CSC2062 AIDA – Assignment 1

Thomas Bloomer

40394874

01/03/2025

Introduction

In this assignment, I created a dataset of 140 images – comprising of 10 letters and 3 symbols – and converted them into 18x18 binary matrices. I then engineered 16 numerical features that capture both pixel-level properties and higher-level structural characteristics. Extensive statistical analyses, including descriptive statistics, hypothesis testing, and correlation analysis were conducted to explore these features. Finally, I applied multiple regression and logistic regression models to predict aspect ratio and classify images as letters versus non-letters, respectively, to validate the discriminative power of the features.

Section 1 – Creating the Dataset

In this section, I created a dataset of 140 hand-drawn images and converted them into 18x18 binary matrices. I hand-drew the symbols using GIMP, which included 10 letters (a - j) with 8 samples each, and 3 symbols (happy face, sad face, and exclamation mark) with 20 samples each.

For each image, I prepared an 18x18 pixel greyscale canvas in GIMP, and used a 1-pixel pencil tool to draw the character ensuring it appropriately filled the grid. After drawing I exported each image as an ASCII PGM file. It was essential to ensure that the file was in ASCII format (indicated by the “P2” header) rather than binary (which would have a “P5” header)

```
P2
# Created by GIMP version 2.10.38 PNM plug-in
18 18
255
```

Figure 1: The typical header of an ASCII PGM file.

This header, along with the subsequent 324 pixel values allowed me to verify the correct structure of the file. For additional clarity on the PGM format, I referred to the Netpbm documentation [1].

Following this, I developed a Python script to process each PGM file. The script verifies each file before iterating through each pixel value. If the value is below the threshold value, which by default is 128, it is set to 1 to signify black; otherwise, it is set to 0 to signify white. The flat list is then processed by moving a sliding window of width 18 across the list, with each slice of 18 corresponding to one complete row of the image. This matrix is then saved as a comma-delimited CSV file (named using the format *40394874_LABEL_INDEX.csv*) in the “images/CSV” folder, while retaining the original PGM files.

```
matrix = []
for i in range(height):
    row = binary_pixels[i * width:(i + 1) * width]
    matrix.append(row)
```

Figure 2: The python expression used to extract the slice of 18 consecutive values

Finally, I developed an additional Python script which chooses a random CSV file in the “images/CSV” folder and ensures that it meets the specification required for this assignment.

Section 2 – Feature Engineering

For this section, I extracted 16 numerical features from each 18x18 binary image produced in Section One. These features capture a wide range of properties, including both basic pixel-level properties such as total black pixel count and row/column statistics, and higher-level structural characteristics such as the aspect ratio and neighbourhood connectivity. Together, these features form a robust descriptor for each handwritten symbol, which will serve the basis for subsequent statistical analyses.

After loading each CSV image into a NumPy array, I computed the following features:

1. **nr_pix**: The total number of black pixels, calculated by taking the sum of all values in the matrix.
2. **rows_with_1** and **cols_with_1**: The counts of rows and columns, respectively, that contain exactly one black pixel. I achieved this by summing along each row, or column, and counting how many of these equal one.
3. **rows_with_3p** and **cols_with_3p**: The number of rows and columns that contain three or more black pixels. This was achieved in a similar fashion to calculating rows and columns with one pixel, but instead, by counting how many are greater than or equal to three.
4. **aspect_ratio**: Using NumPy's argwhere function, which finds the indices of array elements that are non-zero grouped by element [2], I was able to calculate the minimal bounding box around the symbol. The aspect ratio was then calculated as the width divided by the height of the bounding box, capturing the overall shape of the symbol.
5. **neigh_1**: I implemented a helper function, "count_neighbours", to count the 8-connected neighbours for a pixel. I then iterated through every pixel, tallying the number of black pixels that had exactly one black neighbour.
6. **Directional Isolation Features:**
    - **no_neigh_above**, **no_neigh_below**, **no_neigh_left**, **no_neigh_right**: For each black pixel, I examined the immediate neighbours in the specified directions using a sliding window approach and counted those pixels that had no adjacent black pixels in that region.
    - **no_neigh_horiz** and **no_neigh_vert**: These features aggregate the isolation checking by only checking the black neighbours on both the left and right, or top and bottom, simultaneously.
7. **connected_areas**: To calculate this feature, I used an 8-connected neighbourhood- defined using SciPy's "generate_binary_structure(2, 2)" [3] – to determine connectivity between pixels. This means that any two foreground pixels with value 1 are considered connected if they are adjacently horizontally, vertically, or diagonally. The "nd_label" function then labels each unique connected component. I then calculated the total number of these labels, which represents the number of connected areas in the image.
8. **eyes**: According to the assignment specification, an "Eye" is a region of whitespace that is completely surrounded by lines of the character. To compute this, I started by inverting the binary image so that white becomes 1, and black becomes 0. I then applied connected component analyses, as seen in **connected_areas**, but instead using a 4-connected structure. I then counted the connected white regions that do not touch the image border, as these represent truly enclosed regions.
9. **custom - Normalised Horizontal Symmetry**: Initially, I was planning on using the Euler number as my custom feature, however I found it to be too similar to the **eyes** feature, so I decided to base my custom feature on horizontal symmetry. I started by calculating the symbols bounding box, as seen in the **aspect_ratio** feature, and then calculating the midpoint column. I then flip the right half of the

bounding box horizontally and calculate the absolute differences between the left and right halves. This value is then normalised by the total number of pixels in the halves to yield a value between 0 and 1, with 1 indicating perfect symmetry. Additionally, to handle edge cases, I return a default score of 1.0, under the assumption that insufficient width implies a lack of asymmetry.

Finally, I aggregated these features from each image into a master CSV file (named "40394874_features.csv"), where each row includes the label, index, and the complete set of computed features.

label	Index	nr_pix	rows_with_1	cols_with_1	rows_with_3p	cols_with_3p	aspect_ratio	neigh_1	no_neigh_above	no_neigh_below	no_neigh_left	no_neigh_right	no_neigh_horiz	no_neigh_vert	conected_areas	eyes	custom
a	1	42	0	2	9	7	1.3333333333333300	0	8	9	8	8	2	8	1	1	0.4814814814814820
a	2	33	0	2	8	7	1.1111111111111100	0	6	6	5	5	3	5	1	1	0.5333333333333330
a	3	28	0	2	6	6	1.2857142857142900	0	6	8	8	7	7	7	1	1	0.3928571428571430
a	4	34	0	1	7	7	1.1111111111111100	0	5	5	6	6	3	4	1	1	0.5111111111111110

Figure 3: The first 4 rows of the generated features CSV

Section 3

In this section, I perform extensive statistical analysis on the 16 features extracted in Section Two from the 140 handwritten images.

Section 3.1 – Descriptive Statistics and Visualisations

In this section, I explored the distributions of the first six features by constructing histograms for each using the seaborn library [4].

Figure 4: Histogram of nr_pix

In Figure 4, the histogram of **nr_pix** (total number of black pixels) is shown. This graph exhibits a roughly unimodal distribution with a principal cluster around 25-30 pixels. At the 35-40 pixel range a mild right skew emerges, indicating a small subset of symbols are drawn with heavier lines. At the lower end, around 5-10 pixels, the histogram includes symbols that are especially sparse, which are most likely caused by simple symbols such as the exclamation mark ("!"). These few symbols create a left tail, though it remains less

pronounced than the right tail. Overall, whilst many handwritten characters have similar pixel counts, there is considerable diversity at both extremes.

