American Journal of Mathematical and Computer Modelling
2025; 10(1): 19-28
http://www.sciencepublishinggroup.com/j/ajmcm
doi: 10.11648/j.ajmcm.20251001.13
ISSN: 2578-8272 (Print); ISSN: 2578-8280 (Online)

Comparative Study of Extreme Gradient Boosting (XGBOOST), K-Nearest Neighbors (KNN), and Random Forest for Migraine Classification

Boniface Ngugi Kamau*, Bonface Malenje, Charity Wamwea, Lena Anyango Onyango

Department of Statistics and Actuarial Sciences, Jomo Kenyatta University of Agriculture and Technology (JKUAT), Nairobi, Kenya

Email address:

bonifacekamau345@gmail.com (Boniface Ngugi Kamau), bmalenje@jkuat.ac.ke (Bonface Malenje),
cwamwea@jkuat.ac.ke (Charity Wamwea), lenaonyango75@gmail.com (Lena Anyango Onyango)
*Corresponding author

To cite this article:

Boniface Ngugi Kamau, Bonface Malenje, Charity Wamwea, Lena Anyango Onyango. (2025). Comparative Study of Extreme Gradient Boosting (XGBOOST), K-Nearest Neighbors (KNN), and Random Forest for Migraine Classification. *American Journal of Mathematical and Computer Modelling, 10*(1), 19-28. https://doi.org/10.11648/j.ajmcm.20251001.13

Received: 3 March 2025; **Accepted:** 14 March 2025; **Published:** 31 March 2025

Abstract: Migraine is a common neurological disorder that can seriously compromise the quality of life of the affected individuals. Migraine's typical diagnosis is solely dependent on traditional diagnostic methods which relies on patient self-reporting and clinical judgment, which can be subjective and prone to errors. The main objective of this study was to model migraine classification using Extreme Gradient Boosting (XGBoost), Random Forest, and K-Nearest Neighbors (KNN) algorithms, integrating Least Absolute Shrinkage and Selection Operator (LASSO) for feature regularization. Through this study, the classifications abilities of these machine learning models were evaluated to determine which among them is superior in terms of classifying the type of migraine one is suffering from. To prevent overfitting and enhance interpretability, LASSO regression was utilized for feature regularization. The models were trained with a labeled data set, hyperparameter tuning was achieved through Grid Search to systematically explore different combinations of hyperparameters and identify the optimal settings that maximize models performance. The models were evaluated based on accuracy, precision, recall, ROC-AUC, F1-score and computation time. The top-performing model was deployed into a web-based application using Spring Boot. XGBoost outperformed the other models, achieving an accuracy of 92.4%, an AUC of 96.0%, an F1-score of 91.65%, and a sensitivity of 92.24%, with a false positive rate of 1.59% and a computation time of 2.08s. Random Forest followed closely with 91.6% accuracy, a 94.0% AUC, an F1-score of 90.49%, and a sensitivity of 86.45%, but required 4.65s of computation time. K-Nearest Neighbors (KNN) demonstrated the lowest performance, with an accuracy of 86.6%, an AUC of 91.0%, F1-score of 80.53%, a sensitivity of 79.32%, and the highest computation time of 9.51s. XGBoost was found to be the most appropriate choice for migraine classification. This study highlights the promise of machine learning in enhancing migraine diagnosis through objective and data-driven means.

Keywords: Random Forest, K-Nearest Neighbors, Extreme Gradient Boosting, Least Absolute Shrinkage and Selection Operator

1. Introduction

Migraine is a common and disabling neurological disorder in which frequent headaches of different intensity are accompanied by symptoms like nausea, photophobia and phonophobia. The influence of migraine stretches beyond the acute physical pain by impacting the overall well being, private life, professional performance as well as livelihood of an individual [2]. Migraine can be categorized into different subtypes based on their clinical features and related neurological symptoms. One rare type is basilar-type aura, which impacts the brainstem and can cause symptoms like vertigo, double vision, and trouble speaking. Familial hemiplegic migraine is a genetic form that leads to temporary

paralysis or weakness on one side of the body, often accompanied by aura symptoms. The most common type is migraine without aura, which presents as throbbing headaches along with nausea, phonophobia, and photophobia, but without any preceding neurological issues. Sporadic hemiplegic migraine has similar characteristics to familial hemiplegic migraine but occurs without a known family history. Typical aura with migraine involves transient neurological symptoms such as visual disturbances, sensory changes, or speech difficulties before the headache, while typical aura without migraine includes these symptoms without the headache following. The "Other" category is for migraine cases that don't neatly fit into established subtypes. Understanding these classifications is essential for accurate diagnosis and the creation of effective treatment strategies [10].

