A Case Study Report on

Tabular Text Data Extraction Using EasyOCR

Jeyakumar N K

06/06/2024

Case Study Overview

This case study travels through the critical process of extracting text data from images and converting it into a structured tabular format, such as Excel or CSV. Beyond just data retrieval, this conversion enables seamless analysis, manipulation, and integration into existing workflows, unlocking the full potential of the data for tasks like analysis and decision-making. The study focuses on evaluating EasyOCR, a versatile Python module designed for text extraction from images, to address these challenges. Despite EasyOCR lacking native table recognition support, its adaptability for reading both natural scene text and dense document text makes it a promising solution.

The objectives of the study include detecting tabular regions within input images, extracting text data accurately, converting it back into a tabular format, and identifying areas for performance enhancement. To do this, I used various techniques from OpenCV to prepare the image and identify text regions. Then, I used EasyOCR to recognize the text within these regions. Methodologically, the process involves preprocessing images, detecting text regions, passing the region of interest to EasyOCR for text detection, and converting the result back into a tabular form using Python scripts.

Challenges such as handling captions, resizing images, and watermarks are encountered, yet EasyOCR proves efficient, particularly for tables with empty columns. By combining EasyOCR for text extraction and a custom approach for table recognition, effective data extraction from image-based tables is achieved, addressing diverse needs effectively.

This project focuses on turning text found in images into organized tables, like those in Excel or CSV files. Extracting text from images is important because it allows us to analyze and use the information more easily. However, traditional methods struggle with converting image text into structured tables. EasyOCR, a Python tool, helps with this task, even though it doesn't directly recognize tables.

Table of Contents

Click on Chapter Titles for Direct Navigation

I. Introduction:

The importance of extracting text data from images extends beyond just data retrieval; it lies in the subsequent conversion of this extracted text into a structured and easily manipulable format. Converting the extracted text into a tabular format, such as Excel or CSV, holds immense significance for enabling seamless analysis, manipulation, and integration into existing workflows. By structuring the extracted text into rows and columns, organizations can unlock the full potential of the data, facilitating tasks such as data analysis, reporting, and decision-making.

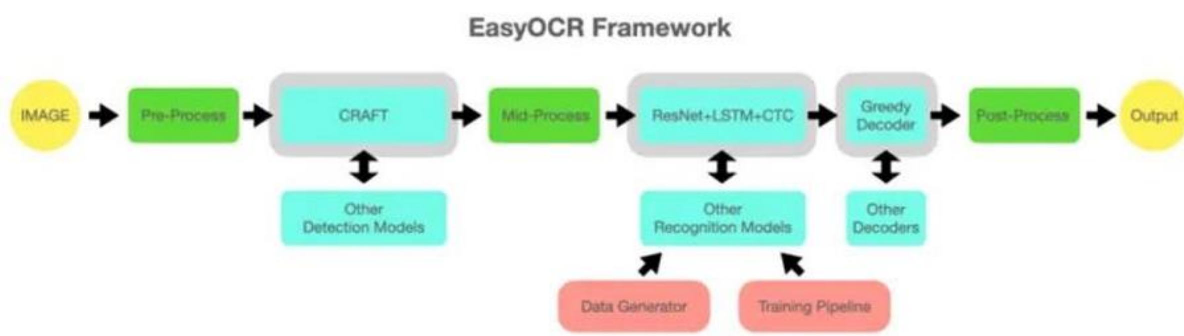


Figure 1.1: EasyOCR Framework

The purpose of this case study is to explore the efficacy of EasyOCR, a Python module specifically designed for extracting text from images, in addressing the aforementioned challenges. EasyOCR distinguishes itself by its versatility in reading both natural scene text and dense text in documents, offering a comprehensive solution for a wide range of applications. With support for over 80 languages, EasyOCR presents a promising tool for organizations seeking to streamline their text extraction processes across diverse linguistic contexts.

Objectives of the Case Study:

1. Tabular Region Detection: The primary objective of this case study is to detect tabular regions within given input images by performing a sequence of OpenCV operations.

2. Text Extraction: Building upon the detection of tabular regions, this case study aims to evaluate EasyOCR's efficacy in accurately extracting text data located within these identified tabular regions.

3. Tabular Conversion: In addition to text extraction, this case study seeks to convert the extracted text data back into a tabular format accurately.

4. Identification of Performance Enhancements: Beyond assessing EasyOCR's current capabilities, this case study endeavors to identify potential areas for performance enhancement. By pinpointing any shortcomings or challenges encountered during the tabular region detection, text extraction, or conversion processes, the study aims to propose recommendations for refining EasyOCR's functionality and optimizing its utility in real-world applications.

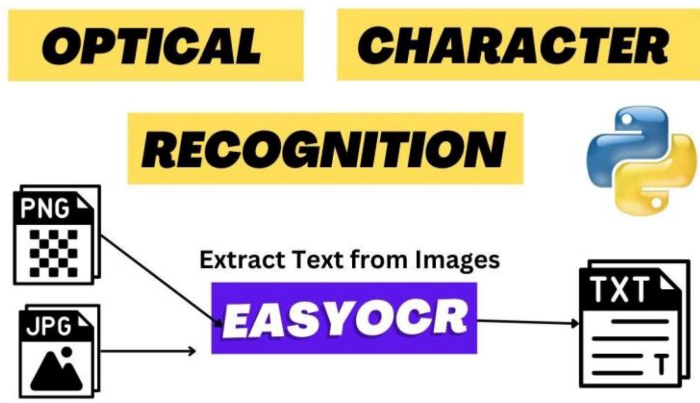


Figure 1.2: Text Detection using EasyOCR

II. Methodology:

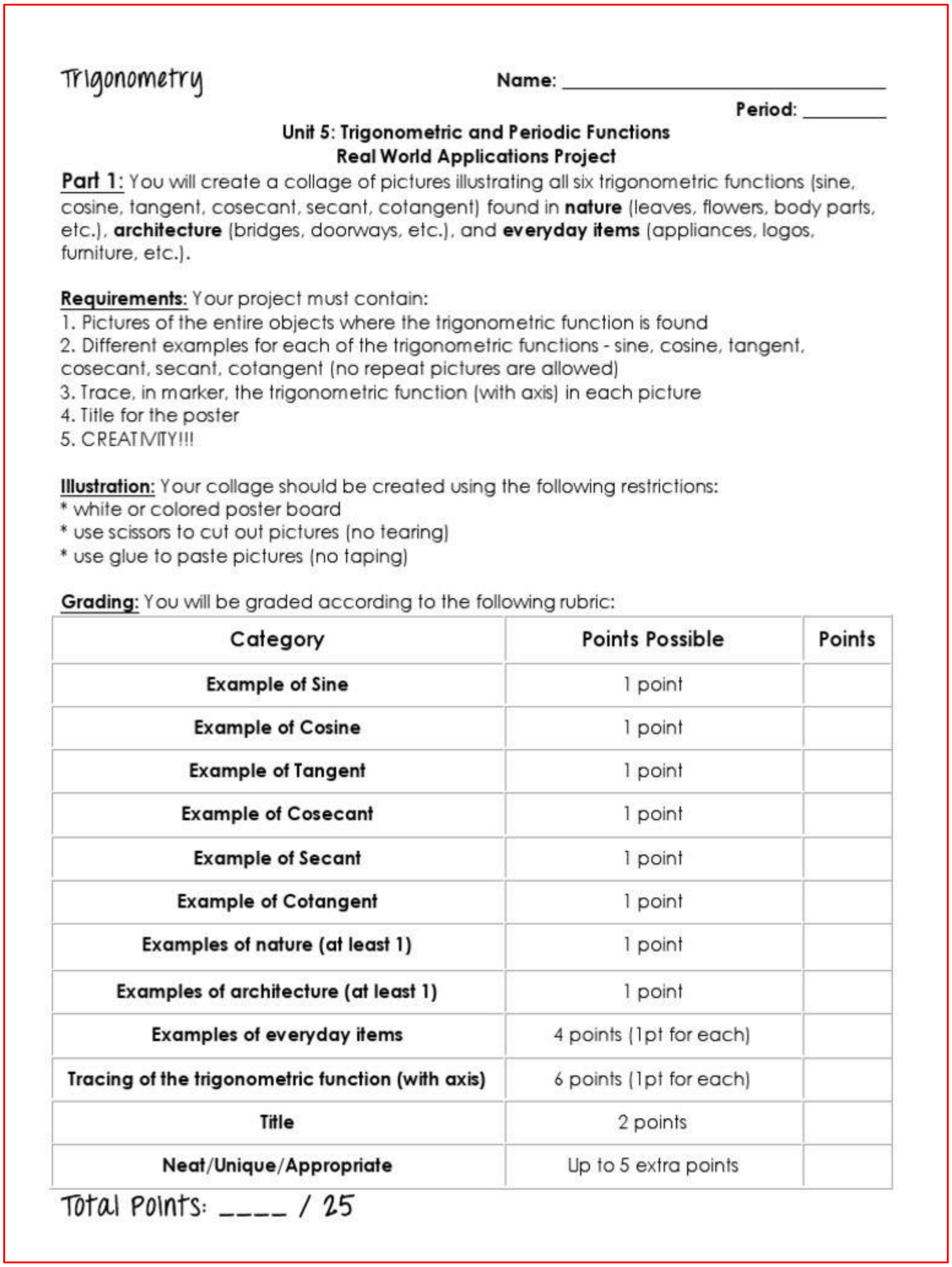
Back to Index

1. Text Extraction Process from Images:

- The text extraction process from images involves several steps to accurately identify and capture textual content embedded within the visual data. This process begins with reading the input image and proceeds through various image processing techniques to enhance the visibility and clarity of the text regions. Once the text regions are identified, optical character recognition (OCR) technology is applied to extract the textual content from the image.

