ROC and AUC

(used for binary classification)

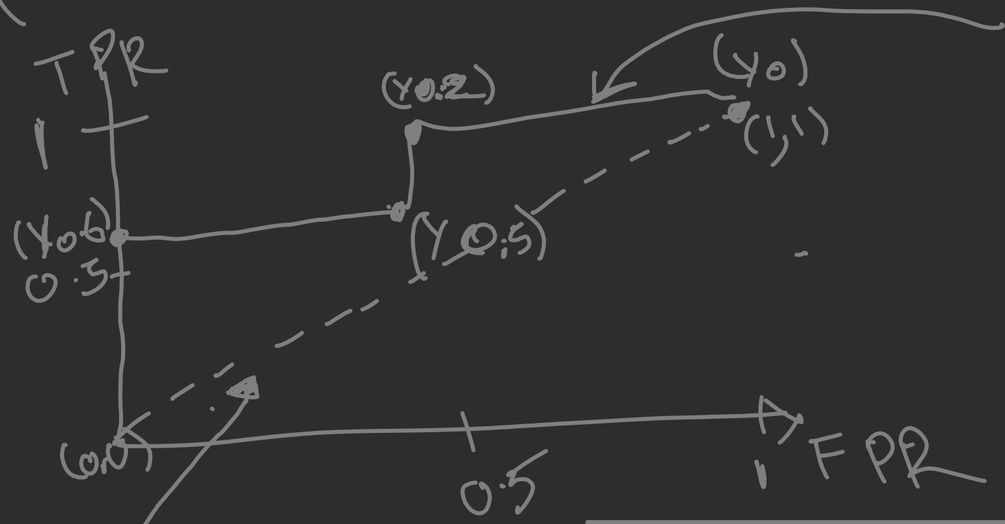
used for threshold selection for algo like logistic Regression

True P. rate = TP / TP + FN

False P rate = FP / FP + TN

y	$\hat{y}$	$\hat{y}(0)$	$\hat{y}(0.2)$	so on
1	0.8	1	1	
0	0.9	1	0	

	y_0	$y_{0.2}$	$y_{0.5}$
TPR	1	1	0.5
FPR	1	0.5	0.6



This curve is ROC and Area under it is AUC

good ROC is which do not come below dashed line

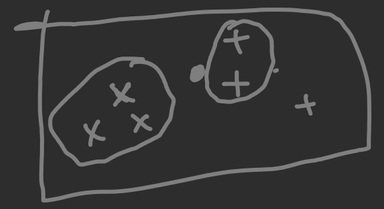
for	TPR only use	$y_{0.6}$
"	FPR only	$y_{0.5}$
"	Mixed	$y_{0.2}$

KNN (classification)

K = number of Neighbours

and from the new point which is prominent class

use distance euclidean, Manhatan, Maximan



K=5
classified as
X class

Take No imbalanced data, No outlier

(regression use)

Average of K neighbours and assign the point

Ensamble Technique

many model are used

$\boxed{\text{Hyperparam is Num. of tree / models}}$

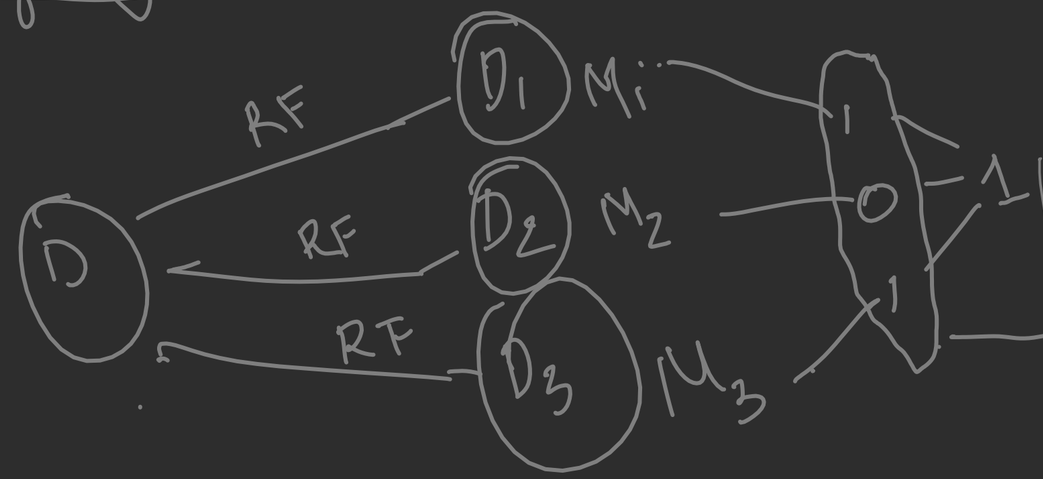
Bagging - it is the act of observing different models to get a unbiased result (Random forest)

Bootstrap - Aggrigation => it is

High variance convert to low variance

Data ($D_1 \neq D_2 \neq D_3$)
Model ($M_1 \neq M_2 \neq M_3$)

Row sampling = RF
Feature " "



Note for regression we just take mean of these But in Classification take mode

Undersampling must be used when have large amount of data
example (near miss function in Imbalance Package)
Oversampling - efficent and No data loss
(SMOTETomek in I " ")

SMOTE - it is a oversampling method (minority oversample tech)
Tomek - it is a undersampling "

Curse of Dimensionality - more features / large like 100 is bad for ML Algo Aka High Varience

Note every PC is rightangle to each previous

1) PCA - principal component analysis is unsupervise ML Algo
step 1 -> standard Normalize data then apply PCA

IMP

data -> covarience -> eign value/vector -> sort wrt eign values in Desc
take top n eign value and vector that resemble most IMP features
n = no. " " you want

Randomize Search -> it is like grid search but take permutation Randomely for best parameter to be found for to used in any Algorithm for good results.

Note -> used for Hyperparameter Optimization / setting

Boosting

Base learner (aka models) are created in which is sequentialy increasing Accuracy as compare to predecessor

i) XG Boost - Binary classifier used in Backend when using multiclass-classifier.

extreem gradiant boost was created to deal with large data

Residual = original − pred

initial pred = 0.5

step X → gain (left + R - Root ∈ {Similarity score (Residual)})

step1 → randomly select a root for Decision tree and its random / specific value then do step X

step2 → step1 with different value of a feature

doing Regression

step3 → select which information gain is more

step4 → iterate the 1-3 to extend leafs of tree according to information gain

step 5 → prune using λ (good = 1)

must be positive if -ve prune the branch

Pruning must be of leaf not root even they are eligable for it or not

λ (lambda) - regularization parameter

Ada Boost → the misslable data of 1st model is passed to that of second and that of second passed to third

weight are change for misslabled data by previous

new weight = old weight × $e^{-\text{Performance (similarity score)}}$

then normalized values of weight

then next model will take the more weighted as input

next is like Xgboost

Adaboost - Aka apes are strong together

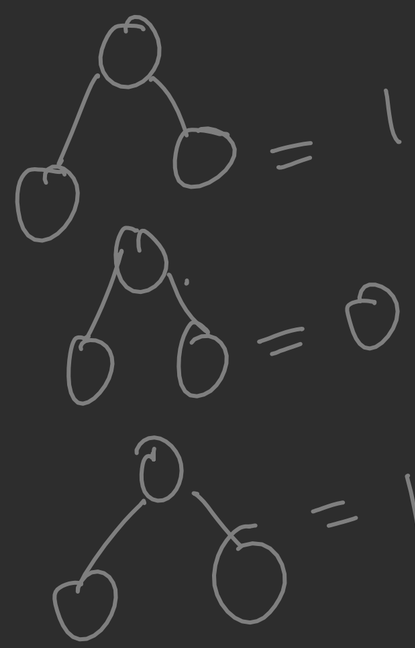
apes = stumps (a tree with one root two leaf not good for accuracy when used alone)

Ada have sequential stump making to overcome the error from previous and consider by say value

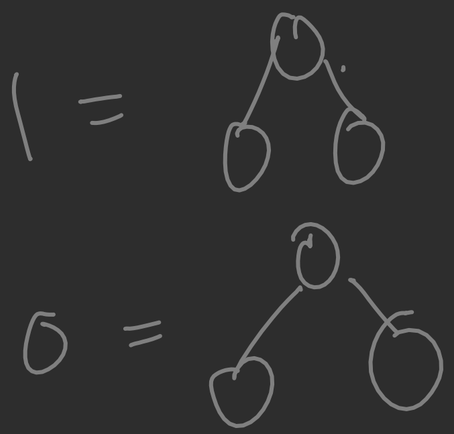
$$\text{Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total error}}{\text{Total error}}\right)$$ row wise error

yes

result = yes



No



K mean Clustering

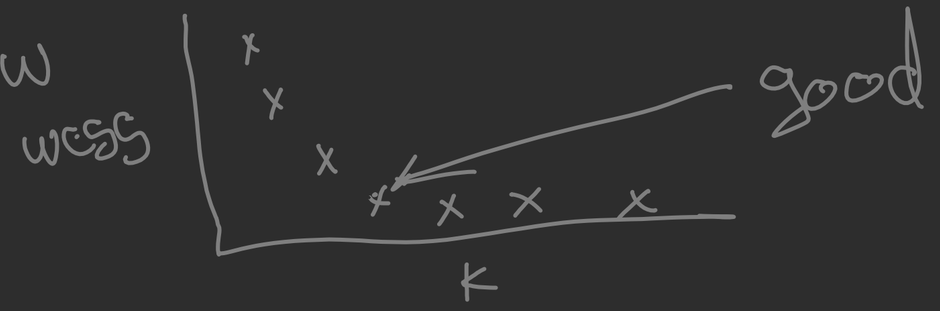
- K = no. of classes to divide the data into or centroids
- elbow method used to find optimum K value
- matrix used are mahatan, Eucladian

main steps

1) K value select 2) initialize K centroids in randomly
3) draw straight line with a normal → sumate avg distance
4) assign centroid the
5) iterate 2 – 4 6) centroids will be such that their mutual distance is maximum and the group SD is minimum

6) aka WCSS (within cluster sum of square)
that must minimize

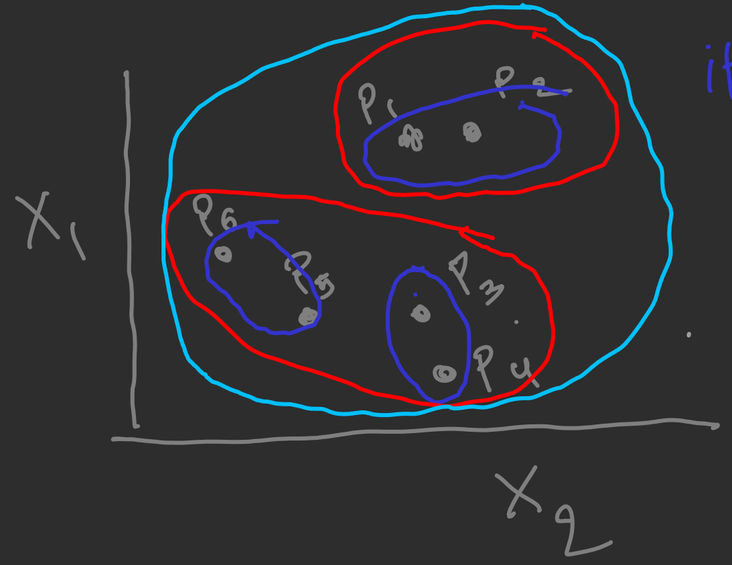
Elbow



<u>Hierarchical</u> <u>Clustering</u> - initially N data will be N cluster

step 1 = find pairs of all most near clusters. label them cluster

step 2 = repeat step 1 with new clusters

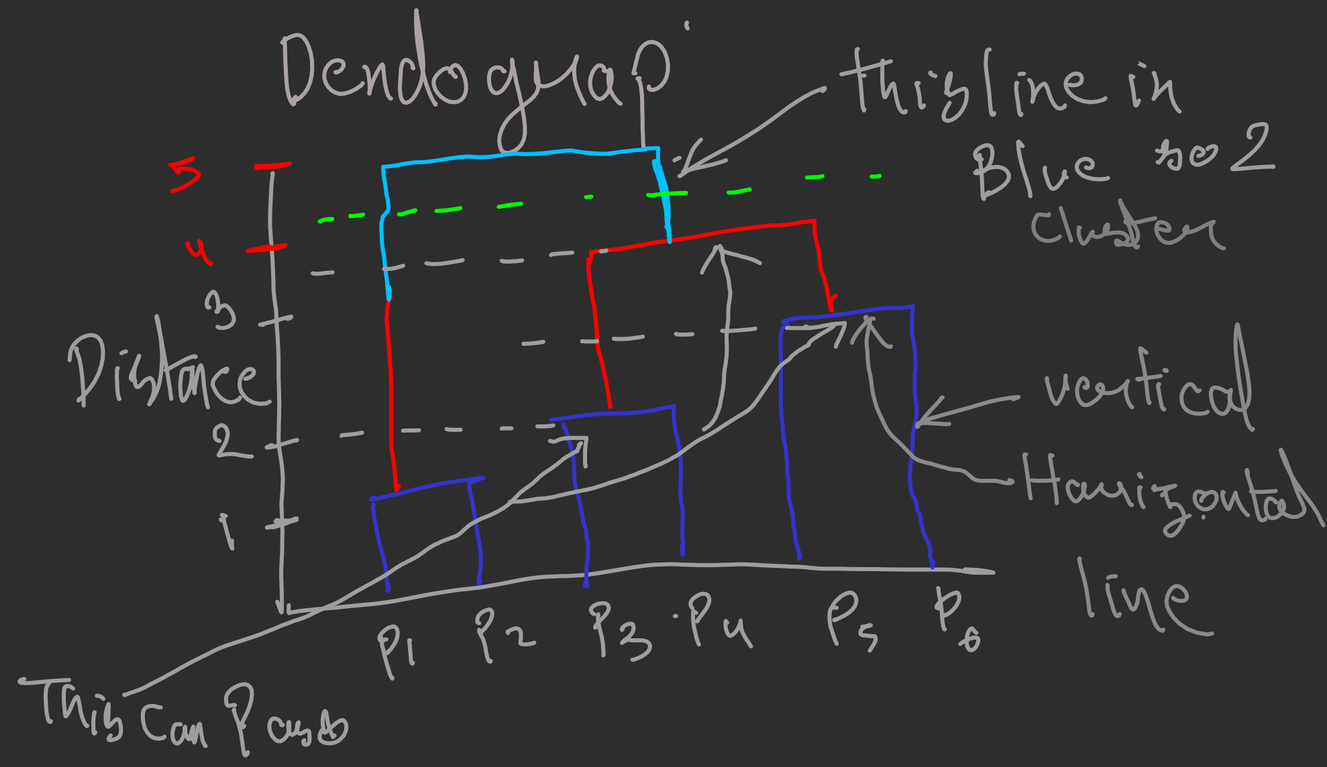


iteration 1 = ●

" 2 = ●

iteration 3 = ●

as $6/2 = 3$



find number of cluster?

1) longest vertical line with no horizontal line passing through it extends.

<u>DBScan</u> - <u>D</u>ensity <u>B</u>ased <u>S</u>patial <u>C</u>lustering of <u>a</u>pplications with <u>N</u>oise

Epsilon → some radius around each point of data

Min Points → number of mini point that must lie in the boundary of a point with Epsilon(ε)

If a point with $0 < E$ and Minpoint is a <u>core point</u>

if " " $0 < E$ only one core point is <u>Border Point</u>

If a point with $0 < E$ and no point <u>Noise point</u>

DB work good with noise and <u>density</u> <u>based</u> Cluster and collect core point and Border Point create a cluster.

Silhoutte validation technique –

a) find the intracluster distance => a

b) find the intercluster " => b

=> min(b)

if a<<b
good
else
Not

<u>Curse of Dimensionality</u> – more features / large like 100 is bad for ML Algo Aka High Varience

Note every PC is rightangle to each previous.

1) PCA – principal component analysis is unsupervise ML Algo

step 1 -> standard Normalize data then apply PCA

data -> covariance -> eign value/vector ---> sort wrt eign values in Desc
take top n eign value and vector that resemble most IMP features
n = no. " " you want

Cross validation (CV)

Problem random state of train test split can change the accuracy of model

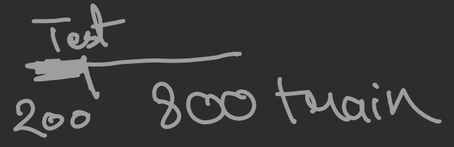
Solution

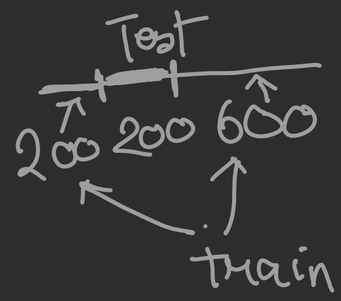
1) leave one out CV - only one point for test in a iteration then move test point to next (Bad)
low bias but high variance

2) <u>K-fold CV.</u> – K is the number of experiment perform

let $K = 5$

let $\frac{1000 \text{ data}}{5} = 200$ frame/window with 200 stride

Item1 = 

, Item2 = 

so, or

then avrage mean with K accuracy

bad for data with low spread of catigories or imbalance

no. of A << B

example class

K=1	Train	Test	K=2	Train	Test
A	50	25		20	10
B	1000	500		1030	515

Stratified CV - consider the random sample is eleminating data imbalance first then its K-fold

like = 60:40, 50:50, 70:30 only

Time Series CV - data must include Time Series means it will use only previous date or time

Baye's theorem - if this happens then what is the probability of happening next if the prior happend

Dependent

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$ = probablity of A happening if B happend

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

$$\boxed{P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}}$$

$P(A)$ → Prior P

$P(B)$ → Marginal P

$P(A|B)$ → Posterior P

$P(B|A)$ → Liklyhood P

Naive Baye's Classification

- feature - x_1, x_2

$$P(y|x_1, x_2) = \frac{P(x_1|y)\,P(x_2|y)\,P(y)}{\boxed{P(x_1)\,P(x_2)}}$$

$\boxed{P(x_1)\,P(x_2)}$ → This will be constant

$$P(y \mid x_1, x_2) \propto P(x_1 \mid y)\, P(x_2 \mid y)\, P(y)$$

$$y = \underset{\substack{\text{from} \\ y_1 \to y_n}}{\operatorname{argmax}} \left(P(x_1 \mid y)\, P(x_2 \mid y)\, P(y) \right)$$

example -

Q

Today(Sun, Hot)

$$P(\text{yes} \mid \text{Sun}, \text{Hot}) = \frac{P(\text{Sun} \mid \text{yes})\, P(\text{Hot} \mid \text{yes})\, P(\text{yes})}{P(\text{today})}$$

$P(\text{today}) \to P(\text{Sun})$, $P(\text{today}) \to P(\text{Hot})$

for yes Ans $= 0.031$

$$P(\text{No} \mid \text{Sun}, \text{Hot}) = \frac{P(\text{Sun} \mid \text{No})\, P(\text{Hot} \mid \text{No})\, P(\text{No})}{P(\text{today})}$$

$P(\text{today}) \to P(\text{Sun})$, $P(\text{today}) \to P(\text{Hot})$

for NO Ans $= 0.08571$

Normalize

$$P(yes) = \frac{0.031}{0.031 + 0.08571} = 0.27$$

$$P(No) = 1 - 0.27 = 0.73$$

$P(NO) > P(yes)$

classified $\rightarrow$ NO

Naive Bias for NLP

Sentiment analysis

Data

This is good

This is not good

$$P(+ve \mid not\ good) = \frac{P(not \mid +ve)\, P(good \mid +ve)\, P(+ve)}{P(sentance)}$$

This	is	good	Not	+ve
1	1	1	0	1
1	1	1	1	0

$P(+ve \mid not\ good) = 0.032$, $P(-ve \mid not\ good) = 0.078$

$$P(+ve) = \frac{0.032}{0.032 + 0.078} = \frac{0.032}{0.110} = 0.29$$

$$P(-ve) = 0.81$$

as $P(-ve) > P(+ve)$

The word "Not good" is not good

SVM - support vector Machine

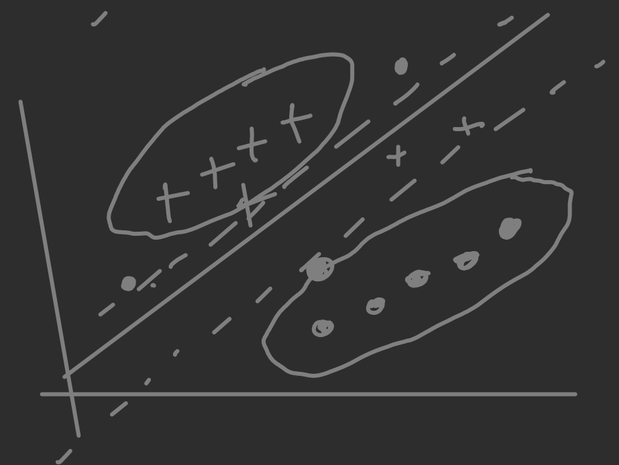
1) three Hyperplain (one touch a near +ve, one for -ve, one for buffer)
2) marginal distance must be maximum b/w that of +ve and -ve
3) support vector are those lie on marginal plane (+ve, -ve)
4) SVM kernel used to convert lower dimension data points to higher

5) it is like logistic regression but with margin

Regularization is done by adding No of misclasses and and magnitude of those misclasses from respective marginal plane/vector

Hard margin have 0 misclasses -

Soft margin have some -



SVM Kernels = to solve nonlinear seperable classification using Transformation of data to higher dimension

Default is **linear kernel**

1) polynomial kernel - standardize data then use $x \to x^2$ or $x^3 \dots x^n$

where X can be nonlinear seperable

Transpose

$$f(x_1, x_2) = \left(x^T_{1,2} \cdot x_{1,2} + 1\right)^d$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 \end{bmatrix} = \begin{bmatrix} x_1^2 & x_1 x_2 \\ x_1 x_2 & x_2 \end{bmatrix} = \text{select distinct } \{x_1^2, x_2^2, x_1 x_2\}$$

so feature synthesis give $\to x_1, x_2, x_1^2, x_2^2, x_1 x_2$

other Types

1) RBF kernel

2) Sigmoid Kernel

gini impurity - is used to get how impure the split created by a feature is of certain node

gini for leaf = $1 - (\text{probab of yes})^2 - (\text{probab of No in node})^2$

and for root node

(lower gini is good)

gini for root = weighted average of gini of leaf

$$= \left(\frac{144}{144+159}\right) \times \text{gini of A} + \left(\frac{159}{144+159}\right) \times \text{gini of B}$$

Alternative of gini

Information gain not usefull as computation heavy

Entropy - measure the impurity for a feature of split

(lower the better)

Random forest – original data set → Bootstrapped data set (AKA Random select with repl) (1) → Tree with random columns of Bootstrap data (2) → select best splitting variable with random exclusion of some variable (3) → Repeat (3) excluding variable of previous to make of any height > 2 (4) → Repeat (1) to (4) for n > 100–50 depending on problem (5) → Random forest created → Run new data to every tree in the forest → use polling to find result ← a type of aggri (mean, max, mod)

Bagging → Process of (1) is done then aggrigating to make decision

Out of bag data → data from original that is not used in Bootstrap

when this data used to find error it is called Out of bag error

this is then used to find accuracy

then tweak Hyperparameter base on found accuracy

Gradient Boosting — $\boxed{\{x_1, x_2\} \text{ indep}, \{y\} \text{ depend}}$ let

compute base model with output as average of $Y \Rightarrow \hat{y}$

→ compute Residual

$R_1 = \hat{Y} - Y_1 \in R^{1 \cdots n}$

$R_n = \hat{Y} - Y_n$ (1)

→ sequentially add decision tree input as x_1, x_2 with depend as R' of (1) (2)

get output R^2 repeat (2) with dependent as R^2 (3)

→ Repeat (2) (3) till R^n is very less

XGBoost

$\{x_1, x_2\}$ indep, $\{y\}$ depend

left → Classification

Compute base model with output as $\hat{y}$

y − total probability $= R$

→ Construct tree

Calculate similarity

$= \sum R / \sum TP(1-TP)$

(1)

→ for A, B, C repeat (1) (2) using other root

get output R^2

repeat (2) with dependent as R^2

(3)

→ Repeat (2) (3) till R^n is very less