Correct classification and diagnosis of migraine should be given much priority, which will enable the medical personnel to come up with the best strategies to handle the attacks as well as improve the patients' outcomes. Common diagnosing methods for migraine are mostly made by the doctor with the help of patients' descriptions and medical history. However, these methods appear subjective and their genuine accuracy is often highly dependent on the individual's perceptiveness [10]. With the increasing number of datasets, particularly more specialized ones, it is now feasible to take advantage of advanced machine learning approaches, to consequently improve the classification of the migraine variants [4]. This research included Least Absolute Shrinkage and Selection Operator features regularization technique so as to achieve excellent model performance. Least Absolute Shrinkage and Selection Operator regulates features by adding a penalty equivalent to the absolute value of the magnitude of coefficients, effectively shrinking some coefficients to zero and thus performing variable regularization [7]. Least Absolute Shrinkage and Selection Operator inclusion to the machine learning apparatus could as well as help to avoid over-fitting, and to thus increase the models' performance.

A review of existing literature underscores the importance of this research in connecting traditional diagnostic methods with machine learning-based classification models. Previous studies have investigated various techniques for classifying migraine, yet the use of LASSO for feature regularization has not been thoroughly examined. By enhancing feature regularization and boosting model interpretability, this study contributes to the growing area of Artificial Intelligence in medical diagnostics. The findings hold considerable importance for healthcare, offering a more objective and data-driven approach to migraine classification. This advancement could result in improved diagnostic accuracy, facilitate early intervention, and ultimately enhance patient care.

2. Methodology

2.1. Lasso Regularization Technique

The Least Absolute Shrinkage and Selection Operator (LASSO) is a regression analysis method that enhances model prediction accuracy and interpretability by performing features regularization [7].

2.2. Standard Linear Regression

In standard linear regression, we aim to minimize the residual sum of squares (RSS):

$$\text{RSS} = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 \tag{1}$$

where:

1. y_i is the response variable for the i-th observation.
2. β_0 is the intercept.
3. β_j are the regression coefficients.
4. x_{ij} are the predictor variables.
5. n is the number of observations.
6. p is the number of predictors.

2.3. Introduction of LASSO Penalty

LASSO modifies the standard linear regression by adding a penalty term to the RSS. This penalty is the sum of the absolute values of the regression coefficients (excluding the intercept):

$$\text{LASSO Penalty} = \lambda \sum_{j=1}^{p}|\beta_j| \tag{2}$$

The parameter λ controls the strength of the penalty. The LASSO objective function to minimize becomes:

$$\text{LASSO Objective Function} = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j| \tag{3}$$

Effect of the LASSO Penalty

1. When $\lambda = 0$, the LASSO solution is the same as the ordinary least squares (OLS) solution.
2. As λ increases, the penalty term forces some of the β_j coefficients to shrink towards zero. When λ is sufficiently large, some coefficients become exactly zero, effectively performing variable selection.

2.4. Optimization

To find the LASSO solution, we minimize the LASSO objective function. This optimization problem can be solved using various algorithms, including coordinate descent and least angle regression (LARS).

Coordinate Descent Algorithm

Coordinate descent is an iterative optimization algorithm that updates one coefficient at a time, holding the others fixed. The update rule for each β_j can be derived from the partial derivative of the LASSO objective function with respect to

β_j[8]: For each predictor j:

$$\beta_j^{(new)} = \frac{S\left(\sum_{i=1}^{n} x_{ij}(y_i - \beta_0 - \sum_{k \neq j} \beta_k x_{ik}), \lambda\right)}{\sum_{i=1}^{n} x_{ij}^2} \quad (4)$$

where $S(z, \gamma)$ is the soft-thresholding operator defined as:

$$S(z, \gamma) = \text{sign}(z) \max(|z| - \gamma, 0) \quad (5)$$

2.5. Interpretation of Results

Once the optimal λ is selected, the final LASSO model is fitted using the entire dataset. The resulting model includes only the predictors with non-zero coefficients, providing a simplified and interpretable model.

2.6. General Non Parametric Models

A statistical non parametric model takes the form of;

$$Y = M(X_i) + \epsilon_i \quad \text{for} \quad i = 1, 2, 3, ...n \quad (6)$$

Where:

1. Y is the dependent variable, which represents the output of the classification model.
   In our case the output has seven classes namely; Basilar-type aura, Familial hemiplegic migraine , Migraine without aura, Other, Sporadic hemiplegic migraine, Typical aura with migraine, and Typical aura without migraine.
2. $X_i = x_1, x_2, x_3, \ldots, x_n$ are the independent variables.
   In our case we have both continuous valuables and binary variables (yes/no)
3. ϵ_i is the error term.
4. $M(X)$ is the mean function, sometimes expressed as $E(Y|X)$.

The error term ϵ has the following assumptions:

1. E$[\epsilon] = 0$, i.e., the mean of the error is expected to be zero.
2. var$[\epsilon] = \sigma_\epsilon^2$, i.e., the variance of the error function is σ^2.
3. There is no interaction between the error function and the independent variables, i.e., cor$(X, \epsilon) = 0$.

To estimate the mean function $M(X)$, we can use different models such as the Random Forest model, XGBoost model, and the K-Nearest Neighbors model.

