Experiment No.10

Implement ensemble methods: AdaBoost and XGBoost for classification tasks.

Aim: Implement ensemble methods: AdaBoost and XGBoost for classification tasks.

Theory:

Machine learning models often face limitations when used individually—some are too simple, others overfit. This is where ensemble methods come in. Ensemble learning combines the predictions from multiple models to build a stronger and more accurate classifier. The idea is simple but powerful: "A group of weak learners can come together to form a strong learner."

- **Boosting**

Among the various ensemble techniques, boosting is one of the most effective. It builds models sequentially—each new model learns from the errors of the previous one. Over time, the model becomes more accurate by focusing more on the mistakes that were previously made.

There are two popular boosting techniques we'll discuss here:

1. AdaBoost (Adaptive Boosting)

Let's start with AdaBoost. Imagine you're giving a quiz to a group of students (your weak learners). The first student gets some questions wrong. So, in the next round, you make those questions count more—increasing their importance. The next student tries harder on those questions. This process repeats, and by the end, you take a weighted vote from all students based on how well they performed. That's essentially how AdaBoost works!

- AdaBoost combines multiple weak learners, usually decision stumps (trees with one split).
- Initially, all samples have equal weight.
- After each iteration, the weights of misclassified samples are increased, forcing the next model to focus on them.
- Each learner gets a weight depending on its accuracy.
- Final prediction = Weighted majority vote of all learners.

This makes AdaBoost adaptive, as it constantly adjusts to where it went wrong. It works well on clean datasets but can be sensitive to noisy data and outliers, as it might focus too much on them.

2. XGBoost (Extreme Gradient Boosting)

Now, let's talk about the champion: XGBoost. If AdaBoost is a smart student, XGBoost is the student who studies smarter, faster, and deeper.

- XGBoost is an optimized and regularized version of gradient boosting.
- It uses gradient descent to minimize errors and includes both first and second derivatives to improve learning.

- It adds regularization (L1 & L2) to reduce overfitting and improve model generalization.
- Supports parallel processing, tree pruning, handling missing data, and sparsity awareness.
- Extremely fast, scalable, and performs exceptionally well on structured/tabular data.

Unlike AdaBoost, which focuses on reweighting samples, XGBoost builds new trees that predict the residuals (errors) of previous models, using gradients to optimize. The result? A highly efficient algorithm that dominates in Kaggle competitions and production environments.

Conclusion:

Both AdaBoost and XGBoost are effective boosting techniques for improving classification accuracy. AdaBoost is straightforward and works well on smaller datasets, but it may overfit in the presence of noise. XGBoost, on the other hand, offers greater control, regularization, and scalability, making it better suited for large and complex datasets. In practical scenarios, XGBoost is often the preferred choice due to its optimization capabilities and performance.

Viva Questions:

1. What is ensemble learning?
2. How does boosting differ from bagging?
3. What is the basic principle of AdaBoost?
4. What kind of learners does AdaBoost use?

Program Code :

```
# Importing libraries

from sklearn.datasets import load_breast_cancer

from sklearn.model_selection import train_test_split

from sklearn.ensemble import AdaBoostClassifier

from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score, classification_report


# Load sample dataset

data = load_breast_cancer()

X, y = data.data, data.target
```

```
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# --------------------
# AdaBoost Classifier
# --------------------
ada_model = AdaBoostClassifier(n_estimators=50, learning_rate=1.0, random_state=42)
ada_model.fit(X_train, y_train)
ada_preds = ada_model.predict(X_test)

# --------------------
# XGBoost Classifier
# --------------------
xgb_model = XGBClassifier(n_estimators=100, use_label_encoder=False,
eval_metric='logloss', random_state=42)
xgb_model.fit(X_train, y_train)
xgb_preds = xgb_model.predict(X_test)

# --------------------
# Evaluation
# --------------------
print("AdaBoost Accuracy:", accuracy_score(y_test, ada_preds))
print("XGBoost Accuracy:", accuracy_score(y_test, xgb_preds))

print("\nClassification Report - AdaBoost:")
print(classification_report(y_test, ada_preds))

print("\nClassification Report - XGBoost:")
print(classification_report(y_test, xgb_preds))
```