CSC2062 AIDA – Assignment 1

ZE HUEI LIM

40443486

12/03/2025

Introduction

In this assignment, I am required to:

(a) Create a dataset of handwritten letters and symbols (which you will use for your analyses and experiments in the rest of Assignment 1, and in Assignment 2).

(b) Perform feature engineering, i.e. calculate features (variables) from the handwritten letters and symbols which may be useful for identifying the handwritten symbols automatically.

(c) Perform statistical analysis of the datasets, using methods of statistical inference.

(d) Implement and evaluate introductory machine learning models that perform classification on the dataset.

Section 1

The 140 images containing of {a, b, c, d, e, f, g, h, i, j, sad, smiley, xclaim} are all hand drawn in GIMP application. OS, numpy, pandas, matplotlib and PIL.Image libraries are used in order to handle file directory, work with arrays for image processing, saves processed data as CSV file and opening PGM files. All files entered into input_folder will run through the code where it will open the file-> convert to Numpy array-> invert the pixels-> convert it to pandas-> Save data as CSV File.

Commented [LH1]: Methods to interact with Operating System such as opening a file, creating file .etc

Commented [LH2]: Used for working with arrays

Commented [LH3]: Supports in applying data into CSV files/ Excel Spreadsheets. With numpy it is much easier to input data.

Commented [LH4]: Data visualization library for generating plots, histograms, chart .etc with certain function

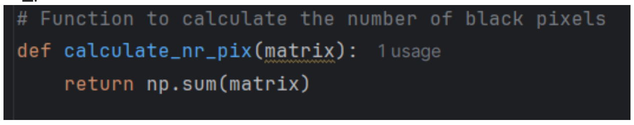
Commented [LH5]: Provides functions to load images and create images from program or files.

Section 2

Same libraries from Section 1 are used with an addition of scipy library to help calculate seprerate objects in my images. All features are coded with the required feature description in the start as functions to improve code readability and reducing complexity. Difficulty faced for feature index 14 and 15 as the demand is quite complicated where research in other libraries/ data structure and algorithms as many test runs were conducted. For my custom feature, COUNT NUMBER OF CORNERS was chosen by me as it is useful for determine different images symbols such as a and b for example where b has generally lesser corners at an average of 2.75 where a with an average of 5.25. The custom feature is calculated in a nested loop for x-axis and y-axis were if it reaches a black pixel it will check if pixel below AND pixel right is white; if condition meets corners will increase by 1.

Commented [LH6]: Library that provides algorithms such as t-test values and p-values

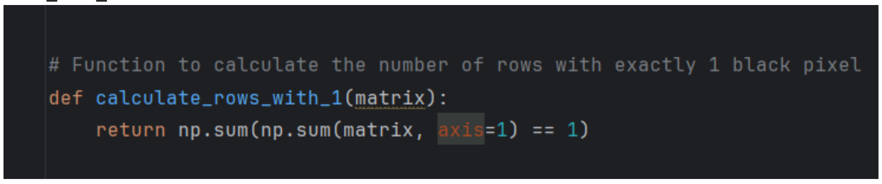
- nr_pix



```
# Function to calculate the number of black pixels
def calculate_nr_pix(matrix):  1 usage
    return np.sum(matrix)
```

Using sum function to calculate all black pixels.

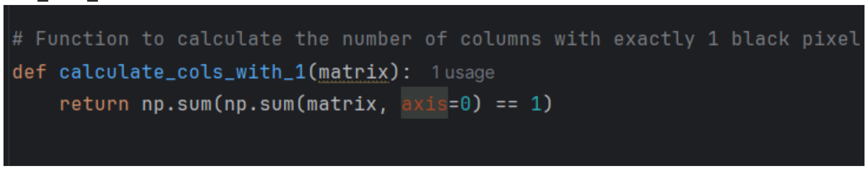
- rows_with_1



```
# Function to calculate the number of rows with exactly 1 black pixel
def calculate_rows_with_1(matrix):
    return np.sum(np.sum(matrix, axis=1) == 1)
```

Using nested recursion uses sum function to calculate all rows with 1 black pixel.

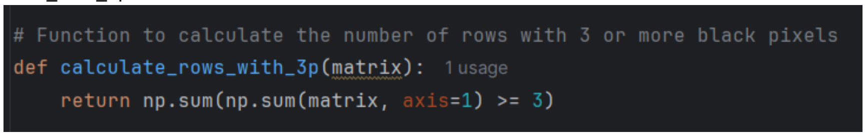
- cols_with_1



```
# Function to calculate the number of columns with exactly 1 black pixel
def calculate_cols_with_1(matrix):  1 usage
    return np.sum(np.sum(matrix, axis=0) == 1)
```

Using nested recursion uses sum function to calculate all columns with 1 black pixel.

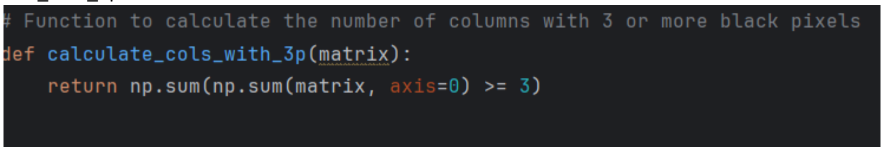
- rows_with_3p



```
# Function to calculate the number of rows with 3 or more black pixels
def calculate_rows_with_3p(matrix):  1 usage
    return np.sum(np.sum(matrix, axis=1) >= 3)
```

Using nested recursion uses sum function to calculate all rows with 3 black pixel or more.

