### **Vehicle Movement Analysis and Insight Generation**

**Code Documentation:**

#### **Problem Statement**

Vehicular Movement analysis and insight generation by finding patterns using edge Computing techniques.

**Proposed Solution:**

* Using YOLO object recognition and EasyOCR to detect the entry and exit of vehicles in real time and capture their images and extract the License plate text along with it’s entry/exit time
* Maintenance of a database in order to store the details of day-to-day vehicular movement activities.
* Using the database to generate valuable insights such as frequency of un-registered vehicles, peak time of vehicular activity, average staying time of vehicles etc.
* Providing an interactive User Interface using Streamlit in order to generate summary and other insights and visualization.

**Documentation**

The project was divided into two modules.

* + One to train a object recognition model to extract text from the license plates.
  + The second module was focused on creating a full-fledged database to handle the input data and to create an interactive UI to access details about the data.

**Text Extraction from License Plates**

**Objective:** To extract license plate text from video footage using a trained YOLO model and save the extracted data along with timestamps to a file.

**Dependencies:**

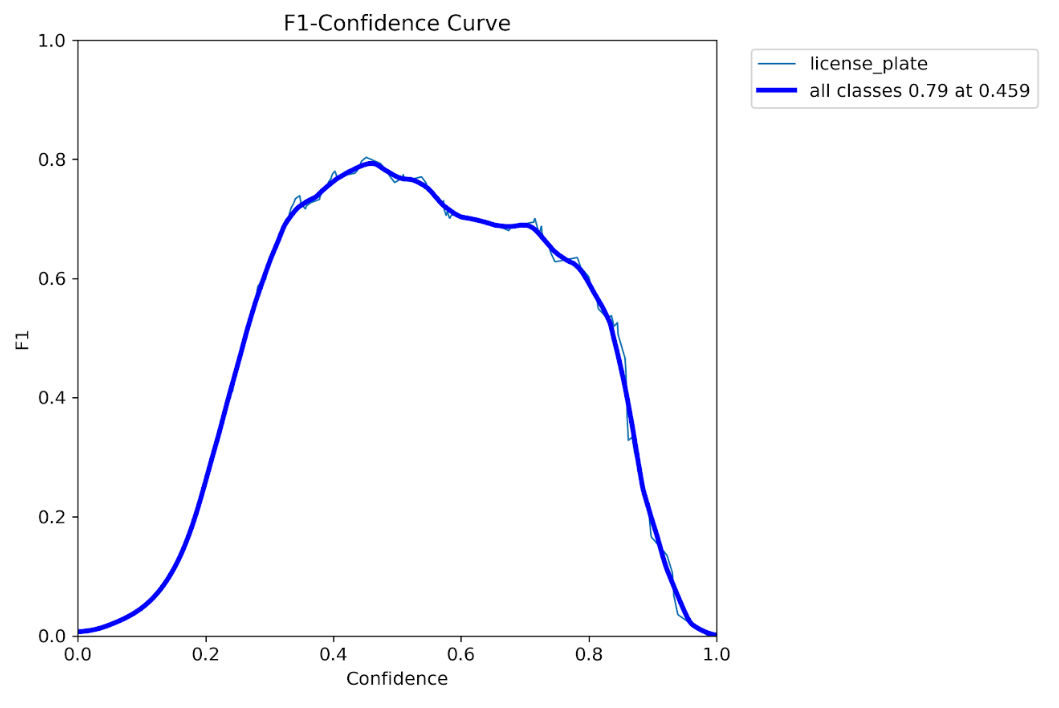
* OpenCV
* Pandas
* Ultranytics YOLO
* cvzone
* NumPy
* EasyOCR
* Datetime

**Training the YOLO model**

* The dataset of vehicle license plate images was manually annotated using “Makesense” online annotation tool. The labels were stored in the form of text document.
* The labelled images are then trained using the YOLO object detection model.
* The command to train the model is
* from ultralytics import YOLO
* # Load a model
* model = YOLO("yolov8n.yaml")  # build a new model from scratch
* # Train the model using the annotated data
* results = model.train(data="data.yaml", epochs=20)
* Here the “data.yaml” file is responsible for storing the meta data of the dataset, such as path of the dataset and the number of classes present in the dataset.

