Machine Learning

★ Pandas data structure

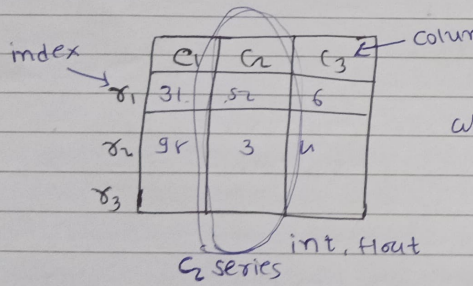
⇒ Pandas has 2 types of data structures:

a) **series**: It's a one dimention array with indexes, it stores a single column ~~of~~ or row of data in a Dataframe

b) **Dataframe**: It's tubular spreadsheet like structure representing row ~~of~~ each of which contain one or multiple column.

⇒ A 1D array (labeled) capable of holding any type of data-series

⇒ A 2D data (labeled) structure with columns of potentially different types of data - Dataframs



int, float

whole datafrume

datafrume is form by combination of series

Accuracy = $\frac{TP+TN}{TP+FP+FN+TN}$

★ Precision, Recall, confusion matix & F1 score

		Predicted Negative (0)	Predicted Positive (1)
0	Actual Negative	7, 9 / 1 5 TN True Negative	8 FP Type 2
1	Actual Positive (If 2 then actual Positive)	2 2 FN Type 1	2 / 2 TP

Positive prediction value

Precision: what persent of positive predictions made acc correct

$$\text{precision} = \frac{TP}{TP+FP}$$

TPR

Recall: what percent of actual positive value were correctly classified by classifier (sensitivity)

$$\text{Recall} = \frac{TP}{TP+FN}$$

Here we chose TP but miss some of them which are actual positive but we marked -ve so we recall them

F1-score = harmonic mean of Precision and recall

$$\text{F1-score} = \frac{2 \times \text{Precison} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{1}{\frac{\frac{1}{Pre}+\frac{1}{Rec}}{2}}$$

★ Linear regression

Linear regression cost function $J = \frac{1}{n}\sum_{i=0}^{n}(Pred_i - y_i)^2$

J is fun of m, c Here $Pred_i = mx_i + c$

$J(m,c)$ $J = \frac{1}{n}\sum_{i=0}^{n}\cdot(mx_i + c - y_i)^2$ y_i pred

⇒ our goal is to minimize cost to get best fit line we are not going to try all permutation and combinations of m & c but we gradient decent algo to get fit line

⇒ gradient decent: algorithum to find best fix for a given traing dataset in smaller number of iterations

For linear regression feature must be dependent of each others

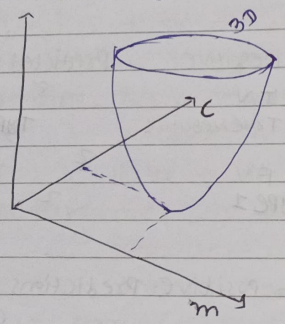
⇒ Linear regration why we don't use direct Formula for because $\beta = (x^T x)^{-1} x^T y$ has tots of computation $(x^T x)^{-1}$ is very expensive computationally

For GD $= L = \Sigma(y - \hat{y})^2 = (y - \hat{y})(y - \hat{y})$

$\omega = \omega - n \frac{\partial L}{\partial \omega} \Rightarrow \frac{\partial L}{\partial \omega} = \frac{\partial}{\partial \omega}$

$\omega = \beta$

Plot m&C against MSE



⇒ For some combination of m&C we get min error this combination of m&C gives best fit line

⇒ The algoritham is start from some value of m&C (usually stats with m=0, C=0)

⇒ calculat MSE at m=0, C=0 let assume MSE = 100

⇒ Then we reduce value of m&C by some amount (Learning step) we will notice decrease in MSE. we will continue doing the same untill Loss function is very small value or ideally 0 (which means 100% accuracy)

Step by steap algoritham

1) Let m=0, C=0 & LR = 0.01

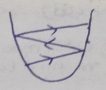
LR gives the speed where the gradient moves during the gradient decent. setting LR to high make our path instable to low makes convere slow zero means our model is not learning anything from gradients

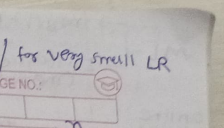
2) Calculate partial derivative of cost function w.r.t m. (with little change of m how cost function changes)

$$\frac{\partial J}{\partial m} = \frac{1}{n} \frac{\partial}{\partial m} \left(\sum_{i=0}^{n} (y_i^2 - m^2 x_i^2 + C^2 + 2mx_iC - 2y_imx_i - 2y_iC) \right)$$

$$\frac{\partial J}{\partial m} = \frac{-2}{n} \sum_{i=0}^{n} x_i (y_i - (mx_i + C))$$

$$\frac{\partial J}{\partial m} = \frac{-2}{n} \sum_{i=0}^{n} x_i (y_i - y_{i\,pred})$$





$$\frac{\partial L}{\partial \beta_m} = \frac{-2}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)\, x_{im}$$

column

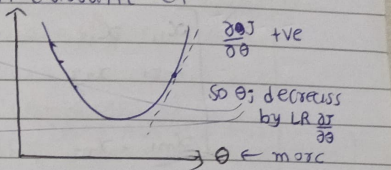
Similarly for C

$$\frac{\partial J}{\partial C} = \frac{-2}{n} \sum_{i=0}^{n} (y_i - y_{i\,pred})$$

3) update value of m&C in current eqn

$$m = m - LR \frac{\partial J}{\partial m}$$

$$C = C - LR \frac{\partial C}{\partial m}$$



4) repeat this process untill cost function J is very small (ideally =0)

* Advantages of gredient decent

Flexibility: GD is used for various cost function and It can handle non-linear regression problems

Scalability: GD is scalable for large data set since it updates parameters for each traing example one at a time.