2.6.1. Random Forest

The mathematical formula for a Random Forest algorithm involves the aggregation of predictions from multiple decision trees, typically represented as [18]:

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} h(x, \theta_b) \quad (7)$$

Where:

1. $\hat{f}(x)$ represents the predicted outcome for input x.
2. B denotes the number of trees in the forest.
3. $h(x, \theta_b)$ signifies the prediction made by the bth tree with parameters θ_b.

For each tree $b = 1, \ldots, B$, a random sample n is drawn from the training dataset with replacement, typically encompassing about two-thirds of the data. Subsequently, a random subset of predictors, selected without replacement from the full set of predictors, is chosen. Utilizing these two samples, the first random forest tree T_b is constructed, following the same methodology as standard decision trees, with the randomly selected subset of the training data [1, 12, 13]. This process iterates until the terminal node size is achieved. These steps are replicated for a predefined maximum number of trees, resulting in the ensemble of decision trees denoted as $T_{b=1}^{B}$. In order to make a prediction in the case of classification, the prediction is given as follows:

Let $\hat{C}_b(x)$ be the class predicted by the b-th tree. Then the prediction for the random forest ensemble $\hat{C}_{\text{rf}}(x)$ is calculated as:

$$\hat{C}_{\text{rf}}(x) = \text{majority vote} \left\{ \hat{C}_b(x) \right\}_{b=1}^{B} \quad (8)$$

The equation presented above depict how the final classification prediction is made in a random forest. In the classification scenario, a majority vote is conducted among the predictions of individual decision trees to determine the final class choice [12]. In the context of branching within decision trees in a random forest, the Gini Index is employed to determine how the trees branch from root nodes to their daughter nodes. The Gini Index (GI) is calculated as:

$$GI = 1 - \sum_{c=1}^{C} p_i^2 \quad (9)$$

The Gini Index (GI) is a metric utilized in decision tree algorithms, where p_i represents the frequency of each class observed in the dataset, and C denotes the number of classes in the classification problem. This index, computed for each tree, utilizes class information to determine the most probable branching decisions at each node [11]. Entropy measures the amount of uncertainty or randomness in the data, indicating how mixed the classes are within a dataset. It is calculated using the formula:

$$\text{Entropy}(D) = -\sum_{i=1}^{C} p_i \log_2(p_i) \quad (10)$$

where p_i represents the probability of class i in the dataset D. The algorithm evaluates different thresholds to find the best split by minimizing the impurity of the resulting subsets. The effectiveness of a split is determined using the formula:

$$\begin{aligned} \text{Split}(D, j, t) = &\left(\frac{|D_{\text{left}}|}{|D|} \times \text{Impurity}(D_{\text{left}}) \right) + \\ &\left(\frac{|D_{\text{right}}|}{|D|} \times \text{Impurity}(D_{\text{right}}) \right) \end{aligned} \quad (11)$$

where $|D_{\text{left}}|$ and $|D_{\text{right}}|$ are the sizes of the left and right

subsets, respectively, and Impurity is a measure such as entropy or Gini impurity.

2.6.2. Extreme Gradient Boosting

The loss function of XGBoost is minimized by the objective function. That consists of two major parts. These include the loss function and the regularization function. The function for XGBoost, denoted by L, is defined as follows:

$$L = \sum_{i=1}^{n} \text{loss}(y_{\text{res}}, h(x)) + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 + \alpha \sum_{j=1}^{T} |w_j| \quad (12)$$

the term

$$\sum_{i=1}^{n} \text{loss}(y_{\text{res}}, h(x)) \quad (13)$$

represents the loss function, which measures the prediction loss errors. The subsequent terms

$$\frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 and \alpha \sum_{j=1}^{T} |w_j| \quad (14)$$

represent the regularization function, which penalizes model complexity to prevent overfitting. XGBoost employs a gradient boosting framework to refine the model by minimizing the gradient of the loss function concerning the predicted outcomes [15]. The loss function described is represented as:

$$\text{loss} = \sum_{i=0}^{n} (y_i - \hat{y})^2 \quad (15)$$

where:

$\hat{y}$: predicted value

y_i: actual values

During each iteration, the algorithm computes the negative gradient of the loss function, and subsequently constructs a tree to accommodate these gradients. XGBoost utilizes a sequential approach where decision trees are constructed parallel to each other, preventing overfitting while capturing essential patterns, thereby enhancing model generalization [15].

The learning rate parameter in XGBoost controls the impact of each tree on the ensemble, preventing overfitting and enhancing generalization by adjusting the step size during optimization.

In XGBoost, final predictions are computed by summing up the predictions from all decision trees in the ensemble, and for classification tasks, a softmax transformation is applied to obtain class probabilities [15].

Hyperparameter tuning in XGBoost optimizes parameters like learning rate and regularization to balance model complexity, enhancing performance through techniques like grid search.

2.6.3. K-Nearest Neighbors

K-Nearest Neighbors is a straightforward algorithm that retains all existing data points and categorizes new instances by measuring their similarity to the existing ones, typically using distance functions [17].

