- **Project Flowchart:** (Phase 1 + Phase 2 + Phase 3 + Phase 4)

1. Project Understanding:

- **Understanding Face Expression Recognition via Measurable Features:**

Facial Expression Recognition (FER) is a fundamental task in computer vision and affective computing, aiming to classify human emotions based on facial features. While deep learning has recently dominated this field, earlier approaches relied on extracting specific measurable features, such as distances and proportions between key facial landmarks, to train machine learning models. This project follows the latter approach, using a structured dataset derived from facial measurements.

- **Problem Statement:**

The goal of this project is to analyse a dataset consisting of 210 instances from the Cohn-Kanade database, where each instance is represented by 25 facial measurements. These measurements capture key structural aspects of facial expressions, including eyebrow position, eye dimensions, mouth shape, and relationships between these components. The dataset is labelled with one of seven possible facial expressions: **Neutral, Disgust, Sadness, Fear, Surprise, Anger, and Joy**.

- **What I Want to Accomplish:**

1. **Feature Selection:** Identify which facial measurements are the most critical for FER.
2. **Evaluation Strategy:** Define the best approach for training, testing, and validation.
3. **Classification:** Compare multiple machine learning classifiers to determine the most effective for FER.
4. **Clustering:** Analyse the dataset without labels and evaluate how well clustering methods can group expressions naturally.

2. Data Understanding:

- **Exploring the Facial Expression Dataset:**

The dataset consists of 210 instances from the Cohn-Kanade database, each with 25 facial measurement features and a class label representing one of seven facial expressions.

The features represent different facial components:

- Eyebrows (H1–H8, L1, L2)
- Eyes (H9–H12, W1, W2)
- Mouth (H13–H15, W3, L3)
- Relationships between facial components (R1–R4)

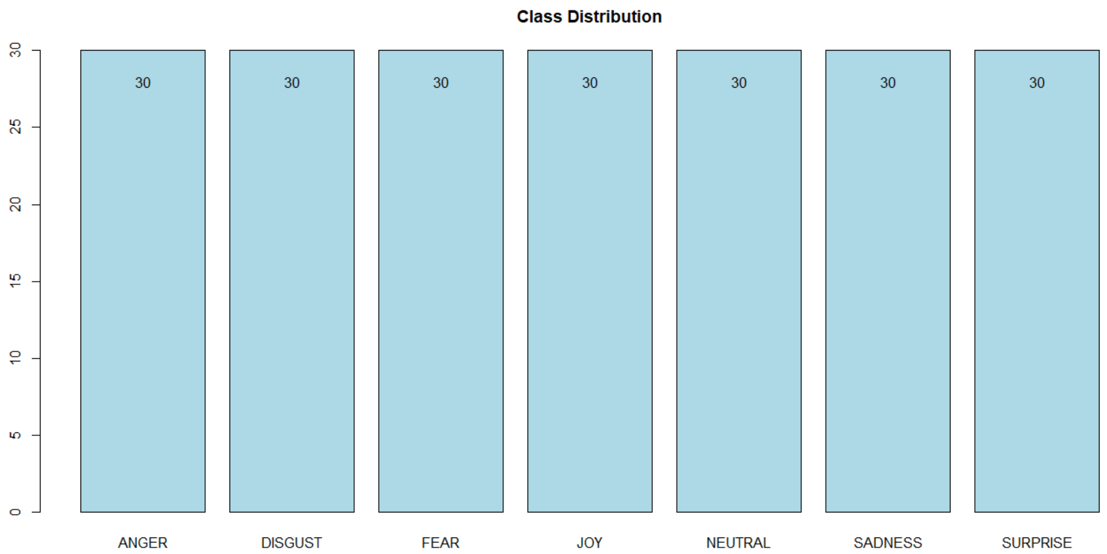
- **<u>What I Did:</u>**

- Imported the dataset.

Facial_data

	H1	H2	H3	H4	L1	H5	H6	H7	H8	L2	H9	H10	W1	H11	H12	W2	H13	H14	H15	W3	L3	R1	R2	R3	R4	...26	...27	...28
1	22	17	19	23	38	20	18	16	23	25	15	19	17	14	17	10	17	16	23	9	24	33	23	14	73	1	1	
2	21	15	18	23	39	24	17	14	23	27	14	19	17	13	15	10	17	15	20	9	21	38	15	25	65	1	0	
3	21	16	18	23	38	17	17	14	15	31	15	17	17	13	16	10	17	15	22	9	30	36	22	16	74	0	0	
4	21	15	18	23	38	18	17	14	23	26	16	18	17	12	15	10	19	16	22	9	23	35	25	14	76	0	1	
5	21	14	16	23	38	17	17	14	23	24	17	18	17	11	15	10	20	14	27	9	27	31	26	11	77	0	1	
6	21	17	19	23	40	21	19	17	19	38	13	19	17	15	13	10	17	13	17	9	18	36	21	17	71	1	0	
7	21	19	19	23	37	23	19	18	17	28	16	19	17	14	16	11	20	17	16	9	15	53	11	48	61	0	0	
8	21	15	17	21	34	18	18	13	14	31	22	17	17	14	10	12	16	8	25	22	25	39	4	97	69	1	1	
9	14	9	11	14	27	10	8	6	8	28	27	10	17	18	8	14	16	7	29	22	34	32	13	24	85	0	1	
10	20	14	17	21	33	18	17	14	14	32	24	16	17	14	10	13	17	5	14	19	17	61	9	67	72	0	0	
11	18	12	16	18	33	15	15	11	11	24	22	14	14	11	7	10	14	6	24	18	23	42	2	210	66	0	0	
12	21	16	18	21	35	19	18	15	15	24	22	17	17	14	11	13	16	6	23	21	24	43	2	215	66	0	1	
13	21	16	20	23	33	22	20	15	14	30	23	18	17	16	9	14	16	7	22	17	22	35	4	87	63	1	0	
14	20	14	17	20	34	18	17	14	14	31	22	16	17	13	8	13	17	7	23	20	23	39	1	390	65	1	0	
15	21	13	17	23	37	24	17	17	23	36	18	19	17	13	12	10	18	14	28	13	28	36	16	22	70	1	1	
16	22	17	19	23	34	23	20	6	16	35	20	18	17	13	7	11	21	10	32	11	35	29	24	12	77	0	1	
17	21	15	13	21	41	10	13	13	14	21	8	15	14	18	15	10	30	3	28	12	23	35	23	15	80	0	0	
18	21	13	17	23	37	23	17	15	17	34	19	18	17	15	13	10	17	10	32	14	32	30	21	14	77	0	1	
19	18	17	19	23	39	23	19	19	22	39	13	20	17	13	12	10	19	8	23	10	25	36	25	14	75	1	0	
20	21	16	18	23	38	23	18	18	22	37	17	19	17	13	9	12	18	8	26	12	25	40	13	30	66	1	0	
21	21	13	17	23	37	22	17	17	23	35	20	19	17	12	15	12	20	9	16	13	17	54	13	41	67	0	0	
22	22	20	20	23	33	23	20	20	21	36	20	21	17	12	9	16	20	8	15	11	15	50	3	166	53	1	1	
23	16	15	17	20	23	19	19	18	17	27	27	17	15	16	6	12	17	2	16	12	22	37	8	46	63	0	1	
24	23	18	17	23	39	24	21	21	23	37	13	21	17	10	9	10	22	9	17	11	30	54	11	49	61	0	0	
25	22	21	21	23	32	23	22	22	22	30	28	22	17	13	8	10	21	8	12	10	12	49	14	35	62	1	0	

- Checked **class distribution** using a bar plot to visualize how expressions are distributed.

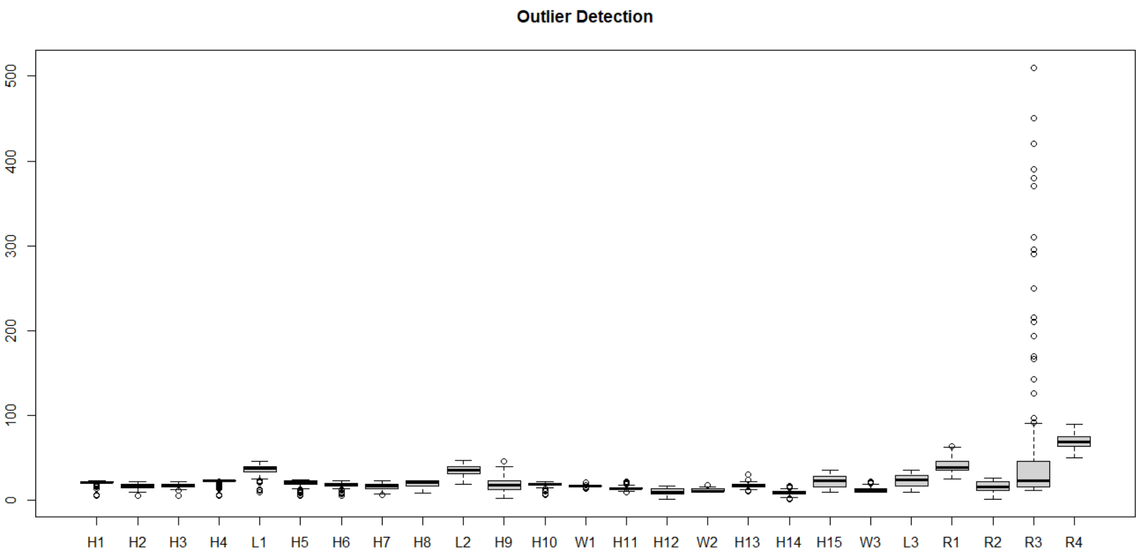


- Extracted the **features data**.

