• Project Flowchart:

1. Project Understanding:

- **<u>Understanding Face Expression Recognition via Measurable Features:</u>**

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.

- **<u>Problem Statement:</u>**

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**.

- **<u>What I Want to Accomplish:</u>**

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:

- **<u>Exploring the Facial Expression Dataset:</u>**

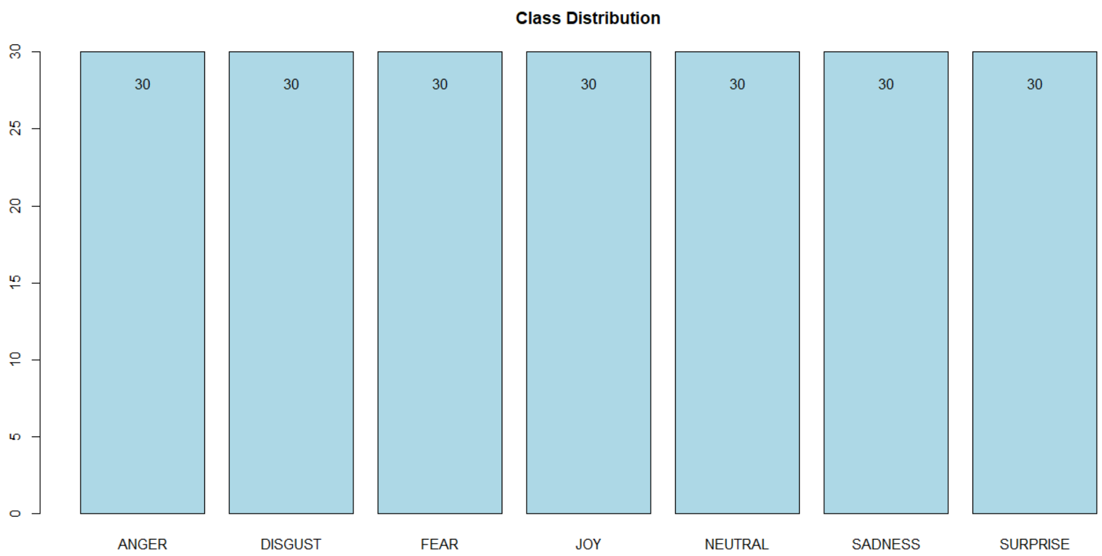
The dataset consists of 210 instances from the Cohn-Kanade database, each with 25 facial measurement features and a class label representing one of seven facial expressions.

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>**

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



- Extracted the **features data**.

features_data

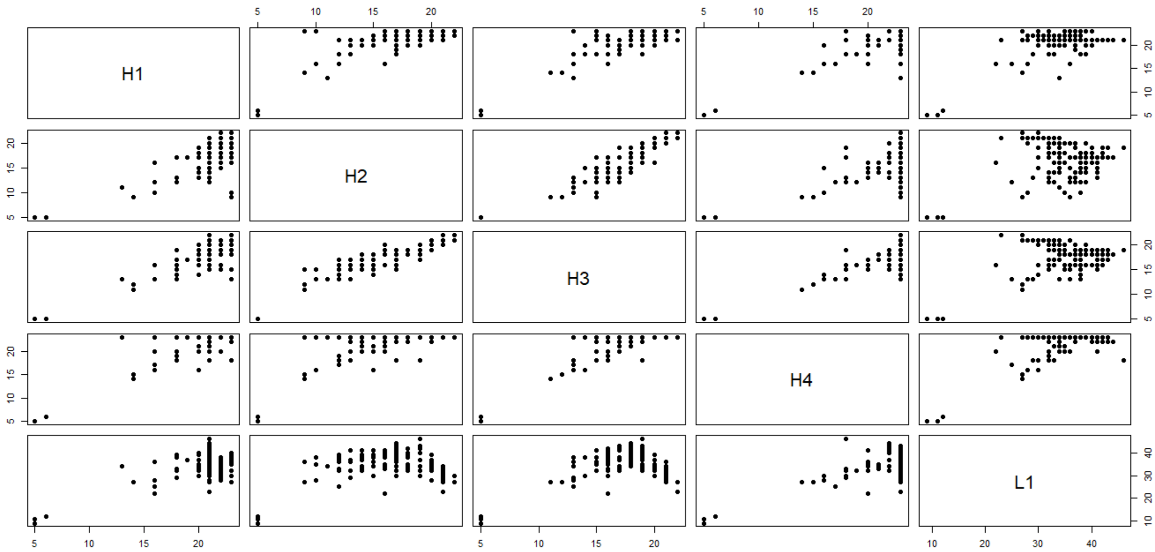
	H1	H2	H3	H4	L1	H5	H6	H7	H8	L2	H9	H10	W1	H11	H12	W2	H13	H14	H15	W3	L3	R1	R2	R3	R4
1	22	17	19	23	38	20	18	16	23	25	15	19	17	14	17	10	17	16	23	9	24	33	23	14	73
2	21	15	18	23	39	24	17	14	23	27	14	19	17	13	15	10	17	15	20	9	21	38	15	25	65
3	21	16	18	23	38	17	17	14	15	31	15	17	17	13	16	10	17	15	22	9	30	36	22	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.
- **Stratified train-test split (80%-20%)** to maintain expression class distribution.
- Verified that both train and test sets have a balanced class distribution**.**

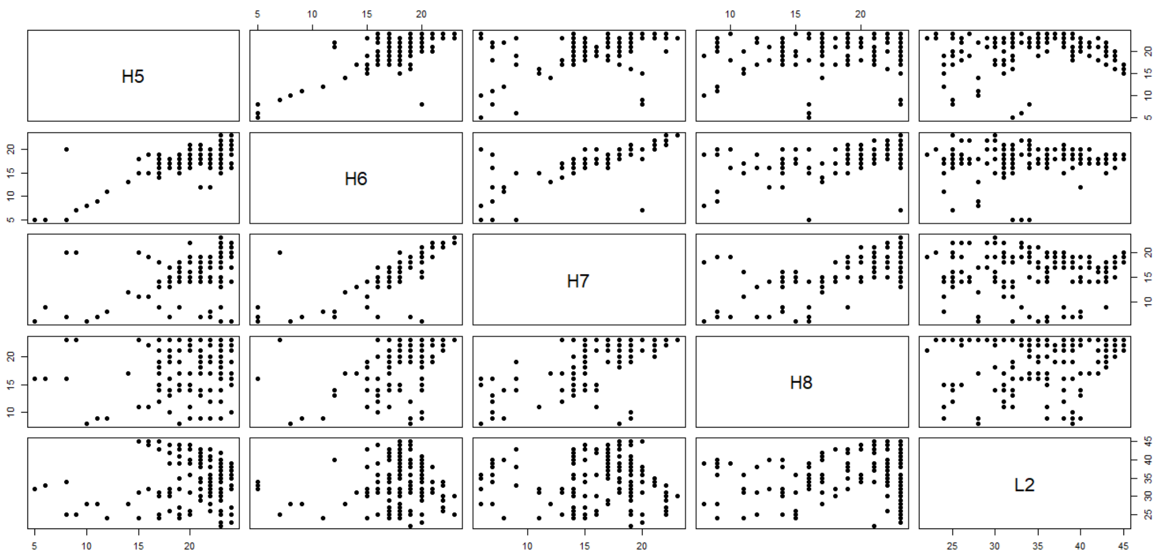
```
> # Split the dataset (80% train, 20% test)
> split_result <- split_process(features_data, seed = GLOBAL_SEED)

  ANGER  DISGUST     FEAR      JOY  NEUTRAL  SADNESS SURPRISE
     24       24       24       24       24       24       24

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

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

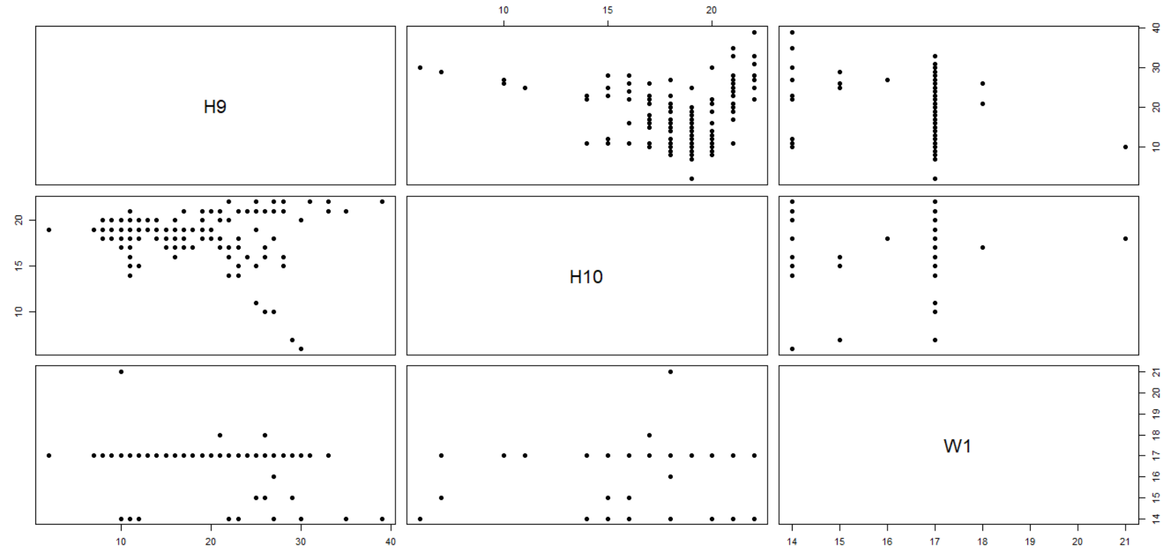
Pairwise Scatter Plot for the Left_Eyebrow



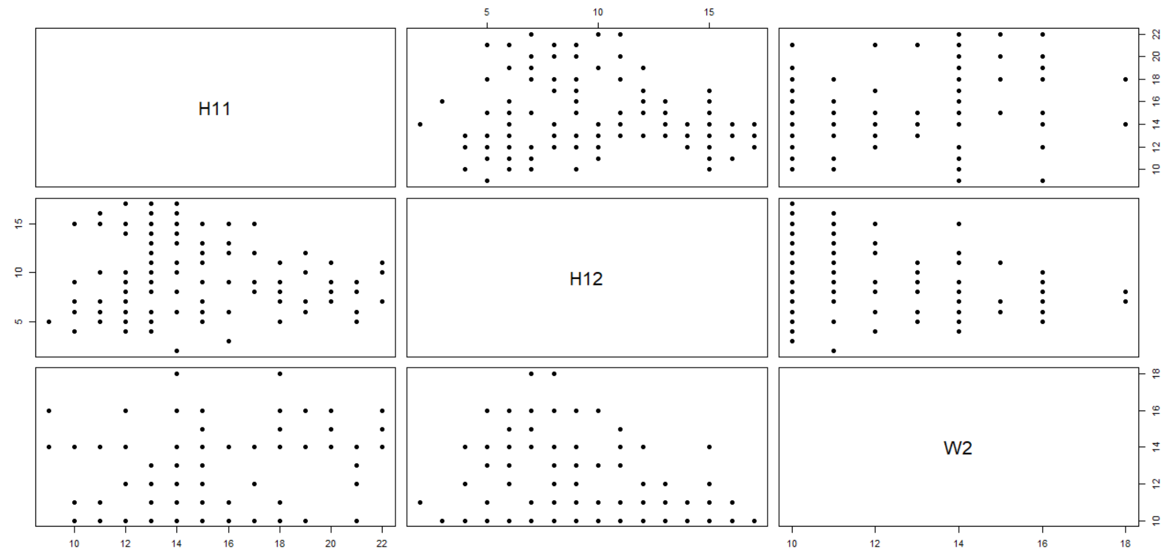
Pairwise Scatter Plot for the Right_Eyebrow



