# Comparative Analysis of Machine Learning Models for EEG-Based Emotion Recognition

## 1. Introduction

Emotion recognition using electroencephalogram (EEG) signals has garnered significant attention due to its vast applications in affective computing, human-computer interaction, and mental health monitoring. This study focuses on identifying the most effective machine learning algorithms for classifying emotional states using EEG data. The "EEG Brainwave Dataset: Feeling Emotions" was utilized to evaluate the performance of various machine learning techniques, including logistic regression, support vector machine (SVM), Gaussian Naïve Bayes (GNB), decision tree, and ensemble models such as random forest, AdaBoost, LightGBM, XGBoost, and CatBoost.

To optimize the dataset for analysis, preprocessing techniques were applied, including handling missing values, feature scaling, and label encoding. Additionally, LASSO (Least Absolute Shrinkage and Selection Operator) regression was used for feature selection, reducing the dataset's dimensionality by retaining only the most relevant features. This step was crucial in improving model efficiency and mitigating overfitting.

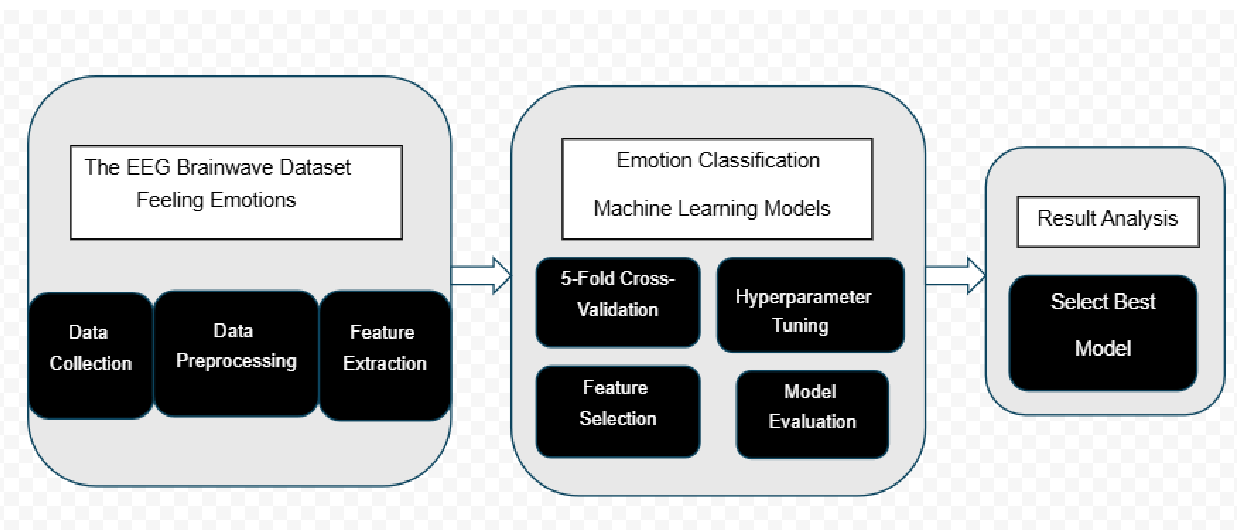


Fig. 1 - System overview of EEG-Based emotion recognition

This study addresses the challenges associated with high-dimensional EEG data, such as computational complexity and susceptibility to overfitting, particularly for simpler models like GNB. By implementing feature selection, model performance was significantly enhanced. The comparative analysis of machine learning algorithms provides insights into their effectiveness for EEG-based emotion recognition, aiding in the development of more precise and robust systems for understanding human emotions and their neural mechanisms.

# 2. Dataset Overview

The dataset comprises EEG recordings from two STEM students (one male, one female, aged 21 ± 1) exposed to visual stimuli designed to evoke three emotional states: Positive, Neutral, and Negative. The recordings were captured at a resampled frequency of 150 Hz, ensuring high temporal resolution. Each participant contributed 36 minutes of EEG data, covering all emotional states.

## 2.1 Stimuli Used

- **Negative stimuli:** Clips from *Marley and Me*, *Up*, and *My Girl*.
- **Positive stimuli:** Clips from *La La Land*, *Slow Life*, and *Funny Dogs*.
- **Neutral state:** Resting-state recordings without stimuli.

The dataset includes 2132 EEG samples, each labeled as Positive, Neutral, or Negative. A total of 2548 features were extracted using statistical, frequency-domain, and advanced feature selection methods.

