**Data Collection and Preprocessing Phase**

|  |  |
| --- | --- |
| Date | 18 June 2025 |
| Team ID | SWTID1749712812 |
| Project Title | Unlocking Silent Signals: Decoding Body Language with Mediapipe |
| Maximum Marks | 6 Marks |

**Data Exploration and Preprocessing Report**

Dataset variables will be statistically analyzed to identify patterns and outliers, with Python employed for preprocessing tasks like normalization and feature engineering. Data cleaning will address missing values and outliers, ensuring quality for subsequent analysis and modeling, and forming a strong foundation for insights and predictions.

|  |  |
| --- | --- |
| **Section** | **Description** |
| Data Overview | Dimension: 2068 rows × 67 columns  Descriptive statistics: |
| Univariate Analysis |  |

|  |  |
| --- | --- |
| Bivariate Analysis |  |
| Multivariate Analysis |  |

|  |  |
| --- | --- |
| Outliers and Anomalies | - |
| **Data Preprocessing Code Screenshots** | |
| Loading Data | # Load body language dataset extracted using Mediapipe  import pandas as pd  data = pd.read\_csv('/content/drive/MyDrive/Dataset/body\_language\_data.csv')  data.head()  **Output:**  nose\_x nose\_y left\_eye\_x left\_eye\_y ... right\_ankle\_x right\_ankle\_y class  0 0.5213 0.4941 0.4982 0.4815 ... 0.5582 0.9892 happy  1 0.5198 0.4967 0.4969 0.4831 ... 0.5539 0.9875 sad  2 0.5209 0.4930 0.4976 0.4809 ... 0.5590 0.9881 victory  3 0.5221 0.4927 0.4995 0.4813 ... 0.5601 0.9864 fighting  4 0.5190 0.4952 0.4960 0.4825 ... 0.5556 0.9853 happy |
| Handling Missing Data | # Check for missing values  data.isnull().sum()  **Output:**  nose\_x 0  nose\_y 0  left\_eye\_x 0  ...  right\_ankle\_y 0  class 0  dtype: int64  # Drop missing values (if any)  data.dropna(inplace=True) |
| Data Transformation | # Normalize landmark features using StandardScaler  from sklearn.preprocessing import StandardScaler  X = data.drop(columns=['class'])  y = data['class']  scaler = StandardScaler()  X\_scaled = scaler.fit\_transform(X)  # Normalize landmark features using StandardScaler  from sklearn.preprocessing import StandardScaler  X = data.drop(columns=['class'])  y = data['class']  scaler = StandardScaler()  X\_scaled = scaler.fit\_transform(X)  **Output (first 3 rows of transformed features):**  array([[-0.054, 0.012, -0.057, ..., 0.132, -0.089],  [-0.060, 0.016, -0.063, ..., 0.125, -0.091],  [-0.058, 0.010, -0.059, ..., 0.130, -0.090]]) |
| Feature Engineering | import numpy as np  # Distance between left and right wrists  data['wrist\_distance'] = np.sqrt(  (data['left\_wrist\_x'] - data['right\_wrist\_x'])\*\*2 +  (data['left\_wrist\_y'] - data['right\_wrist\_y'])\*\*2  )  data[['left\_wrist\_x', 'right\_wrist\_x', 'left\_wrist\_y', 'right\_wrist\_y', 'wrist\_distance']].head()  **Output:**  left\_wrist\_x right\_wrist\_x left\_wrist\_y right\_wrist\_y wrist\_distance  0 0.4510 0.5591 0.7812 0.7892 0.109  1 0.4533 0.5567 0.7840 0.7889 0.104  2 0.4525 0.5589 0.7822 0.7903 0.108  3 0.4509 0.5600 0.7831 0.7896 0.110  4 0.4515 0.5578 0.7844 0.7882 0.106 |
| Save Processed Data | # Save the processed and scaled dataset  processed\_df = pd.DataFrame(X\_scaled, columns=X.columns)  processed\_df['class'] = y.values  processed\_df.to\_csv('processed\_body\_language\_data.csv', index=False)  **Output (file created):**  'processed\_body\_language\_data.csv' saved in working directory. |