***

Gradient Boosting

* with in Gradient Boost, The Decision Trees are Pruned To Maximum Leaf nodes from 8 To 32.

Ex:-

X_1 Age	X_2 City	y Income		Prediction $\hat{y}$ (using D.T-1)	Residuals Error	Prediction (D.T-2)	Residuals error		y-Error
32	A	51000		53500	-2500	-5500	3000		48000
30	B	78000	model:	61000	17000	8000	9000	model	69000
21	A	20000	1 →	28500	-8500	-5500	-3000	2 →	23000
27	B	44000		61000	-17000	-4300	-12700		56700
36	B	89000		90500	-15000	8000	-9500		98500
25	A	37000		28500	8500	8000	5800		36500

3000−51000 = 48000

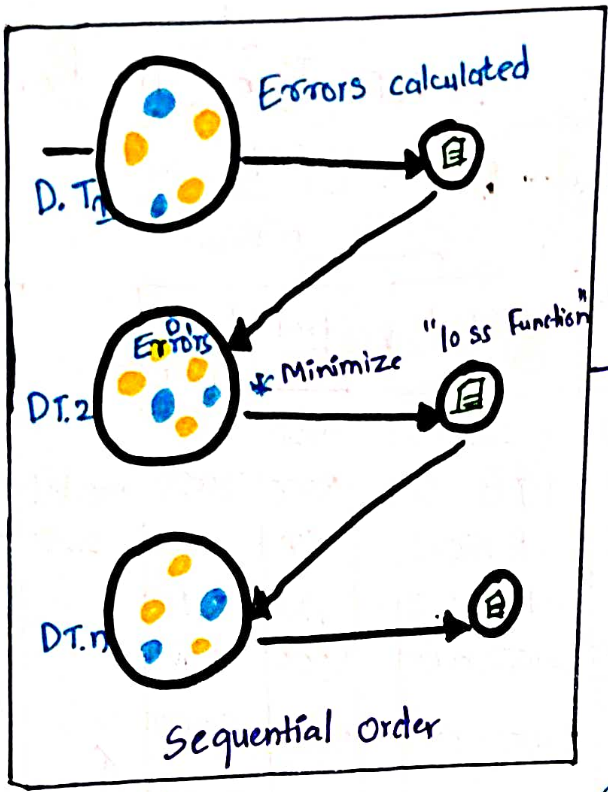
Target $(\text{error})_{min}$

Decision Tree -2

MODEL 2 Income = MODEL 1 income + Predicted Errors

	Model$_0$	M_1	M_2	M_n
Feature	X	X	X	X
Target	Y $\xrightarrow{y-y'}$	e_0 $\xrightarrow{y-y^0}$	e_1 . . . . →	e_{n_1}
	$H_0(x,y)$	$H_1(x,e_0)$	$H_2(x,e_1)$	$H_n(x,e_n)$

* it Continues, till it gets "Minimum Error".



→ GBM
"Gradient Boosting method"

↓ How it works

1. A loss function to be optimized
2. A weak learner to make Predictions
3. An additive mode to add weak learners to minimum, (or) minimize the loss function.

* **Loss Function** :- (Sum of Error)$_{min}$

(or)

(Overall Error should be Minimum)

Que :- What is difference between cost function & loss function ?

Total = 100

Ans :- Ex :-

	Exam	Error
Student 1	70	30
Stundent 2	85	15
Student 3	60	40

→ cost function. (Focusing on only one student)

→ loss function

: cost function :- Error of individual record

loss function :- Overall Error

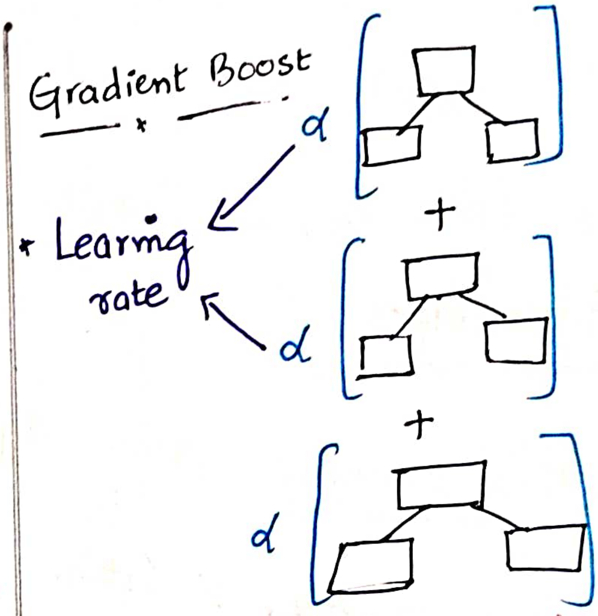
Que :- What is difference b/w "Adaboost" and "gradient boost";

