**Department of Electrical and Computer Engineering**

**North South University**



**CSE445 Report**

**Dengue Prediction Using Machine Learning**

**Minhazur Rahman ID # 2211255042**

**Faculty:**

**Riasat Khan**

**Assistant Professor**

**ECE Department**

**Spring, 2025**

Dengue Prediction Using Machine Learning

Minhazur Rahman   
Electrical and Computer Engineering  
North South UniversityDhaka, Bangladesh  
minhazur.rahman12@northsouth.edu

*Abstract*-Dengue is a rapidly spreading viral disease caused by the Aedes mosquito. It is one of the major health concerns in Bangladesh. Early detection is necessary to reduce severe adverse outcomes and the mortality rate. Our project aims to use clinical data to develop a machine learning-based classification model to predict dengue infection. We performed exploratory data analysis to clean and preprocess the dataset by removing duplicate values. We used label encoding to deal with categorical variables and applied feature scaling. The dataset was split into 80% for training and 20% for testing. Several machine learning models were used, including individual classifiers such as Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Naive Bayes. Ensemble techniques, including stacking and blending, increased predictive performance and model stability. The Random Forest classifier achieved the highest accuracy of 82.03 % and F1 score of 0.80.

Keywords—dengue prediction, machine learning, classification, ensemble techniques, random forest

# Introduction

Dengue is a viral disease caused by the bites of an infected Aedes mosquito. The main cause of dengue is the dengue virus, which spreads by mosquito bites. These mosquitoes breed in standing water or urban areas, especially during warm, humid weather. [1] Every year, around the world, about 400 million people are infected with dengue, and around 40000 people die from it. In Bangladesh, around 320000 cases of dengue were reported, and 1705 people died from it. [2] There are three grades of dengue: Grade I Dengue fever is mild, Grade II Dengue Hemorrhagic Fever (DHF) is more severe, and Grade III Dengue Shock Syndrome (DSS) is the most serious. [3] Through physical exams and blood tests, doctors detect dengue. These Dengue symptoms can be managed by resting and taking fluids and paracetamol to reduce fever. [1][2][3] However, there is no cure for dengue.

Gupta et al. [4] used machine learning algorithms to create a diagnostic model for quick detection of dengue. The authors collected data from the DengAI competition, which consisted of data from San Juan and Iquitos. The dataset had 1872 samples, and the mean method was used to replace missing data. They used a 10-K-fold cross-validation technique to measure accuracy and found that random forest had the highest accuracy score of 8.72.

Nusrat [5] and colleagues detected an early stage of dengue fever using various ensemble methods based on machine learning algorithms. They made a custom-built dataset of 320 samples and 14 hematological features from two hospitals in Dhaka, where the ratio was 5:3 (male to female). One-hot encoding, synthetic oversampling, and removal of redundant features are used in this work. From various ensemble learning and transformer-based models, the best result was obtained by stacking an ensemble classifier, with an F1 score of 0.9646 and an accuracy of 96.88%.

Qaiser [6] and colleagues developed machine learning model to diagnose dengue in its early stage using clinical data. The authors studied a dataset containing 300 dengue patients in Pakistan, from August to October of 2023. The used dataset needed cleaning. For the missing data, they used mean and mode values. The cleaned dataset was used to train and test machine learning models. The SVM model achieved 71.4% accuracy and 97.4% recall, making it the most efficient model.

Bianca [7] and her colleagues employed machine learning for dengue case screening using a dataset sourced from SINAN, consisting exclusively of complete case records. To ensure a balanced dataset, they arbitrarily selected 10,000 positive cases and 10,000 negative cases of dengue while also cleaning and classifying the data to accommodate calculation boundaries. Additionally, they applied data normalization and feature selection during the preprocessing phase. The authors tested several models, including KNN, logistic regression, decision trees, and MLP. All models achieved comparable accuracy, nearing 93%.

We propose a machine learning–based system for the early detection of dengue in Bangladesh, where annual outbreaks continue to take thousands of lives. As there is no cure for dengue, quick detection is necessary so patients can get proper medication. Our approach begins with collecting routine hematological data, followed by data preprocessing: removing duplicate values, outlier detection, and encoding categorical variables. We will train and compare several classifiers, including Random Forest, Support Vector Machine, and several ensemble techniques using standard metrics (accuracy, F₁-score, recall). We will apply LIME to explain the model's prediction to increase confidence in the best model.

The proposed system is discussed in Section II, with appropriate tables and figures. It also includes details of the models used and the project flowchart. The results of the project are discussed in Section III. Finally, Section IV Concludes the paper with discussions on how to improve the prediction capabilities of our models.