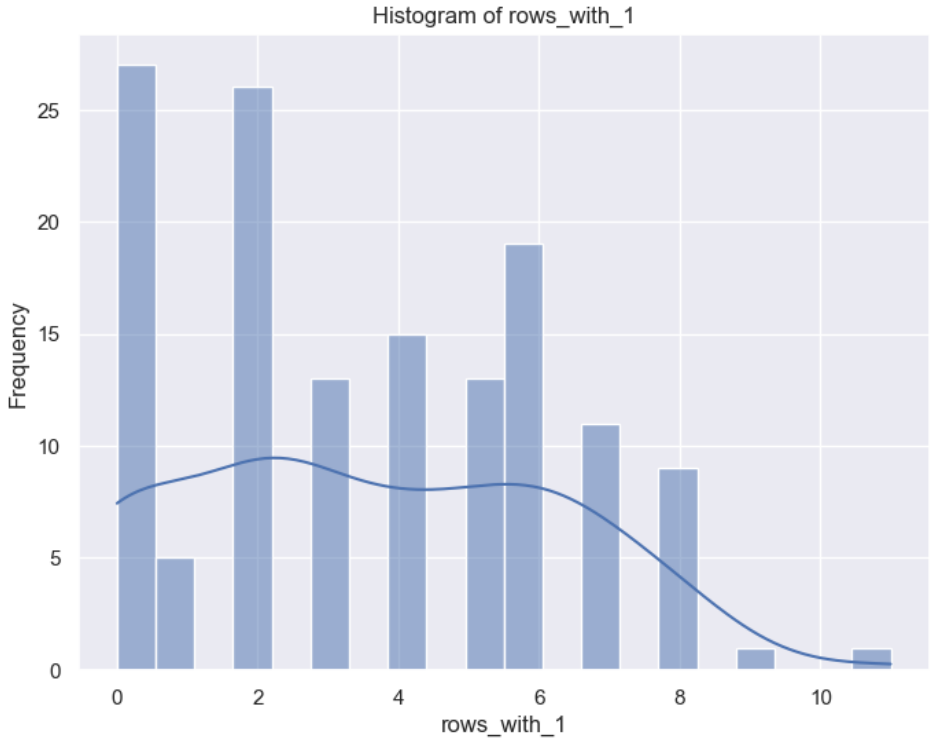

Figure 5: Histogram of rows_with_1

In Figure 5, the histogram of **rows_with_1** (rows with only 1 black pixel) is shown. Visually, the distribution is discrete and highly skewed, with the highest frequency being at zero. This indicated that for most symbols, any row containing a stroke typically has multiple pixels. As we move away from zero, a tail is formed that stretches to around 10, as the frequency gradually declines. This suggests that whilst uncommon, certain symbols are characterised by sparse or interrupted strokes, for instance the exclamation mark (“!”).

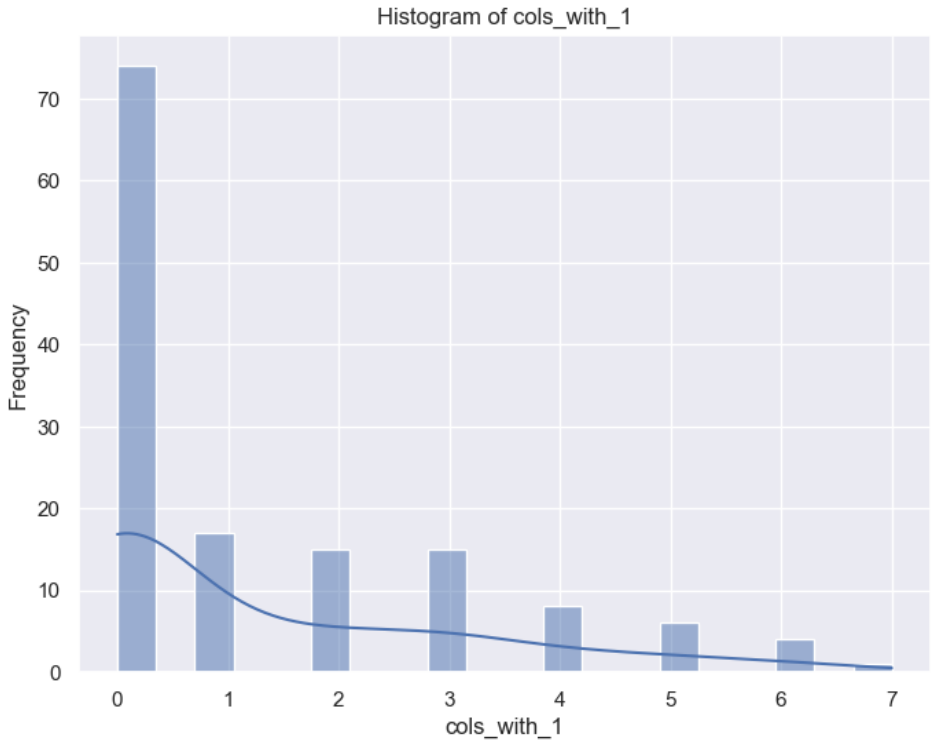

Figure 6: Histogram of cols_with_1

In Figure 6, the histogram for **cols_with_1** (columns containing exactly one black pixel) is highly discrete and strongly skewed toward zero. In fact, most symbols fall into the zero bin, indicating that their columns either contain no pixels, or multiple pixels, rather than exactly one. As the number of single-pixel columns increases beyond zero, the frequency declines sharply, reflecting a relatively small proportion of symbols that exhibit sparse, isolated horizontal strokes, which could be attributed to characters like the exclamation mark (“!”). The long right tail up to seven indicates that whilst uncommon, some symbols are particularly prone to isolated columns.

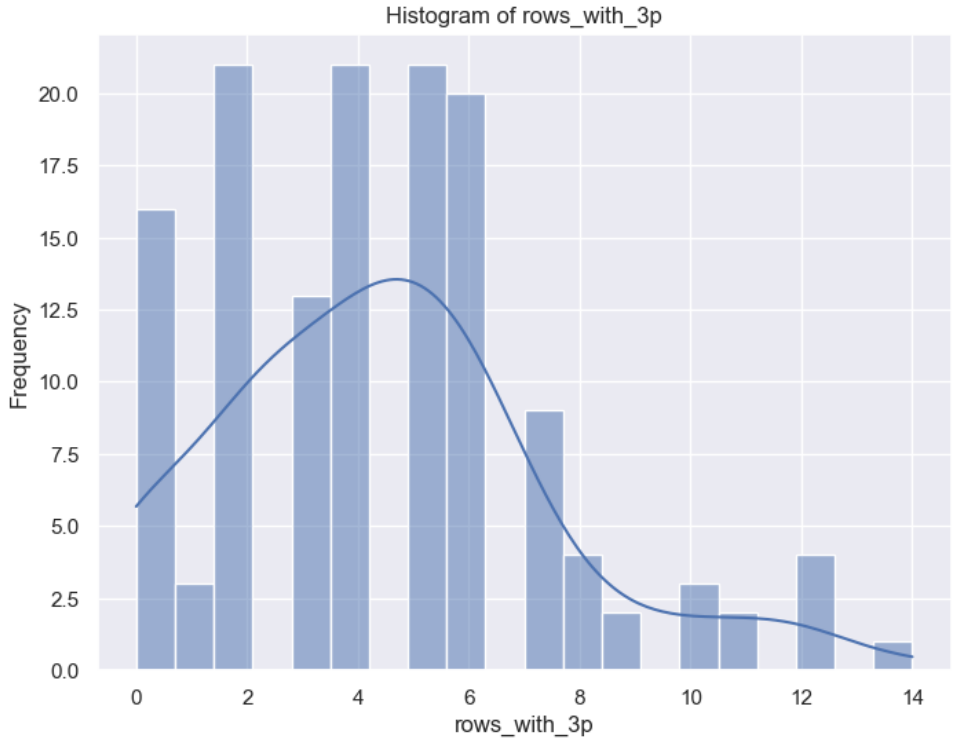

Figure 7: Histogram of rows_with_3p

In Figure 7, the histogram for **rows_with_3p** (rows with at least three black pixels) appears roughly unimodal, with the highest frequency centring around 4-6 rows. This indicates that many symbols utilise a moderate number of vertically dense rows. Beyond this central cluster, a gentle right skew emerges, extending out to around 14 possible reflecting more elaborate designs. Conversely, the lower end of the distribution captures relatively sparse vertical coverage. Ultimately this histogram highlights a spectrum of vertical density within the dataset.

