

## Deep Learning :-

Used for

1. Artificial neural network [ ANN ] [ Regression/classification ]
  2. Convolution neural network [ CNN ] [ computer vision ]  
[ image classification ]
  3. Recurrent neural network [ RNN ] [ Time series analysis ]
- and
- long short term memory

[ LSTM ]

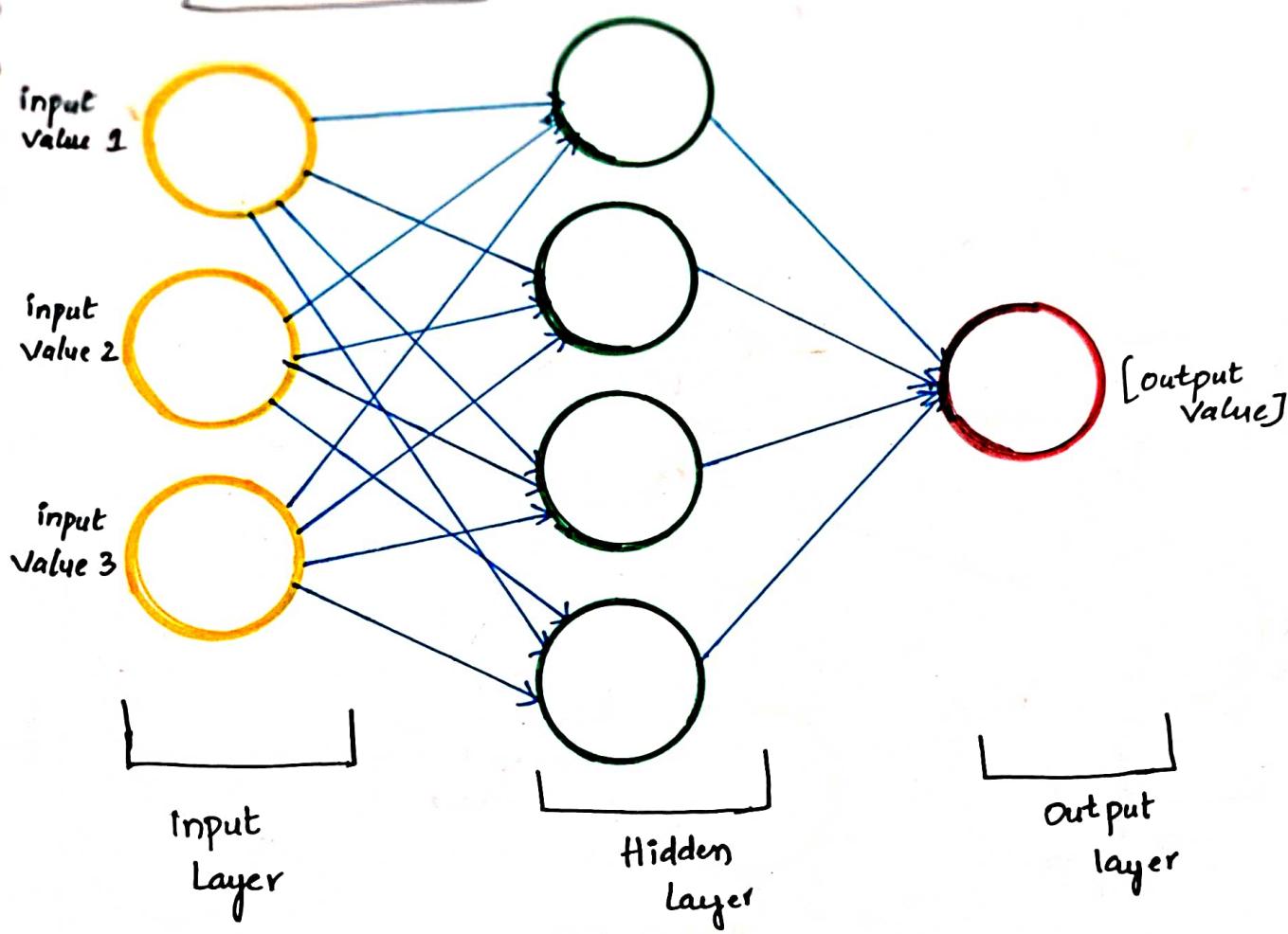
### 1. Artificial neural network

\* Artificial intelligence :- application which can do its own task with out any human intervention

Ex:- netflix app

- \* self driving cars
- \* amazon application
- \* Sofia (robot)

\* Neural network :- Connecting the neurons making as One Connection is known as "Neural network".



Ex:-

$x_1$	$x_2$	$x_3$	$y$
-	-	-	-
-	-	-	-
-	-	-	-

Here, we have  
\* Three input layers :

$x_1, x_2, x_3$

\* One output layer :

" $y$ "

" $y$ " can be

continuous  
discrete  
binary  
multiclass

1. **Perceptron** :- Connecting input layer to output layers

1. Fundamental concepts of data preprocessing  
directly without any hidden layers.  
2. Linear Regression

3. Deep Learning

4. Convolutional Neural Network

5. Support Vector Machines

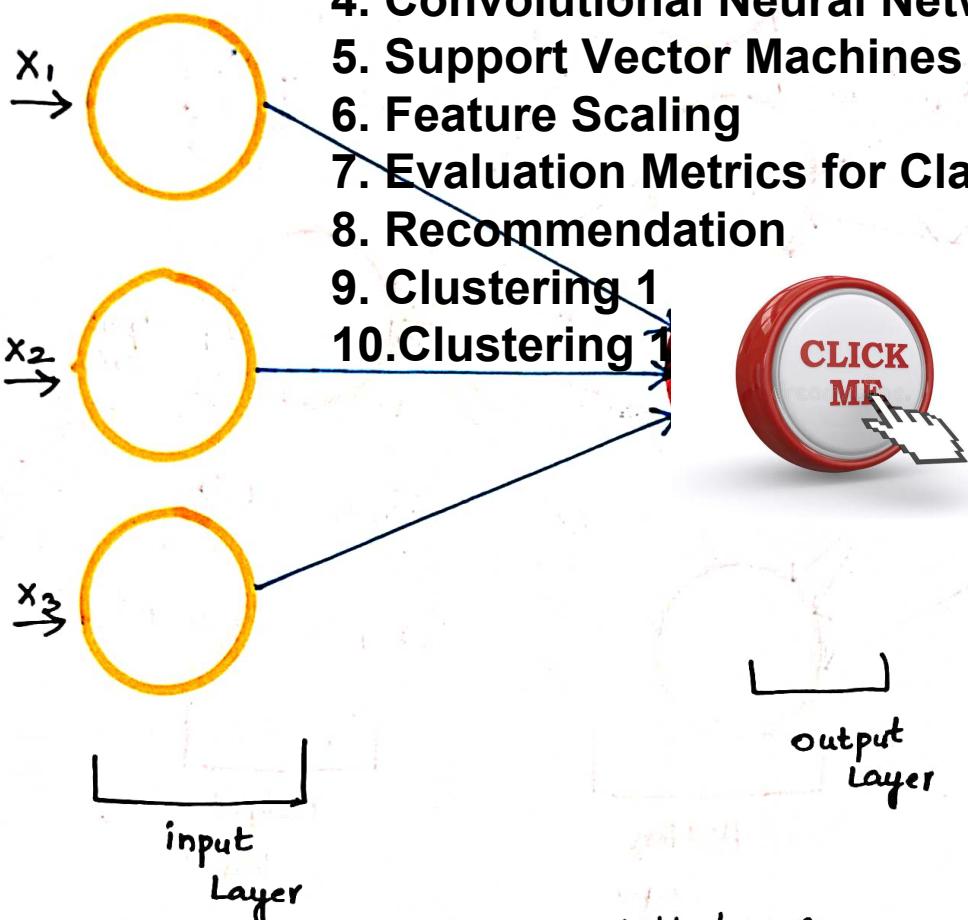
6. Feature Scaling

7. Evaluation Metrics for Classification

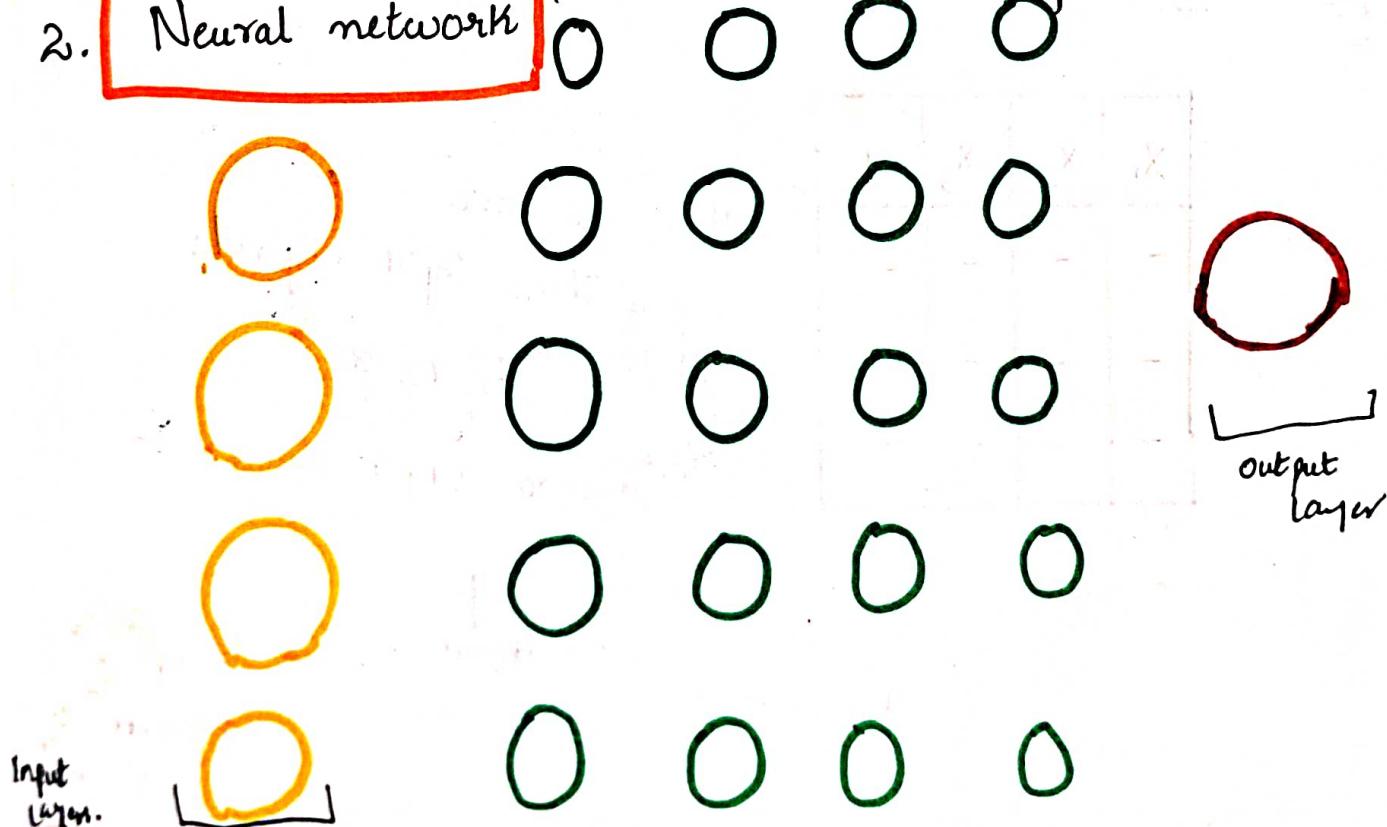
8. Recommendation

9. Clustering 1

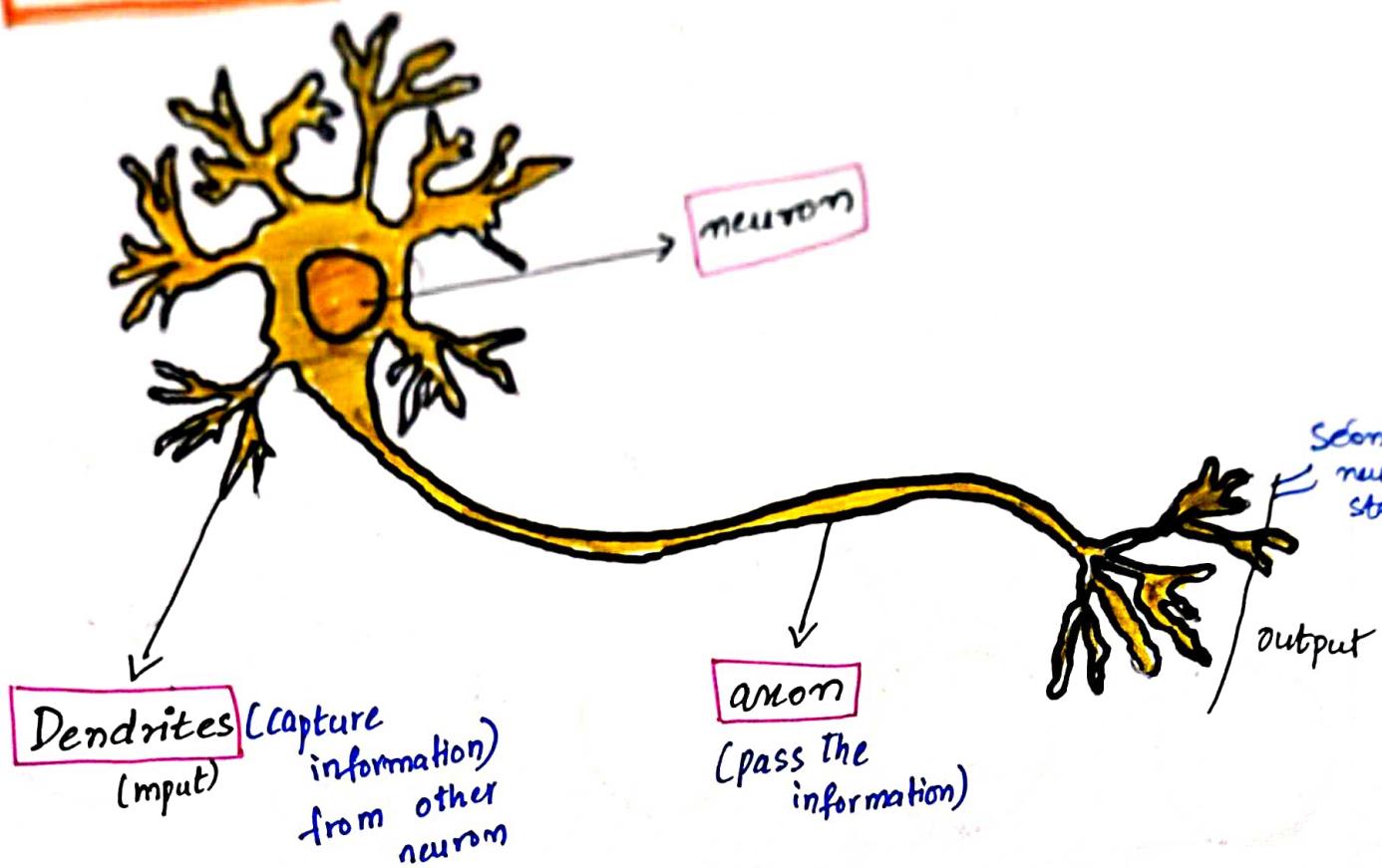
10. Clustering 1



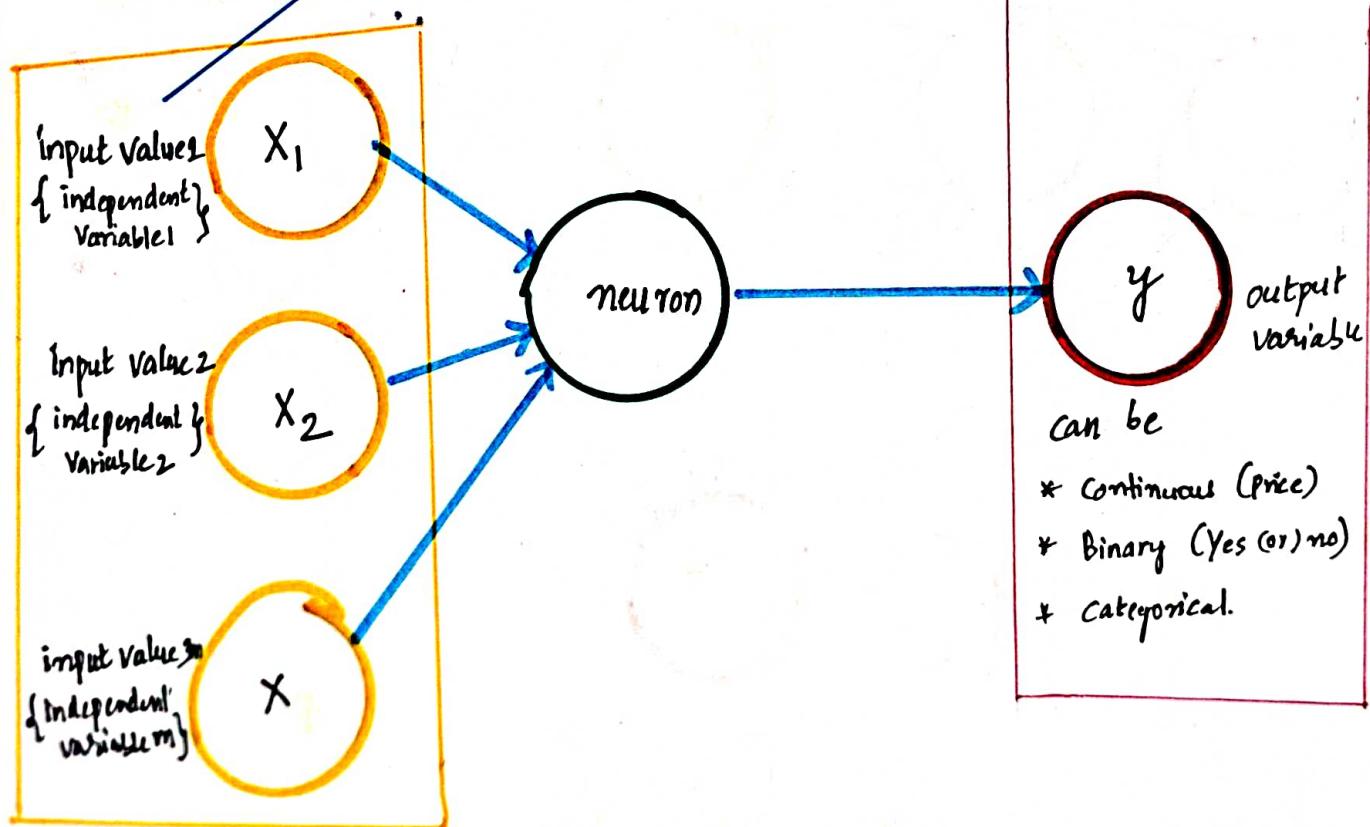
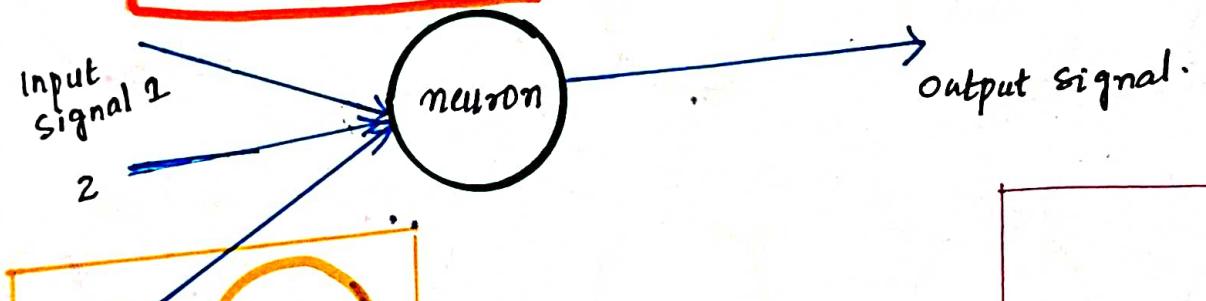
2. **Neural network**



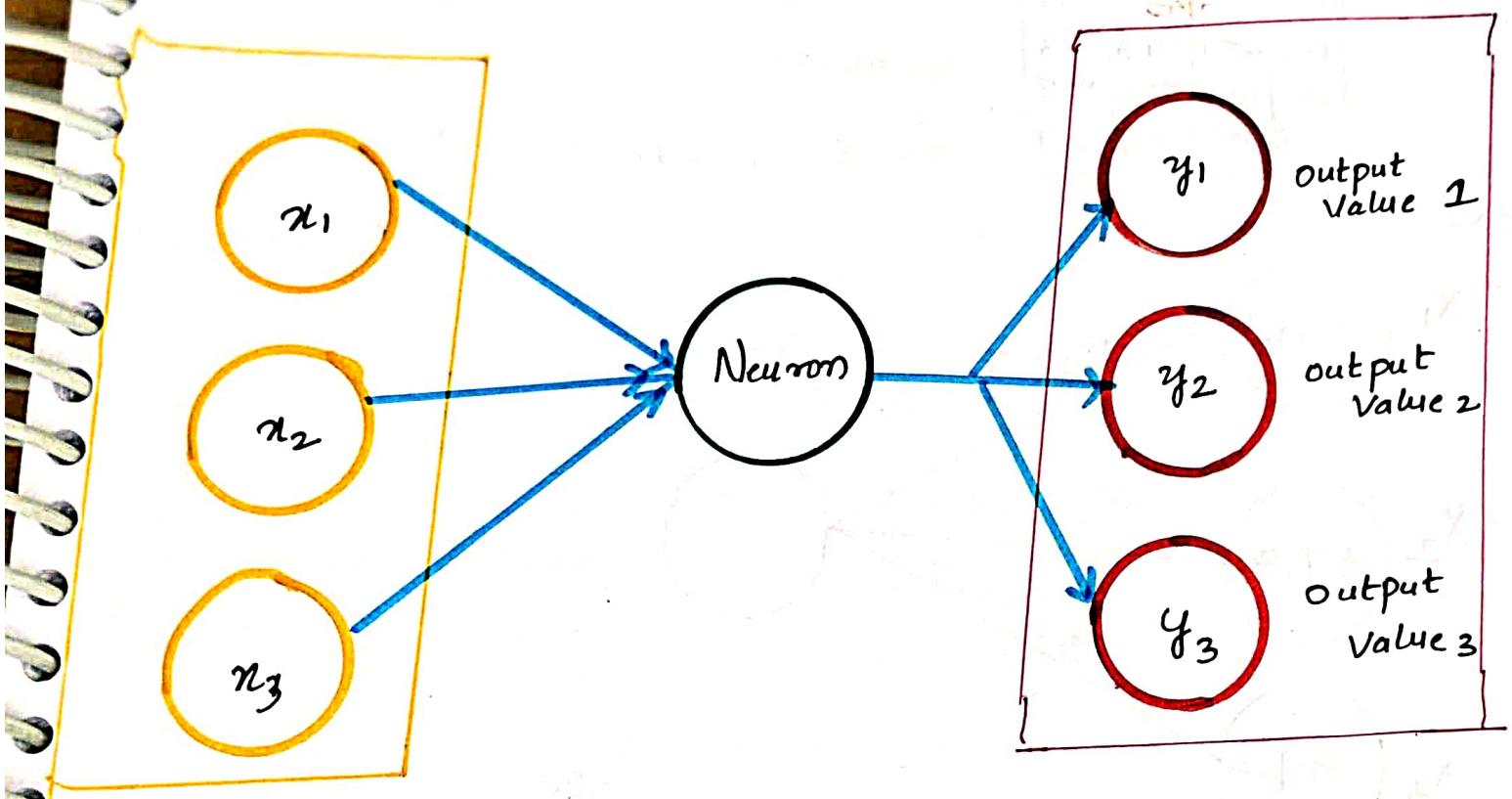
\* The neuron connected to each other to pass the information



\* Artificial Neuron



- \* Binary classification  $\rightarrow$  o/p neuron  $\Rightarrow$  1 neuron
- \* Regression  $\rightarrow$  output neuron  $\Rightarrow$  1 neuron
- \* Multiclassification  $\rightarrow$  output neuron  $\Rightarrow$  no. of categories (6) classes.  
Ex:- penguin  
Age, chi, Gentoo



### \* Weights

Machine learning:  $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 = (y - \hat{y})_{\min}$

deep learning:  $b + w_1 x_1 + w_2 x_2 + w_3 x_3 = (y - \hat{y})_{\min}$

It can be written as  $\sum w_i x_i + b$

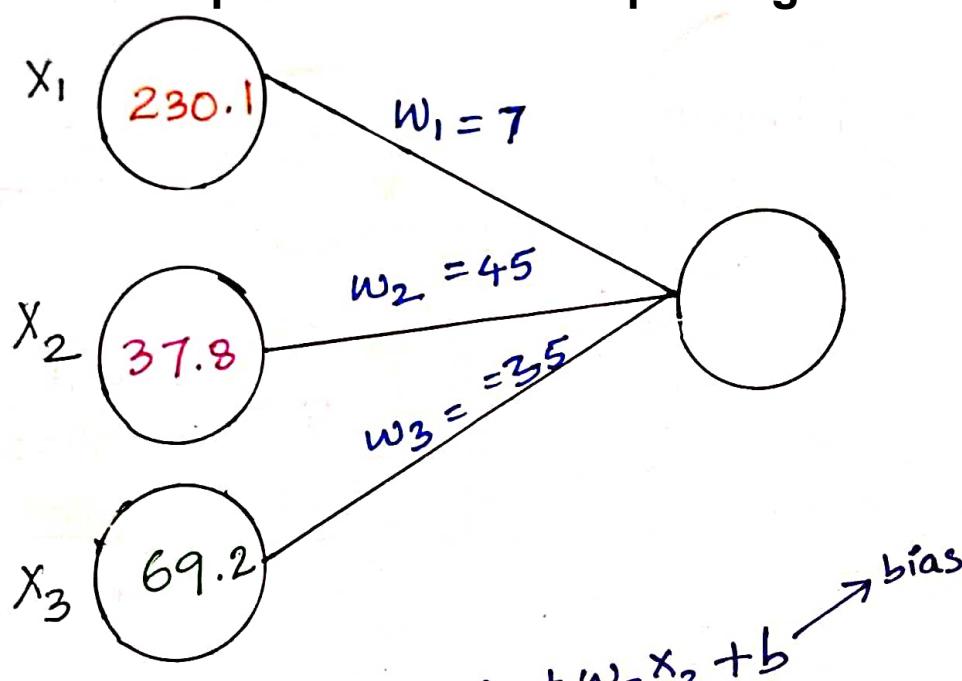
Ex:- Linear Regression.

$X_1$	$X_2$	$X_3$	$y$
TV	Radio	Newspaper	Sales
230.1	37.8	69.2	221
46.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

we consider weights  
Initialization of value  
 $x = 100 \rightarrow 2(100) + 5$

Ex:- Equations ( $2x + 5 = y$ )  
 $y$   $y$  error  
 $x = 99 \rightarrow 2(99) + 5$  203 9 -196  
 $x = 98 \rightarrow$  . . .  
 $x = 2 \rightarrow 2(2) + 5$  9 9 0

Ex:- <https://t.me/AIMLDeepThaught/612>



$$* w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

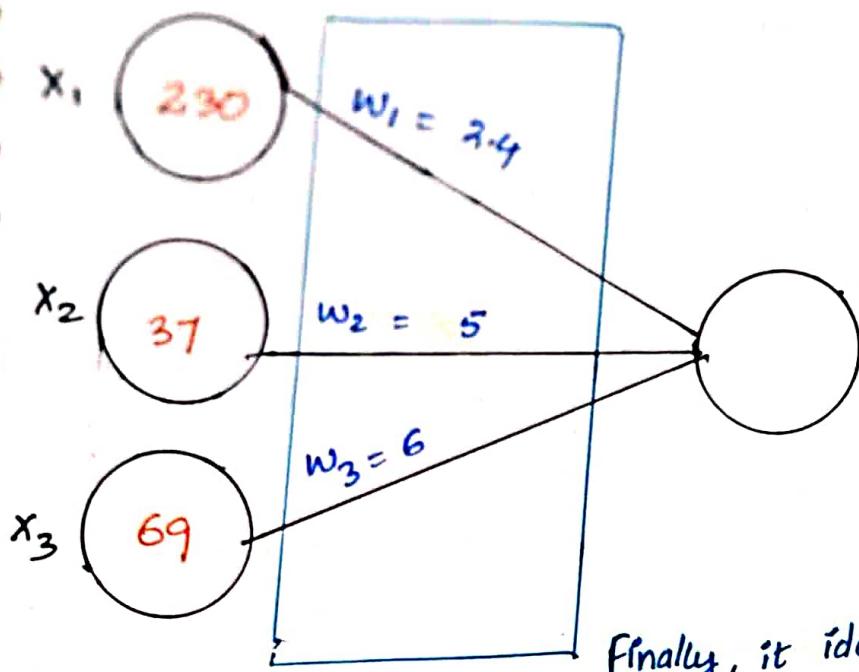
$$* [7 \times 230] + [45 \times 37] + [35 \times 69] + 100$$

$$\hat{y} = 5790.$$

$y$	$\hat{y}$	$y - \hat{y}$
221	5790	5767.9

So,  $(y - \hat{y})$  largest amount  
of difference is there,  
- weights will be readjusted

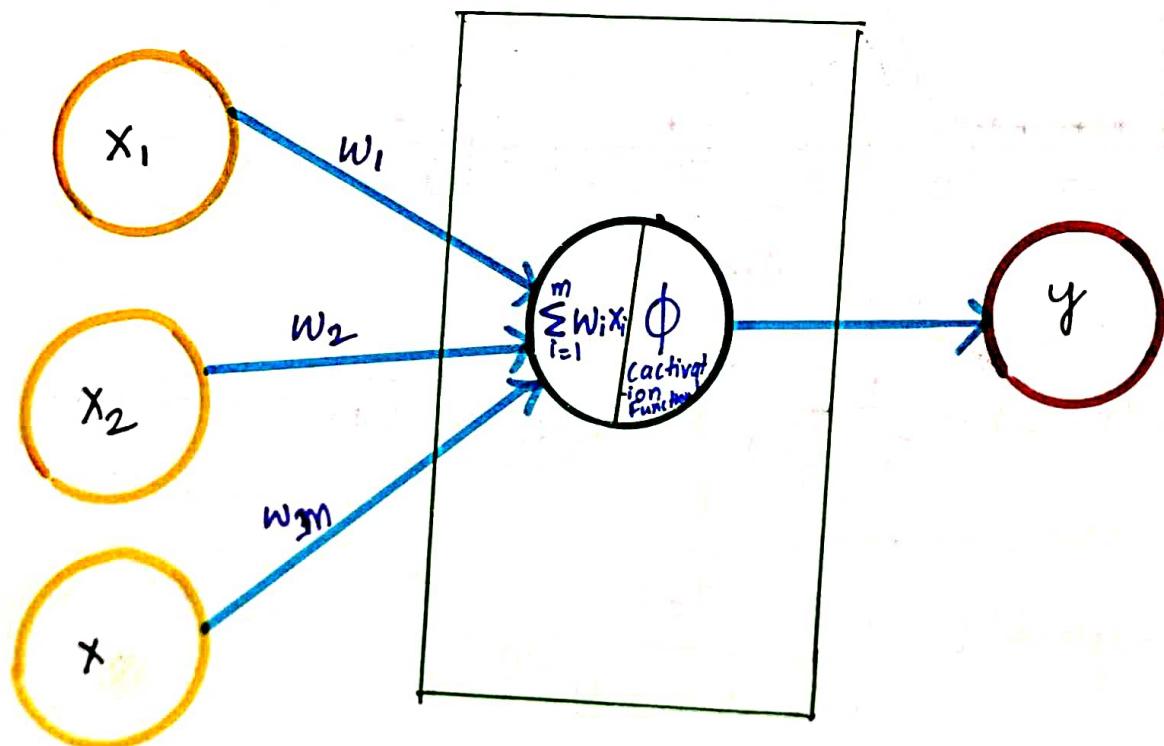
\* weights will be adjusted



Finally, it identifies the best weights which gives sum of square errors to minimum.

Here, it's identify only weights.  
 $(SSE)_{min}$ .

#### \* Steps in neuron



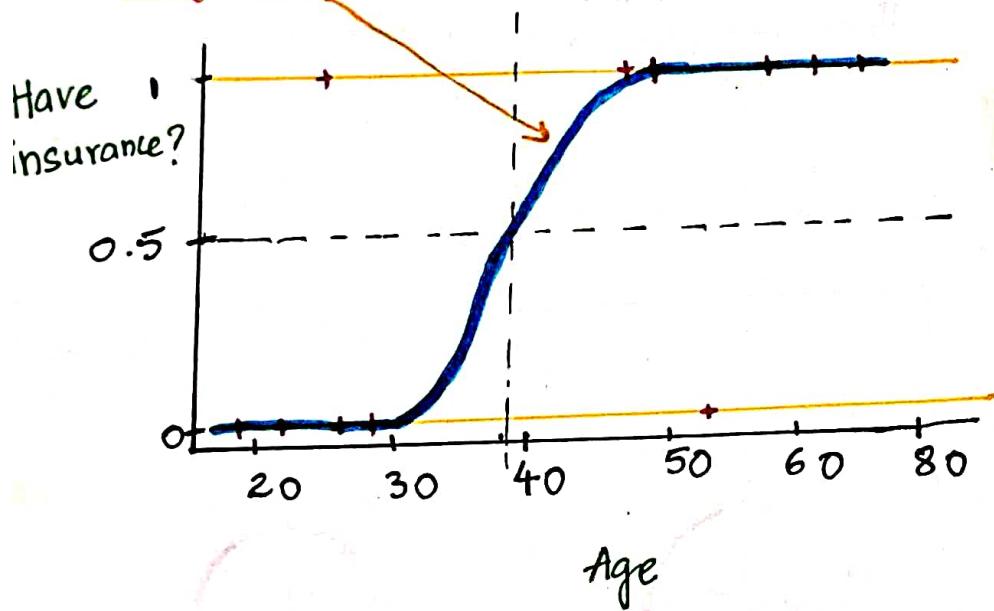
Ex :-

## Binary classification

age	have-insurance
22	0
25	0
47	1
52	0
46	1
56	1

Given, an age of a person, come up with a function that can predict if person will buy insurance (or) not.