# Proposed System

## Dataset

We used a public dataset available on Mendeley Data called Dengue Fever Hematological Dataset: Clinical Insights for Improved Diagnosis and Patient Management [8]. The dataset contains detailed hematological data of patients diagnosed with dengue fever. The dataset has 18 features: gender, age, hemoglobin, neutrophils, lymphocytes, monocytes, eosinophils, RBC, HCT, MCV, MCH, MCHC, RDW-CV, total platelet count, MPV, PDW, PCT, and total WBC. There are 1523 samples, with 1042 positive samples (68%) and 482 negative samples (32%).

Table I shows us the mean, minimum, and maximum values of some important features of the dataset. Here, the values help us to identify the range of typical and extreme values.

Table I. Mean, minimum, and maximum of some features of the dengue public dataset

|  |  |  |  |
| --- | --- | --- | --- |
| Features | Minimum | Maximum | Mean |
| Total Platelet Count (/cumm) | 56,000 | 299,803 | 173,127.17 |
| Total WBC count (/cumm) | 3,500 | 14,900 | 5,483.02 |
| Age | 5 | 78 | 40.31 |
| PCT (%) | 0.00002 | 234.0 | 0.29 |
| MCV (fl) | 80.0 | 100.0 | 89.82 |

Fig. 1. Shows the Violin Plot of HCT (Hematocrit), Total Platelet Count and Hemoglobin. Violin plot shows how spread out the dataset is. It shows the difference between positive and negative dengue cases. For HCT, the distributions for positive and negative dengue cases are very similar meaning HCT does not influence dengue outcomes. For Total Platelet Count, the positive group shows a wider range distribution and the negative group has a more centered higher distribution meaning dengue-positive patients tend to have lower platelet counts. For Hemoglobin, the distribution for positive cases is slightly higher than negative cases meaning Hemoglobin does not influence dengue outcomes.

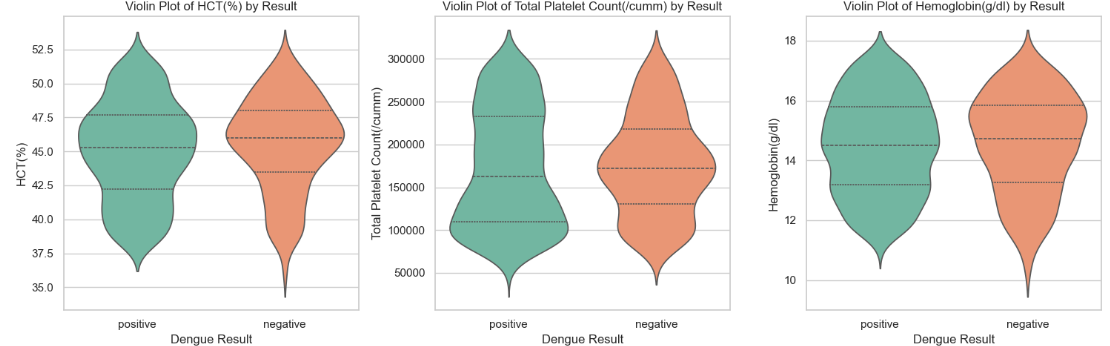


Fig. 1. Violin Plot for HCT, Total Platelet Count, and Hemoglobin

Fig. 2. Shows Histograms for all the features. Histogram helps to visualize the distribution of numerical data. For the age feature, the range is 0 to 80, with multiple peaks between 20 to 60, which shows that the dataset has a diverse range of data. This process is followed for all the features of the dataset.

