Ensemble Technique in Decision Tree

Ensemble technique → It is a technique where involve multiple decision maker. (Used to control the overfitting in Decision Tree) inbuilt

- Bagging → In it multiple inside decision maker. Here every one will make decision and final decision will be there. (Same nature of decision maker)

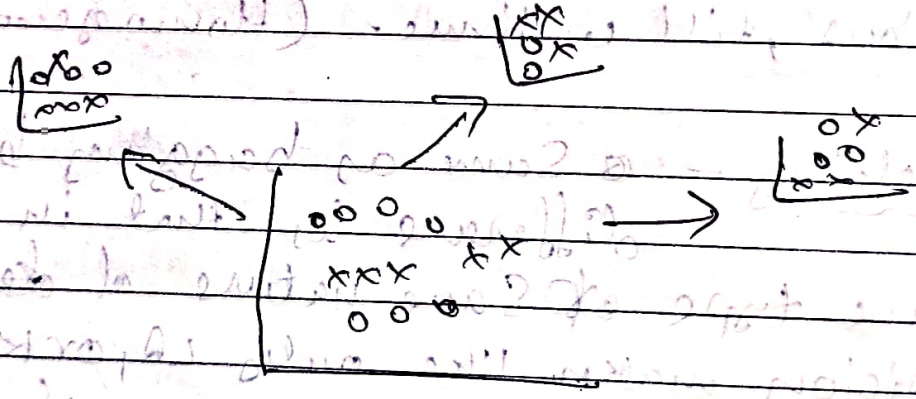
- Boosting - Multiple decision maker but one will be a weak decision maker, next one will better than previous, till continue. (Homogenous Algo)

- Stacking - Same as bagging but the differance is that in bagging same type of ~~same nature of decision~~ decision maker like only LR, or KNN in bagging. But in Stacking all could be different one could be LR one could be KNN.

Note :- In Bagging homogeneous Algorithm are there where as in Stacking hetero-geneous Algorithm.

- Ensemble technique — one technique that aims at improving the accuracy of results in models by combining multiple model instead of using a single model.

- Bootstrap Aggregation (Bagging) — It is a way in which we divided our dataset into a different, different sample with & overlap or replacement.



- it decrease the Variances without increases of Bias. (because it is dividing into multiple dataset sample)

- Here data might overlap.

But

In Pasting technique the data will not overlap.

Random-forest

Bootstrap tech is also known as Bagging,

Ensemble Technique

Bagging	Boosting	Stacking
S.V.M	Ada-Boost	
KNN	Xg Boost	
Naive Bais	Gradient Boost	
Decision Tree		
Random forest		

Random-forest

Random-forest – It is one of the algorithm and a part of ensembel technique work with bagging process.

Note → we want to take ML model we always try to take a model which have low bias & low variance but to get two simultaneously is very hard and difficult because when bias is low than Variance is high & when Variance is low than bias is high. So, To solve this problem we use Regularization, Bagging & Boosting.

Date
Page

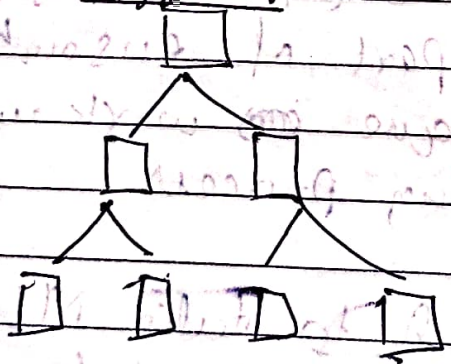
Note ⇒ In Bagging we should always try to take a model who has low Bias & High Variance (LB & HV) like (Decision Tree (fully) (Knn)

but

In Boosting we should always try to take a model who has High Bias & low Variance for ex (shallow decision tree) (dept should be very less), (decision stumps)

Note - In Bagging model learn parrallel & In Boosting model learn sequence

Bagging



Boosting

Note — Ensemble Technique is used to control the Overfitting in Decision Tree.

Boosting

Boosting $\rightarrow$ $w_y = w_o \pm \eta \Delta F_w$

$\eta \rightarrow$ city

$\Rightarrow \rightarrow$ better

Boosting $\rightarrow$ In Boosting we combine weak learner to make a overall Strong learner.