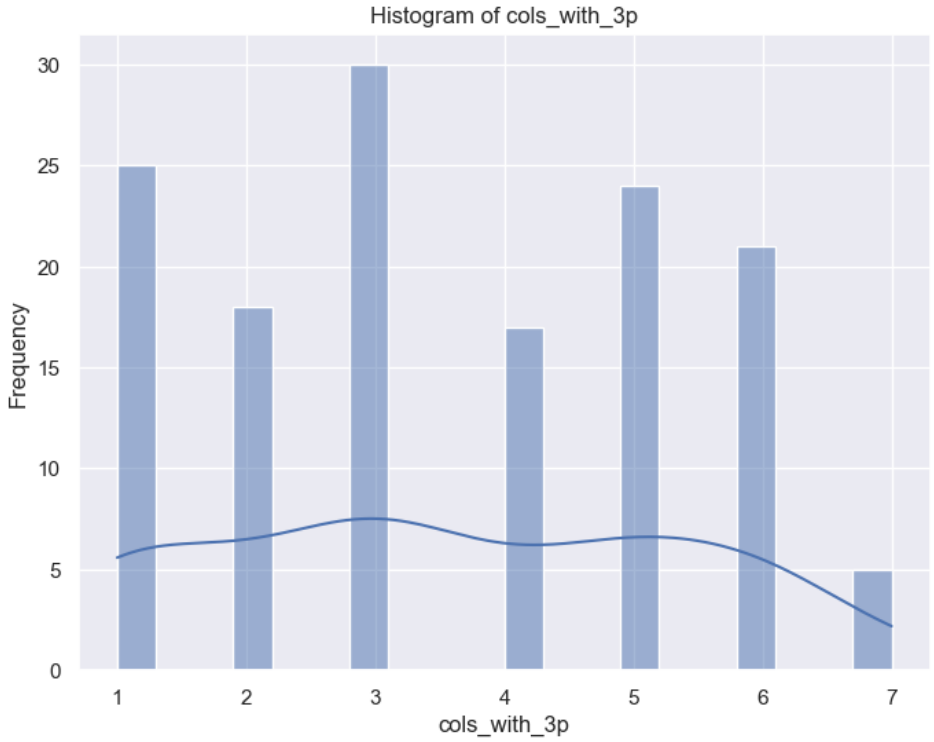

Figure 8: Histogram of cols_with_3p

In Figure 8, the **cols_with_3p** histogram (columns with at least three black pixels) appears more uniformly spread compared to **rows_with_3p**, with no single dominant peak. This relatively even spread suggests that many symbols utilise a moderate portion of the horizontal axis to form strokes, rather than clustering heavily around a specific count of dense columns. Although slight variations exist across the range, no extreme skew is evident, however there is a slight decline in frequency around 7 columns.

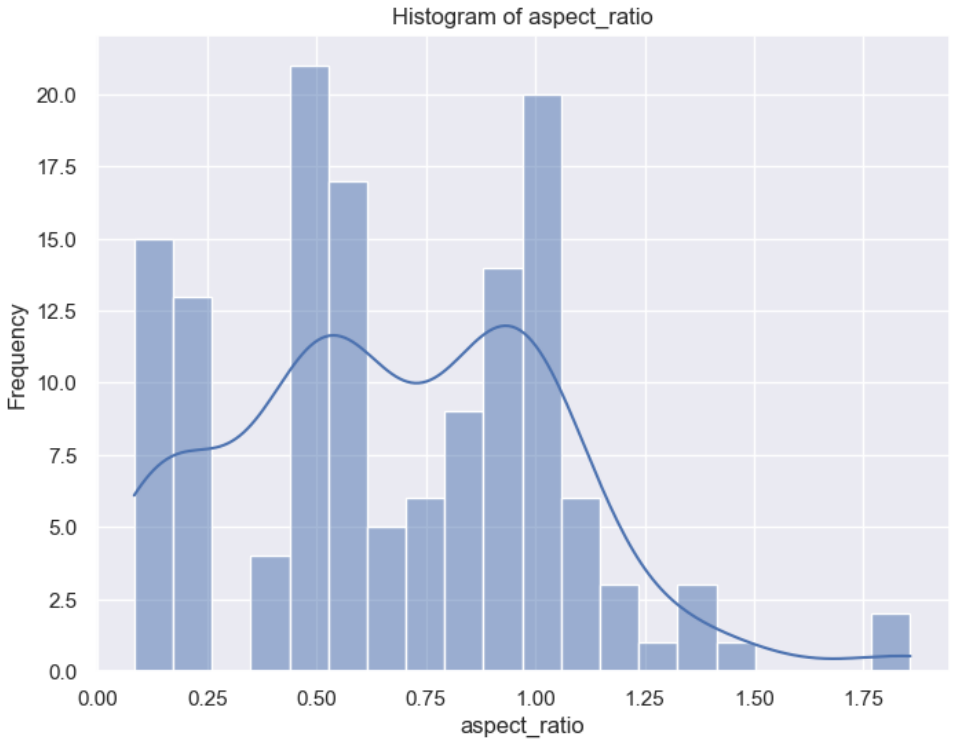

Figure 9: Histogram of aspect_ratio

In Figure 9, the histogram for **aspect_ratio** (width-to-height ratio) suggests two mild concentrations: one peaking around 0.5 and another around 0.8-1.0, with a notable tail extending towards 1.75. This indicates that while many symbols are roughly square, with an aspect ratio near 1.0, a sizeable number are drawn either more vertically elongated, or more horizontally stretched. The absence of a single sharp peak points to a very diverse set of shapes within the dataset. A small number of symbols appear at the extreme left of the histogram, close to 0.0, which could correspond to tall, narrow characters. Conversely, the right-side tail captures wider characters or symbols.

Overall, these histograms reveal that whilst some features, like **nr_pix** and **aspect_ratio** show continuous distributions, the isolation measures (**rows_with_1** and **cols_with_1**) are highly discrete.

Section 3.2 – Group-Specific Summary Statistics and Discriminative Visualisations

In this section, I divided the dataset into two groups – letters (symbols a-j) and non-letters (all other symbols) – and computed summary statistics for all 16 features, including the mean, standard deviation, and median for each group:

group	mean	std	median
letter	22.575	8.032805	23.0
non-letter	26.700	7.578516	27.5

Figure 10: Table showing nr_pix summary statistics

For Figure 10, the letters group has a mean of approx. 22.58 with a median of 23, whereas the non-letter group shows a higher mean of 26.70 and median of 27.5. This indicates that non-letter symbols are generally drawn with greater pixel density, suggesting more elaborate designs compared to letter symbols. The similar standard deviation implies that, despite the difference in central tendency, the overall variability is comparable between the groups.

group	mean	std	median
letter	4.637500	2.398543	5.0
non-letter	2.216667	2.336821	2.0

Figure 11: Table showing rows_with_1 summary statistics

In Figure 11, letters exhibit a mean of approx. 4.64 (median 5) compared to non-letters with a mean of approx. 2.22 (median 2). This suggests that letters tend to have more rows where only one pixel is needed, contrasting non-letters which appear to have such isolated rows.

group	mean	std	median
letter	0.750000	1.163800	0.0
non-letter	2.083333	2.141532	1.5

Figure 12: Table showing cols_with_1 summary statistics

For Figure 12, the letters group has a mean of 0.75 (median 0), while non-letters show a higher mean of approx. 2.08 (median 1.5). This implies that non-letter symbols frequently exhibit columns with a single black pixel, which could be due to discrete vertical elements in their design. This discrimination could be useful for distinguishing symbols based on their vertical structure.

group	mean	std	median
letter	3.5875	2.197200	4.0
non-letter	5.5500	3.441644	5.5

Figure 13: Table showing rows_with_3p summary statistics

In Figure 13, letters have a mean of approximately 3.59 (median 4), whereas non-letter have a higher mean of 5.55 with a median of the same number. This indicates that non-letters typically contain more rows with dense pixel clusters, which could reflect a more complex design compared to the letter group.

group	mean	std	median
letter	3.475	1.967939	3.0
non-letter	3.700	1.576308	3.0

Figure 14: Table showing cols_with_3p summary statistics

For Figure 14, the letters group shows a mean of 3.48 (median 3) compared to a slightly higher mean of 3.70 (median 3) in non-letters. As the groups have a very subtle difference, this feature won't be useful in differentiation between the groups.

group	mean	std	median
letter	0.643715	0.344523	0.545455
non-letter	0.723759	0.401154	0.916667

Figure 15: Table showing aspect_ratio summary statistics

In Figure 15, letters have a lower mean of approx. 0.64 (median around 0.55), while non-letters exhibit a higher mean of around 0.72 with a median of approx. 0.92. This indicates very distinct shape types, which is a strong indicator of design differences between letters and non-letters.

group	mean	std	median
letter	0.025000	0.223607	0.0
non-letter	0.166667	0.642207	0.0

Figure 16: Table showing neigh_1 summary statistics

For Figure 16, both groups display very low values with letters having a mean of around 0.03 and non-letters approx. 0.17, with medians of 0 in both cases. This suggests that isolated pixels are rare in both groups, implying that this feature will be less discriminative and won't be useful in differentiation between the groups.

group	mean	std	median
letter	5.412500	2.463371	5.0
non-letter	8.183333	3.820070	9.0

Figure 17: Table showing no_neigh_above summary statistics

In Figure 17, letters have a mean of around 5.41 (median 5) while non-letters exhibit a higher mean of around 8.18 (median 9). This implies that non-letter symbols tend to have more pixels that are not connected to any stroke above them, which could be important in highlighting design differences between the groups.

group	mean	std	median
letter	5.575000	2.443151	5.0
non-letter	8.183333	4.102466	9.0

Figure 18: Table showing no_neigh_below summary statistics

For Figure 18, the letters have a mean of approx. 5.58 (median 5), whereas non-letters have a higher mean of approx. 8.18 (median 9). This reinforces the pattern observed in **Figure 17**, which could prove useful in discriminating between the two groups.