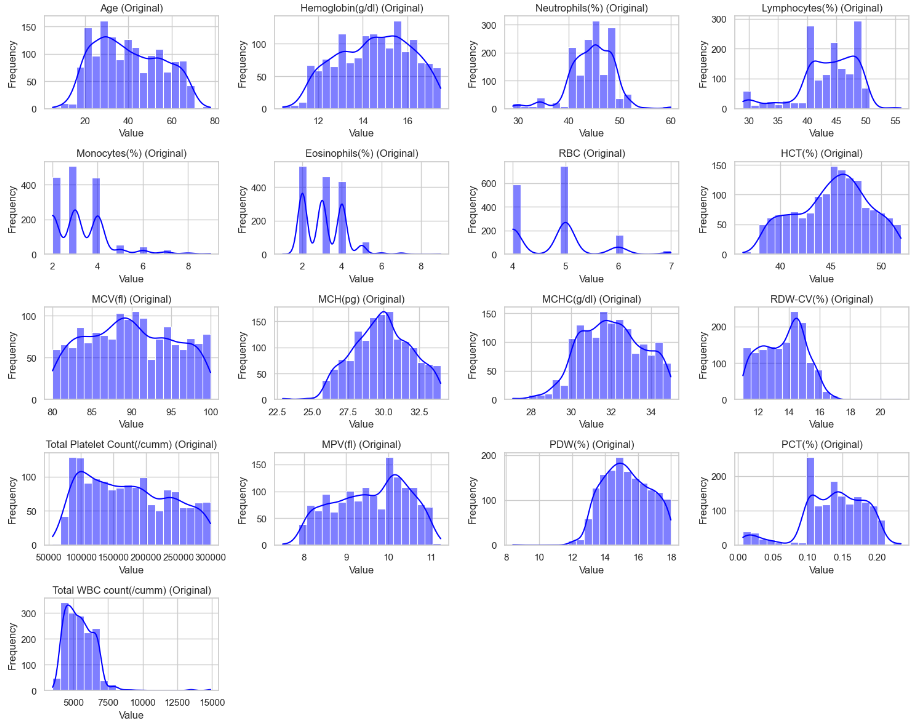


Fig. 2. Histograms for all features

Fig. 3 shows boxplot of all the continuous feature of our dataset. A boxplot can be used for outlier detection. Fig. 3. identifies the outliers of every feature. In the age feature, no outlier is present, but in the neutrophils feature, some outliers are present, as we can see in this figure.

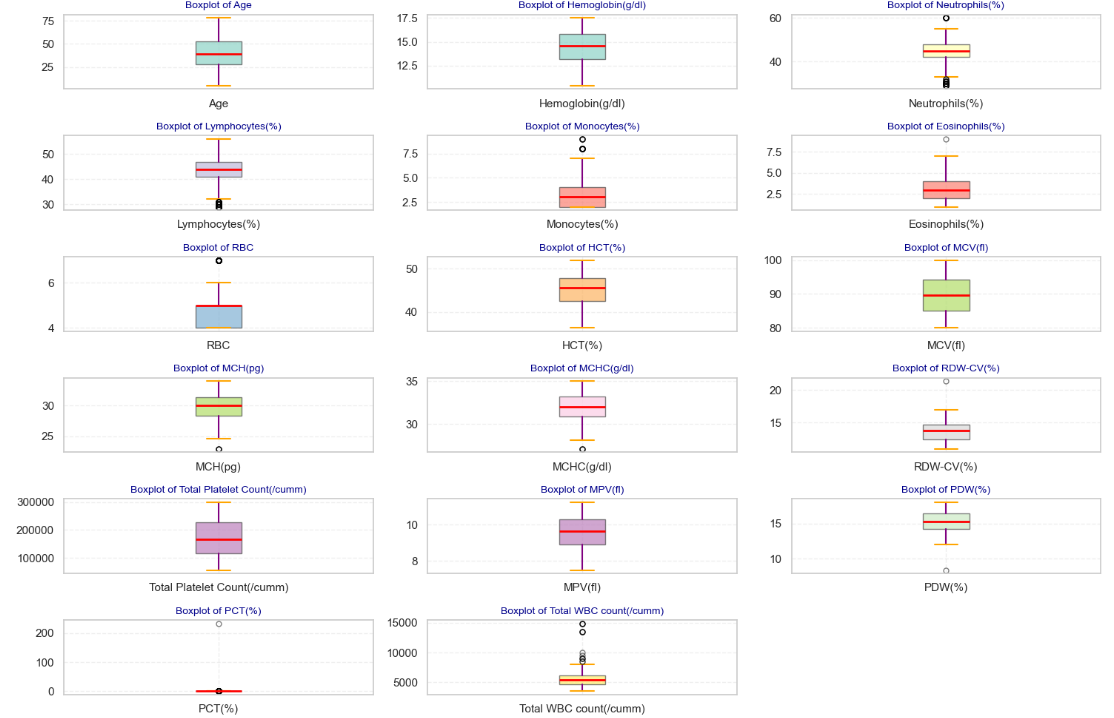


Fig. 3. Boxplot for all continuous features

Fig. 4. shows the KDE plot of different features. In KDE Plot for MCV vs RDW-CV, high data concentration is represented by redder areas, and light areas indicate unusual data. In KDE Plot for Age vs Hemoglobin, high data concentration is represented by bluer areas, and light areas indicate unusual data. In KDE Plot for MCH vs MCHC, high data concentration is represented by greener areas, and light areas indicate unusual data.