2. Modules or Packages Used for Text Extraction and Tabular Conversion:

- In this case study, the primary tool utilized for text extraction from images are OpenCV for extracting tabular regions and EasyOCR, a Python module specifically designed for extracting text from images. EasyOCR leverages deep learning-based models to recognize and extract text from images with high accuracy and efficiency. Additionally, for the conversion of the extracted text data into a tabular format, standard Python libraries such as Numpy, Pandas may be employed to structure the data into rows and columns resembling a spreadsheet.



Trigonometry

Name: ____________________

Period: ________

Unit 5: Trigonometric and Periodic Functions
Real World Applications Project

Part 1: You will create a collage of pictures illustrating all six trigonometric functions (sine, cosine, tangent, cosecant, secant, cotangent) found in **nature** (leaves, flowers, body parts, etc.), **architecture** (bridges, doorways, etc.), and **everyday items** (appliances, logos, furniture, etc.).

Requirements: Your project must contain:
1. Pictures of the entire objects where the trigonometric function is found
2. Different examples for each of the trigonometric functions - sine, cosine, tangent, cosecant, secant, cotangent (no repeat pictures are allowed)
3. Trace, in marker, the trigonometric function (with axis) in each picture
4. Title for the poster
5. CREATIVITY!!!

Illustration: Your collage should be created using the following restrictions:
* white or colored poster board
* use scissors to cut out pictures (no tearing)
* use glue to paste pictures (no taping)

Grading: You will be graded according to the following rubric:

Category	Points Possible	Points
Example of Sine	1 point	
Example of Cosine	1 point	
Example of Tangent	1 point	
Example of Cosecant	1 point	
Example of Secant	1 point	
Example of Cotangent	1 point	
Examples of nature (at least 1)	1 point	
Examples of architecture (at least 1)	1 point	
Examples of everyday items	4 points (1pt for each)	
Tracing of the trigonometric function (with axis)	6 points (1pt for each)	
Title	2 points	
Neat/Unique/Appropriate	Up to 5 extra points	

Total Points: ____ / 25

Figure 2.1: Input Image

3. Overview of Preprocessing Steps Applied to the Images:

Prior to text extraction, preprocessing steps are applied to the input images to enhance the quality and clarity of the textual content. These preprocessing steps include:

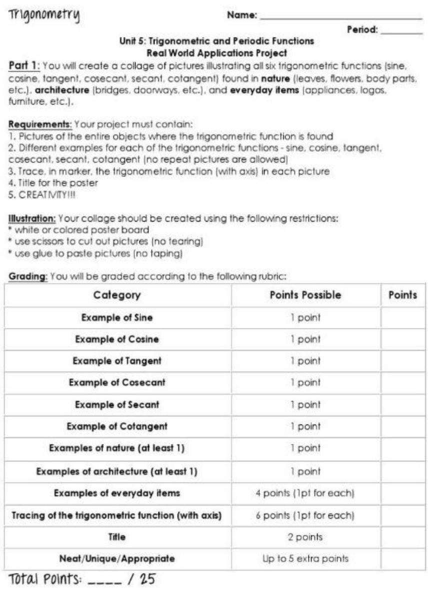
- ***Reading the image:*** The input image is read into the environment for further processing.
- ***Converting it to grayscale:*** The image is converted from color to grayscale to simplify the subsequent image processing operations.
- ***Sharpening edges:*** Edge enhancement techniques are applied to sharpen the edges of text regions, making them more distinguishable from the background.
- ***Thresholding:*** A thresholding operation is performed to segment the image into foreground (text) and background regions based on pixel intensity values.
- ***Canny edge detection:*** The Canny edge detection algorithm is applied to identify prominent edges in the image, further enhancing the visibility of text regions.
- ***Closing operation on Canny:*** Morphological closing operations are performed on the Canny edges to bridge small gaps and smooth the contours of text regions.
- ***Contour detection:*** Contours of text regions are detected using contour detection algorithms, allowing for the identification of tabular regions.
- ***Identifying the largest contour as the tabular region of interest (RoI):*** Among the detected contours, the largest contour is identified as the tabular region containing the text data.
- ***Cropping the input image to the RoI:*** The input image is cropped to isolate the tabular region of interest, facilitating focused text extraction and analysis.

```
image = cv2.imread('WhatsApp Image 2024-05-03 at 15.57.16_82fdd2a1.jpg')
gray_image = cv2.cvtColor(image_file, cv2.COLOR_BGR2GRAY)
kernel = np.array([
    [-1,-1,-1],
    [-1,10,-1],
    [-1,-1,-1]
    ])
```

```
sharpened_image = cv2.filter2D(gray_image, -1, kernel)

threshold = cv2.threshold(sharpened_image, 180, 255, cv2.THRESH_BINARY)[1]

canny = cv2.Canny(threshold, 180, 200, apertureSize = 7)

closing = cv2.morphologyEx(canny, cv2.MORPH_CLOSE, kernel)

contour = cv2.findContours(closing, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]

cv2.drawContours(image_file.copy(), contour, -1, (0, 0, 255), 1)

largest_contour = None
largest_bbox = None
largest_area = 0

for cnt in contour:
    area = cv2.contourArea(cnt)

    if area > largest_area:
            largest_contour = cnt
            largest_bbox = cv2.boundingRect(cnt)
            largest_area = area
  imgcontour = cv2.drawContours(image_file.copy(),[largest_contour],-1,(0,0,255),1)

x, y, w, h = largest_bbox

img_with_bbox = cv2.rectangle(imgcontour, (x, y), (x+w, y+h), (0, 255, 0), 2)

roi = image_file[y : y + h, x : x + w]
```

Table 2.1: Table Detection Python Script

- The code begins by reading an input image file named 'input_image.jpg' using the OpenCV library's **imread()** function as numpy array, storing it in the variable 'image'. Subsequently, the image is converted to grayscale using the **cvtColor()** function with **COLOR_BGR2GRAY**, resulting in a grayscale representation stored in the variable 'gray_image'.
- Next, a 3x3 kernel is defined for sharpening the edges of the image. This kernel is then applied to the grayscale image using the **filter2D()** function, resulting in a sharpened image stored in the variable 'sharpened_image'.



Trigonometry Name: ____________________

Period: ________

Unit 5: Trigonometric and Periodic Functions
Real World Applications Project

Part 1: You will create a collage of pictures illustrating all six trigonometric functions (sine, cosine, tangent, cosecant, secant, cotangent) found in **nature** (leaves, flowers, body parts, etc.), **architecture** (bridges, doorways, etc.), and **everyday items** (appliances, logos, furniture, etc.).

Requirements: Your project must contain:
1. Pictures of the entire objects where the trigonometric function is found
2. Different examples for each of the trigonometric functions - sine, cosine, tangent, cosecant, secant, cotangent (no repeat pictures are allowed)
3. Trace, in marker, the trigonometric function (with axis) in each picture
4. Title for the poster
5. CREATIVITY!!!

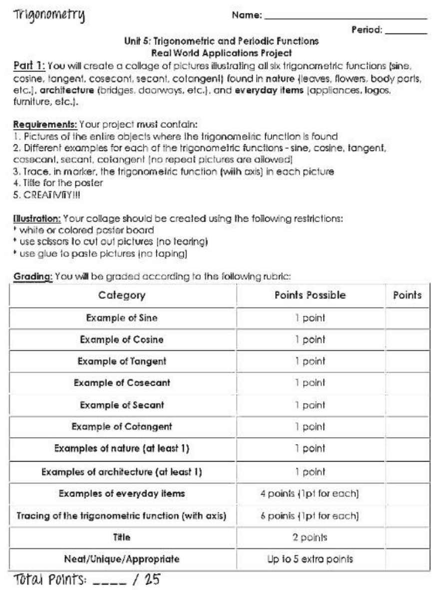
Illustration: Your collage should be created using the following restrictions:
* white or colored poster board
* use scissors to cut out pictures (no tearing)
* use glue to paste pictures (no taping)

Grading: You will be graded according to the following rubric:

Category	Points Possible	Points
Example of Sine	1 point	
Example of Cosine	1 point	
Example of Tangent	1 point	
Example of Cosecant	1 point	
Example of Secant	1 point	
Example of Cotangent	1 point	
Examples of nature (at least 1)	1 point	
Examples of architecture (at least 1)	1 point	
Examples of everyday items	4 points (1pt for each)	
Tracing of the trigonometric function (with axis)	6 points (1pt for each)	
Title	2 points	
Neat/Unique/Appropriate	Up to 5 extra points	

Total Points: ____ / 25

(a) Gray Scale Image



Trigonometry Name: ____________________

Period: ________

Unit 5: Trigonometric and Periodic Functions
Real World Applications Project

Part 1: You will create a collage of pictures illustrating all six trigonometric functions (sine, cosine, tangent, cosecant, secant, cotangent) found in **nature** (leaves, flowers, body parts, etc.), **architecture** (bridges, doorways, etc.), and **everyday items** (appliances, logos, furniture, etc.).

Requirements: Your project must contain:
1. Pictures of the entire objects where the trigonometric function is found
2. Different examples for each of the trigonometric functions - sine, cosine, tangent, cosecant, secant, cotangent (no repeat pictures are allowed)
3. Trace, in marker, the trigonometric function (with axis) in each picture
4. Title for the poster
5. CREATIVITY!!!

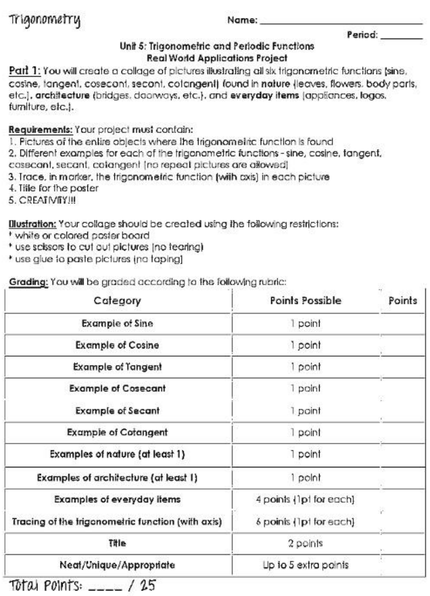
Illustration: Your collage should be created using the following restrictions:
* white or colored poster board
* use scissors to cut out pictures (no tearing)
* use glue to paste pictures (no taping)

Grading: You will be graded according to the following rubric:

Category	Points Possible	Points
Example of Sine	1 point	
Example of Cosine	1 point	
Example of Tangent	1 point	
Example of Cosecant	1 point	
Example of Secant	1 point	
Example of Cotangent	1 point	
Examples of nature (at least 1)	1 point	
Examples of architecture (at least 1)	1 point	
Examples of everyday items	4 points (1pt for each)	
Tracing of the trigonometric function (with axis)	6 points (1pt for each)	
Title	2 points	
Neat/Unique/Appropriate	Up to 5 extra points	

Total Points: ____ / 25

(b) Sharpened Edges



Trigonometry Name: ____________________

Period: ________

Unit 5: Trigonometric and Periodic Functions
Real World Applications Project

Part 1: You will create a collage of pictures illustrating all six trigonometric functions (sine, cosine, tangent, cosecant, secant, cotangent) found in **nature** (leaves, flowers, body parts, etc.), **architecture** (bridges, doorways, etc.), and **everyday items** (appliances, logos, furniture, etc.).

Requirements: Your project must contain:
1. Pictures of the entire objects where the trigonometric function is found
2. Different examples for each of the trigonometric functions - sine, cosine, tangent, cosecant, secant, cotangent (no repeat pictures are allowed)
3. Trace, in marker, the trigonometric function (with axis) in each picture
4. Title for the poster
5. CREATIVITY!!!

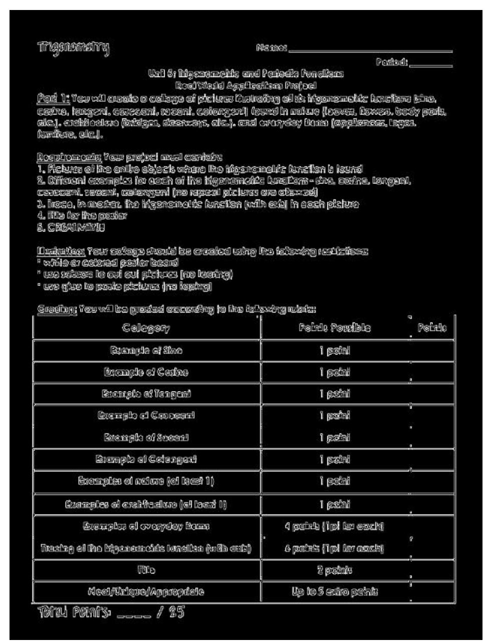
Illustration: Your collage should be created using the following restrictions:
* white or colored poster board
* use scissors to cut out pictures (no tearing)
* use glue to paste pictures (no taping)

Grading: You will be graded according to the following rubric:

Category	Points Possible	Points
Example of Sine	1 point	
Example of Cosine	1 point	
Example of Tangent	1 point	
Example of Cosecant	1 point	
Example of Secant	1 point	
Example of Cotangent	1 point	
Examples of nature (at least 1)	1 point	
Examples of architecture (at least 1)	1 point	
Examples of everyday items	4 points (1pt for each)	
Tracing of the trigonometric function (with axis)	6 points (1pt for each)	
Title	2 points	
Neat/Unique/Appropriate	Up to 5 extra points	

Total Points: ____ / 25

(c) Thresholding



(d) Canny Edge Detection

Table 2.2: Preprocessing Techniques on Input Image

- The sharpened image undergoes a thresholding operation using the **threshold()** function, where pixels with intensity values greater than 180 are set to 255 (white), while others are set to 0 (black). This binary thresholded image is stored in the variable 'threshold'.

- Subsequently, the Canny edge detection algorithm is applied to the thresholded image using the **Canny()** function, resulting in the detection of prominent edges, which are stored in the variable 'canny'.

- A morphological closing operation is then performed on the edges detected by the Canny algorithm using the **morphologyEx()** function with **MORPH_CLOSE**. This operation fills gaps between edges and smooths the contours, producing a closed image stored in the variable 'closing'.

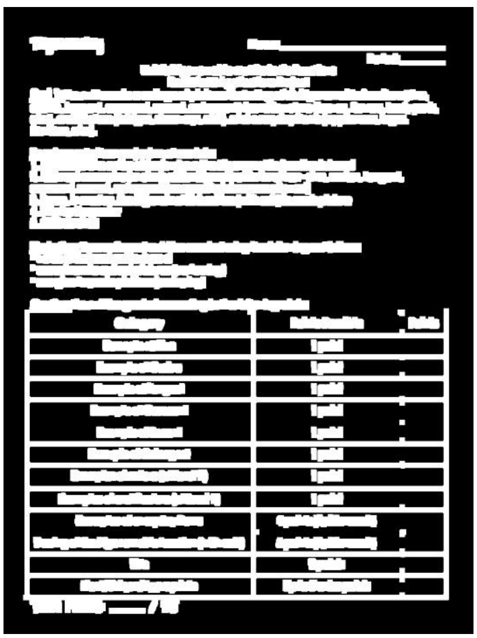


Figure 2.2: Dilated Image

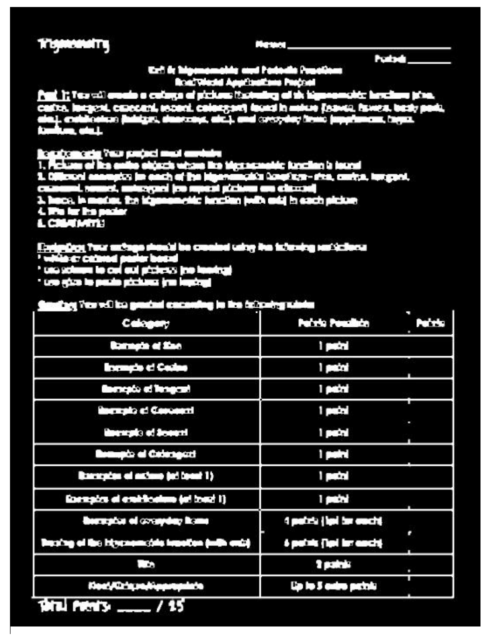
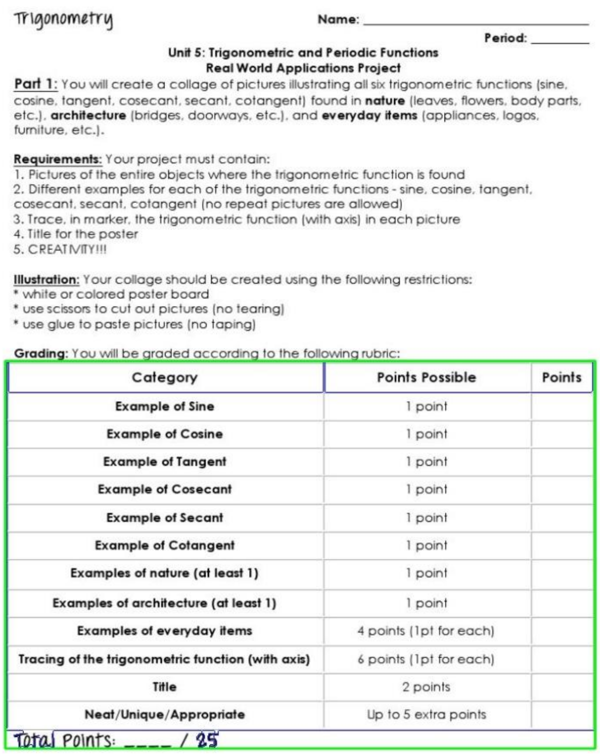


Figure 2.3: Closing Operation On Input Image

- I chose to use a closing operation instead of dilation to specifically ignore any text that's directly above the table or attached to its top. Closing helps smooth out edges and merge nearby elements, making it easier to focus on just the main table content. This way, we can avoid extracting text that might be close to the table but isn't part of it.
- Contours are then detected in the closed image using the **findContours()** function, which returns a list of contours stored in the variable 'contour'.
- The largest contour is identified by iterating through the list of contours and comparing their areas. The bounding box coordinates (x, y, width, height) of the largest contour are extracted and stored in individual variables.



Trigonometry Name: ____________________

Period: ________

Unit 5: Trigonometric and Periodic Functions
Real World Applications Project

Part 1: You will create a collage of pictures illustrating all six trigonometric functions (sine, cosine, tangent, cosecant, secant, cotangent) found in **nature** (leaves, flowers, body parts, etc.), **architecture** (bridges, doorways, etc.), and **everyday items** (appliances, logos, furniture, etc.).

Requirements: Your project must contain:
1. Pictures of the entire objects where the trigonometric function is found
2. Different examples for each of the trigonometric functions - sine, cosine, tangent, cosecant, secant, cotangent (no repeat pictures are allowed)
3. Trace, in marker, the trigonometric function (with axis) in each picture
4. Title for the poster
5. CREATIVITY!!!

Illustration: Your collage should be created using the following restrictions:
* white or colored poster board
* use scissors to cut out pictures (no tearing)
* use glue to paste pictures (no taping)

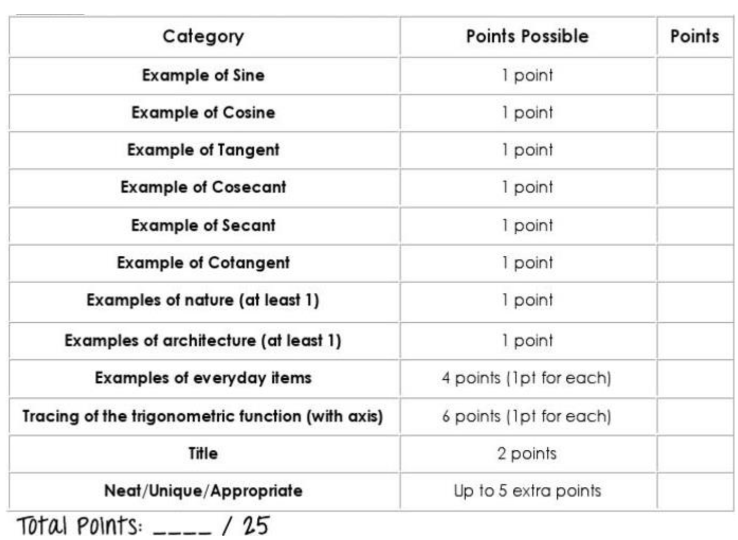
Grading: You will be graded according to the following rubric:

Category	Points Possible	Points
Example of Sine	1 point	
Example of Cosine	1 point	
Example of Tangent	1 point	
Example of Cosecant	1 point	
Example of Secant	1 point	
Example of Cotangent	1 point	
Examples of nature (at least 1)	1 point	
Examples of architecture (at least 1)	1 point	
Examples of everyday items	4 points (1pt for each)	
Tracing of the trigonometric function (with axis)	6 points (1pt for each)	
Title	2 points	
Neat/Unique/Appropriate	Up to 5 extra points	

Total Points: ____ / 25

Figure 2.4: Largest Contour Detected

- Finally, a rectangle is drawn around the region of interest (ROI) defined by the bounding box coordinates on a copy of the original image, highlighting the detected tabular region. The region of interest is then extracted from the original image using the bounding box coordinates and stored in the variable 'roi' for further processing.



Category	Points Possible	Points
Example of Sine	1 point	
Example of Cosine	1 point	
Example of Tangent	1 point	
Example of Cosecant	1 point	
Example of Secant	1 point	
Example of Cotangent	1 point	
Examples of nature (at least 1)	1 point	
Examples of architecture (at least 1)	1 point	
Examples of everyday items	4 points (1pt for each)	
Tracing of the trigonometric function (with axis)	6 points (1pt for each)	
Title	2 points	
Neat/Unique/Appropriate	Up to 5 extra points	

Total Points: ____ / 25

Figure 2.5: Input Image Cropped to Region of Interest

4. Passing RoI to EasyOCR for Text Detection:

After extracting RoI by preprocessing the given input image, we can simply pass the RoI to EasyOCR module for detecting only text present in the tabular region.

```
import easyocr as oc
reader = oc.Reader([‘en’])
result = reader.readtext(roi)
```

Table 2.3: Passing RoI to EasyOCR Module

This code snippet imports the EasyOCR library for text recognition on a designated region of interest (ROI) extracted from an image. An OCR reader object is instantiated using the `Reader()` function from the EasyOCR library, configured to recognize text in the English language ('en'). This object, named 'reader', is then utilized to invoke the `readtext()` method, passing the ROI extracted from the image as its parameter. The `readtext()` method processes the text within the specified ROI using optical character recognition techniques and returns the result, which typically comprises the detected text and its corresponding bounding box coordinates. This result is captured in the variable 'result' for subsequent analysis or further processing.

```
Detected Test Results: [([[145, 8], [228, 8], [228, 35], [145, 35]], 'Category', 0.9775009221672206), ([[433, 11], [555, 11], [555, 31],
 [433, 31]], 'Points Possible', 0.9985647332397556), ([[625, 11], [677, 11], [677, 31], [625, 31]], 'Points', 0.9948524298369781), ([[12
9, 51], [247, 51], [247, 69], [129, 69]], 'Example of Sine', 0.8702974277998347), ([[476, 46], [523, 46], [523, 70], [476, 70]], 'point'
, 0.9665031258880472), ([[115, 81], [258, 81], [258, 106], [115, 106]], 'Example of Cosine', 0.9817100291745914), ([[481, 87], [521, 87]
, [521, 105], [481, 105]], 'point', 0.9293741400607151), ([[114, 117], [262, 117], [262, 141], [114, 141]], 'Example of Tangent', 0.7430
791363523259), ([[476, 116], [523, 116], [523, 140], [476, 140]], 'point', 0.9696398408886909), ([[109, 155], [269, 155], [269, 175], [1
09, 175]], 'Example of Cosecant', 0.962936743247506), ([[481, 157], [521, 157], [521, 173], [481, 173]], 'point', 0.9638947131924763), (
[[117, 191], [259, 191], [259, 211], [117, 211]], 'Example of Secant', 0.8150827687667433), ([[479, 191], [521, 191], [521, 211], [479,
211]], 'point', 0.9634583272227314), ([[105, 227], [273, 227], [273, 247], [105, 247]], 'Example of Cotangent', 0.8167816188413394), ([[
480, 226], [522, 226], [522, 246], [480, 246]], 'point', 0.8965821886855843), ([[77, 261], [299, 261], [299, 281], [77, 281]], 'Examples
 of nature (at least 1)', 0.7540243659569185), ([[479, 261], [521, 261], [521, 281], [479, 281]], 'point', 0.953802130288594), ([[55, 29
9], [321, 299], [321, 319], [55, 319]], 'Examples of architecture (at least 1)', 0.8848066390058598), ([[476, 296], [523, 296], [523, 32
0], [476, 320]], 'point', 0.6853968662412122), ([[84, 334], [293, 334], [293, 355], [84, 355]], 'Examples of everyday items', 0.83100356
3001235), ([[422, 332], [576, 332], [576, 356], [422, 356]], 'points (Ipt for each)', 0.5366906584483264), ([[11, 367], [360, 367], [360
, 392], [11, 392]], 'Tracing of the trigonometric function (with axis)', 0.849662599043716), ([[422, 368], [576, 368], [576, 392], [422,
 392]], 'points (Ipt for each)', 0.7108143711468203), ([[171, 407], [203, 407], [203, 423], [171, 423]], 'Title', 0.9990850638569154), (
[[465, 406], [523, 406], [523, 426], [465, 426]], '2 points', 0.731434124724312), ([[89, 437], [284, 437], [284, 463], [89, 463]], 'Neat
/Unique / Appropriate', 0.7508336074564068), ([[420, 438], [568, 438], [568, 462], [420, 462]], 'Up to 5 extra points', 0.86897098256237
63), ([[8, 470], [134, 470], [134, 493], [8, 493]], 'Total Polnts:', 0.883665811232402), ([[220, 470], [252, 470], [252, 493], [220, 493
]], '25', 0.9999665355662857)]
```

Figure 2.6: List of Tuples held by `result` variable

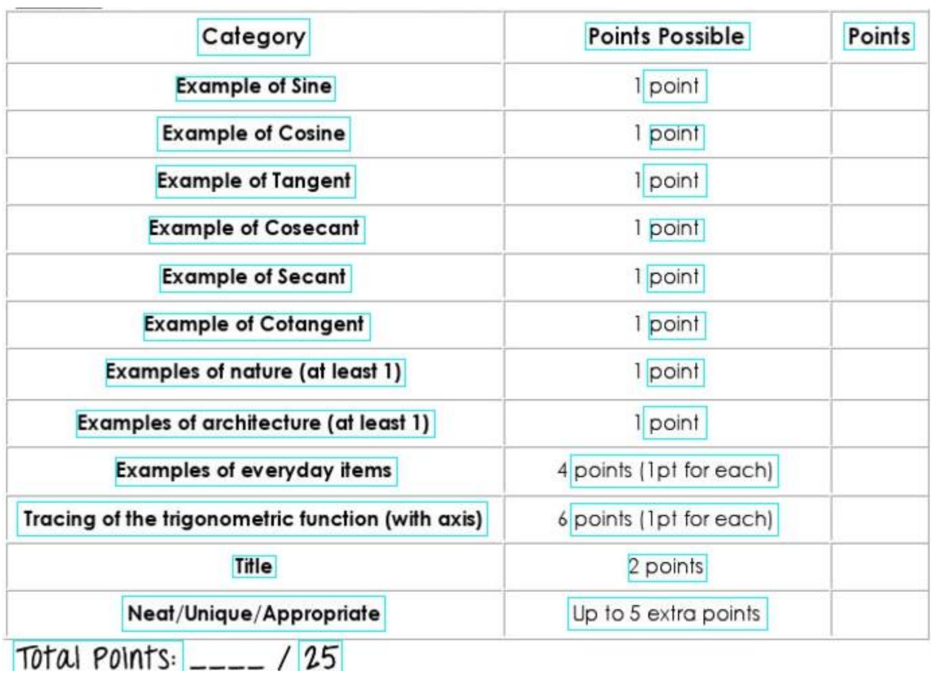
The list contains collection of tuples which encompass three values in it. They are as follows,

([[145, 8], [228, 8], [228, 35], [145, 35]], 'Category', 0.9775009221672206) – 0^{th} Index of List

1^{st} Index of Tuple: [[145, 8], [228, 8], [228, 35], [145, 35]] – It specifies the bounding box information around the text.

2nd Index of Tuple: 'Category' – It specifies the actual detected text content from RoI Image

3rd Index of Tuple: 0.9775009221672206 – This is the confidence value of the detected text information.



Category	Points Possible	Points
Example of Sine	1 point	
Example of Cosine	1 point	
Example of Tangent	1 point	
Example of Cosecant	1 point	
Example of Secant	1 point	
Example of Cotangent	1 point	
Examples of nature (at least 1)	1 point	
Examples of architecture (at least 1)	1 point	
Examples of everyday items	4 points (1pt for each)	
Tracing of the trigonometric function (with axis)	6 points (1pt for each)	
Title	2 points	
Neat/Unique/Appropriate	Up to 5 extra points	

Total Points: ____ / 25

Figure 2.7: Detected Text in RoI After Passing it to EasyOCR Module

5. Converting Back to Tabular Form:

The process of converting back to tabular representation involves several steps aimed at structuring data into a format called a table. It begins with

- Detecting the number of columns from the resultant list of tuples. This step is crucial as it determines how the data will be organized horizontally.

```
def detect_columns(result, start = 0):
    cols = []
    col_ref = []
    x1 = []
    y1 = []
    for tup in range(len(result)):
        x, y = result[tup][0][0]
        x1.append(x)
        y1.append(y)
```

```python
        beg = y1[start] - 10
        end = y1[start] + 10
        for i in range(start, len(result)):
            # st.write(i, y1[i])
            if (y1[i] >= beg) and (y1[i] <= end):
                cols.append(result[i][1])
                col_ref.append(x1[i])
        return (x1, cols, col_ref)

x1, cols, col_ref = detect_columns(result)
```

Table 2.4: Detecting Number of Columns from `result` variable

- Once the number of columns is determined, the next step involves appending the detected row information to the corresponding columns. Each row of data is assigned to its respective column based on the arrangement determined earlier. This ensures that the data is properly aligned within the table structure.

```python
for i in range(len(cols)):
    table.append([])

for tup in range(len(cols)+counter, len(result)):
    x_val = int(x1[tup])
    beg = x_val - 30
    end = x_val + 30
    if (x_val >= beg) and (x_val <= end):
        for i in range(len(cols)):
            if (x_val > (col_ref[i] - 100)) and (x_val < (col_ref[i] + 100)):
                table[i].append(result[tup][1])
```

Table 2.5: Appending Remaining Detected Text Information to Respective Columns

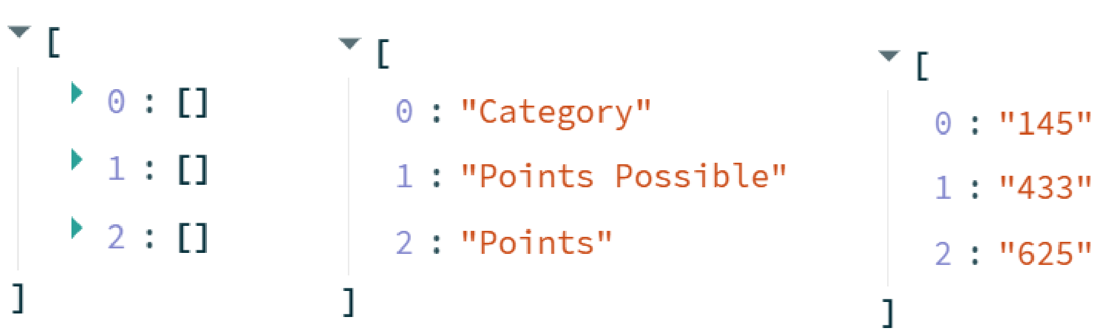


Figure 2.8: (a) Initial Empty Table (b) List of Columns (c) Column References for Appending Rows

- Finally, after all the row information has been appended to the columns, the resultant list is converted into a Pandas DataFrame. Pandas is a powerful Python library commonly used for data manipulation and analysis, and converting the list to a DataFrame allows for further processing, analysis, and visualization of the tabular data. By structuring the data into a tabular representation, it becomes easier to interpret, analyze, and derive insights from the dataset.

```
df = pd.DataFrame(table).transpose()
df = df.rename(columns = {i:cols[i] for i in range(len(cols))})
df.to_excel("results.xlsx")
print(df)
```

Table 2.6: Converting List to DataFrame and Loding them to Excel/ CSV Files

```
                                Category        Points Possible Points
0                        Example of Sine                  point   None
1                      Example of Cosine                  point   None
2                     Example of Tangent                  point   None
3                    Example of Cosecant                  point   None
4                      Example of Secant                  point   None
5                   Example of Cotangent                  point   None
6        Examples of nature (at least 1)                  point   None
7  Examples of architecture (at least 1)                  point   None
8              Examples of everyday items  points (Ipt for each)   None
9                                   Title  points (Ipt for each)   None
10            Neat/Unique / Appropriate                2 points   None
11                                     25   Up to 5 extra points   None
```

Figure 2.9: DataFrame Outcome

	A	B	C	D
1		**Category**	**Points Possible**	**Points**
2	**0**	Example of Sine	point	
3	**1**	Example of Cosine	point	
4	**2**	Example of Tangent	point	
5	**3**	Example of Cosecant	point	
6	**4**	Example of Secant	point	
7	**5**	Example of Cotangent	point	
8	**6**	Examples of nature (at least 1)	point	
9	**7**	Examples of architecture (at least 1)	point	
10	**8**	Examples of everyday items	points (lpt for each)	
11	**9**	Title	points (lpt for each)	
12	**10**	Neat/Unique / Appropriate	2 points	
13	**11**	25	Up to 5 extra points	
14				

Figure 2.10: DataFrame Loaded in Excel File

III. Challenges:

Back to Index

Here are some of the challenges that I faced while doing this study,

- I came to know that EasyOCR does not offer native support for table recognition. For tasks specifically involving the recognition of tables within images, a more suitable solution would be *PaddleOCR's PP-Structure* model. This model is specifically designed to recognize and extract structured information from tables present in images. So, I found it difficult to translate detected texts information into tabular formats like Excel, CSV, TSV files. I just overcame this challenge by analyzing the outcome returned by `result` variable

After analyzing the output of `result` variable, I just highlighted tuples with different colors based on the pattern that I identified. The tuples highlighted with different color in the below image implies different meaning.

- Green Color Indicates – Number of columns of Input Image
- Yellow Color Indicates – Individual row information of Column 1
- Unhighlighted Text – Indicates row information of Column 2

```
([[143, 7], [230, 7], [230, 35], [143, 35]], 'Category', 0.7159639437158294),

([[432, 8], [556, 8], [556, 32], [432, 32]], 'Points Possible', 0.8149897168967521),

([[625, 11], [677, 11], [677, 31], [625, 31]], 'Points', 0.9109005020581996),

([[129, 51], [247, 51], [247, 69], [129, 69]], 'Example of Sine', 0.8702974277998347),

([[480, 50], [521, 50], [521, 69], [480, 69]], 'point', 0.793118081793997),

([[115, 81], [258, 81], [258, 106], [115, 106]], 'Example of Cosine', 0.9817100291745914),

([[481, 87], [521, 87], [521, 105], [481, 105]], 'point', 0.9293741400607151),

([[114, 117], [262, 117], [262, 141], [114, 141]], 'Example of Tangent', 0.7430791363523259),

([[476, 116], [523, 116], [523, 140], [476, 140]], 'point', 0.9696398408886909),

([[108, 154], [269, 154], [269, 175], [108, 175]], 'Example of Cosecant', 0.8046274847951407)
```

Figure 3.1: A Snapshot of output returned by `result` variable

I found out the columns just by taking into account only (x1, y1) coordinates which bounds the detected text region. That is, `result[i][0][0] => 143, 7` here the value of `i` ranges from 0 to n, where n is the number of texts detected and stored as tuple information.

I figured out, for columns, the value of y1 seems to be varying slightly whereas x1 varies greatly. The value of y1 which changes slightly can be adjusted by maintaining some threshold value. So, whatever the values that are ranging between this threshold can be appended to the `columns` list. Once the actual columns of the table are appended to `columns` list, I just appended the corresponding value of x1 (whose value changes greatly) to `column_ref` list. This is just to place the remaining data items to respective columns.

Now we have Columns and Column References for appending data items that corresponds to respective column. By measuring the length of `columns`, I just appended that many empty lists to the `table` list. Similar to the above case, for rows, the value of x1 seems to be varying slightly whereas y1 varies greatly. The value of x1 which changes slightly can be adjusted by maintaining some threshold value. So, whatever the values that are ranging between this threshold can be appended to the corresponding empty lists which we created earlier within `table` list with reference to `column_ref` list.

- The next challenge that I faced is, whenever I resize an image or preprocess it before passing it to the EasyOCR model, the model lacks its ability to detect the correct text which resides in the given input image. The wrongly detected texts are highlighted in the Figure 3.4

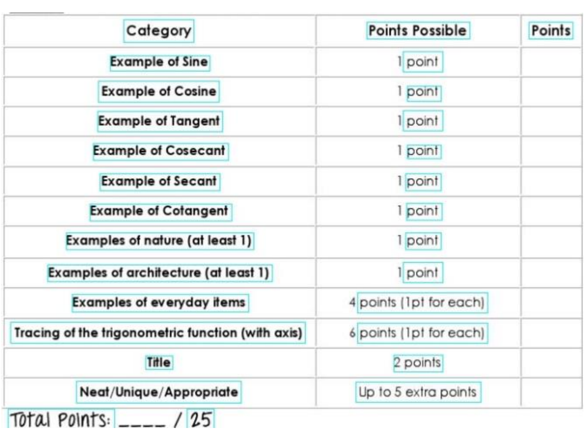


Figure 3.2: Before Preprocessing

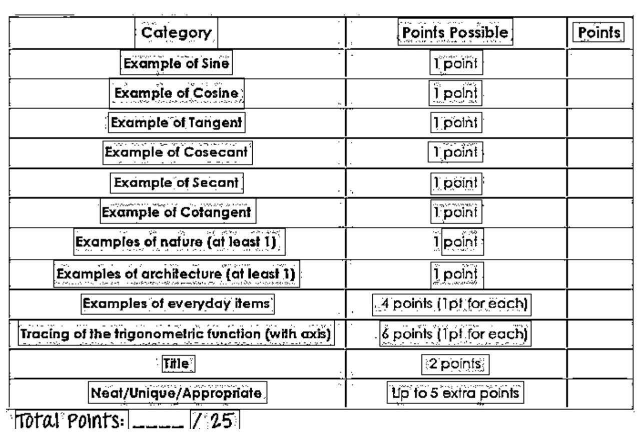
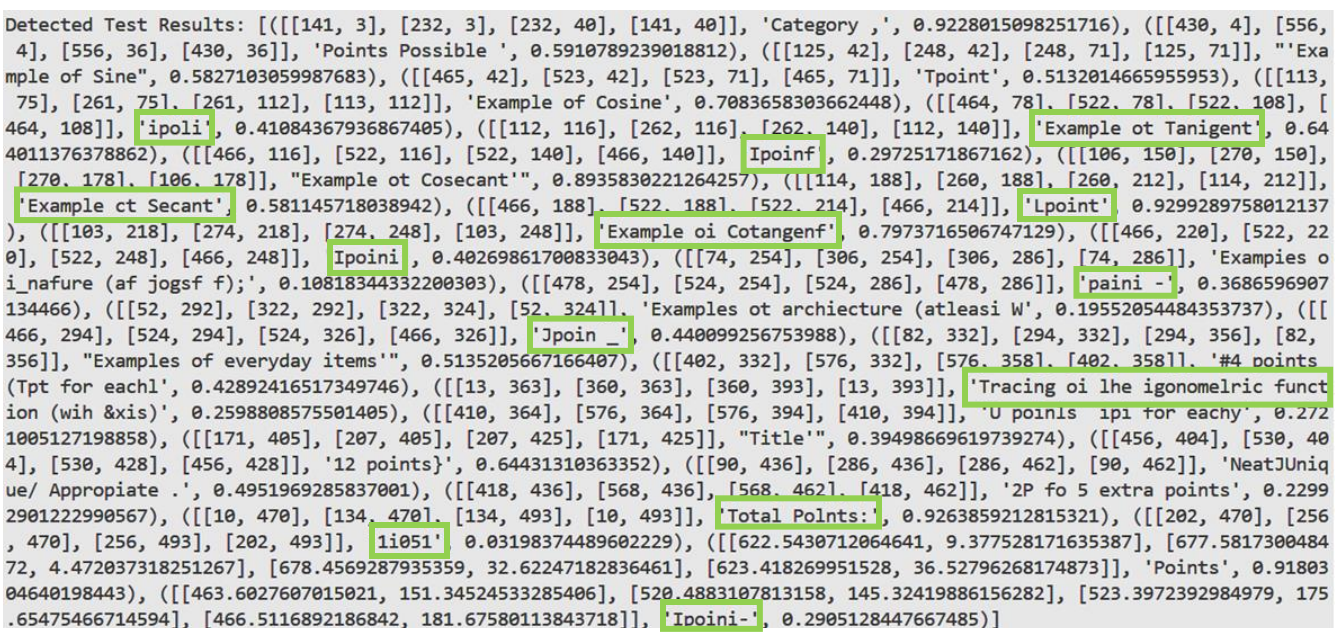


Figure 3.3: After Preprocessing



```
Detected Test Results: [([[141, 3], [232, 3], [232, 40], [141, 40]], 'Category ,', 0.9228015098251716), ([[430, 4], [556, 4], [556, 36], [430, 36]], 'Points Possible ', 0.5910789239018812), ([[125, 42], [248, 42], [248, 71], [125, 71]], "'Example of Sine", 0.5827103059987683), ([[465, 42], [523, 42], [523, 71], [465, 71]], 'Tpoint', 0.5132014665955953), ([[113, 75], [261, 75], [261, 112], [113, 112]], 'Example of Cosine', 0.7083658303662448), ([[464, 78], [522, 78], [522, 108], [464, 108]], 'ipoli', 0.41084367936867405), ([[112, 116], [262, 116], [262, 140], [112, 140]], 'Example ot Tanigent', 0.644011376378862), ([[466, 116], [522, 116], [522, 140], [466, 140]], Ipoinf', 0.29725171867162), ([[106, 150], [270, 150], [270, 178], [106, 178]], "Example ot Cosecant'", 0.8935830221264257), ([[114, 188], [260, 188], [260, 212], [114, 212]], 'Example ct Secant', 0.581145718038942), ([[466, 188], [522, 188], [522, 214], [466, 214]], 'Lpoint', 0.9299289758012137), ([[103, 218], [274, 218], [274, 248], [103, 248]], 'Example oi Cotangenf', 0.7973716506747129), ([[466, 220], [522, 220], [522, 248], [466, 248]], 'Ipoini', 0.40269861700833043), ([[74, 254], [306, 254], [306, 286], [74, 286]], 'Exampies oi_nafure (af jogsf f);', 0.10818344332200303), ([[478, 254], [524, 254], [524, 286], [478, 286]], 'paini -', 0.3686596907134466), ([[52, 292], [322, 292], [322, 324], [52, 324]], 'Examples ot archiecture (atleasi W', 0.19552054484353737), ([[466, 294], [524, 294], [524, 326], [466, 326]], 'Jpoin _', 0.440099256753988), ([[82, 332], [294, 332], [294, 356], [82, 356]], "Examples of everyday items'", 0.5135205667166407), ([[402, 332], [576, 332], [576, 358], [402, 358]], '#4 points (Tpt for eachl', 0.42892416517349746), ([[13, 363], [360, 363], [360, 393], [13, 393]], 'Tracing oi lhe igonomelric function (wih &xis)', 0.2598808575501405), ([[410, 364], [576, 364], [576, 394], [410, 394]], 'U poinls ipi for eachy', 0.2721005127198858), ([[171, 405], [207, 405], [207, 425], [171, 425]], "Title'", 0.39498669619739274), ([[456, 404], [530, 404], [530, 428], [456, 428]], '12 points}', 0.64431310363352), ([[90, 436], [286, 436], [286, 462], [90, 462]], 'NeatJUnique/ Appropiate .', 0.4951969285837001), ([[418, 436], [568, 436], [568, 462], [418, 462]], '2P fo 5 extra points', 0.22992901222990567), ([[10, 470], [134, 470], [134, 493], [10, 493]], 'Total Polnts:', 0.9263859212815321), ([[202, 470], [256, 470], [256, 493], [202, 493]], 1i051', 0.03198374489602229), ([[622.5430712064641, 9.377528171635387], [677.581730048472, 4.472037318251267], [678.4569287935359, 32.62247182836461], [623.418269951528, 36.52796268174873]], 'Points', 0.9180304640198443), ([[463.6027607015021, 151.34524533285406], [520.4883107813158, 145.32419886156282], [523.397239298497, 175.65475466714594], [466.5116892186842, 181.67580113843718]], 'Ipoini-', 0.2905128447667485)]
```

Figure 3.4: EasyOCR fails to detect text accurately after preprocessing

IV. <u>Limitations</u>:

Back to Index

- This is not applicable for table with too many empty cells and for blurred images, but it outperforms in case an entire column is empty as shown in the figure below. It is **prone to None value**

Category	Points Possible	Points
Example of Sine	1 point	
Example of Cosine	1 point	
Example of Tangent	1 point	
Example of Cosecant	1 point	
Example of Secant	1 point	
Example of Cotangent	1 point	
Examples of nature (at least 1)	1 point	
Examples of architecture (at least 1)	1 point	
Examples of everyday items	4 points (1pt for each)	
Tracing of the trigonometric function (with axis)	6 points (1pt for each)	
Title	2 points	
Neat/Unique/Appropriate	Up to 5 extra points	

Total Points: ____ / 25

	Category	Points Possible	Points
0	Example of Sine	point	None
1	Example of Cosine	point	None
2	Example of Tangent	point	None
3	Example of Cosecant	point	None
4	Example of Secant	point	None
5	Example of Cotangent	point	None
6	Examples of nature (at least 1)	point	None
7	Examples of architecture (at least 1)	point	None
8	Examples of everyday items	points (Ipt for ea	None
9	Title	points (Ipt for ea	None
10	Neat/Unique / Appropriate	2 points	None
11	25	Up to 5 extra poi	None

Figure 4.1: Input and Outcome

- If a null value is present in between any cells in a particular column then the data that has to be placed below the null value will be **shifted up to Null value's position** and the place for null value will be preserved at the end of the table.

	Procedure	Trial 1	Trial 2
a	mass of dish, and watch glass	74.14 g	
b	mass of dish, glass and tin	76.20 g	
c	mass of tin =b-a	2.06 g	
d	moles of tin	0173 mol	
e	mass of dish, glass, and product	76.76 g	
f	mass of oxygen =e-b	.56 g	
g	moles of oxygen	.0350 mol	
h	mole ratio	2.02 : 1	
i	accepted ratio	2 : 1	
j	% error	1.00 %	

	Procedure	Trial	Trial 2
0	mass of dlsh; and watch glass	74.14 9	74.14 9
1	Mila5s	76,20	76,20
2	of dish; glass and tin	2.06 9	2.06 9
3	mass of tin =b-a	0173 mol	0173 mol
4	moles of tin	76.76	76.76
5	mass of dish,glass;	0350 mo	0350 mo
6	and product	2.02	2.02
7	mass of oxygen =e-b	LOo	LOo
8	moles of oxygen	None	None
9	mole ratio	None	None
10	accepted ratio	None	None

Figure 4.2: Left Image: RoI, Right Image: Tabular Representation with None values at end of Table

- **Preprocessing/ Resizing** original image comparably affects the EasyOCR's model ability to produce a correct outcome.
- As mentioned in the EasyOCR documentation, the detected text will not follow **natural human reading** so depending on the context the columns may interchange in the output by preserving its data items in the corresponding columns.

- **Watermarks/ background** information present in an image may affect the positioning of data items in the table slightly.

TEST RESULT

S.No	Parameter	Specifications	Result	Test Method
1.	Mashed Potato	Negative	Negative	FSSAI Manual
2.	Sweet Potato	Negative	Negative	FSSAI Manual
3.	Other Starch	Negative	Negative	FSSAI Manual
4.	Rancid Stuff (Old Ghee)	Negative	Negative	FSSAI Manual
5.	Synthetic Colouring Matter	Negative	Negative	FSSAI Manual
6.	Vegetable Oil & Fat	Negative	Negative	FSSAI Manual
7.	Test For Vanaspati	Negative	Negative	FSSAI Manual
8	Curcumin	Negative	Negative	FSSAI Manual
9.	Dalda	Negative	Negative	FSSAI Manual
10.	Lead, mg/kg	Max. 2.5	Not Detected	FSSAI Manual
11.	Arsenic, mg/kg	Max. 1.1	Not Detected	FSSAI Manual
12.	Mercury, mg/kg	Max. 1.0	Not Detected	FSSAI Manual
13.	Cadmium, mg/kg	Max. 1.5	Not Detected	FSSAI Manual

Figure 4.3: RoI with Watermark/ Background Information

	SNo	Parameter	Specifications	Result	Test Method
0	Mashed Potato	Mashed Potato	Negative	Negative	FSSAI Manual
1	Sweet Potato	Sweet Potato	Negative	Negative	FSSAI Manual
2	Other Starch	Other Starch	Negative	Negative	FSSAI Manual
3	Rancid Stuff (Old Ghee)	Rancid Stuff (Old Ghee)	Negative	Negative	FSSAI Manual
4	Synthetic Colouring Matter	Synthetic Colouring Matter	Negative	Negative	FSSAI Manual
5	Vegetable Qil & Fat	Vegetable Qil & Fat	Negative	Negative	FSSAI Manual
6	Test For Vanaspati	Test For Vanaspati	Negative	Negative	FSSAI Manual
7	Curcumin	Curcumin	Negative	Negative	FSSAI Manual
8	Dalda	Tabb	Negative	Negative	FSSAI Manual
9	10.	Dalda	Max 2.5	Not Detect	FSSAI Manual

Figure 4.4: Labs in the previous image detected as Tabb and placed in between actual data items

V. <u>Problem Identified</u>:

Back to Index

- In some cases, the region of interest was detected along with the caption information of the table as shown in the Figure 4.3. Since here I'm considering the text data that is present in first few tuples as the columns for a given input image.

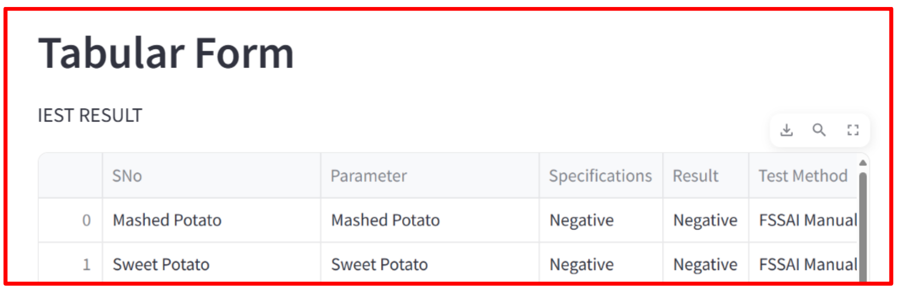
- To overcome this, I just measured the length of columns detected. If it is less than 2 then that will be stored in the 'header' variable.

```
if len(cols) < 2:
   header = cols[0]
   counter += 1
   x1, cols, col_ref = detect_columns(result,
counter)
```

Table 5.1: Python Script for Checking Caption/Header Information

```
def detect_columns(result, start = 0):
```

Figure 5.1: By default, the 'start' is set to zero in function definition



	SNo	Parameter	Specifications	Result	Test Method
0	Mashed Potato	Mashed Potato	Negative	Negative	FSSAI Manual
1	Sweet Potato	Sweet Potato	Negative	Negative	FSSAI Manual

Figure 5.2: Detected Caption Information Displayed Separately Outside the Table

VI. <u>Conclusion</u>:

Back to Index

In conclusion, while EasyOCR may not be suitable for tables with excessive empty cells, it excels when entire columns are empty. Preprocessing or resizing images may impact its accuracy slightly, but overall, EasyOCR reliably preserves data integrity, even if column order may vary. Despite potential challenges like watermarks, EasyOCR remains a valuable tool for efficient and accurate table data extraction. By combining EasyOCR for general text extraction and the approach that I followed for specialized table recognition, table image-based data extraction can be achieved, serving to diverse needs effectively.

VII. <u>Related Works</u>:

Back to Index

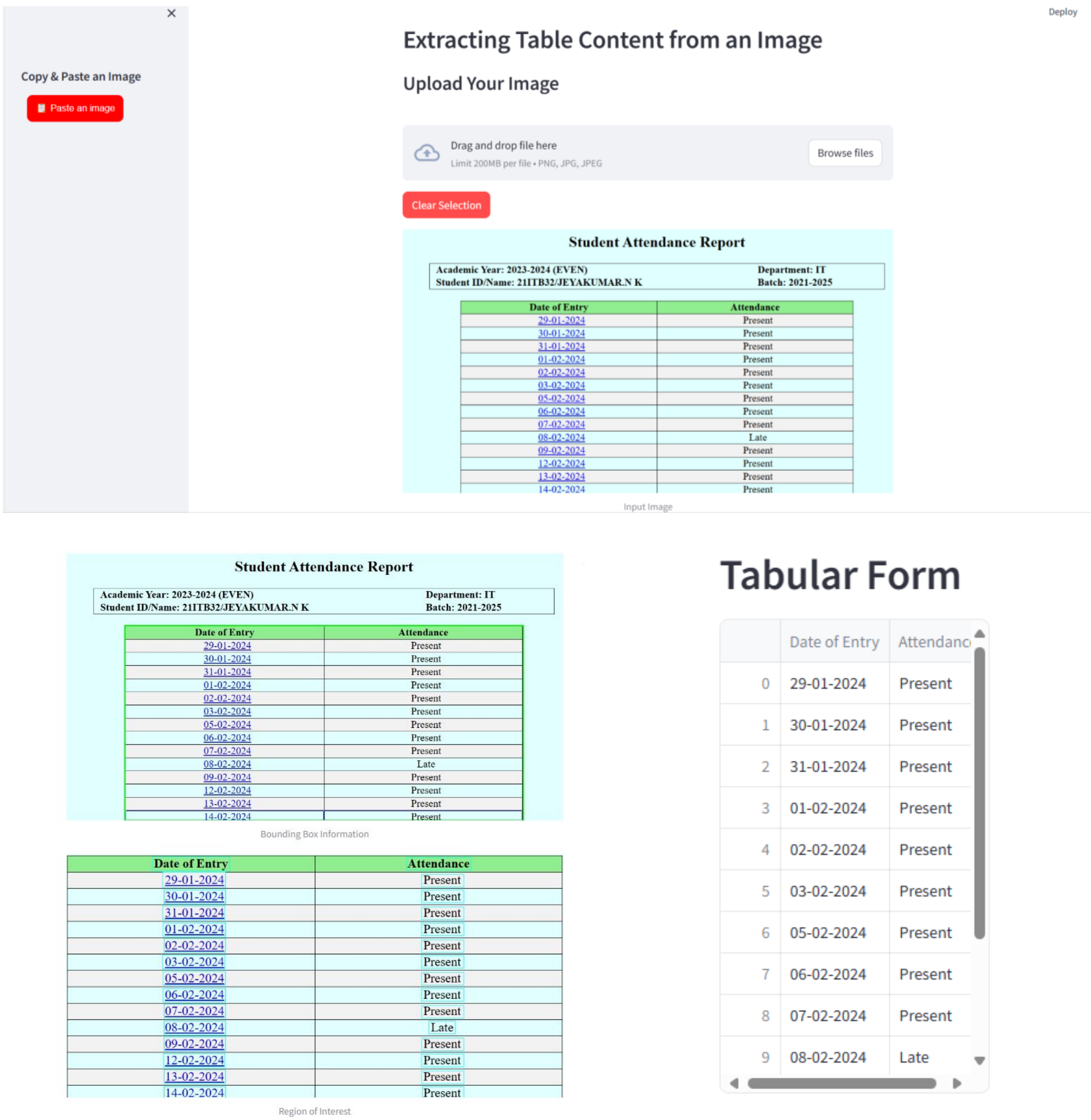


Figure 6.1: Deployed as a Web Application Using Streamlit

<table>
<tr><th>No</th><th>INPUT</th><th>OUTPUT</th></tr>
<tr><td>1</td><td>
<table>
<tr><th>Lab Test</th><th>Result</th><th>Reference Range</th></tr>
<tr><td>Total bilirubin</td><td>20.9 mg/dL</td><td>0.2 - 1.2 mg/dL</td></tr>
<tr><td>Conjugated bilirubin</td><td>12.5 mg/dL</td><td>0.0 - 0.5 mg/dL</td></tr>
<tr><td>Alkaline phosphatase</td><td>327 Units/L</td><td>40 - 150 Units/L</td></tr>
<tr><td>Gamma glutamyltransferase</td><td>185 Units/L</td><td>9 - 64 Units/L</td></tr>
<tr><td>Alanine aminotransferase (ALT)</td><td>34 Units/L</td><td>0-55 Units/L</td></tr>
<tr><td>Aspartate aminotransferase (AST)</td><td>29 Units/L</td><td>5-34 Units/L</td></tr>
<tr><td>Prothrombin time</td><td>18.7 seconds</td><td>12.1 - 14.8 seconds</td></tr>
<tr><td>International normalized ratio (INR)</td><td>1.6</td><td>0.9-1.1</td></tr>
<tr><td>Albumin</td><td>2.5 g/dL</td><td>3.4 - 5.0 g/dL</td></tr>
<tr><td>White blood cell count with differential</td><td>8.9×10^3/uL</td><td>$3.5\text{-}10.5 \times 10^3$/uL</td></tr>
<tr><td>Neutrophils</td><td>81%</td><td>35-70%</td></tr>
<tr><td>Lymphocytes</td><td>11.30%</td><td>20-50%</td></tr>
<tr><td>Monocytes</td><td>18%</td><td>3-15%</td></tr>
</table>
</td><td>
<table>
<tr><th></th><th>Lab Test</th><th>Result</th><th>Reference Range</th></tr>
<tr><td>0</td><td>Total bilirubin</td><td>20.9 mg/ dL</td><td>0.2 -1.2 mg,</td></tr>
<tr><td>1</td><td>Conjugated bilirubin</td><td>12.5 mg/ dL</td><td>0.0 - 0.5 mg</td></tr>
<tr><td>2</td><td>Alkaline phosphatase</td><td>327 Units / L</td><td>40</td></tr>
<tr><td>3</td><td>Gamma glutamyltransferase</td><td>185 Units / L</td><td>150 Units/ L</td></tr>
<tr><td>4</td><td>Alanine aminotransferase (ALT)</td><td>34 Units/L</td><td>64 Units/ L</td></tr>
<tr><td>5</td><td>Aspartate aminotransferase (AST)</td><td>29 Units/ L</td><td>0-55 Units/L</td></tr>
<tr><td>6</td><td>Prothrombin time</td><td>18.7 seconds</td><td>5-34 Units/L</td></tr>
<tr><td>7</td><td>International normalized ratio (INR)</td><td>1.6</td><td>121</td></tr>
<tr><td>8</td><td>Albumin</td><td>2.5 g/dL</td><td>14.8 seconds</td></tr>
<tr><td>9</td><td>White blood cell count with</td><td>8.9 x 10'/uL</td><td>0.9-1.1</td></tr>
<tr><td>10</td><td>differential</td><td>81 %</td><td>34 - 5.0 g/dL</td></tr>
<tr><td>11</td><td>Neutrophils</td><td>11.30%</td><td>3.5-10.5 X 103 /ul</td></tr>
<tr><td>12</td><td>Lymphocytes</td><td>18%</td><td>35-70%</td></tr>
<tr><td>13</td><td>Monocytes</td><td>None</td><td>20-50%</td></tr>
</table>
</td></tr>
<tr><td>2</td><td>
THE UNIVERSITY OF MELBOURNE
HEALTH & SAFETY
EXAMPLE OF A LABORATORY CHEMICAL INVENTORY
<table>
<tr><th>Cas No</th><th>Manufacturer's/Supplier's Chemical Name</th><th>Manufacturer's Name</th><th>Supplier</th><th>Maximum Quantity (L or kg)</th><th>DG Class</th><th>Sub risk</th><th>PG</th><th>UN No</th><th>Haz Sub Y/N</th><th>Poison Schedule</th><th>Current SDS (Yes or No)</th><th>Room Number/ Location</th></tr>
<tr><td>64-19-7</td><td>Acetic acid</td><td>Acetic acid</td><td>Ajax Merck</td><td>2.5L</td><td>8</td><td>3</td><td>II</td><td>2789</td><td>Yes</td><td>6</td><td>16/08/11</td><td>Lab 301</td></tr>
<tr><td>50-78-2</td><td>Acetyl salicylic acid</td><td>Aspirin</td><td>Bayer</td><td>0.3kg</td><td>6.1</td><td>nil</td><td>III</td><td>3249</td><td>Yes</td><td>2</td><td>15/06/12</td><td>Lab 301</td></tr>
<tr><td>7664-41-7</td><td>Ammonia Anhydrous Liquefied</td><td>Ammonia (gas)</td><td>Quenos</td><td>43L</td><td>2.3</td><td>8</td><td>nil</td><td>1005</td><td>Yes</td><td>6</td><td>2/03/13</td><td>Cyl cage</td></tr>
<tr><td>nil</td><td>Ammonia solution 32%</td><td>Ammonia solution 32%</td><td>Ajax Finechem</td><td>2.5L</td><td>8</td><td>nil</td><td>III</td><td>2672</td><td>Yes</td><td>6</td><td>31/01/12</td><td>Lab 301</td></tr>
<tr><td>71-43-2</td><td>Benzene</td><td>Benzene</td><td>Acros</td><td>5L</td><td>3</td><td>nil</td><td>II</td><td>1114</td><td>Yes</td><td>7</td><td>5/06/12</td><td>Lab 301</td></tr>
<tr><td>10043-52-4</td><td>Calcium chloride</td><td>Caltac</td><td>APS Specialty Chemicals</td><td>.5kg</td><td>nil</td><td>nil</td><td>nil</td><td>nil</td><td>Yes</td><td>nil</td><td>23/04/10</td><td>Lab 301</td></tr>
<tr><td>630-08-0</td><td>Carbon monoxide</td><td>Carbon monoxide</td><td>Matheson</td><td>12L</td><td>2.3</td><td>2.1</td><td>nil</td><td>1016</td><td>Yes</td><td>nil</td><td>21/04/10</td><td>Cyl cage</td></tr>
<tr><td>67-66-3</td><td>Chloroform</td><td>Chloroform</td><td>Ajax</td><td>2.5L</td><td>6.1</td><td>nil</td><td>III</td><td>1888</td><td>Yes</td><td>6</td><td>26/04/13</td><td>Lab 301</td></tr>
<tr><td>50-99-7</td><td>D-(+)Glucose</td><td>D-Glucose</td><td>ICN</td><td>1 kg</td><td>nil</td><td>nil</td><td>nil</td><td>nil</td><td>No</td><td>nil</td><td>21/07/11</td><td>Lab 301</td></tr>
<tr><td>3615-56-3</td><td>D-(+)Sorbose</td><td>D-Sorbose</td><td>ICN</td><td>3kg</td><td>nil</td><td>nil</td><td>nil</td><td>nil</td><td>No</td><td>nil</td><td>12/05/10</td><td>Lab 301</td></tr>
<tr><td>7664-39-3</td><td>Hydrofluoric acid</td><td>Hydrofluoric acid</td><td>APS Specialty Chemicals</td><td>1L</td><td>8</td><td>6.1</td><td>I</td><td>1790</td><td>Yes</td><td>7</td><td>10/03/13</td><td>Lab 301</td></tr>
<tr><td>87-79-6</td><td>L-Sorbose</td><td>Sorbin</td><td>Sigma</td><td>5kg</td><td>nil</td><td>nil</td><td>nil</td><td>nil</td><td>No</td><td>nil</td><td>30/03/11</td><td>Lab 301</td></tr>
<tr><td>7783-54-2</td><td>Nitrogen trifluoride</td><td>Nitrogen trifluoride</td><td>Air Liquide</td><td>0.8L</td><td>2.2</td><td>5.1</td><td>nil</td><td>2451</td><td>Yes</td><td>nil</td><td>24/06/10</td><td>Lab 301</td></tr>
<tr><td>7647-14-5</td><td>Sodium chloride</td><td>Common Salt</td><td>APS Specialty Chemicals</td><td>50kg</td><td>nil</td><td>nil</td><td>nil</td><td>nil</td><td>No</td><td>nil</td><td>24/06/10</td><td>Lab 301</td></tr>
</table>
</td><td>
<table>
<tr><th></th><th>Cas Nlo</th><th>manuracturel</th><th>'s Supplier '$</th><th>manuracturel $</th><th>Supplier</th></tr>
<tr><td>0</td><td>Chemical Name</td><td>Chemical Name</td><td>Chemical Name</td><td>Acetic acid</td><td>Acetic acid</td></tr>
<tr><td>1</td><td>64-19-7</td><td>64-19-7</td><td>Acetic acid</td><td>Aspirin</td><td>Ajax Merck</td></tr>
<tr><td>2</td><td>Acetic acid</td><td>Acetic acid</td><td>Acetic acid</td><td>Armonia (gas)</td><td>Aspirin</td></tr>
<tr><td>3</td><td>50-78-2</td><td>50-78-2</td><td>Acetyi</td><td>Armonia</td><td>Bayer</td></tr>
<tr><td>4</td><td>Acetyi</td><td>Acetyi</td><td>salicylic acid</td><td>solution 326</td><td>Armonia (gas)</td></tr>
<tr><td>5</td><td>salicylic acid</td><td>salicylic acid</td><td>Aspirin</td><td>Jenzenes</td><td>Ouenos</td></tr>
<tr><td>6</td><td>7664-41-7</td><td>7664-41-7</td><td>Armonia Anhydrous</td><td>Caltac</td><td>Armonia</td></tr>
<tr><td>7</td><td>Armonia Anhydrous</td><td>Armonia Anhydrous</td><td>Armonia (gas)</td><td>Carbon</td><td>Ajax Finecher</td></tr>
<tr><td>8</td><td>Liguefied</td><td>Liguefied</td><td>Liguefied</td><td>monoxidelMatheson</td><td>solution 326</td></tr>
<tr><td>9</td><td>Ammonia solution 32</td><td>Ammonia solution 32</td><td>Ammonia solution 32</td><td>Chloroforn</td><td>Jenzenes</td></tr>
</table>
</td></tr>
</table>

Table 6.1 Passing Different Inputs to the application

Back to Top

VIII. References:

- https://www.jaided.ai/easyocr/tutorial/
- https://www.analyticsvidhya.com/blog/2021/06/text-detection-from-images-using-easyocr-hands-on-guide/
- https://towardsdatascience.com/pre-processing-in-ocr-fc231c6035a7