**Results of Trained Model**

**F1 score vs Confidence plot**

****

This plot shows that the model provides good F1 scores with convincing confidence levels of the predictions.

**Test results**

Some of the test results after training the model



**Text extraction**

The below code uses the trained YOLO model in order to detect the license plate images and extract the text from it along with the time stamps, using the easyOCR library. It can take both pre-recorded videos and real time video feed as input.

import cv2

import pandas as pd

from ultralytics import YOLO

import cvzone

import numpy as np

import easyocr

from datetime import datetime

# Initialize EasyOCR reader

reader = easyocr.Reader(['en'], gpu=False)

model = YOLO('model/best.pt')

cap = cv2.VideoCapture('mycarplate.mp4')

my\_file = open("coco1.txt", "r")

data = my\_file.read()

class\_list = data.split("\n")

area = [(32, 398), (16, 456), (1015, 451), (978, 392)]

count = 0

list1 = []

processed\_numbers = set()

# Open file for writing car plate data

with open("car\_plate\_data.txt", "a") as file:

    file.write("NumberPlate\tDate\tTime\n")  # Writing column headers

while True:

    ret, frame = cap.read()

    count += 1

    if count % 3 != 0:

        continue

    if not ret:

       break

    frame = cv2.resize(frame, (1020, 500))

    results = model.predict(frame)

    a = results[0].boxes.data

    px = pd.DataFrame(a).astype("float")

    for index, row in px.iterrows():

        x1 = int(row[0])

        y1 = int(row[1])

        x2 = int(row[2])

        y2 = int(row[3])

        d = int(row[5])

        c = class\_list[d]

        cx = int(x1 + x2) // 2

        cy = int(y1 + y2) // 2

        result = cv2.pointPolygonTest(np.array(area, np.int32), ((cx, cy)), False)

        if result >= 0:

           crop = frame[y1:y2, x1:x2]

           gray = cv2.cvtColor(crop, cv2.COLOR\_BGR2GRAY)

           gray = cv2.bilateralFilter(gray, 10, 20, 20)

           # Use EasyOCR for text extraction

           # The extracted text will be saved in a text file along with current time and date

           result = reader.readtext(gray)

           if result:

               text = result[0][-2].strip()

               text = text.replace('(', '').replace(')', '').replace(',', '').replace(']','').replace('!','').replace('"','').replace('“','')

               if text not in processed\_numbers:

                  processed\_numbers.add(text)

                  list1.append(text)

                  current\_datetime = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

                  with open("car\_plate\_data.txt", "a") as file:

                       file.write(f"{text}\t{current\_datetime}\n")

                       cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 1)

    # Incase if the code is stuck in infinite loop

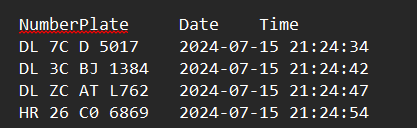
    if count > 1000:  # Remove or Adjust this value as needed

        break

cap.release()

**Extracted Data:**

Executing the above code with the required video input, the output will be stored in a text file.

****

**Database Handling and UI:**

**Objective:** To manage the entry and exit of vehicles in a parking area using a database and provide a user interface for viewing and analyzing the data.

**Database Features:**

* Tables for temporary records and permanent records of vehicle data
* Tracking number of available slots
* Average filled slots per day
* Number of new unauthorised visitors and its average count
* Tracking the average unavailable slots.
* Tracking maximum filled slots every day.
* Tracking peak vehicular activity per day.

**Dependencies:**

* sqlite3
* datetime
* pandas
* streamlit
* matplotlib
* seaborn
* numpy

**The code consists of two major functions**

* the entry function is used to keep track on the vehicles that are entering and also keeping track of the currently occupied slot counts by storing in a temporary table.
* The exit function is called whenever a vehicle seems to leave the parking area. Then the currently occupied slot count is lowered and also the car’s data along with it’s entry and exit time is stored in a permanent table in the database.
* The customer\_data function is used to update the authorized vehicle’s details by uploading them in a table in the database.
* Whenever a vehicle enters, it is compared with the customer table to find if it is a newly entering vehicle or not.

**User Interface:**

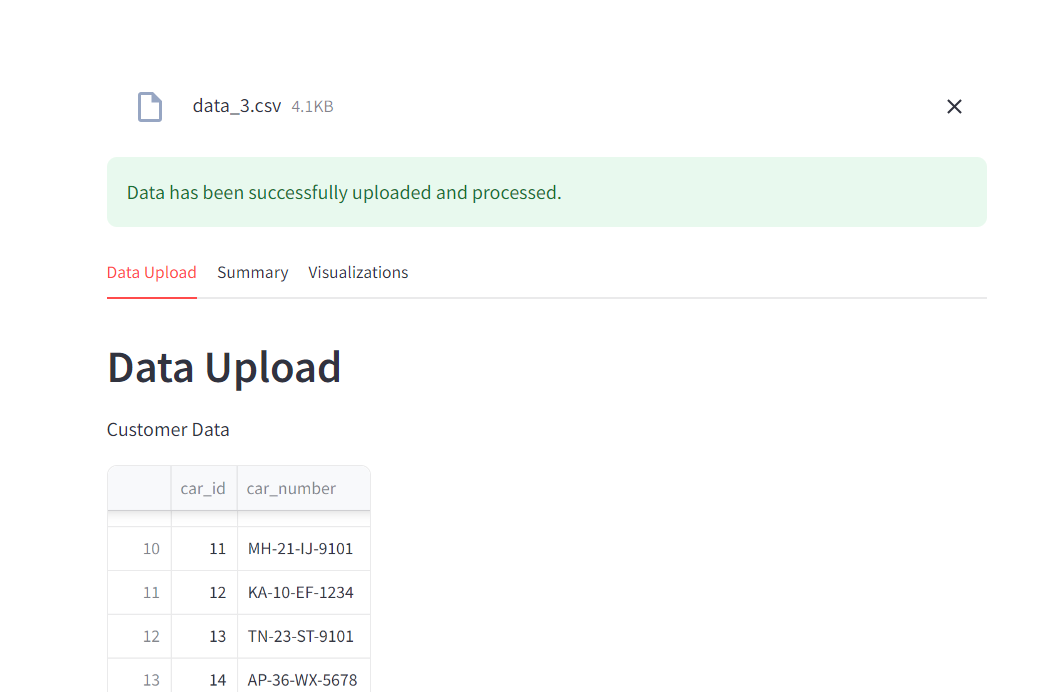
The database class is linked with the UI such that when the input is given a new object is created and the input is tracked.  
The UI prompts the user to upload the dataset of the customer data and the entry\_exit data.