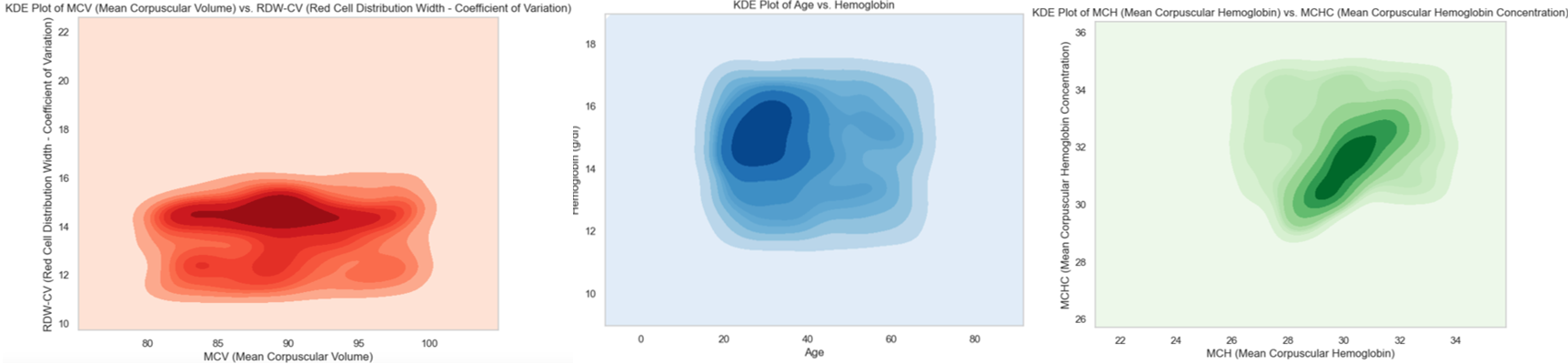


Fig. 4. KDE Plot for MCV vs RDW-CV, Age vs Hemoglobin, and MCH vs MCHC

## Dataset Preprocessing

We processed the dataset to find missing and duplicate values. There were no missing values; after dropping all duplicate samples, we had 1511 samples. Boxplot was used for outlier detection, and min-max scaling was used for feature scaling. None of the curves from our histograms were bell-shaped, which indicated that feature selection was needed. However, using Pearson’s correlation coefficient, we obtained Figure 5.

Fig. 5. showed no value was close to 0.9 or 1, so there is no perfect positive linear relationship. So, we didn't use any feature selection technique. The only categorical data in the dataset was gender, and we encoded it using label encoding, with male being 0 and female being 1.

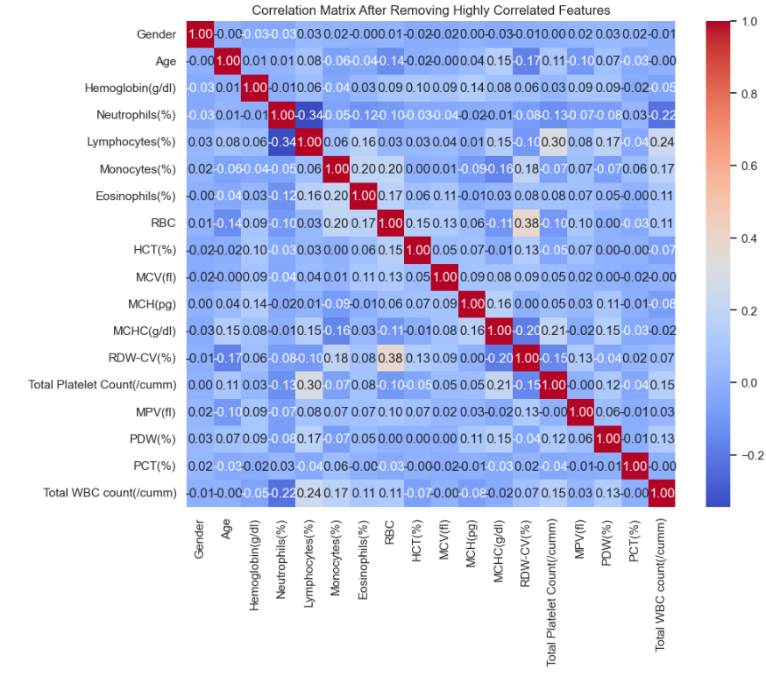


Fig. 5. Pearson Correlation Matrix

## Machine Learning Models

* **Random forest:** Random Forest is an ensemble method that uses many decision trees to make an accurate prediction. It works by combining the results of each tree to get a more reliable outcome. Important hyperparameters include the number of trees in the forest, the maximum number of features, the maximum depth of each tree, and the minimum number of samples required to split into a leaf. Tuning these hyperparameters helps optimize the model’s performance.

(1)

(2)

