274 rows x 6 columns]
```

In [ ]:

```python
from glob import glob
print("Train images:", len(glob(os.path.join(yolo_images_dir, 'train', '*.jpg'))))
print("Val images:", len(glob(os.path.join(yolo_images_dir, 'val', '*.jpg'))))
```

```
Train images: 4500
Val images: 1126
```

In [ ]:

```python
#create the yolo config file
yolo_dataset_path = "./sample_data/yolo_dataset/"
yaml_path = "./sample_data/yolo_dataset/data.yaml"
with open(yaml_path, "w") as f:
    f.write(f"""
path: {yolo_dataset_path}
train: images/train
val: images/val

nc: {len(class_map)}
names: {list(class_map.keys())}
""")
```

3. Create an CNN architecture for object detection of your choice to train an object detection model. Please note that algorithm or architecture selection is a very important aspect of ML model training, and you must pick the one that works the best for your dataset

In [ ]:

```python
model = YOLO("yolov8n.pt")  # Use yolov8n for speed, or yolov8s/m/l/x for accuracy
results = model.train(data=yaml_path, epochs=20, imgsz=640)
```

```
Ultralytics 8.3.141  Python-3.11.12 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
engine/trainer: agnostic_nms=False, amp=True, augment=False, auto_augment=randaugment, batch=16, bgr=0.0, box=7.
5, cache=False, cfg=None, classes=None, close_mosaic=10, cls=0.5, conf=None, copy_paste=0.0, copy_paste_mode=fli
p, cos_lr=False, cutmix=0.0, data=./sample_data/yolo_dataset/data.yaml, degrees=0.0, deterministic=True, device=
None, dfl=1.5, dnn=False, dropout=0.0, dynamic=False, embed=None, epochs=20, erasing=0.4, exist_ok=False, fliplr
=0.5, flipud=0.0, format=torchscript, fraction=1.0, freeze=None, half=False, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4,
imgsz=640, int8=False, iou=0.7, keras=False, kobj=1.0, line_width=None, lr0=0.01, lrf=0.01, mask_ratio=4, max_de
t=300, mixup=0.0, mode=train, model=yolov8n.pt, momentum=0.937, mosaic=1.0, multi_scale=False, name=train2, nbs=
64, nms=False, opset=None, optimize=False, optimizer=auto, overlap_mask=True, patience=100, perspective=0.0, plo
ts=True, pose=12.0, pretrained=True, profile=False, project=None, rect=False, resume=False, retina_masks=False,
save=True, save_conf=False, save_crop=False, save_dir=runs/detect/train2, save_frames=False, save_json=False, sa
ve_period=-1, save_txt=False, scale=0.5, seed=0, shear=0.0, show=False, show_boxes=True, show_conf=True, show_la
bels=True, simplify=True, single_cls=False, source=None, split=val, stream_buffer=False, task=detect, time=None,
tracker=botsort.yaml, translate=0.1, val=True, verbose=True, vid_stride=1, visualize=False, warmup_bias_lr=0.1,
warmup_epochs=3.0, warmup_momentum=0.8, weight_decay=0.0005, workers=8, workspace=None
Overriding model.yaml nc=80 with nc=11

                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22        [15, 18, 21]  1    753457  ultralytics.nn.modules.head.Detect           [11, [64, 128, 256]]
Model summary: 129 layers, 3,012,993 parameters, 3,012,977 gradients, 8.2 GFLOPs

Transferred 319/355 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
AMP: checks passed ✅
train: Fast image access ✅ (ping: 0.0±0.0 ms, read: 815.5±333.6 MB/s, size: 24.9 KB)
train: Scanning /content/sample_data/yolo_dataset/labels/train.cache... 4500 images, 0 backgrounds, 0 corrupt: 1
00%|██████████| 4500/4500 [00:00<?, ?it/s]
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01, method='w
eighted_average', num_output_channels=3), CLAHE(p=0.01, clip_limit=(1.0, 4.0), tile_grid_size=(8, 8))
val: Fast image access ✅ (ping: 0.0±0.0 ms, read: 634.1±367.1 MB/s, size: 23.5 KB)
val: Scanning /content/sample_data/yolo_dataset/labels/val.cache... 1126 images, 0 backgrounds, 0 corrupt: 100%|
██████████| 1126/1126 [00:00<?, ?it/s]
Plotting labels to runs/detect/train2/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'l
r0' and 'momentum' automatically...
optimizer: AdamW(lr=0.000667, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005),
63 bias(decay=0.0)
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/detect/train2
Starting training for 20 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/20      2.16G      1.231      2.372       1.08         14        640: 100%|██████████| 282/282 [01:22<0
0:00,  3.43it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:11<00:00,  3.27it/s]
                   all       1126       3806      0.451      0.228      0.215      0.155

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/20       2.3G       1.16      1.595      1.044         23        640: 100%|██████████| 282/282 [01:33<0
0:00,  3.00it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:09<00:00,  3.98it/s]
                   all       1126       3806      0.394      0.302      0.285      0.194

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
       3/20      2.3G      1.14     1.403      1.04        25       640: 100%|██████████| 282/282 [01:18<0
0:00,  3.58it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:08<00:00,  4.02it/s]
                   all       1126       3806      0.462      0.365      0.343      0.232

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/20      2.3G     1.115     1.259     1.032        22       640: 100%|██████████| 282/282 [01:17<0
0:00,  3.62it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:08<00:00,  4.32it/s]
                   all       1126       3806      0.541      0.417      0.419      0.291

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       5/20      2.3G      1.08     1.151     1.019        16       640: 100%|██████████| 282/282 [01:18<0
0:00,  3.60it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:09<00:00,  3.89it/s]
                   all       1126       3806      0.439      0.437       0.41      0.279

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       6/20      2.3G     1.068     1.086     1.013        20       640: 100%|██████████| 282/282 [01:16<0
0:00,  3.69it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:09<00:00,  3.80it/s]
                   all       1126       3806      0.418      0.526      0.452      0.319

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       7/20     2.31G     1.037     1.018     1.003        28       640: 100%|██████████| 282/282 [01:15<0
0:00,  3.73it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:08<00:00,  4.38it/s]
                   all       1126       3806       0.55      0.463      0.475      0.334

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/20     2.33G      1.04    0.9863     1.004        22       640: 100%|██████████| 282/282 [01:16<0
0:00,  3.67it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:08<00:00,  4.43it/s]
                   all       1126       3806      0.554      0.433       0.46      0.318

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       9/20     2.33G     1.024    0.9512    0.9983        17       640: 100%|██████████| 282/282 [01:16<0
0:00,  3.69it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:09<00:00,  3.90it/s]
                   all       1126       3806      0.478      0.538       0.51      0.357

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/20     2.33G     1.014    0.9296    0.9955        26       640: 100%|██████████| 282/282 [01:16<0
0:00,  3.71it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:08<00:00,  4.08it/s]
                   all       1126       3806      0.474       0.55      0.528      0.378

Closing dataloader mosaic
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01, method='w
eighted_average', num_output_channels=3), CLAHE(p=0.01, clip_limit=(1.0, 4.0), tile_grid_size=(8, 8))

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      11/20     2.34G    0.9834    0.8624    0.9731         8       640: 100%|██████████| 282/282 [01:15<0
0:00,  3.74it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:08<00:00,  4.31it/s]
                   all       1126       3806      0.571      0.517      0.554      0.392

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      12/20     2.37G    0.9693    0.8148    0.9642        11       640: 100%|██████████| 282/282 [01:12<0
0:00,  3.88it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:08<00:00,  4.05it/s]
                   all       1126       3806      0.581      0.554      0.562      0.396

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      13/20     2.37G    0.9593    0.7927    0.9597        18       640: 100%|██████████| 282/282 [01:12<0
0:00,  3.92it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36
[00:08<00:00,  4.12it/s]
                   all       1126       3806      0.507      0.582      0.544      0.389