group	mean	std	median
letter	8.550000	3.744701	8.0
non-letter	9.583333	1.543622	9.0

Figure 19: Table showing no_neigh_left summary statistics

In Figure 19, the letters group has a mean of 8.55 (median 8) whilst non-letters have a slightly higher mean of 9.58 (median 9). As this is a very subtle difference, it won't be useful in discriminating between the groups.

group	mean	std	median
letter	8.412500	3.355305	9.0
non-letter	9.516667	1.641534	9.0

Figure 20: Table showing no_neigh_right summary statistics

For Figure 20, the letters group shows a mean of approx. 8.41 (median 9), while non-letters have a slightly higher mean of around 9.52 (median 9). This is like **Figure 19**, however whilst slightly higher, it still won't be extremely useful in differentiating between groups.

group	mean	std	median
letter	8.762500	3.646548	8.0
non-letter	4.333333	2.765812	4.0

Figure 21: Table showing no_neigh_horiz summary statistics

In Figure 21, letters have a notably higher mean (around 8.76, median 8) than in non-letters (approx. mean 4.33, median 4). This significant different indicates that letters tend to have more horizontal isolation in contrast to non-letters, making this feature a strong discriminator.

group	mean	std	median
letter	6.012500	3.643943	6.0
non-letter	3.616667	2.718996	4.0

Figure 22: Table showing no_neigh_vert summary statistics

For Figure 22, the letters group has a mean of approx. 6.01 (median 6) whilst non-letters have a lower mean of around 3.62 (median 4). This suggests that letters tend to have greater vertical isolation, which could help distinguish between the groups.

group	mean	std	median
letter	1.200000	0.402524	1.0
non-letter	3.333333	0.950765	4.0

Figure 23: Table showing connected_areas summary statistics

In Figure 23, letters have a mean of 1.2 (median 1) and a relatively low standard deviation, whilst non-letters have a significantly higher mean of 3.33 (median 4). This contrast indicates that non-letter symbols are typically composed of multiple regions. This clear difference suggests that this feature is highly discriminative.

group	mean	std	median
letter	0.5	0.503155	0.5
non-letter	0.0	0.000000	0.0

Figure 24: Table showing eyes summary statistics

For Figure 24, letters exhibit a mean and median of 0.5 (with a standard deviation of approximately 0.5), whereas non-letters have a value of 0 across all statistics. This indicates that letters incorporate an enclosed white region, which would serve as a critical feature in discriminating between the groups.

group	mean	std	median
letter	0.682140	0.150350	0.666667
non-letter	0.938432	0.065474	0.956818

Figure 25: Table showing custom (normalised horizontal symmetry) summary statistics

In Figure 25, letters have a mean of approx. 0.68 (median around 0.67), whilst non-letters have a much higher mean of around 0.94 (median approx. 0.96). This indicates that letters tend to display asymmetry. The low variability in non-letters (standard deviation of around 0.07) suggests that this is a consistent characteristic, which would make it an effective discriminator between both groups.

After reviewing these statistics for all 16 features, I decided upon **aspect_ratio**, **connected_areas**, and **custom** as the three features which would be interesting for discrimination between the groups. Together they capture the stroke connectivity, symmetry, and overall shape of the symbols, which should offer a robust basis for discriminating between letters and non-letters. To illustrate this, I generated Violin plots for these three features:

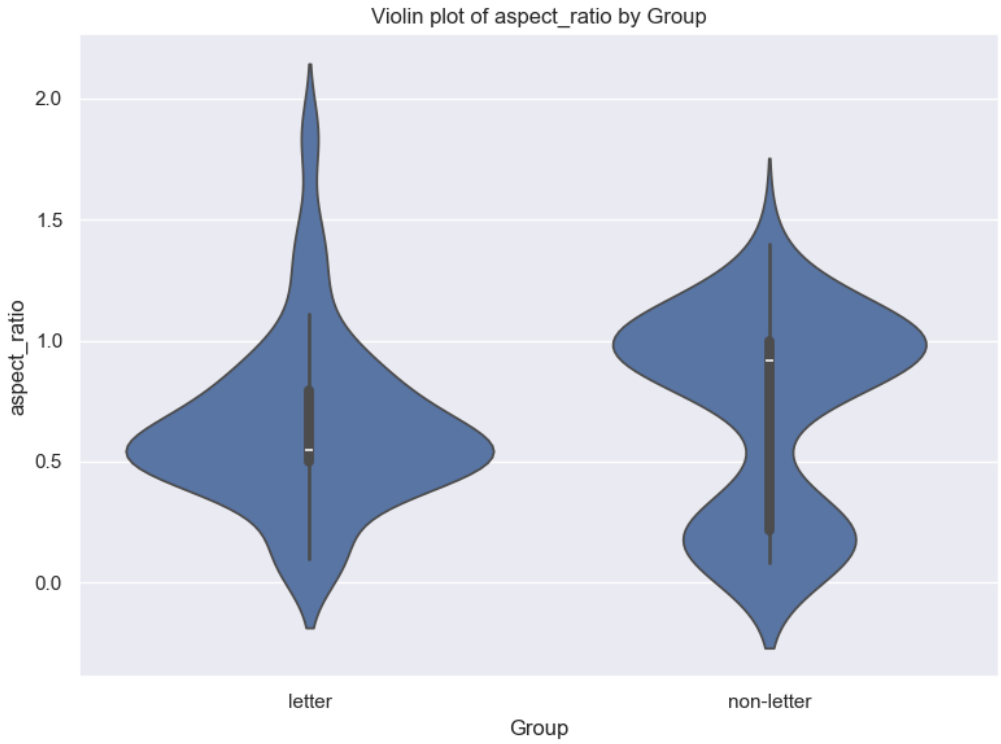

Figure 26: Violin plot of aspect_ratio by Group

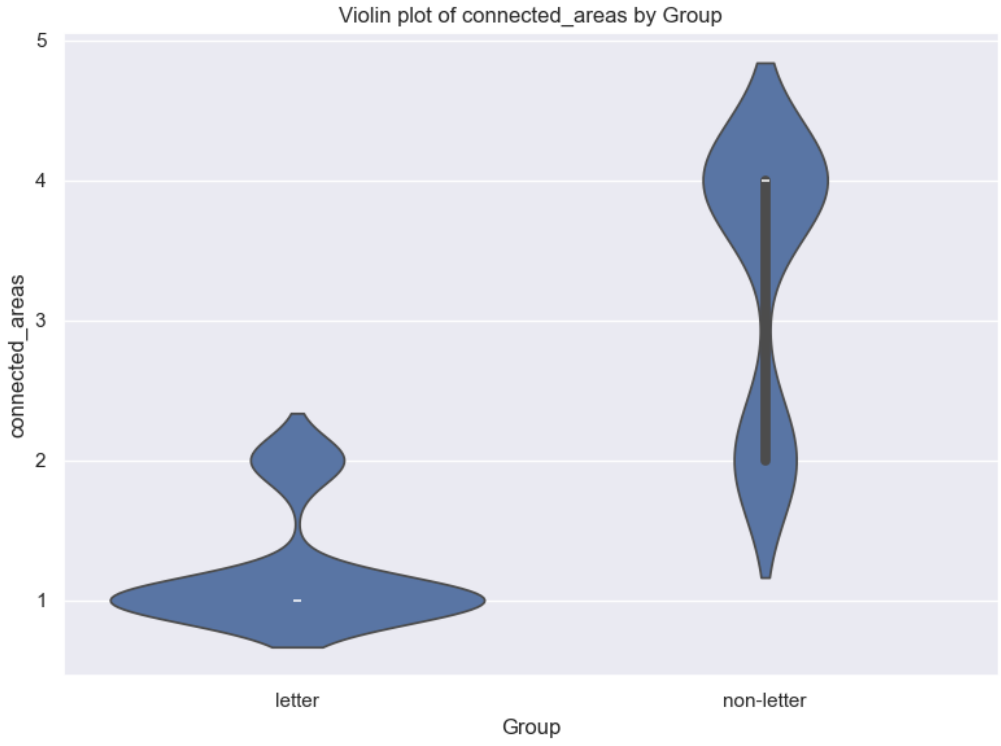

Figure 27: Violin plot of connected_areas by Group

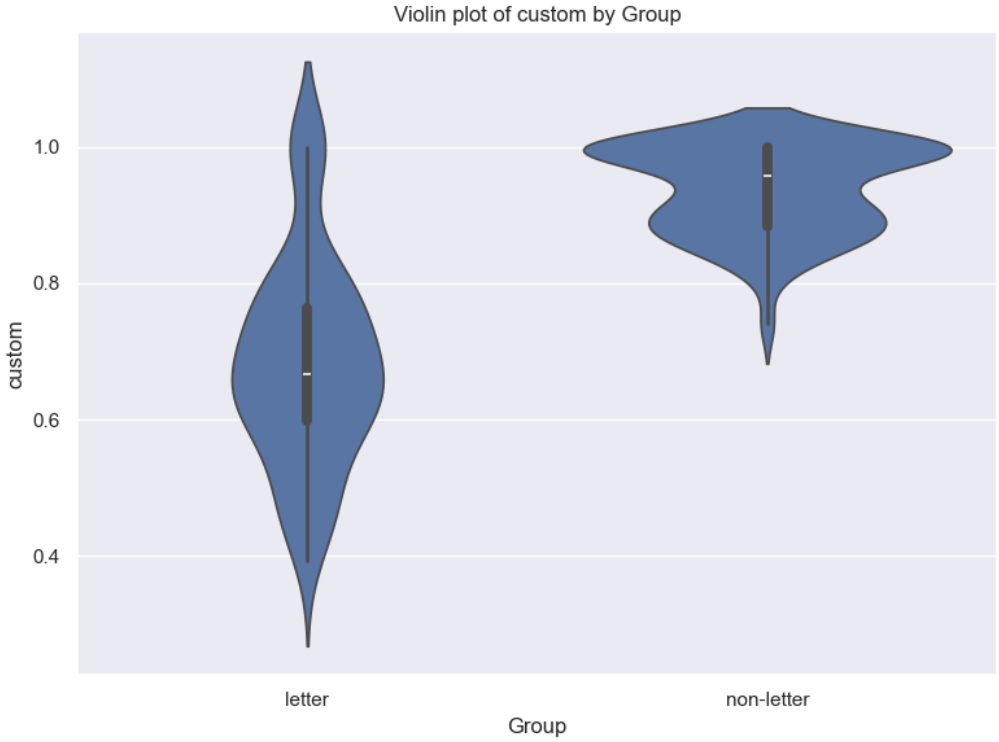

Figure 28: Violin plot of custom by Group

The violin plots for these features, **Figures 26 through to 28**, exhibit minimal overlap between letters and non-letters, proving their strong discriminative power between the groups.

Section 3.3 – Hypothesis Testing of Group Differences

In this section, I performed statistical tests to determine whether the differences in feature values between letters and non-letters and non-letters are statistically significant. For each of the 16 features, an independent t-test, assuming unequal variances, is conducted comparing the two groups.

A p-value less than 0.05 indicates that the difference in means is unlikely due to chance, suggesting that the feature is useful for group discrimination. The results of the tests can be seen in **Figure 29** below:

	Feature	t-test	p-value
0	nr_pix	-3.10598	0.00233
1	rows_with_1	5.99751	0.00000
2	cols_with_1	-4.36359	0.00004
3	rows_with_3p	-3.86546	0.00020
4	cols_with_3p	-0.75074	0.45409
5	aspect_ratio	-1.24016	0.21742
6	neigh_1	-1.63595	0.10635
7	no_neigh_above	-4.90534	0.00000
8	no_neigh_below	-4.37701	0.00003
9	no_neigh_left	-2.22856	0.02785
10	no_neigh_right	-2.56273	0.01161
11	no_neigh_horiz	8.17262	0.00000
12	no_neigh_vert	4.45514	0.00002
13	connected_areas	-16.31819	0.00000
14	eyes	8.88819	0.00000
15	custom	-13.62158	0.00000

Figure 29: The results of the independent t-tests

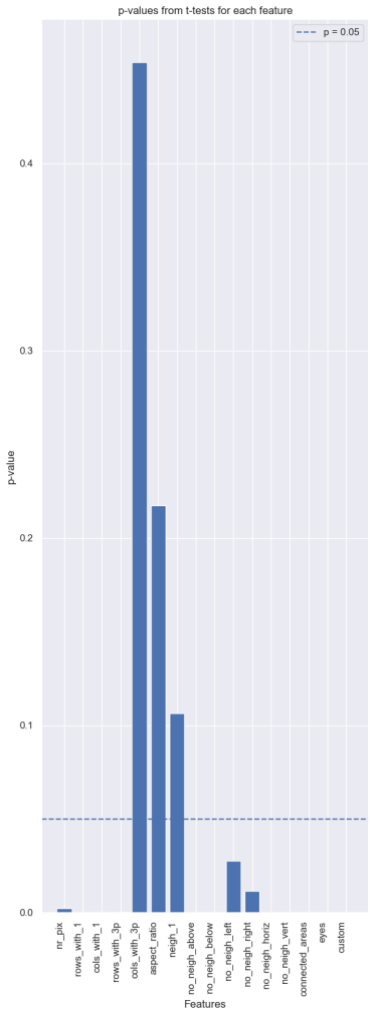

Figure 30: A bar chart of the calculated p-values

From the t-test results, several features, such as **connected_areas**, **custom**, **rows_with_3p**, **no_neigh_horiz**, and **eyes**, exhibit p-values well below 0.05. This strongly suggests that these features differ between letter and non-letters and may be useful for discrimination. For instance, the extremely low p-value for **connected_areas** ($p<0.00001$) shows that letters are consistently drawn as a single connected region, whereas non-letters often consist of multiple connected regions. Similarly, the **custom** feature and **no_neigh_horiz** reveal distinct group differences in horizontal symmetry and isolation, respectively.

```
connected_areas: t-statistic = -16.318192880331893, p-value = 2.246709663042953e-26
```

Figure 31: The full p-value for connected_areas

Unfortunately, one of the features I chose in Section 3.2, **aspect_ratio**, has a p-value approx. 0.21742. This means that the difference in the mean aspect ratio between the groups is not statistically significant, despite my initial expectation. Whilst it does capture meaningful differences in the overall shape of the symbols, the variability does not differ enough between the groups, and as a result it is less effective compared to others.

Figure 32: Violin plot of no_neigh_horiz by group

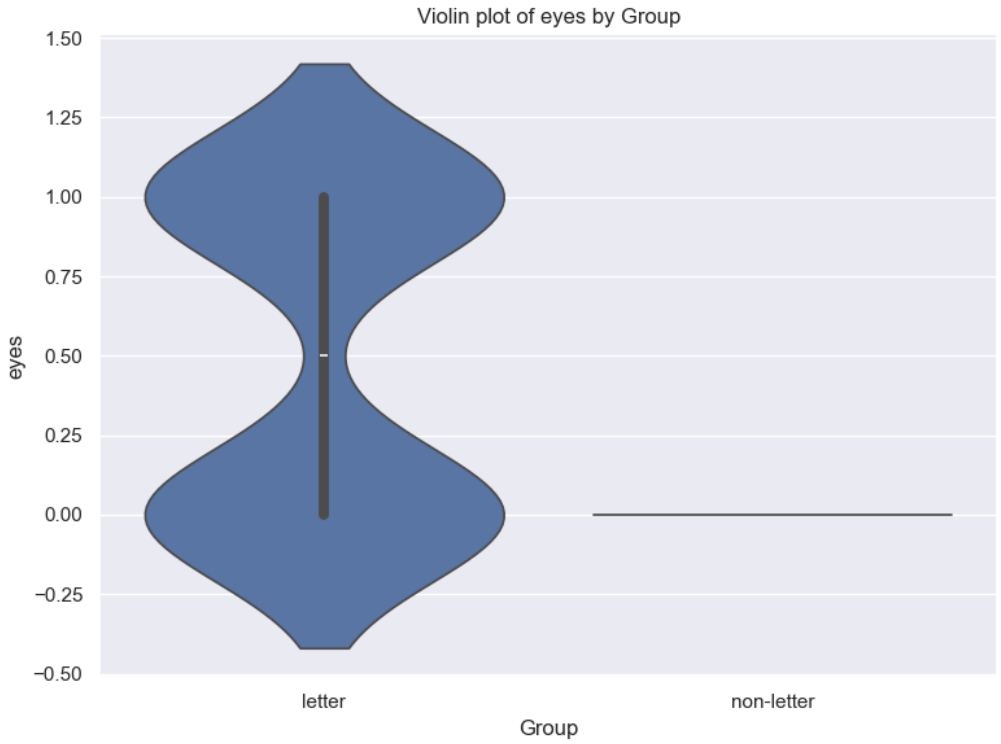

Figure 33: Violin plot of eyes by group

Figure 34: Violin plot of rows_with_1 by group

The Violin plots for **no_neigh_horiz**, **eyes**, and **rows_with_1**, as seen in **Figures 32-34**, visually address these claims by demonstrating minimal overlap.