A:- * Adaboost :- We consider $\boxed{\text{Only stump}}$ depth = 1

* Gradient boost :- We Consider Some what grown The Tree, $\boxed{\text{Max. leaf nodes = 8 to 32}}$

How it works

* $\hat{y}$ hat
* [y_actual - y_hat]^2
* y = m(x) + error 1
* error 1 = G [x] + error 2
* error 2 = H [x] + error 3
* y = m(x) + G[x] + H [x] + error 3
* y = alpha * M(x) + beta * G(x) + gamma * H(x) + error 4



1st Tree . calculate residuals / Errors

2nd Tree . Tree after adjusting For Residuals / Errors

Nth Tree . Final Tree after adjusting Errors multiple times.

Data :-

This Data Set includes descriptions of hypothetical Samples corresponding to 23 species of gilled mushrooms in agaricus and Lepiota Family (pp 500-525). Each species in indentified as definetly edible, definitely poisonous, or of unknown edibility and not recommended. this latter class was combined with the poisinous one. the Guide clearly states that there is no simple rule for determing the edibility of a mushroom; no rule like "leaflets three, Let it be" for poisinous Oak and Ivy.

Attribute Information

1. cap_shape : bell = b, conical = c, convex = x, flat = f, knobbed = k, sunken = s
2. Cap - Surface :- fibrous = f, grooves = g, scaly = y, smooth = s
3. Cap - colour :- brown = n, buff = b, cinnamon = c, gray = g, green = r, pink = p, purple = u, red = e, white = w, yellow = y.
4. bruises ? : bruises = t, no = F
5. odor : almond = a, anise = l, creoste = c, fishy = y, foul = f, musty = m, none = n, punget = p, spicy = s

6. gill-attachment : black = k, brown = n, buff = b, chocolate = h, gray = g, green = r, Orange = o, pink = p, purple = u, red = e, white = w, Yellow = y

7. gill-spacing : close = c, crowded = w, distant = d

8. gill-size : broad = b, narrow = n

1. gill-color : black = k, brown = n, buff = b, chocolate = h, grave = g, green = r, orange = o, pink = p, purple = u, red = e, white = w, yellow = y.

. stalk-shape : enlarging = e, tapering = t

stalk-root : bulbous = b, club = c, cup = u, equal = e, rhizomorphs = z, rooted = r, missing = ?

Stalk Surface - above - ring : fibrous = f, scaly = y, silky = k, smooth = s

Stalk Surface - below - ring : fibrous = f, sclay = y, silky = k, Smooth = s

Stalk - colour - above - ring : brown = n, buff = b, Cinnamon = c, gray = g, orange = o, pink = p, red = e, white = w, Yellow = y.

stalk - colour - below - ring : "

Veil - type : Partial = p, Universal = u

Veil - color : brown = n, Orange = o, white = w, Yellow = y

18. ring - number :- none = n, One = o, two = t

19. ring - type :- cobwebby = c, evanescent = e, flaring = f, large = l, none = n, Pendent = p, sheathing = s, zone = z,

20. Spore - print Colour :- black = k, brown = n, buff = b, chocolate = h, green = r, orange = o, Purple = u, white = w, Yellow = y

21. Population :- abundant = a, clustered = c, numerous = n, Scattered = s, Several = v, solitary = y

22. habitat :- grasses = g, Leaves = l, meadows = m, paths = p, urban = u, waste = w, woods = d.

23. Class (Output)

Bussiness Problem

Goal here isto see if we can harness the power of machine Learning and boosting to help create not just a predective model, but general Guideline for features People should look out for when picking mushrooms.

```
# df = Pd.redd_csv("mushrooms.csv")
# df.head()
```

class	cap shape	cap surface	cap color	bruises	odor	gill attachment	gill spacing	gill size	gill color	...	stalk surface above ring	stalk color above ring	stalk color below ring	veil type	veil color	ring number	ring type	spore print color	population	habitat
p	x	s	n	t	p	f	c	n	k		s	w	w	p	w	o	p	k	s	u
e	x	s	y	t	a	f	c	b	k		s	w	w	p	w	o	p	n	n	g
e	b	s	w	t	l	f	c	b	n		s	w	w	p	w	o	p	n	n	m
p	x	y	w	t	p	f	c	n	n		s	w	w	p	w	o	p	k	s	u
e	x	s	g	f	n	f	w	b	k		s	w	w	p	w	o	e	n	a	g

df. shape

Out : [8124, 23]

df. info ()

Out : Range Index : 8124 Entries, 0 to 8123

column	Non null Count	Dtype.
* class	8124 Non null	Object
* cap-shape	"	"
* cap surface	"	"
:		"
* habitat	"	

df. isnull (). sum ()

Out : 0.

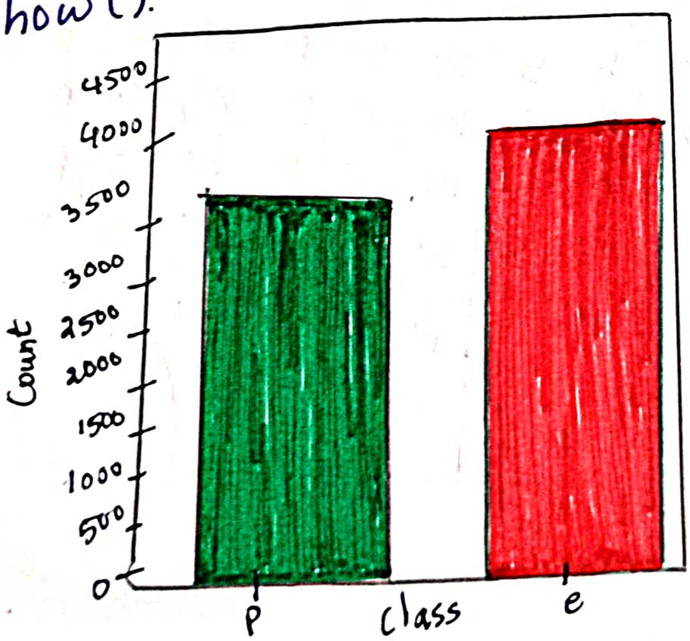
EDA

: sns. countplot (data = df, x = "class", palette = "Dark2")

output

plt. show ().

Out



Transpose. gives total no. columns. without....

df.describe().transpose()

Out

	Count	Unique	top	Freq
(o/p) Class	8124	2	e	4208
Cap shape	8124	6	x	3656
cap-surface	8124	4	y	3244
cap-color	8124	10	n	2284
bruises	8124	2	f	4748
odor	8124	9	n	3528
gill attachment	8124	2	f	7914
gill spacing	8124	2	c	6812
gill size	8124	2	b	5612
gill-color	8124	12	b	1728
Stalk-shape	8124	2	t	4608
Stalk-root	8124	5	b	3776
Stalk-Surface-above ring	8124	4	s	5176
Stalk-Surface-below ring	8124	4	s	4936
Stalk-color above ring	8124	9	w	4464
Stalk-color below ring	8124	9	w	4384
Veil-type	8124	1	p	8124
Veil-color	8124	4	w	7924
ring-number	8124	3	o	7488
ring-type	8124	5	p	3968
Spore-Print color	8124	9	w	2388
Population	8124	6	v	4040
habitant	8124	7	d	3148

* Posinous
* non posinous

CODE :

Gradient Boost :-

Same Data. and Same Procedure from import to X and Y (Mushroom DataSet)

X & Y

```
# x = pd.get_dummies(df.drop("class"), axis = 1 drop_first = True)
# y = df["class"]
```

```
# x.shape, y.shape
```

Out : (8124, 95), (8124,))

train Test split

```
from sklearn.model_selection import train_test_split
# x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.
                                                       random_state = 29)
```

modelling

```
from sklearn.ensemble import GradientBoostingClassifier
# gradmodel = GradientBoostingclassifier()
# gradmodel.fit(x_train, y_train)
```

Out : GradientBoostingclassifier()

Prediction

```
# ypred_train = gradmodel.predict (x_train)
# ypred_test = gradmodel.predict (x_test)
```

Evaluation

* **Accuracy**

```
from sklearn.metrics import accuracy_score

# print ("train accuray", accuracy_score (y_train, ypred_train)
# print ("test accuray:", accuracy_score (y_test, ypred_test)
```

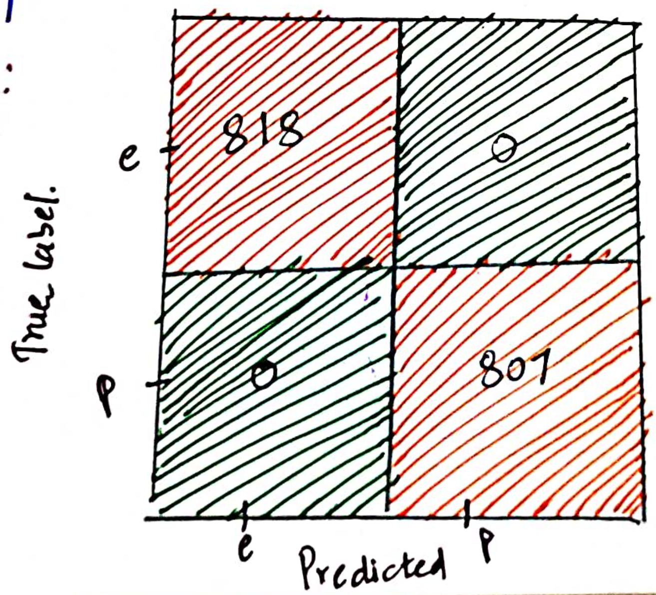
Out: train accuracy : 1.0
Test accuracy : 1.0

* **Confusion Matrix**

```
from sklearn.metrics import plot_confusion_matrix

# plot_confusion_matrix (gradmodel, x_test, y_test)
# plt.show ()
```

Out:



* Classification report

from Sklearn .metrics import Classification_report

```
# Print (classification_report (y_test, ypred_test))
```

Out :

	Precision	recall	f1score	Support
e	1.00	1.00	1.00	818
p	1.00	1.00	1.00	807
				1625
accuracy			1.00	1625
macro avg	1.00	1.00	1.00	1625
Weighted Avg	1.00	1.00	1.00	1625

* Cross-validation Score

from Sklearn. model_selection import . Cross_Val_Score

```
# scores = cross_Val_score (gradmodel, x, y, cv=5)
# Print ( "cross_val_score :", scores.mean())
```

Out : cross_val_score : 0.9192

* feature importance

```
# gradmodel. feature_importances_
```

Out : array ([0.00e+00 , 1.0465 , 1.93018 , 0.000e+0,
6. 19492 , 1.36 263 , 4.49731 -3.5778, ...])

Hyper Parameter Tunning

from sklearn.model_selection import GridSearchCV

Important Parameter for GradientBoost

\# estimator = GradientBoostingClassifier()

\# param_grid = {"n_estimators": [1,5,10,20,40,100], "learning rate": [0.1, 0.2, 0.5, 0.9]}

\# grid = GridSearchCV(estimator, param_grid, scoring="accuracy", cv=5)

\# grid.fit(x_train, y_train)

\# grid.best_params_

out: {"learning_rate": 0.2, "n_estimators": 100}

Final Model

\# final model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.2)

\# final model.fit(x_train, y_train)

\# ypred_train = final model.predict(x_train)

\# ypred_test = final model.predict(x_test)

\# print("train accuracy", accuracy_score(y_train, ypred_train))

\# print("test accuracy", accuracy_score(y_test, ypred_test))

out: train accuracy: 1.0

test accuracy : 1.0

final model . feature_ importances_

Out : array ([0.000e+00, 2.45746e-16, 1.27008 , 4.735
1.137228 1.485440 1.923168 , 3.20965)
: . 0.000be0]

important = pd. Data Frame (index = x.columns, data = final model. feature_ importances , Columns = ["importance_feature"]

important

Out :

	importance_Feature
Cap_shape_c	0.0000e+00
cap_ shape_f	2.45745e-16
cap_ shape_k	1.270085e-23
cap_ shape_s	4.735679e-08
⋮	⋮
habitat_u	9.502012e-05
habitat_w	0.00000e+00

Jack = important [important ["importance_feature"] > 0.01]

Jack. Sort_Values (" importance_ Feature")

Out :

	importance_features
gill_size_n	0.010084
Spore print color_h	0.014901
odor_l	0.019874
Spore print color_r	0.032914
stalk root_r	0.052216
bruises_t	0.060678 . . . .

plt.figure (figsize=(14,6), dpi=200)

# sns. barplot (data= Jack. sort_values ("importance_feature"),
x = Jack. index , y= "importance_feature")

plt . xticks(rotation = 90)

plt. show()

Out :