

# Machine learning Algorithms

STUDY GUIDE

SUBIN VASANTHAN

## Table of Contents

<b>1.</b>	<b><i>Overview of Machine Learning</i></b>	2
<b>2.</b>	<b><i>Types of Data</i></b>	3
<b>3.</b>	<b><i>Supervised Learning / Model &amp; Un- Supervised Learning / Model</i></b>	3
a.	<b><i>Supervised Learning / Model</i></b>	3
b.	<b><i>Un supervised Learning</i></b>	3
<b>4.</b>	<b><i>Regression problem and Classification problem</i></b>	4
a.	<b><i>Regression problem</i></b>	4
b.	<b><i>Classification problem</i></b>	4
<b>5.</b>	<b><i>Building a Model</i></b>	5
<b>6.</b>	<b><i>Machine Learning Algorithms</i></b>	7
a.	<b><i>SUPERVISED MODELS</i></b>	7
i.	<b><i>Linear Regression Algorithm – (Supervised Regression)</i></b>	7
ii.	<b><i>Logistics Regression</i></b>	18
iii.	<b><i>Decision Trees</i></b>	27
iv.	<b><i>K-Fold Validation for GridSearchCV Function</i></b>	38
v.	<b><i>SVM-Support Vector Machines</i></b>	40
b.	<b><i>UN-SUPERVISED MODELS</i></b>	43
i.	<b><i>K- Means</i></b>	44
ii.	<b><i>DB-Scan</i></b>	48
iii.	<b><i>Hierarchical Clustering</i></b>	49

## 1. Overview of Machine Learning

Machine Learning

- Field of making machine learn from Data ×
- Field of writing algorithms that enable computer to learn from the data and make a decision using it !!

## Agenda

Tuesday, 28 September 2021 8:05 AM

- Overview/Introduction to Machine Learning
- Machine Learning Algorithms
  - Linear Regression - Supervised Reg
  - Logistic Regression - Sup Class
  - Decision Trees - Sup Reg&Class
  - Random Forest - Sup Reg&Class
  - Support vector Machines - Sup Reg&Class
  - K-Means - UnSup
  - DB Scan - UnSup
  - Hierarchical Clustering - UnSup

Tuesday, 28 September 2021 9:07 AM

Type of data

- Training data
- Validation data
- Testing data

Type of problem

- Regression problem
- Classification problem

[the type of target column]

Type of approach

- Supervised approach
- Unsupervised approach

[presence / absence of target column in the training dataset].

## 2. Types of Data

Type of data

- Training data → Data using which the model is trained!
- Validation data / dev data
- Testing data → Data on which the model is tested !!

More the training data

&

more the variance in the training data

↳ helps in building a  
really intelligent  
model !!

## 3. Supervised Learning / Model & Un- Supervised Learning / Model

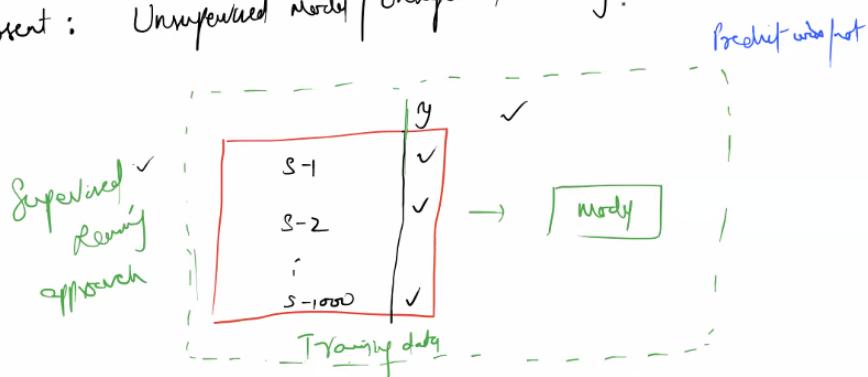
### a. Supervised Learning / Model

If the training data contains some target results for the scenarios given it is called supervised learning

on the basis of target value:

Present : Supervised model | Supervised learning ✓

Absent : Unsupervised model | Unsupervised learning ✓



### b. Unsupervised Learning

If the training data does not contain any target results for the scenarios given it is called unsupervised learning

## 4. Regression problem and Classification problem

Machine Learning can solve 2 kinds of Problems

### a. Regression problem

→ Build a model that can predict what should be the price of house

Target is numeric continuous column that is a regression problem

Problem:

Loca	# bed	# bath	Gated	Target	
				House	House
House	1	1	No	10000	
House	1	2	Yes		15000
:	:	:	:	:	:

→ Model

Supervised model to solve regression problem !!

Given the above scenarios the model will learn the Target values.

### b. Classification problem

Classification Problem  
objective: Build a ML model that can predict if a customer would default on a loan given his financial details

The target values will be categorical that is 1's or 0's whether he will default or not default. This is called classification problem

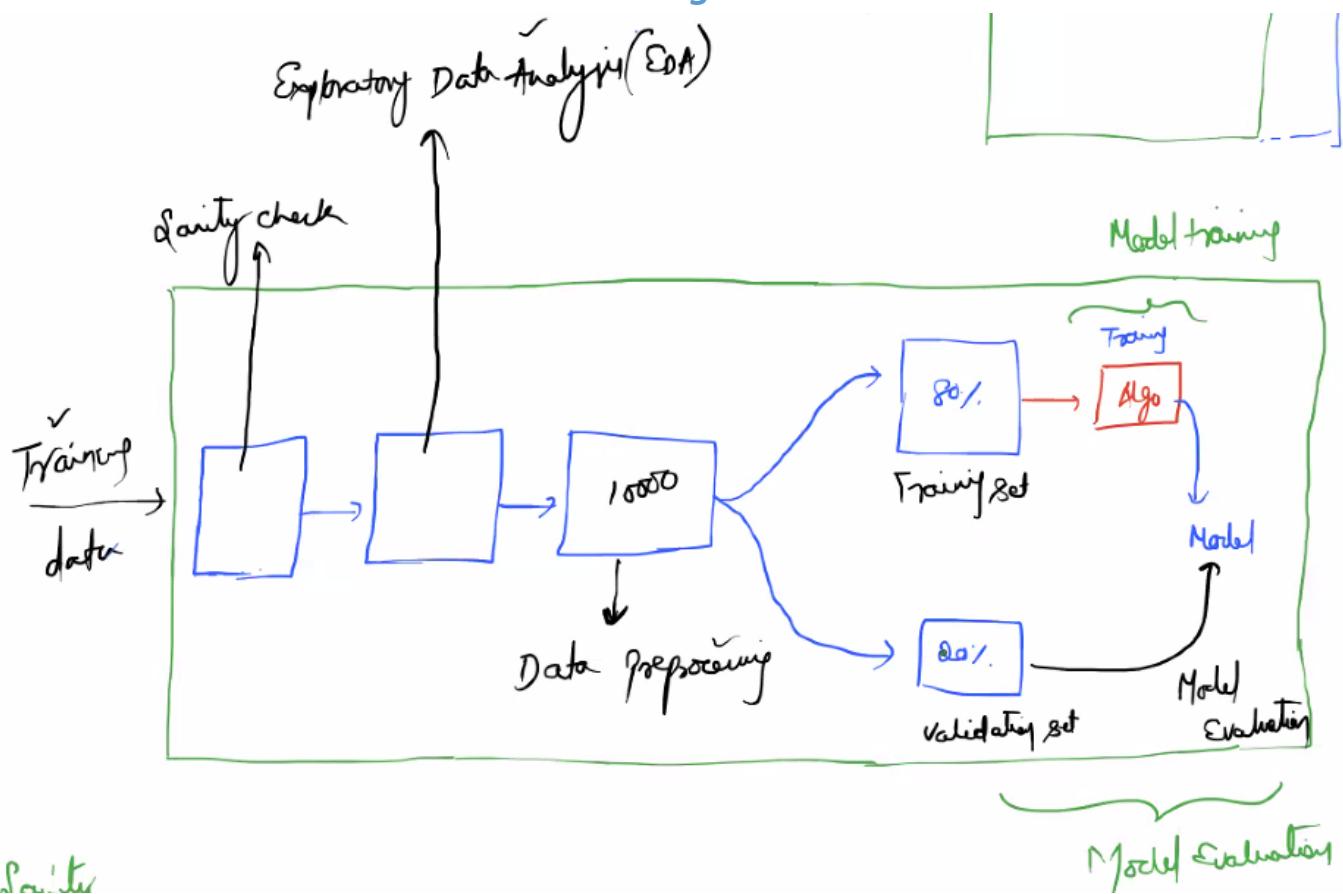
Target :- 1 | 0

The classification problem can also have more than 2 values.

Like ratings

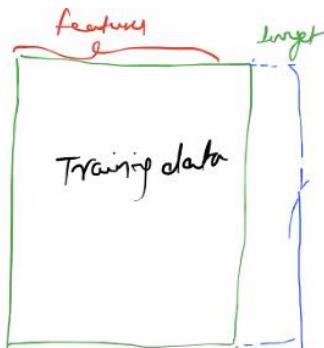
Target :- 5, 4, 3, 2, 1 (Numeric discrete column)

## 5. Building a Model



Sanity

- Do I have enough observations/ good quantity of data to build a model
- Is the target column present on the training data & what kind of column it is
- Are the features are logically related to the target !!

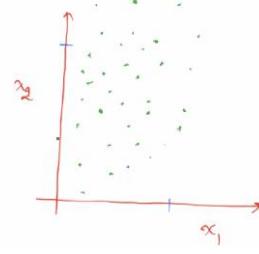


## Exploratory Data Analysis

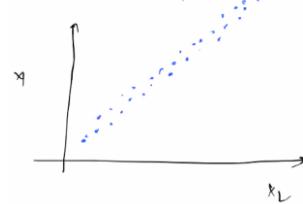
- Are there missing values present in the data
- Feature composition of the data
- Are there any outliers in the data
- Distribution of each of the features
- Scale of the features
- Correlation b/w features & the target

### Outliers

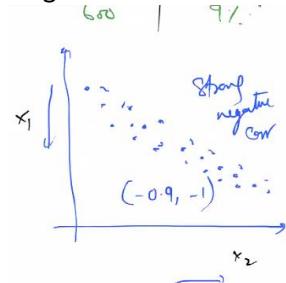
$x_1$	$x_2$	$y$
84	36	1
6	32	0
16	19	1
1	0	0



Positive correlation



Negative correlation



Zero correlation

The points that are significantly different are called **outliers** like the one red and blue in the above plot

Correlation is a method to measure the linear relationship between two numeric continuous columns

$$-1 \leq \text{Correlation} \leq 1$$

Correlation = 1 (very strongly positively related)

Correlation = -1 (very strongly negatively related)

Correlation = 0 (there is no relationship b/w the columns)

Between Feature  $\square$  Low correlation is desired

Between Feature and Target  $\square$  High Correlation is desired

## Data Preprocessing:

At this stage we treat all the problems like outliers, missing data etc. encountered at the EDA

## 6. Machine Learning Algorithms

### a. SUPERVISED MODELS

#### i. Linear Regression Algorithm – (Supervised Regression)

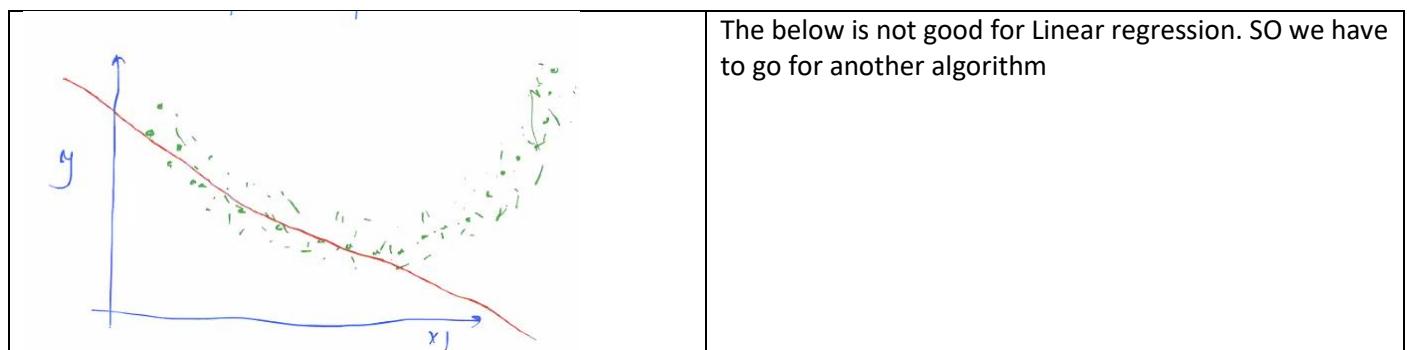
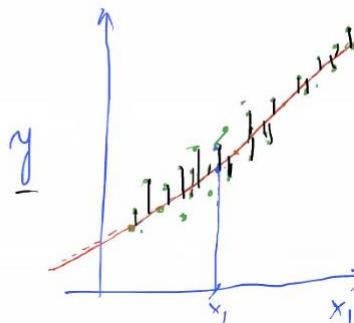
##### Linear Regression

- Linear Regression is a Supervised Technique / algorithm
- Linear Regression is used to solve Regression Problem

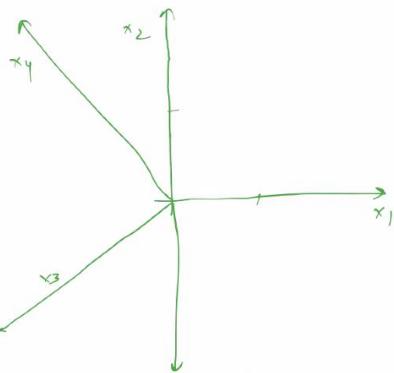
##### Linear Regression

- Linear Regression is a Supervised Regression algorithm that can find out  
[Linear relationship] b/w features & target !!

- find a line/plane that gives the least Error



$$\begin{array}{c|c|c|c|c} x_1 & x_2 & \dots & x_n & y \\ \hline \checkmark & \checkmark & \dots & \checkmark & \checkmark \end{array}$$



In real time there will be many dimensions that is many inputs for a target.

→ Linear Regression data  
→ Compute the Error

$$\text{Mean error} = \frac{1}{n}$$

If the error is low, we will use linear regression otherwise if the error is big we will choose other algorithms

### Linear Regression Equation

Linear Regression

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

(Linear Equation)  
b/w features & target

Coefficients / parameters / weight

objective of linear regression algorithm is to find the value of weights that corresponds to a line/plane such that the errors are minimum.

## Learning Steps

Linear Reg. Equation

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Learning Steps

Step-0: Randomly assign the values of  $\beta_0, \beta_1, \beta_2$

$$\beta_0 = 2, \beta_1 = 10, \beta_2 = 3 \quad (\hat{y} = 2 + 10x_1 + 3x_2)$$

Step-1: Using the current equation, predict on the training set

$x_1$	$x_2$	$y$	$\hat{y}$
1	1	6	15
2	2	8	28
3	3	9	41

Step-2: Compute the loss (error) due to the current prediction

$x_1$	$x_2$	$y$	$\hat{y}$	Error
1	1	6	15	9
2	2	8	28	20
3	3	9	41	32

$$\text{Loss func} = \text{Mean squared Error} = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

$x_1$	$x_2$	$y$	$\hat{y}$	Sum Error
1	1	6	12	$(12-6)^2 = 36$
2	2	8	20	$(20-8)^2 = 144$
3	3	9	30	$(30-9)^2 = 441$

Mean ✓

Step-3: Magic it will see the current error (due to the current set of  $\beta$ 's) and update the values of  $\beta$  such that the error reduces

$$(\beta_0 = 1.8, \beta_1 = 5, \beta_2 = 2.3)$$

After this it will go to step-1 and repeat this until the model is improved to give a least value of error.

The Word Magic used in Step 3 is called **Optimizers**

Below is how we optimize

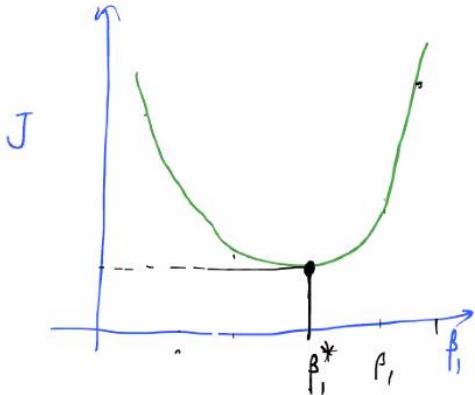
Thursday, 30 September 2023

9:31 AM

$$\hat{y} = \beta_0 + \beta_1 x_1$$

$$\begin{aligned}
 J &= \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2 \\
 &= \frac{1}{m} \sum_{i=1}^m (y_i - \beta_0 - \beta_1 x_{1i})^2 \\
 &= \frac{1}{m} \sum_{i=1}^m \left( y_i^2 + \beta_0^2 + \beta_1^2 x_{1i}^2 - 2 y_i \beta_0 - 2 y_i \beta_1 x_{1i} \right) \\
 &= \frac{1}{m} \left[ \sum_{i=1}^m y_i^2 + \sum_{i=1}^m \beta_0^2 x_{1i}^2 - \sum_{i=1}^m 2 y_i \beta_0 - \sum_{i=1}^m 2 y_i \beta_1 x_{1i} \right] \\
 &= \frac{1}{m} \left[ k_1 + \beta_0^2 \sum_{i=1}^m x_{1i}^2 - 2 \beta_0 \sum_{i=1}^m y_i x_{1i} \right] \\
 &= \frac{1}{m} \left[ k_1 + \beta_0^2 k_2 - 2 \beta_0 k_3 \right] \\
 J &= \left[ \beta_0^2 c_1 - \beta_0 c_2 + c_3 \right]
 \end{aligned}$$

This equation is similar to  $y = ax^2 + bx + c$  which is the equation of a parabola



So here we have the best value of  $\beta$ , where the error is low.

The formulae to get this is

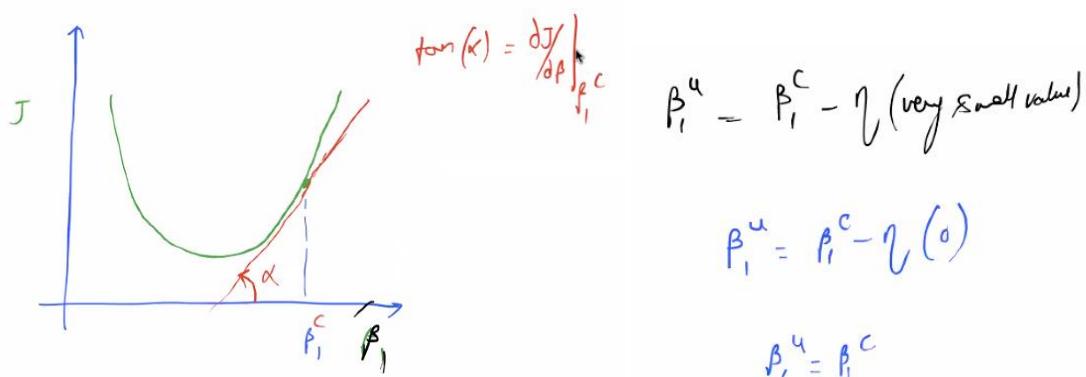
[Optimizer: Gradient Descent Algorithm](#)

## Gradient descent

$$\beta_i^u = \beta_i^c - \eta \left( \frac{\partial J}{\partial \beta_i} \right) \Big|_{\beta_i^c}$$

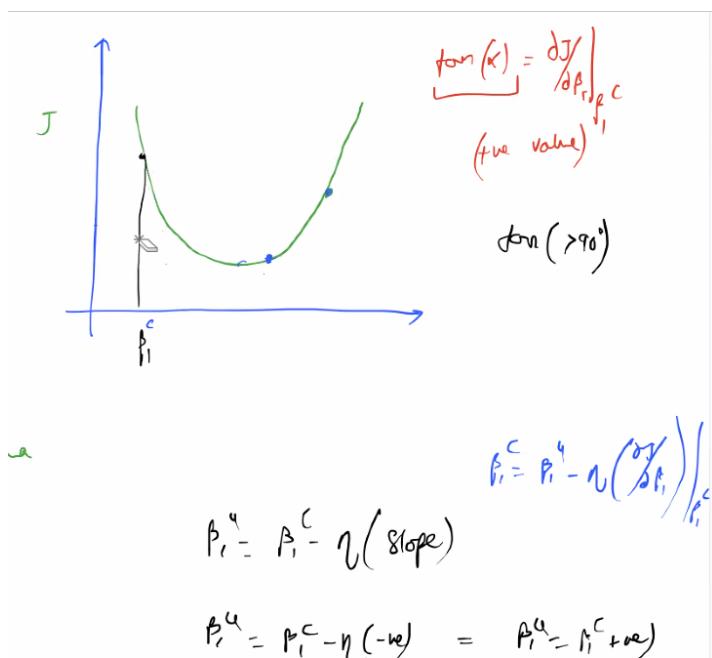
Gradient descent function  
will help me to slide down the  
Cost function & search minimum

In the above  $\frac{\partial J}{\partial \beta}$  is the derivative of  $J$ . Derivative is the angle that a value makes on the x axis.



We update the values until the tan of the value is close to zero.

If the  $\beta$  value chosen is on the left of the parabola,  $\tan$  of the angle will be  $-ve$ . So in the equation the  $\beta$  will be increased to reach the bottom of the curve.



The above is explained using 2 parameters for the ease of explaining

But in real time we will have many parameters so in this case we will use **gradient descent algorithm** for all the values.

$$\beta_0^u = \beta_0^c - \eta \left( \frac{\partial J}{\partial \beta_0} \right) \Big|_{\beta_0^c}$$

$$\beta_1^u = \beta_1^c - \eta \left( \frac{\partial J}{\partial \beta_1} \right) \Big|_{\beta_1^c}$$

$$\beta_2^u = \beta_2^c - \eta \left( \frac{\partial J}{\partial \beta_2} \right) \Big|_{\beta_2^c}$$

Optimizer  $\rightarrow$  Gradient descent

$$\boxed{\beta_j^u = \beta_j^c - \eta \left( \frac{\partial J}{\partial \beta_j} \right) \Big|_{\beta_j^c}}$$

Slope of the  
Cost function  
in direction  
 $\beta_j$

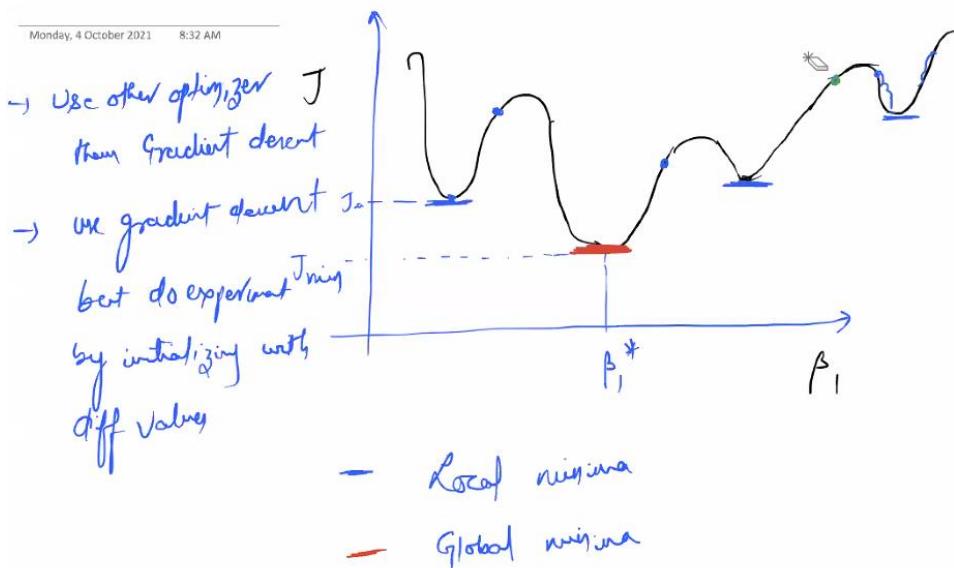
Learning rate (typically  $\eta$  is around  $10^{-3}$ )

$\eta \rightarrow$  Learning rate  $\rightarrow$  Controls the update that is made to the Parameter

Advantage of  $\eta \rightarrow$  we gradually come to minima

Disadvantage of  $\eta \rightarrow$  we will have to do multiple iteration before minima is reached !!

The above is good for a linear regression equation, but in case of other functions the curve may not be always a inverted parabola. It can be wriggly curve as below



Below is just an extra

Compute loss

$$\hat{y}_{(s0,)} = \text{feature}_{(s0 \times 3)} \times \text{weights}_{(3,)}$$

$$\text{Loss} = \frac{1}{m} \cdot \eta \cdot \text{sum} \left( \left[ \text{target}_{(s0,)} - \hat{y}_{(s0,)} \right]^{**2} \right)$$

Gradients

$$\begin{bmatrix} \frac{\partial J}{\partial \beta_0} \\ \frac{\partial J}{\partial \beta_1} \\ \frac{\partial J}{\partial \beta_2} \end{bmatrix} = -\frac{2}{m} * \left( \text{feature}_{(3, s0)}^T \times (\text{target} - \hat{y})_{(s0,)} \right)$$

$$= \text{vector} = (3, )$$

$$\text{weights}_{(3,)} = \text{weights}_{(3,)}^* - \eta \times \text{feature}_{(3,)}$$

```

features_df = features.copy(deep = True) ### (50,3)
targets = y ### (50,)
weights = np.random.random(size = (3,)) ##### Step-0 :random initialization ### (3,)

def prediction(weights, features_df):
    predicted_vals = np.dot(features_df,weights) ## This will be an vector which contains predicted values for all obs in features_df, shape = (50,)
    return (predicted_vals)

def error (targets,predicted):
    error_val = (targets-predicted) ### (50,)
    return (error_val)

def cost_val(error_val):
    cost = np.mean(error_val**2) ##### mean squared error value ### scalar
    return (cost)

##### pred_y = w0*x0 + w1*x1 + w2*x2

##### cost_funct = (y - (w0*x0 +w1*x1 +w2*x2))^2

##### gradient (cost_funct)/w0 = 2 * (y- w0*x0 + w1*x1 + w2*x2) * x0

def gradients(error_val,features_df):
    temp_gradients = np.dot(features_df.T, error_val) ### vector of shape (3,)
    #error_val = np.reshape(error_val,newshape = (1,50))
    #temp_gradients = np.dot(error_val,features_df) ### (1,3) ###[grad(w0), grad(w1), grad(w2)]
    #temp_gradients = temp_gradients.T #### (3,1)
    temp_gradients = temp_gradients*(-2)

    temp_gradients = temp_gradients/(features_df.shape[0]) ## (3,1)
    grads = temp_gradients
    #grads = np.reshape(temp_gradients,-1) ###(3,)
    return(grads)

```

```

temp_gradients = temp_gradients/(features_df.shape[0]) ## (3,1)
grads = temp_gradients
#grads = np.reshape(temp_gradients,-1) ###(3,)
return(grads)

def linearReg(features_df,targets,weights,learning_rate,n_iter):
    cost_list= [] ### Store error after every update of weights

    for i in range(n_iter):
        ##### Step-1 : Compute prediction ###

        predict_vector = prediction(weights,features_df) ### shape (50,)

        ##### Step -2 : Compute cost #####
        error_vector = error(targets,predict_vector)

        cost_value = cost_val(error_vector) ## Mean squared error for the current predictions

        ##### Step -3 : Gradient descent #####
        weights = weights - learning_rate * gradients(error_vector,features_df)

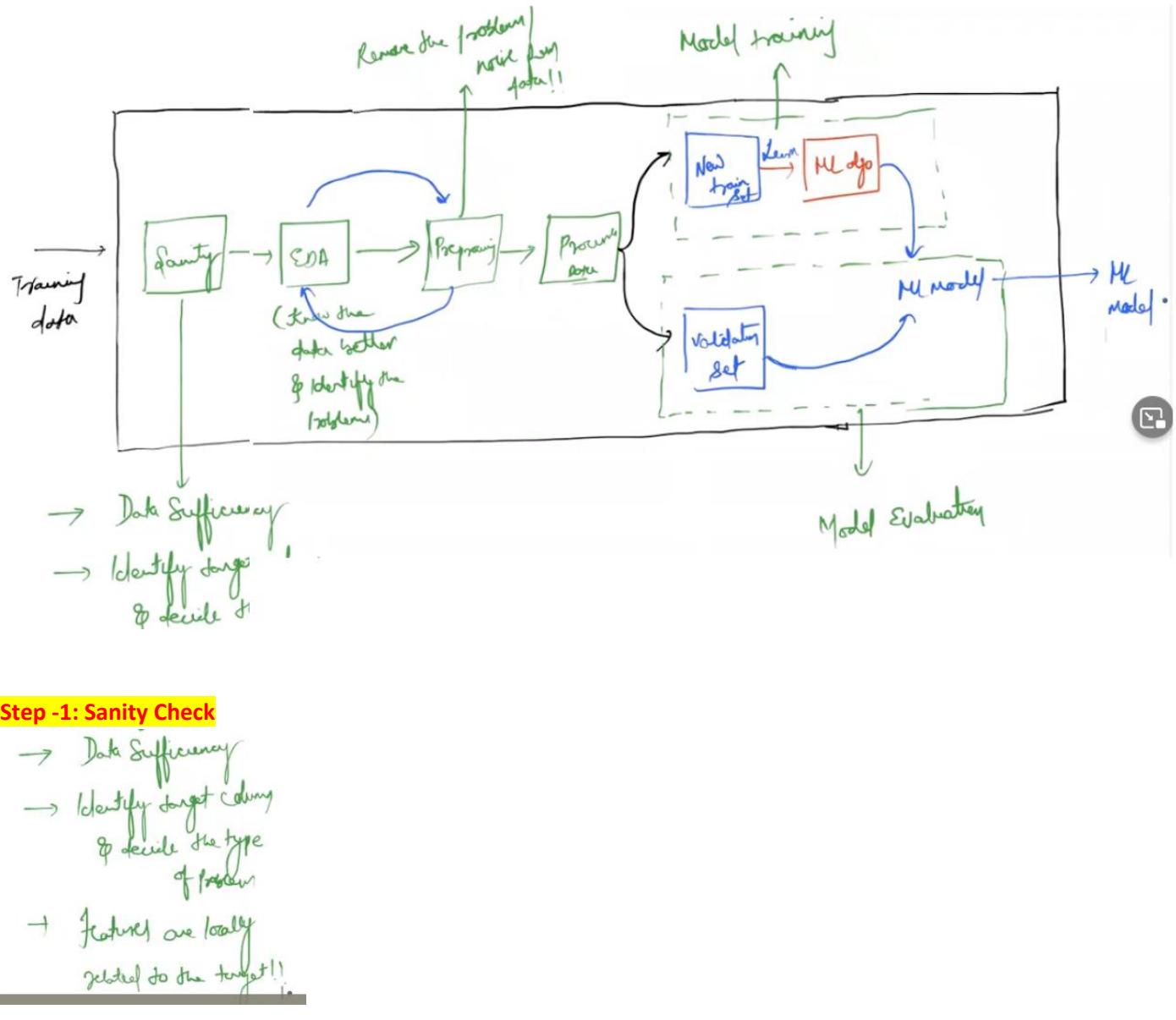
        cost_list.append(cost_value)

    #print("iteration : {} Cost : {}".format(i,cost_value))

    return(weights,error_list)

```

## Implementing linear Regression by solving an End to End Problem



## EoA

- Whether missing values present in the data
- Whether outliers are present in the data
- Look at the distribution of the features
- What is the composition of data
- Correlation between features - feature & target !!

## **Step -3: Pre-processing**

TUESDAY, 2 JULY 2019 04:10 PM

$1 \rightarrow 0.95$

### Preprocessing

- Treat the missing values → Mandatory
- Convert all columns into numeric types → Mandatory
- Treat outliers → optional
- Normalize the scale of the data → optional
- Remove very highly correlated features ( $\geq 0.95/0.90$ ) → optional
- Normalize the data distribution → optional

## Preprocessing

- Convert all the column into numeric types → Mandatory ✓
- Missing value treatment → Mandatory ✓
- outlier treatments → optional
- Normalization of Scale → optional \*
- Normalize the distribution → optional
- Remove very highly correlated features → optional

## **Step -4: Model Evaluation**

## ii. Logistics Regression

Tuesday, 12 October 2021 8:13 AM

### Classification Problem

↳ 2 unique values in the target

✓ → Binary Classification Problem !!

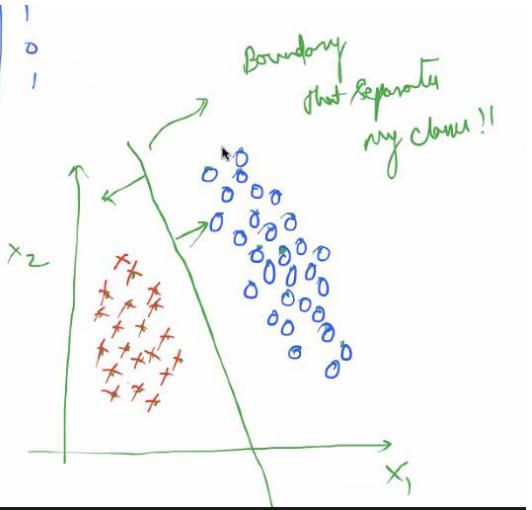
↳ more than 2 unique values

✓ → Multi-class Classification Problem !!

Logistic regression helps us to solve both kinds of problems !!

$x_1$	$x_2$	$x_3$	...	$x_n$	$y$
-	-	-	-	-	0
-	-	-	-	-	1
-	-	-	-	-	0
-	-	-	-	-	1
-	-	-	-	-	2
-	-	-	-	-	1
-	-	-	-	-	0
-	-	-	-	-	1
-	-	-	-	-	2
-	-	-	-	-	1
-	-	-	-	-	2
-	-	-	-	-	1
-	-	-	-	-	2

### i. Binary Classification



### Objective

→ Objective of any classification algorithm  
is to find a function that would  
serve as a boundary that separates  
(Decision boundary)  
the classes in the data !!

## Logistic Regression

- Logistic Regression is a linear classifier
- Linear classifier means that it can only find linear functions for decision boundaries/linear decision boundaries

i.e. Logistic Regression can only identify lines (2D) or planes (3D/higher) as the decision boundaries

→ Logistic Regression is a modification of linear separation technique, to support classification tasks !!

(  
2D linear → line  
3D/higher linear → plane)

## Linear Regression

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

Range:  $(-\infty, \infty)$

### a. Equation

## Logistic Regression

$$\hat{y} = \text{Sigmoid} \left( \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n \right)$$

— Logistic Regression function

Transformation function      Equation of line

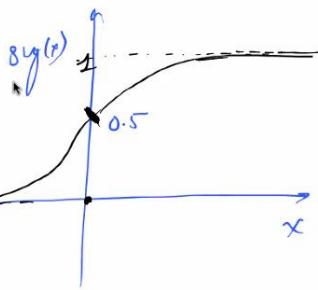
Domain :  $(0, 1)$

Y lies between 0 and 1

Now we will find out the values between 0 and 1 be classified to either 0 or 1.

$$* \hat{y} = \text{sig}(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)$$

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



$$\hat{y} = \text{sig}(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)$$

$$\hat{y} = \frac{1}{1+e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}$$

$$* \text{sig}(x) \rightarrow 0.5$$

$\rightarrow$  Probability that  
the model thinks  
the observation  
is class 1

$$\hat{y} = \text{sig}(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)$$

$$\hat{y} = P(y=1 | x_1, \dots, x_n) = \underbrace{\text{sig}(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}_{\text{Probability of given obs belonging to class 1}}$$

\* Logistic Regression will output probabilities !

\* once we get the probabilities, we can apply some threshold ...

$$\hat{y} = \text{sig}(x) \begin{cases} > 0.5 & \rightarrow 1 \\ \leq 0.5 & \rightarrow 0 \end{cases} \quad \begin{array}{c} \uparrow \\ \text{ideal/different threshold} \end{array}$$

But we can change the thresholds as per the experience

## b. Learning Steps

$$\hat{y} = \text{sig}(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)$$

Step-0: Random Initialization of weights  
ie.  $\beta_0, \beta_1, \beta_2, \dots, \beta_n$  are randomly chosen

Step-1: Predict the probabilities using the current set of  $\beta$ 's

Step-2: Compute the loss/Error in the prediction  
 $\text{loss} = \text{Mean log loss} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)]$

Step-3: Apply an optimizer (Gradient descent) to optimize the values of  $\beta$ 's

### Linear Regression

Loss func: Mean Sq. Error

Logistic Regression

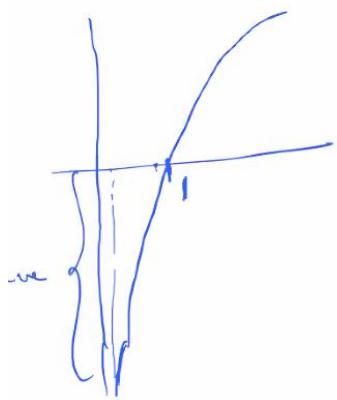
$$\text{Mean log loss} = \frac{1}{n} \sum_{i=1}^n -[y_i \log \hat{y}_i + (1-y_i) \log(1-\hat{y}_i)]$$

Binary Cross Entropy loss

$$\text{Log loss} = -[\underline{y_i \log \hat{y}_i} + \underline{(1-y_i) \log(1-\hat{y}_i)}]$$

$y_i = 1$ $\hat{y}_i \approx 1$ → <del>low loss</del>	$y_i = 1$ $\hat{y}_i \approx 0$ <span style="float: right;">Large loss</span>  $L.L = -(y_i \log \hat{y}_i)$ $= -\left(1 \log(\approx 0)\right)$ $= \text{high value}$	$y_i = 0$ $\hat{y}_i \approx 0$ <span style="float: right;">Low loss</span>  $L.L = -[(1-y_i) \log(1-\hat{y}_i)]$ $= -\left[1 \log(1-\approx 0)\right]$ $= \text{high value}$	$y_i = 0$ $\hat{y}_i \approx 1$ <span style="float: right;"><del>high loss</del></span>  $L.L = -[(1-y_i) \log(1-\hat{y}_i)]$ $= -\left[(1-0) \log(1-\approx 1)\right]$ $= -\left[1 \log(\approx 0)\right]$ $= \text{high value}$
--	---	--	---

Log x graph



### Evaluation metrics

$y_{act}$	$\hat{y}_{pred}$
0	1
1	1
0	0
1	1
1	0
1	1
0	0
1	1

- \* Convert the soft classes to hard classes using some threshold  
 $\rightarrow$  Threshold  $\boxed{0.5}$  i.e. if  $\hat{y} > 0.5 \rightarrow 1$   
 $\qquad\qquad\qquad \hat{y} \leq 0.5 \rightarrow 0$

Metric-1 = 
$$\frac{\text{Total Correct Predictions}}{\text{Total no. of obs.}} = \frac{6}{8} = \frac{3}{4} = 0.75$$

$\hat{y}$	$y$
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
1	0
1	0
1	0
1	0
0	0
0	0
0	0

Accuracy =  $\frac{9000}{10000} = 90\%$

Can be misleading especially in the scenarios where there is  
CLASS IMBALANCE

---

$\hat{y}$	$y$
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
1	0
1	0
1	0
1	0
0	0
0	0
0	0

Accuracy of class 0  $\rightarrow \frac{\text{Total class 0 corr}}{\text{Total class 0 obs.}} = 100\%$

Confusion matrix

		Prediction	
		0	1
Actual	0	9000 ✓	0 ✗
	1	1000 ✗	0 ✓

Given an ideal case one fault

\* Recall =  $\frac{\text{Total no. of } 1's \text{ captured correctly}}{\text{Total no. of } 1's}$

$\checkmark$  Recall = Accuracy of day  $\pm$  =  $\frac{\text{Total no. of } 1's \text{ cap. correctly}}{\text{Total no. of act. } 1's} \times 100\%$

Precision = How many times  
 my predicted 1's are  
 correct =  $\frac{\text{Total no. of } 1's \text{ q. Captured correctly}}{\text{Total no. of } 1's \text{ predicted}} \times 100\% = \text{Nan}$

Recall  $\rightarrow$  out of all 1's ... How many 1's are captured correctly

Precision  $\rightarrow$  out of all predictions  $\rightarrow$  How many are correct  
of 1's

Recall  $\rightarrow 100\%$

Precision  $\rightarrow 100\%$

		0	1	
		9000	0	}
0	0	1000		
1				

(A good model is a model that has high values of both precision & recall)

		Prediction	
		0	1
Actual	0	(True neg)	False positive
	1	False neg	(True positive)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{F.N.}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{F.P.}}$$

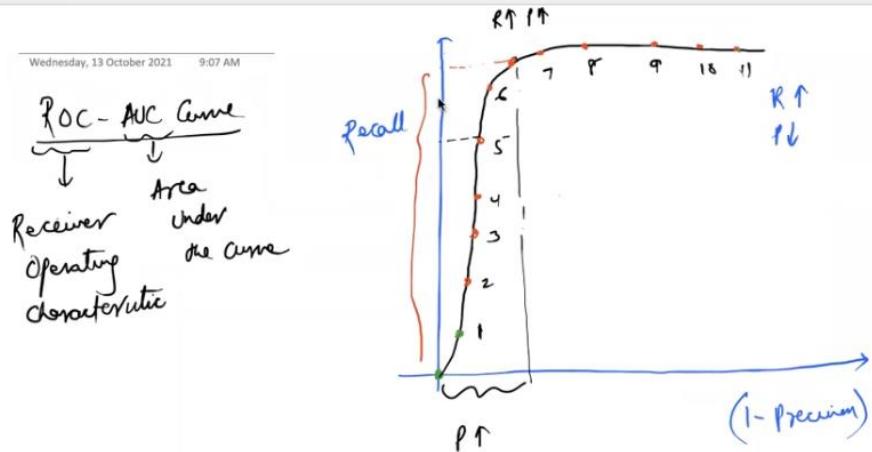
$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

[0, 1]

Any F1 score closer to 1 represents good model, any F1 score closer to 0 represents a bad model!!

Higher the F1 score better the model !!

0.1 → F1 score	$\uparrow$ Consider that value as the threshold that gives you highest F1 score !!
0.2 → F1 score	
0.3 → F1 score	
0.4 → :	
0.5 → :	
0.6 → :	
0.7 → :	
0.8 → :	
0.9 → :	
1.0 → :	



A good threshold is a threshold that gives us highest F1 score

High F1 score means high precision & high recall

## ii. Parameters for logistics regression

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True,
intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto',
verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)
```

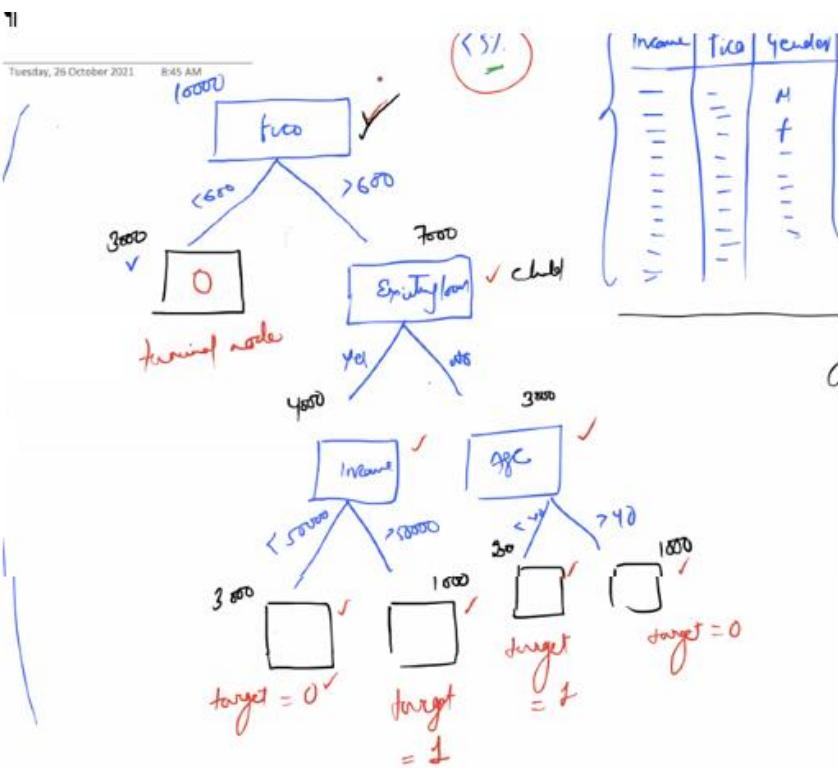
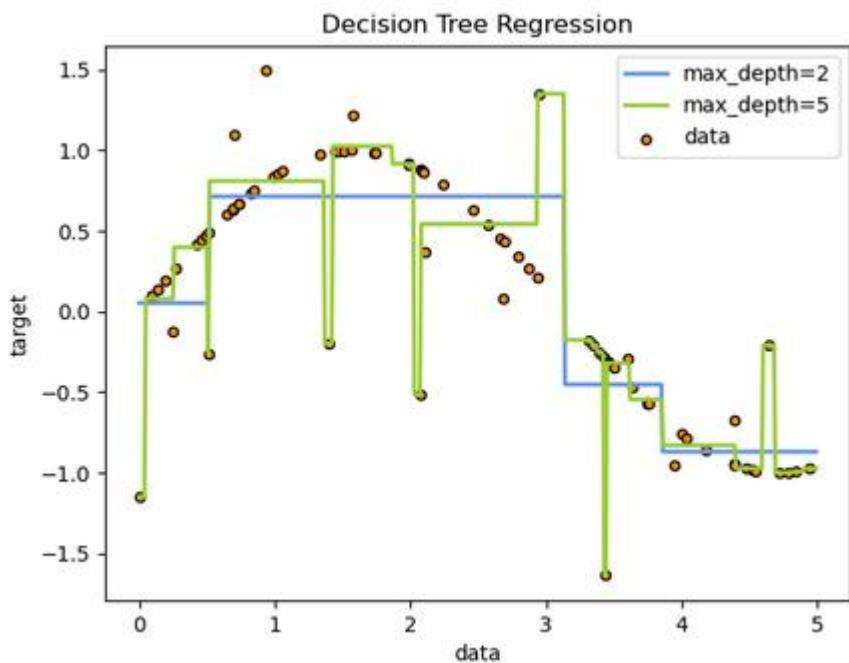
Refer to [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)  
For all the definitions

### iii. Decision Trees

Decision Trees are supervised learning algorithms which can be used for solving both classification and regression tasks. ✓

Unlike logistic and linear regressions, decision trees can find both non linear and linear functions.

Decision Trees is a powerful algorithm when compared to linear and logistic regression.



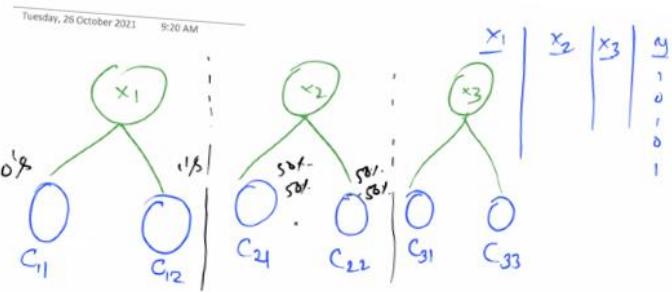
## i. Learning Steps

- 
- Step-1 : Identify the feature to split on
- Step-2 : For every child node generated, evaluate if the node can be split further !
- Step-3 : If the child nodes become terminal nodes assign a label to it
- 

## ii. Learning Questions

- \* How does the algorithm decide the feature to split on? ✓
- \* What are the different criteria that can help decide whether to split the node further or not? ✓
- \* How does a splitting rule get decided for a numeric feature ?? ✓
- \* How does decision tree work for Regression problems ?? ✓

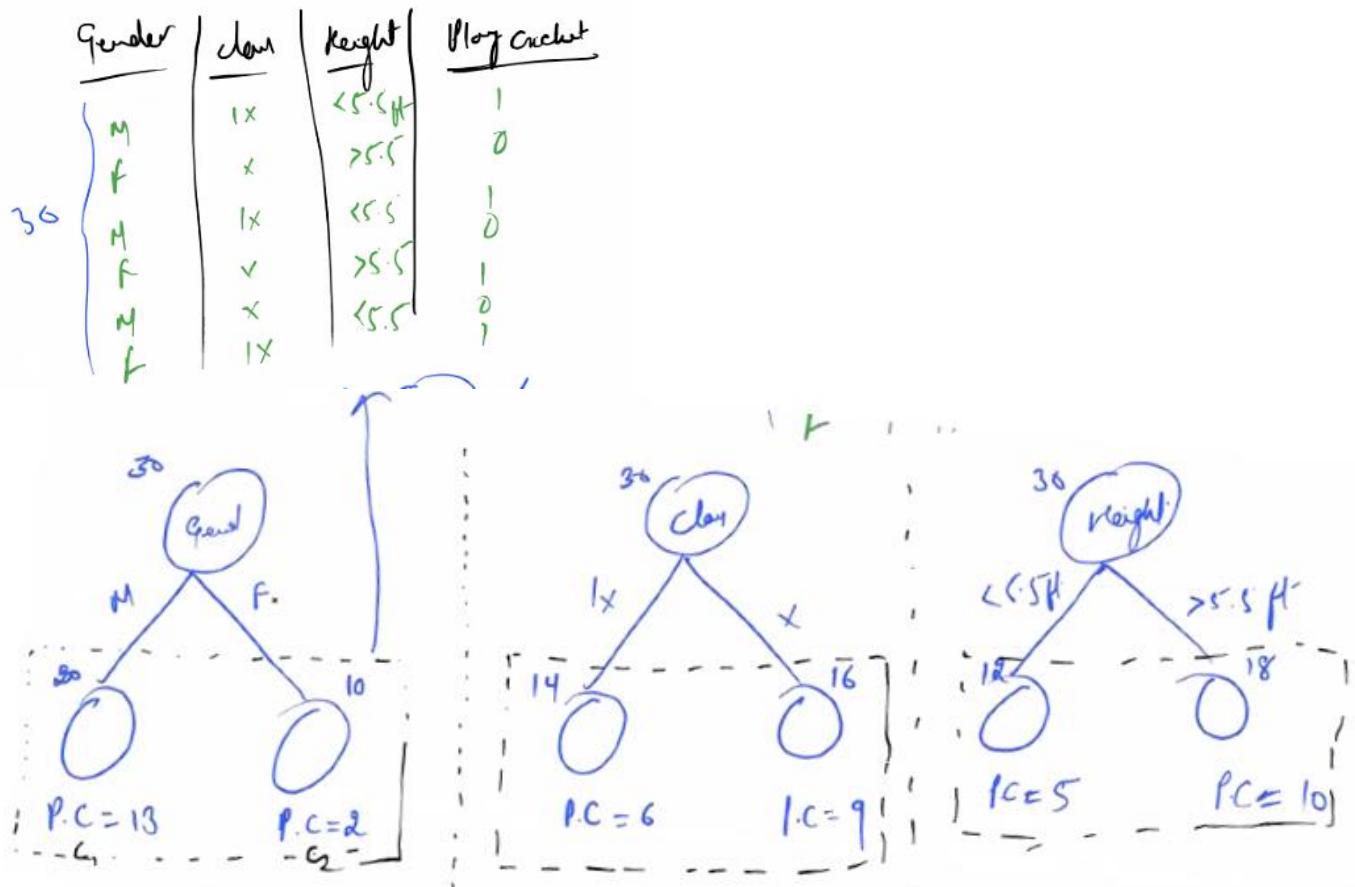
## a. Splitting



- \* It will select that split that separates out the positive & negative classes the best
- \* In other words, it will split on that feature that gives the most homogenous child nodes!!

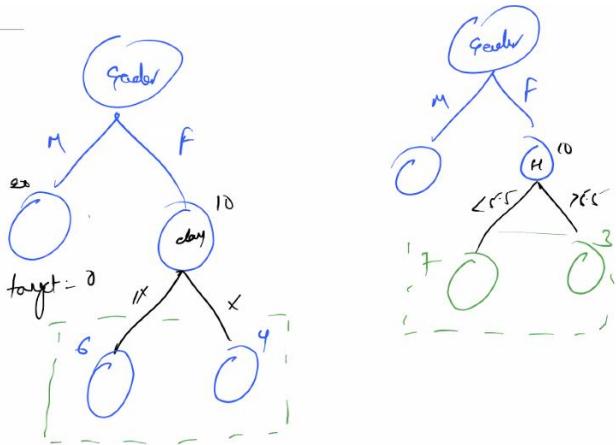
### Gini Score

Gini score is used for determining which is the best feature to split on first  
An example is given below



Gene Score for gender who plays cricket	Gene Score for classr who plays cricket	Gene Score for classr who plays cricket
<p>Gender</p> $G_{in} = p^2 + q^2$ $= \left(\frac{13}{20}\right)^2 + \left(\frac{7}{20}\right)^2$ $= 0.545$ $G_2 = p^2 + q^2$ $= (0.2)^2 + (0.8)^2$ $= 0.68$ $\text{weighted } G_{in} = \frac{20}{30} \times (0.545) + \left(\frac{10}{30}\right) \times (0.68)$ $= 0.588$	$G = \left(\frac{6}{14}\right)^2 + \left(\frac{8}{14}\right)^2$ $= 0.5$ $G_2 = \left(\frac{7}{16}\right)^2 + \left(\frac{9}{16}\right)^2$ $= 0.5$ $\text{weighted } G_{in} = \left(\frac{14}{30}\right) \times (0.5) + \left(\frac{16}{30}\right) \times (0.5)$ $= 0.5$	$G = \left(\frac{5}{12}\right)^2 + \left(\frac{7}{12}\right)^2$ $= 0.5$ $G_2 = \left(\frac{10}{18}\right)^2 + \left(\frac{8}{18}\right)^2$ $= 0.5$ $\text{weighted } G_{in} = 0.5$

Since the gene score is higher for gender, we know where to start the split and then go to the other branch to calculate the gene score and then so on



$$\text{Gene Score} = p^2 + q^2$$

$p \rightarrow \text{Probability of class 1}$   
 $q \rightarrow \text{Probability of class 0}$

Higher the Gene Score, more homogeneous the nodes!

## Entropy

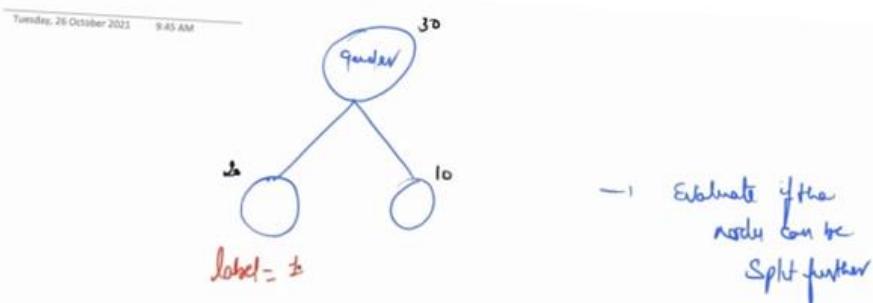
Entropy  $\rightarrow$  is a metric that measures homogeneity of the child nodes !!

$$\rightarrow \text{Entropy} = [-p \log p - q \log q]$$

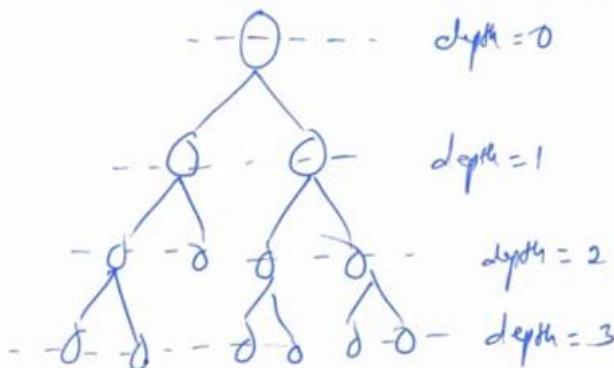
$\rightarrow$  lower the entropy value higher the homogeneity of the node !!

Weighted average will calculate in the same way as the Gini score

### b. Stopping criteria's (Stop Splitting)



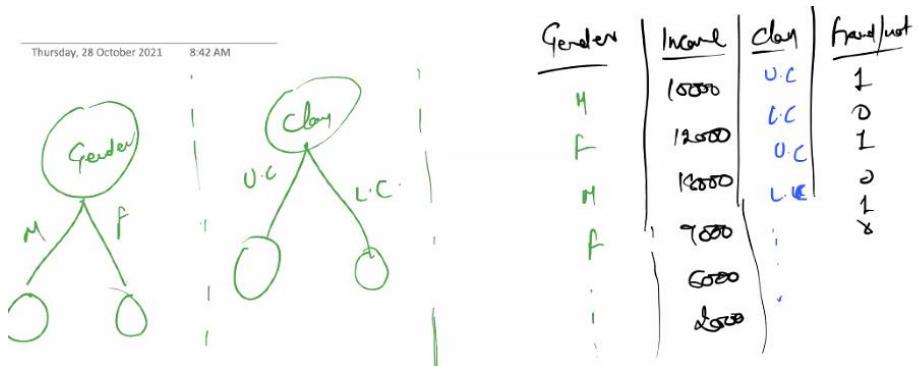
- \* Maximum impurity in a node !!
  - $\rightarrow$  if the minority class in a node is less than 5% then don't split the node further !!
- \* If the no. of observations in a node is less than 10, make it a terminal node !!
- \* The max depth till the tree can grow is say 3 !!



We should specify the depth in such a way that the depth will learn to the optimum.

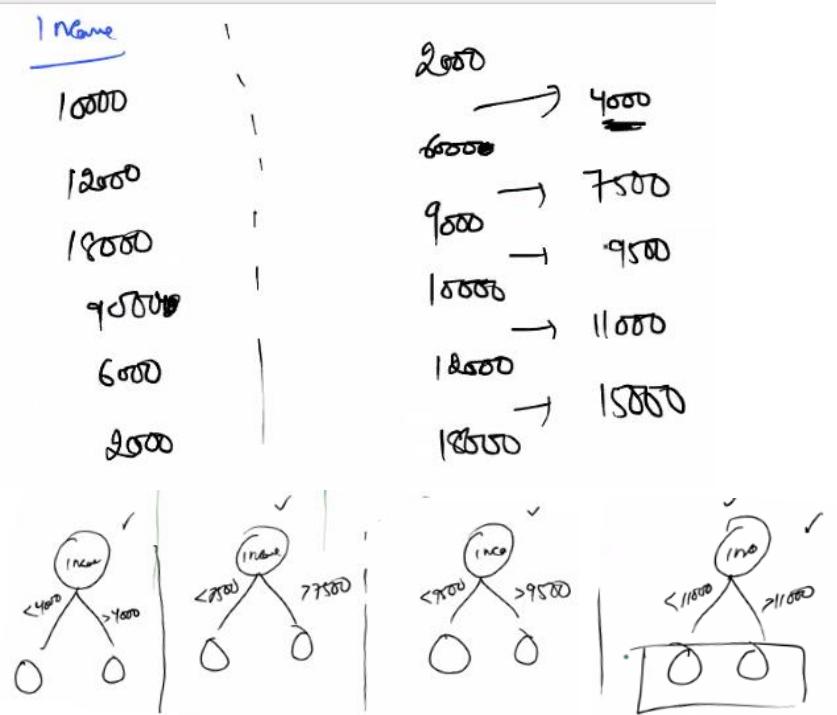
Stopping Criteria are necessary to prevent Underfitting / overfitting of a tree !!

### c. Splitting decision



For Gender and class, it's a straight forward split, but for the Income the splitting will be difficult to decide on the range to split.

To split income, the value of income will be first sorted and then median of the values between will be taken for the range

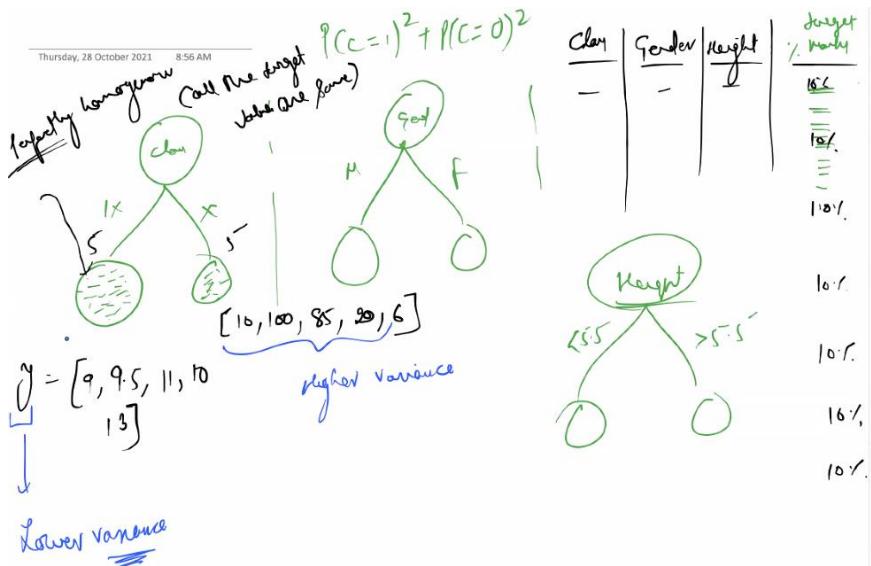


Now we calculate the Gini score for Gender, class and all the different Income ranges and from that we decide on which to split

→ If there are numeric continuous columns, the decision tree algorithm has to evaluate many possible splits before deciding the best split to split the node !!

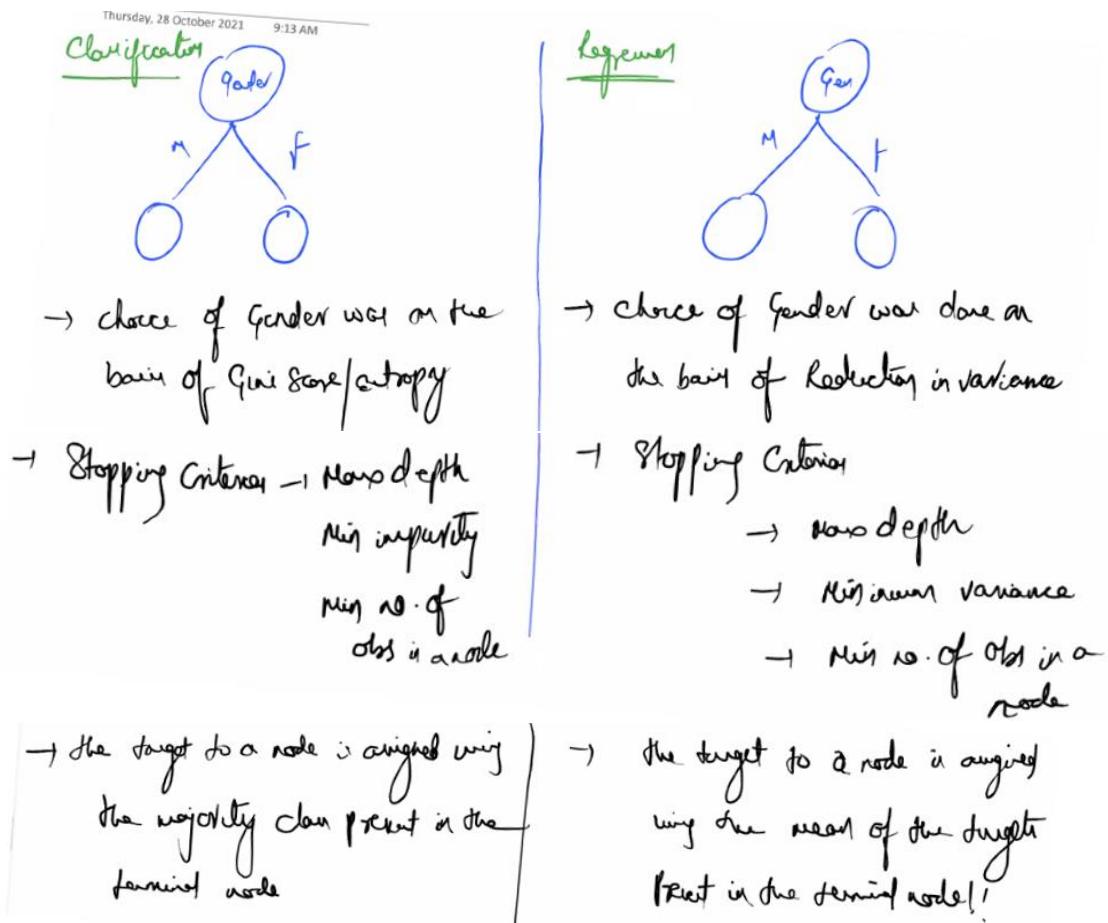
#### d. How does decision tree work for regression problems?

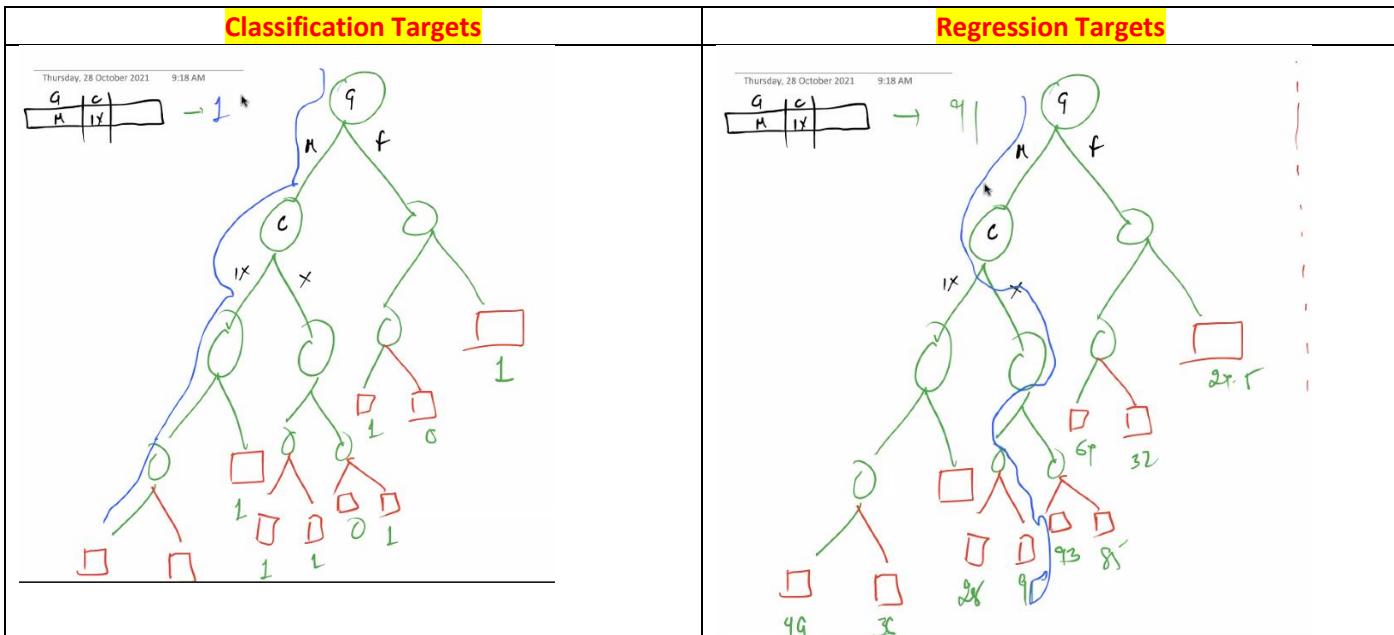
The below is a regression



So in case of a regression problem, variance is the key to find out the way to tree splitting

We calculate the weighted average on the variance and then we see which has the lowest weighted average and split on that node





Decision trees are not widely used for regression problems.

## A. Random Forest

- very similar to decision tree
- Supervised learning algorithm → Can be used for both classification & regression tasks
- Generally, better than decision tree, logistic & linear regression algorithms
- Can capture non-linear relationships b/w features & target!!

Example given below

If there is an exam with 100 chapters

- 1 option → write the exam individually ✓
- form a group of 10 people & write the exam combined!! ✓

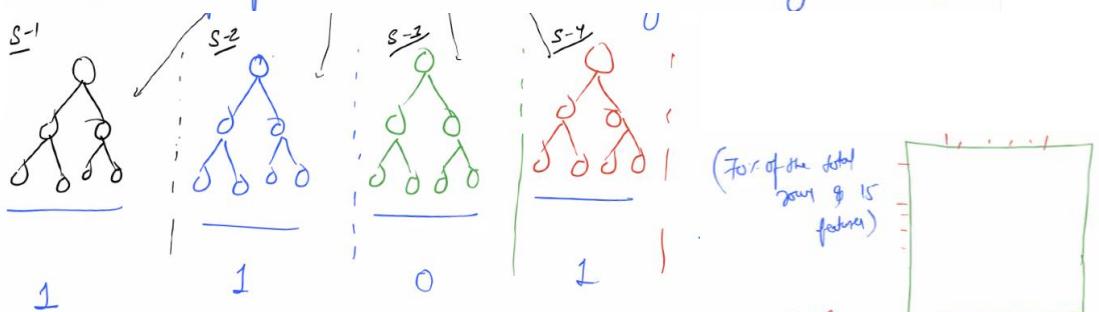
The second option is better as the chapters can be split into different individuals to save time

### 1 Decision tree

- might not be capable to learn the relationship properly!

### Random forest

- Build multiple decision tree & each tree only learns a part!



- Prediction made by the majority of trees in the final prediction assigned to the observation!!

And if it's a regression problem, then the average of the prediction will be the result

→ Random forest is an ensemble of independently grown decision trees  
 ↓  
 Combination      ↓  
 ↴                  ↴  
 ↴ no dependency

→ Random forest is a bagging method

Ensemble {  
 → Bagging → if the algo that get combined are independent of each other  
 → Boosting → if the algo that get combined are dependent on each other

## i. Implementation

### a. Classification problems

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

## sklearn.tree.DecisionTreeClassifier

```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0)
```

[source]

### b. Regression problems

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>

## sklearn.tree.DecisionTreeRegressor

```
class sklearn.tree.DecisionTreeRegressor(*, criterion='squared_error', splitter='best', max_depth=None, min_samples_split=2,
min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, ccp_alpha=0.0)
```

[source]

## ii. Under fitting

Underfitting :-

After fitting the model,  
if evaluated on training set you see a poor performance  
" " " Validation " " "

MSE  
Training set → 35%  
Validation set → 33% Underfitting  
exhibiting poor performance on both  
training & validation  
set i.e. the model is  
Underfit!!

## iii. Overfitting

Overfitting

If a model is evaluated on  
Training dataset → very good performance  
Validation dataset → the performance is not so good

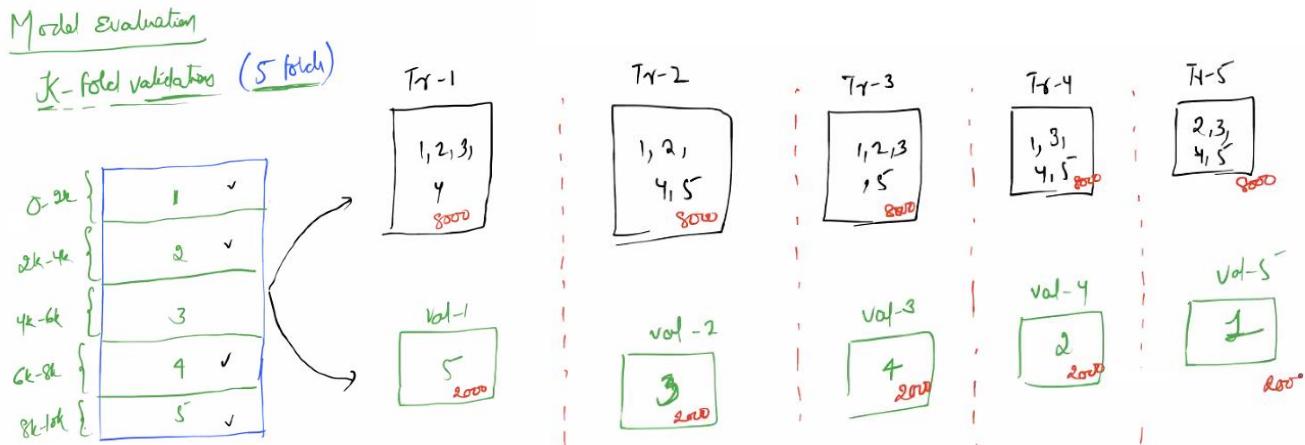
#### iv. K-Fold Validation for GridSearchCV Function

CV parameter of GridSearchCV ↗ the below is the explanation of the cv parameter of the GridSearchCV function

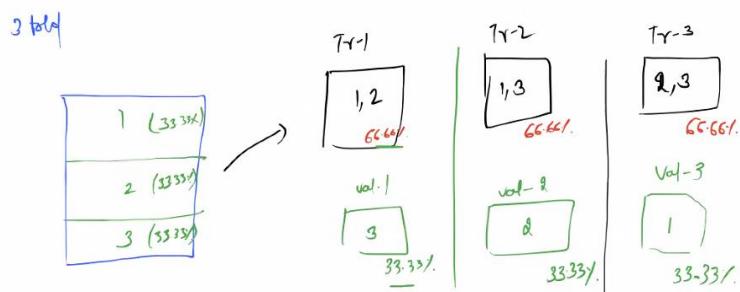
K-Fold is a means of splitting the data into n parts and training it on these n parts.

We are evaluating the parameter combination on different datasets to get a better evaluation

In the below we are splitting the data into 5 parts.



For each we would get a train f1 and validation f1 score



10 fold  
→ 10 set of (90% - 10% train - val split)

**Note:** We will use either 5 or 10 folds for this K-folds as we are dividing the data into 2 sets in case of 2-fold. The train data would contain only 50% of the combination which would be an under fit model

<p>② C-1 : <math>C.W = \text{balanced}</math>  <math>\alpha_{\text{ut}} = \text{gini}</math>  <math>\text{max-depth} = 2</math></p> <p>↓</p> <p>fitted on each <math>T_1, T_2, T_3, T_4, T_5</math></p>	<p>C-2 : <math>C.W &lt; \text{balanced}</math>  <math>\alpha_{\text{ut}} = \text{gini}</math>  <math>\text{max-depth} = 3</math></p> <p>↓</p> <p>fitted on each <math>T_1, T_2, T_3, \dots, T_5</math></p>
<p>Mean-train-f<sub>i</sub> :</p> <p>Mean-val-f<sub>i</sub> :</p>	<p>Mean-train-h<sub>i</sub> :</p> <p>Mean-val-h<sub>i</sub> :</p>

Now that we did a GridSearchCV and found the best combination we can now fit a decision tree with these parameters

GridSearch-CV →  $m_i, C_i, w_i$  → max-depth, Criteria, class-weight

D.T. ( $m.d = m_i$ ,  $C.g = C_i$ , weight =  $w_i$ )

model =  $a.\text{fit}(\text{feature-train}, \text{target-train})$

## v. SVM-Support Vector Machines

→ Support vector Machines (SVM) is a supervised learning algorithm

→ It can be used to solve both classification & regression tasks

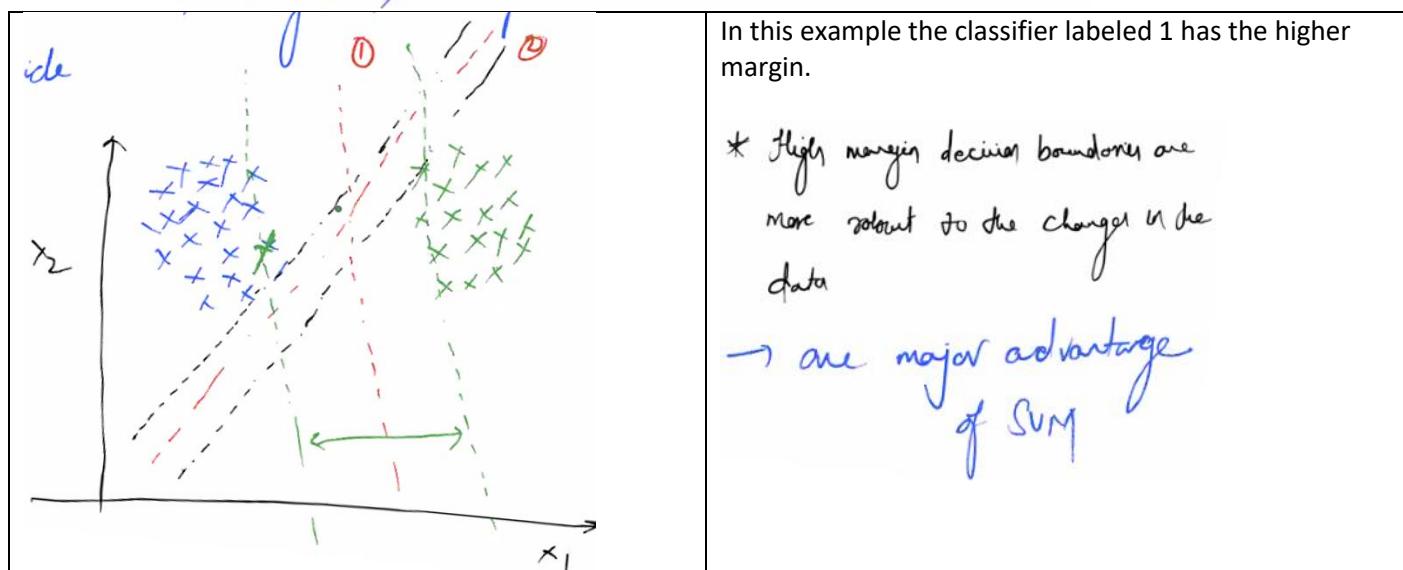
Similar to logistic & linear regression SVM is a linear classifier/regressor

i.e. - SVM can only find linear decision boundaries / linear relationship between features & targets !!

→ SVM's are also known as large margin classifiers / large margin regressors !!

→ SVM algorithm will always find a decision boundary that has highest margin !!

Margins :- distance of the decision boundary with closest point on either side



SVM algorithm is a linear algorithm

SVM + Kernel → can help to capture non linear decision boundaries

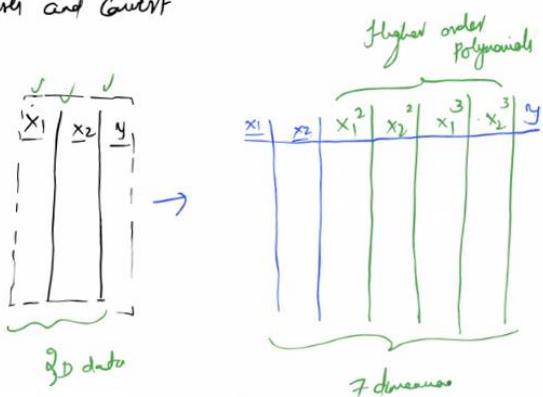
Kernel :- Transformation that can be applied on features and convert the data to a high dimension data

- \* Polynomial Kernel
- \* Gaussian Kernel
- \* Radial Kernel
- \* Tree based Kernel
- \* Custom Kernel

### i. Polynomial Kernel

Polynomial kernel with degree 3. What this would do is, it will create additional feature with higher order polynomials

structure and convert



On this newly formed dimension we will apply the SVM.

**Note:** We have to check which kernel is good for our model by passing it to the gridsearchCV

SVM + Poly	SVM + Quad	SVM + Radial	SVM + Tree
T or F	-	-	-
val h	-	-	-

All the earlier steps will be done as the same for all the MLs.

Only the parameter differs below is the way to fit for SVM

```
from sklearn.svm import SVC
clf = SVC()
#clf = LogisticRegression()
params = {'C':[0.1,0.01,0.001,1,2,5,10],
          'kernel' : ['linear', 'rbf', 'poly', 'sigmoid'],
          'degree' : [2,3,4],
          'class_weight':{'balanced'}}
```

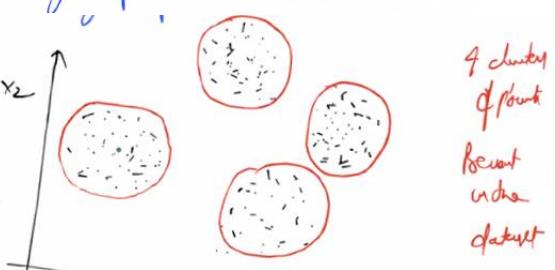
## b. UN-SUPERVISED MODELS

In unsupervised models we do not have the target columns

\* with unsupervised data, classification/ regression techniques don't work as there is no labelled target column data!!

\* with Unsupervised data, we try to find cluster present in the data

→ with unsupervised techniques we find group(s) cluster present in the data!!



→ Unsupervised methods are also called as clustering techniques/methods !!



\* 2 popular algorithms

→ K-means → find clusters present in the data

→ DBScan → find clusters present in the data

## i. K-Means

This is the most popular algorithm

For visual please check this website

<http://shabal.in/visuals/kmeans/2.html>

<http://shabal.in/visuals/kmeans/3.html>

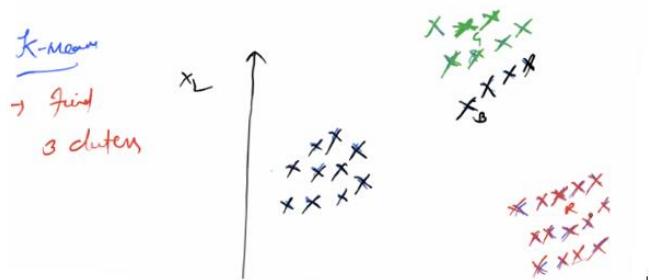
We have to specify the number of clusters. This is a limitation in the K-means.

K means method will find randomly pick up cluster centers in the data

With the clusters centers, it will calculate the distance of the points to these cluster center points and assigns as

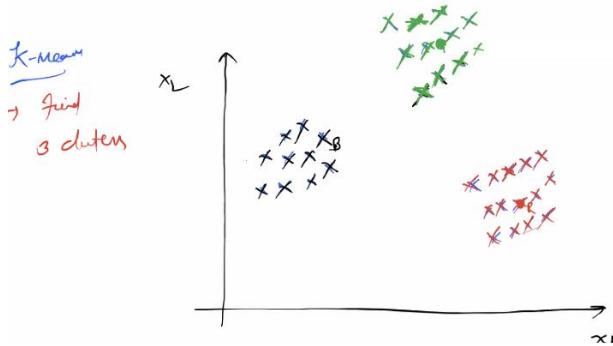
- Randomly select point as cluster center
- for every point, it will compute the distance of the point to the each cluster center. The point will be assigned to the cluster just in closest to it.

First Iteration



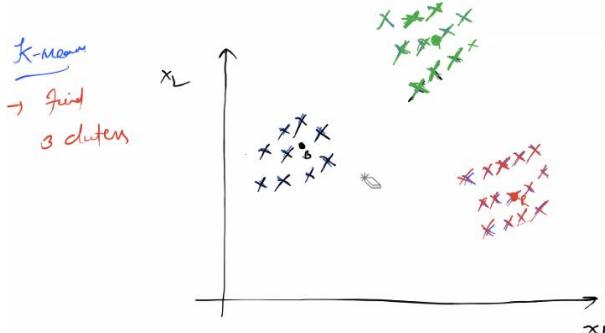
Second Iteration

Now it will find the mid of the cluster it has formed and again assigns its nearest neighbors to its cluster



3rd Iteration

In the 3<sup>rd</sup> iteration the points will align itself to the cluster center and from here on there will be no more changes in the clusters center points.



### i. Learning Steps

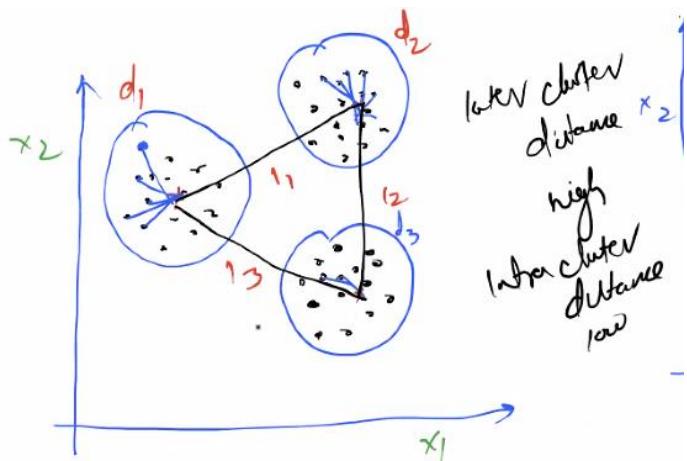
- Step-0 : Randomly select 'k' points as the cluster center !!
- Step-1 : Assign a cluster label to all other points on the basis of the distance of the point from the cluster center ↗
- Step-2 : Recalculate the cluster centers  
The Step1 & Step2 will repeat till the time no points are changing their existing cluster !!

### ii. Limitations of K-Means

#### Limitation of K-Means

- \* It needs the number of clusters to find as an input & this is something that we will not know beforehand !!
- \* K-means algo is sensitive to outliers  
→ outlier treatment before applying K-means !!
- \* K-means algo is impacted by scale difference in the features  
→ Scaling before implementing K-means algo

### iii. Finding the robustness of a cluster



#### Silhouette score

$$\text{Metric } J = \frac{\text{Within cluster distance}}{\text{Between cluster distance}} = \frac{(d_1 + d_2 + d_3)}{(l_1 + l_2 + l_3)}$$

Within cluster distance  $\rightarrow$  ✓  
Between cluster distance  $\rightarrow$  ↑

Silhouette Score is a metric to measure the goodness/robustness of the clusters formed

$$-1 < \text{Silhouette} < 1$$

Higher the Silhouette score more the robustness of the cluster.

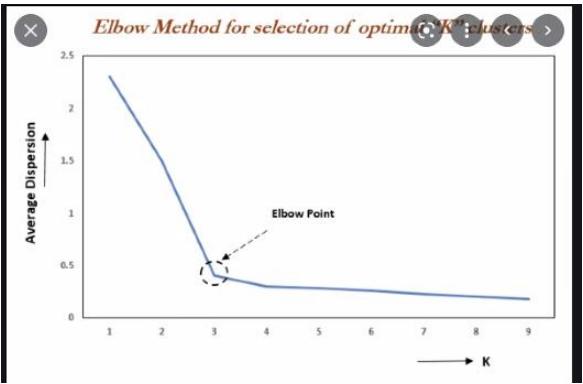
#### Inertia

Inertia is a metric

Elbow curve  $\rightarrow$  a plot that helps us to decide on number of clusters!!

Inertia also uses inter cluster and intra cluster average distances to calculate the number of K's

In the elbow curve we plot the inertia vs the number of clusters and the point where we can see an elbow is the best value of K



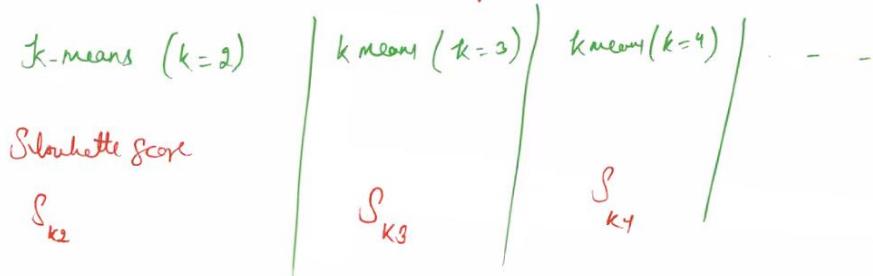
#### iv. How to determine the best value of K

We can never know before hand is the answer 😊

So we fit the k-Means for k values from 2 to 10

Then we calculate the Silhouette score is close to 1, we will choose that cluster k

*How to determine the best value of k?*



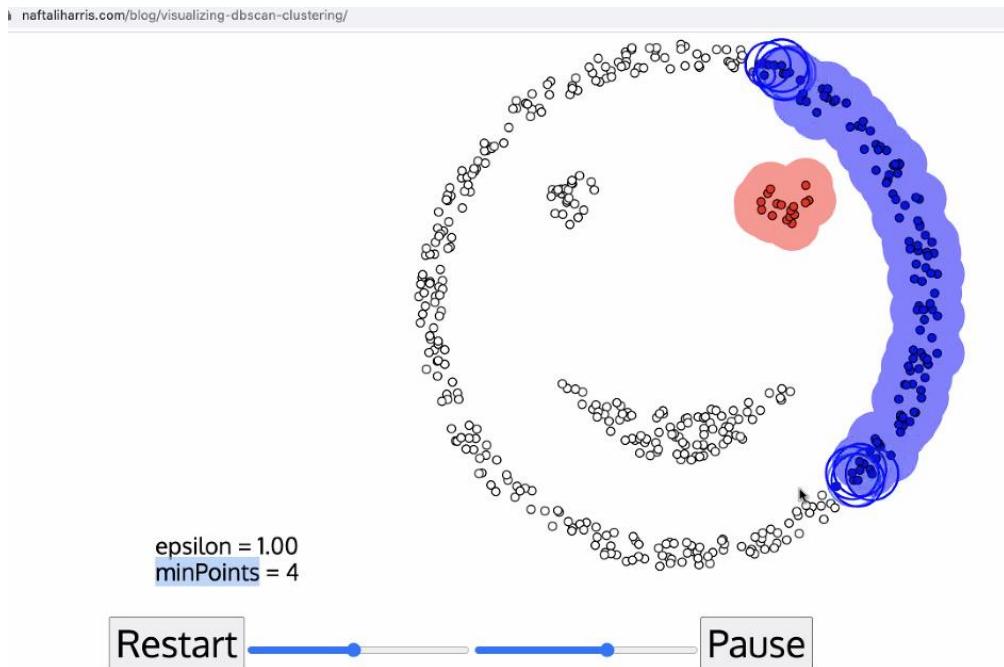
\* whenever the Silhouette score is highest, that implies the best cluster for the data!!

#### v. Implementation in Python

check it yourself

## ii. DB-Scan

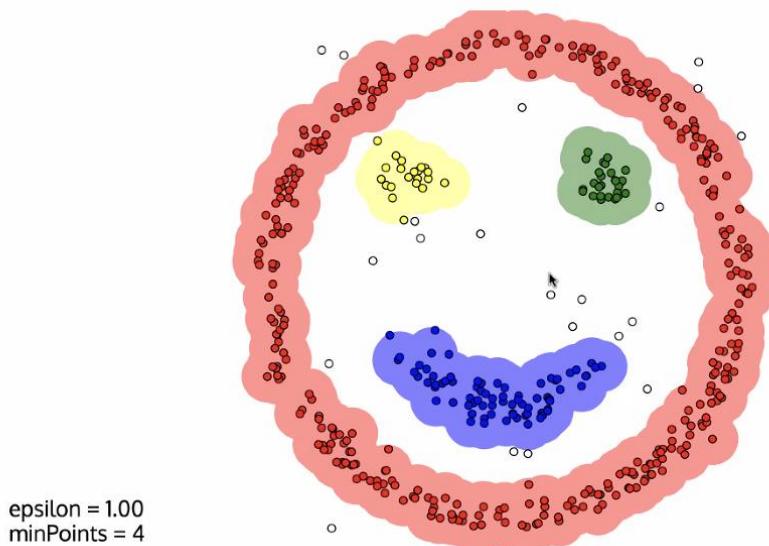
<https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/>



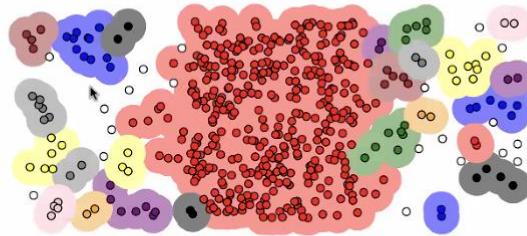
DB scan works on 2 parameters

1. Epsilon  $\square$  is the radius of circle of to be created
2. minPoints  $\square$  is the minimum number of points required to form the cluster around the epsilon radius formed

In the below we can see that there are points which can be thought as outliers. For the outlier point when DB scan forms a circle there will be no minPoints of a given number and DBscan thus treats them as outliers.



Changing the value of epsilon and minPoints we get differing results. So it is important to set the right values for epsilon and minPoints



epsilon = 0.68  
minPoints = 2

(  
Lower epsilon → More clusters  
Lower min points → less clusters  
)

### i. Advantages and limitation of DB Scan

Advantages to DBScan

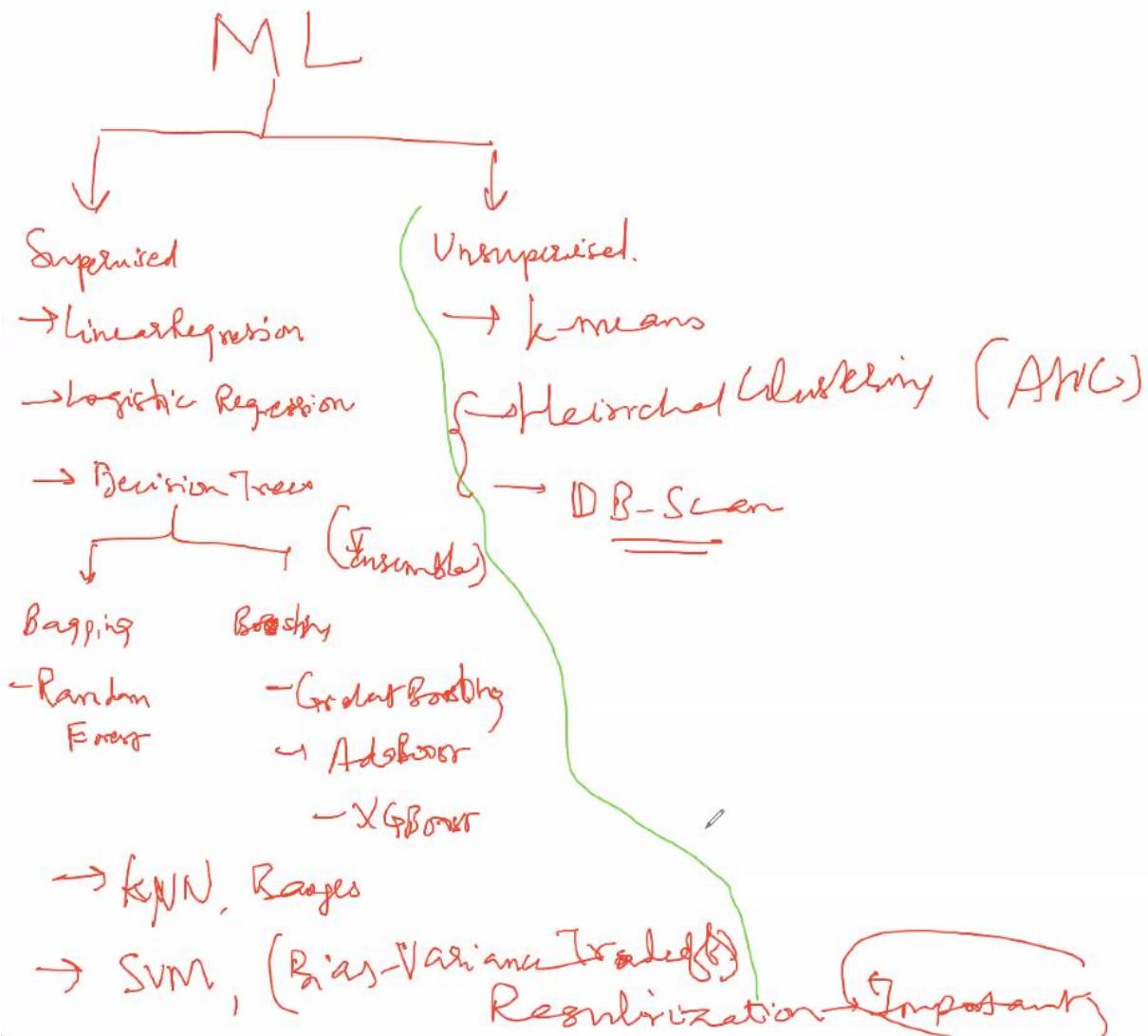
- \* Not Sensitive to outliers
- \* It does not require the number of clusters to be explicitly passed.

Limitations

- \* It is also impacted by scale
- \* we don't know a good value for epsilon (radius)  
& the min-point

### iii. Hierarchical Clustering

Check it yourself



## Resources

- Towards Data science
- Analytics Vidya
- Kaggle solutions- Machine learning
- Machine learning mastery