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

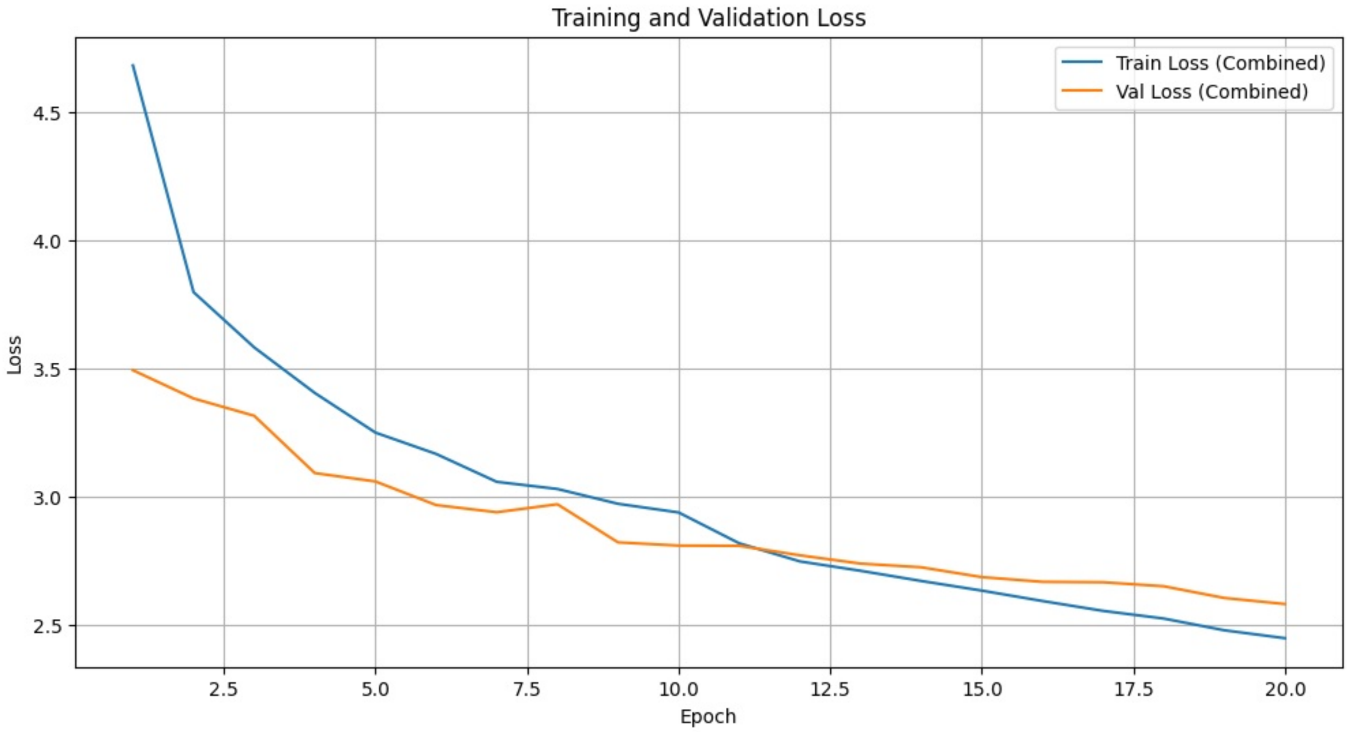
```
      14/20      2.37G     0.9468      0.767     0.9581         10        640: 100%|██████████| 282/282 [01:12<00:00,  3.87it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36 [00:08<00:00,  4.17it/s]
                   all       1126       3806      0.524      0.575      0.568      0.408

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      15/20      2.37G     0.9396     0.7394     0.9553         11        640: 100%|██████████| 282/282 [01:13<00:00,  3.84it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36 [00:09<00:00,  3.95it/s]
                   all       1126       3806      0.568      0.571      0.571      0.413
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      16/20      2.37G     0.9285     0.7166     0.9487         13        640: 100%|██████████| 282/282 [01:12<00:00,  3.87it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36 [00:08<00:00,  4.42it/s]
                   all       1126       3806      0.574      0.598      0.597      0.426
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      17/20      2.37G     0.9121     0.6981      0.945          9        640: 100%|██████████| 282/282 [01:12<00:00,  3.89it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36 [00:08<00:00,  4.02it/s]
                   all       1126       3806      0.551      0.617      0.588      0.428
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      18/20      2.37G     0.9094     0.6752     0.9408         12        640: 100%|██████████| 282/282 [01:11<00:00,  3.93it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36 [00:08<00:00,  4.06it/s]
                   all       1126       3806      0.618      0.581      0.601      0.441
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/20      2.37G     0.8942     0.6542     0.9312          7        640: 100%|██████████| 282/282 [01:12<00:00,  3.89it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36 [00:08<00:00,  4.50it/s]
                   all       1126       3806      0.608      0.619      0.615      0.448
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/20      2.37G     0.8829     0.6359     0.9299          7        640: 100%|██████████| 282/282 [01:13<00:00,  3.85it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36 [00:08<00:00,  4.13it/s]
                   all       1126       3806      0.625       0.59      0.616      0.452
20 epochs completed in 0.474 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 6.2MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 6.2MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics 8.3.141  Python-3.11.12 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
Model summary (fused): 72 layers, 3,007,793 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 36/36 [00:10<00:00,  3.49it/s]
                   all       1126       3806      0.624      0.589      0.615      0.451
          pickup_truck        311        434      0.795      0.786      0.844      0.688
                   car        944       2544      0.835      0.882      0.912      0.667
     articulated_truck         70         81      0.655      0.751      0.725      0.542
                   bus        105        121      0.931      0.897      0.938       0.83
     motorized_vehicle        232        303       0.57      0.363      0.408      0.245
              work_van         99        104      0.629      0.654      0.615        0.5
     single_unit_truck         57         60      0.558       0.55      0.518      0.357
            pedestrian         45         92      0.457      0.266      0.331      0.166
               bicycle         22         26      0.481      0.577       0.64      0.408
 non-motorized_vehicle         19         19      0.218      0.105     0.0913     0.0615
            motorcycle         20         22      0.739      0.645      0.742      0.497
Speed: 0.2ms preprocess, 1.7ms inference, 0.0ms loss, 1.8ms postprocess per image
Results saved to runs/detect/train2
```

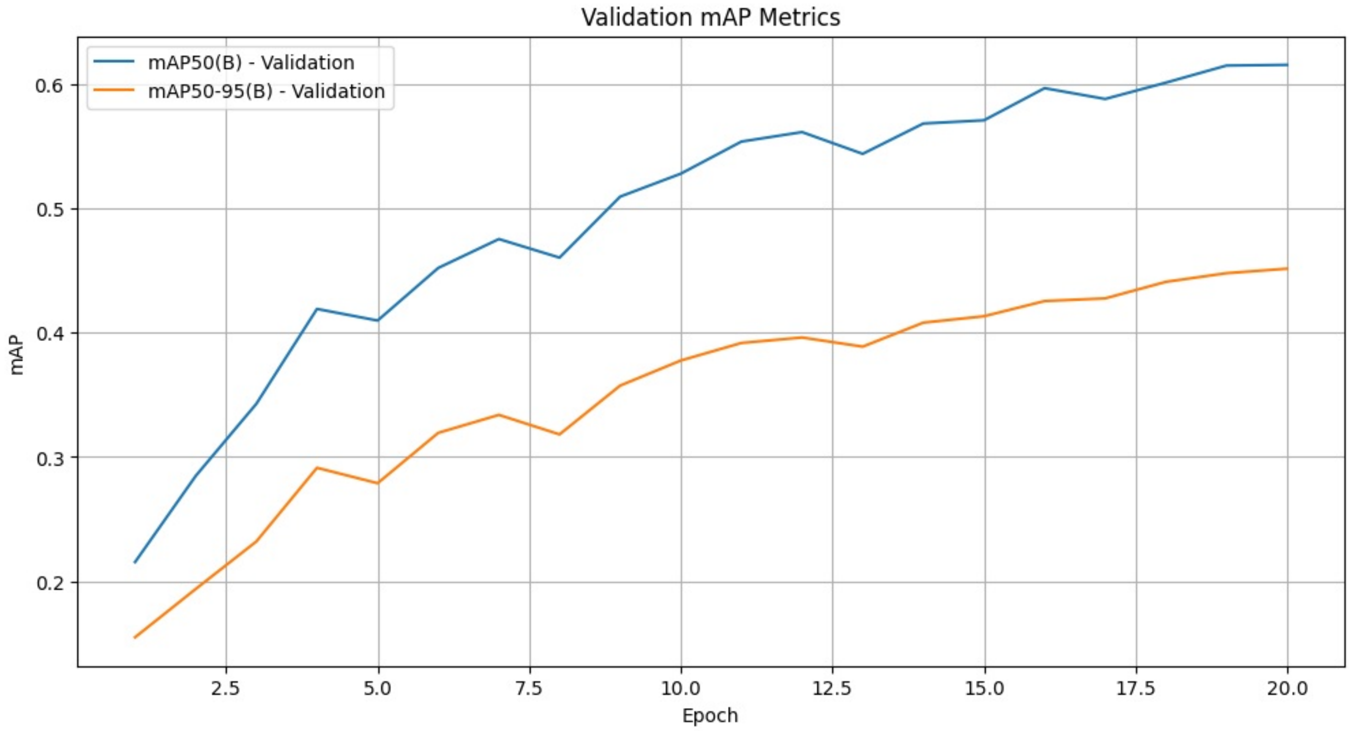
4. Evaluate the model and check the test results

In [ ]:

```python
import matplotlib.pyplot as plt
import pandas as pd


# Replace with the actual path to your results.csv
results_path = './runs/detect/train/results.csv'

# Read the CSV, skipping initial comments (if any)
# You might need to adjust 'skipinitialspace' or 'header' based on your file's exact format
try:
```

```python
    df = pd.read_csv(results_path)
except FileNotFoundError:
    print(f"Error: {results_path} not found. Please check the path.")
    exit()

# Clean column names (YOLOv8 often has leading spaces)
df.columns = [col.strip() for col in df.columns]

# Plotting Loss
plt.figure(figsize=(12, 6))
plt.plot(df['epoch'], df['train/box_loss'] + df['train/cls_loss'] + df['train/dfl_loss'], label='Train Loss (Cor
plt.plot(df['epoch'], df['val/box_loss'] + df['val/cls_loss'] + df['val/dfl_loss'], label='Val Loss (Combined)'
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.show()

# Plotting mAP
plt.figure(figsize=(12, 6))
plt.plot(df['epoch'], df['metrics/mAP50(B)'], label='mAP50(B) - Validation')
plt.plot(df['epoch'], df['metrics/mAP50-95(B)'], label='mAP50-95(B) - Validation')
plt.title('Validation mAP Metrics')
plt.xlabel('Epoch')
plt.ylabel('mAP')
plt.legend()
plt.grid(True)
plt.show()
```





5. Run inferences on sample images and see if vehicles are detected accurately

In [ ]:

```
results_in = model.predict("./sample_data/yolo_dataset/images/val/00004523.jpg", save=True, conf=0.25)
print(results_in)

# Extract first result
r = results_in[0]

# Show class names
print("Class Names:", model.names)

# Print all predictions
for box, conf, cls_id in zip(r.boxes.xyxy, r.boxes.conf, r.boxes.cls):
    print(f"Class: {model.names[int(cls_id)]}, Confidence: {conf:.2f}, Box: {box.tolist()}")
```

```
image 1/1 /content/sample_data/yolo_dataset/images/val/00004523.jpg: 448x640 5 cars, 1 work_van, 121.2ms
Speed: 3.0ms preprocess, 121.2ms inference, 2.2ms postprocess per image at shape (1, 3, 448, 640)
Results saved to runs/detect/train22
[ultralytics.engine.results.Results object with attributes:

boxes: ultralytics.engine.results.Boxes object
keypoints: None
masks: None
names: {0: 'pickup_truck', 1: 'car', 2: 'articulated_truck', 3: 'bus', 4: 'motorized_vehicle', 5: 'work_van', 6:
'single_unit_truck', 7: 'pedestrian', 8: 'bicycle', 9: 'non-motorized_vehicle', 10: 'motorcycle'}
obb: None
orig_img: array([[[ 31,  12,   9],
        [ 37,  18,  13],
        [ 24,   6,   0],
        ...,
        [ 31,  14,  17],
        [ 22,   8,  10],
        [ 24,  10,  12]],

       [[ 18,   0,   0],
        [ 66,  47,  42],
        [ 95,  77,  70],
        ...,
        [ 36,  19,  22],
        [ 19,   5,   7],
        [ 21,   7,   9]],

       [[ 58,  39,  36],
        [166, 147, 142],
        [240, 222, 215],
        ...,
        [ 30,  16,  18],
        [ 13,   0,   1],
        [ 19,   5,   7]],

       ...,

       [[  1,  17,  16],
        [ 26,  42,  41],
        [ 50,  65,  67],
        ...,
        [  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0]],

       [[  3,  19,  18],
        [ 27,  43,  42],
        [ 49,  64,  66],
        ...,
        [  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0]],

       [[  3,  19,  18],
        [ 28,  44,  43],
        [ 49,  64,  66],
        ...,
        [  0,   0,   0],
        [  0,   0,   0],
        [  0,   0,   0]]], dtype=uint8)
orig_shape: (228, 342)
path: '/content/sample_data/yolo_dataset/images/val/00004523.jpg'
probs: None
save_dir: 'runs/detect/train22'
speed: {'preprocess': 2.9612039998028195, 'inference': 121.22001700026885, 'postprocess': 2.164272000300116}]
Class Names: {0: 'pickup_truck', 1: 'car', 2: 'articulated_truck', 3: 'bus', 4: 'motorized_vehicle', 5: 'work_va
n', 6: 'single_unit_truck', 7: 'pedestrian', 8: 'bicycle', 9: 'non-motorized_vehicle', 10: 'motorcycle'}
Class: work_van, Confidence: 0.80, Box: [69.21305847167969, 55.57606506347656, 138.12481689453125, 89.4680099487
3047]
Class: car, Confidence: 0.75, Box: [151.51165771484375, 27.57723617553711, 182.0041046142578, 35.35712432861328]
Class: car, Confidence: 0.63, Box: [69.44784545898438, 55.727657318115234, 138.2427215576172, 89.75243377685547]
Class: car, Confidence: 0.61, Box: [226.2789764404297, 27.015281677246094, 247.7737274169922, 35.183013916015625
]
Class: car, Confidence: 0.58, Box: [110.9125747680664, 24.340269088745117, 129.16390991210938, 31.64859199523925
8]
Class: car, Confidence: 0.34, Box: [27.07554054260254, 43.211185455322266, 45.96173095703125, 55.497798919677734
]
```

