Ensemble Methods

Ensemble Methods

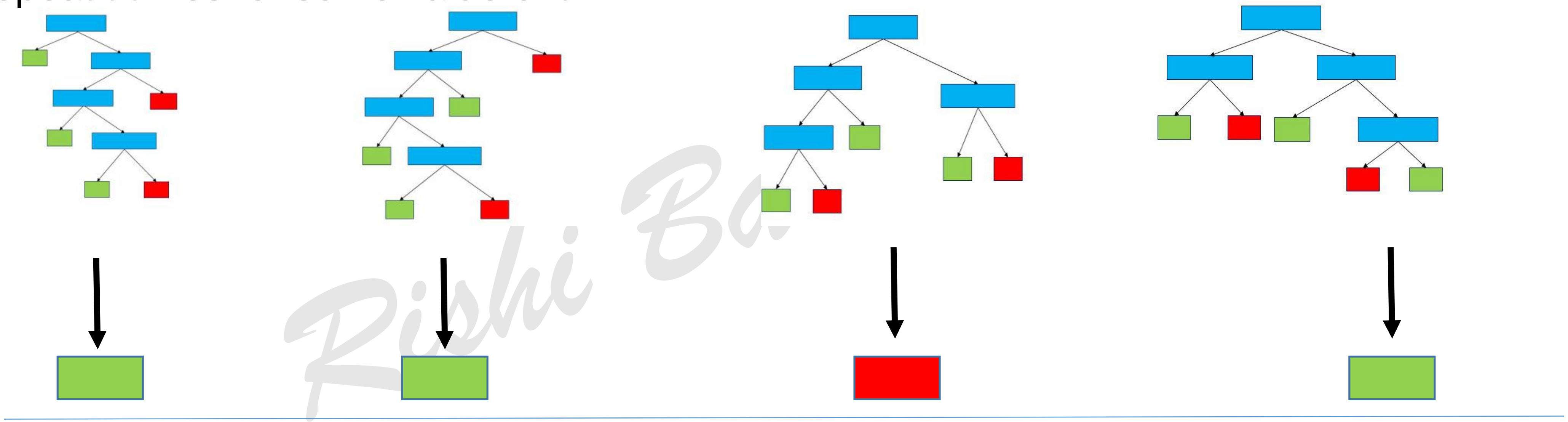
- Main cause of error while learning are due to noise, bias and variance
- Minimize above factors
- Group of weak learners combined to form a strong learner

Two Types:

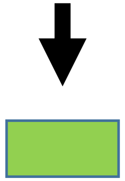
1. Homogeneous Algorithm – Bagging, Boosting
2. Heterogeneous Algorithm - Stacking

Bagging

- Build several estimators independently & average their predictions
- Take random n sample with replacement – Bootstrapping
- Train decision tree on these n sample
- Repeat t times for some value of t



Parallel Execution, Majority



Bagging

- Minimize variance
- Average of all predictions are taken – Regression
- Majority of all predictions are taken – Classification
- Handles Overfitting
- E,g: Random Forest (try subset of features at a time)

Random Forest

- Ensemble Algorithm
- model made up of many decision trees

Key Concepts:

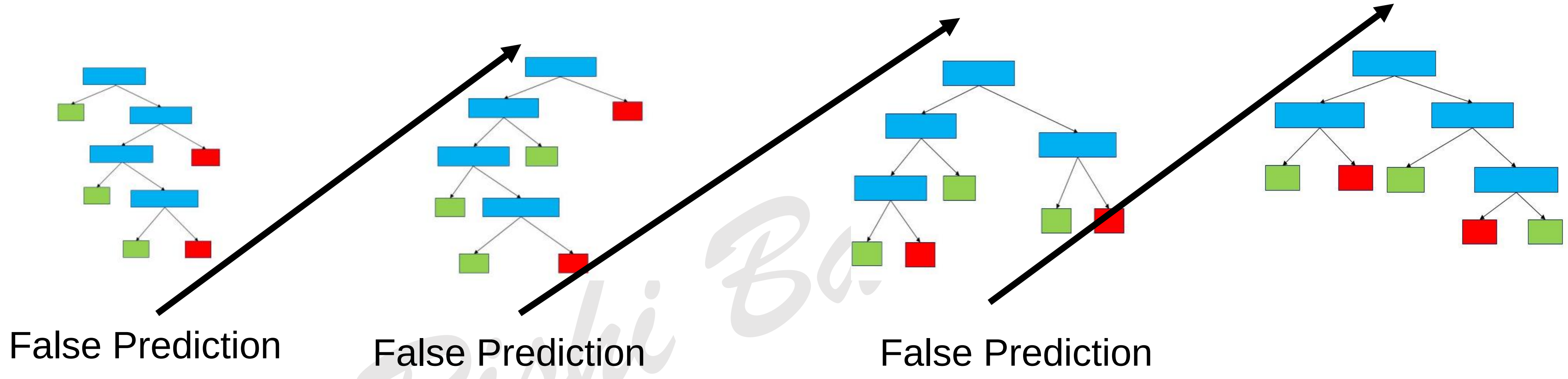
- While building trees it performs random sampling of training data points
- While splitting nodes it performs random subsets of features

