Experiment No. 10

Title: Study and Implement Bagging/Boosting using Random Forests.

Objectives:

1. Understand the concepts of Bagging and Boosting and their impact on model performance.
2. Implement Bagging and Boosting techniques using Random Forests.
3. Compare the performance of Bagging and Boosting with standard Random Forest classifiers.
4. Evaluate accuracy,

Problem Statement:

Develop and analyze ensemble learning techniques (Bagging and Boosting) using Random Forests to improve classification accuracy and reduce overfitting in machine learning models. Compare the effectiveness of these techniques on real-world datasets.

Outcomes:

1. Implementation of Bagging and Boosting using Random Forests.
2. Performance comparison between Bagging, Boosting, and standard Random Forest classifiers.
3. Evaluation metrics demonstrating improvement in accuracy metrics.

Tools Required: 4GB RAM, Anaconda, Notebook

Theory:

Bagging (Bootstrap Aggregating) and Boosting are ensemble learning techniques that improve the stability and accuracy of machine learning algorithms by combining multiple base learners.

- Bagging:
  - Creates multiple subsets of training data via bootstrapping.
  - Trains separate models on each subset independently.
  - Aggregates predictions through averaging (regression) or majority voting (classification).
  - Example: Random Forest.
- Boosting:
  - Sequentially trains weak models, each focusing on errors of the previous ones.
  - Assigns higher weights to misclassified instances.
  - Combines weak learners into a strong predictive model.
  - Examples: AdaBoost, Gradient Boosting, XGBoost.

Algorithm:

1. Load and preprocess dataset.

2. Split data into training and testing sets.
3. Implement Random Forest classifier as a baseline.
4. Implement Bagging with Random Forest:
    - Use BaggingClassifier from scikit-learn.
    - Train multiple decision trees on bootstrapped samples.
    - Aggregate predictions.
5. Implement Boosting with Random Forest:
    - Use AdaBoost or GradientBoostingClassifier from scikit-learn.
    - Sequentially train models, adjusting for errors.
6. Compare performance metrics (accuracy)
7. Visualize.

Source Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier,
  AdaBoostClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.datasets import make_classification
data, labels = make_classification(n_samples=1000, n_features=20, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2,
  random_state=42)
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
bagging = BaggingClassifier(estimator=RandomForestClassifier(), n_estimators=50,
  random_state=42)
bagging.fit(X_train, y_train)
y_pred_bagging = bagging.predict(X_test)
print("Bagging Accuracy:", accuracy_score(y_test, y_pred_bagging))
boosting = AdaBoostClassifier(estimator=RandomForestClassifier(n_estimators=10,
  random_state=42), n_estimators=50, random_state=42)
boosting.fit(X_train, y_train)
y_pred_boosting = boosting.predict(X_test)
print("Boosting Accuracy:", accuracy_score(y_test, y_pred_boosting))
```

Output:

```
Random Forest Accuracy: 0.9
Bagging Accuracy: 0.88
Boosting Accuracy: 0.86
```

Conclusion:

_____

_____

_____

_____