University School of Automation & Robotics
GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY
East Delhi Campus, Surajmal vihar
Delhi - 110092

Machine Learning

(ARM210)

Project Report

Project Title: **EEG-Based Eye State Classification**

Name: **Vigyat Goel**

Enrollment Number: **04119051622**

Email ID: **vigyatgoel@gmail,com**

Contact Number: **9310444371**

Github link :https://github.com/VigyatGoel/EEG_Eye_State_Classification

Title:

EEG-Based Eye State Classification

Abstract:

This report presents an investigation into classifying eye states using EEG data. The dataset comprises 14 EEG values recorded over time, alongside labels indicating the corresponding eye state. The proposed methodology encompasses dataset preprocessing, feature extraction, classification, and evaluation. Results demonstrate the effectiveness of the approach in accurately classifying eye states as open and close.

Keywords:

EEG, Eye state prediction, Classification, Comparative analysis, Support Vector Machine, Multi layer Perceptron

Objective:

Classification of state of eye(open or close) using EEG data.

Proposed Methodology:

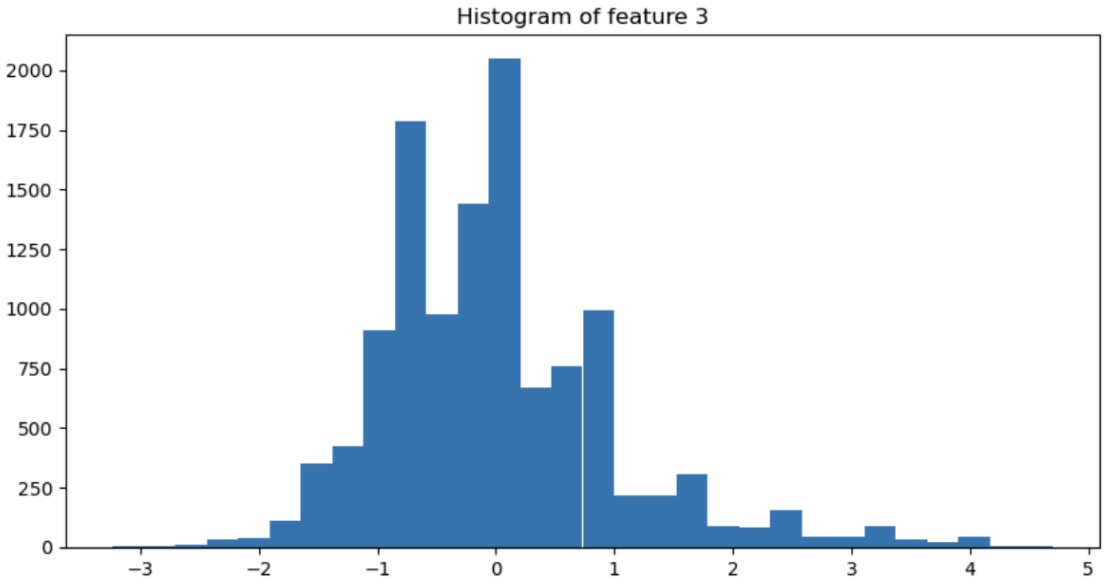
1. Datasets: The dataset consists of 14 EEG values for a total of 14980 instances, alongside labels indicating the corresponding eye state.
2. Preprocessing: Data preprocessing involves cleaning, normalization, and feature extraction from sequential EEG time-series data.
3. Model Selection: Classification models (e.g SVM, Random Forest, Neural Networks) are chosen for comparison.
4. Training and Evaluation: Models are trained on preprocessed data and evaluated using appropriate metrics such as accuracy, precision, recall, F1-score for classification.

Dataset

Link: https://archive.ics.uci.edu/dataset/264/eeg+eye+state

- The dataset consists of 14 EEG features and corresponding eye state labels, providing sequential time-series data for analysis. Each instance represents a specific time interval, capturing brain activity related to different eye states.
- Dataset has a total of **14 Row and 14980 columns**.
- Dataset consists of 14 input features and 2 output labels
- The dataset contain the following features:
    - AF3, F7, F3, FC5, T7, P7, O1 ,O2, P8, T8, FC6, F4, F8, AF4
- The dataset contains following label:
    - eyeDetection(closed or open)
- The task is to classify the output as state of eye based on inputs.

Below is a histogram of a features made using matplotlib library



Data Preprocessing

Preprocessing serves as a cornerstone in enhancing the quality of EEG signals and improving classification accuracy.

Key preprocessing steps include:

- **Train-Test Split**: The dataset is divided into training and testing sets to evaluate model performance on unseen data.
    - Splitting was done using the **train_test_split** function from sklearn library.
    - After Splitting the data:
        - Training data size: 11984 (80% of original data)
        - Testing data size: 2996 (20% of original data)

- **Standardization**: Feature standardization is performed to rescale feature values to have a **mean of 0** and a **standard deviation of 1**, ensuring uniformity across features.
    - Standardization was done using the **StandardScaler()** function of sklearn library.

- **Outlier Removal**: Outliers, if any, are identified and removed from the dataset to mitigate their adverse effects on model training and performance.
    - Outlier removal was done by calculating **Z-Score** of each data point in the dataset using zscore() function of scipy library.
    - Points whose **Z-score was more than 3 were removed** from the dataset.

Model Selection

Various machine learning algorithms are explored for their efficacy in classifying eye states based on EEG features.

Models which were employed include:

- **Logistic Regression**: A simple yet powerful model for binary classification tasks, often used as a baseline for more complex algorithms.
- **Random Forest Classifier**: It aggregates the predictions of multiple decision trees to improve accuracy and reduce overfitting.
- **Gradient Boosting Classifier**: Iteratively builds a strong predictive model by focusing on the mistakes of previous models, achieving high performance in various tasks.
- **Support Vector Machine**: Constructs a hyperplane in a high-dimensional space to separate classes, making it effective in both linear and non-linear classification.
- **K Neighbors Classifier**: Makes predictions based on the majority class among its k nearest neighbors, adapting well to local patterns in the data.
- **Decision Tree Classifier**: It partitions the feature space into a tree structure to make predictions based on simple rules.
- **Perceptron**: A fundamental unit of neural networks, it learns to classify inputs by adjusting weights through iterative training.
- **Multilayer Perceptron Classifier**: A deep learning model composed of multiple layers of perceptrons, capable of learning complex patterns and relationships in data.

Hyper Parameter Tuning

Hyper Parameter means selecting optimal parameters for a model to enhance the accuracy and efficiency of the model.

Hyper parameter tuning was done using the **RandomizedSearchCV** function of the sklearn library.

- Hyperparameter tuning using random search optimizes model performance by systematically sampling from a predefined hyperparameter space.
- This approach efficiently explores a wide range of hyperparameter combinations, helping to find optimal configurations that enhance model accuracy and generalization.
- By leveraging random search, models can be fine-tuned effectively, saving computational resources compared to exhaustive search methods while still achieving competitive results.

Example of Randomized search for multi layer perceptron:

```
param_distributions = {
    'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 50)],
    'activation': ['logistic', 'tanh', 'relu'],
    'solver': ['lbfgs', 'sgd', 'adam'],
    'alpha': [0.0001, 0.001, 0.01],
    'learning_rate': ['constant', 'adaptive']
```

Training and Evaluation

- The selected machine learning models are trained using the preprocessed dataset to learn the underlying patterns in the egg_eye_state dataset.
- Subsequently, the trained models are evaluated using appropriate performance metrics such as accuracy, cross validation, and F1-score.
- This evaluation provides insights into the effectiveness of each algorithm in accurately classifying eye state based on input features.

In this section, we present the training and evaluation results of the selected machine learning algorithms using the provided metrics for the classification of eye state based on input features.

- **Logistic Regression**:
    - Average cross-validation score: 0.6402250336750697
    - Accuracy: 62.839879154078545%
    - F1 score: 0.5408544172542513

- **Random Forest Classifier**:
    - Average cross-validation score: 0.9215148020561259
    - Accuracy: 93.68915743538099%
    - F1 score: 0.9311355311355312

- **Gradient Boosting Classifier**:
    - Average cross-validation score: 0.9381353295611617
    - Accuracy: 94.86404833836858%

    - F1 score: 0.9441401971522454

- **Support Vector Machine**:
    - Average cross-validation score: 0.9743982089760488
    - Accuracy: 98.05303793219201%
    - F1 score: 0.9790613718411553

- **K Neighbors Classifier**:
    - Average cross-validation score: 0.9580297820448722
    - Accuracy: 96.67673716012085%
    - F1 score: 0.9641952983725136

- **Decision Tree Classifier**:
    - Average cross-validation score: 0.8301851943852216
    - Accuracy: 83.04800268546492%
    - F1 score: 0.8176236908631275

- **Perceptron**:
    - Average cross-validation score: 0.5531770838288141
    - Accuracy: 57.09969788519638%
    - F1 score: 0.4384885764499121

- **Multi Layer Perceptron Classifier**:
    - Average cross-validation score: 0.9715438878735572
    - Accuracy: 97.44880832494125%
    - F1 score: 0.9723837209302325

Result & Discussion

- The results indicate that **SVM performs the best** among the classifiers, achieving the highest **accuracy of 98.05%** and an **F1 score of 0.979.**
- It is closely followed by the **Gradient Boosting Classifier** and the **Multilayer Perceptron Classifier**.
- These models demonstrate strong performance in accurately classifying eye states based on EEG data, suggesting their suitability for real-world applications in health monitoring and cognitive analysis.
- However, models such as **Perceptron and Logistic Regression exhibit comparatively lower performance**, indicating that they may not be well-suited for this classification task.

- One notable observation is the significant discrepancy in performance between SVM and simpler models like Logistic Regression and Perceptron.
- This suggests that the decision boundary for classifying eye states in EEG data is likely complex and nonlinear, making more sophisticated models like SVM better suited to capture these intricacies.

Conclusion & Future Work

In conclusion, the study demonstrates the feasibility of using machine learning techniques for EEG-based eye state classification. The results highlight the effectiveness of models such as Support Vector Machine, Gradient Boosting Classifier, and Multilayer Perceptron Classifier in accurately classifying eye states.

Moving forward, efforts should be directed towards investigating ensemble methods that combine the strengths of multiple models to further enhance classification performance and robustness. Additionally, incorporating domain knowledge, such as incorporating temporal dependencies or hierarchical relationships in EEG signals, could lead to more informative feature representations and improved model interpretability.

Lastly, with the rapid advancements in deep learning techniques, exploring the applicability of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) for EEG-based eye state classification could open up new avenues for research.

Overall, the findings from this study lay the foundation for future research endeavors aimed at advancing our understanding of cognitive processes and developing innovative technologies for health monitoring and cognitive analysis.

References

- https://in.mathworks.com/campaigns/offers/next/choosing-the-best-machine-learning-classification-model-and-avoiding-overfitting.html
- https://www.quora.com/Which-machine-learning-model-is-best-for-classification-1
- https://scikit-learn.org/stable/modules/preprocessing.html
- https://scikit-learn.org/stable/supervised_learning.html
- https://stackoverflow.com/questions/31621215/what-classification-model-should-i-use-new-to-machine-learning-recommendation
- https://chatgpt.com/
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
- https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.zscore.html
- https://pandas.pydata.org/docs/user_guide/index.html
- https://numpy.org/doc/stable/user/index.html#user