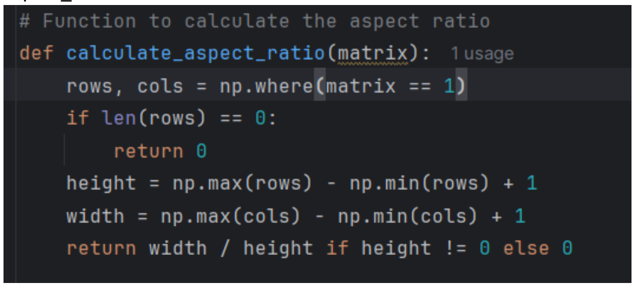
- cols_with_3p



```
# Function to calculate the number of columns with 3 or more black pixels
def calculate_cols_with_3p(matrix):
    return np.sum(np.sum(matrix, axis=0) >= 3)
```

Using nested recursion uses sum function to calculate all columns with 3 black pixel or more.

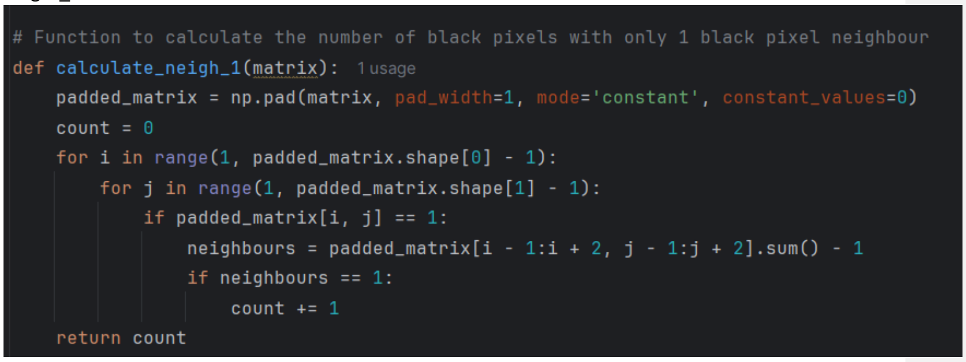
- aspect_ratio



```
# Function to calculate the aspect ratio
def calculate_aspect_ratio(matrix):  1 usage
    rows, cols = np.where(matrix == 1)
    if len(rows) == 0:
        return 0
    height = np.max(rows) - np.min(rows) + 1
    width = np.max(cols) - np.min(cols) + 1
    return width / height if height != 0 else 0
```

Find the maximum height and width difference then returns width divided height.

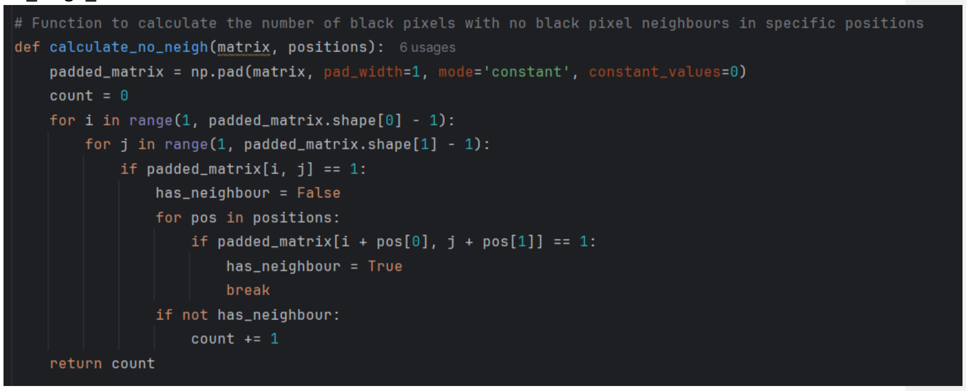
- neight_1



```
# Function to calculate the number of black pixels with only 1 black pixel neighbour
def calculate_neigh_1(matrix):  1 usage
    padded_matrix = np.pad(matrix, pad_width=1, mode='constant', constant_values=0)
    count = 0
    for i in range(1, padded_matrix.shape[0] - 1):
        for j in range(1, padded_matrix.shape[1] - 1):
            if padded_matrix[i, j] == 1:
                neighbours = padded_matrix[i - 1:i + 2, j - 1:j + 2].sum() - 1
                if neighbours == 1:
                    count += 1
    return count
```

Uses numpy function .pad() to find surrounding pixels and uses nested for loops for rows and column. If only one black pixel is found, count increases.

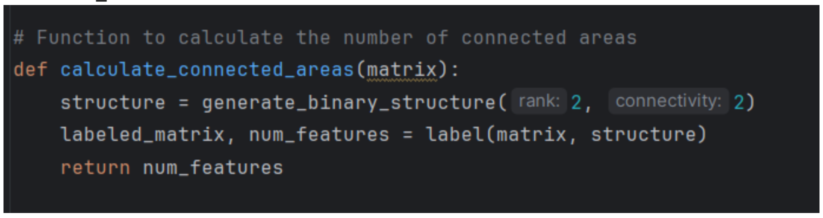
- no_neigh_above



```
# Function to calculate the number of black pixels with no black pixel neighbours in specific positions
def calculate_no_neigh(matrix, positions):  6 usages
    padded_matrix = np.pad(matrix, pad_width=1, mode='constant', constant_values=0)
    count = 0
    for i in range(1, padded_matrix.shape[0] - 1):
        for j in range(1, padded_matrix.shape[1] - 1):
            if padded_matrix[i, j] == 1:
                has_neighbour = False
                for pos in positions:
                    if padded_matrix[i + pos[0], j + pos[1]] == 1:
                        has_neighbour = True
                        break
                if not has_neighbour:
                    count += 1
    return count
```

Similar to neightbour however only dedicates to one position to check if there is no black pixel