Convergence: GD can converge to the globle minimum of cost function provided LR is set appropriately

* Disadvantage of gredient decent

Slow convergence: GD may requied more iteration to converge to min. since it updates the parameters for each traing example at a time

Local minima: GD stuck to local minima if cost function has multiple local minima

Noisy updates: upade in GD are noisy and have high variance which can make optimization process less stable and leads to oscillations around the minima

ML → model-Based learning (generalized)
→ Instance Based learning: give o/p instantly (memorized) Ex: K nearest, RBF, Kernel fun — no need to train

online / offline / out of core learning: ~~Training is done like online~~ (batch learning) but first we ~~take~~ model offline and then

* For multilinear regression (two or more features)

X = feature matrix of n×p where x_{ij} is jth feature for ith observation

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & & & \\ x_{n1} & x_{n2} & & x_{np} \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Y = response vector
y_i = response for ith observation

$y_{i\,pred} = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$, β is regression coeff.

$y_i = y_{i\,pred} + \varepsilon_i$ ← residual error

let we represent our feature matrix X as

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & & & & \\ 1 & x_{n1} & \cdots & & x_{np} \end{bmatrix}$$

$L = (y-\hat{y})^2$, $\frac{\partial L}{\partial m} = 0$, $\frac{\partial L}{\partial b} = 0$

$y = X\beta + \varepsilon$ where $\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$, $\varepsilon = \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}$

now we calculate β for which error is minimized (using least squares method)

As already explained the least squares method tends to determine $\hat{\beta}$ for which total residual error is so we present result directly

Error $E = \Sigma(y_i - \hat{y})^2$ minimized

Matrix form $E = (y-\hat{y})^T(y-\hat{y})$

$E = (y^T - \hat{y}^T)(y-\hat{y}) = [y^T - (X\beta)^T](y - X\beta)$

$E = y^Ty - y^TX\beta - (X\beta)^Ty + (X\beta)^TX\beta$

$E = y^Ty - 2y^TX\beta + \beta^TX^TX\beta$ ($y^TX\beta = (X\beta)^Ty$ ∵ 1D matrix so $A = A^T$)

$\hat{\beta} = (X^TX)^{-1}X^Ty$

$\hat{y} = X\hat{\beta}$ estimated response vector

$\frac{dE}{d\beta} = 0 = \frac{d}{d\beta}[y^Ty - 2y^TX\beta + \beta^TX^TX\beta]$

$= 0 - 2y^TX + \frac{d}{d\beta}(\beta^TX^TX\beta)$ (For this there is matrix rule: $\frac{dy}{dA} = 2XA^T$ If $y = A^TXA$)

$= -2y^TX + 2X^TX\beta^T = 0 \Rightarrow \beta = (X^TX)^{-1}X^Ty$ ($1D\ A^T = A$ so $(X^TX)^T = (X^TX)^{-1}$)

Odds: The proportion of an event's chances of happening to its chances of not happening

⇒ Regularization is also solved by direct form formula & gradient

⇒ big regu. high w & more them compare to low weight

★ Regularization

Regularization is process where we eliminate some feature which is not useful

⇒ Regularization is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting

⇒ commonly used regularization technique

L1 regularization = LASSO (Least Absolute Shrinkage and selection operator)

L2 " = Ridge regression

Dropout "

⇒ Lasso Regression add "absolute value of magnitude" of coefficient as penalty term in loss function

$\|w\|_1 = |w_1| + |w_2| + \cdots + |w_N|$

⇒ Ridge adds "squard magnitude" of coefficient as penalty term to the loss function

$\|w\|_2 = (|w_1|^2 + |w_2|^2 + \cdots + |w_N|^2)^{1/2}$

Note: During the regularization the output function y_{pred} does not change. The change is only in the loss function

The output function

$y_{i\,pred} = w_1x_1 + w_2x_2 + \cdots + w_Nx_N + w_0$

Loss function before regularization

Loss = Error($y, y_{i\,pred}$)

After Regularization

For Lasso Loss = Error($y, y_{i\,pred}$) + $\lambda \sum_{i=1}^{N} |w_i|$ λ is hyperparameter know as regularization constant & λ>0

For Ridge Loss = Error($y, y_{i\,pred}$) + $\lambda \sum_{i=1}^{N} w_i^2$

For Ridge: Error $L = (Xw-y)^T(Xw-y) + \lambda\|w\|^2$

$L = (Xw-y)^T(Xw-y) + \lambda w^Tw$

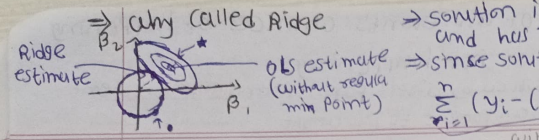
$\frac{\partial L}{\partial \beta}$ (1 β)

