Decision Tree Algorithms and Ensemble Methods

In this mini-project, you will apply Decision Tree algorithms and Ensemble Methods (such as Bagging, Boosting, and Stacking) to a real-world classification task. You will implement and evaluate Decision Trees, Random Forests, AdaBoost, and Stacking techniques on a dataset, and compare the performance of each method.

For this project, you will use the **"Credit Card Fraud Detection"** Dataset available on Kaggle. This dataset contains credit card transactions where the task is to predict whether a transaction is fraudulent or not. Alternatively, you may use any other binary classification dataset such as predicting customer churn, spam email detection, or loan default prediction.

Instructions:

1. **Step 1: Data Exploration and Visualization**
    - Import Libraries: Import necessary Python libraries such as pandas, numpy, matplotlib, seaborn, and sklearn.
    - Load the Dataset: Load your chosen dataset (e.g., Credit Card Fraud Detection dataset) into a Pandas DataFrame.
    - Explore the Dataset:
        - Display basic information about the dataset (e.g., number of rows, columns, missing values).
        - Analyze the distribution of the target variable (fraudulent or non-fraudulent).
        - Visualize relationships between features using correlation matrices, pair plots, and histograms.
2. **Step 2: Data Preprocessing**
    - Data Splitting: Split the dataset into training and testing sets (typically 80% training, 20% testing).
    - Handle Missing Values: If the dataset has missing values, handle them either by removing rows or filling with mean/median.
    - Data Scaling: Use StandardScaler to scale the features as it is crucial for ensemble methods like Random Forest and AdaBoost.

3. **Step 3: Implement Decision Tree Classifier**

   Train a Decision Tree: Use DecisionTreeClassifier to train a model on the training data.

   Evaluate the Model:

   - Evaluate using accuracy, precision, recall, F1-score, and confusion matrix.
   - Since the dataset is imbalanced (fraudulent transactions are rare), focus on precision, recall, and F1-score.

4. **Step 4: Implement Random Forest Classifier (Bagging)**

   Train a Random Forest: Use RandomForestClassifier to train a model on the training data.

   Evaluate the Model:

   - Use accuracy, precision, recall, F1-score, and confusion matrix to evaluate the performance.

5. **Step 5: Implement AdaBoost Classifier (Boosting)**

   Train an AdaBoost Classifier: Use AdaBoostClassifier with a base estimator like DecisionTreeClassifier.

   Evaluate the Model:

   - Evaluate using accuracy, precision, recall, F1-score, and confusion matrix.

6. **Step 6: Implement Stacking Classifier (Stacking)**

   Train a Stacking Classifier: Use multiple base models (e.g., Decision Tree, Logistic Regression, K-Nearest Neighbors) and a meta-model (e.g., Logistic Regression) to build a stacking classifier.

   Evaluate the Model:

   - Evaluate the performance using accuracy, precision, recall, F1-score, and confusion matrix.

7. **Step 7: Hyperparameter Tuning (Optional)**

   - Hyperparameter Tuning: Use techniques like GridSearchCV or RandomizedSearchCV to tune the hyperparameters of your models.
   - Optimize performance: Focus on parameters like the number of estimators (n_estimators), maximum depth (max_depth), and learning rate (learning_rate) for boosting methods.

8. **Step 8: Model Evaluation and Comparison**
    - Compare all models you implemented: Decision Tree, Random Forest, AdaBoost, and Stacking.
    - Create a table comparing the accuracy, precision, recall, and F1-score for each model.

Deliverables:

- Code Implementation: Submit your Python code either as a .py script or a Jupyter notebook (.ipynb).
- Model Comparison: Provide a table comparing the performance (accuracy and other metrics) of each model.
- Performance Evaluation: Include confusion matrices, classification reports, and performance scores for all models.
- Discussion: Provide a brief discussion summarizing your findings, including which model performed the best and why, as well as the advantages and disadvantages of each ensemble method.