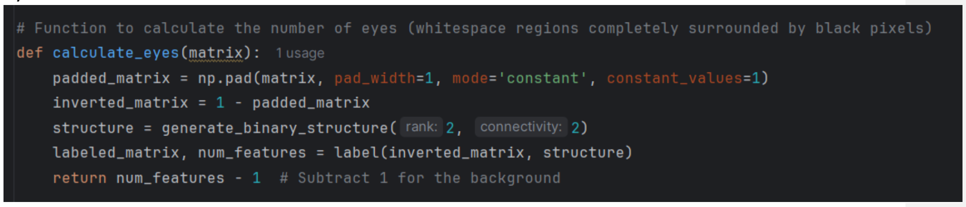
- no_neigh_below- same as no_neigh_above, but difference area
- no_neigh_left- same as no_neigh_above, but difference area
- no_neigh_right- same as no_neigh_above, but difference area
- no_neigh_horiz- same as no_neigh_above, but difference area
- no_neigh_vert- same as no_neigh_above, but difference area
- connected_areas



```
# Function to calculate the number of connected areas
def calculate_connected_areas(matrix):
    structure = generate_binary_structure( rank: 2,  connectivity: 2)
    labeled_matrix, num_features = label(matrix, structure)
    return num_features
```

Uses scipy function generate_binary_structure() to calculate connect black pixels. Hence return number of feature = number of connected areas.

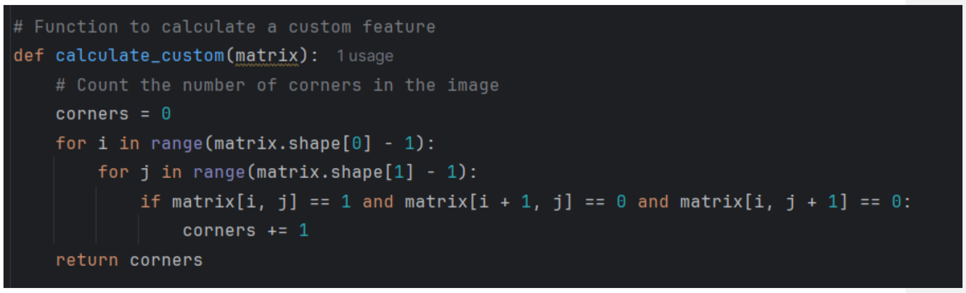
- eyes



```python
# Function to calculate the number of eyes (whitespace regions completely surrounded by black pixels)
def calculate_eyes(matrix):  1 usage
    padded_matrix = np.pad(matrix, pad_width=1, mode='constant', constant_values=1)
    inverted_matrix = 1 - padded_matrix
    structure = generate_binary_structure( rank: 2,  connectivity: 2)
    labeled_matrix, num_features = label(inverted_matrix, structure)
    return num_features - 1  # Subtract 1 for the background
```

Similar to connected areas however uses numpy pad() to find surroundings and return feature minus 1 as it also adds up our background.

- custom



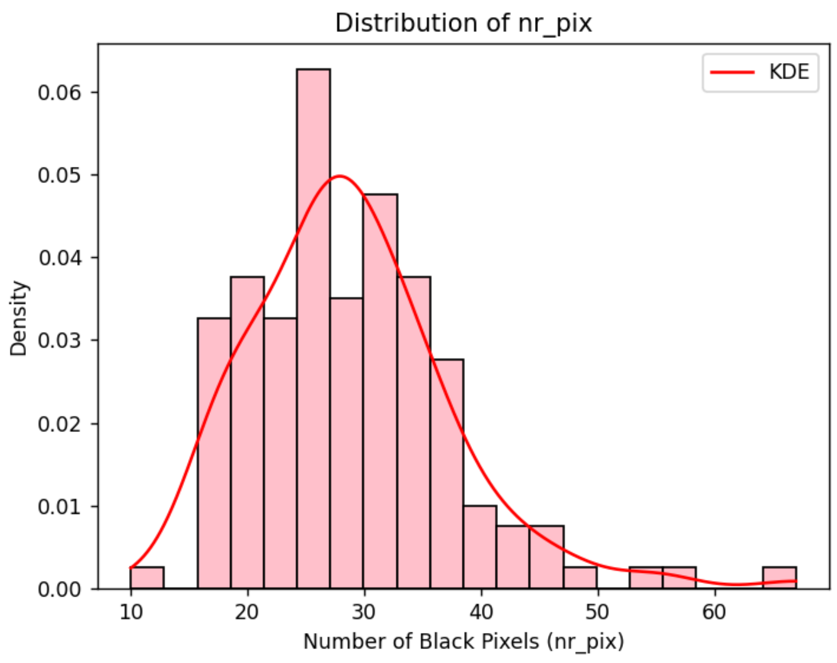
```python
# Function to calculate a custom feature
def calculate_custom(matrix):  1 usage
    # Count the number of corners in the image
    corners = 0
    for i in range(matrix.shape[0] - 1):
        for j in range(matrix.shape[1] - 1):
            if matrix[i, j] == 1 and matrix[i + 1, j] == 0 and matrix[i, j + 1] == 0:
                corners += 1
    return corners
```

Uses nested for loops for rows and column to find sharp edges for example an edge of a square (sharp edges) and increases corners by 1 every time it is found. Return to number of corners found.