$\beta \rightarrow w = (X^TX + \lambda I)^{-1}X^Ty$

For gradient decend

$w = w - \eta\frac{\partial L}{\partial w}$

$\frac{dL}{dw} = X^TXw - X^Ty + \lambda w$



⇒ Why called Ridge → solution is always on circle parameters and has min distance from OLS point
⇒ since solution is on Ridge of circle so it called

$$\sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}))^2 + \lambda(\beta_1^2 + \beta_2^2)$$

loss Function part ↑ circle

wikipedia Analysis Vidhya

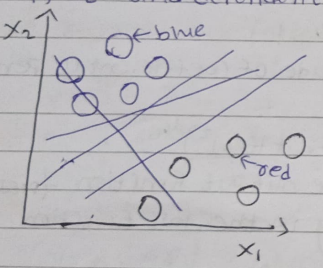
☆ Support Vector Machine

⇒ SVM is a supervised ML algo used both for classification and regression

⇒ **Objective** is of SVM to find hyperplane in N-dimentional space that distinctlly classifies the data points

⇒ The dimention of hyperplane is depends on number of input features If input features are 2 the hyperplane is just aline, If input features is three then the hyper plane become 2D plane.
It is difficult to imagine when the number exceed 3.

⇒ Lets consider input variable are only 2 (features) X_1, X_2 and dependent variable is either a blue or red circle

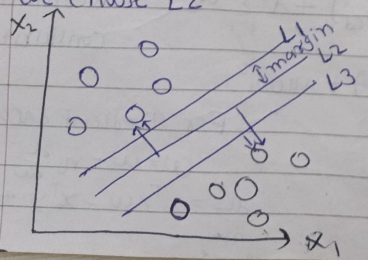


⇒ Here hyper plane is line (2 inde var)
⇒ From graph its very clear that there are multiple lines that segregrate our data points between red & blue
⇒ In this case our resonable choice as the best hyper plane

Linearly separable data Point is the one that represent largest separation or margin between the two classes

⇒ So we choose the hyper plane whose distance from it to the nearest data point on each side is maximized
⇒ If such a hyper plane exists it known as maximum-margin hyperplane or hard margin so we choose L2

case 1



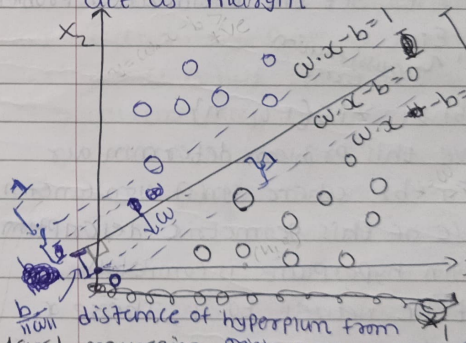
Hyperplane: In geometry, a hyperplane is a subspace whose dimention is one less than that of its ambient space.

⇒ Support Vectors are data point are closer to hyperplane and influence the position and orientation of hyperplane using these support vectors, we maximize the margin of the classifier. Deleting the support vector will change position of hyperplane. These point are help us to buid our SVM.

★ Large margin intuition

⇒ In logistic regression we take the linear function's output and squash the value within range of [0, 1] using sigmoid function and If the o.p > threshold value (0.5) then label it as 1 otherwise 0

⇒ In SVM we take the o.P of linear function and it is greater than 1 we identify it with one class and if output is less than 1 we identify is with another class since the threshold values are values ([-1, 1]) which act as margin



hyper plane can be written as set of the point x satisfying

$w^T x - b = 0$

where w is the (not necessarily normalized) normal vector to the hyperplane

Hard-margin

⇒ If traing data is linearly seperated then we can select two Parallel Hyper Plane that seperates two classes of data The region boundy by those is called "**margin**"
⇒ The maximum margin hyper plane is the ~~margin~~ hyperplane that lies halfway between them.
⇒ with normalized or standardized data set those hyperplanes can be described by the equations

$w^T x - b = 1$ (anything on or above this boundary is one class) with label 1

& $w^T x - b = -1$ (anything on or below this boundary is other class) label -1

normal to plane $(\vec{r}-\vec{a})\cdot\vec{n}_0 = 0$ → Now let Point a is Point in plane from where normal to the plane pass through origin so angle between $\vec{n}$ and $\vec{a}$ is zero

$\vec{r}\cdot\vec{n}_0 = \vec{a}\cdot\vec{n}_0$

any point on plane $d = \vec{a}\cdot\vec{n}_0$

so $d = a$

$\vec{a}\cdot\vec{n} = |a||n|\cos\theta = |a|\cdot 1\cdot 1$

$= |a|$ = distance of Point a from origin

⇒ distance between those hyperplane is $\frac{2}{\|w\|}$ so the maximize distance between them we need to minimize $\|w\|$

This distance is computed using the distance from a point to plane eqⁿ.