i = frequency

i = class i

c = total number of classes

t = node t

* **Decision tree:** Decision tree is a machine learning model that predicts by asking a series of yes/no questions. It works by splitting the data into smaller parts based on the answers to these questions until it reaches a final decision. Important hyperparameters include the maximum depth of the tree, the minimum number of samples needed to split a node, and the minimum number of samples to be a leaf node. Tuning these hyperparameters helps optimize the model’s performance.
* **AdaBoost:** AdaBoost is a machine learning method that combines many simple models to make one strong model. It works by training models one after another, where each new model focuses more on the mistakes made by the previous ones. Important hyperparameters include several estimators (models used) and the learning rate (controls how much each model affects the final result).
* **XGBoost:** XGBoost is a machine learning method that is used for classification and regression problems. It works by building many decision trees. Each new tree tries to correct the mistakes made by the tree before. Important hyperparameters include the number of trees, the maximum depth of the tree, and the learning rate (controls how fast the model learns).
* **GradientBoost:** Gradient Boosting is a machine learning method that builds a strong model by combining many small and simple models. It works by adding new models that have already corrected the errors made by the model used before. Important hyperparameters include the number of trees, the maximum depth of the tree, and the learning rate (controls how fast the model learns).
* **Support Vector Machine (SVM):** Support Vector Machine (SVM) is a machine learning model that classifies data into different categories. It works by drawing a decision boundary called a hyperplane that divides the classes, aiming to leave the widest margin between the closest points from each class, which are called supportvectors. Important hyperparameters include the regularization parameter (C), the degree of the kernel function, the type of kernel, and the gamma for non-linear problems.

(3)

w = weight vector

x = input vector

b = bias

* **Logistic Regression:** Logistic Regression is a machine learning model that uses a probability-based predictive analysis algorithm. It works by calculating the probability of a given input belonging to a particular class and picking the class with the highest chance. Important hyperparameters include C (controls how much the model tries to avoid overfitting), solver (decides which algorithm is used to find the best model weight), maximum number of times the algorithm will try to learn from the data, and penalty (sets the type of regularization).

(4)

x = input feature

(5)

θ = model parameter

m = number of samples

i = class i

y(i)= true label of i

p(i) = predicted probability of i

* **Naive Bayes:** Naive Bayes is a machine learning model used mainly for classification tasks. It uses probabilities of an event occurring based on the features, assuming each feature is independent. Important hyperparameters include var\_smoothing, which helps prevent errors by adding a small value to the variance in calculations.

(6)

A, B = event

P(A), P(B) = independent probability of A, B

P(A|B) = probability of A given B is true  
P(B|A) = probability of B given A is true

* **K- Nearest Neighbor:** K-Nearest Neighbors (KNN) is a lazy machine learning algorithm for classification and regression. It works by looking at the “k” closest data points to a new point and predicting the result based on the majority (for classification) or average (for regression) of those neighbors. Important hyperparameters include k (number of neighbors), weighting scheme (near or far), and distance (Euclidean or Manhattan).

(7)

(8)

1, x2 = x values of graph

y1, y2 = y values of the graph

* **Blending:** Blending is an ensemble technique that combines the predictions of multiple different models to improve overall performance. It works by training several base models and using another model (a blender) to learn how to mix their predictions best. Important hyperparameters include several estimators (base models), the final estimator (meta model), and CV (number of cross-validation folds).
* **Stacking:** Stacking is an ensemble learning method that combines multiple models to make better predictions. It works by training several models (called base learners) and then using a final model (called a meta-learner) to learn from their outputs. Important hyperparameters include several estimators (base models), the final estimator (meta model), and CV (number of cross-validation folds).



Fig. 6. Working sequences of the proposed dengue prediction system

Fig. 6. illustrates the working sequences of the proposed dengue prediction system. Data is processed to drop duplicates and outliers. Min-max scaling and label encoding are done. The dataset is split into train and test samples using a stratified train-test split with an 80:20 ratio. Different models are trained so that their ability to predict dengue successfully can be compared. Models are evaluated based on their test and train accuracy, precision, recall, F1 score, AUC-ROC, and confusion matrix. SMOTE was used for dataset sampling, and LIME, an explainable AI, is used to explain the decisions taken by the best model.

# Results and Discussion

This section represents the results of machine learning models for detecting dengue. Accuracy, precision, recall, and F1 score are important performance metrics in this section.

(9)

(10)

(11)

(12)

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

Table II represents the hyperparameter value range and the optimized values of all the models we used. The optimized hyperparameters in GridSearch and RandomSearch help to achieve the best performance.

