1. **AdaBoost with Decision Tree Base Model on Heart Disease Dataset**
    - **Dataset**: Use the *Heart Disease Dataset* from the UCI repository or Kaggle.
    - **Tasks**:
        1. Load and preprocess the data (handle any missing values and encode categorical variables).
        2. Split the data into training and testing sets (80% train, 20% test).
        3. Initialize an AdaBoost classifier with a Decision Tree as the base estimator and set `n_estimators=50`.
        4. Train the model with Decision Trees of varying depths (`max_depth=1`, `max_depth=3`, and `max_depth=5`).
        5. Evaluate each model using a confusion matrix and accuracy score, and report the results.
        6. Analyze how the depth of the base estimator affects the model's performance, discussing trade-offs between underfitting and overfitting.

2. **Feature Impact Analysis with AdaBoost on the Iris Dataset**
    - **Dataset**: Use the *Iris Dataset* from `sklearn.datasets`.
    - **Tasks**:
        1. Load the Iris dataset and split it into training and testing sets (70% train, 30% test).
        2. Initialize an AdaBoost classifier with a Decision Tree as the base estimator (`max_depth=2`) and `n_estimators=100`.
        3. Train the model and calculate feature importance scores.
        4. Display the importance scores and identify the top two features.
        5. Visualize these two features in a 2D scatter plot, coloring points by their class.
        6. Discuss how well these top features separate the classes.

3. **AdaBoost Hyperparameter Tuning on the Titanic Dataset**
    - **Dataset**: Use the *Titanic Dataset* from Kaggle or `seaborn` library.
    - **Tasks**:
        1. Load the Titanic dataset, preprocess it (handle missing values and encode categorical variables), and split it into training and testing sets (80% train, 20% test).
        2. Initialize an AdaBoost classifier with `n_estimators=50` and a Decision Tree base estimator (`max_depth=1`).
        3. Perform hyperparameter tuning on `n_estimators` (values: 50, 100, 150) and `learning_rate` (values: 0.1, 0.5, 1.0).
        4. For each configuration, evaluate accuracy and F1-score on the test data.
        5. Plot accuracy and F1-score for each configuration and analyze how tuning affects performance in terms of bias-variance trade-off.