Section 3

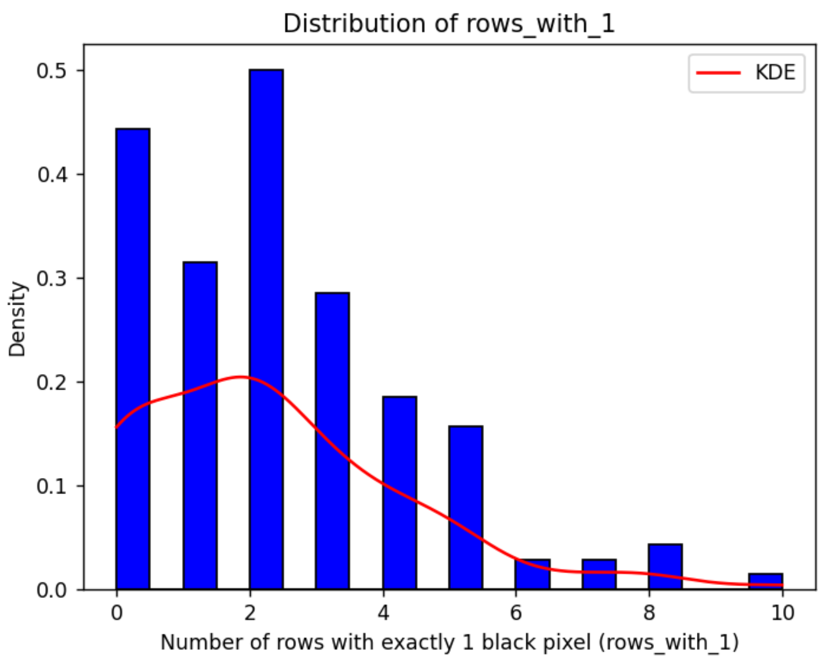
In this section, I must perform statistical analyses of the feature data, in order to explore which features are important for distinguishing between different kinds of handwritten symbols.

Section 3.1



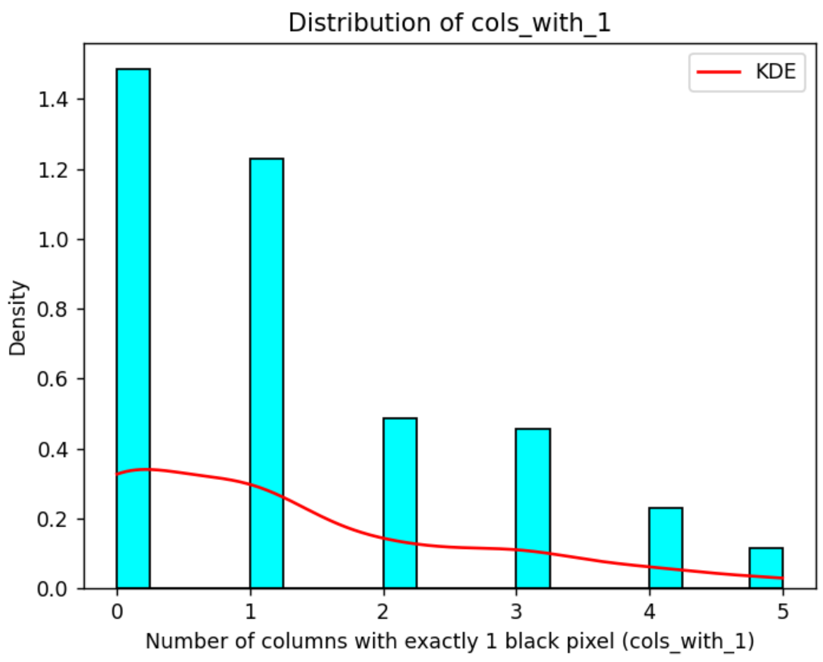
SHAPE OF DISTRIBUTION: Right Skewed Distribution

INTERESTING PATTERNS: Most images have small number of black pixels however few images have large number of black pixels. Right-side of graph is significantly flatter.



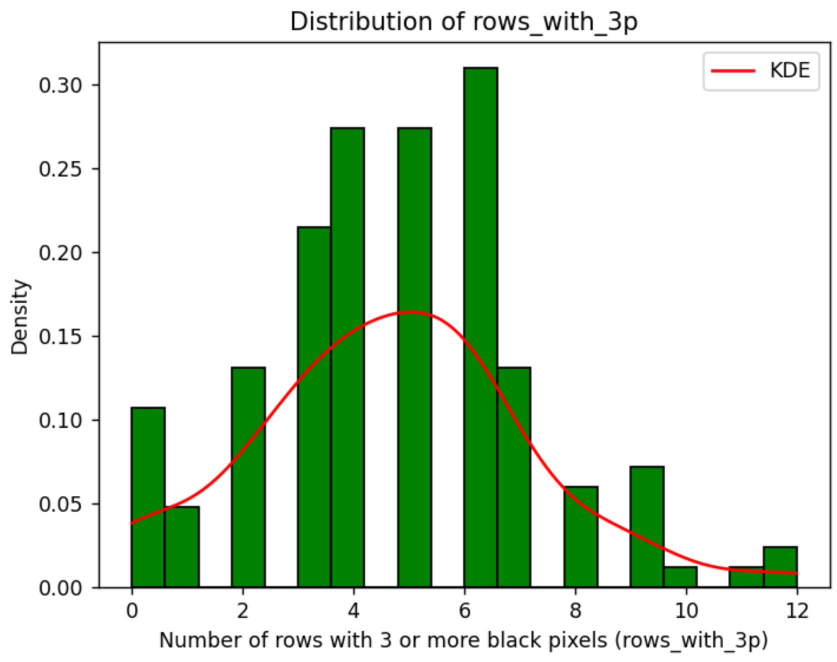
SHAPE OF DISTRIBUTION: Right Skewed Distributions

INTERESTING PATTERNS: Most images have few or no rows with a single black pixel, while only a small number of images contain a high number of such rows. Right-side of graph is significantly flatter.