Ultimately **connected_areas** and **custom** are the top discriminators, as they show very low p-values and clear separation between groups. Following this would be **no_neigh_horiz**, as letters have a significantly higher horizontal isolation than non-letters. Another very discriminative feature is **eyes**, with letters showing an enclosed white region, and non-letters showing none. Finally, **rows_with_1** can add additional evident, as letters tend to have more isolated single pixel rows. All five of these features fall far below the 0.05 threshold, indicating that they are statistically significant, and making them effective candidates for classification.

Section 3.4 – Correlation and Linear Association Analysis among features

In this section, I investigated the linear relationships amongst the 16 features extracted from the handwritten images. Using Pearson's correlation coefficient, I computed a correlation matrix that quantifies the degree to which each pair of features vary together using the Python Data Analysis Library (pandas), this data was then passed into seaborn to generate a heatmap, see **Figure 35**, to better visualise these relationships.

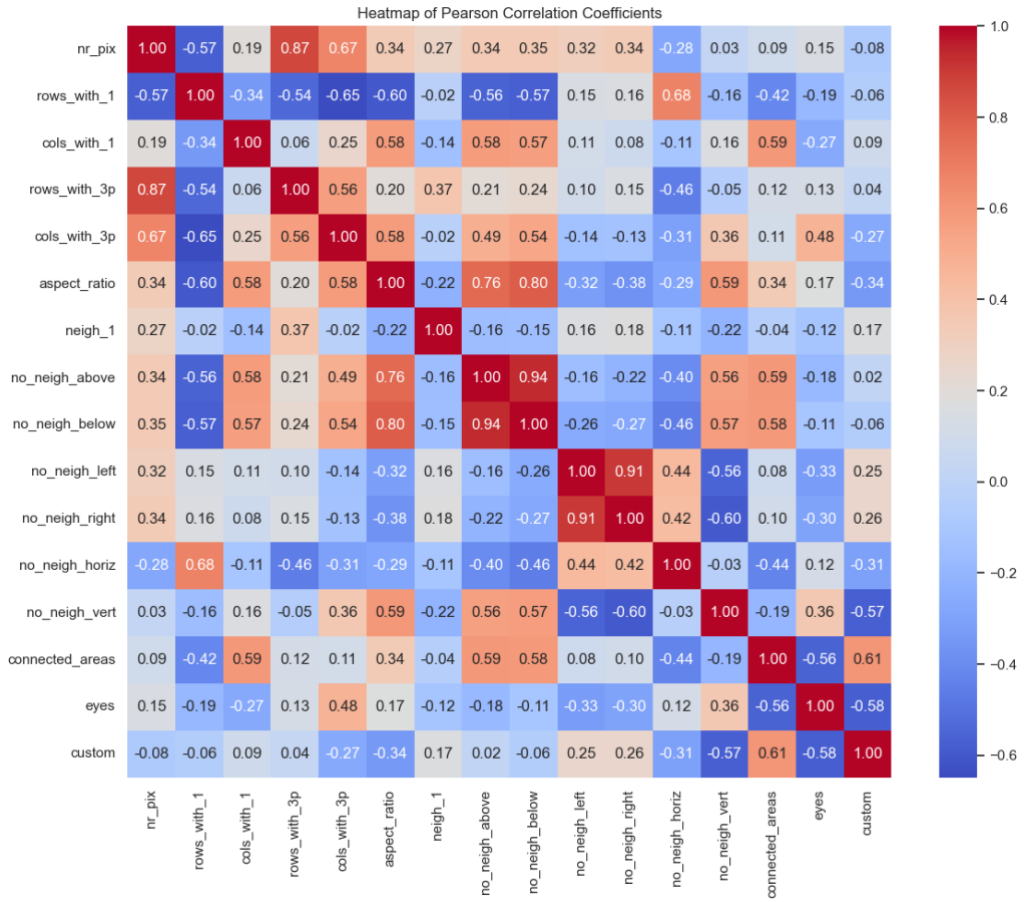

Figure 35: Heatmap of Pearson Correlation Coefficients

Several feature pairs show strong correlations (with $|r| > 0.7$), suggesting potential redundancy in the dataset, as seen in **Figure 36**. A common theme that emerges is that features that are computed using similar underlying properties (such as the isolation measures) tend to be highly correlated.

```
Highly Correlated Feature Pairs (|r| > 0.7):
nr_pix and rows_with_3p: correlation = 0.87
aspect_ratio and no_neigh_above: correlation = 0.76
aspect_ratio and no_neigh_below: correlation = 0.80
no_neigh_above and no_neigh_below: correlation = 0.94
no_neigh_left and no_neigh_right: correlation = 0.91
```

Figure 36: Pairs identified with absolute correlation > 0.7

Figure 37: Scatter Plot of nr_pix against rows_with_3p

In Figure 37, we observe a clear positive trend: as the total number of black pixels increases, the number of rows containing three or more black pixels also rises, validating their correlation.

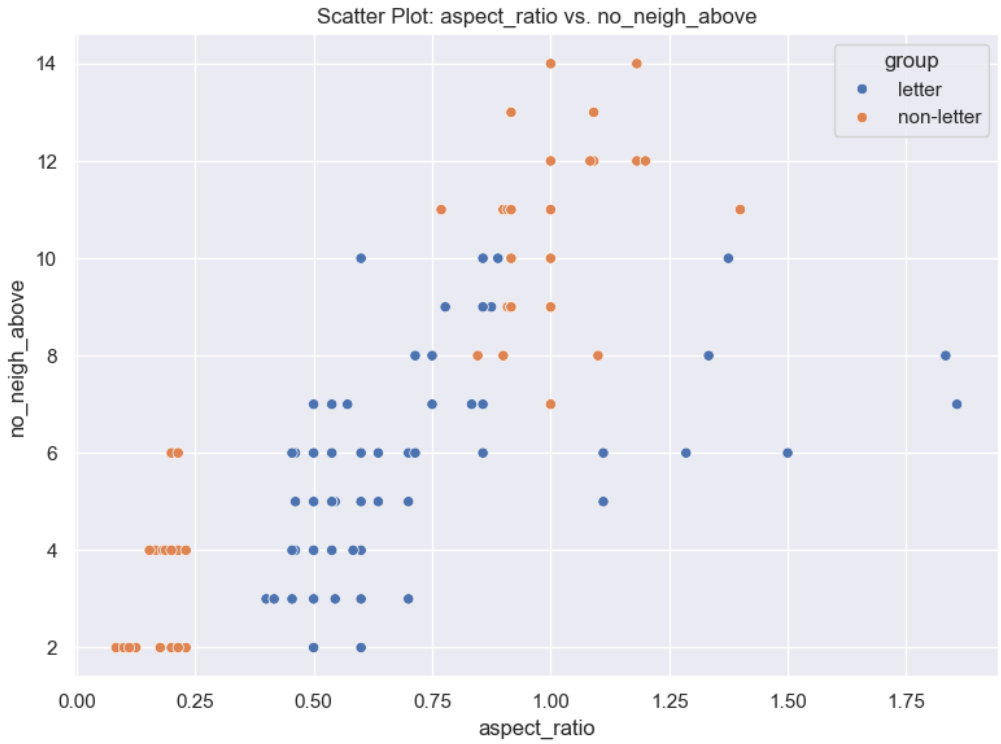

Figure 38: Scatter Plot of aspect_ratio against no_neigh_above

For Figure 38, we can see a partial positive trend: non-letters occupy higher no_neigh_above values especially at aspect_ratios above 0.8, while letters cluster at lower aspect_ratios. This suggests letters are compact and vertically oriented whilst non-letters are more horizontally extended and spread out. Overall, the trend still shows a correlation between the two.