features_data

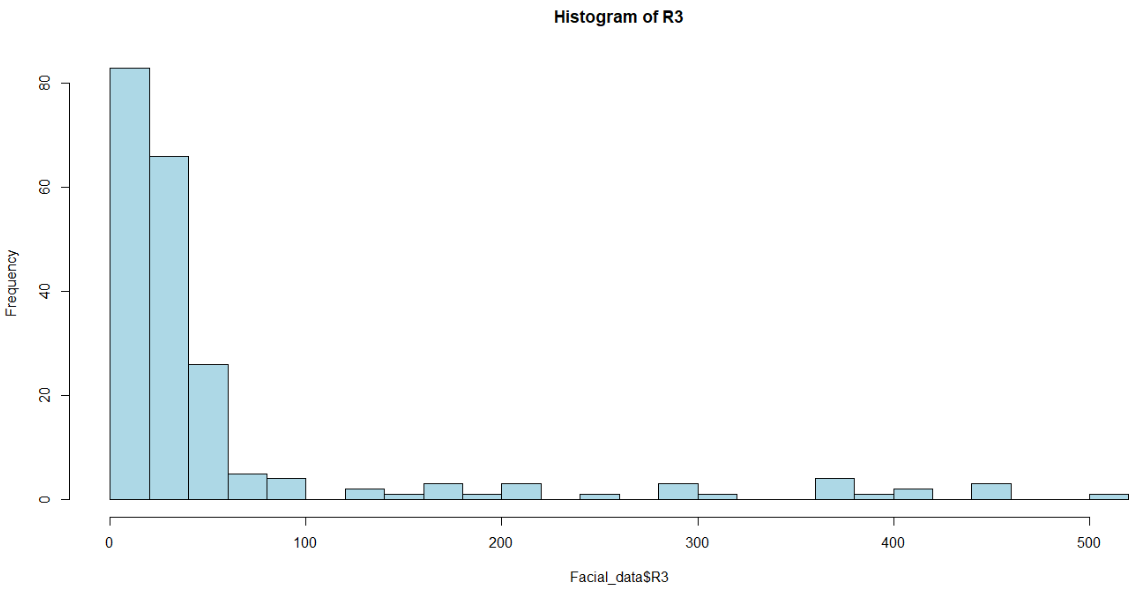
	H1	H2	H3	H4	L1	H5	H6	H7	H8	L2	H9	H10	W1	H11	H12	W2	H13	H14	H15	W3	L3	R1	R2	R3	R4
1	22	17	19	23	38	20	18	16	23	25	15	19	17	14	17	10	17	16	23	9	24	33	23	14	73
2	21	15	18	23	39	24	17	14	23	27	14	19	17	13	15	10	17	15	20	9	21	38	15	25	65
3	21	16	18	23	38	17	17	14	15	31	15	17	17	13	16	10	17	15	22	9	30	36	22	16	74
4	21	15	18	23	38	18	17	14	23	26	16	18	17	12	15	10	19	16	22	9	23	35	25	14	76
5	21	14	16	23	38	17	17	14	23	24	17	18	17	11	15	10	20	14	27	9	27	31	26	11	77
6	21	17	19	23	40	21	19	17	19	38	13	19	17	15	13	10	17	13	17	9	18	36	21	17	71
7	21	19	19	23	37	23	19	18	17	28	16	19	17	14	16	11	20	17	16	9	15	53	11	48	61
8	21	15	17	21	34	18	18	13	14	31	22	17	17	14	10	12	16	8	25	22	25	39	4	97	69
9	14	9	11	14	27	10	8	6	8	28	27	10	17	18	8	14	16	7	29	22	34	32	13	24	85
10	20	14	17	21	33	18	17	14	14	32	24	16	17	14	10	13	17	5	14	19	17	61	9	67	72
11	18	12	16	18	33	15	15	11	11	24	22	14	14	11	7	10	14	6	24	18	23	42	2	210	66
12	21	16	18	21	35	19	18	15	15	24	22	17	17	14	11	13	16	6	23	21	24	43	2	215	66
13	21	16	20	23	33	22	20	15	14	30	23	18	17	16	9	14	16	7	22	17	22	35	4	87	63
14	20	14	17	20	34	18	17	14	14	31	22	16	17	13	8	13	17	7	23	20	23	39	1	390	65
15	21	13	17	23	37	24	17	17	23	36	18	19	17	13	12	10	18	14	28	13	28	36	16	22	70
16	22	17	19	23	34	23	20	6	16	35	20	18	17	13	7	11	21	10	32	11	35	29	24	12	77
17	21	15	13	21	41	10	13	13	14	21	8	15	14	18	15	10	30	3	28	12	23	35	23	15	80
18	21	13	17	23	37	23	17	15	17	34	19	18	17	15	13	10	17	10	32	14	32	30	21	14	77
19	18	17	19	23	39	23	19	19	22	39	13	20	17	13	12	10	19	8	23	10	25	36	25	14	75
20	21	16	18	23	38	23	18	18	22	37	17	19	17	13	9	12	18	8	26	12	25	40	13	30	66
21	21	13	17	23	37	22	17	17	23	35	20	19	17	12	15	12	20	9	16	13	17	54	13	41	67
22	22	20	20	23	33	23	20	20	21	36	20	21	17	12	9	16	20	8	15	11	15	50	3	166	53
23	16	15	17	20	23	19	19	18	17	27	27	17	15	16	6	12	17	2	16	12	22	37	8	46	63
24	23	18	17	23	39	24	21	21	23	37	13	21	17	10	9	10	22	9	17	11	30	54	11	49	61
25	22	21	21	23	32	23	22	22	22	30	28	22	17	13	8	10	21	8	12	10	12	49	14	35	62

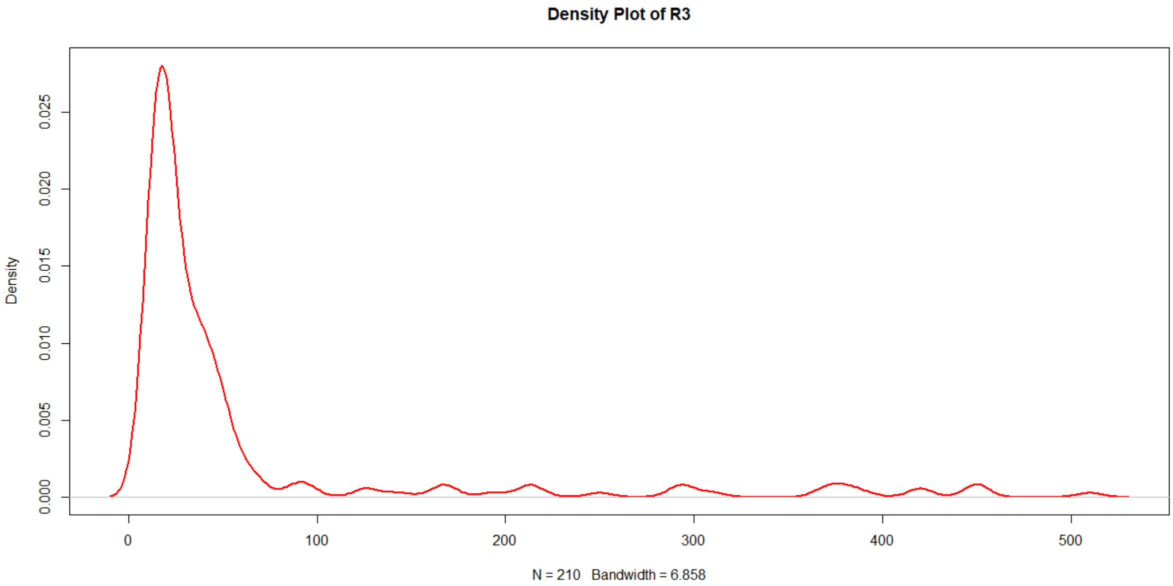
- Checked for **missing values** and found out that there were none.
- Used **boxplots** to identify potential outliers.

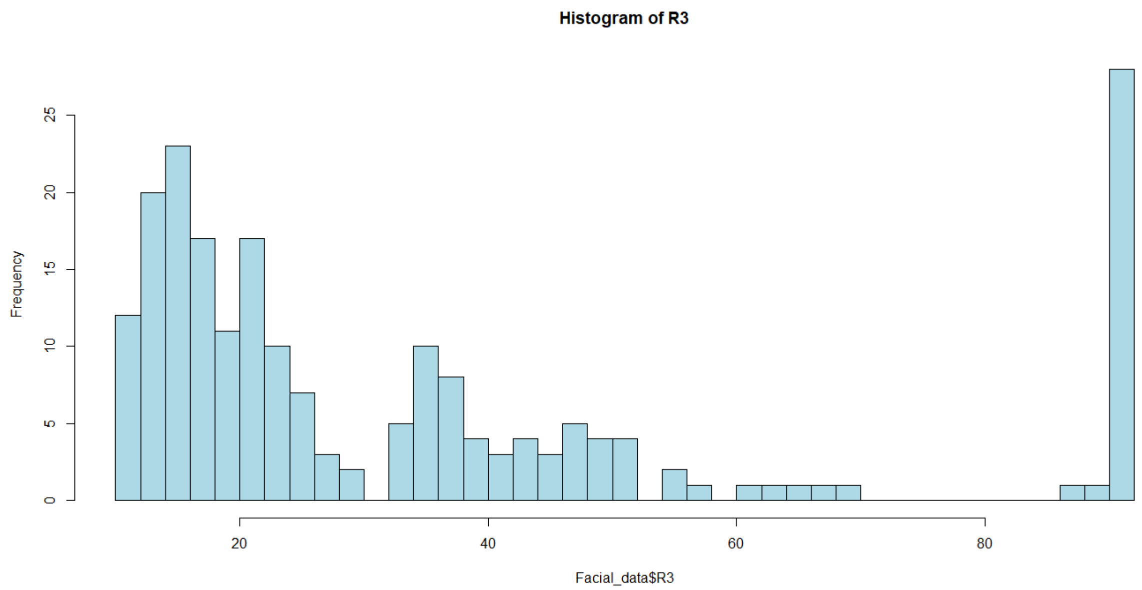


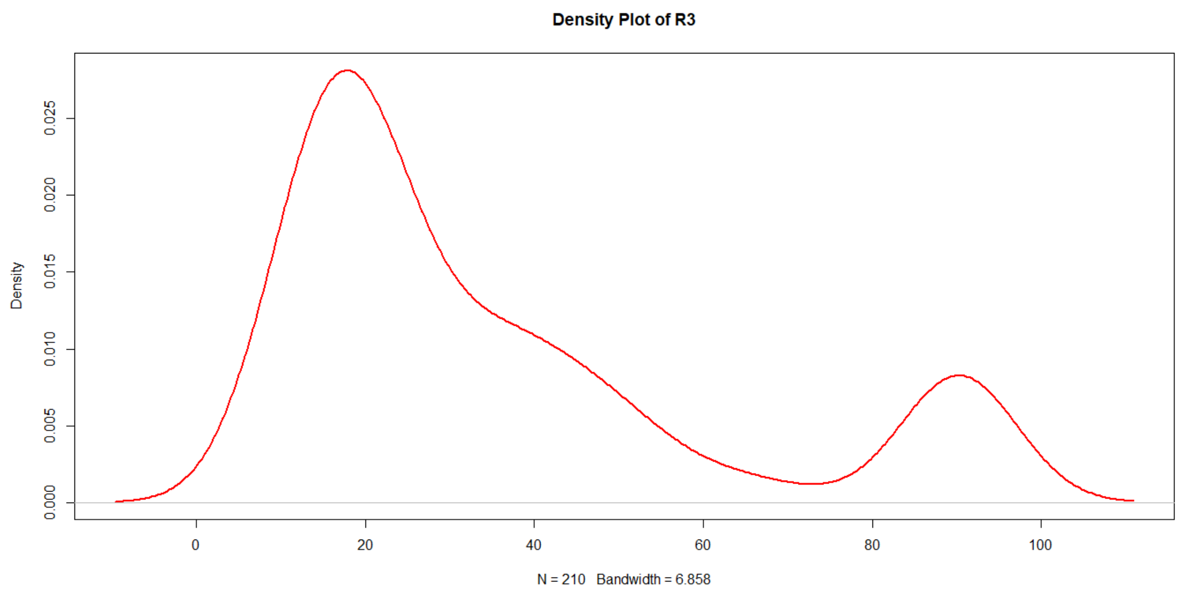
- Focused on feature **R3**, detecting extreme values.
- Used **IQR-based capping** to replace extreme values beyond Q3 + 1.5 * IQR with the upper limit.

- Plotted **histograms and density plots** before and after capping to visualize changes.

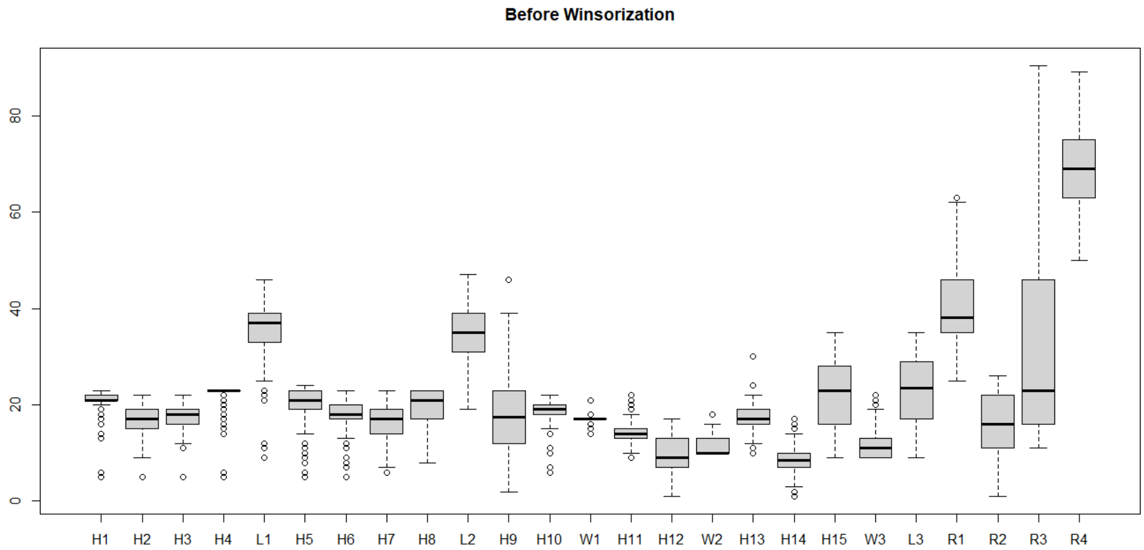




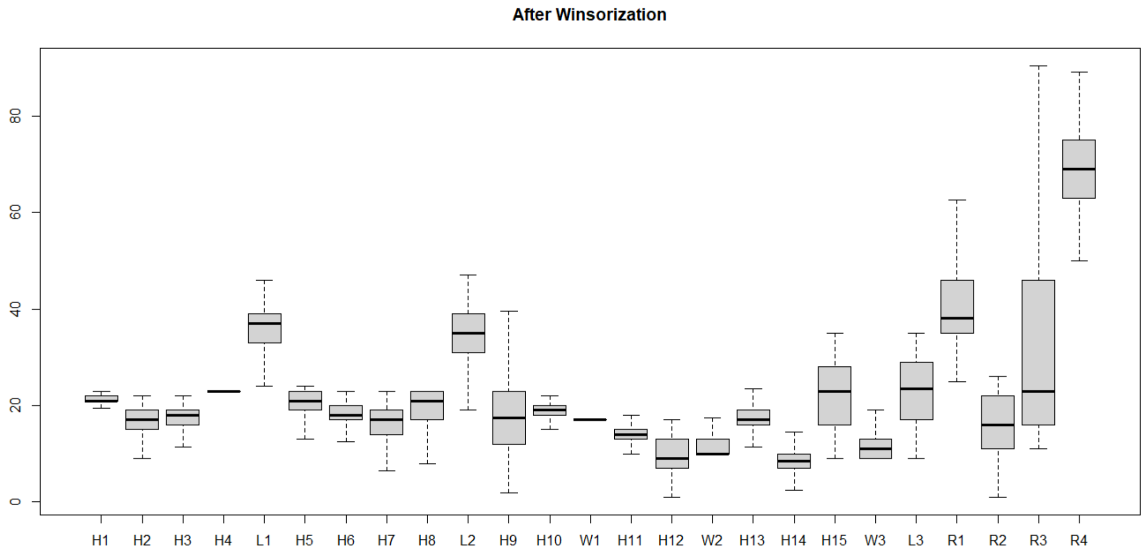

Histogram of R3
Frequency
0
5
10
15
20
25
20
40
60
80
Facial_data$R3


Density Plot of R3
Density
0.000
0.005
0.010
0.015
0.020
0.025
0
20
40
60
80
100
N = 210 Bandwidth = 6.858

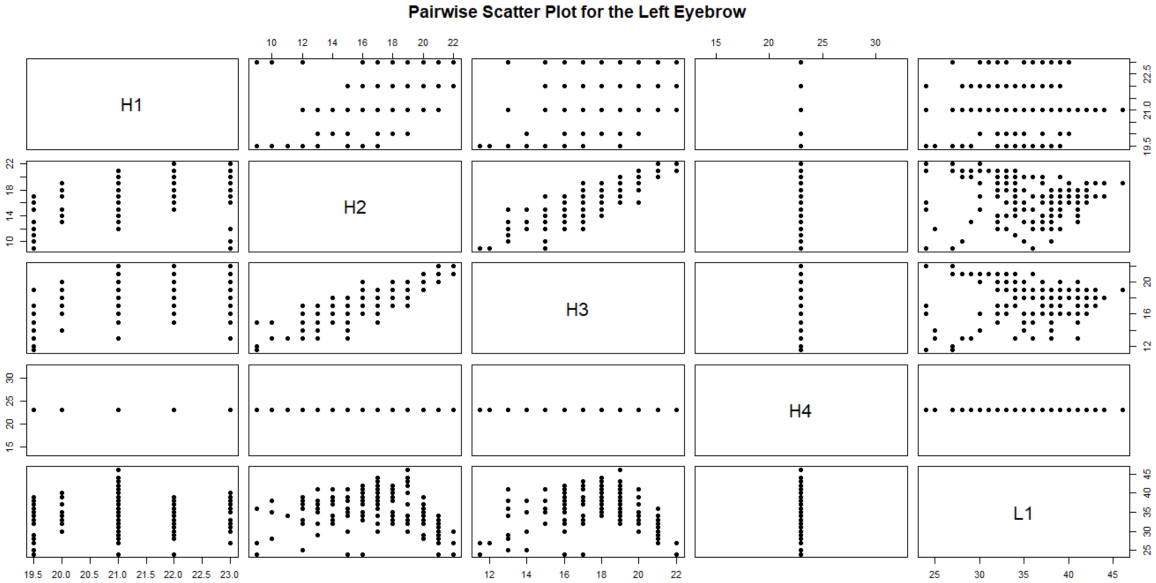
- Used **boxplots to identify further potential outliers.**

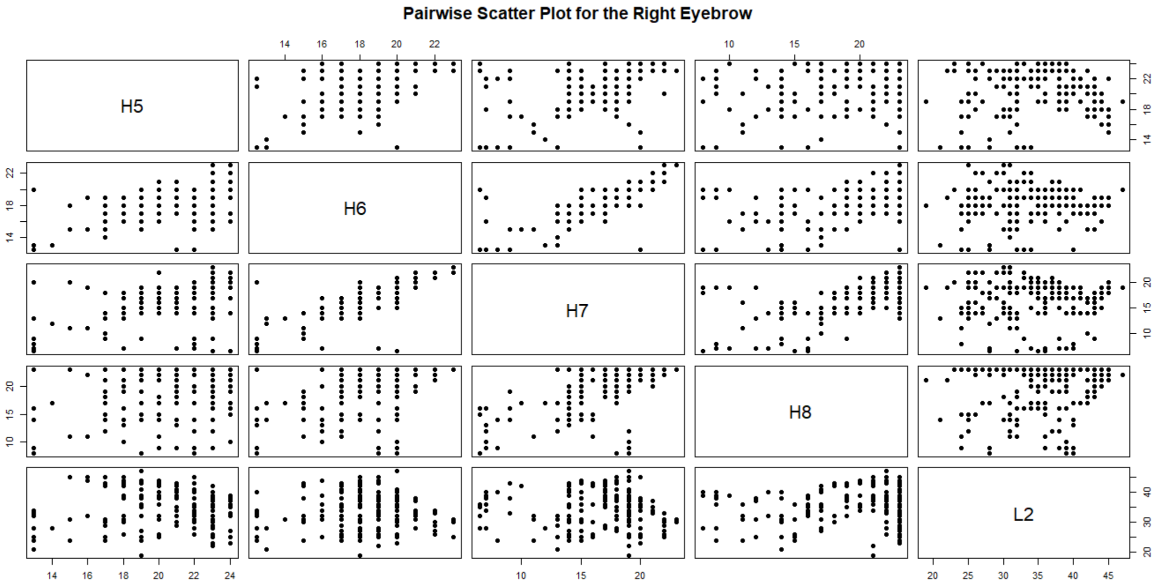


- **Applied Winsorization to cap outliers**.
- Created **boxplots to visualize after Winsorization**.

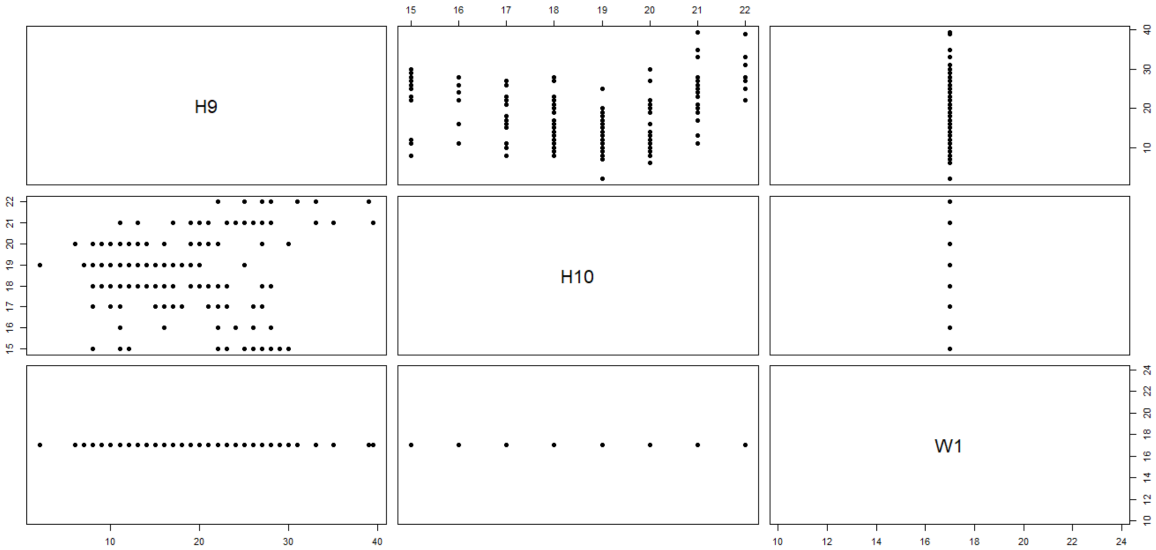


- Created **pairwise scatter plots** for feature groups (eyebrows, eyes, mouth, relationships) to observe correlations.

