Exploratory data analysis (EDA)

Student: Ammar Alhashimi (13836676)
Email: ammar.alhashmi@student.uva.nl
Supervisor UvA: Maarten Marx (M.J.Marx@uva.nl)

1 Introduction

This paper offers an exploratory data analysis(EDA) for a specific set of data. EDA aims to provide insights into the data at hand. E.g. EDA aims to check if there are outliers or disclose underlying structure in the data at hand. In this case, the dataset consists of 365 PDF files which contain in total 32 thousand textual pages. The documents can be downloaded from the dutch government website webcovid19 (https://wobcovid19.rijksoverheid.nl/). The goal is to use the dataset to build a classifier that detects whitewashed regions. Whitewashed regions refer to the regions in a pdf document that have been censored as will be shown in the next part in figure 1.

2 Performing EDA

This section provides insights and descriptions of some of the samples of the dataset. First, in order to perform preprocessing approaches, it is preferable that the PDF document should be turned into images. After choosing some pdf samples and turning them into images

First, the needed packages should be imported. And after that, the image is read as shown below in figure 1.

Figure 1

In figure 1, the yellow regions are the whitewashed regions. The reason behind whitewashing is to hide sensitive information from the public such as names or signatures, it could also be any kind of informative text that is not meant yet to be seen by the public.

As shown below in figure 2, hiding information can be done in different ways.

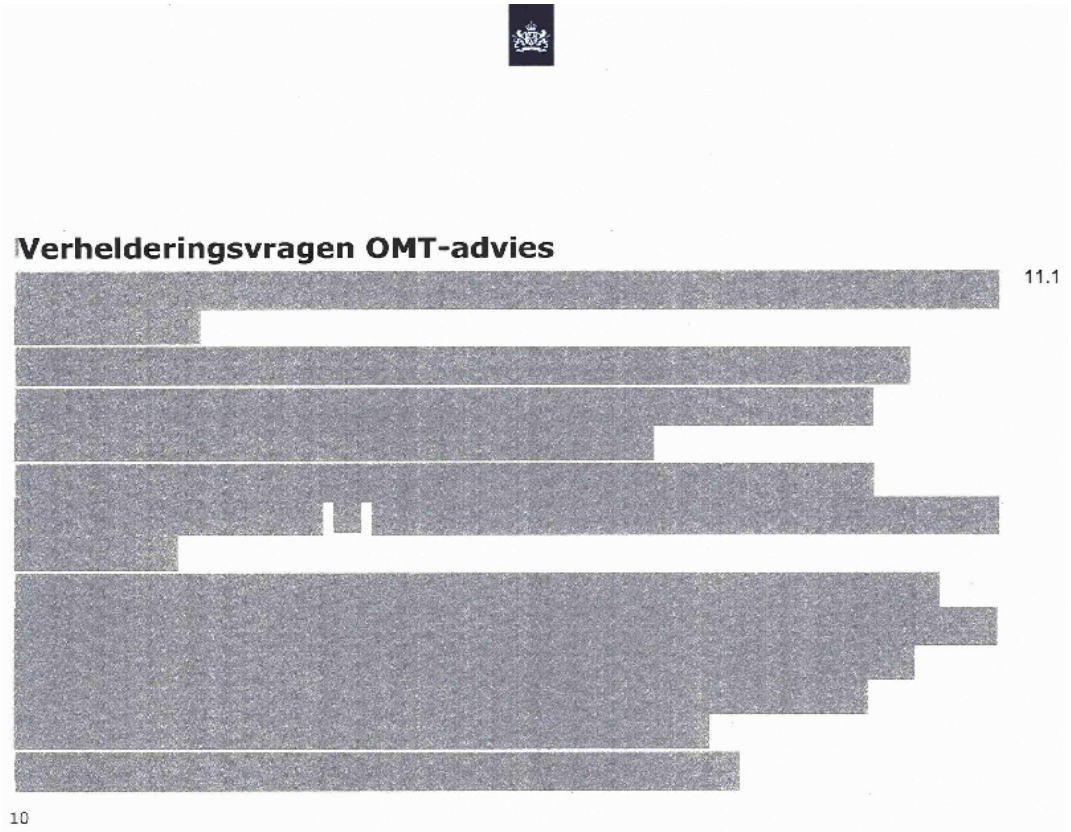

Figure 2

After taking a look at some samples of the dataset that include whitewashed information, it is time to check if it is possible to perform some data cleaning as part of the preprocessing. The script below shown in 'code 1' extracts the whitewashed regions.

```
# Convert BGR to HSV
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
# Threshold the HSV image to get only specified colors
mask = cv2.inRange(hsv, light_yelloy, dark_yellow)
#bitwise
output = cv2.bitwise_and(frame,frame, mask= mask)
#show output
cv2.imshow('output',output)
cv2.waitKey(0)
```

Code 1

Figure 3 below shows the output of the code above. This way the image has more value when detecting the whitewashed regions.

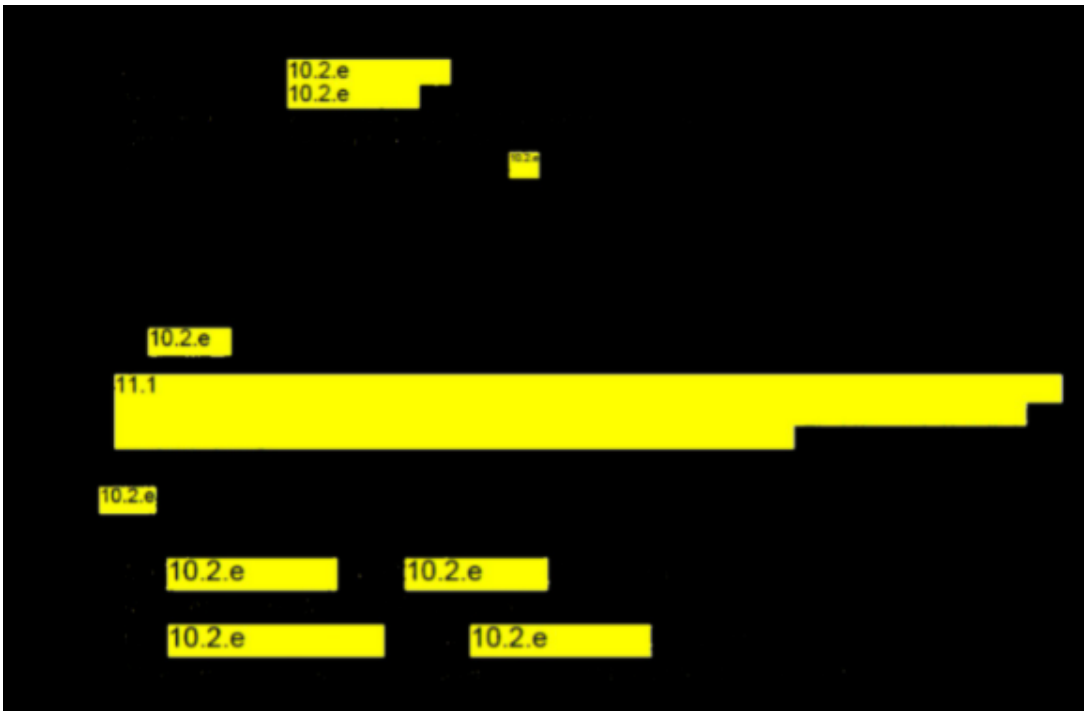

Figure 3

The next step would be to extract features from the last preprocessed image (figure 3). Figure 4 demonstrates extracting the numbers in the image such as 10.2.e. It would be helpful to extract this feature because the numbers refer to articles in the Dutch law.

```
# Looping through the identified contours
# Then rectangular part is cropped and passed on
# to pytesseract for extracting text from it
# Extracted text is then written into the text file

# note, some of the text(10.2) does not get recognised
for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)

    # Drawing a rectangle on copied image
    rect = cv2.rectangle(im2, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # Cropping the text block for giving input to OCR
    cropped = im2[y:y + h, x:x + w]

    # Open the file in append mode
    file = open("recognized.txt", "a")

    # Apply OCR on the cropped image
    text = pytesseract.image_to_string(cropped)
    print ("ROI output",text)

    # Appending the text into file
    file.write(text)
    file.write("\n")

    # Close the file
    file.close
```

Output of the pytesseract:

```
10.2.e

11.1
```

Figure 4

Another step that could be helpful is extracting the coordinates of the whitewashed regions in the image and that is demonstrated below in Figure 5

```
#get coordinates
X,Y = np.where(np.all(frame== light_yelloy, axis=2))
X1,Y1 = np.where(np.all(frame== dark_yellow, axis=2))

zippedLight = np.column_stack((X,Y))
zippedDark = np.column_stack((X1,Y1))

print("the coordinates are:.....")
print("light color:  ",zippedLight)
print("dark color:  ",zippedDark)
```

Coordinates of the colored regions

```
light color: [] dark color: [[ 301 718] [ 301 744] [ 918 662] [1026 307] [1202 235] [1373 755] [1579 588] [1657 344] [1658 700]]
```

Figure 5.

Having these features extracted from all images could help when creating and training a model to detect whitewashed regions fast in a lot of images.