## 2.2 EEG Data Collection Protocol

EEG signals were recorded using a Muse EEG headband, which provides a cost-effective yet reliable method for capturing brainwave activity. The four electrodes (TP9, AF7, AF8, TP10) were positioned according to the 10-20 international system, ensuring optimal signal acquisition for emotion recognition. Preprocessing techniques were applied to remove artifacts caused by eye blinks, muscle movements, and environmental noise.

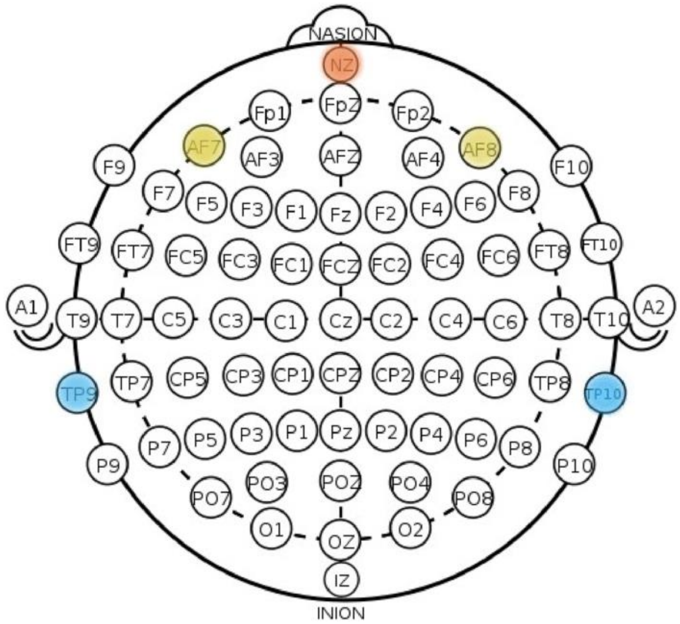


Fig. 2 - EEG electrode sensor TP9, TP10, AF7, AF8

## 2.3 Experimental Setup

Participants were seated in a quiet, dimly lit environment to minimize distractions. Each participant was exposed to six different video clips per emotional state, ensuring a diverse range of emotional responses. EEG signals were continuously recorded for one minute per stimulus, maintaining consistency across trials. To ensure high-quality EEG recordings, participants were instructed to minimize head movements and blinking.

## 2.4 Feature Extraction and Dataset Composition

- **Time-Domain Features:** Mean, standard deviation, moments, max/min values, covariance, and correlation were extracted to capture statistical characteristics.
- **Frequency-Domain Features:** Power spectral densities were computed using the Fast Fourier Transform (FFT) to analyze EEG signal distributions across frequency bands.
- **Advanced Features:** Eigenvalues, logarithmic transformations, and quantile-based statistics were included for enhanced feature representation.

The dataset was carefully balanced across emotional states to ensure fair model training and evaluation, reducing bias and improving classification robustness.

# 3. Data Preprocessing

Preprocessing techniques were applied to improve data quality and optimize model performance:

## 3.1 Handling Missing Values

Missing values were imputed using mean imputation via *SimpleImputer*, preserving dataset integrity while ensuring that no samples were lost.

## 3.2 Feature Scaling

EEG features were standardized using *StandardScaler* to have a mean of 0 and a standard deviation of 1. This step was particularly crucial for distance-based models such as SVM and KNN.

## 3.3 Label Encoding

Emotional states were converted into numerical values using *LabelEncoder*:

- Negative → 0
- Neutral → 1
- Positive → 2

## 3.4 Feature Selection Using LASSO Regression

Given the high dimensionality of the dataset (2548 features), LASSO regression was employed to retain the most significant features while discarding irrelevant or redundant ones.

**LASSO Selection Process:**

- **Regularization-Based Selection:** LASSO applies an L1 penalty, which forces the regression coefficients of less important features to shrink to zero, effectively removing them from the model.

- **Optimal Feature Selection:** Features with nonzero coefficients after LASSO regularization were retained, ensuring that only the most informative features contributed to model training.

A total of 20 highly relevant EEG features were selected based on their impact on emotion classification.

