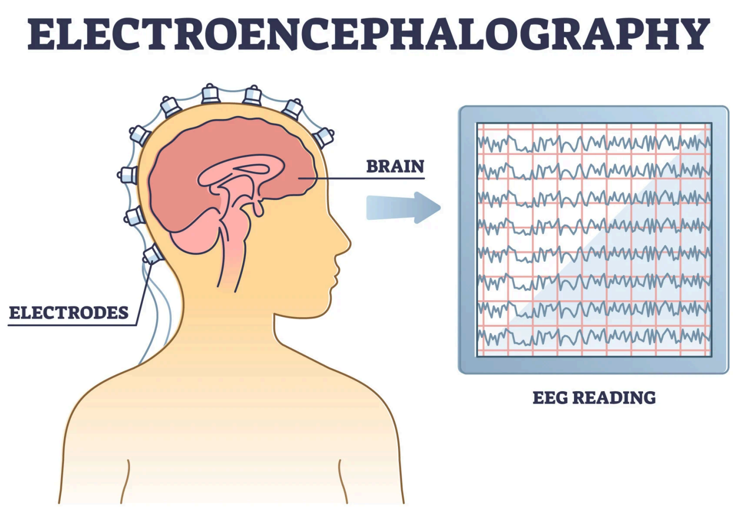
CRISP-DM model

Enhancing EEG Classification for Seizure Detection: A Data-driven Approach



4/29/2024
New York Tech
Introduction To Data Mining

Michael Serra
Denzel Green
Michael Valenzuela
Team Data Goats 2

Business Understanding:

The dataset consists of EEG recordings from subjects under different conditions, with the aim of classifying whether a subject is experiencing epileptic seizure or not based on their brain activity. This classification task is crucial for medical diagnosis and treatment planning, as identifying seizure activity can lead to timely intervention and management.Timely identification of epileptic seizures can significantly impact patient outcomes. Firstly, early detection allows for swift medical intervention, potentially averting serious consequences such as injury or status epilepticus, a life-threatening condition characterized by prolonged seizures. Secondly, accurate classification enables healthcare professionals to tailor treatment plans to individual patients, optimizing therapy effectiveness and minimizing adverse effects.This classification task is crucial as it forms the foundation for various use cases, such as the development of wearable EEG monitoring devices equipped with seizure detection algorithms. These devices could provide individuals with epilepsy and their caregivers with real-time alerts, enabling prompt intervention during seizures, thereby enhancing safety and quality of life

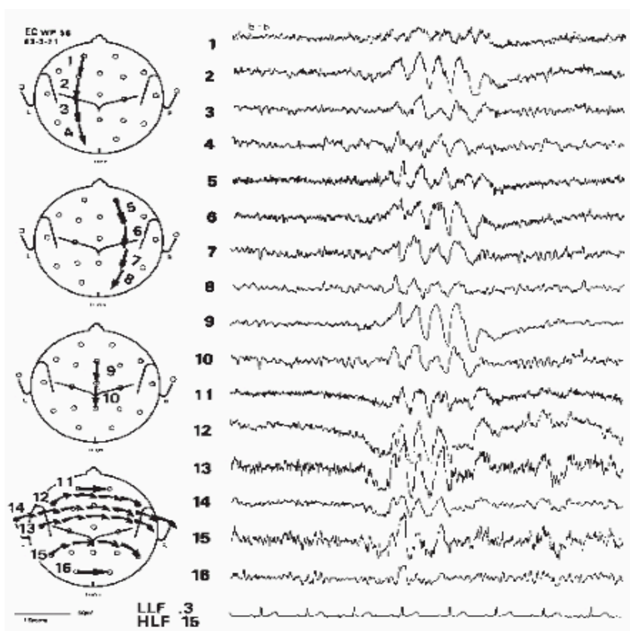


Figure 0 - EEG Readings

Data Understanding:

Data Structure:

a. The dataset comprises 500 files (5 folders, each with 100 files), each representing a single subject.
b. Each file contains recordings of brain activity for 23.6 seconds, sampled into 4097 data points.
c. Each data point represents the EEG recording at a different point in time.
d. The response variable, denoted as y, is located in column 179 and contains categorical labels {1, 2, 3, 4, 5}.
e. Classes 2, 3, 4, and 5 represent non-seizure activities, while class 1 represents epileptic seizure activity.
f. Explanatory variables X1 to X178 represent the EEG recordings.

Response Variable (y):

g. Class 1 indicates epileptic seizure activity.
h. Classes 2, 3, 4, and 5 represent different non-seizure activities, such as:
   i. Class 2 eyes open
   ii. Class 3 eyes closed
   iii. Class 4 recording from a healthy brain area
   iv. Class 5 recording from the area with a tumor,

Explanatory Variables (X1 to X178):

i. Each row contains a 178-dimensional vector representing a randomly selected 1-second sample from the EEG recordings.
j. These vectors capture the EEG activity at different time intervals.

Data Quality:

k. Data has been pre-processed and ready to use. There are no missing values, outliers, or noise. The dataset is clean.

Objective:

l. The primary objective is to build a classification model to distinguish between epileptic seizure and non-seizure activities using EEG recordings.
m. According to the dataset description. Given the binary nature of the classification task (seizure vs. non-seizure), authors typically perform binary classification, focusing on class 1 (seizure) against the rest. Which is what we will do. By understanding the business context and the dataset structure, we can proceed to the next steps of the CRISP-DM model: Data Preparation and Modeling, with a focus on optimizing the use case of developing a wearable EEG monitoring device with seizure detection algorithms.

Data Preparation:

In our data preparation phase, we initially intended to apply various techniques such as data cleaning, transformation, reduction, and integration to ensure the quality and suitability of the dataset for our classification task. However, upon closer examination, we found that the dataset provided was already pre-processed and did not require extensive cleaning or transformation. While traditional data preparation tasks were not utilized due to the dataset's preprocessed nature, we encountered a crucial modification requirement. To satisfy the requirements of our classification models, particularly for logistic regression, we needed to modify the dataset by classifying it into binary categories.

To achieve this, we performed the following steps:

Verified Preprocessing:

1. We ensured that the dataset was properly formatted and structured, ready for further analysis.

Data Classification:

1. Prior to modeling, we assigned labels to each attribute based on the classification task. Specifically, we classified the EEG recordings into two categories:
   a. 0: Non-seizure activity (classes 2, 3, 4, and 5)
   b. 1: Epileptic seizure activity (class 1)

Element Addition Filter:

1. To facilitate the binary classification, we employed the AddElement filter to create a new attribute based on a conditional operation involving the existing class attribute (A179). This operation classified instances as 1 if the class was indicative of epileptic seizure activity (class 1) and 0 otherwise.

Changing Class to Nominal:

1. Finally, we modified the class attribute to be nominal, ensuring compatibility with the classification algorithms utilized in our analysis.

Modeling:

K-nearest Neighbors (k-NN)

K-nearest neighbors (k-NN) is a pattern recognition technique that uses training datasets to identify the k closest relatives in subsequent cases.

When using k-NN for classification, it calculates to place data in the category of its nearest neighbor. If k = 1, it would be assigned to the class closest 1. K is classified based on a plurality survey of its neighbors. (5 Types of Classification Algorithms in Machine Learning (monkeylearn.com))

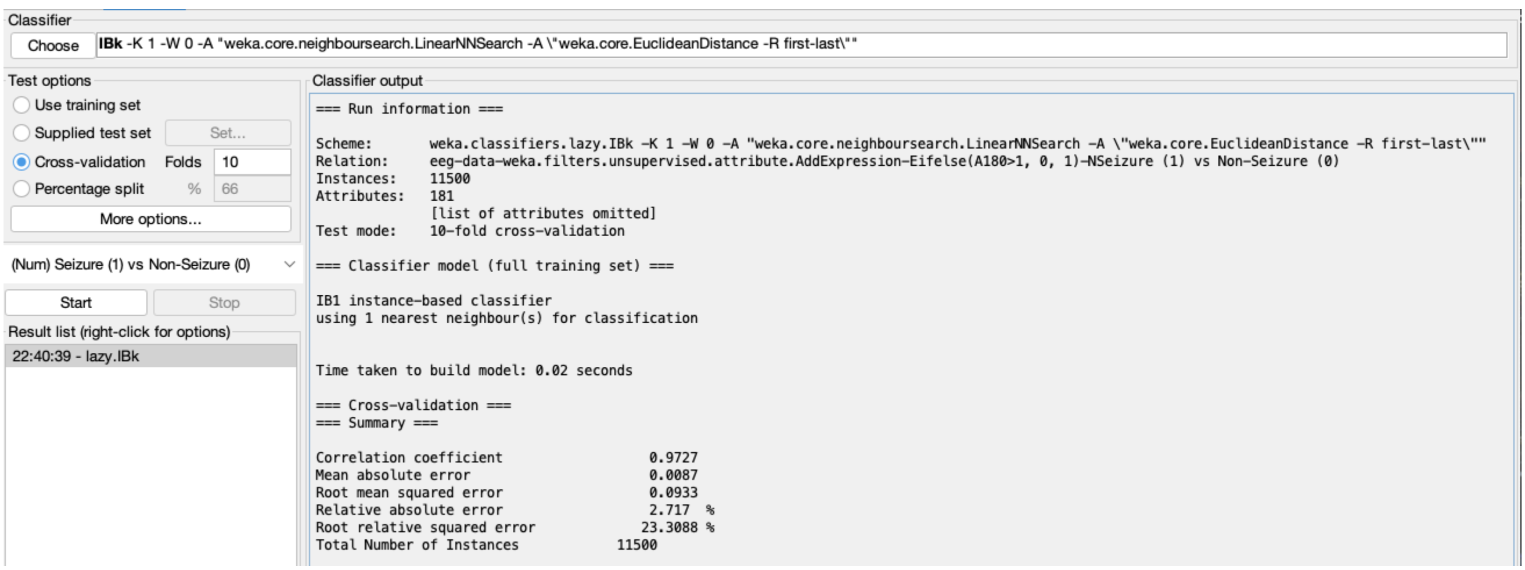


Figure 1 - k-NN Default Settings

Number of Neighbors (k):

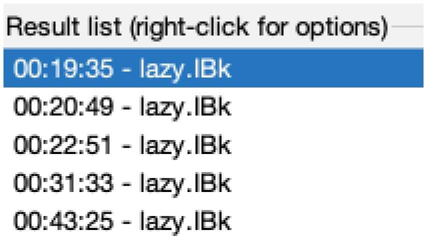
1. Values: 1, 3, 5, 7, 9
2. The parameter k has an important effect on the model's bias-variance trade-off. Smaller k values result in more complicated decision boundaries, which can contribute to overfitting, whereas bigger k values may increase bias while decreasing variance.

Distance Metric:

1. Values: Euclidean distance, Manhattan distance, Chebyshev distance, Minkowski distance (with varying p values), Mahalanobis distance
2. Different distance metrics can influence how k-NN assesses similarity between data points, hence impacting decision boundaries and classification accuracy.

***Disclaimer* My computer's CPU was taking a toll. So I used the Explorer tab**

Figure 2 - k-NN list



00:19:35 - k = 1, d = Euclidean distance
00:20:49 - k = 3, d = Manhattan distance
00:22:51 - k = 5, d = Chebyshev distance
00:31:33 - k = 7, d = Minkowski distance
00:43:25 - k = 9, d = Mahalanobis distance

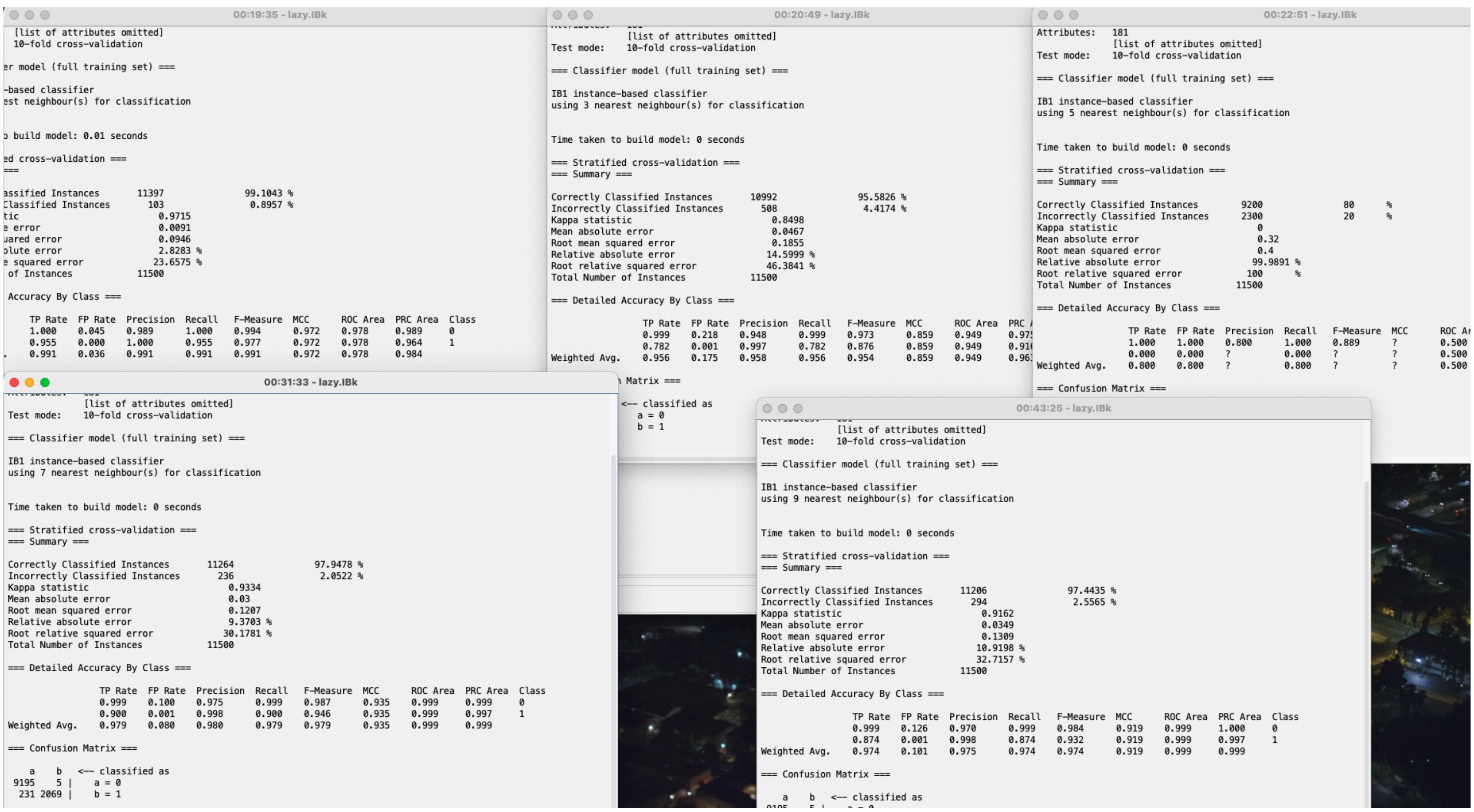


Figure 3 - k-NN list Results

Decision Tree

A decision tree is a supervised learning algorithm that is perfect for classification problems, as it's able to order classes on a precise level. It works like a flow chart, separating data points into two similar categories at a time from the "tree trunk" to "branches," to "leaves," where the categories become more finitely similar. This creates categories within categories, allowing for organic classification with limited human supervision.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11500              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances            11500

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     0
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     1
Weighted Avg.    1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000

=== Confusion Matrix ===

    a    b   <-- classified as
 9200    0 |    a = 0
    0 2300 |    b = 1
```

Figure 4 - Decision Tree Results Default

Minimum Number of Instances per Leaf (MinNumObj):

1. Values: 1, 5, 10, 20, 50
2. This parameter determines the halting condition for tree development. Smaller numbers may result in overly complicated trees that are prone to overfitting, whereas bigger values may result in simpler trees that fail to recognize data patterns.

Confidence Threshold for Pruning (ConfidenceFactor):

1. Values: 0.1, 0.25, 0.5, 0.75, 0.9
2. Pruning is a strategy for avoiding overfitting by deleting nodes that do not significantly increase forecast accuracy. The confidence factor sets the threshold for trimming choices.

Test output

```
Tester:     weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix
Analysing:  Percent_correct
Datasets:   1
Resultsets: 5
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       5/5/24, 1:21 AM


Dataset                   (1) trees.J48 | (2) trees. (3) trees. (4) trees. (5) trees.
-------------------------------------------------------------------------------------
'eeg-data-weka.filters.un(100)   100.00 |   100.00     100.00     100.00     100.00
-------------------------------------------------------------------------------------
                                (v/ /*) |    (0/1/0)    (0/1/0)    (0/1/0)    (0/1/0)


Key:
(1) trees.J48 '-C 0.1 -M 1' -217733168393644444
(2) trees.J48 '-C 0.25 -M 5' -217733168393644444
(3) trees.J48 '-C 0.5 -M 10' -217733168393644444
(4) trees.J48 '-C 0.75 -M 20' -217733168393644444
(5) trees.J48 '-C 0.9 -M 50' -217733168393644444
```

Figure 5 - Decision Tree List Results (Percent Correct)

Logistic Regression

One method of predicting a binary result is logistic regression: either something happens or nothing happens. This can be expressed as Yes/No, Pass/Fail, Alive/Dead, etc.
The independent variables are examined to provide a binary outcome, with the findings falling into one of two groups. The independent variables may be categorical or numerical, but the dependent variable is always categorical. Written like this:

$$\mathbf{P(Y=1|X) \text{ or } P(Y=0|X)}$$

It determines the probability of the dependent variable Y given the independent variable X. This may be used to determine if a word has a good or negative meaning (0, 1, or a scale in between). Alternatively, it may be used to identify the item in a photograph (tree, flower, grass, etc.), using each.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       11495               99.9565 %
Incorrectly Classified Instances         5                0.0435 %
Kappa statistic                          0.9986
Mean absolute error                      0.0004
Root mean squared error                  0.019
Relative absolute error                  0.1372 %
Root relative squared error              4.7602 %
Total Number of Instances            11500

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.002    0.999      1.000    1.000      0.999    1.000     1.000     0
                 0.998    0.000    1.000      0.998    0.999      0.999    1.000     1.000     1
Weighted Avg.    1.000    0.002    1.000      1.000    1.000      0.999    1.000     1.000

=== Confusion Matrix ===

    a    b   <-- classified as
 9200    0 |    a = 0
    5 2295 |    b = 1
```

Figure 6 - Logistic Regression Results Default

Regularization Strength (Ridge):

1. Values: 0.01, 0.1, 1, 10, 100
2. The regularization strength parameter, also known as 'ridge' or 'lambda (λ)', determines the level of regularization in logistic regression. It penalizes coefficients that are excessive to prevent overfitting. Higher ridge values produce stronger regularization, which can assist minimize overfitting but may also increase bias.

MaxIts (Maximum Number of Iterations):

1. Values to try: 50, 100, 200, 500, 1000
2. The maxIts parameter limits the number of iterations (epochs) used in the optimization procedure during model training. It affects the logistic regression model's convergence behavior and training duration. Changing this parameter can influence how efficiently the model converges to the optimal solution.

Test output

```
Tester:     weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matri
Analysing:  Percent_correct
Datasets:   1
Resultsets: 5
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       5/5/24, 2:23 AM


Dataset                   (1) function | (2) functi (3) functi (4) functi (5) funct
-----------------------------------------------------------------------------------
'eeg-data-weka.filters.un(100)   98.72 |   100.00 v   100.00 v   100.00 v   99.83 v
-----------------------------------------------------------------------------------
                               (v/ /*) |    (1/0/0)    (1/0/0)    (1/0/0)   (1/0/0)


Key:
(1) functions.Logistic '-R 0.01 -M 50 -num-decimal-places 4' 3932117032546553727
(2) functions.Logistic '-R 0.1 -M 100 -num-decimal-places 4' 3932117032546553727
(3) functions.Logistic '-R 1.0 -M 200 -num-decimal-places 4' 3932117032546553727
(4) functions.Logistic '-R 10.0 -M 500 -num-decimal-places 4' 3932117032546553727
(5) functions.Logistic '-R 100.0 -M 1000 -num-decimal-places 4' 3932117032546553727
```

Figure 7 - Logistic Regression List Results (Percent Correct)

Evaluation

To evaluate the classifiers developed using the K-Nearest Neighbors (k-NN), Decision Trees, and Logistic Regression methods, we will use the following relevant assessment metrics:

> **Accuracy:** Accuracy refers to the proportion of properly categorized occurrences out of total instances. It gives an overall assessment of the classifier's performance.
>
> **F1-Score:** The F1-Score is the harmonic average of accuracy and recall. It balances both false positives and false negatives, which is especially valuable when the class distribution is skewed.

K-nearest Neighbors (k-NN)

Best classifier configuration:
k = 1, d = Euclidean distance
Accuracy: 99.1043%
F1-Score: 0.994

Decision Tree

Best classifier configuration:
C = .1, M = 1
Accuracy: 100%
F1-Score: 1.0

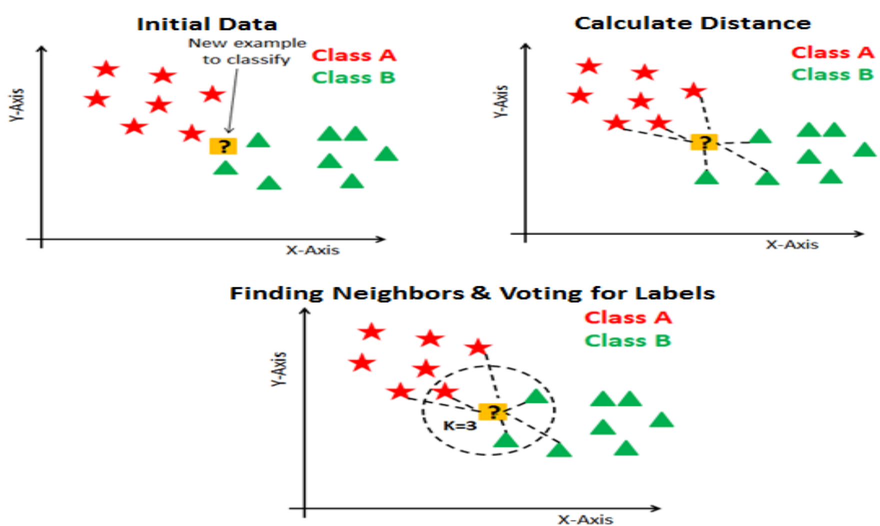
Logistic Regression

Best classifier configuration:
R = 0.1, M = 100
Accuracy: 100%
F1-Score: 1.0

ZeroR

Best classified configuration:
Batch Size: 100
Accuracy: 80%
F1-Score: 0.889

Figure 8 - k-NN vs Logistic Regression



Evaluation Explanation

k-NN: With the best configuration of k = 1 and the Euclidean distance metric, the k-NN classifier scored a high F1-Score (0.994) and accuracy (99.1043%). This performance demonstrates that the classifier efficiently uses closest neighbor information to categorize instances. Unlike ZeroR, which just predicts the majority class, k-NN examines instance similarity and assigns labels based on their nearest neighbors, resulting in much greater accuracy and F1-Score.

Decision Tree: With the ideal configuration of C = 0.1 and a minimal number of cases per leaf (M) = 1, the Decision Tree classifier achieved 100% accuracy and an F1-Score of 1.0. Decision Trees use a tree-based structure to split the feature space, capturing complicated decision boundaries in data. Unlike ZeroR, which ignores feature information, Decision Trees examine several qualities at the same time, yielding greater classification results.

Logistic Regression: The Logistic Regression classifier likewise obtained perfect accuracy (100%) and F1-Score (1.0) with the ideal regularization strength (R) of 0.1 and maximum number of iterations (M) of 100. Logistic Regression uses a logistic function to describe the likelihood of belonging to a class, capturing complicated connections between input data and class labels. Unlike ZeroR, which does not use feature information, Logistic Regression considers the full feature space, resulting in higher classification accuracy and F1-score.

Overall

Each classifier (k-NN, Decision Tree, and Logistic Regression) outperformed the baseline ZeroR classifier in terms of accuracy and F1-Score. Unlike ZeroR, which ignores feature information and relies simply on the majority class, the classifiers use a variety of methods and strategies to efficiently harness feature space, resulting in better performance when categorizing EEG data for seizure detection.

Deployment Plan:

Infrastructure Setup:

- Implement AWS EC2 instances to host the model, which ensures scalability and stability.
- Containerize the model with Docker, which allows for consistent deployment across several environments.
- Deploy the containers to Kubernetes clusters for orchestration and administration.

API Integration:

- Deploy Flask or Django to create a RESTful API with endpoints for model inference.
- Implement JWT-based authentication to ensure safe access to API endpoints.
- Utilize API Gateway services, such as AWS API Gateway, for API management and monitoring.

Monitoring and Maintenance:

- Configure AWS CloudWatch to watch critical metrics such as CPU utilization, memory consumption, and request latency.
- Use centralized logging services such as AWS CloudTrail or the ELK stack to monitor model activity and discover abnormalities.
- Set up alerts and notifications with AWS SNS or PagerDuty to notify stakeholders of any issues or performance decline.
- Schedule frequent maintenance tasks including system upgrades, container patching, and model retraining.

Data Management:

- Use AWS Glue or Apache Airflow to create data pipelines for ingestion, preprocessing, and storage.
- Store raw and processed data in Amazon S3 buckets to ensure longevity and scalability.
- Use AWS Athena to query and analyze S3 data, making it easier to explore and train models.
- Use AWS DataSync to transmit data between on-premises systems and cloud storage.

Documentation and Training:

- Use tools such as Confluence or GitHub Wiki to create comprehensive documentation covering deployment processes, API usage, and troubleshooting tips.
- Conduct training sessions for developers, DevOps engineers, and end users to get them acquainted with the deployed model and its integration points.
- Make available online resources such as video lessons, API documentation, and code repositories for ongoing learning and reference.

Monitoring and Maintenance Requirements:

- Regularly monitor AWS CloudWatch metrics, logs, and alarms to maintain system health and performance.
- API endpoint functionality and stability are validated using automated testing with tools such as Postman or Newman.
- Continuous integration and delivery (CI/CD) pipelines automate model deployment, testing, and version control.
- Scheduled backups and disaster recovery strategies are in place to prevent data loss and preserve company continuity.

Conclusion:

The deployment strategy describes a reliable and scalable infrastructure for hosting the seizure detection model in a real-world hospital setting. The deployed solution offers high availability, reliability, and security by using cloud-native services as well as best monitoring and maintenance practices. With continual monitoring and maintenance, the model adapts to changing data patterns and healthcare needs, providing doctors with correct insights for epilepsy diagnosis and management.

Citation:

Agarwal, M. (2024). EEG-Datasets: A list of all public EEG-datasets. https://github.com/meagmohit/EEG-Datasets?tab=readme-ov-file

Andrzejak, RG. (2024). Nonlinear Time Series Analysis. Retrieved from https://www.upf.edu

MonkeyLearn. (2020). 5 Types of Classification Algorithms in Machine Learning [Web article]. Retrieved from https://monkeylearn.com/blog/classification-algorithms/

University of California, Irvine. (Year). Epileptic Seizure Recognition Dataset. data.world. https://data.world/uci/epileptic-seizure-recognition/workspace/project-summary?agentid=uci&datasetid=epileptic-seizure-recognition

University of Georgia. (2011). Weka-Tutorial-Exercises.pdf [PDF document]. Retrieved from https://cobweb.cs.uga.edu/~khaled/DMcourse/Weka-Tutorial-Exercises.pdf

University of Waikato. (Year). Using the mathexpression filter - Weka Wiki [Web page]. Retrieved from https://waikato.github.io/weka-wiki/using_the_mathexpression_filter/