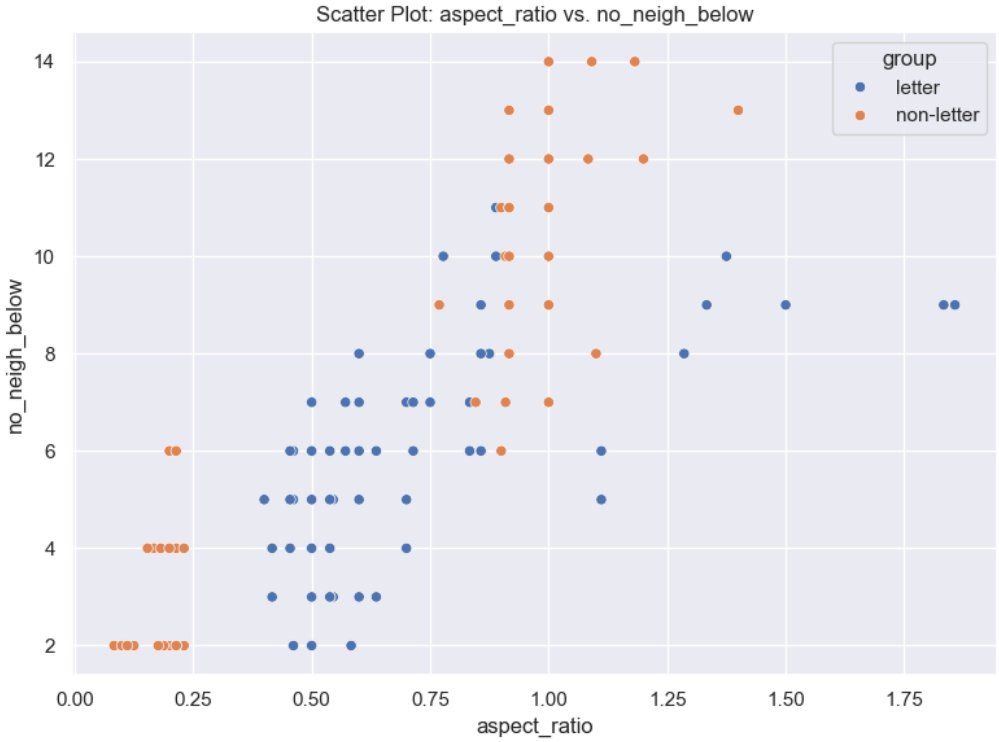

Figure 39: Scatter Plot of aspect_ratio against no_neigh_below

In Figure 39, like **Figure 38**, we can see a partial positive trend: non-letters occupy higher no_neigh_below values as aspect_ratio increases, whereas letters remain clustered at lower aspect_ratios. This implies again that non-letters are horizontally elongated and frequently exhibit more pixels without neighbours below them, while letters tend to be more vertically compact.

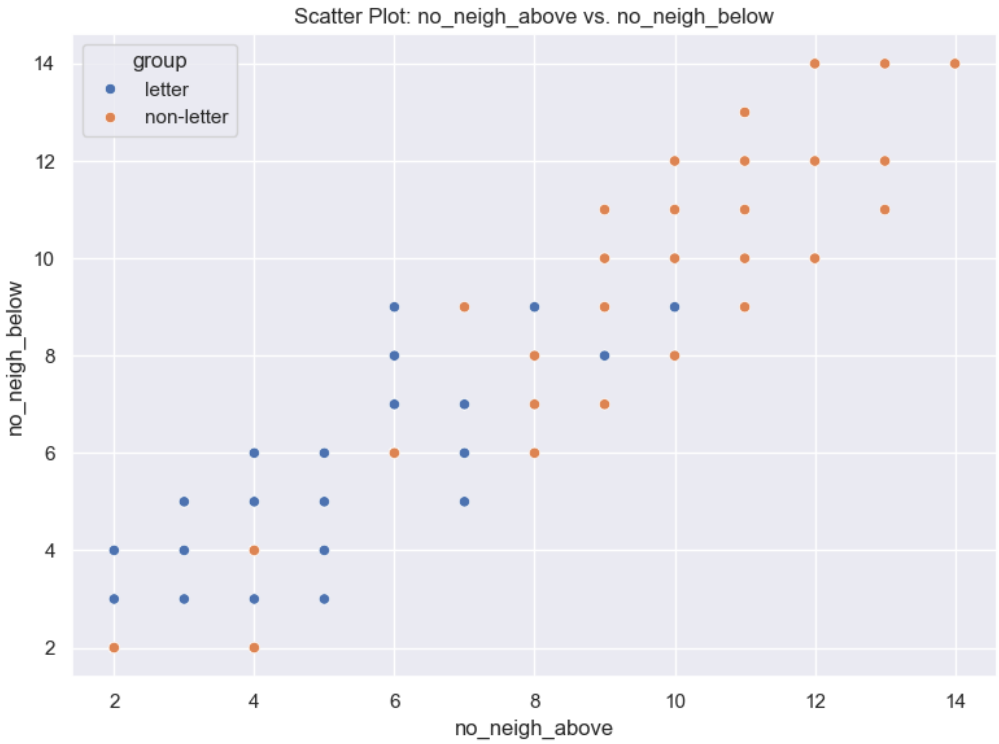

Figure 40: Scatter Plot of no_neigh_above against no_neigh_below

For Figure 40, a strong positive relationship can be seen: as no_neigh_above increases, no_neigh_below also tends to rise, validating their correlation.

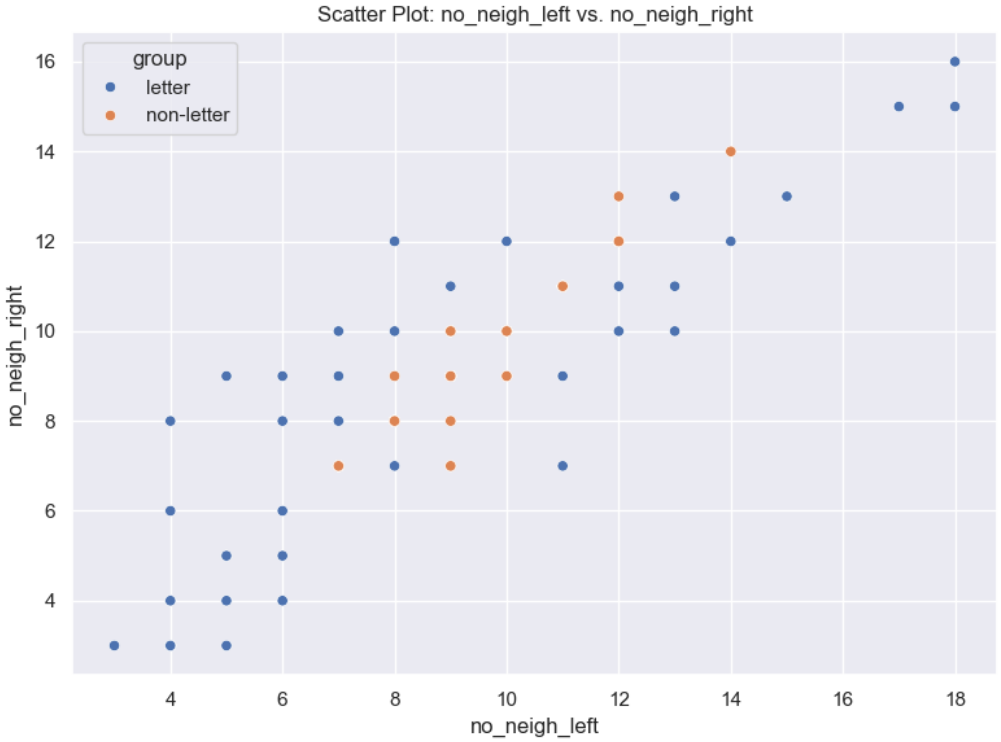

Figure 41: Scatter Plot of no_neigh_left against no_neigh_right

In Figure 41, a positive relationship is evident: symbols with higher no_neigh_left values also tend to show higher no_neigh_right values. Letters mostly cluster at lower isolation on both sides, whereas non-letters often occupy the upper-right section, indicating more pronounced horizontal isolation overall.

Ultimately, by knowing and understanding these relationships, I will be able to ensure that features capture complementary rather than redundant information.

Section 4 – Regression and Machine Learning

In this section, I apply machine learning methods to further validate and utilise the features extracted in Section 2.

Section 4.1 - Multiple Regression Model to predict aspect ratio

In this section, the goal was to predict the aspect_ratio of each handwritten symbol using a parsimonious set of predictors selected from the other 15 features. To achieve this, I fit a multiple regression model with Ordinary Least Squares (OLS).

