Assignment 2: From Trees to Boosting Power

Due Date: Wednesday, October 22, 2025 at 12:00 AM (midnight, Eastern Time)
Policy: https://columbia-coms4995.github.io/aml-fall2025/#policies

📃 Submission

- A runnable Jupyter/Colab Notebook (`.ipynb`)
- A PDF report (5-7 pages with clear structure and visualizations)

🎯 Objective

This assignment expands upon A1 and challenges you to explore the **power of Boosting methods**. You will move beyond a single model and investigate how Boosting approaches improve generalization by correcting residual errors in sequential learners.

By the end of this assignment, you should be able to:

- Train and tune tree-based and boosting models.
- Analyze the bias-variance trade-off through validation results and boosting dynamics.
- Visualize feature importance and interpret how boosting models evolve during training.

📁 Datasets

Choose one of the following datasets:

🏦 Bank Marketing Dataset

Task: Predict whether a customer will subscribe to a term deposit based on demographic and campaign features.
Link: https://archive.ics .uci.edu/dataset/222/bank+marketing

💰 Home Credit Default Risk Dataset

Task: Predict whether an applicant will repay a loan, using a wide range of application, credit, and behavioral data.
Link: https://www.kaggle.com/c/home-credit-default-risk

Both datasets are suitable for tree-based and boosting algorithms, featuring mixed data types, non-linear relationships, and moderate feature dimensionality.

🛠️ Steps

1. Data Preparation

a. Handle missing values and inconsistent types.
b. Apply label encoding or one-hot encoding to categorical variables.
c. Optionally construct new features (e.g., ratios, age groups, spending levels).
d. Split data into train / validation / test sets (e.g., 70 / 15 / 15).

2. Baseline Model: Decision Tree

a. Train a DecisionTreeClassifier using Gini impurity.
b. Tune `max_depth`, `min_samples_split`, and `ccp_alpha` (pre-pruning).
c. Evaluate train and test accuracy to illustrate overfitting vs underfitting.
d. Plot the learning curve to show how training and validation scores change with depth.

3. Boosting Methods

Your main focus in this assignment is to **understand and implement boosting algorithms**, and compare them with a baseline tree-based method.

a. Baseline

Train either a simple **Decision Tree** or a **Random Forest** as your baseline model. Tune its depth, minimum samples, and regularization parameters. Record training and validation accuracy.

b. Gradient Boosting

Train a **GradientBoostingClassifier** using scikit-learn. Explore how the following parameters affect performance and overfitting:

- `learning_rate`
- `n_estimators`
- `max_depth`
- `subsample`

c. XGBoost

Install and use XGBoostClassifier (`xgboost` library). Perform similar tuning and visualize:

- training vs. validation loss (using `eval_set` or early stopping)
- feature importance (`plot_importance`)
- effect of learning rate and tree depth on bias-variance balance

d. Optional (Bonus)

Implement [LightGBM] and compare their speed and accuracy against XGBoost. Discuss how different boosting frameworks handle categorical features and regularization.

4. Model Evaluation and Visualization

a. Metrics: Accuracy, Precision, Recall, F1, AUCPR.
b. Plot:
- Confusion matrix
- PR curve
- Feature importance (bar chart)
- Learning curve (bias-variance illustration)

c. Discuss differences between baseline Decision Tree vs Boosting performance.

Additionally, for your **boosting models**, plot and analyze:

- **Training vs Validation Loss over Boosting Iterations** (e.g., using `eval_set` or custom matplotlib visualization)
- **Effect of Learning Rate:** Compare results for 0.01, 0.1, and 0.3
- **Bias-Variance Trade-off:** Discuss how increasing the number of estimators affects generalization

These visualizations will help you interpret how boosting methods incrementally reduce bias while controlling variance.

5. Discussion and Interpretation

a. Explain why boosting methods improve generalization.
b. Relate your results to the bias-variance trade-off.
c. Comment on computational cost and interpretability.
d. Reflect on how feature importance helps interpret predictions.

6. Report Writing (6 - 10 pages)

a. **Introduction:** Problem definition & dataset description.
b. **Methods:** Preprocessing and modeling steps (summary table optional).
c. **Results:** Evaluation metrics and visualizations.
d. **Discussion:** Interpretation of findings, limitations, bias-variance reflection.
e. **AI Tool Disclosure:** List any AI tools (e.g., ChatGPT, Gemini, Claude) and their roles.

🔑 Pointers & Hints

- Start with DecisionTreeClassifier or RandomForestClassifier as a baseline before moving to boosting.
- Use `GridSearchCV` or `RandomizedSearchCV` for hyperparameter tuning.

- Visualize feature importance using `model.feature_importances_` or `xgb.plot_importance()`.
- `cross_val_score`, `learning_curve`, `validation_curve` in scikit-learn are helpful.
- For XGBoost, use the `eval_set` and `early_stopping_rounds` parameters to monitor model convergence and prevent overfitting.
- Control random seeds for reproducibility.

📊 Grading Rubric

Category	Weight	Description
Data Preparation & Baseline Decision Tree	15	Proper preprocessing and baseline model tuning.
Boosting Implementation (Gradient Boosting / XGBoost)	**30**	Correct setup, parameter tuning, and visualization of training dynamics.
Evaluation & Visualization	20	Metrics (Accuracy, F1, PR), plus boosting loss curves and feature importance.
Discussion & Interpretation	20	Strong explanation of how boosting reduces bias and improves generalization.
AI Tool Usage Disclosure	10	Transparent acknowledgment of AI tools and your personal contributions.
Bonus (+ up to 10%)	-	Compare XGBoost vs LightGBM; analyze effect of learning rate and number of estimators.