‘A quantitative analysis using feature engineering to train and predict AI models using Machine learning’

M.D’Arcy 40302488

13/03/2022

Introduction

This report guides the reader through the process of understanding AI and Machine learning. Today more and more industry related business are using AI to improve their daily work load. From using it to aid statistical analysis to training a model to caption what is in an image, Machine Learning certainly has cemented its purpose in an industry wide work.

Throughout my report, I will explore the creation of a portable bitmap image dataset from handwritten letters,faces,and exclamation marks, performance of feature engineering, a statistical analysis and finally the implentation of a machine learning model.All code is written in R through R studio

Section 1

———— DataSet ————

The project uses a dataset comprised of 140 individual images, these break down into 20 smiley face, 20 sad face, 20 exclamation mark and letters a -j with 8 of each. The symbols are created using gimp specifically contained in an 18x18 grid. Each set has their own criteria to ensure a base standard;

each face consists of two eyes, a nose and a mouth, each symbol has to be in an upright position,and must possess a style of “handwritten”.

———— Creation of dataset ————

The 140 files were exported as portable bitmap format (PGM), type ASCII. Once finished, I open the dataset in R Studio and read in the list of files and loop through them. Skipping the first four lines of the PGM file header and added the values iteratively to a matrix. To analyse the images we turn each one into a comma delimited file, any pixel value >128 is represented using a 1 and <128 is 0. Using this a matrix is created and stored in a .csv file.A for loop is created to circle through all 140 images.

Section 2

————Introduction————

In Section 2 we explore feature engineering,where we examine different aspects which could aid in the identification of our images. This array of characteristics describes the image,with 16 features in total. We define a pixel as having 8 neighbors, and create a matrix first from our csv files.

We create identifiers for the images using index and for the features using labels. I read a single CSV file and created a new test matrix with the same dimensions from section one. However, this time they needed transposed once or twice before the shape of the image normalized and I could run the various calculations on it. Once I had a functional matrix, I started from feature one.

————Feature 1 ————

We created a function called “nr_pix” which measures the number of black pixels in the image.

————Feature 2-5 ————

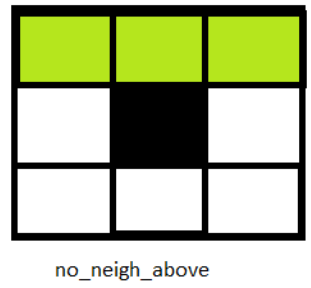
We build on the function “nr_pix” using it for functions “rows_with_1”, “cols_with_1”,“rows_with_3p” and “cols_with3p” which measures the number of black pixels in rows and columns specifically 1 and 3 or more pixels respectively. I made use of built in features such as sum,col sum, and row sum to have both ease of understanding and a documentated function. The first five features have small chamhges in the pattern but is simliar in caluculations.

———— Feature 6————

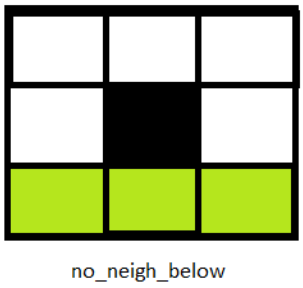
Following on, we examine aspect ratio which is The vertical distance, in pixels, between the topmost and bottommost black pixels in the image which are measured from the pixel centre called the height. The horizontal distance, in pixels, between the leftmost and rightmost black pixels in the image measured from the pixel centre called width.I again made use of R built-in function this time using the which, min and max functions to iteratively search thorough the columns or row depending on if I was calculating height or width. I .This feature is width/height.

————Feature 7-14 ————

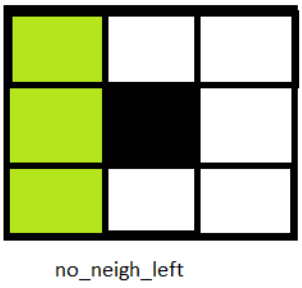
The next 7 features examine the neighbors of the pixel,following on similarly from how we first created “nr_pix” we first create “neigh_1” which is the number of black pixels with only 1 corresponding pixel in the 8 tiles around it. From this we take each the next features and explore its pixels neighbours shown in the diagram where the green represent pixel spaces without a black pixel present.



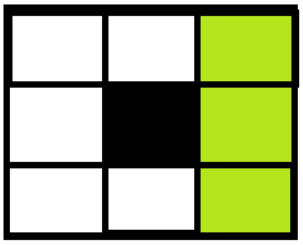
no_neigh_above



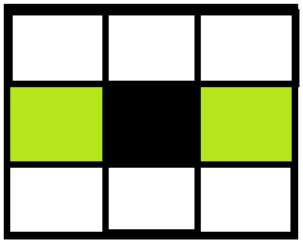
no_neigh_below



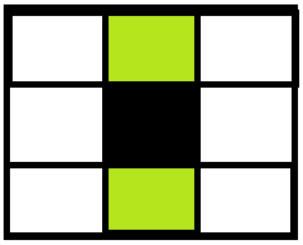
no_neigh_left



no_neigh_right



no_neight_horiz



no_neigh_vert

We used a counter to count the number of times a specific pattern was seen within a given matrix. We then created a nested for loop that would search through rows and columns using the index of for nested loop and incrementing by one to depending on the pattern. This was all wrote within a single if statement and printed the counter at the end of the loop

Most important thing was when I was tracing my code.I wanted to able to reuse my code and have less confusing code,which is where I impleneted y vertical and horizontal functions.

———— Feature 15-16 ————

The last two features involve the use of the clump and raster libraries where we examine “eyes” and “connected_areas” both functions where a region of white space is enclosed with black pixels.

I encountered difficulty both understanding the raster and clump libraries, and transforming my information into a usable function. The program was returning -inf until I released I needed to create a new matrix to invert it to.

————Custom Features ———— For my custom features I examined different structures of matrices. I first thought that an inclusion of upper and lower triangular matrices would help in identification of curves present in “a”,“b” and smiley faces,

However, I could only hard-code the matrix and was only identified a handful of times in further look into the features.csv. I abandoned these features and pursued a separate type of matrix analysis called “stocastic matrix”, Which examines all the pixels in the matrices and then returns the probability of a black pixel present. From this I wanted to link the connected areas aspect and use it as a feature. Further analysis occurs in section 3.

————Creation of Features CSV ————

Once the code had been tested, a function was created to could call them iteratively inside a for loop which I used to create the final feature csv

The data collected from our 16 features is stored in Features.csv using the write table function.

————Reflection ————

For this section I had difficulty visualizing the neighbor pixels and the concept of why we needed each function. It wasn't until section 3 that I understood through the statistical analysis the importance of each feature. Upon discussion with classmates about section 2 it appeared that many choose not to write it as functions. For my code personally I prefer to use functions as it allows me to return and test each feature individually. I then called each function with the matrix, where each function has a row matching 1 of the 140 images each. I then used the bind function to add column titles and exported to the csv. Both the data frame and csv is useful for section 3 analysis.

Section 3 Introduction

Section 3 of our report dives into the Statistical analysis of feature data. To explore which features are important for distinguishing between our handwritten symbol images we perform descriptive statistics, hypothesis testing and represent the results in visualizations. Further to this analysis,an interesting comparison between 1 pixel and 3 pixels or more with both rows and columns respectively would be interesting to confirm the possibilities set out above in our figure captions.Another graphical represenation of the above histograms is exploring them as qq plots which is a plot of the quantiles of the first data set against the quantiles of the second data set. A 45-degree reference line is also plotted. If the two sets come from a population with the same distribution, the points should fall approximately along this reference line. The more points clung to the line the closer it is to a normal distrubtion.

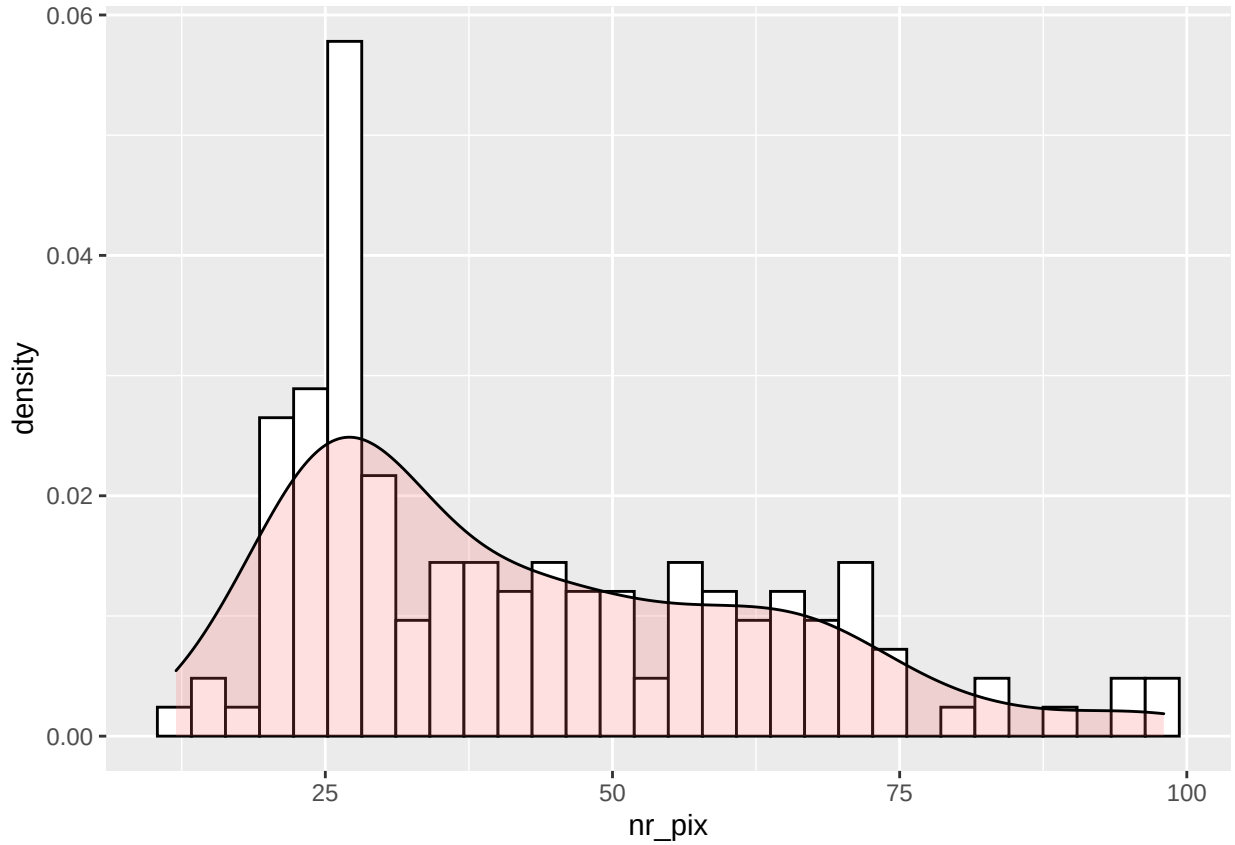
Section 3.1

We first examine the results of our 6 features using histograms with a density overlay to help aid understanding of the distribution.

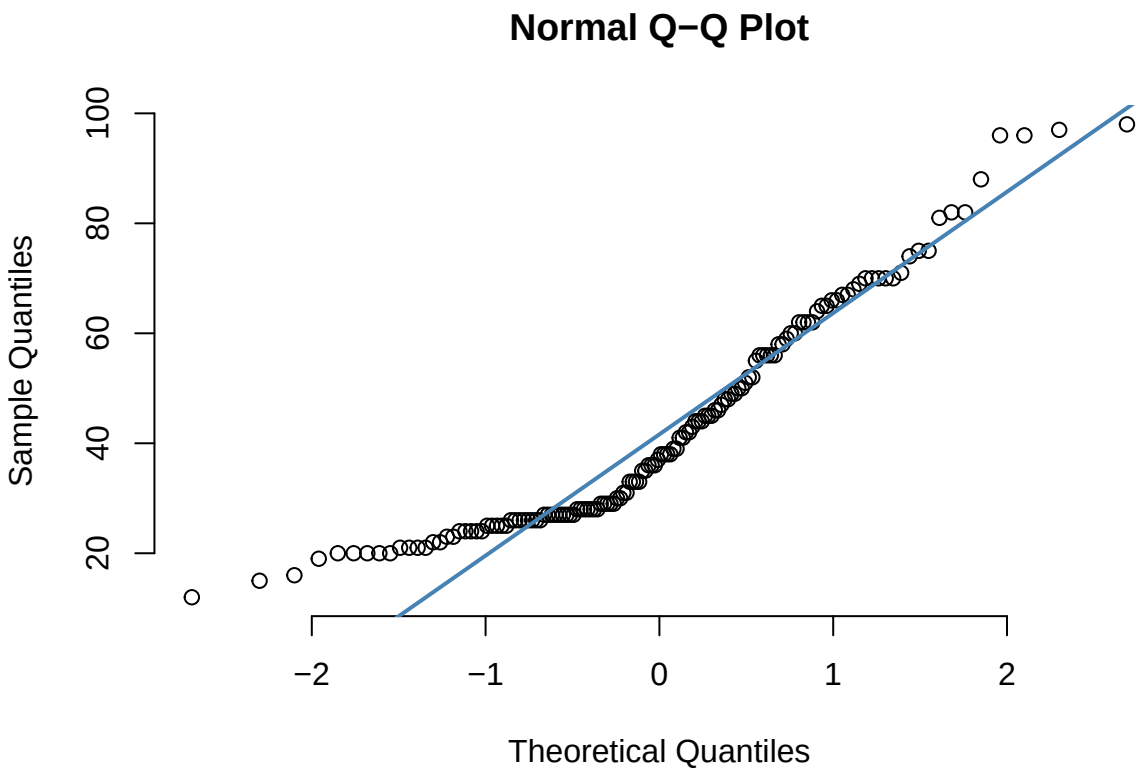
##Nr_pix (Number of black pixels)

```
p
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
 qqnorm(data$nr_pix, pch=1, frame=FALSE)
qqline(data$nr_pix, col = "steelblue", lwd=2)
```

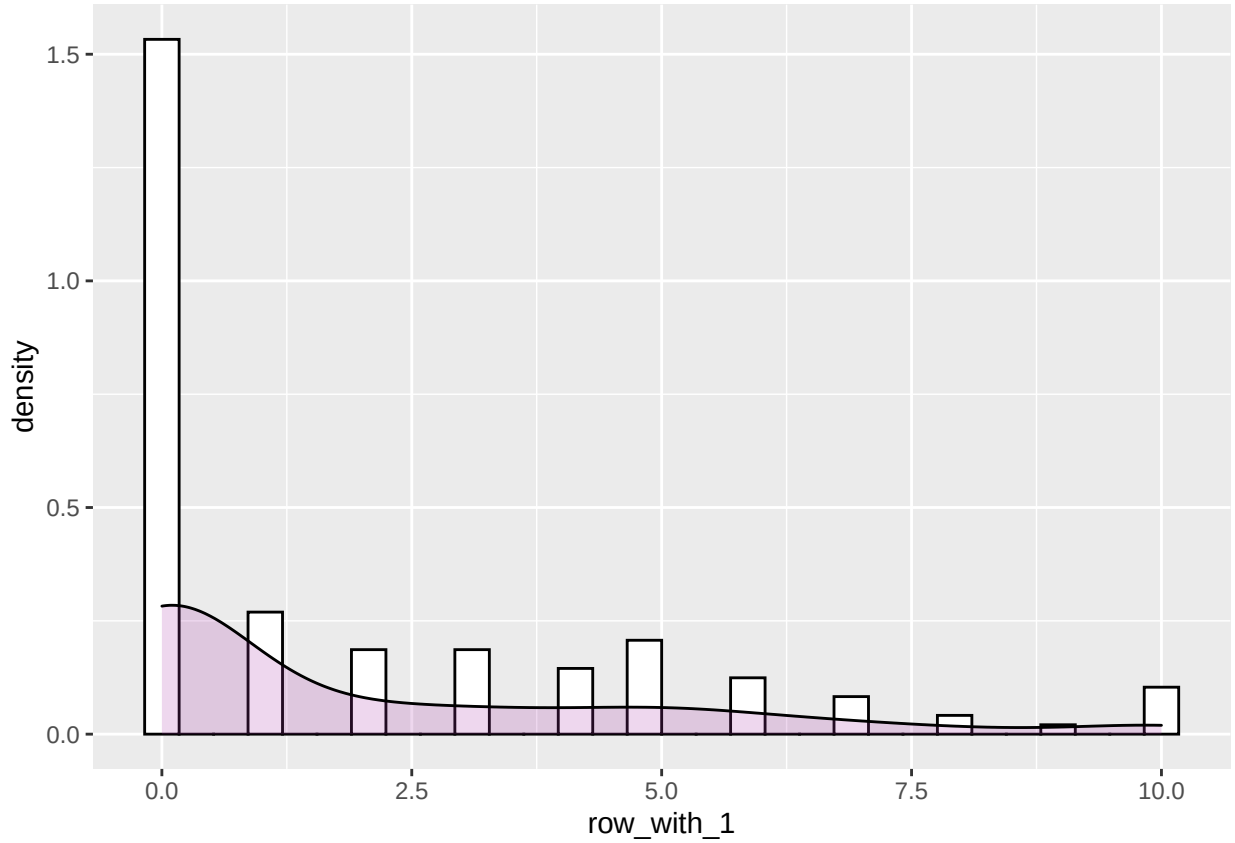


Fig(1) shows nr_pix represented as a histogram.The function has been successful in collecting its data which means that the following histograms can be anaylsed and explored. The histrogram appears to be mutlimodal with 1 prominent peak,exploration of the QQ-Plot shows it to be normally distributed, The three peaks could possibly be the three classfications, letters,faces and exclamation mark as they are similar to each other.

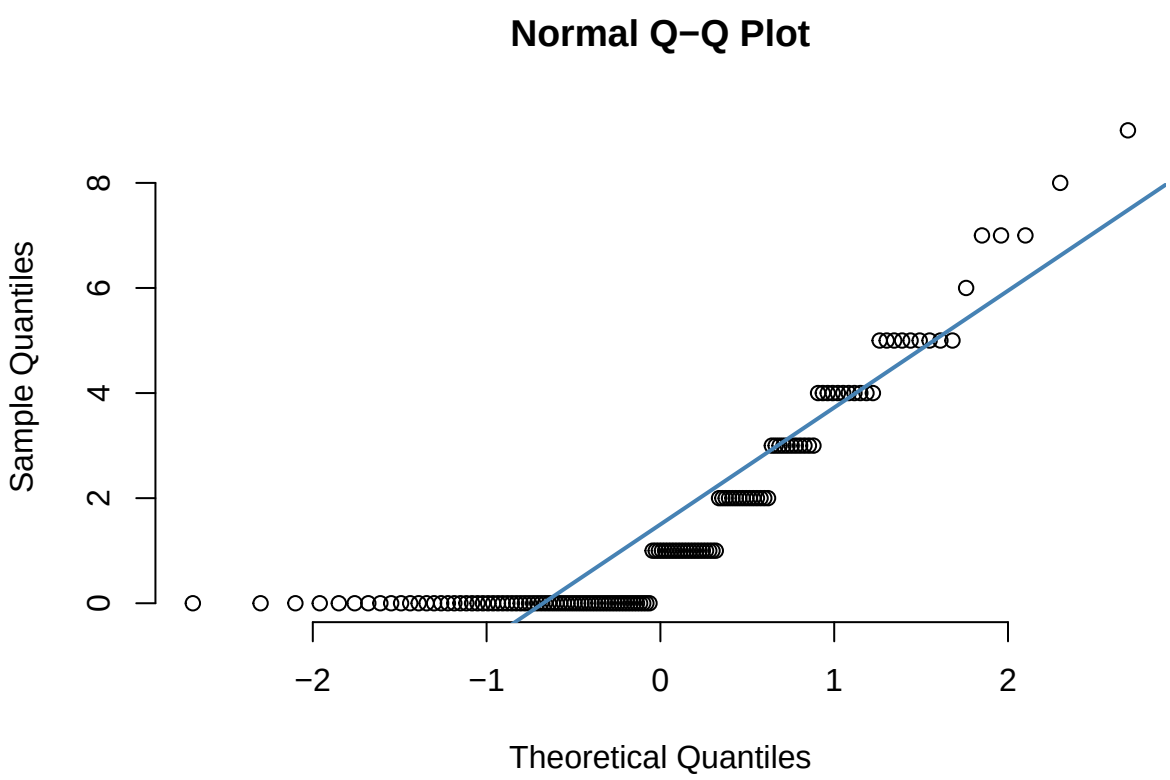
##Row_with_1p(Rows with only 1 pixel)

```
p1
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
qqnorm(data$cols_with_1, pch=1, frame=FALSE)
qqline(data$cols_with_1, col = "steelblue", lwd=2)
```

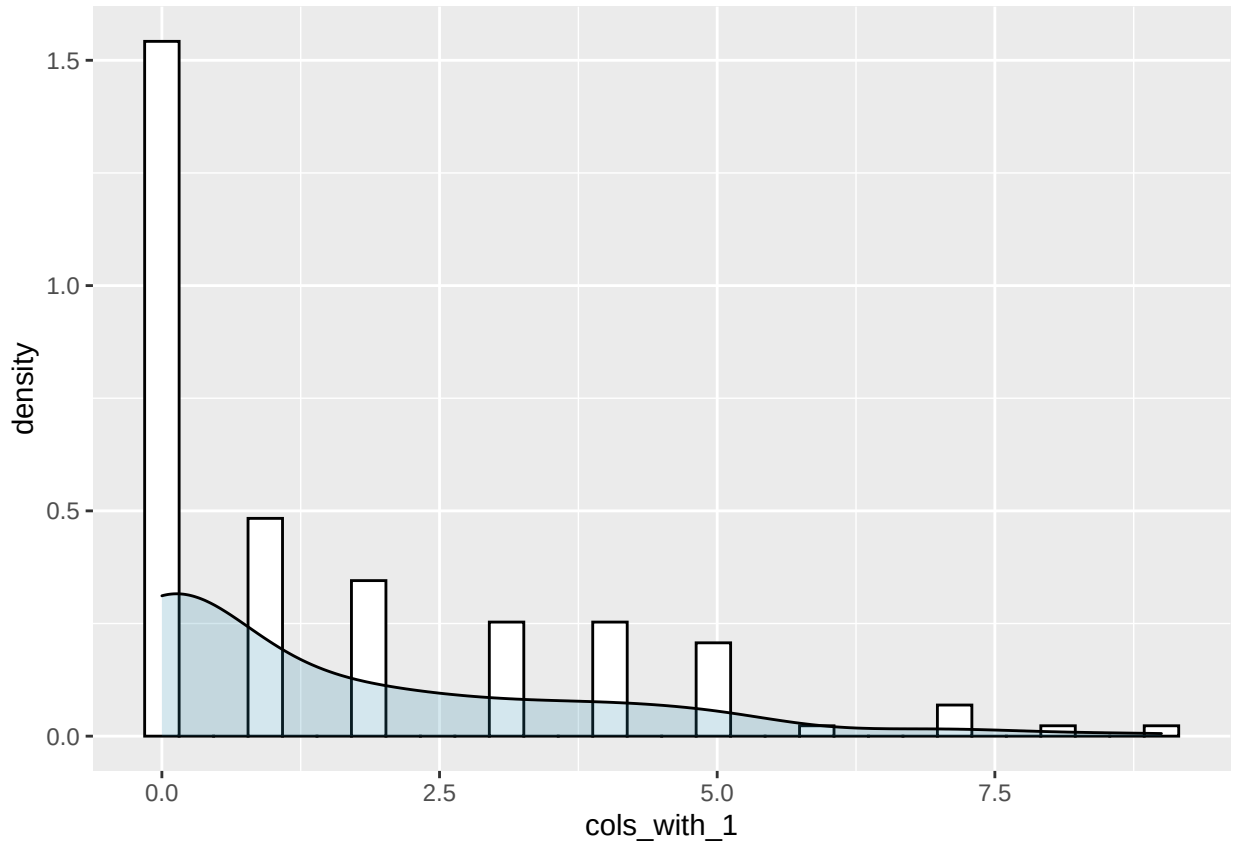


Fig(2) rows_with_1 : The density is low which is expected, any symbol which contains a curve would count at least 2 pixels in the row so the only symbols which could be represented here is tail ended letters such as b,f,j,e,the eyes in the faces.There appears to be a right skew present.The data is poorly fitted to the distribution line in the qq-plot.

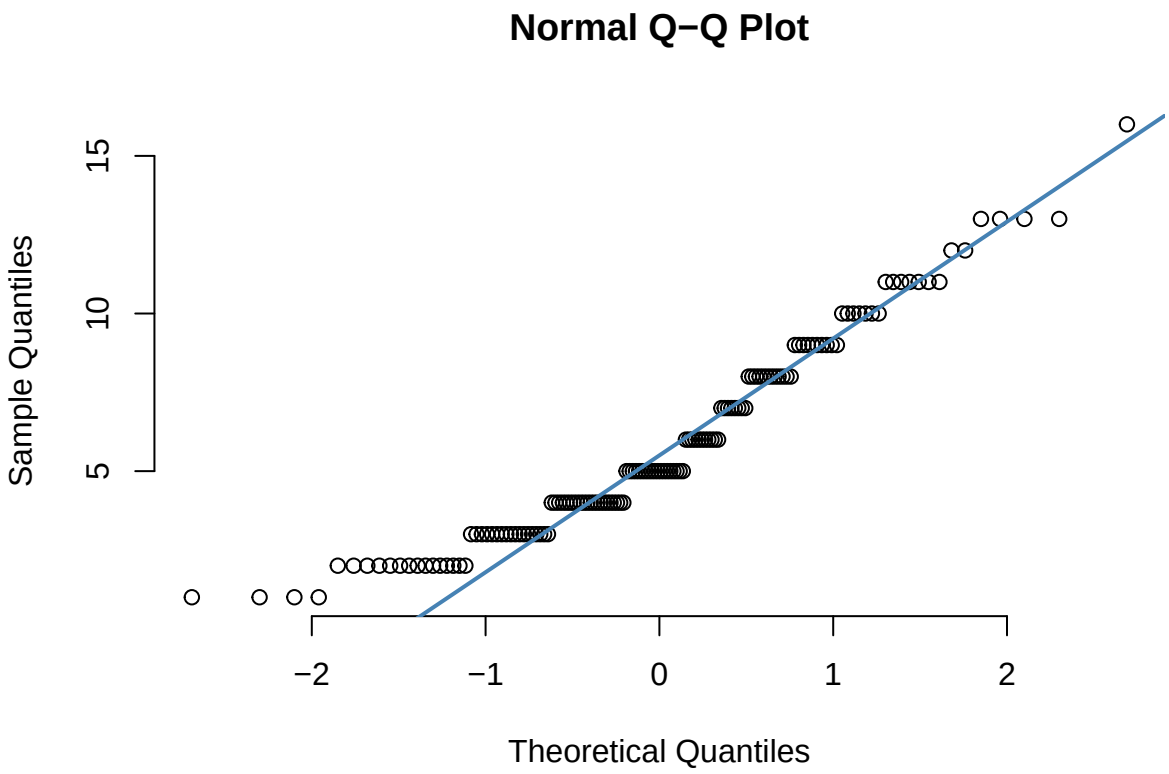
##Cols_with1p(Columns with 1 pixel)

```
p2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
qqnorm(data$cols_with_3p, pch=1, frame=FALSE)
qqline(data$cols_with_3p, col = "steelblue", lwd=2)
```

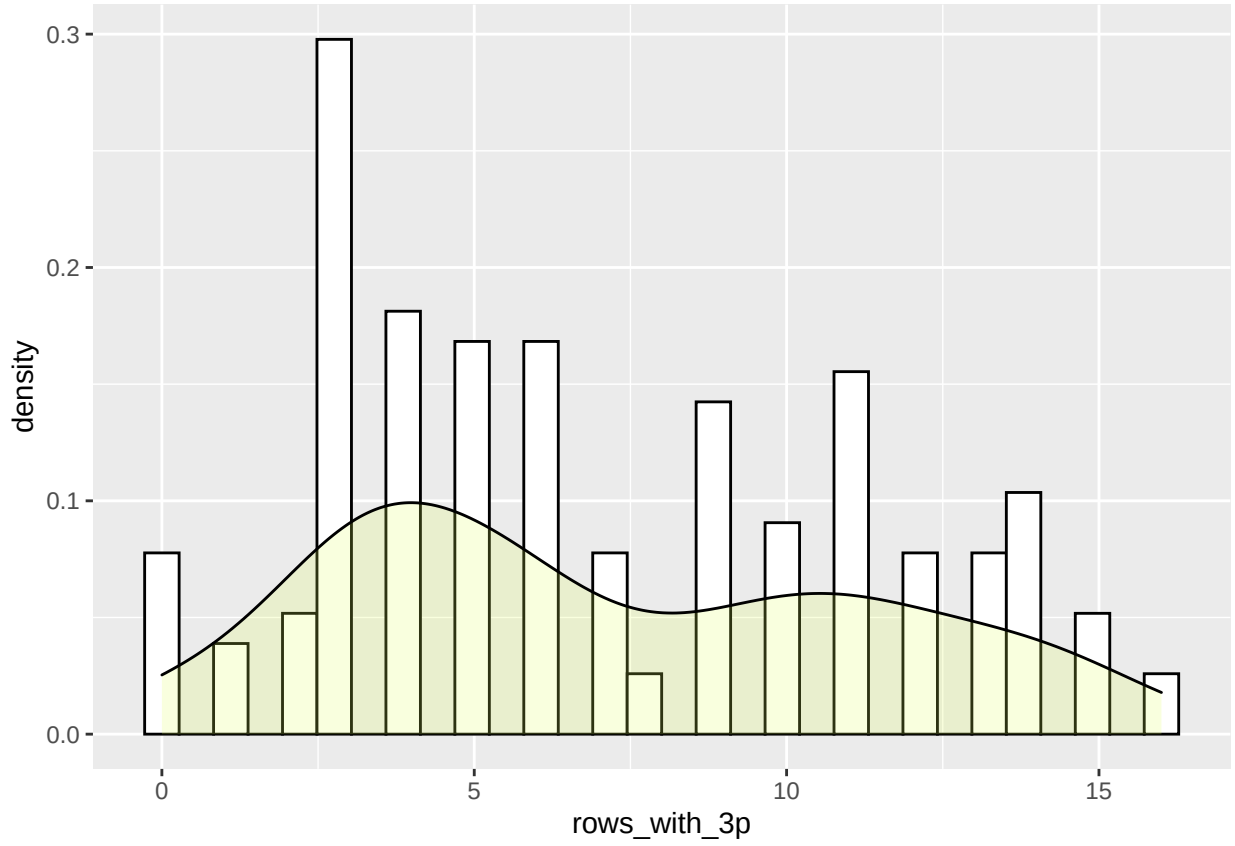


Fig(3) cols_with_1 : The density is low,which is expected, further to the exclusion listed in in Fig(2) but also the overlap which would neagate the tail ended letters,curve letters,faces,exclamation mark theres only a few possible cases where we can see cols_with_1 such as open ended curves, "j", overly wide mouths.Also simliar to Fig(2)it is skewd right but more fitted to the distribution line.

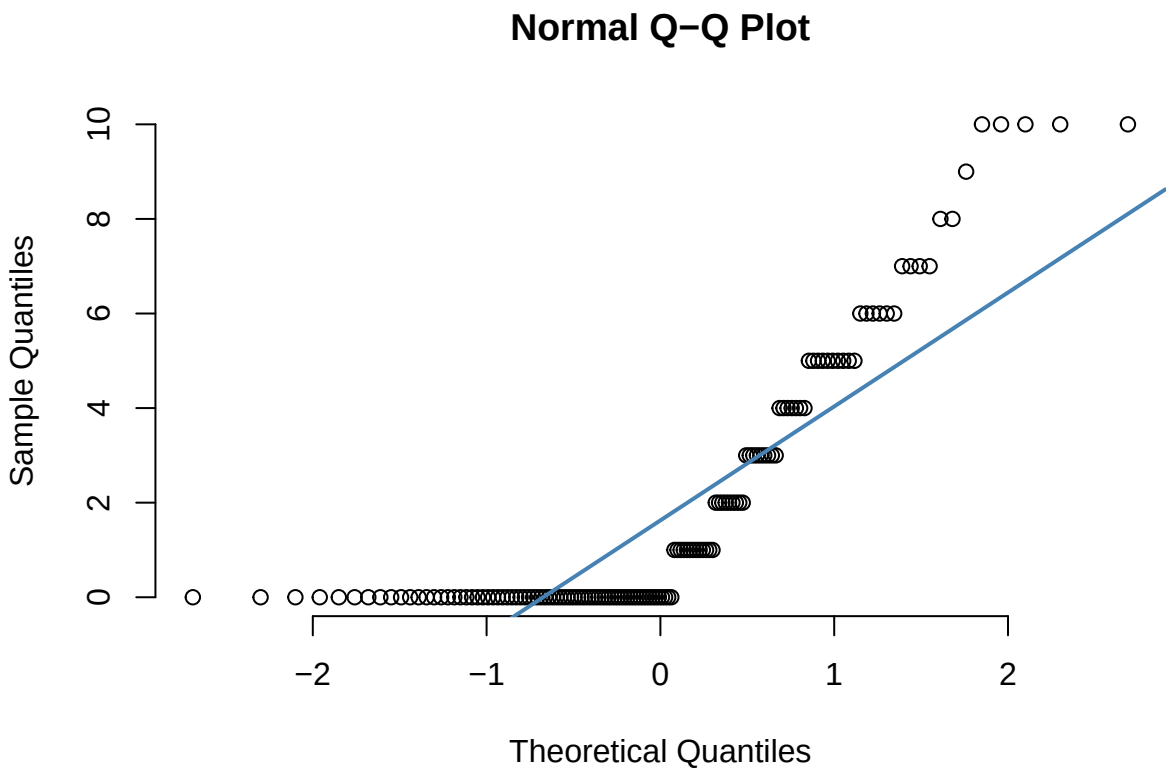
##Rows_with3p(Rows with 3 or more pixels)

```
p3
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
qqnorm(data$row_with_1, pch=1, frame=FALSE)
qqline(data$row_with_1, col = "steelblue", lwd=2)
```

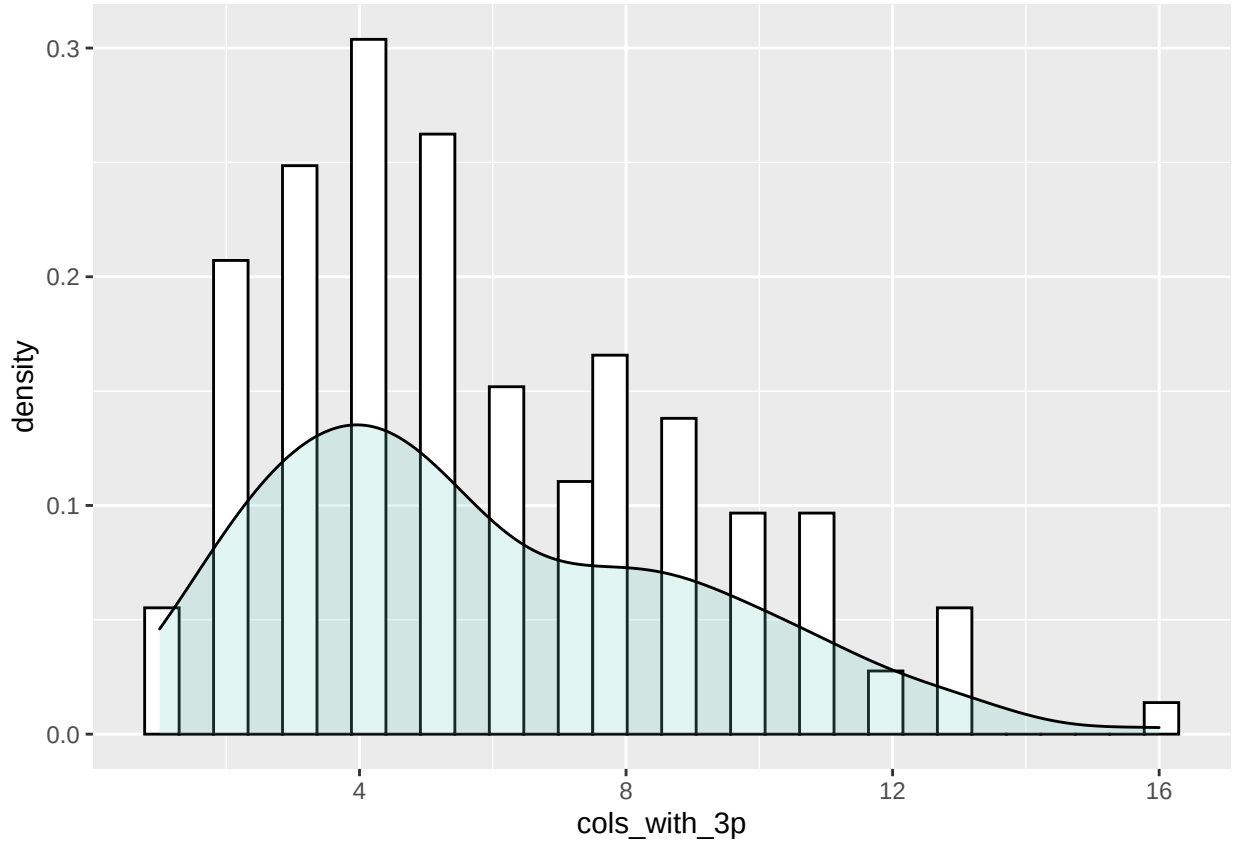


Fig(4) rows_with_3p: The Density is consistently spread across all symbols which is expected,if we examine the graph we see 2 spikes meaning its is bimodal, where if we go back to Fig(1) we discussed how tail end letters would be included such as h,k,l,j,i which accounts for this dip.The data is fitted loosley to the distribution line expect for a small portion.

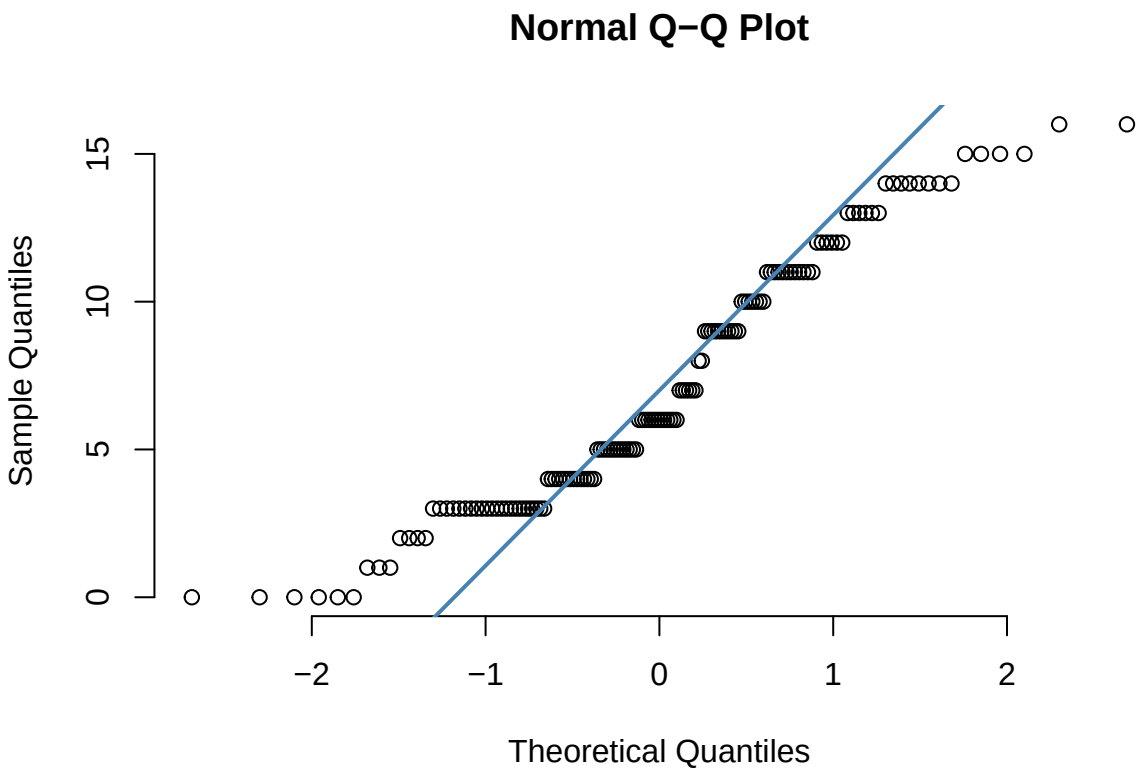
##Cols_with3p(Columns with 3 or more pixels)

```
p4
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
qqnorm(data$rows_with_3p, pch=1, frame=FALSE)
qqline(data$rows_with_3p, col = "steelblue", lwd=2)
```

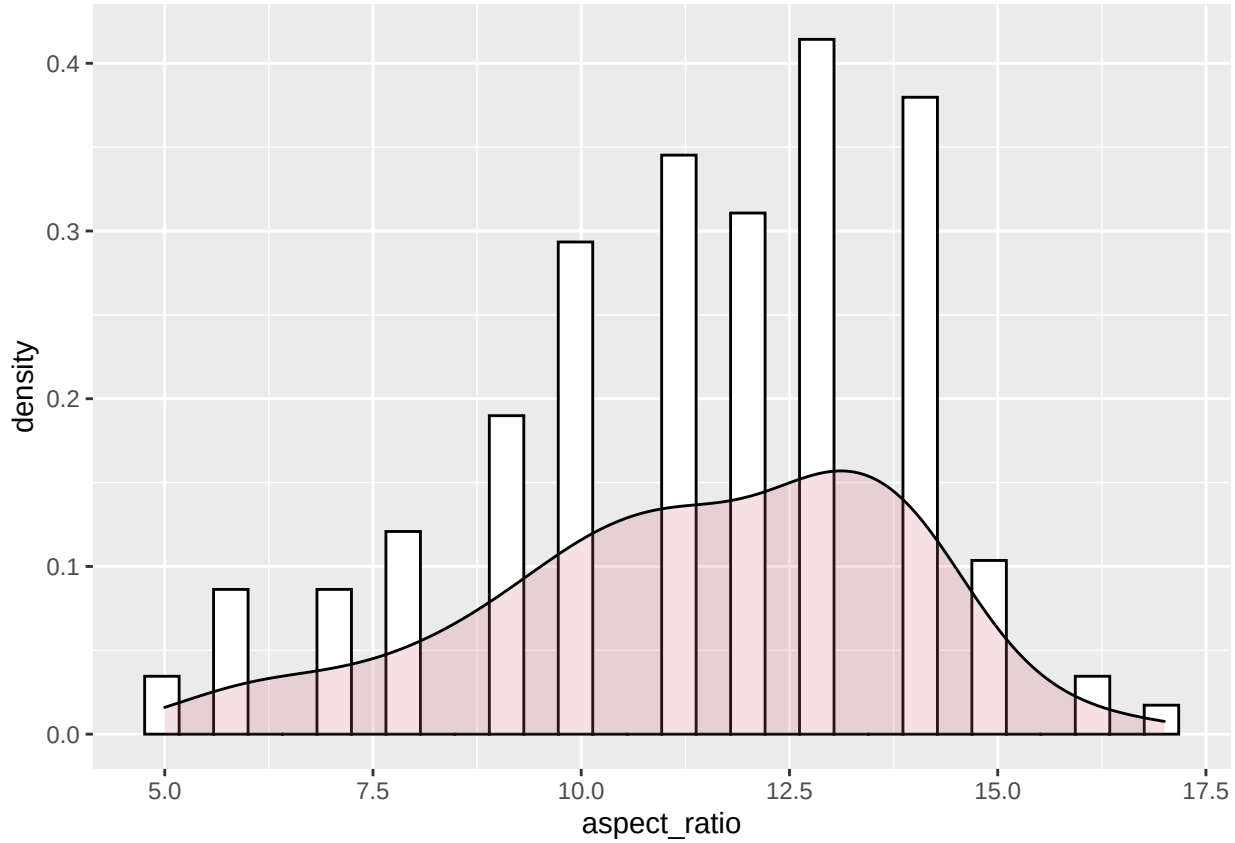


Fig(5) cols_with_3p : Similarity to Fig(4) we notice an even spread, however it is skewed to the right and drops off. The explanation for this being the exclamation mark has at most 1-3 columns so would only be able to have at most a small spread compared to the wideness achieved by the other symbols.This is more fitted to the density line comaped to fig(4)

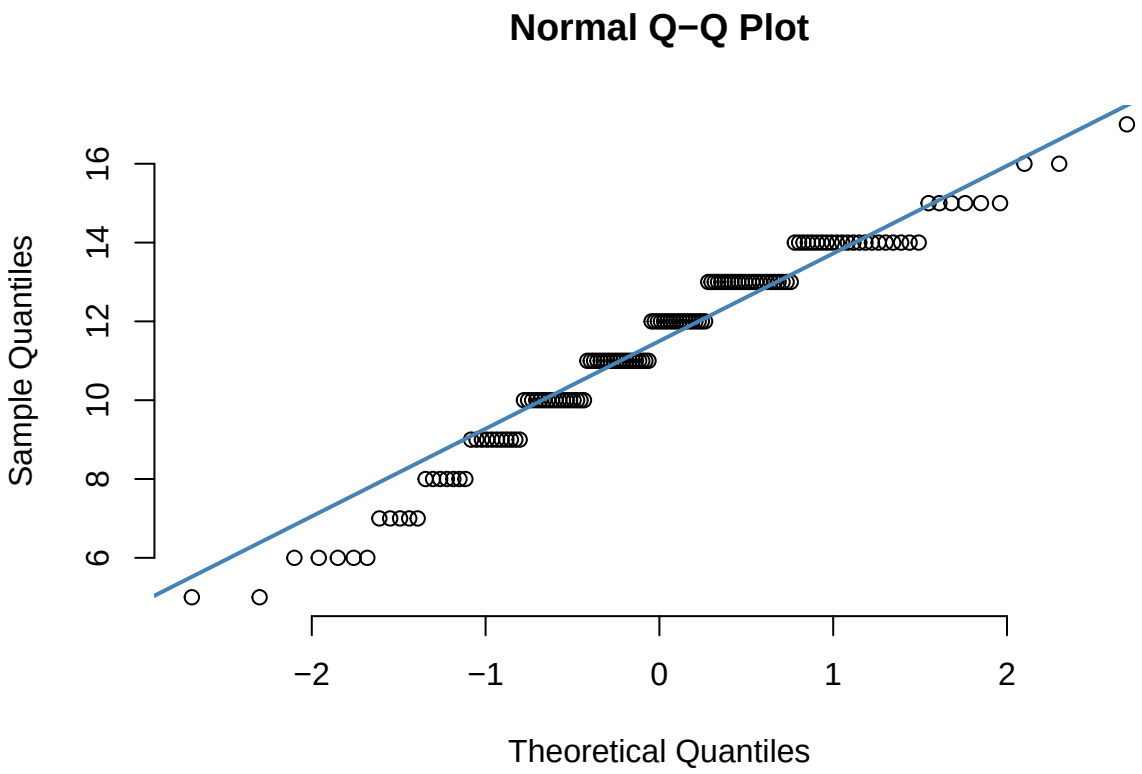
##Aspect_ratio

```
p5
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
qqnorm(data$aspect_ratio, pch=1, frame=FALSE)
qqline(data$aspect_ratio, col = "steelblue", lwd=2)
```



Fig(6) aspect_ratio: For this we expected a curve which was dense but level. Aspect ratio measures the relationship between width and height. From this we see that most symbols have an almost consistent ratio meaning it is square like in shape roughly such as a,c,e,faces,and exclamation points.This is realtivly normally distributed to the Q-Q plot.

Section 3.2

In Section 3.2 We use summary statistics particularly mean, median, range and interquartile range. After we explore the averages,we look to explore the standard deviation. For this we split the data into two sets: "data_letters" and "data_nonletters" and examine them below using the table. We will examine the tables individually before we compare and contrast.Next We explore the standard deviation, Standard deviation is a statistic that measures the dispersion of a dataset relative to its mean,below is the fomrulatic expression.

$$s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}} \tag{1}$$

s = Sample S.D. x_i = individual value $\bar{x}$ = Sample mean n = Sample size

If the data points are further from the mean, there is a higher deviation within the data set; thus, the more spread out the data, the higher the standard deviation. A standard deviation close to zero indicates that data points are close to the mean, whereas a high or low standard deviation indicates data points are respectively above or below the mean.

```
kable(output_letters_3.2,capation = "Statsical analysis into letters ")
```

	nr__pix	row__with	cols_1	with	rows_1	with	cols_3p	with	aspect__ra	tio	neigh__le	ft	neigh__ri	ght	neigh__ab	ove	neigh__be	low	neigh__ve	rt	neigh__horiz
letters__mi	12.000000	0.000000	0.000000	0.000000	1.000000	5.000000	0.000000	1.00000	3.000000	0.130434	8	1.000000	1.000000								
letters__q1	24.000000	0.000000	0.000000	3.000000	3.000000	9.000000	2.000000	6.75000	6.000000	0.299074	1	4.000000	4.000000								
letters__me	27.000000	0.000000	1.000000	4.000000	4.000000	11.000000	0.000000	9.00000	9.000000	0.425824	2	6.000000	6.000000								
letters__mea	28.837500	0.037500	0.562500	4.062500	4.225000	10.612500	2.750000	9.01250	9.025000	0.463000	2	6.875000	6.450000								
letters__q3	33.000000	0.000000	3.000000	5.000000	5.250000	13.000000	6.000000	11.00000	11.250000	0.585483	9	9.000000	8.000000								
letters__ma	56.000000	0.000000	7.000000	11.000000	11.000000	16.000000	12.000000	16.00000	17.000000	1.090909	12	20.000000	20.000000								
letters__sd	8.220757	2.965519	1.741844	2.235183	2.164004	2.587672	2.963811	3.35455	3.337455	0.223719	5	3.756429	3.655757								

```
kable(output_nonletters_3.2,capation = "Statsical analysis into letters ")
```

	nr__pix	row__with	cols_1	with	rows_1	with	cols_3p	with	aspect__ra	tio	neigh__le	ft	neigh__ri	ght	neigh__ab	ove	neigh__be	low	neigh__ve	rt	neigh__horiz
nonletters	27.000000	0.000000	0.000000	5.000000	3.000000	8.000000	0.000000	0.000000	7.000000	0.048387	1	2.00000	1.000000								
nonletters	48.475000	0.000000	0.000000	9.000000	5.000000	11.000000	6.000000	12.000000	12.000000	0.230219	8	8.00000	8.000000								
nonletters	61.000000	0.000000	0.000000	11.000000	8.000000	13.000000	8.000000	14.000000	13.500000	0.401219	5	11.000000	12.000000								
nonletters	61.150000	0.083333	3.333333	11.166667	7.000000	12.433333	7.388333	14.300000	14.100000	0.372718	2	10.733311	11.450000								
nonletters	70.000000	0.000000	2.250000	13.000000	10.000000	14.000000	10.250000	17.000000	16.250000	0.464761	6	14.000001	15.250000								
nonletters	98.000000	0.000000	9.000000	16.000000	16.000000	17.000000	13.000000	28.000000	27.000000	0.695652	29	19.000002	20.000000								
nonletters	15.879063	0.340328	3.875487	2.718996	3.047644	1.977558	3.622396	4.187552	3.657822	0.144927	24	6.364210	4.770762								

———————Letters ———————

When studying our tables I have moved from left to right as if it is normally distrubted to be able to view the boundraies. we notice trends emerging within the data set. One noticeable example is the similarity between the mean and median values which suggests our data is symmetrically distributed. It is interesting to see the difference in values <1, <2 and >2. With the majority being less than 1, these results seem like valuable features for symbol analysis as it has lower differences but overall a great distinable feature if our data was thrown into a set of new random images. We can however only choose which features are valuable in identification when we compare it against non_letter data.

For the interquartile range of an observation variable is the difference of its upper and lower quartiles. It is a measure of how far apart the middle portion of data spreads in value. Within this inter-quartile range, favourable data occurs with lower results. We can discount nr_pix because the data is dense and would produce a large inter-quartile range. What is interesting is the low inter-quartile ranges for both rows and columns that have 1 pixel. I expected it to have a larger inter quartile range. I suspect along with the range calculation of the group that this is wrong and rule them both out in this analyis.

Overall I am happy with the leading on the mean and median values with guidance from the range and inter-quartile range.

———————Standard deviation———————

From the below set of letters, I have noticed 3 values <1 which are cols_with1,cols_with3p,no_neigh_vert.This is suprising and a correlation is present between data collected and organised via inspection of pixels in columns. This is also present in non_letters too,both aspect ratio has a low standard deviation also.

———————Non-Letters———————

Next We examine the sumery statistics in the non_letter subset which contains the faces and exclamation marks. We notice that the difference between median and mean here is not consistent and is more present being >1. We meet a lot more zero values here this means that if the right features are selcted they are not applaicable for any of the summery analysis therefore being a faovarabe marker.

———————Features ———————

Particualry 3 features met all of the desrible criteria outlined above,this was row_with_1,rows_with3p and no_neigh_above.

————————Row_with_1————————

I selcted this one due to the favaorble closeness of mean and median but more the fact it presents 0 for 4 categories comapred to only 1 in letters.

————————Rows_with_3p————————

Noticably in my standard devaition particylar low results were any row analysis I pair this with also the consistent data presneted from the summery statstics

————————no_neigh_vert ————————

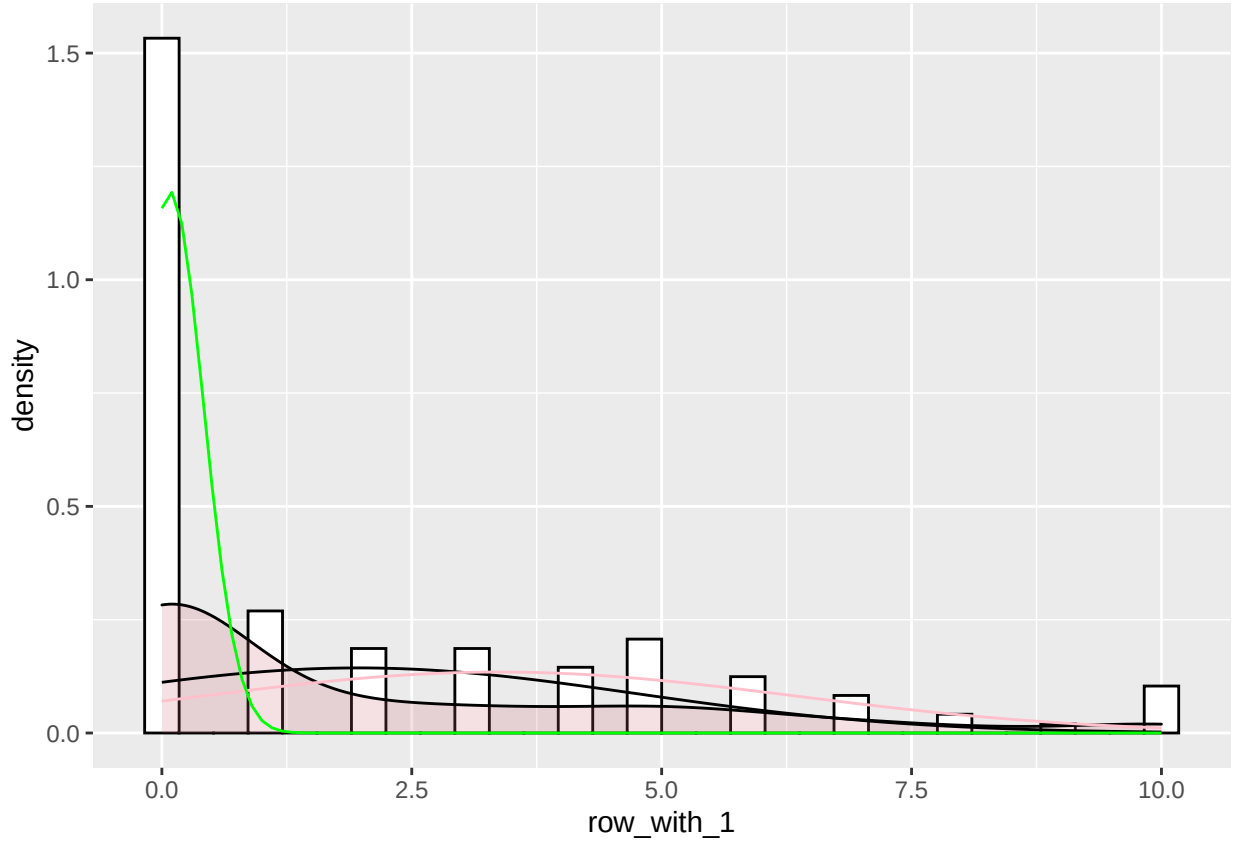
This feature discrimates letters from non-letters as all non-letters have an overhaed pixel this is suported by the summery statistics and nice whole numbers

####Section 3.2 graphs

————————Legend ———————— Pink line = Letters Green line = non letters black = entrie set

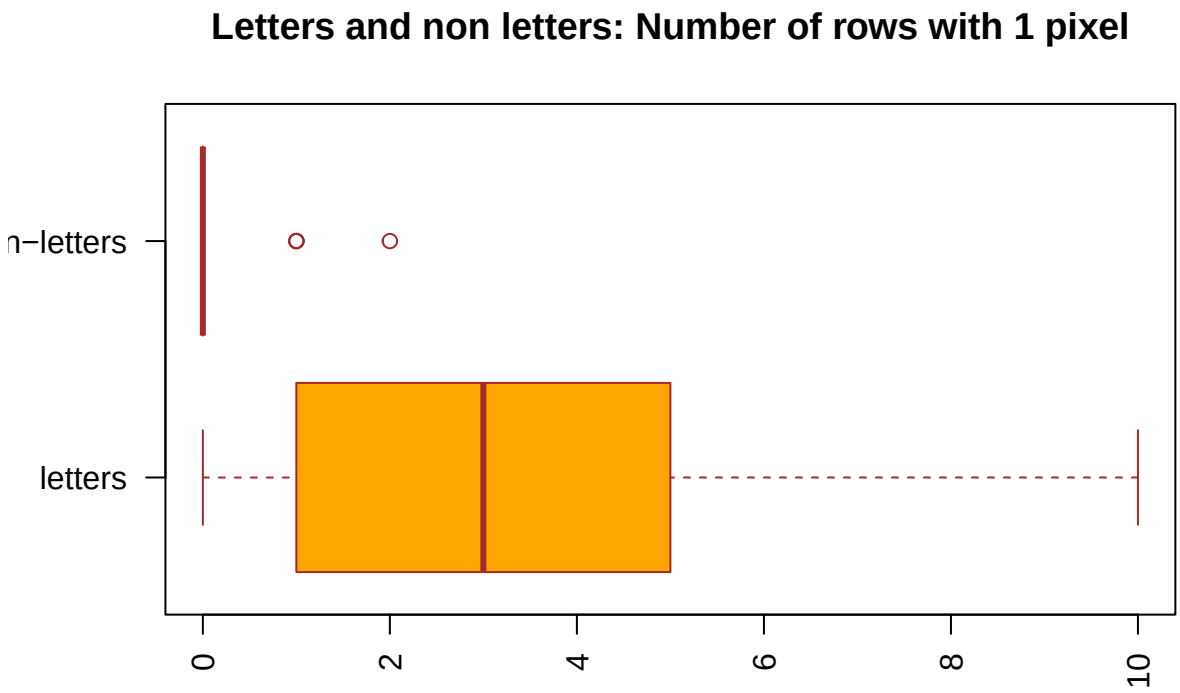
```
f1
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
boxplot(letters_row_with1, nonletters_row_with1,
        main = "Letters and non letters: Number of rows with 1 pixel",
        at = c(1,2),
```

```
        names = c("letters", "non-letters"),
        las = 2,
        col = c("orange","lightblue"),
        border = "brown",
        horizontal = TRUE,
        notch = FALSE)
```



fig(7) As seen above there appears to be a higher density of non_letters,then it drops off and is not present,We see a small slight rise in letters.This could be useful when we distinauble the boundary and use it for classfication.The box plot shows us that non letters is very contained and letters has most of its data not over lapping.This has been a successfull choice

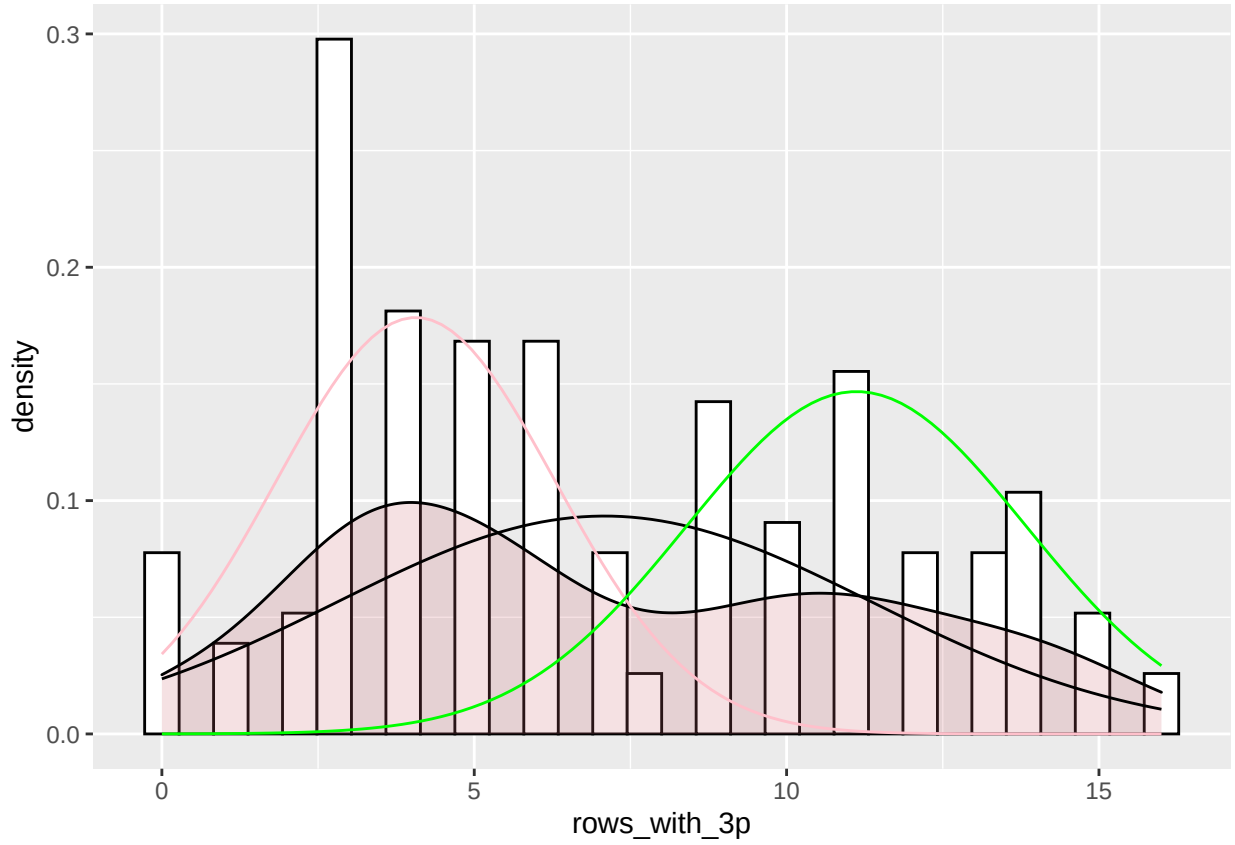
```
geom_histogram(color="black", fill="white",binwidth = 2)
```

```
## geom_bar: na.rm = FALSE, orientation = NA
## stat_bin: binwidth = 2, bins = NULL, na.rm = FALSE, orientation = NA, pad = FALSE
## position_stack
```

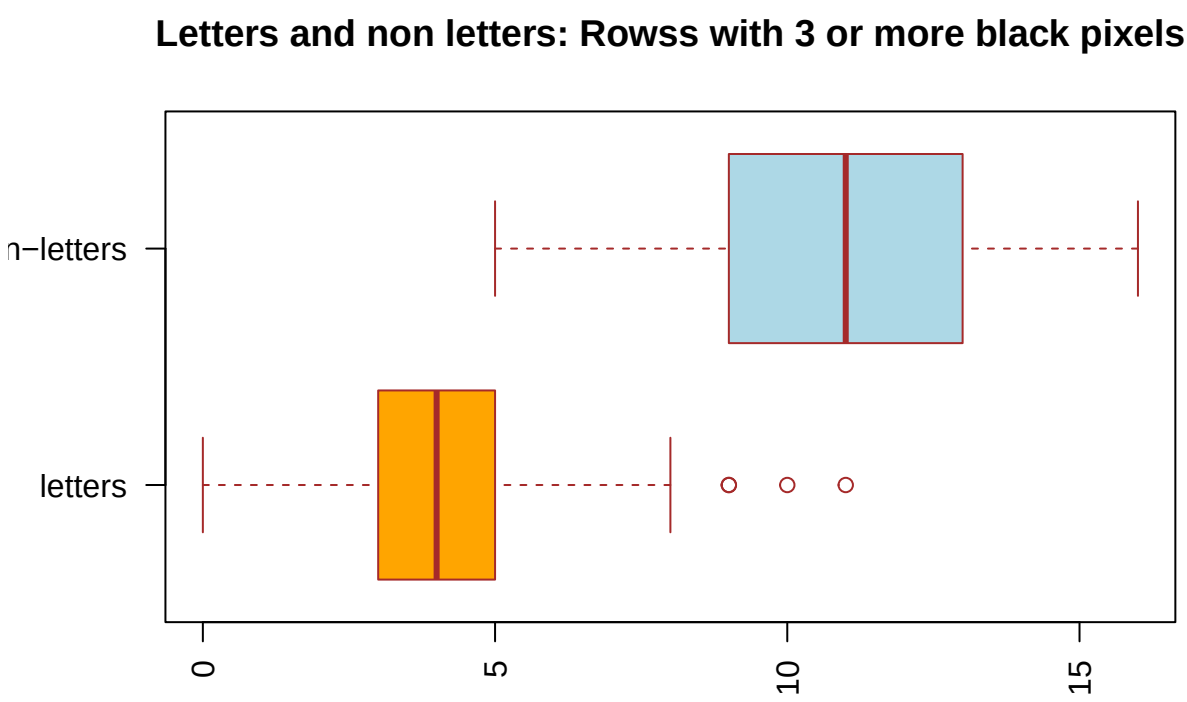
```
f2<-ggplot(dataFeature, aes(x=rows_with_3p)) +
geom_histogram(aes(y=..density..), colour="black",fill="white")+
geom_density(alpha=.2, fill="#D16A76")  +
 stat_function(fun = dnorm, args = list(mean = mean(data$rows_with_3p), sd = sd(data$rows_with_3p)))+
  stat_function(fun = dnorm, args = list(mean = mean(data_letters$rows_with_3p), sd = sd(data_letters$
  stat_function(fun = dnorm, args = list(mean = mean(data_nonletters$rows_with_3p), sd = sd(data_nonle
```

```
f2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# prepare the data
letters_rows_with_3p <- data_letters$rows_with_3p
nonletters_rows_with_3p <- data_nonletters$rows_with_3p

# box plot to compare the number of diagonals data
boxplot(letters_rows_with_3p, nonletters_rows_with_3p,
        main = "Letters and non letters: Rowss with 3 or more black pixels",
        at = c(1,2),
        names = c("letters", "non-letters"),
        las = 2,
        col = c("orange","lightblue"),
        border = "brown",
        horizontal = TRUE,
        notch = FALSE
)
```



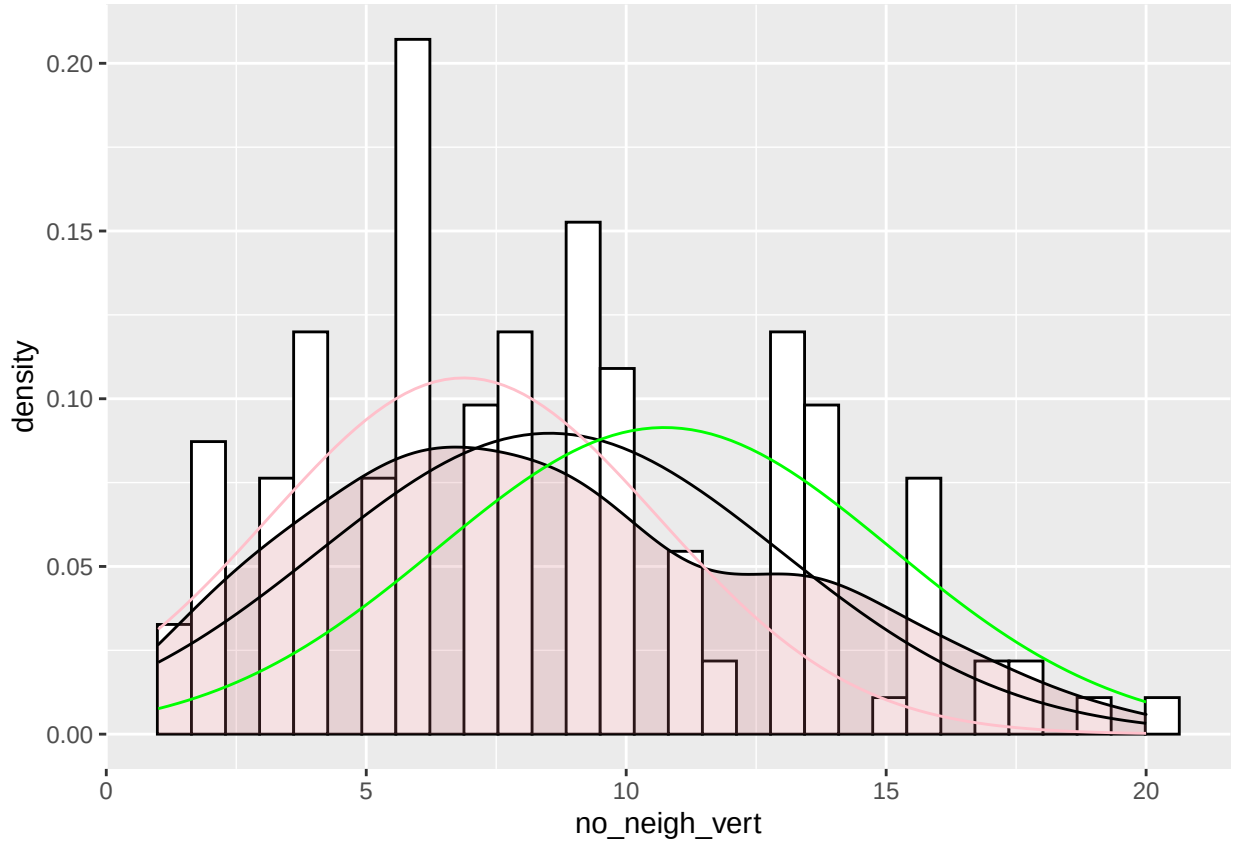
Fig(8) Th black line suggests that when combined both data sets produce a normal distribution,howver when separted they are unimodal which is helpful for separtion between the two sets.From the box plot we see no overlap of data from the mean and where ost of the data is equaly distributed does not overlap.

```
geom_histogram(color="black", fill="white",binwidth = 2)
```

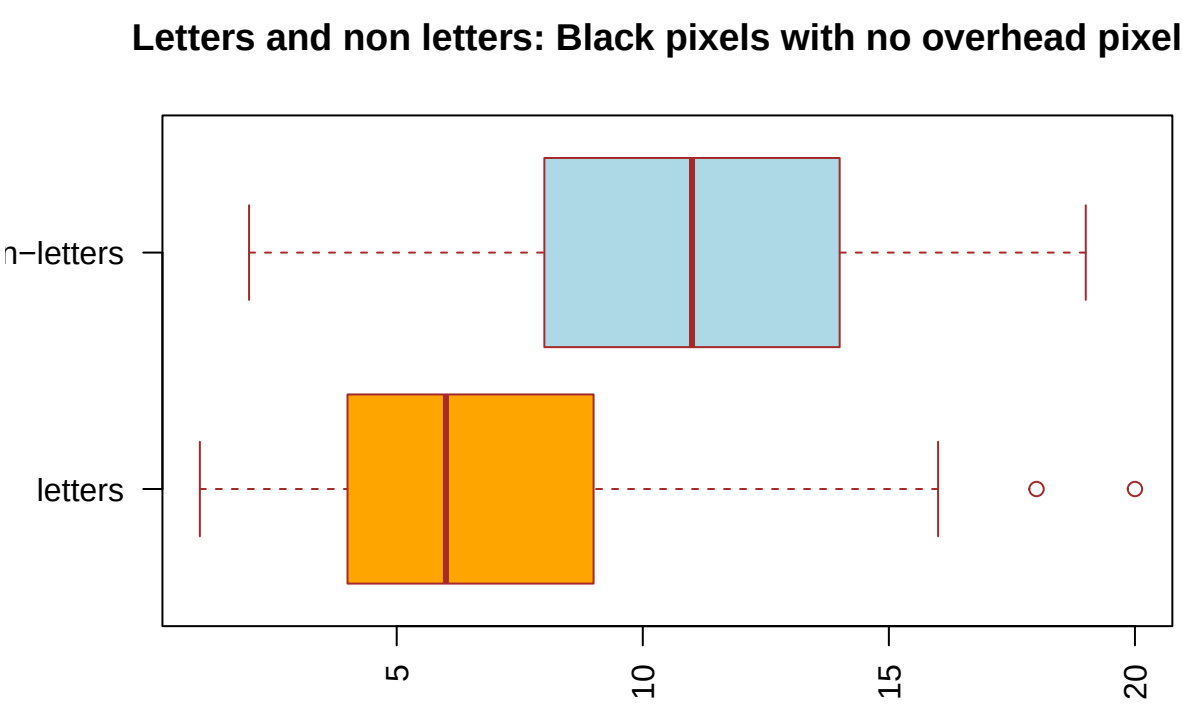
```
## geom_bar: na.rm = FALSE, orientation = NA
## stat_bin: binwidth = 2, bins = NULL, na.rm = FALSE, orientation = NA, pad = FALSE
## position_stack
```

```
f3<-ggplot(dataFeature, aes(x=no_neigh_vert)) +
geom_histogram(aes(y=..density..), colour="black",fill="white")+
geom_density(alpha=.2, fill="#D16A76")  +
 stat_function(fun = dnorm, args = list(mean = mean(data$no_neigh_vert), sd = sd(data$no_neigh_vert)))+
  stat_function(fun = dnorm, args = list(mean = mean(data_letters$no_neigh_vert), sd = sd(data_letters$
  stat_function(fun = dnorm, args = list(mean = mean(data_nonletters$no_neigh_vert), sd = sd(data_nonle



f3
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# prepare the data
letters_no_neigh_vert<- data_letters$no_neigh_vert
nonletters_no_neigh_vert <- data_nonletters$no_neigh_vert

# box plot to compare the number of diagonals data
boxplot(letters_no_neigh_vert, nonletters_no_neigh_vert,
        main = "Letters and non letters: Black pixels with no overhead pixels",
        at = c(1,2),
        names = c("letters", "non-letters"),
        las = 2,
        col = c("orange","lightblue"),
        border = "brown",
        horizontal = TRUE,
        notch = FALSE
)
```



Fig(9) When examinging the no_neigh_vert histogram is produces no useable analysis both data sets are close to each other and the mean.The only use of this feature is if another data set complety differnet is thrown in,then used to spearate the two.In the boxplot there is an overlap of the data spread and the end of letters seems to be too close to the mean of non_letters meaning 1/2 of the data is mixed in.

Section 3.3 ———— Introduction————

The p-values given by the t-tests will determine whether there is a difference in the true population means between letters and non-letters of each feature. This will help us determine whether a feature will be useful to distinguish between a letter and a non-letter. Using this T statistic we can then workout the P value using the formula below .

$$t = \frac{m - \mu}{s/\sqrt{n}} \tag{2}$$

t = T-test m = mean μ = theorical value s = Standard deviation n = varaible set size

For these t-tests We assume the conditions for inference are met, meaning a random sample and varaibles are independent.

———— Hypothesis testing ————

Our null hypothesis will be that for each feature, there will be no difference in their population means.

Our alternative hypothesis will be that the mean of the features will differ from letters and non-letters. We follow first on the assumpation of the null hypotheis until proven different , significance level will be 0.05 for 95% confidence. where pval is the orgainal p value and pval1 is the p-value logged to base 10

```
kable(output_t,capation = "Statsical analysis using t-tests")
```

feature_name	pval	pval1
nr_pix	2.44293554456541e-32	-72.7895225676046
row_with_1	2.08078410942649e-14	-31.5034465035551
cols_with_1	0.933305433546108	-0.0690227645629155
rows_with_3p	2.92353913244709e-35	-79.5176833407689
cols_with_3p	1.84981110014758e-14	-31.6211077760681
aspect_ratio	1.19379594297862e-05	-11.3357873666332
no_neigh_left	1.52107847566848e-09	-20.3038462302149
no_neigh_right	8.86488680431937e-14	-30.0540931313023
no_neigh_above	2.1494243577241e-14	-31.4709912362194
no_neigh_below	0.00726155393223076	-4.92516143279661
no_neigh_vert	1.06943277839776e-07	-16.0509672566914
no_neigh_horiz	9.05989230182333e-11	-23.1245787901662
eyes	1.47593288994688e-06	-13.4262203003353

P values

The lowest P-Values are cols_with1 ,no_neigh_below,eyes. The low P value indicates a low chance that the sample means are different by chance alone, meaning that there is a high chance that the true population means are different. As the P values for these 3 features below our significance level of 0.05, we have enough evidence to suggests that we can reject the null hypothesis. Which suggest that the alternative hypothesis is accepted, that the population means are not the same. This will be true forall the features except no_neigh_right,no_neigh_horiz. but I will be considering the top 3 features. Since the population means are determined to be different it is implied that they can be useful in distinguishing between letters and non-letters. Upon readinfg more material I realsied that a pre requizste of t-tests is are that the data is not extremely skewed,I can relate to my histograms for section 3.1 fig(1-6) to examine skew.My eyes contain Nan which affects the skew so I will have to discount that feature,cols with one is skewed but nor past the 1 varaible for most of its data so is accepted and no_neigh_below is accpeted from calacuations done to calucate the skew on a qq-plot.