⇒ we also have to prevent data points from falling into the margin, we add the following constraint for each i either

$w^T x_i - b \geq 1$ if $y_i = 1$

or

$w^T x_i - b \leq -1$ if $y_i = -1$

(point should be 1 or more than 1 distance away from middle line (w is unit vector))

This constrains stelte that each data point must lie on correct side of margin. This can be written as

$y_i(w^T x_i - b) \geq 1$, for all $1 \leq i \leq n$

(y_i: actual value, $w^T x_i - b$: predicted value)

soft margin

we can put this together to get the optimization problem

we have to change w & b in such a way max $\frac{2}{\|w\|}$ →

minimize $\|w\|\frac{1}{2} + C\sum_{i=1}^{n}\varepsilon_i$ over w, b

($\|w\|\frac{1}{2}$: margin error, ε_i: classification error)

subjected to $y_i(w^T x_i - b) \geq 1 \quad \forall i \in \{1, 2, \ldots n\}$

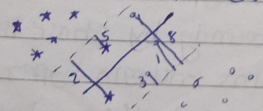
The w, & and b solve this problem determin our classifier $x \mapsto \text{sgn}(w^T x - b)$ where sgn(·) sign function

⇒ An important consequnce of this geometric description is that the max-margin hyperplane is completely determined by x_i that lie nearst to it These x_i are called support vectors.

→ Here C = How many errors model can consider

soft margin (extended)

ε_i = value of the errors



$\varepsilon = 2+5+3+1+8$

error is sum of distance between ~~time~~ Point inside and their correspondy line

$(\vec{r}-\vec{a})\cdot\vec{n} = 0$ & $(\vec{r}-\vec{a})\cdot\vec{n}_0 = 0$ (∵ divide both side by |n|)

(n not nessesary unit vector; n_0 unit vector)

$\vec{r}\cdot\vec{n} - \vec{a}\cdot\vec{n} = 0$ $\quad\vec{r}\cdot\vec{n}_0 - \vec{a}\cdot\vec{n}_0 = 0$

$\vec{r}\cdot\vec{n} - d = 0$ $\quad\vec{r}\cdot\vec{n}_0 = d$ ← distance from origin

$\vec{r}\cdot\vec{n} = d$ when normal $\vec{n}$ passes throught origin (not necessarily)

Here d is distance of plane from the origin

but $\frac{d}{\|n\|}$ is distance of plane from origin

Normal to plane constant

Features changing for each data

$w^T x - b = 0$ (hyper)

⇒ w is normal vector to plane

⇒ $\frac{b}{\|w\|}$ distance from origin

dist of Point x to plane or line

$= d = \frac{w^T x + b}{\|w\|}$ ← eqⁿ

(origin so x = 0)

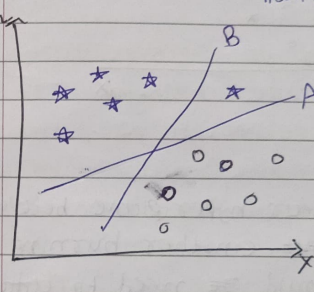
$w^T x - b = 1$ dis from origin is $= \frac{|1+b|}{\|w\|}$

correct $w^T x - b = -1$ dis from origin $= \frac{|-1+b|}{\|w\|}$

dis bet this two plane $= \frac{|1+b| - |-1+b|}{\|w\|}$ $b \geq 0$

so $\frac{2b}{\|w\|}$

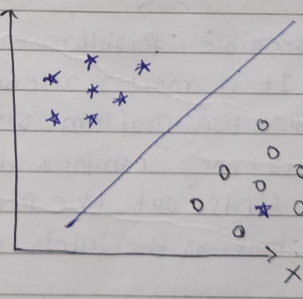
case 2



⇒ SVM select the hyper plane which classifies the class accurately prior to maximizing margin

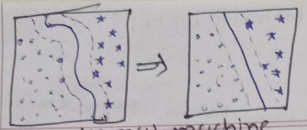
⇒ Plane B has classification error and plane A classified all correctly so plane A is correct hyper plane

case 3

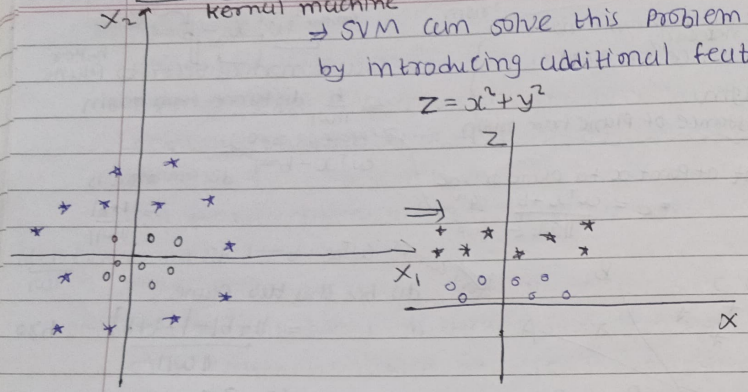


⇒ Here we unable to di segregate two class using a straight line as of one of the stars is in the Territory of other class as an outlier.

⇒ so SVM algorithm has a feature to ignor outliers and find hyper plane that has max margin



Kernal machine



⇒ SVM can solve this problem ~~easily~~ easily by introducing additional feature

$z = x^2 + y^2$

⇒ In SVM classifer, having linear hyper plane between those two classes is easy but another burning question arises that if we should ~~be~~ need to add this feature manually to have a hyperplan., NO. SVM algorithm has a technique called the Kernal trick

⇒ Kernal trick: is a function that take low dimention vector space and transfer it to higher dimention space

⇒ It convert non seperable problem to separable problem. It is mostly useful in non linear data separation problem. Simply put it does some extremely complex data transformations, then finds out the process to separate the data base on the labels or output you define.