Table II. HYPERPARAMETER VALUES’ RANGES FOR VARIOUS ML MODELS

|  |  |  |
| --- | --- | --- |
| Model | Hyperparameter Value Range | Optimized value |
| Random Forest | estimators: [100,300,50],  max\_features: ['sqrt', 'log2'],  max\_depth: [10, 50, 100],  min\_samples\_split: [2,5,10],  min\_samples\_leaf: [1,2,4],  criterion: ['gini', 'entropy'] | 'criterion': 'gini', 'max\_depth': 100, 'max\_features': 'sqrt', 'min\_samples\_leaf': 4, 'min\_samples\_split': 10, 'n\_estimators': 300 |
| Decision Tree | criterion: ['gini', 'entropy'],  max\_depth: [10,50,100, None],  min\_samples\_split: [2,5,10],  min\_samples\_leaf: [1,2,4],  max\_features: ['sqrt', 'log2', None] | 'criterion': 'gini', 'max\_depth': 10, 'max\_features': 'sqrt', 'min\_samples\_leaf': 1, 'n\_samples\_split': 10 |
| AdaBoost | estimator: [dt\_stump],  n\_estimators:[50,100,200,300, 500],  learning\_rate: [0.01, 0.1, 1.0], algorithm: ['SAMME',  'SAMME.R'] | algorithm: 'SAMME',  estimator:DecisionTreeClassifier(max\_depth=10, random\_state=42),  learning\_rate: 0.1,  n\_estimators: 500 |
| XGBoost | n\_estimators: [50,100,200,  300],  max\_depth: [3, 5, 7, 10],  learning\_rate: [0.01,0.1,0.2],  subsample: [0.6,0.8,1.0], colsample\_bytree': [0.6,0.8,1.0] | colsample\_bytree:0.6,  learning\_rate: 0.1,  max\_depth: 3,  n\_estimators: 50,  subsample': 0.6 |
| Stacking using RF, XGBoost, AdaBoost, and Logistic Regression | Using the same as other hyperparameters in that table |  |
| Blending model using Rf, GradientBoost, Logistic Regression, and XGboost | Using the same as other hyperparameters in that table | [RF] Best Params: {'max\_depth': 5,'n\_estimators': 100}  [GB] Best Params: {'learning\_rate':0.1,'max\_depth': 3, 'n\_estimators': 100}  [LR] Best Params:'clf\_\_C': 1.0}  [XGB] Best Params: {'learning\_rate': 0.1, 'max\_depth': 5, 'n\_estimators': 100} |
| SVM | C: [0.1,1,10,100],  gamma: [0.001,0.01,0.1,1],  kernel: ['rbf', 'linear', 'poly'] | C: 1, gamma: 0.1,  kernel: 'rbf' |
| Logistic Regression | penalty: ['l1', 'l2'],  C: [0.01, 0.1, 1, 10, 100] | C: 0.01, penalty: l2 |
| Naive Bayes | var\_smoothing:  np.logspace(-10, -2, 9) | var\_smoothing: 1e-10 |
| KNN | n\_neighbors: [3, 5, 7, 9, 11],  weights: ['uniform', 'distance'],  metric: ['euclidean',  'manhattan'] | metric: 'manhattan',  n\_neighbors: 9,  weights: 'distance' |
| Blending Ensemble (KNN + NB → RF) | Using the same as other hyperparameters in that table | Using the same hyperparameters as default |
| Blending Ensemble (KNN + NB → SVM) | Using the same as other hyperparameters in that table | Using the same hyperparameters as default |

Table III shows the performance metrics of the various machine learning models evaluated for dengue prediction without optimizing the hyperparameters. The SVM model achieved the highest accuracy of 78.89% with a precision of 0.74 and an F1 score of 0.71. Other significant performances include the Random Forest Classifier with an accuracy of 78.03%, a precision of 0.79, and an F1 score of 0.75.

Table III. PERFORMANCE METRICS OF VARIOUS ML MODELS WITH DEFAULT HYPERPARAMETERS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy  (%) | Precision | Recall | F1-score |
| Random Forest | 78.03 | 0.79 | 0.78 | 0.75 |
| Decision Tree | 65.25 | 0.68 | 0.65 | 0.66 |
| AdaBoost | 76.07 | 0.76 | 0.76 | 0.73 |
| XGBoost | 74.43 | 0.73 | 0.74 | 0.72 |
| Stacking | 77.70 | 0.80 | 0.78 | 0.74 |
| Blending | 77.38 | 0.78 | 0.77 | 0.75 |
| SVM | 78.89 | 0.74 | 0.75 | 0.71 |
| Logistic Regression | 74.630 | 0.73 | 0.73 | 0.69 |
| Naive Bayes | 72.13 | 0.72 | 0.72 | 0.66 |
| KNN | 71.14 | 0.69 | 0.71 | 0.68 |
| Blending Ensemble (KNN + NB → RF) | 71.85 | 0.73 | 0.72 | 0.69 |
| Blending Ensemble (KNN + NB → SVM) | 73.77 | 0.74 | 0.74 | 0.69 |

Table IV shows the performance metrics of the various machine learning models evaluated for dengue prediction after optimizing the hyperparameters. The Random Forest classifier optimized by GridSearchCV achieved the highest accuracy of 82.30% with a precision of 0.82 and an F1 score of 0.80, which indicates a significant improvement compared to its performance with default hyperparameters (78.03%). The accuracy of stacking also improved from 77.70% to 80.98%.

Table IV. PERFORMANCE METRICS OF VARIOUS ML MODELS WITH OPTIMIZED HYPERPARAMETERS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy  (%) | Precision | Recall | F1-score |
| Random Forest | 82.30 | 0.82 | 0.82 | 0.80 |
| Decision Tree | 77.38 | 0.75 | 0.77 | 0.75 |
| AdaBoost | 72.79 | 0.72 | 0.73 | 0.72 |
| XGBoost | 79.02 | 0.80 | 0.79 | 0.76 |
| Stacking | 80.98 | 0.80 | 0.81 | 0.78 |
| Blending | 78.03 | 0.79 | 0.78 | 0.75 |
| SVM | 74.75 | 0.74 | 0.75 | 0.71 |
| Logistic Regression | 74.75 | 0.75 | 0.75 | 0.71 |
| Naive Bayes | 72.13 | 0.72 | 0.72 | 0.66 |
| KNN | 74.42 | 0.73 | 0.74 | 0.71 |
| Blending Ensemble (KNN + NB → RF) | 64.26 | 0.64 | 0.64 | 0.64 |
| Blending Ensemble (KNN + NB → SVM) | 73.44 | 0.73 | 0.73 | 0.69 |

Fig. 7. Shows the confusion metrics for the best-performing Random Forest classifier using GridSearch. The model can accurately classify 33 instances of class 0 and 217 instances of class 1. There are only nine misclassifications of class 1 and 47 misclassifications of class 0, which demonstrates the effectiveness of the dengue prediction model.

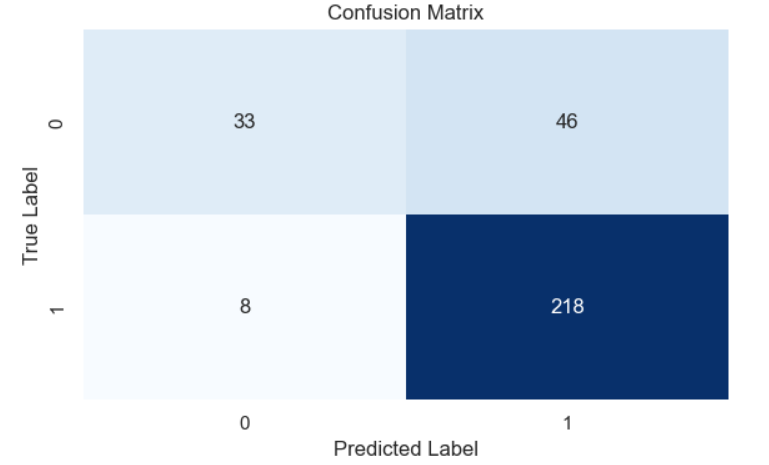


Fig. 7. Confusion Matrix of the best-performing model, Random Forest

Fig. 8. Shows the ROC curve for the best-performing Random Forest classifier. The x-axis represents the False Positive Rate (FPR) and the y-axis shows the True Positive Rate (TPR). Here, AUC tells us how good Random Forest model is at telling the difference between positive and negative dengue cases. Our AUC is 0.73. If AUC is 0.5, then it is considered that model is random guessing. If AUC is 1.0, then it is considered that model is perfect at prediction. Our AUC is 0.73 which is considered acceptable.

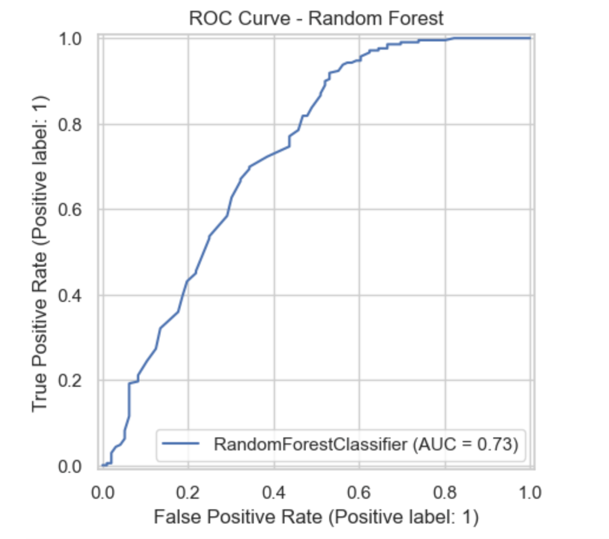


Fig. 8. ROC Curve of the best-performing model, Random Forest

Local Interpretable Model-agnostic Explanation (Lime) based Explainable AI (XAI) is used to explain how the Random Forest model predicts a result. This framework approximates the outcomes locally, making it more efficient. For example, the change in the data input uses the simple model to see whether the change in input data affects the result. It helps to understand why the main model predicts whether a patient is dengue positive or negative.

Fig. 9. demonstrates the confidence score of 69% for the positive case, where neutrophils, PCT, monocytes, RDW-CV, MCV, and RBC are the most impactful features and played a significant role in determining the positive class. However, the total WBC count and total platelet count played a prominent role in classifying the negative class.

