Ensembling Techniques.

- Bagging ✓
  - Random Forest
- Boosting. ✓
  - Adaboost
  - XGboost

→ Hyperparameter Tuning.

Pre-requisite. ✓

- → Basic Knowledge of ML } Covered in Part -1
- → classifier Algo
  - KNN
  - → Logistic Regression.
  - → Decision Tree.

Simple Ensembling. → Combining the result from diff. Models

Majority Voting. → Classifier.

1. KNN → Non-spam
2. logistic Regression → Spam
3. Decision Tree. → Spam

2/3 → Spam.

Averaging: → Regression.

Advance techniques:

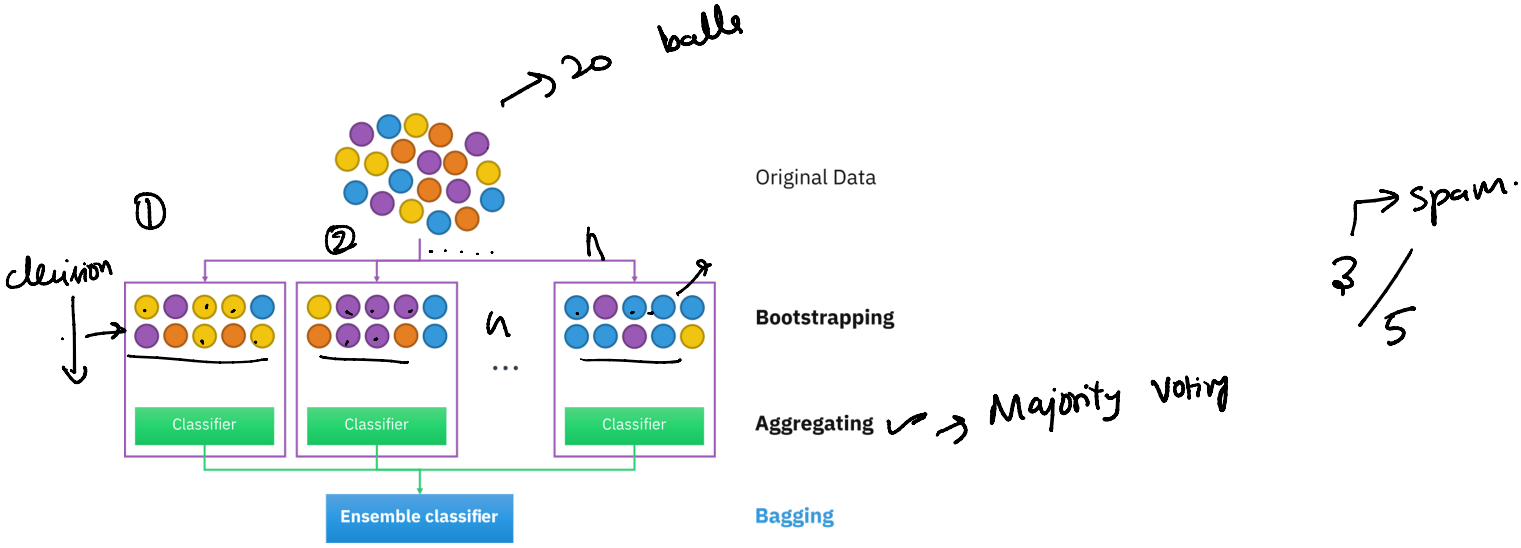
1. Stacking.
2. Bagging.
3. Boosting.

Bagging → Random Forest.