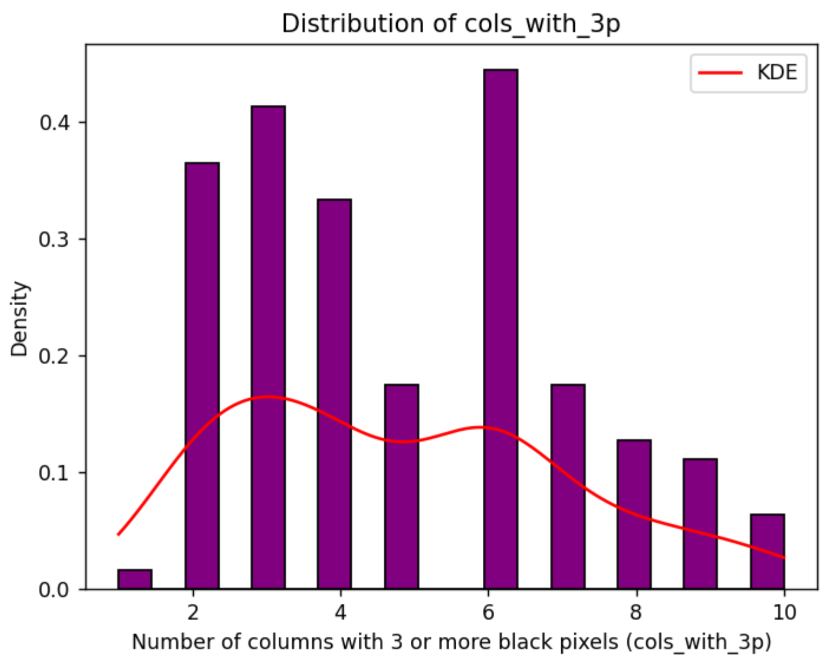
SHAPE OF DISTRIBUTION: Uniform Distribution

INTERESTING PATTERNS: Number of columns with exactly 1 black pixel is evenly distributed. Graph have almost a straight line.



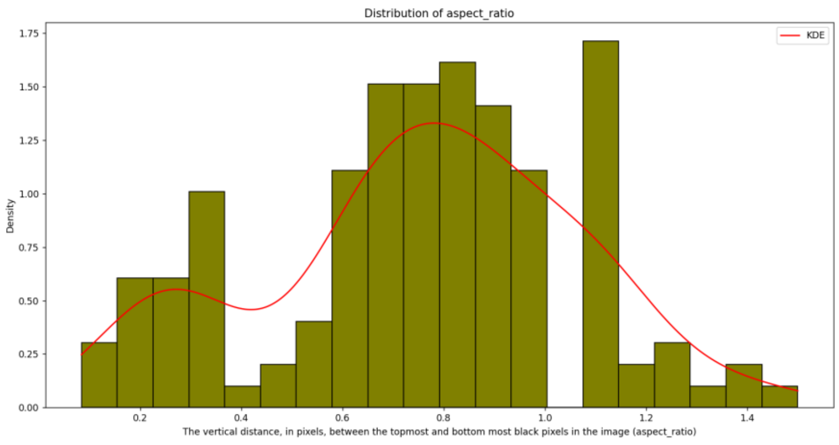
SHAPE OF DISTRIBUTION: Right Skewed Distribution

INTERESTING PATTERNS: Most images have low amount of rows with 3 or more black pixels whereas a few images have large amount of rows with 3 or more black pixels. Right-side of graph is much flatter.



SHAPE OF DISTRIBUTION: Bimodal Distribution

INTERESTING PATTERNS: Have 2 peaks where most images have low number of columns with 3 or more black pixels and another at its average. Graph have 2 humps providing 2 peak points.



SHAPE OF DISTRIBUTION: Bimodal and Right Skewed Distribution

INTERESTING PATTERNS: it has 2 peaks at around 0.2-0.3 and another at 0.8-1.0. Not only that another sharp peak occur around 1.0 however the KDE line smooths out the histogram. Graph have small hump at beginning and a peak hump.

Section 3.2

(a) the full set of letters

	index	nr_pix	rows_with_1	cols_with_1	rows_with_3p	cols_with_3p	aspect_ratio	neigh_1
mean	4.5	26.525	2.85	1.125	4.55	4.325	0.784766	1.15
median	4.5	26	3	1	4	4	0.727273	1
std	2.305744	7.70989	2.289768	1.362936	2.272009	1.763017	0.285776	1.159441

	no_neigh_above	no_neigh_below	no_neigh_left	no_neigh_right	no_neigh_horiz
mean	7.175	7.2125	7.9625	7.775	6.775
median	7	7	8	8	7
std	2.791375	2.910886	3.099382	3.085183	3.471311

	no_neigh_vert	connected_areas	eyes	custom
mean	6.6375	1.2	0.325	3.6
median	7	1	0	3
std	3.959426	0.402524	0.47133	1.547558

(b) the full set of non-letters

	index	nr_pix	rows_with_1	cols_with_1	rows_with_3p	cols_with_3p	aspect_ratio	neigh_1
mean	10.5	31.75	1.516667	1.45	4.916667	5.5	0.715173	1.233333
median	10.5	32	1.5	1	5	6	0.851648	1
std	5.814943	9.047193	1.200165	1.358277	2.695204	2.626462	0.336249	1.681068

	no_neigh_above	no_neigh_below	no_neigh_left	no_neigh_right	no_neigh_horiz
mean	10.38333	10.76667	9.733333	9.283333	3.85
median	11	12	9	9	3
std	4.847126	5.241188	1.858254	1.766848	3.139821

	no_neigh_vert	connected_areas	eyes	custom
mean	5.433333	3.333333	0.066667	5.133333
median	6	4	0	5.5
std	3.928888	0.950765	0.251549	2.310379

