**CSC2062 AIDA – Assignment 1**

**Section 1**

I used the code snippets provided in the practical sessions as well as a number of basic loops and conditions to read through every file in the folder where I store my PGM files. This information was then converted to a numpy array. Once this process had finished it was saved to a .csv file of the same name as the original file.

I was able to easily install and use GIMP to create my dataset and export it to PGM Files. I tried my best to keep the images fairly neat in order to prevent anomalous results, however, there remained a number of inconsistencies as I was using a computer mouse. I had struggled for the first number of days to successfully get Jupyter Notebook working on my home computer due to an issue with anaconda being unable to process file paths containing a space character. Once this had been overcome I was able to set it up relatively easily.

The coding of this section was straightforward, I was only hindered by my small logical errors such as mis-inputting the size of the 2D matrix causing my output .csv file to be too small. Many of these minor issues were caused by an unfamiliarity with python and lack of experience with file conversion. The greatest issue I experienced while coding this was the errors caused by not having the .pgm files in their own folder. This led to the creation of the try catch block I used to discover which I have left in the notebook in the case I needed to use it for further debugging.

**Section 2**

In my code I iterated through each file and used for loop nesting to get the value of each coordinate. I then used a number of if statements to apply the conditions specified in the assignment sheet. Once I had successfully stored all of the values I required into variables I assigned them to the respective row in the numpy array and once the program had successfully iterated through every file in the folder it saved it as 40403863\_features.csv.

My custom feature was ‘densest\_quadrant’ which finds the midpoint of the drawn shape and goes through every pixel of the square and finds out where it is in relation to the midpoint. I then find the quadrant with the greatest amount of black pixels. Judging by the features csv this will prove very useful in differentiating between the letters.

Section 2 took me significantly longer to complete than section 1 due to substantially more difficult processes being required. I began by creating all of the necessary code such as numpy array and loading the first row with the feature names. I was now familiar with iterating through files having done it in the first section so this also spawned no issues. The logic required for automatically filling in the rest of the data was the cause of most of my problems in this section. There were a number of problems such as the neigh\_1 feature for which my for loop was too short and didn’t encompass all of the neighbours of the pixel, leading to almost of the values being 0. The fix for this caused all of the values to go to zero as the pixel was now counting itself as a neighbour. Other than this there were very few logical errors that I noticed and had to correct.

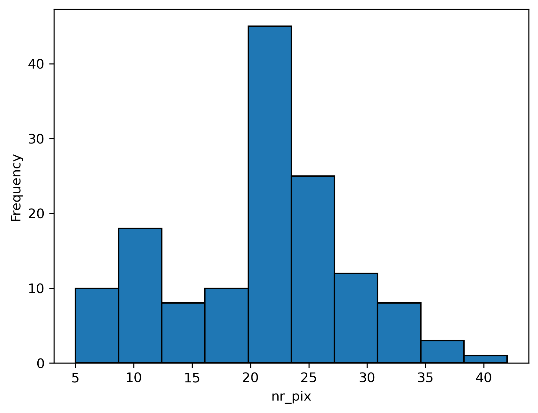
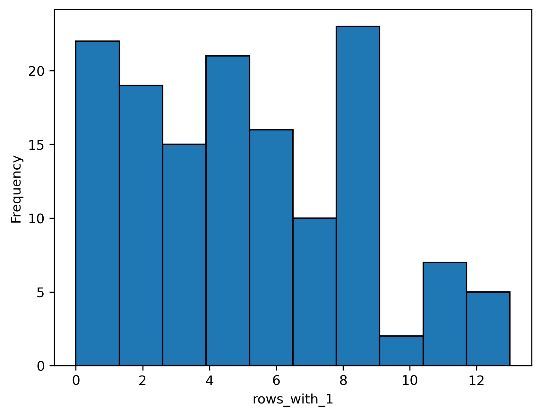
I found the connected\_areas and eyes features particularly because of the use of scipy. I spent a number of hours researching which libraries to use and how to use them. To learn this I used SciPy’s own documentation on the scipy.org website.

**Section 3**

**Section 3.1**

This part provided very few challenges as all of the necessary code was available through the practicals. The results are as follows.

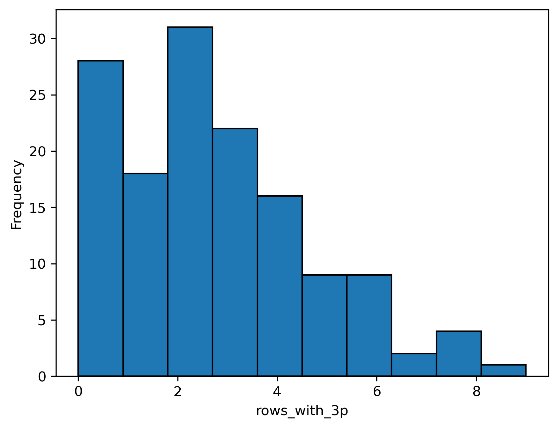
A graph of a number of blue bars

AI-generated content may be incorrect.