Ada. Boosting

Ex-	f_1	f_2	f_3	O/p	Weight	Update weight
	-	-	-	✓ Yes	1/7	0.058
	-	-	-	✗ No	1/7	0.058
	-	-	-	Yes	1/7	0.058
Incorrect Record ←	-	-	-	↑	1/7	0.349
	-	-	-	-	1/7	0.058
	-	-	-	-	1/7	0.058
	-	-	-	-	1/7	0.058

∴ Here Weight is created for Ada Boosting

TE - Total Error = 1/7

Performance of Stump (P.S) $= \frac{1}{2} \log_e \left(\frac{1 - TE}{TE} \right)$

$$PS \Rightarrow \frac{1}{2}\log_e\left(\frac{1-1/7}{1/7}\right) = 0.895$$

Now for New sample —

for,

Correct Records

$\Rightarrow$ old weight $\times e^{-Ps}$

$\Rightarrow \frac{1}{7} \times e^{-0.895}$

$\Rightarrow 0.05$

} for correct Record and we will decrease in new sample of weights

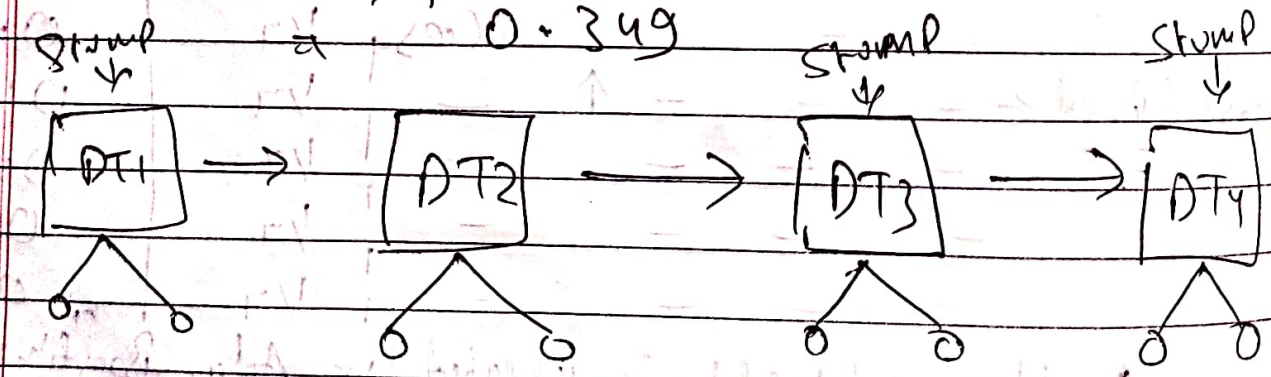
and

for, Incorrect Record

$\Rightarrow$ weight $\times e^{+Ps}$

$\Rightarrow 1/7 \times e^{+0.895}$

$\Rightarrow 0.349$



Weak learner $\Rightarrow$

output 0 1 1 1

Majority Voting $\Rightarrow$ Classification Stump

Mean/Average $\Rightarrow$ Regression Ans = 3/4

① What is the white box & Black box

LR / Decision Tree
⇓
White Box

Ada Boosh
⇓
Black box

In Deep learning what ever Neural Network is used all are black box.

② Data leakage → Dataset — Train → Model
— Test

:- Gradient - boosting - tree :-

i) Gradient boosting - It is a succer of Adaptive Boosting where the approach is little different fro Ada Boost.

New value = Old value + learning rate * residual

where, old value is average of outcome label
learning rate range from (1.0 - 0.0001)

∴ We always try to keep residual as low as possible.

∴ Note → Gradient Boosting used for both Regression & classification problem.

- Pseudo Residual in G.B is the kind of error

$$f(x) = h_0(x) + \alpha_1 h_1(x_1) + \alpha_2 h_2(x_3) + \dots \alpha_n h_n(x_n)$$

base model → DT1 → DT2

h_0 = hypothesis, α_1 = learning rate [0-1] (Kind of hyperparameter)

∴ In boosting or in G.B we use α (alpha), learning rate for weak learner to ensure learning.

$$f_n = \text{Base Model} + \alpha_1(DT_1) + \alpha_2(DT_2) \dots \alpha_n(DT_n)$$

here α (alpha) is used to reduce overfitting

— XG Boost Algorithm —

- It improves the gradient boosting even further.