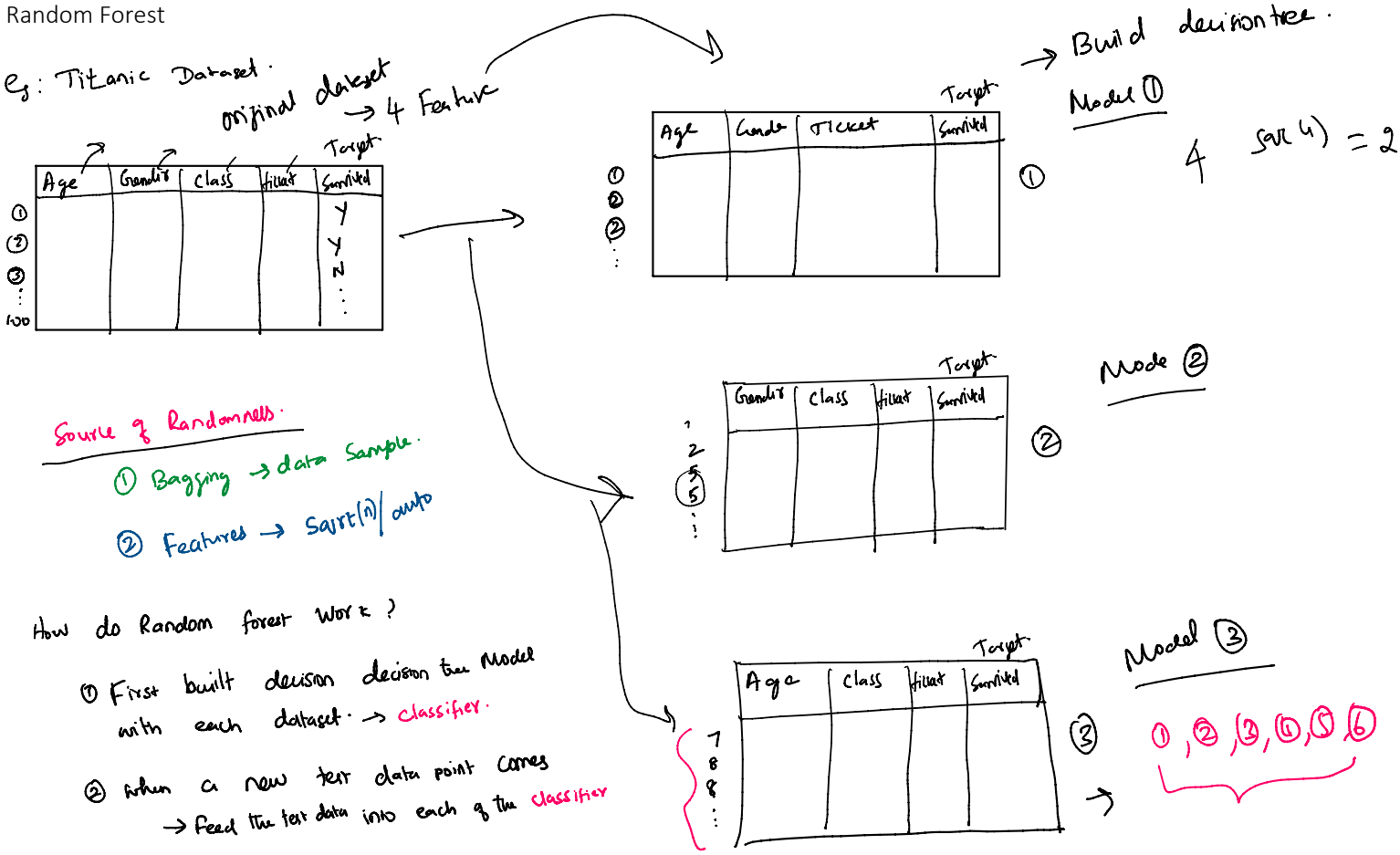
Baggiy → Boostrap + Aggregation.

Boostrap

↳ Sampling with replacement.



Credit : https://commons.wikimedia.org/wiki/File:Ensemble_Bagging.svg


Eg: Titanic Dataset.
original dataset → 4 Feature
Age
Gender
Class
Hillar
Survived
Target
Build decision tree.
Model ①
Model ②
Model ③
Source of Randomness.
① Bagging → data Sample.
② Features → sqrt(n)/ auto
How do Random forest work ?
① First built decision decision tree Model with each dataset. → classifier.
② When a new test data point comes → Feed the test data into each of the classifier
③ Use Majority Voting to finalize the prediction.
① ② ③ ④ ⑤ ⑥
Hyper parameter
n_estimator = No. of trees.
oob = out of bag sample.
oob_score → Test score

Boosting

→ R.F

Bagging :1) The trees are independent.

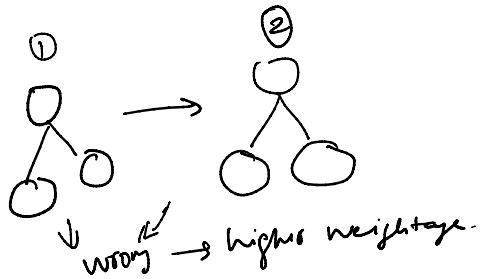
2) The result of one tree doesn't affect another tree. — Can run in parallel

Issue:

Bagging doesn't address the issues of previous tree

Base learner:

Complete decision tree



Boosting :

1) Boosting address the issue of bagging.

