WESAD database

Stress Analysis Using ECG, EDA, and EMG Signals

Team HARmonizers
Kanishk Goel- 2021325
Milind Jain- 2021165

TABLE OF CONTENT

PROBLEM STATEMENT

- "Understanding the physiological response to stress using non-invasive biomedical signals."
- Importance: "Timely detection and quantification of stress can lead to effective interventions, improving mental and physical well-being."

01.

02.

DATA PREPROCESSING

1. Data Cleaning
   - Removal of irrelevant data points.
   - Handling and imputation of missing values.
   - Rectification of inconsistencies in the dataset.
2. Normalization (Handling Noisy Data)
   - Applied Min-Max scaling to transform the data into the range [0, 1].
   - Reduced the effect of outliers and ensured uniformity in the data distribution.

```
# Drop NaN values
ecg_raw = ecg_raw[~np.isnan(ecg_raw)]
eda_raw = eda_raw[~np.isnan(eda_raw)]
emg_raw = emg_raw[~np.isnan(emg_raw)]
```

```
def normalize_data(data):
    """Normalize data using Min-Max scaling."""
    return (data - np.min(data)) / (np.max(data) - np.min(data))
```

3. Feature Selection

- Identified and retained the most informative features: ECG, EDA, and EMG.
- Excluded other features to focus on primary physiological signals related to stress.

4. Data Splitting

- Divided the dataset into training and evaluation subsets.
- Ensured the model has unseen data for validation and performance assessment.

```
# Load data from .pkl file
pkl_path = os.path.join(subject, subject + '.pkl')
with open(pkl_path, 'rb') as file:
    data = pickle.load(file, encoding='latin1')

# Extracting chest data
chest_data = data['signal']['chest']

ecg_raw = chest_data['ECG']
eda_raw = chest_data['EDA']
emg_raw = chest_data['EMG']
```

03.

CHALLENGES

PROBLEM

Complex Structure of WESAD Dataset.

Noisy Data and Signal Interference.

SOLUTION

Reviewed dataset documentation and extracted relevant signals systematically.

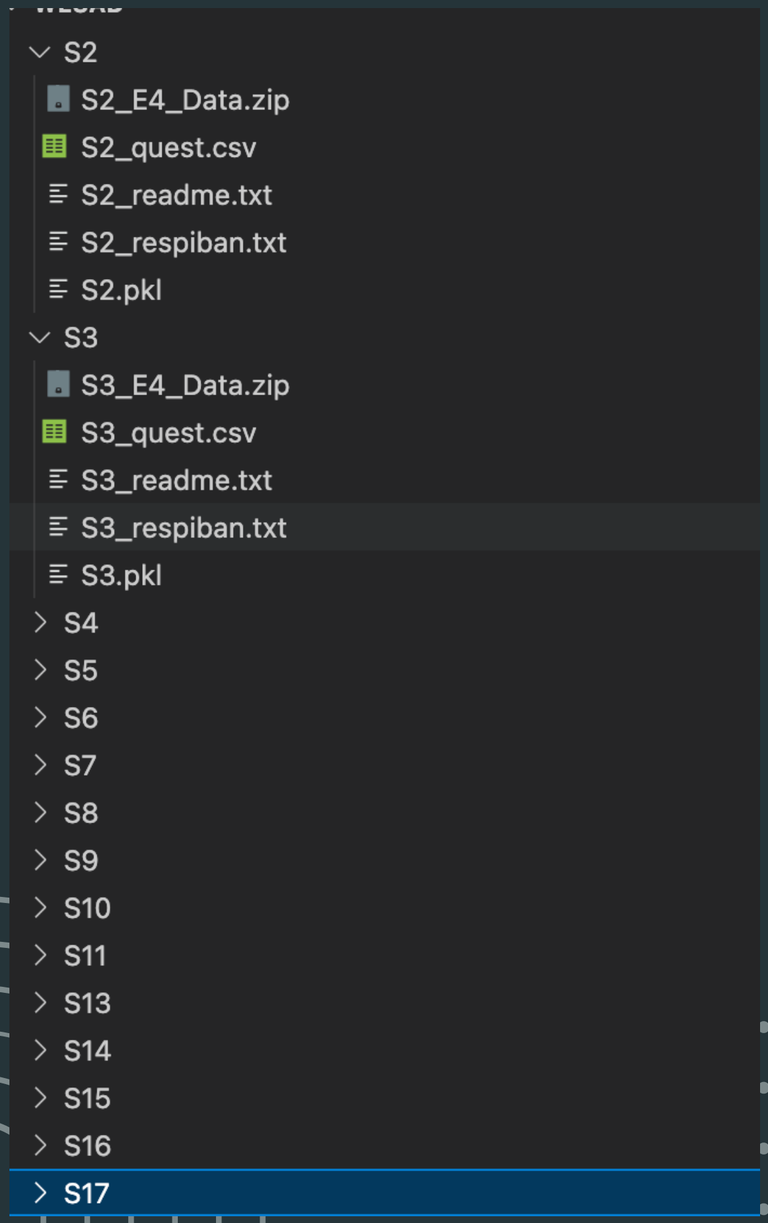
Implemented signal processing techniques, including filtering, to reduce noise and enhance the clarity of physiological signals.