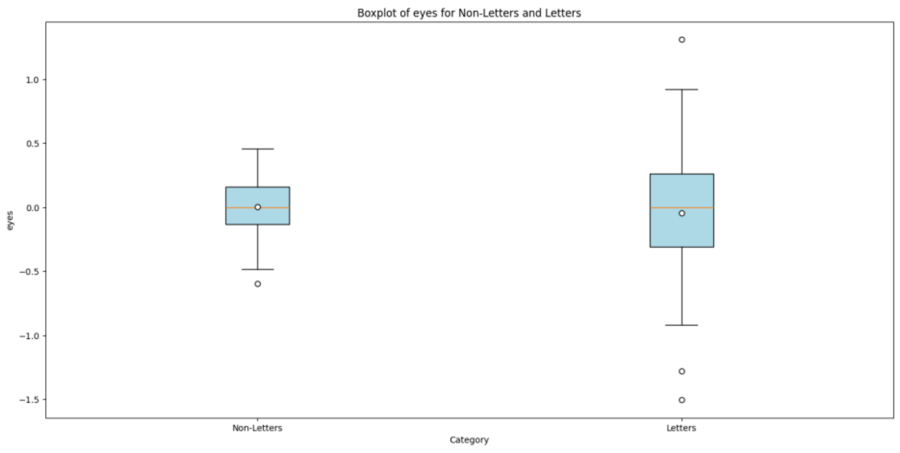
The summary statistics, including the mean, median and standard deviation for full set of letter and non-letters are presented above. These will be compared to identify which feature may be useful for differentiation.

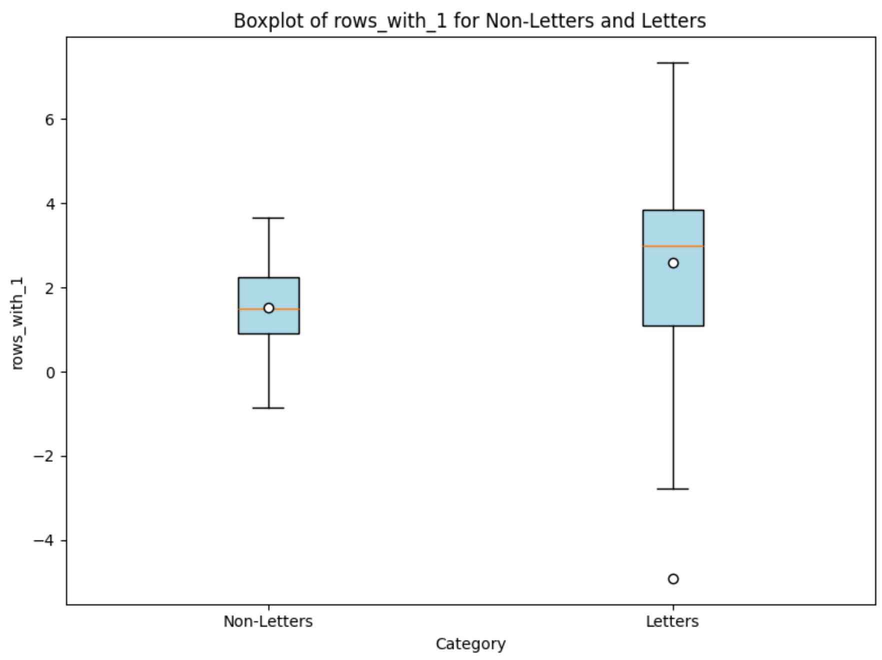
Mean-average value for each feature

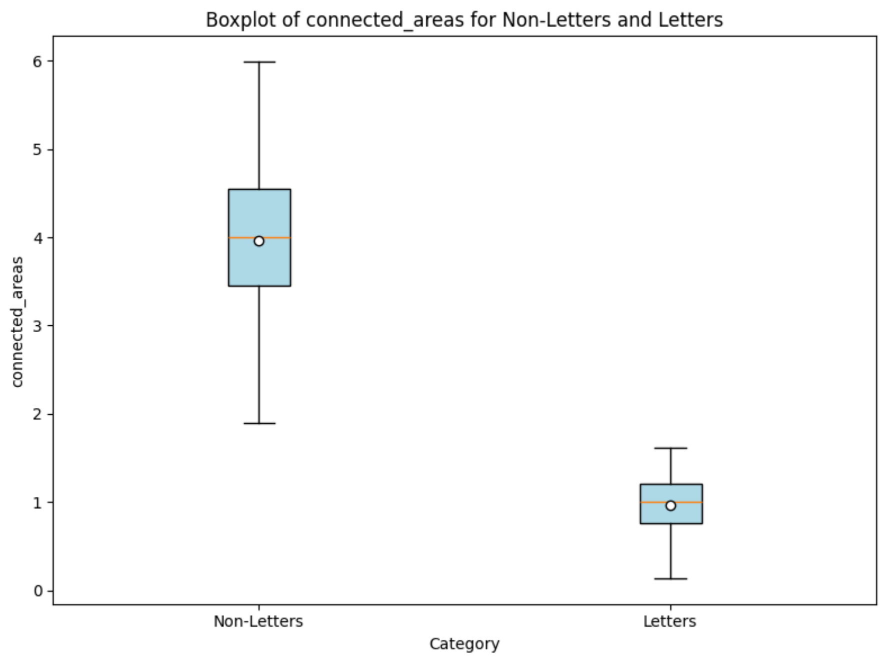
Median-middle value for each feature

Standard deviation(std)- measures variability for each feature

In my opinion, rows_with_1, connected areas and eyes as their mean difference is the most compared to others and also have low standard deviation. Not only that, using Cohen,s d formula it provides a large Cohen,s d value. Below are boxplot for better visualisation:




Boxplot of rows_with_1 for Non-Letters and Letters
rows_with_1
6
4
2
0
−2
−4
Non-Letters
Letters
Category


Boxplot of connected_areas for Non-Letters and Letters
connected_areas
6
5
4
3
2
1
0
Non-Letters
Letters
Category

Section 3.3

Using t-tests to find significant difference:

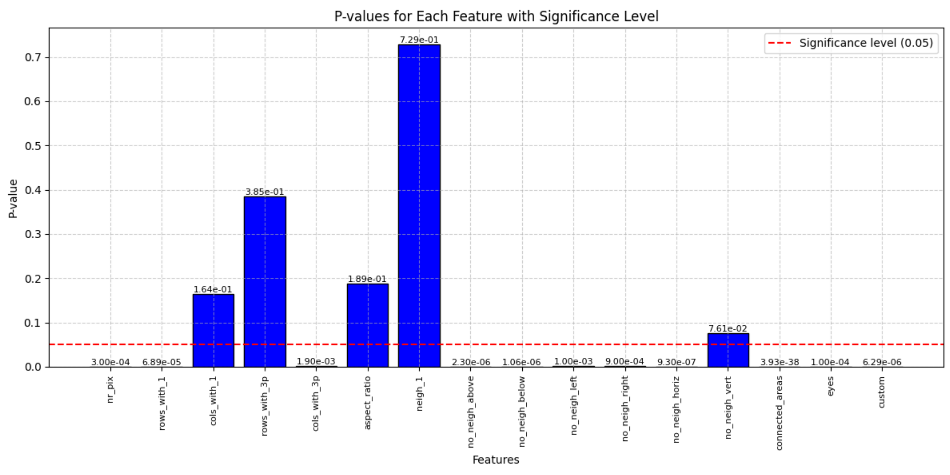
	nr_pix	rows_with_1	cols_with_1	rows_with_3p	cols_with_3p	aspect_ratio	neigh_1
t-stat	-3.6825	4.1049	-1.3983	-0.8721	-3.1639	1.3215	-0.347
p-value	0.0003	6.89E-05	0.1643	0.3847	0.0019	0.1885	0.7291

no_neigh_above	no_neigh_below	no_neigh_left	no_neigh_right	no_neigh_horiz	no_neigh_vert
-4.9325	-5.1086	-3.926	-3.39	5.1376	1.7866
2.30E-06	1.06E-06	0.001	0.0009	9.30E-07	0.0761

connected_areas	eyes	custom
-18.0444	3.8517	-4.6974
3.93E-38	0.0001	6.29E-06

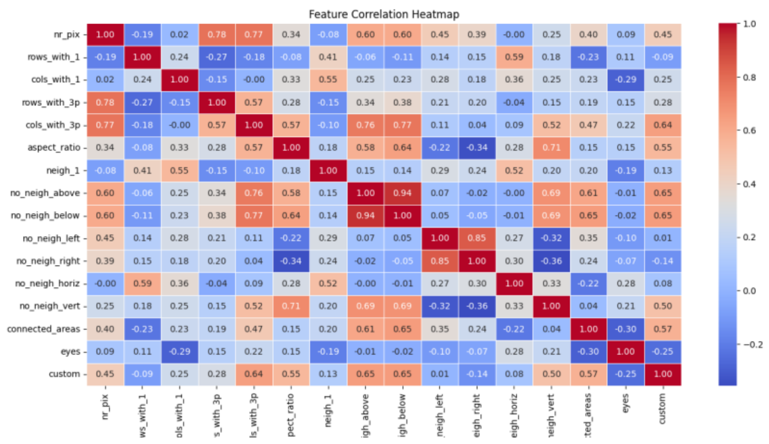
p-values are obtained by using scipy.stats library with stats.ttest_ind() which generate the t-value and p-value.

With these statistics, (nr_pix, rows_with_1, cols_with_3p, no_neigh_above, no_neigh_below, no_neigh_left, no_neigh_right, no_neigh_horiz, connected_areas, eyes and custom) are features that are most useful to discriminate whether image is a letter or a non-letter. It is because with the given significance level of 0.05 for analysis, we can compare feature's p-value to it where p-value < significance level of 0.05; it will be statistically significant. Confirming that it could not have been occurred by chance solidifying discrimination.



Section 3.4

The features with the highest degree of linear association between features are no_neigh_below and no_neight_above with degree of linear association of 0.94. Below is the heat map comparing all features:



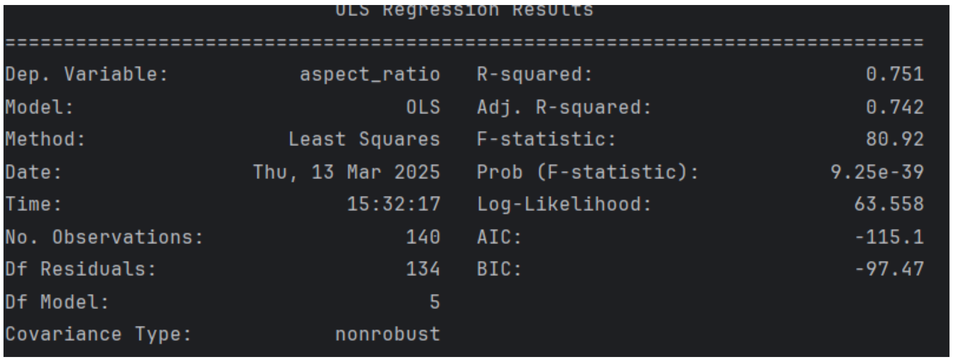
Top 3 features that have high association is (no_neigh_below and no_neight_above) , (no_neigh_right and no_neight_left) and (rows_with_3p and nr_pix). These features have high association due to its mean value being similar.

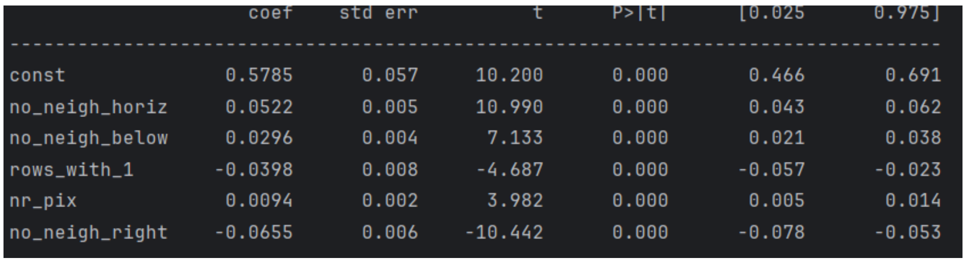
Section 4

Section 4.1

Using Feature selection: forwards-selection to find highest adjusted R squared possible

Figure below shows highest adjusted R squared possible with 'no_neigh_horiz','no_neigh_below','rows_with_1','nr_pix','no_neigh_right':


```
                            OLS Regression Results
==============================================================================
Dep. Variable:           aspect_ratio   R-squared:                       0.751
Model:                            OLS   Adj. R-squared:                  0.742
Method:                 Least Squares   F-statistic:                     80.92
Date:                Thu, 13 Mar 2025   Prob (F-statistic):           9.25e-39
Time:                        15:32:17   Log-Likelihood:                 63.558
No. Observations:                 140   AIC:                            -115.1
Df Residuals:                     134   BIC:                            -97.47
Df Model:                           5
Covariance Type:            nonrobust
```



```
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const             0.5785      0.057     10.200      0.000       0.466       0.691
no_neigh_horiz    0.0522      0.005     10.990      0.000       0.043       0.062
no_neigh_below    0.0296      0.004      7.133      0.000       0.021       0.038
rows_with_1      -0.0398      0.008     -4.687      0.000      -0.057      -0.023
nr_pix            0.0094      0.002      3.982      0.000       0.005       0.014
no_neigh_right   -0.0655      0.006    -10.442      0.000      -0.078      -0.053
```

Interpretation of Results:

R-squared: Value is 0.751, meaning approximately 75.1% of variance is explained by the model. It indicates a strong fit however some portion is unexplained.

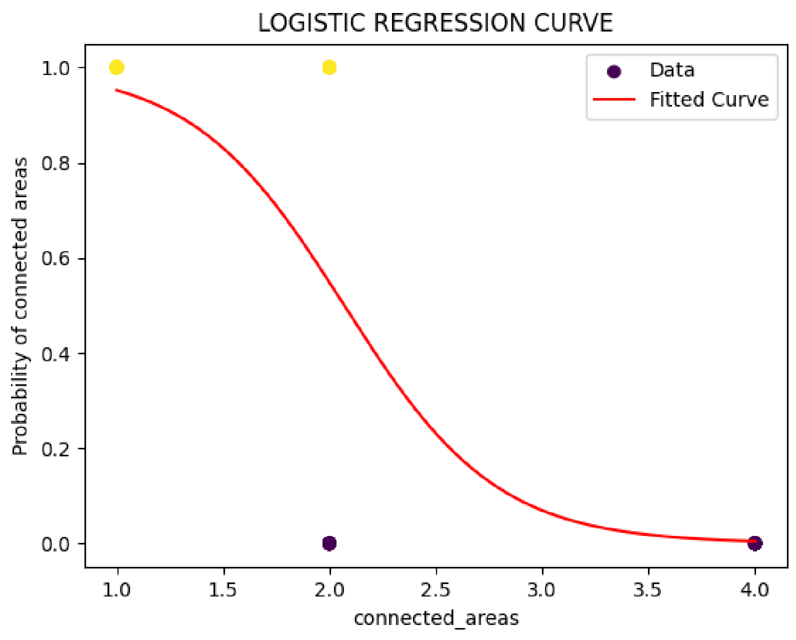
Adjusted R-squared: Value is 0.742, slightly lower than R-squared which indicates some predictors may not be contributing to model.

Standard Error: All features have low to no standard error which indicates there is little to no difference between the population mean from the sample mean.

All feature (no_neigh_horiz, no_neigh_below, rows_with_1, nr_pix and no_neigh_right) shows highly significant effects with P > | t | less than 0.05 showing a strong relationship between features and aspect_ratio. Not only that, (no_neigh_horiz, no_neigh_below, nr_pix) have positive relationships with aspect ratio with positive coefficient whereas; (rows_with_1, no_neigh_right) have negative relationships with aspect_ratio as they have negative coefficient.

Section 4.2

With results from section 3.3 above, connected_areas is the most useful discriminatory feature hence it is used.



The diagram shows a inverted S-shape diagram due to its coefficient being negative. This means when variable (connected_areas) increases, the probability of connected_areas will decrease.

Section 4.3

	Split 1	Split 2	Split 3
Letter	0.3500	0.3875	0.2875
Face	0.8000	0.9250	0.4250
Exclamation Marks	0.2500	0.0000	0.0000

Conclusions

The assignment is done with the use of Python programming with various use of libraries to achieve specific data for each section. Through data analysis and data visualisation, the assignment is completed with various diagrams to corroborate further explanations.