2) It builds on the mistake of previous tree

3) It runs sequentially and give more importance to mistakes made by previous tree.

Base learner: stump / tree with 2-3 level deep

: Weak learners.

Ada boost

→ Base learner: Stump 

→ Each stump has diff. say in the final decision.

→ At start each record has equal weightage.

→ Identify the best split [here is based on Weighted (Gini impurity)]

→ We check the prediction which went wrong.

→ Change and give higher weightage to the data sample which are wrong.

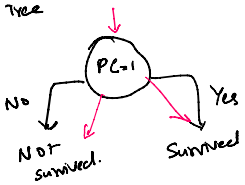
→ Repeat the same process again with the new weight we calculated.

Lets go through an example.

Passenger Class	Age	Gender	Survived	Initial Weight	We off 1st tree	Normalized Weight
1	35	M	Y	1/6	0.12	0.125
1	47	F	N	1/6	0.24	0.25
1	40	F	Y	1/6	0.12	0.125
3	45	M	N	1/6	0.12	0.125
3	35	F	N	1/6	0.12	0.125
3	55	M	Y	1/6	0.24	0.25
					0.96	

First

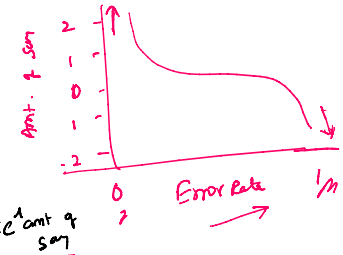
① Decision Tree



② Error Rate = Wrong prediction. / Total Sample = 2/6

③ Amt of say = 1/2 log (1 − Error Rate / Error Rate)

graph



④ New weight

Incorrect Sample = existing weight × $e^{\text{amt of say}}$

Correct Sample = existing weight × $e^{-\text{amt of say}}$

5 Tree

Survived (0.8)

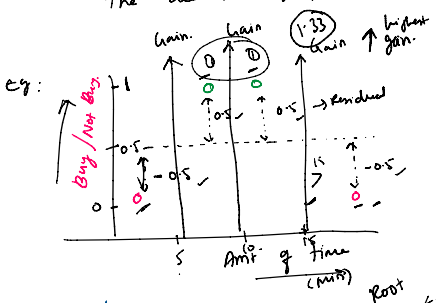
① 0.2
② 0.4
③ 0.2

Non Survive (0.9)

① 0.4
② 0.5

XGBoost tree

The tree building process involved in XGBoost is different.



1) Initial probability = 0.5 (first tree)

2) Calculate the residual

Residual = Actual − Predicted

3) Similarity Score = $\frac{(\sum \text{Residuals})^2}{[\text{prev. probability} \times (1 - \text{prev. probability})] + \lambda}$

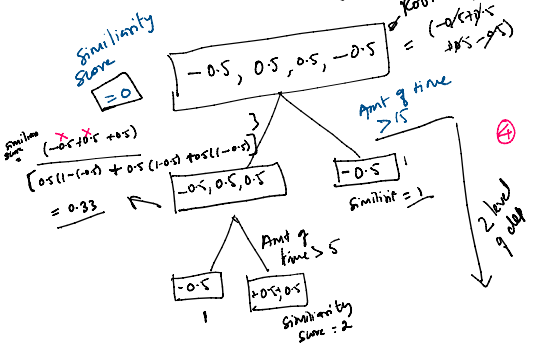
λ, eta, γ

→ to determine which points to split based on.

→ Regularization parameter.

4) Gain = Left similarity score + Right similarity score − Root similarity score.

↳ choose the split which has highest gain.



Gain = 0.33 + 1 − 0 = 1.33

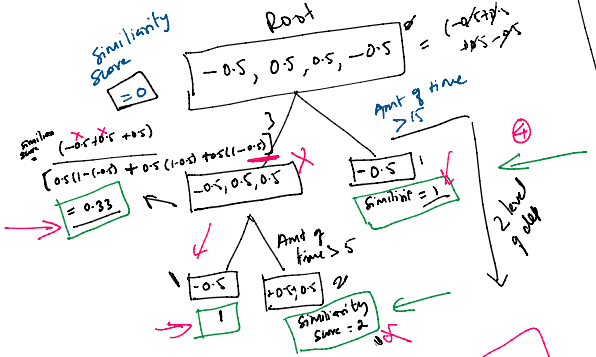
Gain = 1 + 2 − 0.33 = 2.66

Gamma (γ) — pruning parameter

→ Avoid overfitting.

