**** 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** :-

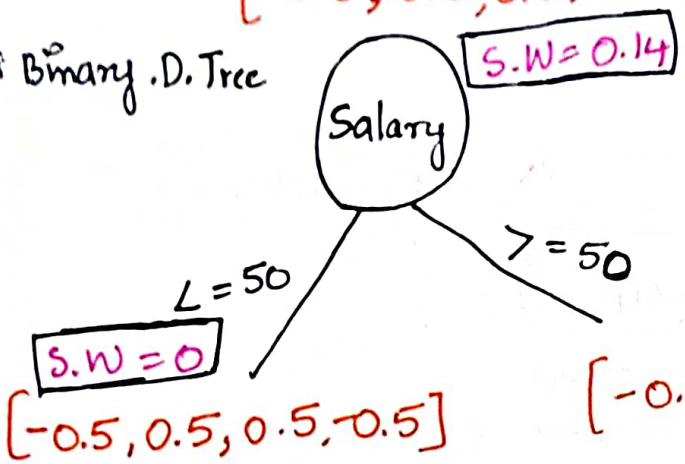
1. Create Binary Decision Tree using Features
2. Calculate Similarity weight

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

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

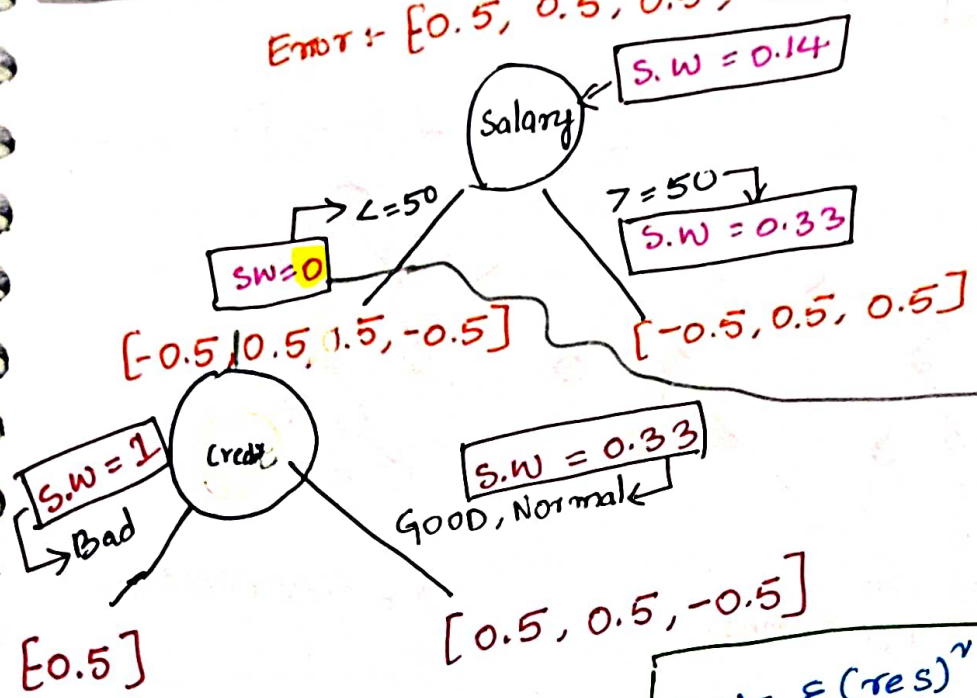
$$= \frac{[-0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]^2}{0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]+0.5[1-0.5]}$$

