**Introduction**

This report delves into a machine learning project conducted in Python, focusing on Electroencephalography (EEG) signal analysis to discern between eyes open and eyes closed states. EEG signals, which capture electrical activity in the brain, provide valuable insights into cognitive processes and neurological conditions.

Leveraging Python's libraries such as NumPy, pandas, scikit-learn, and matplotlib, the project's primary objective was to develop a robust classification model capable of accurately identifying the state of a subject's eyes based on 14 distinct features derived from EEG measurements.

**Problem Definition**

This project focuses on accurately classifying EEG signals into eyes open and eyes closed states. EEG signals reflect brain electrical activity and exhibit complex patterns representing different cognitive states. The objective is to use machine learning techniques to develop a model capable of discerning these states based on 14 distinct features extracted from EEG measurements.

The significance of this problem spans multiple domains. In medical diagnostics, the model can detect neurological disorders. In human-computer interaction, it aids in developing brain-computer interfaces. In cognitive research, it contributes to understanding brain dynamics.

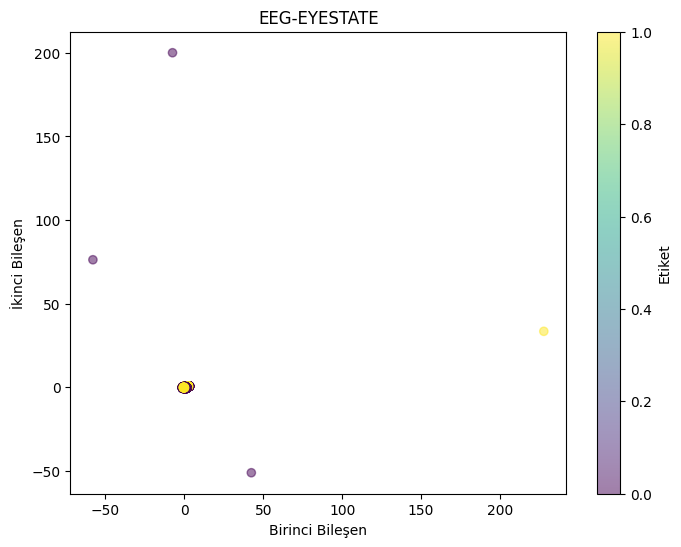
The challenges include EEG signal complexity, individual variability, and real-time classification requirements. Addressing these necessitates robust preprocessing, effective feature selection, and suitable machine learning algorithms.

**Dataset**

All data is from one continuous EEG measurement with the Emotiv EEG Neuroheadset. The duration of the measurement was 117 seconds. The eye state was detected via a camera during the EEG measurement and added later manually to the file after analysing the video frames. '1' indicates the eye-closed and '0' the eye-open state. All values are in chronological order with the first measured value at the top of the data. There are 14980 instances.

**1.Visualization:**

We utilized various visualization techniques, including scatter plots, histograms, and correlation matrices, to gain insights into the dataset's structure and the relationship between EEG features and eye states. Additionally, Principal Component Analysis (PCA) was employed to reduce the dimensionality of the dataset while preserving its variance. This technique allowed us to visualize the data in a lower-dimensional space, making it easier to discern patterns and clusters related to eye states.



**2.Preprocessing:**

Prior to model training, the dataset underwent preprocessing steps. This included scaling features to a common range, such as min-max scaling or standardization, to ensure uniformity and prevent certain features from dominating others during model training. Additionally, we explored feature engineering techniques to extract relevant information and enhance the model's predictive power.

**3. Handling Missing Data (loss):**

The dataset was carefully examined for missing values. Missing data in each feature was addressed by replacing them with the mean value of that feature, thereby maintaining data integrity and consistency throughout the dataset. This approach ensured that missing values were filled, preserving the integrity and consistency of the dataset.

**Model Definition**

In this section, we delve into the model definition phase where we explored various machine learning algorithms to classify EEG signals into eyes open and eyes closed states. Through rigorous experimentation, we evaluated seven different models and obtained the following accuracy scores:

* Logistic Regression: 58.0% (C=1.0, solver='lbfgs')
* **K Nearest Neighbors (KNN): 91.2%** (k=5, metric: minkowski)
* Support Vector Machine (SVM): 60.6% (C=1.0, kernel='linear')
* Kernel-SVM: 71.5% (C=1.0, kernel='rbf', gamma='scale')
* **Naive Bayes: 51.0%** (GaussianNB)

These accuracy scores provide insights into the performance of each model in distinguishing between eye states. We will further discuss the implementation details, hyperparameter tuning, and evaluation metrics for each model in the subsequent sections.

**Evaluation**

**Logistic Regression:**

**Accuracy Score: 58.01%**

**Precision Score:56.34%**

**Recall Score:36.61%**

**F1 Score:44.38%**

-The accuracy score is low, and the confusion matrix is unbalanced. The model seems not to be fitting the data properly.

**K Nearest Neighbors (KNN):**

**Accuracy Score: 91.28%**

**Precision Score:91.60%**

**Recall Score:89.13%**

**F1 Score:90.35%**

-The KNN model performs remarkably well with a high accuracy score and a balanced confusion matrix.

**Support Vector Machine (SVM):**

**Accuracy Score: 60.68%**

**Precision Score:67.07%**

**Recall Score:27.64%**

**F1 Score:39.15%**

-The SVM model provides lower accuracy compared to KNN, and the confusion matrix is unbalanced.

**Kernel-SVM:**

**Accuracy Score: 71.56%**

**Precision Score:79.86%**

**Recall Score:50.61%**

**F1 Score:61.96%**

- The SVM model with the RBF (Radial Basis Function) kernel performs slightly better than the previous SVM model but still exhibits imbalance.

**Naive Bayes:**

**Accuracy Score: 51.00%**

**Precision Score:47.71%**

**Recall Score:73.96%**

**F1 Score:58.00%**

- Naive Bayes model provides a lower accuracy compared to others, and the confusion matrix is unbalanced.

**Decision Tree Classification:**

**Accuracy Score: 83.07%**

**Precision Score:82.23%**

**Recall Score:80.37%**

**F1 Score:81.29%**

- The decision tree classifier exhibits moderate performance among the machine learning algorithms. The confusion matrix is balanced, but the accuracy score is relatively lower.

**Random Forest Classification:**

**Accuracy Score: 89.45%**

**Precision Score:93.63%**

**Recall Score:82.56%**

**F1 Score:87.75%**

- The random forest classifier outperforms other models by providing high accuracy.

**Conclusion**

-K Nearest Neighbors (KNN) and Random Forest Classification models exhibit the best performance on the EEG Eye State dataset.

-Some other models have produced unbalanced results or low accuracy.

-When choosing a model, it's essential to select the algorithm that best fits the dataset and the specific problem. For instance, for balanced results, models like KNN or Random Forest can be preferred.

-Additionally, effective preprocessing of the data and selection of appropriate hyperparameters are crucial to enhance model performance.

https://colab.research.google.com/drive/1q\_wJczROz\_qTqqcNowJUhByjwQcz45KW