Name: Durva Kadam Roll No. 23

St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2025-2026
Class: BE-ITA/B, Semester: VII
Subject: Data Science Lab

Experiment – 7

1. **Aim:** To implement Supervised Learning algorithm - Ada Boost.
2. **Objectives:** Students should be familiarize with Learning Architectures and Frameworks
3. **Prerequisite:** Python basics

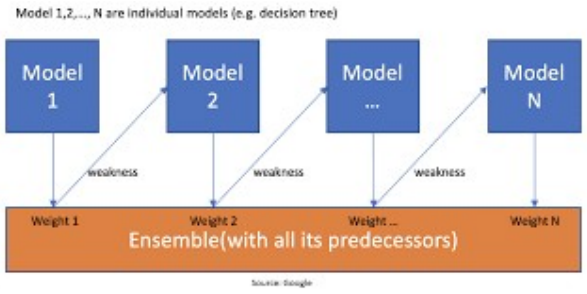
4. **Pre-Experiment Exercise:**
   Theory:

AdaBoost, short for Adaptive Boosting, is a statistical classification meta-algorithm formulated by Yoav Freund and Robert Schapire in 1995, who won the 2003 Gödel Prize for their work. It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. Usually, AdaBoost is presented for binary classification, although it can be generalized to multiple classes or bounded intervals on the real line.

AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

Although AdaBoost is typically used to combine weak base learners (such as decision stumps), it has been shown that it can also effectively combine strong base learners (such as deep decision trees), producing an even more accurate model.

Every learning algorithm tends to suit some problem types better than others, and typically has many different parameters and configurations to adjust before it achieves optimal performance on a dataset. AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier. When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify examples.



6. **Laboratory Exercise**
   Procedure
   i. Build a model and make predictions.
   ii. Assign higher weights to miss-classified points.
   iii. Build next model.

iv. Repeat steps 3 and 4.
v. Make a final model using the weighted average of individual models.
vi. Add relevant comments in your programs and execute the code. Test it for various cases.

Post-Experiments Exercise:

A. Extended Theory:

a. Explain Hyper tuning parameters.

B. Conclusion:

1. Write what was performed in the program (s) .
2. What is the significance of program and what Objective is achieved?

C. References:

[1] https://www.analyticsvidhya.com/blog/2021/03/introduction-to-adaboost-algorithm-with-python-implementation/
[2] . https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble

---

Code:

```
# Dataset: Breast Cancer Dataset (from sklearn)
# Step 1: Import Required Libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, classification_report
# Step 2: Load Dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split into Training and Testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Step 3: Build the Base (Weak) Learner
# ------------------------------------------------
# We use a shallow Decision Tree (stump) as a weak learner
base_model = DecisionTreeClassifier(max_depth=1)

# Step 4: Build the AdaBoost Classifier
# ------------------------------------------------
# n_estimators = number of weak learners
# learning_rate = how much each weak learner contributes
ada_model = AdaBoostClassifier(
    estimator=base_model,
    n_estimators=50,
    learning_rate=1.0,
    random_state=42
)
# Step 5: Train the Model
ada_model.fit(X_train, y_train)
# Step 6: Make Predictions
# ------------------------------------------------
y_pred = ada_model.predict(X_test)
# Step 7: Evaluate Model Performance
# ------------------------------------------------
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_
report(y_test, y_pred))
# Step 8: Test for Various Cases
# ------------------------------------------------
# Predicting for a few random samples
sample = X_test[:3]
print("\nSample Predictions:", ada_model.predict
(sample))
print("Actual Labels:", y_test[:3])
```

Output:

```
Accuracy: 0.9649122807017544

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.93      0.95        43
           1       0.96      0.99      0.97        71

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114


Sample Predictions: [1 0 0]
Actual Labels: [1 0 0]
```