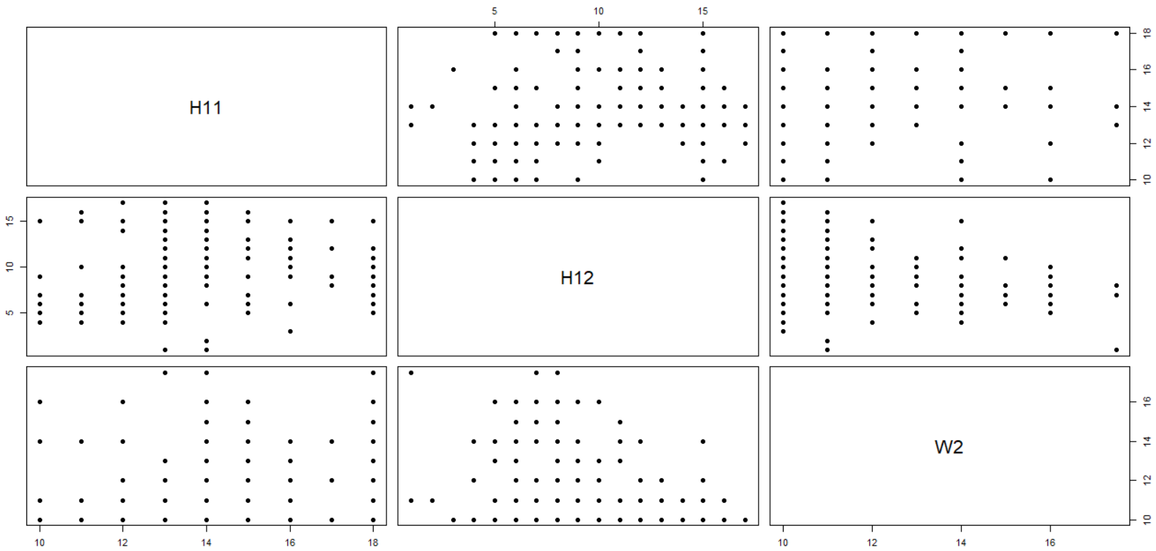




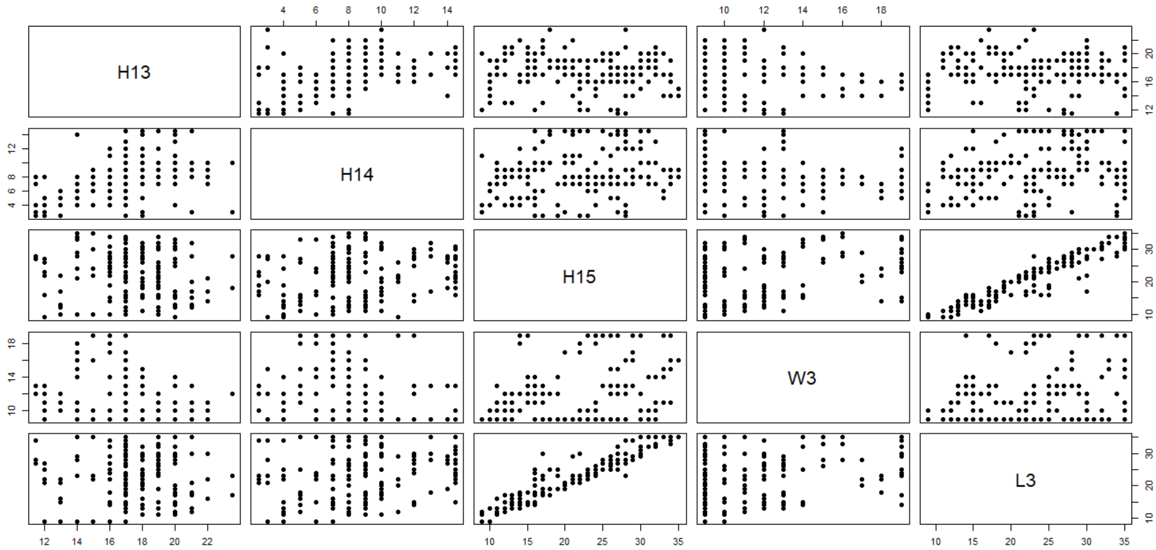
Pairwise Scatter Plot for the Left Eye



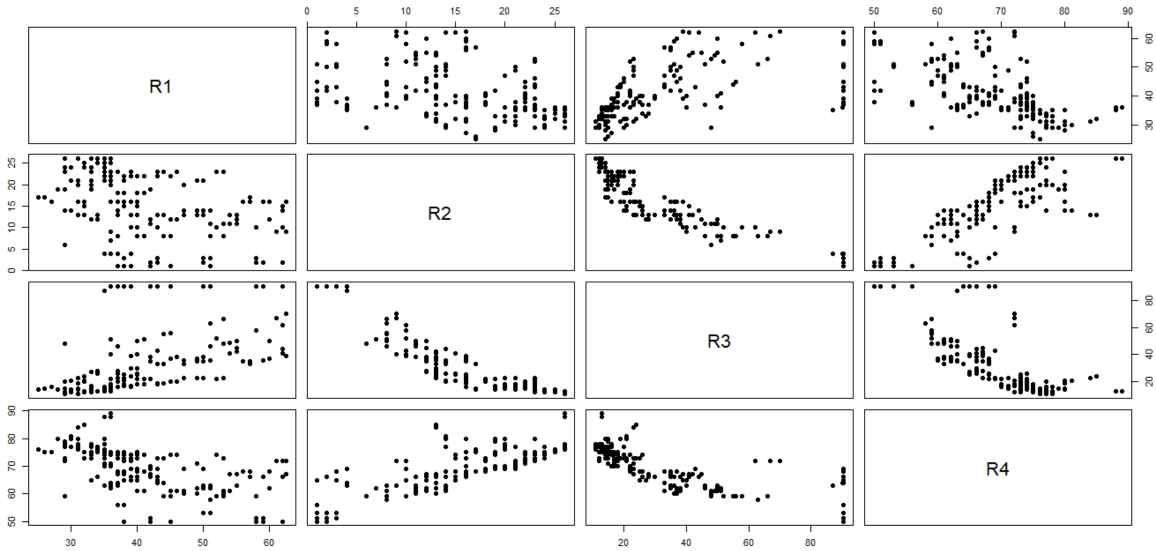
Pairwise Scatter Plot for the Right Eye



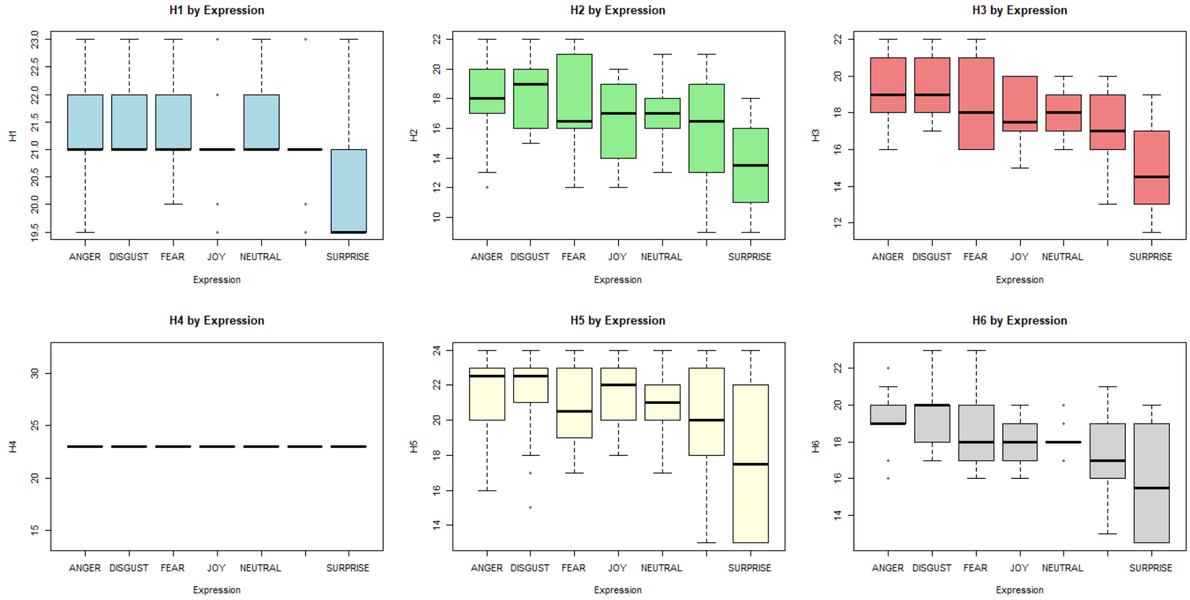
Pairwise Scatter Plot for the Mouth

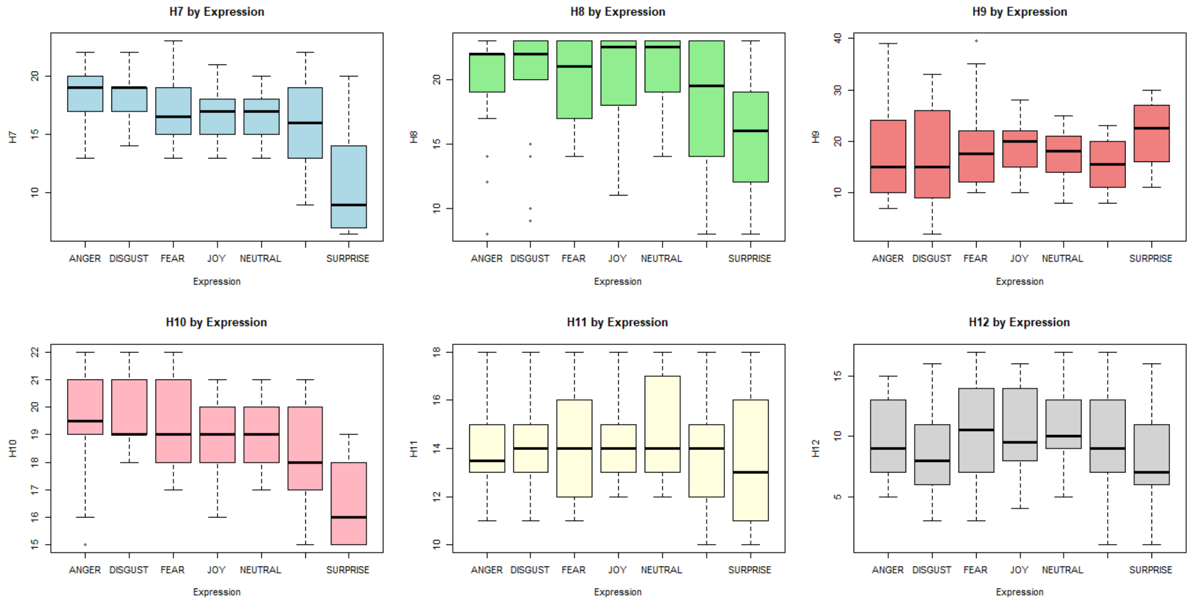


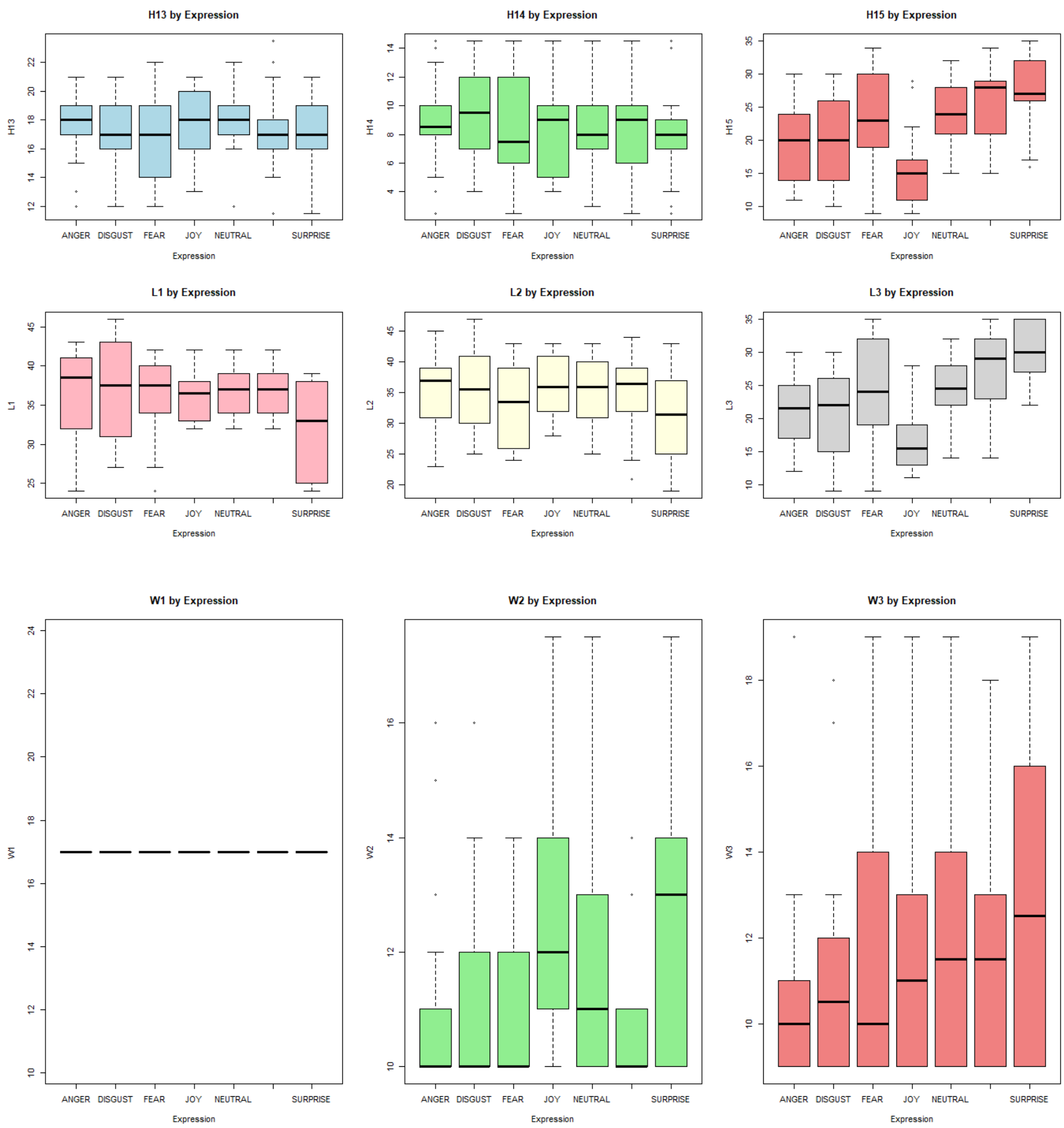
Pairwise Scatter Plot for the Relationships



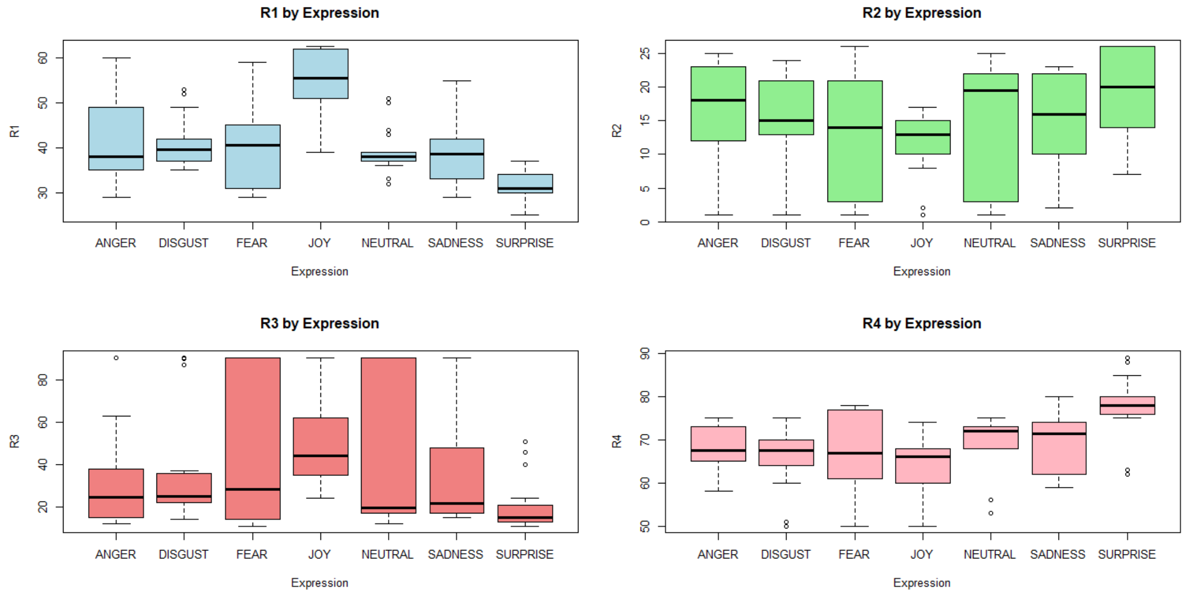
- Plotted **boxplots** of each feature grouped by **Expression** to analyse how facial features vary across emotions.







H13 by Expression
H13
Expression
H14 by Expression
H14
Expression
H15 by Expression
H15
Expression
L1 by Expression
L1
Expression
L2 by Expression
L2
Expression
L3 by Expression
L3
Expression
W1 by Expression
W1
Expression
W2 by Expression
W2
Expression
W3 by Expression
W3
Expression
ANGER DISGUST FEAR JOY NEUTRAL SURPRISE