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

* 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

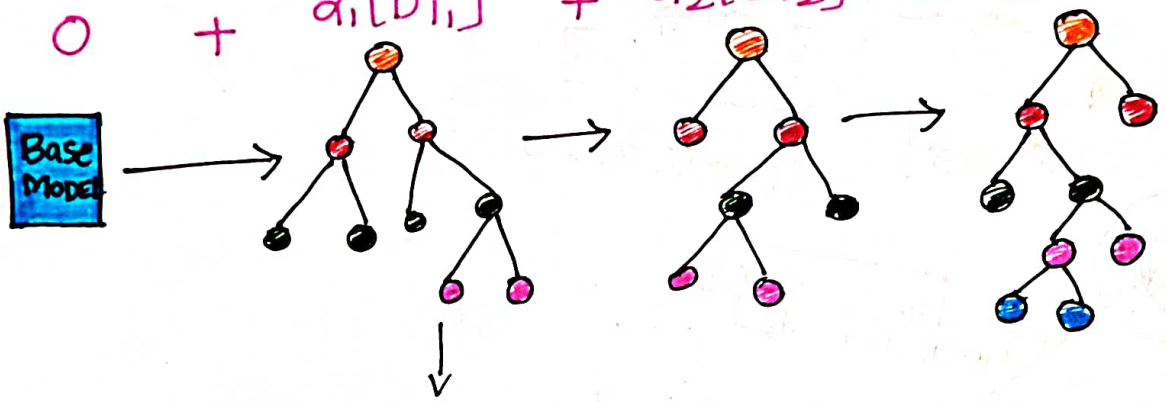
Real probability = $\log\left(\frac{p}{1-p}\right) = \log\left(\frac{0.5}{1-0.5}\right) = 0$

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]

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

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

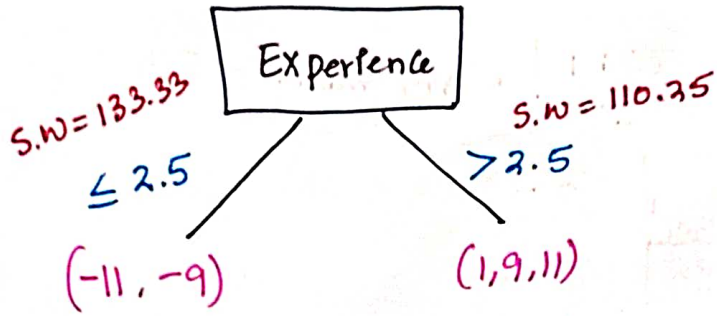
* Similarity weight = $\frac{\Sigma [\text{Residual}]^2}{\text{no. of residuals} + \lambda}$

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

$\therefore \lambda = 1$

* 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	"	"

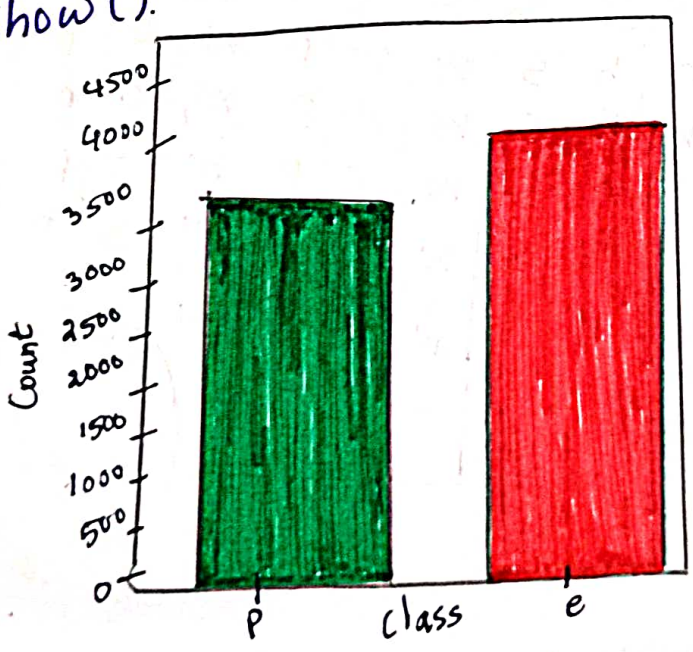
df.isnull().sum()

Out: 0.

EDA

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

plt.show().