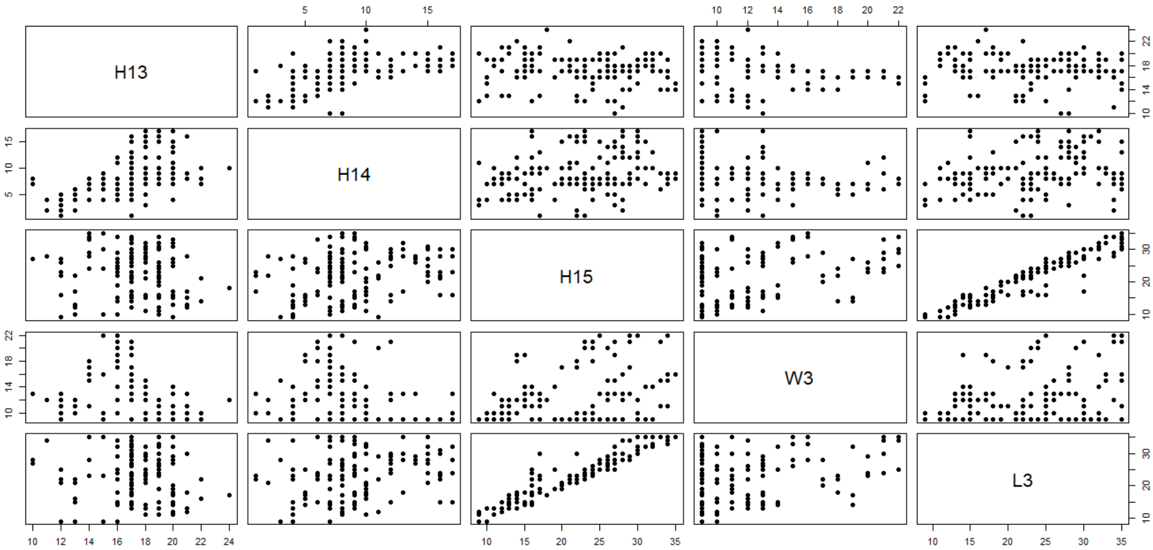
Pairwise Scatter Plot for the Left_Eye



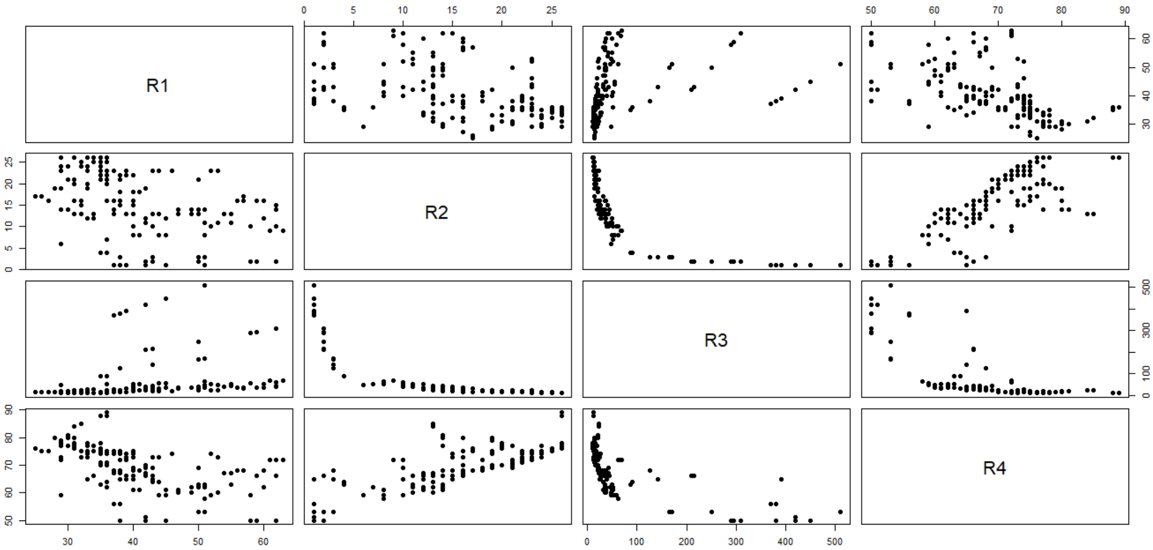
Pairwise Scatter Plot for the Right_Eye



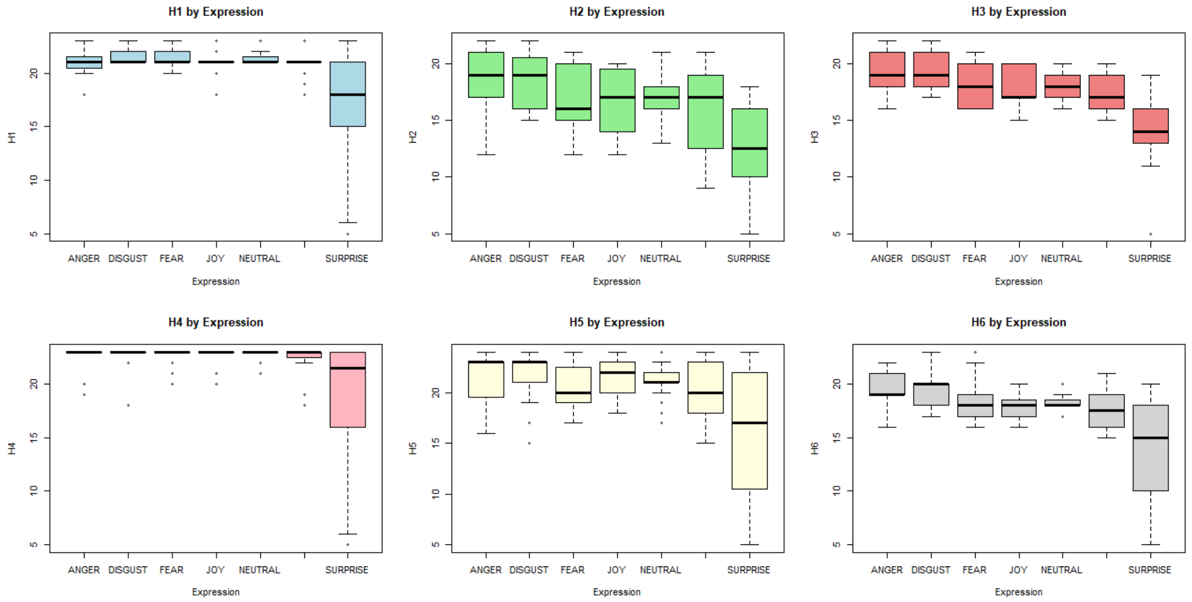
Pairwise Scatter Plot for the Mouth

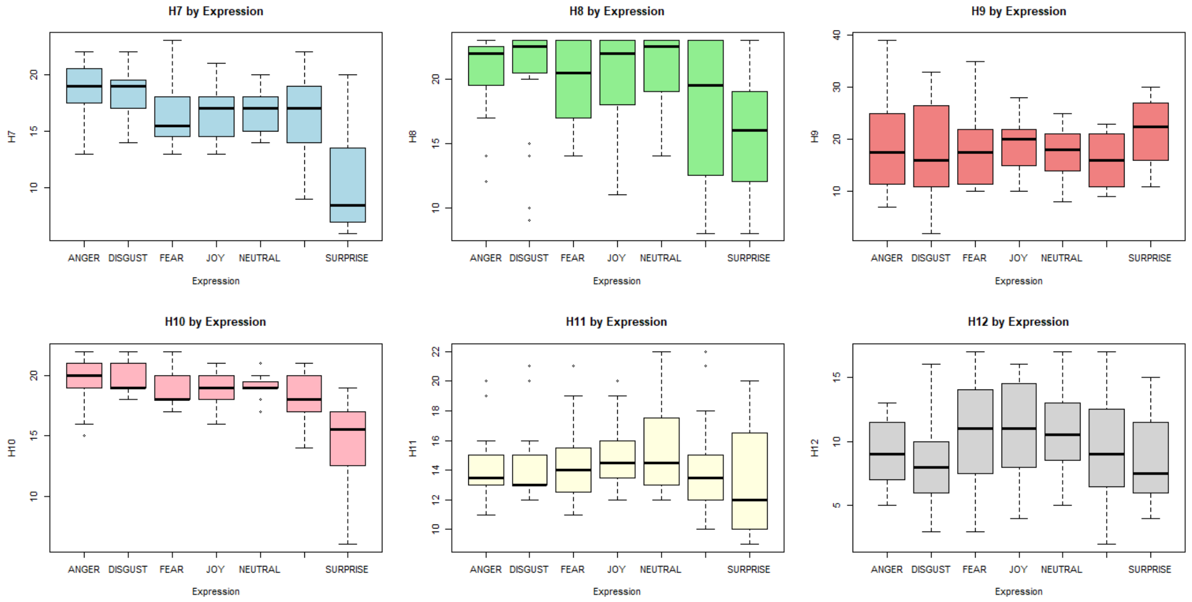


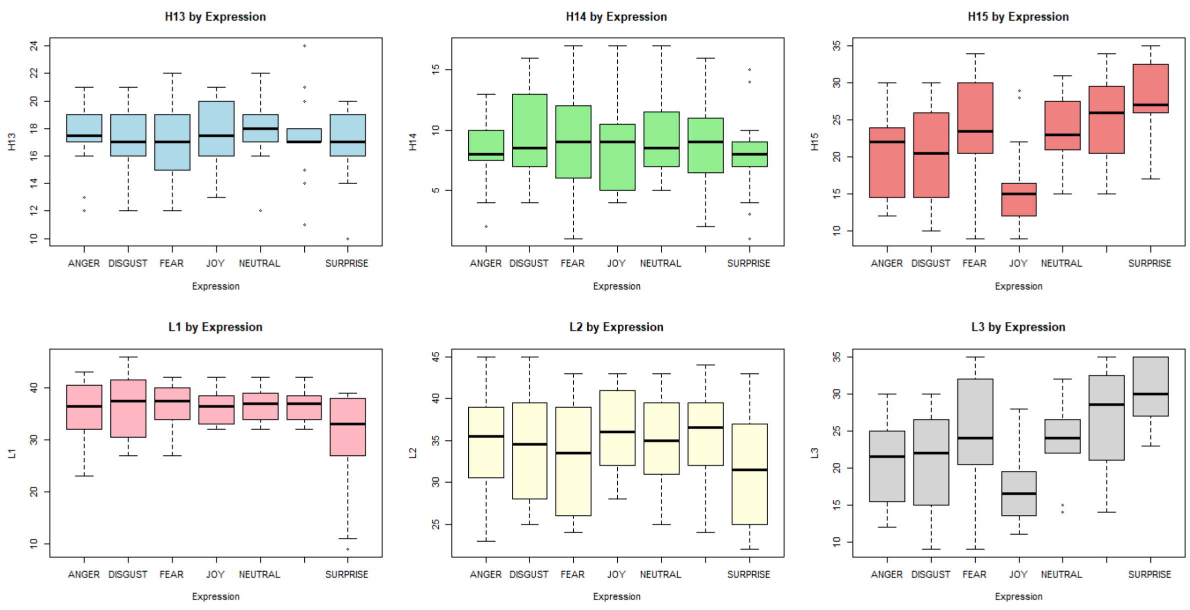
Pairwise Scatter Plot for the Relationships

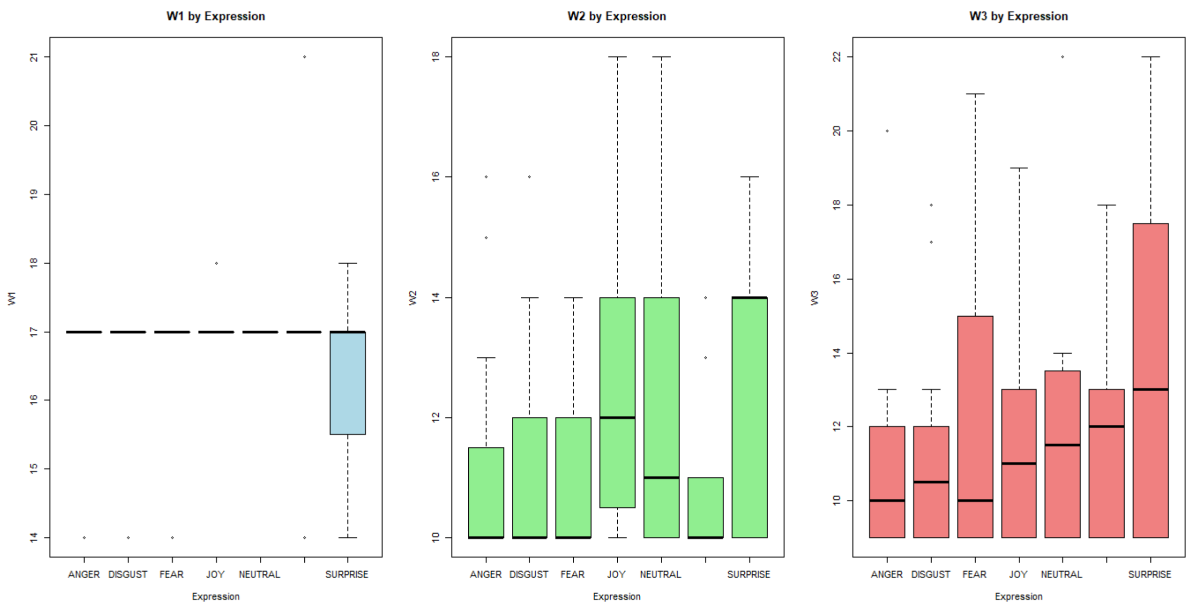


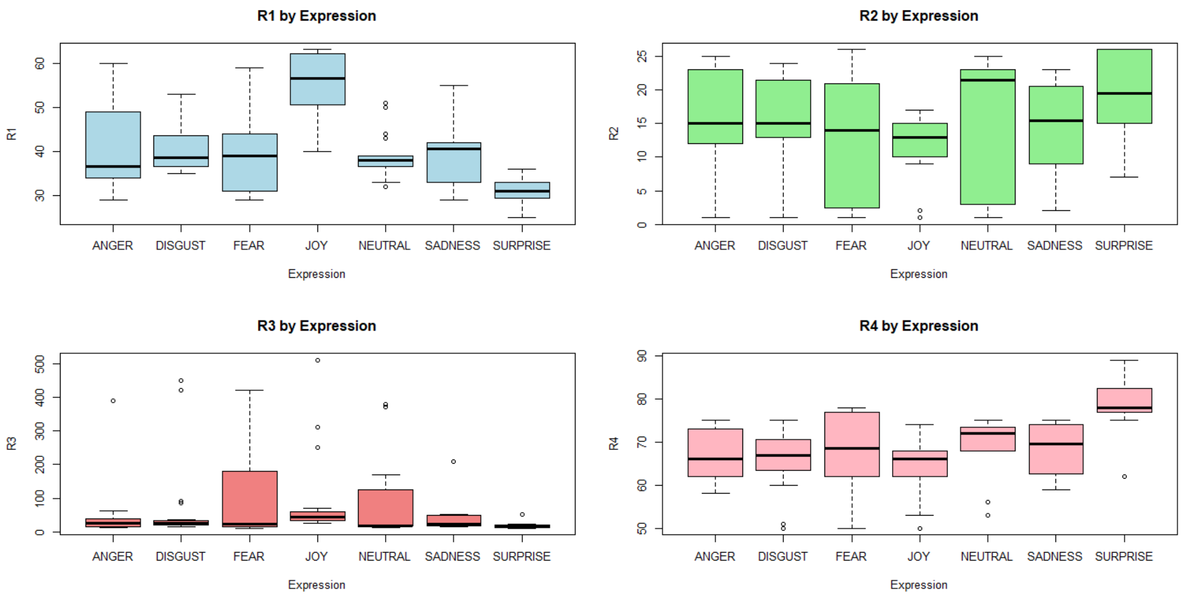
- Plotted **boxplots** of each feature grouped by **Expression** to analyse how facial features vary across emotions.



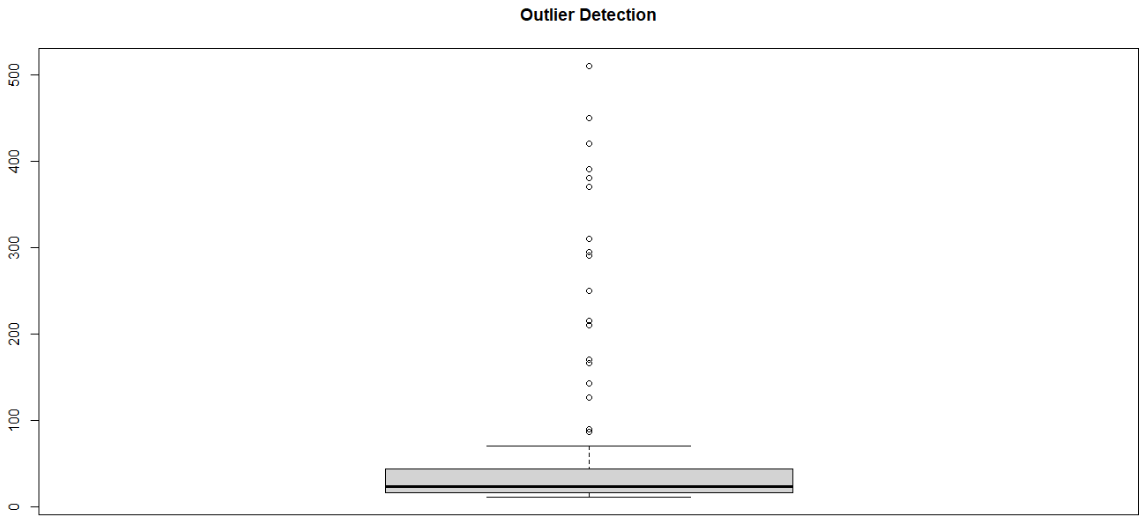


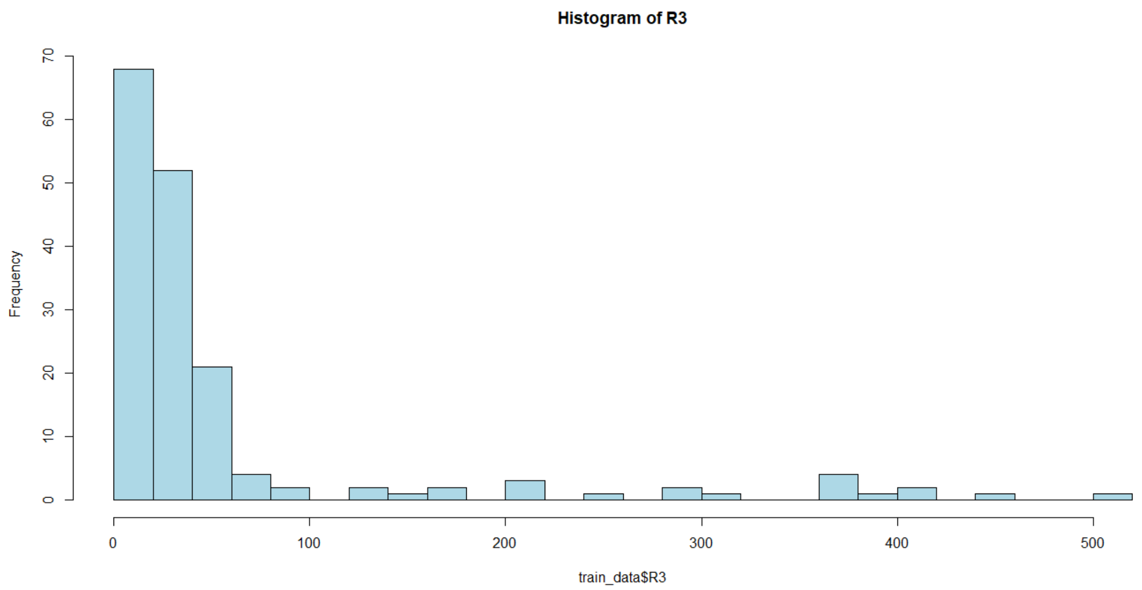

H13 by Expression
H14 by Expression
H15 by Expression
L1 by Expression
L2 by Expression
L3 by Expression
ANGER DISGUST FEAR JOY NEUTRAL SURPRISE
Expression

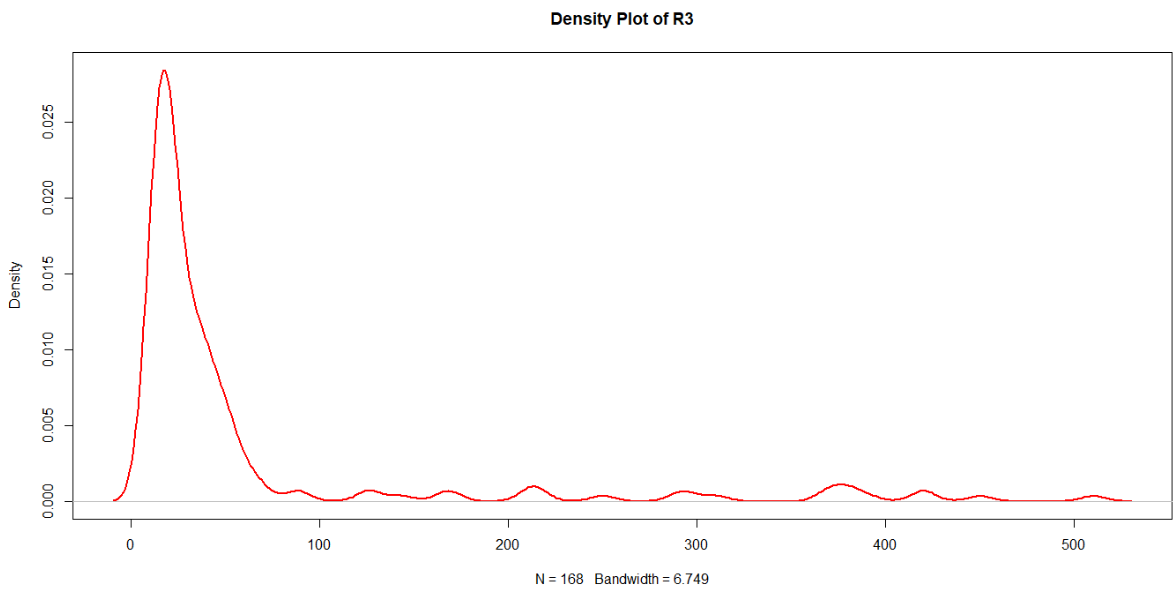

W1 by Expression
W2 by Expression
W3 by Expression
ANGER DISGUST FEAR JOY NEUTRAL SURPRISE
Expression



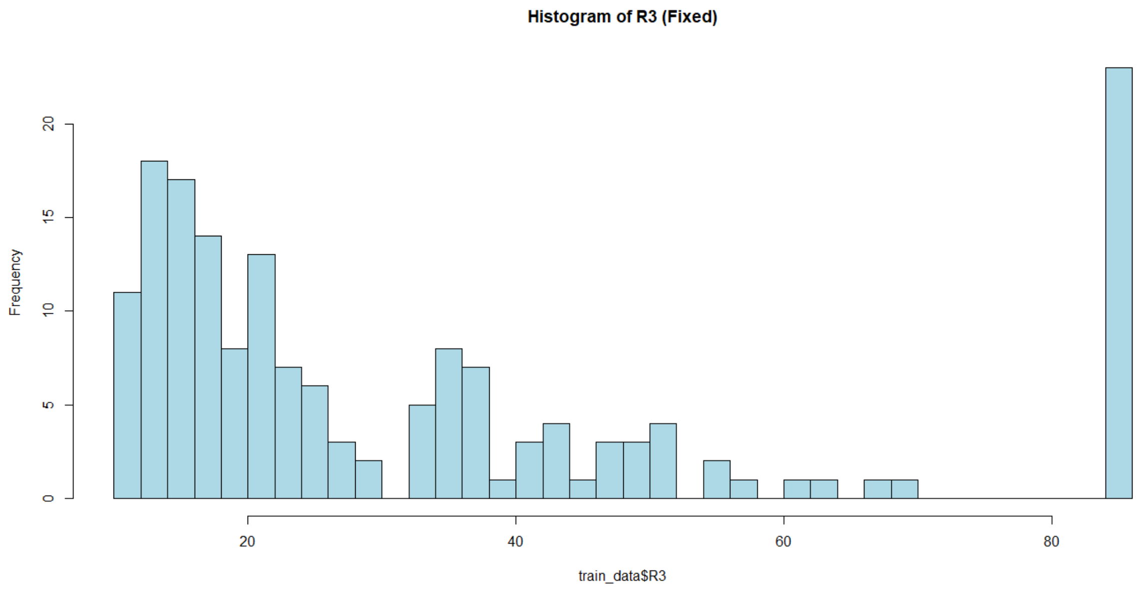
- Focused on feature **R3**, detecting extreme values.

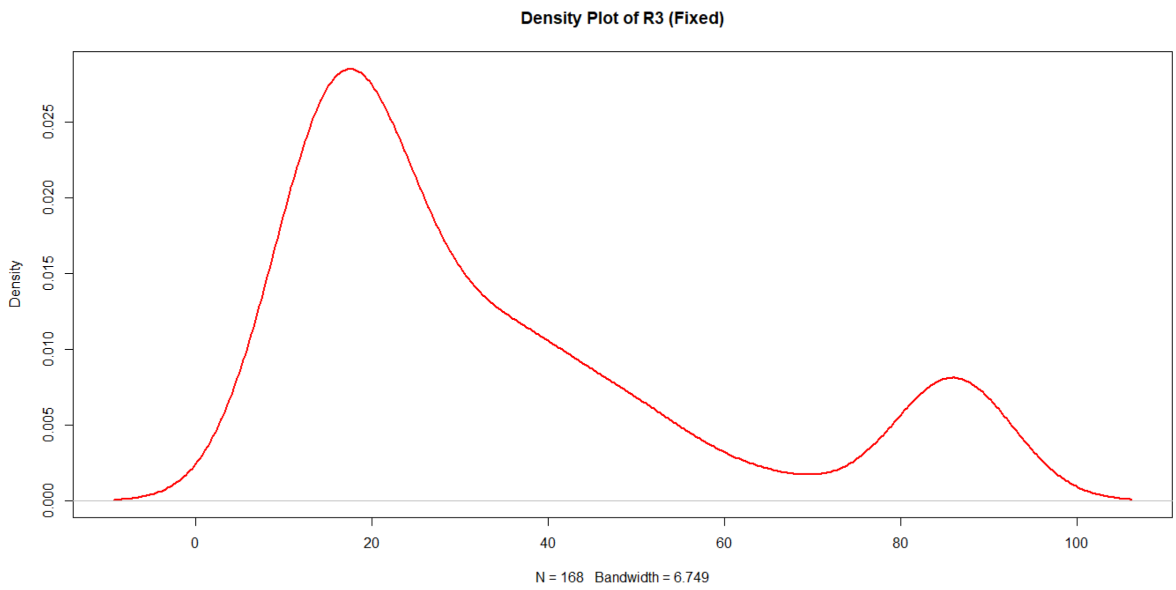



Histogram of R3
Frequency
0
10
20
30
40
50
60
70
0
100
200
300
400
500
train_data$R3


Density Plot of R3
Density
0.000
0.005
0.010
0.015
0.020
0.025
0
100
200
300
400
500
N = 168   Bandwidth = 6.749

- Used **IQR-based capping** to replace extreme values beyond Q3 + 1.5 * IQR with the upper limit.





- **Applied Winsorization to cap outliers on data**.

3. Data Preparation:

- <u>**Dataset Preprocessing Overview:**</u>

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.

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

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

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

```
> data_for_anova <- cbind(train_data)
> anova_results <- anova_analysis(data_for_anova)
[1] "ANOVA p-values:"
          H1           H2           H3           H4           L1           H5           H6 
4.272848e-01 5.920407e-10 6.730479e-17 4.272848e-01 1.949558e-03 1.666010e-05 1.298621e-13 
          H7           H8           L2           H9          H10           W1          H11 
8.756404e-18 5.373599e-05 1.222030e-01 3.330201e-01 1.826468e-14 4.272848e-01 8.702703e-02 
         H12           W2          H13          H14          H15           W3           L3 
1.181246e-01 7.910576e-03 9.521240e-01 8.505060e-01 4.733985e-11 8.337680e-02 7.477352e-12 
          R1           R2           R3           R4 
9.298742e-21 1.152568e-02 7.065387e-04 2.970516e-10 
[1] "Significant features based on ANOVA:"
 [1] "H2"  "H3"  "L1"  "H5"  "H6"  "H7"  "H8"  "H10" "W2"  "H15" "L3"  "R1"  "R2"  "R3"  "R4"
```

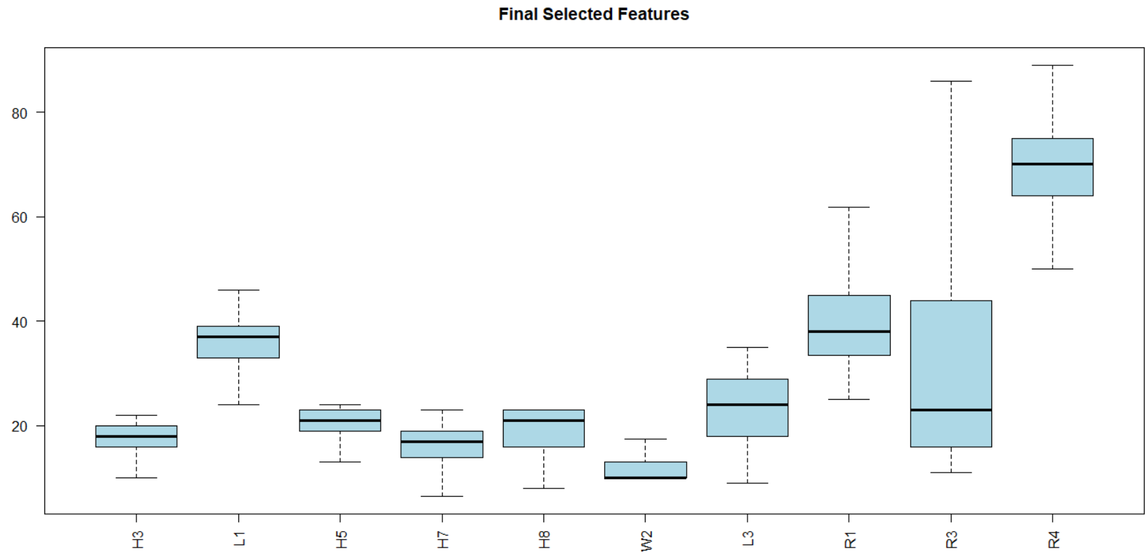
- Computed the **correlation matrix** to identify highly correlated features.

- Selected features with **correlation > 0.8** to reduce redundancy.

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

```
> print("Final selected features after combining ANOVA and correlation results:")
[1] "Final selected features after combining ANOVA and correlation results:"
> print(final_features)
 [1] "H3" "L1" "H5" "H7" "H8" "W2" "L3" "R1" "R3" "R4"
```

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



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

```
> # Perform PCA
> pca_result <- pca_process(train_final, test_final)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8     PC9
Standard deviation     2.0649 1.3511 1.1053 0.90162 0.74385 0.69741 0.60575 0.44266 0.43776
Proportion of Variance 0.4264 0.1826 0.1222 0.08129 0.05533 0.04864 0.03669 0.01959 0.01916
Cumulative Proportion  0.4264 0.6089 0.7311 0.81240 0.86773 0.91637 0.95306 0.97266 0.99182
                          PC10
Standard deviation     0.28603
Proportion of Variance 0.00818
Cumulative Proportion  1.00000
[1] 168   5
[1] 42  5
```

- **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 (3 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):** Selected **split decision metric**, and optimized **tree depth**, **minimum split size**, and **bucket size**.

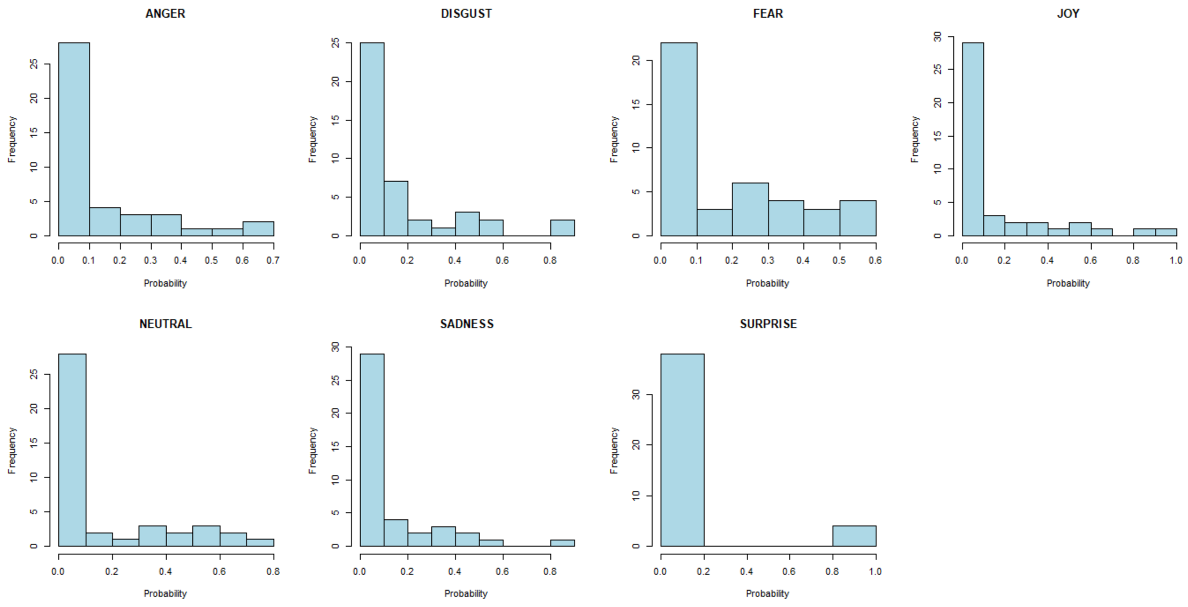
- **K-Nearest Neighbours (KNN):** Tuned **k-value (kmax)**, distance metric (**Manhattan vs. Euclidean**), and **kernel function** for optimal performance.

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

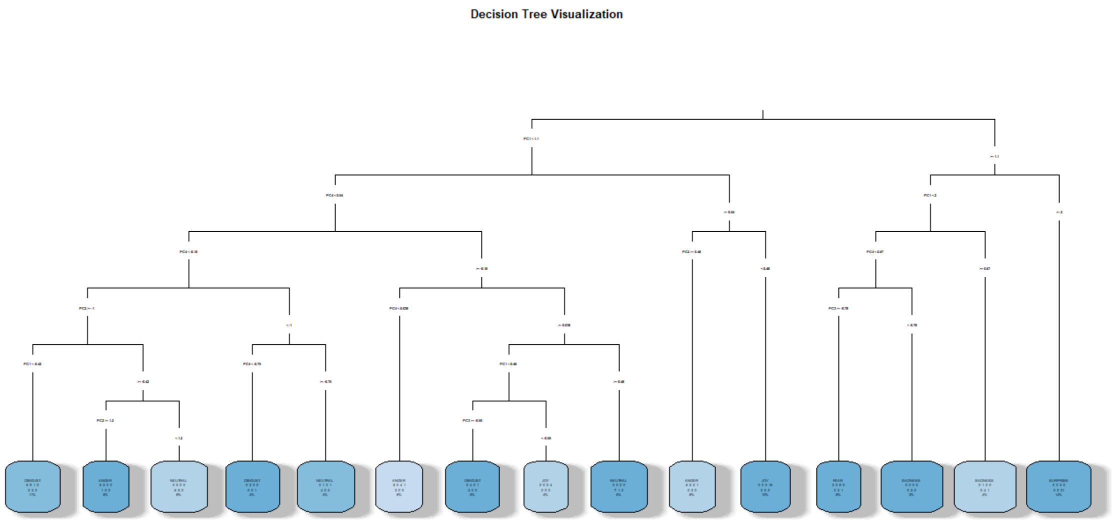
```
Performance Metrics:
> print(metrics)
          Model  Accuracy Precision    Recall  F1_Score
1   Naive Bayes 0.4523810 0.4636054 0.4523810 0.4452991
2 Decision Tree 0.4523810 0.4886209 0.4523810 0.4468491
3           KNN 0.7142857 0.7387755 0.7142857 0.7045169
```

- **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**.

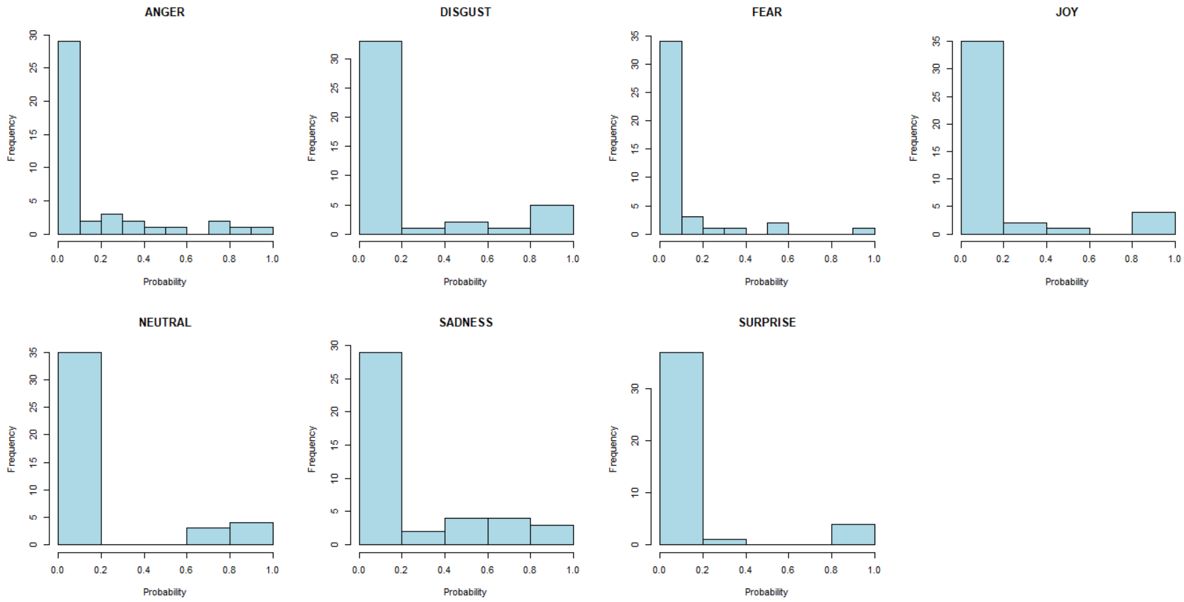


- Evaluated **Naïve Bayes, Decision Tree, and K-Nearest Neighbours (KNN) without applying PCA (raw data)**.

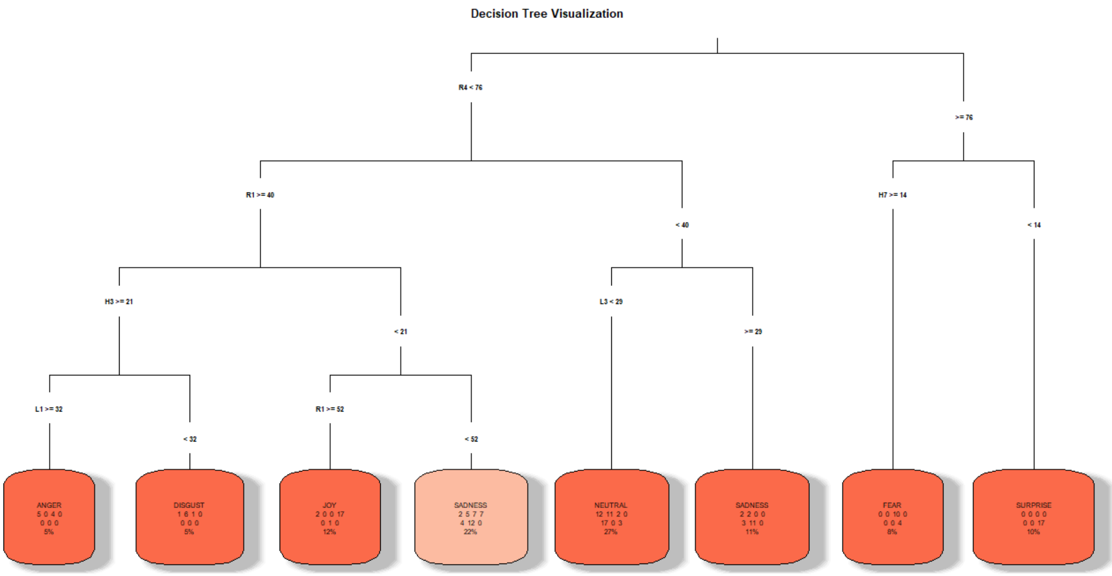
```
Performance Metrics:
> print(metrics_before)
          Model  Accuracy Precision    Recall  F1_Score
1   Naive Bayes 0.6666667 0.6969079 0.6666667 0.6537908
2 Decision Tree 0.3333333 0.3439876 0.3333333 0.3581338
3           KNN 0.9047619 0.9166667 0.9047619 0.9059988
```

- **Compared model performance without PCA**.

- **Plotted the probability distributions without PCA for each class in the Naïve Bayes model using histograms.**



- **Plotted and visualized the decision tree without PCA.**



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

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

Model	Without PCA	With PCA	Change
Naïve Bayes	66.7%	45.2%	-21.5%
Decision Tree	33.3%	45.2%	+11.9%
KNN	90.5%	71.4%	-19.1%

- **Naïve Bayes** experiences the most significant decline in accuracy, **dropping by 21.5%.** This suggests that the original features contained crucial information that was lost during dimensionality reduction. The **conditional independence** assumption of Naïve Bayes is more strongly violated after PCA, contributing to the sharp accuracy decline.

- **Decision Tree** profits from PCA**,** with **a 11.9% increase** in accuracy. This indicates that the raw features provided worst decision splits than the transformed principal components. The principal components created by PCA likely provided a cleaner and more effective feature space, while raw features were probably noisy.

- **KNN** remains the most resilient model**,** exhibiting **a 19.1% drop** in accuracy. This suggests that PCA had an impact on its classification ability and the original feature space was more informative. PCA probably caused a **distortion of distance metrics** that, combined with the **small dataset size,** resulted to the observed decrease.

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

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

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

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

- Based on the confusion matrix, **K-Nearest Neighbours (KNN) with PCA** performed best for **ANGER, DISGUST, JOY, NEUTRAL, SADNESS, and SURPRISE**, but struggled with **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      1       6    2   0       0       0        1
  FEAR         0       0    2   0       0       1        1
  JOY          0       0    0   5       0       0        0
  NEUTRAL      0       0    1   1       6       0        0
  SADNESS      2       0    1   0       0       4        0
  SURPRISE     0       0    0   0       0       0        4
```

- Based on the confusion matrix, **Naïve Bayes (NB) without PCA** performed best for **DISGUST, JOY, NEUTRAL, SADNESS, and SURPRISE**, but struggled with **ANGER, FEAR**.

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

- Based on the confusion matrix, **Decision Tree (DT) without PCA** performed best for **JOY, and SADNESS**, but struggled the most with **ANGER, DISGUST, FEAR, NEUTRAL, and SURPRISE**.

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

- Based on the confusion matrix, **K-Nearest Neighbours (KNN) without PCA** performed best for **ANGER, DISGUST, FEAR, JOY, NEUTRAL, SADNESS, and SURPRISE**. It demonstrated no difficulties distinguishing between classes.

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

- **What I Did:**

- 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):** Tuned **k**.

- **DBSCAN:** Tuned **eps** and **minPts** values.

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

```
> print(res_km$silhouette_avg)
[1] 0.3359572
> print(res_km$ARI)
[1] 0.08295301
```

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

```
> print(res_gmm$silhouette_avg)
[1] 0.3193624
> print(res_gmm$ARI)
[1] 0.06686721
```

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

```
> print(res_dbscan$silhouette_avg)
[1] 0.2305485
> print(res_dbscan$ARI)
[1] 0.01782748
```

- **What I Learned:**

- **K-Means:** With a silhouette score of ~0.34, K-Means achieved slightly better moderate separation than the other models. Its ARI of ~0.08 suggests a weak alignment with true classes.

- **GMM:** GMM's silhouette score of ~0.32 indicates a level of moderate separation nearly identical to K-Means. Its ARI of ~0.07 similarly confirms a weak correspondence with the true classes.

- **DBSCAN:** DBSCAN performed the worst with the lowest silhouette score (~0.23), indicating the poorest separation. Its minimal ARI of ~0.02 shows it had the most difficulty aligning with the true classes.

- Based on the confusion matrix, K-Means's clusters are highly mixed, indicating a failure to isolate specific expressions. For example, Cluster 2 contains a blend of **Fear (7)**, **Sadness (6)**, and **Surprise (10)**, while Cluster 6 mixes **Anger (10)** and **Joy (12)**. The model did not successfully separate the similar "negative" expressions, nor did it isolate any single emotion effectively. The number of clusters (7) appears to be an appropriate attempt to match the true classes, but the distribution within them is poor.

```
> print(res_km$confusion)
       Label
Cluster ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
      1     3       2    1   2       3       3        2
      2     3       0    7   0       1       6       10
      3     4       6    3   4       4       6        1
      4     1       4    6   6       5       0        0
      5     9      15    8   6      17       8        2
      6    10       3    5  12       0       6        0
      7     0       0    0   0       0       1       15
```

- Based on the confusion matrix, GMM shows a weak but notable ability to isolate some expressions. For instance, Cluster 2 is almost exclusively composed of **Surprise (15)**. However, most other clusters are highly mixed, suggesting that while the model found some distinct patterns, it also struggled to separate overlapping expressions. Cluster 1 is particularly mixed, blending **Anger (12)**, **Disgust (15)**, and **Neutral (17)**.

```
> print(res_gmm$confusion)
       Label
Cluster ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
      1    12      15   15   6      17      11        3
      2     0       0    0   0       0       1       15
      3     3       3    3   4       3       3        1
      4     6       7    7   9       6       3        2
      5     5       1    5  11       0       3        0
      6     1       3    0   0       0       6        9
      7     3       1    0   0       4       3        0
```

- Based on the confusion matrix, DBSCAN's performance is poor, creating only 3 clusters, which is too few for the 7 expressions. Cluster 1 acts as a large, undifferentiated group containing the majority of all emotions. The model does show a weak tendency to isolate **Surprise** in Cluster 2, but this is a very limited success. There is no clear evidence that the model is detecting variations within expressions, rather, it appears to be failing to differentiate them at all.

```
> print(res_dbscan$confusion)
       Label
Cluster ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
      0     0       0    0   4       1       1       10
      1    30      30   30  26      29      29       14
      2     0       0    0   0       0       0        6
```

- 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.