EEG Signal Classification Using Machine Learning

Muneezé Malik

April 2025

1 Introduction

1.1 Background

Electroencephalography (EEG) is a non-invasive method for monitoring brain activity through electrodes placed on the scalp. These signals reflect underlying neural oscillations and are integral to numerous applications such as neurological diagnostics, cognitive science, sleep analysis, and Brain-Computer Interfaces (BCIs). Despite its importance, EEG data is inherently challenging due to its high dimensionality, non-stationarity, and susceptibility to noise and artifacts. Furthermore, manual EEG analysis is time-consuming, prone to variability, and difficult to scale.

The emergence of Machine Learning (ML) provides a powerful framework to address these challenges. ML algorithms can automatically learn patterns within EEG data, enabling scalable, accurate classification of neurological events including epileptic seizures, cognitive states, and brain disorders.

1.2 Objective

This project aims to develop a complete machine learning pipeline for classifying EEG signals using a dataset sourced from Mendeley Data. The primary goals include:

- Preprocess raw EEG data from EDF files.
- Visualize signal patterns and class distributions.
- Train and evaluate three classifiers: Logistic Regression, Random Forest, and Support Vector Machine (SVM).
- Assess the impact of feature engineering and synthetic oversampling via SMOTE.
- Evaluate models using weighted F1-score, confusion matrices, and PCA visualization.

2 Dataset Overview

The dataset used is sourced from Mendeley Data and comprises EEG recordings from six patients. Each 1-second segment (500 samples) of EEG is labeled into one of four classes:

- Class 0: Normal EEG
- Class 1: Complex Partial Seizures (CPS)
- Class 2: Electrical Seizures
- Class 3: Non-Organized Convulsions

The data was collected in clinical settings and includes annotations for seizure events. Raw EEG files are in EDF format and were accompanied by metadata that assisted in labeling.

3 Data Preprocessing and Visualization

3.1 Preprocessing Pipeline

Using Python libraries such as `mne`, `numpy`, and `scikit-learn`, the preprocessing steps included:

1. **EDF Import:** Loaded raw data using `mne.io.read_raw_edf()`.
2. **Channel Cleanup:** Removed or reordered unused/noisy channels.
3. **Segmentation:** Split data into 1-second windows (500 samples).
4. **Labeling:** Applied seizure metadata to assign class labels.
5. **Feature Engineering:** Computed statistical features (mean, std, min, max) per channel, producing 76 features.
6. **Normalization:** Applied `StandardScaler` to normalize features.
7. **Train/Test Split:** 90% training, 10% testing.
8. **Balancing:** Used SMOTE to balance class distribution.

3.2 Visualization

To better understand the data distribution and structure:

- Sample EEG signal window visualisations highlighted significant noise and variability in raw brainwave patterns.
- Class distribution analysis before applying SMOTE revealed a major imbalance, with Class 3 notably underrepresented.

- F1 score comparison between raw and balanced data showed significant improvement in classification performance after applying SMOTE and feature engineering.

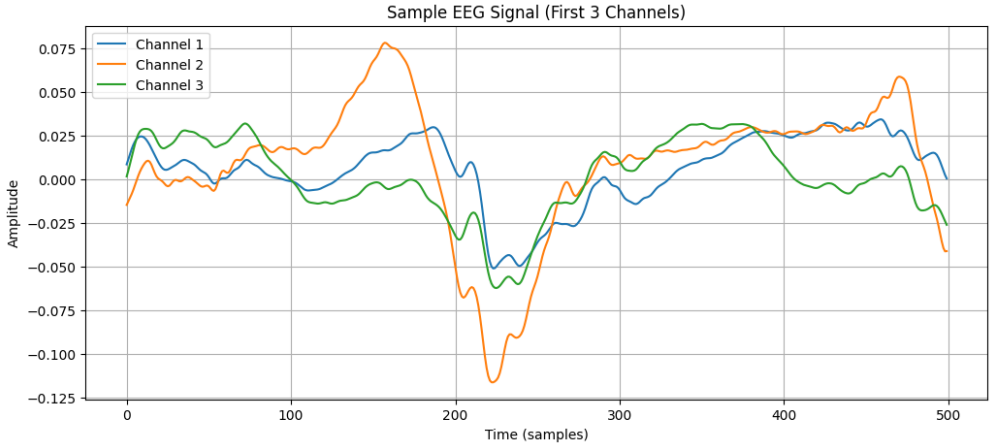

Figure 1: Sample EEG Signal Window

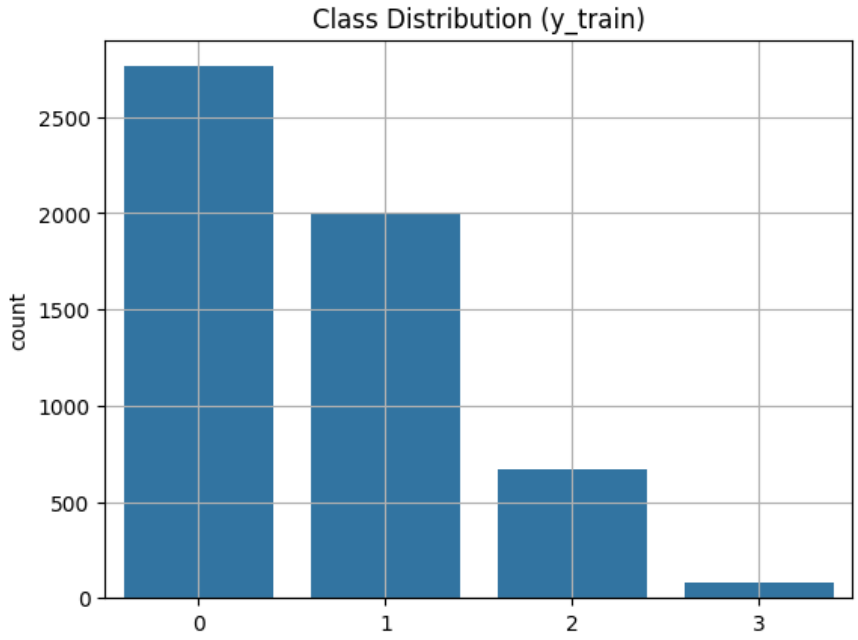

Figure 2: Class Distribution Before SMOTE

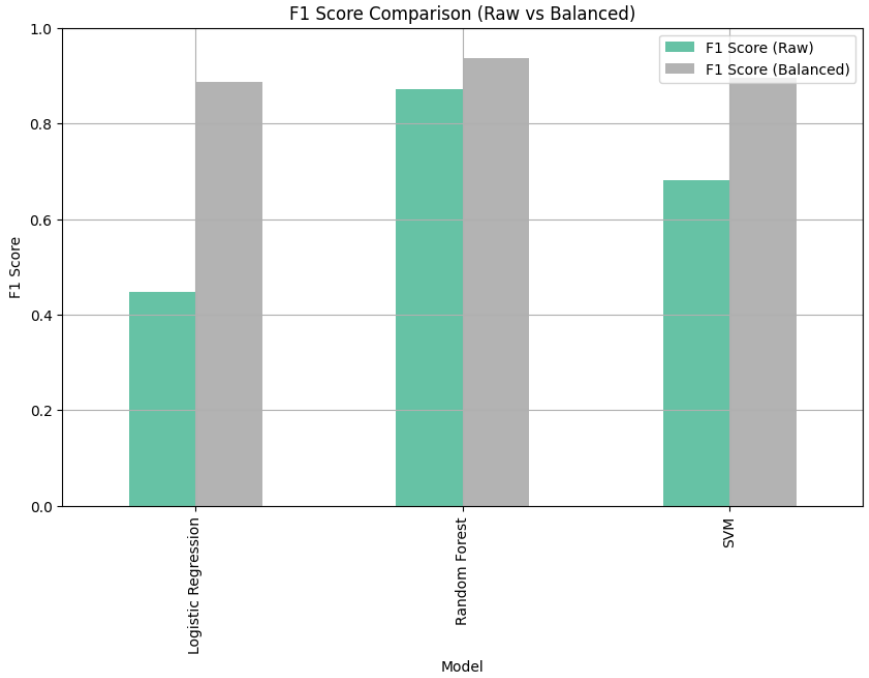

Figure 3: F1 Score Comparison (Raw vs Balanced

4 Machine Learning Models

4.1 Logistic Regression

A linear baseline model ideal for binary classification, adapted here using the one-vs-rest strategy. It uses the following hypothesis and cost functions:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \quad \text{(Sigmoid Function)}$$

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

4.2 Random Forest

An ensemble model of decision trees trained using bagging. Key advantages:

- Handles nonlinear relationships.
- Reduces overfitting.
- Provides feature importance.

4.3 Support Vector Machine (SVM)

SVM seeks a hyperplane that maximizes the margin between different classes. We used the RBF kernel with:

- C (Regularization Parameter)
- Gamma (Kernel Coefficient)

4.4 Evaluation Metric: Weighted F1-Score

To manage class imbalance:

$$\text{F1}_{\text{weighted}} = \sum_{i=1}^{N} \frac{n_i}{n} \cdot \text{F1}_i$$

Where n_i is the number of true instances in class i, and F1_i is the F1-score for that class.

5 Experiments and Results

5.1 Experiment Settings

Models were evaluated under two configurations:

1. Without Feature Engineering and SMOTE
2. With Feature Engineering and SMOTE

5.2 F1-Scores for All Models

Model	Raw Data F1	Engineered + SMOTE F1
Logistic Regression	0.68	0.89
Random Forest	0.85	0.9375
SVM	0.72	0.91

Table 1: Weighted F1-scores across different configurations

5.3 Analysis

Random Forest consistently outperformed other models, showing robustness to noise and better generalization. Logistic Regression struggled with raw features due to linear constraints but improved with engineered features. SVM showed sensitivity to SMOTE's synthetic data, performing best with feature engineering.

6 Conclusion

6.1 Summary

This study demonstrated a full EEG classification pipeline using machine learning techniques. It emphasized the value of feature engineering and balancing techniques like SMOTE. The Random Forest classifier achieved the highest weighted F1-score (0.9375).

6.2 Key Takeaways

- Feature engineering significantly boosts model accuracy.
- SMOTE effectively handles class imbalance.
- Random Forest is a dependable model for EEG classification.

6.3 Future Work

- Employ deep learning architectures (CNN, LSTM) for temporal pattern recognition.
- Integrate attention mechanisms for dynamic feature weighting.
- Apply the pipeline in real-time for BCI control and seizure prediction.

7 References

- Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. JMLR, 12, 2825–2830.
- Chawla, N.V., et al. (2002). SMOTE: Synthetic Minority Over-sampling Technique. JAIR, 16, 321–357.
- Gramfort, A., et al. (2013). MEG and EEG data analysis with MNE-Python. Frontiers in Neuroscience.
- MNE Documentation: `https://mne.tools/stable/index.html`
- EEG Dataset: `https://data.mendeley.com/datasets/5pc2j46cbc/1`