26/04/22 10:06pm

****

Ensemble Learning

Ex :- Avengers.

Boosting Technique :- it is methodology, not a m.L Algorithm, it is applied to an Existing m.L, mostly applied on Decision Tree.

Ex :- Student

* Exam

⇒ not prepared well, Focus on wrong answers only

* Attempt Exam again

⇒ rectified but missed another section

* Again Exam

⇒ Total correctly done.

* Boosting :-

⇒ Decision Tree 1 :- misclassified records

⇒ Decision Tree 2 :- misclassified records

⇒ Decision Tree 3 :-

* 2nd Weak learner is dependent on 1st weak learner

↓

algorithm doesn't give more accuracy.

* Formula for Boosting :-

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \rightarrow f_t(x) = \alpha_t \, h(x)$$

Summation of t=1 to T

Learning rate

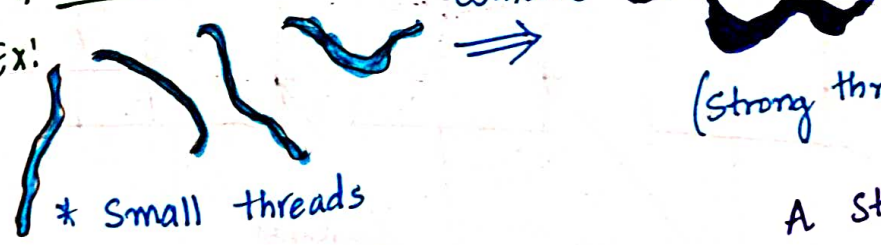
$h(x)$ → [Each model]

* boosting is the Combination of weak learners.

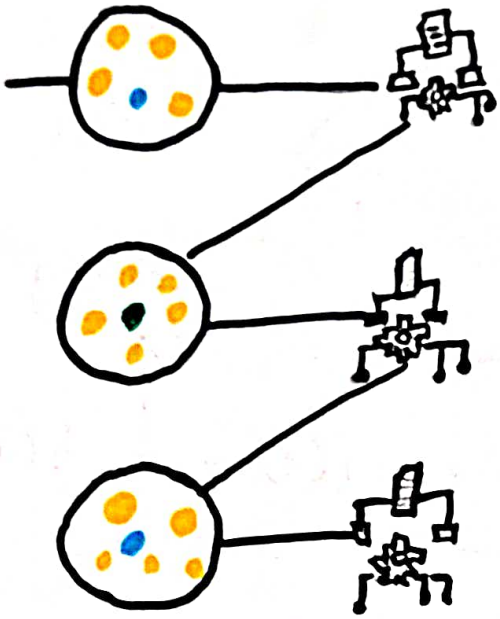
* Implies that Combination of Estimators (models) with an applied Coefficient Could act as an effective Ensemble Estimator.

Boosting

Ex:



* Multiple Weak classifer (Stumps)



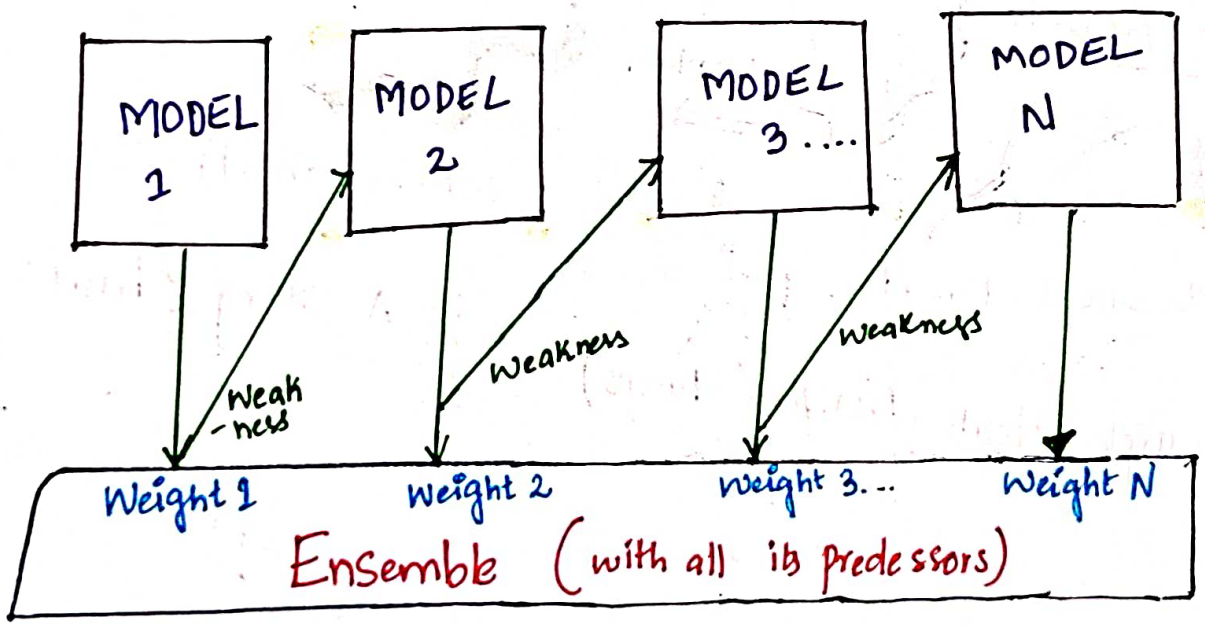
"Sequential learning":

* From the first algorithm whatever mistakes was done
* we share to the second algorithm.
* again it shared to third algorithm. - - -
* it is in "Sequential Order"

Boosting - "Sequential" Learning.

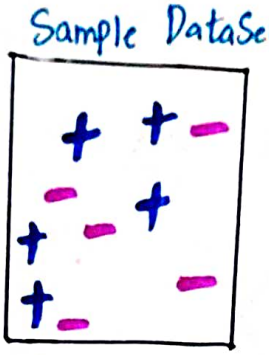
1. Ada Boost → Boosting.
   - Adaptive

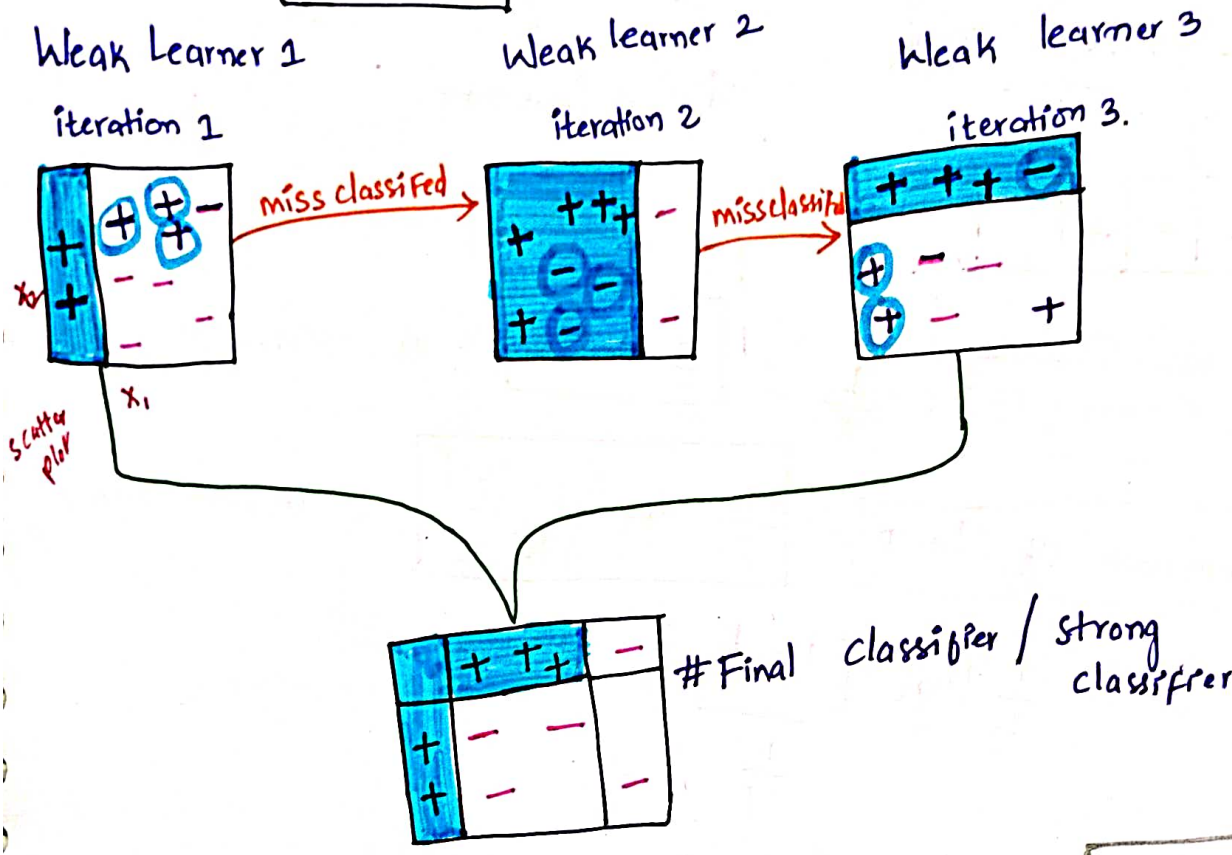
Ex:- Model, 1, 2, ... N are individual models (eg. Decision Tree)