* Advantage of SVM

⇒ effective in the higher dimention cases

⇒ It memory efficient as it uses a subset of taring point in the decision function called support vector

⇒ different Kernal function can be specified for the disision function and its possible to specify custom Kernels

* Disadvantage of SVM

difficult to choose ∴ Kernal func ↑ C ↑ varue

⇒ It's does not perform well on large data set because the required taring time is large

⇒ It's does not perform well when dataset has more noise i.e., target ~~set~~ classes are overlapping

⇒ SVM doesn't directly provide probability estimates; these are calculated using an expensive five-fold cross-validation. it is included in the related SVC method of the Python scikit learn library

⇒ SVM is work best on small data set which contain the number of dimention is grater then number of samples.

Soft margin

To extend the SVM to cases in which data are not linearly separable the hinge loss function is helpful

$\varepsilon_i = \text{Loss} = \max\left[0,\ 1 - y_i(w^T x_i - b)\right]$ $\quad y_i$ is i^{th} target (± 1)

$i \in (1, 2 \dots n)$ $\quad w^T x_i - b$ is the i^{th} output

⇒ This Loss function is zero when condition $y_i(w^T x_i - b) \geq 1$ for all $1 \leq i \leq n$ satisfied - (means x_i lies on correct side of margin)

⇒ For data on the wrong side of the margin, the function's value is proportional to the distance from the margin

⇒ The goal of the optimization then is to minimized

$$\lambda \|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \varepsilon_i$$

where, $\lambda > 0$ determines the trade-off between increasing margin size and ensuring that x_i lie on the correct side of the margin, by deconstructing the hinge loss

⇒ This optimization problem can be messaged in to the following:

So we have calculate minimum value of $\left(\frac{1}{n}\sum_{i=1}^{n}\varepsilon_i + \lambda\|w^2\|\right)$

$\|w\|^2 + C\sum_{i=1}^{n}\varepsilon_i \quad \left(C \propto \frac{1}{\lambda}\right)$

with respect to

(subject to) $y_i(w^T x_i - b) \geq 1 - \varepsilon_i$, $\varepsilon_i \geq 0 \ \forall i \in (0, 1, 2 \dots n)$

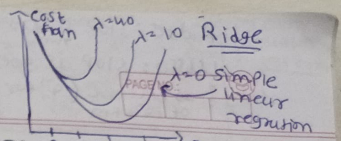
For the large value of C it behaves similar to the hard margin SVM.

Feature which is not highly correlated is minimize but not zero

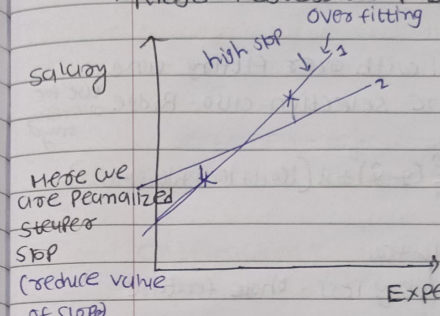
Regularization

$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots$

Here by ↑ λ value θ value become low but not zero



★ Ridge Regression & Lasso regression



Cost function for Ridge regression

$\text{Loss} = \sum_{i=1}^{n}(y - \hat{y})^2 + \lambda(\text{slope})^2$ $\quad y = mx + c$

$\lambda > 0$

We have to reduce this cost function to deal with overfitting

If there are more then one feature then $y = w_1x_1 + w_2x_2 + \dots + w_nx_n + c$

$\text{Loss} = 0 + \lambda(w_1^2 + w_2^2 + w_n^2)$

slop1 slop2 slopn

in case of overfitting

$\text{Loss} = \sum_{i=1}^{n}(y - \hat{y})^2 + \lambda(\text{slop})^2$

$0 < \lambda < \infty$

$\text{Loss} = \lambda(\text{slop})^2$

⇒ w_i is slop with respect to x_i If increase x_i by one unit then how much effect in y is calculated from w_i

let assume line 1 has slop = 1.3 & λ = 1

So $\text{Loss}_1 = 1.69$

but we can find another line 2 with has less slop then Line 1 let this line has slop = 1 $\quad \lambda = 1$

so $\text{Loss} = \sum_{i=1}^{n}(y - \hat{y})^2 + \lambda(\text{slop})^2$

value of $\sum_{i=1}^{n}(y - \hat{y})^2$ has small value let 0.39

so $\text{Loss} = 0.39 + 1(1)$

$\text{Loss}_2 = 1.39$

$\text{Loss}_1 > \text{Loss}_2$

⇒ Here we reduce overfitting by reducing slop so there will be more cost for training data then overfitting but for test data Its give generalised solution

⇒ and we reduce slop with has higher value so in unite increasing in x is less change in y

Ridge : slop is goes to zero but it is shrink not

Lusso : slop is goes to zero

Why ? ⇒ because of eqn of m (eqn for slope of Ridge & Lusso can explain (will this)

⇒ If λ is higher value of slops are goes toward zero

★ **Lasso regression** : use to deal with over fitting and use for feature selection also Ridge use for overfitting only

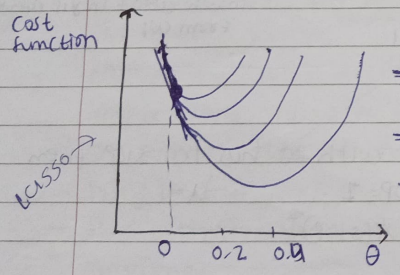
bias not change only w change

$$Loss = \sum_{i=1}^{n} (y-\hat{y})^2 + \lambda\,[slop] = \sum_{i=1}^{n} (y-\hat{y})^2 + \lambda(|\omega_1|+|\omega_2|+|\omega_3|+|\omega_4|)$$

$y = \omega_1 x_1 + \omega_2 x_2^2 + \omega_3 x_3 + \omega_4 x_4 + \omega_5$ (less slop)

⇒ where ever slope value is very very less those feature will be removed.

⇒ when ever we use |slop| magnitude in Loss function (cost fun) slop will be moving toward zero so some of slop values of feature is reduce to zero.



~~Features with not related~~

⇒ Features which not ~~high~~ highly correlated is become zero

⇒ Feature selection

⇒ reduce features which are not importan

For direct formula $\frac{\partial L}{\partial m} = 0$

Ridge : $m = \dfrac{\sum_{i=1}^{n} (y_i-\bar{y})(x_i-\bar{x})}{\sum_{i=1}^{n} (x_i-\bar{x})^2 + \lambda}$ (λ also called as α)

$\frac{\partial L}{\partial m} = 0$ $\quad L = \sum_{i=1}^{n} (y_i-\hat{y}_i)^2 + \lambda|m|$

Lusso : $m = \dfrac{\Sigma(y_i-\bar{y})(x_i-\bar{x}) \mp \lambda}{\Sigma(x_i-\bar{x})^2}$ (− for +ve, + for −ve ($\because \frac{\partial |m|}{\partial m}$ is −ve) so that term will be −ve)

$m \rightarrow \omega$ = weights = slopes

$0 < \lambda < \infty$

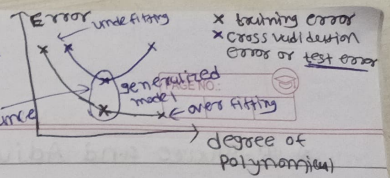
$L = \omega^T x^T x \omega - 2\omega^T x^T y + y^T y + \lambda \omega^T \omega$

$\frac{\partial L}{\partial \omega} = x^T x \omega - x^T y + \lambda \omega = 0$

$\omega = (x^T x + \lambda I)^{-1} x^T y$

Bias : accuracy in training data

Variance : accuracy in testing data



★ Elasticnet Regression (Ridge + Lasso)

$$\text{Cost function} = \underbrace{\frac{\sum_{i=1}^{n} (y_i-\hat{y}_i)^2}{2n}}_{\text{MSE}} + \lambda_1 \underbrace{\sum_{i=1}^{n} (slop)^2}_{\text{Ridge}} + \lambda_2 \underbrace{\sum_{i=1}^{n} |slop|}_{\text{Lasso}}$$

⇒ Reduce overfitting

⇒ Feature selection

⇒ use for big dataset and we haven't knowledge about which regression is better

⇒ use when data has multi colinearity

L1 ratio = $\frac{\lambda_1}{\lambda_1+\lambda_2}$

$\lambda = \lambda_1 + \lambda_2$

$L1 = \frac{\lambda_1}{\lambda}$

$\lambda_1 = \lambda \times L_1$

$\lambda_2 = \lambda - \lambda_1$

★ R-Square and Adjusted R square

⇒ In classification we check accuracy using confussion matrix, accuracy score, TP value, Recall value, precission, F1-score

⇒ But in regression problem there are different method call R square Technique.

$$R^2 = 1 - \frac{SS_{res}}{SS_{total}}$$

SS_{res} = Sum of Residual or error

SS_{Total} = Sum of Avg total

goodness of best fit line

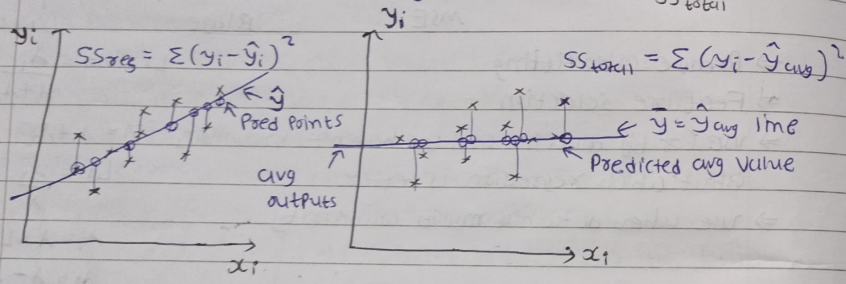
Real Points

$$R^2 = 1 - \frac{\Sigma(y_i - \hat{y}_i)^2}{\Sigma(y_i - \bar{y}_i)^2}$$

Predicted Points

Predicted avg value

$$R^2 = \frac{SS_{reg}}{SS_{total}}$$



⇒ If R^2 is less then zero then predicted model is very bad. (∵ It is worst then avg)

⇒ $R^2 \to 1$ model is best

$y = \beta_0 + \beta_1 x_1$ (simple)

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ (multiple)

$+ \beta_3 x_3$

as we adding new and new independent feachers R^2 is usally increases.

