**Assignment Report**

**Title: Data Extraction and Formatting from PDF Files**

**Prepared by:** Dollu Harsha Vardhan  
2101070  
CS31, IIIT Guwahati

**1. Introduction**

This report outlines the process of developing a solution for extracting and formatting data from PDF files using OCR (Optical Character Recognition) techniques. The primary goal was to accurately extract structured data, such as voter details, from complex PDF layouts and organize it into a usable format (EXCEL format). This report details the approach taken, tools used, challenges encountered, and the final outcome of the project.

**2. Tools and Technologies Used**

* **Python**: Programming language used for scripting the data extraction process.
* **OpenCV**: A library used for image processing tasks such as contour detection and box segmentation.
* **Tesseract OCR**: The OCR engine employed to extract text from the segmented boxes.
* **Matplotlib**: A plotting library used to visualize the extracted image boxes.
* **Pandas**: For organizing and exporting the extracted data into structured formats like CSV.

**3. Problem Approach**

**3.1 Initial Strategy: Full-Page Text Extraction**

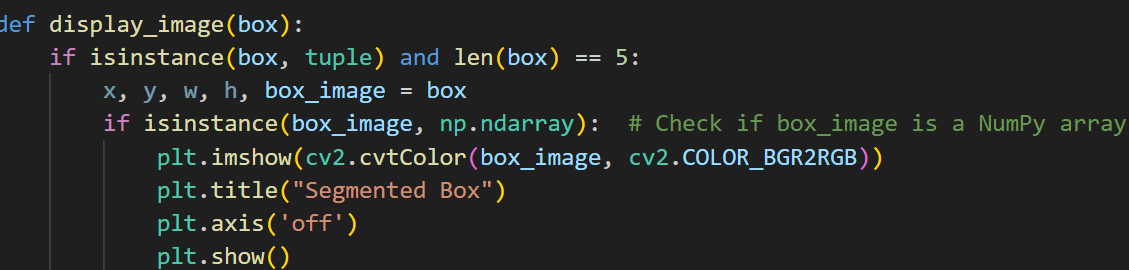
* The initial approach involved converting each page of the PDF into an image (JPG) and applying OCR to extract all text at once. The intent was to split the text based on line breaks and other patterns to organize the data. However, this method proved challenging due to the scattered nature of the text on the page, making it difficult to maintain structure and correctly associate data points.

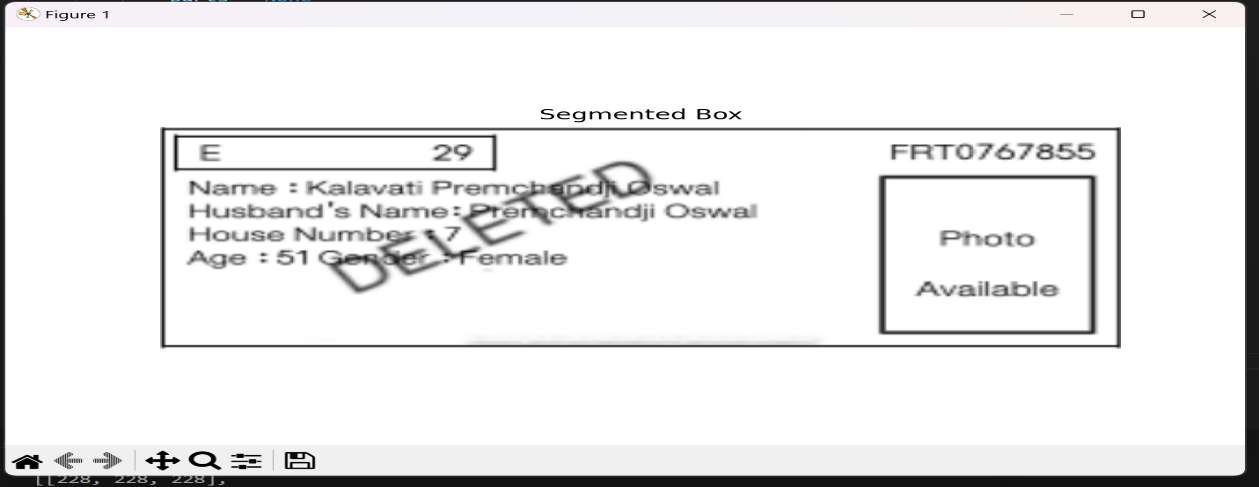
**3.2 Pivot to Box Segmentation**

* To address the issues encountered with full-page extraction, I adopted a box segmentation approach. This method involved detecting and isolating specific boxes within the page that contained structured data, such as voter details. The boxes were identified using contour detection in OpenCV, and then each box was processed individually with Tesseract OCR. This ensured that the extracted data remained well-organized and correctly formatted.

**3.3 Image Display for Debugging and Validation**

* During the development process, it was crucial to visually verify that the correct boxes were being detected and processed. To achieve this, I implemented the display\_image() function, which allowed me to display the segmented boxes as images. This function was essential in debugging and validating the segmentation process.
* The display\_image() function checked if the input was a tuple containing the expected number of elements (x, y, width, height, and the box image itself). It then displayed the box image using Matplotlib after converting it to the RGB color space. Here's the function and Image of single box respectively;





**4. Challenges Faced**

**4.1 Handling Text Overlays and Watermarks**

One significant challenge was the presence of watermarks(“DELETED”) and text overlays, which interfered with the OCR process. These elements distorted the text, leading to inaccurate or incomplete data extraction. Although I applied preprocessing steps like thresholding and noise reduction, the issue persisted, particularly in cases where the watermark was heavily overlaid on critical text. This remains an area for potential improvement in future iterations.

**4.2 OCR Inconsistencies with EPIC Numbers**

Another challenge was the inconsistent extraction of EPIC numbers, due to variations in image quality, the OCR occasionally failed to capture these numbers accurately. This inconsistency made it difficult to rely fully on the extracted data. While I implemented validation checks to catch obvious errors, some inaccuracies still slipped through, reflecting the limitations of the OCR engine in handling diverse text formats and conditions.

**4.3 Time Constraints and Remaining Issues**

Due to the tight deadline, there were some challenges that I couldn't fully address:

* Symbol Misinterpretation: During the OCR process, certain symbols were extracted incorrectly. For example, a colon (:) might be interpreted as an ‘=’ sign or a ‘+’ sign . These inconsistencies sometimes led to merged or incorrect data entries, particularly in sensitive fields like age and gender.
* Inconsistencies in Age and Gender Columns: The OCR occasionally produced inaccuracies in the extracted text, especially for the age and gender columns. These errors were due to the misinterpretation of symbols and the varying quality of the scanned images. Unfortunately, there wasn't enough time to implement a robust solution for these inconsistencies.

Despite these issues, the current solution is effective for the majority of cases. However, further refinement and additional post-processing techniques would be needed to handle these edge cases more accurately**.**

**5. Conclusion**

* Over the past three days, I've had quite a journey exploring and working with Tesseract OCR, diving deeper into improving pandas Data Frames, and tackling the challenges that came my way. Although I didn't manage to fully implement the technique for removing overlapping text, I did explore how to resolve it, which was a valuable learning experience in itself.
* While I know the project isn't completely finished, I feel that I've gained a lot of new skills and revisited some that I already knew. This process has been both challenging and rewarding, and it's given me a better understanding of how to approach similar tasks in the future. Overall, it's been a great learning experience, and I'm proud of the progress I've made, even if there's still more to be done.