PROBLEM

Combining Multiple Data Streams

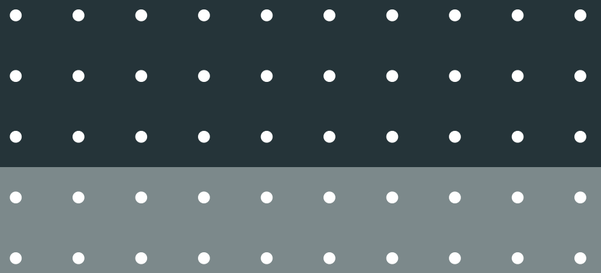
SOLUTION

Developed a systematic pipeline for data concatenation and alignment, preserving the temporal integrity of the signals.



04.

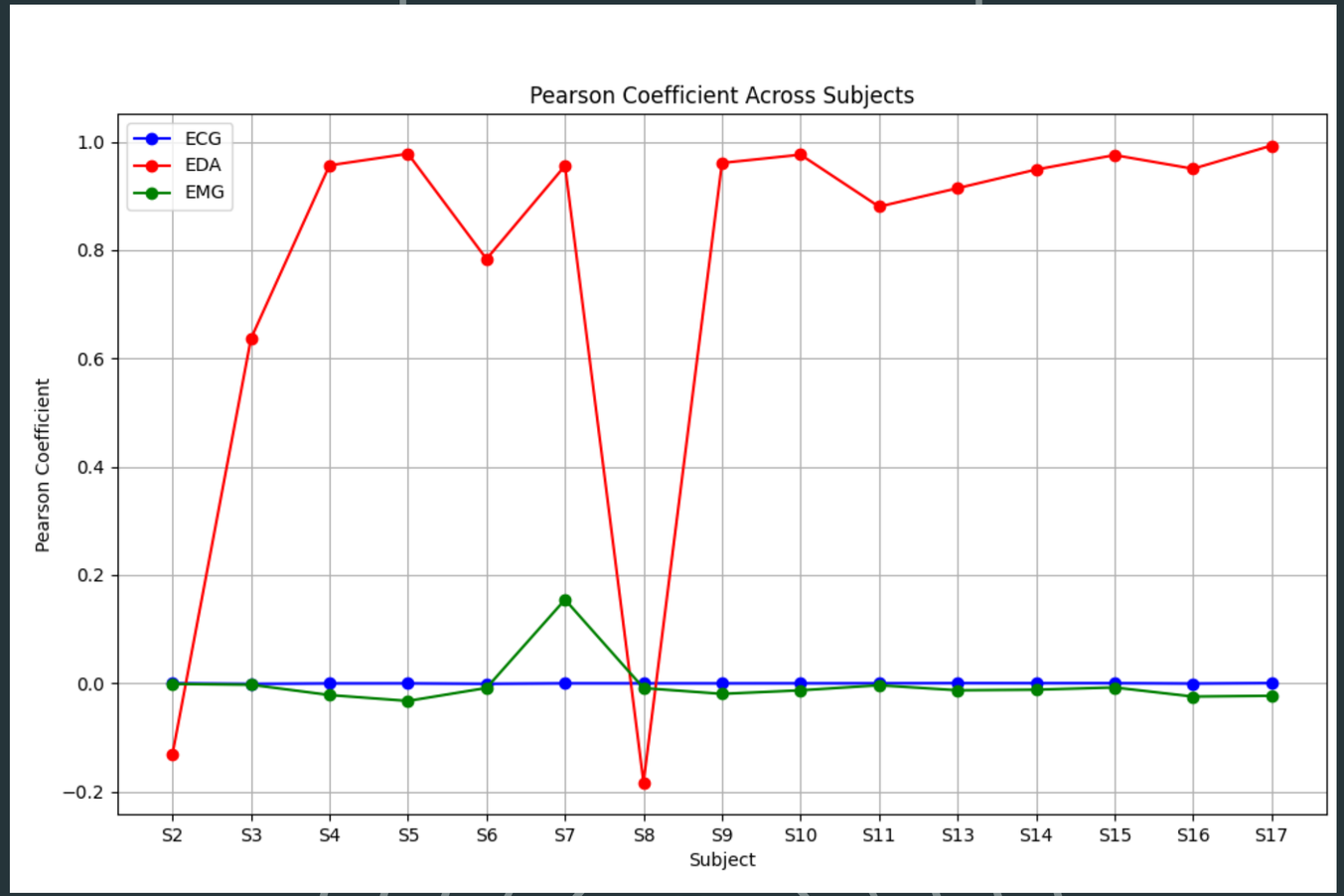
HYPOTHESIS TESTING



PEARSON COEFFICIENT

We decided to first check the pearson correlation coefficient between baseline and the stress level for ECG, EDA and EMG. Here is the plot for the same for all the subjects.

Here, we observed that the EDA and stress levels were highly correlated(positive linear relationship)



