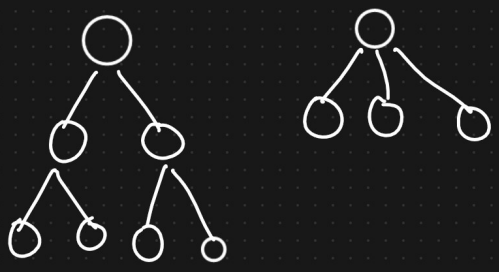
Decision Tree Classifier

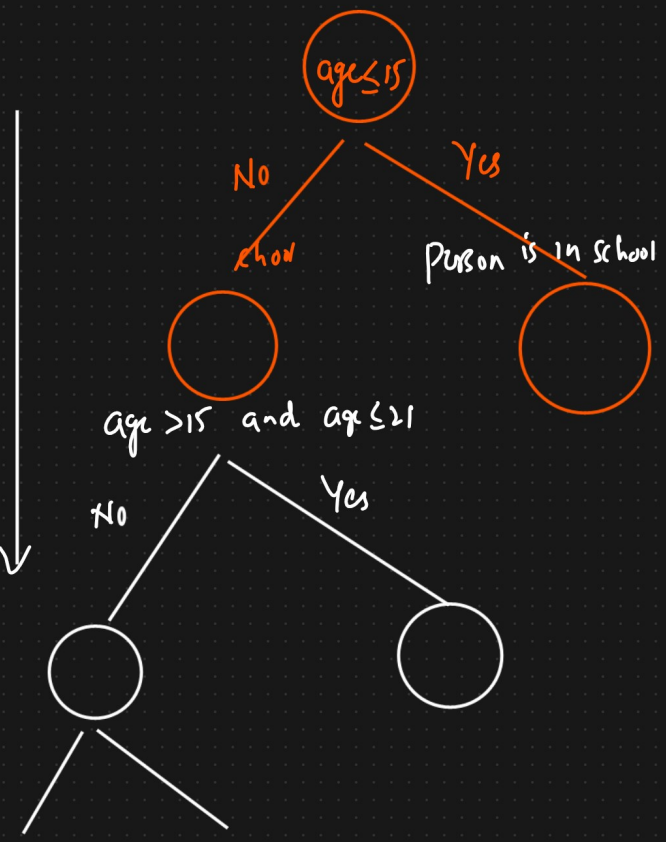
Decision Tree Classifier → ID3
→ CART ✓



a) Entropy and Gini Index → Purity Split

b) Information Gain → features to select for DT construction

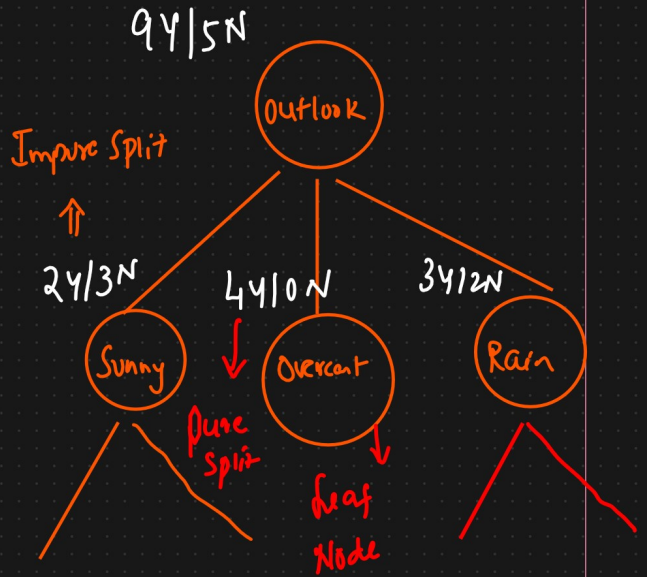
```
age = 14
if (age ≤ 15):
    Print ("The person is in School")
elif (age > 15 and age ≤ 21):
    Print ("The person may be college")
else:
    Print ("The person has passed")
```



Dataset

Binary classification

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



① Purity → Pure or Impure Split

↳ Entropy

↳ Gini Impurity

② What feature you need select for Splitting → Information Gain

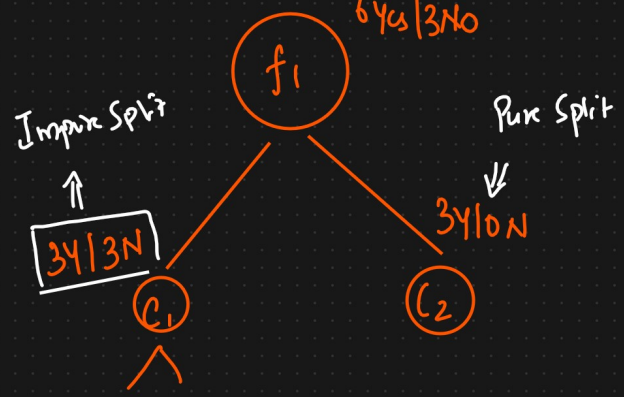
1

0

{Binary classification}

1) Entropy

$$H(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

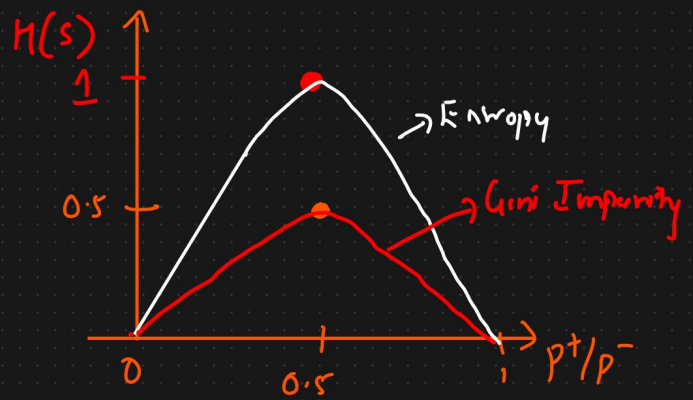


$$H(C_1) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

$$= -\frac{3}{6} \log_2 3/6 - \frac{3}{6} \log_2 3/6$$

2) Gini Impurity

$$G.I = 1 - \sum_{i=1}^{n} (p)^2$$



$= 1 \Rightarrow$ Impure Split

$$H(C_2) = -\frac{3}{3}\log_2\frac{3}{3} - 0\log_2 0$$

$= -1\log_2 1 \Rightarrow 0 \Rightarrow$ Pure Split

② Gini Impurity

$$G.I = 1 - \sum_{i=1}^{n} (p)^2$$

$= 1 - ((p_+)^2 + (p_-)^2)$

$= 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right)$

$= 0.5 \Rightarrow$ Impure Split

3Y/0N

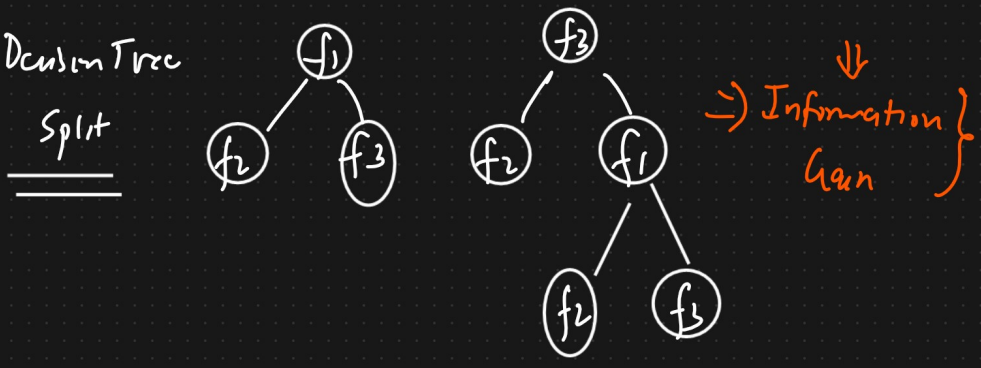
$= 1 - \left(\left(\frac{3}{3}\right)^2\right)$

$= 1 - 1$

$= 0 \Rightarrow$ Pure Split

f1 f2 f3

Decision Tree Split

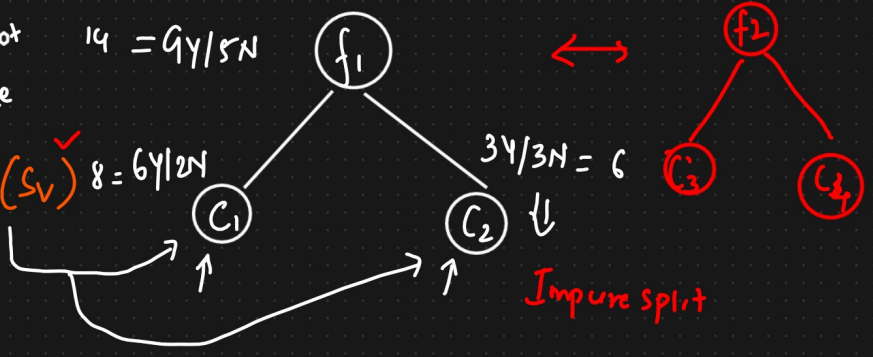


⇓ ⇒ Information Gain

Information Gain

f1 f2 f3 O/P

14 = 9Y/5N



$$Gain(S, f_1) = \boxed{H(S)} - \sum_{v \in val} \frac{|S_v|}{|S|} H(S_v)$$

↑ Entropy of the root node

↑ Root Node

$$H(S) = -P_+\log_2 P_+ - P_-\log_2 P_-$$

$= -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14}$

≈ 0.94

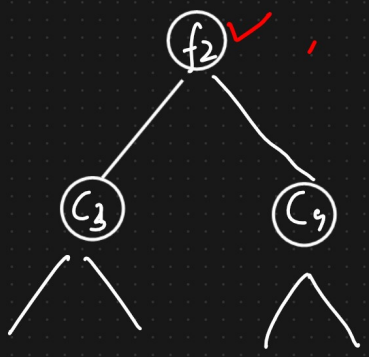
$H(C_1) = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8}$

$H(C_1) = 0.81$

$H(C_2) = 1$

$Gain(S, f_1) = 0.94 - \left[\frac{8}{14} \times 0.81 + \frac{6}{14} \times 1\right]$

$Gain(S, f_1) = 0.049$



$Gain(S, f_2) = 0.051 > Gain(S, f_1) = 0.049$

Gain

Information is Basically calculated.

Entropy Vs Gini Impurity

$H(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$

$G.I = 1 - \sum_{i=1}^{n} (p)^2 \Rightarrow$

O/P = 3 categories

$H(S) = -p_{C_1} \log_2 p_{C_1} - p_{C_2} \log_2 p_{C_2} - p_{C_3} \log_2 p_{C_3}$

Whenever dataset is small → Entropy

large → Gini Impurity

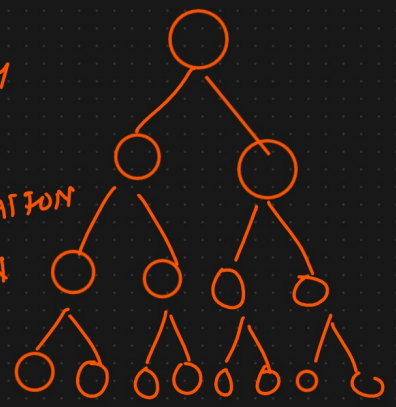
Decision Tree Regression

Dataset

Exp	Career Gap	Salary
2	Yes	40K
2.5	Yes	42K
3	No	52K
4	No	60K
4.5	Yes	56K

op

Entropy
GINI
INFORMATION GAIN



CART

[40K, 42K, 52K, 60K, 56K]

50K

60.8

≤ 2

Yes → 40K

No → 42, 52, 60, 50

V(C1) = 100

V(C2) = 51

[40K, 42K, 52K, 60K, 56K]

60.8

≤ 2.5

40, 42

52, 60, 56

Decision Tree Regressor

Var(C1) = $\frac{1}{2}\left[(40-50)^2 + (42-50)^2\right]$

$= \frac{1}{2}\left[100 + 64\right]$

$= \frac{164}{2} = 82$

Var(child2) = $\frac{1}{3}\left[4 + 100 + 36\right]$

$= \frac{140}{3} = 46.66$

Variance Reduction (Regression Problem)

① Variance $= \frac{1}{n}\sum_{i=1}^{n}(y-\bar{y})^2$

Mean Square Error

→ Average

Variance (Root) $= \frac{1}{5}\left[(40-50)^2 + (42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2\right]$

$= \frac{1}{5}\left[100 + 64 + 4 + 100 + 36\right]$

$= 60.8$

Variance Reduction

$= 60.8 - \left[\frac{2}{5} \times 82 + \frac{3}{5} \times 46.66\right]$

$= 60.8 - \left[32.8 + 27.996\right]$

$= +0.004$

Variance (C1) $= \frac{1}{n}\sum_{i=1}^{n}(y-\bar{y})^2$

$$= \frac{1}{1} (40-50)^2$$

$$= 100$$

$$\text{Variance (c2)} = \frac{1}{n} \sum_{i=1}^{n} (y-\bar{y})^2$$

$$= \frac{1}{4} \left[(42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2 \right]$$

$$= \frac{1}{4} \left[64 + 4 + 100 + 36 \right]$$

$$= 51$$

Variance Reduction ↓

$$= \text{Var(Root)} - \sum w_i \text{Var(child)}$$

$$= 60.8 - \left[\frac{1}{5} \times \overset{20}{\cancel{100}} + \frac{4}{5} \times 51 \right]$$

$$= 60.8 - 20 - 40.8$$

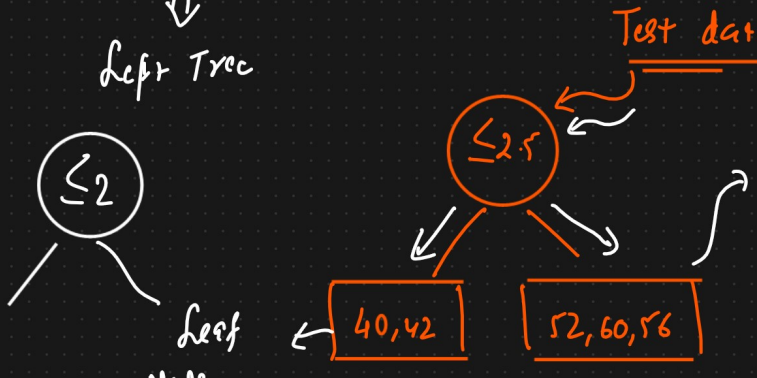
Variance Reduction = 0

0 0.004

Variance Reduction (Left Split) < VR (Right Split)

⇓

Left Tree



Test data

Decision Tree O/P Reg.

$\frac{52+60+56}{3} = 56$

O/P = $\left[\frac{40+42}{2} \right]$

$= 41$

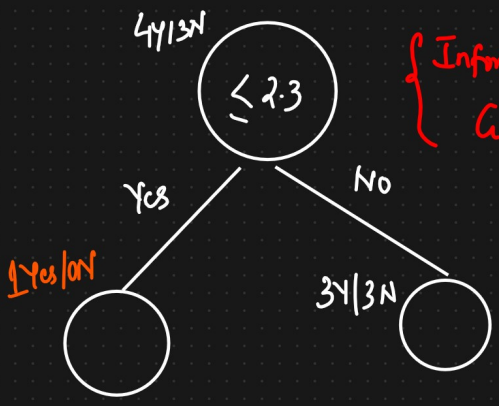
Decision Tree Split for Numerical Feature.

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

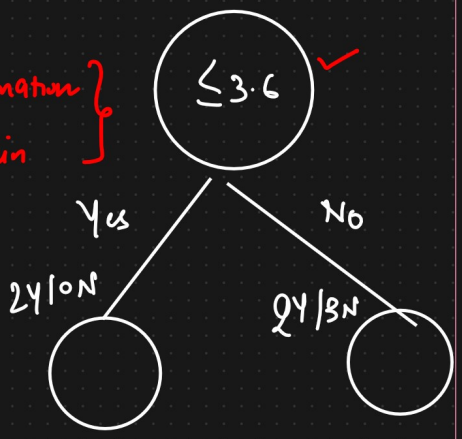
① Sort the feature value

f1	O/P
2.3	Yes
3.6	Yes
4	No
5.2	No
6.7	Yes
8.9	No
10.5	Yes

① Threshold = 2.3



② Threshold = 3.6

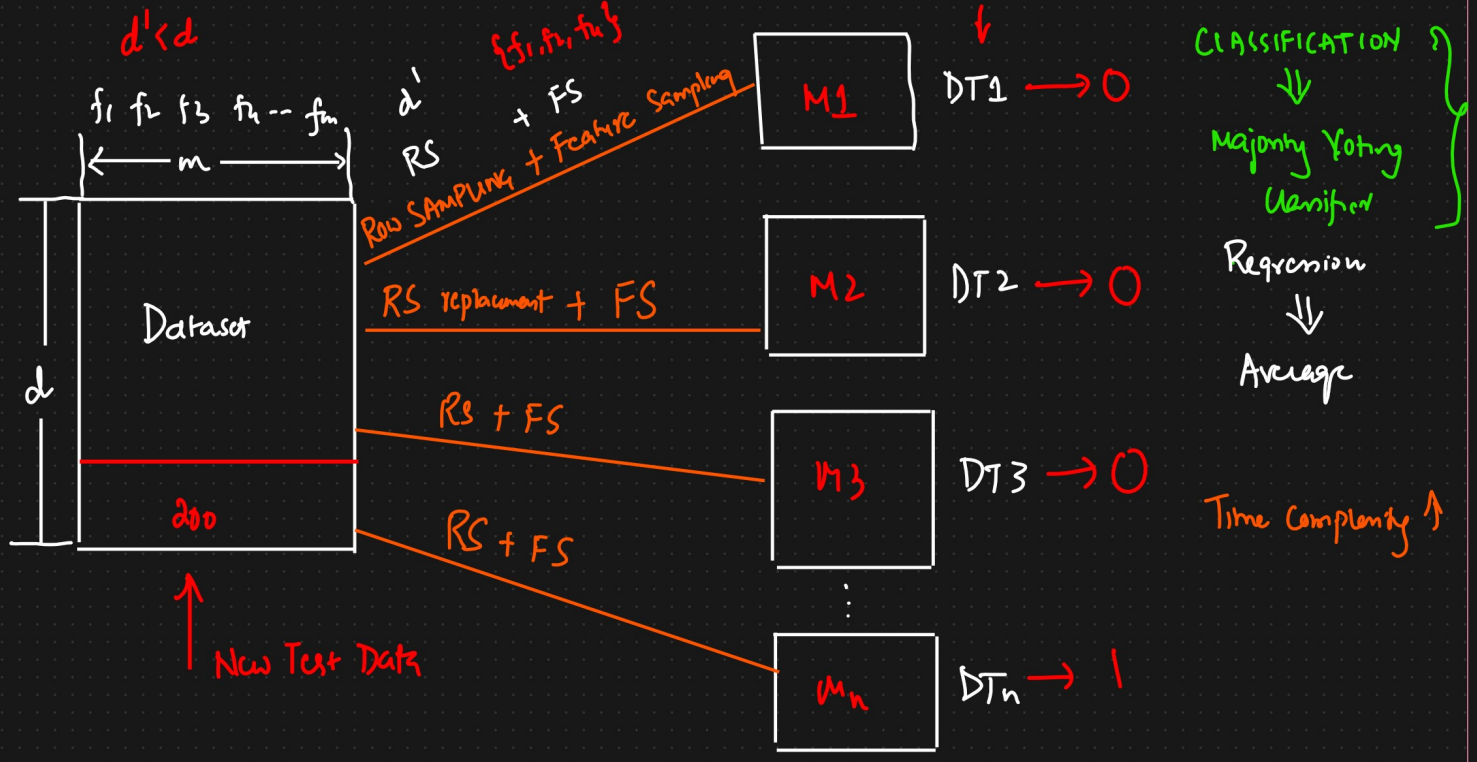


{ Information Gain }

Millions of records

Time Complexity ↑↑

Random Forest Classification And Regression



CLASSIFICATION - MAJORITY VOTING CLASSIFIER

REGRESSION - Average O/P of the Models.

Why should we use Random Forest instead of DT?

Decision Tree

Generalized Model → Low Bias
→ Low Bias

Overfitting

Train Acc ↑ → Low Bias

Test Acc ↓ → High Variance

Random Forest

→ Low Bias

→ Low Variance

AdaBoost Machine Learning Algorithms

Decision Trees

Ensemble Techniques

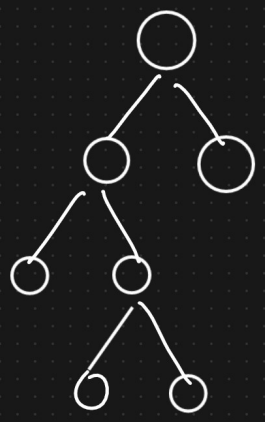
{Base Learners} → Bagging

{weak learners} → Boosting

{
① Random Forest Classifier
② Random Forest Regressor
}

① Adaboost
② Gradient Boosting
③ xgboost

Decision Tree :



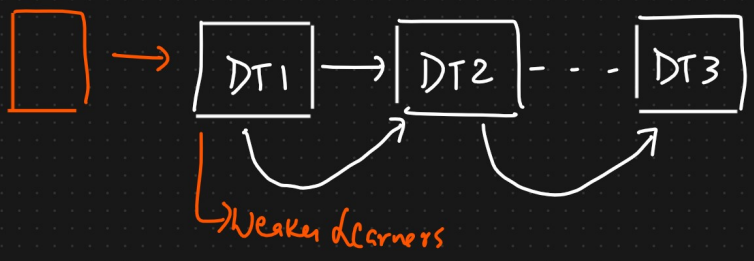
Overfitting : Train Acc ↑
Test Acc ↓

{
Low Bias
High Variance
}

Random Forest {Bagging}

{
Low Bias
Low Variance
}

Boosting {Sequentially connected}



⇒ Strong Learner

Weak Learner → Haven't learnt much from the Training Dataset

Random Forest → Majority Voting classifier
Average of (O/P)

AdaBoost ⟶ Assignment weights to the weak learner

$M_1 \dots M_n$ → Decision Tree Stumps

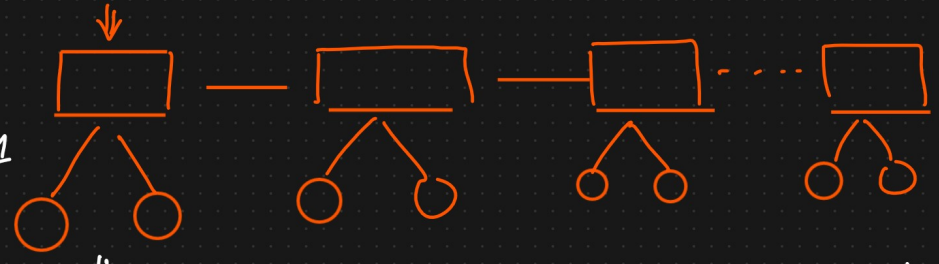
$$f = \alpha_1(M_1) + \alpha_2(M_2) + \alpha_3(M_3) + \dots \alpha_n(M_n)$$

→ CLASSIFICATION
→ REGRESSION

$\{\alpha_1, \alpha_2, \alpha_3 \dots \alpha_n\}$ ⇒ weights

Decision Stump

Depth = 1



Weak Learner

↳ Underfitting

↳ Train Acc ↓ 40%

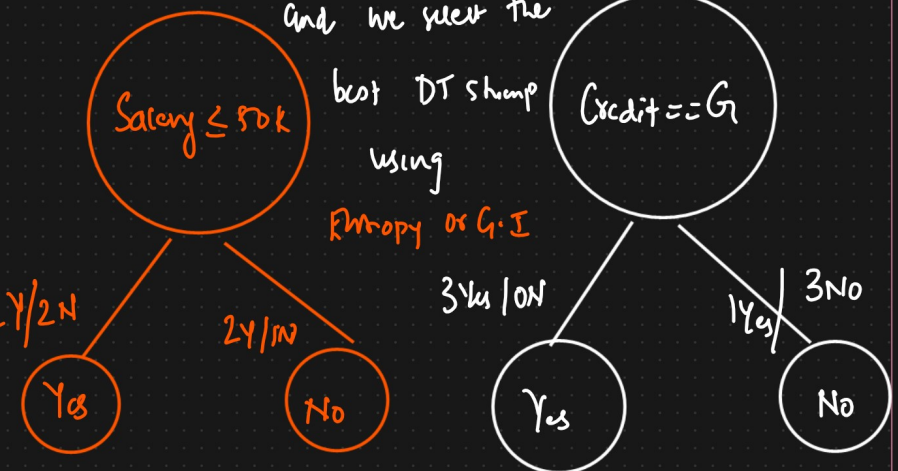
↳ Test Acc ↑ 45%

Decision Tree Stump ⇓ { High Bias, Low Variance } ⇒ { Low Bias, High Variance }

Adaboost Classifier Maths Indepth Intuition

Salary	Credit	Approval
<=50K	B	No
<=50K	G	Yes
<=50K	G	Yes
>50K	B	No
>50K	G	Yes
>50K	N	Yes
<=50K	N	No

① We create Decision Tree Stump and we select the best DT stump using Entropy or G.I



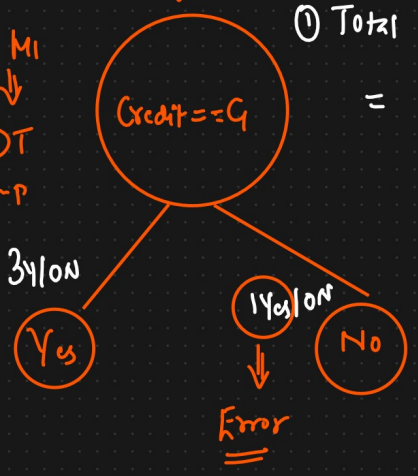
Entropy or Gini Impurity

$$H(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

② Sum of the Total Errors And Performance of Stump

Salary	Credit	Approval	Sample Weights
<=50K	B	No	1/7
<=50K	G	Yes	1/7
<=50K	G	Yes	1/7
>50K	B	No	1/7
>50K	G	Yes	1/7
>50K	N	Yes	1/7
<=50K	N	No	1/7

M1 or DT Stump



① Sum of all the Total Error = 1/7

② Performance of Stump $= \frac{1}{2} \ln\left[\frac{1 - TE}{TE}\right]$

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

$= \frac{1}{2} \ln[6] \approx 0.896$

Performance of Stump ≈ 0.896

$f = \alpha_1(M_1) + \alpha_2(M_2) + \cdots \alpha_n(M_n)$

$\alpha_1 = 0.896$ ⇒ weight

③ Update the weights for correctly and Incorrectly classified points

Salary	Credit	Approval	Sample Weights	Update wts
<=50K	B	No	1/7 ↓ →	0.058
<=50K	G	Yes	1/7 ↓ →	0.058
<=50K	G	Yes	1/7 ↓ →	0.058
>50K	B	No	1/7 ↓	0.058
>50K	G	Yes	1/7 ↓	0.058
>50K	N	Yes	1/7 ↑ →	0.349
<=50K	N	No	1/7 ↓ →	0.058

For correct classified points

= Weight * $e^{-\text{Performance of stump}}$

= $\frac{1}{7} * e^{-(0.896)}$

= 0.058

For Incorrect classified

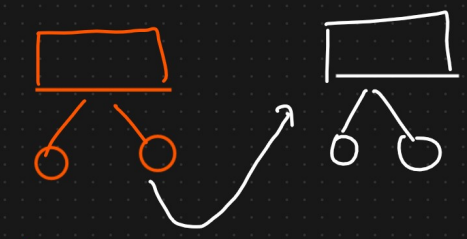
= Weight * $e^{\text{Performance of stump}}$

= $\frac{1}{7} * e^{(0.896)}$

= 0.349

(4) Normalized Weights Computation And Assigning Bins

Salary	Credit	Approval	Update wts	Normalized weights	Bins Assignment
<=50K	B	No	0.058	0.08	0 - 0.08
<=50K	G	Yes	0.058	0.08	0.08 - 0.16
<=50K	G	Yes	0.058	0.08	0.16 - 0.24
>50K	B	No	0.058	0.08	0.24 - 0.32
>50K	G	Yes	0.058	0.08	0.32 - 0.40
>50K	N	Yes	0.349	0.50	0.40 - 0.70
<=50K	N	No	0.058	0.08	0.50 - 0.58
			0.697	≈ 1	



$\lambda_1 = 0.896$ Prepare datapoints

(5) Select data points to send to Next Stump

(1) Iteration process selecting random value between 0 and 1

Salary	Credit	Approval	Bins Assignment
<=50K	B	No	0 - 0.08
<=50K	G	Yes	0.08 - 0.16
<=50K	G	Yes	0.16 - 0.24
>50K	B	No	0.24 - 0.32
>50K	G	Yes	0.32 - 0.40
>50K	N	Yes	0.40 - 0.70
<=50K	N	No	0.50 - 0.58

S	Credit	Approval	Random
>50K	N	Yes	0.50
<=50K	G	Yes	0.10
>50K	N	Yes	0.60
>50K	N	Yes	0.75
<=50K	G	Yes	0.24
>50K	B	No	0.32
>50K	N	Yes	0.87

(6) These records will be sent to next DT Stump.

S	Credit	Approval	Sample weight
>50K	N	Yes	1/6
<=50K	G	Yes	1/6
>50K	N	Yes	1/6
>50K	N	Yes	1/6
<=50K	G	Yes	1/6
>50K	B	No	1/6
>50K	N	Yes	1/6

TE

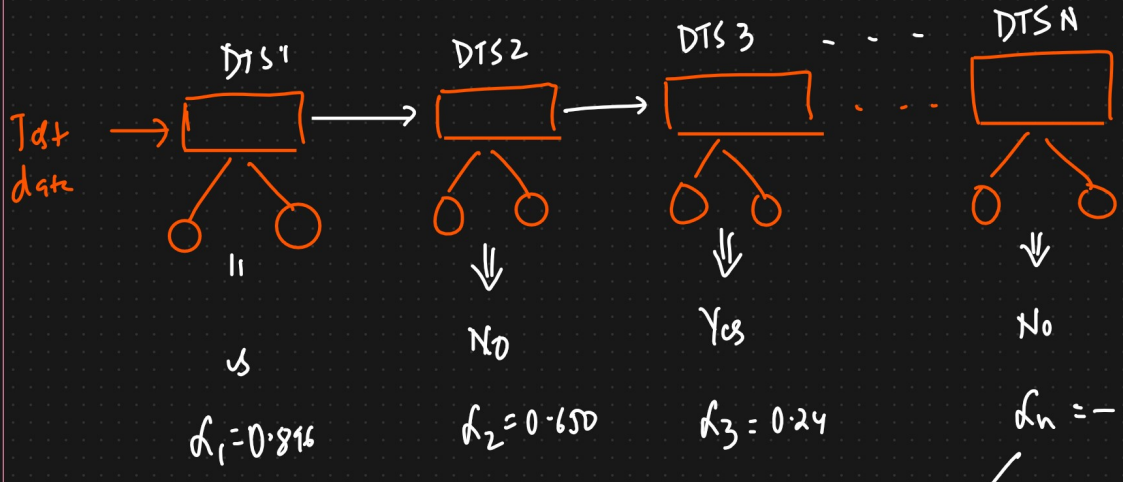
Performance Stump ⇒ 0.65

$f_1 = \alpha_1(M_1) + \alpha_2(M_2)$

$\alpha_1 = 0.896$ $\quad \alpha_2 = 0.65$

(7) Final Prediction

Test data $(\leq 50K, G)$



$\alpha_1 = 0.896$ $\quad \alpha_2 = 0.650$ $\quad \alpha_3 = 0.24$ $\quad \alpha_n = -0.30$

$f = \alpha_1(M_1) + \alpha_2(M_2) + \alpha_3(m_3) + \cdots \alpha_4(M_4)$

$= 0.896(\text{Yes}) + (0.650)(\text{No}) + 0.24(\text{Yes}) - 0.30(\text{No})$

$$\begin{array}{r} 1\ 0.896 \\ 0.240 \\ \hline 1.136 \end{array}$$

$= \boxed{1.136}(Y_{es}) + 0.350\,(No)$ ⇒ O/p : Yes

Performance of say (Yes) = 1.136 >

Performance of say (No) = 0.350

Gradient Boosting Algorithm

① Regression

② Classification

Regression Dataset

Exp	Degree	Salary (y)	$\hat{y}$	$(y-\hat{y})$ R_1	R_2	$\hat{y}$	R_3	R_4
→ 2	B.E	50K	75K	-25K	-23	72.7	-22.7	-
3	Masters	70K	75K	-5K	-3	74.7	-4.7	-
5	Masters	80K	75K	5K	3	-	-	-
6	PHD	100K	75K	25K	20	-	-	-
		75K						

Steps

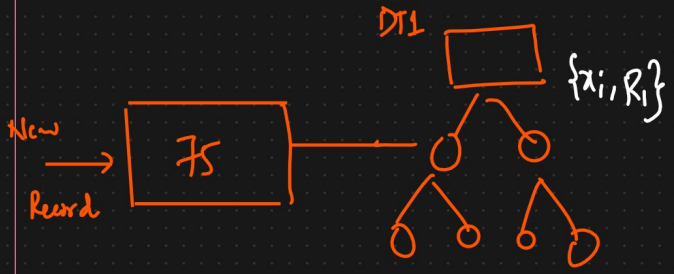
① Create a Base Model

Average = $\frac{50+70+80+100}{4} = 75$

② Compute Residuals, Error

③ Construct a Decision Tree

Consider inputs x_i and O/p R_i



Predicted O/p ⇒ 75 + (-23) = 75 - 23 = 52 {Overfitting}

Predicted O/P $\Rightarrow$ $75 + \alpha(DT_1) = 75 + (0.1)(-23)$

α = learning (0 to 1)

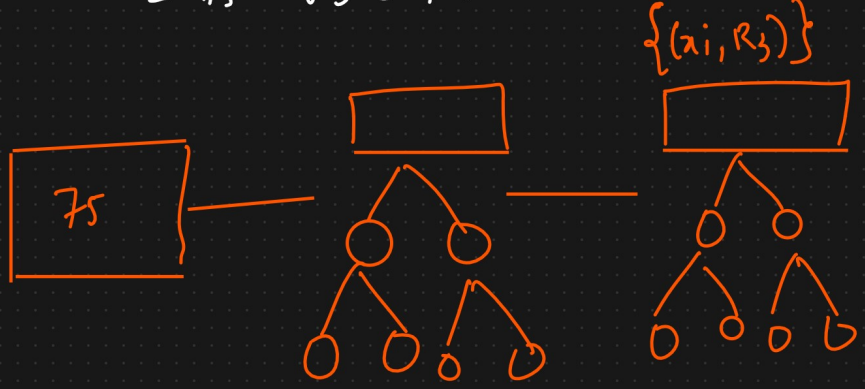
$\alpha = 0.1$

$= 75 - 2.3$

$= 72.7$

$\Rightarrow 75 + 0.1(-3)$

$= 75 - 0.3 = 74.7$



$$F(x) = \alpha_0 h_0(x) + \alpha_1(h_1(x)) + \alpha_2(h(x)) + \alpha_3(h(x)) - - + \alpha_n(h(x))$$

learning Rate $\alpha = 0.1$

$$F(x) = \sum_{i=0}^{n} \alpha_i h_i(x)$$

$\Rightarrow$ Final Function of Gradient Boosting

Xgboost ML Algorithm (Classification)

$\hat{y} = 0.5$

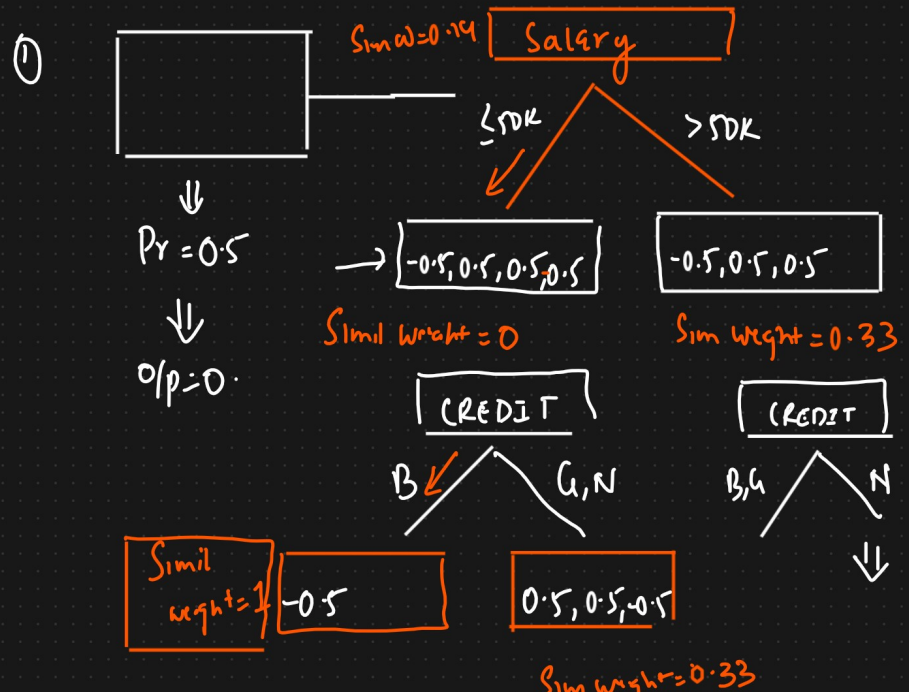
Dataset

Salary	Credit	y Approval	R1	$\hat{y}$	R2
<=50K	B	0	-0.5	0.52	-0.48
<=50K	G	1	0.5	0.58	0.42
<=50K	G	1	0.5	—	—
>50K	B	0	-0.5	—	—
>50K	G	1	0.5	—	—
>50K	N	1	0.5	—	—
<=50K	N	0	-0.5	—	—

Steps

1. Construct a base Model
2. Construct a Decision Tree with root.
3. Calculate Similarity Weight
   $= \sum (\text{Residual})^2$
   Cover value $\leftarrow \sum Pr(1-Pr) + \lambda$ (hyperparameter)
4. Calculate Gain

[-0.5, 0.5, 0.5, 0.5, -0.5, 0.5, 0.5]



Test data

$$\log(\text{odds}) = \log\left(\frac{P}{1-P}\right)$$

$$\log(\text{odds}) = \log\left(\frac{0.5}{0.5}\right)$$

$$= 0$$

$$\text{Similarity (Root)} = \frac{0.25}{0.25} = 1$$

$$\text{Similarity (Rest)} = \frac{0.25}{0.75} = 0.33$$

$$\text{Gain} = 1 + 0.33 - 0 = \boxed{1.33}$$

Similarity weight $(\lambda) = \frac{\Sigma (\text{Residual})^2}{\Sigma P_r(1-P_r)}$

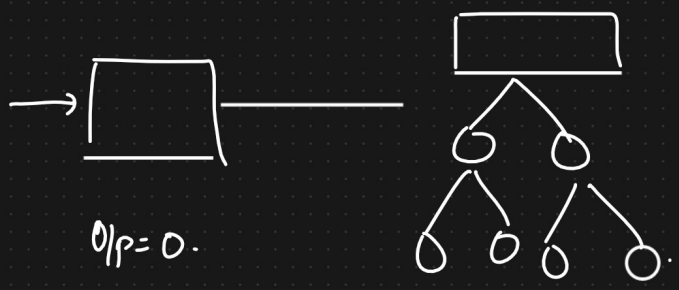
Similarity (RC) $= \frac{(-0.5+0.5+0.5)^2}{0.75} = \frac{0.25}{0.75} = \frac{1}{3}$

$= [(-0.5+0.5+0.5-0.5)^2] = 0$

$1 \Leftarrow [0.5(1-.5)+0.5(1-.5)+ .5(1-.5)+0.5(1-.5)]$

Gain $= 0 + 0.33 - .14 = .21$

Final O/p



New Data $= \sigma\left(0 + \lambda(1)\right)$ ← (Similarity weight), ↓

$\lambda = 0.1$

Logistic function ⇐ Sigmoid Activation $= \sigma\left(0 + (0.1)(1)\right)$

$= \frac{1}{1+e^{-0.1}} = 0.52$

Second Record O/p $= \sigma\left(0 + \lambda(0.33)\right)$

$= \sigma\left(0 + 0.1(0.33)\right)$

$$= \frac{1}{1+e^{-0.033}} = 0.508$$

Xgboost classifier

$$O/P = \sigma\left(\text{Base learner} + \alpha_1(DT1) + \alpha_2(DT2) + \alpha_3(DT3)\right).$$

Xgboost Regressor Mh Algorithm

Dataset {Regression}

Exp	Gap	Salary	R_1	$\hat{y}$	R_2
2	Yes	40K	-11	49.9	-9.9
2.5	Yes	42K	-9	49.9	-7.9
3	No	52K	1	51.5	0.5
4	No	60K	9	51.5	8.5
4.5	Yes	62K	11	52.1	9.9
		≈ 51K			

$[51 + (0.1)(-10)] = 51 - 0.1 = 49.9$

$[51 + (0.1)(5)] = 51.5$

$[51 + 0.1(11)] = 51 + 1.1 = 52.1$

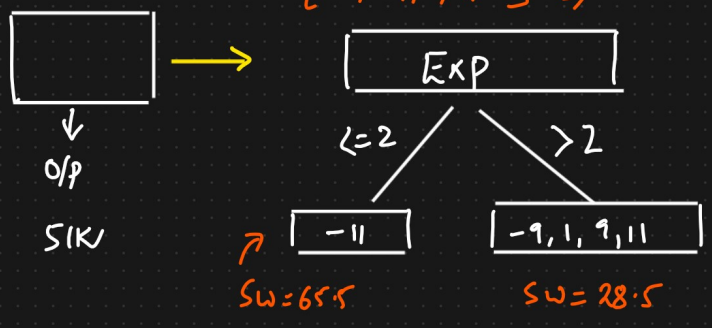
$$\text{Similarity weight} = \frac{\sum(\text{Residual})^2}{\sum p_r(1-p_r)}$$

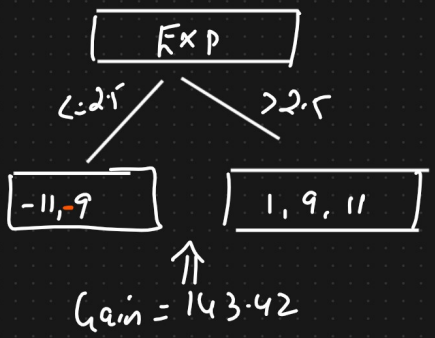
Gain

Steps

① Create a Base Model ② Residual Computation ③ Construct DT1 using $\{x_i, R_1\}$

$[-11, -9, 1, 9, 11] \Rightarrow SW = 0.16$.





$$\text{Similarity weight} = \frac{\sum(\text{Residual})^2}{\text{No. of Residuals} + \lambda}$$

λ → Hyperparameter

$\lambda \uparrow$ SW ↓

$\lambda = 1$

$$SW(\text{Right child}) = \frac{121}{1+1} = 121/2 = 65.5 //$$

$$SW(\text{Right child}) = \frac{(-9+1+9+11)^2}{4+1}$$

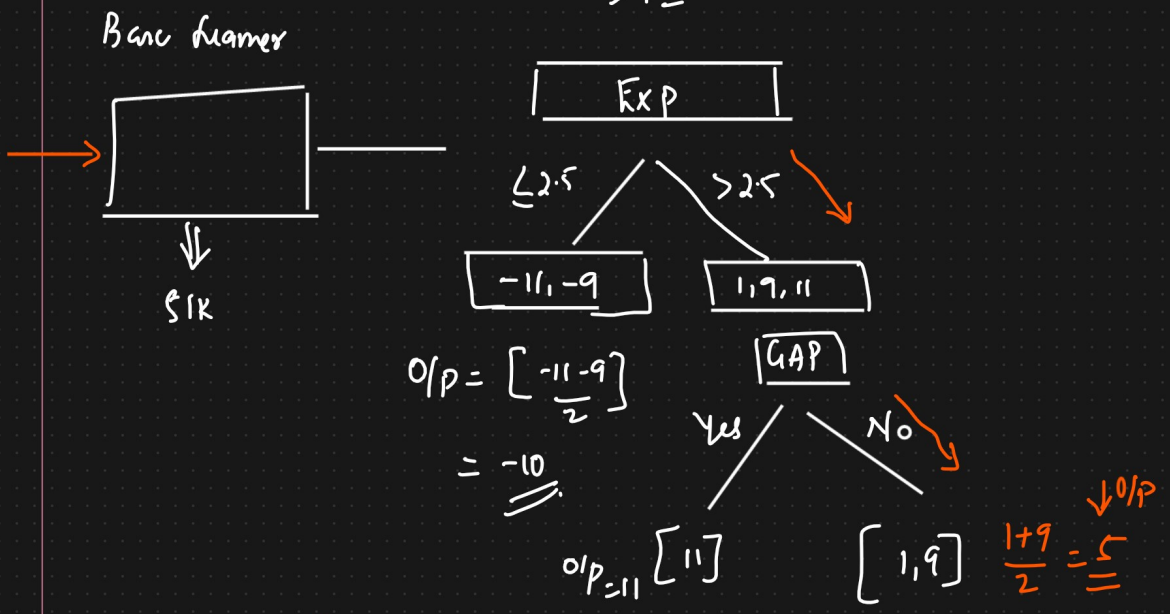
$$= \frac{144}{5} = 28.5$$

(5) Calculate Gain

$$\text{Gain} = 65.5 + 28.5 - 0.16$$

$$= 98.34$$

DT 1



α = Learning Rate $\alpha = 0.1 \Rightarrow$ Hyperparameter

$$\text{XGBoost Classifier} = \text{Base Learner} + \alpha_1(DT1) + \alpha_2(DT2) + \cdots + \alpha_n(DTn)$$

$$\text{XGBoost Regressor} = \overset{O/P}{51K} + 0.1(5)$$

$$= 51 + 0.5$$

$$= 51.5$$

$$\text{Similarity weight} = \frac{\Sigma(\text{Residual})^2}{\text{No. of Residuals} + \lambda}$$

{Regression}

$$\text{Similarity Weight} = \frac{\sum (\text{Residual})^2}{\sum Pr(1-Pr) + \boxed{\lambda}}$$

Principal Component Analysis (PCA) [Dimensionality Reduction]

① Curse of Dimensionality

→ Dimensions →

Dataset : 500 features

Price of the House

House size
No. of bedrooms
No. of bathrooms

3 features → M1 → Acc 1

6 features → M2 → Acc 2 ↑↑ > Acc 1

15 features → M3 → Acc 3 ↑↑↑ > Acc 2

50 features → M4 → Acc 4 ↓↓ < Acc 3

← Model is overfitted

100 features → M5 → < Acc 5 ↓↓↓ < Acc 4

500 features → M6 → Acc 6 ↓↓↓↓ < Acc 5

② Model performance Degrade

Domain

Person → ↓↓↓ → Confused

Accurate

f1 2BhK

Loca ← 450K - 550K

3BhK ← 500K - 600K

Beach ← ↑↑↑

Near to a celebrity house ← ↑↑↑

Grocery shop ← ↑↑

School ←

Two different ways to remove curse of Dimensionality

① Feature Selection ↑↑

② Dimensionality Reduction (PCA) ↓↓↓

original f1 f2 f3 ... fn

Reduce ↓ Feature Extraction

D1 D2 ... O/P

Imp features

↓

Feature Extraction

Feature Selection Vs Feature Extraction

↳ Dimensionality Reduction

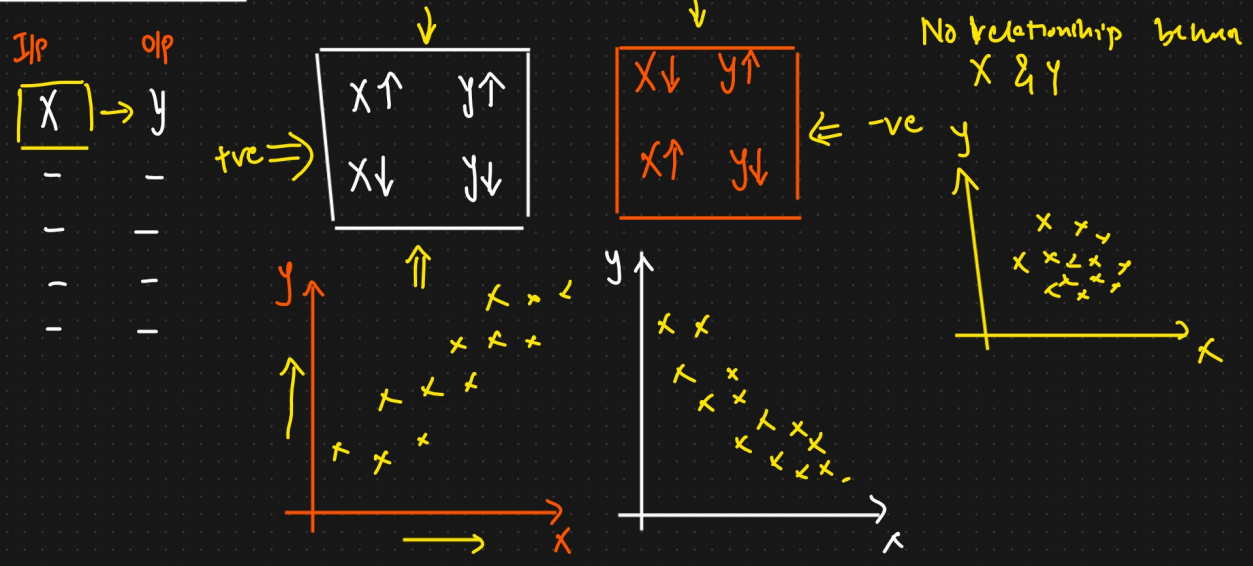
① Why Dimensionality Reduction?

- Prevent ⟶ Curse of Dimensionality
- Improve the performance of the model
- Visualize the data ⟶ understand the data

3d → 2d

100 d → 3d or 2d

Feature Selection



$$Cov(x,y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x}) * (y_i - \bar{y})}{N-1}$$

= +ve

≈0

= -ve

≈0 {No Relationship}

-ve correlated

$$\text{Pearson Correlation} = \frac{Cov(x,y)}{\sigma_x * \sigma_y} = \boxed{-1 \text{ to } 1}$$

The more toward the value of ±1 the

more +ve Correlated X & Y is

Dataset Housing

Independent feature

House Size | Fountain Size | Price → O/P feature

{Feature Selection}

DROP

Price

{Linear Relationship}

Covariance = +ve

-ve

House Size

Price of House

0 - 0.2

Fountain Size

Domain Expertise

Feature Extraction

+ve

-ve

2 feature → 1 feature

Dimensionality Reduction

{ Room Size | No. of Rooms | Price | Price

Transformation To extract New feature

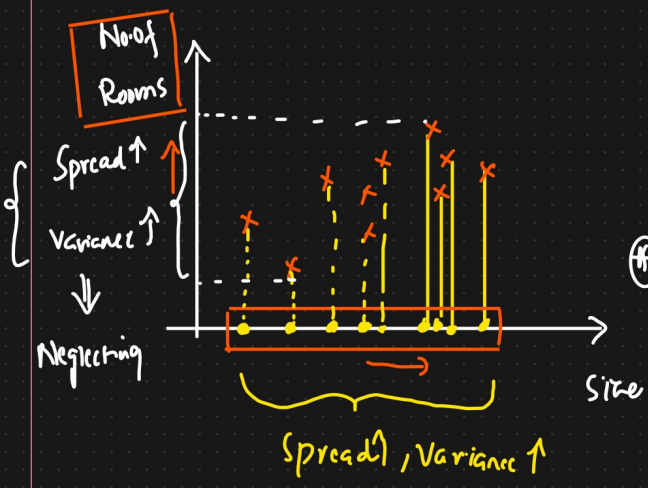
House Size | Price

PCA Geometric Intuition

{Dimensionality Reduction}

Housing Dataset

Size House	No. of Room	Price ↗ O/P



PCA

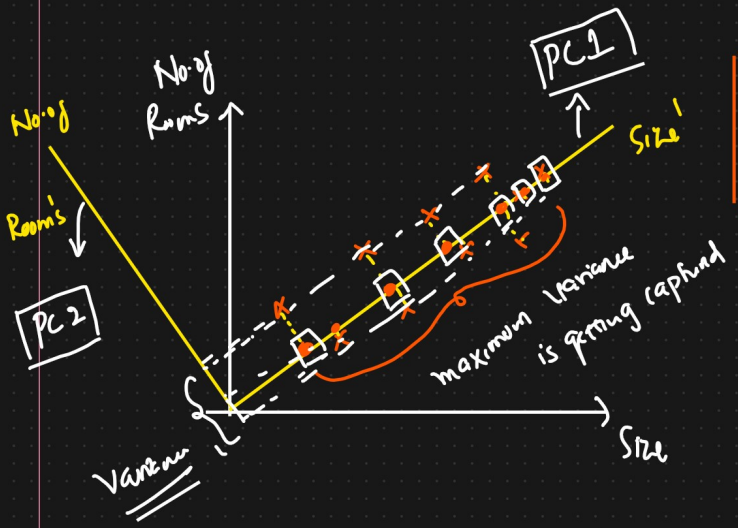
2 dimension ⟶ 1 dimension

2D → 1D

* Loss of information (No. of Rooms)

⇓

Feature Extraction



Eigen decomposition on Matrix

⇑

Transformation

2 Dimension

PC1, PC2

Var(PC1) > Var(PC2)

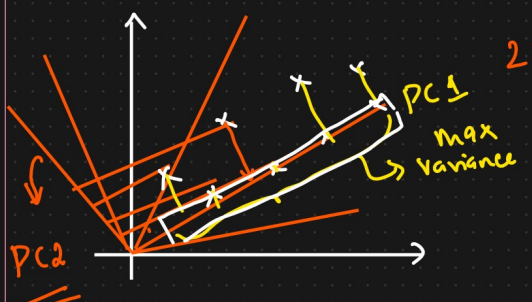
3 Dimension

PC1, PC2, PC3

Var(PC1) > Var(PC2) > Var(PC3)

2D ⟶ 1 Dimension

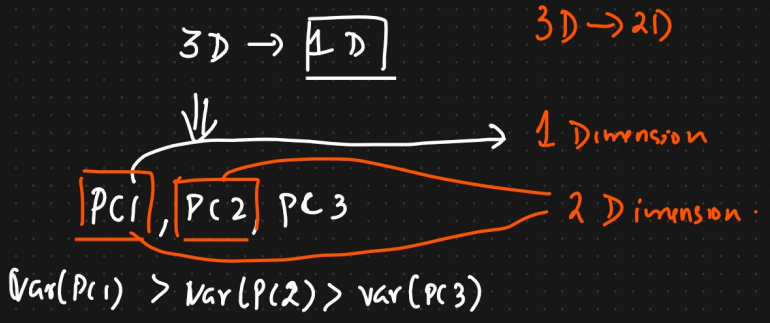
much information is not lost



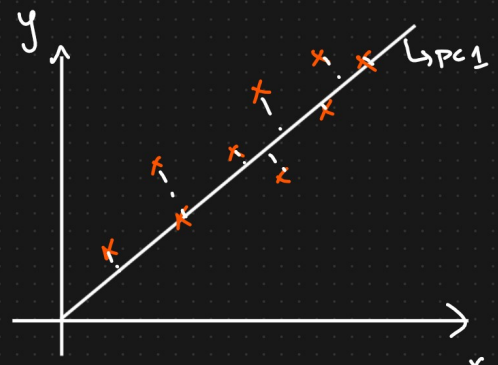
2D → 1D

2 Best Principal Component

To get the best Principal Component which captures maximum variance

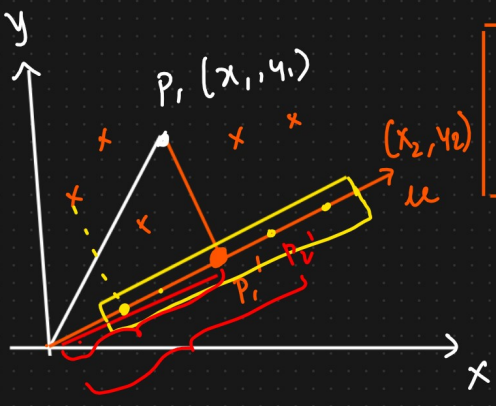


Maths Intuition behind PCA Algorithm



① Projection ✓

② Cost function → Variance ✓

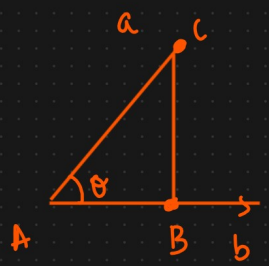


$$Proj_{P_1} u = \frac{P_1 \cdot u}{\|u\|}$$

$\|u\| = 1$ → unit vector

$$Proj_{P_1} u = P_1 \cdot u$$ ⇒ Scalar value

P_1'



$P_0', P_1', P_2', P_3', P_4', \ldots P_n'$

⇓

Scalar values

⇓

Variance

$P_0', P_1', P_2', P_3', P_4', \ldots P_n'$

⇓

$x_0', x_1', x_2', x_3', x_4' \ldots x_n'$

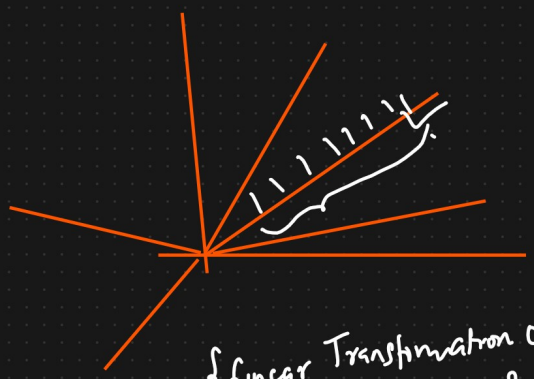
Max Variance = $\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}$ ↑

{Goal : Find the best unit vector which captures maximum variance}

⇓

Cost function

⇓ Eigen vectors And ⇓ Eigen values.



① Covariance Matrix between features

② Eigen vectors and Eigen values will found out from this covariance matrix

{Linear Transformation of matrix}

$Av = \lambda v$

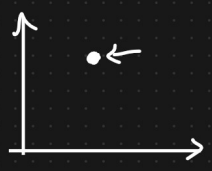
③ Eigen vector ⟶ Eigen value → magnitude of the Eigen vector ⟶ Capture the maximum variance

Eigen vectors And Eigen values [Linear Transformation]

[Eigen decomposition of covariance Matrix]

⇓

Eigen vector & Eigen values



$$\begin{bmatrix} & \\ & \end{bmatrix} * [v] = \lambda * v$$

λ ↓ Eigen value

$$A * v = \lambda * v$$

Eigen vector → Maximum magnitude

Eigen vector → Max Magnitude → Max Eigen vector → Best Principal Component → PC1

Principal Component → Max Var

Steps to calculate Eigen value and vectors

① Covariance of features

$[X, Y]$ → Z → X'

$$Cov(X,Y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{N-1}$$

2×2

A =	X	Y
X	Var(X)	Cov(X,Y)
Y	Cov(Y,X)	Var(Y)

$Cov(X,X) = Var(X)$

$Cov(Y,Y) = Var(Y)$

	X	Y	Z
X			

$\lambda_1, \lambda_2, \lambda_3$

f_1 & f_2

$A \cdot v = \lambda \cdot v$

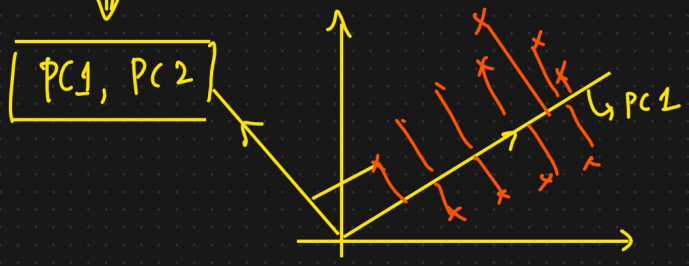
λ_1 λ_2

λ_1 ⇓ PC1 . λ_2 ⇓ PC2

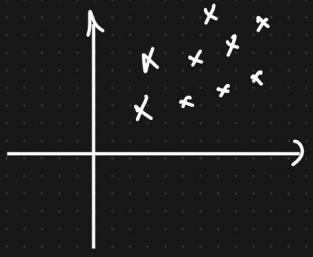
λ_1, λ_2 → Eigen values

⇓

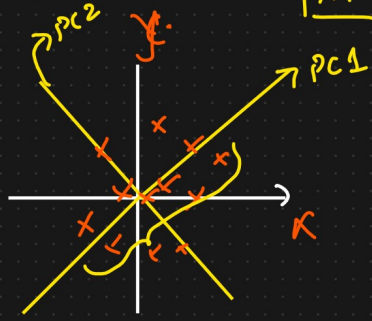
PC1, PC2



(4)



2D → 1D



λ_1 ⇒ magnitude of the Eigen vector

① Standardize the data

② Covariance Matrix of X & Y

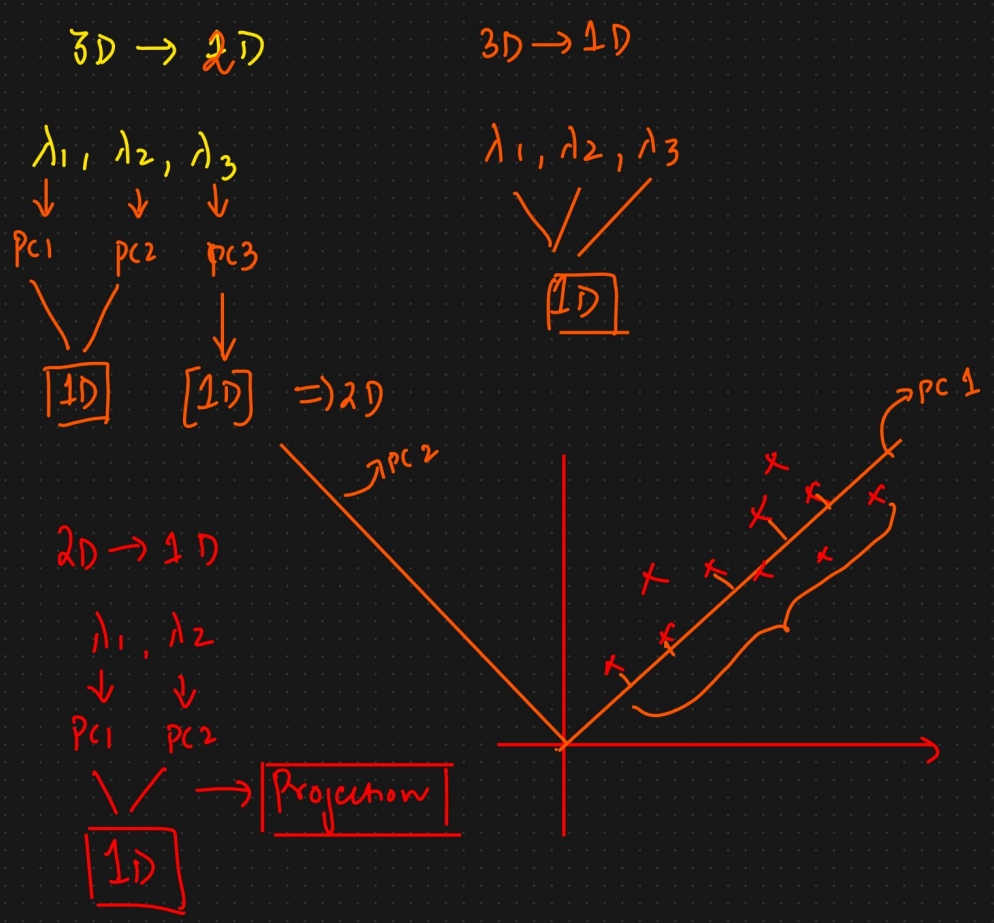
$$A = \begin{array}{c|cc} & X & Y \\ \hline X & Var(x) & Cov(x,y) \\ Y & Cov(y,x) & Var(y) \end{array} \quad 2 \times 2$$

③ Find out Eigen vectors And value

$A v = \lambda v$

λ_1, λ_2 ⇒ Eigen values.

↓ PC1 ↓ PC2


3D → 2D
λ1, λ2, λ3
PC1
PC2
PC3
1D
1D
=>2D
PC2
2D → 1D
λ1, λ2
PC1
PC2
Projection
1D
3D → 1D
λ1, λ2, λ3
1D
PC 1

Unsupervised Machine Learning

Supervised ML → Regression
→ Classification

Dataset

I/p features | Dependent or O/p features

f1 f2 f3 f4 | O/p

Unsupervised ML → Clustering {group your data into similar clusters}

Dataset

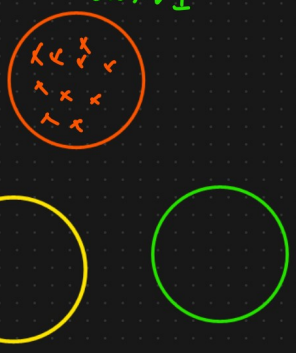
Age	Experience	Salary
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—

No O/p

⇒ Unsupervised ML

⇓

Group or cluster this data

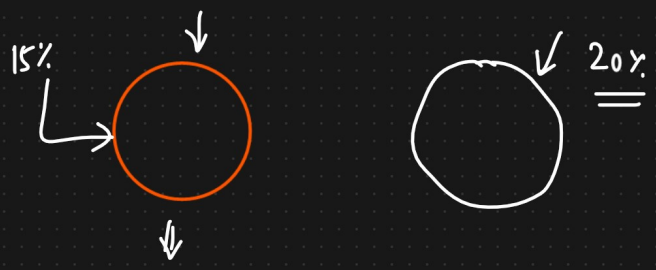


Eg: Customer Segmentation

Apple

Product : Data | Salary | Spending Score

↳ New Product ← Discount



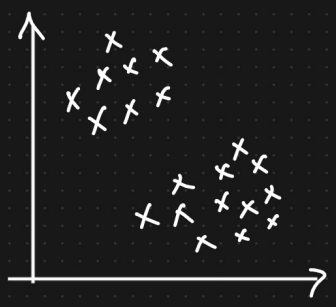
Unsupervised ML

1. K Means Algorithm
2. Hierarchal Clustering
3. DBScan Clustering

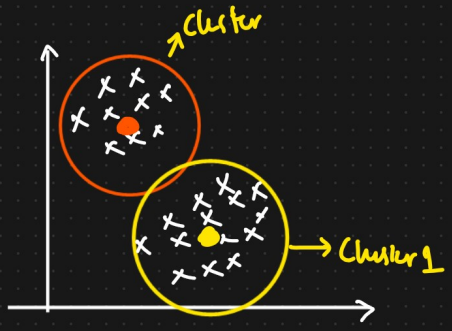
Validate

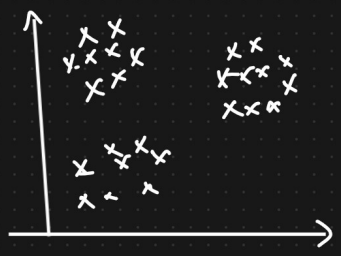
4. Silhoutte Scoring

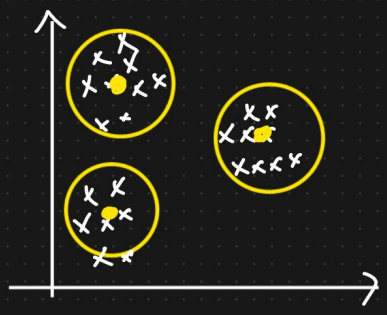
K Means Clustering Algorithm



K Means







Euclidian Distance
OR
MANHATTAN Distance

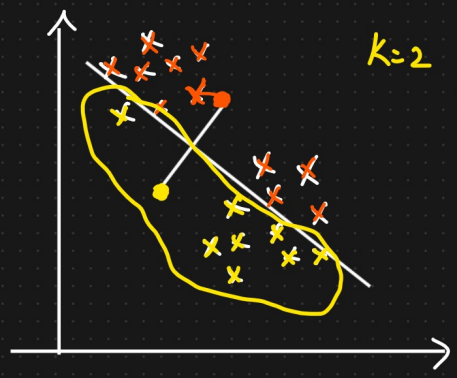
K=2

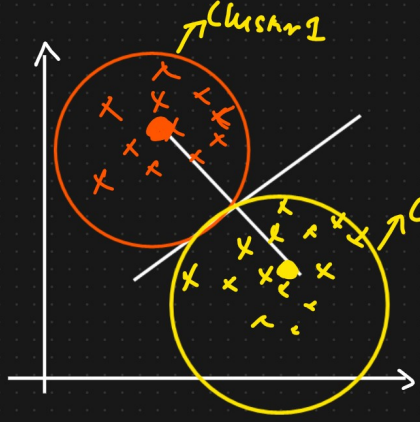
K Mean

① Initialize Some K → centroids

② Points that are nearest to the Centroids → Group

③ Move the centroids → Average

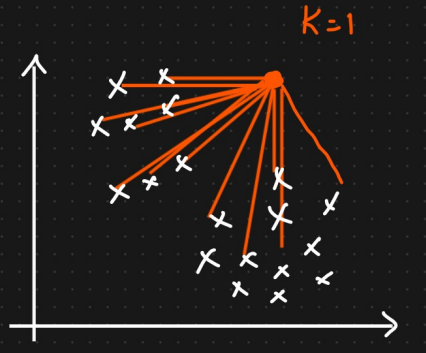




How do we select the K value?

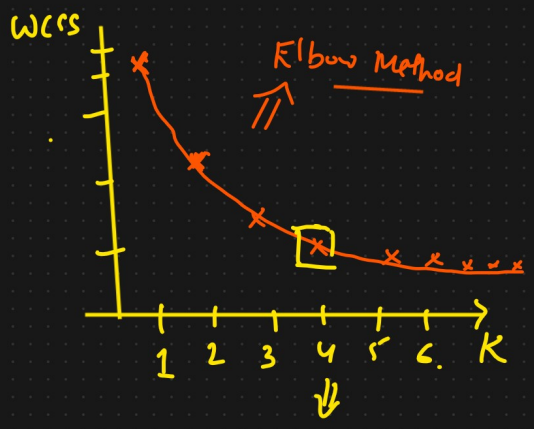
WCSS = Within Cluster Sum of Squares

Initialize K=1 to 20

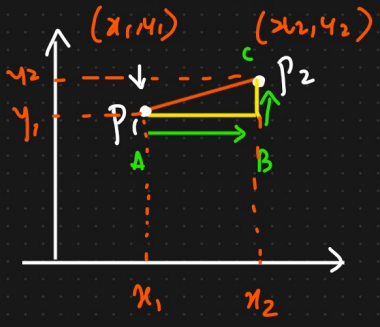


$$WCSS = \sum_{i=1}^{n} \left(\text{distance between points to nearest centroid}\right)^2$$

WCSS ↓



ⓕ Euclidean Distance



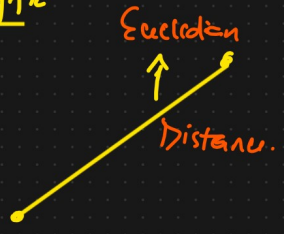
$$\text{Euclidean dist} = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

$$\text{Manhattan dist} = |x_2-x_1| + |y_2-y_1|$$

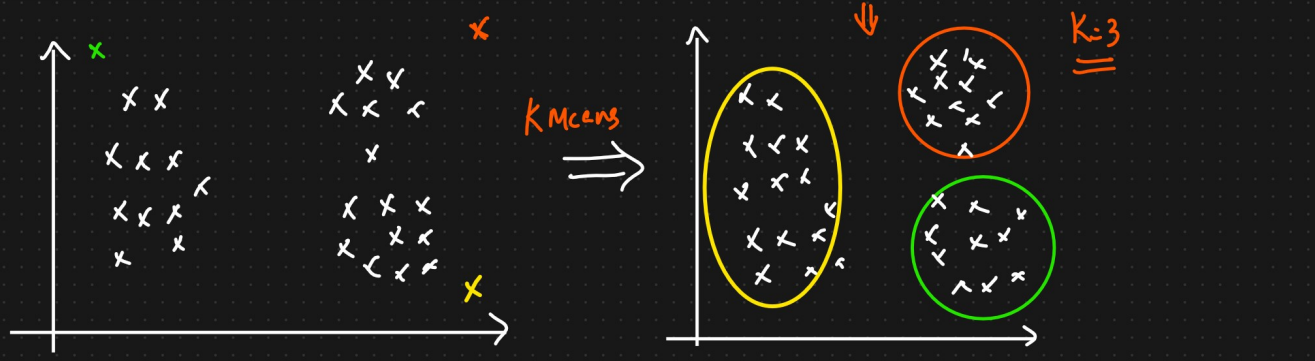
IRON MAN → U.S

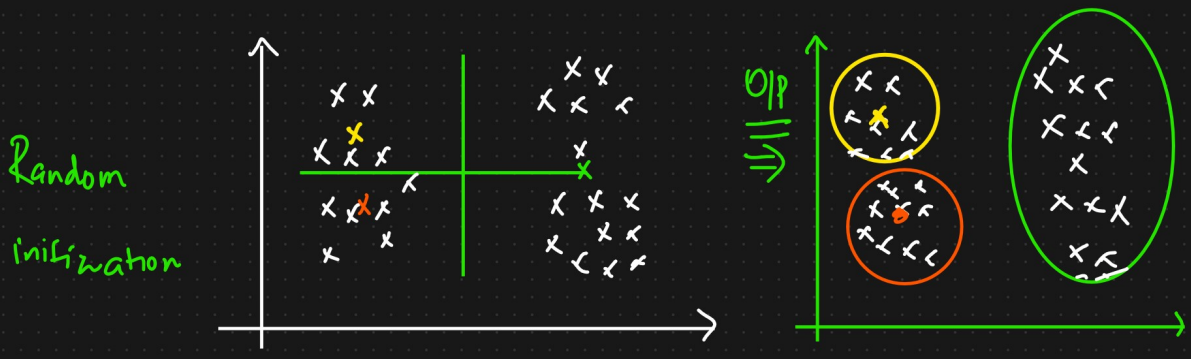
⇒ Manhattan distance

Air Traffic



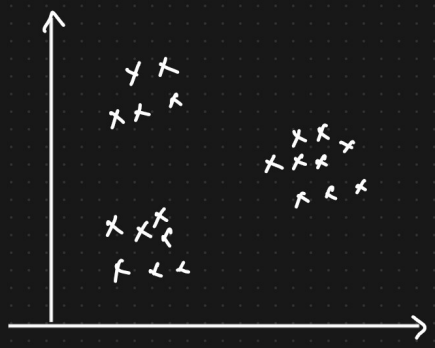
Random Initialization TRAP (Kmeans ++)



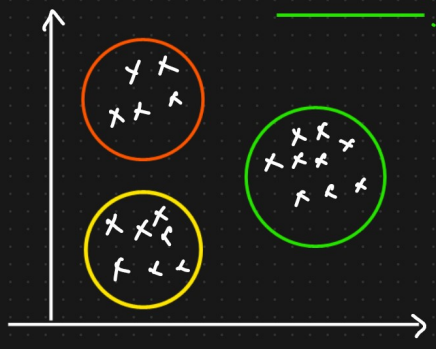


Kmeans ++ Initialization Technique

Hierarichal Clustering

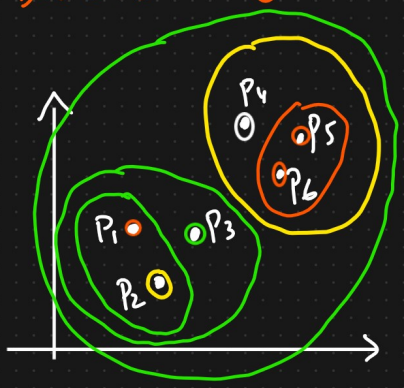


⇒ Hicrarichal Clustering



HC

① Agglomerative
② Divisive
⇒ Geometric Intuition.



Steps

① For each point initially will consider it as a Seperate Cluster

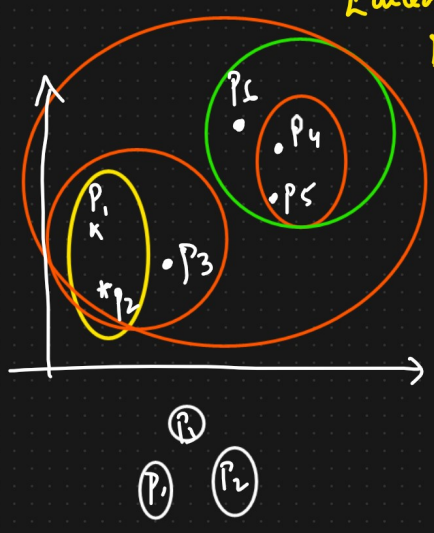
② Find the nearest point and create a new Cluster

③ Keep on doing the same process until we get a single Cluster

Dendogram

No. of clusters

Euclidian Distance



Threshold

K=2

K=4

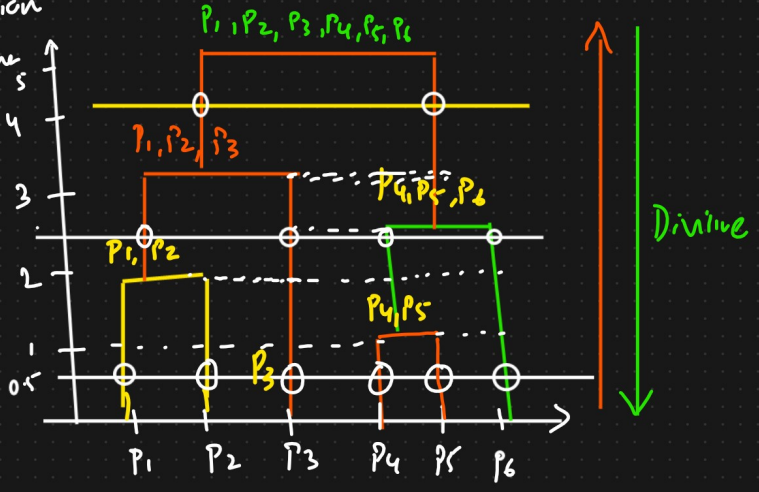
Dendogram

K=2

K=1

Agglomerative

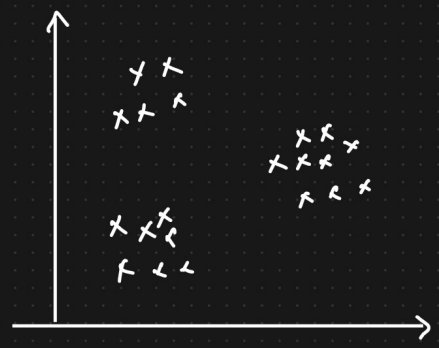
Euclidian Distance



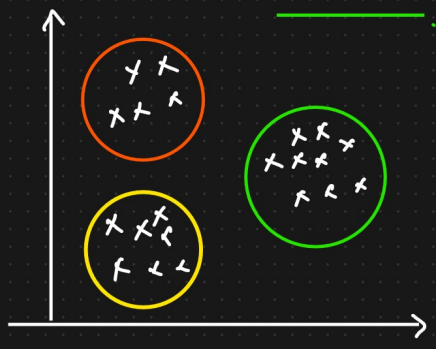
(K) Select the longest vertical line such that no horizontal line passes through it }

Threshold {Euclidean Distance}.

Hieraridal Clustering

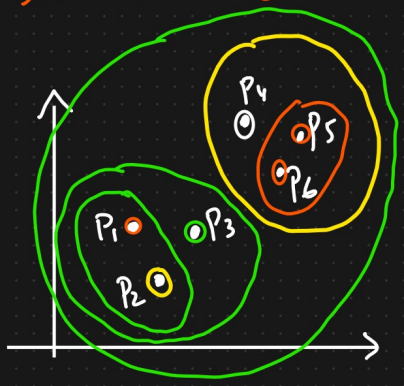


⇒ Hieraridal Clustering



HC

① Agglomerative
② Divisive
⇒ Geometric Intuition.



Steps

① For each point initially will consider it as a separate cluster

② Find the nearest point and create a new cluster

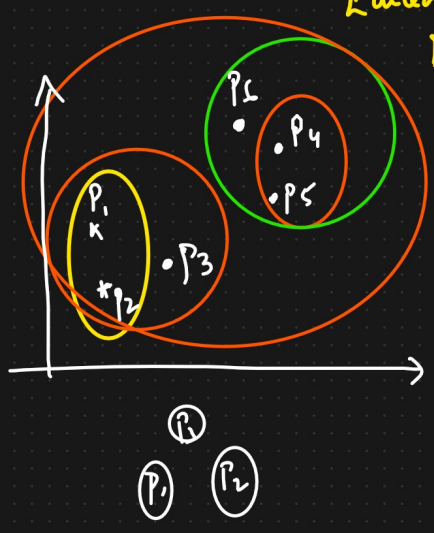
③ Keep on doing the same process until we get a single cluster

Cosine Similarity

No. of clusters

Dendogram

Euclidian Distance



Threshold

K=2

↑↑↑↑

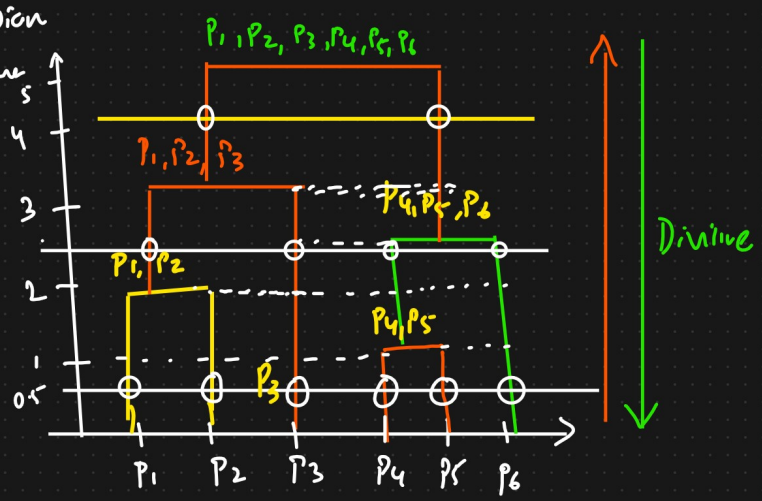
K=4

Dendogram

K=2

K=1

Agglomerative



④ Select the longest vertical line such that no horizontal line passes through it

Threshold {Euclidean Distance}.

K Means Vs Hierarchical Clustering

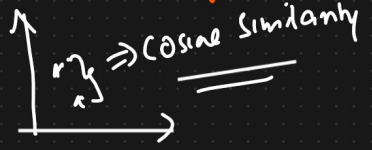
Scalability And Flexibility

① Dataset size → Huge → K Means

Small → Hierarchical Clustering

② Kmean → Numerical data

Hierarchical clustering → Variety of data.



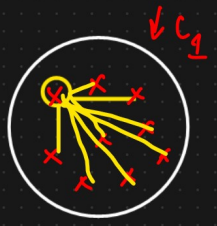
③ Centroids → Elbow method → No. of centroids

→ No. of Clusters

Silhouette (Clustering)

①

a(i)



For data point $i \in C_I$ (data point i in the cluster C_I), let

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I, i \neq j} d(i, j)$$

be the mean distance between i and all other data points in the same cluster, where $|C_I|$ is the number of points belonging to cluster i, and $d(i, j)$ is the distance between data points i and j in the cluster C_I (we divide by $|C_I| - 1$ because we do not include the distance $d(i, i)$ in the sum). We can interpret $a(i)$ as a measure of how well i is assigned to its cluster (the smaller the value, the better the assignment).

a(i)

②

We then define the mean dissimilarity of point i to some cluster C_J as the mean of the distance from i to all points in C_J (where $C_J \neq C_I$).

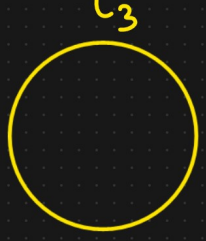
For each data point $i \in C_I$, we now define

$$b(i) = \min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i, j)$$

⇒ b(i)

a(i) << b(i)

to be the *smallest* (hence the $\min$ operator in the formula) mean distance of i to all points in any other cluster, of which i is not a member. The cluster with this smallest mean dissimilarity is said to be the "neighboring cluster" of i because it is the next best fit cluster for point i.



a(i)

② Silhouette Score

We now define a *silhouette* (value) of one data point i

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_I| > 1$$

−1 to 1

and

$$s(i) = 0, \text{ if } |C_I| = 1$$

More near to 1 better

Clustering model we have created

Which can be also written as:

+ve. ≈ 1

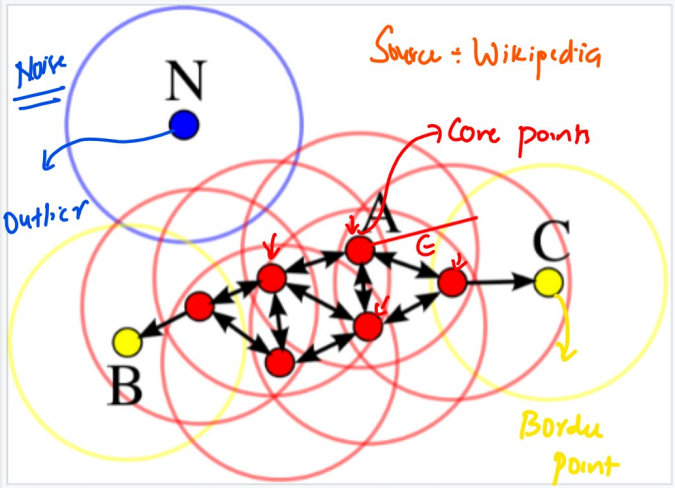
$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

≈ −1

From the above definition it is clear that

$$-1 \leq s(i) \leq 1$$

DBSCAN Clustering.



In this diagram, minPts = 4. Point A and the other red points are core points, because the area surrounding these points in an ε radius contain at least 4 points (including the point itself). Because they are all reachable from one another, they form a single cluster. Points B and C are not core points, but are reachable from A (via other core points) and thus belong to the cluster as well. Point N is a noise point that is neither a core point nor directly-reachable.

- Core point
- border point
- Outlier

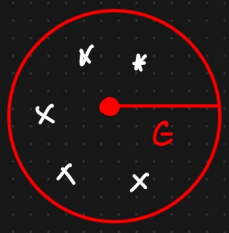
} Non linear Clustering

minpts = 4 ϵ = radius

Core point

minpts = 4

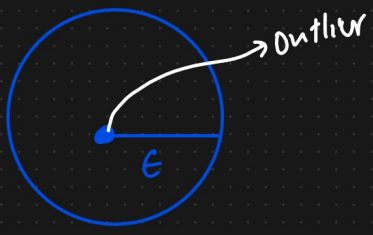
① No. of points within the ϵ should be greater ≥ 4



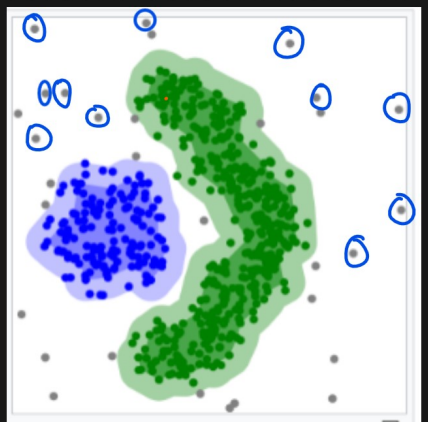
Border point

no. of data points within this radius will be less than minpts

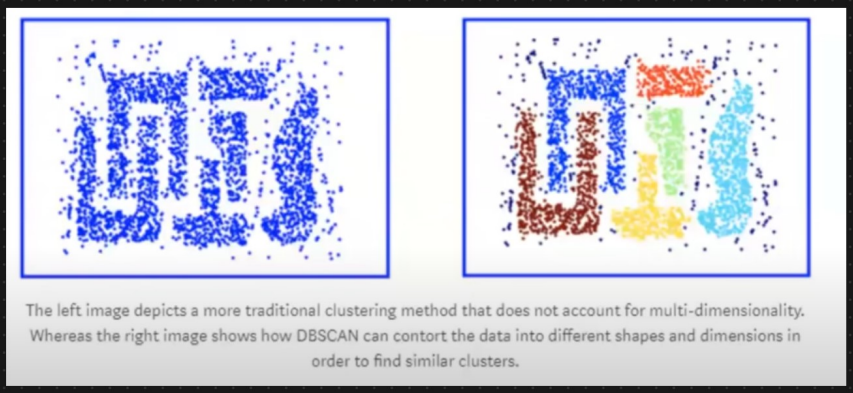
Outlier (Noise)



Some Examples after we apply DBScan Clustering



DBSCAN can find non-linearly separable clusters. This dataset cannot be adequately clustered with k-means or Gaussian Mixture EM clustering.



The left image depicts a more traditional clustering method that does not account for multi-dimensionality. Whereas the right image shows how DBSCAN can contort the data into different shapes and dimensions in order to find similar clusters.