Guided Project Report

Text Detection

Name: Shruti Verma
Course: AI and ML
(Batch 4)
Duration: 10 months

Problem Statement: Implement a text detection and extraction model using OpenCV and OCR

Prerequisites

What things you need to install the software and how to install them:

Python 3.8 or higher versions This setup requires that your machine has latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not- recognized-as-an-internal-or-external- command/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.8 then run below commands in command prompt/terminal to install these packages pip install -U scikit-learn pip install numpy pip install scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy . Install the pytesseract using the command, pip install pytesseract and open CV using pip install opencv-python.

Video Link

https://drive.google.com/drive/folders/1nKjHKic6ZATLwmXNwcyb1Guxb1LNOrrO

Dataset used

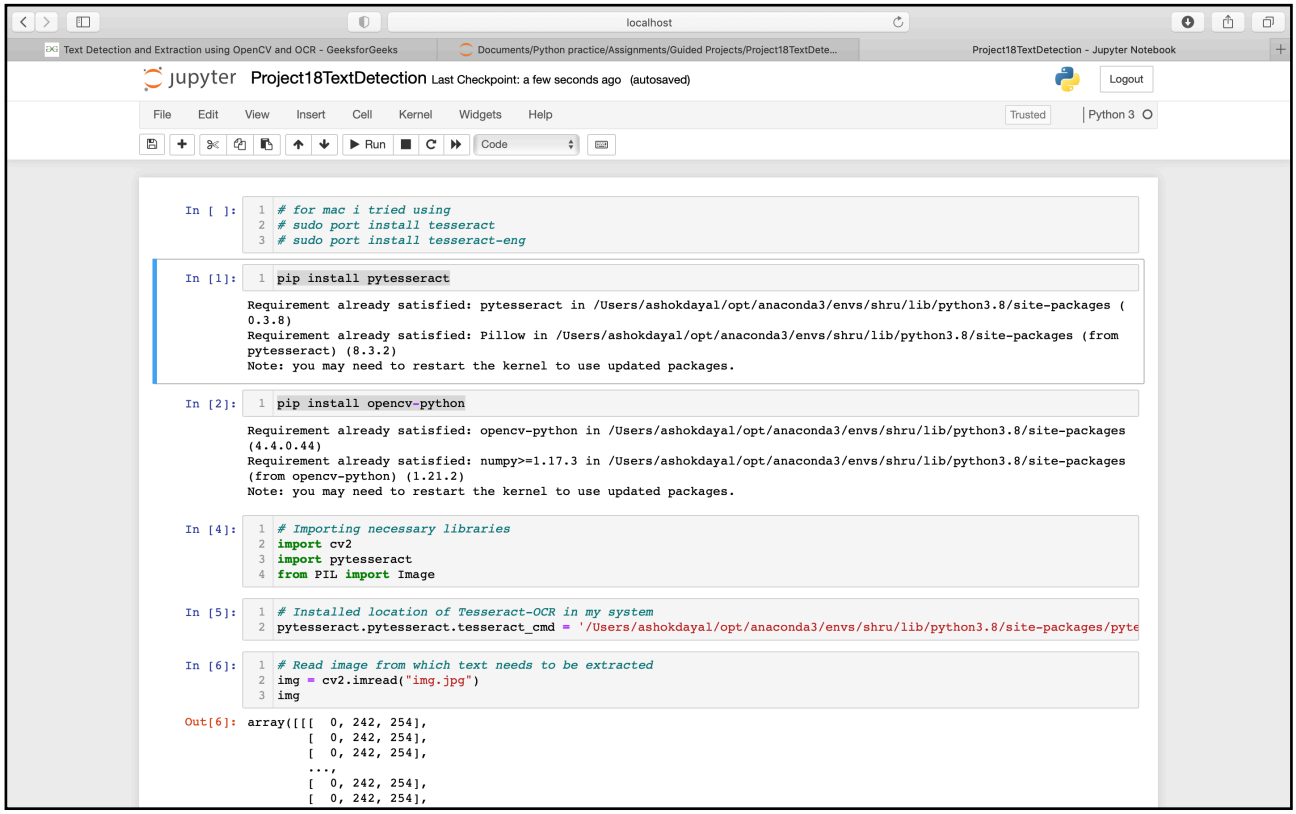
Sample image with text - img.jpeg

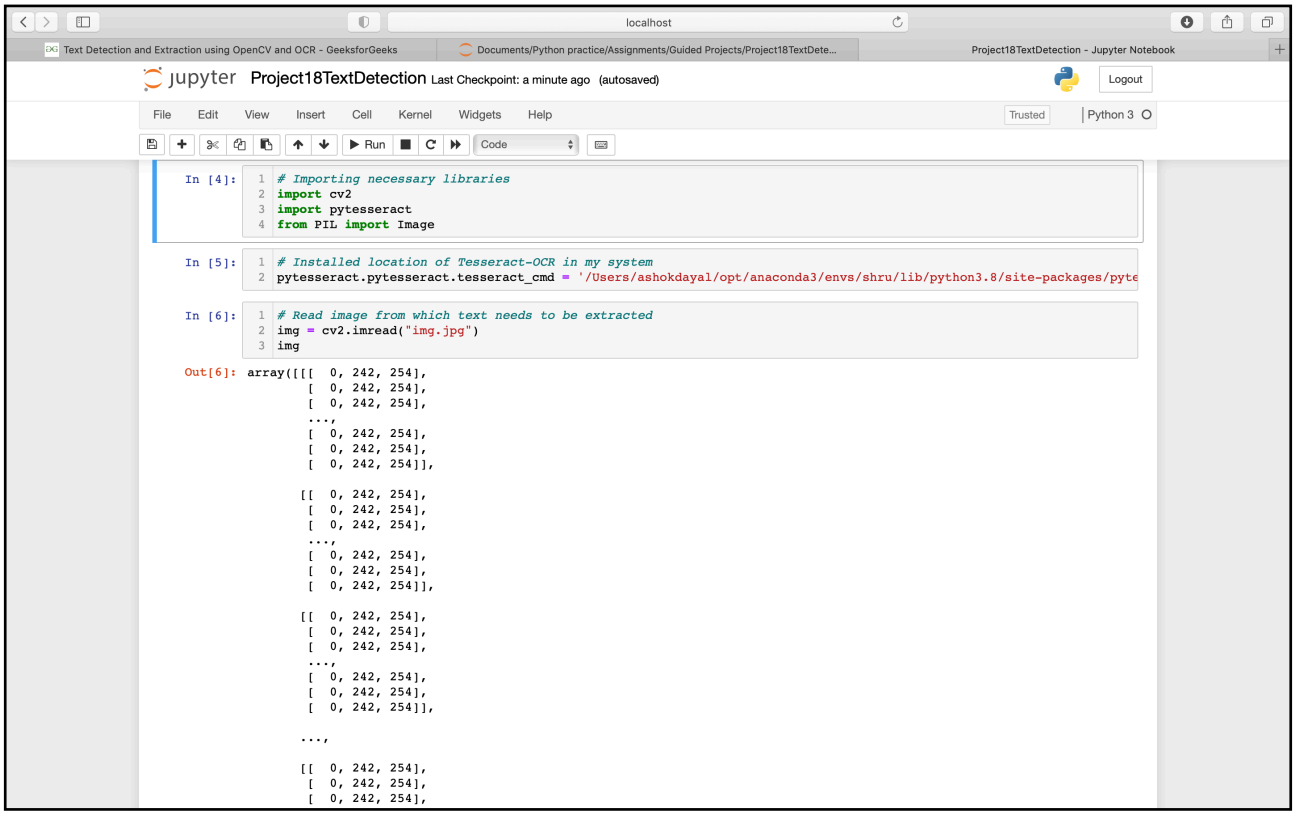
Method used for detection

- Image Preprocessing
- Contours representing the text areas
- Applying OCR using python-tesseract

Importing the libraries and capturing images:

Installing python-tesseract and openCV

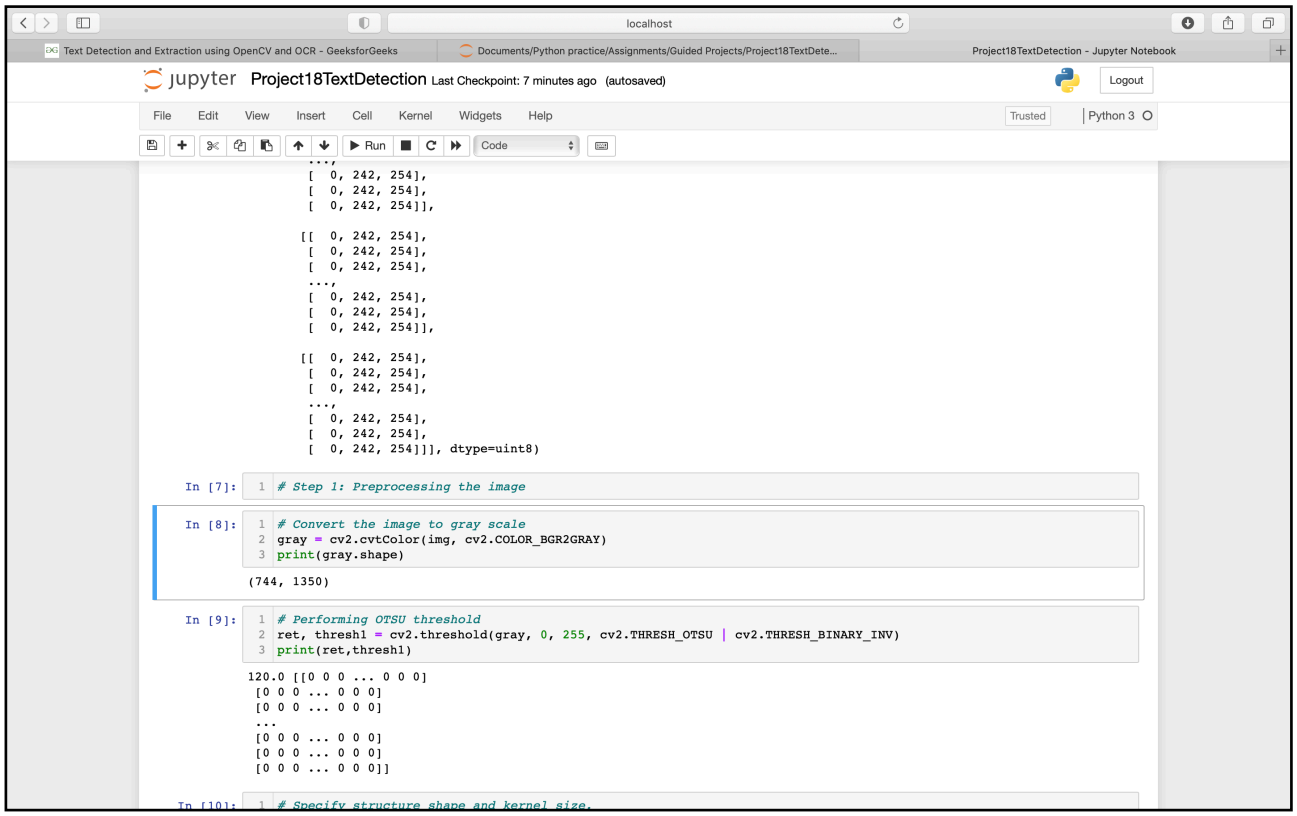




```
In [4]:
# Importing necessary libraries
import cv2
import pytesseract
from PIL import Image

In [5]:
# Installed location of Tesseract-OCR in my system
pytesseract.pytesseract.tesseract_cmd = '/Users/ashokdayal/opt/anaconda3/envs/shru/lib/python3.8/site-packages/pyte

In [6]:
# Read image from which text needs to be extracted
img = cv2.imread("img.jpg")
img

Out[6]: array([[[  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254],
                ...,
                [  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254]],

               [[  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254],
                ...,
                [  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254]],

               [[  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254],
                ...,
                [  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254]],

               ...,

               [[  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254],
```