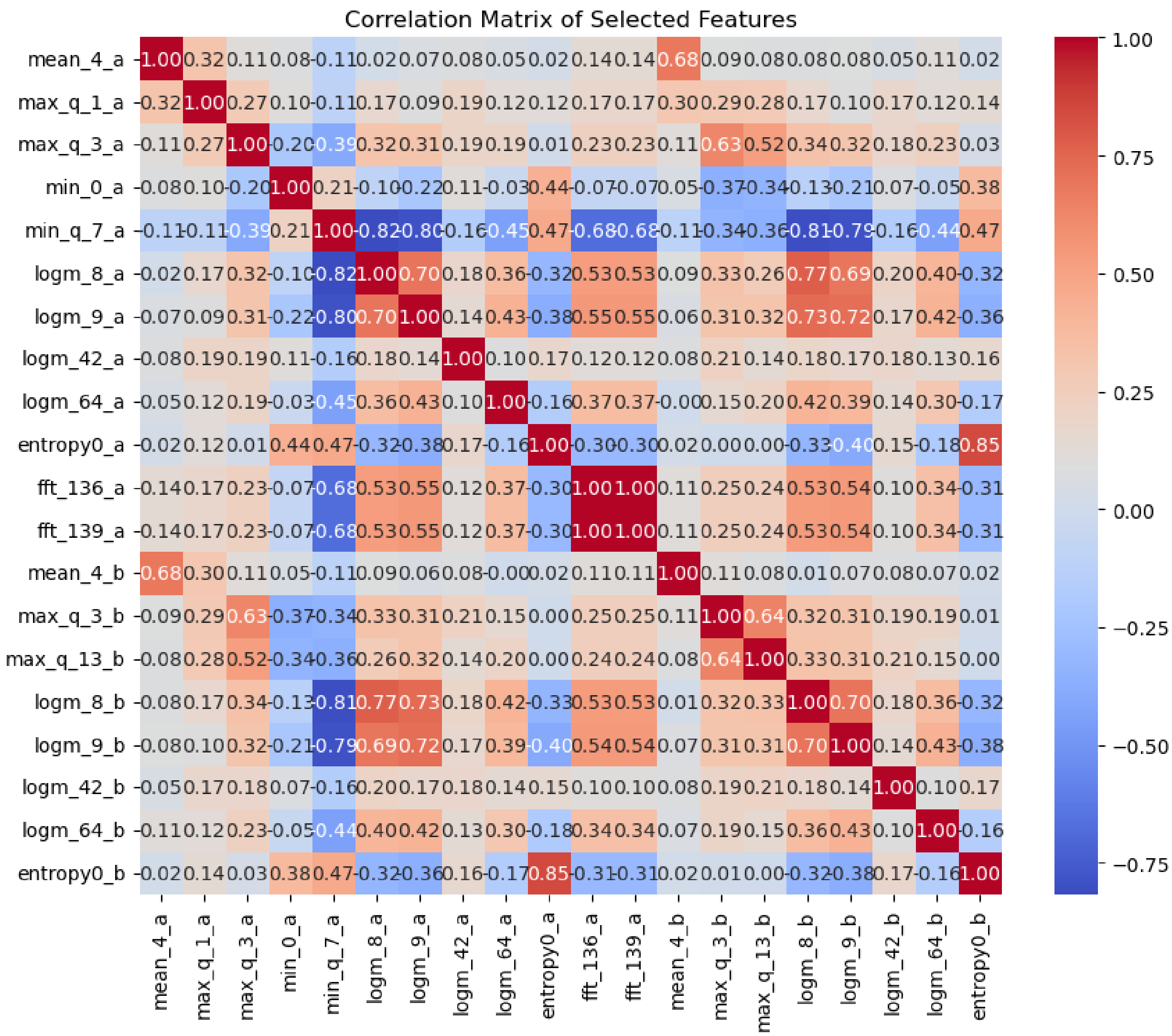


Fig. 3 - Correlation Matrix of selected feature's

## 3.5 Data Splitting

The dataset was split into 80% training and 20% testing sets. A stratified split was performed to maintain class balance, ensuring fair model evaluation.