RF- Real Life Analogy

- To predict a company A stock will go up or down
- A team of analysts is formed having no idea about the company A
- Each analyst has low bias as they dont have any prior assumption
- They will learn from news reports datasets
- These news reports may have high noise or irrelevant data
- Analysts are prone to capture these noise
- Each analyst might have different prediction with high variance based on different training set of reports
- Allow each analyst access to section of reports
- The noise in these section will cancelled out while sampling
- Take majority vote to decide

Boosting Training

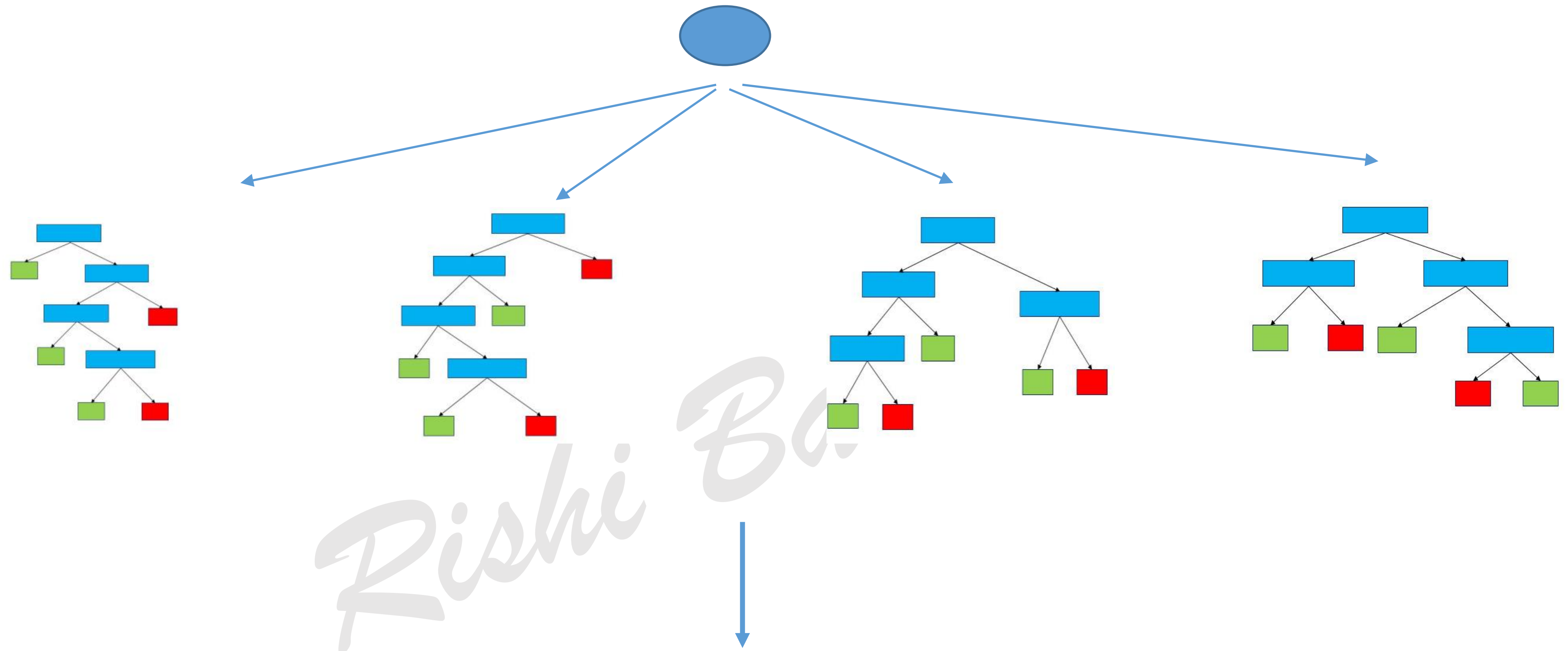
- The base algorithm reads the data and assigns equal weight to each sample observation
- False predictions from base learner assigned to next iteration with higher weightage



- Focus more on miss classified predictions

Boosting Predictions

- Parallel Predictions



Majority

Boosting

- Training is sequential
- Prediction is parallel
- Aim is to minimize Bias
- Higher vote/weightage to misclassified samples
- Tend to overfit
- E.g: Adaboost, Gradient Boost, xGBoost

Adaboost and Gradient Boost

Adaboost:

- Increases the accuracy by giving more weightage to the target which is misclassified
- Next sample – repeat same
- Weak learners combine to form a strong learners

Gradient Boost:

- increases the accuracy by minimizing the loss function and keep this as target for next decision tree building.

Gradient Boosting

Parameters:

- **n_estimators:** Number of boosting stages
- **max_depth:** Maximum depth of each estimator tree
- **min_samples_split:** Minimum samples in each subset when splitting the data set
- **learning_rate:** Defines the rate at which to converge to the optimal value
- **loss:** Type of loss function to optimize (ls == least squares)

Voting Classifier

- As of now the same base estimator is used
- How about if we want to combine different type of base estimators
- Hard Voting - Same weightage for different algorithms
- Soft voting - Different weightage for different algo
- How to fig out the best combination of weightage