⇒ let we add $\beta_3 x_3$ feacture so our linear regression moldel assign β_3 some value such that $SS_{res} \downarrow$ so $R^2 \uparrow$

⇒ so $R^2 \uparrow$ as we added new independed features

⇒ So adding independent feature R^2 always increass it will never decreass. (∵ Even by adding Feature If weights is not change but $y_i - \bar{y}$ always $\geq y_i - \hat{y}_i$ so...)

★ ⇒ even If new added independed feacture has no correlation with target output even though $R^2 \uparrow$ so It is not penilizing the new added features. to overcome this problem we introduce adjusted R^2

$$R^2_{adjusted} = 1 - \frac{(1-R^2)(N-1)}{N-P-1}$$

where R^2 = sample R-square

P = Number of predictors (Independed features)

N = Total sample size

If ~~Independed~~ when ever Attributes increasses

$R^2_{adj} \downarrow$

⇒ If independed feacture is correlated with output then $R^2 \to 1$ and R^2_{adj} also high

⇒ If variables are not correlated R^2_{adj} slightly decreass compare to orignal R^2 value.

& If It is complatly correlated there will be slight increass in R^2_{adj}

⇒ Every time you add a independed variable to model the $R^2 \uparrow$ even If the independed variable is insignificate It never declines. whereas ★ $R^2_{adj} \uparrow$ only when ind variable is significate and affects dependent variable

$$R^2_{adj} \leq R^2$$

Hypothesis Testing

⇒ Evaluate 2 (can also more them 2) mutual exclusive statement (either this true of that) on population data using sample data

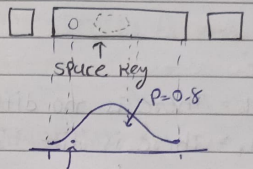
1) make Initial Assumption (H_0) (Null hypothisis (let innocent)) H_1 = alternet hypothesis (guilty)
2) Collect Data (evidencess)
3) gather evidence to Reject OR none reject null hypothesis

Actual →	H_0	H_1
H_0 DO not Reject (predicted)	OK (TP)	Type 2 error (FP)
H_0 Reject (so H_1)	Type 1 error (FN)	OK (TN)

Truth

⇒ Type 1 error is more harmful or type 2 error is more harm ful is defeds on problem statement

★ P value : Probability for null hypothesis to be true ★



Null hypothesis : Treats everything same or equal

that means If we repeat this experiment 100 times number of touching this area 0 1 time

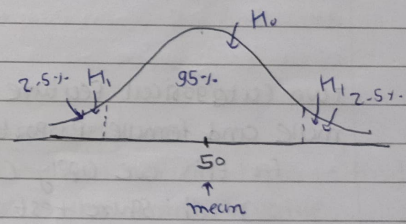
Toss experiment :

H_0 = Null hypothesis = fair coin

Toss 100 times

alternet hypothesis

H_1 = Coin is not fair



$$\text{Chi-Square} = \sqrt{\frac{(\text{Actual} - \text{Expected})^2}{\text{Expected}}}$$

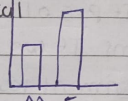
PAGE NO.:

★ Ttest, chi square test, ~~&~~ Anova test

Gender	Age group	weight (kg)	Height (m)
M	Elderly	70	1.4
F	Adult	65	1.2
M	Adult	65	1.4
M	Child	20	1.1
F	Adult	75	1.3
M	Elderly	80	1.3

H_0 = There is no diff
H_1 = There is diff

one categorical ratio of M & F is diff or not



$P \leq 0.05$ ← significance value (P value)

one sample Proprosion Test → If $P \leq 0.05$ then Reject H_0 and There is ~~d~~eff in male & female ratio

Two categorical feature : Is there any difference in male and female Proportion base on Age grop
for this we apply Chi square test
If Chi Squre test gives value < 0.05 then reject H_0

one numerical Variable

Continous variable (let consider Height)
now what is statement that coms w.r.t continous variable height! ~~easy~~
ave height = 1.3 is there any diff w.r.t mean height
Here we apply → Test : T test
base of T test If values ≤ 0.05 then reject H_0

two numerical variable

Continous variable
Test : correlation (-1 to +1) / T test
If correlation value → 0 so no correlation
H_0 = There is relation bet^n two variable
H_1 = " " no relation " " "

base on this we apply T test If $P \leq 0.05$ reject H_0

more then 2 category → Test : Anova $P \leq 0.05$ reject H_0
In case of 2 category we use T-test
(Ex In Age elderly, Adult, Child) ⇒ correlation between categories

★ Metrix In classification

Classification Problem

Class Lables ← default Probability is 0.5

In binary classification there are two classes A, B
In this method threshold value is taken 0.5
If greater then 0.5 then Its one class otherwise another class

Probabilities

In Probabilities we have to find ~~three~~ Right threshold value
It may be 0.3, 0.4 or anything bet 0 to 1

$$F_{Beta} = (1+\beta)^2 \frac{\text{Precision} \times \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

