• Project Flowchart:

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 and Preparation:

- **Exploring the Facial Expression Dataset:**

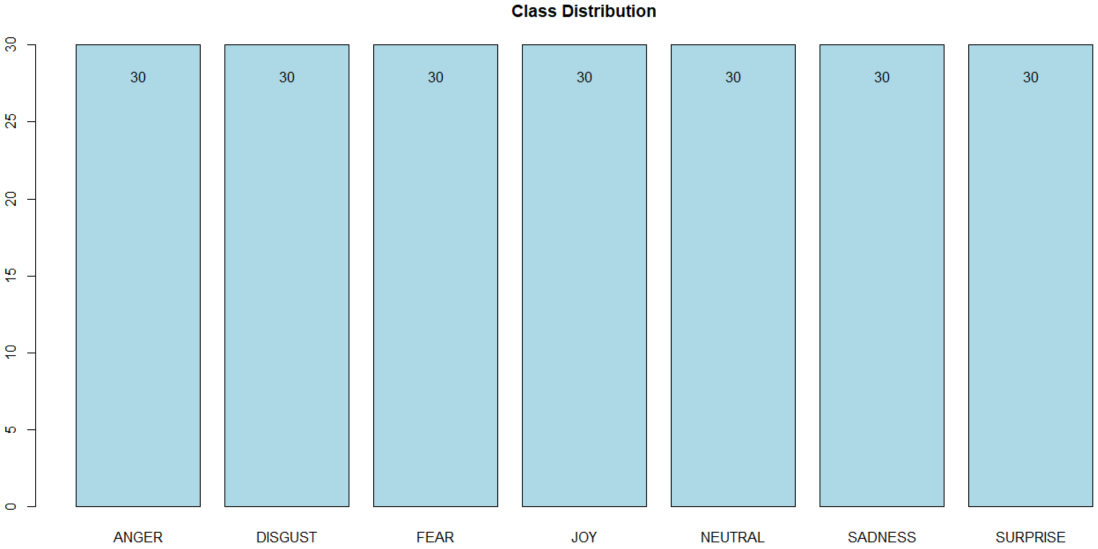
The dataset consists of 210 instances from the Cohn-Kanade database, each with 25 facial measurement features and a class label representing one of seven facial expressions.

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)

- **What I Did:**

- Checked **class distribution** using a bar plot to visualize how expressions are distributed. This verified that the dataset is perfectly balanced



- Extracted the **features data**.

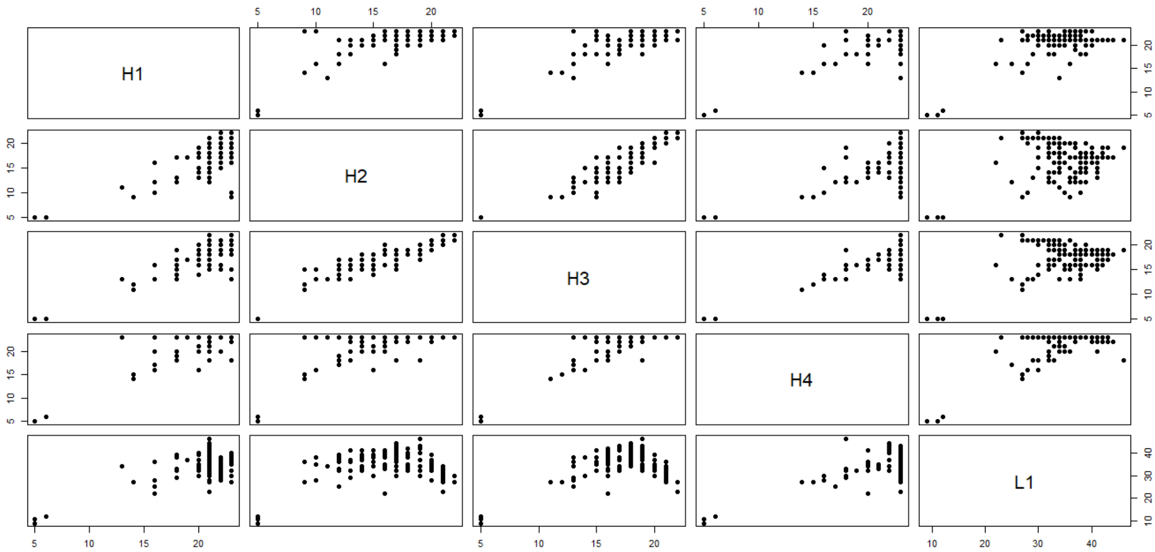
	H1	H2	H3	H4	L1	H5	H6	H7	H8	L2	H9	H10	W1	H11	H12	W2	H13	H14	H15	W3	L3	R1	R2	R3	R4
1	22	17	19	23	38	20	18	16	23	25	15	19	17	14	17	10	17	16	23	9	24	33	23	14	73
2	21	15	18	23	39	24	17	14	23	27	14	19	17	13	15	10	17	15	20	9	21	38	15	25	65
3	21	16	18	23	38	17	17	14	15	31	15	17	17	13	16	10	17	15	22	9	30	36	22	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**.**

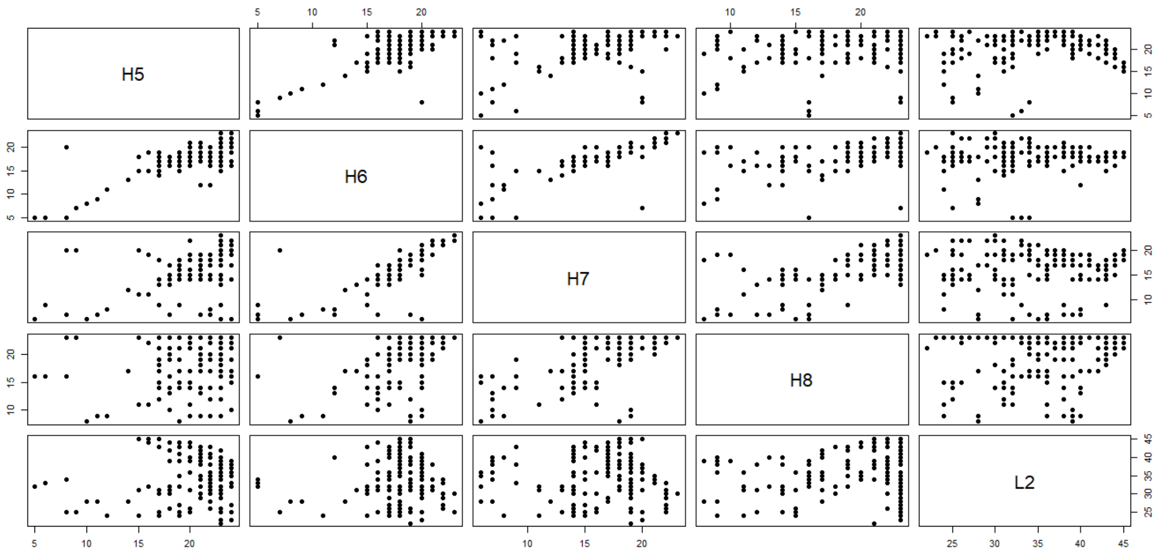
```
> # Split the dataset (80% train, 20% test)
> split_result <- split_process(features_data, seed = GLOBAL_SEED)

   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.

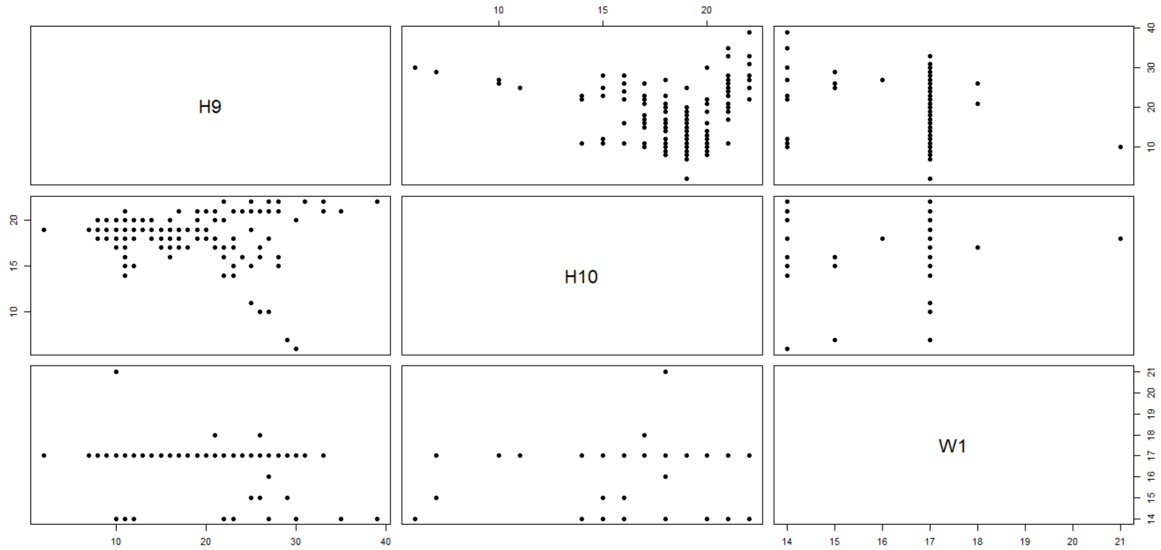
Pairwise Scatter Plot for the Left_Eyebrow



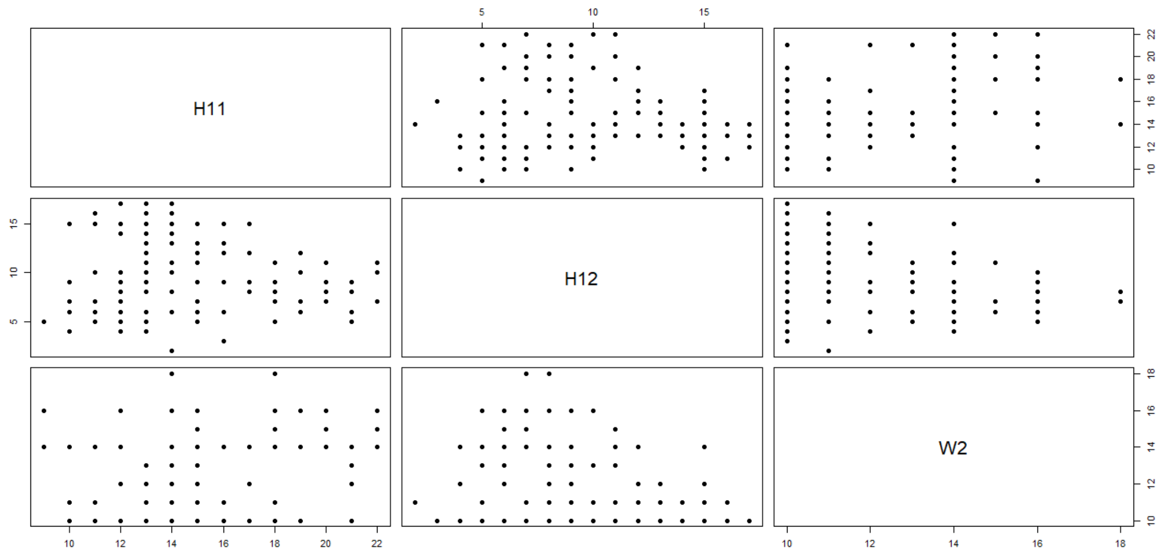
Pairwise Scatter Plot for the Right_Eyebrow



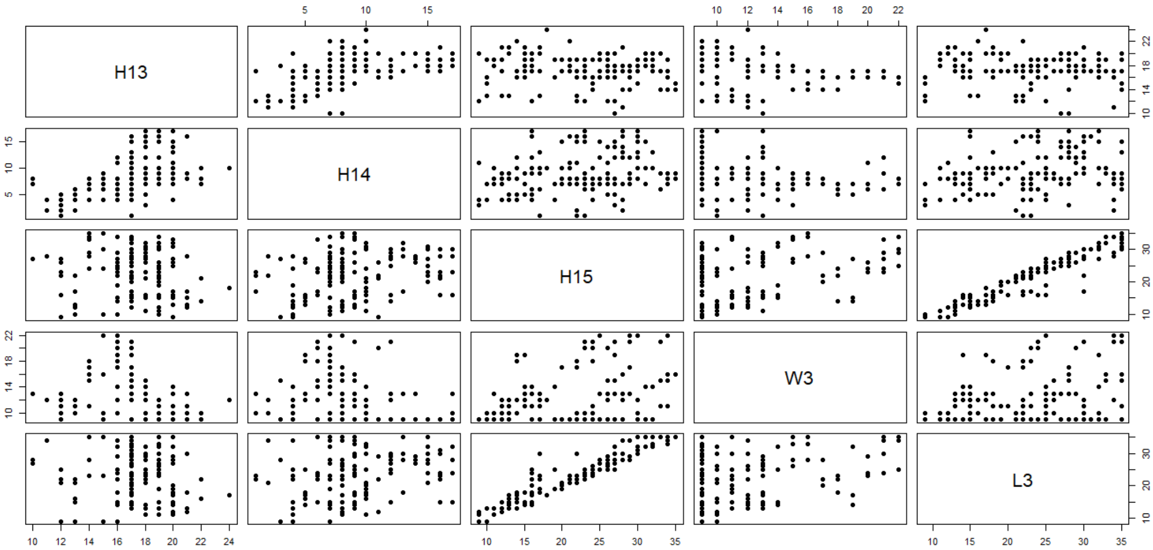
Pairwise Scatter Plot for the Left_Eye



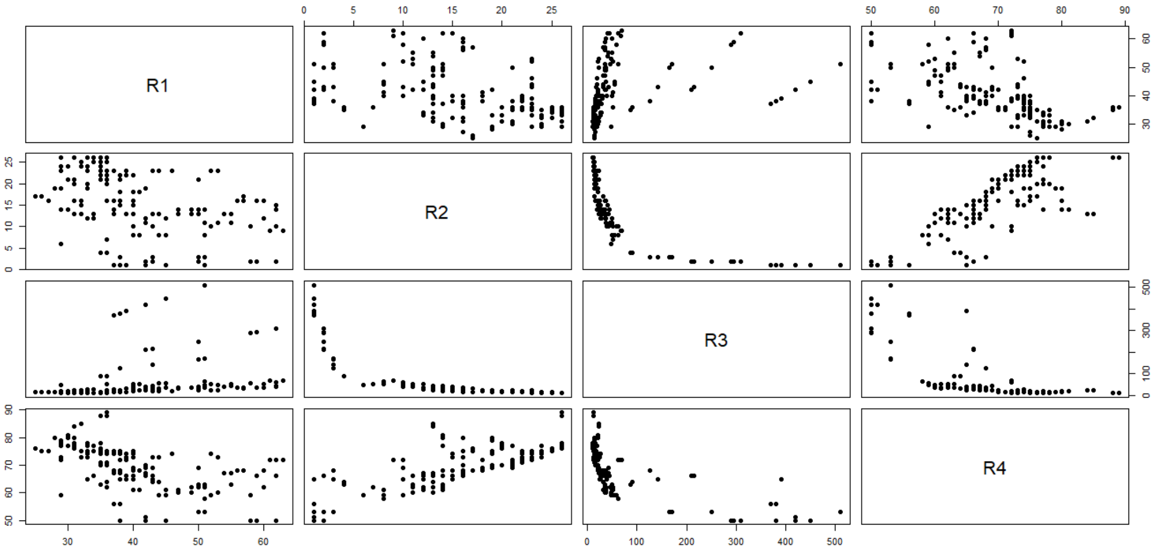
Pairwise Scatter Plot for the Right_Eye



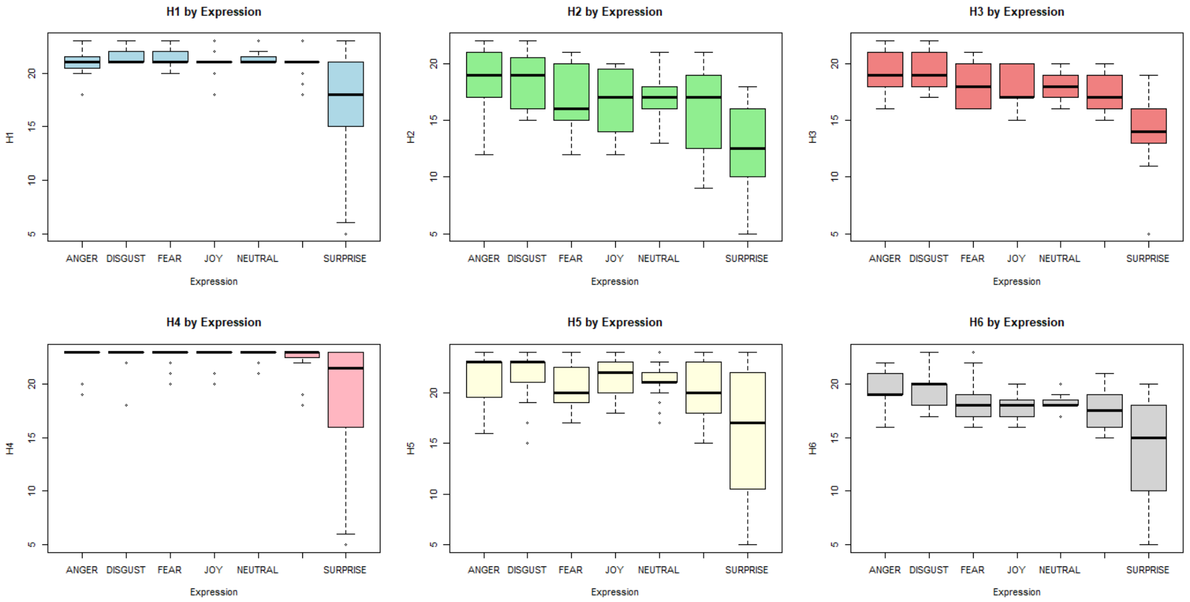
Pairwise Scatter Plot for the Mouth

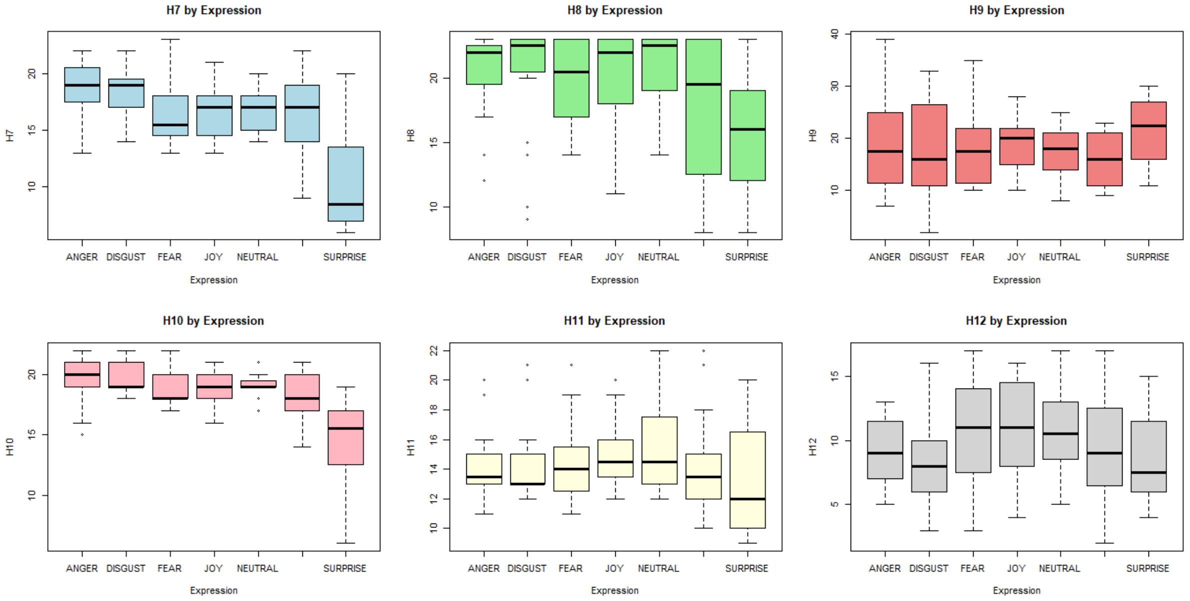


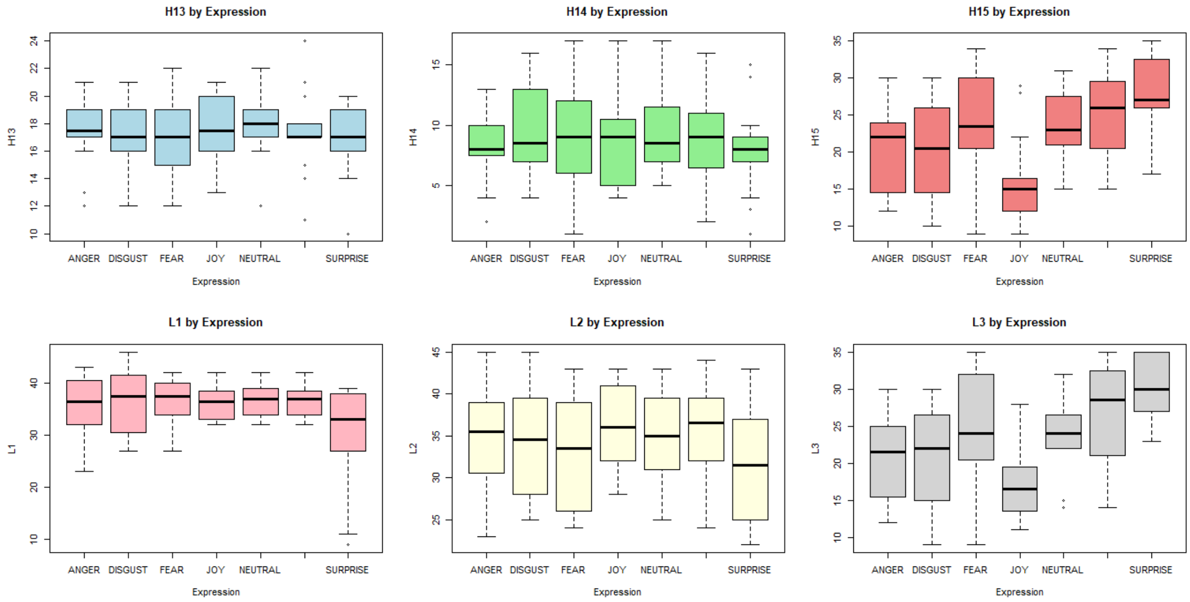
Pairwise Scatter Plot for the Relationships

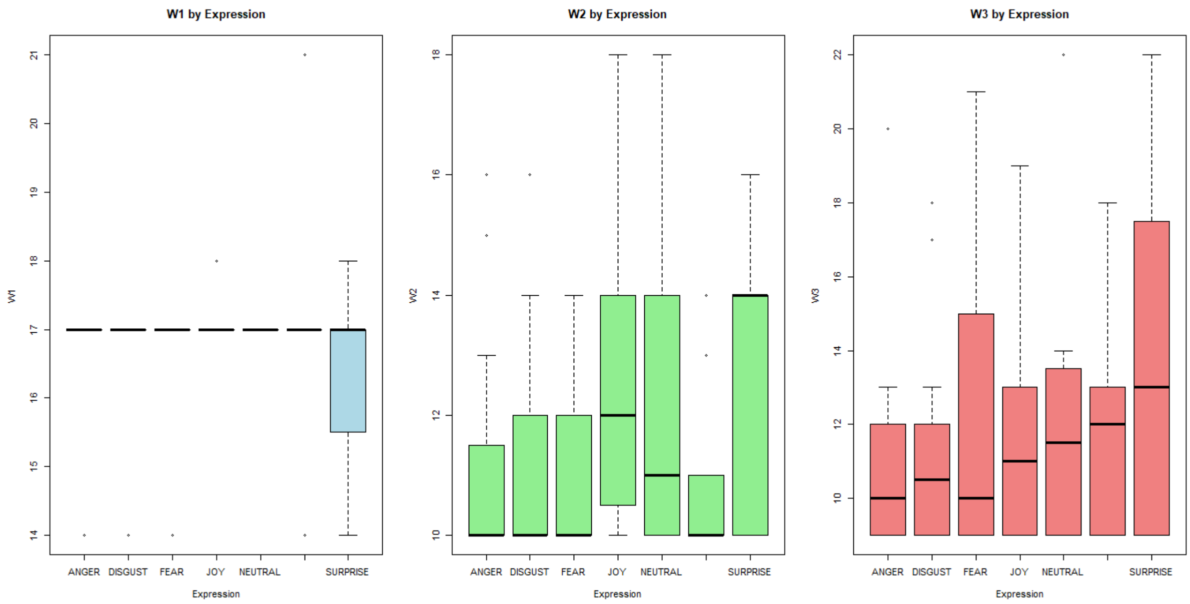


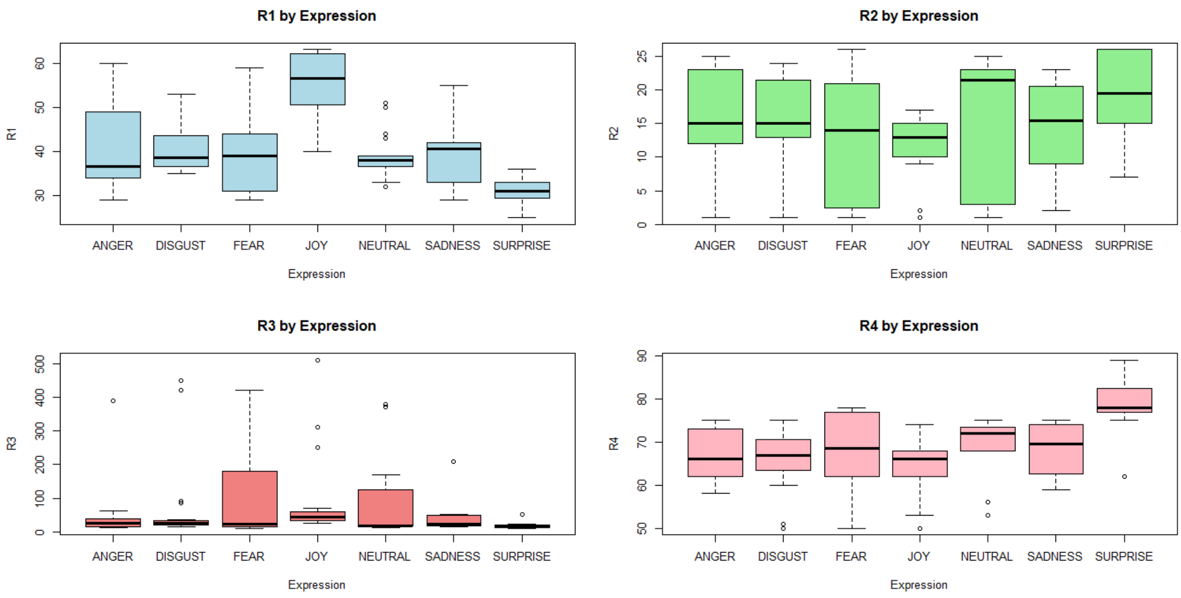
- Plotted **boxplots** of each feature grouped by **Expression** to analyse how facial features vary across emotions.






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


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



3. **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)**.

- For each model, I followed two distinct training paths within the cross-validation loops. **Raw Features with Recursive Feature Elimination (RFE)** and **PCA-Transformed Features with RFE,** to automatically select the best features

- **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**.

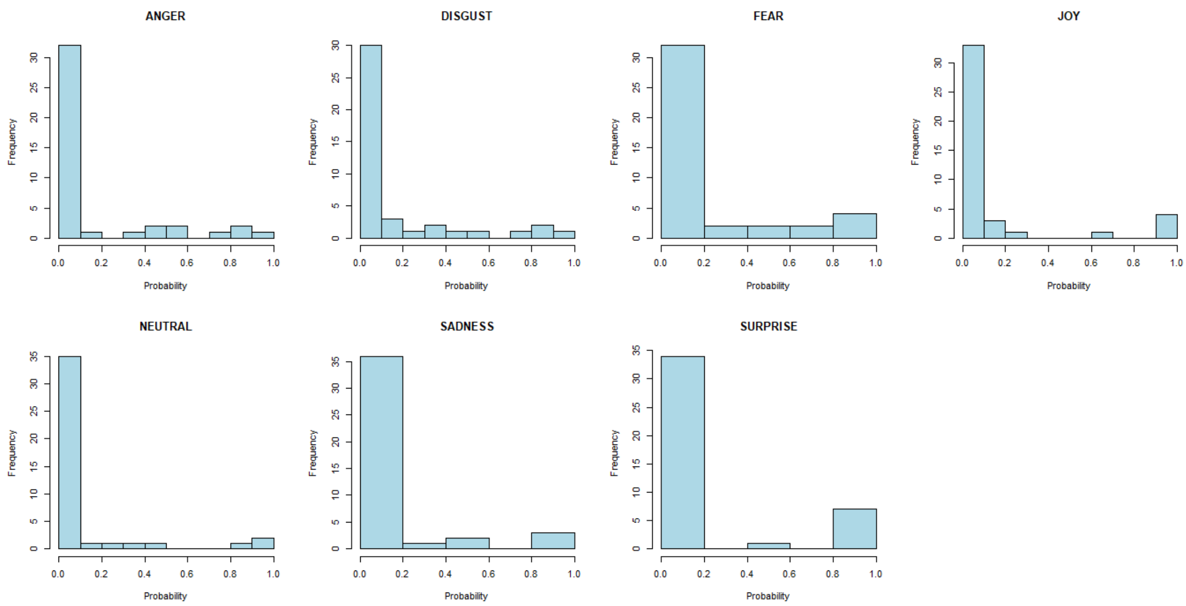
- **K-Nearest Neighbours (KNN):** Tuned **k-value (kmax)**, distance metric (**Manhattan vs. Euclidean**), and **kernel function** for optimal performance.

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

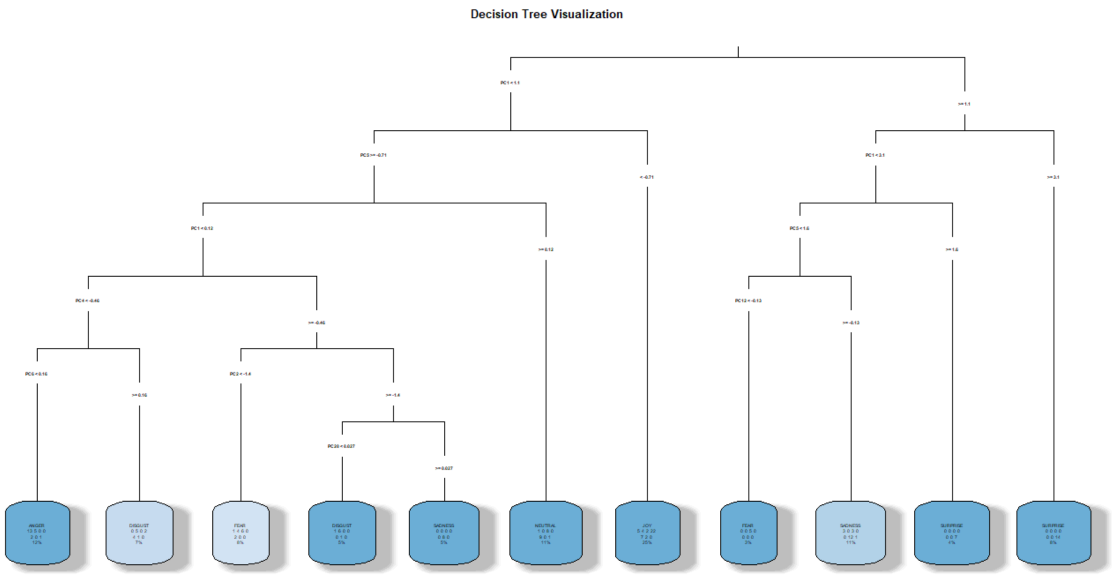
```
Performance Metrics:
> print(metrics)
          Model  Accuracy Precision    Recall  F1_Score
1   Naive Bayes 0.5952381 0.6270408 0.5952381 0.5995005
2 Decision Tree 0.3809524 0.4141156 0.3809524 0.3636054
3           KNN 0.7857143 0.8103741 0.7857143 0.7894201
```

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

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



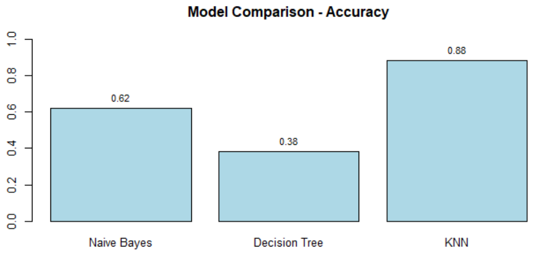
- **Plotted and visualized the decision tree**.

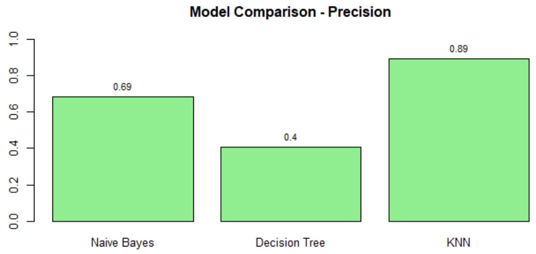


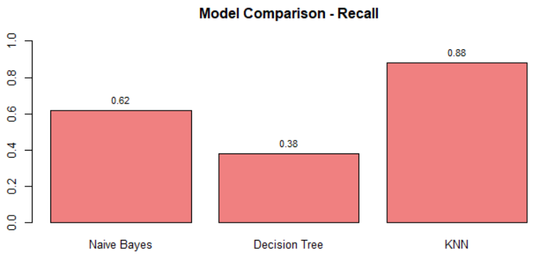
- Evaluated **Naïve Bayes, Decision Tree, and K-Nearest Neighbours (KNN) without applying PCA (raw data)**.

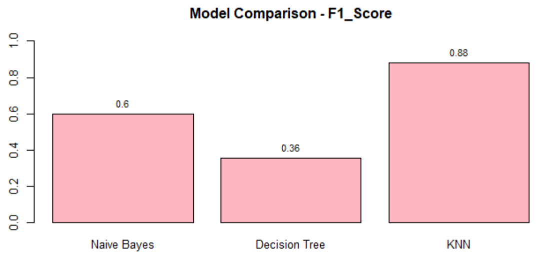
```
Performance Metrics:
> print(metrics_before)
          Model  Accuracy Precision    Recall  F1_Score
1   Naive Bayes 0.6190476 0.6850649 0.6190476 0.5973771
2 Decision Tree 0.3809524 0.4047619 0.3809524 0.3560606
3           KNN 0.8809524 0.8911565 0.8809524 0.8824509
```

- **Compared model performance without PCA**.

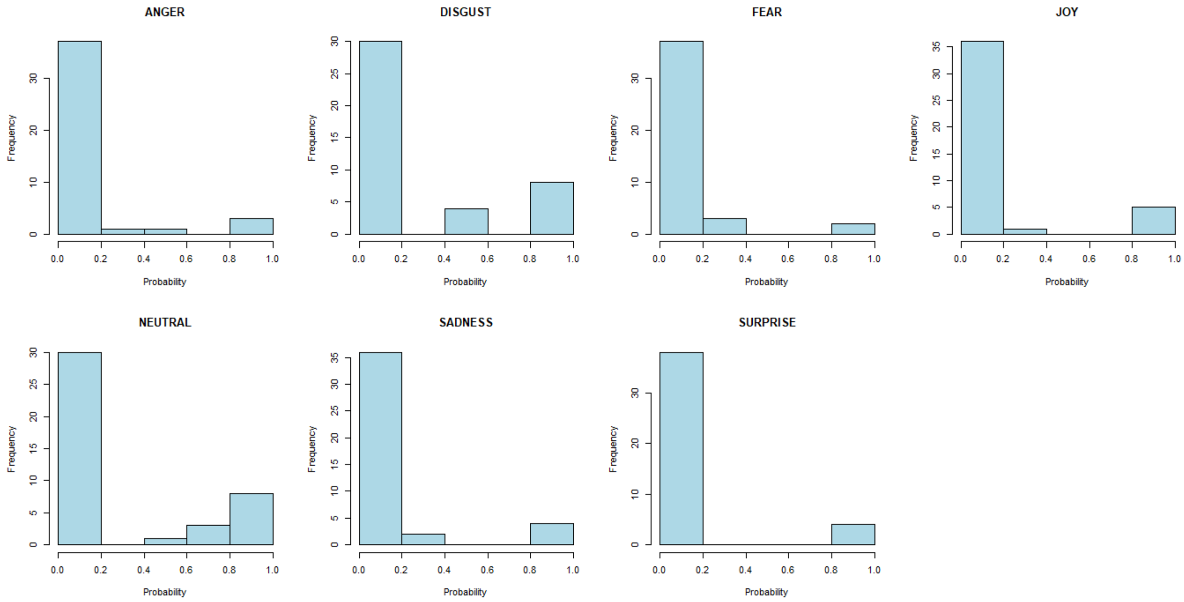




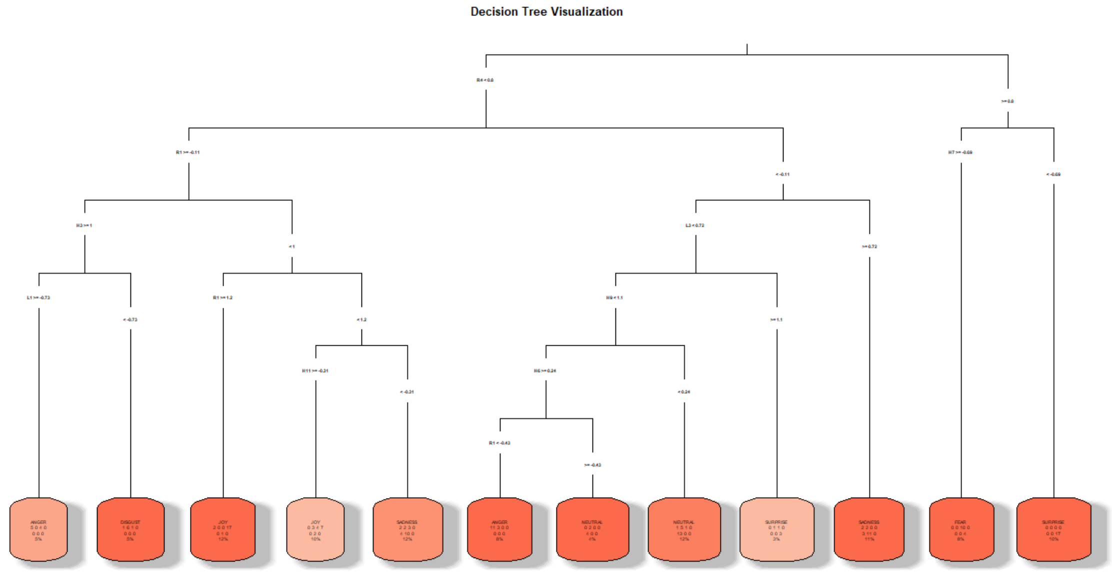




- **Plotted the probability distributions without PCA for each class in the Naïve Bayes model using histograms.**



- **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	61.9%	59.5%	▼ -2.4%
Decision Tree	38.1%	38.1%	● 0.0%
KNN	88.1%	78.6%	▼ -9.5%

- **Naïve Bayes** experiences a small decline in accuracy, **dropping by 2.4%.** This suggests that PCA subtly violated the conditional independence assumption. The original facial measurements likely contained distinct, independent signals for each expression that were partially muddled by the PCA transformation, leading to the small drop in performance.

- **Decision Tree** model's performance **remains unchanged**. This indicates that the principal components derived from the facial measurements neither improved nor harmed the model's ability to classify the data. The model's low overall accuracy suggests that neither the raw facial measurements nor the principal components are easily separable for this type of model.

- **KNN** is the most affected model**, showing a significant 9.5% drop in accuracy**. PCA, while preserving overall variance, can distort the local relationships between data points. This distortion caused the distances between points of the same class to increase and those of different classes to decrease, leading to the large drop in classification accuracy.

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

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

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

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

- Based on the confusion matrix, **K-Nearest Neighbours (KNN) with PCA** performed best for **ANGER, DISGUST, FEAR, JOY, NEUTRAL, SADNESS, and SURPRISE**, and struggled with none of the expressions.

```
Confusion Matrix for KNN:
> print(knn_cm$table)
          Reference
Prediction ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
  ANGER        5       0    0   0       0       0        1
  DISGUST      0       5    1   0       0       1        1
  FEAR         0       0    5   0       1       0        0
  JOY          0       0    0   5       0       0        0
  NEUTRAL      0       1    0   0       5       1        0
  SADNESS      1       0    0   1       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, and NEUTRAL**, but struggled with **ANGER, FEAR, SADNESS, and SURPRISE**.

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

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

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

- 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       0       0        0
  DISGUST      0       6    1   0       0       0        0
  FEAR         0       0    5   0       0       2        0
  JOY          0       0    0   5       0       0        0
  NEUTRAL      0       0    0   0       6       0        0
  SADNESS      1       0    0   1       0       4        0
  SURPRISE     0       0    0   0       0       0        6
```

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

4. 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):** Tuned **k**.
- **DBSCAN:** Tuned **eps** and **minPts** values.
- Evaluated **K-Means** clusters using **Silhouette Score** and **ARI**.

```
> print(res_km$silhouette_avg)
[1] 0.2976775
> print(res_km$ARI)
[1] 0.07586069
```

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

```
> print(res_gmm$silhouette_avg)
[1] 0.1499203
> print(res_gmm$ARI)
[1] 0.06439163
```

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

```
> print(res_dbscan$silhouette_avg)
[1] 0.542801
> print(res_dbscan$ARI)
[1] 0.06161083
```

- **What I Learned:**

- **K-Means:** With a silhouette score of approximately **0.298**, K-Means shows a **moderate level of cluster separation**. Its Adjusted Rand Index (ARI) of around **0.076** indicates a **weak alignment with the true classes**. This suggests that while K-Means found moderately distinct clusters, they don't strongly correspond to the real-world categories of the data.

- **GMM:** The Gaussian Mixture Model's silhouette score of about **0.150** shows a **poor level of cluster separation** and is the lowest of the three models. Its ARI of approximately **0.064** confirms a **very weak correspondence with the true classes**, performing the worst in this regard.

- **DBSCAN:** DBSCAN performed the **best** in terms of cluster separation, with the highest silhouette score of approximately **0.543**. This score indicates that the clusters it formed were **well-separated**. However, its ARI of around **0.062** shows that despite finding well-defined clusters, they had the **weakest alignment with the true classes**, performing slightly worse than GMM in this aspect.

- Based on the confusion matrix, K-Means created **7 clusters**, which is an appropriate number given the 7 distinct facial expressions. However, the clusters are **highly mixed**, indicating a failure to isolate specific expressions. For example, **Cluster 6** contains a blend of **Anger (11), Disgust (11), and Joy (14)**. Similarly, **Cluster 4** is a mixture of **Fear (13), Neutral (8), and Sadness (6)**. The model did not successfully separate the expressions, and the distribution within the clusters is poor.

```
> print(res_km$confusion)
       Label
Cluster ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
      1     0       0    0   0       0       0        3
      2     0       0    0   0       0       0       11
      3     2       2    3   3       2       4        3
      4     2       3   13   0       8       6        7
      5     6       5    4   4       5       3        0
      6    11      11    2  14       6       9        0
      7     3       3    2   3       3       2        0
```

- Based on the confusion matrix, GMM created **7 clusters**, which aligns with the number of true classes. The clusters are highly mixed, indicating a failure to isolate specific expressions. For example, **Cluster 3** is a blend of **Fear (7), Sadness (8), and Surprise (15)**. **Cluster 4** is predominantly **Joy (12)**, but also includes a significant number of **Neutral (5)** expressions. The model did not successfully separate the expressions and the distribution within the clusters is poor.

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

- Based on the confusion matrix, DBSCAN created **6 clusters** (numbered 0-5), which is too few for the 7 expressions. The model's performance is poor, with **Cluster 0** acting as a large, undifferentiated group containing the majority of all emotions, particularly **Joy (21), Surprise (22), and Sadness (16)**. This indicates that the model is failing to differentiate the expressions, lumping many of them into a single, massive cluster. There is no clear evidence that DBSCAN is detecting variations within expressions.

```
> print(res_dbscan$confusion)
       Label
Cluster ANGER DISGUST FEAR JOY NEUTRAL SADNESS SURPRISE
      0     6       6   12  21       6      16       22
      1    11      11    9   3      13       6        2
      2     1       2    3   0       2       0        0
      3     3       2    0   0       0       0        0
      4     0       0    0   0       3       2        0
      5     3       3    0   0       0       0        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.