HYPOTHESIS

- "Is there a significant difference in the means of biomedical signals (ECG, EDA, EMG) between baseline and stress conditions?"
- "Ho (Null Hypothesis): There isn't a significant difference between the means."
- "H1 (Alternative Hypothesis): There is a significant difference between the means."

PROCEDURE

Data Segregation:

- Segregated the data based on labels, distinguishing between baseline and stress conditions.

Sampling:

- Randomly sampled 10% of the data for each physiological signal under both conditions.

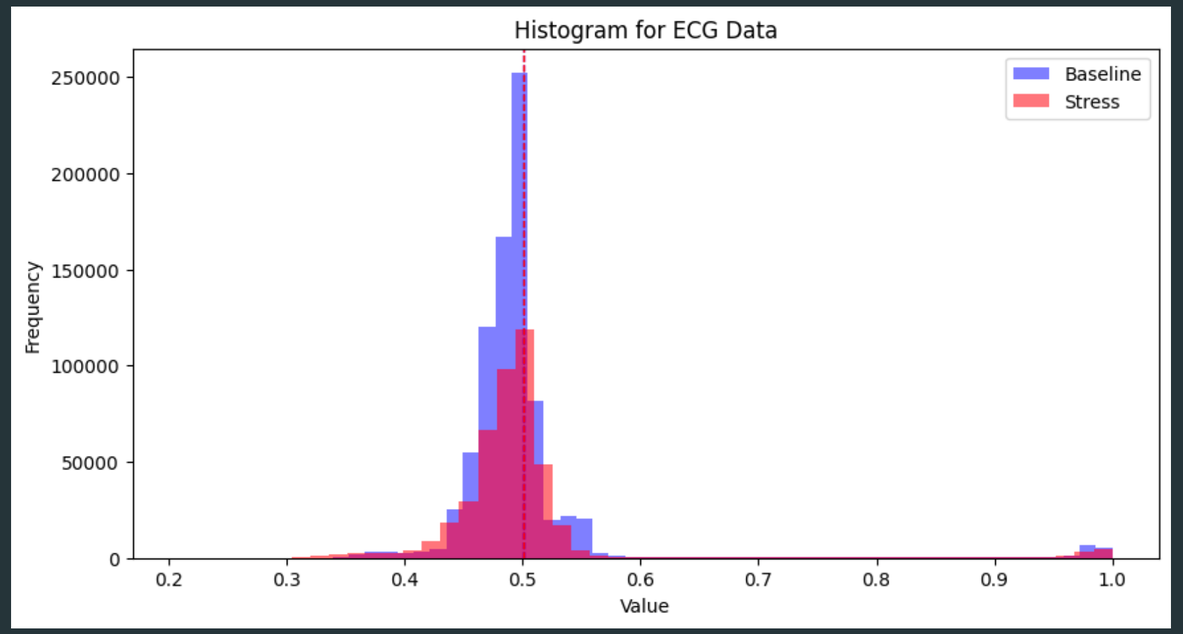
Z-test Application:

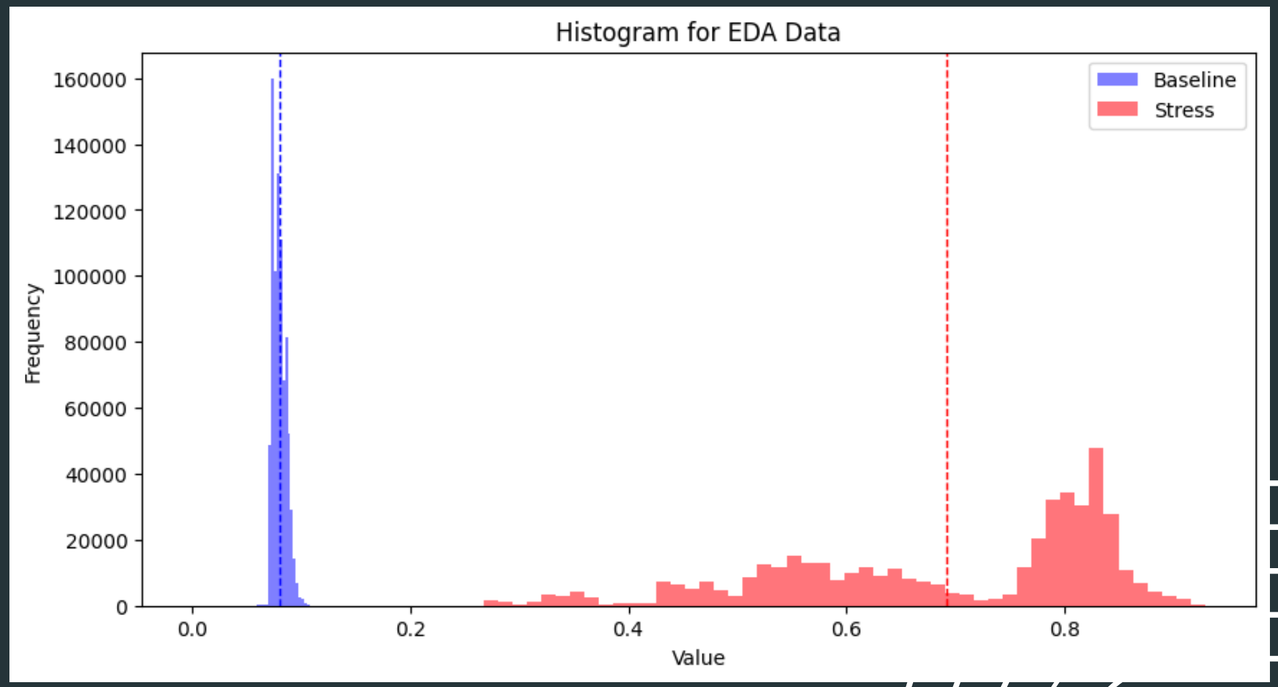
- Applied the Z-test to compare the means of the sampled data for baseline and stress conditions.