# 4. Emotion Classification

To classify emotional states from EEG signals, ten machine learning models were implemented. Each model was chosen for its ability to handle high-dimensional data and extract meaningful patterns from EEG features. Below is an overview of their significance in EEG-based emotion recognition:

- **Logistic Regression:** A probabilistic classification model that predicts the likelihood of an EEG sample belonging to a particular emotional state.
- **Support Vector Machine (SVM):** Finds an optimal hyperplane to separate different emotional states, utilizing kernel functions for non-linear data.
- **Gaussian Naïve Bayes (GNB):** A probabilistic classifier based on Bayes' theorem, assuming feature independence.
- **Decision Tree (DT):** Creates a tree-like model by splitting EEG data into branches based on feature values.
- **K-Nearest Neighbors (KNN):** Classifies EEG signals based on similarity to the closest training samples.
- **Random Forest (RF):** An ensemble method that combines multiple decision trees to enhance classification accuracy.
- **Adaptive Boosting (AdaBoost):** Focuses on misclassified instances to improve classification performance.
- **LightGBM (Light Gradient Boosting Machine):** A fast, efficient gradient boosting framework designed for large datasets.
- **XGBoost (Extreme Gradient Boosting):** An advanced gradient boosting algorithm with regularization and parallel processing.
- **CatBoost (Categorical Boosting):** Optimized for handling categorical features efficiently without extensive preprocessing.

# 5. Model Evaluation & Results

This section assesses the performance of various machine learning models for EEG-based emotion recognition. Several evaluation metrics—accuracy, precision, recall, F1-score, and training time—were used to measure the reliability, efficiency, and generalization capabilities of each model. The analysis compares results on both the full dataset and the feature-selected dataset, highlighting the impact of feature selection on model performance.

## 5.1 Evaluation Metrics

- **Accuracy:** The proportion of correctly classified samples out of the total test samples. Higher accuracy indicates better overall model performance.
- **Precision:** The ratio of correctly predicted positive instances to total predicted positives, reducing false positives.
- **Recall:** The proportion of correctly predicted positive instances out of all actual positives, minimizing false negatives.
- **F1-Score:** The harmonic mean of precision and recall, balancing both metrics for imbalanced datasets.
- **Training Time:** The computational time required for model training, a crucial factor for real-time EEG-based applications.

## 5.2 Performance without Feature Selection

The performance of different models varied significantly when trained on the full dataset Table 1.

- **High-performing models:** LightGBM (100% accuracy), XGBoost (99.77%), and CatBoost (99.77%) emerged as the top classifiers. LightGBM achieved perfect accuracy with an efficient training time of 45.60 seconds. XGBoost, while slightly less accurate, balanced performance and computational efficiency with 99.77% accuracy and a training time of 22.56 seconds. CatBoost matched XGBoost's accuracy but was computationally expensive, requiring 987.56 seconds, making it impractical for real-time use.
- **Moderate performers:** Random Forest (98.36%), SVM (97.66%), and Logistic Regression (97.42%) delivered strong accuracy with varying trade-offs. Random Forest effectively handled non-linear data but had a longer training time (5.45 seconds). SVM performed well on high-dimensional data but required 3.76 seconds to train. Logistic Regression was the most computationally efficient (0.87s training time) but showed slightly lower accuracy.

- **Lower-performing models:** Decision Tree (95.55%), KNN (93.21%), and AdaBoost (91.10%) had noticeable declines in accuracy compared to ensemble methods. Decision Trees were prone to overfitting despite reasonable accuracy. KNN, the fastest model (0.005s training time), struggled with classification accuracy. AdaBoost delivered only 91.10% accuracy and had a significantly longer training time (66.31s), making it inefficient.
- **Weakest performer:** Gaussian Naïve Bayes (GNB) achieved only 66.04% accuracy due to its assumption of feature independence, which does not hold for EEG signals. Despite a very low training time (0.06s), the poor accuracy rendered it impractical.
-

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1 Score** | **Training Time (s)** |
|---|---|---|---|---|---|
| Logistic Regression | 97.42% | 97.41% | 97.36% | 97.37% | 0.87 |
| Random Forest | 98.36% | 98.34% | 98.33% | 98.33% | 5.45 |
| Decision Tree | 95.55% | 95.53% | 95.44% | 95.46% | 3.58 |
| KNN | 93.21% | 93.56% | 92.96% | 92.95% | 0.005 |
| SVM | 97.66% | 97.64% | 97.62% | 97.63% | 3.76 |
| AdaBoost | 91.10% | 91.59% | 90.85% | 90.88% | 66.31 |
| Gaussian Naive Bayes | 66.04% | 66.00% | 65.20% | 63.96% | 0.06 |
| LightGBM | 100% | 100% | 100% | 100% | 45.60 |
| XGBoost | 99.77% | 99.77% | 99.75% | 99.76% | 22.56 |
| CatBoost | 99.77% | 99.77% | 99.75% | 99.76% | 987.56 |

