Stress detection using werable sensors and machine learning

Under supervision of:Dr. Prof/ ibrahim sadik

Members:

1. Gerges marzouk
2. Fady bakheet

1. Introduction

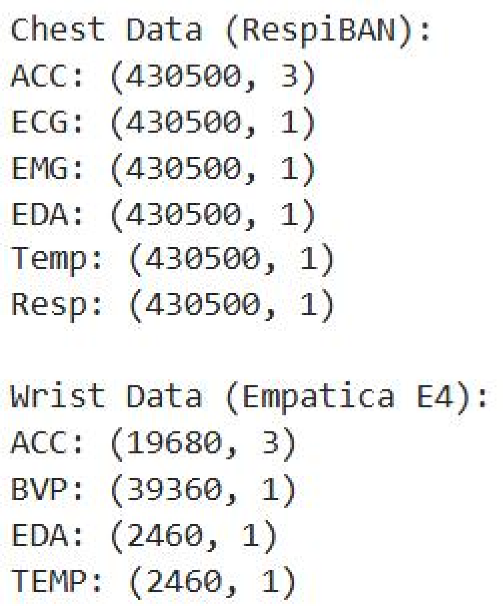
Stress detection using wearable sensors and machine learning involves collecting physiological signals, such as galvanic skin response (GSR) or heart rate, from devices like smartwatches. These signals are processed to extract features, which are then analyzed by machine learning models to classify stress levels, enabling real-time monitoring and personalized interventions.

---

2. Before the paper and our project how we understand the data-set?

The data-set contain multiple folders and files we tried to open each file in a single way and the only file that contain important data (it contain all the ways that the data collected from patients) is .pkl file.

This file contain two ways to extract the data from patients



```
Chest Data (RespiBAN):
ACC: (430500, 3)
ECG: (430500, 1)
EMG: (430500, 1)
EDA: (430500, 1)
Temp: (430500, 1)
Resp: (430500, 1)

Wrist Data (Empatica E4):
ACC: (19680, 3)
BVP: (39360, 1)
EDA: (2460, 1)
TEMP: (2460, 1)
```

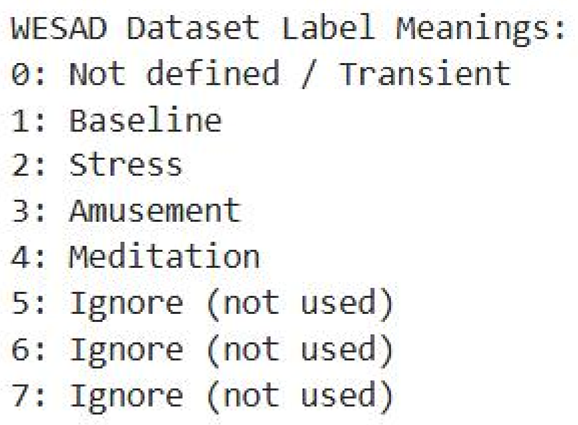
Samples from S2 only

As explained in the photo each way extracted many features that will help in the stress detection.

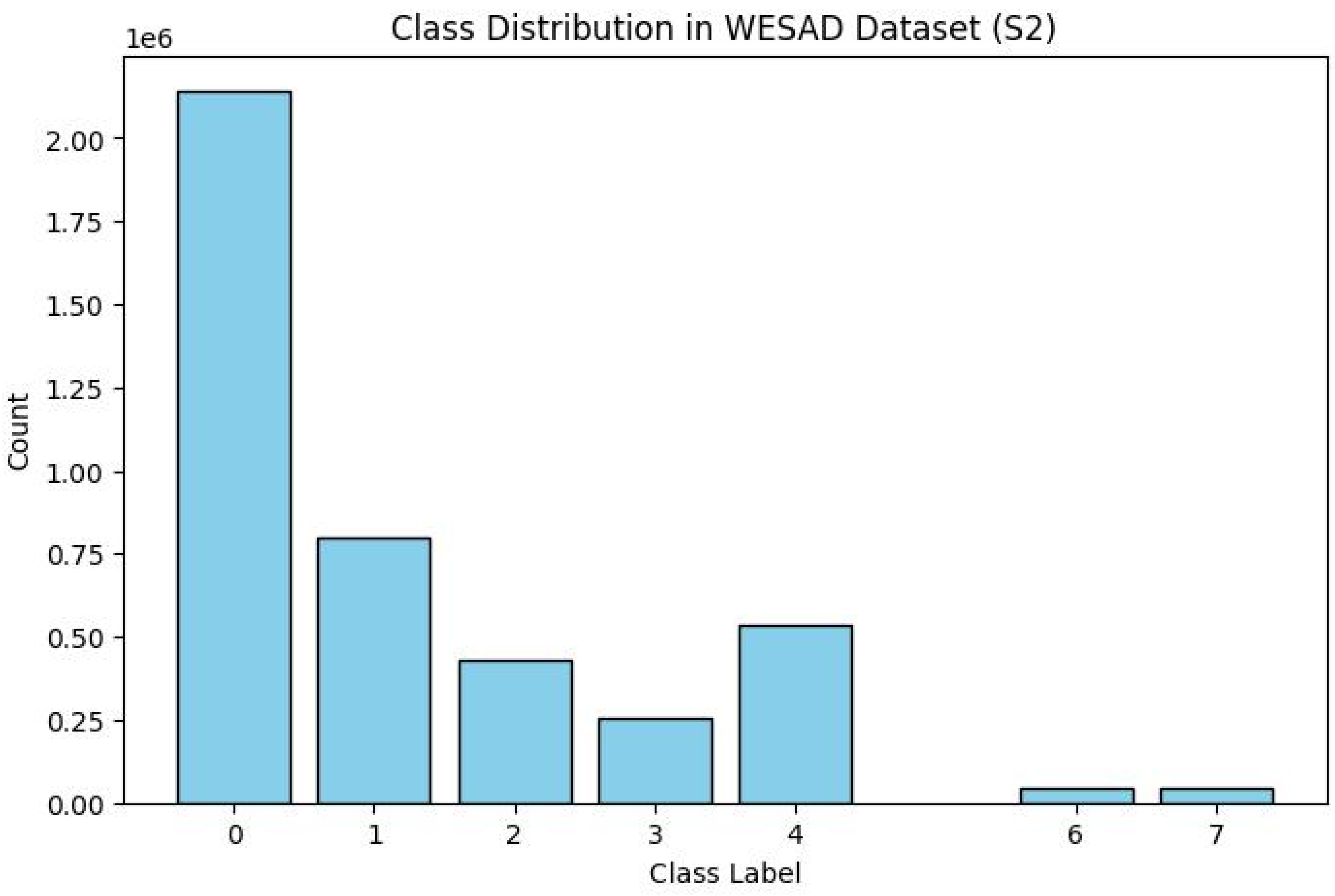
1st :-mean number of samples

2nd :-I how many direction can explained

Also the data has labels and each label mean something different:



```
WESAD Dataset Label Meanings:
0: Not defined / Transient
1: Baseline
2: Stress
3: Amusement
4: Meditation
5: Ignore (not used)
6: Ignore (not used)
7: Ignore (not used)
```



3. Research_Paper

We used a research paper that guide us with the steps of the project how be done.

What we did	Paper aproach
Wesad on wrist	Wesad on both wrist and chest
ML	DL
Random forest	Convolutional Neural Network (CNN)
Training-test split (70:30)	Cross-validation

What the paper did:

1.Introduction:

- The paper introduces the problem of stress detection, which is important for health and well-being, and proposes using wearable sensors and machine learning to identify stress automatically.

- Stress can harm health, but it's hard to measure. Wearable devices (like smartwatches or chest bands) collect body signals (e.g., heart rate, skin response). The paper uses these signals with a smart computer model (CNN) to detect when someone is stressed.

2.Dataset (WESAD)

- The study uses the WESAD dataset, which includes data from 15 people wearing two devices: RespiBAN (chest) and Empatica E4 (wrist), collecting 14 biosignals during different activities (baseline, amusement, stress, meditation, recovery).

3.Signal Processing Techniques

- Raw biosignals are processed using three techniques—Fast Fourier Transform (FFT), Cube Root (CR), and Constant Q Transform (CQT)—to turn them into frequency-based data that the CNN can use.

4.CNN Model

- The CNN is the machine learning model that takes the processed signal data and learns to classify stress states (e.g., Stress vs. Non-Stress, or distinguishing between baseline, stress, amusement, meditation, recovery).

5.Training Strategy

- The CNN is trained using a method called Leave-One-Subject-Out (LOSO) cross-validation, with 10 epochs, a batch size of 50, and the RMSprop optimizer. Hyper-parameters are tuned to ensure the model works well for new people.

6.Evaluation and Results

The model is tested to see how well it detects stress. It gets:

- 96.62% accuracy for spotting stress vs. non-stress (very good!).
- 85.03% accuracy for separating baseline, stress, and amusement.
- 81.15% accuracy for all five states (harder task, still good).

The best results come from combining all signal processing tricks (FFT, CR, CQT).

What we did

After we sucessfuly read our dataset we applied

2.Low-Pass Filter

- Inputs:
  signal: The input EDA signal.(raw signal)
  cutoff: Cutoff frequency (default 0.1 Hz).
  fs: Sampling frequency (default 4 Hz for EDA in WESAD).
  order: Filter order (default 4).
- Process:
  Calculate the Nyquist frequency (0.5 * fs).
  Normalize the cutoff frequency (cutoff / nyquist).
  Design a Butterworth filter using butter with the specified order and cutoff.
  Apply the filter to the signal using filtfilt for zero-phase filtering.
- Purpose: Removes high-frequency noise from the EDA signal to focus on low-frequency components relevant to stress

2.Define Feature Extraction Function

- Process:
  Squeeze the signal to ensure it's a 1D array.
  Apply the low-pass filter to the signal.
- Compute time-domain features:
  Mean (np.mean).
  Standard deviation (np.std).
  Minimum (np.min).
  Maximum (np.max).
  Skewness (skew).
  Kurtosis (kurtosis).
- Compute frequency-domain features:
  Extract the dominant frequency (freqs[np.argmax(psd)]).
  Calculate total power (np.sum(psd)).
- Output: A list of 8 features: [mean, std_dev, min, max, skewness, kurtosis, dominant_freq, total_power].
- Purpose: Extracts meaningful statistical and spectral features from EDA signals for stress classification.

3.Convert Data to DataFrame

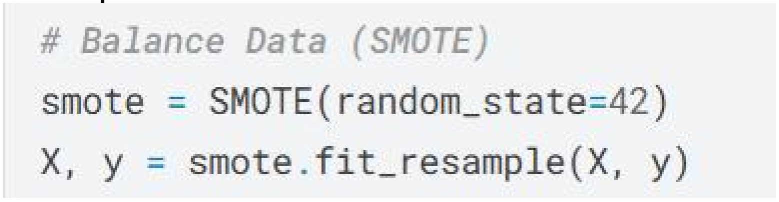
- Create a pandas DataFrame from data_list:
  -Extract features (x['Features']) and labels (x['Label']) from each dictionary.
  -Drop any rows with missing values using df.dropna.
  -Rename columns for clarity: ['Mean_GSR', 'Std_Dev_GSR', 'Min_GSR', 'Max_GSR', 'Skewness', 'Kurtosis', 'Dominant_Frequency', 'Total_Power', 'Label'].
- Purpose: Organizes the data into a structured format for analysis and modeling.

4.Normalize Features

- Use StandardScaler to normalize the feature columns (excluding the Label column):
  -Fit and transform the features (scaler.fit_transform(df.drop(columns=['Label']))) to have zero mean and unit variance.
  -Store the normalized features in X and labels in y.
- Purpose: Ensures features are on the same scale, which is important for machine learning algorithms like Random Forest.

5.Balance Data Using SMOTE

- Apply Synthetic Minority Oversampling Technique (SMOTE) to balance the dataset:
  -Use SMOTE(random_state=42) to generate synthetic samples for the minority class.
  -Resample X and y to create a balanced dataset.
- Purpose: Addresses class imbalance (e.g., more non-stress than stress samples) to improve model performance.



```
# Balance Data (SMOTE)
smote = SMOTE(random_state=42)
X, y = smote.fit_resample(X, y)
```

6.Train Random Forest and Compute Losses

- Iterate over a range of n_estimators (1 to MAX_TREES, step=1):
  -Train a RandomForestClassifier with:
    -n_estimators: Current number of trees.
    -max_depth=10: Limit tree depth to prevent overfitting.
    -class_weight='balanced': Adjust weights for imbalanced classes.
    -random_state=42: Ensure reproducibility.
  -Compute log loss for training and test sets:
    -Use model.predict_proba to get probability predictions.
    -Calculate log_loss for y_train and y_test against their respective probabilities.
  -Store losses in train_losses and test_losses.
- Purpose: Evaluates how the model's performance (log loss) changes with the number of trees.

- Train a final RandomForestClassifier with n_estimators=MAX_TREES (100), max_depth=10, class_weight='balanced', and random_state=42.
- Fit the model on X_train and y_train.
- Purpose: Creates the final model for predictions and evaluation.

4.Program flow

1. Import Required Libraries
2. Define Configuration Parameters
3. Define Low-Pass Filter Function
4. Define Feature Extraction Function
5. Define Data Loading and Preprocessing Function
6. Load and Process Data
7. Convert Data to DataFrame
8. Normalize Features
9. Balance Data Using SMOTE
10. Split Data into Training and Test Sets
11. Train Random Forest and Compute Losses
12. Train Final Random Forest Model
13. Make Predictions
14. Create Output Directory
15. Print Stress and Non-Stress Sample Counts
16. Calculate and Print Training Accuracy
17. Calculate and Print Test Accuracy
18. Print Classification Report
19. Print Confusion Matrix
20. Plot Confusion Matrix
21. Plot Loss Curve
22. Perform Loss Analysis
23. Compute and Print Feature Statistics
24. Create GUI for Stress Prediction
25. Define Prediction Function for GUI
26. Display the GUI

5.References

Human stress detection with wearable sensors using convolutional neural networks

https://oa.upm.es/78548/1/Stress.pdf