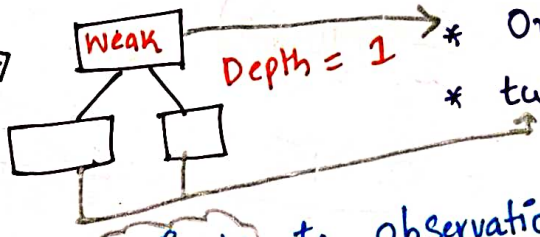
covid (positive / negative)

Ex:-





* In Adaboost :- Each weak learner is called "stump"

* Stump ⇒ 

* One root node
* two Leaf nodes.

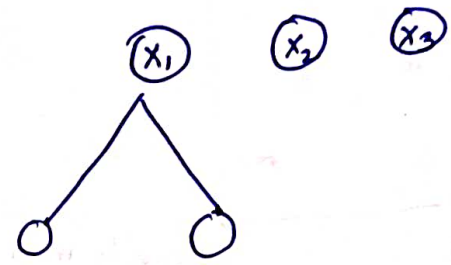
* Step 1 :- Adaboost assigns weights to observations which are incorrectly predicted

Step 2 :- Next model works to predict these values correctly which are wrongly predicted before.

Ex:-

Taken by Entropy / Information gain

	X_1	X_2	X_3	X_4	O/p	weights
1	–	–	–	–	Yes	1/7 ✓
2	–	–	–	–	No	1/7 ✓
3	–	–	–	–		1/7 ✓
X 4	–	–	–	–		1/7 X
5	–	–	–	–		1/7 ✓
6	–	–	–	–		1/7 ✓
7	–	–	–	–		1/7 ✓



{weak Learner}

*1. **Total Error [TE]** :- $\frac{1}{7}$:- $\varepsilon_t = \varepsilon_w$ ∴ ε_t = "Episinol" T

2. **Performance of Stump** :- $\frac{1}{2}\log_e\left[\frac{1-TE}{TE}\right]$ ∴ TE = Total Error

$$= \frac{1}{2}\log_e\left[\frac{1-\frac{1}{7}}{\frac{1}{7}}\right]$$

$$= \frac{1}{2}\log_e\left[\frac{6/7}{1/7}\right]$$

$$= \frac{1}{2}\log_e(6)$$

$$= 0.895$$

New Sample weight = $\text{Weight} * e^{-Ps}$ → correct records ∴ Ps = Performance Stump

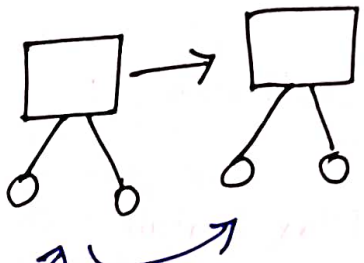
$$= \frac{1}{7} * e^{-0.895} = 0.05$$

+ **incorrect record** = $\text{weight} * e^{Ps}$ → incorrect records.

$$= \frac{1}{7} * e^{0.895} = 0.349$$

is it 1??

Weights	New weights	Normalized weights	Buckets
$\frac{1}{7}$	0.05 ÷ 0.649	0.077	(0 - 0.07)
$\frac{1}{7}$	0.05 ÷ 0.649	0.077	(0.07 - 0.14)
$\frac{1}{7}$	0.05	0.077	(0.14 - 0.21)
$\frac{1}{7}$	0.349	0.537	(0.21 - 0.747) ✓
$\frac{1}{7}$	0.05	0.077	[1.747 - 0.751] ✓
$\frac{1}{7}$	0.05	0.077	—
$\frac{1}{7}$	0.05	0.077	—
	0.649	~ 1	



∴ 1.537 + 0.21 = 1.747

* Steps :-

⇒ Weight = $\frac{1}{n}$

⇒ Total Error_t = $\frac{[\text{correct Prediction} - n]}{n}$

⇒ Total weighted error_t = $\frac{\text{Sum (weight (i) * Error (i))}}{\text{Sum(weight)}}$

⇒ $\alpha_t = \frac{\ln [1 - TE]}{TE\ [\text{Total Error}]}$

⇒ Weight = weight * exp (α_t * Total error)

⇒ total error = 0 if ($y == y^{hat}$), else 1.

* Parameters

* n_estimors :- no. of weak learners
* Cretierion :- Gini / Entropy
* Max_Feature :- all Feature / selected ones
* Max_depth :
* min_samples_split
* n_jobs :- System is quadcore :- no. of cores = 4
  dual core :- no. of cores = 2
  octo core : no. of cores = 8
  -1 (all cores)
* Random_state :
* Learning rate :

CODE :- AdaBoost

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.


→ Mushroom

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

Cap_colour : brown = n, buff = b, cinnamon = c, gray = g, green = r, pink = p, purple = u, red = e, white = w, Yellow = y.

bruises ? : bruises = t, no = F

dor : 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, scaly = 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 is to 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 below 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	"	"

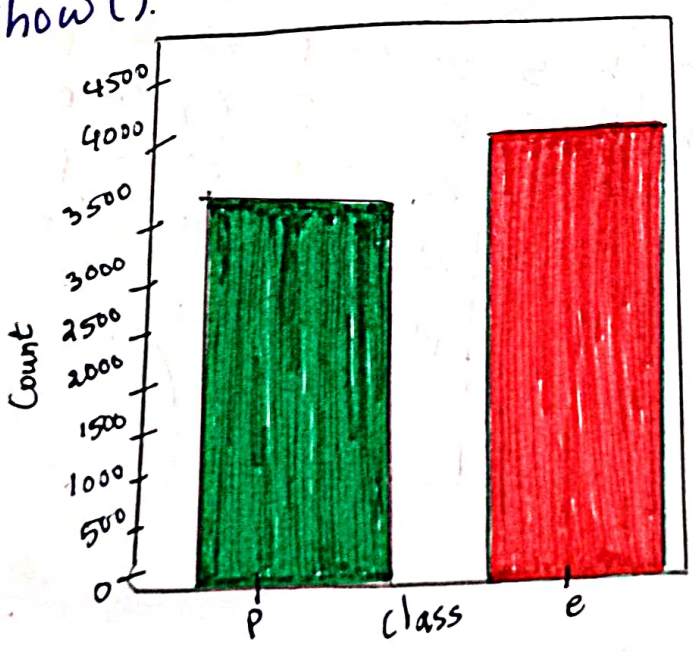
df.isnull().sum()

Out: 0.

EDA

sns.countplot(data=df, X="class", palette="Dark2")
output

plt.show().



