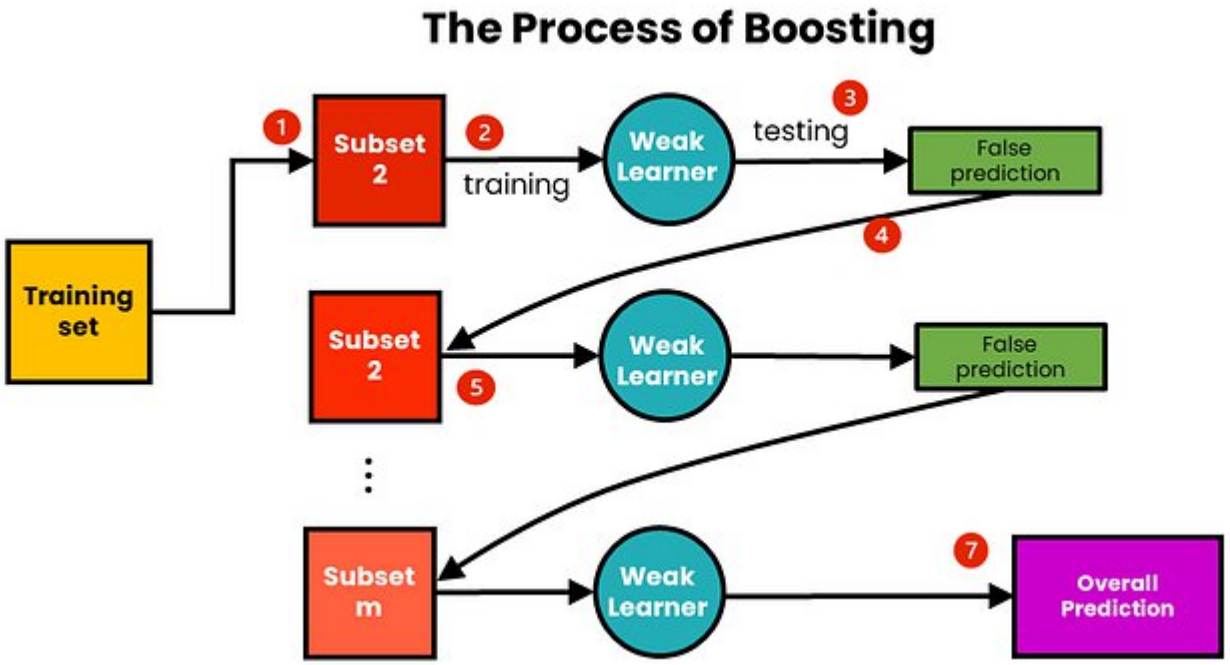
Boosting algorithms in Machine Learning

•**Boosting-**combines the predictions of multiple weak models to generate a powerful ensemble model
•It prioritizes samples that were incorrectly categorized in previous iterations
•It focuses on reducing errors in sequential model training.



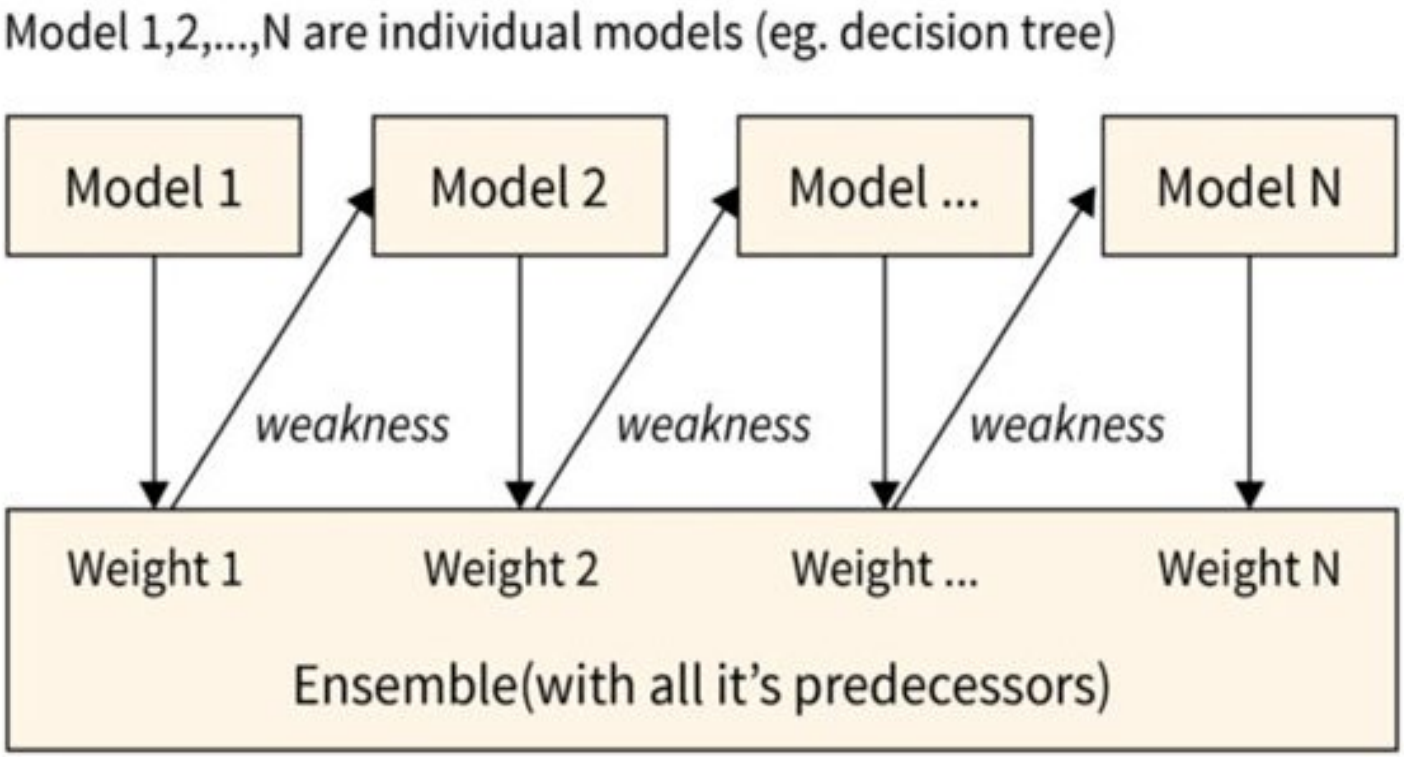
STEPS IN BOOSTING ALGORITHM

1. **Select Initial Weights:** Assign initial weights to all data points
2. **Train Sequentially**: Train the first weak learner on the data. This makes the next weak learner focus more on the harder cases.
3. **Iterate the Process**: Repeat the process of adjusting weights and training subsequent learners.
4. **Combine the Results**: Aggregate the predictions of all weak learners to form the final output

Types of Boosting Algorithms

1. AdaBoost (Adaptive Boosting)
2. XGBoost (Extreme Gradient Boosting)
3. Light GBM (Light Gradient Boosting Machine)
4. Cat Boost

1. Ada Boost (Adaptive Boosting)

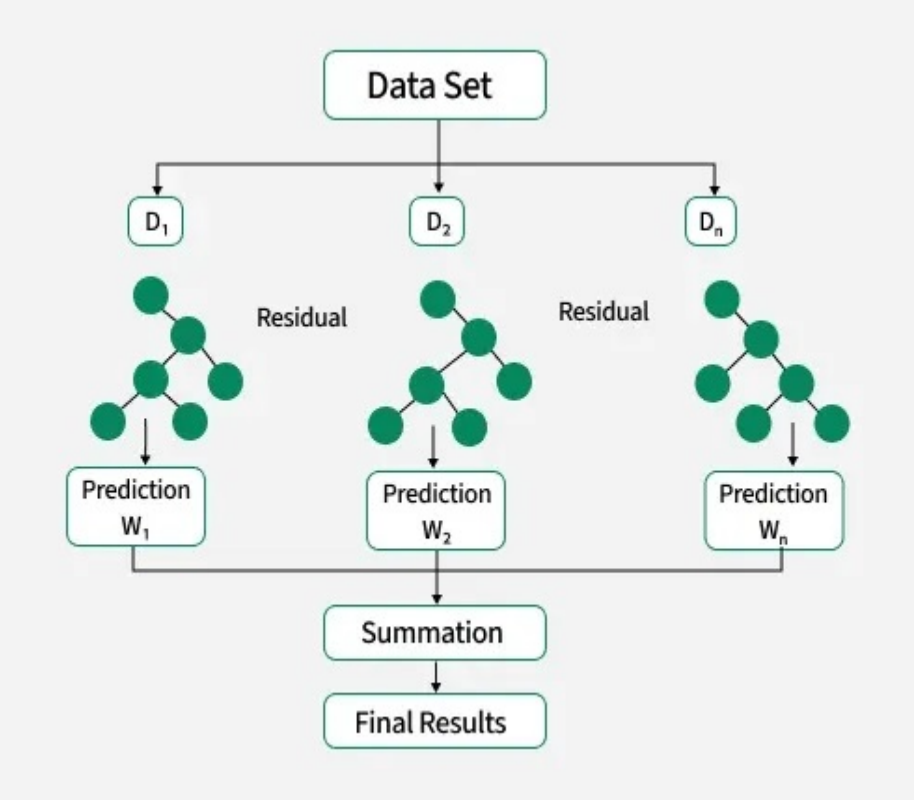


1. Ada Boost (Adaptive Boosting)

- It is the most popular boosting algorithm, which assigns weights to training instances and adjusts these weights based on the performance of weak learners.
- It focuses on misclassified instances, allowing subsequent weak learners to concentrate on these samples
- The final prediction is determined by aggregating the predictions of all weak learners

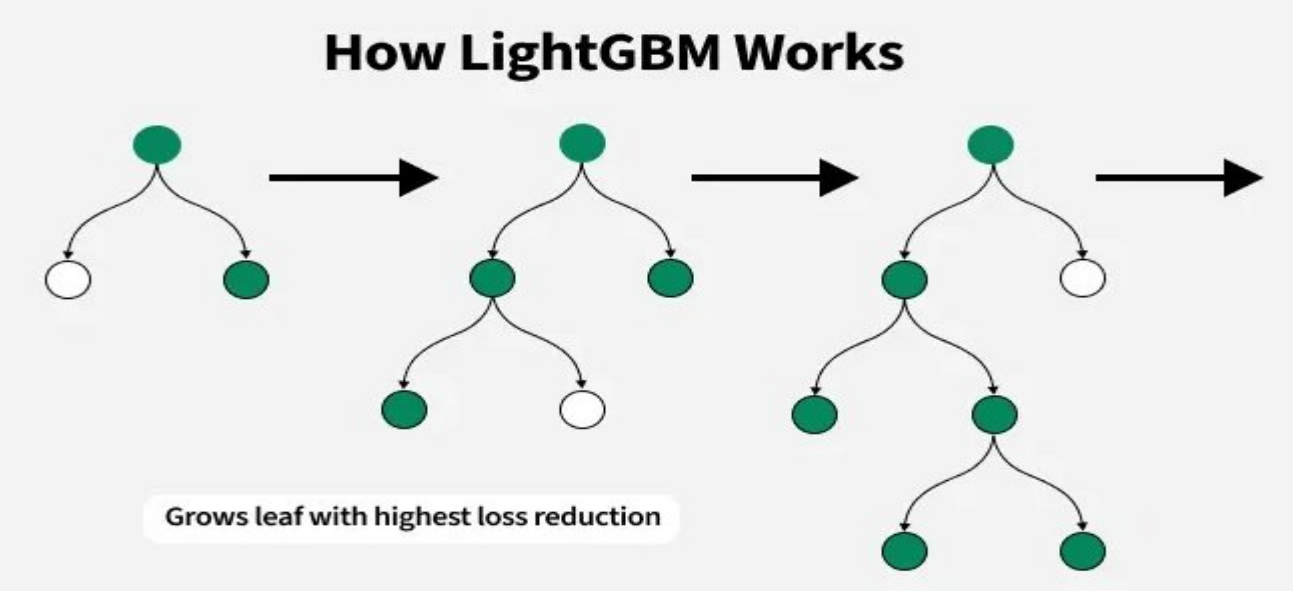
2. XGBoost (Extreme Gradient Boosting)

- XG is an advanced boosting algorithm that combines gradient boosting with regularization techniques
- It incorporates both tree-based models & linear models to enhance performance and efficiency
- It combines both, gradient boosting and regularization strategies to prevent overfitting
- It is known for its speed, scalability, and ability to handle large-scale datasets effectively



3. Light GBM (Light Gradient Boosting Machine)

- Light GBM is a high-performance boosting algorithm that uses a leaf-wise approach to construct decision trees
- It prioritizes growing the leaf nodes that reduce the loss the most, resulting in faster training times.
- It is particularly efficient when dealing with large datasets and is widely used in competitions and industry applications.



4.Cat Boost

- CatBoost is a boosting algorithm designed specifically for categorical data
- It handles categorical features directly, eliminating the need for pre-processing, such as one-hot encoding

5.Stochastic Gradient Boosting

- It randomly selects a subset of features and samples, providing diversity in the weak learners
- This randomness helps prevent overfitting and improves the generalization ability of the model

Advantages of Boosting

- Improved Performance
- Ability to Handle Complex Data
- Robustness to Noise
- Flexibility
- Interpretability

Applications of Boosting

- Image and Object identification
- Text and Natural Language Processing
- Fraud Detection
- Medical Diagnosis
- Recommendation Systems
- Time Series Analysis

Building Model in Python

1. Importing Required Libraries

```
# Load libraries
from sklearn.ensemble import AdaBoostClassifier
from sklearn import datasets
# Import train_test_split function
from sklearn.model_selection import train_test_split
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
```

Building Model in Python

2. Loading Dataset

```
# Load data
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

3. Split dataset

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

Building Model in Python

4. Building the AdaBoost Model

```
# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=50,
                         learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test)
```

Building Model in Python

4. Building the AdaBoost Model

```
# Create adaboost classifer object
abc = AdaBoostClassifier(n_estimators=50,
                         learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test)
```

5. Evalu

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```