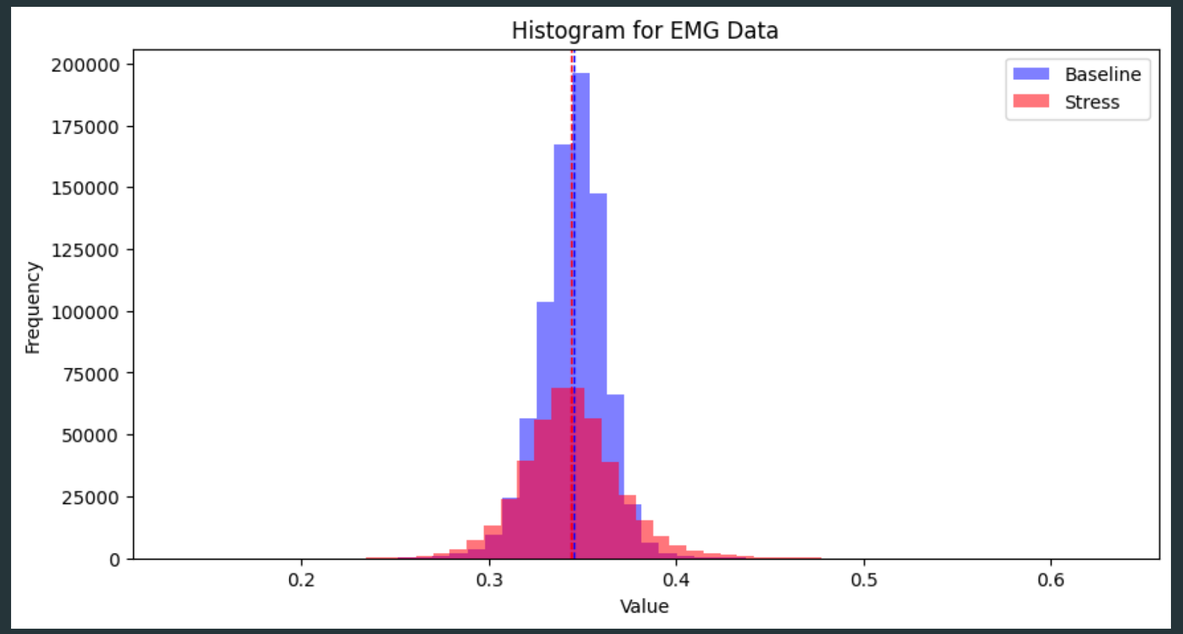
Decision Criteria:

- Used a significance level (α) of 0.05.
- If p-value < α, reject the null hypothesis.
- If p-value > α, fail to reject the null hypothesis.

VISUALISATION







HYPOTHESIS RESULTS AND VALIDATION

For nearly all subjects, a significant difference in means was observed between baseline and stress conditions, both in sampled and entire dataset. This aligns with the strong correlation seen between EDA and stress.

For ECG and EMG, no significant correlation was observed.

```
Subject: S17
ECG-Baseline vs Stress Correlation: 0.0002656950030589104
EDA-Baseline vs Stress Correlation: 0.9930300191900447
EMG-Baseline vs Stress Correlation: -0.02310494688339912

Validation for ECG data:

Hypothesis Test on 10.0% of ECG data:
Z-score: 0.4353304115961519, p-value: 0.663322613490565
Fail to reject the null hypothesis for ECG on the sample: There isn't a signific
ant difference between the means.
Validation on entire dataset contradicts this, as means of ECG Baseline and Stre
ss are different.

Validation for EDA data:

Hypothesis Test on 10.0% of EDA data:
Z-score: -3490.9355480744925, p-value: 0.0
Reject the null hypothesis for EDA on the sample: There is a significant differe
nce between the means.
Validation on entire dataset supports this, as means of EDA Baseline and Stress
are different.

Validation for EMG data:

Hypothesis Test on 10.0% of EMG data:
Z-score: 11.050213504346823, p-value: 0.0
Reject the null hypothesis for EMG on the sample: There is a significant differe
nce between the means.
Validation on entire dataset supports this, as means of EMG Baseline and Stress
are different.
--------------------------
```

For EDA, findings remained consistent for nearly all subjects, validating the results.