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 bebw 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

X & Y

Nomal ... (nominal data) ⇒ One hot Encoding

```
# X = pd.get_dummies(df.drop("class", axis=1), drop_first=True)

# y = df["class"]
```

x

Out

cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x	cap-surface_g	cap-surface_s	cap-surface_y	cap-color_c	cap-color_e	...	population_c	population_s	population_v	population_y	habitat_g	habitat_l	habitat_m	habitat_p	habitat_u
0	0	0	0	1	0	1	0	0	0		0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	1	0	0	0		1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0		1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	1	0	0		0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	1	0	0	0		0	0	0	0	1	0	0	0	0

8124 rows x 95 columns

y

Out:

0	p
1	e
2	e
3	p
4	e

```
# 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 = 0.2, random_state=29)
```

MODELLING

```
from sklearn.ensemble import AdaBoostClassifier

# model = AdaBoostClassifier ()   → n_estimator = 100 (stumps)

# model.fit (x_train, y_train)
```

out : AdaBoostClassifier()

PREDICTION

```
# ypred_train = model.predict (x_train)

# ypred_test = model.predict (x_test)
```

Evaluation

* Accuracy

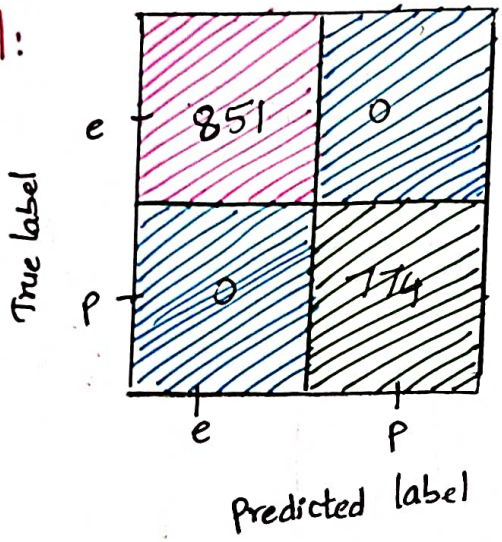
```
from sklearn.metrics import accuracy_score

# Print ("Train accuracy" : , accuracy_score (y_train, ypred_train)
# print ("Test accuray": , accuracy_score (y_test, ypred_test)
```

out:
Train Accuray : 1.0
Test Accuray : 1.0

* Confusion Matrix

```
from sklearn.metrics import plot_confusion_matrix
# plot_confusion_matrix (model, x_test, y_test)
# plt.show()
```

Out:



* classification Report

```
from sklearn.metrics import classification_report
# print (classification_report (y_test, y_pred_test))
```

Out:

	Precision	recall	f1 score	Support
e	1.00	1.00	1.00	851
p	1.00	1.00	1.00	774
			1.00	1625
Accuracy	1.00	1.00	1.00	1625
macro avg		1.00	1.00	1625
weighted Avg	1.00			

* Cross-Validation-Score

```
from sklearn.model_selection import cross_val_score.

# scores = cross_val_score (model, x, y, cv=5)

# print (" cross validation score:", scores.mean())
```

Out : Cross validation score : 0.925

* model.feature_importances

```
# model.feature_importances_

Out  array([0. , 0. , 0. 0, 0. , 0, 0., 0.,
            0. . 0,  0. , 0.02, 0. 0.04,
            0.06, 0.02, 0.0, 0, 0.1, 0.12,
            0 ]

# f_imp = pd.DataFrame (index = X.columns, data = model.feat
                        -re = importances_ columns = ["Feature Import
                                                      ances.
```

```
# f_imp
```

Out: Feature Importances

cap_shape_C	0.00
cap_shape_f	0.00
cap_shape_K	0.00
cap_shape_s	0.00
cap_shape_x	0.00
⋮	⋮

95 rows x 1 columns

Data Extraction (Pandas)

```
# f_imp[f_imp["feature importance"]>0]
```

Out :

	Feature Importance
cap_color w	0.02
odor_c	0.04
odor_f	0.04
odor_n	0.06
odor_p	0.02
gill_spacingw	0.10
⋮	⋮

Hyper Parameter Tuning

```
from sklearn.model_selection import GridSearchCV
# estimator = AdaBoostClassifier()
# param_grid = {"n_estimators": list(range(1,101))}
# grid = GridSearchCV(estimator, param_grid, cv=5, scoring="Accuray")
# grid.fit(x_train, y_train)
# grid.best_params_
```

Out : {"n_estimator : 20}

final model

```
# final_model = AdaBoostClassifier (n_estimators = 20)
# final_model.fit (x_train, y_train)

# Pred_train = final_model.predict (x_train)
# Pred_test = final_model.predict (x_test)

# print ("train accuray:", accuray score (y_train, Pred_train)
# print ("test accuray:", accuray score (y_test, Pred_test)
```

Out: train accuray : 1.0
Test accuray : 1.0

25/04/22
10:00pm

```
# final model.feature_importances_
```

Out: array ([0. , 0. , 0., 0. 0.
0.0434 0. 0, 0, 0.
0 0 0 0 0.434...]]

27/4/22
4:30pm.

```
# f_imp2 = Pd.DataFrame (index = X.columns, data = final model
        .feature_importances_, columns = ["importance-
        Feature"])
```

f_imp2

Out:

	importance_Feature
cap_shape_c	0.0
cap-shape-f	0.0
⋮	
habitat-p	0.0
habitat-u	0.0

important = f_imp2 [f_imp2 ["importance_feature"] >0]

important . Sort_Values (" importance-feature")

Out:

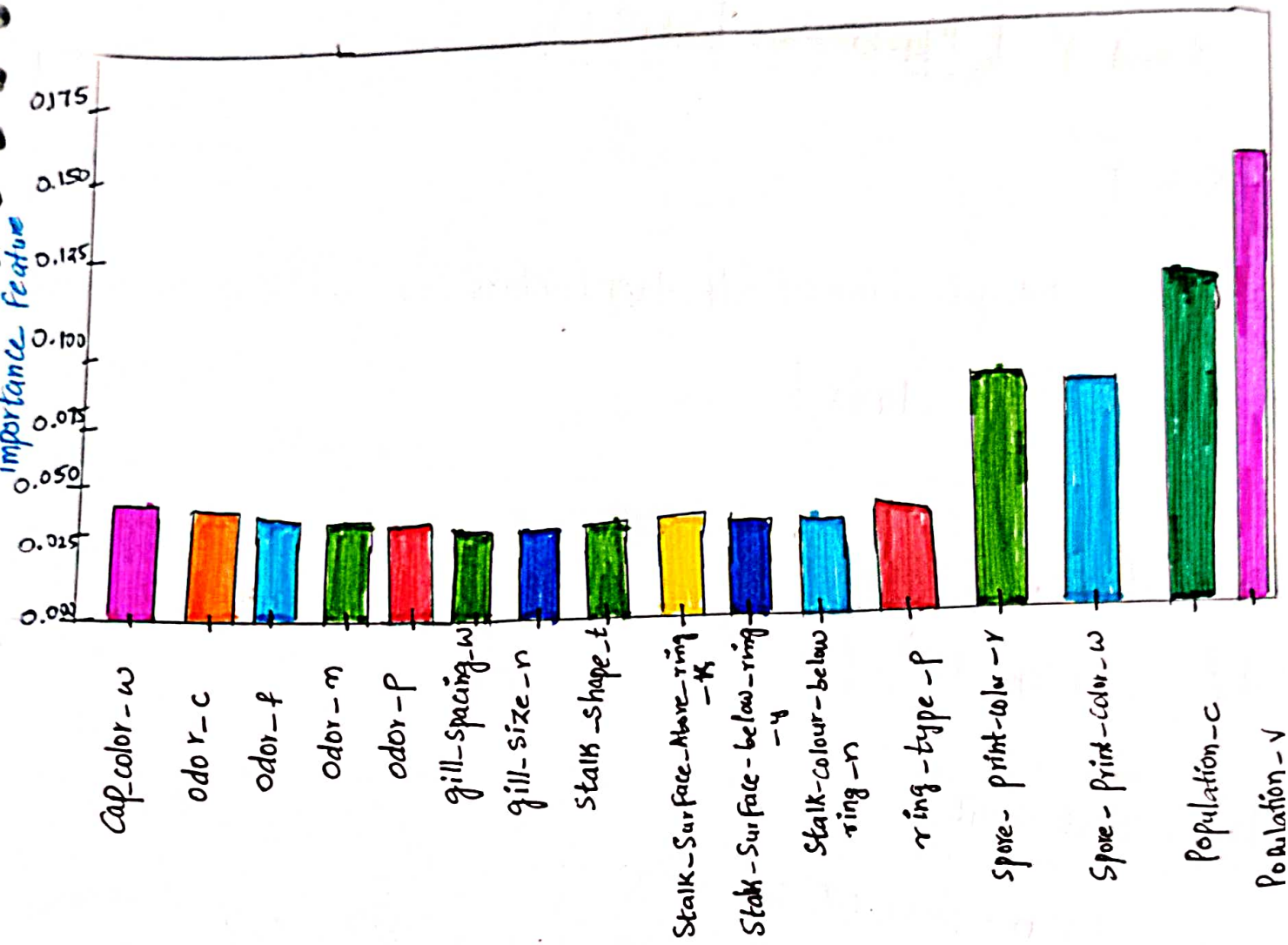
	importance_Feature
cap_color_w	0.043478
odor_c	0.043478
odor_f	0.043478
odor_p	0.043478
⋮	
order_n	0.130435
gill_size_n	0.173913