In [ ]:

```python
# r.plot() returns an annotated image (numpy array)
annotated_img = r.plot()

# Display it with matplotlib
plt.figure(figsize=(10, 8))
```

```
plt.imshow(annotated_img)
plt.axis('off')
plt.title("YOLOv8 Prediction")
plt.show()
```

YOLOv8 Prediction

Part 2

Data Science

1. Preliminary data inspection and cleaning

   a. Perform preliminary data inspection, checking for data types, missing values, and duplicates b.Remove any columns that might not be relevant for the analysis

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

In [ ]:
```
# Load the dataset
df_full = pd.read_csv("./sample_data/Tesla - Deaths.csv")

df = df_full.iloc[:294, :]
col_maxima = df.tail(10)
print(col_maxima)
#shape
print("Dataset Shape:", df.shape)
# Display basic info
print("Dataset Info:")
print(df.info())

# Show first few rows
print("\nSample Data:")
print(df.head())

# Check for missing values
print("\n Missing Values:")
print(df.isnull().sum())

# Check for duplicates
print("\nDuplicate Rows:")
print(df.duplicated().sum())

df.columns = df.columns.str.strip()
print(" Cleaned column names:")
print(df.columns.tolist())
```

```
# Define columns to drop (you can customize this)
irrelevant_cols = [
     "Note", "Source","Unnamed: 16","Unnamed: 17"
]

# Drop the columns if they exist in the dataset
df = df.drop(columns=[col for col in irrelevant_cols if col in df.columns])

# Verify cleanup
print("\nCleaned DataFrame:")
print(df.columns)

# Identify numeric and object columns
num_cols = df.select_dtypes(include=['int64', 'float64']).columns
str_cols = df.select_dtypes(include=['object']).columns
print(num_cols)
#fill empty columns with NaN
df[df.columns] = df[df.columns].replace(['-', ' - ', ''], np.nan)



df.tail(100)
```

```
     Case #    Year        Date  Country   State   \
284    10.0  2015.0  12/22/2015   Canada        -
285     9.0  2015.0  11/18/2015      USA      CA
286     8.0  2015.0   6/22/2015      USA      CA
287     7.0  2015.0   1/22/2015      USA      CA
288     6.0  2014.0  12/30/2014      USA      CA
289     5.0  2014.0   7/14/2014      USA      CA
290     4.0  2014.0    7/4/2014      USA      CA
291     3.0  2014.0    7/4/2014      USA      CA
292     2.0  2013.0   11/2/2013      USA      CA
293     1.0  2013.0    4/2/2013      USA      CA

                      Description    Deaths   Tesla driver   \
284             Struck by dumptruck       1.0             1
285          Tesla kills pedestrian       1.0             -
286          Tesla drives off cliff       1.0             1
287          Tesla drives off cliff       1.0             1
288          Tesla drives off cliff       1.0             1
289        Tesla kills motorcyclist       1.0             -
290      Thief crashes stolen Tesla       1.0             1
291     Tesla rear ends stopped car       3.0             -
292             Tesla kills cyclist       1.0             -
293   Tesla veers into opposite lane       2.0             -

     Tesla occupant   Other vehicle   ...  Verified Tesla Autopilot Deaths   \
284               -               -   ...                                -
285               -               -   ...                                -
286               -               -   ...                                -
287               -               -   ...                                -
288               -               -   ...                                -
289               -               1   ...                                -
290               -               -   ...                                -
291               -               3   ...                                -
292               -               -   ...                                -
293               -               2   ...                                -

     Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO   \
284                                                  -
285                                                  -
286                                                  -
287                                                  -
288                                                  -
289                                                  -
290                                                  -
291                                                  -
292                                                  -
293                                                  -

                                             Unnamed: 16  \
284   https://web.archive.org/web/20220817120756/ht...
285   https://web.archive.org/web/20220817120754/ht...
286   https://web.archive.org/web/20220817120755/ht...
287   https://web.archive.org/web/20220817120837/ht...
288   https://web.archive.org/web/20220817120806/ht...
289   https://web.archive.org/web/20220817120807/ht...
290   https://web.archive.org/web/20220817120839/ht...
291   https://web.archive.org/web/20220412004559/ht...
292   https://web.archive.org/web/20220817121049/ht...
293   https://web.archive.org/web/20150425055520/ht...

                                             Unnamed: 17  \
```

```
284   https://web.archive.org/web/20220817120756/ht...
285   https://web.archive.org/web/20220817120754/ht...
286   https://web.archive.org/web/20220817120755/ht...
287   https://web.archive.org/web/20220817120837/ht...
288   https://web.archive.org/web/20220817120806/ht...
289   https://web.archive.org/web/20220817120807/ht...
290   https://web.archive.org/web/20220817120839/ht...
291   https://web.archive.org/web/20220412004559/ht...
292   https://web.archive.org/web/20220817121049/ht...
293   https://web.archive.org/web/20150425055520/ht...

                                          Source   Note   \
284   https://web.archive.org/web/20220817120756/ht...    NaN
285   https://web.archive.org/web/20220817120754/ht...    NaN
286   https://web.archive.org/web/20220817120755/ht...    NaN
287   https://web.archive.org/web/20220817120837/ht...    NaN
288   https://web.archive.org/web/20220817120806/ht...    NaN
289   https://web.archive.org/web/20220817120807/ht...    NaN
290   https://web.archive.org/web/20220817120839/ht...    NaN
291   https://web.archive.org/web/20220412004559/ht...    NaN
292   https://web.archive.org/web/20220817121049/ht...    NaN
293   https://web.archive.org/web/20150425055520/ht...    NaN

                   Deceased 1                Deceased 2   Deceased 3   \
284                       NaN                       NaN          NaN
285                       NaN                       NaN          NaN
286             Tim Devine                          NaN          NaN
287            Peter Kleis                          NaN          NaN
288   Louis Francis Thoelecke                       NaN          NaN
289                       NaN                       NaN          NaN
290            Joshua Slot                          NaN          NaN
291                       NaN                       NaN          NaN
292                       NaN                       NaN          NaN
293   Alberto Casique-Salinas    Armando Garcia-Gonzales           NaN

     Deceased 4
284         NaN
285         NaN
286         NaN
287         NaN
288         NaN
289         NaN
290         NaN
291         NaN
292         NaN
293         NaN

[10 rows x 24 columns]
Dataset Shape: (294, 24)
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294 entries, 0 to 293
Data columns (total 24 columns):
 #   Column                                                            Non-Null Count  Dtype
---  ------                                                            --------------  -----
 0   Case #                                                            294 non-null    float64
 1   Year                                                              294 non-null    float64
 2   Date                                                              294 non-null    object
 3    Country                                                          294 non-null    object
 4    State                                                            294 non-null    object
 5    Description                                                      294 non-null    object
 6    Deaths                                                           294 non-null    float64
 7    Tesla driver                                                     289 non-null    object
 8    Tesla occupant                                                   285 non-null    object
 9    Other vehicle                                                    290 non-null    object
 10   Cyclists/ Peds                                                   291 non-null    object
 11   TSLA+cycl / peds                                                 292 non-null    object
 12   Model                                                            294 non-null    object
 13   Autopilot claimed                                                276 non-null    object
 14   Verified Tesla Autopilot Deaths                                  290 non-null    object
 15   Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO   293 non-null    object
 16  Unnamed: 16                                                       289 non-null    object
 17  Unnamed: 17                                                       289 non-null    object
 18   Source                                                           294 non-null    object
 19   Note                                                             9 non-null      object
 20   Deceased 1                                                       87 non-null     object
 21   Deceased 2                                                       17 non-null     object
 22   Deceased 3                                                       4 non-null      object
 23   Deceased 4                                                       0 non-null      float64
dtypes: float64(4), object(20)
memory usage: 55.3+ KB
None
```

```
Sample Data:
   Case #    Year        Date  Country   State   \
0   294.0  2022.0   1/17/2023      USA       CA
1   293.0  2022.0    1/7/2023   Canada        -
2   292.0  2022.0    1/7/2023      USA       WA
3   291.0  2022.0  12/22/2022      USA       GA
4   290.0  2022.0  12/19/2022   Canada        -

                         Description    Deaths   Tesla driver   \
0    Tesla crashes into back of semi       1.0             1
1                      Tesla crashes       1.0             1
2   Tesla hits pole, catches on fire       1.0             -
3             Tesla crashes and burns      1.0             1
4      Tesla crashes into storefront       1.0             -

   Tesla occupant   Other vehicle   ...  Verified Tesla Autopilot Deaths   \
0               -               -   ...                                -
1               -               -   ...                                -
2               1               -   ...                                -
3               -               -   ...                                -
4               -               -   ...                                -

   Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO   \
0                                                  -
1                                                  -
2                                                  -
3                                                  -
4                                                  -

                                         Unnamed: 16  \
0   https://web.archive.org/web/20221222203930/ht...
1   https://web.archive.org/web/20221222203930/ht...
2   https://web.archive.org/web/20221222203930/ht...
3   https://web.archive.org/web/20221222203930/ht...
4   https://web.archive.org/web/20221223203725/ht...

                                         Unnamed: 17  \
0   https://web.archive.org/web/20221222203930/ht...
1   https://web.archive.org/web/20221222203930/ht...
2   https://web.archive.org/web/20221222203930/ht...
3   https://web.archive.org/web/20221222203930/ht...
4   https://web.archive.org/web/20221223203725/ht...

                                             Source   Note   \
0   https://web.archive.org/web/20230118162813/ht...    NaN
1   https://web.archive.org/web/20230109041434/ht...    NaN
2   https://web.archive.org/web/20230107232745/ht...    NaN
3   https://web.archive.org/web/20221222203930/ht...    NaN
4   https://web.archive.org/web/20221223203725/ht...    NaN

         Deceased 1   Deceased 2   Deceased 3   Deceased 4
0                NaN          NaN          NaN          NaN
1   Taren Singh Lal          NaN          NaN          NaN
2                NaN          NaN          NaN          NaN
3                NaN          NaN          NaN          NaN
4                NaN          NaN          NaN          NaN

[5 rows x 24 columns]

 Missing Values:
Case #                                                                  0
Year                                                                    0
Date                                                                    0
 Country                                                                0
 State                                                                  0
 Description                                                            0
 Deaths                                                                 0
 Tesla driver                                                           5
 Tesla occupant                                                         9
 Other vehicle                                                          4
 Cyclists/ Peds                                                         3
 TSLA+cycl / peds                                                       2
 Model                                                                  0
 Autopilot claimed                                                     18
 Verified Tesla Autopilot Deaths                                        4
 Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO     1
Unnamed: 16                                                             5
Unnamed: 17                                                             5
 Source                                                                 0
 Note                                                                 285
 Deceased 1                                                           207
 Deceased 2                                                           277
 Deceased 3                                                           290
```

```
 Deceased 4                                                             294
dtype: int64

Duplicate Rows:
0
 Cleaned column names:
['Case #', 'Year', 'Date', 'Country', 'State', 'Description', 'Deaths', 'Tesla driver', 'Tesla occupant', 'Other vehicle', 'Cyclists/ Peds', 'TSLA+cycl / peds', 'Model', 'Autopilot claimed', 'Verified Tesla Autopilot Deaths', 'Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO', 'Unnamed: 16', 'Unnamed: 17', 'Source', 'Note', 'Deceased 1', 'Deceased 2', 'Deceased 3', 'Deceased 4']

Cleaned DataFrame:
Index(['Case #', 'Year', 'Date', 'Country', 'State', 'Description', 'Deaths',
       'Tesla driver', 'Tesla occupant', 'Other vehicle', 'Cyclists/ Peds',
       'TSLA+cycl / peds', 'Model', 'Autopilot claimed',
       'Verified Tesla Autopilot Deaths',
       'Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO',
       'Deceased 1', 'Deceased 2', 'Deceased 3', 'Deceased 4'],
      dtype='object')
Index(['Case #', 'Year', 'Deaths', 'Deceased 4'], dtype='object')
```