05.

CONCLUSION

- Successfully pre-processed and analyzed the complex WESAD dataset.
- Identified significant correlations and patterns in ECG, EDA, and EMG signals related to stress.
- We found a strong positive correlation between stress levels and EDA signals.
- Validated hypotheses revealing differences in baseline and stress states across subjects.

MOVING FORWARD

01. *Recalling Problem Statement*

02. *Data preprocessing*

03. *ML Models*

04. *Random Forest, SVM and Logistic Regression*

05. *Comparison of the different Ml models*

PROBLEM STATEMENT

This project aims to leverage the WESAD (WEarable Stress and Affect Detection) dataset to develop personalized machine learning models that accurately predict stress levels for individuals. By analyzing physiological and demographic data collected through wearable sensors, the project seeks to understand the variance in stress responses among different subjects. The goal is to create a robust model for each individual that effectively distinguishes between stress and non-stress states, potentially contributing to personalized stress management solutions.

FURTHER DATA PREPROCESSING

ML MODELS

- Random Forest: Achieved unexpectedly high accuracy, suggesting potential overfitting or data leakage. Our dataset was limited in the number of data points for each subject.
- Support Vector Machine (SVM): Lower accuracy compared to Random forrest, indicating a more nuanced understanding of the data, potentially reducing overfitting.
- Logistic Regression: Logistic Regression outperformed other models due to its suitability for smaller datasets, where complex models like Random Forest tend to overfit.

RANDOM FOREST

```
Processing Subject: S17
Accuracy for S17: 0.64
Classification Report for S17:
              precision    recall  f1-score   support

       False       0.76      0.72      0.74        18
        True       0.38      0.43      0.40         7

    accuracy                           0.64        25
   macro avg       0.57      0.58      0.57        25
weighted avg       0.66      0.64      0.65        25


Final Results:
Accuracies: {'S2': 0.782608695652174, 'S3': 0.9166666666666666, 'S4': 0.7391304347826086,
'S5': 0.8333333333333334, 'S6': 0.75, 'S7': 0.9166666666666666, 'S8': 0.9166666666666666,
'S9': 0.9166666666666666, 'S10': 0.84, 'S11': 0.875, 'S13': 0.7916666666666666, 'S14': 0.8
333333333333334, 'S15': 0.7916666666666666, 'S16': 0.8333333333333334, 'S17': 0.64}
```

```python
subject_ids = [f"S{i}" for i in range(2, 18) if i != 12]  # Subjects S2 to S17 excluding S12
base_path = 'data/subject_feats/'

# Initialize storage for results
accuracy_scores = {}
classification_reports = {}

for subject_id in subject_ids:
    print(f"\nProcessing Subject: {subject_id}")

    # Load the data
    file_path = f"{base_path}{subject_id}_feats_4.csv"
    data = pd.read_csv(file_path)

    # Select only EDA-related columns for features
    eda_columns = [col for col in data.columns if 'EDA' in col]
    X = data[eda_columns]  # Use only EDA columns as features
    y = data['2']  # Assuming '2' is the stress label

    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8, random_state=42)

    # Initialize and train the Random Forest Classifier
    rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
    rf_classifier.fit(X_train, y_train)

    # Predictions
    y_pred = rf_classifier.predict(X_test)

    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    accuracy_scores[subject_id] = accuracy
    classification_reports[subject_id] = report

    print(f"Accuracy for {subject_id}: {accuracy}")
    print(f"Classification Report for {subject_id}:\n{report}")

# Results
print("\nFinal Results:")
print("Accuracies:", accuracy_scores)
```