plt. figure [fig_size = (14,6), dpi=200)

sns. barplot (data = important. Sort_values ("importance_feature"), x = important. index, y = "importance_Feature")

plt.xticks (rotation = 90)

plt. show ()



***

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
32	A	51000		53500	-2500
30	B	78000	model: 1 →	61000	17000
21	A	20000		28500	-8500
27	B	44000		61000	-17000
36	B	89000		90500	-15000
25	A	37000		28500	8500

D.T-2

Prediction	Residuals error
-5500	3000
8000	9000
-5500	-3000
-4300	-12700
8000	-9500
8000	5800

3000-5100 = 48000

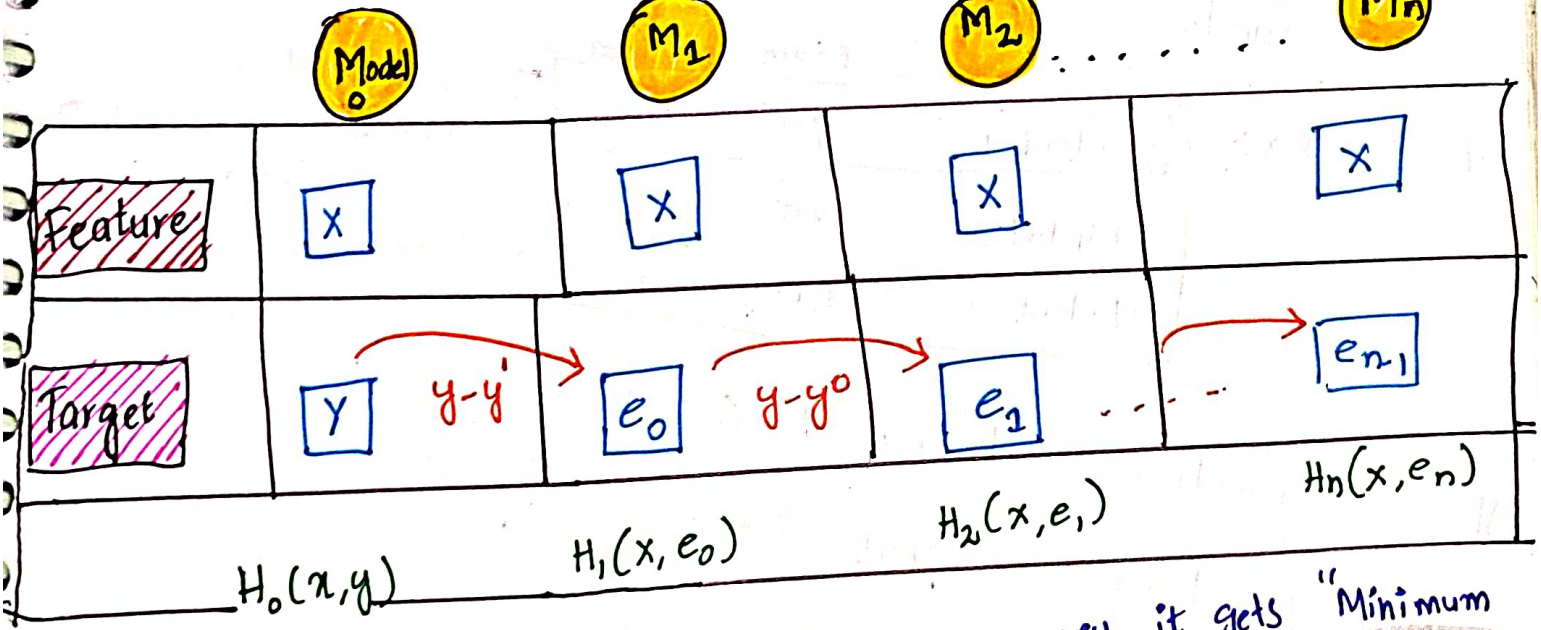
y-Error
48000
69000
23000
56700
98500
36500

model 2 →

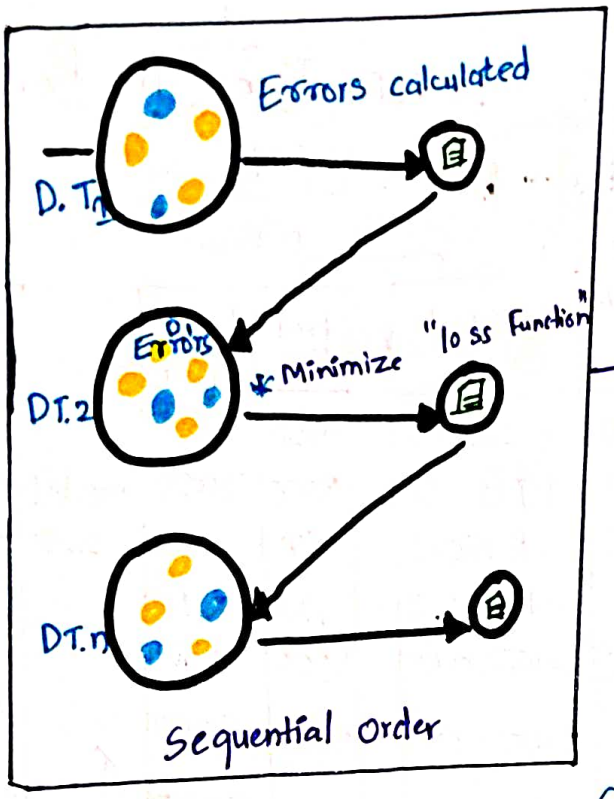
Decision Tree -2

Target (error)min

MODEL 2 Income = MODEL 1 income + Predicted Errors



* 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** :- $(\text{Sum of Error})_{min}$

(or)

(Overall Error should be Minimum)

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

Ans :- Ex :-

Total = 100

	Exam	Error
Student 1	70	30
Student 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 only stump/ depth = 1

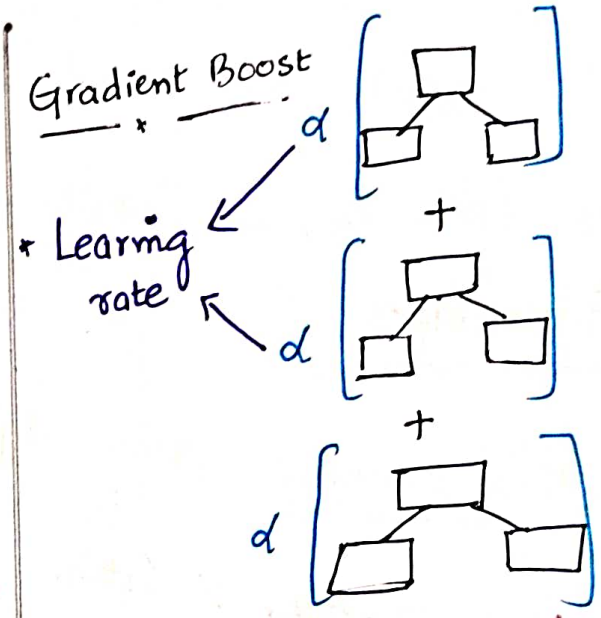
* Gradient boost :- We Consider some what grown The Tree, Max. Leaf nodes = 8 to 32