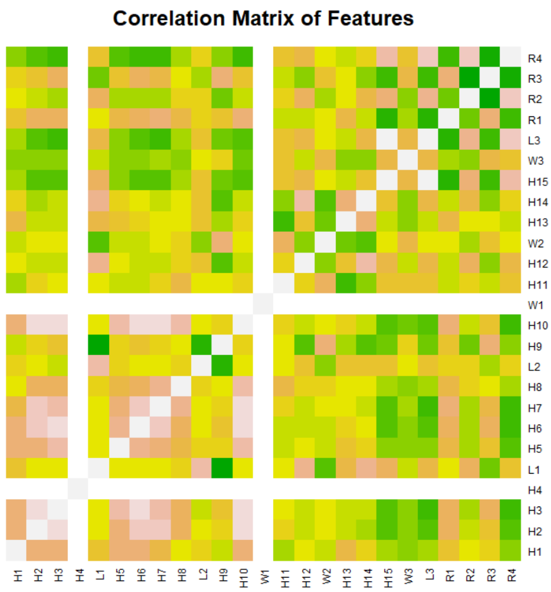
3. Data Preparation:

- **Dataset Preprocessing Overview:**

After understanding the dataset, the next step is to refine and prepare the data for machine learning. This involves feature selection, scaling, and dimensionality reduction to improve model performance.

- **What I Did:**

- Computed the **correlation matrix** to identify highly correlated features.



- Selected features with **correlation > 0.9** to reduce redundancy.

```
> # Features to remove based on correlation analysis
> features_to_remove <- colnames(features_data[, 1:25])[unique(high_corr_pairs[, 2])]
> print("Features to remove based on correlation analysis:")
[1] "Features to remove based on correlation analysis:"
> print(features_to_remove)
[1] "H9" "L3" "R3"
```

- Performed **ANOVA** to determine which features significantly vary across different expressions.

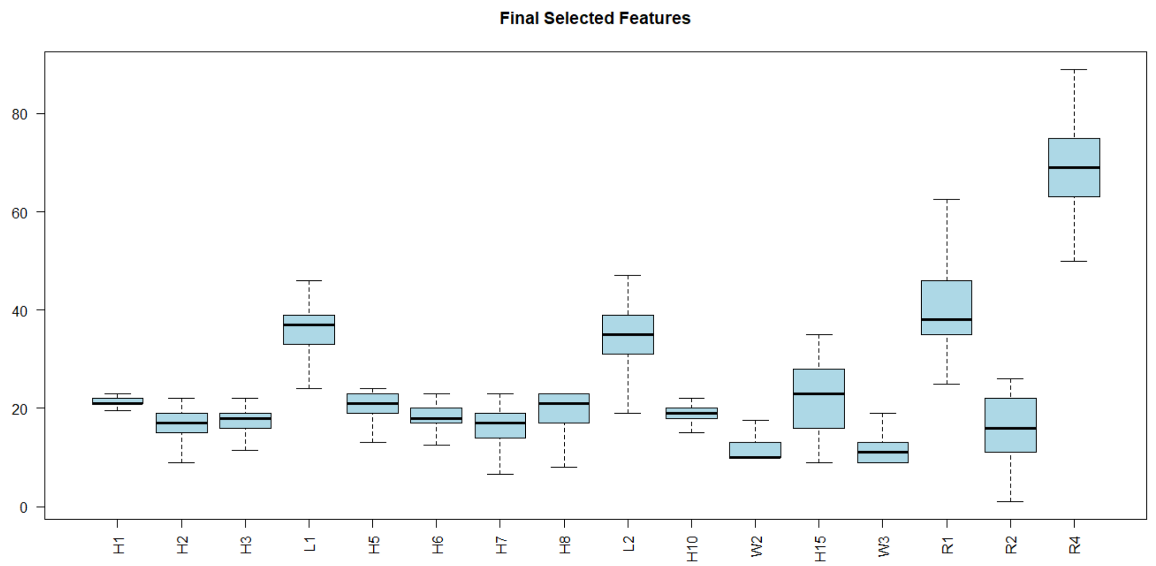
- Selected features with a **p-value < 0.05**.

```
> print("Significant features based on ANOVA:")
[1] "Significant features based on ANOVA:"
> print(significant_features)
 [1] "H1"  "H2"  "H3"  "L1"  "H5"  "H6"  "H7"  "H8"  "L2"  "H10" "W2"  "H15" "W3"  "L3"  "R1"
[16] "R2"  "R3"  "R4"
```

- Combined the results from **Correlation and ANOVA** to retain only meaningful features.

```
> print("Final selected features after combining correlation and ANOVA results:")
[1] "Final selected features after combining correlation and ANOVA results:"
> print(final_features)
 [1] "H1"  "H2"  "H3"  "L1"  "H5"  "H6"  "H7"  "H8"  "L2"  "H10" "W2"  "H15" "W3"  "R1"  "R2"
[16] "R4"
```

- Plotted **boxplots** of the **final selected features**.



- **Stratified train-test split (80%-20%)** to maintain expression class distribution.

- Verified that both train and test sets have a balanced class distribution**.**

```
> table(train_data$Expression)

   ANGER  DISGUST     FEAR      JOY  NEUTRAL  SADNESS SURPRISE
      24       24       24       24       24       24       24
> table(test_data$Expression)

   ANGER  DISGUST     FEAR      JOY  NEUTRAL  SADNESS SURPRISE
       6        6        6        6        6        6        6
```

- Applied **(PCA)** to reduce dimensionality while retaining **80% variance**.

```
> summary(pca_model)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6    PC7     PC8     PC9
Standard deviation     1.9391 1.5759 1.1773 1.00852 0.98752 0.76905 0.7574 0.59445 0.55140
Proportion of Variance 0.3133 0.2069 0.1155 0.08476 0.08127 0.04929 0.0478 0.02945 0.02534
Cumulative Proportion  0.3133 0.5203 0.6358 0.72055 0.80181 0.85110 0.8989 0.92835 0.95368
                          PC10    PC11    PC12
Standard deviation     0.48312 0.41559 0.38690
Proportion of Variance 0.01945 0.01439 0.01247
Cumulative Proportion  0.97313 0.98753 1.00000
```

- **Transformed train and test sets using selected principal components**.

4. Model Training & Evaluation:

- **<u>What I Did:</u>**

- I used repeated **10-fold cross-validation (5 times)** to get a more reliable accuracy estimate since my dataset is small.

- Evaluated three models: **Naïve Bayes, Decision Tree, and K-Nearest Neighbours (KNN)**.

- **Naive Bayes (NB):** Tuned **Laplace smoothing (fL)**, enabled/disabled **kernel-based density estimation**, and adjusted **smoothing bandwidth**.

- **Decision Tree (DT):** Optimized **complexity parameter (cp)**, **tree depth**, **minimum split size**, and **bucket size**.

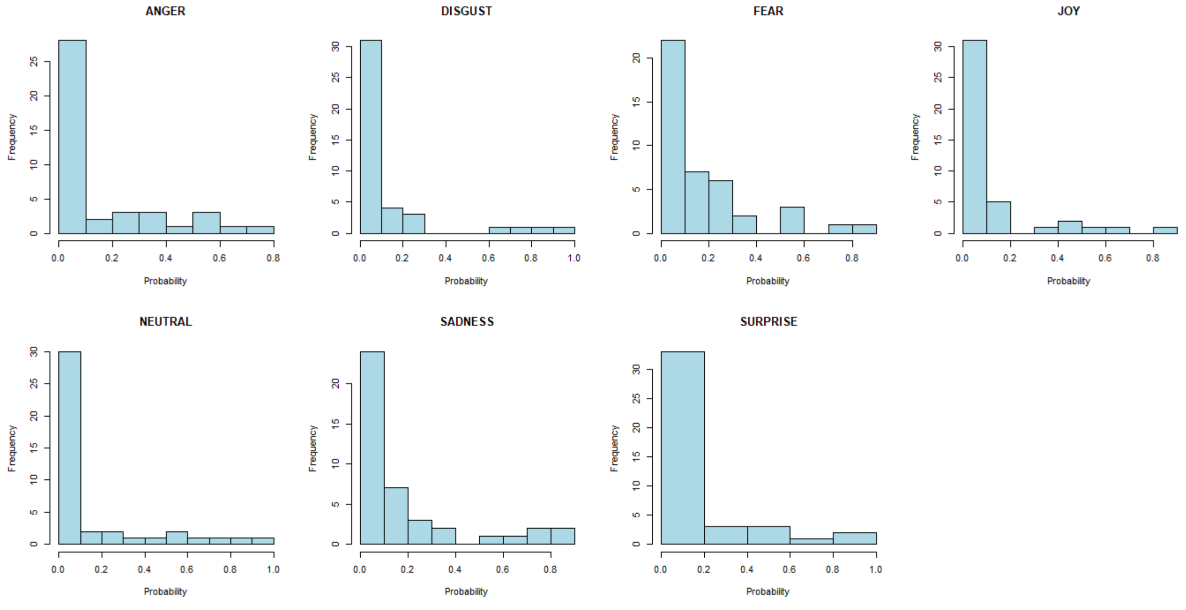
- **K-Nearest Neighbours (KNN):** Tuned **k-value (kmax)**, distance metric (**Manhattan vs. Euclidean**), and **kernel function** for optimal performance.

- Used **confusion matrices** to compute **accuracy, precision, recall, and F1-score**.

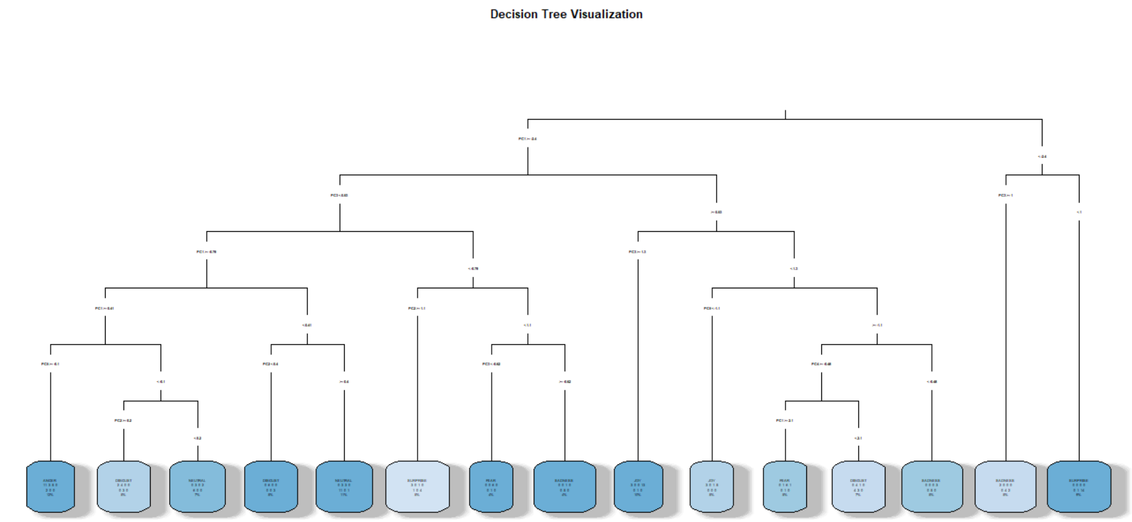
```
Performance Metrics:
> print(metrics)
          Model  Accuracy Precision    Recall  F1_Score
1   Naive Bayes 0.5952381 0.6374150 0.5952381 0.6074925
2 Decision Tree 0.4523810 0.5126984 0.4523810 0.4330035
3           KNN 0.7380952 0.7474490 0.7380952 0.7352076
```

- **Compared model performance using bar plots.**

- **Visualized the probability distributions for each class in the Naïve Bayes model using histograms.**



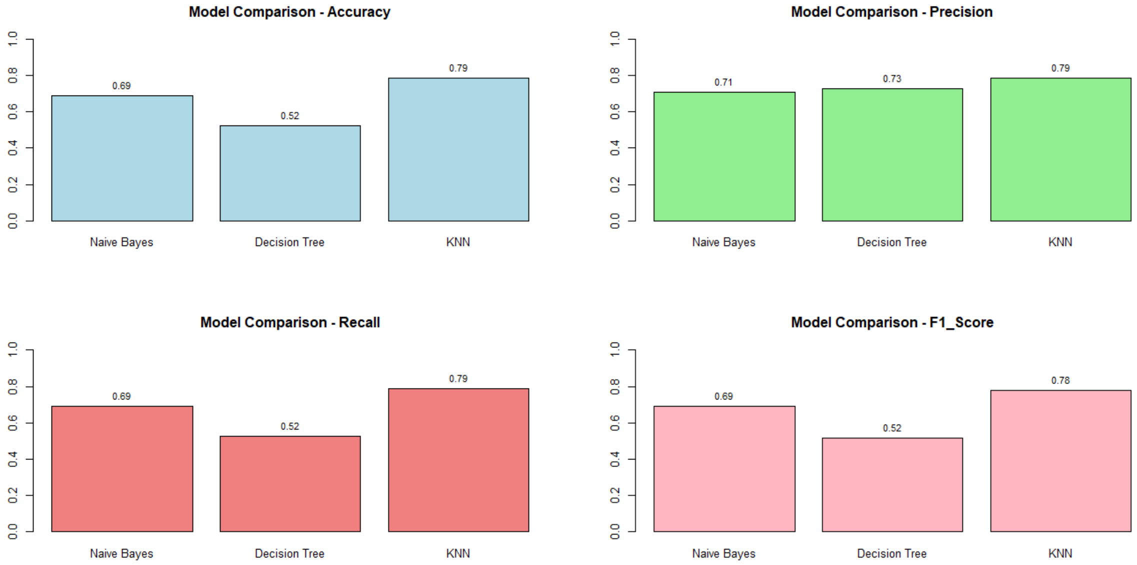
- **Plotted and visualized the decision tree**.



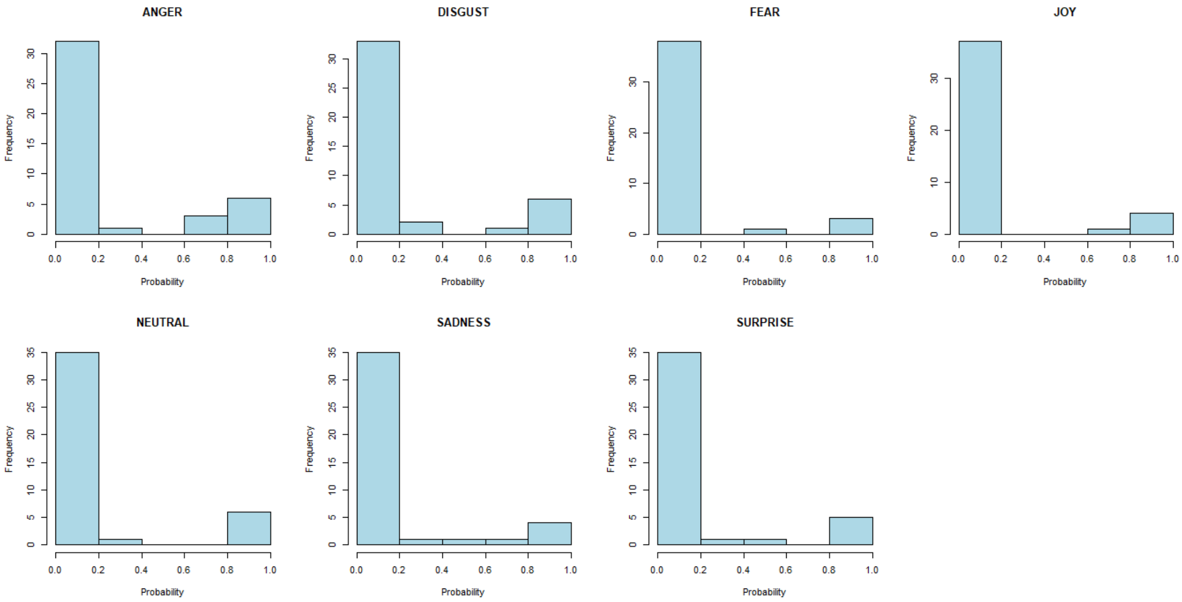
- Evaluated **Naïve Bayes, Decision Tree, and K-Nearest Neighbours (KNN) before applying PCA**.

```
Performance Metrics:
> print(metrics_before)
          Model  Accuracy Precision    Recall  F1_Score
1   Naive Bayes 0.6904762 0.7083900 0.6904762 0.6912754
2 Decision Tree 0.5238095 0.7284580 0.5238095 0.5160815
3           KNN 0.7857143 0.7853741 0.7857143 0.7799581
```

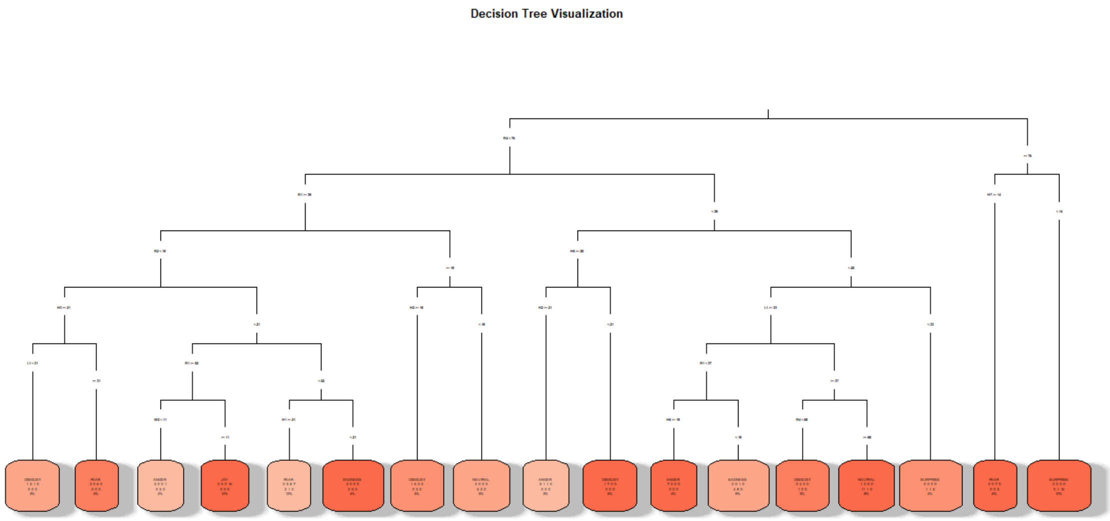
- **Compared model performance before PCA**.



- **Plotted the probability distributions before PCA for each class in the Naïve Bayes model using histograms**.



- **Plotted and visualized the decision tree before PCA**.



- <u>**What I Learned:**</u>

- **PCA** appears to degrade model performance.

Model	Before PCA	After PCA	Change
Naïve Bayes	69%	59.5%	▼ -9.5%
Decision Tree	52.4%	45.2%	▼ -7.2%
KNN	78.6%	73.8%	▼ -4.8%

- **Naïve Bayes** experiences the most significant decline in accuracy, **dropping by 9.5%.** This suggests that the original features contained crucial information that was lost during dimensionality reduction.

- **Decision Tree** also suffer from PCA**,** with **a 7.2% decrease** in accuracy. This indicates that the raw features provided better decision splits than the transformed principal components.

- **KNN** remains the most resilient model**,** exhibiting only **a 4.8% drop** in accuracy. This suggests that PCA had a smaller impact on its classification ability.

- Based on the confusion matrix, **Naïve Bayes (NB) after PCA** performed best for **DISGUST, SADNESS, and SURPRISE**, but struggled with **ANGER, FEAR, JOY, and NEUTRAL**.

```
Confusion Matrix for Naive Bayes:
> print(nb_cm$table)
          Reference
Prediction ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
  ANGER        3       1    0   3       1       1        0
  DISGUST      0       4    0   0       0       0        0
  FEAR         0       0    3   0       1       0        1
  JOY          1       0    0   3       0       1        0
  NEUTRAL      0       1    3   0       3       0        0
  SADNESS      2       0    0   0       0       4        0
  SURPRISE     0       0    0   0       1       0        5
```

- Based on the confusion matrix, **Decision Tree (DT) after PCA** performed best for **DISGUST, NEUTRAL, and SURPRISE**, but had the most difficulty with **ANGER, FEAR, JOY, and SADNESS**.

```
Confusion Matrix for Decision Tree:
> print(dt_cm$table)
          Reference
Prediction ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
  ANGER        1       1    0   1       0       0        0
  DISGUST      2       4    1   1       2       2        0
  FEAR         0       0    1   1       0       2        1
  JOY          0       0    0   2       0       0        0
  NEUTRAL      2       1    0   1       4       0        0
  SADNESS      1       0    0   0       0       2        0
  SURPRISE     0       0    4   0       0       0        5
```

- Based on the confusion matrix, **K-Nearest Neighbours (KNN) after PCA** performed best for **DISGUST, JOY, NEUTRAL, SADNESS, and SURPRISE**, but struggled with **ANGER, FEAR**.

```
Confusion Matrix for KNN:
> print(knn_cm$table)
          Reference
Prediction ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
  ANGER        3       0    0   0       0       1        0
  DISGUST      0       6    0   1       0       0        0
  FEAR         2       0    3   0       0       0        1
  JOY          0       0    0   5       0       0        0
  NEUTRAL      0       0    2   0       4       0        0
  SADNESS      1       0    1   0       1       5        0
  SURPRISE     0       0    0   0       1       0        5
```

- Based on the confusion matrix, **Naïve Bayes (NB) before PCA** performed best for **ANGER, DISGUST, JOY, NEUTRAL, SADNESS, and SURPRISE**, but struggled with **FEAR**.

```
Confusion Matrix for Naive Bayes:
> print(nb_cm_before$table)
          Reference
Prediction ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
  ANGER        4       1    0   2       0       2        0
  DISGUST      1       5    1   0       0       0        0
  FEAR         0       0    2   0       1       0        1
  JOY          1       0    0   4       0       0        0
  NEUTRAL      0       0    1   0       5       0        0
  SADNESS      0       0    2   0       0       4        0
  SURPRISE     0       0    0   0       0       0        5
```

- Based on the confusion matrix, **Decision Tree (DT) before PCA** performed best for **FEAR, JOY, and SADNESS**, but struggled the most with **ANGER, DISGUST, NEUTRAL, and SURPRISE**.

```
Confusion Matrix for Decision Tree:
> print(dt_cm_before$table)
          Reference
Prediction ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
  ANGER        1       0    0   0       0       0        0
  DISGUST      2       3    1   0       0       0        0
  FEAR         2       3    5   1       2       2        3
  JOY          0       0    0   5       0       0        0
  NEUTRAL      0       0    0   0       1       0        0
  SADNESS      1       0    0   0       2       4        0
  SURPRISE     0       0    0   0       1       0        3
```

- Based on the confusion matrix, **K-Nearest Neighbours (KNN) before PCA** performed best for **ANGER, DISGUST, JOY, NEUTRAL, SADNESS, and SURPRISE**, but had difficulties with **FEAR**.

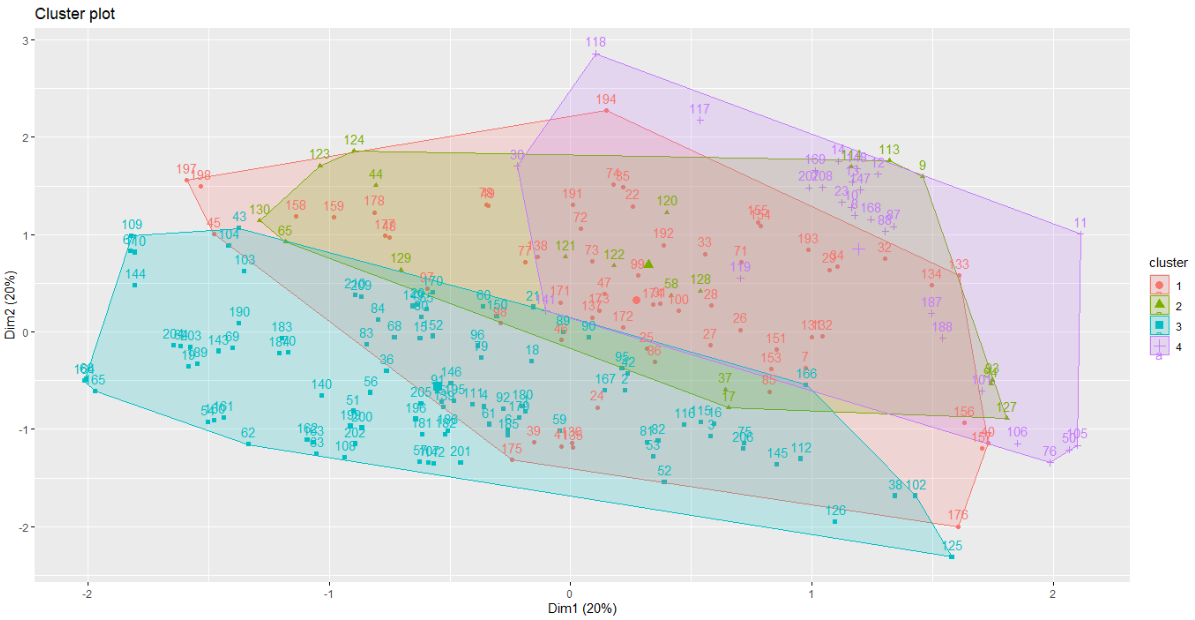
```
Confusion Matrix for KNN:
> print(knn_cm_before$table)
          Reference
Prediction ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
  ANGER        4       1    0   0       0       0        0
  DISGUST      0       5    1   0       0       0        0
  FEAR         1       0    2   0       1       0        1
  JOY          0       0    0   6       0       0        0
  NEUTRAL      0       0    2   0       5       0        0
  SADNESS      1       0    1   0       0       6        0
  SURPRISE     0       0    0   0       0       0        5
```

- **K-Nearest Neighbours (KNN)** appears to be the most effective model, as it achieved the highest performance and classified expressions with the greatest accuracy.