\* **Sigmoid** or **Logit function**



$$\text{Sigmoid}(z) = \frac{1}{1 + e^{-y}}$$

$\therefore e = \text{Euler's number} \sim 2.71828$

$$\text{Sigmoid}(200) = \frac{1}{1 + 2.71^{-200}} = \text{almost close to } 1$$

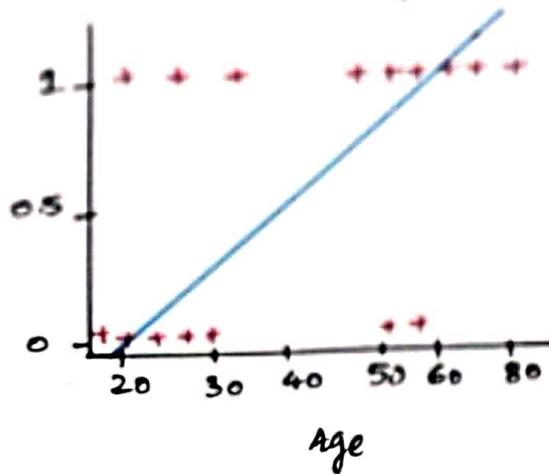
$$\text{Sigmoid}(-200) = \frac{1}{1 + 2.71^{+200}} = \text{almost close to } 0$$

\* Sigmoid function converts input into range 0 to 1.

STEP : 2

$$y = m \cdot x + b$$

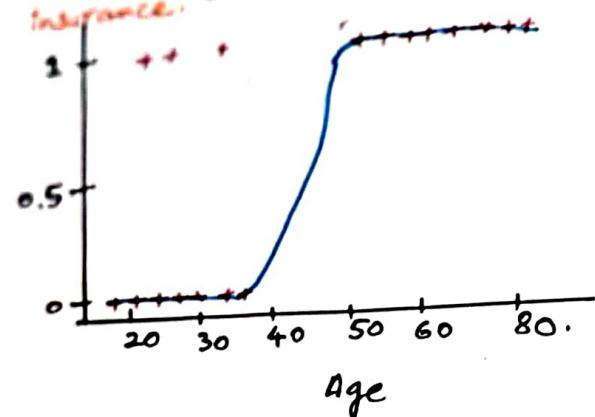
$\hookrightarrow$  age



Step : 2

$$z = \frac{1}{1 + e^{-y}}$$

If  $\downarrow$   
Person will buy  
insurance.



$$y = 0.042 \cdot x - 1.53$$

$\hookrightarrow$  Age.

• STEPS :-

$$y = 0.042 \cdot x - 1.53$$

$$z = \frac{1}{1 + e^{-y}}$$

- value  $< 0.5$  = Person will  $\nwarrow$  buy
- value  $> 0.5$  = Person will buy insur.

Age = 35

$$y = 0.042 \cdot 35 - 1.53$$

$$= -0.06$$

$$z = \frac{1}{1 + 2.71^{0.06}}$$

$$= 0.48$$

$\rightarrow 0.48$

Age = 43

$$y = 0.042 \cdot 43 - 1.53$$

$$= 0.216$$

$$z = \frac{1}{1 + 2.71^{0.216}}$$

$$= 0.568$$

$\rightarrow 0.51$

$$y = 0.042 * x_1 - 1.53$$

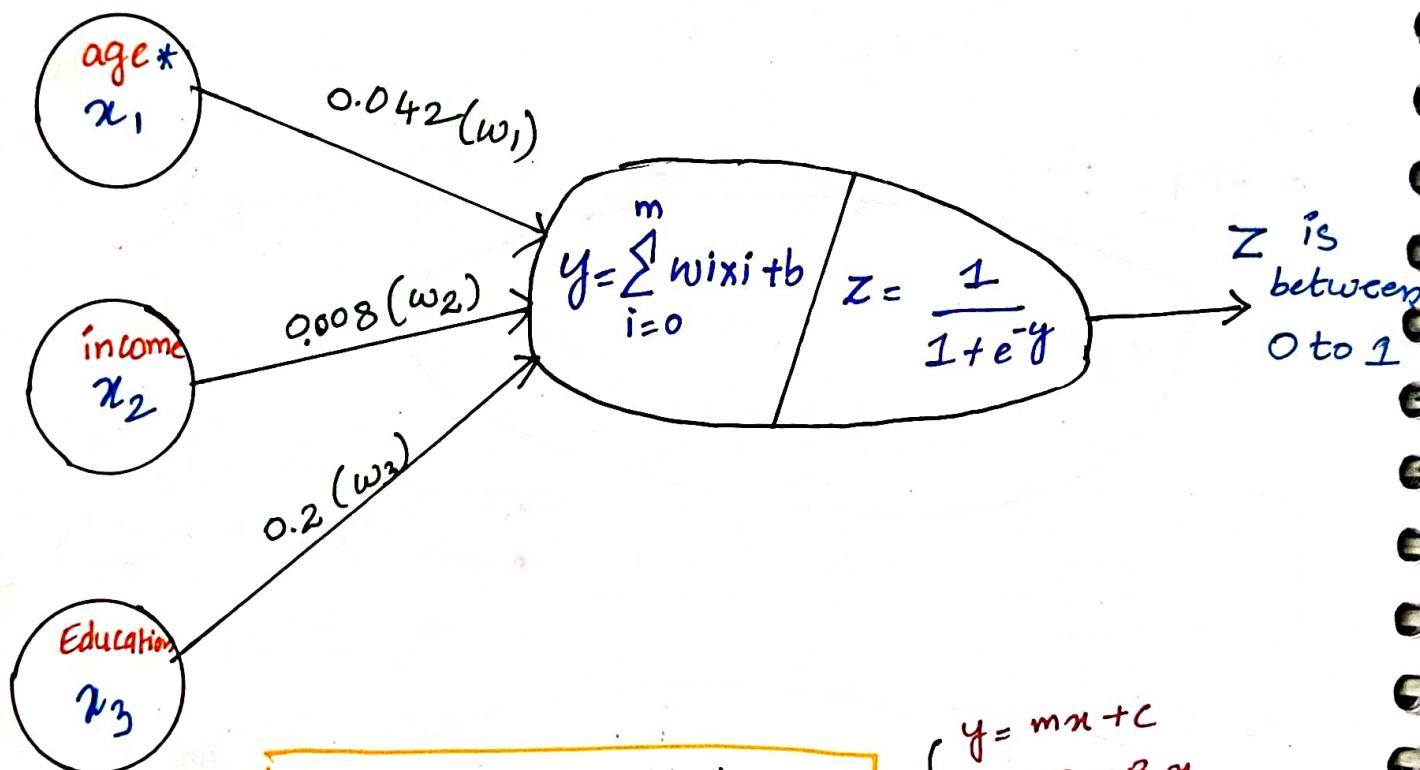
$\hookrightarrow$  age  $\nearrow b$

$$y = [0.042 * x_1] + [0.008 * x_2] + [0.2 * x_3] - 1.53$$

$\hookrightarrow$  Age  $\hookrightarrow$  income  $\nearrow b$   $\hookrightarrow$  Education.

$$y = [w_1 * x_1] + [w_2 * x_2] + [w_3 * x_3] + b$$

$$y = \sum_{i=0}^m w_i x_i + b$$



\* "weights" and "b" values

can be  
+ve, -ve

$$\begin{cases} y = mx + c \\ y = \beta_0 + \beta_1 x \\ y = w_i x_i + b \end{cases}$$

all are  
same (but change in  
notation)

Ex:-

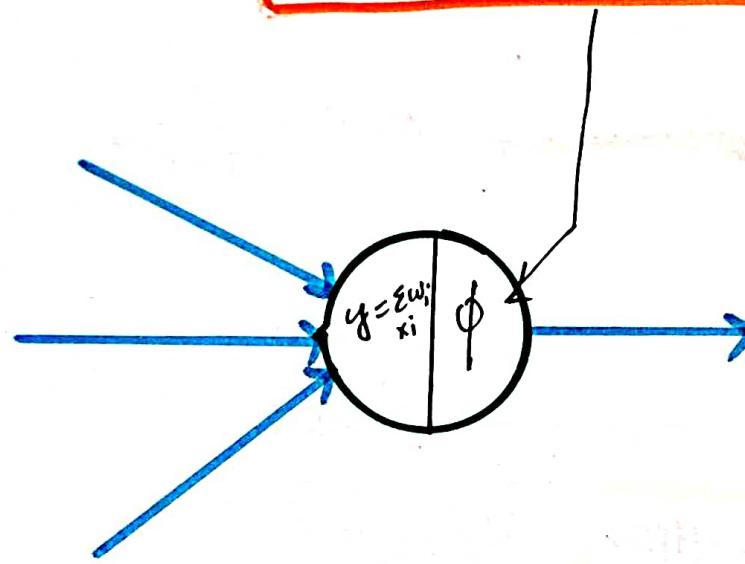
X	y
1	7
2	17
3	27
4	37
5	47

$$\therefore y = 10x - 3$$

$$= w_i x_i + b$$
$$= 10[x] + [-3]$$

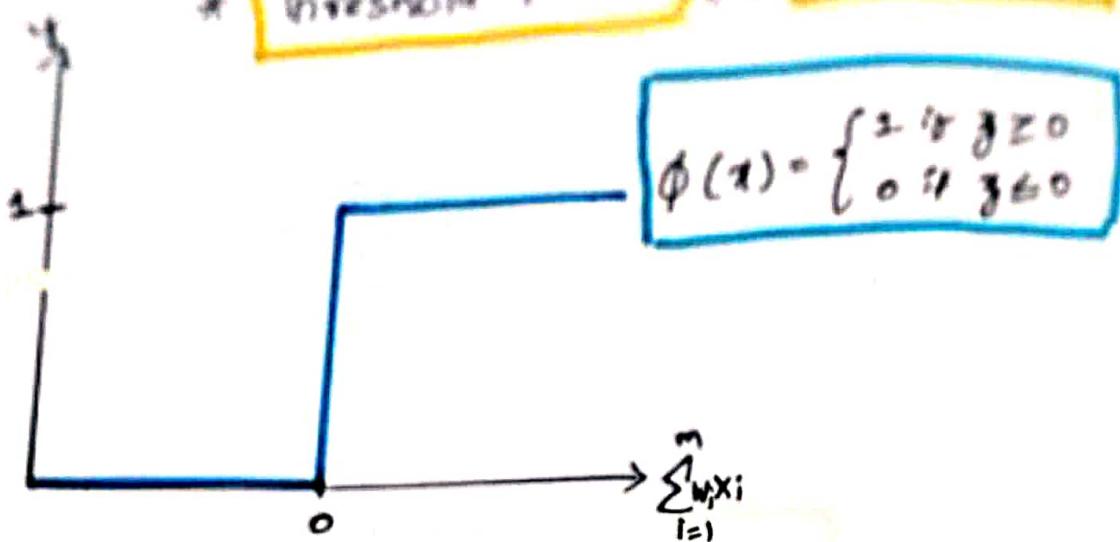
w, b  
can be (+ve) or (-ve)

### Activation function

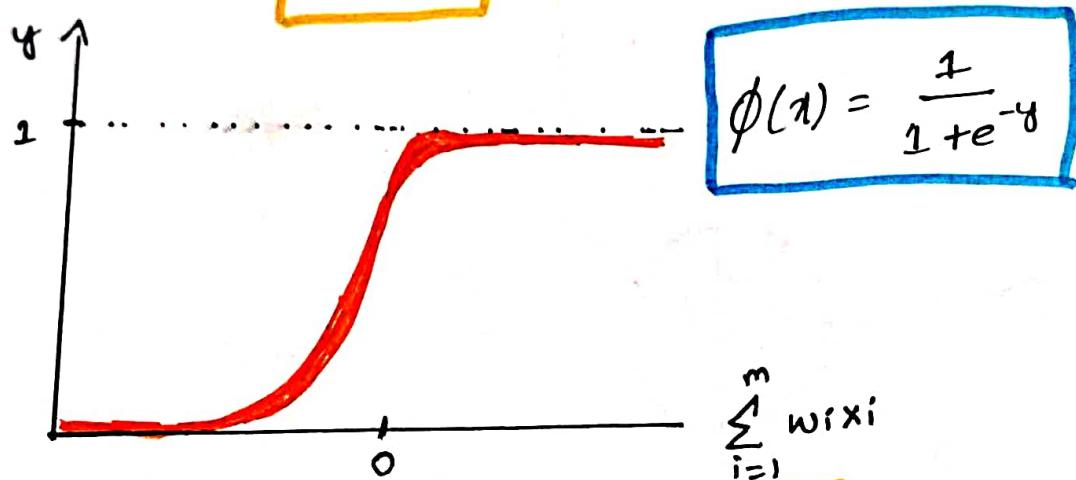


1. Threshold Function (or) step function
2. Sigmoid Function
3. Rectifier (or) ReLU
4. Hyperbolic Tangent (tanh)

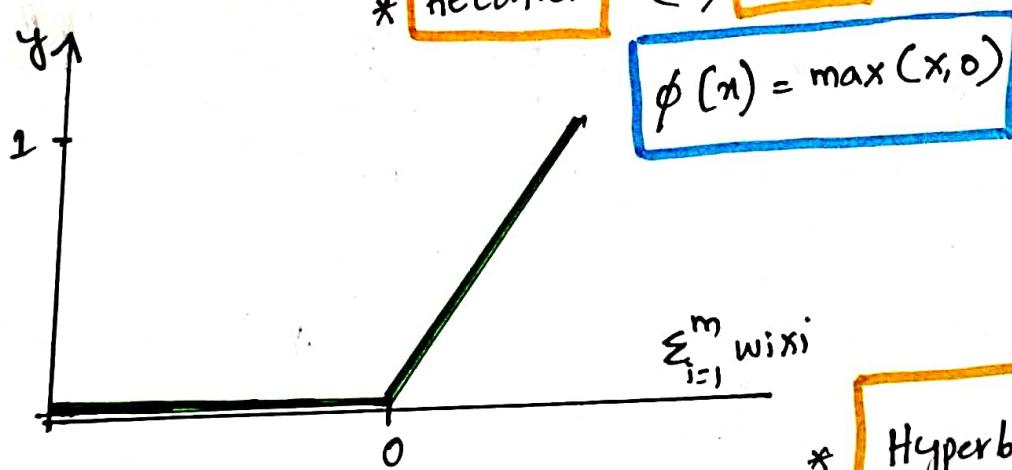
\* Threshold function (or) Step function



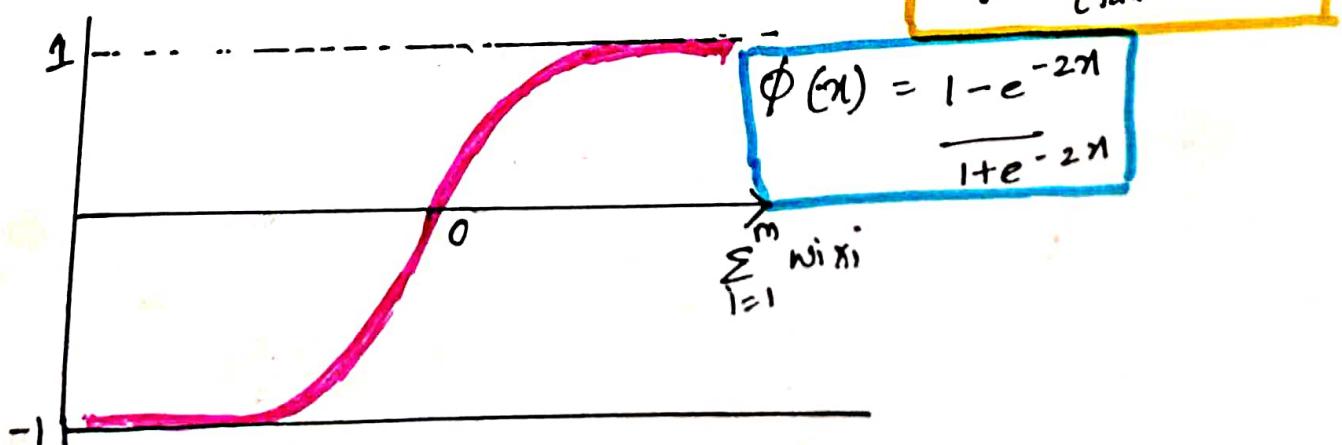
\* Sigmoid



\* Rectifier (or) Relu



\* Hyperbolic Tangent (tanh)



\* Threshold function (or) Step function

$$\phi(x) = \begin{cases} 1 & \text{if } y \geq 0 \\ 0 & \text{if } y \leq 0 \end{cases}$$

\* Sigmoid function

$$\phi(x) = \frac{1}{1+e^{-x}}$$

$$\Rightarrow \frac{e^x}{1+e^x}$$

<https://t.me/AIMLDeepThaught/612>

\* Rectifier (or) Relu

• Relu = Rectifier Linear Unit.

$$\phi(x) = \max(x, 0)$$

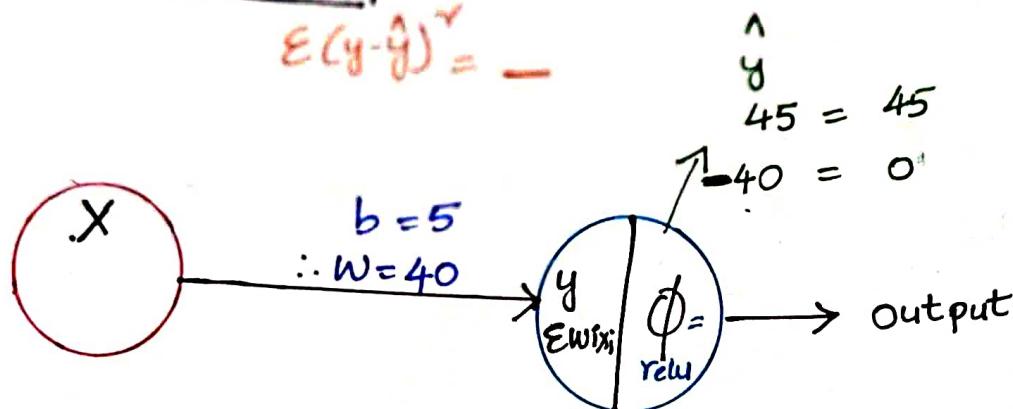
\* Tanh (or) hyperbolic Tangent

$$\phi(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$$

Ex:- Regression  $\rightarrow$  Relu

X	y	$\hat{y}$	$y - \hat{y}$
1	11	45	-
2	12	85	-
3	13	125	-
4	14	165	-
5	15	205	-

$$E(y - \hat{y})^2 = -$$

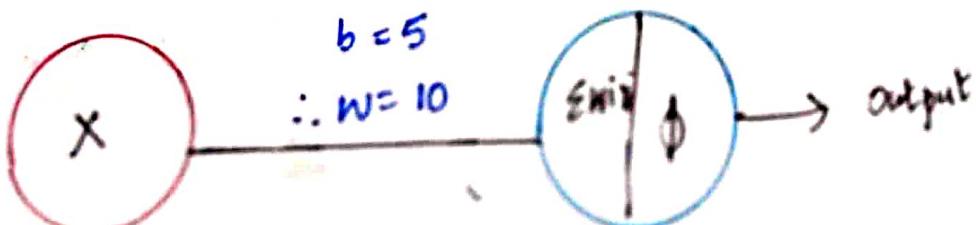
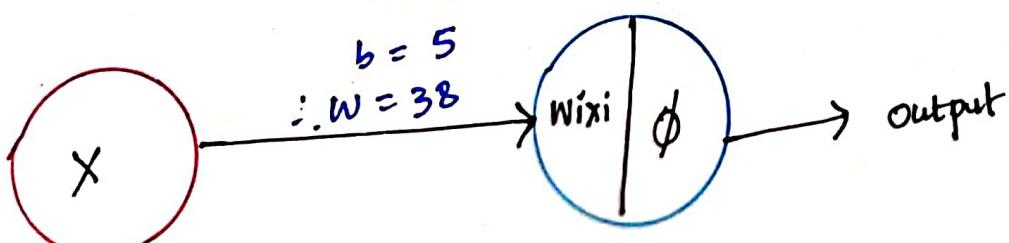
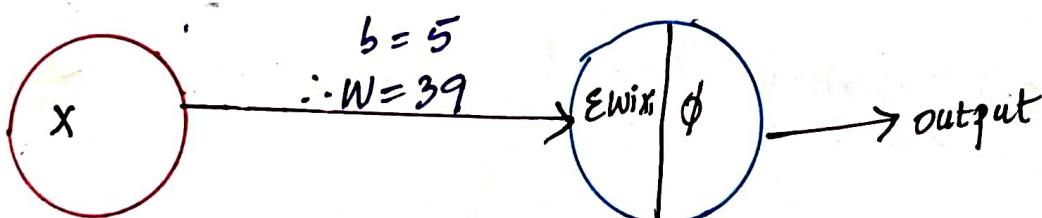


$$\therefore \hat{y} = wX + b$$

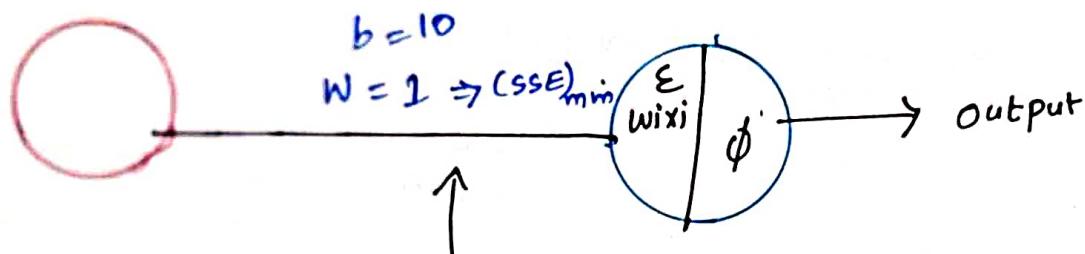
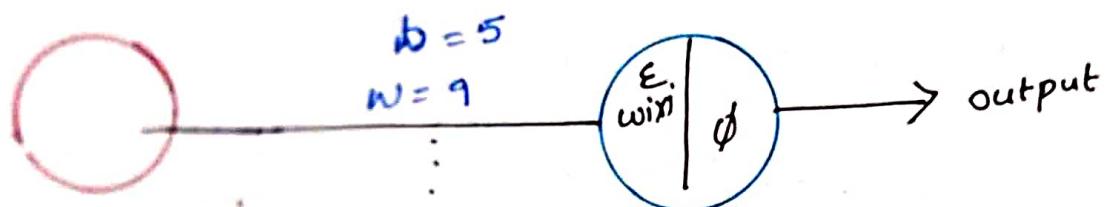
$$x=1, \hat{y} = 40(1) + 5 = 45$$

$$x=2, \hat{y} = 40(2) + 5 = 85$$

\* **adjusting weights**



- after that it changes "bias"  $\Rightarrow b$



$$\therefore y = wx + b$$

$$\Rightarrow y = 1(x) + 10$$

- **iterations** :- try & Errors [ <sup>apply</sup> loop ]  
For better answer.

### important points

- no. of **input** Neurons = no. of input features.
- no. of **Output** Neurons =
  - Regression :- 1
  - binary class :- 1
  - Multiclass :- no. of categories

## \* Activation function of output Neuron:

1. regression  $\Rightarrow$  **relu**
2. binary classification  $\Rightarrow$  **Sigmoid**
3. Multiclass  $\Rightarrow$  **Soft max**

↓  
calculate the probability of  
Each class. and which  
of them having highest probability  
we consider into that class.

Ex :-  $\text{drug}(x), \text{drug}(y), \text{drug}(z)$

$P(\text{drug } x)$   
 $\text{prob}(\text{drug } y)$   
 $\text{prob}(\text{drug } z)$  }  $\rightarrow$  highest

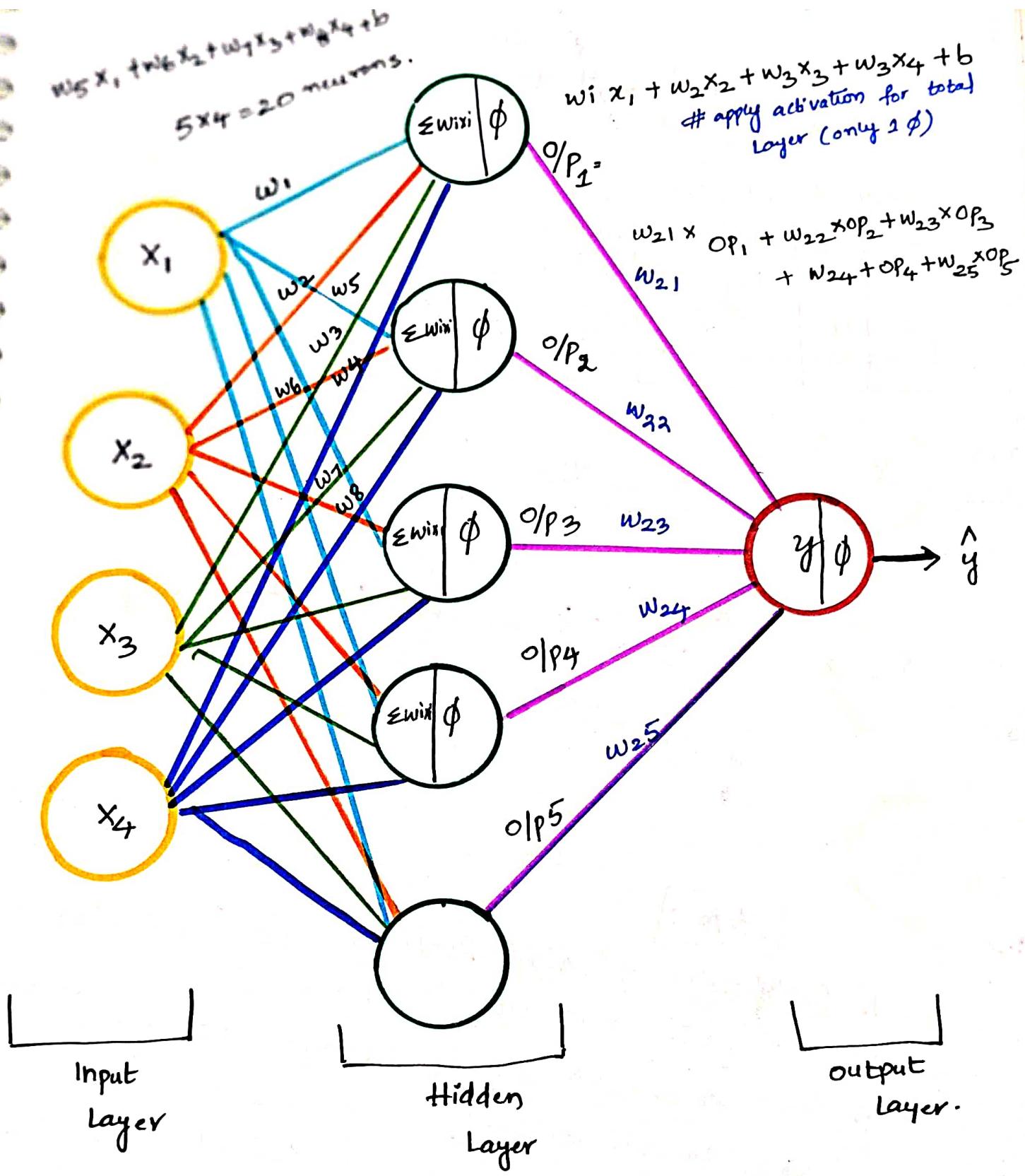
dt:- 09/05/22

7:40pm.

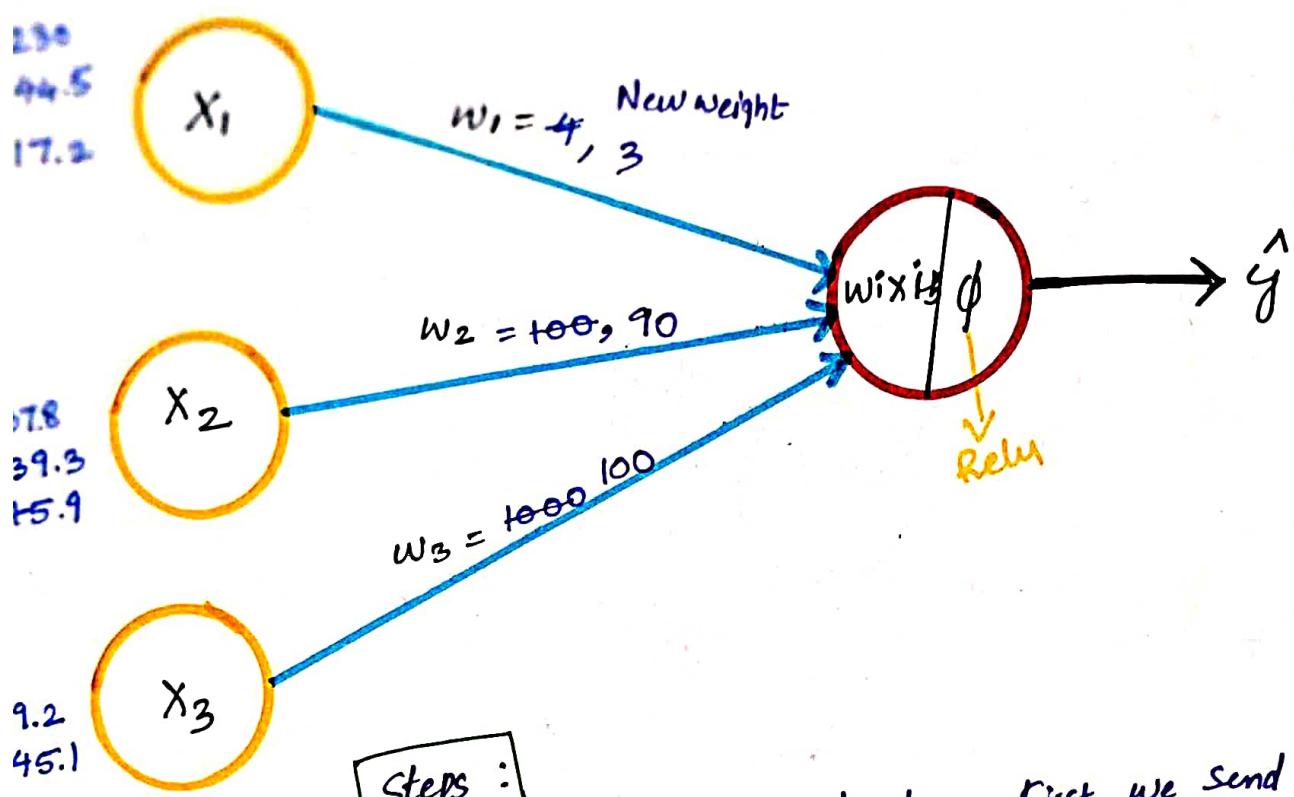
✓  
09/05/22  
5:30pm.

## # How do neural network work?

- # We can Take as per choice, **Hidden Layers**.
- # and Same hidden Layers, we can Take as many as no. of **neurons** in the **neural network**.



TV	Radio	new price	Sales			
230.4	37.8	69.2	22.1	—	—	—
44.5	39.3	45.1	10.4	—	—	—
17.2	46.7	69.3	7.3	—	—	—
931.5	41.3	58.5	16.5	—	—	—
180.8	10.9	58.4	12.9	—	—	—



Steps :

1. initialization of weights Randomly, first we send 230.4, 37.8, 69.2 into input layer (x, x<sub>2</sub>, x<sub>3</sub>) with weights  $230 \times 4$ ,  $37.8 \times 100$ ,  $69.2 \times 1000$  and
2. Then, we calculate  $230 \times 4$ ,  $37.8 \times 100$ ,  $69.2 \times 1000$  and  $(\hat{y})$ . Going to calculate  $(\hat{y})$ . In the neuron, we calculate  $(w_i x_i + b)$  and  $(\phi)(ReLU)$  and
3. In the neuron, we calculate  $(w_i x_i + b)$  and  $(\phi)(ReLU)$  and  $\text{output is predicted } (\hat{y})$ . Similarly same as first, we continue with all data. --
4. Then, it calculate Error  $(y - \hat{y})$
5. After that, it calculate  $(y - \hat{y})^2$  (predicted - Actual)<sup>2</sup>.

7. at last, we want  $(SSE)_{\min}$ .
8. In order to reduce Error, the weights will reupdate to 3, 90, 100
9. again, it will do some pattern, in loop. (In forward & backward).
10. Finally it make weights such that, it gives  $(SSE)_{\min}$ . till that, it process.