There are different ways of calculating the distance function. These ways differs with the type of data, for instance if we have continuous data we use the following ways to calculate the distance;

1. Euclidean
2. Manhattan
3. Minkowski

$$d_{\text{Euclidean}}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (16)$$

$$d_{\text{Manhattan}}(x, y) = \sum_{i=1}^{n} |x_i - y_i| \quad (17)$$

$$d_{\text{Minkowski}}(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} \quad (18)$$

For the categorical data we use the Hamming distance function. The Hamming distance formula calculates the number of positions at which corresponding symbols differ between two strings of equal length.

Let s_1 and s_2 be two strings of length n, and let $\delta(s_1[i], s_2[i])$ be a function that returns 0 if the symbols are the same and 1 if they are different. Then, the Hamming distance $H(s_1, s_2)$ between s_1 and s_2 is given by:

$$H(s_1, s_2) = \sum_{i=1}^{n} \delta(s_1[i], s_2[i]) \quad (19)$$

Where $s_1[i]$ and $s_2[i]$ are the symbols at position i in the two strings, and $\delta(s_1[i], s_2[i])$ is a function that returns 0 if the symbols are the same and 1 if they are different [16].

Voting is done in two way.

1. Each of the k nearest neighbors votes for its class, and the class with the most votes is assigned to the new data point.
2. Weighted voting mechanism, where weights are assigned due to how far the distance is.

$$\text{class}(x) = \arg\max_{c \in C} \sum_{i=1}^{k} w_i \delta(x, y) \quad (20)$$

In order to choose the value of the number of nearest neighbors k, for example, between noise reduction and computational efficiency, the following trade-offs should be considered. In the case of a small k value the K-NN algorithm will be sensitive to noise and biased towards neighbouring points when classifying. However, selecting a large k value gives rise to computational ineffectiveness and may be antithetical to the naive principle of KNN, that is, points close together should usually have similar classes or densities. A simple and intuitive approach to selecting k is to set it equal to the square root of the number of data points n. This method balances both noise reduction and computational efficiency (size of the dataset are taken into account, while the algorithm

can still learn local patterns efficiently [6].

$$K = \sqrt{(n)} \tag{21}$$

2.7. Performances Comparison

For us to compare the performance of the models, we used the following: Accuracy, Sensitivity (Recall), Precision, Error Rate, True Positive Rate (TPR), False Positive Rate (FPR), AUC, ROC, Computational time and F1 Score.

Accuracy:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{22}$$

Sensitivity (Recall):

$$\text{Recall} = \frac{TP}{TP + FN} \tag{23}$$

Sensitivity denotes the count of actual positive instances that are correctly classified as positive.

Specificity

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{24}$$

Specificity represents the ratio of actual negative instances correctly classified as negative.

Precision:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{25}$$

Error Rate:

$$\text{ER} = \frac{FN + FP}{TP + FP + FN + TN} \tag{26}$$

True Positive Rate (TPR):

$$\text{TPR} = \frac{TP}{TP + FN} \tag{27}$$

False Positive Rate (FPR):

$$\text{FPR} = \frac{FP}{FP + TN} \tag{28}$$

F1 Score:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{29}$$

AUC:

$$AUC = \sum_{i=1}^{n-1} (FPR_{i+1} - FPR_i) \times \frac{(TPR_{i+1} + TPR_i)}{2} \tag{30}$$

2.8. Deployment

Model deployment in machine learning is the process of smoothly embedding a trained model into operational systems or applications and automating task execution or producing predictions [9]. In our particular scenario, our focus was to implement the best model selected from a set composed of Random Forest (RF), XGBoost, and K-Nearest Neighbors (KNN). This section comprehensively explores the step-by-step process of deploying the machine learning model, utilizing the popular framework Spring boot.

Leveraging Spring boot, a highly acclaimed framework for crafting interactive web applications with java, facilitates the deployment process [14]. With Spring boot we were able to create easy-to-use interfaces for our models so that the user can easily interact with and receive predictions from the deployed model. One of the major considerations in this deployment task is to serialize the trained model, setup a spring boot application, plug the model, in fact, and deploy it in a web server. In addition, proper data preprocessing, efficient user input processing, and understandable prediction output visualization are all equally important when deploying the model [5]. In the end, via a structured methodology and the use of tools such as Spring boot, we intended to effortlessly implement predictive capabilities of our machine learning models in real world applications, rendering end-user able to act upon results and predict in real-world applications. The deployment followed this process. Serialization involves converting the trained machine learning model into a format that can be easily stored and loaded, facilitating its retrieval from disk and loading into memory for making predictions without the need for retraining. After serialization, the configuration of the web application, library framework, spring boot, consists of the configuration of the basic structure, including routes, views and files, and thus the foundation, on which Hypertext Transfer Protocol requests are handled, and on which responses are constructed. Subsequently, embedding the serialized model in the web application simply entails getting the model into memory when the application starts and, it can be directly used for predictions. Developing an Application Programming Interface endpoint in the web application allows the submission of input data for prediction using Hypertext Transfer Protocol POST requests. After the input data is received, data preprocessing guarantees its conformity to the format of the training data, including the data scaling, normalization, or also categorical variable encoding. Based on the preprocessed input, the web app retrieves predictions from the deployed machine learning model and presents the results in an appropriate response format to the client for convenient application [6]. Finally, a web application processes the response, displaying the predictions to the user (i.e., displaying the predictions on a browser or retrieving predictions from the client in a different format).