If $\gamma >$ Gain, We prune the tree.

$\gamma \geq 0$



Gain = 0.33 + 1 − 0 = 1.33 → ① for $\gamma = 2$

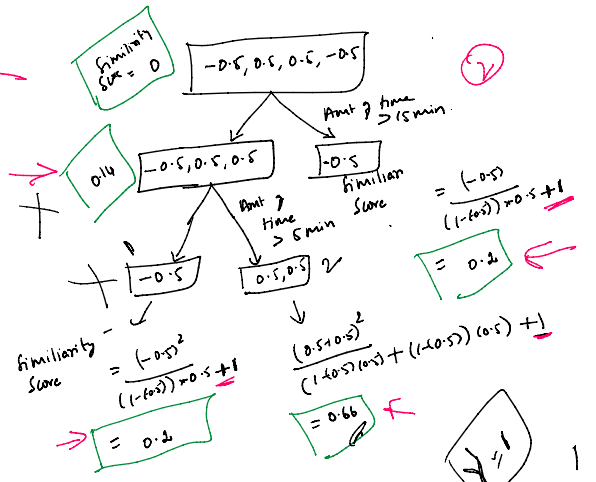
Gain = 1 + 2 − 0.33 = 2.66 → ②

Overfitting the training data

Regularization parameter — λ

① Reduces similarity score / Gain. $\lambda = 1$

② Reduces overfitting → influence of single/isolated value.



Gain = 0.14 + 0.2 − 0 = 0.34

Gain = 0.2 + 0.66 − 0.14 = 0.72

$\gamma = 1$

1 → 0.2 (less)

2 → 0.66 (66%) ↓ dissimilarity.

Random , Grid Search & Bayesian Optimization

Parameters:

n_estimators : *int, default=100* → RF hyper-parameter.
The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *{"gini", "entropy"}, default="gini"* → split
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : *int or float, default=1*
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.
- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Grid Search.

n_estimator = [100, 150, 200, 500]

criterian = ["gini", "entropy"]

max depth = [100, 200, 500, 1000]

min_sample split = [2, 4, 6, 16]

min_sample_leaf = [1, 2, 4, 8]

Grid search.

1) Exhaustive Search.

$4 \times 2 \times 4 \times 4 \times 4$

$= 512$ Combinations

2) But timing consume

Random Search.

$n = 30$ iteration

↓

30 combination at Random

3) Bayesian Optimization.

1) Probabilistic model based approach.

2) Evidence based.

Objective function → "Loss function"

↳ Minimize fn.

by changing the hyperparameter

3) starts with Random hyper parameter.

↳ identify the space with good performance

↳ then choose other hyper parameter based on the evidence found earlier.

Hyper opt.

1. Parameter space
2. fix the objective fn.
3. Choose an algo
   - Gaussian methods
   - Parzen Estimator etc...

Steps:

① Start with default prediction → 0.5 → first classifier.

② Calculate similiarity score

Residual = Actual − predicted

Similarity Formula = $\frac{(\Sigma \text{ sum of residuals})^2}{\Sigma \text{ prev probability} \times (1 - \text{prev probability}) + \lambda}$

③ Calculate Gain.

Gain = Left similiarity score + Right similiarity score − Root similiarity score. ↑ >max.

④ Prune the tree if Gamma (γ) > Gain.

⑤ Calculate the output of the leaf using.

$$\text{output of leaf} = \frac{\Sigma \text{ Sum of Residuals}}{\Sigma \text{ prev probabilities} \times (1 - \text{prev probability})}$$

→ out put in Log (odds)

⑥ Final prediction

output (log of odds) in log odds = log of odds of previous probability + learning rate (eta) × output of leaf

= 0 + 0.3 × −2

= −0.6

output probability = $\frac{e^{\text{log of odds}}}{1 + e^{\text{log of odds}}}$ | $= \frac{e^{-0.6}}{1+e^{-0.6}}$