-

**Table 1 - Performance without Feature Selection**

## 5.3 Performance with Feature Selection

Feature selection using LASSO regression reduced feature dimensions, improving computational efficiency and mitigating overfitting. The results varied across models Table 2:

- **High-performing models:** LightGBM, XGBoost, and CatBoost maintained high accuracy (97.66%, 97.89%, and 98.59%, respectively). LightGBM had the most significant efficiency gain, reducing training time from 45.60s to 0.2872s, making it ideal for real-time classification. XGBoost reduced its training time from 22.56s to 0.2013s with a minor accuracy drop. CatBoost, despite being slower, significantly improved its efficiency from 987.56s to 9.1728s.
- **Moderate performers:** Random Forest experienced only a slight accuracy drop (98.36% to 97.42%) while drastically improving training efficiency (5.45s to 0.5144s). However, SVM and Logistic Regression suffered substantial accuracy losses, dropping to 89.93% and 89.23%, respectively, indicating heavy reliance on the removed features.
- **Lower-performing models:** Decision Tree dropped from 95.55% to 91.80%, but training time improved significantly (3.58s to 0.0382s). KNN showed a slight accuracy improvement (93.21% to 94.38%), suggesting the removal of noisy data rather than crucial features. AdaBoost was most affected, with accuracy falling from 91.10% to 65.81%, proving its heavy dependence on removed features.
- **Notable improvement:** Gaussian Naïve Bayes (GNB) improved the most, increasing from 66.04% to 83.14% after feature selection. This indicated that redundant features previously hindered its performance. Training time, already low at 0.06s, was further optimized to 0.0000s, making it the fastest model, albeit still outperformed by ensemble methods.
-

| Model | Accuracy | Precision | Recall | F1 Score | Training Time (s) |
|---|---|---|---|---|---|
| Logistic Regression | 89.23% | 89.48% | 89.08% | 89.07% | 0.0221 |
| Random Forest | 97.42% | 97.42% | 97.38% | 97.38% | 0.5144 |
| Decision Tree | 91.80% | 91.70% | 91.70% | 91.70% | 0.0382 |
| KNN | 94.38% | 94.39% | 94.35% | 94.31% | 0.0010 |

| | | | | | |
|---|---|---|---|---|---|
| SVM | 89.93 % | 90.74 % | 89.62% | 89.60 % | 0.1905 |
| AdaBoost | 65.81 % | 53.33 % | 64.37% | 54.59 % | 0.4967 |
| Gaussian Naive Bayes | 83.14 % | 84.76 % | 83.20% | 83.02 % | 0.0000 |
| LightGBM | 97.66 % | 97.64 % | 97.62% | 97.62 % | 0.2872 |
| XGBoost | 97.89 % | 97.92 % | 97.83% | 97.85 % | 0.2013 |
| CatBoost | 98.59 % | 98.59 % | 98.57% | 98.57 % | 9.1728 |

-

**Table 2 - Performance with Feature Selection**

## 5.4 Comparative Analysis: Feature Selection vs. Full Dataset

Feature selection significantly reduced training times while maintaining competitive accuracy in many models Table 3:

- LightGBM (-2.34%) and XGBoost (-1.88%) retained high accuracy while achieving drastic speed improvements.
- Random Forest (-0.94%) remained competitive, balancing accuracy and efficiency.
- SVM (-7.73%) and Logistic Regression (-8.19%) suffered significant accuracy drops, proving their dependence on removed features.
- AdaBoost (-25.29%) became unreliable after feature selection.
- GNB (+17.10%) had the greatest improvement, showing the impact of feature selection in reducing noise.
- **Comparative Analysis: Full Dataset vs. Feature-Selected Dataset**

| Model | Accuracy (Full) | Accuracy (Selected) | Training Time Reduction | Performance Change |
|---|---|---|---|---|
| Logistic Regression | 97.42% | 89.23% | 0.87s → 0.0221s (**~40x faster**) | ⬇ **8.19% drop** (largest decline) |
| Random Forest | 98.36% | 97.42% | 5.45s → 0.5144s (**~10x faster**) | ⬇ 0.94% drop (minimal impact) |
| Decision Tree | 95.55% | 91.80% | 3.58s → 0.0382s (**~90x faster**) | ⬇ 3.75% drop |
| KNN | 93.21% | 94.38% | 0.005s → 0.001s (**~5x faster**) | ⬆ **1.17% gain** (only improver) |
| SVM | 97.66% | 89.93% | 3.76s → 0.1905s (**~20x faster**) | ⬇ 7.73% drop |
| AdaBoost | 91.10% | 65.81% | 66.31s → 0.4967s (**~130x faster**) | ⬇ **25.29% drop** (worst performer) |
| Gaussian Naive Bayes | 66.04% | 83.14% | 0.06s → 0.0000s (**instant**) | ⬆ **17.1% gain** (best relative boost) |
| LightGBM | 100% | 97.66% | 45.60s → 0.2872s (**~160x faster**) | ⬇ 2.34% drop |
| XGBoost | 99.77% | 97.89% | 22.56s → 0.2013s (**~100x faster**) | ⬇ 1.88% drop |
| CatBoost | 99.77% | 98.59% | 987.56s → 9.1728s (**~99% faster**) | ⬇ 1.18% drop |

- **Table 3 - Feature Selection vs. Full Dataset**

## 5.5 Hyperparameter Tuning

Hyperparameter tuning using GridSearchCV optimized each model's parameters to enhance accuracy and efficiency.

- **KNN and AdaBoost:** Performance significantly improved after hyperparameter tuning, highlighting their sensitivity to parameter choices.
- **Other models:** Logistic Regression, SVM, Decision Tree, Random Forest, and GNB showed minor improvements, indicating well-suited default parameters.
- **Computational efficiency:** Hyperparameter tuning optimized training times across models, making them more viable for real-time applications.

**Hyperparameter Grid vs. Best Selections**

| Model | Hyperparameter Grid (All Tested Options) | Best Hyperparameters |
|---|---|---|
| **Random Forest** | n_estimators: [50, 100, 200]<br>max_depth: [None, 10, 20]<br>min_samples_split: [2, 5, 10]<br>min_samples_leaf: [1, 2, 4] | {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 50} |
| **Logistic Regression** | C: [0.001, 0.01, 0.1, 1, 10, 75]<br>penalty: ['l1']<br>solver: ['liblinear']<br>max_iter: [100, 200, 300, 500] | {'C': 1, 'max_iter': 500, 'penalty': 'l1', 'solver': 'liblinear'} |
| **KNN** | n_neighbors: [1, 3, 5, 7, 9]<br>weights: ['uniform', 'distance']<br>p: [1, 2] | {'n_neighbors': 1, 'p': 1, 'weights': 'uniform'} |
| **Decision Tree** | criterion: ['gini', 'entropy']<br>max_depth: [None, 10, 20, 30]<br>min_samples_split: [2, 5, 10]<br>min_samples_leaf: [1, 2, 4]<br>max_features: ['auto', 'sqrt', 'log2', None] | {'criterion': 'entropy', 'max_depth': 30, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2} |
| **SVM** | C: [0.1, 1, 10]<br>kernel: ['linear', 'rbf', 'poly']<br>gamma: ['scale', 'auto', 0.001, 0.01, 0.1, 1] | {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'} |
| **AdaBoost** | n_estimators: [50, 100, 200] | {'learning_rate': 0.01, 'n_estimators': 50} |

| | | |
|---|---|---|
| | learning_rate: [0.01, 0.1, 1.0] | |
| **Gaussian Naive Bayes (GNB)** | priors: [None, [0.2, 0.3, 0.5], [0.3, 0.4, 0.3], [0.1, 0.1, 0.8]]<br>var_smoothing: [1e-9, ..., 1e-1] | {'priors': [0.1, 0.1, 0.8], 'var_smoothing': 1e-05} |
| **LightGBM** | n_estimators: [50, 100]<br>learning_rate: [0.01, 0.1]<br>max_depth: [5, 10]<br>num_leaves: [31, 63] | {'learning_rate': 0.1, 'max_depth': 5, 'n_estimators': 100, 'num_leaves': 31} |
| **XGBoost** | max_depth: [3, 5]<br>learning_rate: [0.01, 0.1]<br>subsample: [0.5, 0.7] | {'learning_rate': 0.1, 'max_depth': 3, 'subsample': 0.7} |
| **CatBoost** | iterations: [50, 100]<br>learning_rate: [0.01, 0.1]<br>depth: [4, 6] | {'depth': 6, 'iterations': 100, 'learning_rate': 0.1} |

**Table 4 - Hyperparameter Tuning**

## 5.6 Cross-Validation Results

Cross-validation results ensured models generalized well to unseen data. The average classification performance of models across multiple folds is summarized:

- **Ensemble models:** Random Forest had the highest accuracy (98.50%), demonstrating its robustness. AdaBoost also performed well (97.28%) but required extensive computational time (2421.40s).
- **Single models:** Logistic Regression and SVM maintained strong accuracy (97.80% and 97.61%, respectively). Decision Tree and KNN provided moderate accuracy (96.95% and 96.81%, respectively). GNB remained the weakest performer (66.18%).

- **Generalization ability:** Most models showed consistent performance across folds, indicating they effectively learned classification patterns.
- **Computational efficiency:** KNN was the fastest (25.59s), while AdaBoost was the slowest (2421.40s), highlighting the importance of computational trade-offs in EEG classification.
-

**Cross-Validation Results**

**Model Performance Comparison (5-Fold CV)**

| Model | Fold-1 | Fold-2 | Fold-3 | Fold-4 | Fold-5 | Avg Accuracy | Avg Precision | Avg Recall | Avg F1-Score | Training Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| **Logistic Regression** | 98.36% | 96.49% | 96.95% | 98.36% | 98.36% | 97.70% | 97.73% | 97.70% | 97.70% | 1.82 |
| **Random Forest** | 98.83% | 98.36% | 97.65% | 99.06% | 99.77% | 98.73% | 98.74% | 98.73% | 98.73% | 2.27 |
| **Decision Tree** | 97.19% | 95.08% | 97.18% | 97.89% | 97.89% | 97.05% | 97.06% | 97.05% | 97.04% | 2.76 |
| **KNN** | 96.49% | 95.32% | 96.95% | 96.24% | 98.12% | 96.62% | 96.69% | 96.62% | 96.60% | 0.005 |
| **SVM** | 98.36% | 97.42% | 96.48% | 96.95% | 98.83% | 97.61% | 97.63% | 97.61% | 97.61% | 7.66 |
| **AdaBoost** | 70.49% | 89.93% | 91.55% | 92.72% | 69.48% | 82.84% | 88.11% | 82.84% | 80.45% | 27.45 |
| **Gaussian Naive Bayes** | 66.04% | 68.15% | 63.85% | 67.61% | 65.26% | 66.18% | 66.33% | 66.18% | 64.36% | 0.06 |
| **LightGBM** | 99.77% | 98.59% | 99.30% | 99.30% | 99.53% | 99.30% | 99.30% | 99.30% | 99.30% | 23.22 |
| **XGBoost** | 99.77% | 98.59% | 99.30% | 99.30% | 99.30% | 99.25% | 99.26% | 99.25% | 99.25% | 22.27 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **CatBoost** | 99.30% | 98.13% | 98.36% | 99.06% | 99.77% | 98.92% | 98.94% | 98.92% | 98.92% | 75.84 |

### 5.7 Final Insights & Recommendations

- **Best models:** LightGBM and XGBoost achieved the best balance of accuracy and computational efficiency, making them ideal for EEG-based emotion recognition.
- **Limitations:** CatBoost, although accurate, remained too slow for real-time applications. Feature selection significantly improved model efficiency but caused severe accuracy reductions in models like AdaBoost and SVM, making them less viable for EEG classification.
- **Future work:** Further exploration of deep learning techniques and hybrid models may improve performance beyond traditional machine learning methods.

## 6. Conclusion

This study evaluated various machine learning models for EEG-based emotion recognition, with **LightGBM** emerging as the best-performing model, achieving **high accuracy (100% before and 97.66% after feature selection)** while significantly improving computational efficiency. Feature selection reduced training time by up to **160x**, making models more practical for real-time applications.

Ensemble models like **LightGBM, XGBoost, and CatBoost** consistently delivered superior accuracy, while **Logistic Regression and SVM** provided a balance between interpretability and efficiency. **Gaussian Naïve Bayes**, initially weak, improved significantly after feature selection, highlighting the impact of noise reduction.

This research contributes to the advancement of EEG-based emotion recognition systems, with potential applications in **neuroscience, mental health monitoring, and human-computer interaction**. Future studies can explore **deep learning and advanced signal processing** to further enhance classification performance.