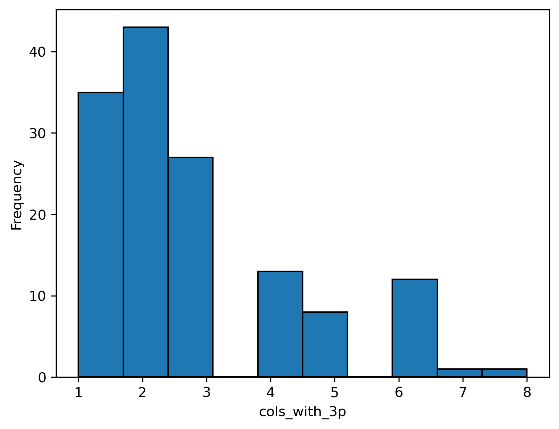
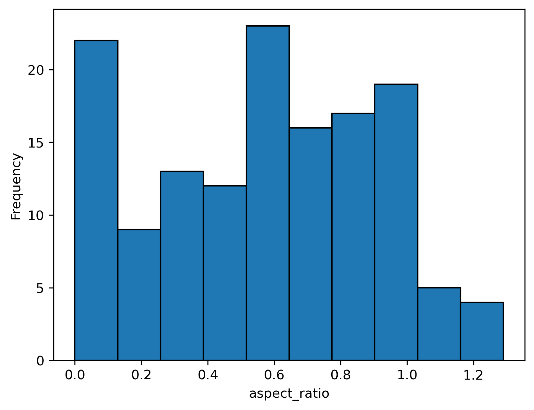
Left-skewed (-0.142 skewness) bimodal histogram peaking at 20-24. Given the concentration around the centre, it is likely to be useful in differentiating letters and non-letters

Right-skewed (0.404 skewness) multimodal histogram. This is not likely to be useful given it’s unpredictability caused by its uneven distribution

Right-Skewed (1.021 skewness) multimodal histogram which has been log transformed in accordance with the recommendations in the lecture slides. The skew being this high suggest that this feature may be useful in discriminating letters from non-letters



A right-skewed (0.752 skewness) bimodal histogram suggesting most images have a low amount of rows with 3 pixels. Given the value of the skew being large this statistic could be used to differentiate between characters



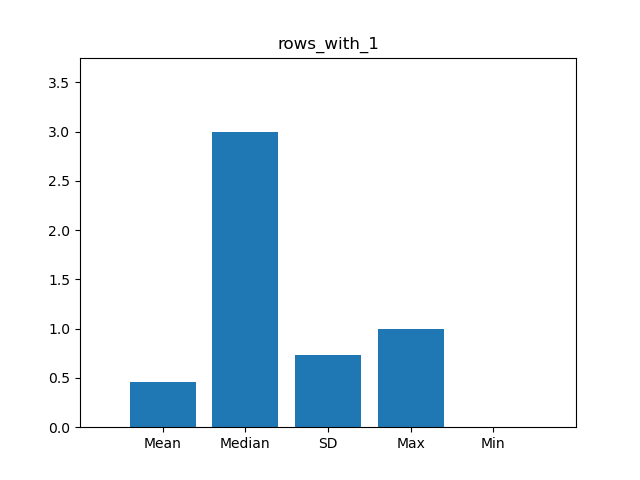
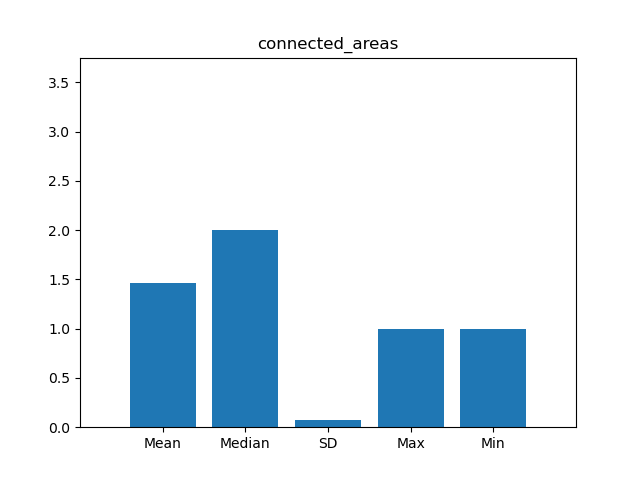
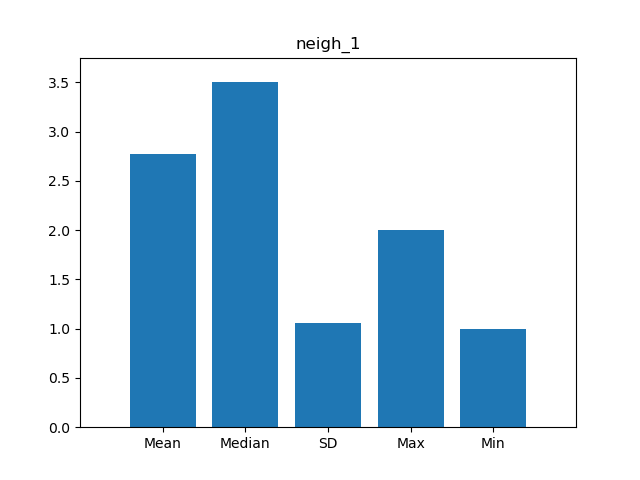
A left-skewed (-0.110 skewness) multimodal histogram peaking in the centre. The large spike on the left is likely caused by narrow characters such as ‘i’ and ‘xclaim’. Most values are less than 1, indicating they are taller than they are wide. However, the negative skew indicates that most values lie between 0.6 and 1.2. These factors make this a key value in discriminating between characters.

Right-skewed (0.973 skewness) multimodal histogram comprising of 3 columns separated by values of zero. This may prove useful as the separated columns may encompass different characters and could help to distinguish letters and non-letters. Alternatively, the separate columns could be outliers and may not be useful

All graphs presented in section 3.1 are either bimodal or multimodal. This could be a key indicator in each graph of distinct categories of characters. These categories could be used to distinguish between characters and more specifically letters and non-letters which we look at later.

**Section 3.2**

|  |  |
| --- | --- |
| **Summary Statistics** | **Analysis** |
| **A white background with black text  AI-generated content may be incorrect.** | For nr\_pix, the difference in each statistic is significant, likely resulting from letters such as ‘d’ having more complicated structures than non-letters like ‘xclaim’. Values less than 8 or greater than 34 are likely letters as these are the minimum and maximum values of non-letters. This is likely useful in distinguishing between them |
| **A black and white text  AI-generated content may be incorrect.** | Rows\_with\_1 has a significant difference between letters and non-letters. Likely caused by letters such as ‘i’ and ‘b’ which all have single pixel rows, images such as ‘smiley’ are often vertically symmetrical and as such, rows have more than 1 pixel. Based on maximum values, images with values of 13 are likely to be non-letters |
| A black and white text  AI-generated content may be incorrect. | Cols\_with\_1 has a large separation when comparing letters and non-letters, this is likely due to ‘smiley’ and ‘sad’ images which contain high amounts of single pixel columns. Although the minimums are equal. Based on maximum values, values over 4 are likely to be non-letters |
|  | Rows\_with\_3p has a significant separation between letters and non-letters likely caused by the structural difference between letters (eg. ‘a’) and non-letters (eg. ‘xclaim’) suggesting this will be a useful metric. We can also infer from the maximum letters that if the value is greater than 6 it is likely to be a letter. |
|  | Cols\_with\_3p has a similar average difference to rows\_with\_3p despite a lower total. From this we can infer that it will be equally useful. From the maximum value we know values above 4 are likely to be a letter. |
|  | While the average difference between letters and non-letters for the aspect ratio is small it should be noted that the average values are also small. Therefore, this metric is likely to be useful in distinguishing between letters and non-letters. |
|  | The difference in the number of pixels with 1 neighbour is likely due to more separate parts found in non-letters such as ‘smiley’ and ‘sad’. The maximum value here could prove highly useful as any value greater than 4 is definitely a non-letter. |
|  | I have grouped these values together as they are all similar in definition and result but will specify individual points.  For the values no\_neigh\_above and no\_neigh\_below, the only differences lie in the mean and the standard deviation which have very little variance. Given the size of the values relative to the average difference and the fact that the maximum and minimum are equal, it is unlikely that these will be useful.  Similarly, the values no\_neigh\_left and no\_neigh\_right are very similar. They have noticeably more variation in their standard deviation which could prove useful in differentiating between letters and non-letters.  Their most noticeable asset is the difference in maximum and minimum. If the no\_neigh\_left is greater than 17 or less than 7, it is likely that it is a letter. Similarly, if no\_neigh\_right is greater than 16 or less than 7, it is likely to be a letter. As such these letters are likely to be useful in differentiating between letters and non-letters  These features are not as similar as the previous pairs, with the number of pixels with no horizontal neighbours being greater than the number with no vertical neighbours. This further supports the claim made previously that the letters are taller than they are wide.  The maximum and minimum of both of no\_neigh\_horiz and no\_neigh\_vert are very close and as such cannot be reliably used.  For these reasons these values are unlikely to be useful |
|  | The average difference between the values connected\_areas is low however seeing as the maximum value is 3 this may be significant.  It should be noted that when written in the standard format have 3 connected areas. Meaning that if there are 3 connected areas that the character is almost certainly a non-letter  As such this feature is highly likely to be useful for differentiating letters and non-letters |
|  | The number of eyes is highly likely to be a useful feature as there are no non-letters in the standard format that contains an eye. This means if the character has an eye there is a very high likelihood that it is a letter. |
|  | My custom feature densest\_quadrant has a very small statistic separation. This in combination with the same maximum and minimum features means it may not be useful for distinguishing between letters and non-letters. |

The 3 Features that I have selected that I feel will be the most useful for differentiating between groups are, rows\_with\_1, aspect\_ratio and connected\_areas. I have plotted their values on the following histograms. I was hoping to put them all into the one histogram however they overlapped, I didn’t have the knowledge to fix this and was unable to find how to do it in the matplotlib documentation.