SVM

```
Processing Subject: S16
Accuracy for S16: 0.75
Classification Report for S16:
              precision    recall  f1-score   support

       False       0.80      0.89      0.84        18
        True       0.50      0.33      0.40         6

    accuracy                           0.75        24
   macro avg       0.65      0.61      0.62        24
weighted avg       0.72      0.75      0.73        24


Final Results:
Accuracies: {'S2': 0.782608695652174, 'S3': 0.7916666666666666, 'S4': 0.7826086
95652174, 'S5': 0.75, 'S6': 0.75, 'S7': 0.7916666666666666, 'S8': 0.875, 'S9':
0.75, 'S10': 0.72, 'S11': 0.75, 'S13': 0.7916666666666666, 'S14': 0.75, 'S15':
0.75, 'S16': 0.75}
```

```python
subject_ids = [f"S{i}" for i in range(2, 18) if i != 12]  # Subjects S2 to S17 excluding S12
base_path = 'data/subject_feats/'

# Initialize storage for results
accuracy_scores = {}
classification_reports = {}

count = 0
for subject_id in subject_ids:
    count=count+1
    if count == 15:
        break
    print(f"\nProcessing Subject: {subject_id}")

    # Load the data
    file_path = f"{base_path}{subject_id}_feats_4.csv"
    data = pd.read_csv(file_path)

    # Select only EDA-related columns for features
    eda_columns = [col for col in data.columns if 'EDA' in col]
    X = data[eda_columns]  # Use only EDA columns as features
    y = data['2']  # Assuming '2' is the stress label

    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    # Initialize and train the S
    svm_classifier = SVC(kernel=
    svm_classifier.fit(X_train, y_train)

    # Predictions
    y_pred = svm_classifier.predict(X_test)

    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    accuracy_scores[subject_id] = accuracy
    classification_reports[subject_id] = report

    print(f"Accuracy for {subject_id}: {accuracy}")
    print(f"Classification Report for {subject_id}:\n{report}")

# Results
```

(variable) y_train: Any

LOGISTIC REGRESSION

```
Processing Subject: S17
Accuracy for S17: 0.64
Classification Report for S17:
              precision    recall  f1-score   support

       False       0.76      0.72      0.74        18
        True       0.38      0.43      0.40         7

    accuracy                           0.64        25
   macro avg       0.57      0.58      0.57        25
weighted avg       0.66      0.64      0.65        25


Final Results:
Accuracies: {'S2': 0.782608695652174, 'S3': 0.9166666666666666, 'S4': 0.7391304
347826086, 'S5': 0.8333333333333334, 'S6': 0.75, 'S7': 0.9166666666666666, 'S8'
: 0.9166666666666666, 'S9': 0.9166666666666666, 'S10': 0.84, 'S11': 0.875, 'S13
': 0.7916666666666666, 'S14': 0.8333333333333334, 'S15': 0.7916666666666666, 'S
16': 0.8333333333333334, 'S17': 0.64}
```

```python
subject_ids = [f"S{i}" for i in range(2, 18) if i != 12]  # Subjects S2 to S17 excluding S12
base_path = 'data/subject_feats/'

# Initialize storage for results
accuracy_scores = {}
classification_reports = {}

count = 0
for subject_id in subject_ids:
    print(f"\nProcessing Subject: {subject_id}")

    # Load the data
    file_path = f"{base_path}{subject_id}_feats_4.csv"
    data = pd.read_csv(file_path)

    # Select only EDA-related columns for features
    eda_columns = [col for col in data.columns if 'EDA' in col]
    X = data[eda_columns]  # Use only EDA columns as features
    y = data['2']  # Assuming '2' is the stress label

    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    # Initialize and train the Logistic Regression Classifier
    logistic_classifier = LogisticRegression(random_state=42)
    logistic_classifier.fit(X_train, y_train)

    # Predictions
    y_pred = logistic_classifier.predict(X_test)

    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    accuracy_scores[subject_id] = accuracy
    classification_reports[subject_id] = report

    print(f"Accuracy for {subject_id}: {accuracy}")
    print(f"Classification Report for {subject_id}:\n{report}")

# Results
print("\nFinal Results:")
print("Accuracies:", accuracy_scores)
```

COMPARISON OF THE DIFFERENT ML MODELS