3. Results

3.1. Selected Features

In the Lasso model, feature selection was guided by the absolute value of their coefficients.

Location “the side of the head in which patients perceive

pain from migraine" turned out to be the most relevant predictor with the largest coefficient, suggesting its importance for predicting the outcome variable (type of migraine). Other, e.g., "Diplopia", "Phonophobia", and "Paresthesia", also showed significant coefficients, indicating their high correlation to the type of migraine and helpfulness for the model. On the other hand, characteristics such as "Age of the patient", "Duration of headache", and "Number of headache episodes a day" achieved lower coefficients which implies less predictive role. However, these characteristics still provided informative supplementary data that could be useful to differentiate between various subtypes of migraine. Exhausted features, "Character", "Dysarthria" and "Ataxia", coefficients were given a coefficient of zero, as they do not contribute significantly to the predictive outcome of the presence of the chosen features or their effect was already captured in the model via another variable. By this selective process, the model is kept efficient and generalizable to unseen data.

Table 1. *Features Selected by Lasso.*

Rank	Feature	Coefficient
1	Location	10.8
2	Diplopia/Double vision	6.28
3	Phonophobia	5.98
⋮	⋮	⋮
18	Frequency	0.320
19	Duration	0.291
20	Age	0.0543

3.2. Fitted Models

3.2.1. Random Forest

Random Forest model was applied to a dataset to classify migraine based on various features, using 500 decision trees. With 4 variables randomly selected at each split and an Out-of-Bag (OOB) error estimate of 8.4%, the model demonstrates good performance. The OOB error rate, a reliable performance measure for Random Forests, indicates that the model is well-balanced, neither underfitting nor overfitting, and generalizes well to new data. It is helpful, so that 4 variables are tried at each split, to keep the diversity of the trees, so that correlation is reduced and prediction is made more robust.

Table 2. *Confusion Matrix for Random Forest Model: Out-of-Sample.*

Reference	0	1	2	3	4	5	6
Prediction							
0	3	1	0	1	0	0	0
1	0	3	0	0	0	0	0
2	0	0	21	0	1	0	0
3	0	0	0	2	0	0	0
4	0	0	0	0	1	0	0
5	0	2	0	1	5	72	0
6	0	0	0	0	0	0	6

3.2.2. K-Nearest Neighbors

The model was trained and tested by 10-cross-validation, a powerful method for performance prediction. In this study, the performance of the model fluctuated depending on the value of k. The maximum accuracy 86.55% was achieved with k 5, and the highest Kappa score 72.92% was achieved, indicating a strong agreement between the corresponding predicted class and the actual class. As k increased, the model's accuracy gradually decreased, with the lowest accuracy 73.41% observed at k 23 , where the Kappa value dropped to 42.10%. This trend is an indication that, fewer neighbors leads to better performance, as the model is increasingly sensitive to the local structure of the data. Nevertheless, when k is large, the model is relatively insensitive, adding noise as far from the considered neighbor, in this way decreasing the quality of the classification. Thus, the best value of K with which to train the model as a hyperparameter for this dataset is 5, which captures a compromise between accuracy and reliability.

Table 3. *Performance Summary of k-NN Across Different Values of k.*

k	Accuracy	Kappa
5	0.8655	0.7292
7	0.8145	0.6506
9	0.8152	0.6431
11	0.8078	0.6224
13	0.8011	0.6017
15	0.7837	0.5573
17	0.7696	0.5220
19	0.7697	0.5187
21	0.7590	0.4886
23	0.7341	0.4210

The k-Nearest Neighbors model was evaluated using the confusion matrix and overall statistics, as detailed in Table 4. This model classifies instances into seven distinct classes and achieved an overall accuracy of 86.66%. The confidence interval for this accuracy ranges from 79.09% to 92.12%, indicating a robust performance.

Table 4. *Confusion Matrix for K-Nearest Neighbors Model: Out-of-Sample.*

Reference	0	1	2	3	4	5	6
Prediction							
0	2	1	0	0	0	0	0
1	1	2	0	1	3	0	0
2	0	0	14	0	0	2	0
3	0	0	0	3	0	0	0
4	0	0	0	0	0	1	0
5	0	3	7	0	4	69	0
6	0	0	0	0	0	0	6

3.2.3. Extreme Gradient Boosting

The model was set up with a maximum tree depth of 3, ie. each sample in the ensemble was limited to three splits on each decision tree. Because this is a relatively shallow depth, there is the potential to avoid overfitting by having each tree identify broad, general patterns instead of fine interactions within the data. The model was trained for more than 100 iterations (boosting rounds), in which each iteration introduces a new tree so as to fix the errors made by the previous trees. The monotonic reduction from 1.2713 to 0.0344 of training log loss between these iterations show that the model learned the data and further enhanced its performance with each iteration. In this framework the complexity and the learning are combined, because the model converges to a good solution while staying generalizable.