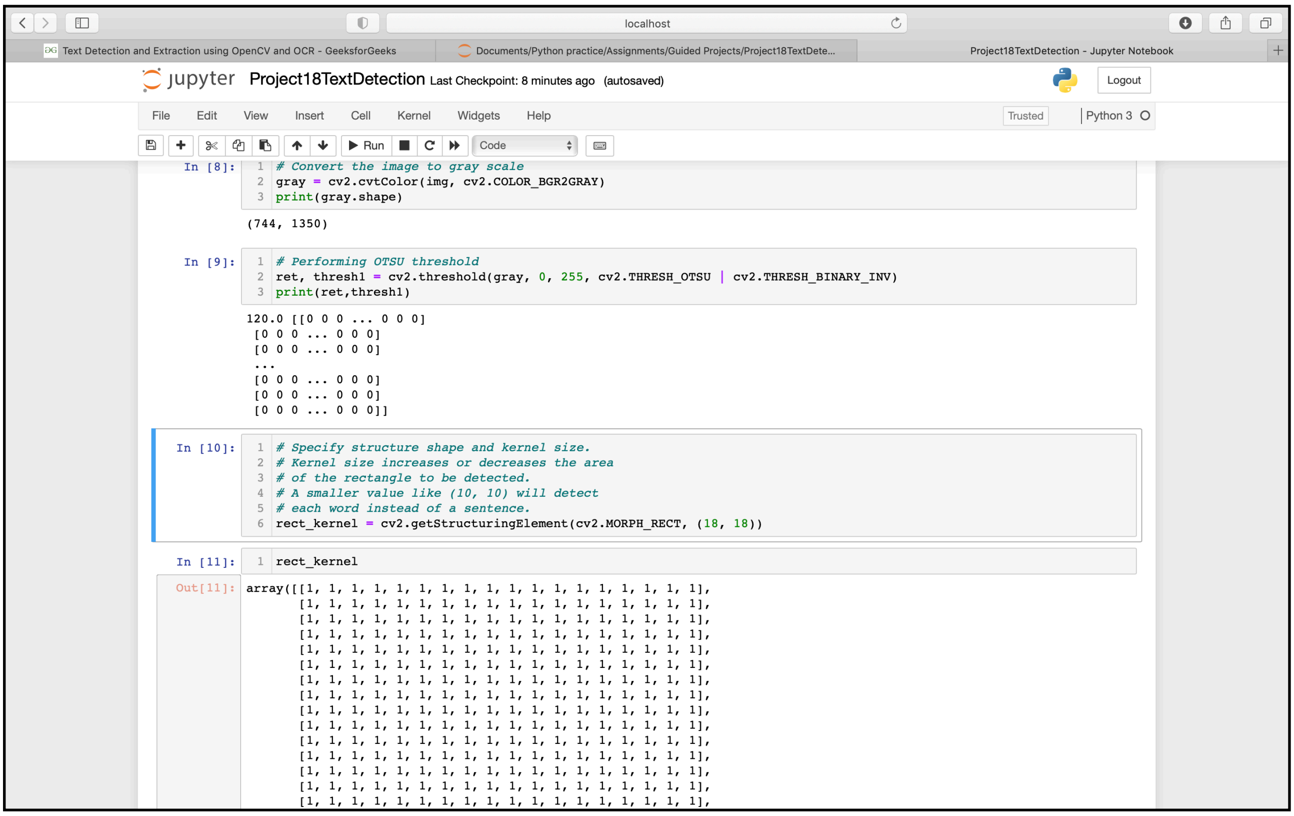
Importing necessary libraries and reading the image ‘img’

Preprocessing the image



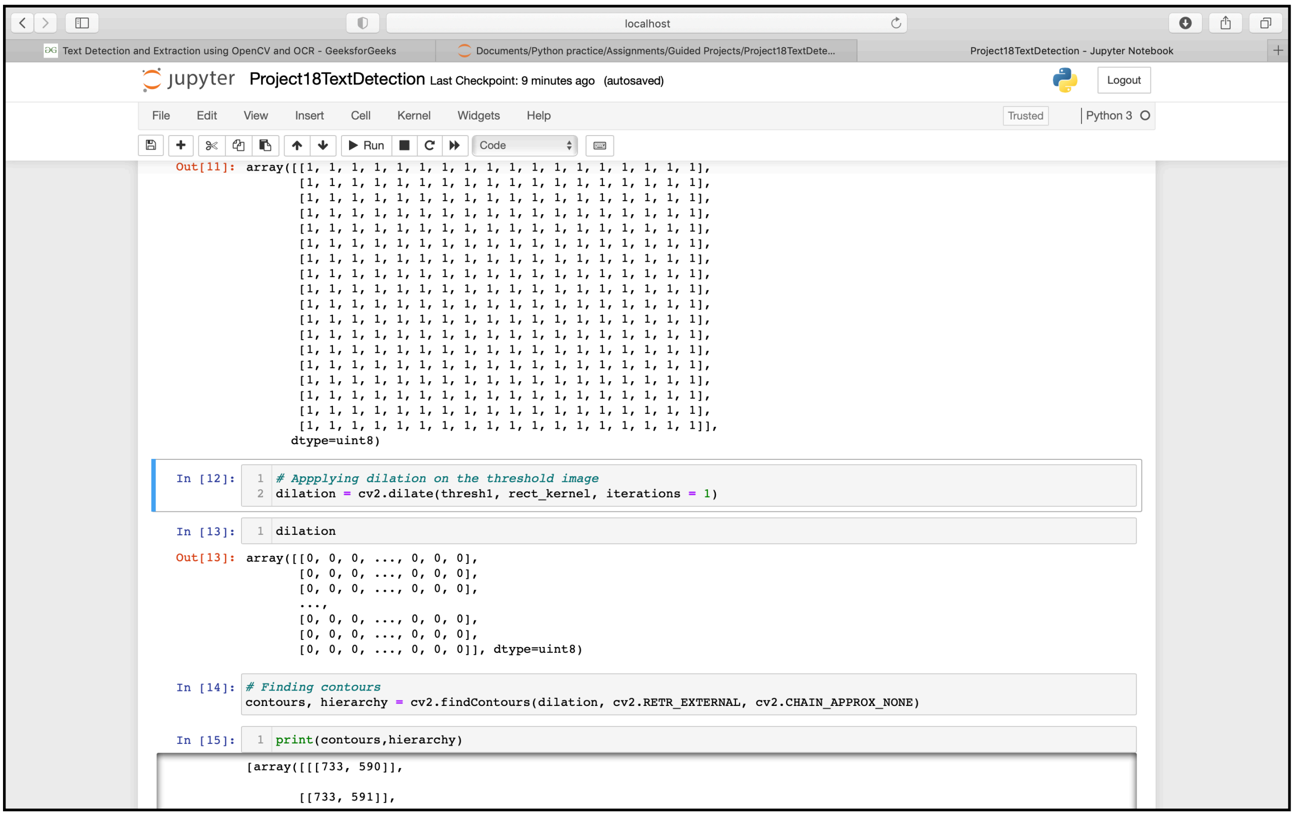
```
                ...,
                [  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254]],

               [[  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254],
                ...,
                [  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254]],

               [[  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254],
                ...,
                [  0, 242, 254],
                [  0, 242, 254],
                [  0, 242, 254]]], dtype=uint8)

In [7]:
# Step 1: Preprocessing the image

In [8]:
# Convert the image to gray scale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print(gray.shape)

(744, 1350)

In [9]:
# Performing OTSU threshold
ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY_INV)
print(ret,thresh1)

120.0 [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]

In [10]:
# Specify structure shape and kernel size.
```

Rectangle kernel specifications



```
In [8]:  # Convert the image to gray scale
         gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
         print(gray.shape)

         (744, 1350)

In [9]:  # Performing OTSU threshold
         ret, thresh1 = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU | cv2.THRESH_BINARY_INV)
         print(ret,thresh1)

         120.0 [[0 0 0 ... 0 0 0]
          [0 0 0 ... 0 0 0]
          [0 0 0 ... 0 0 0]
          ...
          [0 0 0 ... 0 0 0]
          [0 0 0 ... 0 0 0]
          [0 0 0 ... 0 0 0]]

In [10]: # Specify structure shape and kernel size.
         # Kernel size increases or decreases the area
         # of the rectangle to be detected.
         # A smaller value like (10, 10) will detect
         # each word instead of a sentence.
         rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (18, 18))

In [11]: rect_kernel

Out[11]: array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
```

Applying dilation and finding contours



```
Out[11]: array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
                [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]],
               dtype=uint8)

In [12]: # Appplying dilation on the threshold image
         dilation = cv2.dilate(thresh1, rect_kernel, iterations = 1)

In [13]: dilation

Out[13]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)

In [14]: # Finding contours
         contours, hierarchy = cv2.findContours(dilation, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

In [15]: print(contours,hierarchy)

         [array([[[733, 590]],

                [[733, 591]],
```

Copy the image

```
[array([[[733, 590]],

        [[733, 591]],

        [[733, 592]],

        ...,

        [[736, 590]],

        [[735, 590]],

        [[734, 590]]], dtype=int32), array([[[62, 30]],

        [[62, 31]],

        [[62, 32]],

        ...,
```

In [16]:
```
# Creating a copy of image
img2 = img.copy()
```

In [17]:
```
img2
```

Out[17]:
```
array([[[  0, 242, 254],
        [  0, 242, 254],
        [  0, 242, 254],
        ...,
        [  0, 242, 254],
        [  0, 242, 254],
        [  0, 242, 254]],

       [[  0, 242, 254],
        [  0, 242, 254],
        [  0, 242, 254],
        ...,
        [  0, 242, 254],
        [  0, 242, 254],
        [  0, 242, 254]],

       [[  0, 242, 254],
```

Creating an empty text file and using image to string from pytesseract

```
        ...,
        [  0, 242, 254],
        [  0, 242, 254],
        [  0, 242, 254]],

       [[  0, 242, 254],
        [  0, 242, 254],
        [  0, 242, 254],
```

In [18]:
```
# A text file is created and flushed
file = open("recognized.txt", "w+")
file.write("")
file.close()
```

In [24]:
```
dir(pytesseract)
```

Out[24]:
```
['ALTONotSupported',
 'Output',
 'TSVNotSupported',
 'TesseractError',
 'TesseractNotFoundError',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 '__version__',
 'get_languages',
 'get_tesseract_version',
 'image_to_alto_xml',
 'image_to_boxes',
 'image_to_data',
 'image_to_osd',
 'image_to_pdf_or_hocr',
 'image_to_string',
 'pytesseract',
 'run_and_get_output']
```

Applying OCR on the cropped text and appending in the text file and closing file