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

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

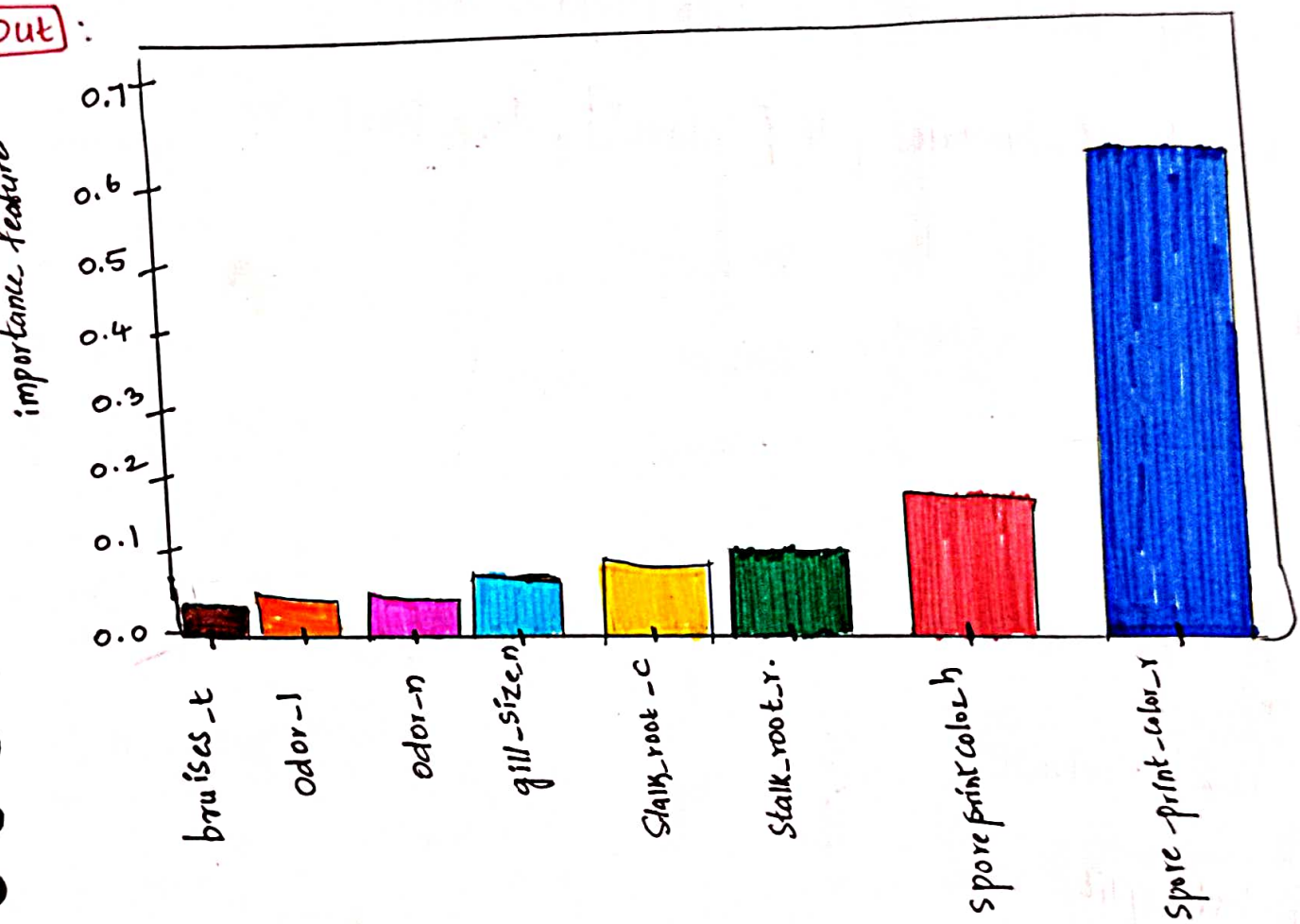
Out

	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

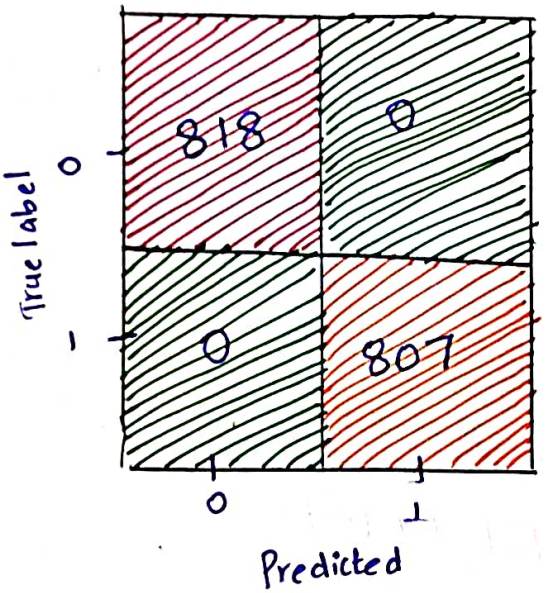
```
from sklearn.metrics import accuracy_score

# 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: