1. Pseudocode (Step-by-Step Logic in Words)

---

2. 1. Gradient Boosting (GBM)

1. Start with an initial model (e.g., mean of target for regression).
2. Compute the residuals (errors) from the current model predictions.
3. Fit a new weak learner (e.g., decision tree) on these residuals.
4. Add this learner to the model with a learning rate multiplier.
5. Repeat steps 2–4 for a set number of iterations or until convergence.
6. Final model = sum of all weak learners.

---

3. 2. AdaBoost (Adaptive Boosting)

1. Assign equal weights to all training samples initially.
2. Train a weak learner (e.g., decision stump).
3. Evaluate the learner’s error and compute its weight in the final model.
4. Increase the weights of the misclassified samples.
5. Normalize the weights.
6. Repeat steps 2–5 for several iterations.
7. Final model = weighted sum of all learners.

---

4. 3. XGBoost (Extreme Gradient Boosting)

1. Initialize a base prediction.
2. For each iteration:
    - Compute gradient and hessian (2nd derivative) of loss function.
    - Fit a tree to the gradient/hessian using regularized objective.
    - Add this tree to the model using a shrinkage (learning rate).
3. Apply regularization (L1/L2) to prevent overfitting.
4. Use column subsampling and tree pruning to speed up and regularize.
5. Final model = sum of all trees with regularization.

---

5. 4. CatBoost (Categorical Boosting)

1. Encode categorical features internally using **ordered statistics** (target-based).
2. Initialize model predictions.
3. Compute gradients of the loss function.
4. Build oblivious (symmetric) trees using permutations to avoid target leakage.
5. Add each tree to the ensemble after applying learning rate.

6. Use ordered boosting to reduce prediction shift and overfitting.
7. Final model = ensemble of symmetric trees with categorical awareness.

---

6. 5. LightGBM (Light Gradient Boosting Machine)

1. Initialize prediction.
2. Use **gradient-based one-side sampling (GOSS)** to select data subset.
3. Use **exclusive feature bundling (EFB)** to reduce features.
4. Build trees leaf-wise (with depth limits), not level-wise (faster, deeper splits).
5. Add trees iteratively to model based on gradient info.
6. Final model = sum of leaf-wise grown trees.

---

7. 📊 Comparison Table of Boosting Algorithms

Feature / Algo	**Gradient Boosting**	**AdaBoost**	**XGBoost**	**CatBoost**	**LightGBM**
Base Learner	Decision Trees	Decision Stumps/Trees	Regularized Trees	Symmetric (Oblivious) Trees	Leaf-wise Trees
Boosting Type	Gradient-based	Error-based	Gradient + Hessian	Ordered Gradient Boosting	Histogram-based Gradient Boosting
Categorical Support	Manual Encoding	Manual Encoding	Manual Encoding	Built-in (smart encoding)	Requires Encoding or Native Handle
Regularization	No	No	L1, L2 Regularization	Yes (through ordering, early stop)	Yes (depth, leaf, GOSS)
Parallelization	No	No	Yes	Limited	Yes
Tree Growth	Level-wise	Level-wise	Level-wise	Level-wise (Symmetric Trees)	**Leaf-wise (faster but riskier)**
Speed	Moderate	Slow	Fast	Moderate	**Very Fast**
Accuracy	Good	Moderate	**High**	**High with Cat Features**	**High**

Feature / Algo	**Gradient Boosting**	**AdaBoost**	**XGBoost**	**CatBoost**	**LightGBM**
Best for	General Purpose	Simple Tasks	Large-Scale Structured Data	Categorical-Rich Data	Large Datasets with Many Features