Ex:- clear Example of updating weights

$$2x + 5 = 8$$

$\hat{y}$

$\hat{y}$

$\hat{y} - \hat{y}$

$$* x = 100$$

$$2(100) + 5 = 205 \quad 8 \quad -197$$

AS "x" value increasing, Error also increasing.

$$* x = 105$$

$$2(105) + 5 = 215 \quad 8 \quad -207$$

$$* x = 95$$

$$2(95) + 5 = 195 \quad 8 \quad -187$$

as "x" value decreasing, Error also decreasing

$$* x = 90$$

$$* x = 85$$

$$* x = 80$$

$$\frac{dx}{\Delta x} = 5 \quad (\text{change in } x = 5)$$

$$\begin{matrix} \vdots \\ * x = 5 \end{matrix}$$

$$2(5) + 5 = 15 \quad 8 \quad -7$$

$$2(0) + 5 = 5 \quad 8 \quad 3$$

so, The Error is in between -5 & 0

We updated:

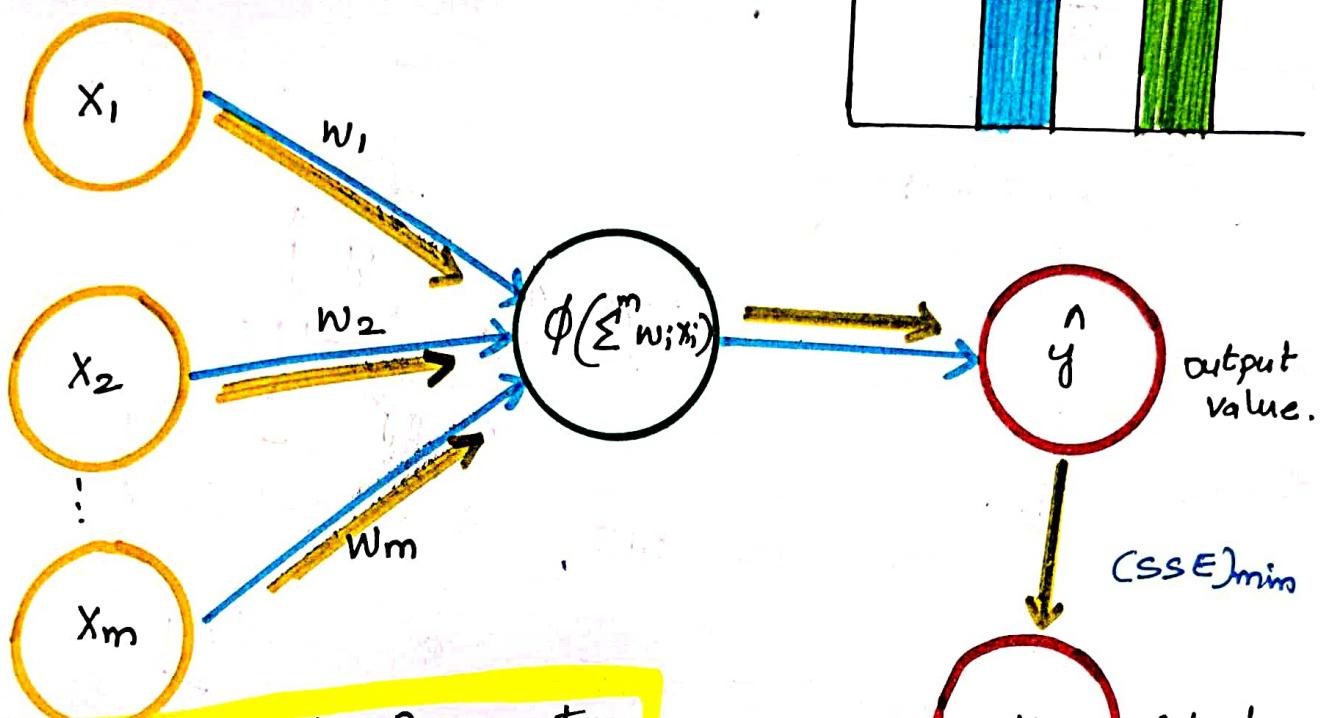
till we get minimum value

$$\begin{matrix} * x = 1 \\ * x = 2 \\ * x = 1.5 \end{matrix}$$

# How do neural networks Learn ?

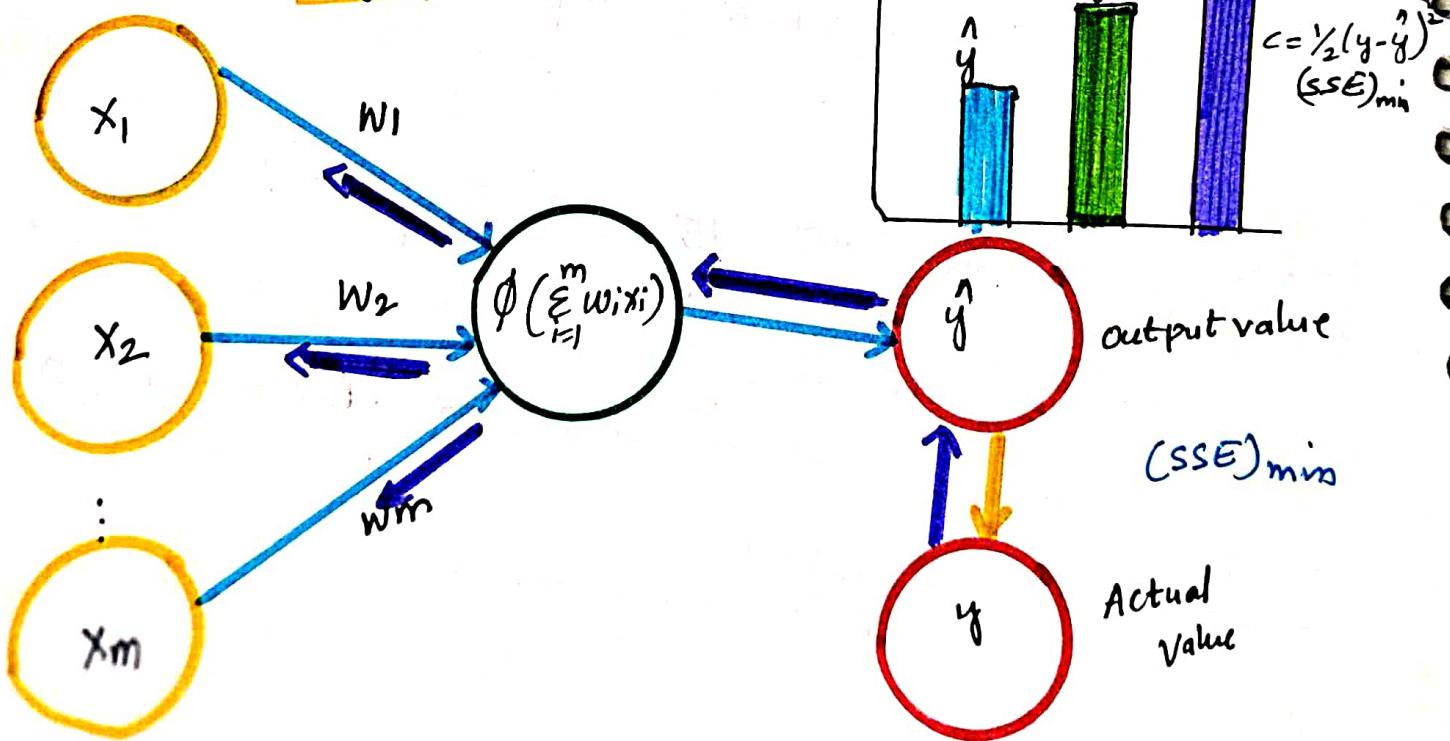
# Neural networks learn based on connections.

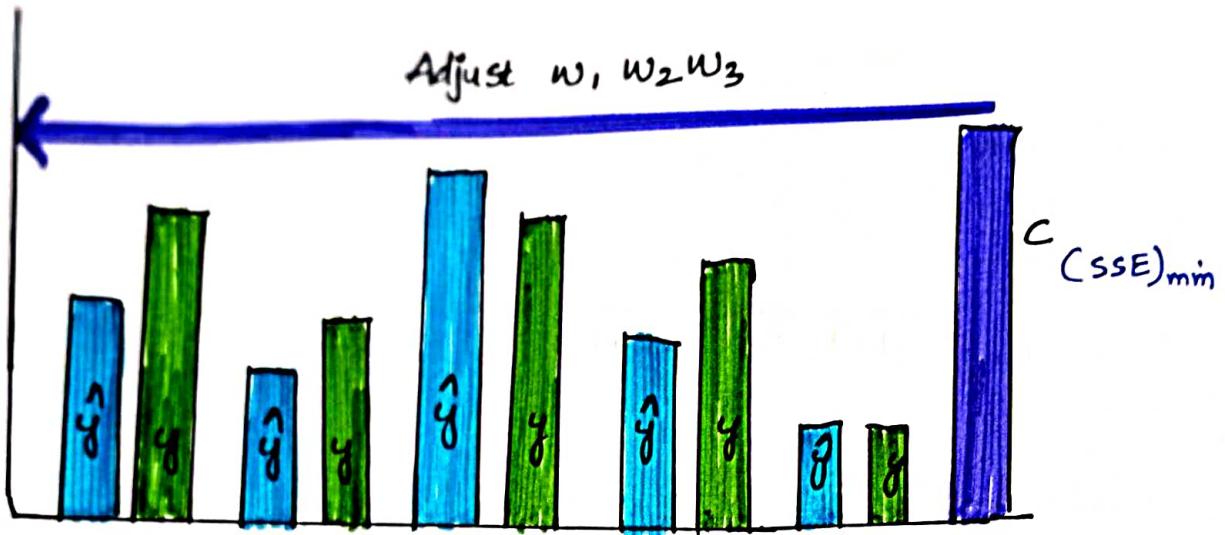
# Forward propagation



# Backward Propagation

# For minimum Error, we have to update weights.





# forward propagation :-

First, calculating  $\sum w_i x_i + b$  and calculating the output is known as Variable in the forward direction.  
forward propagation.

# backward propagation :-

Adjusting (or) updating the weights, coming backward direction is known as backward propagation.  
 initially, weights are assigned randomly, and from that we update weights. The weights will be updated in a way that  $(SSE)_{min}$

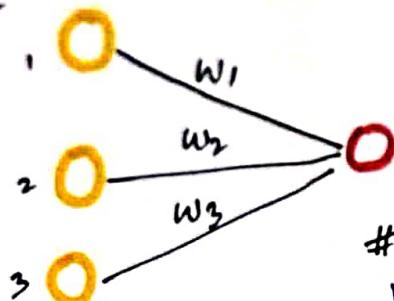
## Weight initialization

### 1. Uniform

$$= \left[ \frac{-1}{\sqrt{\text{inputs}}}, \frac{+1}{\sqrt{\text{inputs}}} \right]$$

# we have select the weights, that forms, uniform distribution.

Ex:-



# Backend code.  
np.random.uniform()

$\left[ \frac{-1}{\sqrt{3}}, \frac{+1}{\sqrt{3}} \right]$  # we have select values in these only.  
# These weights should form uniform distribution.



### 2. Xavier Uniform

$$= U \left[ -\sqrt{\frac{6}{\text{intout}}}, +\sqrt{\frac{6}{\text{intout}}} \right]$$

### 3. Xavier normal

$$= N \left[ 0, \sqrt{\frac{2}{\text{intout}}} \right]$$

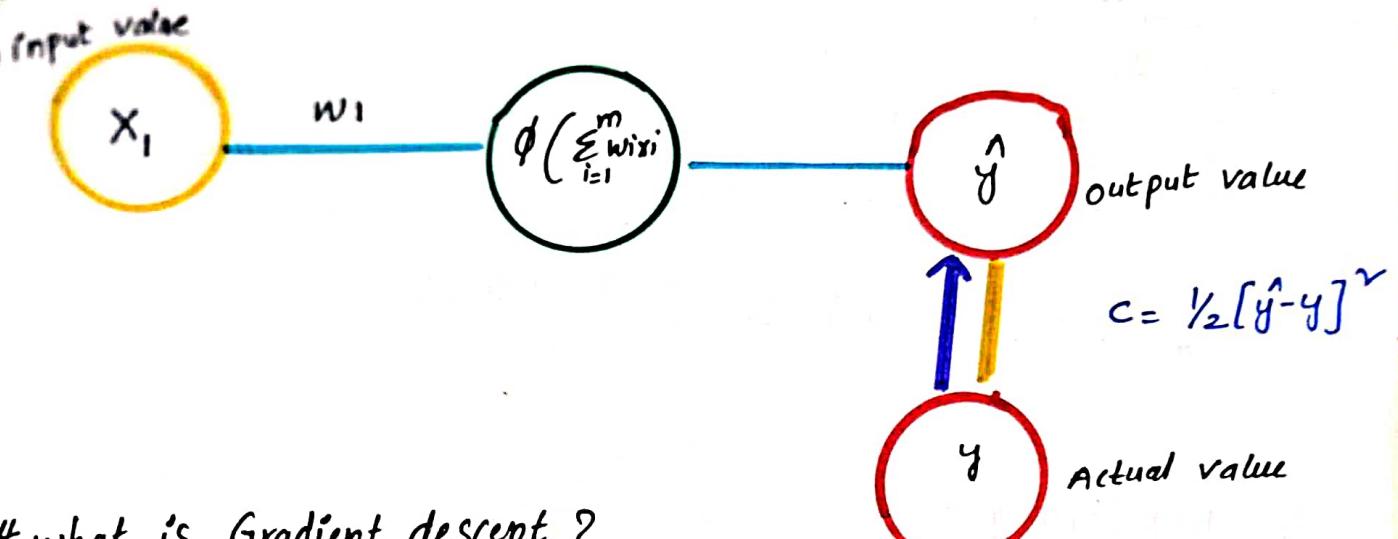
### 4. He - int Uniform

$$= U \left[ -\sqrt{\frac{6}{\text{in}}}, +\sqrt{\frac{6}{\text{in}}} \right]$$

### He - int normal

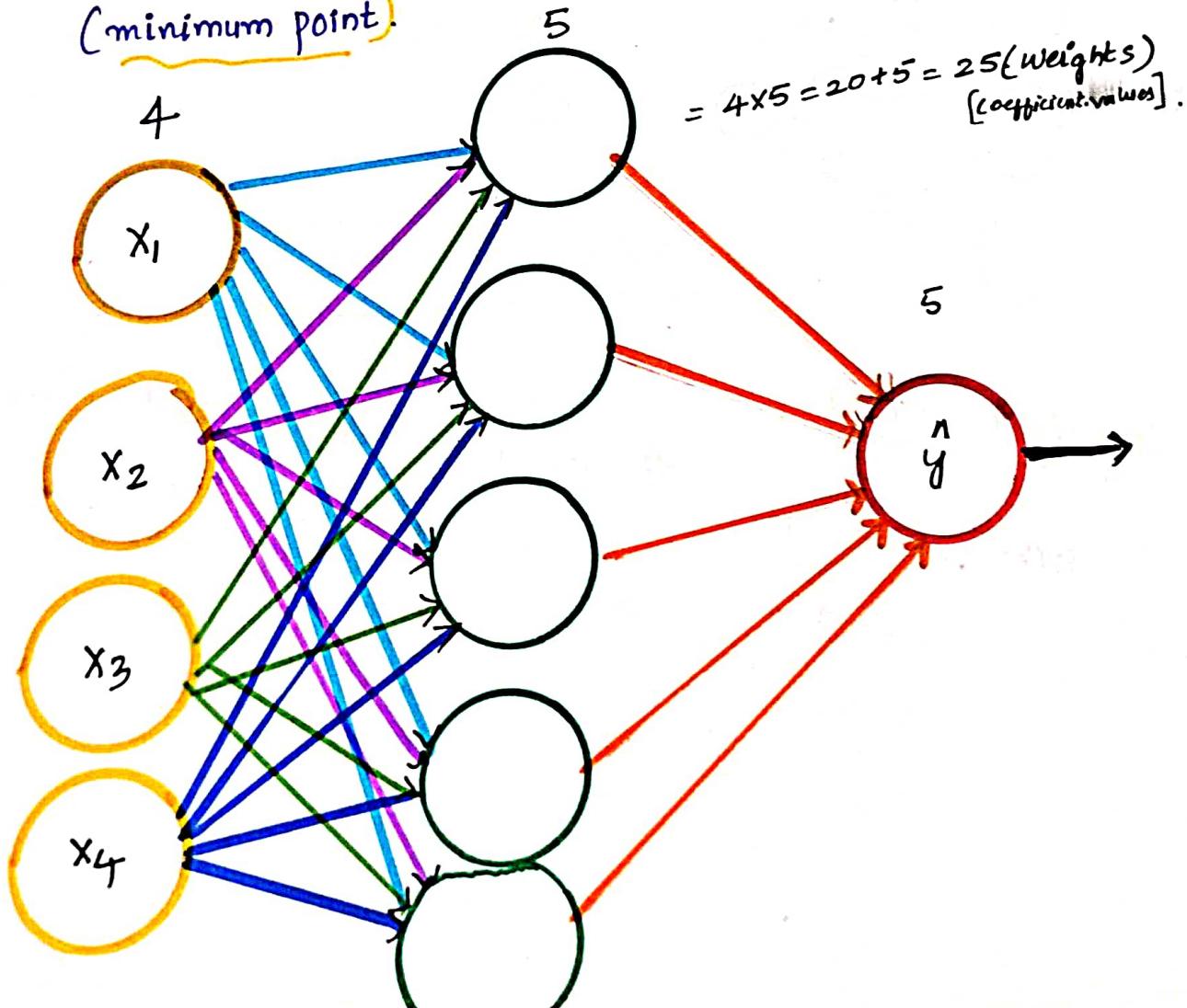
$$= N \left[ 0, \sqrt{\frac{2}{\text{int}}} \right]$$

## Gradient Descent



# what is Gradient descent ?

# Gradually decreasing and identifying the Local minima (minimum point).



In machine learning :

Ex:- In multiple linear regression

model. intercept — , model. coeff —

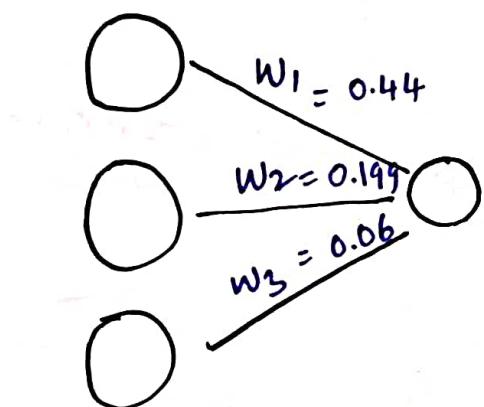
out: (2.708), array (0.44, 0.199, 0.006)

$\beta_0 \quad \beta_1 \quad \beta_2 \quad \beta_3$

$$\hat{y} = b + w_1 x_1 + w_2 x_2 + w_3 x_3$$

$$= 2.7 + 0.44[x_1] + 0.199[x_2] + 0.06[x_3]$$

In deep learning :



# By using this Techinque, it gives "accurate results".

<https://t.me/AIMLDeepThaught/612>

# Gradient descent

$$\text{Ex:- } 2x + 5 = 8$$

$$x = 10^0$$

$$x = 9^9$$

⋮

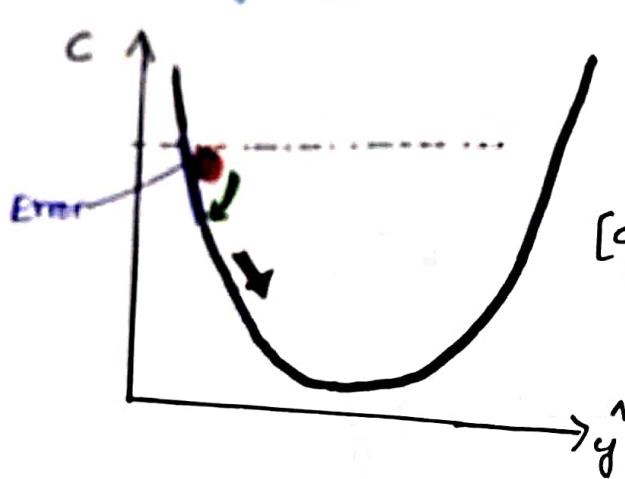
$$x = 0$$

$$x = 1$$

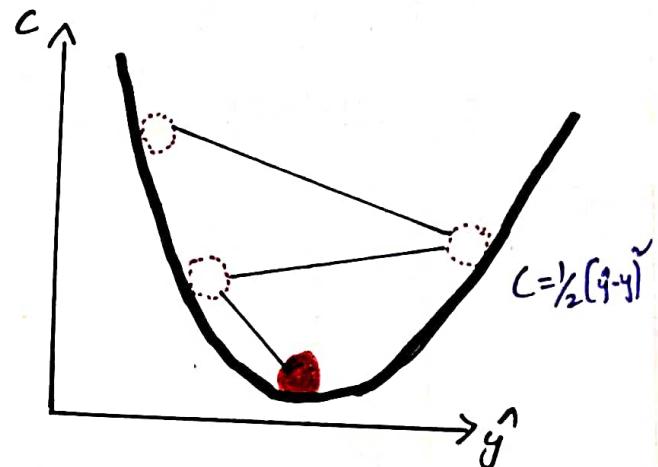
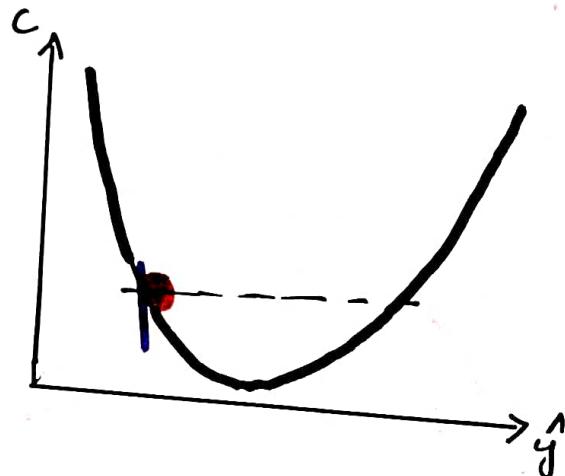
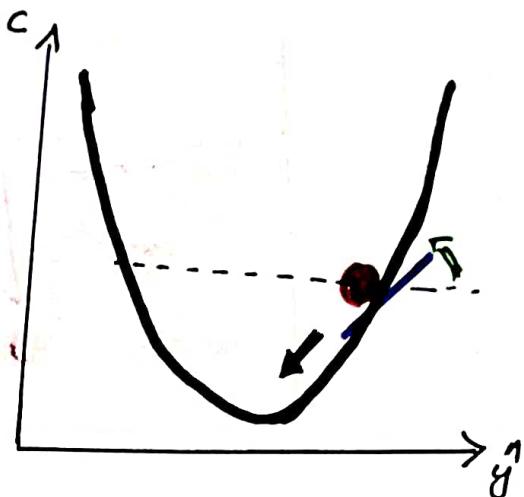
$$x = 1.5$$

# Gradually reducing weights and making Error as minimum value.

\* By applying different "x" values and identifying the "perfect "x" value till, we get error as "0"  $\Rightarrow$  gradient decent



$$[C = \frac{1}{2}(\hat{y} - y)^2]$$



### Stochastic Gradient Descent

- \* Local minima

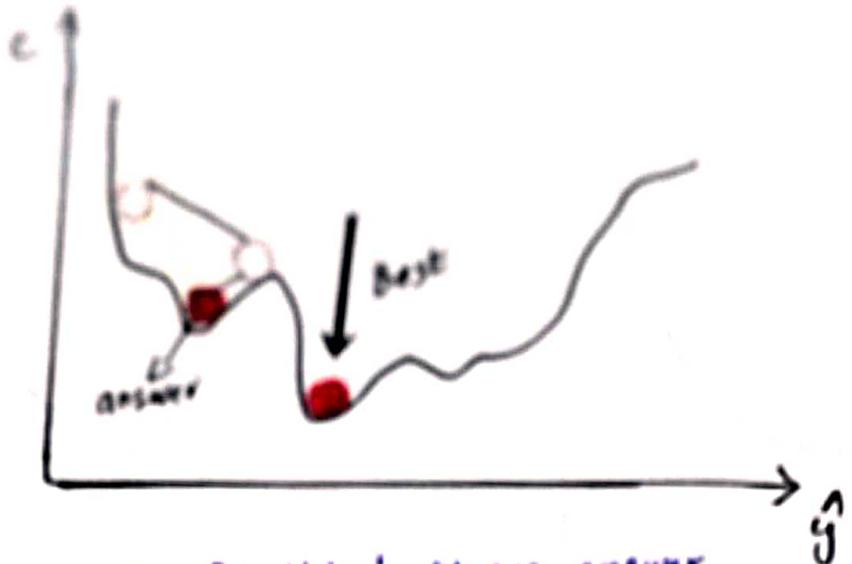
- \* Global minima

EX:-

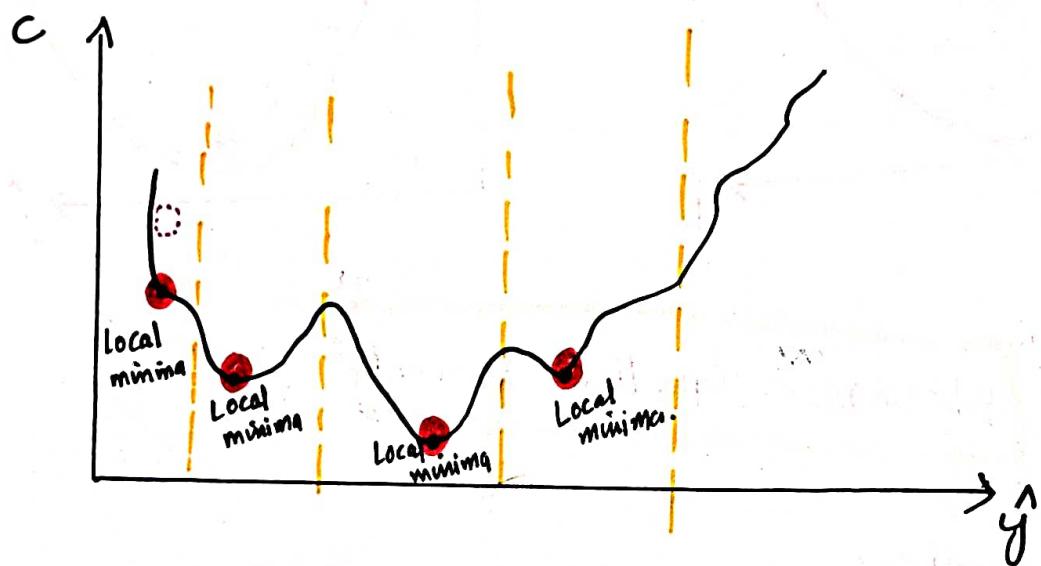
- Every state

1<sup>st</sup> ranker (local minima)

- All India ranker 1<sup>st</sup> ranker (global minima)



# Here, we predicted wrong answer,  
 But, there is another curve. So, How we find that?  
 by dividing into parts.



# identifying the Global minima is called  
 "Stochastic Gradient Descent"

# Entire data we divide into parts (or) batches. Each part  
 We are identifying the "Local minima" and from  
 that overall identify "Global minima".  
 ↓  
 "Error".

02/05/22  
 10:00pm

# ANN - Classification

Code : on Tensorflow, Keras.

Tensorflow, CNTKs, theano  
Google Microsoft Facebook

# PIP install tensorflow  
# PIP install keras

```
import numpy as np
import Pandas as pd
import tensorflow as tf
import keras
```

# from tensorflow import Keras.

# df = pd.read\_csv ("churn-modelling.csv")

# df.head()

out

Row no.	customer id	Surname	Credit Score	Geo graphy	Gender	Age	Tenure	Balance	num of products	Has card	is active member	Estimated Salary
1	1563469	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348
2	15647311	Hill	608	Spain	Female	41	1	83807	1	0	1	112542
3	15619304	Onio	502	France	Female	42	8	159660	3	1	0	1139315
4	15701354	Bonni	699	France	Female	39	1	0.00	2	0	0	93826.6
5	15737889	Mitchell	850	Spain	Female	43	2	125510	1	1	1	79084.

# df.info()

out : RangeIndex : 10000 entries , 0 to 9999

Data columns.

→ Row number

→ Customer Id

Nonnull columns

10000 Non null

1000 Non null

Dtype.

int64

int64

3. Credit score	10000 non-null	int64
2. Surname	10000 non-null	object
categorical 4. Geography	10000 non-null	object
5. Gender	"	object
6. Age	"	int64
7. Tenure	"	float64
8. Balance	"	int64
9. No. of Products	"	int64
10. HasCrCard	"	int64
11. Is Active member	"	float64
12. Estimated Salary	"	int64
13.Exited	"	int64

# we can use drop function also [x & y]

#  $x = df. iloc[:, 3:13]. values$

#  $y = df. iloc[:, 13]. values$

$\begin{matrix} \text{:} \\ \text{all} \end{matrix} \quad \begin{matrix} 3:13 \\ 3 \text{ to } 12 \end{matrix} \quad \begin{matrix} [ : , 13 ] \\ \text{only} \\ 13 \text{ column} \end{matrix}$

#  $x. shape$

[out]: (10000, 10)

Original Data  
13 input  
1 output  
10 inputs  
1 column  
3 drop (row no. customer id, surname)

#  $y. shape$

[out]: (10000, )

# encoding [categorical data]

from sklearn. pre processing import Label Encoder

#  $x[:, 2]$

[out]: array [ "Female", "Female", ... "Male", "Female" ].

natural  
ordinal data (sequence)

```
# labelencoder = LabelEncoder()
```

```
# x[:,2] = labelencoder.fit_transform(x[:,2])
```

# changed to numerical.

```
# x[:,2]
```

```
[out]: array [0,0,0 ... 0,1,0]
```

# one hot encoding (nominal data = Geography (column)) # we can use get-dummies.

```
# one hot encoding
```

```
from sklearn.preprocessing import OneHotEncoder
```

```
from sklearn.compose import ColumnTransformer
```

\*\*\* from sklearn.compose import ColumnTransformer  
↳ transforms data.

```
# x[:,1]
```

```
[out]: array ([ "France", "Spain", "France" ... "France", "Germany" ] ).  
dtype: object.
```

```
# ct = ColumnTransformer ([{"encoder": OneHotEncoder(), [1]},  
, remainder = "passthrough")
```

```
# x = np.array (ct.fit_transform(x))
```

# changed to numerical

```
# x[:,1]
```

```
[out]: array ([0.0, 0.0, 0.0, ... 0.0, 1.0, 0.0]).
```

## train-test

```
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)
```

## # Shape.

```
# x-train.shape, x-test.shape, y-train.shape, y-test.shape  
out: (8000, 12), (2000, 12), (8000), (2000),  
↳ after applying  
dummie it converts into 12 columns.
```

## Scaling

```
from sklearn.preprocessing import StandardScaler
```

```
# sc = StandardScaler()
```

```
# x-train = sc.fit_transform(x-train)
```

```
# x-test = sc.fit_transform(x-test)
```

## MODELLING

### initializing The ANN

```
from keras.models import Sequential
```

```
# ann = Sequential()
```

## Adding The input layer and first hidden layer.

from Keras.layers import Dense.

```
# ann.add(Dense(input_dim = 12, units = 6, kernel_initializer = "uniform", activation = "relu"))
```

connections  
Hidden Layer  
input\_dim = 12, units = 6, kernel\_initializer = "uniform", activation = "relu"  
12x6 = 72 Dense weights  
[-1/sqrt(n), +1/sqrt(n)]  
[-1/sqrt(12), +1/sqrt(12)]

## Adding the Second hidden Layer.

```
# ann.add(Dense(units = 6, kernel_initializer = "uniform", activation = "relu"))
```

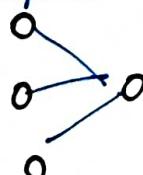
activation = "relu"  
relu  
relu

## Adding the Output layer

```
# ann.add(Dense(units = 1, kernel_initializer = "uniform", activation = "sigmoid"))
```

output  
sigmoid.  
relu  
relu

Ex:- Perception model.



(ann.add(Dense(input\_dim = 3, units = 1, kernel\_initializer = "uniform", activation = "relu"))).

## Part-3 Training the ANN

## Compiling the ANN

if past: connection is made (writing), current (data)  $E(y-y')$   
Regression = mean\_squared\_error

training the ANN on training Set

# ann. fit (x-train, y-train, epochs = 100)  
↳ no. of iteration.  
(backward, forward  
directions)

Out Epoch 1/100

Epoch 1/100  
250/250 [=====] - 2s 1ms/step - loss: 0.6457, -accuracy: 0.7830

Epoch 2/100

— — — —

Epoch 100

25 - 100

## Part-4 Predictions & Evaluating model

## making the predictions

#  $y_{-pred} = \text{ann}.\text{Predict}(x_{-test})$  # Test accuracy.

# y-pred = 0.5

$$\# \quad y_{-pred} = (y_{-pred} > 0.5)$$

## Evaluating the model

```
from sklearn.metrics import Confusion_matrix, accuracy_score  
# print ("Test Accuracy:", accuracy_score(y-test, y-pred))  
# Confusion_matrix(y-test, y-pred)  
[out]: Test accuracy : 0.8455  
array([[1536, 59  
       [250, 155]]])
```

## Cross validate the model

# user defined function

```
def build_cross_classifier():  
    ann (or) classifier = Sequential()  
    anything we can use input layer  
    classifier.add(Dense(input_dim=12, units=6, kernel_initializer="uniform", activation="relu"))  
    classifier.add(Dense(units=6, kernel_initializer="uniform", activation="relu"))  
    classifier.add(Dense(units=1, kernel_initializer="uniform", activation="sigmoid"))  
    classifier.compile(optimizer="adam", loss="binary_crossentropy", metrics=["Accuracy"])  
    return classifier
```

### classifier

```
from keras.wrappers.scikit_learn import KerasClassifier
```

```
# classifier = Kerasclassifier (build_fn = build_cross_classifier  
                                , batch_size = 10 , epochs = 100)  
                                User defined function  
                                if stochastic descent Iterations  
                                With
```

```
from sklearn.model_selection import cross_val_score
```

# accuracies = cross\_val\_score (estimator = Classifier,  
X = x\_train, y = y\_train, cv=5)

# print (accuracies) <https://t.me/AIMLDeepThaught/612>

```
# accuracies.mean()
```

```
# accuracies.mean()  
[0.8237, 0.8343, 0.8362, 0.8362, 0.8374]
```

out 6.83362

## ⇒ Hyperparameter Tuning

### Part-5 - Improving and Tuning the ANN

This can be done by 3 options

1. Hyperparameter tuning
2. Regularization ( $L_1$  &  $L_2$ ) to reduce overfitting if needed (for regression problems)
3. Dropout

Hyperparameter tuning can be done for identifying no. of hidden layers & no. of neurons in each hidden layer.

- Layers =  $[[20], [10], [30], [40]] \rightarrow$  1 hidden layer with different options of no. of neurons in hidden layer
- Layers =  $[[20], [40, 20], [45, 30, 15]] \rightarrow$  Multiple hidden layers with different options of no. of neurons.

Hyperparameter tuning can be done for identifying best activation function for hidden layers.

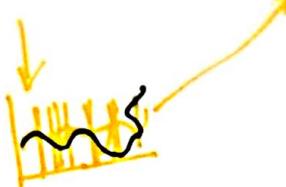
- activations = `["relu", "sigmoid"]`

Hyperparameter tuning can be done for identifying best optimizers

- Optimizer = `["adam", "rmsprop"]`  
gradient descent ↳ Stochastic gradient descent

Hyper parameter tuning for batchsize for building/training model.

• batch size =  $[10, 16, 32, 64, 128, 256]$



The diagram illustrates a single layer of a neural network. It shows a vertical column of input data (represented by small yellow squares) and a vertical column of weights (represented by small blue squares). The weights are multiplied by the inputs, and the results are summed to produce the final output. An arrow points from the text 'batch size' to this diagram.

Code for tuning :-

```
def build_classifier(optimizer):  
    classifier = Sequential()  
    classifier.add(Dense(input_dim=12, units=6, kernel_initializer="uniform",  
                         activation="relu"))  
    classifier.add(Dense(units=6, kernel_initializer="uniform",  
                         activation="relu"))  
    classifier.add(Dense(units=1, kernel_initializer="uniform",  
                         activation="sigmoid"))  
    classifier.compile(optimizer=optimizer, loss="binary_crossentropy",  
                        metrics=["accuracy"])  
    return classifier
```

```
# classifier = KerasClassifier(build_fn=build_classifier)  
# parameters = {"batch_size": [10, 32], "epochs": [50, 100],  
#               "optimizer": ["adam", "rmsprop"]}
```

```
from sklearn.model_selection import GridSearchCV  
# grid = GridSearchCV (estimator = classifier, param_grid =  
# Parameters, scoring = "accuracy", cv=5)  
# grid_result = grid.fit (x-train, y-train)
```

```
[out]: Epoch 1/100  
200/200 [=====] - os 7110s/step - loss: 0.5877, accu. 0.717 ]  
Epoch 2/100  
200/200 [=====] - os 6300s/step - loss: 0.4392 - acc - 0.802 ]  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
Epoch 100/100  
200/200 [=====] - os 8300s/step - loss: 0.6392. acc - 84 ]
```

```
# grid_result . best - params -
```

```
[out]: { "batch_size": 32, "epochs": 50, "optimizer": "adam" }
```

# best accuracy

```
# grid_result . best - score -
```

```
[out]: 0.844125
```

```
# y-pred = grid_result . predict (x-test)
```

```
# y-pred = (y-pred > 0.5)
```

# accuracy, confusion-matrix

```
from sklearn.metrics import Confusion_matrix, accuracy_score.
```

```
# print ("test accuracy", accuracy_score(y-test, y-pred))
```

```
# Confusion-matrix (y-test, y-pred)
```

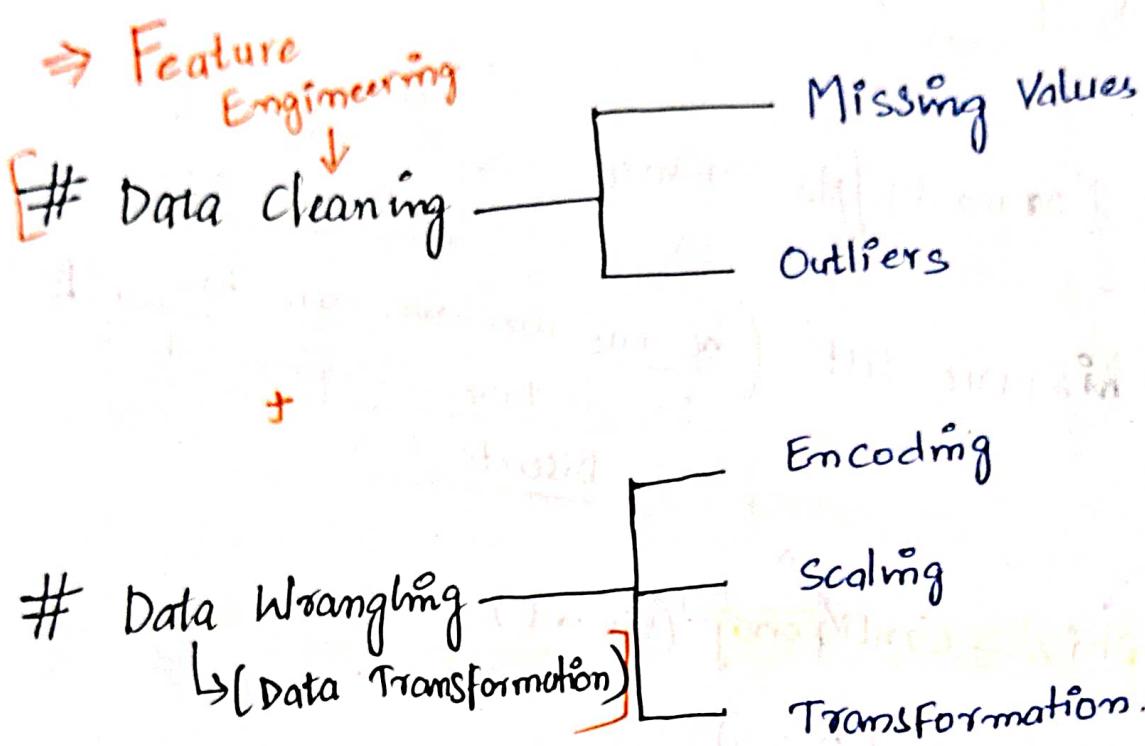
out: Test accuracy : 0.8405

array ([[1530, 65]  
[254, 151]],

2/10/2022  
10.05/22  
5:30 pm.

02/24/22  
8:50pm

## \* Data Preprocessing / Data Preparation



Us.  
(Car Insurance) # Data cleaning (In spyder)

# import numpy as np

# import pandas as pd

why two dataset  
Because to check old  
and new dataset.

⇒ Load Data Set

# df\_raw = pd.read\_csv ("claimants.csv")

# df = pd.read\_csv ("claimants.csv")

[Out]: df\_raw | DataFrame | 1340, 7 | column names : CASENUM,  
df | DataFrame | 1340, 7 | column names : "

⇒ Check is there null values.

# df.isnull().sum()

out :-	CASENUM	0
	CLMSEX	12
	CLMINSUR	41
	SEATBELT	48
	CLMAGE	189
	LOSS	0
	ATTORNEY	0

MISSING values.

# Data Understanding

Case Num :-	Number of case (insurance)	
Discrete Clm sex :-	0, 1 [Male] [Female]	Person who
Discrete Clm Insur :-	1 [Yes], 0 [No]	# having Insurance is
Discrete Seat belt :-	0 [No], 1 [Yes]	or not Clamming
Continuous Clm age :-	Age of the person	Insurance is called
Loss :-	Loss / Damage happened	Seat belt "claimants"
Attorney :-	Whether they hire a lawyer	During the accident
	(or) not, [0] No, [1] Yes.	

# There are various methods for filling null values

\* For Continuous Data

mean (no outliers)

median (outliers)

Because Mean is

\* For Discrete Data

Mode

(Less affected) Impacted with outliers

# Replacing null values For **discrete** Variables

~~~~ \* ~~~\* ~~~~~ So, we have to Replace Them with "**mode**"

# df[ "CLMSEX" ]. value\_counts() (mode)

**Out** :- 1.0 742 → Highest, so we Replace with "1"

# df[ "CLMSEX" ]. fillna(1, inplace = True)

↓ changing in original Data = TRUE

**Out** Filled with "1"

# df[ "CLMINSUR" ]. value\_counts()

**Out** :- 1.0 1820 → MODE

# df[ "CLMINSUR" ]. fillna(1, inplace = True)

**Out** Filled with "1", so replaced with "1" value

```
# df[["SEATBELT"]].valuecounts()
```

Out : 0.0 1270 → Mode.  
1.0 22

```
# df[["SEATBELT"]].fillna(0, inplace=True)
```

Out : Filled with "0"

Time  
11:30pm

# Replacing null Values For Continuous Variables  
By  
# Pandas

```
# df[["CLMAGE"]].mean()
```

Out 28.414

```
# df[["CLMAGE"]].median()
```

Out 30.00

```
# df[["CLMAGE"]].fillna(28.414, inplace=True)
```

Out Filled with 28.414

If we ask again what is median value.

```
# df[["CLMAGE"]].median()
```

28.414, so it changes 30.00 To 28.414

Because we Filled with null values  
28.414.

# # sklearn (scikit Learn) (One stop shop For ml)

## \* SIMPLE IMPUTER

# Load data set again  
Library → (module) (Function)

From

Sklearn. impute

import SimpleImputer

Ex: missing values = "???"

calculating mean Value

strategy = "mean" →  $\frac{\Sigma}{n}$

# mean\_imputer = SimpleImputer (strategy = "mean")  
(Function) → (Argument)

name  
as

Function. (or) Store in  
mean\_imputer

Ex:- Core Python

[Keyword

argument]

If we don't  
write.  
This time  
it gives

Same : a = 4

# Converting array  
To data frame  
To Remove  
array. From mean value.

# df [ "CLMAGE" ] = pd. DataFrame (mean\_imputer. fit\_trans

array [ 50.,  
18.,  
5.,  
:  
]

storing

- Trans

form

(df [ [ "CLMAGE" ] ] )

Fill in This column  
(or) Replacing

Calculate  
The mean  
value

Out

Filled with 28.414

For Median Only change mean To median

# median\_imputer = SimpleImputer (strategy = "Median")

# df[["CLMAGE"]] = pd.DataFrame (median\_imputer. fit -  
Transform (df[["CLMAGE"]]))

For Mode [Discrete Data]

mode\_imputer = SimpleImputer (strategy = "mode")

# df[["CLMSEX"]] = pd.DataFrame (mode\_imputer. fit\_transform (df[["CLMSEX"]]))

# df[["CLMINSUR"]] = pd.DataFrame (mode\_imputer. fit\_transform  
(df[["CLMINSUR"]]))

# df[["SEATBELT"]] = pd.DataFrame (mode\_imputer. fit\_transform  
(df[["SEATBELT"]]))

# **Output** Fills Directly The mode  
Value in to the dataset.

# MODE = it is for Discrete Data.

# MEAN  
# MEDIAN → Continuous Data

3/4/22 2:08 AM

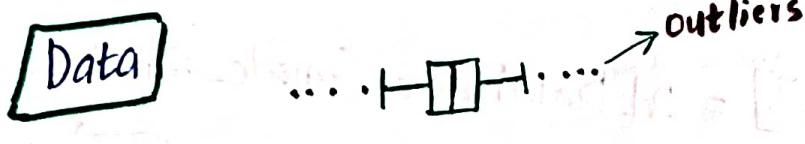
# Outliers → Applicable Only For

Continuous Data

<https://t.me/AIMLDeepThought/612>

1 Que :- What is Outlier?

Ans :- An Outlier is a data point in a dataset.  
that is distant from all other observation  
which is significantly different from the remaining



(or)

- The data point that lies outside the overall distribution of the dataset.

2 Que :- What are Impacts having Outliers in a DataSet?

Ans :- It will cause various problems in statistical analysis. (It may cause a significant impact on mean and standard deviation)

Ex :-  $\bar{x}$  = mean (affect)

$\frac{\sum (x - \bar{x})^2}{n-1}$  = variance

$$\sqrt{\frac{\sum (x - \bar{x})^2}{n-1}} \quad (\text{standard deviation})$$

all are affected by outliers

\* In addition, Some **Machine learning** models are **Sensitive to Outliers**, which may **Decrease** **Performance** **There**

3 Que : Reasons For Outliers ?

- Ans : 1. **Data Entry Errors** (Ex:- Salary = 100,000, But 10,000 Entering)
2. **Measurement Errors** (Ex:- Measuring in meters instead of KM)
3. **Instrumental Error**

4 Que :- Types of Outliers ?

- Ans :- **Univariate Outlier**  $\rightarrow$  Identifying Outlier For **Single Variable**
- \* **Bivariate Outlier**  $\rightarrow$  Identifying as Outlier by **Analyzing 2 Variables at a Time**

# Importing Libraries

- # Import numpy as np
- # Import pandas as pd
- # Import matplotlib as plt
- # Import Seaborn as sns

## Real Time Case study

### boston - House Price Dataset

```
# boston = pd.read_csv("D:\\data science\\Shubham\\Krishna\\class\\6. Data cleaning\\boston.csv")
```

```
boston.head()
```

out

|   | CRIM    | ZN   | INDUS | CHAS | NOX   | RM    | AGE  | DIS   | RAD | TAX  | PTRATIO |
|---|---------|------|-------|------|-------|-------|------|-------|-----|------|---------|
| 0 | 0.00632 | 18.0 | 2.31  | 0    | 0.538 | 6.575 | 65.2 | 4.090 | 1   | 29.6 | 15.3    |
| 1 | 0.02731 | 0.0  | 7.07  | 0    | 0.469 | 6.421 | 78.9 | 4.967 | 2   | 24.2 | 17.8    |
| 2 | 0.02729 | 0.0  | 7.07  | 0    | 0.469 | 7.185 | 61.1 | 4.967 | 2   | 24.2 | 17.8    |
| 3 | 0.03237 | 0.0  | 2.18  | 0    | 0.458 | 6.998 | 45.8 | 6.082 | 3   | 22.2 | 18.7    |
| 4 | 0.06905 | 0.0  | 2.18  | 0    | 0.458 | 7.147 | 54.2 | 6.062 | 3   | 22.2 | 18.7    |

```
# boston.info()
```

506 rows, 14 columns

out

|    |         |         |
|----|---------|---------|
| 0  | CRIM    | float64 |
| 1  | ZN      | float64 |
| 2  | INDUS   | float64 |
| 3  | CHAS    | int64   |
| 4  | NOX     | float64 |
| 5  | RM      | float64 |
| 6  | AGE     | float64 |
| 7  | DIS     | int64   |
| 8  | RAD     | int64   |
| 9  | TAX     | float64 |
| 10 | PTRATIO | float64 |
| 11 | B       | float64 |
| 12 | LSTAT   | float64 |
| 13 | PRICE   | float64 |

| B      | LSTAT | PRICE |
|--------|-------|-------|
| 396.90 | 4.98  | 24.0  |
| 396.90 | 9.14  | 21.6  |
| 392.83 | 4.03  | 34.7  |
| 394.63 | 2.94  | 33.4  |
| 396.90 | 5.33  | 36.2  |

(float64(11), int64(3))

## Various ways of finding The Outlier

Changing Every Value to Z score.

option ①

**Z score**

$$|Z\text{ score}| > 2$$

$$\left[ \begin{array}{l} \therefore Z < -2 \\ Z > +2 \end{array} \right] \left| \begin{array}{l} \text{modulus} \end{array} \right|$$

Ex :-

| X | Z <sub>score</sub> |
|---|--------------------|
| 6 | 4.56               |
| 4 | 9.56               |
| 9 | 7.56               |
| 7 | 2.56               |
| 2 | -1.56              |

$$Z = \frac{X - \mu}{\sigma}$$

$$I = \frac{6 - 5.6}{1}$$

$$\therefore \mu = \frac{6 + 4 + 9 + 7 + 2}{5}$$

$$\mu = 28/5 = 5.6$$

$$Ex: \sigma = 1$$

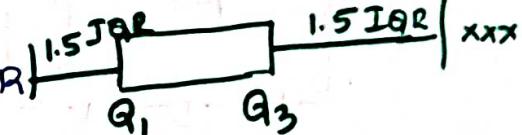
option ②

**IQR**

(Inter Quartile Range)

any Value  $< Q_1 - 1.5 \text{ IQR}$

any Value  $> Q_3 + 1.5 \text{ IQR}$



# any value Less than  $Q_1 - 1.5 \text{ IQR}$  is Outlier

# any value More Than  $Q_3 + 1.5 \text{ IQR}$  is Outlier

option ③

**Visualization**

\* Box plot, Histogram For Univariate Outliers

\* Scatter plot For bivariate Outliers.

option ③

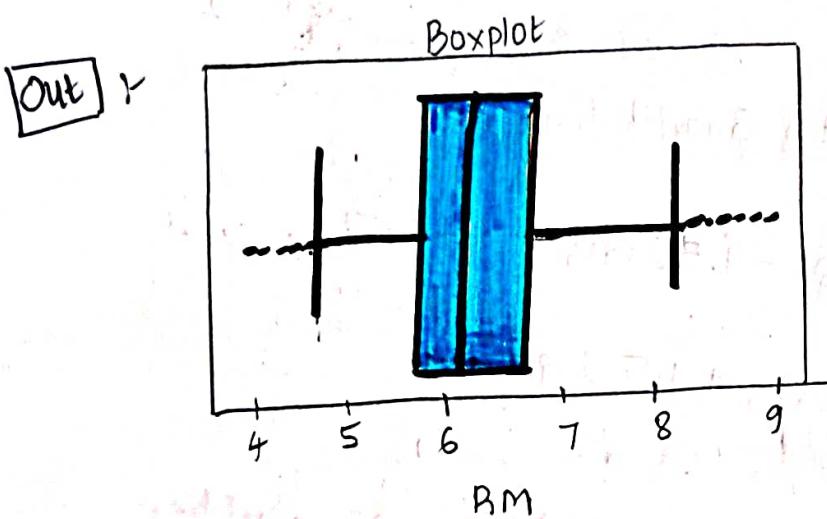
Detection of Outliers of "RM" column (Based on Boxplot)

# sns. boxplot (boston.RM)

plt. title ("Boxplot")

plt. show()

\* Overall view of Outlier



Outliers in Both Tails of "RM".

option ①

Detection of Outliers of "RM" column (Based on Zscore)

\* For Exact Outliers in Data

$$Z\text{score} = \frac{x - \mu}{\sigma}$$

# boston["RM"].mean(), boston["RM"].std()

[out] (6.284, 0.702)

# boston["RM\_Zscore"] = (boston["RM"] - boston["RM"].mean()) / boston["RM"].std()

creating new column  
and storing Zscores

$$\mu \quad \sigma$$

Output :-

clarity :-

$$\frac{x - \mu}{\sigma} = \frac{(\text{boston}['RM']) - \text{boston}['RM'].mean())}{\text{boston}['RM'].std()}$$

$$\frac{x - \mu}{\sigma}$$

$$\frac{x - \mu}{\sigma}$$

E Answers Ami

$$\text{boston}['RM\_Zscore'] =$$

↑ Endulo store age + thage.

ZSCORE

|     | CRIM    | ZN   | INDUS | CHAS | NOX   | RM  | AGE | DIS | RAD | TAX | PTRATIO | B   | ... RM SCORE |
|-----|---------|------|-------|------|-------|-----|-----|-----|-----|-----|---------|-----|--------------|
| 0   | 0.0632  | 18.0 | 2.31  |      | 6.515 |     |     |     |     |     |         |     | 0.413263     |
| 1   | 0.02731 | 0.0  | 7.07  |      | 6.421 |     |     |     |     |     |         |     | 0.194082     |
| 2   | 0.02729 | 0.0  | 2.18  |      | 7.185 |     |     |     |     |     |         |     | 1.281446     |
| ... | ...     | ...  | ...   | ...  | ...   | ... | ... | ... | ... | ... | ...     | ... | ...          |
| 505 | 0.04741 | 0.0  | 11.93 |      | 6.030 |     |     |     |     |     |         |     | -0.362408    |

506 x 15 columns

Now, Identify Outliers in this column ↑

# Outlier =  $\text{boston}[(\text{boston}['RM\_Zscore'] > +2) \text{ or } (\text{boston}['RM\_Zscore'] < -2)]$

2 conditions

Greater Than +2  
Less Than -2

[Out] :- Outlier

| CRIM | RM SCORE |
|------|----------|
| 97   | 2.539    |
| 98   | 2.185    |
| 162  | 3.47     |
| 163  | -2.121   |
| 66   | -3.055   |

3/4/22 4:30 AM

Time  
1:00pm

## # Outlier. Shape

**Out** :- (31, 15)

option 2  
# Detection of Outlier of RM (based on IQR)

- Calculate First ( $Q_1$ ) and third quartile ( $Q_3$ )
  - Find interquartile range ( $Q_3 - Q_1$ )
  - Find Lower bound  $Q_1 - 1.5$  & Find upper bound  $Q_3 + 1.5$
- for Exact no. of range + - of outliers

#  $Q_3 = \text{boston}["RM"].quantile(0.75)$

#  $Q_1 = \text{boston}["RM"].quantile(0.25)$

$$Q_1 = 0.25(w)$$

$$Q_3 = 0.75(w)$$

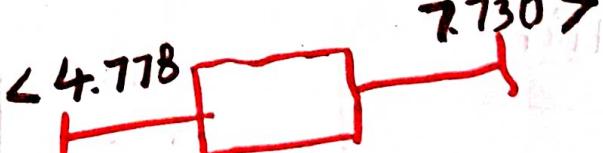
$$IQR = Q_3 - Q_1$$

$$\text{Lower-limit} = Q_1 - (IQR * 1.5)$$

$$\text{Upper-limit} = Q_3 + (IQR * 1.5)$$

Lower-limit, upper-limit

**Out** :- 4.778, 7.730



- \* Which is having less than 4.778 is outlier
- \* Which is having greater than 7.730 is outlier

## Methods of deal The Outliers

- 3R → \* Remove → Deleting The Outliers
- Technique \* Replace → changing values / Data Manipulation
- \* Retain → Treat Them Separately.  
Work without outlier / with outlier  
"Work" Separately.

Outliers should be detected and "REMOVED" Only From Training data set, NOT from The Test set. So we should first divide our data set into train and test. and remove Outliers in the train set, but keep those in test set, and Measure how well our model is doing.

option 1 Removing \* (Let's trimm The dataset) 74.77  
# boston\_trimmed = boston[(boston["RM"]  $\rightarrow$  Lower-Limit) & (boston["RM"]  $\leftarrow$  Upper-Limit)] 7.73

boston\_trimmed  
Out: Which ever Data is Greater than 4.77 and Less Than 7.73 Consider That data Only, Remove Remaining data.

|   | RM    | RM - Zscore |
|---|-------|-------------|
| 0 | 6.515 | 0.413263    |
| 1 | 6.421 | 0.194082    |
| 2 | 7.185 | 1.2814416   |
| 3 | 6.794 | 0.724955    |
| 4 | 6.030 | -0.362408   |

∴ From 506 records, Removed 30 records

that Means 476 having (Or) there.

⇒ it's about 6% of the data is Removed

⇒ it is Only in One Variable (1 column) = 6%.

⇒ If Total Variable (all columns)  $\frac{1,2,3,4, \dots}{(14) \rightarrow (10) \text{ continuous data}}$  Remove May Delete 100 rows  
20% of Data

So, In real Life, we don't use outliers "Remove"

Even, Though, If we Want Remove, still it Contains

Outliers.

<https://t.me/AIMLDeepThaught/612>

⇒ Outliers in The Trimmed dataset

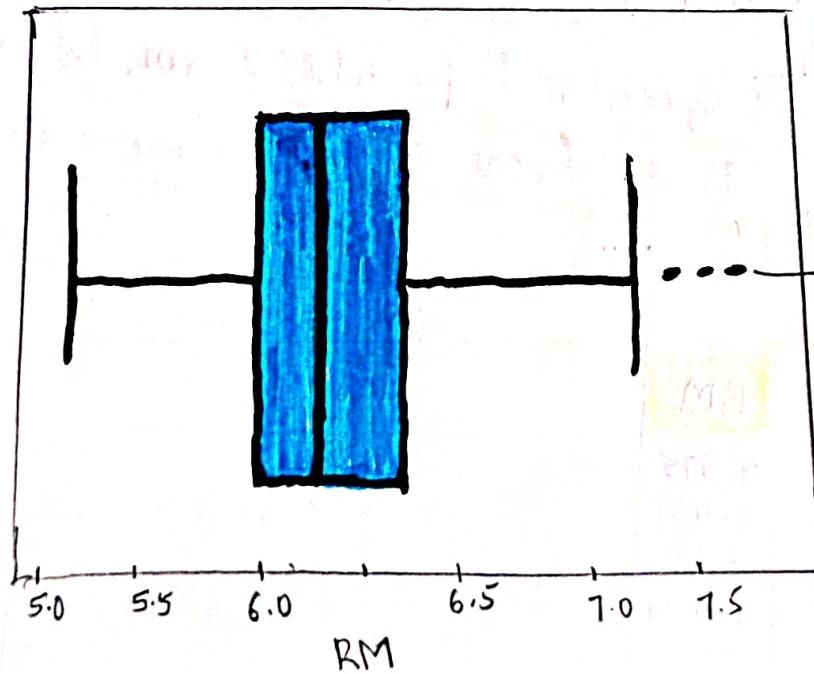
# sns.boxplot(boston\_trimmed.RM)

plt.title("Boxplot")

plt.show()

Boxplot

Out



still, we have Outliers, after Removing The Outlier records

Q: Why? it showing Outliers after Removing records of outliers

A: When we remove datapoints From Our dataset, all the parameters of The distribution are re-calculated.

For 476 records, Again, it's calculating  $(Q_1, Q_3, IQR)$

for 506

$Q_1 = 4.77$ ,  $Q_1$  = New value (again)  
 $Q_3 = 7.730$ ,  $Q_3$  = New value (again)

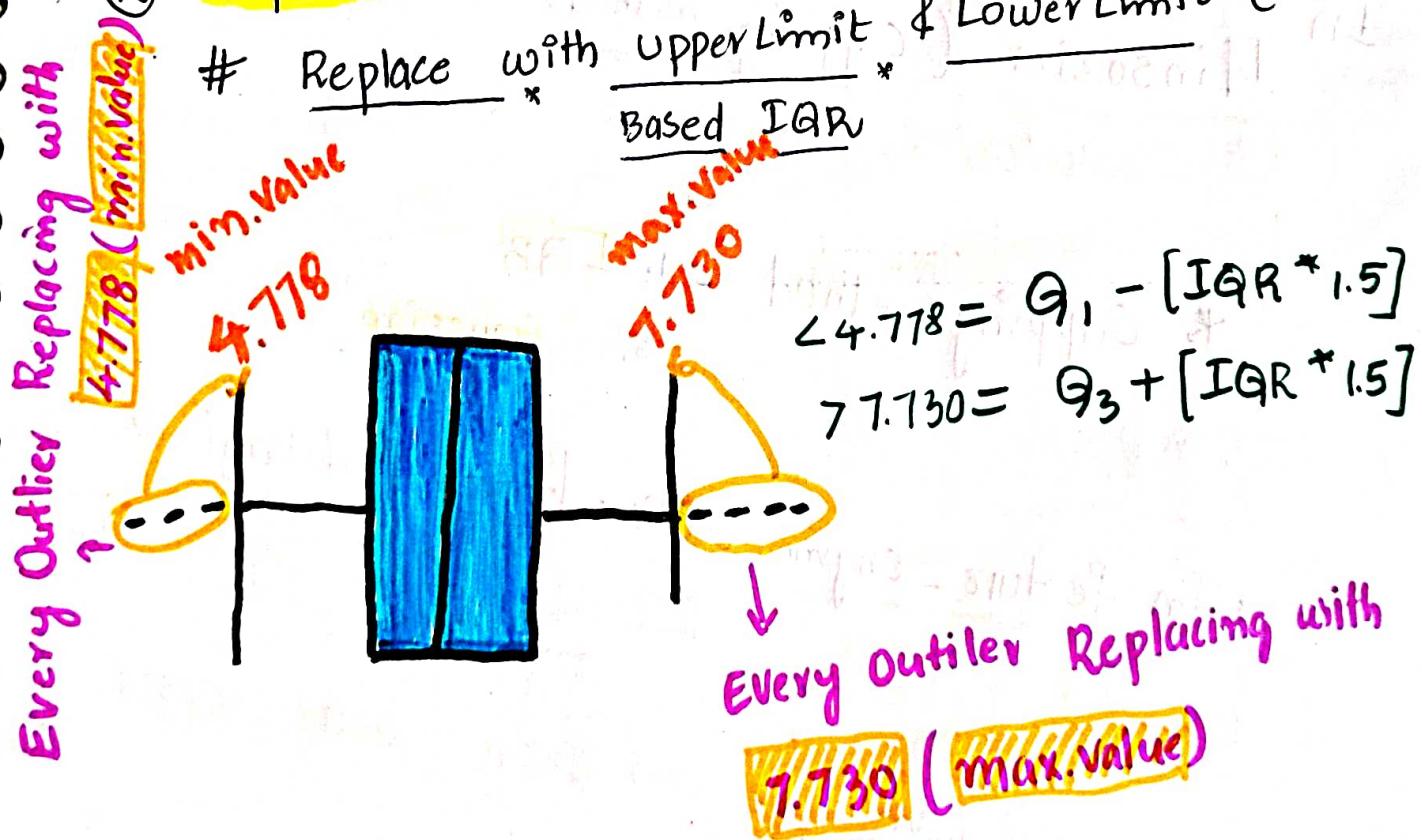
\* Therefore, In The new-trimmed variable, Values

that before were not considered Outliers.

That's why, it showing new Outliers after removing some records also.

option ②. Replace

# Replace with Upper Limit & Lower Limit (calculated Based IQR)



⇒ More Detailness

Replacing with  $Q_1, Q_3$  Values

| CRIM | ...                                                                           | RM                               | ...                              | ... <del>RM</del> <del>sose</del> |
|------|-------------------------------------------------------------------------------|----------------------------------|----------------------------------|-----------------------------------|
| 97   | More Than<br>7.730, so, They<br>are Outliers.<br>Reduce with<br>$Q_3 = 7.730$ | 8.069<br>7.820<br>7.902<br>8.375 | 7.730<br>7.730<br>7.730<br>7.730 |                                   |
| 98   |                                                                               |                                  |                                  |                                   |
| 162  |                                                                               |                                  |                                  |                                   |
| 163  |                                                                               |                                  |                                  |                                   |

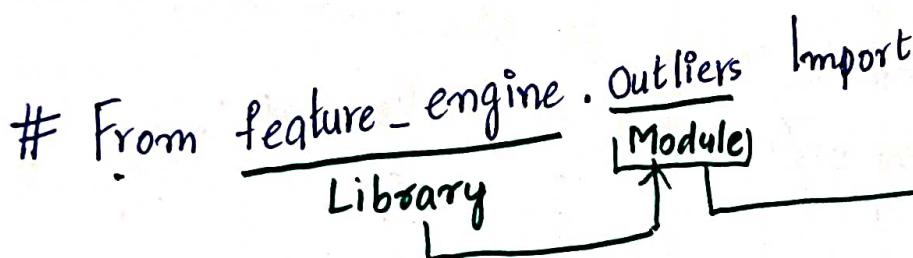
$$Q_3 - IQR * 1.5 < [4.778] \rightarrow \begin{array}{l} \text{less than 4.778} \\ \text{= outlier} \end{array}$$

$$Q_1 + IQR * 1.5 > [7.730] \rightarrow \begin{array}{l} \text{greater than 7.730} \\ \text{= outlier} \end{array}$$

\* **Winsorizer**

# PIP install

**feature-engine**



**Winsorizer**

Winsorizer  
Function

In Winsorizer (capping method)  
Function argument

- \* Capping method = 1. IQR
- 2. "Gaussian"

# In feature-engine . Outliers Import ArbitraryOutlierCapper

```

graph LR
    FE[feature-engine] --> Outliers[Outliers]
    Outliers --> Import[Import]
    subgraph Library [Library]
        Outliers
        Import
    end
    Outliers --> AOC[ArbitraryOutlierCapper]
    AOC --> Function[Function]
    
```

We can give our own Lower, upper values.

1. By IGP method

## In Depth of

Firstly,

# we calculate  $G_1$  value

```
# Q1 = boston["RM"].quantile(0.25)
```

506  
Records

$$\therefore Q_1 = 5.885$$

$$Q_3 = 6.623$$

Out :- 5.855 calculate  $Q_3$  value.

Out: # we calculate  $G_3$  value.  
#  $G_3 = \text{quantile}(\text{boston}["RM"], 0.75)$

# 93  
out : 6.623

$$IQR = Q_3 - Q_1 \\ = 0.737$$

Distance between  $Q_3$  and  $Q_1$   $\rightarrow$

$$\# IGR = Q_3 - Q_1$$

IGR

out

# Winf = Wimsozier

```
# Wint = Winsorizer (capping-  
↑ store in  
# boston_t = Wint.fit_transform (boston[["RM"]])  
↑ store in  
original data
```

# boston\_ ← store in win. right-tail-caps\_)  
# Print (win. left-tail-caps\_, win. right-tail-caps\_)  
# 4.7184 ← 7.7302

# Sns. boxplot (boston\_t . RM)

plt.title ("boxplot")

# plt.show()

7.730E-12 AM Edited (replaced IGR) RM

again

From feature\_engine.Outliers import Winsorizer

win = Winsorizer (capping\_method="lqr", fold=1.5, Tail="both")

Variables = ["RM"]

boston\_t = win.fit\_transform(boston[["RM"]])

Print (win.left\_tail\_caps\_, win.right\_tail\_caps\_)

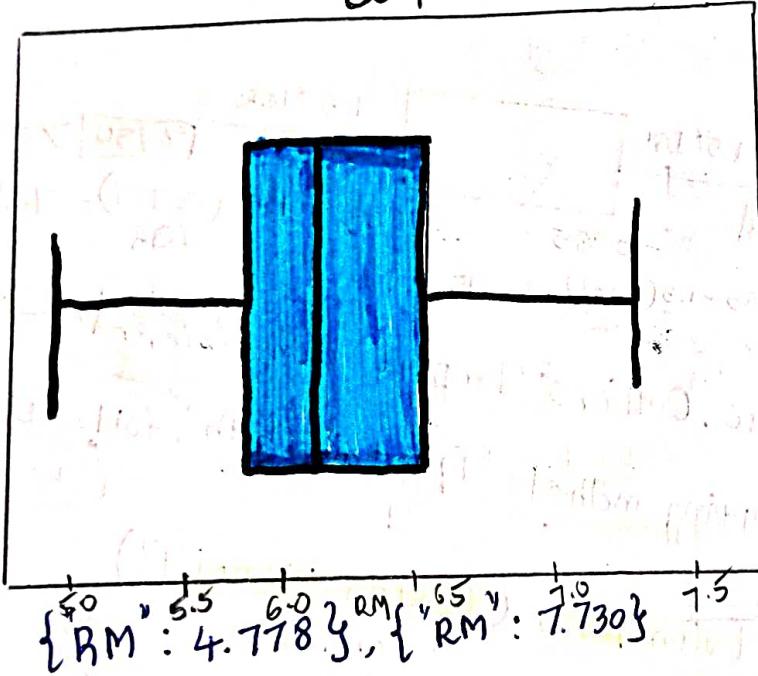
Sns.boxplot (boston\_t.RM)

plt.title ("boxplot")

plt.show()

Boxplot

Out



2, Replacing outliers using Winsorizer (min, max, automatically taken by "Gaussian method")

# Gaussian method

= "Normal"

Same

Distribution method



```
# from feature_engine.outliers import Winsorizer
```

```
# Wim = Winsorizer(capping_method = "gaussian", tail = "both")
```

```
fold = 1.5, variables = ['RM'])
```

```
boston_t = Wim.fit_transform(boston[['RM']])
```

```
print(Wim.left_tail_caps_, Wim.right_tail_caps_)
```

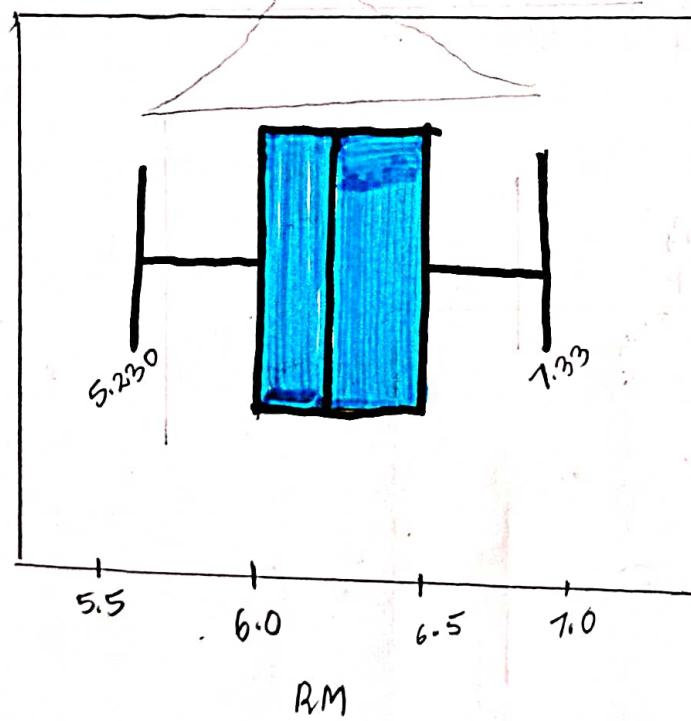
```
sns.boxplot(boston_t.RM)
```

```
plt.title("Boxplot")
```

```
plt.show()
```

```
{'RM': 5.230} {'RM': 7.33}
```

Out



3. Replace arbitrary Outlier Capper (The min, max, by user)

# we get the **min** and **max** values based on "Domain Expert" (or) by **Our Own Research**

# from `feature_engine.Outliers import ArbitraryOutlierCapper`

`Capper = ArbitraryOutlierCapper (max_capping_dict = { "RM": 7.5}, min_capping_dict = { "RM": 4.8})`

`boston_c = Capper.fit_transform(boston[['RM']])`

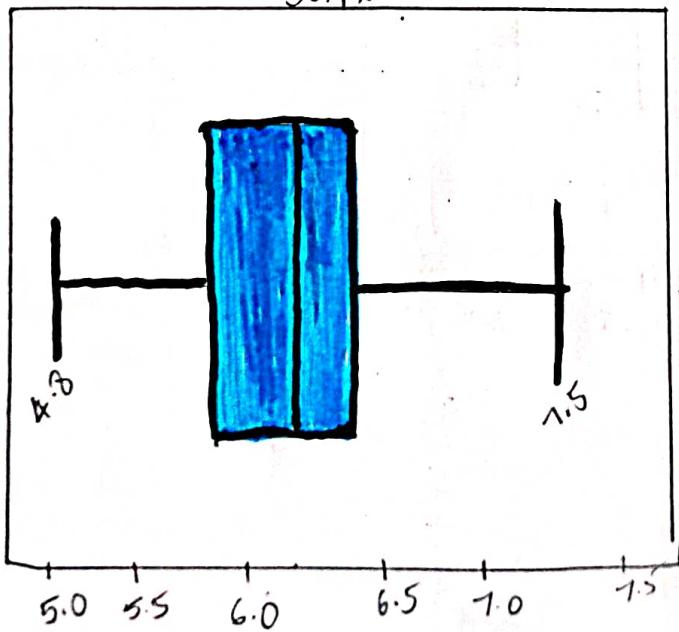
`print(Capper.right_tail_caps_, Capper.left_tail_caps_)`

`sns.boxplot(boston_c.RM)`

`plt.title("Boxplot")`

`plt.show()`

`{ "RM": 4.8}, { "RM": 7.5}`



3/4/2022 5:20 PM

5/4/22  
10:30pm

3

## Discretization

Converting Continuous Data To Discrete Data

also called as "Binning"

code :-

Import Pandas as pd

```
# stroke = pd.read_csv("stroke prediction.csv")  
# stroke.head()  
https://t.me/AIMLDeepThaught/612
```

Out

| id    | gender | age  | hyperTension | heart-disease | Ever_married | work_type    | Residence_Type | avg-gluco_Level | bmi  | smok_Status     | stroke |
|-------|--------|------|--------------|---------------|--------------|--------------|----------------|-----------------|------|-----------------|--------|
| 30669 | Male   | 3.0  | 0            | 0             | No           | children     | Rural          | 95.12           | 18.0 | NaN             | 0      |
| 30468 | Male   | 58.0 | 1            | 0             | Yes          | private      | urban          | 87.96           | 39.2 | Never           | 0      |
| 16523 | Female | 8.0  | 0            | 0             | No           | private      | urban          | 110.89          | 17.6 | NAN             | 0      |
| 56543 | Female | 70.0 | 0            | 0             | Yes          | private      | Rural          | 69.04           | 35.9 | Formerly smoked | 0      |
| 46136 | male   | 14.0 | 0            | 0             | No           | Never_Worked | Rural          | 161.28          | 19.1 | NAN             | 0      |

discrete (cat)      continuous      discrete (cont.)      discrete (cont.)      discrete      discrete  
↓                    ↓                    ↓                    ↓                    ↓                    ↓

# stroke.shape

Out : (43400, 12)

# stroke.info()

information about stroke data.

Out

⇒ Create Bins

$\# \text{ intervals} = [0, 12, 19, 30, 60, 90] \rightarrow \# \text{ bins} \Rightarrow$  Each interval

# Intervals = 5  
# Categories = ["child", "teenager", "young adult", "middle aged", "senior citizen"]

• Creating New Column & Storing The data of  
"Age" Divide.

# Størke ["Age-category"] = Pd. cut [x =

$\downarrow$   
 $X = \text{Stroke ['Age']}$ ,  $\text{bins} = \text{intervals}$ ,  
 $\text{Labels} = \text{categories}$ ]

Age column is  
Continuous Data. So,  
We converting into  
Discrete Variable  
intervals categorized

- \* 0-12 → child
- \* 12-19 → teenager
- \* 19-30 → young adult
- \* 30-60 → Middle aged
- \* 60-90 → Senior citizen



# stroke.head()

## Changed Continuous Data

Discrete Categorical data

## Newly-ads.

## 4

## \* Encoding \*

Applicable For Categorical Variables

Discrete

\* Machine Can't Understand Text Data. So, We Convert the "discrete categorical" to "discrete count" Data

\* There are two types of Categorical Data

1. Nominal

(No Natural Sequence)

2. Ordinal

(Natural Sequence)

# we should treat  
Every variable  
Equally.

→ To Convert Categorical Data

To Numeric:

# Ordinal Data

| Grades | Score |
|--------|-------|
| O      | 10    |
| At     | 9     |
| A      | 8     |
| B+     | 7     |
| B      | 6     |
| F      | 0     |

→ O > A >  
10 > 9

| Citynames | Number |
|-----------|--------|
| Hyd       | 0      |
| Delhi     | 1      |
| Mumba     | 2      |
| Kerala    | 3      |
| Gujarat   | 4      |

In this case, we can't  
decide by numbering  
given to state.

# Nominal Data → (dummies)

which is  
greater

1. get Dummies (Pandas)

2. One hot Encoding (sklearn)

# Ordinal Data

1. map (pandas)

2. Label Encoder (sklearn)

| # One Hot Encoding | # Label Encoding |
|--------------------|------------------|
| * Nominal          | * Ordinal        |

```

import Pandas as pd
import Numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
# df = pd.read_csv("homeprices.csv")
# df.head()

```

**Out** :-

|   | Town     | Area | Price   |
|---|----------|------|---------|
| 0 | Chennai  | 2600 | 5500000 |
| 1 | Chennai  | 3000 | 5650000 |
| 2 | Chennai  | 3200 | 6100000 |
| 3 | Chennai  | 3600 | 6800000 |
| 4 | Banglore | 2600 | 5850000 |

# Town

- 1. Chennai
- 2. Bangalore
- 3. Hyderabad

Nominal Data → Categorical Data

→ **Pd. get\_dummies** [ Nominal Variable Encoding using pandas ]

# dummies = Pd. get\_dummies (df[ "Town" ])

# dummies

**Out** :-

|    | Banglore | Chennai | Hyderabad |
|----|----------|---------|-----------|
| 0  | 0        | 1       | 0         |
| 1  | 0        | 1       | 0         |
| 2  | 0        | 1       | 0         |
| 3  | 0        | 1       | 0         |
| 4  | 1        | 0       | 0         |
| 5  | 2        | 0       | 0         |
| 6  | 1        | 0       | 0         |
| 7  | 1        | 0       | 0         |
| 8  | 0        | 0       | 0         |
| 9  | 0        | 0       | 0         |
| 10 | 0        | 0       | 1         |

Out :-

|    | Banglore | Chennai | Hyderabad |
|----|----------|---------|-----------|
| 0  | 0        | 1       | 0         |
| 1  | 0        | 1       | 0         |
| 2  | 0        | 1       | 0         |
| 3  | 0        | 1       | 0         |
| 4  | 1        | 0       | 0         |
| 5  | 1        | 0       | 0         |
| 6  | 1        | 0       | 0         |
| 7  | 0        | 0       | 1         |
| 8  | 0        | 0       | 1         |
| 9  | 0        | 0       | 1         |
| 10 | 0        | 0       | 1         |
| 11 | 0        | 0       | 1         |

Dummies

# which ever record is there  
it shows = 1

\* For other columns it  
shows = 0

→ Adding → original → converted one  
# df\_dummies = pd.concat ([df, dummies], axis = "columns")  
by

df\_dummies <https://t.me/AIMLDeepThaught/612>

Out :-

|    | Town      | Area | Price   | Banglore | Chennai | Hyderabad |
|----|-----------|------|---------|----------|---------|-----------|
| 0  | Chennai   | 2600 | 5500000 | 0        | 1       | 0         |
| 1  | Chennai   | 3000 | 5650000 | 0        | 1       | 0         |
| 2  | Chennai   | 3200 | 6100000 | 0        | 1       | 0         |
| 3  | Chennai   | 3600 | 6800000 | 0        | 1       | 0         |
| 4  | Banglore  | 3600 | 5850000 | 1        | 0       | 0         |
| 5  | Banglore  | 2600 | 6150000 | 1        | 0       | 0         |
| 6  | Banglore  | 2800 |         | 1        | 0       | 0         |
| 7  | Banglore  | 3300 | 6500000 | 1        | 0       | 0         |
| 8  | Hyderabad | 3600 | 6100000 | 0        | 0       | 1         |
| 9  | Hyderabad | 2600 | 7100000 | 0        | 0       | 1         |
| 10 | Hyderabad | 2900 | 5750000 | 0        | 0       | 1         |
| 11 | Hyderabad | 3100 | 6000000 | 0        | 0       | 1         |

# Delete The Original column

# df\_dummies . drop ("town", axis = "columns", inplace = True)

df\_dummies

Out:

|    | Area | Price   | Banglore | chennai | Hyderabad |
|----|------|---------|----------|---------|-----------|
| 0  | 2600 | 5500000 | 0        | 1       | 0         |
| 1  | 3000 | 5650000 | 0        | 1       | 0         |
| 2  | 3200 | 6100000 | 0        | 1       | 0         |
| 3  | 3800 | 6800000 | 0        | 1       | 0         |
| 4  | 2600 | 5850000 | 1        | 0       | 0         |
| 5  | 2800 | 6150000 | 1        | 0       | 0         |
| 6  | 3300 | 7100000 | 1        | 0       | 0         |
| 7  | 3600 | 5750000 | 1        | 0       | 0         |
| 8  | 2600 | 6000000 | 0        | 0       | 1         |
| 9  | 2900 | 6200000 | 0        | 0       | 1         |
| 10 | 3100 | 6900000 | 0        | 0       | 1         |
| 11 | 3600 | 6500000 | 0        | 0       | 1         |

# after deleting original column in New Data (i.e. df\_dummies)

Old\_data (df) is still remains same.



Dummy Variable Trap



|   | Bang | chenn | Hyder |
|---|------|-------|-------|
| 0 | 1    | 0     | 0     |
| 0 | 1    | 0     | 0     |
| 1 | 0    | 0     | 0     |
| 1 | 0    | 0     | 0     |
| 0 | 0    | 1     | 0     |
| 0 | 0    | 1     | 0     |



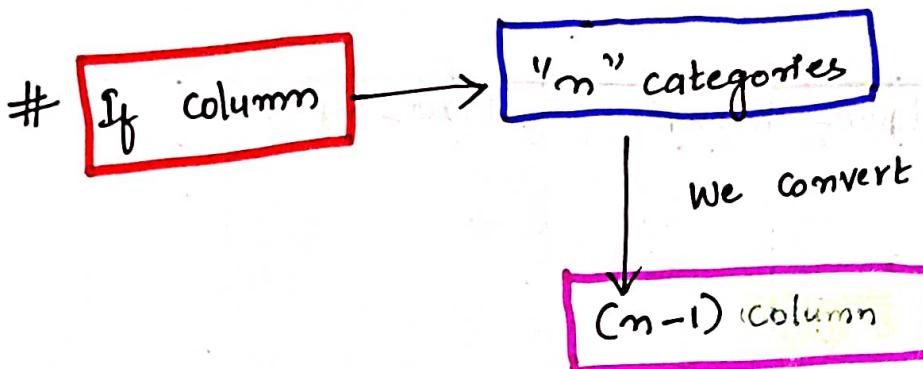
|   | Bang | chenn |
|---|------|-------|
| 0 | 1    | 0     |
| 0 | 1    | 0     |
| 1 | 0    | 0     |
| 1 | 0    | 0     |
| 0 | 0    | 0     |
| 0 | 0    | 0     |

→ hyderabad

If we Remove One of The column also, still we can identify, col name by (0,0)

# Town - 3 categories

↓ Dummies  
3 columns [ Multi-collinearity / collation problems ]



# df\_dummies.drop("chennai", axis="columns", inplace=True)

# df\_dummies

out

| Area | Price | Bangalore | Hyderabad |
|------|-------|-----------|-----------|
| 0    | 2600  | 5500000   | 0         |
| 1    | -     | -         | 0         |
| 2    | -     | -         | 0         |
| 3    | -     | -         | 0         |
| 4    | -     | -         | 0         |
| 5    | -     | -         | 0         |
| 6    | -     | -         | 0         |
| 7    | -     | -         | 0         |
| 8    | -     | -         | 0         |
| 9    | -     | -         | 0         |
| 10   | -     | -         | 0         |
| 11   | 36000 | 6500000   | 0         |

# where ever  
Both column  
having equal  
zero's, it is  
another column's  
data.

chennai = 0 0

# We Can Write **One line** of Code To do all these

# df\_dum = pd.get\_dummies (df, drop\_first=True)

# df\_dum

| out : | Area | Price | town-chennai | town-hyderabad |
|-------|------|-------|--------------|----------------|
| 0     |      |       | 1            | 0              |
| 1     |      |       | 1            | 0              |
| 2     |      |       | 1            | 0              |
| 10    |      |       | 1            | 0              |
| 11    |      |       | 1            | 0              |

# This Diagram is Exact of (Previous) last page diagram. But deleted banglore Replaced chennai  $\rightarrow$  town-banglore [deleted]  $\rightarrow$  Because, B  $\checkmark$  (Alphabet order) replaced with  $(0 0)$   $\leftarrow$  chennai  $\leftarrow$  hyd

⇒ **One Hot Encoding**  $\rightarrow$  **Nominal using sklearn**

From sklearn.preprocessing import OneHotEncoder

# enc = OneHotEncoder (drop = "First")  
↑  
Stores in enc

# enc\_df = pd.DataFrame (enc.fit\_transform (df[["town"]]).toarray())  
↑  
Converting  
↓  
Output

# enc-df <https://t.me/AIMLDeepThaught/612>

Out

|    | 0   | 1   |
|----|-----|-----|
| 0  | 1.0 | 0.0 |
| 1  | 1.0 | 0.0 |
| 2  | 1.0 | 0.0 |
| 3  | 1.0 | 0.0 |
| 4  | 0.0 | 0.0 |
| 5  | 0.0 | 0.0 |
| 6  | 0.0 | 0.0 |
| 7  | 0.0 | 0.0 |
| 8  | 0.0 | 1.0 |
| 9  | 0.0 | 1.0 |
| 10 | 0.0 | 1.0 |
| 11 | 0.0 | 1.0 |

# merge with main df

# df\_ohe = df.join(enc-df)

# df\_ohe.drop("town", axis="columns", inplace=True)

# df\_ohe

Out

|    | area | Price   | 0   | 1   |
|----|------|---------|-----|-----|
| 0  | 2600 | 5500000 | 1.0 | 0.0 |
| 1  | 3000 | 5650000 | 1.0 | 0.0 |
| 2  | -    | -       | 1.0 | 0.0 |
| 3  | -    | -       | 1.0 | 0.0 |
| 4  | -    | -       | 0.0 | 0.0 |
| 5  | -    | -       | 0.0 | 0.0 |
| 6  | -    | -       | 0.0 | 0.0 |
| 7  | -    | -       | 0.0 | 0.0 |
| 8  | -    | -       | 0.0 | 0.0 |
| 9  | -    | -       | 0.0 | 1.0 |
| 10 | -    | -       | 0.0 | 1.0 |
| 11 | -    | -       | 0.0 | 1.0 |

## Alphabetical Order

⇒ **Label Encoder** → **Ordinal Variable** [sklearn]

# Used for binary category Variable

# `dfnew = df.copy()` → creating "copy" from original Data.

# `dfnew`

**Out**

|    | town      | Area | Price    |
|----|-----------|------|----------|
| 0  | Chennai   | 2600 | 5500000  |
| 1  | Chennai   | 3000 | 5650000  |
| 2  | Chennai   | 3250 | 61000000 |
| ⋮  | ⋮         | ⋮    | ⋮        |
| 11 | Hyderabad | 3600 | 6950000  |

From `sklearn.preprocessing` import `LabelEncoder`

# `Le = LabelEncoder()`

# `dfnew.town = Le.fit_transform(dfnew.town)`

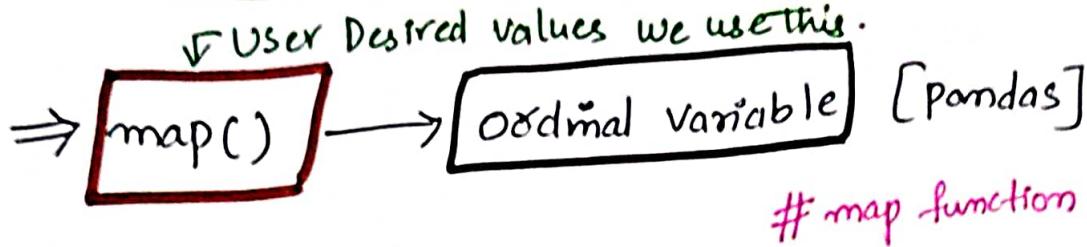
# `dfnew`

**Out** :

|    | town | Area | Price    |
|----|------|------|----------|
| 0  | 1    | 6.0  | 5500000  |
| 1  | 1    | 6.0  | 5650000  |
| 2  | 1    | 6.0  | 61000000 |
| 3  | 1    | 6.0  | 6950000  |
| 4  | 0    | 6.0  | 5500000  |
| 5  | 0    | 6.0  | 5650000  |
| 6  | 0    | 6.0  | 61000000 |
| 7  | 0    | 6.0  | 6950000  |
| 8  | 2    | 6.0  | 5500000  |
| 9  | 2    | 6.0  | 5650000  |
| 10 | 2    | 6.0  | 61000000 |
| 11 | 2    | 6.0  | 6950000  |

Alphabetical Order

↓  
Banglore [B] — 0  
Chennai [C] — 1  
Hyderabad [H] — 2



# df\_m = df.copy()

# df\_m ["town"] = df\_m ["town"]. map ({ "Chennai":0, "Bangalore":2, "Hyderabad":1})

# df\_m

**Out** :-

|    | town | Area | Price |
|----|------|------|-------|
| 0  | 0    |      |       |
| 1  | 0    |      |       |
| 2  | 0    |      |       |
| 3  | 0    |      |       |
| 4  | 2    |      |       |
| 5  | 2    |      |       |
| 6  | 2    |      |       |
| 7  | 2    |      |       |
| 8  | 1    |      |       |
| 9  | 1    |      |       |
| 10 | 1    |      |       |

Which ever order we write it takes in the order

# Taken by our choices

Here

Chennai = 0, 20

Bangalore = 2, 15

Hyderabad = 1, 25

# We can replace with any value

<https://t.me/AIMLDeepThaught/612>



From `sklearn.preprocessing import OrdinalEncoder`

# df\_new1 = df.copy()

# oe = OrdinalEncoder (categories = [[ "Bangalore", "Hyderabad", "Chennai" ]])

# df\_new1["town"] = oe.fit\_transform (df\_new1[["town"]])

# df\_new1

**Out** :-

|        | Town | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|--------|------|---|---|---|---|---|---|---|---|---|---|
| 0 - 20 | 2    | 2 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

inf/ 04/22  
3:00 AM

Dt: 3/4/22  
11:30pm

"1"

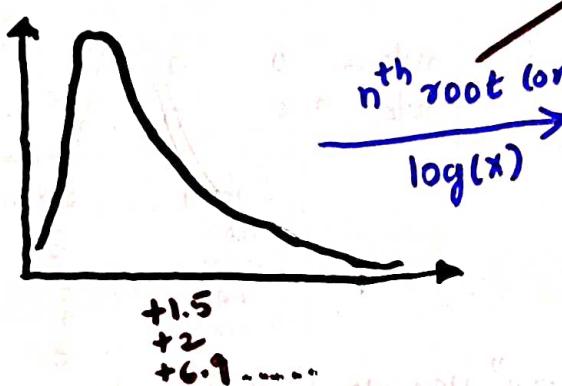
Feature Transformation

Only For  
Continuous  
Data

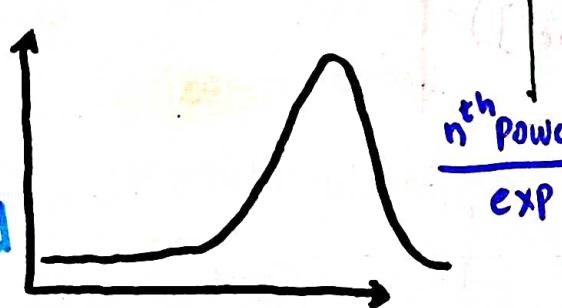
Feature processing : Transformation.

columns / variables / features (same)

Right  
Skewed



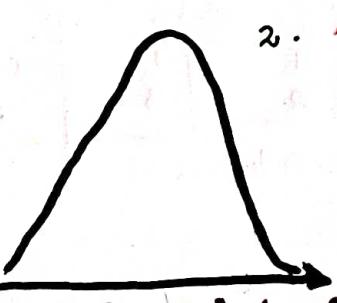
Left  
Skewed



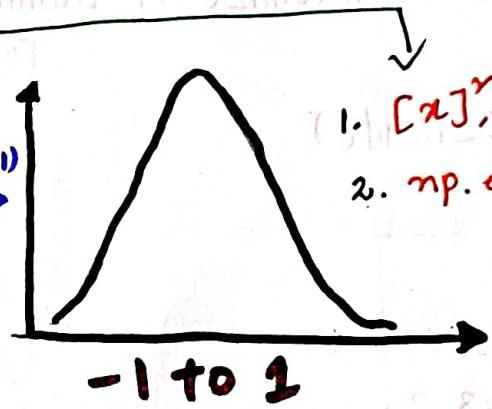
|     |    |
|-----|----|
| Ex: | x  |
|     | 2  |
|     | 4  |
|     | 9  |
|     | 12 |
|     | 25 |

$$x^{0.5}, \sqrt{x}, x^{1/3}, x^{1/4}$$

1.  $[x]^{1/2}, [x]^{1/3}, [x]^{1/4}$
2. np.log



-1 to +1 (normal)  
Distribution



1.  $[x]^r, [x]^3, [x]^4$
2. np.exp

# import Libraries / packages.

# import numpy as np

# import pandas as pd

# import matplotlib.pyplot as plt

# import matplotlib inline

# %matplotlib inline

# import seaborn as sns

1. Root Transformations
2. log Transformations
3. Reciprocal Transformations
4. Boxcox Transformations

```
# titanic = pd.read_csv("titanic-csv")  
# titanic = pd.read_csv("titanic-csv")  
# titanic = pd.read_csv(location of csv)
```

titanic

| Passenger Id | Survived | Pclass | Name        | Sex    | Age  | SibSp | Parch | Ticket  | Fare    |
|--------------|----------|--------|-------------|--------|------|-------|-------|---------|---------|
| 1            | 0        | 3      | Owen Harris | male   | 22.0 | 1     | 0     | A/52771 | 7.2500  |
| 2            | 1        | 1      | cumming     | Female | 38.0 | 1     | 0     |         | 71.2833 |
| 3            | 1        | 3      | Larina      | Female | 26.0 | 0     | 0     |         | 7.9250  |
| 4            | 1        | 1      | Futrelle    | Female | 35.0 | 1     | 0     |         | 53.1000 |
| 889          | 890      | 1      | Behr        | male   | 26.0 | 0     | 0     |         | 30,0000 |
| 890          | 891      | 0      | Doolay      | male   | 32.0 | 0     | 0     |         | 7.7500  |

891 rows x 12 columns.

| Cabin | Embarked |
|-------|----------|
| NAN   | S        |
| C85   | C        |
| NAN   | S        |
| C123  | S        |
| :     |          |
| Q48   | C        |

```
# titanic = pd.read_csv("titanic.csv", usecols=[  
    "Fare"])
```

titanic\_head()

out

| Fare |         |
|------|---------|
| 0    | 7.2500  |
| 1    | 71.2833 |
| 2    | 7.9250  |
| 3    | 53.1000 |
| 4    | 8.0500  |

particular column

usecols = "Fare"  
col. Name.

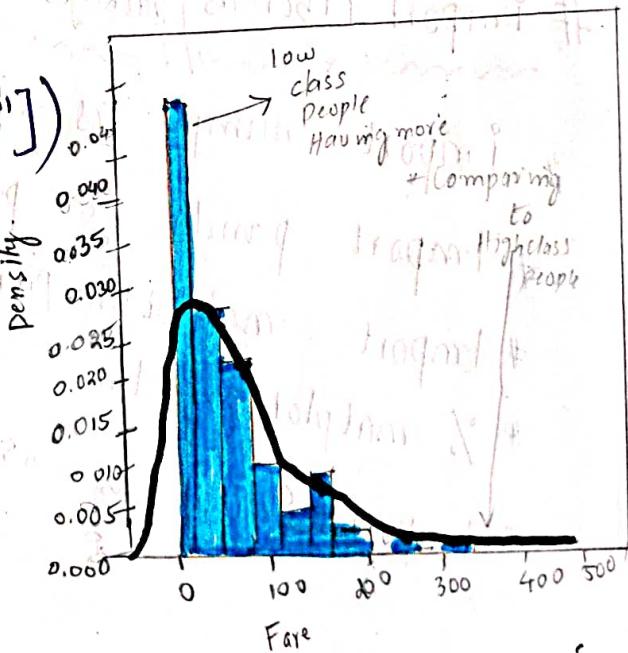
```
# sns.distplot(titanic["Fare"])
```

plt. show()

```
titanic[["Fare"]].skew()
```

Out ')

+ 4.7873



# another way

```
t - ["Fare"].hist()  
plt.show()
```

```
titanic["Fare"].skew()
```

Out :- +4.7873

Values

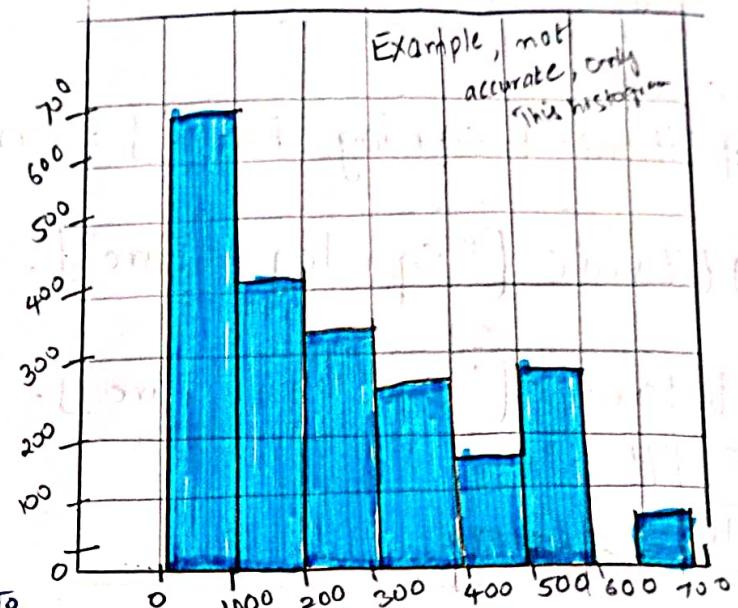
$$\text{Skew} = +0.085 (2^{\text{th}} \text{ root})$$

$$\text{Skew} = +0.5 (4^{\text{th}} \text{ root})$$

$$\text{Skew} = +0.21 (5^{\text{th}} \text{ root})$$

$$\text{Skew} = -0.95 (6^{\text{th}} \text{ root})$$

consider,  
This  
which is  
close to  
"0".



\* ROOT Transformation # For Right Skewed

```
# titanic["Sqr-Fare"] = titanic["Fare"] ** (1/2)
```

```
# titanic["Sqr-Fare"].skew()
```

```
# titanic["Sqr-Fare"].hist()
```

Out :- +2.085 More Than +1  
or -1

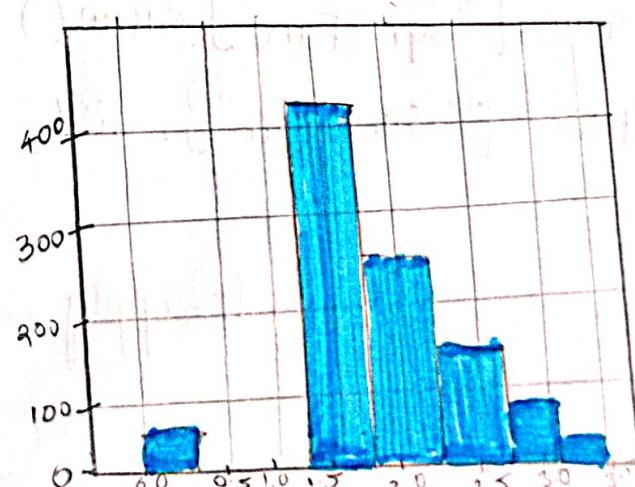
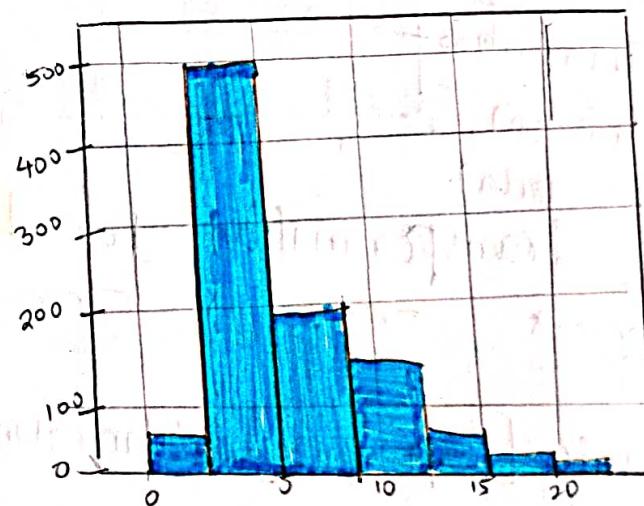
\* Try with  $\sqrt[5]{}, \sqrt[6]{}, \dots$

```
# titanic["Sqr-Fare"] = titanic["Fare"] ** (1/5)
```

```
# titanic["Sqr-Fare"].hist()
```

```
# titanic["Sqr-Fare"].skew()
```

Out :- -0.212676 Less Than  
-1 to +1  
Final



## → Log Transformation

Q. Is This mandatory, To write Every time?  
 - No, Only if we "0" in The Data.

#titanic[["sqr log - "Fare"]] =  $n \cdot \log(titanic["Fare"] + 0.01)$

#titanic[["sqr log - "Fare"]].skew() Why?

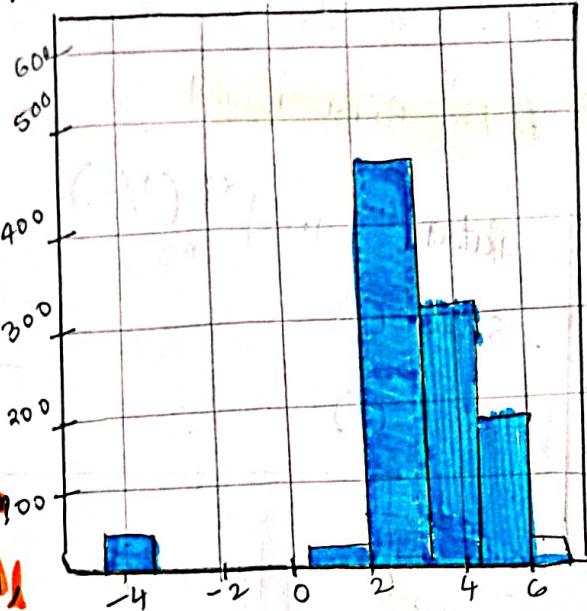
#titanic[["sqr log - "Fare"]].hist()

# plt.show()

Out

-2.41004

Here, The Value, in This Have more, So we don't use log Transformation For This (0) Data)



Because In Data "Fare" column, we have some records "0" values. So,

If we calculate with log. it give a (infinity) value, so, we add +0.01 to "0" column.

we use "n" power

Root Transformation For (left skewed)

fare data is not left skewed, But To understand, Left skewed

titanic[["sqr-Fare"]] = titanic[["Fare"]<sup>\*\*</sup>(2)]

it can be any value

titanic[["sqr-Fare"]].skew()

(2) (3) (4) ...

titanic[["Sqr-Fare"]].hist()

\* When The Data is left

Skewed.

Out :

\* We apply  $n^{\text{th}} \text{ power}$  To The Data (or) column.

## \* Exponential

`titanic["sqr-exp Fare"] = np.exp(titanic["Fare"])`

`titanic["sqr-exp Fare"].skew()`

`titanic["sqr-exp Fare"].hist()`

`plt.show`

another method  
For Left Skewed  
Distribution

\* Evaluates  $e^x$  For Each  
Element in The given  
Input.

## \* Reciprocal Transformation

`titanic["Rec-Fare"] = 1 / [titanic["Fare"] + 0.01]`

`titanic["Rec-Fare"].skew()`

`titanic["Rec-Fare"].hist()`

`plt.show()`

"Reciprocal" is that  
we divide Every Value  
with  $(1) / [\text{column}] + 0.01$   
of The data to Eliminate  
"0" value In The Data.

## \* Box Cox Transformation

| Fare  |
|-------|
| 7250  |
| 71283 |
| -     |
| -     |
| -     |
| -     |

$$\rightarrow \frac{7250 - 1}{71283 - 1}$$

$$T[x] = \frac{x^\lambda - 1}{\lambda}$$

$\therefore \lambda$  [lambda] Varies From -5 to +

$\therefore$  Where  $x$  is The response Variable and  
 $\Rightarrow$  In This Transformation, all Values of  
 $\lambda$  are Considered and The optimal  
Value for a given Variables are Selected.

from help

we see two return values

See [two](#)

1. Array values ( $\max \log$ )
2. optimal Lamda parameter ( $\max \log$ )

which ever gives best result

## # Fitting data, Fitting Lambda

# titanic ["Fare\_boxcox"], param = stats.boxcox  
(titanic ["Fare"] + 1) out :-  $a, b = \overline{1}, \overline{1}$   
stores

# print ("λ=", param) (or) param

$$\boxed{\text{Out}} \therefore -0.09$$

```
# titanic [ "Fare_boxcox" ]. skew()
```

$$\boxed{\text{out}} := \underline{-0.04}$$

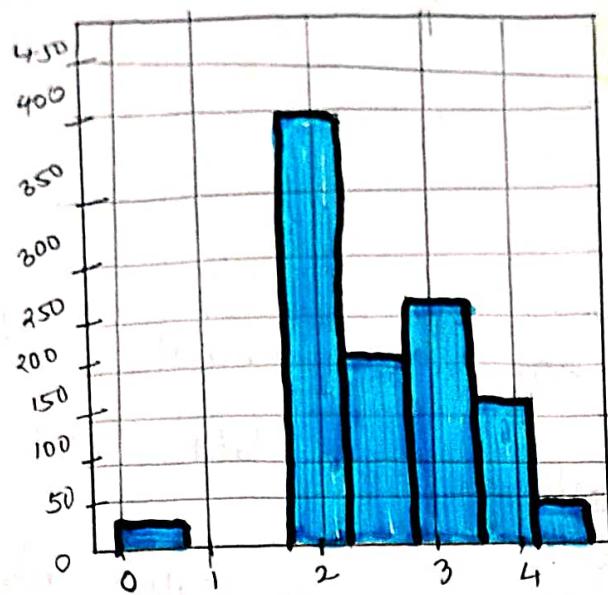
## Code

```
# titanic[ "Fare - boxcox" ], param = stats. boxcox (titanic. Fare + 1)
```

```
# titanic["Fare_boxcox"].hist()
```

```
# titanic [ "Fare_boxcox" ] . skew()
```

out :- -0.04



graph of water level

graph of water level

~~inf~~  
4/4/22

5:30pm

5/4/22  
11:00 AM

2  $\Rightarrow$  Feature Scaling  $\Rightarrow$  Applicable only for **Continuous data**

Feature Scaling ?

Ans:- it refers (or) Techniques used to **normalize** the ranges in our data. (or)

of **Independent Variables**  
**input variables**

\* The methods to set the features value range within a **similar scale**.

\* Variables with bigger magnitude / larger value range dominate over those with smaller magnitude / value range.

Ex:- 10,000,000  
if we take  
10  
10

Both are equal important range.

\* Scale of the features is an important consideration when building **Machine Learning models**.

\* Feature scaling is generally the **last step** in the data **pre processing pipeline**, performed just before **Training the machine learning algorithms**.

5. Gradient descent **converges faster** when **Features** are on **similar scales**

Example :-

| C.C  | Mileage |
|------|---------|
| 1600 | 14      |
| 1800 | 15      |
| 2200 | 16.5    |
| 2100 | 18      |
| 2600 | 22.5    |
| 1800 | 19.4    |
| 1900 | 25.4    |

Continuous Data

\* Feature Scaling \*

of a car

a. When we ask Machine which is important in the both columns?

\* Machine is giving importance to C.C

Because it is having large values.

\* But, we know both are important

So, we have train's model in <sup>Initially</sup> such that,

it should consider both the columns similar

So, we reduce the value

$$= \frac{800}{1000} = \frac{80}{100} = \frac{8}{10} = \frac{4}{5} \quad (\text{Every one is same})$$

We are scaling down

Dividing every value with "10" in this case.

\* To reduce the value.

\* Feature Scaling

Importance in some ML Algorithms

## 1. Linear Regression & logistic Regression

\* The regression coefficients of linear models are directly influenced by scale of the variable.

## 2. Support Vector Machines :-

\* Feature scaling helps decrease the time to find support vectors for SVM's

## 3. K-means clustering

\* Euclidean distances are sensitive to feature magnitude

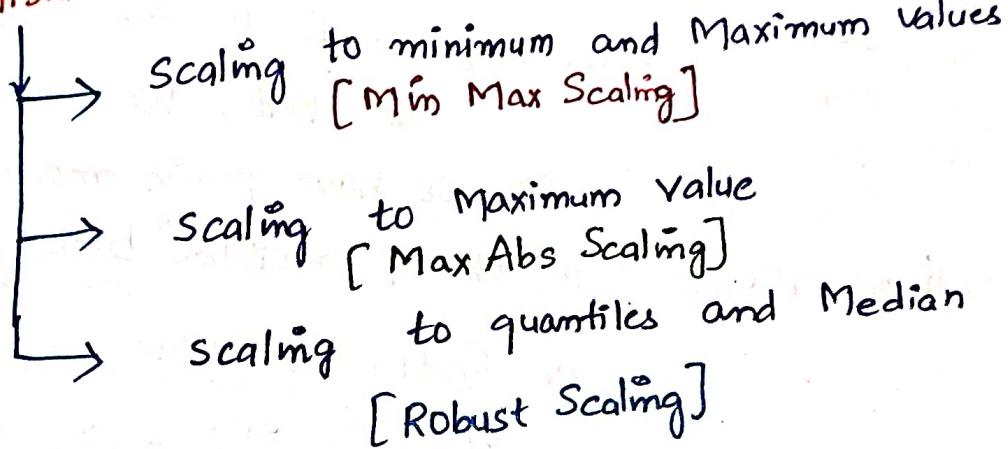
## 4. Principal Component analysis (PCA) :-

\* PCA requires the features to be centered at "0".

## \* Various Feature Scaling Techniques :-

1. Standardisation

2. Normalisation Technique



# import pandas as pd

# import matplotlib.pyplot as plt

1. matplotlib inline.

→ Using (titanic Data Set)

code:  
# titanic = pd.read\_csv("titanic.csv", usecols = ["Age"])

titanic.head()

out

| Age  |
|------|
| 22.0 |
| 38.0 |
| 26.0 |
| 35.0 |
| 35.0 |

# check Null values in "Age" column.

# titanic . isnull() . sum()

[Out] Age 177 → Null values

# Replacing with "Median Value" For Null values

# titanic ["Age"] . fillna (titanic ["Age"] . median(),  
inplace = True)

# titanic ["Age"] . isnull() . sum()

[Out] : Age 0 → zero Null values

I

\* Standardisation

Converting Each Value To Zscore

Ex :-

| X  | $\frac{x-\mu}{\sigma} = Z$ score |
|----|----------------------------------|
| 23 | $\frac{23-60}{2} = -18.5$        |
| 45 | $\frac{45-60}{2} = -7.5$         |
| 67 | $\frac{67-60}{2} = 3.5$          |
| 89 | $\frac{89-60}{2} = 14.5$         |
| 78 | $\frac{78-60}{2} = 9$            |

These Five values converting To Zscores  
is called as "standard" scaling

Avg / mean = 60.

(std)  $\sigma = \sqrt{\frac{\sum (x - \mu)^2}{n}}$

Code :-

From sklearn.preprocessing import

Standard Scaler (sc)

```
# sc = StandardScaler() # call The Function (short cut)
# main (don't forget)
# titanic["Age-SC"] = sc.fit_transform(titanic[["Age"]]) # fit-transform
# creating new column
# titanic["Age-SC"] # stores in
# titanic[["Age-SC"]]
```

calculation

Converting The value

Out :-

<https://t.me/AIMLDeepThaught/612>

|     | Age-SC  |
|-----|---------|
| 0   | -0.5651 |
| 1   | 0.6638  |
| 2   | -0.2583 |
| 3   | 0.43312 |
| ... |         |
| 890 | 0.20272 |

# Here Every Value is converted To Zscore.

$$\left[ \frac{x - \mu}{\sigma} \right] = Z \text{ score.}$$

EX:- Age [22] # First record  $x = 22$

$$\# \text{titanic}.mean[\frac{x - \mu}{\sigma}] = \left[ \frac{22 - 29.361}{13.01} \right]$$

# titanic

Original Data

"Age-SC"  $\rightarrow$  [-0.5651]

Out

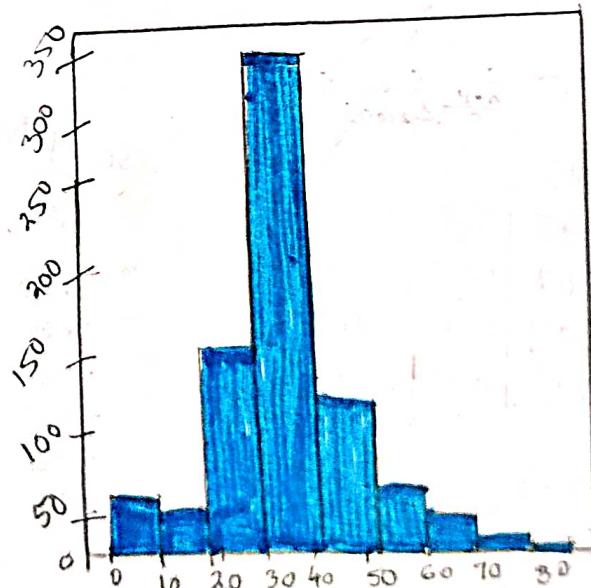
|     | Age  | Age-SC  |
|-----|------|---------|
| 0   | 22.0 | -0.5651 |
| 1   | 38.0 | 0.6638  |
| 2   | 25.0 | -0.2583 |
| ... |      |         |
| 889 | 26.0 | -0.2583 |
| 890 | 32.0 | 0.20272 |

Converted To Z-Score

# histogram of  
Original ["Age"]

# titanic["Age"].hist()

plt. show

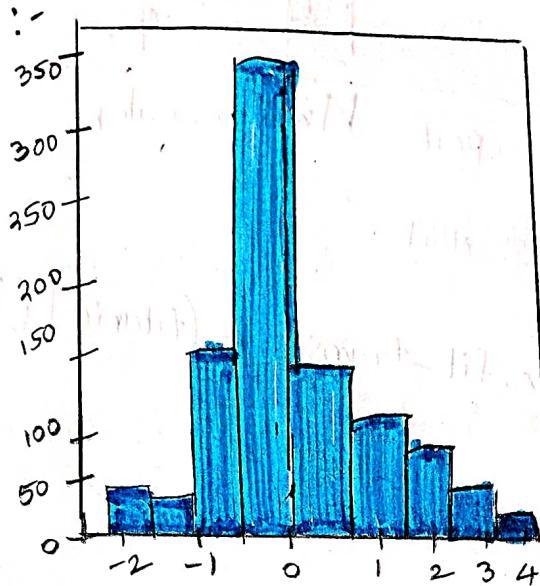


# Histogram of Converted "Age"

# titanic[ "Age\_sc" ].hist()

# plt.show()

Out :-



# Here, we see, Histogram is not going to change, But, Values does.

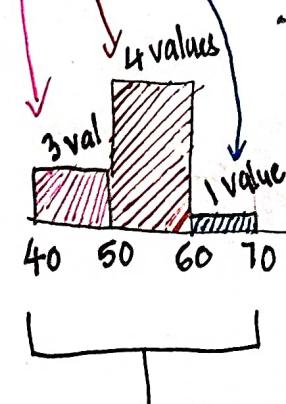
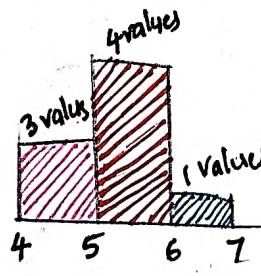
Example :-

\* How ? Histogram Remains Same, But changed Values

Ans :-

Ex :-

| X  | Changed X |
|----|-----------|
| 40 | 4         |
| 50 | 5         |
| 42 | 4.2       |
| 43 | 4.3       |
| 54 | 5.4       |
| 58 | 5.8       |
| 59 | 5.9       |
| 62 | 6.2       |



# Here only "X" Values Only changes But Histogram Remains Same.

## II # Normalisation

~~~~\* ~~~

### \* MinMax Scaling

→ MinMax Scaling Scales The Values between "0" to "1" (Every Thing is one value) No -ve value

Ex:-

| X  |                        |
|----|------------------------|
| 65 | $\frac{65-34}{53}$     |
| 45 | $\frac{45-34}{53}$     |
| 34 | $\frac{34-34}{53} = 0$ |
| 67 | $\frac{67-34}{53}$     |
| 67 | $\frac{67-34}{53}$     |
| 87 | $\frac{87-34}{53} = 1$ |

$$\frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

MinMaxScaler

Code: From sklearn.preprocessing import

# min\_max = MinMaxScaler() # Shortcut

# titanic["Age\_mm"] = min\_max.fit\_transform(titanic[["Age"]])

fit\_transform  
Creating new column

# titanic (calling/show data)

Out :-

|   | Age  | Age_sc  | Age_mm |
|---|------|---------|--------|
| 0 | 22.0 | -0.5637 | 0.2911 |
| 1 | 38.0 | 0.6487  | 0.472  |
| 2 | 26.0 | 0.259   | 0.32   |
| 3 |      |         |        |
| 4 |      |         |        |
| 5 | 26.0 | -0.25   | 0.32   |
| 6 | 32.0 | 0.20    | 0.39   |

### \* Robust Scaling

$$\frac{X - X_{\text{median}}}{IQR}$$

$$\therefore IQR = Q_3 - Q_1$$

## Code :

From sklearn.preprocessing import RobustScaler

# rs = RobustScaler() → shortcut()

# titanic["Age-rs"] = rs.fit\_transform(titanic[["Age"]])

titanic["Age-rs"] (or) = RobustScaler().fit\_transform(...)

If not using any shortcut like rs = RobustScaler()

# titanic

Out

| Age  | Age-SC | Age-MM | Age-RS |
|------|--------|--------|--------|
| 22.0 | 0.565  | 0.461  | 0.4615 |
| 38.0 | 0.66   | 0.769  | 0.769  |
| 26.0 | 0.25   | -0.153 | -0.153 |
| ...  |        |        |        |
| 26.0 | -0.25  | 0.153  | 0.153  |
| 32.0 | 0.2    | 0.396  | 0.3076 |

## \* Max Abs Scaling

$$\frac{x}{x_{\max}}$$

Ex:-

| x  |           |
|----|-----------|
| 68 | 68/80     |
| 44 | 44/80     |
| 52 | 52/80     |
| 69 | 69/80     |
| 80 | 80/80 = 1 |

Max. value

## code

From sklearn.preprocessing import

MaxAbsScaler

# mas = MaxAbsScaler()

# titanic["Age-mas"] = mas.fit\_transform(titanic[["Age"]])

# titanic

Out :-

|     | Age | Age - sc | Age - mm | Age - rs | Age - mas | Original Variable |
|-----|-----|----------|----------|----------|-----------|-------------------|
| 0   | 1   | 0.2750   |          |          |           | Standard Scaler   |
| 1   | 1   | 0.4950   |          |          |           | Min max Scaler    |
| 2   | 1   | 0.3250   |          |          |           | Robust Scaler     |
|     |     |          |          |          |           | Max Abs Scaler    |
| 889 | 1   | 0.3250   |          |          |           |                   |
| 890 | 1   | 0.4000   |          |          |           |                   |

891 x 5 col

Original Variable  
Standard Scaler  
Min max Scaler  
Robust Scaler  
Max Abs Scaler

5/04/22  
4:58 pm.

Example :- Discretization

Continuous Data

|      |
|------|
| 12.5 |
| 22.8 |
| 25.2 |
| 18.3 |
| 12.4 |
| 15.8 |
| 23.3 |

⇒ Mileage of a Car (Discrete, categorical)

$\leq 15 \rightarrow$  low. mileage

$15 - 20 \rightarrow$  Avg. mileage

$20 - 25 \rightarrow$  good. mileage

$\geq 25 \rightarrow$  very good

By our  
own choice  
To divide the  
into parts

## 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 Exam  
again

⇒ rectified but missed another section

$\Rightarrow$  Total correctly done.

## Boosting

⇒ Decision Tree 1 :- misclassified records

## ⇒ Decision Tree 1

misclassified records

## ⇒ Decision Tree

misclassified records

## ⇒ Decision Tree 3

- \* 2<sup>nd</sup> weak learner is dependent on 1<sup>st</sup> weak learner
  - algorithm doesn't give more accuracy.

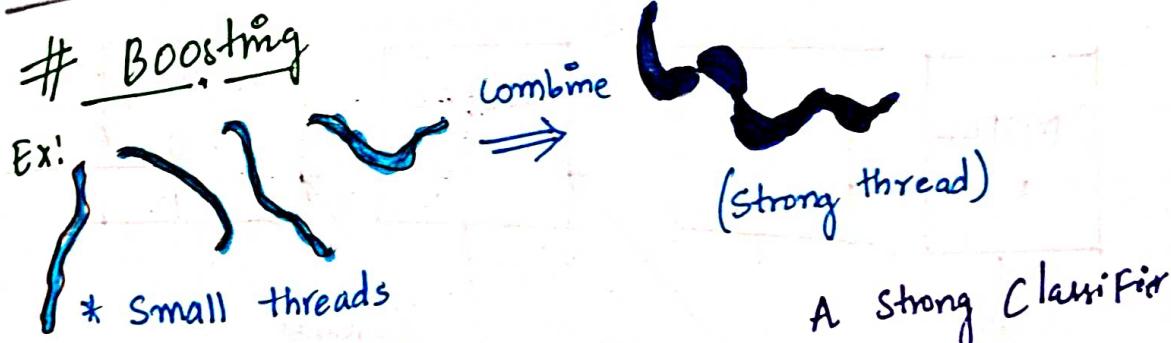
\* Formula for Boosting :-

$$F_T(x) = \sum_{t=1}^T f_t(x) \quad \text{Summation of } f_t(x) \text{ up to } T$$

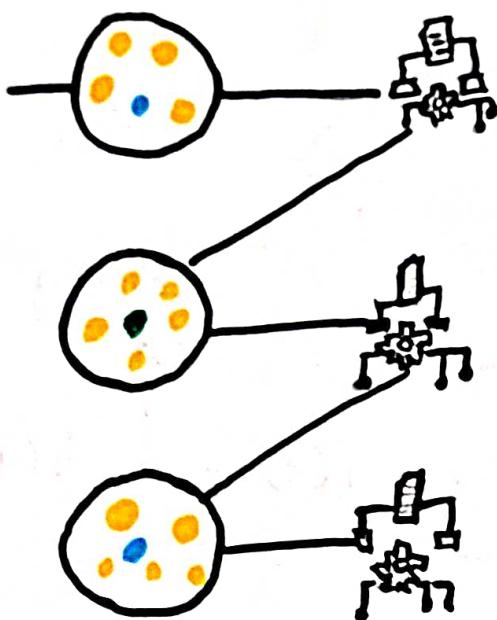
$$f_t(x) = \alpha_t \frac{h(x)}{\text{Learning rate}} \quad \text{Each model}$$

- \* boosting is the Combination of weak learners:

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



- \* Multiple Weak classifier (stumps)



"Sequential learning":

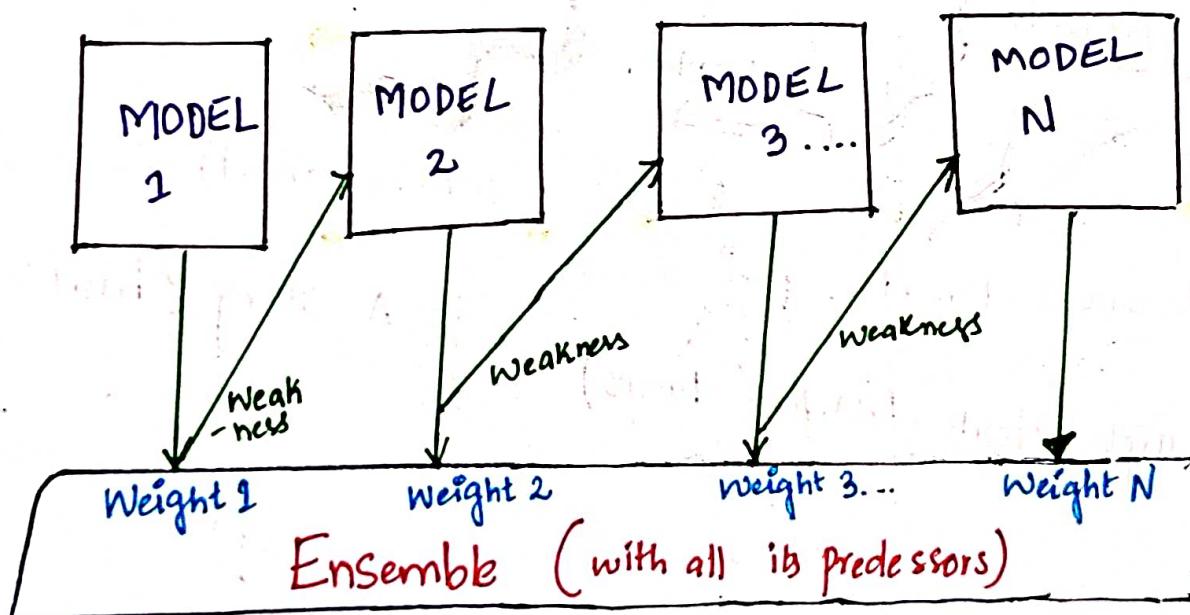
- \* From the first algorithm whatever mistakes was done.
- \* we share to the second algorithm.
- \* again it shared to third algorithm....
- \* it is in "Sequential order"

# Boosting - "Sequential" Learning.

1. Ada Boost  $\rightarrow$  Boosting.

Adaptive

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



covid (positive/negative)

Ex:- ✓

Sample DataSet

|   |   |   |
|---|---|---|
| + | + | - |
| - | - | + |
| + | - | + |
| + | - | - |

Weak Learner 1

Weak learner 2

Weak learner 3

iteration 1

|   |   |   |   |
|---|---|---|---|
| + | + | + | - |
| - | - | - | - |
| + | - | - | - |
| - | - | - | - |

miss classified

iteration 2

|   |   |   |   |
|---|---|---|---|
| + | + | + | - |
| - | - | - | - |
| + | - | - | - |
| - | - | - | - |

miss classified

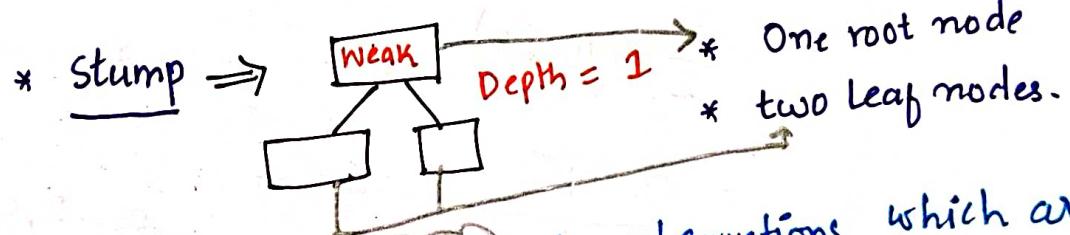
iteration 3.

|   |   |   |   |
|---|---|---|---|
| + | + | + | - |
| - | - | - | - |
| + | - | - | - |
| - | - | - | - |

# Final classifier / strong classifier.

|   |   |   |   |
|---|---|---|---|
| + | + | + | - |
| - | - | - | - |
| + | - | - | - |

\* In AdaBoost : Each Weak learner is called "stump"



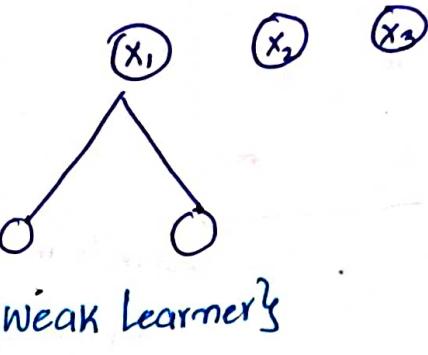
\* Step 1 : AdaBoost assigns weights to observations which are incorrectly predicted to predict these values.

Step 2 : Next model works which are wrongly predicted before.

Ex:-

|   | $X_1$ | $X_2$ | $X_3$ | $X_4$ | O/p | weights |
|---|-------|-------|-------|-------|-----|---------|
| 1 | -     | -     | -     | -     | Yes | $y_1$ ✓ |
| 2 | -     | -     | -     | -     | No  | $y_1$ ✓ |
| 3 | -     | -     | -     | -     |     | $y_1$ ✓ |
| 4 | -     | -     | -     | -     |     | $y_1$ ✗ |
| 5 | -     | -     | -     | -     |     | $y_1$ ✓ |
| 6 | -     | -     | -     | -     |     | $y_1$ ✓ |
| 7 | -     | -     | -     | -     |     | $y_1$ ✓ |

# Taken by Entropy/Inform. gain



\*1. Total Error [TE] :-  $\frac{1}{7}$

$$E_t = E_w \quad \therefore E_t = \text{"Episimol"} T$$

2. Performance of Stump :-

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

$\therefore TE = \text{Total Error}$

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

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

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

$$= 0.895$$

$$= \text{Weight} * e^{-PS}$$

correct records

$\therefore PS = \text{Performance Stump}$

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

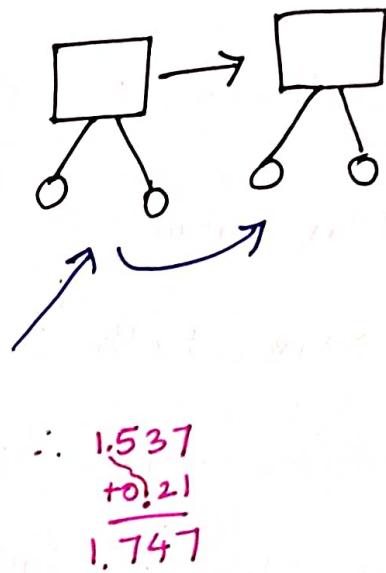
$$+ \text{incorrect record} = \text{Weight} * e^{PS} \rightarrow \text{incorrect records.}$$

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

New Sample weight

is it 1??

| Weights       | New weights       | Normalized weights | Buckets          |
|---------------|-------------------|--------------------|------------------|
| $\frac{1}{7}$ | $0.05 \div 0.649$ | 0.077              | $(0 - 0.07)$     |
| $\frac{1}{7}$ | $0.05 \div 0.649$ | 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              | $(1.747 - 0.75)$ |
| $\frac{1}{7}$ | 0.05              | 0.077              | -                |
| $\frac{1}{7}$ | 0.05              | 0.077              | -                |
| <hr/>         |                   | $\sim 1$           |                  |



\* STEPS :-

$$\Rightarrow \text{Weight} = \frac{1}{n}$$

$$\Rightarrow \text{Total Error}_t = \frac{[\text{Correct Prediction} - n]}{n}$$

$$\Rightarrow \text{Total weighted error}_t = \frac{\sum (\text{Weight}(i) * \text{Error}(i))}{\sum(\text{Weight})}$$

$$\Rightarrow \alpha_t = \ln \left[ \frac{1 - \text{TE}}{\text{TE}} \right]$$

$$\Rightarrow \text{Weight} = \text{Weight} * \exp(\alpha_t * \text{Total error})$$

$$\Rightarrow \text{total error} = 0 \text{ if } (y == \hat{y}), \text{ else } 1.$$

## \* Parameters

- \* n-estimators :- no. of weak learners
- \* Criterion :- Gini / Entropy
- \* Max\_Feature :- all feature / selected ones
- \* Max\_depth : <https://t.me/AIMLDeepThaught/612>
- \* 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 :

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 is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets three, let it be" for poisonous Oak and Ivy.

Attribute  $\rightarrow$  Information



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

6. gill-attachment : black = k, brown = n, buff = b, chocolate = h,  
gray = g, green = t, 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 = m

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

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

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

12. stalk-surface-above-ring : fibrous = f, scaly = y, silky = k,  
smooth = s

13. stalk-surface-below-ring : fibrous = f, scaly = y, silky = k,  
smooth = s

14. stalk-colour-above-ring : brown = n, buff = b, cinnamon = c,  
gray = g, orange = o, pink = p, red = e,  
white = w, yellow = y.

15. stalk-colour-below-ring :

16. veil-type : partial = p, universal = u

17. veil-color : brown = n, orange = o, white = w, yellow = y

18. Ring - number :- none = n, One = o, two = t, large = l, flaring = f.
19. Ring - type :- cobwebby = c, evanescent = e, sheathing = s, zone = z, none = n, Pendent = p.
20. Spore - Print :- black = b, brown = n, buff = b, chocolate = h, green = g, 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)

#### Business Problem

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

# df = pd.read\_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 surface below 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         | b          | s                        | w                        | w                      | w                      | p         | w          | o           | p         | b                 | s          | u       |
| P     | x         | s           | y         | t       | a    | f               | c            | b         | n          | s                        | w                        | w                      | p                      | w         | o          | p           | n         | n                 | g          | m       |
| E     | b         | s           | w         | t       | p    | f               | c            | n         | n          | s                        | w                        | w                      | p                      | w         | o          | p           | n         | n                 | s          | u       |
| P     | x         | y           | w         | f       | n    | f               | w            | b         | b          | s                        | w                        | w                      | w                      | p         | w          | o           | e         | n                 | a          | g       |
| E     | x         | s           | g         | f       | n    | f               | w            | b         | b          | s                        | w                        | w                      | w                      | p         | w          | o           |           |                   |            |         |

# df.shape

out: [8124, 23]

# df.info()

out: RangeIndex: 8124 Entries, 0 to 8123

| column        | Nonnull Count | Dtype  |
|---------------|---------------|--------|
|               | 8124          | Object |
| * class       | "             | "      |
| * cap-shape   | "             | "      |
| * cap-surface | "             | "      |
| * habitat     | "             | "      |

F df.isnull().sum()

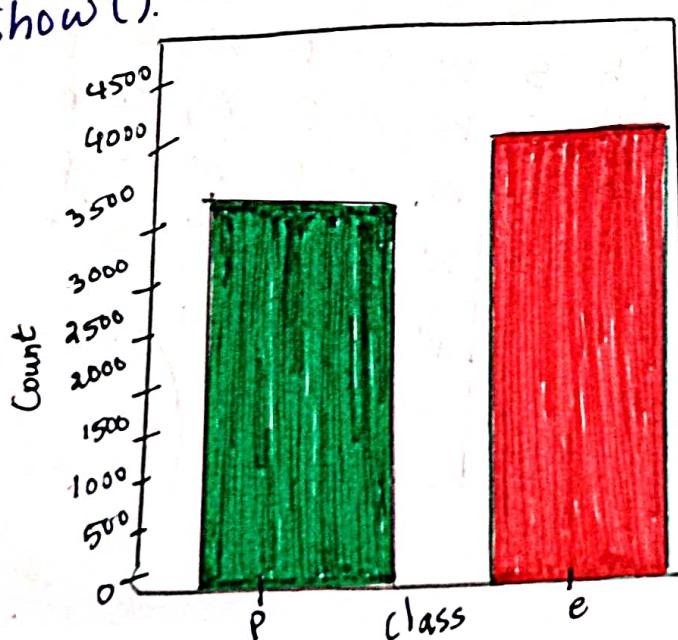
out: 0.

EDA

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

plt.show()

AE



# Transpose. gives total no. columns, without ...

# df.describe().transpose()

Out

|                             | Count | Unique | top | freq |
|-----------------------------|-------|--------|-----|------|
| (0p) 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<br>ring | 8124  | 4      | s   | 5176 |
| Stalk-Surface-below<br>ring | 8124  | 4      | s   | 4936 |
| Stalk-color above<br>ring   | 8124  | 9      | w   | 4464 |
| Stalk-color below<br>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 |

\* posinou  
\* non posinou

x & y

Names  
(nominal data)  $\Rightarrow$  One hot Encoding

#  $X = \text{pd.get_dummies}(\text{df.drop}(\text{"class", axis=1}), \text{drop\_first} = \text{True})$

# y = df [ "class" ]

廿  $x$

out

8124 rows x 95 columns

<https://t.me/AIMLDeepThaught/612>

# Y

|     |   |   |   |
|-----|---|---|---|
| out | : | 0 | P |
|     |   | 1 | e |
|     |   | 2 | e |
|     |   | 3 | P |
|     |   | 4 | e |

# x.shape, y.shape

out:  $(8124, 95), (8124, 1))$

## 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 ()  
# model.fit (x-train, y-train)  
out: AdaBoostClassifier()
```

## PREDICTION

```
# ypred-train = model.predict (x-train)  
# ypred-test = model.predict (x-test)
```

## Evaluation

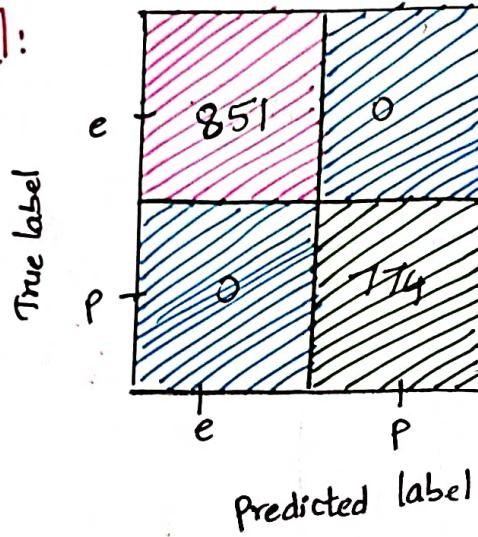
x Accuracy

```
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

```
# 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     |
| Accuracy     | 1.00      | 1.00   | 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 (model, x, y, cv=5)  
# print ("cross validation score:", scores.mean())  
# Out: cross validation score: 0.925
```

## \* model.features.importances

```
# model.feature_importances -
```

```
# Out array([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.feature_importances_, columns = ["Feature Importances"])
```

```
# 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 × 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-spacing w | 0.10               |
| ⋮              | ⋮                  |

## Hyper Parameter Tuning

```
from sklearn.model_selection import GridSearchCV
```

```
# estimator = AdaBoostClassifier()
```

```
# estimator = AdaBoostClassifier(n_estimators=101)
```

```
# param_grid = { "n_estimators": list(range(1,101)) }
```

```
# grid = GridSearchCV(estimator, param_grid, cv=5, scoring="Accuracy")
```

```
# grid . fit (x-train, y-train)
```

```
# grid . best_params_
```

```
# grid . best_params_
```

Out :

```
{ "n_estimators": 20 }
```

## final model

```
# final_model = AdaBoost Classifier (n-estimators = 20)
```

```
# final_model . fit (x-train, y-train)
```

```
https://t.me/AIMLDeepThaught/612 (x-train)
```

```
# pred-train = final_model . predict (x-test)
```

```
# pred-test = final_model . predict (x-test)
```

```
# print ("train accuracy:", accuracy score (y-train, pred-train))
```

```
# print ("test accuracy:", accuracy score (y-test, pred-test))
```

**Out:** train accuracy : 1.0

Test accuracy : 1.0

28/04/22  
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... ])

anjali  
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\_f 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

odor\_n 0.130435

gill\_size\_n 0.173913

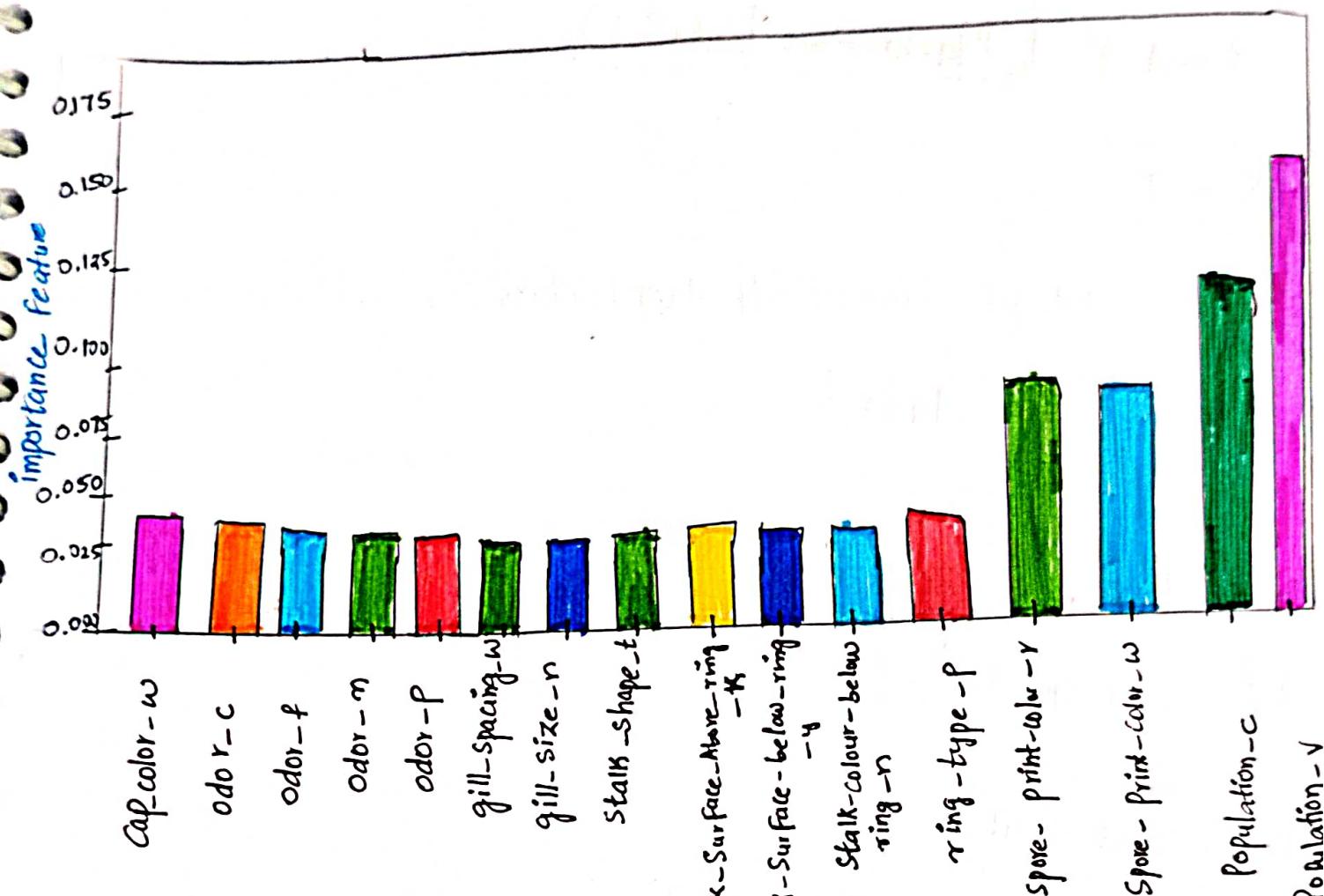
# plt.figure ( fig\_size = (14, 6), dpi = 200 )

# sns.barplot ( data = important, sort\_values ( "importance\_Feature" ), x = important.index, y = "importance\_Feature" )

# sns.barplot ( data = important, sort\_values ( "importance\_Feature" ), x = important.index, y = "importance\_Feature" )

# plt.xticks ( rotation = 90)

# plt. show()



## \*\*\* Gradient Boosting

\* within Gradient Boost, The Decision Trees are Pruned To Maximum Leaf nodes from 8 to 32.

Ex:-

| X <sub>1</sub><br>Age | X <sub>2</sub><br>City | y<br>Income | using D.T-1 | D.T-2           |                    | y Err. |
|-----------------------|------------------------|-------------|-------------|-----------------|--------------------|--------|
|                       |                        |             |             | Prediction<br>y | Residuals<br>Error |        |
| 32                    | A                      | 51000       |             | 53500           | -2500              | 48000  |
| 30                    | B                      | 78000       | model: 1 →  | 61000           | 17000              | 69000  |
| 21                    | A                      | 20000       |             | 28500           | -8500              | 23000  |
| 27                    | B                      | 44000       |             | 61000           | -17000             | 56700  |
| 36                    | B                      | 89000       |             | 90500           | -15000             | 98500  |
| 25                    | A                      | 37000       |             | 28500           | 8500               | 36500  |

Decision Tree - 2

Target (error)<sub>min</sub>

$3000 - 51000 = 48000$

$$\text{MODEL 2 Income} = \text{MODEL 1 income} + \text{Predicted Errors}$$

Model

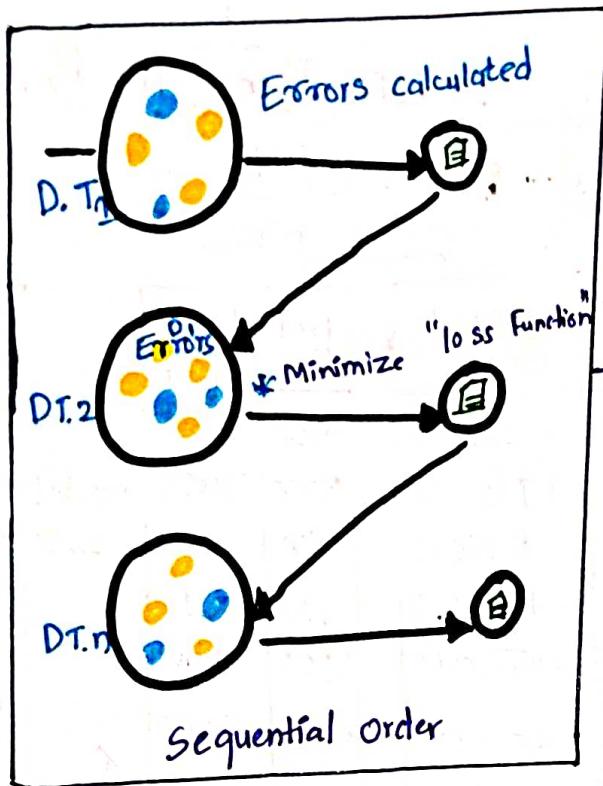
M<sub>1</sub>

M<sub>2</sub>

M<sub>n</sub>

| Feature     | X | X        | X             | X         | X             |
|-------------|---|----------|---------------|-----------|---------------|
| Target      | Y | $y - y'$ | $e_0$         | $y - y^0$ | $e_1$         |
| $H_0(x, y)$ |   |          | $H_1(x, e_0)$ |           | $H_2(x, e_1)$ |
|             |   |          |               |           | $H_n(x, e_n)$ |

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

(Overall Error should be Minimum)

Ques: What is difference between cost function & loss function?

Ans:

Ex:-

|           | Exam | Error | Total = 100 |
|-----------|------|-------|-------------|
| 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

<https://t.me/AIMLDeepThought/612>

Loss function : Overall Error

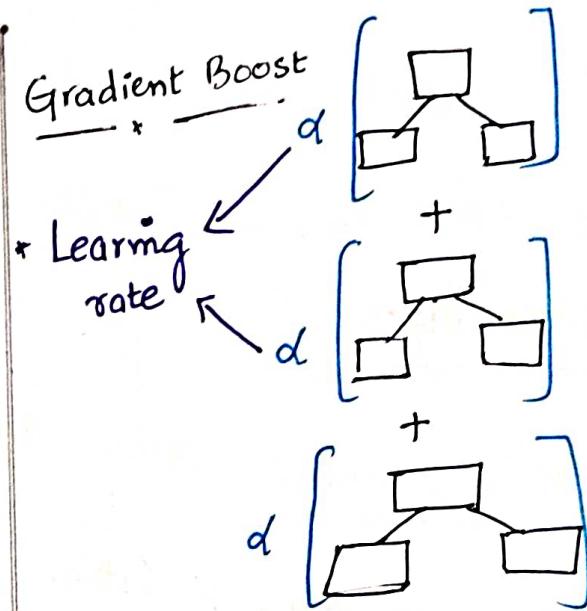
Loss function

Ques:- 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

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



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 is identified as definitely edible, definitely poisonous } or of unknown edibility and not recommended. this latter class was combined with the poisonous one. the Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets for poisonous Oak and Ivy. Let it be



Attribute + Information

1. Cap-shape :- bell = b, conical = c, convex = x, flat = f, knobbed = k, sunken = s → Mushroom
2. Cap-Surface :- fibrous = f, grooves = g, scaly = y, smooth = s
3. Cap-colour :- brown = n, buff = b, cinnamon = c, gray = g, green = o, pink = p, purple = u, red = e, white = w, yellow = y.
4. bruises ? : bruises = t, no = F
- odor : almond = a, anise = l, creosote = 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 = t, 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 = m

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

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

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

12. stalk-surface-above-ring : fibrous = f, scaly = y, silky = k,  
smooth = s

13. stalk-surface-below-ring : fibrous = f, scaly = y, silky = k,  
smooth = s

14. stalk-colour-above-ring : brown = n, buff = b, cinnamon = c,  
gray = g, orange = o, pink = p, red = e,  
white = w, yellow = y.

15. stalk-colour-below-ring :

16. veil-type : partial = p, universal = u

17. veil-color : brown = n, orange = o, white = w, yellow = y

18. Ring - number :- none = n, One = o, two = t, large = l, flaring = f.
19. Ring - type :- cobwebby = c, evanescent = e, sheathing = s, zone = z, none = n, Pendent = p.
20. Spore - Print :- black = b, brown = n, buff = b, chocolate = h, green = g, 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)

#### Business Problem

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

# df = pd.read\_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 surface below 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         | b          | s                        | w                        | w                      | w                      | p         | w          | o           | p         | b                 | s          | u       |
| P     | x         | s           | y         | t       | a    | f               | c            | b         | n          | s                        | w                        | w                      | p                      | w         | o          | p           | n         | n                 | g          | m       |
| E     | b         | s           | w         | t       | p    | f               | c            | n         | n          | s                        | w                        | w                      | p                      | w         | o          | p           | n         | n                 | s          | u       |
| P     | x         | y           | w         | f       | n    | f               | w            | b         | b          | s                        | w                        | w                      | w                      | p         | w          | o           | e         | n                 | a          | g       |
| E     | x         | s           | g         | f       | n    | f               | w            | b         | b          | s                        | w                        | w                      | w                      | p         | w          | o           |           |                   |            |         |

# df.shape

out: [8124, 23]

# df.info()

out: RangeIndex: 8124 Entries, 0 to 8123

| column        | Nonnull Count | Dtype  |
|---------------|---------------|--------|
|               | 8124          | Object |
| * class       | "             | "      |
| * cap-shape   | "             | "      |
| * cap-surface | "             | "      |
| :             | "             | "      |
| * habitat     | "             | "      |

F df.isnull().sum()

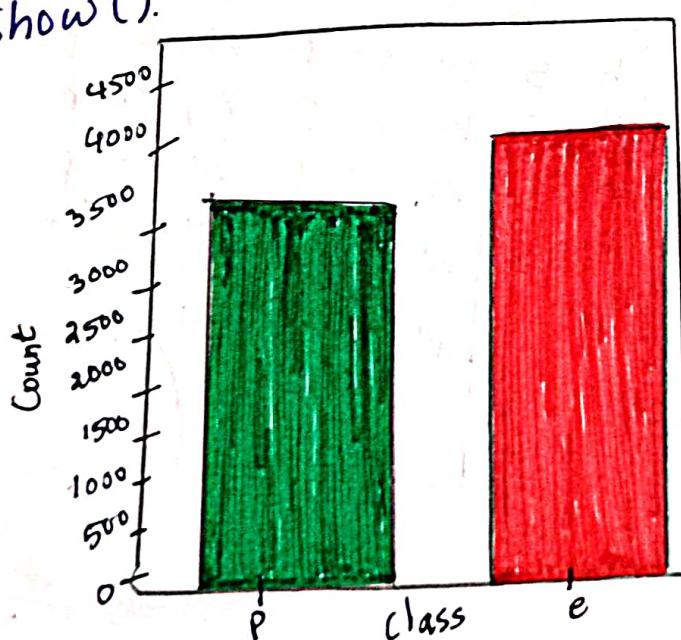
out: 0.

EDA

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

plt.show()

AE



# Transpose. gives total no. columns, without ...

# df.describe().transpose()

Out

|                             | Count | Unique | top | freq |
|-----------------------------|-------|--------|-----|------|
| (0p) 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<br>ring | 8124  | 4      | s   | 5176 |
| Stalk-Surface-below<br>ring | 8124  | 4      | s   | 4936 |
| Stalk-color above<br>ring   | 8124  | 9      | w   | 4464 |
| Stalk-color below<br>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 |

\* posinou  
\* non posinou

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"]` <https://t.me/AIMLDeepThaught/612>

# `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 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 accuracy":, accuracy_score(y-train, ypred-train))  
# print("test accuracy":, accuracy_score(y-test, ypred-test))
```

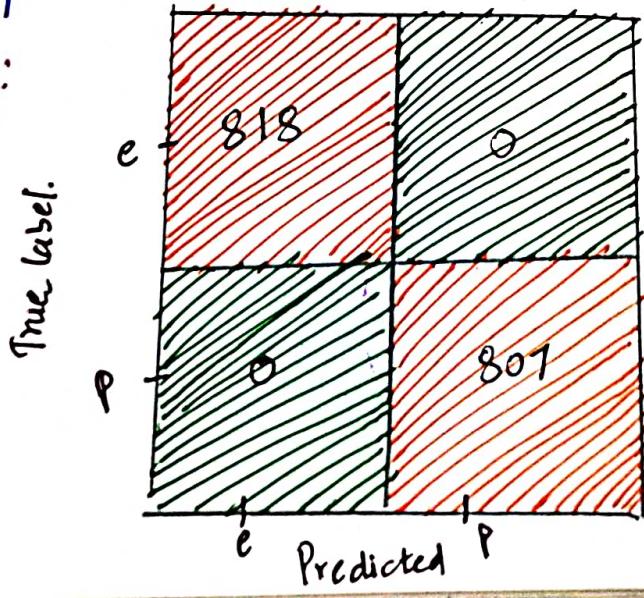
**Out**: train accuracy : 1.0  
Test accuracy : 1.0

### 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     |
| 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+00 ,  
 6.19492 , 1.36263 , 4.49731 -3.5778 , ...])

## Hyper Parameter Tuning

```
from sklearn.model_selection import GridSearchCV
# estimator = Gradient Boosting Classifier()
# 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")
# grid.fit(x_train, y_train)
# grid.best_params_
out: {"learning_rate": 0.2, "n_estimators": 100}
```

Important Parameter for Gradient Boost

## Final Model

```
# final model = Gradient Boosting Classifier(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, 1.27008e-08, 4.735e-08, 1.923168e-08, 3.20965e-08, 0.000bed])
```

```
# important = pd.DataFrame(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_feature
```

```
gill_size-n 0.010084  
Spore print color-b 0.014901  
odor-l 0.019874  
Spore print color-r 0.032914  
stalk root-r 0.052216  
bruises-t 0.060678 . . .
```

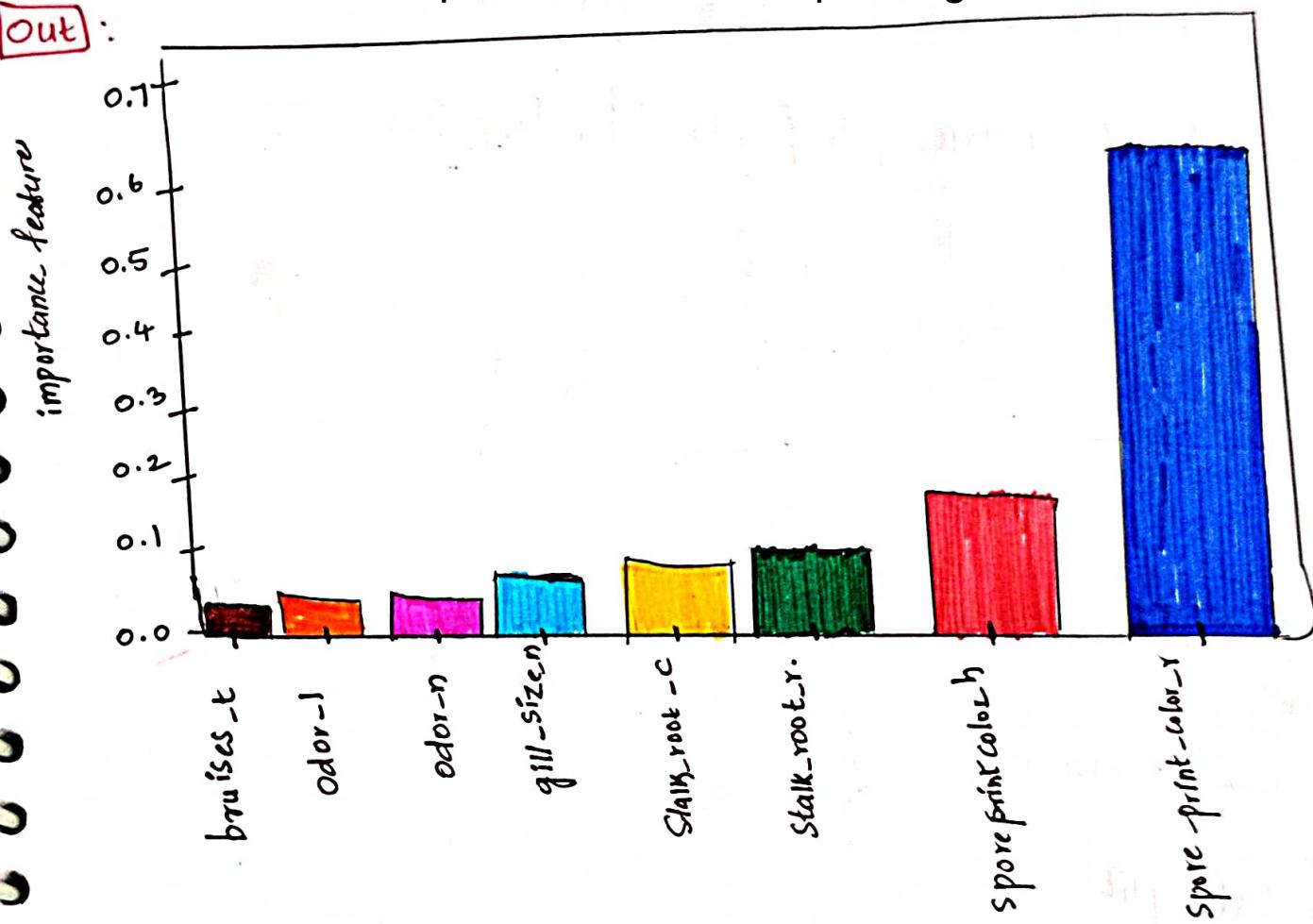
```
# plt.figure(figsize=(14,6), dpi=200)
# sns.barplot(data=Jacks.sort_values("importance-feature"),
#             x=Jacks.index, y="importance-feature")
```

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

```
# plt.show()
```

<https://t.me/AIMLDeepThought/612>

**Out**:



## XG BOOST

Gradient

Xtreme

\* Adding "penalty" term Reduce overall 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-flexibility

: Cache awareness and out of core computing. (-) it uses all cores in system

\* Reduce Overfitting:

"Regularization" for avoiding overfitting

Efficient Handling

missing data

Inbuilt

"cross-validation" Capability

XG-boost

error + Penalty Term

$L_2: \frac{d}{2} (\text{slope})^2$

$L_1: \frac{d}{2} |\text{slope}|$

with in "XGBoost":  
 $\alpha = \frac{d}{2}$   
gamma

## XG boost classifier Black box Model

Base model = Probability = 0.5

Ex:- DataSet

(Pr-Approval)

| Salary     | Credit | Approval | Residuals                 |
|------------|--------|----------|---------------------------|
| $\leq 50k$ | Bad    | 0        | $0 - 0.5$<br>-0.5<br>-0.5 |
| $\leq 50k$ | Good   | 1        | 0.5                       |
| $\leq 50k$ | Good   | 1        | 0.5                       |
| $> 50k$    | Bad    | 0        | -0.5                      |
| $> 50k$    | Good   | 1        | 0.5                       |
| $> 50k$    | Normal | 1        | 0.5                       |
| $\leq 50k$ | Normal | 0        | -0.5                      |

### Steps :-

1. Create Binary Decision Tree using Features

2. Calculate

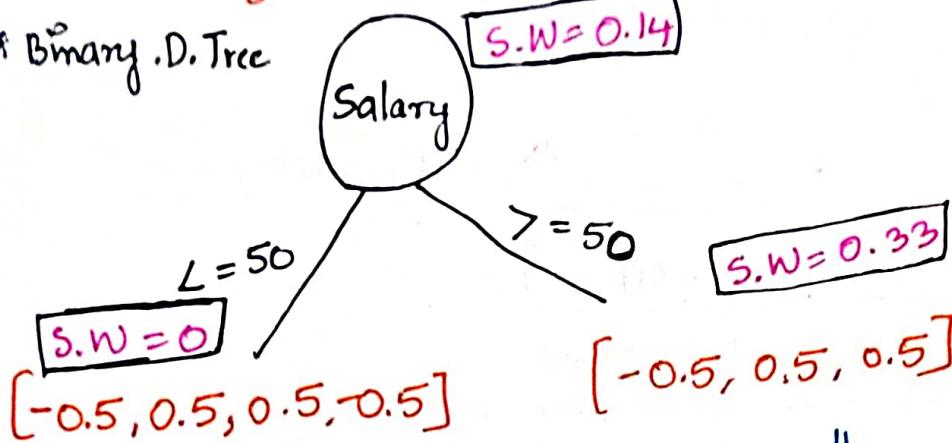
Similarity weight

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

3. Information Gain.

$$\text{Errors} := [-0.5, 0.5, 0.5, -0.5, 0.5, 0.5, 0.5, -0.5]$$

# Binary D. Tree



$$\downarrow$$

Similarity weight =  $\frac{E(\text{res})^{\gamma}}{EP(1-P)\lambda}$

$$\begin{aligned}
 &= \frac{[-0.5 + 0.5 + 0.5]^{\gamma}}{0.5[1-0.5] + 0.5[1-0.5] + 0.5[1-0.5] + 0.5[1-0.5]} \\
 &= \frac{0.25}{0.75} = \frac{1}{3} \Rightarrow 0.33
 \end{aligned}$$

$$= 0$$

# for Root node Similarity weight =  $\frac{E(\text{res})^{\gamma}}{EP(1-P)\lambda}$

$$\begin{aligned}
 &= \frac{[-0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]^{\gamma}}{0.5[1-0.5] + 0.5[1-0.5] + 0.5[1-0.5] + 0.5[1-0.5] + 0.5[1-0.5] + 0.5[1-0.5] + 0.5[1-0.5]} \\
 &= \frac{0.25}{1.75} = \frac{1}{7} = 0.14
 \end{aligned}$$

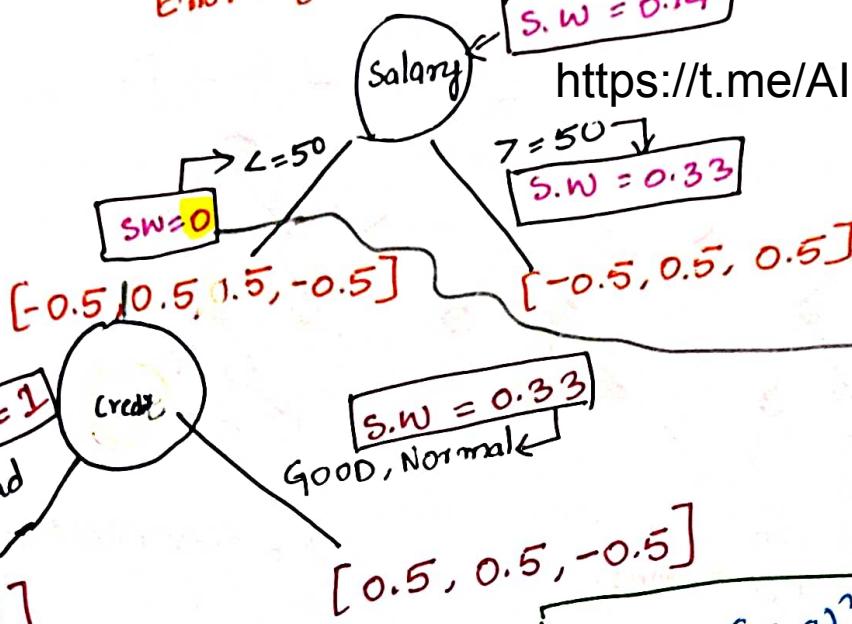
$$* \text{ information Gain} = 0 + 0.33 - 0.14$$

$$\rightarrow = 0.19$$

Error: [0.5, 0.5, 0.5, -0.5, 0.5, 0.5, -0.5]

$$S.W = 0.14$$

<https://t.me/AIMLDeepThaught/612>



$$* S.W = \frac{E(\text{res})^r}{E(P(1-P))}$$

$$* S.W = \frac{E(\text{res})^r}{E(P(1-P))}$$

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

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

= 1

$$* \text{ information Gain} = 1 + 0.33 - 0$$

$$= 1.33$$

| Salary        | credit | Approval |
|---------------|--------|----------|
| $\angle = 50$ | B      | 0        |

activation function  
sigmoid  $\sigma$   $\rightarrow$  [0 +  $\alpha(1)$ ]  $\rightarrow$  Based on "d" value  $\rightarrow$  o/p [0-1]

Base model = 0.5  
Real probability =

$$\log \frac{P}{1-P} = \log \left( \frac{0.5}{1-0.5} \right) = 0$$

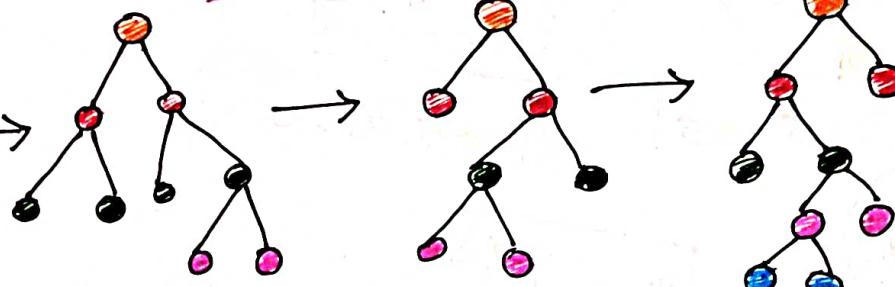
Output :-

$$\sigma [o + \alpha_1 [DT_1] + \alpha_2 [DT_2] + \alpha_3 [DT_3] + \dots + \alpha_n [DT_n]]$$

For any new record  $\uparrow$

$$o + \alpha_1 [DT_1] + \alpha_2 [DT_2] + \alpha_3 [DT_3] + \dots$$

Base Model



Based on  
independent features.

$\therefore \lambda = \text{cross validation.}$

## \* XG boost regressor

\* Base model = 51K

Ex:-

| $x_1$ | $x_2$ | $y$    | Resid<br>-<br>base |
|-------|-------|--------|--------------------|
| Exp   | Gap   | Salary |                    |
| 2     | Yes   | 40K    | -11K               |
| 2.5   | Yes   | 42.5   | 9                  |
| 3     | No    | 52K    | 1                  |
| 4     | No    | 60K    | 9                  |
| 4.5   | Yes   | 62K    | 11                 |

$$\text{Avg} = 51.$$

$$s.w = \frac{1}{6}$$

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

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

Experience

$$s.w = \frac{45.5}{72}$$

$$[-11]$$

$$s.w = 28.8$$

$$[9, 1, 9, 11]$$

$$\frac{[-9, 1, 9, 11]}{4+1} = \frac{12}{5} = 28.8$$

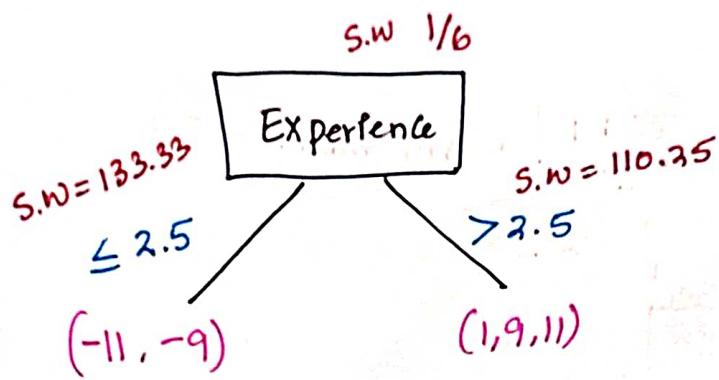
$\lambda = 2$

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

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

$$\therefore \lambda = 1$$

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



$$\begin{aligned}
 S.W. &= \frac{[-11-9]^2}{2+1} = \frac{[1+9+11]^2}{3+1} \\
 &= \frac{[-20]^2}{3} = \frac{[21]^2}{4} \\
 &= \frac{400}{3} \Rightarrow 133.33 \\
 &= 110.25
 \end{aligned}$$

any records goes from base model = 51  
 $\leq 2.5$   
Avg. of S.W

|                                                                                                                                              |                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| $51 + \alpha_1 \left[ \frac{[-11-9]}{2} \right] = (-10)$ <p style="text-align: center;"><small>Arg. of S.W <math>\leq 2.5</math></small></p> | $51 + \alpha_1 \left[ \frac{1+11+9}{3} \right] \Rightarrow \frac{21}{3} = 7$ |
|----------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|

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

↳ 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 is identified as definitely edible, definitely poisonous } or of unknown edibility and not recommended. this latter class was combined with the poisonous one. the Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets for poisonous Oak and Ivy. Let it be



Attribute + Information

1. Cap-shape :- bell = b, conical = c, convex = x, flat = f, knobbed = k, sunken = s → Mushroom
2. Cap-Surface :- fibrous = f, grooves = g, scaly = y, smooth = s
3. Cap-colour :- brown = n, buff = b, cinnamon = c, gray = g, green = o, pink = p, purple = u, red = e, white = w, yellow = y.
4. bruises ? : bruises = t, no = F
- odor : almond = a, anise = l, creosote = 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 = t, 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 = m

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

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

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

12. stalk-surface-above-ring : fibrous = f, scaly = y, silky = k,  
smooth = s

13. stalk-surface-below-ring : fibrous = f, scaly = y, silky = k,  
smooth = s

14. stalk-colour-above-ring : brown = n, buff = b, cinnamon = c,  
gray = g, orange = o, pink = p, red = e,  
white = w, yellow = y.

15. stalk-colour-below-ring :

16. veil-type : partial = p, universal = u

17. veil-color : brown = n, orange = o, white = w, yellow = y

18.

18. Ring - number :- none = n, One = o, two = t, large = l, flaring = f.
19. Ring - type :- cobwebby = c, evanescent = e, sheathing = s, zone = z, none = n, Pendent = p.
20. Spore - Print :- black = b, brown = n, buff = b, chocolate = h, green = g, 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)

#### Business Problem

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

# df = pd.read\_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 surface below 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         | b          | s                        | w                        | w                      | w                      | p         | w          | o           | p         | b                 | s          | u       |
| P     | x         | s           | y         | t       | a    | f               | c            | b         | n          | s                        | w                        | w                      | p                      | w         | o          | p           | n         | n                 | g          | m       |
| E     | b         | s           | w         | t       | p    | f               | c            | n         | n          | s                        | w                        | w                      | p                      | w         | o          | p           | n         | n                 | s          | u       |
| P     | x         | y           | w         | f       | n    | f               | w            | b         | b          | s                        | w                        | w                      | w                      | p         | w          | o           | e         | n                 | a          | g       |
| E     | x         | s           | g         | f       | n    | f               | w            | b         | b          | s                        | w                        | w                      | w                      | p         | w          | o           |           |                   |            |         |

# df.shape

out: [8124, 23]

# df.info()

out: RangeIndex: 8124 Entries, 0 to 8123

| column        | Nonnull Count | Dtype  |
|---------------|---------------|--------|
|               | 8124          | Object |
| * class       | "             | "      |
| * cap-shape   | "             | "      |
| * cap-surface | "             | "      |
| * habitat     | "             | "      |

F df.isnull().sum()

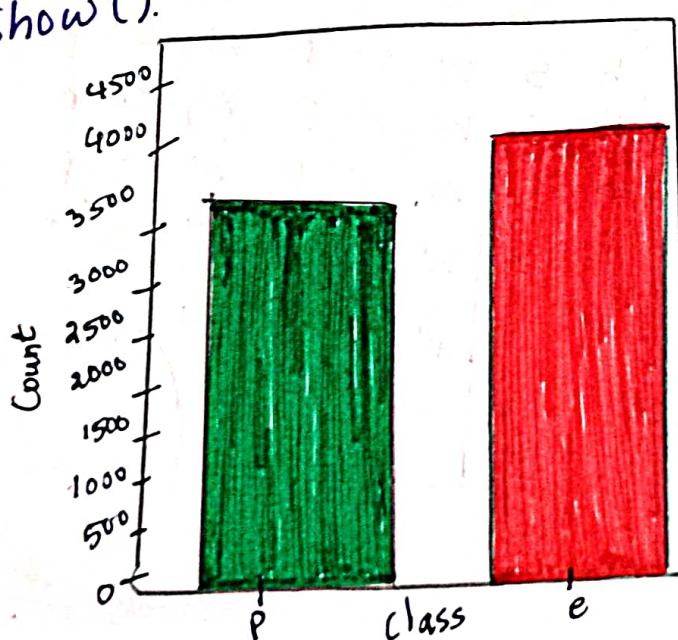
out: 0.

EDA

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

plt.show()

AE



# Transpose. gives total no. columns, without ...

# df.describe().transpose()

Out

|                             | Count | Unique | top | freq |
|-----------------------------|-------|--------|-----|------|
| (0p) 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<br>ring | 8124  | 4      | s   | 5176 |
| Stalk-Surface-below<br>ring | 8124  | 4      | s   | 4936 |
| Stalk-color above<br>ring   | 8124  | 9      | w   | 4464 |
| Stalk-color below<br>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 |

\* posinou  
\* non posinou

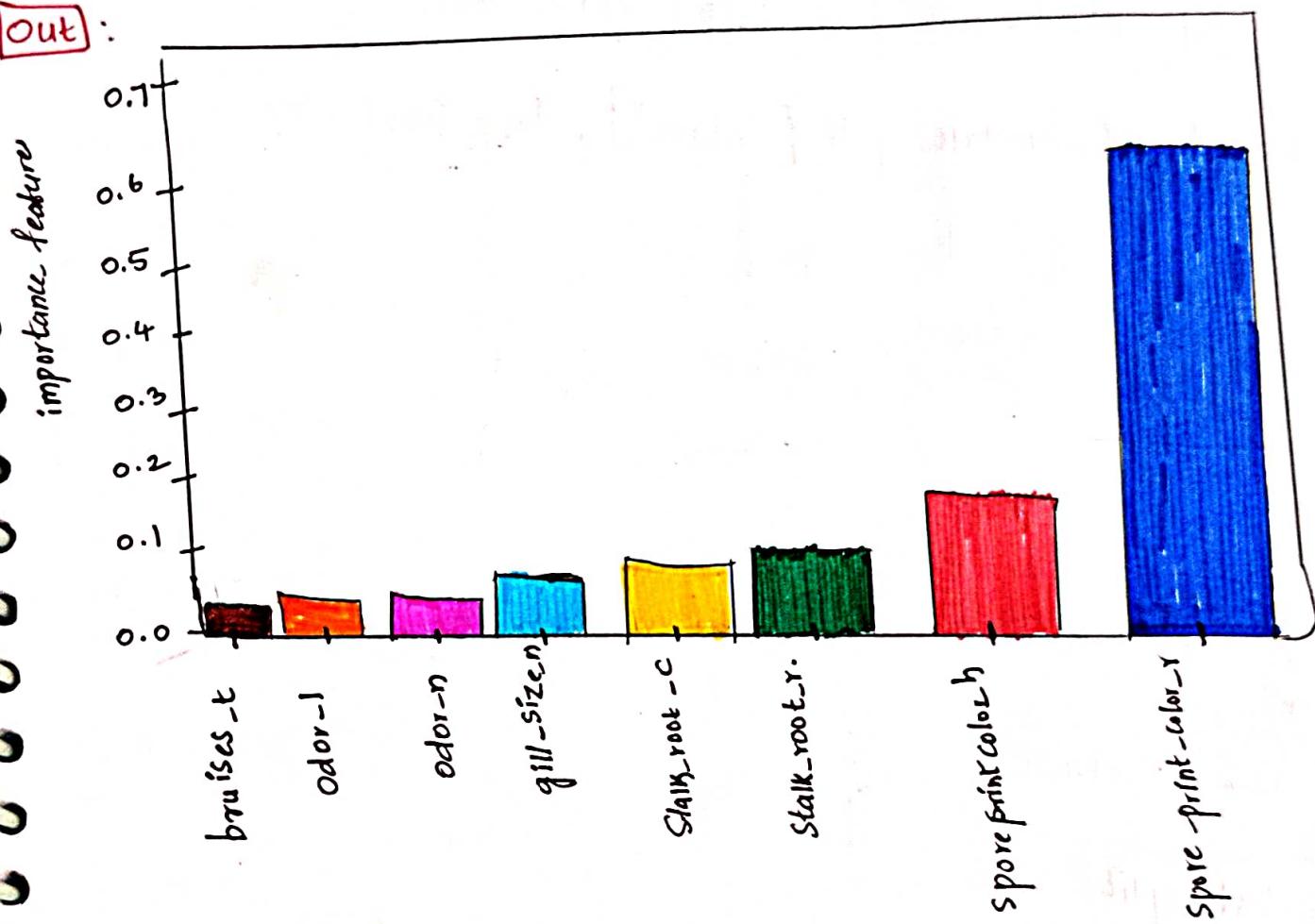
```
# plt.figure(figsize=(14,6), dpi=200)
```

```
# sns.barplot(data=Jacks.sort_values("importance-feature"),  
x=Jacks.index, y="importance-feature")
```

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

```
# plt.show()
```

**Out:**



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)
```

<https://t.me/AIMLDeepThaught/612>

# 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

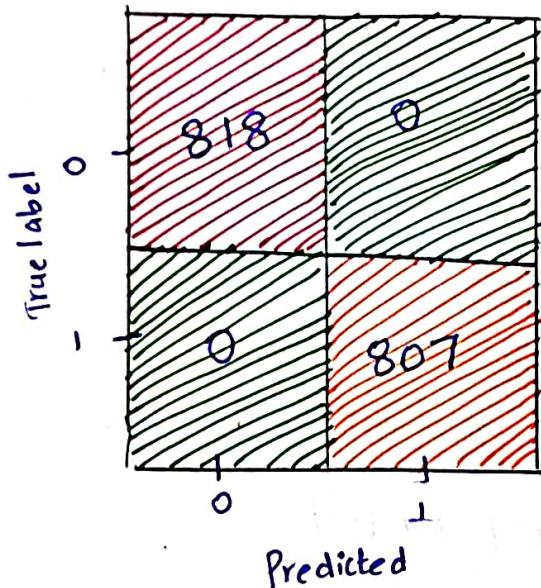
### \* Accuracy

```
from sklearn.metrics import accuracy_score  
# print("train accuracy:", accuracy_score(y-train, ypred-train))  
# print("test accuracy:", accuracy_score(y-test, ypred-test))  
[Out]: Train accuracy : 1.0  
Test accuracy : 1.0
```

## \* Confusion Matrix

```
from sklearn.metrics import plot_confusion_matrix
# plot_confusion_matrix (ngb-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+0,  0.0000e+0,  0.0000e+0,
            3.961512e-5,  0.00000e+0, 9.23225e-04])
```

## Hyper Parameter tuning

```
from sklearn.model_selection import GridSearchCV
```

Important parameters in XGBoost

```
# 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")
```

```
# grid.fit(x_train, y_train) CV = 5
```

```
# 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)  
  
# y_predictions_test = final_model. Predict (x-test)  
# y_predictions_train = final_model. Predict (x-train)  
  
# print ("train accuracy:", accuracy_score (y_prediction_train, y_train))  
# print ("test accuracy:", accuracy_score (y_prediction_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., 0.])

```
# important = pd. DataFrame (index = x. columns, data = final-  
model. feature_importances_, columns = [importance-  
Features])
```

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

```
# Jack . sort_values[["importance_feature"]]
```

### important\_features

**Out:**

|                     |          |
|---------------------|----------|
| odor-p              | 0.015831 |
| spore-print-Color-W | 0.020152 |
| :                   | :        |
| odor-n              | 0.290767 |

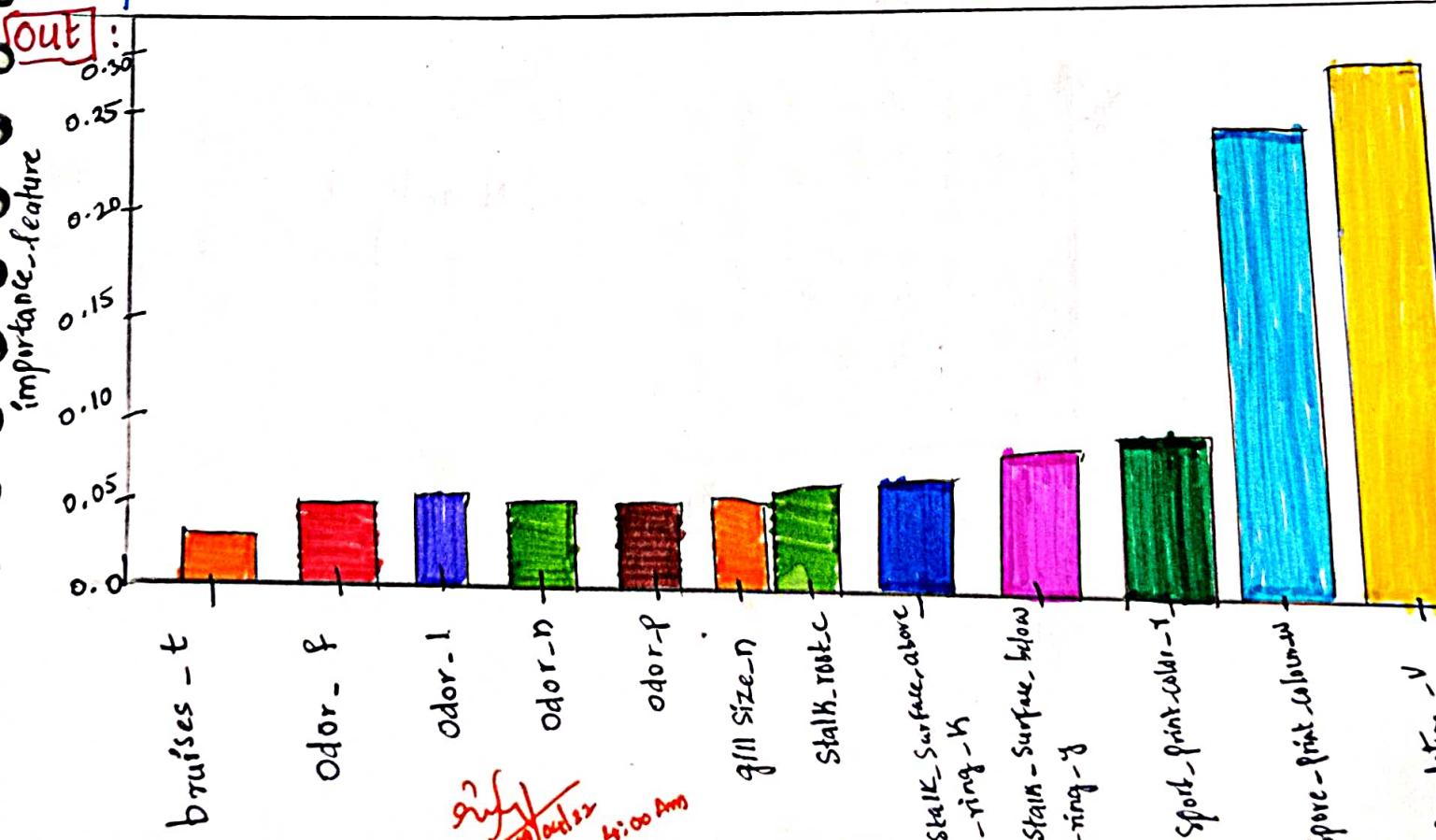
```
# plt.figure(figsize=(14,6), dpi=200)
```

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

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

```
# plt.show()
```

**Out:**



Dt:- 18/4/22  
2: 30 PM

\* Supervised Algorithm :-

it has Label Data

class (Yes/No)

\* Predicting :- Discrete (categories)

Ex:-

\* Binary classification

[0, 1]

\* Multi classification

[drug x, drug, doni]

classe

## #Classification [Algorithms]

→ Logistic Regression

→ KNN [K-Nearest Neighbours]

→ Decision Tree

→ Random Forest

→ Ada Boost

→ Gradient Boost

→ XG Boost

→ SVM [Support Vector Machine]

→ Navies Bayes

## ⇒ Classification Use Cases :

1. Which Category

a customer belongs to ?

2. whether Customer

Switches to another provider / brand ?

3. whether Customer

responds to a particular

Advertising Campaign ?

## ⇒ Evaluation Metrics :

### ⇒ Confusion Matrix

Ex:- whether The person is covid: (+) positive (-) negative.

covid  
(+) positive

| Actual | Predict |
|--------|---------|
| -      | +       |
| -      | +       |
| +      | -       |
| +      | -       |
| -      | -       |
| +      | +       |
| -      | -       |
| -      | -       |
| +      | +       |
| -      | +       |

Actual / True

|            |     | +ve | -ve |
|------------|-----|-----|-----|
| Prediction | +ve | T   | F   |
|            | +ve | 2 P | 3 P |
|            | -ve | F   | T   |
|            | -ve | 2 N | 3 N |

⇒ T/F - Correct Prediction | Incorrect Prediction

⇒ P/N - Predicted Positive | Predicted Negative

\* Another Type

|        |     | Predict |     |
|--------|-----|---------|-----|
|        |     | +ve     | -ve |
| Actual | +ve | TP      | FN  |
|        | -ve | FP      | TN  |

|        |     | Predict |        |
|--------|-----|---------|--------|
|        |     | +ve     | -ve    |
| Actual | +ve | 2 [TP]  | 2 [FN] |
|        | -ve | 3 [FP]  | 3 [TN] |

\* Classification [Accuracy] = 
$$\frac{\text{Correct Predictions}}{\text{total predictions}} = \frac{5}{10}$$

\* Mis-classification [Error] = 
$$\frac{\text{Incorrect prediction}}{\text{total predictions}} = \frac{5}{10}$$

Ques :- Which is more important "False positive", "False Negative"

Ans :- it is dependent on "Business problems"  $\rightarrow$

$\text{FN} \downarrow$   
 $\text{FP} \downarrow$   
 $\text{Error} \downarrow$   
 $\text{Accuracy} \uparrow$

$\Rightarrow$  Person is Having covid

Actual +ve

Predicted -ve

[FN] More important

Because if person is Having covid,  
 You predicted negative, Problem (i) are  
 (ii) spread

$\Rightarrow$  Person is not having covid

Actual -ve

FN

Predicted +ve

[FP]

If person is not having covid,  
 You predicted covid, (i) quarantine  
 (ii) Antibiotics (medicine)

FP

[FP] More important  
 But, Person is Having cancer, But predicted positive.

(i) Kimo therapy

Always Predicted Comes last  
 \* (Second letter)

Que: In business problems, They said to focus of "FN" should be law ?

<https://t.me/AIMLDeepThaught/612>

|      |   | Actual |    |
|------|---|--------|----|
|      |   | 1      | 1  |
| Pred | 1 | TP     | FP |
|      | 0 | FN     | TN |

accuracy = 91%  
FN = 8

|      |   | Actual |    |
|------|---|--------|----|
|      |   | T      | P  |
| Pred | T | 4      | 6  |
|      | P | 3      | 87 |
|      |   | FN     | TN |

Accuracy = 91%  
False Negative = 3

If client wants  
accuracy and FN

Best

|      |   | Actual |    |
|------|---|--------|----|
|      |   | 38     | 24 |
| pred | T | P      | FP |
|      | 0 | 38     | FN |
|      |   |        | TN |

$$\text{Accuracy} = 76\%$$

If client wants  
Only FN should be low,  
Not Accuracy

best

DT: 18/4/12  
10:00pm

$$\frac{TN}{FN+TN}$$

↑ Pred

## Precision (-re)

Correctly predicted (-ve)

Recall

f) actual

|     |   |     |     |
|-----|---|-----|-----|
|     |   | +ve | -ve |
| +ve | T | P   | FP  |
| -ve | F | N   | TN  |

precision (+ve)

↓  
Correctly predicted (tve)

Total predicted (+ve)

$$\frac{TP}{TP + FP}$$

$$\frac{TP}{TP + FP}$$

$$\frac{TP}{TP + FP}$$

$$2) \underline{\text{Recall (+ve)}} = \frac{\text{correctly predicted (+ve)}}{\text{total Actual positive (+ve)}}$$

$$= \boxed{\frac{TP}{TP + FN}}$$

$$\underline{\text{Recall (-ve)}} = \frac{\text{correctly predicted (-ve)}}{\text{total Actual negative (-ve)}}$$

$$= \boxed{\frac{TN}{FP + TN}}$$

Que :- What is difference between Recall and Precision?

Ans :- Precision :-

| True      | Recall                |
|-----------|-----------------------|
| Predicted | Comparing with Actual |

:- Comparing with Actual values.

+ denominator

:- Comparing with Actual values, where numerator is Same

+ denominator

changes, instead of prediction, Take actual

One, instead of predicted

positive

Take actual positive

+ True class

|   |         |         |
|---|---------|---------|
|   | +       | -       |
| + | True +  | False + |
| - | False - | True -  |

→ Type-I (Error)

Predictions

Type-II  
Error

- \* True positive [TP] :- when classifier predicted True and correct class was True.  
Ex: They have disease, predicted correctly.
- \* True Negative [TN] : when model predicted False and correct class was False.  
Ex:- No disease, But predictive not have disease.
- \* False Positive [FP] [Type-I Error] :- classifier predicted True but correct class was False.  
Ex:- Patient did not have disease.
- \* False Negatives [FN] [Type-II Error] :- classifier predicted False, but they actually do have disease.  
Ex:- Patient do not have disease but they actually do have disease.

## Key Performance Indicators [KPI]

# classification accuracy =

$$\frac{TP + TN}{(TP + TN + FP + FN)}$$

# Misclassification rate =  
(error rate)

$$\frac{FP + FN}{(TP + TN + FP + FN)}$$

## Example

|      |   | Actual |         |
|------|---|--------|---------|
|      |   | +      | -       |
| Pred | + | TP = 1 | FP = 1  |
|      | - | FN = 8 | TN = 90 |

$$\# \text{ classification accuracy} = \frac{TP + TN}{TP + FP + FN + TN} = 91\%$$

$$\# \text{ precision (+ve)} = \frac{TP}{TP + FP} = \frac{1}{2} = 50\%$$

$$\# \text{ Recall (+ve)} = \frac{TP}{TP + FN} = \frac{1}{1+8} = \frac{1}{9} = 11\%$$

$$\# \text{ precision (-ve)} = \frac{TN}{FN + TN} = \frac{90}{8+90} = \frac{90}{98} = 0.91\%$$

$$\# \text{ Recall (-ve)} = \frac{TN}{FP + TN} = \frac{90}{1+90} = \frac{90}{91} = 0.98\%$$

$$* \text{TPR} = \text{True Positive Rate} = \frac{TP}{TP + FN} \xrightarrow{\text{Actual (+ve)}}$$

$$* \text{FPR} = \text{False Positive Rate} = \frac{FP}{TP + FN} \xrightarrow{\text{Actual (+ve)}}$$

$$* \text{TNR} = \text{True Negative Rate} = \frac{TN}{TN + FP} \xrightarrow{\text{Actual (-ve)}}$$

$$* \text{FNR} = \text{False Negative Rate} = \frac{FN}{FP + TN} \xrightarrow{\text{Actual (-ve)}}$$

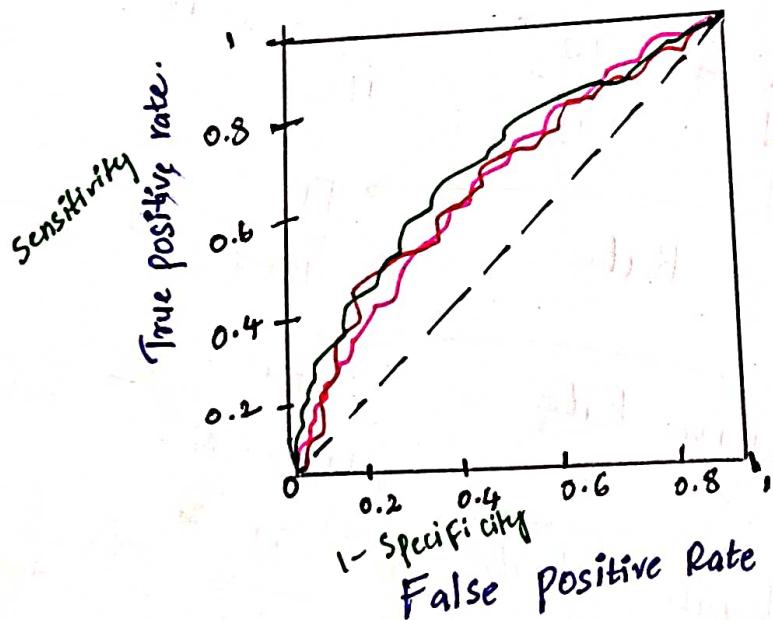
\* specificity =  $TNR \Rightarrow 1 - FPR$

= True Negative Rate =  $\frac{\text{True Negatives}}{\text{True negatives} + \text{False positives}}$

<https://t.me/AIMLDeepThought/612>

\* Sensitivity  $\Rightarrow TPR \Rightarrow \text{Recall} \Rightarrow \frac{TP}{TP + FN}$

# ROC [Reciver operating characteristic curve]

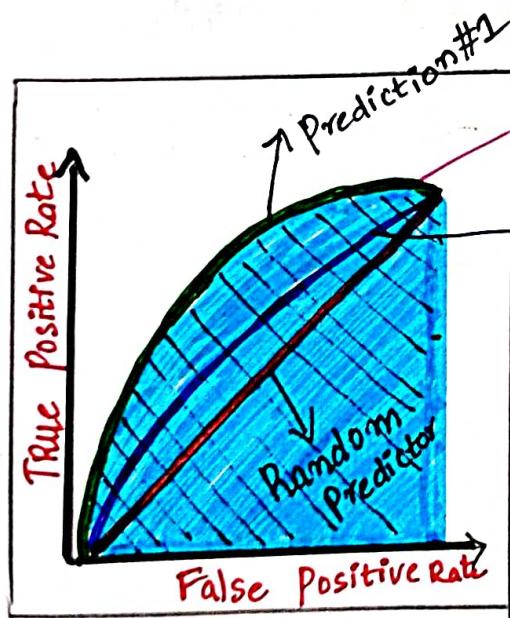


False Positive Rate =  $1 - \text{specificity} = 1 - TNR$

\* ROC curve is a metrics between binary (0 or 1) classes that assess model ability to distinguish

- \* ROC curve is created by plotting the true positive rate (TPR) against the False Positive Rate [FPR] at various threshold settings
- \* True positive rate is also known as Sensitivity or detection in Machine Learning.
- \* False positive rate is also known as the probability of False alarm and can be calculated as  $[1 - \text{specificity}]$

# AUC [Area Under Curve]



Best model!

# the light blue area represents area under curve of receiver operating characteristic (AUROC)

# Higher the AUC, the better the model is at predicting

18/04/22  
11:30 pm

5/4/22  
11:00 AM

2  $\Rightarrow$  Feature Scaling  $\Rightarrow$  Applicable only for **Continuous data**

Feature Scaling ?

Ans:- it refers (or) Techniques used to **normalize** the ranges in our data. (or)

of **Independent Variables**  
**input variables**

\* The methods to set the features value range within a **similar scale**.

\* Variables with bigger magnitude / larger value range dominate over those with smaller magnitude / value range.

Ex:- 10,000,000  
if we take  
10  
10  $\rightarrow$  Both are equal importance

\* Scale of the features is an important consideration when building **Machine Learning models**.

\* Feature scaling is generally the **last step** in the data **pre processing pipeline**, performed just before **Training the machine learning algorithms**.

5. Gradient descent **converges faster** when **Features** are on **similar scales**

Example :-

| C.C  | Mileage |
|------|---------|
| 1600 | 14      |
| 1800 | 15      |
| 2200 | 16.5    |
| 2100 | 18      |
| 2600 | 22.5    |
| 1800 | 19.4    |
| 1900 | 25.4    |

Continuous Data

\* Feature Scaling \*

of a car

a. When we ask Machine which is important in the both columns?

\* Machine is giving importance to C.C

Because it is having large values.

\* But, we know both are important

So, we have train's model in <sup>Initially</sup> such that,

it should consider both the columns similar

So, we reduce the value

$$= \frac{800}{1000} = \frac{80}{100} = \frac{8}{10} = \frac{4}{5} \quad (\text{Every one is same})$$

We are scaling down  
Dividing every value with

"10" In this case.

\* To reduce the value.

\* Feature Scaling

Importance in  
Some ML Algorithms

### 1. Linear Regression & logistic Regression

\* The regression coefficients of linear models are directly influenced by scale of the variable.

### 2. Support Vector Machines :-

\* Feature scaling helps decrease the time to find support vectors for SVM's

### 3. K-means clustering

\* Euclidean distances are sensitive to feature magnitude

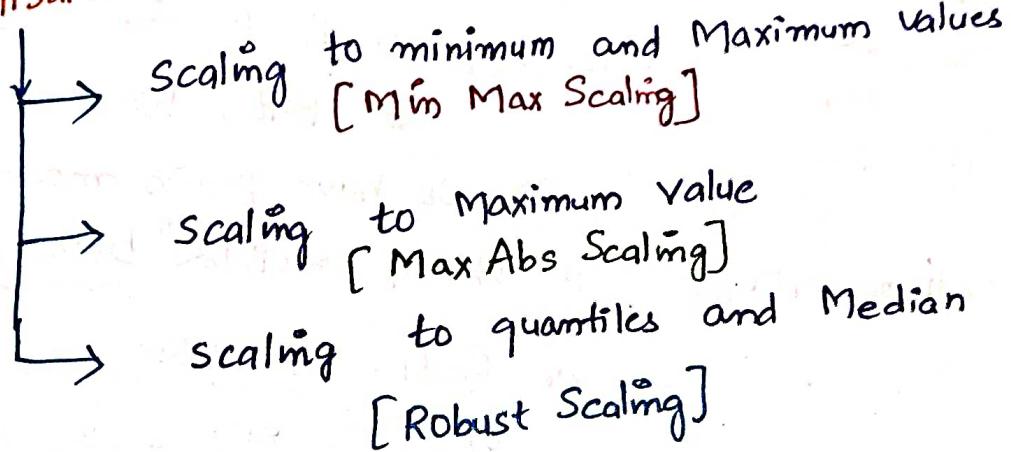
### 4. Principal Component analysis (PCA) :-

\* PCA requires the features to be centered at "0".

## \* Various Feature Scaling Techniques :-

### 1. Standardisation

### 2. Normalisation Technique



```
# import pandas as pd
```

```
# import matplotlib.pyplot as plt
```

```
%matplotlib inline
```

→ Using (titanic Data Set)

code:  
# titanic = pd.read\_csv("titanic.csv", usecols = ["Age"])

```
titanic.head()
```

Out

| Age  |
|------|
| 22.0 |
| 38.0 |
| 26.0 |
| 35.0 |
| 35.0 |

# check Null values in "Age" column.

# titanic . isnull() . sum()

[Out] Age 177 → Null values

# Replacing with "Median Value" For Null values

# titanic ["Age"] . fillna (titanic ["Age"] . median(),  
inplace = True)

# titanic ["Age"] . isnull() . sum()

[Out] : Age 0 → zero Null values



\* Standardisation

Converting Each Value To Zscore

Ex :-

| X  | $\frac{x-\mu}{\sigma} = Z$ score |
|----|----------------------------------|
| 23 | $\frac{23-60}{2} = -18.5$        |
| 45 | $\frac{45-60}{2} = -7.5$         |
| 67 | $\frac{67-60}{2} = 3.5$          |
| 89 | $\frac{89-60}{2} = 14.5$         |
| 78 | $\frac{78-60}{2} = 9$            |

These Five values converting To Zscores  
is called as "standard" scaling

Avg / mean = 60.

(std)  $\sigma = \sqrt{\frac{\sum (x - \mu)^2}{n}}$

Code :-

```
From sklearn.preprocessing import StandardScaler
```

# sc = StandardScaler() # call The Function (ShortCut)

# sc.fit\_transform(titanic[["Age"]]) # fit\_transform

# titanic[["Age-SC"]] # creating new column

Out :-

|     | Age-SC  |
|-----|---------|
| 0   | -0.5651 |
| 1   | 0.6638  |
| 2   | -0.2583 |
| 3   | 0.43312 |
| ..  |         |
| 890 | 0.20272 |

# titanic

Out :-

|     | Age  | Age-SC  |
|-----|------|---------|
| 0   | 22.0 | -0.5651 |
| 1   | 38.0 | 0.6638  |
| 2   | 25.0 | -0.2583 |
| ..  |      |         |
| 889 | 26.0 | -0.2583 |
| 890 | 32.0 | 0.20272 |

# histogram of  
Original ["Age"]

# titanic[["Age"]].hist()

plt. show

# Here Every Value is  
converted To Zscore.

$$\left[ \frac{x - \mu}{\sigma} \right] = Z \text{ score.}$$

EX:- Age [22] # First record  $x = 22$

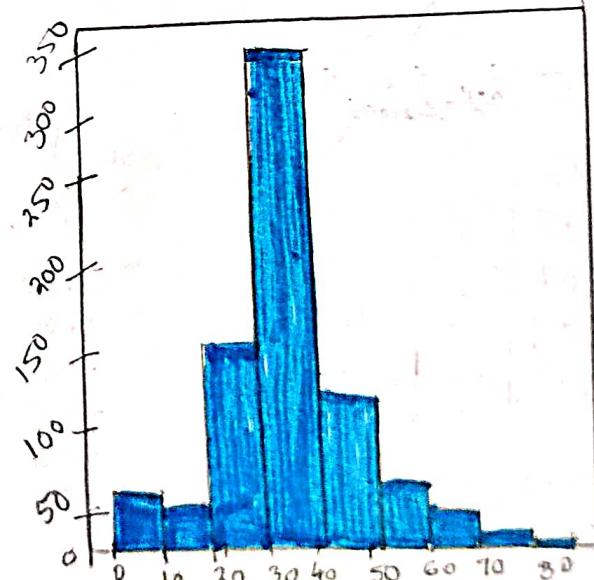
# titanic.mean()  $\mu = 29.361$

# titanic.std()  $\sigma = 13.01$

Original Data "Age-SC"  $\rightarrow [-0.5651]$

$$\left[ \frac{22 - 29.361}{13.01} \right]$$

Converted To Z-Score

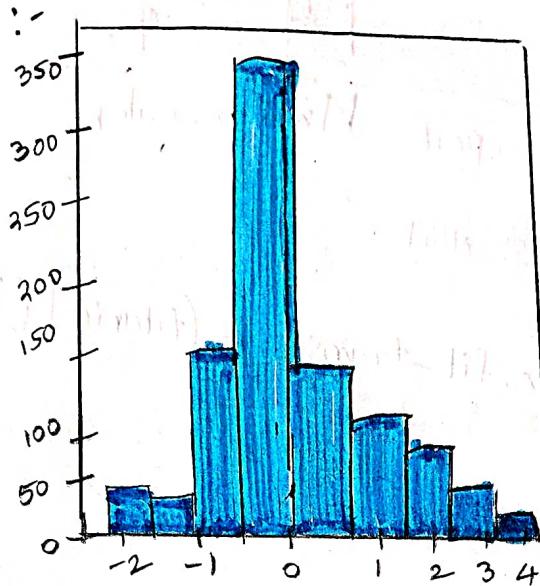


# Histogram of Converted "Age"

# titanic[ "Age\_sc" ].hist()

# plt.show()

Out :-



# Here, we see, Histogram is not going to change, But, Values does.

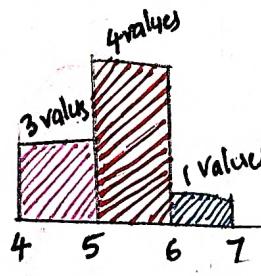
Example :-

\* How ? Histogram Remains Same, But changed Values

Ans :-

Ex :-

| X  | Changed X |
|----|-----------|
| 40 | 4         |
| 50 | 5         |
| 42 | 4.2       |
| 43 | 4.3       |
| 54 | 5.4       |
| 58 | 5.8       |
| 59 | 5.9       |
| 62 | 6.2       |



# Here only "X" Values Only changes But Histogram Remains Same.

## II # Normalisation

~~~~\* ~~~

### \* MinMax Scaling

→ MinMax Scaling Scales The Values between "0" to "1" (Every Thing is one value) No -ve value

Ex:-

| X  |                        |
|----|------------------------|
| 65 | $\frac{65-34}{53}$     |
| 45 | $\frac{45-34}{53}$     |
| 34 | $\frac{34-34}{53} = 0$ |
| 67 | $\frac{67-34}{53}$     |
| 67 | $\frac{67-34}{53}$     |
| 87 | $\frac{87-34}{53} = 1$ |

$$\frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

MinMaxScaler

Code: From sklearn.preprocessing import

# min\_max = MinMaxScaler() # Shortcut

# titanic["Age\_mm"] = min\_max.fit\_transform(titanic[["Age"]])

fit\_transform

creating new column

# titanic (calling/show data)

Out :-

|   | Age  | Age_sc  | Age_mm |
|---|------|---------|--------|
| 0 | 22.0 | -0.5637 | 0.2911 |
| 1 | 38.0 | 0.6487  | 0.472  |
| 2 | 26.0 | 0.259   | 0.32   |
| 3 |      |         |        |
| 4 |      |         |        |
| 5 | 26.0 | -0.25   | 0.32   |
| 6 | 32.0 | 0.20    | 0.39   |

### \* Robust Scaling

$$\frac{X - X_{\text{median}}}{IQR}$$

$$\therefore IQR = Q_3 - Q_1$$

## Code:

From sklearn.preprocessing import RobustScaler

# rs = RobustScaler() → shortcut()

# titanic["Age-rs"] = rs.fit\_transform(titanic[["Age"]])

titanic["Age-rs"] (or) = RobustScaler().fit\_transform(...)

If not using any shortcut like rs = RobustScaler()

# titanic

Out

| Age  | Age-SC | Age-MM | Age-RS |
|------|--------|--------|--------|
| 22.0 | 0.565  | 0.461  | 0.4615 |
| 38.0 | 0.66   | 0.769  | 0.769  |
| 26.0 | 0.25   | -0.153 | -0.153 |
| ...  |        |        |        |
| 26.0 | -0.25  | 0.153  | 0.153  |
| 32.0 | 0.2    | 0.396  | 0.3076 |

## \* Max Abs Scaling

$$\frac{x}{x_{\max}}$$

Ex:-

| x  |           |
|----|-----------|
| 68 | 68/80     |
| 44 | 44/80     |
| 52 | 52/80     |
| 69 | 69/80     |
| 80 | 80/80 = 1 |

Max. value

## Code

From sklearn.preprocessing import

MaxAbsScaler

# mas = MaxAbsScaler()

# titanic["Age-mas"] = mas.fit\_transform(titanic[["Age"]])

# titanic

Out :-

|     | Age | Age - sc | Age - mm | Age - rs | Age - mas | Original Variable |
|-----|-----|----------|----------|----------|-----------|-------------------|
| 0   | 1.0 | 0.2750   | 0.4950   | 0.3250   | 0.4000    | Standard Scaler   |
| 1   | 1.0 | 0.2750   | 0.4950   | 0.3250   | 0.4000    | Min max Scaler    |
| 2   | 1.0 | 0.2750   | 0.4950   | 0.3250   | 0.4000    | Robust Scaler     |
| 889 | 1.0 | 0.2750   | 0.4950   | 0.3250   | 0.4000    | Max Abs Scaler    |
| 890 | 1.0 | 0.2750   | 0.4950   | 0.3250   | 0.4000    |                   |

891 x 5 col

Original Variable  
Standard Scaler  
Min max Scaler  
Robust Scaler  
Max Abs Scaler

5/04/22  
4:58 pm.

Example :- Discretization

Continuous Data

|      |
|------|
| 12.5 |
| 22.8 |
| 25.2 |
| 18.3 |
| 12.4 |
| 15.8 |
| 23.3 |

⇒ Mileage of a Car (Discrete, categorical)

$\leq 15 \rightarrow$  low. mileage

$15 - 20 \rightarrow$  Avg. mileage

$20 - 25 \rightarrow$  good. mileage

$\geq 25 \rightarrow$  very good

By our  
own choice  
To divide the  
into parts