Out[ ]:

	Case #	Year	Date	Country	State	Description	Deaths	Tesla driver	Tesla occupant	Other vehicle	Cyclists/ Peds	TSLA+cycl / peds	Model	Autop clain
194	100.0	2019.0	12/29/2019	USA	CA	Tesla runs red light after exiting freeway	2.0	NaN	NaN	2	NaN	NaN	S	
195	99.0	2019.0	12/23/2019	Slovenia	NaN	Drunk driver strikes Tesla	1.0	NaN	NaN	1	NaN	NaN	S	N
196	98.0	2019.0	12/19/2019	USA	FL	Police SUV strikes Tesla	2.0	1	1	NaN	NaN	2	NaN	N
197	97.0	2019.0	12/11/2019	USA	CA	Tesla strikes turning Lexus	1.0	NaN	NaN	1	NaN	NaN	X	N
198	96.0	2019.0	12/10/2019	USA	NV	Kia rear ends Tesla	1.0	NaN	NaN	1	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
289	5.0	2014.0	7/14/2014	USA	CA	Tesla kills motorcyclist	1.0	NaN	NaN	1	NaN	NaN	NaN	N
290	4.0	2014.0	7/4/2014	USA	CA	Thief crashes stolen Tesla	1.0	1	NaN	NaN	NaN	1	NaN	N
291	3.0	2014.0	7/4/2014	USA	CA	Tesla rear ends stopped car	3.0	NaN	NaN	3	NaN	NaN	NaN	N
292	2.0	2013.0	11/2/2013	USA	CA	Tesla kills cyclist	1.0	NaN	NaN	NaN	1	1	NaN	N
293	1.0	2013.0	4/2/2013	USA	CA	Tesla veers into opposite lane	2.0	NaN	NaN	2	NaN	NaN	S	N

100 rows × 20 columns

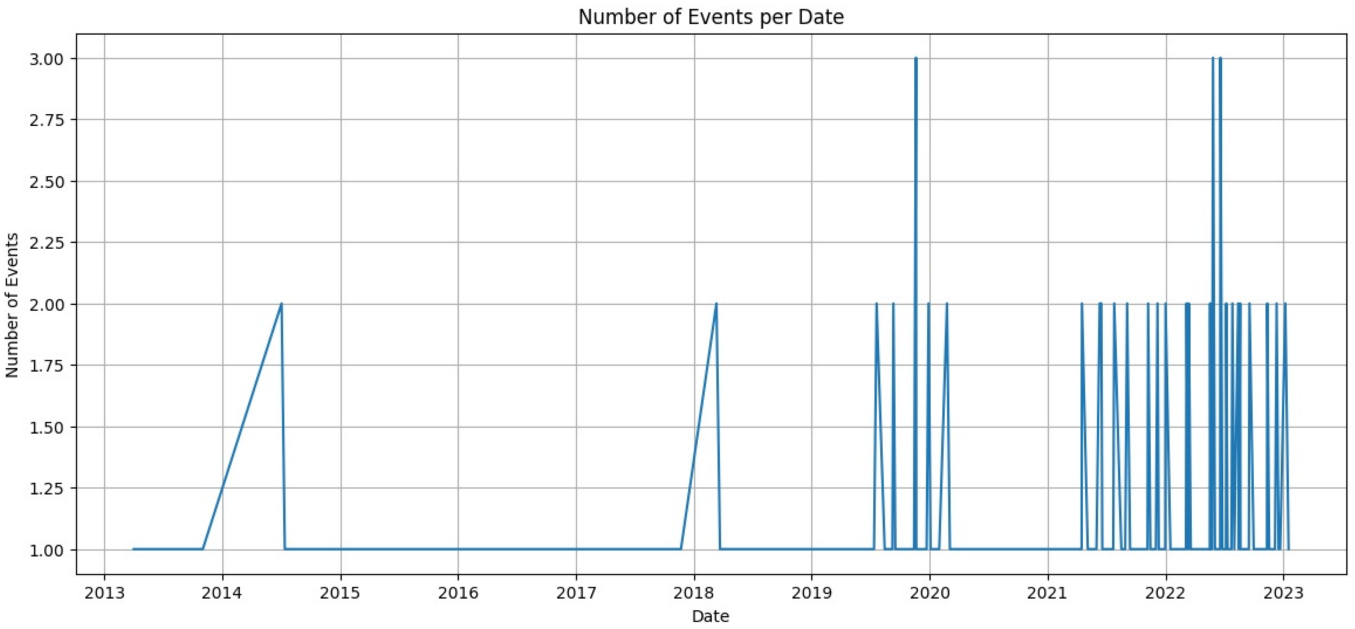
Exploratory Data Analysis

a. Perform an in-depth exploratory data analysis on the number of events by date, per year, and per day for each state and country

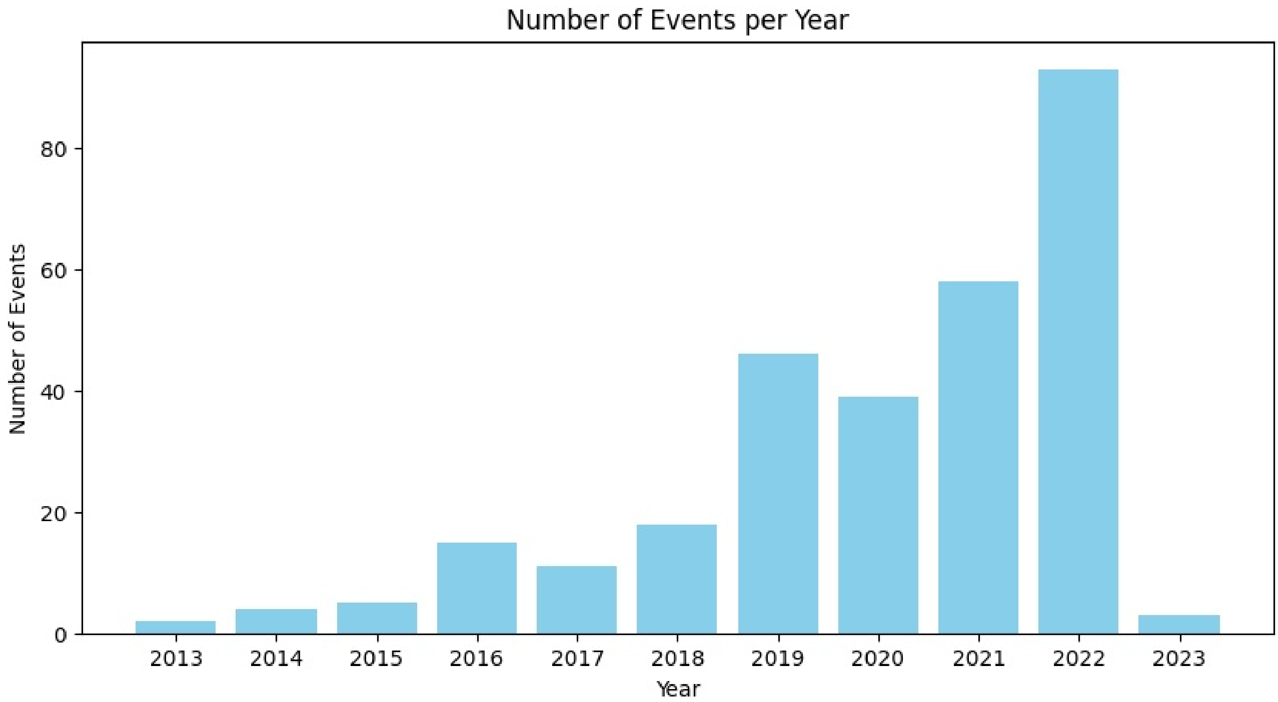
In [ ]:
```
df['Date'] = pd.to_datetime(df['Date'], errors='coerce')  # Convert Date to datetime, coerce errors
```

In [ ]:
```
events_per_date = df.groupby('Date').size().reset_index(name='count')


plt.figure(figsize=(14,6))
plt.plot(events_per_date['Date'], events_per_date['count'])
plt.title('Number of Events per Date')
```

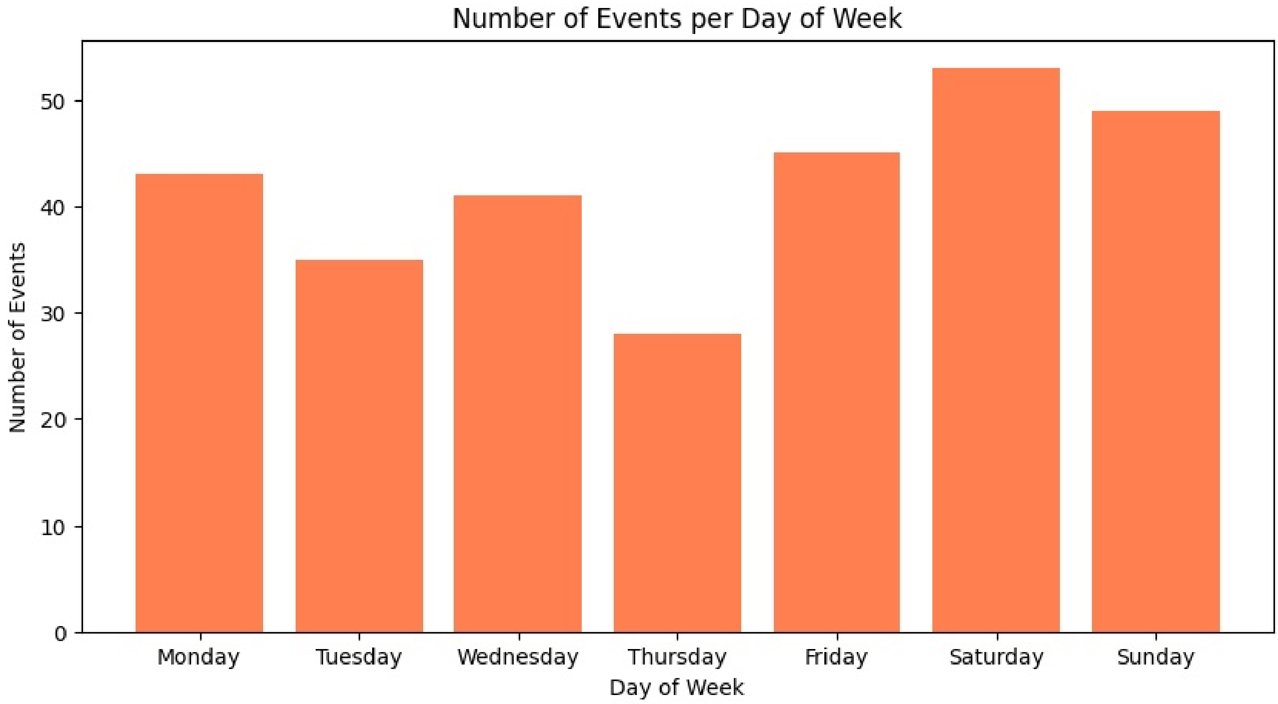
```
plt.xlabel('Date')
plt.ylabel('Number of Events')
plt.grid(True)
plt.show()
```



In [63]:
```
# Events count per year
df['Year'] = df['Date'].dt.year
events_per_year = df.groupby('Year').size().reset_index(name='count')


plt.figure(figsize=(10,5))
plt.bar(events_per_year['Year'], events_per_year['count'], color='skyblue')
plt.title('Number of Events per Year')
plt.xlabel('Year')
plt.ylabel('Number of Events')
plt.xticks(events_per_year['Year'])
plt.show()
```

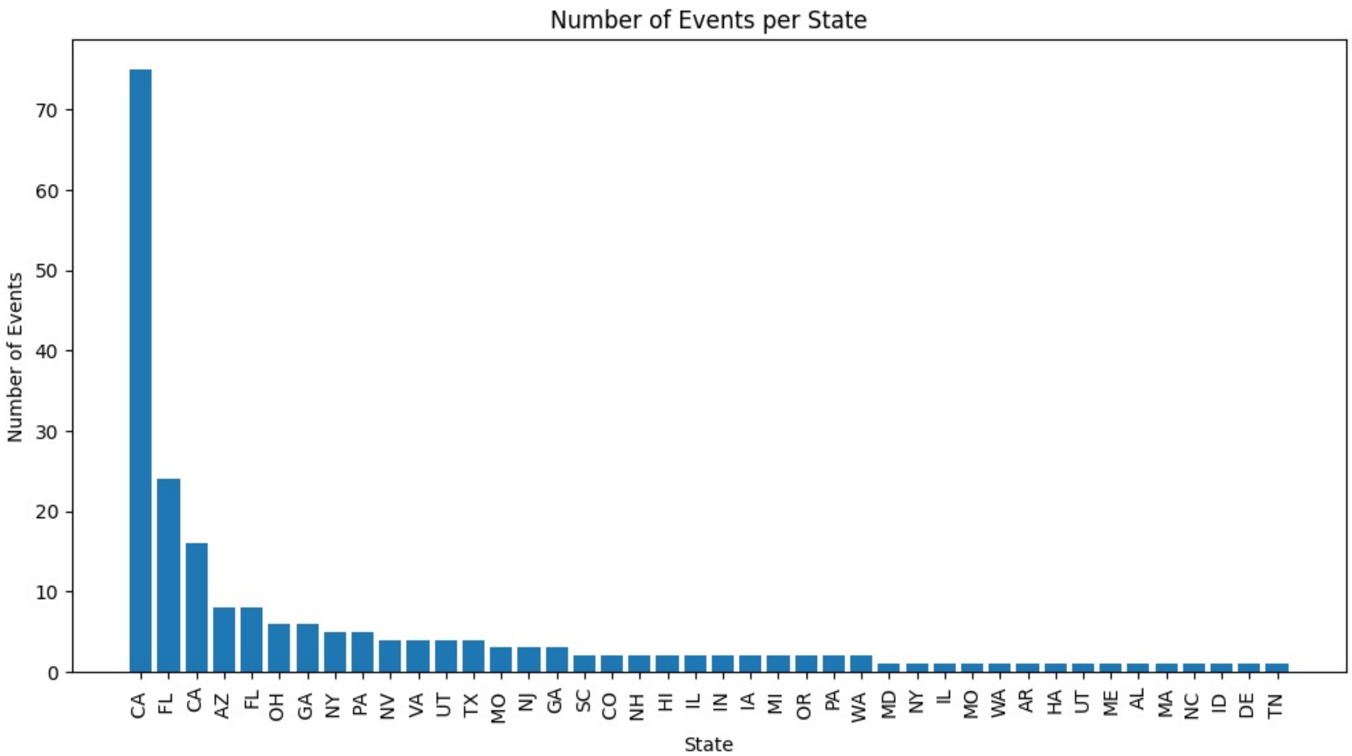


In [ ]:
```
#Events count per day of week
df['DayOfWeek'] = df['Date'].dt.day_name()
events_per_day = df.groupby('DayOfWeek').size().reindex([
    'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'
]).reset_index(name='count')

plt.figure(figsize=(10,5))
plt.bar(events_per_day['DayOfWeek'], events_per_day['count'], color='coral')
plt.title('Number of Events per Day of Week')
plt.xlabel('Day of Week')
plt.ylabel('Number of Events')
plt.show()
```



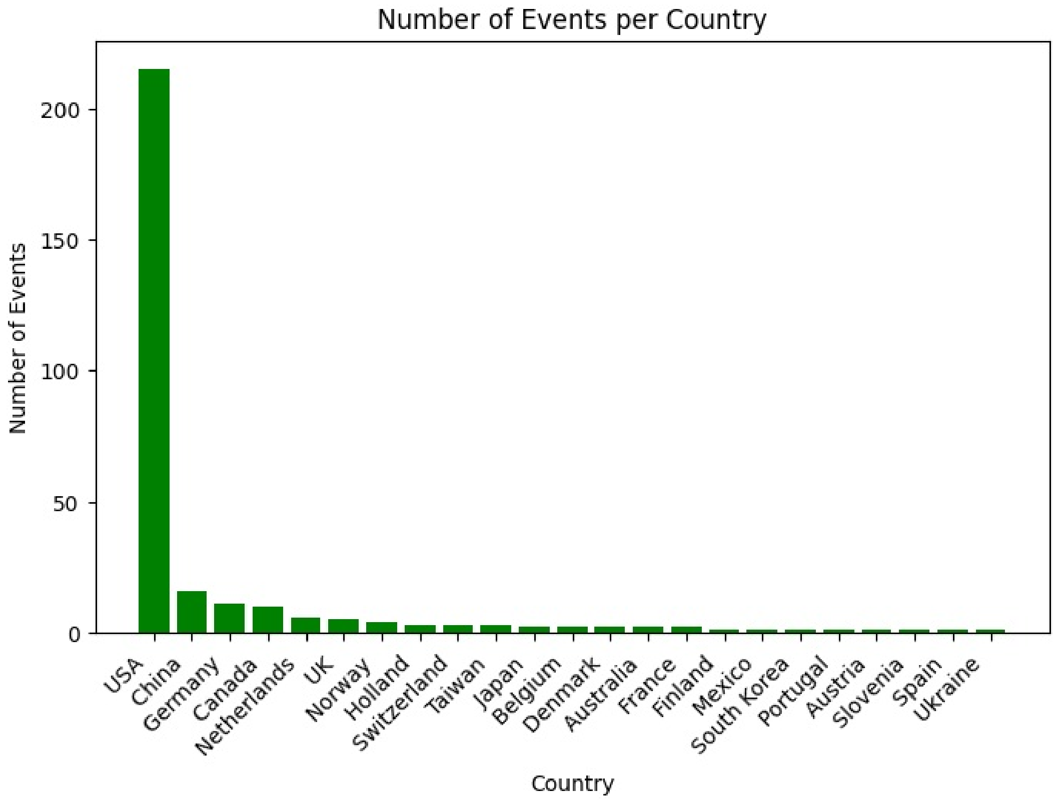
In [ ]:

```
#Events count per State
events_per_state = df['State'].value_counts().reset_index()
events_per_state.columns = ['State', 'count']

plt.figure(figsize=(12,6))
plt.bar(events_per_state['State'], events_per_state['count'])
plt.title('Number of Events per State')
plt.xlabel('State')
plt.ylabel('Number of Events')
plt.xticks(rotation=90)
plt.show()
```



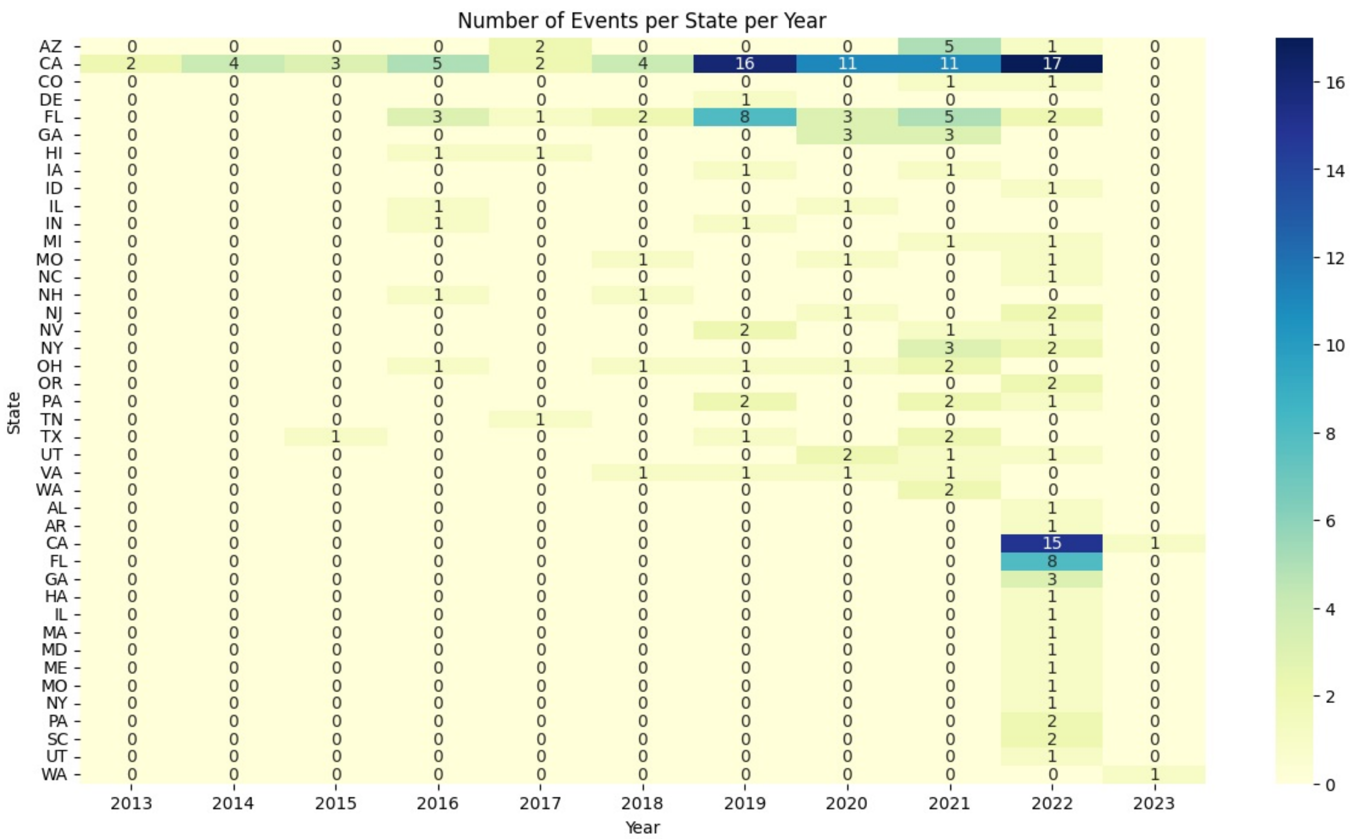
In [ ]:

```
#Events count per Country
events_per_country = df['Country'].value_counts().reset_index()
events_per_country.columns = ['Country', 'count']

plt.figure(figsize=(8,5))
plt.bar(events_per_country['Country'], events_per_country['count'], color='green')
plt.title('Number of Events per Country')
plt.xlabel('Country')
plt.ylabel('Number of Events')
plt.xticks(rotation=45, ha='right')
plt.show()
```



In [ ]:

```
#Heatmap of events per State and Year


state_year = df.groupby(['State', 'Year']).size().unstack(fill_value=0)

plt.figure(figsize=(15,8))
sns.heatmap(state_year, annot=True, fmt='d', cmap='YlGnBu')
plt.title('Number of Events per State per Year')
plt.xlabel('Year')
plt.ylabel('State')
plt.show()
```



b.Analyze the different aspects of the death events. For example:

- What is the number of victims (deaths) in each accident?

In [ ]:
```
# Show accident ID and deaths
if 'Case #' in df.columns:
    print(df[['Case #', 'Deaths']])
else:
    print(df[['Deaths']])
```

```
     Case #  Deaths
0     294.0     1.0
1     293.0     1.0
2     292.0     1.0
3     291.0     1.0
4     290.0     1.0
..      ...     ...
289     5.0     1.0
290     4.0     1.0
291     3.0     3.0
292     2.0     1.0
293     1.0     2.0

[294 rows x 2 columns]
```

In [ ]:
```
print("Deaths per accident statistics:")
print(df['Deaths'].describe())
df['Deaths'] = pd.to_numeric(df['Deaths'], errors='coerce')
# Count how many accidents with zero deaths vs non-zero deaths
print("\nAccidents with zero deaths:", (df['Deaths'] == 0).sum())
print("Accidents with one or more deaths:", (df['Deaths'] >=1).sum())
```

```
Deaths per accident statistics:
count    294.000000
mean       1.200680
std        0.513171
min        1.000000
25%        1.000000
50%        1.000000
75%        1.000000
max        4.000000
Name: Deaths, dtype: float64

Accidents with zero deaths: 0
Accidents with one or more deaths: 294
```

- How many times did tesla drivers die ?

In [ ]:
```
# Count number of deaths
tesla_driver_deaths = (df['Verified Tesla Autopilot Deaths'].dropna().astype(int) >= 1).sum()

print(f"Tesla drivers died in {tesla_driver_deaths} accident(s).")
```

```
Tesla drivers died in 16 accident(s).
```

- What is the proportion of events in which one or more occupants died ?

In [ ]:
```
# Ensure 'Deaths' is numeric
df['Deaths'] = pd.to_numeric(df['Deaths'], errors='coerce')

# Total number of events
total_events = len(df)

# Events with at least one death
fatal_events = (df['Deaths'] >= 1).sum()

# Proportion calculation
proportion = fatal_events / total_events

print(f"Total Events: {total_events}")
print(f"Events with ≥1 Death: {fatal_events}")
print(f"Proportion: {proportion:.2%}")
```

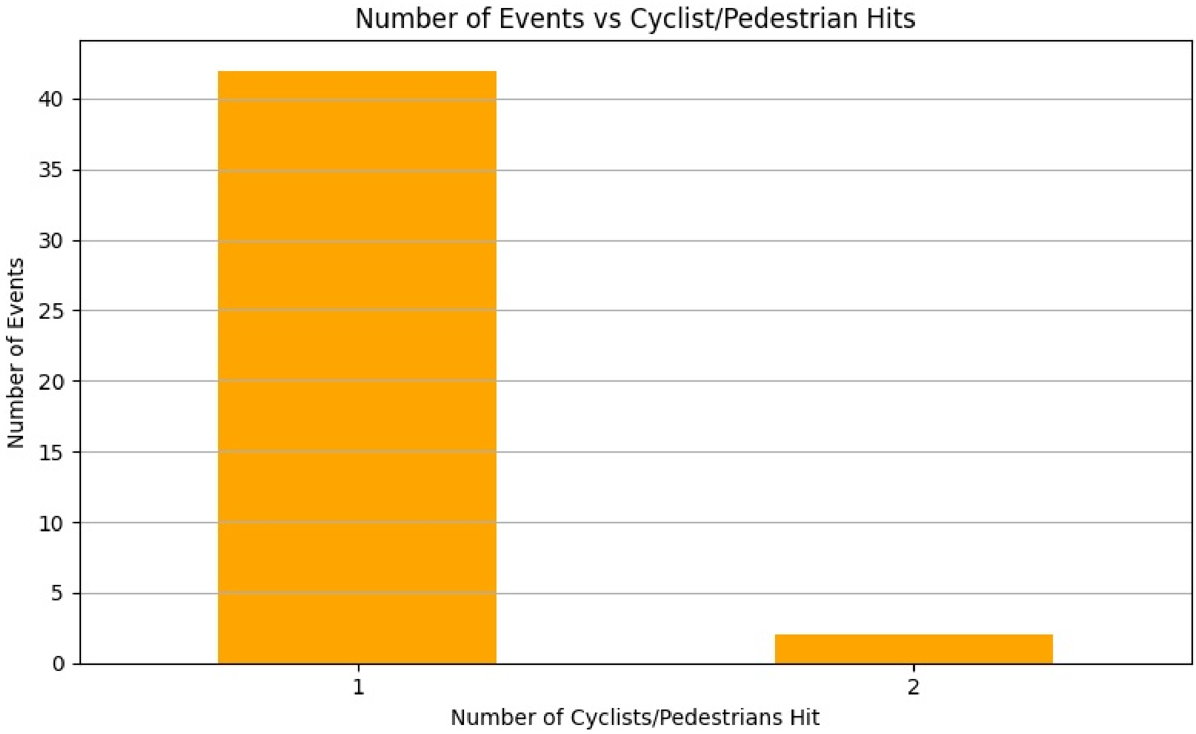
```
Total Events: 294
Events with ≥1 Death: 294
Proportion: 100.00%
```

- What is the distribution of events in which the vehicle hit a cyclist or a pedestrian ?

In [ ]:
```
df['Cyclists/ Peds'] = pd.to_numeric(df['Cyclists/ Peds'], errors='coerce')

print("Distribution of cyclist/pedestrian hits:")
print(df['Cyclists/ Peds'].value_counts().sort_index())
```

```
plt.figure(figsize=(8,5))
df['Cyclists/ Peds'].dropna().astype(int).value_counts().sort_index().plot(kind='bar', color='orange')
plt.title('Number of Events vs Cyclist/Pedestrian Hits')
plt.xlabel('Number of Cyclists/Pedestrians Hit')
plt.ylabel('Number of Events')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

```
Distribution of cyclist/pedestrian hits:
Cyclists/ Peds
1.0    42
2.0     2
Name: count, dtype: int64
```



- How many times did the accident involve the death of an occupant or driver of a Tesla along with a cyclist or pedestrian?

In [ ]:

```
# Convert relevant columns
print(df.columns)
df['Cyclists/ Peds'] = pd.to_numeric(df['Cyclists/ Peds'], errors='coerce')
df['Tesla driver'] = pd.to_numeric(df['Tesla driver'], errors='coerce')
df['Tesla occupant'] = pd.to_numeric(df['Tesla occupant'], errors='coerce')

tesla_death = (
    (df['Tesla driver'] >= 1) |
    (df['Tesla occupant'] >= 1)
)
cycl_peds_hit = df['Cyclists/ Peds'] >= 1
combined_condition = tesla_death & cycl_peds_hit
count = combined_condition.sum()

print(f"Number of accidents involving Tesla driver/occupant death AND cyclist/pedestrian: {count}")
```