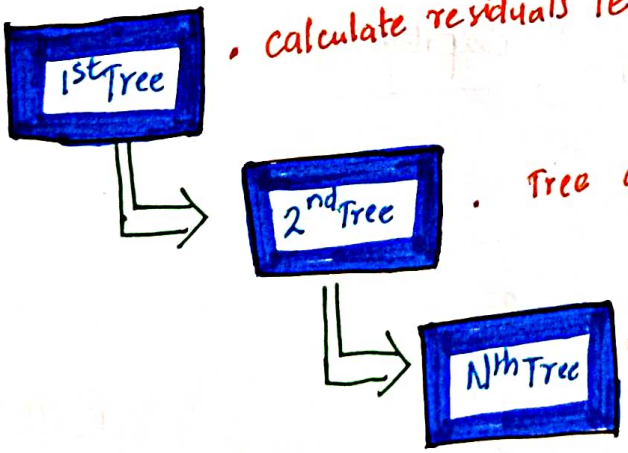
How it works

* $\hat{y}$ 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

Gradient Boost

* Learning rate





. calculate residuals /Errors

. Tree after adjusting for Residuals /Errors

. 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.


→ Mushroom

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
Cap_colour :	brown = n, buff = b, cinnamon = c, gray = g, green = r, pink = p, purple = u, red = e, white = w, Yellow = y.
bruises ? :	bruises = t, no = F
dor :	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, scaly = 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 is to 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 below 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	"	"

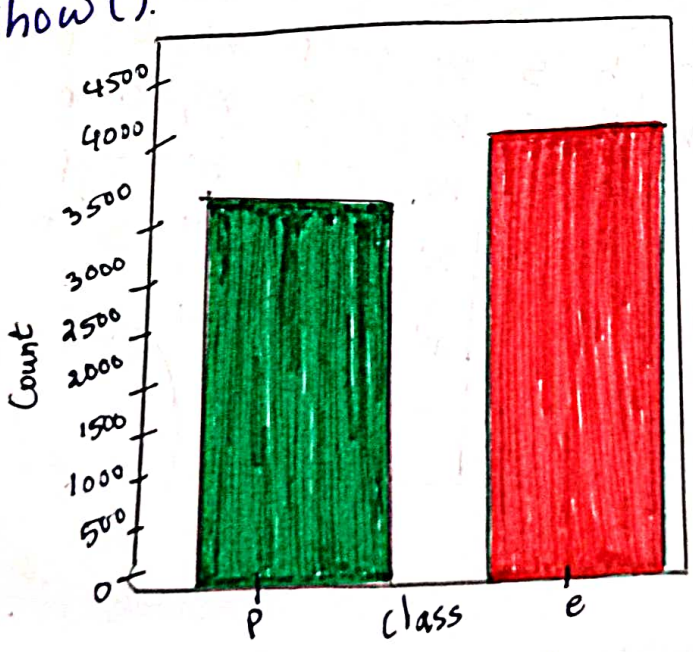
df.isnull().sum()

Out: 0.

EDA

sns.countplot(data=df, X="class", palette="Dark2")
output

plt.show().



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 bebw 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 Xand 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 Gradient Boosting Classifier
# gradmodel = Gradient Boosting classifier ()
# gradmodel.fit (x_train, y_train)
```

Out : Gradient Boosting classifier ()

Prediction

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

Evaluation

* **Accuracy**

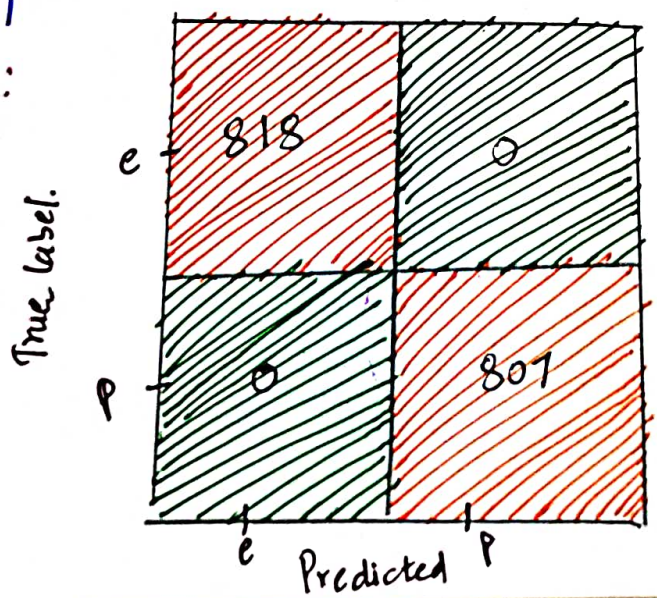
```
from sklearn.metrics import accuracy_score
# print("train accuracy:", accuracy_score(y_train, ypred_train))
# print("test accuracy:", accuracy_score(y_test, ypred_test))
```

Out: train accuracy : 1.0
Test accuracy : 1.0

* **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	f1-score	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, 1.36263, 4.49731 -3.5778, ...])

Hyper Parameter Tunning

```
from sklearn.model_selection import GridSearchCV

# estimator = GradientBoostingClassifier()
# param_grid = {"n_estimators": [1,5,10,20,40,100], "learning_rate":
                                                    [0.1, 0.2, 0.5, 0.9]}
```

Important Parameter for GradientBoost

```
# 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, 2.45745e-16, 1.27008, 4.735
1.137228 1.485440 1.923168 , 3.20965)
0.000 bev]

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	: 0.0000 e+00

Jack = important [important ["importance_ feature"] > 0.01]