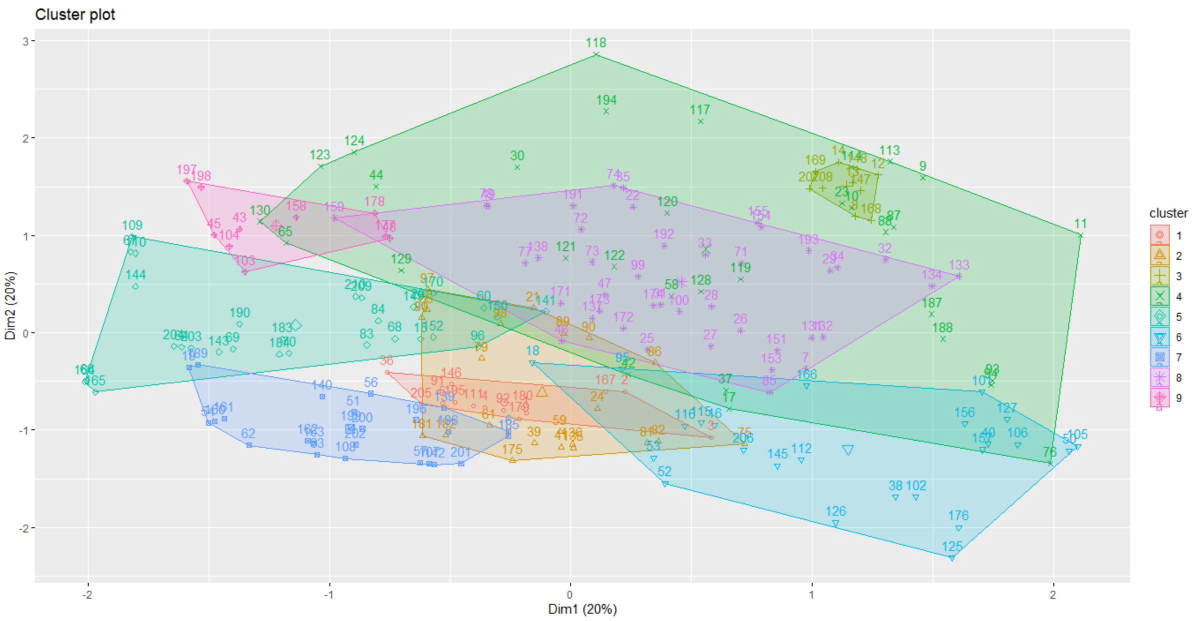
5. Clustering:

- <u>**What I Did:**</u>
- Used only feature columns for **unsupervised clustering**.
- Reduced dimensionality by applying **PCA** and keeping the top components explaining **~80% of the variance**.
- **Applied and evaluated three clustering algorithms**.
- **K-Means:** Tuned **k**.
- **Gaussian Mixture Model (GMM):** Applied **GMM** on **PCA-transformed** data.
- **DBSCAN:** Tuned **eps** and **minPts** values.

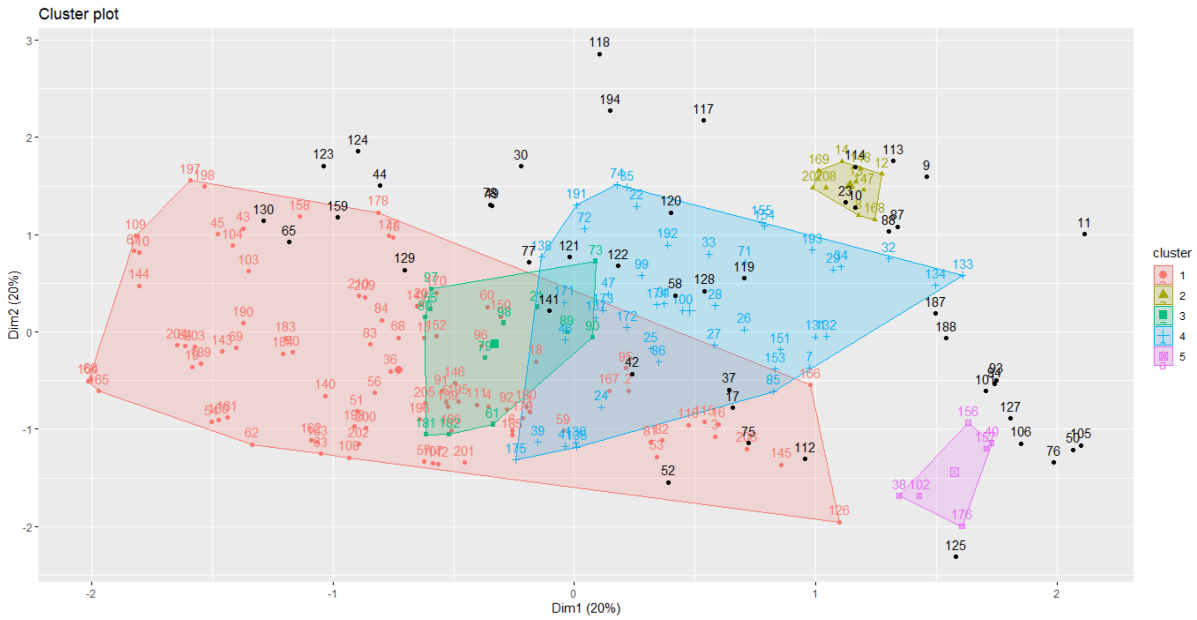
- **Visualized clusters** for **K-Means**.



- **Visualized clusters** for **GMM**.



- **Visualized clusters** for **DBSCAN**.



- Evaluated **K-Means** clusters using **Silhouette Score**.

```
> summary(silhouette_score)
Silhouette of 210 units in 4 clusters from silhouette.default(x = kmeans_result$cluster, dist = dist(p
ca_data)) :
 Cluster sizes and average silhouette widths:
       63        19       101        27
0.3277569 0.2418907 0.3284226 0.2468802
Individual silhouette widths:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.06227  0.19077  0.33332  0.30991  0.43627  0.54855
```

- Evaluated **GMM** clusters using **Silhouette Score**.

```
> summary(silhouette_score_gmm)
Silhouette of 210 units in 9 clusters from silhouette.default(x = gmm_clusters, dist = dist(pca_data))
:
 Cluster sizes and average silhouette widths:
         15          22          10          32          30          24          25          42
 0.34702308  0.21020592  0.69523420 -0.04694377  0.11927039  0.05361440  0.31409806  0.29823997
         10
 0.58512014
Individual silhouette widths:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-0.51485  0.06605  0.23213  0.22083  0.39303  0.75893
```

- Evaluated **DBSCAN** clusters using **Silhouette Score**.

```
> summary(silhouette_score_dbscan)
Silhouette of 210 units in 6 clusters from silhouette.default(x = dbscan_result$cluster, dist = dist(p
ca_data)) :
 Cluster sizes, ids = (0, 1, 2, 3, 4, 5), and average silhouette widths:
        46         92         10         12         44          6
-0.1511204  0.1573108  0.7021293  0.4781748  0.2585512  0.7021414
Individual silhouette widths:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.566052 -0.009097  0.208835  0.170807  0.348887  0.774134
```

- <u>**What I Learned:**</u>

- **K-Means** imposed 4 clusters, but there are 7 true expressions. Average silhouette ~0.31 suggests moderate separation.

- **GMM** chose 9 clusters on its own, more than actual expressions. Two clusters had very high silhouette (> 0.6), indicating strong internal cohesion. One had a negative average silhouette, a warning sign.

- **DBSCAN** detected some very cohesive, small groups (silhouette ~0.7). 46 instances were labelled as noise, nearly 22% of the data.

- Based on the confusion matrix, **K-Means** is able to clearly isolate Surprise (Cluster 2), but struggles to separate similar expressions (especially the "negative" ones like Sadness, Fear, Disgust). Too few clusters for 7 expressions.

```
> table(Cluster = kmeans_result$cluster, Label = final_data$Expression)
       Label
Cluster ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
      1    12      12   12  12       8       7        0
      2     0       0    0   0       0       3       16
      3    15      15   14  14      19      15        9
      4     3       3    4   4       3       5        5
```

- Based on the confusion matrix, **GMM** creates clusters, some of which do isolate specific expressions (especially Surprise and Joy). Others are highly mixed, possibly detecting variation within expressions.

```
> table(Cluster = gmm_result$classification, Label = final_data$Expression)
       Label
Cluster ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
      1     2       2    2   0       4       3        2
      2     5       0    3  11       0       3        0
      3     1       3    3   0       3       0        0
      4     2       0    0   5       1       4       20
      5     4       6    7   3       6       4        0
      6     2       4    2   0       1       7        8
      7     5       6    4   0       7       3        0
      8     6       8    9  11       5       3        0
      9     3       1    0   0       3       3        0
```

- Based on the confusion matrix, **DBSCAN** is catching expressions like Surprise, but struggles with the rest. Noise label (Cluster 0) is rich in actual emotion, which might reflect very strong expressions.

```
> table(Cluster = dbscan_result$cluster, Label = final_data$Expression)
       Label
Cluster ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
      0     2       1    1   9       1       8       24
      1    15      16   14   6      21      14        6
      2     1       3    3   0       3       0        0
      3     3       0    0   7       0       2        0
      4     8       7   12   8       5       4        0
      5     1       3    0   0       0       2        0
```

- Clustering algorithms struggle to distinguish facial expressions due to overlapping feature patterns and subtle differences between emotions, leading to mixed cluster compositions. This suggests unsupervised methods alone are insufficient for clearly separating all seven expressions.