```
Index(['Case #', 'Year', 'Date', 'Country', 'State', 'Description', 'Deaths',
       'Tesla driver', 'Tesla occupant', 'Other vehicle', 'Cyclists/ Peds',
       'TSLA+cycl / peds', 'Model', 'Autopilot claimed',
       'Verified Tesla Autopilot Deaths',
       'Verified Tesla Autopilot Deaths + All Deaths Reported to NHTSA SGO',
       'Deceased 1', 'Deceased 2', 'Deceased 3', 'Deceased 4', 'DayOfWeek'],
      dtype='object')
Number of accidents involving Tesla driver/occupant death AND cyclist/pedestrian: 1
```

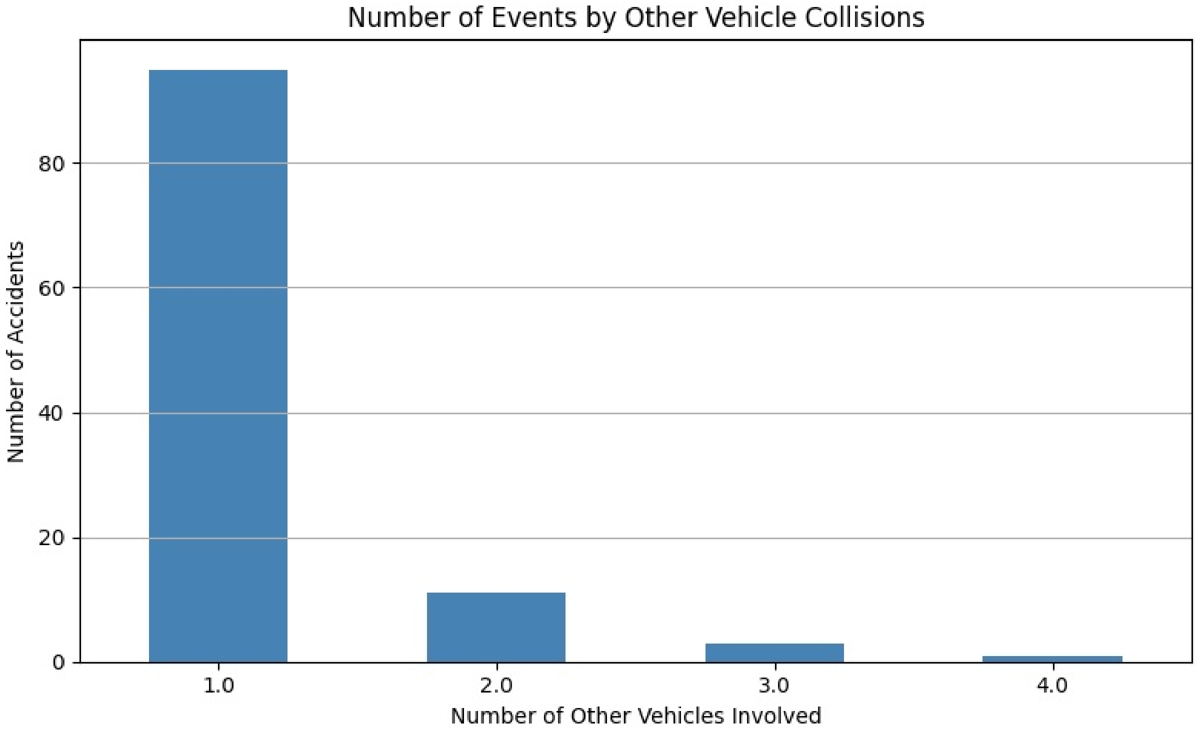
- What is the frequency of Tesla colliding with other vehicles ?

In [ ]:

```
df['Other vehicle'] = pd.to_numeric(df['Other vehicle'], errors='coerce')
print("Frequency of Tesla colliding with other vehicles:")
count_other_vehicles=df['Other vehicle'].value_counts().sort_index()
print(count_other_vehicles)
```

```
plt.figure(figsize=(8, 5))
count_other_vehicles[count_other_vehicles>=1].plot(kind='bar', color='steelblue')
plt.title('Number of Events by Other Vehicle Collisions')
plt.xlabel('Number of Other Vehicles Involved')
plt.ylabel('Number of Accidents')
plt.xticks(rotation=0)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

```
Frequency of Tesla colliding with other vehicles:
Other vehicle
1.0    95
2.0    11
3.0     3
4.0     1
Name: count, dtype: int64
```

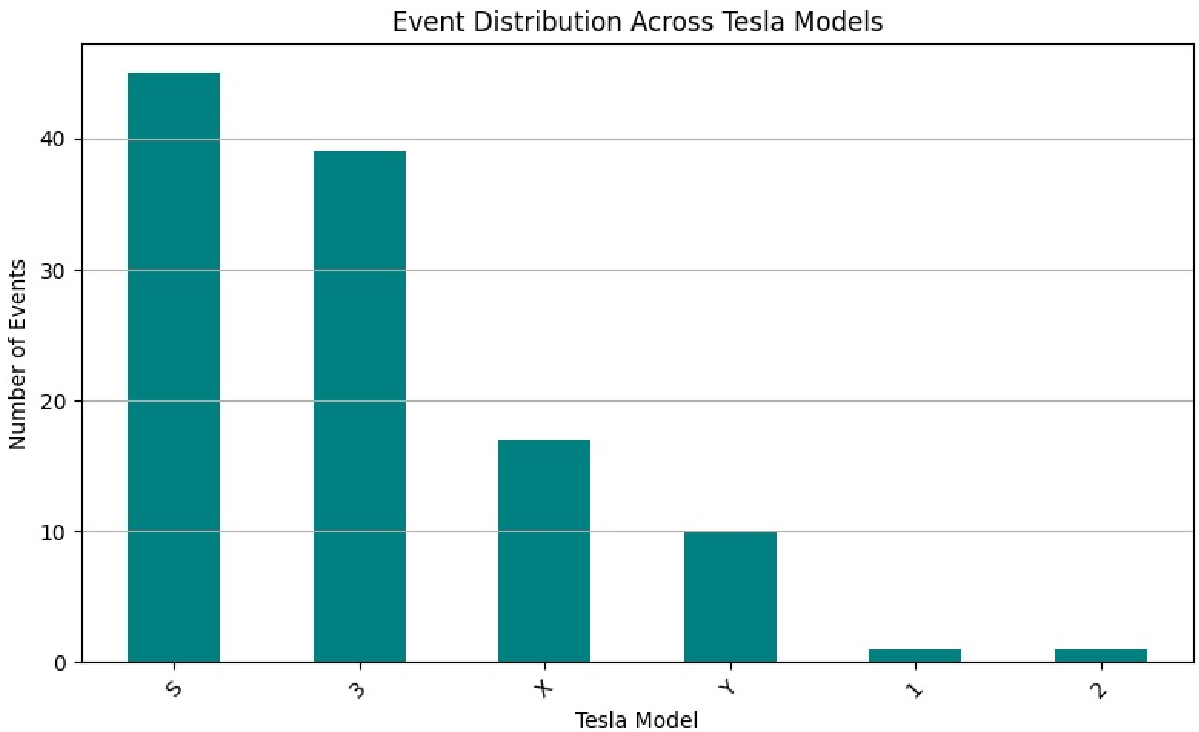


c. Study the event distribution across models

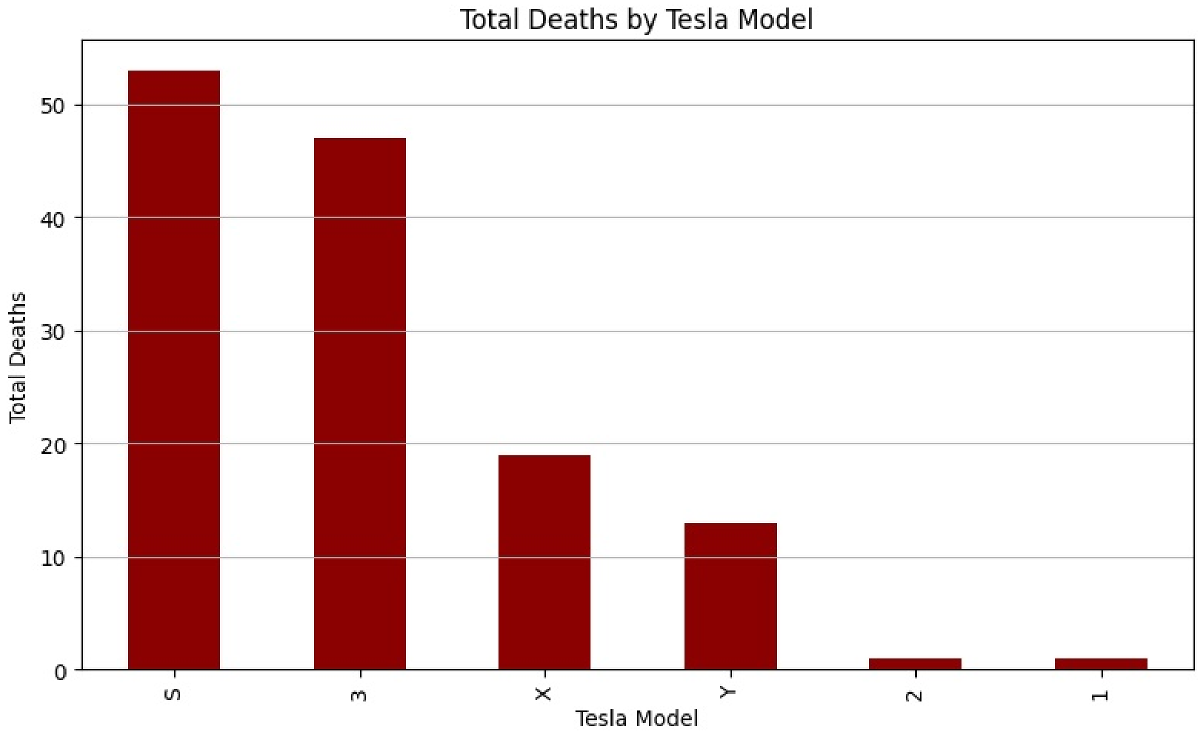
In [ ]:
```
# Standardize formatting
df['Model'] = df['Model'].str.strip().str.title()

# View unique models
print("Unique Tesla Models:")
print(df['Model'].unique())

model_counts = df['Model'].value_counts()

print("Event distribution by Tesla model:")
print(model_counts)



plt.figure(figsize=(8, 5))
model_counts.plot(kind='bar', color='teal')
plt.title('Event Distribution Across Tesla Models')
plt.xlabel('Tesla Model')
plt.ylabel('Number of Events')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()

model_percentage = model_counts / model_counts.sum() * 100
print("Percentage distribution by model:")
print(model_percentage.round(2))


# Ensure 'Deaths' is numeric
df['Deaths'] = pd.to_numeric(df['Deaths'], errors='coerce')

# Total deaths per model
deaths_per_model = df.groupby('Model')['Deaths'].sum().sort_values(ascending=False)
```

```
print("Total deaths per Tesla model:")
print(deaths_per_model)

deaths_per_model.plot(kind='bar', color='darkred', figsize=(8, 5))
plt.title('Total Deaths by Tesla Model')
plt.xlabel('Tesla Model')
plt.ylabel('Total Deaths')
plt.grid(axis='y')
plt.tight_layout()
plt.show()

df['Autopilot claimed'] = df['Autopilot claimed'].str.strip().str.title()

autopilot_claimed_model = df.groupby(['Model', 'Autopilot claimed']).size().unstack(fill_value=0)

print("Autopilot claimed status by Tesla model:")
print(autopilot_claimed_model)


autopilot_claimed_model.plot(kind='bar', stacked=True, figsize=(9, 6), colormap='Paired')
plt.title('Autopilot Claim Distribution by Tesla Model')
plt.xlabel('Tesla Model')
plt.ylabel('Number of Events')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend(title='Autopilot Claimed')
plt.grid(axis='y')
plt.show()

df['Cyclists/ Peds'] = pd.to_numeric(df['Cyclists/ Peds'], errors='coerce')
cycl_peds_per_model = df.groupby('Model')['Cyclists/ Peds'].sum().sort_values(ascending=False)

print("Cyclist/Pedestrian hits per Tesla model:")
print(cycl_peds_per_model)
cycl_peds_per_model.plot(kind='bar', color='blue', figsize=(8, 5))
plt.title('Cyclist/Pedestrian hits per Tesla model')
plt.xlabel('Tesla Model')
plt.ylabel('Total Cyclist/Pedestrian hits')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

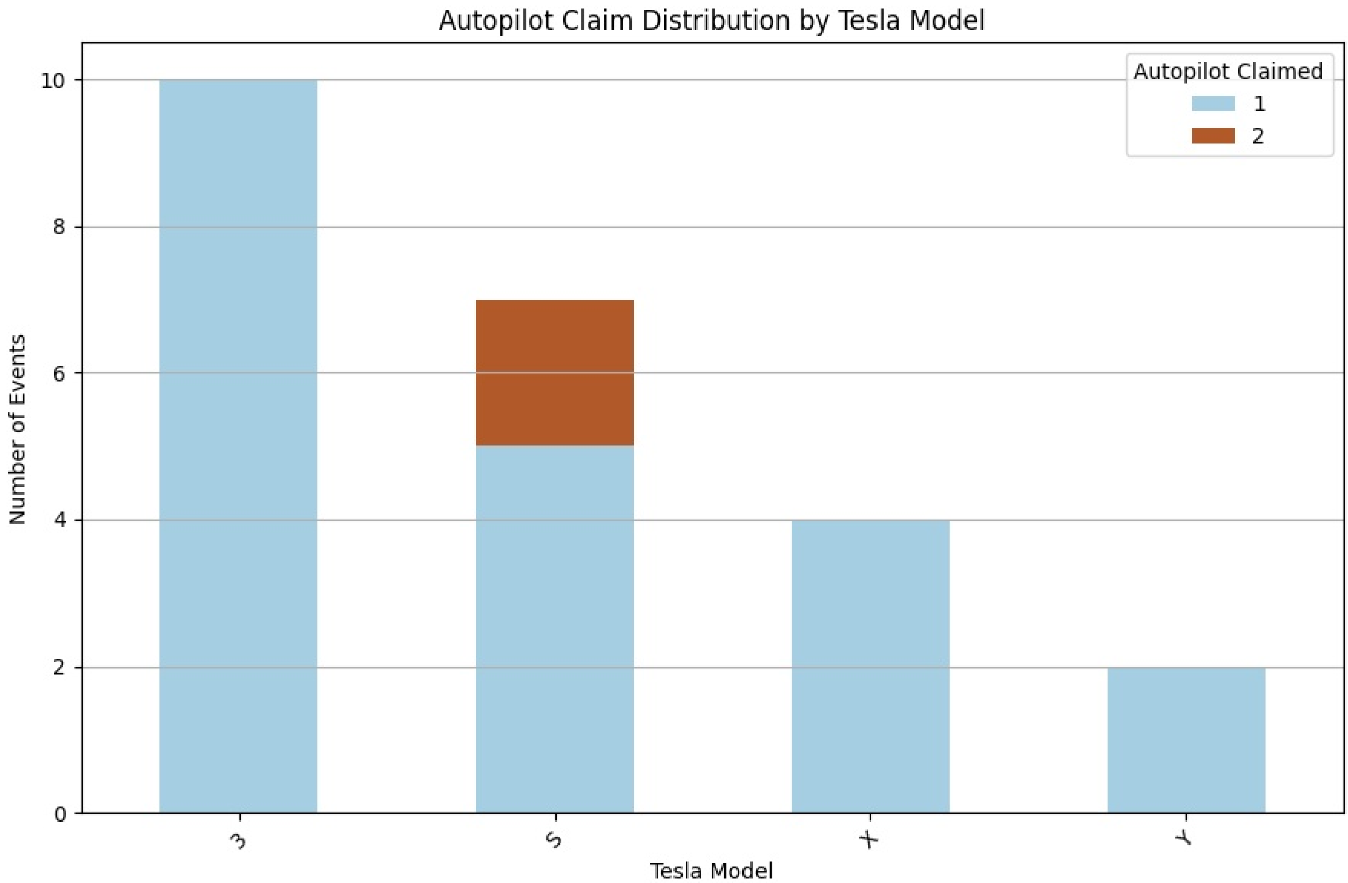
```
Unique Tesla Models:
[nan 'Y' '1' '2' '3' 'S' 'X']
Event distribution by Tesla model:
Model
S    45
3    39
X    17
Y    10
1     1
2     1
Name: count, dtype: int64
```



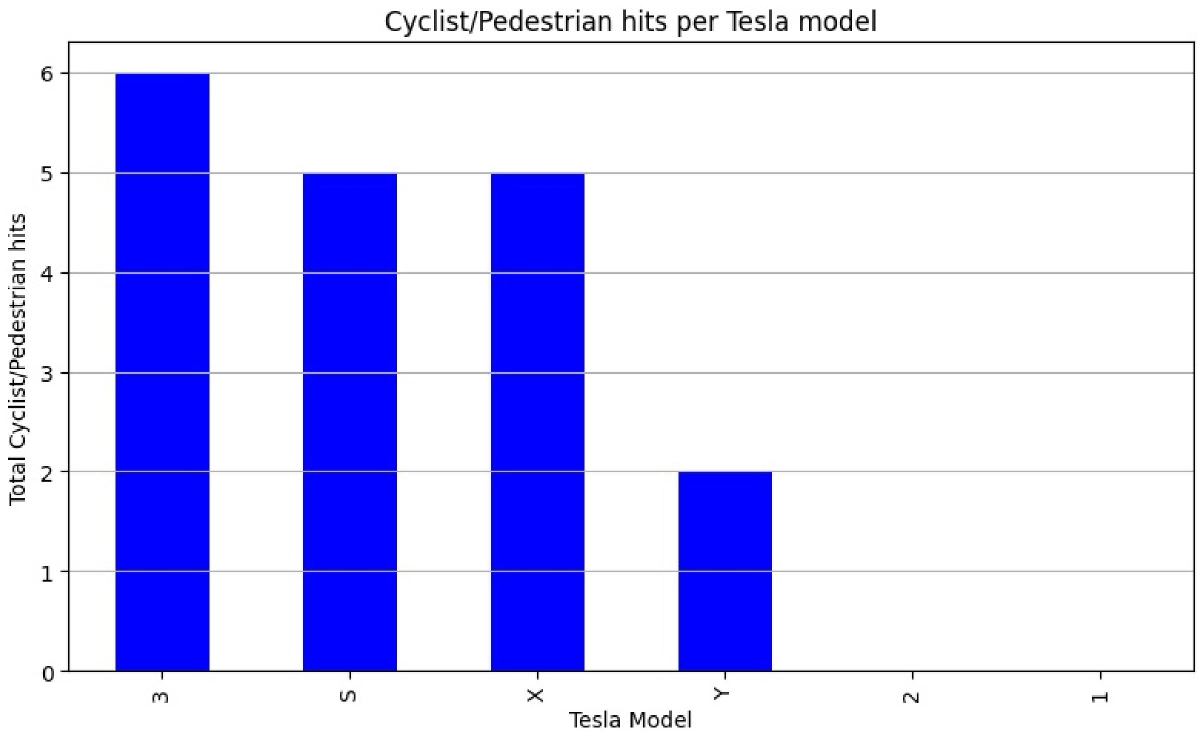
```
Percentage distribution by model:
Model
S    39.82
3    34.51
X    15.04
Y     8.85
1     0.88
2     0.88
Name: count, dtype: float64
Total deaths per Tesla model:
Model
S    53.0
3    47.0
X    19.0
Y    13.0
2     1.0
1     1.0
Name: Deaths, dtype: float64
```




```
Autopilot claimed status by Tesla model:
Autopilot claimed   1  2
Model
3                  10  0
S                   5  2
X                   4  0
Y                   2  0
```




```
Cyclist/Pedestrian hits per Tesla model:
Model
3    6.0
S    5.0
X    5.0
Y    2.0
2    0.0
1    0.0
Name: Cyclists/ Peds, dtype: float64
```



d.Check the distribution of verified Tesla autopilot deaths

```
# Ensure it's numeric
df['Verified Tesla Autopilot Deaths'] = pd.to_numeric(df['Verified Tesla Autopilot Deaths'], errors='coerce')
print("Summary of Verified Tesla Autopilot Deaths:")
print(df['Verified Tesla Autopilot Deaths'].describe())
# Count of how many events had 0, 1, 2,... verified deaths
print(df['Verified Tesla Autopilot Deaths'].value_counts().sort_index())


plt.figure(figsize=(7, 5))
df['Verified Tesla Autopilot Deaths'].dropna().astype(int).plot.hist(bins=range(0, int(df['Verified Tesla Autop
plt.title('Distribution of Verified Tesla Autopilot Deaths')
plt.xlabel('Number of Verified Autopilot Deaths')
plt.ylabel('Number of Events')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
# Assuming 'Year' is available
df['Year'] = pd.to_numeric(df['Year'], errors='coerce')

autopilot_deaths_yearly = df.groupby('Year')['Verified Tesla Autopilot Deaths'].sum()

autopilot_deaths_yearly.plot(kind='line', marker='o', figsize=(8, 5), color='crimson')
plt.title('Yearly Verified Tesla Autopilot Deaths')
plt.xlabel('Year')
plt.ylabel('Number of Verified Deaths')
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Summary of Verified Tesla Autopilot Deaths:
count      23.000000
mean      186.652174
std       579.648295
min         1.000000
25%         1.000000
50%         1.000000
75%         9.500000
max      2022.000000
Name: Verified Tesla Autopilot Deaths, dtype: float64
Verified Tesla Autopilot Deaths
1.0       13
2.0        3
3.0        1
16.0       1
19.0       1
75.0       1
118.0      1
2021.0     1
2022.0     1
Name: count, dtype: int64
```