Table 5. Training Log Loss Across Iterations.

Iteration	Training Log Loss
1	1.2713
2	0.9629
...	...
99	0.0346
100	0.0344

Table 6. Confusion Matrix for XGBoost Model: Out-of-sample.

Reference	0	1	2	3	4	5	6
Prediction							
0	2	0	0	1	0	0	0
1	1	4	0	0	0	0	0
2	0	0	19	0	1	0	0
3	0	0	0	2	0	0	0
4	0	0	0	0	2	0	0
5	0	2	2	1	4	72	0
6	0	0	0	0	0	0	6

XGBoost model was evaluated using the confusion matrix as detailed in Table 6. This model classifies instances into seven distinct classes and achieved an overall accuracy of 92.44%. The confidence interval for this accuracy ranges from 83.05% to 94.68%, indicating a strong performance.

3.3. Performance Evaluation

XGBoost model demonstrated the highest accuracy of 92.4%. This architecture is well known for its efficiency, its scalability¡[a]especially for sparse data, and a capacity to ingest complex relationships between features. The boosting method is used to further reduce errors by correcting errors made by the models that preceded it and thus results in high performing system.

Table 7. Accuracies and 95% Confidence Intervals for Different Models.

Model	Accuracy	95% CI
XGBoost	0.924	(0.8305, 0.9468)
Random Forest	0.916	(0.8406, 0.9529)
K-Nearest Neighbors (KNN)	0.866	(0.7909, 0.9212)

The accuracy of the Random Forest model is 91.6%, which indicates a strong performance in classifying migraine types. This model's ability to handle large datasets with numerous features, along with its robustness to overfitting due to the ensemble approach, contributes to its high accuracy.

K-Nearest Neighbors model resulted in an accuracy of 86.6%. The performance is greatly reliant on the value of k, the number of neighbors, and the distance metric that is used, which may account for the relatively lower accuracy.

Table 8. Performance Metrics for Different Models.

Model	TPR	Spec.	Prec.	F1	FPR	Time (s)
XGBoost	0.9224	0.9841	0.8059	0.9165	0.0159	2.08
RF	0.8645	0.9723	0.8975	0.9049	0.0277	4.65
KNN	0.7932	0.9624	0.7042	0.8053	0.0376	9.51

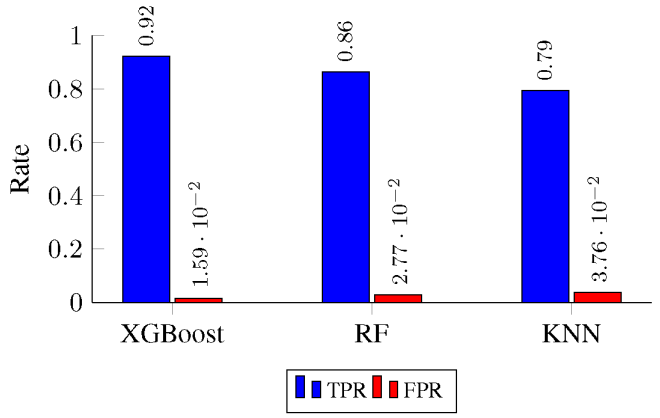

Figure 1. True Positive Rate (TPR) vs False Positive Rate (FPR) for XGBoost, Random Forest (RF), and K-Nearest Neighbors (KNN).

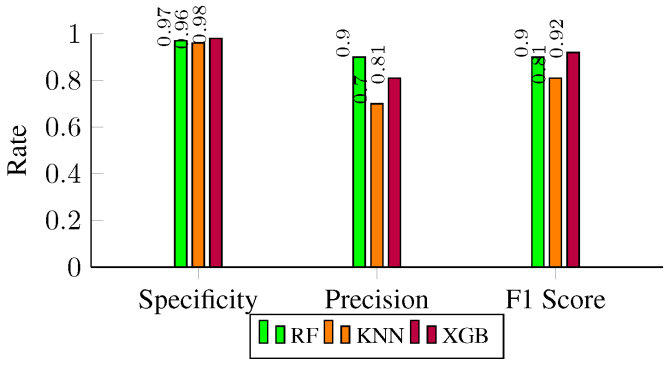

Figure 2. Comparison of Specificity, Precision, and F1 Score for Random Forest (RF), K-Nearest Neighbors (KNN), and XGBoost (XGB).

In the case of data imbalance and its effects on model performance, evaluation metrics for the XGBoost, Random Forest, and K-Nearest Neighbors were studied. The model evaluation criteria given the imbalance demanded that accuracy alone cannot accurately measure the efficacy of the models, so other indicators of performance, sensitivity (TPR), specificity, precision, F1 score, false positive rate (FPR), and computation time were considered. XGBoost outperformed in identifying the true positive cases with extremely fewer false positives of 1.59%, holding together with the highest sensitivity of 92.24% and specificity of 98.41%. Random Forest performed a little higher on FPR with 2.77%, showing values of 86.45% for sensitivity and a good precision of 89.75%. The K-Nearest Neighbors performed poorly since it exhibited the lowest sensitivity of 79.32% and precision of 70.42%. Furthermore, there was a significant disparity in computation time, with XGBoost having the least computation time among the three models.

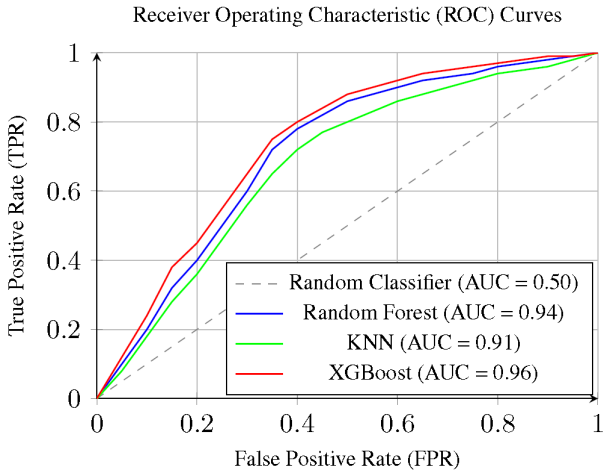

Figure 3. *Receiver Operating Characteristic (ROC) curves comparing the performance of Random Forest, K-Nearest Neighbors, and XGBoost classifiers..*

The models performance was also evaluated using a Receiver Operating Characteristic (ROC) curve, which represents the performance of the model across different threshold values graphically. It gives a fair amount of trade-off between sensitivity and specificity and is particularly useful for imbalanced datasets. The ROC curve with the highest area under it suggests a model that has better classification ability over the data. The classification of migraine types by the XGBoost model outperformed the rest, achieving an AUC of 96%, Random Forest achieved 94%, with K-Nearest Neighbors slightly behind at 91%.

4. Discussion

Performance of three machine learning models (Random Forest, K-Nearest Neighbors, and XGBoost) were evaluated based on a dataset of migraine symptoms and patient parameters. Comparison of three models were made on the basis of metrics such as Accuracy and Sensitivity/Specificity/Precision, F1 Score and computation time. XGBoost model showed the optimal overall performance. This model is especially effective at predicting positive and negative cases correctly, with the best balance and strength in predicting F1 score. Its excellent performance is due to its boosting process, which is very effective at modeling complex relationships as well as reducing errors.

Random Forest model also performed well, although it is slightly less sensitive and specific than XGBoost, it is still a valuable tool, particularly its high Precision and balanced performance. Its ensemble approach contributes to its robustness and reliability.

On the other hand, K-Nearest Neighbors model yielded the least favorable results among the three models. The lower performance is due, in particular, to the adopted voting metric, since it was a challenge to use the weighted voting method because the outcome variable was categorical.

Various studies have examined the performance of the three machine learning models on different diseases with quite different results depending on the dataset, features applied, and model hyperparameters. For instance, it was previously demonstrated that XGBoost excels in handling imbalanced breast cancer datasets and in identifying subtle interactions amongst features [7], which is also what we experienced. Also, there have been research work to demonstrate that deep learning models such as CNNs and LSTMs can outperform traditional ensemble methods in some situations [6]. Although our findings shows KNN being worse compared to XGBoost and Random Forest, research also proves that with careful optimization through techniques like feature scaling and dimensionality reduction, KNN can perform even better [16]. These comparisons highlight the need for thoughtful choice of these models depending on specific contexts and underscore the continued need for research into advancements in migraine classification methodologies.

In general, this work has detected XGBoost as the best model for migraine classification and Random Forest as the second best model. Despite the lower accuracy, K-Nearest Neighbors continues to be a potential choice depending upon the application and requirements of a classification task. This comparison offers insight into the advantages and disadvantages of each model and informs their use for performance considerations.

Despite promising results, this study has limitations; Model interpretability is limited, as feature importance was not extensively analyzed. Class imbalance in the dataset may have affected model performance. Future work should concentrate on the investigation of hybrid models like XGBoost-Recurrent Neural Networks and advanced approaches like Fuzzy Logic for better accuracy and robustness. Implement techniques like Shapley Additive Explanations (SHAP) values to refine feature selection and improve model interpretability. Apply methods like Synthetic Minority Over-sampling Technique to ensure balanced data for classification.

5. Conclusion

Through the comparative study, we proposed Random Forest, k-Nearest Neighbors and XGBoost as migraine classification model, and among them, we suggest XGBoost as the most ideal model due to the highest performance in terms of Accuracy, Sensitivity, Specificity, Computation time and F1 score. Due to its boosting technique, it is robust to the complex relation and has high overall performance. Random Forest, however, is a more powerful alternative providing both high Precision and balanced performance, which is appropriate for tasks where robustness and interpretability play a primary role. KNN had the least effective performance, which can be improved by managing class imbalance. The choice of model should align with the specific requirements and constraints of the application.