- The objective function of the XG Boost is that, it is a function of a lossfunction + a Regularisation.

Note - In regression our function or (loss function) will be MSE or MAE or other.

- In classification problem entropy or information gain for loss function.

$$obj(\theta) = \sum_{i}^{N} l(y_i - \hat{y}_i) + \sum_{j=1}^{j} \Omega(f_j)$$

l : this is loss function

Ω : this a regularisation

Note :- XGBoost doesn't use entropy or Gini Indices. Instead it utilise gradient (the error terms) and hessian for creatives the trees.

:- XG Boost Classifier :-

Salary	Credit Score	Approval	Residual (0.5 -)
<= 50K	B	0	-0.5
<= 50K	G	1	0.5
<= 50K	G	1	0.5
> 50K	B	0	-0.5
> 50K	G	1	0.5
> 50K	N	1	0.5
<= 50K	N	0	-0.5

(i) Construct Base model

(ii) Construct Tree with root node

(iii) Calculate Similarity Weight

$$\Rightarrow \frac{\Sigma (\text{Residual})^2}{\Sigma \text{Prob}(1 - \text{Prob}) + \lambda}$$

Probability

XG Boost Regression —

Exp	Crop	Salary	Res 1
2	Yes	40K	-11
2.5	Yes	42K	-9
3	No	52K	1
4	No	60K	9
4.5	Yes	62K	11

Res 1 = Average of Salary - Salary

(iii) Similarity weight = $\frac{(\Sigma \text{Residuall})^2}{\text{no. of residual} + \lambda}$

∴ Here λ is a hyperparameter tuning.

Bagging

Code 1 >>> from sklearn.ensemble import BaggingClassifier

>>> bag_dt = BaggingClassifier(DecisionTreeClassifier(), n_estimators=10)

>>> bag_dt.fit(x_train, y_train) ↓ no. of tree

Note – Internally Random forest uses Decision Tree but it is also possible to use some other model or other Algorithm as a base model for a particular bag.

It is not necessary to use always Random forest for Bagging we can use any Base algorithm for Bagging.

V.V.I Q. Is it is possible to use Bagging without Decision Tree or Random forest.

Ans. Yes, it is possible, as a base model we can use any other algorithm.

Code 2 >>> from sklearn.neighbors import KNeighborsClassifier

>>> bag_knn = BaggingClassifier(KNeighborsClassifier(), n_estimators=10)

>>> bag_knn.fit(x_train, y_train)

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> rf = RandomForestClassifier(n_estimators=5)
>>> rf.fit(x_train, y_train)
>>> rf.estimators_
>>> plt.figure(figsize=(20,20))
>>> tree.plot_tree(rf.estimators_[0], filled=True
```

or to show the Decision make

```
>>> plt.figure(figsize=(20,20))
>>> for i in range(len(rf.estimators_)):
>>>     tree.plot_tree(rf.estimators_[i], filled=True)
```

Reading purpose = Google AI paper with code

(Boosting)

Ada — Boosting

- Weak learner → It is a ML model which have a accuracy just 50%.
- Decision Stump → A decision tree which have a max depth as 1 or have only one nodes
- +1 and −1 → Instead of 1 & 0 in classification
- Ada Boost is a Stage wise additive method.

Stage wise + add weak learner Sequence wise additive method → In Ada-Boost it multiply add multiply weak learner.

$[\alpha_1 = a(pha_1)]$

$$\text{Ada Boost} = \text{Sign}(\alpha_1 * h_1(x) + \alpha_2 * h_2(x) + \alpha_3 * h^3(x))$$

Error rate = 1 − accuracy

$$\alpha = \frac{1}{2} \log\left(\frac{1 - \text{error}}{\text{error}}\right)$$

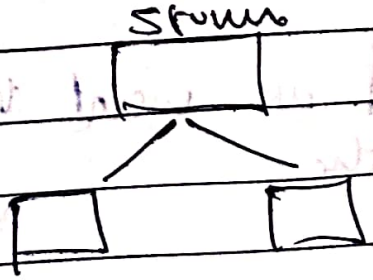
but here error is to add the weight $\left(\frac{1}{n}\right)$ of wrong prediction.

here α is learning rate.

1 Boosting - It is a ensemble modeling technique which improve the prediction power by converting a number of weak learners to strong learner.

2 The principal behind boosting algorithm is that we first build a model on the training dataset and then build a second model to rectify the error present in the first model

3 The most common estimator used with Ada Boost is decision Tree with one level which means Decision tree with only 1 stump



1 Decision tree with one stump / Ada Boost

Date
Page

4 In Ada Boost it builds a model and give equal weight to all the data point. It then assign higher weight to the Wrong classified. Now, all the points with higher weight are given more important in the next model. It will keep training model until and unless a lower error is received.

Model 1 → Model 2 → Model 3 → Model 4
↓ ↓ ↓ ↓
weight 1 weight 2 weight 3 weight 4

Formula of Weight = $\frac{1}{N}$

Code:
```
>>> from sklearn.ensemble import AdaBoostClassifier
>>> clf = AdaBoostClassifier(random_state=96)
>>> clf.fit(x_train, y_train)
>>> clf.score(x_train, y_train)
>>> clf.score(x_test, y_test)
```

∴ Now if we want to do Hyperparameter tuning then

>>> from sklearn.ensemble import RandomForestClassifier

>>> clf = AdaBoostClassifier(random_state = 96,
base_estimator = RandomForestClassifier(random_state=101), n_estimators=10,
learning_Rate = 0.01)

>>> clf.fit(x_train, y_train)

∴ Now check Score of it.

∴ Weight or TE (Total Error) = $\frac{1}{N}$

∴ Performance of Stump (PS) = $\frac{1}{2} \log_e \left(\frac{1 - TE}{TE} \right)$

$\pm PS$

∴ New Sample weight = old weight $\times e^{\pm PS}$

(−) use for correct classified samp

(+) " " wrong " "

Gradient Boosting

1 Gradient Boost algorithm help us to minimize bias error of the model.

2 Similar in ensembling technique the weak learner combine to make a strong model.

3 Gradient Boosting is powerful enough to find any non linear relationship between our model target and features

4 Gradient Boosting - It is one of the variants of ensemble methods where we create multiple weak model and combine them to get better performance as a whole.

- We always try to keep residual as low as possible.

5 Gradient boosting is typically used with decision tree (especially in CART)

formula $\Rightarrow h_0(x) + \alpha_1 h_1(x_1) + \alpha_2 h_2(x_3) + \dots \alpha_n h_n(x_n)$

(Base model = $h_0(x)$; α → hyperparameter, learning rate [0–1]; $h_1(x_1)$ → DT 1; $h_2(x_3)$ → DT 2)

OR

$\Rightarrow$ Basemodel $+ \alpha_1 DT_1 + \alpha_2 DT_2 + \dots$

XG Boosting

1 XG Boosting — It is designed for speed, ease of use, and performance on large Dataset. It does not require optimization of the parameter or tuning.

2 XG Boost Features —

i) XG Boost offer regularization which allows us to control overfitting by introduced L1/L2 penalties on the weights

ii) It also has a block structure for parallel learning

iii) It has a execution speed & model performa

Q How would you explain XG Boost in an interview?

A XG Boost is a robust algo. that can help us improve our ML model accuracy. It is based on gradient boosting and can be used to fit any decision tree-based model.

Stacking

Note :- Bagging & Boosting use Homogenous weak learner model for ensemble whereas, Stacking often consider heterogeneous weak learner.

1. In Stacking we are supposed to do a two level of a data division in general. We are supposed to divided original dataset into first one division and then we are supposed to take a subset of that data as a training and a testing dataset.

Code :-
```
>>> train, val_train, test, val_test = train_test_split
                (x, y, test_size = 0.50,
                 random_state = 30)
```

after this will we create x_train, y_train

```
>>> x_train, x_test, y_train, y_test = train_test_split
                (train, test, test_size = 0.15)
```

∴ Now we will use model

Code 2 [>>> knn = KNeighborsClassifier()
>>> knn.fit(x_train, y_train)

Code 3 [>>> svc = SVC()
>>> svc.fit(x_train, y_train)

Code 4 [>>> prediction_knn = knn.predict(val_train)
>>> prediction_svc = svc.predict(val_train)

∴ We will do prediction from the first split data that is from Code no 1 with val_train

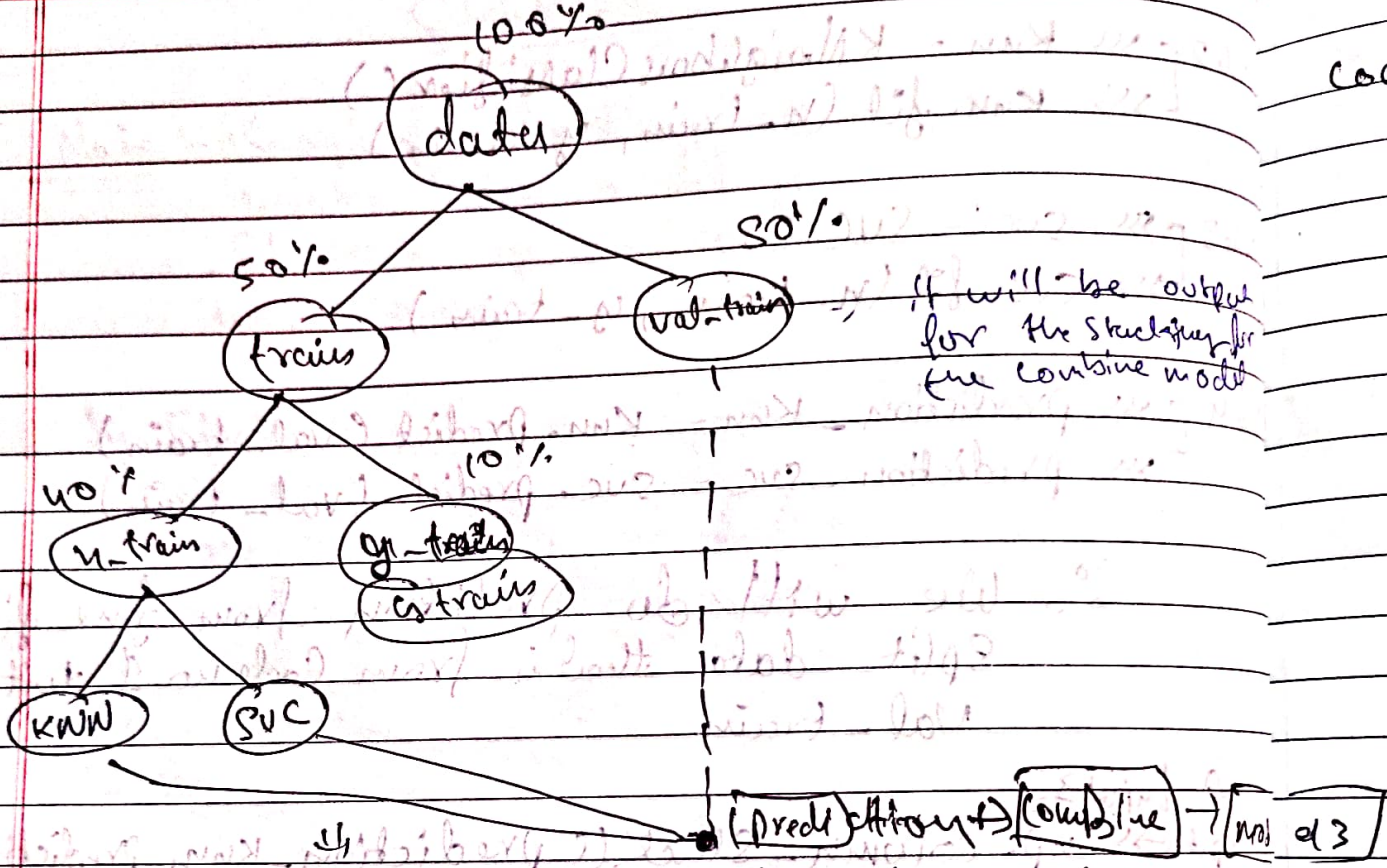
Code 5 input 3 =
input 3 >>> np.column_stack((prediction_knn, prediction_svc))

>>> output = val_test

Code 6

∴ Now we will create model 3 with the input 3 & output

Note - We never pass a data to next Stack we are trying to pass a metadata / derived data of first Stack into a second Stack.



If will be output for the stacking for the combine model

① we will use val_time data for prediction in KNN, SVC and after doing prediction of KNN & SVC we will combine the out put

② After combing the output result we will use it for model 3 or next model.

code (6)
```
>>> rf = RandomForestClassifier()
>>> rf.fit(input3, output)

>>> rf.predict(input3)
```

q3