Selection

In conclusion, the 3 with the lowest valid P Values are, cols_with1, no_neigh_below and rows_with3p, 1 of which I have chosen in the previous section to be likely candidates for the most useful features.

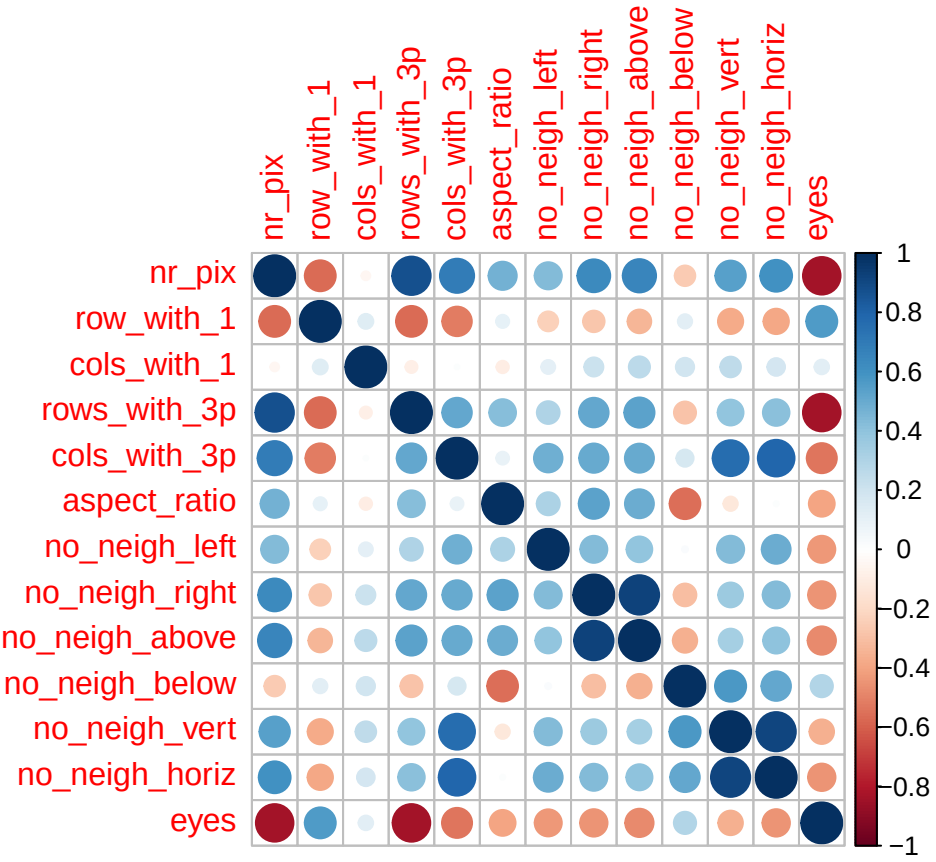
##Section 3.4

```
# Correlation plot with circles:
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.1.3
```

```
## corrplot 0.92 loaded
```

```
# correlation plot with circles
correlations <- cor(data[,3:15])
corrplot(correlations, method="circle")
```




```
corr = round(cor(data[,3:15]),1)
kable(corr,capation = "Correlation analysis ")
```

	nr_pix	row_with_1	cols_with_1	rows_with_3p	cols_with_3p	aspect_ratio	no_neigh_left	no_neigh_right	no_neigh_above	no_neigh_below	no_neigh_vert	no_neigh_horiz	eyes
nr_pix	1.0	-0.6	0.0	0.9	0.7	0.5	0.4	0.6	0.7	-0.3	0.5	0.6	-0.8
row_with_1	-0.6	1.0	0.1	-0.6	-0.5	0.1	-0.2	-0.3	-0.3	0.1	-0.4	-0.4	0.6
cols_with_1	0.0	0.1	1.0	-0.1	0.0	-0.1	0.1	0.2	0.3	0.2	0.3	0.2	0.1
rows_with_3p	0.9	-0.6	-0.1	1.0	0.5	0.4	0.3	0.5	0.5	-0.3	0.4	0.4	-0.8
cols_with_3p	0.7	-0.5	0.0	0.5	1.0	0.1	0.5	0.5	0.5	0.2	0.8	0.8	-0.5
aspect_ratio	0.5	0.1	-0.1	0.4	0.1	1.0	0.3	0.5	0.5	-0.6	-0.1	0.0	-0.4
no_neigh_left	0.4	-0.2	0.1	0.3	0.5	0.3	1.0	0.4	0.4	0.0	0.4	0.5	-0.4
no_neigh_right	0.6	-0.3	0.2	0.5	0.5	0.5	0.4	1.0	0.9	-0.3	0.4	0.4	-0.4
no_neigh_above	0.7	-0.3	0.3	0.5	0.5	0.5	0.4	0.9	1.0	-0.4	0.3	0.4	-0.5
no_neigh_below	0.1	0.2	-0.3	0.2	-0.6	0.0	-0.3	-0.4	1.0	0.6	0.5	0.3	

	nr_pix	row_width_1	rows_1	width_3p	rows_3p	aspect	no_neigh	left_neigh	right_neigh	above_neigh	below_neigh	vert_neigh	eyes	horiz
no_neigh_vert	-0.5	-0.4	0.3	0.4	0.8	-0.1	0.4	0.4	0.3	0.6	1.0	0.9	-0.4	
no_neigh_horiz	-0.6	-0.4	0.2	0.4	0.8	0.0	0.5	0.4	0.4	0.5	0.9	1.0	-0.4	
eyes	-0.8	0.6	0.1	-0.8	-0.5	-0.4	-0.4	-0.4	-0.5	0.3	-0.4	-0.4	1.0	

The table above along withe the correlation plot shows the correlation between 2 features in all possible combinations.We only need to examine either above or below the diagonal center line as the data is repated . Observing the table we can see that the 3 highest correlation values are

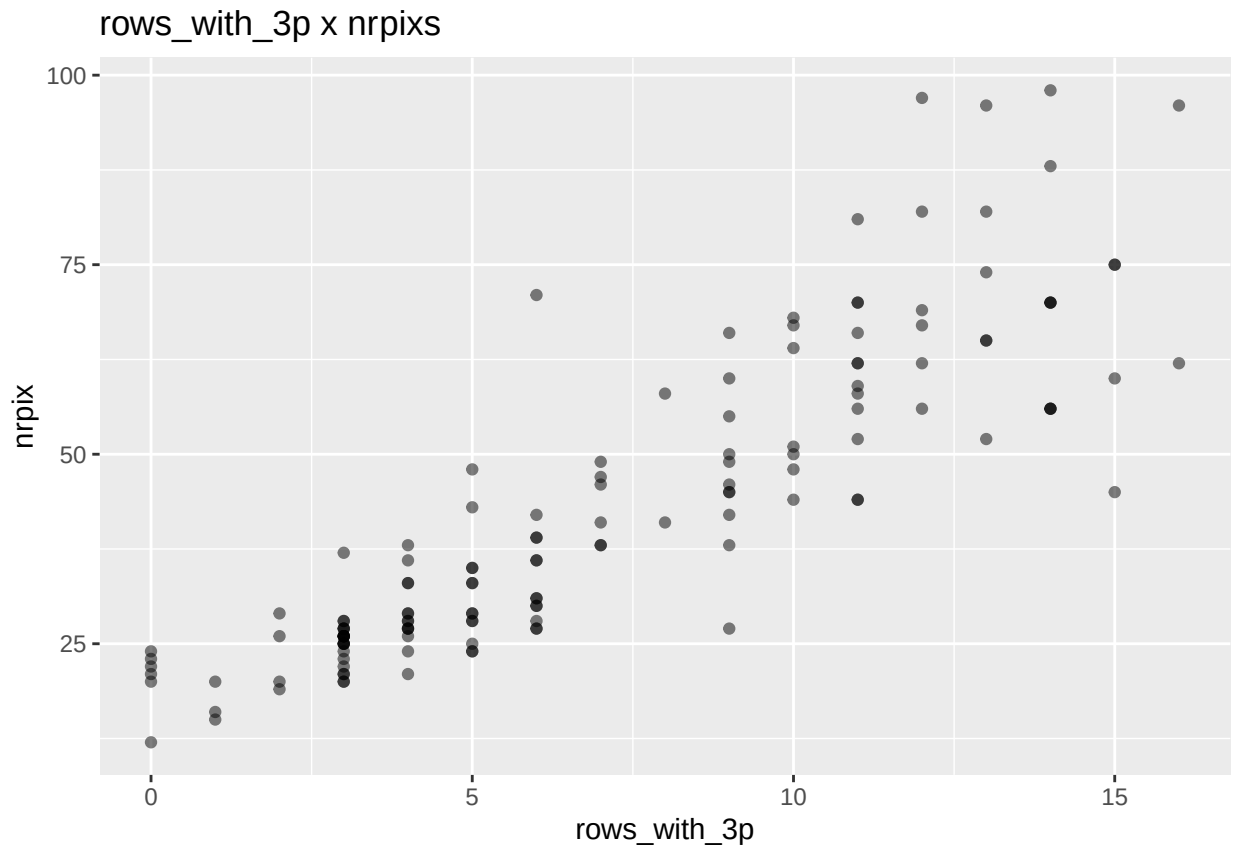
———— Results of top 3 ————

rows_with_3p x nrpix @0.9 no_neigh_horiz x no_neigh_vert@0.9 no_neigh_right x no_neigh_above @ 0.9 I will grahically explore this 3 features.

```
cor.test(data$nr_pix, data$rows_with_3p)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$nr_pix and data$rows_with_3p
## t = 21.112, df = 138, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8279551 0.9080860
## sample estimates:
##       cor
## 0.8738289
```

```
ggplot(mapping = aes(x = data$rows_with_3p, y = data$nr_pix)) +labs(title="rows_with_3p x nrpixs", x = "
  geom_point(alpha = 0.5)
```




```
geom_smooth(method = "lm", se = FALSE, color = "red")
```

```
## geom_smooth: na.rm = FALSE, orientation = NA, se = FALSE
## stat_smooth: na.rm = FALSE, orientation = NA, se = FALSE, method = lm
## position_identity
```

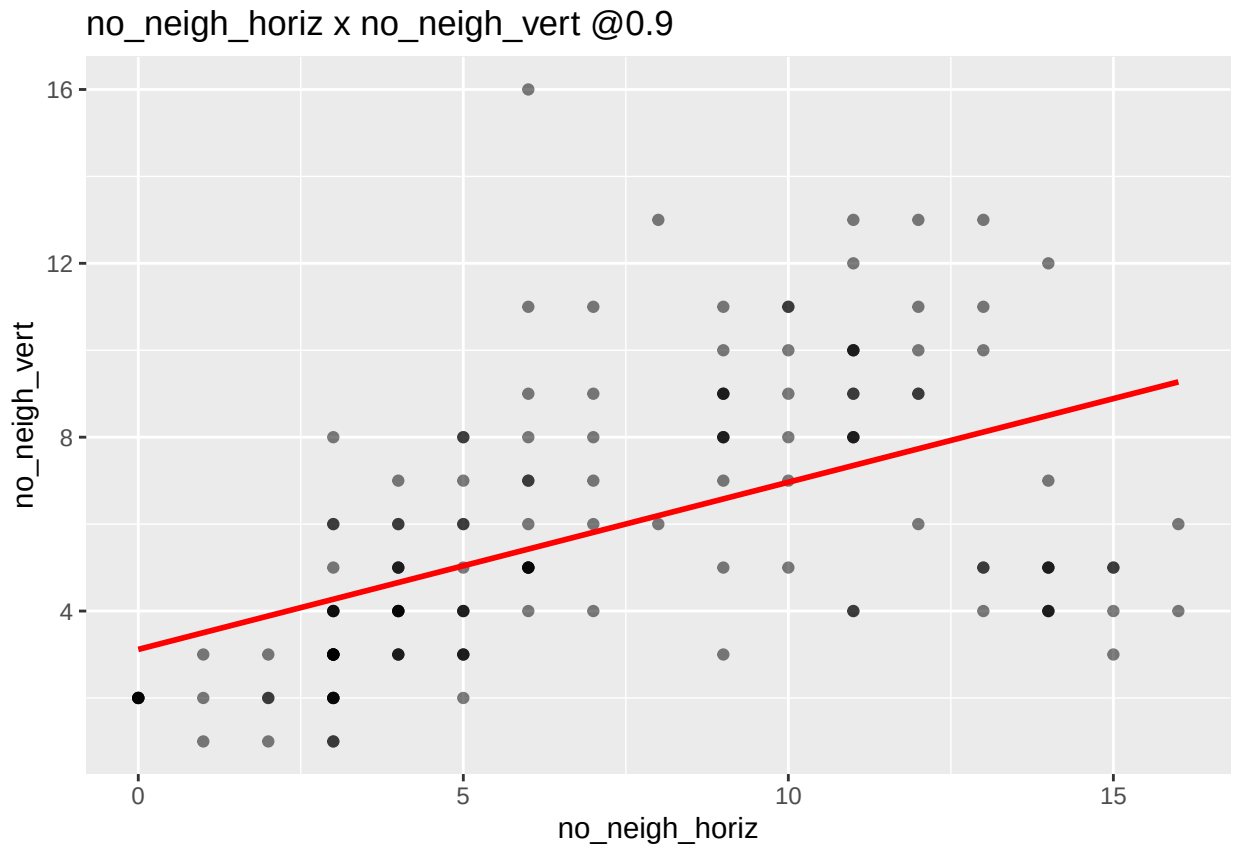
fig(10) Number of pixels has strong linear correlations with rows_with_3p. This is evident due to the definitions of those features. As there are more rows with 3 or more black pixels as does the number of black pixels in the image increase.Th linear correaltion is postive and well fitted.

```
cor.test(data$cols_with_3p, data$rows_with_3p)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$cols_with_3p and data$rows_with_3p
## t = 7.0879, df = 138, p-value = 6.407e-11
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3835851 0.6286365
## sample estimates:
##       cor
## 0.5166122
```

```
ggplot(mapping = aes(x = data$rows_with_3p, y = data$cols_with_3p)) +labs(title="no_neigh_horiz x no_nei
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE, color = "red")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



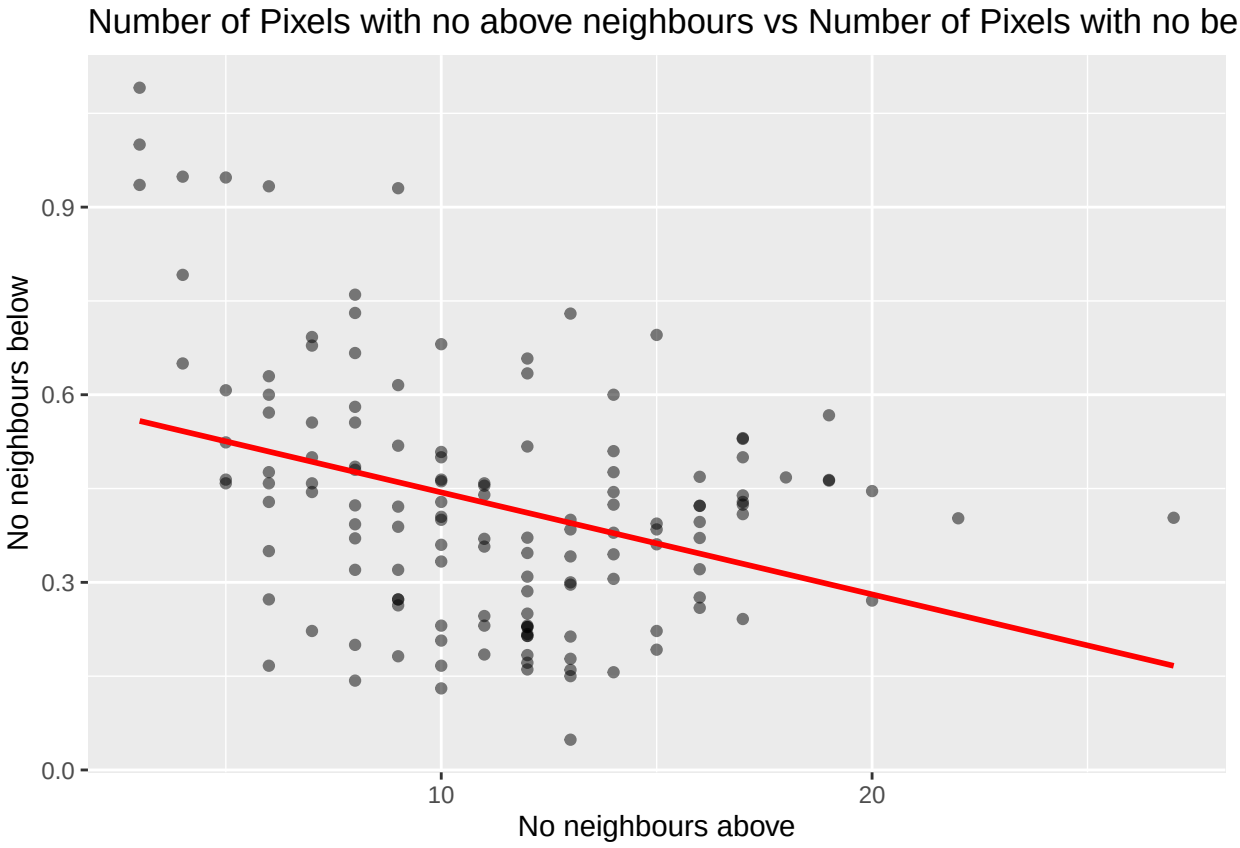
fig(11) Here there appears to be a weak postive correlation,the data points are not clustered and apear both above and below the line.

```
# rows_with_3p and nr_pix = 0.9
# no_neigh_above and no_neigh_below = 0.919
cor.test(data$no_neigh_below, data$no_neigh_above)
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$no_neigh_below and data$no_neigh_above
## t = -4.4225, df = 138, p-value = 1.962e-05
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.4896125 -0.1979985
## sample estimates:
##       cor
## -0.3523283
```

```
ggplot(mapping = aes(x = data$no_neigh_above, y = data$no_neigh_below)) +labs(title="Number of Pixels wi
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE, color = "red")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



fig(12) Our first negavtive correlation is prested there appears to be a few outliers inthe data set but it is strongly correlated.

##Section 4 In this section I will be exploring the impenation regression models and machine learning on the data set's features. ####Section 4.1 To predict the aspect_ratio feature on an image I will fit a multiple regression model I will be using the p-value approach in order to select the features I would like to include in my regression model.

This means that I will start with a model containing all the features and I will be removing the feature that has the highest p-value and refit a smaller model. I will repeat this until all features included in the model are significant.

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following objects are masked from 'package:raster':
##
##     area, select
```

```
model = lm(aspect_ratio~nr_pix + row_with_1 + cols_with_1 + cols_with_3p + rows_with_3p + no_neigh_above

multi_model = stepAIC(model,direction="backward")
```

```
## Start:  AIC=102.42
## aspect_ratio ~ nr_pix + row_with_1 + cols_with_1 + cols_with_3p +
##     rows_with_3p + no_neigh_above + no_neigh_below + no_neigh_left +
##     no_neigh_right + no_neigh_horiz + no_neigh_vert
##
##                   Df Sum of Sq    RSS    AIC
## - no_neigh_above   1     1.031 246.16 101.00
## <none>                         245.12 102.42
## - cols_with_3p     1     3.988 249.11 102.68
## - rows_with_3p     1     4.160 249.28 102.77
## - cols_with_1      1     4.483 249.61 102.95
## - no_neigh_below   1     9.919 255.04 105.97
## - nr_pix           1    10.091 255.21 106.06
## - no_neigh_horiz   1    10.516 255.64 106.30
## - no_neigh_right   1    12.152 257.28 107.19
## - no_neigh_vert    1    12.843 257.97 107.57
## - no_neigh_left    1    18.482 263.61 110.59
## - row_with_1       1   112.008 357.13 153.10
##
## Step:  AIC=101
## aspect_ratio ~ nr_pix + row_with_1 + cols_with_1 + cols_with_3p +
##     rows_with_3p + no_neigh_below + no_neigh_left + no_neigh_right +
##     no_neigh_horiz + no_neigh_vert
##
##                   Df Sum of Sq    RSS    AIC
## <none>                         246.16 101.00
## - cols_with_3p     1     4.317 250.47 101.44
## - rows_with_3p     1     4.332 250.49 101.45
## - cols_with_1      1     6.777 252.93 102.81
## - no_neigh_below   1     9.322 255.48 104.21
## - nr_pix           1     9.586 255.74 104.35
## - no_neigh_horiz   1    10.110 256.27 104.64
## - no_neigh_vert    1    12.394 258.55 105.88
## - no_neigh_left    1    19.431 265.59 109.64
## - no_neigh_right   1    19.723 265.88 109.80
## - row_with_1       1   116.604 362.76 153.29
```

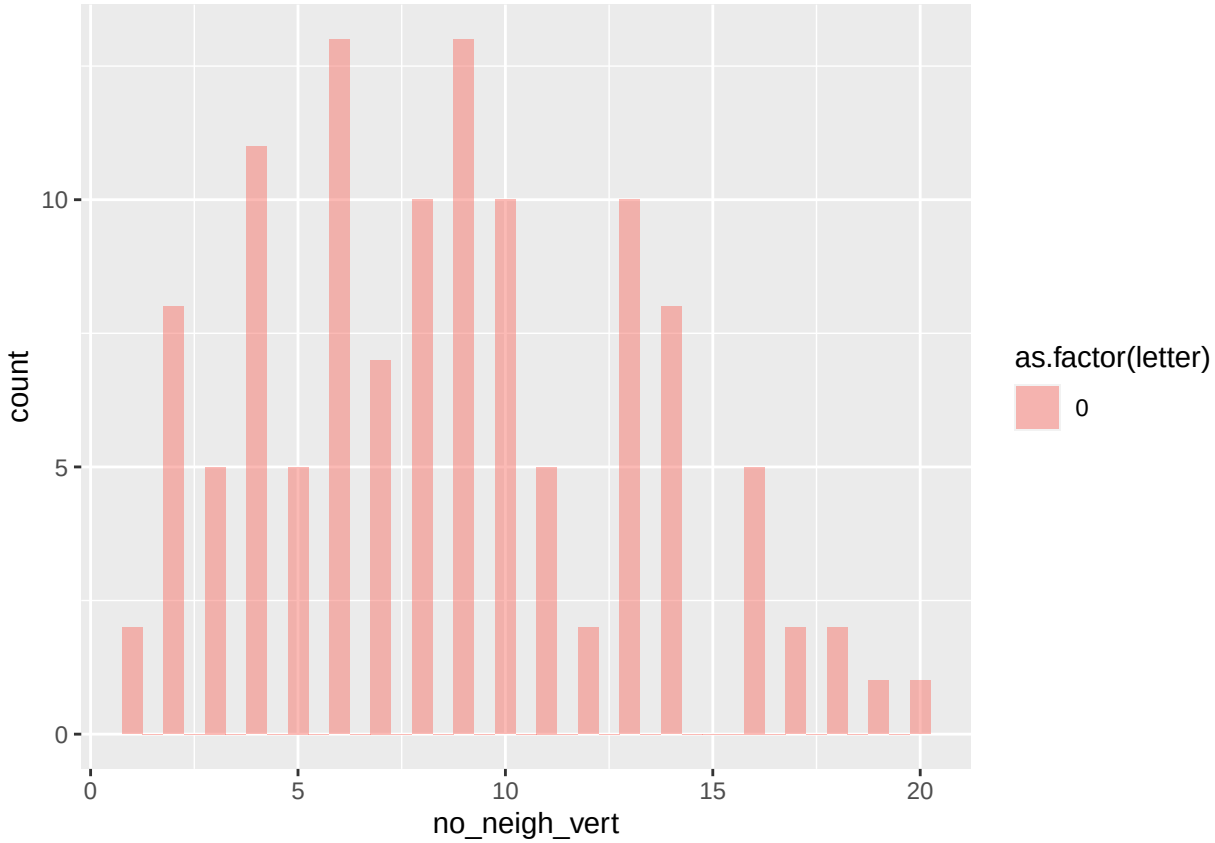
```
summary(multi_model)
```

```
##
## Call:
## lm(formula = aspect_ratio ~ nr_pix + row_with_1 + cols_with_1 +
##     cols_with_3p + rows_with_3p + no_neigh_below + no_neigh_left +
##     no_neigh_right + no_neigh_horiz + no_neigh_vert, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.05885 -0.83053 -0.05417  0.87467  2.91984
```

```
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     8.45387    0.91799   9.209 7.73e-16 ***
## nr_pix          0.04651    0.02075   2.241  0.02672 *
## row_with_1      0.43930    0.05620   7.817 1.67e-12 ***
## cols_with_1    -0.12862    0.06825  -1.885  0.06174 .
## cols_with_3p   -0.11612    0.07720  -1.504  0.13498
## rows_with_3p    0.09274    0.06155   1.507  0.13432
## no_neigh_below -4.14525    1.87540  -2.210  0.02885 *
## no_neigh_left   0.12417    0.03891   3.191  0.00178 **
## no_neigh_right  0.13879    0.04317   3.215  0.00165 **
## no_neigh_horiz  0.19347    0.08405   2.302  0.02295 *
## no_neigh_vert  -0.22044    0.08649  -2.549  0.01199 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.381 on 129 degrees of freedom
## Multiple R-squared:  0.7182, Adjusted R-squared:  0.6963
## F-statistic: 32.87 on 10 and 129 DF,  p-value: < 2.2e-16
```

DISCUSS ####Section 4.2

```
data2 = data

data2$letter = 0
for( i in (1:80)){
  data2[i,15] = 1
}

# randomly shuffle rows:
data_shuffled = data2[sample(nrow(data)),]

# first 80% will be training data
training_data = data_shuffled[1:120,]
test_data = data_shuffled[121:140,]

# plot a histogram: for letter
plt <- ggplot(training_data, aes(x=no_neigh_vert, fill=as.factor(letter))) +
  geom_histogram(binwidth=0.5, alpha=.5, position='identity')
plt
```




```
# logistical regression model

glmfit<-glm(letter ~ no_neigh_vert,
            data = training_data,
            family = 'binomial')
```

```
## Warning: glm.fit: algorithm did not converge
```

```
summary(glmfit)
```

```
##
## Call:
## glm(formula = letter ~ no_neigh_vert, family = "binomial", data = training_data)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -2.409e-06  -2.409e-06  -2.409e-06  -2.409e-06  -2.409e-06
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -2.657e+01  7.118e+04       0        1
## no_neigh_vert -8.222e-17  7.334e+03       0        1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 0.0000e+00  on 119  degrees of freedom
## Residual deviance: 6.9619e-10  on 118  degrees of freedom
## AIC: 4
##
## Number of Fisher Scoring iterations: 25
```

```
glmfit$coefficients
```

```
##   (Intercept) no_neigh_vert
## -2.656607e+01 -8.221751e-17
```

```
beta0 = as.numeric(glmfit$coefficients[1])
beta1 = as.numeric(glmfit$coefficients[2])



linear_combo = beta0 + beta1*1

estimated_probability = exp(linear_combo)/(1+exp(linear_combo))


newdata = as.data.frame(c(2,1,5))
colnames(newdata) = 'no_neigh_vert'

predicted = predict(glmfit, newdata, type="response")

abs(estimated_probability-predicted)
```

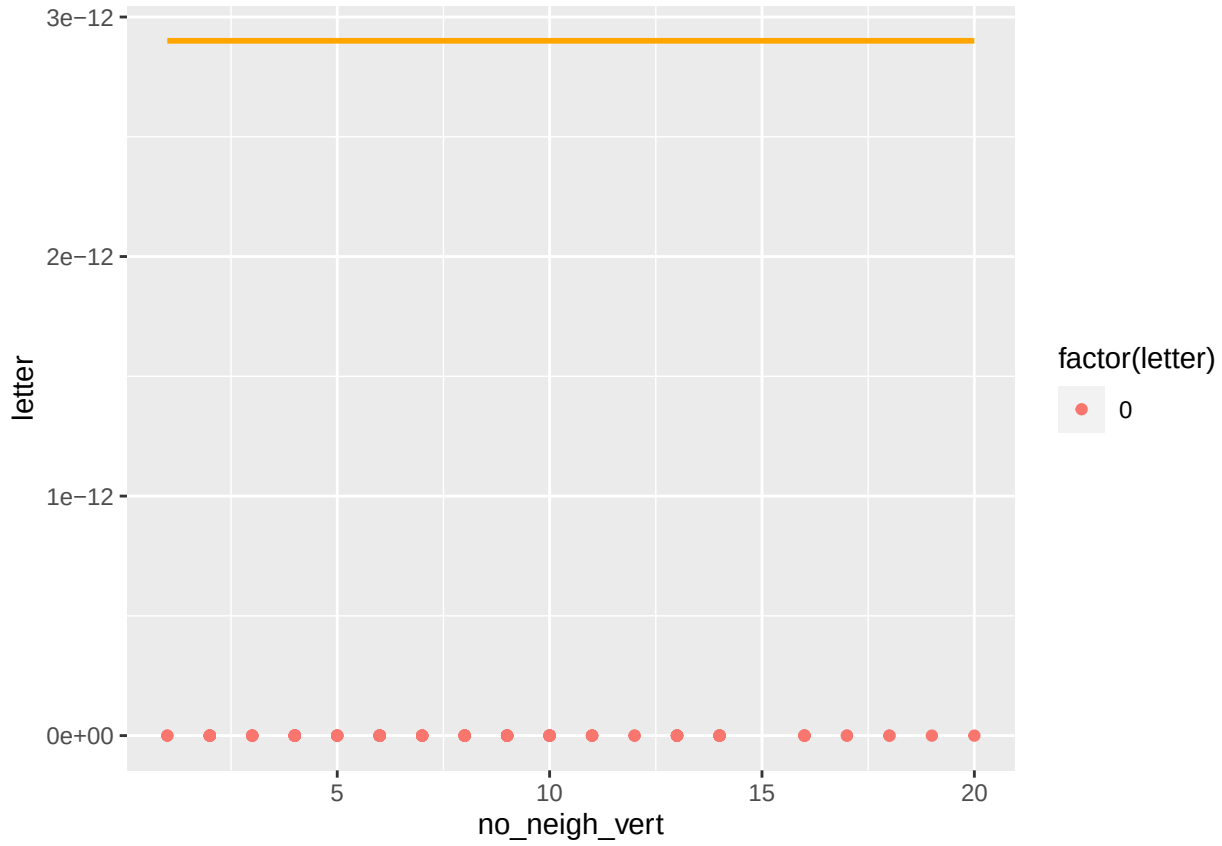
```
## 1 2 3
## 0 0 0
```

```
x.range = range(training_data[["no_neigh_vert"]])

x.values = seq(x.range[1],x.range[2],length.out=1000)

# plots the graph
fitted.curve <- data.frame(no_neigh_vert = x.values)
summary(fitted.curve)
```

```
##  no_neigh_vert
##  Min.   : 1.00
##  1st Qu.: 5.75
##  Median :10.50
##  Mean   :10.50
##  3rd Qu.:15.25
##  Max.   :20.00
```

```
fitted.curve[["letter"]] = predict(glmfit, fitted.curve, type="response")

##
# Plot the training data and the fitted curve:
```

```
plt <-ggplot(training_data, aes(x=no_neigh_vert, y=letter)) +
  geom_point(aes(colour = factor(letter)),
             show.legend = T)+
  geom_line(data=fitted.curve, colour="orange", size=1)

plt
```




```
#  Calculating the accuracy on the training data, assuming a p>0.5 cut-off
###########################################################################

training_data[["predicted_val"]] = predict(glmfit, training_data, type="response")
training_data[["predicted_class"]] = 0
training_data[["predicted_class"]][training_data[["predicted_val"]] > 0.5] = 1

correct_items = training_data[["predicted_class"]] == training_data[["letter"]]
# proportion correct:
print(paste("Percentage of Values correct for training data: ",nrow(training_data[correct_items,])/nrow
```

```
## [1] "Percentage of Values correct for training data:  1"
```

```
# proportion incorrect:
print(paste("Percentage of Values incorrect for training data: ",nrow(training_data[!correct_items,])/n
```

```
## [1] "Percentage of Values incorrect for training data:  0"
```

```
#  Calculating the accuracy on the test data, assuming a p>0.5 cut-off
###############################################################################

test_data[["predicted_val"]] = predict(glmfit, test_data, type="response")
test_data[["predicted_class"]] = 0
test_data[["predicted_class"]][test_data[["predicted_val"]] > 0.5] = 1

correct_items = test_data[["predicted_class"]] == test_data[["letter"]]

# proportion correct:
print(paste("Percentage of Values correct for testing data: ", nrow(test_data[correct_items,])/nrow(tes
```

```
## [1] "Percentage of Values correct for testing data:  1"
```

```
# proportion incorrect:
print(paste("Percentage of Values incorrect for testing data: ",nrow(test_data[!correct_items,])/nrow(t
```

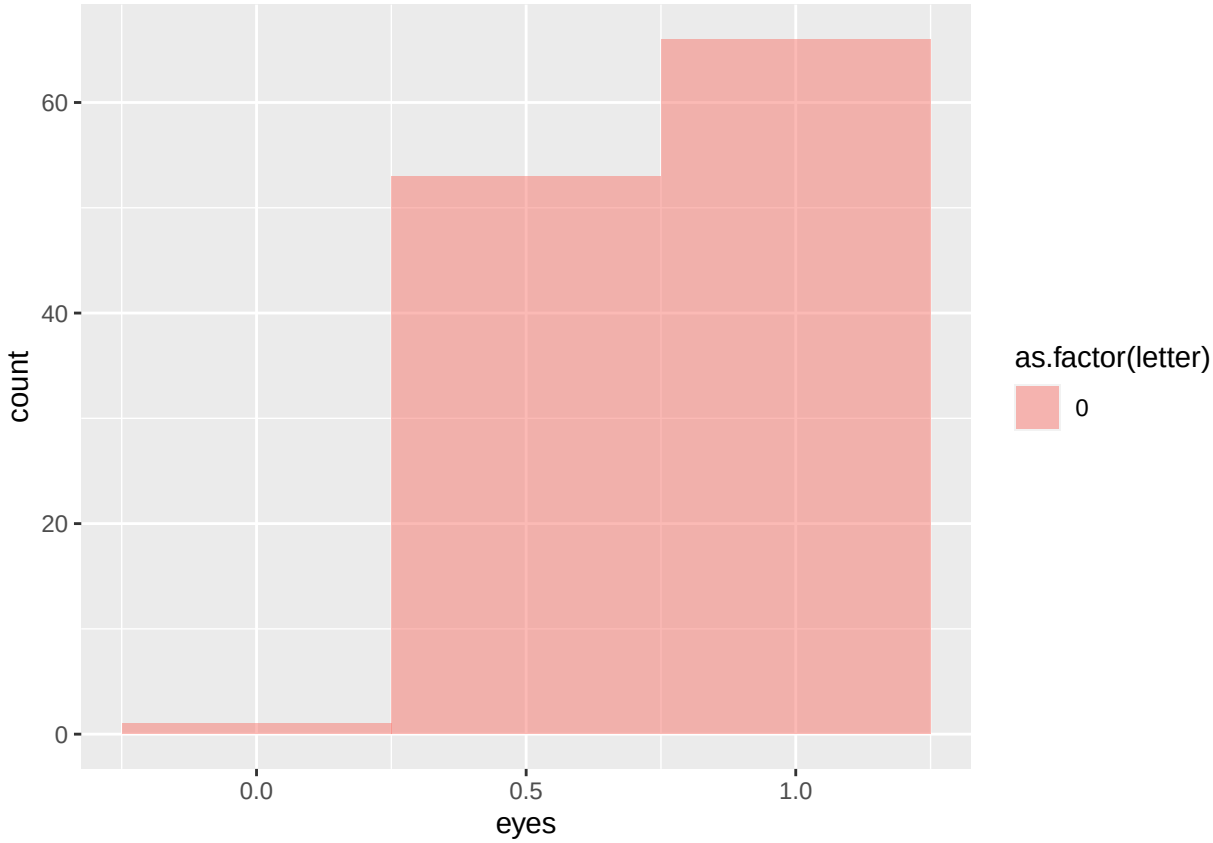
```
## [1] "Percentage of Values incorrect for testing data:  0"
```

As shown in the summary above, the most significant features to predict the feature of 'aspect_ratio' include: 'nr_pix', 'rows_with_1', 'cols_with_1', 'rows_with_3p', 'cols_with_3p', 'no_neigh_above', 'no_neigh_below', 'no_neigh_left', 'no_neigh_right', 'no_neigh_horiz' and 'no_neigh_vert'.

As the R squared value is 0.9081, we can say the model has improved from removing the other features. It may be beneficial in this model to also remove the 'no_neigh_above' and "eyes,

##4.2 ## Section 4.2 I have decided to look at logistical regression for the feature of rows_with3p. as it has a low pvalue,Running through other features it produces the highest rate of 0.84

```
plt1
```



Section 4.3

The table below shows the total number of letters, faces and exclamation marks with values for nr_pixels (split1), aspect_ratio(split2) and rows_with_3p(split3) which are greater than the median value for that feature for that class for their class.

```
kable(output, capation = "Categorical Median Splits")
```

	Split1	Split2	Split3
Letters	37	33	30
Faces	20	16	19
Exclamation Marks	10	7	6

Above is a table for each catergoty,Which has a feature greater than the median.Its interesting to note the particular low numbers as the set of letter conatins 80,faces 40 and exclmation mark 20. Spilt 1 predicts under and on half for the catergories where as it never goes above the half line which is good.

Section 4.4

Naive Bayes Theorem is used to order and categorise the images into letters and non-letters user the table produced above. This is done by using probabilities of certain features in a formula. each feature is independent which is where the name "Naive" Bayes orginates.

```
# Naive Bayes

num_images = 140
# 80 letters
num_letters = 80
# 40 faces
num_faces = 40
# 20 xclaims
num_xclaims = 20


# table from 4.3 is : output

# SPLIT1 = 1, SPLIT2 = 1, SPLIT3 = 1

# p(split1=1 | class = letter) = 36/80
split1_letter = output[1,1]/num_letters
# p(split1=1 | class = face) = 17/40
split1_faces = output[2,1]/num_faces
# p(split1=1 | class = xclaim) = 9/20
split1_xclaims = output[3,1]/num_xclaims

# p(split2=1 | class = letter) = 40/80
split2_letter = output[1,2]/num_letters
# p(split2=1 | class = face) = 16/40
split2_faces = output[2,2]/num_faces
# p(split2=1 | class = xclaim) = 10/20
split2_xclaims = output[3,2]/num_xclaims

# p(split3=1 | class = letter) = 17/80
split3_letter = output[1,3]/num_letters
# p(split3=1 | class = face) = 17/40
split3_faces = output[2,3]/num_faces
# p(split3=1 | class = xclaim) = 0/20
split3_xclaims = output[3,3]/num_xclaims

# if letter
# p(split1,split2,split3 | class = letter)*p(class=letter) = 0.027321
prob1_letter = split1_letter * split2_letter * split3_letter * (num_letters/num_images)

# p(split1,split2,split3 | class = face)*p(class=face) = 0.02064285714
prob1_face = split1_faces * split2_faces * split3_faces * (num_faces/num_images)

# p(split1,split2,split3 | class = xclaim)*p(class=xclaim) = 0
prob1_xclaims = split1_xclaims * split2_xclaims * split3_xclaims * (num_xclaims/num_images)

# hence is split1=1, split2 =1 and split3=1 the predicted class will be a letter
table_predict1 = NULL
table_predict1 = rbind(table_predict1,prob1_letter,prob1_face,prob1_xclaims)

as.matrix(table_predict1[order(-table_predict1[,1]),])
```

```
##                     [,1]
```

```
## prob1_letter  0.04088170
## prob1_face    0.02714286
## prob1_xclaims 0.00750000
```

```
title = names(table_predict1[1,1])

if(grepl("letter", title, fixed=TRUE)){
  print("The predicted class for when split1=1, split2=1 and split3=1 is: Letter")
} else if(grepl("faces",title,  fixed=TRUE)){
  print("The predicted class for when split1=1, split2=1 and split3=1 is: Face")
} else if(grepl("xclaim",title,  fixed=TRUE)){
  print("The predicted class for when split1=1, split2=1 and split3=1 is: Exclamation Mark")
}
```

```
## [1] "The predicted class for when split1=1, split2=1 and split3=1 is: Letter"
```

```
# SPLIT1 = 0, SPLIT2 = 0, SPLIT3 = 0

# p(split1=1 | class = letter) = 44/80
split1_letter = (1-(output[1,1]/num_letters))
# p(split1=1 | class = face) = 23/40
split1_faces = (1-(output[2,1]/num_faces))
# p(split1=1 | class = xclaim) = 11/20
split1_xclaims = (1-(output[3,1]/num_xclaims))

# p(split2=1 | class = letter) = 40/80
split2_letter = (1-(output[1,2]/num_letters))
# p(split2=1 | class = face) = 24/40
split2_faces = (1-(output[2,2]/num_faces))
# p(split2=1 | class = xclaim) = 10/20
split2_xclaims = (1-(output[3,2]/num_xclaims))

# p(split3=1 | class = letter) = 63/80
split3_letter = (1-(output[1,3]/num_letters))
# p(split3=1 | class = face) = 23/40
split3_faces = (1-(output[2,3]/num_faces))
# p(split3=1 | class = xclaim) = 20/20
split3_xclaims = (1-(output[3,3]/num_xclaims))

# if letter
# p(split1,split2,split3 | class = letter)*p(class=letter) = 0.027321
prob1_letter = split1_letter * split2_letter * split3_letter * (num_letters/num_images)

# p(split1,split2,split3 | class = face)*p(class=face) = 0.02064285714
prob1_face = split1_faces * split2_faces * split3_faces * (num_faces/num_images)

# p(split1,split2,split3 | class = xclaim)*p(class=xclaim) = 0
prob1_xclaims = split1_xclaims * split2_xclaims * split3_xclaims * (num_xclaims/num_images)

# hence is split1=0, split2 =0 and split3=0 the predicted class will be a letter
table_predict1 = NULL
table_predict1 = rbind(table_predict1,prob1_letter,prob1_face,prob1_xclaims)

as.matrix(table_predict1[order(-table_predict1[,1]),])
```

```
##                    [,1]
## prob1_letter  0.112779
## prob1_face    0.045000
## prob1_xclaims 0.032500
```

```
title = names(table_predict1[1,1])


if(grepl("letter", title, fixed=TRUE)){
  print("The predicted class for when split1=0, split2=0 and split3=0 is: Letter")
} else if(grepl("faces",title,  fixed=TRUE)){
  print("The predicted class for when split1=0, split2=0 and split3=0 is: Face")
} else if(grepl("xclaim",title,  fixed=TRUE)){
  print("The predicted class for when split1=0, split2=0 and split3=0 is: Exclamation Mark")
}
```

```
## [1] "The predicted class for when split1=0, split2=0 and split3=0 is: Letter"
```

I have chosen to include my code as it is more like a proof and explains each step.

The Naive Bayes Classifier is how likly an image is to be classfied.

The results for the split features = 0,0,0 was not surprising at all. It seemed to predict the image category which has the highest number of the corresponding values. For example the category with the most amount of 0's across all the images which was exclamation mark. This model will not be too accurate as it assumes the features are completely independent but as see above in 3.2 in summaey statiscs that they have a lot in common.

Conclusion

Overall,I enjoyed this project as I have a love for Maths.At every point in my journey I made mistakes or leanrt how to code something more effiently with practice.

During this project I have learnt a range of statistical techniques and regression models.I have complemeted my first R project and what application to statistics AI & ML have.

At the end of this project I was able to determine if features were useful in determining the class of the image and even fit models to predict the class of the image using linear regression, logistical regression and Naive Bayes Classifier.

Websites used for coding help and influence: https://www.statmethods.net/graphs/bar.html https://biostats.w.uib.no/creating-a-circular-bar-chart/ https://www.programmingr.com/tutorial/log-in-r/#:~:text=Log%20transformation%20in%20R%20is,logarithm%20to%20a%200%20value. https://r-graph-gallery.com/web-circular-barplot-with-R-and-ggplot2.html https://stackoverflow.com/questions/27661852/adding-a-density-line-to-a-histogram-with-count-data-in-ggplot2 https://www.google.com/search?q=latex+cheat+sheet+math&sxsrf=APq-WBu-iesg5ccD7bPFWApa72Xfl_cAuQ:1647730935367&source=lnms&tbm=isch&sa=X&sqi=2&ved=2ahUKEwjc4dmIpNP2AhVMA94KHcBVAc8Q_AUoAXoECAEQAw&biw=1536&bih=722&dpr=1.25#imgrc=ArlgSzCEDPpfjM https://bookdown.org/ndphillips/YaRrr/dataframe-column-names.html https://stackoverflow.com/questions/50584209/number-of-rows-of-result-is-not-a-multiple-of-vector-length-arg-2-in-r https://www.datanovia.com/en/blog/how-to-create-a-bubble-chart-in-r-using-ggplot2/ https://www.r-graph-gallery.com/correlogram.html https://cloud.google.com/blog/products/data-analytics/different-types-graphs-charts-uses https://wordcounter.net/ https://tex.stackexchange.com/questions/456051/standard-deviation https://www.nlm.nih.gov/nichsr/stats_tutorial/section2/mod8_sd.html https://r-lang.com/how-to-calculate-standard-deviation-in-r-using-sd/#:~:text=To%20calculate%20the%20standard%20deviation,values%20provided%20in%20the%20object. https://towardsdatascience.com/q-q-plots-explained-5aa8495426c0

and lecture notes from class