= 0.35

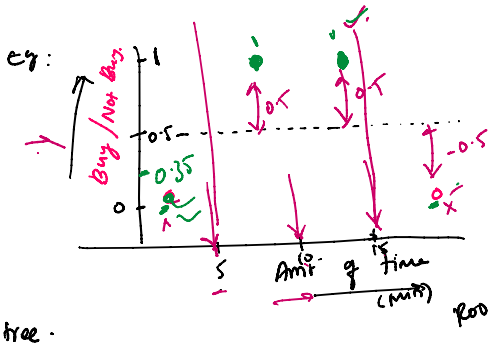
Probability to log odds

$\log\left(\frac{p}{1-p}\right)$ → 0.5 = $\log\left(\frac{0.5}{1-0.5}\right)$

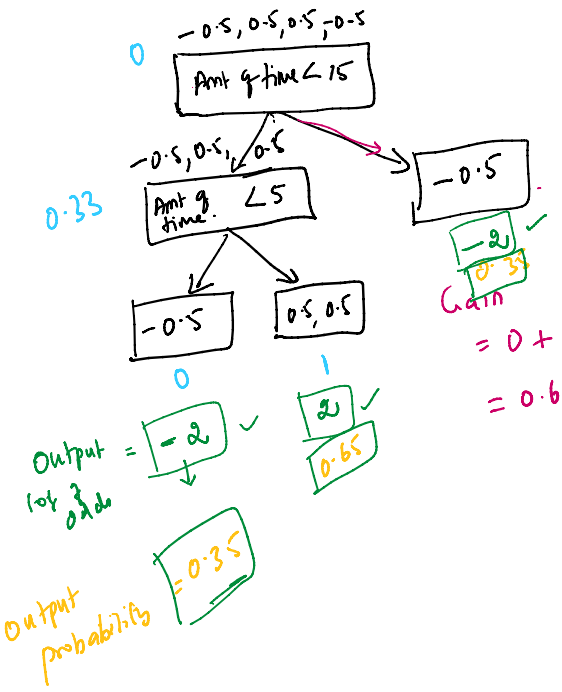
= log 1

= 0

eg:



XGBoost tree.

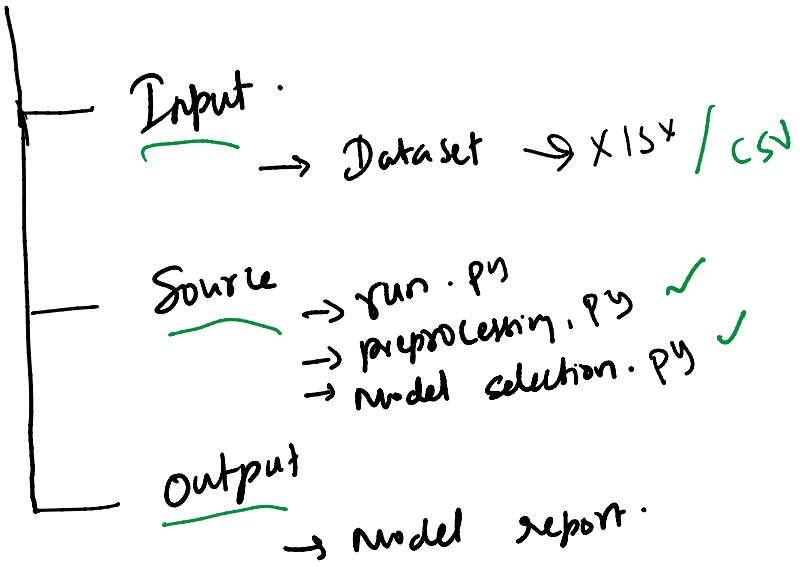


0 to 1

Gain
= 0.33 + 1 − 0 = 1.33

Gain
= 0 + 1 − 0.33
= 0.66

Conclusion & verdict on model performance



	model	precision_score	recall_score	f1_score	model_type
1	LogisticRegression	0.263339	0.242090	0.228505	Imbalanced
5	AdaBoostClassifier	0.542182	0.404654	0.437416	Imbalanced
2	GaussianNB	0.593930	0.749998	0.553084	Imbalanced
0	KNNClassifier	0.837556	0.833380	0.835141	Imbalanced
3	DecisionTreeClassifier	0.876315	0.920471	0.895335	Imbalanced
4	RandomForestClassifier	0.943305	0.944560	0.943913	Imbalanced
6	XGBoostClassifier	0.955169	0.942856	0.948765	Imbalanced