Abbreviations

LASSO	Least Absolute Shrinkage and Selection Operator
ML	Machine Learning
RSS	Residual Sum of Squares
OLS	Ordinary Least Squares
LARS	Least Angle Regression
KNN	K-Nearest Neighbors
RF	Random Forest
OOB	Out-of-Bag
TPR	True Positive Rate
FPR	False Positive Rate
AUC	Area Under the Curve
ROC	Receiver Operating Characteristic
SHAP	Shapley Additive Explanations
SMOTE	Synthetic Minority Over-sampling Technique

Acknowledgments

I herein extend my deepest appreciation to all those involved in the success of this research. I offer special thanks to my mentors and advisors for their constructive criticism, critical input, and unwavering support during this study. I also appreciate the stakeholders and organizations that helped provide the resources and information needed for model development and validation. Last but not least, I would like to thank my family, friends, and colleagues for their inspiration and support, which were crucial for the completion of this migraine classification research.

References

[1] Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.

[2] Burton, W. N., Landy, S. H., Downs, K. E., & Runken, M. C. (2009). The impact of migraines and the effect of migraines treatment on workplace productivity in the United States and suggestions for future research. In Mayo Clinic Proceedings (Vol. 84, pp. 436-445). https://doi.org/10.4065/84.5.436

[3] Chardet, M., Coullon, H., Pertin, D., & Pérez, C. (2018). Madeus: A formal deployment model. In 2018 International Conference on High Performance Computing & Simulation (HPCS) (pp. 724-731). https://doi.org/10.1109/HPCS.2018.00118

[4] Choudhary, K., DeCost, B., Chen, C., Jain, A., Tavazza, F., Cohn, R., et al. (2022). Recent advances and applications of deep learning methods in materials science. npj Computational Materials, 8(159).

[5] Diaby, T., & Rad, B. B. (2017). Cloud computing: A review of the concepts and deployment models. International Journal of Information Technology and Computer Science, 9(6), 50-58. https://doi.org/10.5815/ijitcs.2017.06.07

[6] Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN model-based approach in classification. In On the move to meaningful internet systems 2003: Coopis, doa, and odbase: Otm Confederated International Conferences, Coopis, Doa, and Odbase 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings (pp. 986-996).

[7] Hassan, M. M., Hassan, M. M., Yasmin, F., Khan, M. A. R., Zaman, S., Islam, K. K., et al. (2023). A comparative assessment of machine learning algorithms with the Least Absolute Shrinkage and Selection Operator for breast cancer detection and prediction. Decision Analytics Journal, 7, 100245. https://doi.org/10.1016/j.dajour.2023.100245

[8] Larsson, J. (2024). Optimization and algorithms in sparse regression: Screening rules, coordinate descent, and normalization. Lund University.

[9] Liang, P., Song, B., Zhan, X., Chen, Z., & Yuan, J. (2024). Automating the training and deployment of models in MLOps by integrating systems with machine learning. https://doi.org/10.48550/arXiv.2405.09819

[10] Mahović, D., Bračić, M., & Jakuš, L. (2021). Diagnostic criteria and classification of migraine. Medicus, 301, Migrena, 39-44.: https://hrcak.srce.hr/257514

[11] Miao, Y., Wang, J., Zhang, B., & Li, H. (2022). Practical framework of Gini index in the application of machinery fault feature extraction. Mechanical Systems and Signal Processing, 165, 108333. https://doi.org/10.1016/j.ymssp.2021.108333

[12] Qi, Y. (2012). Random forest for bioinformatics. Ensemble machine learning: Methods and applications, 307-323.

[13] Rigatti, S. J. (2017). Random forest. Journal of Insurance Medicine, 47(1), 31-39.

[14] Selvaraj, S. (2024). Building RESTful APIs with Spring Boot (Java). In Mastering REST APIs: Boosting Your Web Development Journey with Advanced API Techniques (pp. 291-347). Springer. https://doi.org/10.1007/979-8-8688-0309-3-7

[15] Shaik, N. B., Jongkittinarukorn, K., & Bingi, K. (2024). XGBoost based enhanced predictive model for handling missing input parameters: A case study on gas turbine. Case Studies in Chemical and Environmental Engineering, 10, 100775. https://doi.org/10.1016/j.cscee.2024.100775

[16] Wang, Y., & Wang, Z.-O. (2007). A fast KNN algorithm for text categorization. In 2007 International Conference on Machine Learning and Cybernetics (Vol. 6, pp. 3436-3441). https://doi.org/10.1109/ICMLC.2007.4370742

[17] Zhang, S., Li, X., Zong, M., Zhu, X., & Cheng, D. (2017). Learning k for KNN classification. ACM Transactions on Intelligent Systems and Technology (TIST), 8(3), 1-19. https://doi.org/10.1145/2990508

[18] Zhou, X., Lu, P., Zheng, Z., Tolliver, D., & Keramati, A. (2020). Accident prediction accuracy assessment for highway-rail grade crossings using random forest algorithm compared with decision tree. Reliability Engineering & System Safety, 200, 106931. https://doi.org/10.1016/j.ress.2020.106931