The UI code is run using the following command

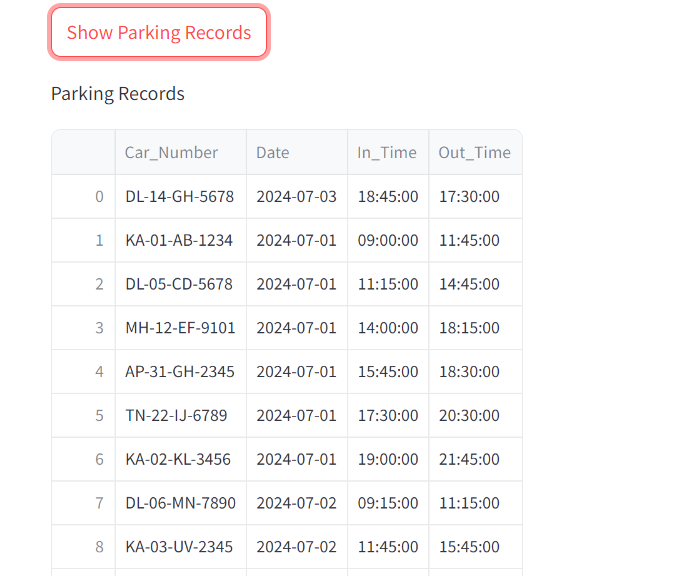
***“streamlit run UI.py”***

****

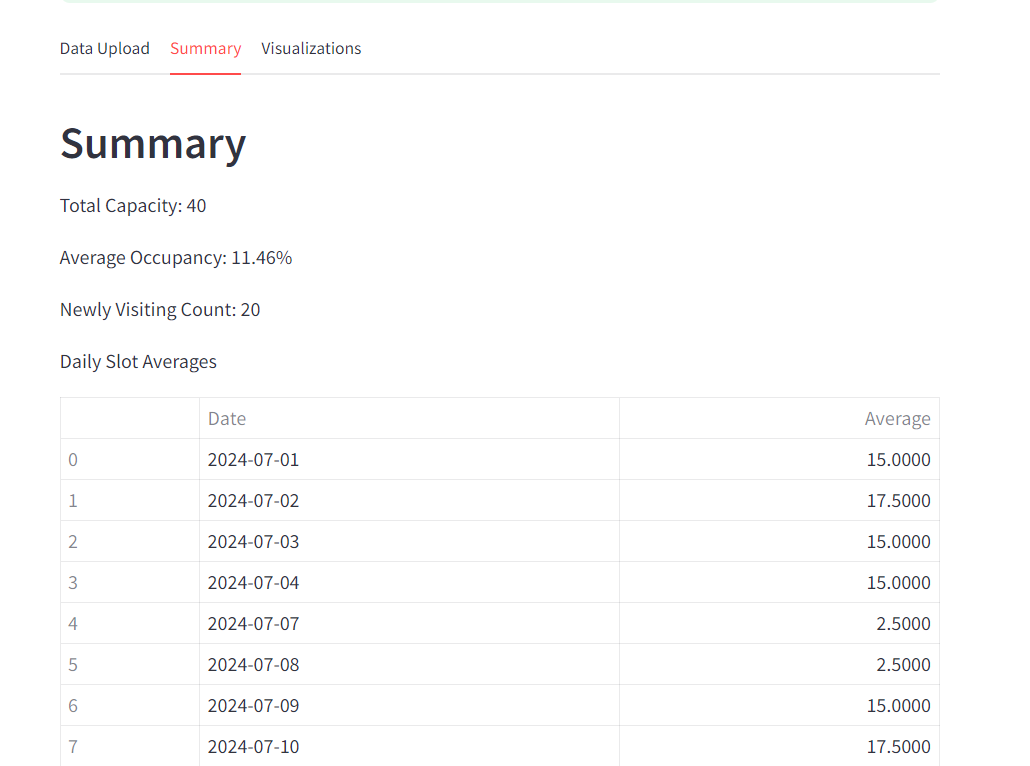
Once the data is uploaded then, it is processed and updated in the database.

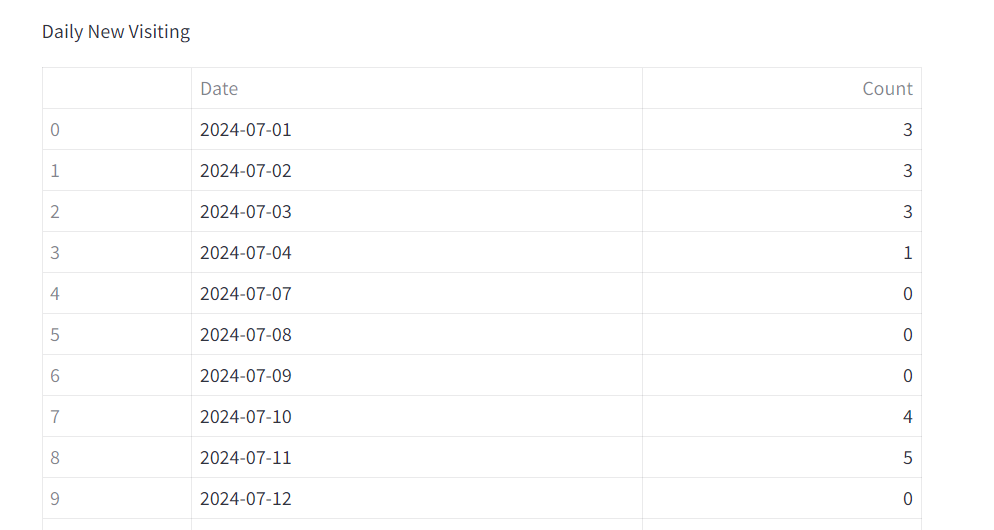


* When the parking records button is clicked, it shows the recorded data of the vehicles that are entered and exited along with in and out time



Hovering through the tabs of the UI gives various insights about the data.





**Visualizations:**

The visualization tab shows various insights about the data in form of plots

Some of the plots shown are

* Distribution of Duration of Stay
* Peak Entry and Exit Times
* Utilization Rates by Hour
* Average Time Between Visits
* Recurrent Visitors

And more