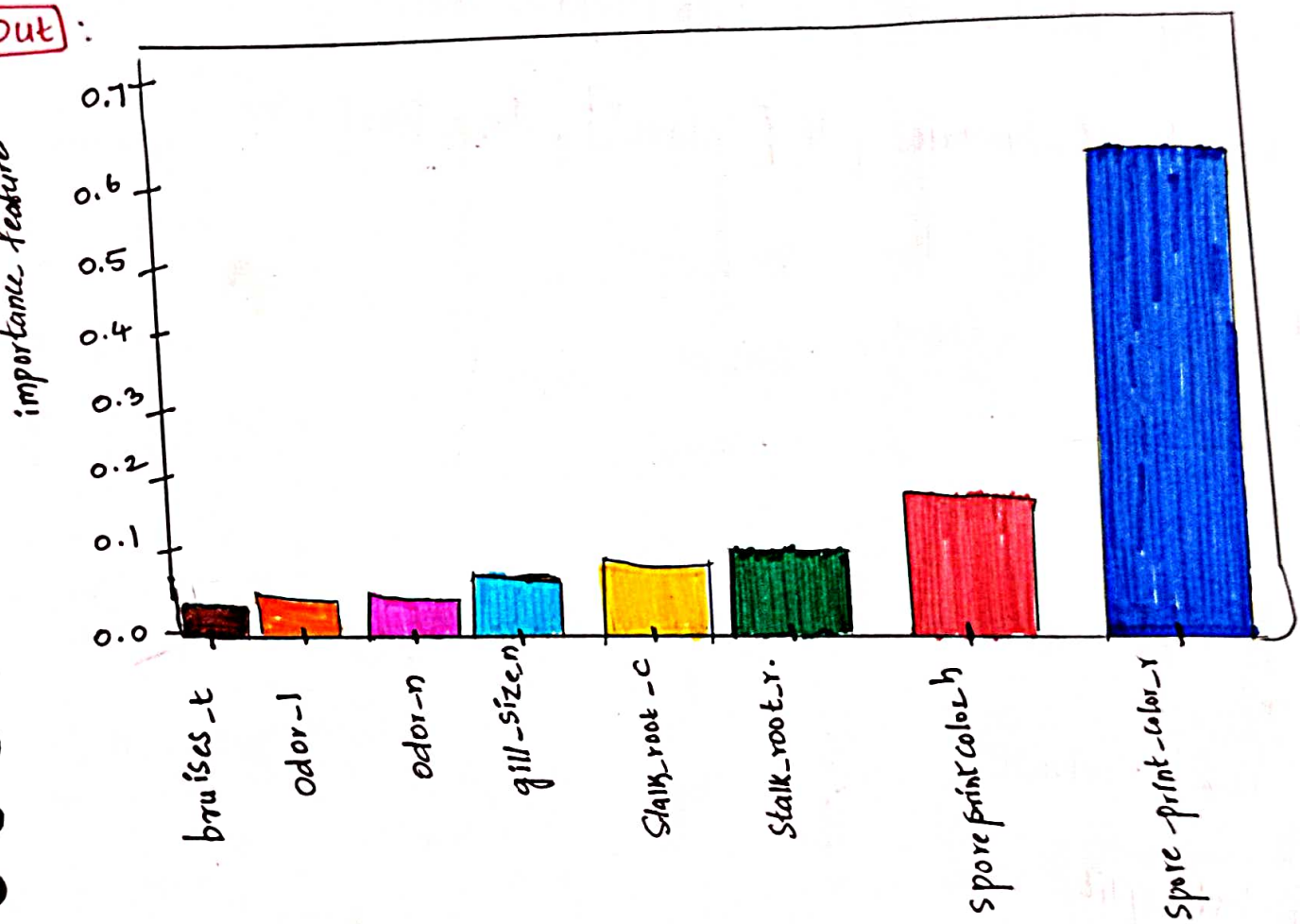
Jack. Sort_ Values (" importance_ Feature")

Out :

	importance_ feature
gill_size_n	0.010084
Spore print color_h	0.014901
odor_l	0.019874
Spore printcolor_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:



**** XG BOOST → Gradient

↓

Xtreme

* Adding "Penality" term Reduce over all Error
* "10" times Faster Than "Gradient Boosting".
* "XG" Boost also called as "Regularized Boosting".
  * Like L_1, L_2 Regularization

* Advantages

* Tree Pruning Using depth first approach
* High-flexibity :- Catche awarness and out of core computing. (-1) it uses all cores in system
* Reduce Overfitting :- "Regularization" for avoiding overfitting

Efficient Handling missing data

Inbuilt "cross-validation" Capability

XG-boost

error + Penality Term

L_2 : α (slope)^2

L_1 : α |slope|

with in "XG Boost :-
α = gamma

* **XG boost classifier** Black box Model.

Base model = Probability = 0.5

Ex:- DataSet (Pr - Approval)

Salary	Credit	Approval	Residuals
<= 50k	Bad	0	0 - 0.5 = -0.5
<= 50k	Good	1	1 - 0.5 = 0.5
<= 50k	Good	1	0.5
> 50k	Bad	0	-0.5
> 50k	Good	1	0.5
> 50k	Normal	1	0.5
<= 50k	Normal	0	-0.5

* **Steps** :-

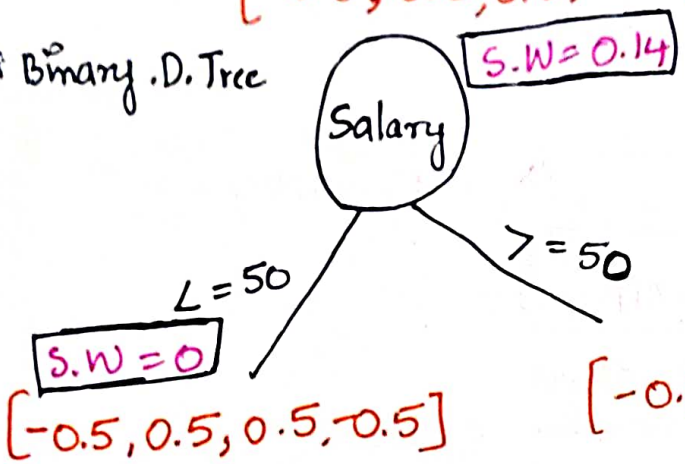
1. Create Binary Decision Tree using Features
2. Calculate Similarity weight

$$= \frac{\Sigma [\text{Residual}]^2}{\Sigma [\text{Probability}\,[1 - \text{Probability}] + \lambda]} \rightarrow + \lambda \text{ (hyper parameter)}$$

3. Information Gain.

Errors :- $[-0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]$

* Binary .D. Tree



$$\text{Similarity weight} \Rightarrow \frac{\varepsilon(res)^2}{\varepsilon p(1-p)\lambda}$$

$$= \frac{[-0.5+0.5+0.5-0.5]^2}{0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0}$$

$\because \lambda = 0$

$$= 0$$

$$\text{Similarity weight} = \frac{\varepsilon(res)^2}{\varepsilon p(1-p)\lambda}$$

$$= \frac{[-0.5+0.5+0.5]^2}{0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+\frac{0}{\lambda}}$$

$$= \frac{0.25}{0.75} = \frac{1}{3} \Rightarrow 0.33$$

For Root node Similarity weight $= \frac{\varepsilon(res)^2}{\varepsilon p(1-p)\lambda}$

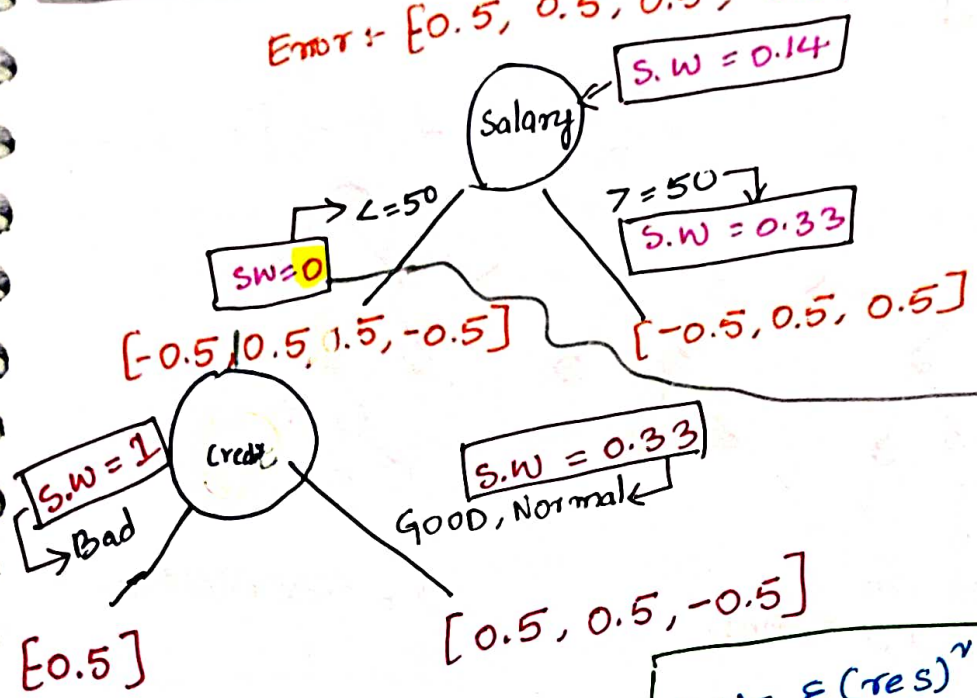
$$= \frac{[-0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]^2}{0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]}$$

$$= \frac{0.25}{1.75} = \frac{1}{7} = 0.14$$

* Information Gain = 0 + 0.33 − 0.14

↳ = 0.19

Error = [0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]



* $S.W = \frac{\varepsilon(res)^2}{\varepsilon p(1-p)}$

$= \frac{[0.5]^2}{0.5[1-0.5]} = \frac{0.25}{0.25}$

= 1

* $S.W = \frac{\varepsilon(res)^2}{\varepsilon p(1-p)}$

$= \frac{[0.5, 0.5, -0.5]^2}{0.5[1-0.5] + 0.5[1-0.5] + 0.5[1-0.5]}$

$= \frac{0.25}{0.75} = \frac{1}{3} = 0.33$

* Information Gain : 1 + 0.33 − 0 → ?

