Accuracy:

What it is: Accuracy measures the proportion of correct predictions (both true positives and true negatives) out of all predictions made.

Use case: Accuracy is useful when the classes (dengue-positive vs. dengue-negative) are balanced

Formula:

- Formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

Precision:

What it is: Precision calculates the proportion of true positive cases among all instances classified as positive (i.e., how many of the predicted dengue-positive cases are actually dengue-positive).

Use case: Precision is important when the cost of false positives is high, such as in medical diagnoses where misclassifying a healthy person as sick could lead to unnecessary treatment.

- Formula:

$$Precision = \frac{TP}{TP + FP}$$

Recall (Sensitivity or True Positive Rate):

- **What it is**: Recall measures the proportion of actual positive cases that were correctly identified by the model (i.e., how many of the actual dengue-positive patients are correctly identified).

- **Use case**: Recall is important when the cost of false negatives is high, such as in medical diagnostics where missing a positive case (i.e., failing to identify a sick person) can be dangerous.

- Formula:

$$Recall = \frac{TP}{TP + FN}$$

F1 Score:

F1-Score:

- **What it is**: F1-Score is the harmonic mean of precision and recall. It balances both metrics and is useful when you need a balance between precision and recall.

- **Use case**: It's particularly helpful when the class distribution is imbalanced (e.g., more negative cases than positive cases, which is common in healthcare).

- Formula:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

ROC-AUC (Receiver Operating Characteristic - Area Under the Curve):

- **What it is**: ROC-AUC evaluates the tradeoff between the true positive rate (recall) and the false positive rate. The area under the ROC curve provides an aggregate measure of the classifier's ability to discriminate between positive and negative classes.
- **Use case**: ROC-AUC is especially useful when you need to evaluate the performance of a model across all classification thresholds.
- **Interpretation**: A higher AUC indicates better model performance. A model with an AUC of 0.5 is equivalent to random guessing, while an AUC of 1 indicates perfect classification.

Confusion Matrix:

- **What it is**: A confusion matrix is a table that visualizes the performance of a classification algorithm. It shows the counts of TP, TN, FP, and FN.
- **Use case**: It's useful for understanding the types of classification errors the model is making.

Data Preprocessing:

1. Handling Missing Values: Check for and handle any missing values.
2. Feature Encoding: Categorical Features: Convert categorical data (e.g., Sex, Race, Residence_Area) to numerical form using methods like one-hot encoding for non-binary categories or binary encoding for binary categories (e.g., Sex as 0 for male, 1 for female).
3. Feature Scaling: Scaling helps normalize numerical features (like Age, Day_Count_of_Symptoms) to ensure they contribute equally.
4. Label Encoding for Target Variable: Classification (0 = no dengue, 1 = dengue) is binary,
5. Drop the 'Gestational_Age' column
6. Convert the age to a string to extract the first digit and the remaining part

Exploratory Data Analysis (EDA)

1. Visualization : the mean age for dengue-positive cases
2. Dengue Cases by Age Group and Gender
3. Calculate Precision, Recall, F1-Score, and Accuracy
4. feature importance
5. count and percentage of dengue positive and negative cases
6. confirmed dengue cases by gender
7. symptom frequency and percentage for dengue-positive cases
8. population count and percentage of each area
9. visualize the count of dengue-positive cases by race
10. frequency of dengue positive cases for each day (Day_Count_of_Symptoms)
11. Calculate total dengue positive and dengue negative cases
12. Confirmed Dengue Cases by Gender
13. Symptom Frequency and Percentage Summary for Dengue-Positive Cases:

Model Selection:
Random Forest: Good for feature importance analysis and generally provides robust performance with minimal tuni ng.
Gradient Boosting or XGBoost: High performance on tabular data, especially for imbalanced classes.

Model Training and Evaluation:
Train-Test Split: Divide the dataset into training and testing sets (e.g., 80%-20% split) to evaluate model performanc e on unseen data.
Cross-Validation: Use k-fold cross-validation (e.g., 5 or 10 folds) to ensure the model's robustness across multiple s ubsets of the data.
Evaluation Metrics:
Accuracy: The percentage of correctly classified instances.
Precision, Recall, F1-Score: Especially important for imbalanced classes; they help capture true dengue cases while minimizing false positives.
ROC-AUC Score: A measure of how well the model distinguishes between the two classes.

Hyperparameter Tuning:

Grid Search or Random Search to optimize parameters.

For example:

Random Forest: Number of trees, max depth, minimum samples for split.

XGBoost: Learning rate, max depth, subsample.

Use cross-validation within the hyperparameter search to avoid overfitting.

How model decide is a dengue positive or negative:

1. Boost (Extreme Gradient Boosting)

XGBoost is a powerful and efficient implementation of gradient-boosted decision trees. Here's how it works:

Decision Trees: XGBoost is made up of many decision trees, where each tree is trained to improve the accuracy of the previous tree. Each tree makes a "decision" by asking questions about the input features, such as:

- Is **the patient's fever present (1) or absent (2)?**
- does **the patient show signs of petechiae?**
- Is **the patient's retroorbital pain high or low?**

Tree Splitting: During training, each tree looks at the features (like fever, myalgia, rash, etc.) and finds the best way to split the data into different groups based on feature values. For example, if the fever is `1` (positive), the model might split the data into one group for positive dengue and another group for negative dengue based on other features.

Boosting: XGBoost builds one tree at a time and adjusts the model based on errors made by the previous tree. Each tree improves the prediction by focusing on the hardest-to-classify examples.

Prediction: After training, when a new patient's data is passed into the model, the model traverses through all the decision trees and outputs a prediction based on the majority vote or weighted vote of the trees. If most trees predict "positive" for dengue, the final output will be "positive"; otherwise, it will be "negative".

2. Random Forest

Random Forest is similar to XGBoost in that it uses multiple decision trees. Here's how it works:

- **Bagging**: Random Forest creates multiple decision trees by using different random subsets of the training data. Each tree is trained independently using a subset of features and data points. This technique is called **bagging** (bootstrap aggregation).
- **Feature Randomness**: Random Forest also selects random features for each split in the decision tree. This randomness helps make the model more robust and prevents overfitting.
- **Decision Making**: Like XGBoost, each tree in the forest predicts whether a patient is dengue-positive or negative. After all trees have made their predictions, the model aggregates the results. The final prediction is based on the majority vote of the trees (i.e., if more than 50% of trees say "positive", the model predicts "positive").

How the Model Learns to Predict Dengue-Positive vs. Dengue-Negative

The models learn to predict whether a patient is dengue-positive or negative by examining relationships between the features (symptoms, demographic information, etc.) and the target label (`Classification`). During training:

1. **Learning the Patterns**: The model looks at many examples (training data) where the **features** (fever, rash, myalgia, etc.) are paired with the **target labels** (dengue-positive or dengue-negative). It learns the statistical relationships between the input features and the target outcome.
2. **Identifying Important Features**: The models identify which features (e.g., fever, rash, etc.) are most important in determining whether a patient is dengue-positive or negative. This is why feature importance plots are useful — they show which symptoms are most predictive of the disease.
3. **Optimizing for Accuracy**: The model adjusts its internal parameters to minimize errors during training. For example, decision trees are split to reduce error (impurity), while SVM maximizes the margin between classes.
4. **Generalization**: After training, the model applies the learned patterns to new, unseen data to make predictions about whether a new patient is dengue-positive or negative.