CS 6375: Machine Learning
Project 2: Evaluation of Tree-Based Classifiers and Their Ensembles

Instructor: Tahrima Rahman

In this project, you will evaluate tree-based classifiers and their ensemble methods as discussed in class. You will use `scikit-learn` and perform the following experiments.

Datasets

- Download the 15 datasets available on eLearning. Each dataset is divided into three subsets: the **training set**, the **validation set**, and the **test set**. The datasets are in CSV format, where each row represents an instance with attribute values separated by commas. The last attribute corresponds to the class variable.
- Assume that all attributes take values from the domain $\{0, 1\}$.
- The datasets are synthetically generated by randomly sampling solutions and non-solutions from a Boolean formula in conjunctive normal form (CNF). Solutions are labeled as class "1," while non-solutions are labeled as class "0."

  Example CNF Formula:

  $$(X_1 \vee \neg X_2) \ \wedge \ (\neg X_1 \vee X_3) \ \wedge \ (X_2 \vee \neg X_3)$$

  Here $\wedge$ denotes logical AND, $\vee$ denotes logical OR, and $\neg$ denotes logical NOT (negation).

  This formula has three clauses:

  - Clause 1: $(X_1 \vee \neg X_2)$
  - Clause 2: $(\neg X_1 \vee X_3)$
  - Clause 3: $(X_2 \vee \neg X_3)$

  A *datapoint* is an assignment of values to (X_1, X_2, X_3), with the label given by whether all clauses are satisfied:

  - $(X_1 = 0, X_2 = 0, X_3 = 0)$: all clauses True $\Rightarrow$ label $= 1$.
  - $(X_1 = 1, X_2 = 0, X_3 = 1)$: clause 3 False $\Rightarrow$ label $= 0$.

  Thus, each datapoint corresponds to a Boolean assignment, and the class label is 1 if the CNF evaluates to true, and 0 otherwise.
- Five CNF formulas were generated with 500 variables and varying numbers of clauses: 300, 500, 1000, 1500, and 1800 clauses (each clause containing exactly 3 literals). From each formula, 100, 1000, and 5000 positive and negative examples were sampled.

- Filenames follow a structured naming convention:
  - `train_c[i]_d[j].csv` contains training data with j examples generated from a formula with i clauses.
  - `test_c[i]_d[j].csv` and `valid_c[i]_d[j].csv` contain the corresponding test and validation sets.
  - Example: `train_c500_d100.csv` contains 100 examples from the formula with 500 clauses.
- **Important:** Do not mix datasets. For instance, do not train on `train_c500_d100.csv` and test on `test_c500_d5000.csv`.

Experiments

1. **(15 points) Decision Tree Classifier:** Train a `sklearn.tree.DecisionTreeClassifier` on each dataset. Use the validation set to tune hyperparameters (e.g., criterion, splitter, max_depth). After tuning, combine the training and validation sets, retrain with the best parameter settings, and report:
   - Best hyperparameter settings found via tuning.
   - Classification accuracy and F1 score on the test set.

   (Each student is expected to obtain slightly different hyperparameter settings.)
2. **(15 points) Bagging with Decision Trees:** Repeat the above experiment using `sklearn.ensemble.BaggingClassifier` with a `DecisionTreeClassifier` as the base estimator. Report:
   - Best hyperparameter settings found via tuning.
   - Classification accuracy and F1 score.
3. **(15 points) Random Forest Classifier:** Repeat the experiment using `sklearn.ensemble.RandomForestClassifier`. Report the best parameter settings, classification accuracy, and F1 score.
4. **(15 points) Gradient Boosting Classifier:** Repeat the experiment using `sklearn.ensemble.GradientBoostingClassifier`. Report the best parameter settings, classification accuracy, and F1 score.
5. **(15 points) Comparative Analysis:** Record the classification accuracy and F1 scores for each testset and classifier in a table. You can arrange all your results in a table shown above (Table 11). Make sure you have two tables: one for classification accuracy and one for F1 Score. Then answer the following questions:
   - Which classifier achieves the best overall generalization accuracy/F1 score? Explain why.
   - How does increasing the training data size impact accuracy/F1 score for each classifier?
   - How does increasing the number of features (clauses) affect classifier performance?
6. (25 points) Download and preprocess the MNIST dataset using the instructions below:
   - The MNIST dataset consists of a training set of 60,000 examples and a test set of 10,000 examples. Each digit is centered within a 28×28 pixel grayscale image.

Table 1: Classification Accuracy

Dataset	DecisionTree	Bagging	RandomForest	GradientBoosting
c300_d100	–	–	–	–
c300_d1000	–	–	–	–
c300_d5000	–	–	–	–
c500_d100	–	–	–	–
c500_d1000	–	–	–	–
c500_d5000	–	–	–	–
c1000_d100	–	–	–	–
c1000_d1000	–	–	–	–
c1000_d5000	–	–	–	–
c1500_d100	–	–	–	–
c1500_d1000	–	–	–	–
c1500_d5000	–	–	–	–

- You can use `scikit-learn` to download and normalize the dataset using the following code:

```
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split

# Load MNIST dataset
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
X = X / 255.0  # Normalize pixel values to [0,1]

# Split into training (60K) and test (10K) sets
X_train, X_test = X[:60000], X[60000:]
y_train, y_test = y[:60000], y[60000:]
```

Task: Evaluate the four classifiers used earlier—*Decision Trees, Bagging, Random Forest, and Gradient Boosting*—on the MNIST dataset. Report their **classification accuracy** (do *not* compute F1 scores).

Analysis: Which classifier achieves the highest classification accuracy on MNIST? Provide a brief explanation for its superior performance.

Submission Instructions

Submit a single ZIP file containing:

- Your code, which demonstrates how the experiments were set up, and a `README` file with instructions for running the code.
- A report describing your results, including answers to all questions.
- Submit your AI chat transcript along with the project report (in the same format as Project 1).

Important: Your code must run without errors. If we cannot replicate your results, no credit will be given.