= 1.33

For classification

Salary	Credit	Approval
<=50	→ B	0

Goes to binary

activation function → sigmoid

$\sigma\left[0 + \alpha(1)\right]^{\alpha=0.01}$ ⇒ Based on "α" value → o/p 0-1

Learning rate, S.W

∴ Base model = 0.5

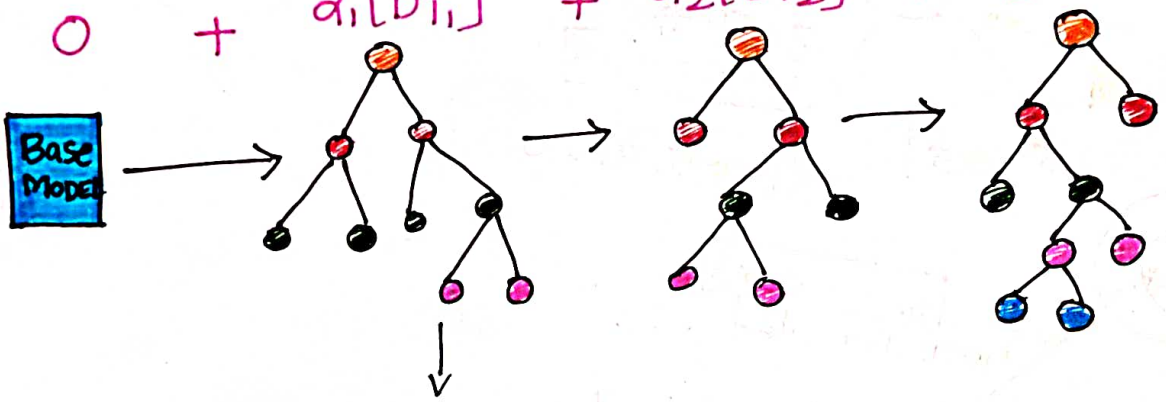
Real probability = $\log\left(\frac{p}{1-p}\right) = \log\left(\frac{0.5}{1-0.5}\right) = 0$

Output :-

$$\left[\sigma\left[0+\alpha_1[DT_1]+\alpha_2[DT_2]+\alpha_3[DT_3]+\cdots\alpha_n[DT_n]\right]\right.$$

For any new record

$$0 + \alpha_1[DT_1] + \alpha_2[DT_2] + \alpha_3[DT_3] + \cdots\cdots$$



Based on independent Features.

$\therefore \lambda$ = Cross validation.

* **XG boost regressor**

* Base model = 51K

Ex:-

X_1 Exp	X_2 Gap	y Salary	Residual
2	Yes	40 K	-11K
2.5	Yes	42 K	9
3	NO	52 K	1
4	No	60 K	9
4.5	Yes	62 K	11

Avg = 51.

S.W = $\frac{1}{6}$

[-11, -9, 1, 9, 11]

$$\frac{[-11 - 9 + 1 + 9 + 11]^2}{5+1} = \frac{1}{6}$$

Experience

S.W = 45.5 (< 2) → [-11]

S.W = 28.8 (> 2) → [-9, 1, 9, 11]

$$\frac{[-9 + 1 + 9 + 11]^2}{4+1} \quad \lambda = 1$$

$$= \frac{[12]^2}{5} = \frac{144}{5} = 28.8$$

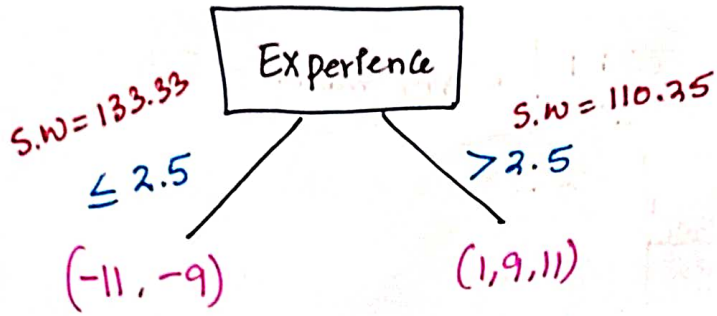
* Similarity weight = $\frac{\Sigma [\text{Residual}]^2}{\text{no. of residuals} + \lambda}$

$$= \frac{[-11]^2}{1+1} = \frac{121}{2} = 60.5$$

$\therefore \lambda = 1$

* Information Gain = $60.5 + 28.8 - \frac{1}{6} = 89.13$

S.W 1/6



$$S.W = \frac{[-11-9]^2}{2+1} = \frac{[-20]^2}{3} = \frac{400}{3} \Rightarrow 133.33$$

$$= \frac{[1+9+11]^2}{3+1} = \frac{[21]^2}{4} = 110.25$$

any records goes from base model = 51

≤ 2.5 Avg. of S.W

$51 + \alpha_1 \left[\frac{[-11-9]}{2}\right] = (-10)$

* $51 + \alpha_1 [-10]$

Avg. of S.W

$51 + \alpha_1 \left[\frac{1+11+9}{3}\right] \Rightarrow \frac{21}{3} = 7$

* $51 + \alpha_1 [7]$

* $51 + \alpha_1 (-10) + \alpha_2 [DT_2] + \alpha_3 [DT_3] - \cdots \alpha_n [DT_n]$

↳ Output.

27/04/22
6:00 Am.

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.


→ Mushroom

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
Cap_colour :	brown = n, buff = b, cinnamon = c, gray = g, green = r, pink = p, purple = u, red = e, white = w, Yellow = y.
bruises ? :	bruises = t, no = F
dor :	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, scaly = 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 is to 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 below 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	"	"

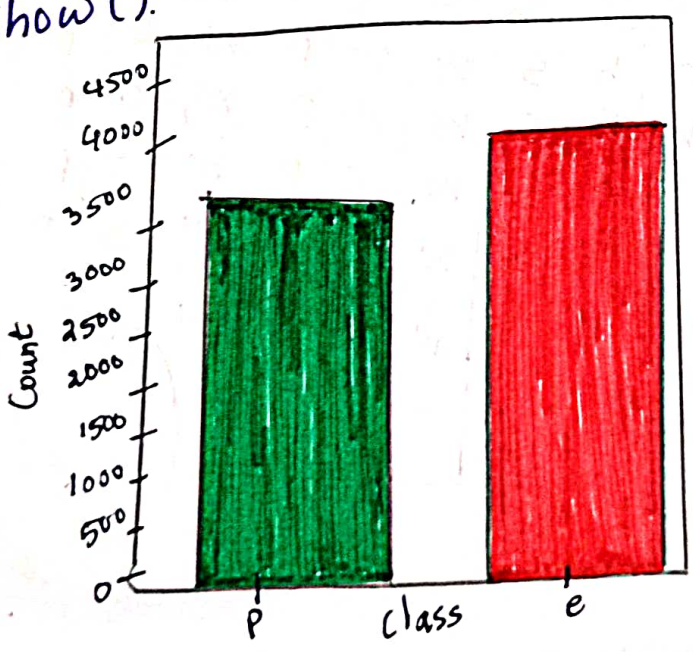
df.isnull().sum()

Out: 0.

EDA

sns.countplot(data=df, X="class", palette="Dark2")
output

plt.show().



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

```
# df.describe().transpose()
```

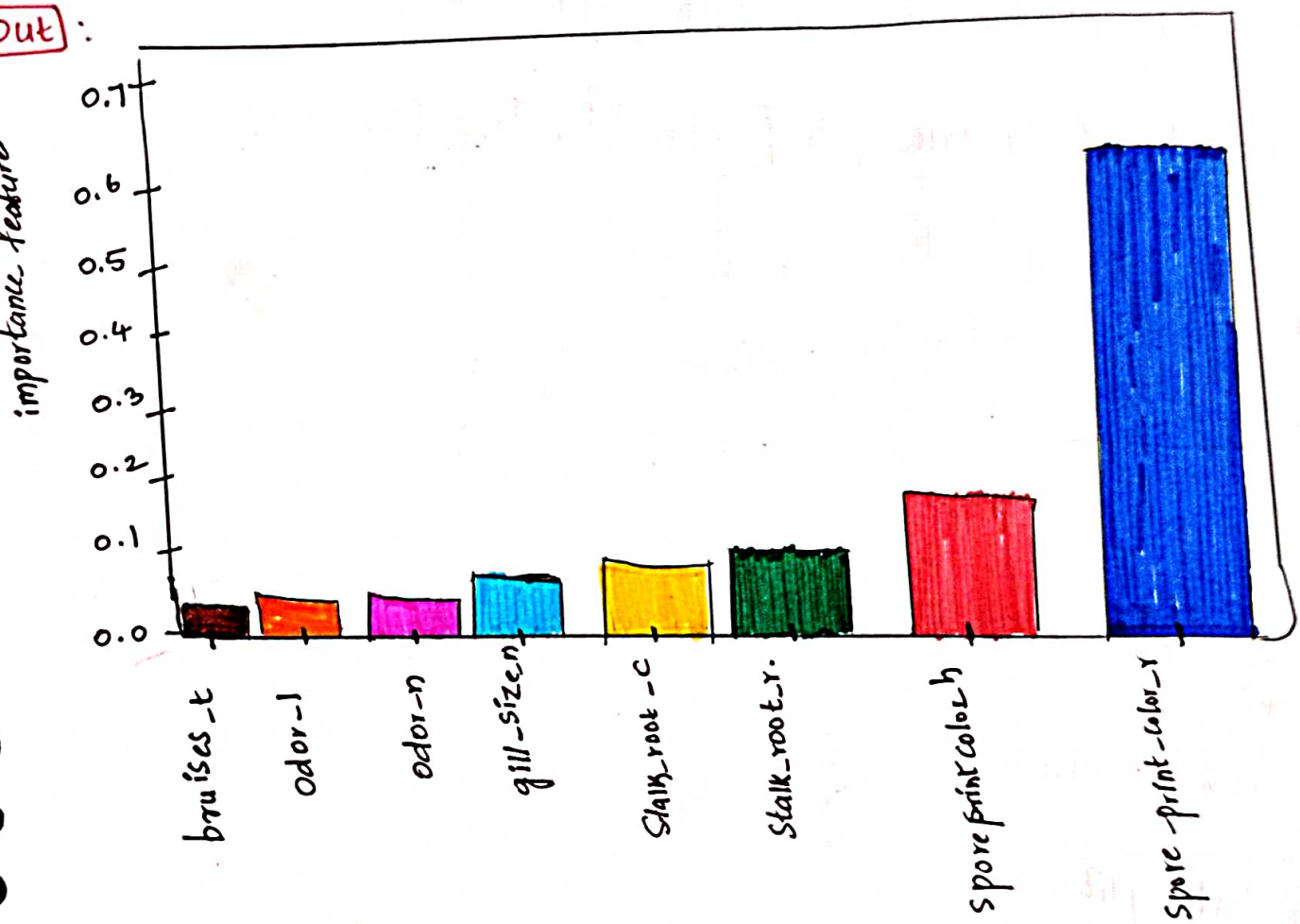
Out

	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 bebw 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

```
# 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:



Code :

XG boost Classifier

Same Example, Same Question of Mushroom dataset.

import to X & Y.

X & Y

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

```
# y
```

out :

	p
0	1
1	0
2	0
3	1
4	0
⋮	⋮

8124 x 1 column

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 = 0.2, random_state=29
```

modelling

```
! Pip install xgboost

from xgboost import XGBClassifier

# xgb_model = XGBClassifier()

# xgb_model.fit(x_train, y_train)
```

Out: XGBClassifier(base_score=0.5, booster='gbtree', ...
.. .. --)

Prediction

```
# ypred_train = xgb_model.predict(x_train)

# ypred_test = xgb_model.predict(x_test)
```

Evaluation

* Accuracy

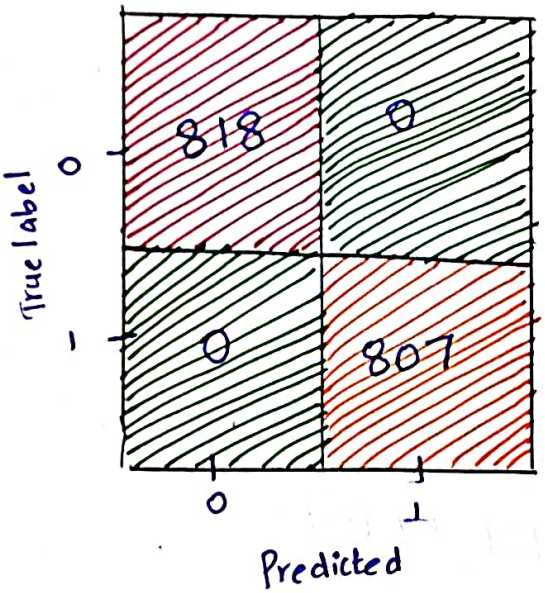
```
from sklearn.metrics import accuracy_score

# print("train accuracy:", accuracy_score(y_train, ypred_train

# print("test accuracy:", accuracy_score(y_test, ypred_tes
```

Out:
Train accuracy : 1.0
Test accuracy : 1.0

* Confusion Matrix

```
from sklearn.metrics import plot_confusion_matrix
# plot_confusion_matrix (xgb_model , 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	f1 score	Support
0	1.00	1.00	1.00	818
1	1.00	1.00	1.00	807
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 (xgb_model, x, y, cv=5)
# print ("Cross_val_score:", scores.mean())
```

Out: Cross-val-score : 0.935

* Feature importance

```
# xgb_model.feature_importances_
```

Out: array([0.0000e+0 , 0.0000e+00 , 0.0000e+0, 0.0000e+0
3.961512e5 0.00000e+0 , 9.23.225 ---])

Hyper Parameter tuning

Important Paramen in XGboos

```
from sklearn.model_selection import GridSearchCV

# estimator = XGBClassifier()
# param_grid = {"n_estimators": [1,5,10,20,40,100], "gamma": [0,1,
                ,0.5,0.9], "max_depth": [2,6
# grid = GridSearchCV(estimator, param_grid, scoring = "accuracy
                      cv= 5)
# grid.fit(x_train, y_train)
# grid.best_params
```

Out: {"gamma":0.1, "max_depth":2, "n_estimators":20}

final Model

```
# final_model = XGBClassifier (n_estimators = 20, gamma = 0.1, max_
                                depth = 2)
# final_model . fit (x_train, y_train)

# ypredictions_test  = final_model. Predict (x_test)
# ypreditions_train  = final_model. Predict (x_train)

# print ("train accuracy:" ; accuracy_score (yprediction_train, y_train))
# print ("test accuracy:", accuracy_score (yprediction_test, y_test))
```

Out: Test accuracy : 0.99815
Train accuracy : 0.9979

```
# final_model. feature_importances_
```

Out: array ([0., 0., 0., 0.,
0., 0., 0. ..])

```
# important = pd. DataFrame ( index = X. columns , data = final_
model . feature_importances_ , columns = ['importance_
Features'])
```

```
# Jacks = important[important["importance_feature"] > 0.01]

# Jacks.sort_values("importance_feature")
```

Out:

	important_features
odor-p	0.015831
spore-print-color-w	0.020152
⋮	⋮
odor-n	0.290767

```
# plt.figure(fig_size=(14,6), dpi=200)
# sns.barplot(data = Jacks.sort_values("importance_feature"),
                x = Jacks.index, y= "importance_feature")

# plt.xticks(rotation=90)
# plt.show()
```

Out: