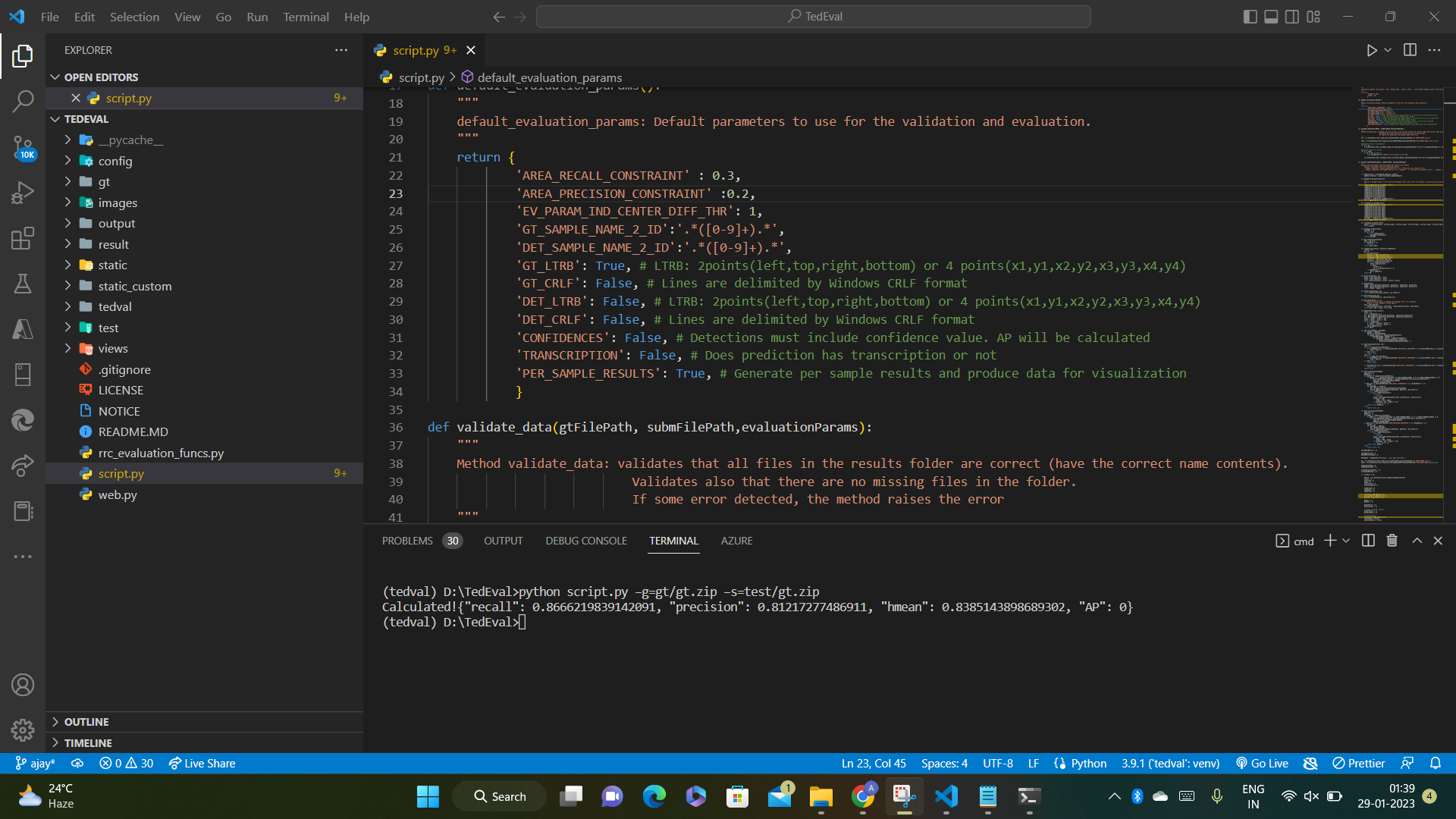
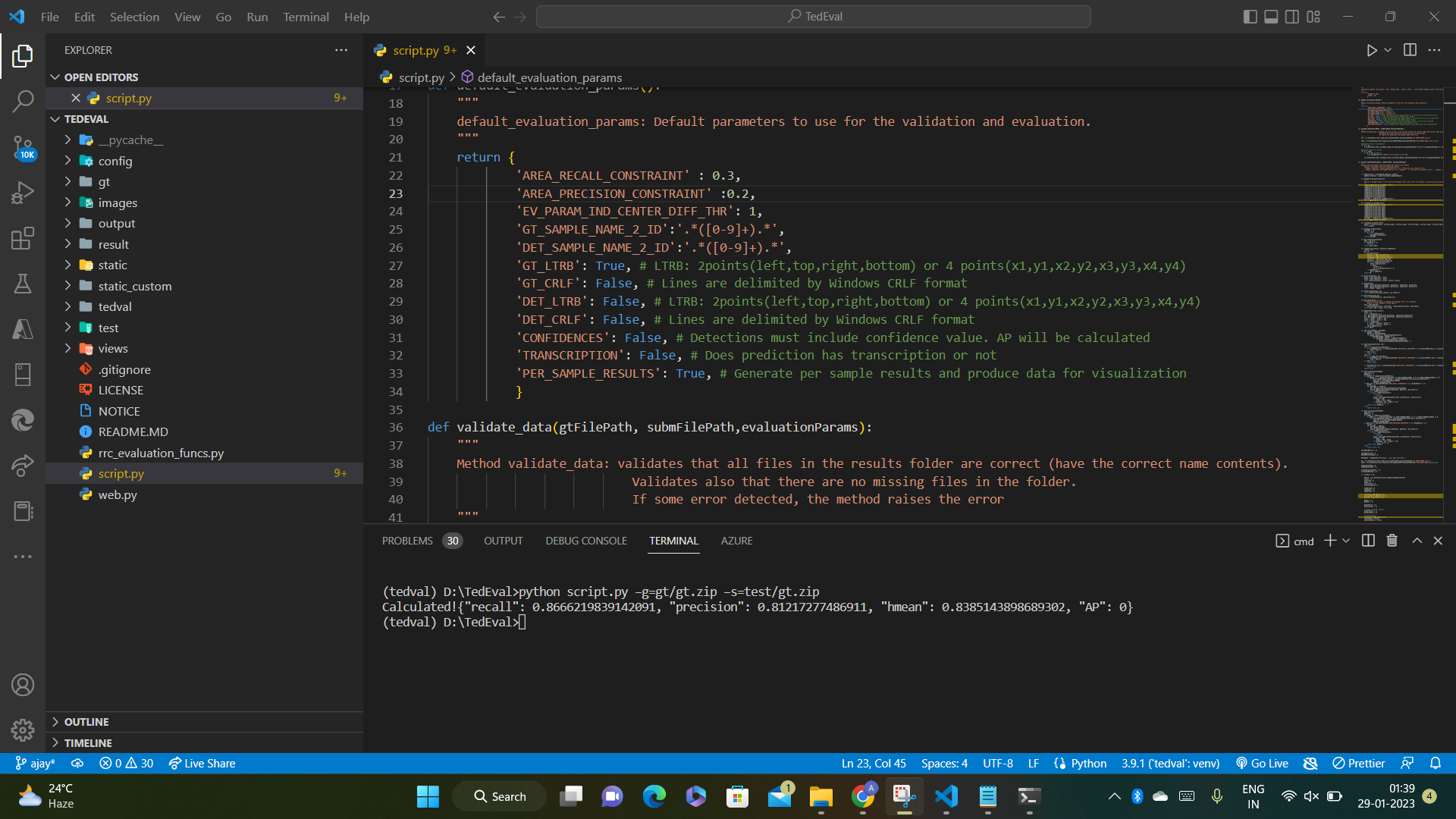
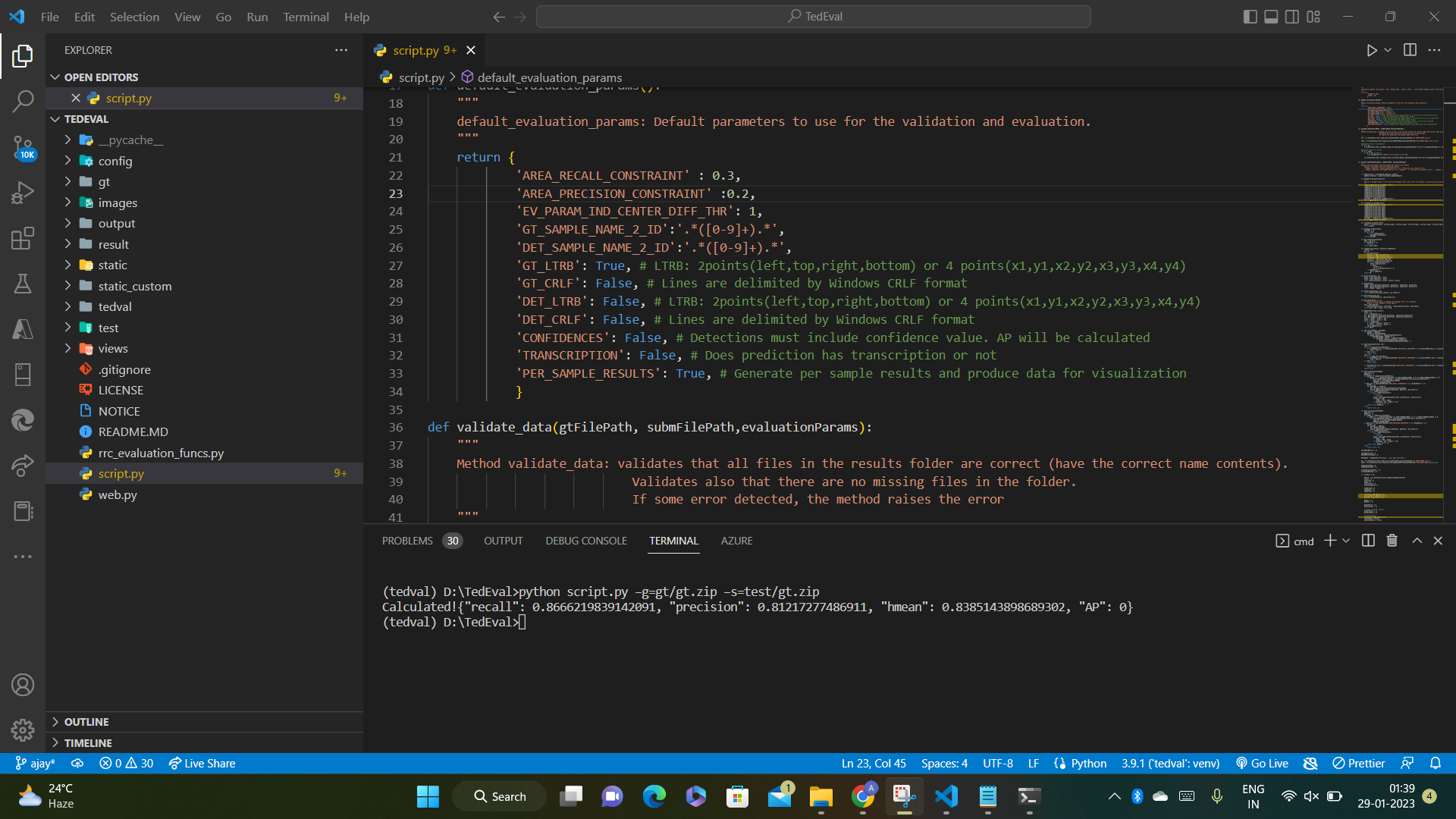
**TedEval metrics:**







Calculated!{"recall": 0.8666219839142091, "precision": 0.81217277486911, "hmean": 0.8385143898689302, "AP": 0}

**Data preparation and training for Keras OCR detector:**

Keras OCR's detector is basically a packaged version of the CRAFT model which needs character level annotations for training it. Using label-studio, 41 images were selected and 9831 character annotations were done on them. This dataset was then used to fine-tune the Keras OCR detector model for customizing it to our OCR tasks.

**Real time streaming:**

Using the IP camera application from Google play store, the real time data stream of the display panel can be wirelessly transferred to our application using a secured RTSP link provided by the IP camera application. Any camera that provides an IP link for wireless transfer can be used here. Using multithreading, a separate daemon thread runs in the background which reads the next available frame from the stream. Because of the huge number of incoming frames, a queue data structure is maintained of length 1 so that latency is reduced and the latest frame is obtained. The main thread can now run our main processing application.

**Processing application:**

Keras OCR's detector model and EasyOCR's text recognition model are loaded with the weights obtained from their training process.

The RTSP (h264\_pcm) link provided by the IP camera application is provided to OpenCV 's VideoCapture method as stream ID. The daemon thread captures frames and the main thread does processing simultaneously. OpenCV shows the video stream for the user to capture a screenshot for further text detection process. This captured frame goes through Keras OCR's detector model and all the text regions are mostly detected in this phase itself. These regions are then drawn on the image frame with its corresponding Region Of Interest (ROI) number for the user to choose from. Suppose some text region is not detected unfortunately, an additional feature is provided which allows the user to draw their own ROI on the frame. ROIs created from both these options are combined to create a ROI template which can then be used for further text recognition processes.

The video streaming process is started again to obtain real time frames. The ROI template is then used to crop images from these frames based on its coordinates. If ROI coordinates have a length of 2, then the frame can be cropped by slicing whereas if it has a length of 4, four point transformation is done initially to obtain a consistent order of coordinates which are then applied with perspective transformation to get cropped images.

Finally these cropped images of each frame are passed to EasyOCR's recognition model to extract the text present in them. A dictionary of these extracted texts with their timestamp when OCR is performed is created which is finally fed into a Pandas Dataframe which logs all the changing values. This Dataframe is then converted to a CSV file which is saved at the end with the final timestamp.

Gayatri Hajare, Utkarsh Kharche, Pritam Mahajan, Apurva Shinde *Automatic Number Plate Recognition System for Indian Number Plates using Machine Learning Techniques,* 2022

Karthick Kanagarathinam, Kavaskar Sekar *Text detection and recognition in raw image dataset of seven segment digital energy meter display,* 2019