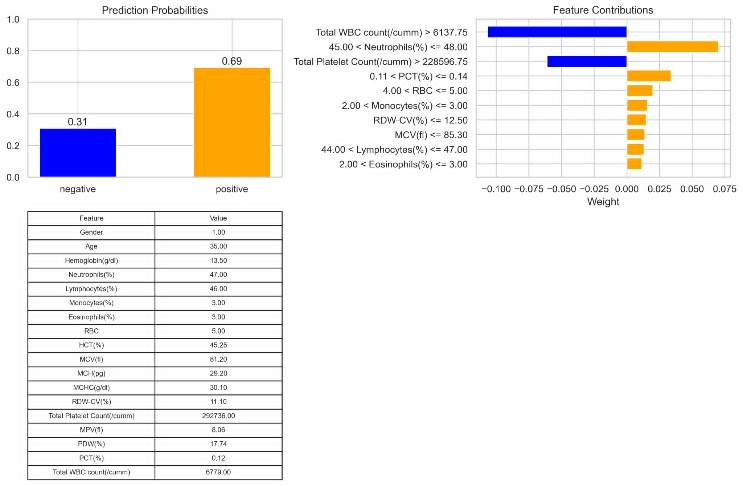


Fig. 9 Machine learning model prediction interpretation by LIME explainable AI for Random Forest

Table V illustrates a comparison of the proposed dengue detection system with previous work done on this subject. We can see that random forest gave similar accuracy (range: 80%-89%) for our model [4]. However, logistic regression and stacking gave better predictions for the dataset used in [5] and [7] compared to our dataset. SVM gave much lower accuracy than any of the other models [6].

Table V. COMPARISON OF THE PROPOSED SYSTEM WITH EXISTING WORKS

|  |  |  |  |
| --- | --- | --- | --- |
| Reference | Model | Accuracy (%) | F1 Score |
| [4] | Random Forest | 87.2 | Not available |
| [5] | Stacking | 96.8 | 0.96 |
| [6] | Support Vector | 71.0 | 0.83 |
| [7] | Logistic Regression | 93.1 | 0.93 |
| Our work | Random Forest | 82.3 | 0.80 |

# Conclusions

This research demonstrates various machine learning models for predicting dengue from a Hematological dataset of a 250-bedded general hospital in Jamalpur, having 1523 samples and 18 features. Various preprocessing techniques, such as removing duplicate values and handling outliers, using min-max scaling, level encoding, and synthetic sampling, are applied. Both positive and negative cases are predicted using various machine learning models. Using GridSearchCV, the hyperparameters of the applied model are tuned. In the ensemble model, we used various boosting techniques like Adaboost, XGBoost, and Gradient Boosting, and the base learners of each case were DecisionTreeClassifier. Among all the models, we got the highest classification accuracy of 82% from the Random Forest classifier that GridSearch tuned. Lastly, the LIME XAI technique has been implemented to examine the notable characteristics and understand the Random Forest predictions. In the future, the employed dataset can be expanded by a private dataset that can be collected and merged. Blood smear photographs for the same patient data can be used to implement a multimodal architecture.

##### References

1. M. S. Hossain *et al*., “Twenty-two years of dengue outbreaks in Bangladesh: epidemiology, clinical spectrum, serotypes, and future disease risks,” Tropical Medicine and. Health, vol. 51, Jul. 2023.
2. I. M. Cismaru et al., “Increasing the awareness of under-diagnosed tropical cases of dengue in Romania,” Tropical Medicine and Infectious Disease, vol. 8, 2023.
3. J. P. Salazar Buenaño *et al*., “Impact of dengue fever on depression, anxiety, and stress symptoms in Esmeraldas Province, Ecuador: a prospective cohort study,” Tropical Medicine and Health, vol. 52, 2024.
4. J. P. Salazar Buenaño *et al*., “Impact of dengue fever on depression, anxiety, and stress symptoms in Esmeraldas Province, Ecuador: a prospective cohort study,” Tropical Medicine and Health, vol. 52, 2024.
5. N. J. Riya, M. Chakraborty and R. Khan, “Artificial Intelligence-Based Early Detection of Dengue Using CBC Data,” IEEE Access, vol. 12, pp. 112355-112367, 2024.
6. A. Qaiser, M. A. Anwar, A. A. Shah, and M. Imran, “Support Vector Machine Outperforms Other Machine Learning Models in Early Diagnosis of Dengue Virus,” Advances in Virology, vol. 2024, pp. 1–10, 2024.
7. B. C. Bohm, *et al*., “Utilization of machine learning for dengue case screening,” BMC Public Health, vol. 24, 2024.
8. M. A. S. Nirob, A. K. M. F. K. Siam, P. Bishshash, S. Mia, T. I. Tareq, and M. Ashikuzzaman, “Dengue fever hematological dataset: Clinical insights for improved diagnosis and patient management,” *Mendeley Data*, vol. 1, 2024.