I trained this model on a subset of features chosen by the RFECV (Recursive feature elimination with Cross-Validation) method.

```
Automatically selected features: ['custom', 'cols_with_1', 'no_neigh_below']
```

Figure 42: The features selected by the RFECV method

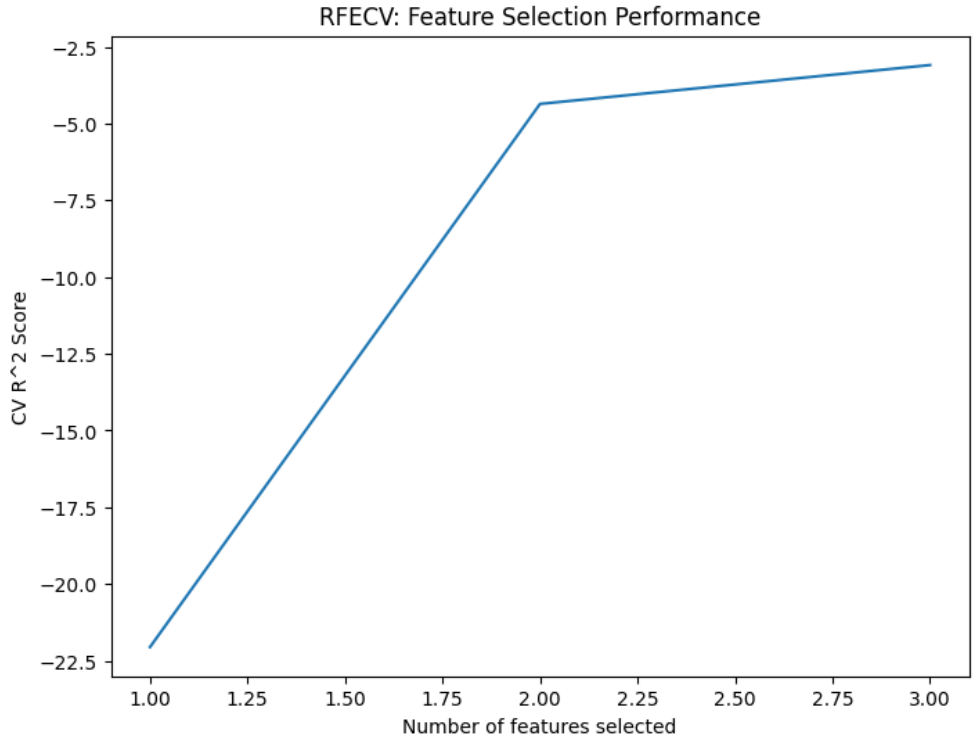

Figure 43: Line Graph to show the effect of increasing the selected features to the CV R^2 score

Once these features were chosen, I included an intercept in the model, so that it could learn a baseline level when all predictors are zero.

OLS Regression Results

Dep. Variable:	aspect_ratio	R-squared:	0.770
Model:	OLS	Adj. R-squared:	0.765
Method:	Least Squares	F-statistic:	151.7
Date:	Mon, 10 Mar 2025	Prob (F-statistic):	3.31e-43
Time:	15:37:54	Log-Likelihood:	43.678
No. Observations:	140	AIC:	-79.36
Df Residuals:	136	BIC:	-67.59
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.7032	0.080	8.837	0.000	0.546	0.861
custom	-0.6958	0.088	-7.920	0.000	-0.870	-0.522
cols_with_1	0.0493	0.011	4.680	0.000	0.028	0.070
no_neigh_below	0.0688	0.005	12.883	0.000	0.058	0.079

Omnibus:	31.516	Durbin-Watson:	1.234
Prob(Omnibus):	0.000	Jarque-Bera (JB):	56.123
Skew:	1.039	Prob(JB):	6.50e-13
Kurtosis:	5.303	Cond. No.	59.5

Figure 44: The Summary Generated by OLS

In Figure 44, you can see that the model explains about 77% of the variance, and all predictors chosen are statistically significant ($p < 0.001$). The model also indicates that symbols with higher values for no_neigh_below and cols_with_1 tend to have higher aspect ratios, whereas greater symmetry corresponds to lower aspect ratios, although some diagnostic tests suggest there are some slight deviations from normality.

Section 4.2 - Logistic Regression for Letter vs. Non-Letter Classification

In this section, I built a logistic regression model to classify images as letters or non-letters using the most discriminative feature identified in Section 3.3, connected_areas.

To begin, I extracted connected_areas and created a binary target variable (1 for letters, 0 for non-letters) using a lambda function. I then divided the data using "train_test_split" from scikit-learn, with 80% being used to train and 20% being used to test, to evaluate model performance. Then I instantiated the logical regression model via "LogisticRegression()" from scikit-learn. It then fit a model that predicts the probability of an image being a letter.

```
Accuracy:  0.8571428571428571
Confusion Matrix:
 [[ 9  4]
 [ 0 15]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.69      0.82        13
           1       0.79      1.00      0.88        15

    accuracy                           0.86        28
   macro avg       0.89      0.85      0.85        28
weighted avg       0.89      0.86      0.85        28
```

Figure 45: Output from the Logical Regression Model

In Figure 45, you can see how I evaluated model performance, using accuracy, a confusion matrix, and a classification report. The model achieved an overall accuracy of approx. 85.7%, indicating that connected_areas is effective in distinguishing between the two groups. For the non-letter class (labelled as 0), there were 13 true samples, and the model correctly predicted 9 as non-letters, but misclassified 4 as letters. For the letter class (labelled as 1), there were 15 true samples, and the model correctly classified all 15 as letters. This shows that the model is very good at identifying letters, with 100% recall, but over-classifies some non-letters as letters.

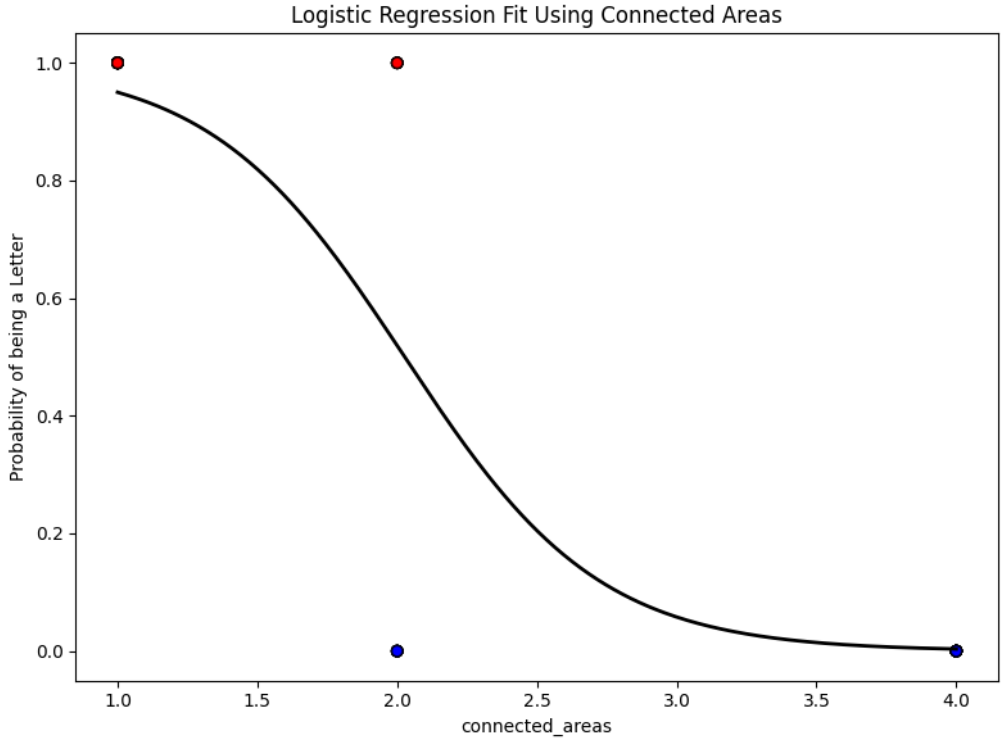

Figure 46: Graph to show the logistic regression fit

In Figure 46, the curve indicates that as connected_areas increases beyond 2-3, the model's confidence in the symbol being a letter drops sharply toward zero. The red points, indicating letters, mostly appear on the left with higher probabilities, whilst blue points, indicating non-letters, cluster on the right where the probability is close to zero.

Section 4.3 - Categorical Features via Median Splits

In this section, I transformed three continuous features into categorical features by applying a median split. To do this I computed the median for each of the chosen features and assigned a 1 if the value was above the median, and 0 if otherwise. I then mapped the original labels into three classes, "Letter", "Face", and "Exclamation Mark". Finally, I grouped the data by these classes and computed the proportion of "1"s for each split feature. The output can be seen below, in Figure 47.

class	split1	split2	split3
Letters	0.400	0.375	0.0125
Faces	0.575	1.000	0.0000
Exclamation Marks	0.550	0.000	0.2000

Figure 47: Proportion of "1"s by class for each split feature

Conclusions

In conclusion, the analyses demonstrate that features such as connected_areas, custom (normalised horizontal symmetry), and no_neigh_horiz are highly effective in distinguishing letters from non-letters. The regression and logistic models confirm their significance, with strong predictive performance and statistically significant differences. These findings highlight the importance of robust feature engineering and provide a good solid foundation for more advanced analyses in Assignment 2.

References

[1] J. Poskanzer, "PGM Format Specification," [Online]. Available: https://netpbm.sourceforge.net/doc/pgm.html.

[2] "NumPy Documentation," [Online]. Available: https://numpy.org/doc/2.1/reference/generated/numpy.argwhere.html.

[3] "SciPy - generate_binary_structure," [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.generate_binary_structure.html#scipy.ndimage.generate_binary_structure.