If $\beta = 1$ $F_{Beta} = 2 \frac{Pre \times RE}{Pre + RE}$

when FP & FN both are equally important then we select $\beta = 1$

$0 < \beta < 1$

⇒ If FP having more impact then FN then we Reduce β (Type 2 error)

⇒ If FN having more impact (Type 1 error) then FP the we increase β value then $1 < \beta < 10$

Logistic Regression

Maximum likelihood:

⇒ For maximum likelihood we multiple Prob of each Point (Prob of that Point is correctly classified If Point is +ve then take P(x) of that Point vice versa)

⇒ Max likelihood = 0.7 × 0.4 × 0.4 × 0.8

⇒ we chose model which has maximum likelihood

This Product is very low so we take -log of that so eq become $= -\log(0.7) - \log(0.4) - \log(0.4) - \log(0.8)$. This eqn is corss entropy should be minimum

$y = mx + c$ (slop, intercept)

$y = \beta_0 + \beta_1 x$

$y = \omega^T x + b$

binary classification

+ve Points = +1

-ve Points = -1

Plane

distance from point to Plane

$$d = \frac{\omega^T x + b}{\|\omega\|}$$

If Point above the plane distance +ve otherwise -ve

Total sum of dis = $\sum_{i=1}^{n} \omega_i^T x_i$

y_i is actual value (target) of point then

If $y_i d > 0$ then our prediction is correct

$y_i \sum_{i=1}^{n} \omega_i^T x_i > 0$

$y_i \frac{\omega^T x + b}{\|\omega\|} > 0$

$y_i (\omega^T x + b) > 0$

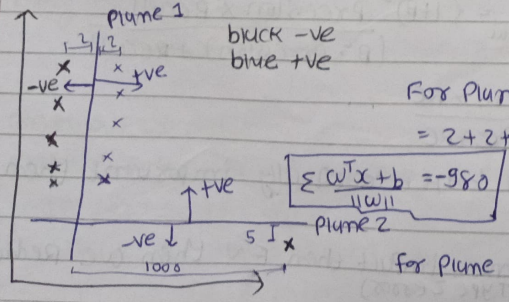
$y = \omega^T x + b = [m][x_1] + b$

$y = \omega_1^T x = \begin{bmatrix} m \\ b \end{bmatrix} [x_1 \; 1]$ ← If add b in matrix

$mx_1 + b$ Here we add bin w f i in x

Cost function is $\max \sum_{i=1}^{n} y_i (\omega_i^T x_i + b)$

for maximize $\sum y_i(\omega_i^T x_i + b)$ we need to update only ω_i (y_i given, x_i given) until get max of $\sum_{i=1}^{n} y_i(\omega_i^T x_i + b)$



For Plane 1 Sum of distance is

$= 2+2+2+2+2+2+2+2+2+2 - 1000$

$\sum \frac{\omega^T x + b}{\|\omega\|} = -980$

for Plane 2 sum of dis = 25

⇒ Impact of σ $\omega_n = \omega_0 + \eta(y_i - \hat{y}_i) x_i$

⇒ instead of using step fun If use σ then $\hat{y}_i - \hat{y}$ will be not 0 and weight update every time either Point is correctly classified or mis classified

⇒ positive Points (⊕) move line ↙ always

-ve point (⊖) move ↗ always

So here because of outlier we select Plane 2 but Plane 1 is best Plane so overcome with this here we take σ function to prevent this

$$\max \sum_{i=1}^{n} \sigma(y_i + \omega^T x_i)$$

↑ z

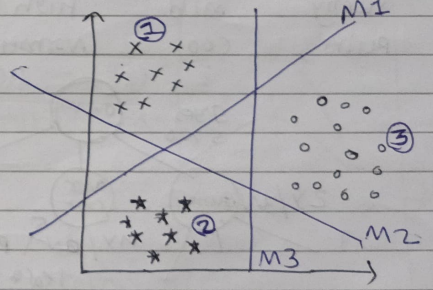
here $\sigma = \frac{1}{1+e^{-z}}$, $0 < \sigma < 1$ at $z=0$ $\sigma = 1/2$

(∵ we take -log here so) Cross entropy eqn $-y_i \log(\hat{y}_i) - (1-y_i)\log(1-\hat{y}_i)$ ← If • and $\hat{y}$ is Prob of ⊗

In pre calculation for Plane 1 sum of distance is -980 This is big value so we transform this in to between 0 to 1 by using σ function. by doing this we neglect large value due to outlier

* Logistic Regression (one vs Rest)

f_1	f_2	f_3	O/P	O_1	O_2	O_3
I_1	I_2	I_3	O_1	+1	-1	-1
I_4	I_5	I_6	O_2	-1	+	-1
I_7	I_8	I_9	O_3	-1	-1	+1
I_{10}	I_{11}	I_{12}	O_1	+1	-1	-1
I_{13}	I_{14}	I_{15}	O_2	-1	+	-1
				↑	↑	↑
				M_1	M_2	M_3



m1 will be able to find out that O1 is +1 or not +1 for O1, m2 for O2, m3 +1 for O3

whenever we give new Input feature we Pass all Input in M1 model then Pass all feature in M2 & M3 Respectively let m1 give Probability 0.2

m2 " " 0.25

m3 " " 0.55 (Probability of $m_1 + m_2 + m_3 = 1$)

Probability of m3 highest so data belongs to O3 category

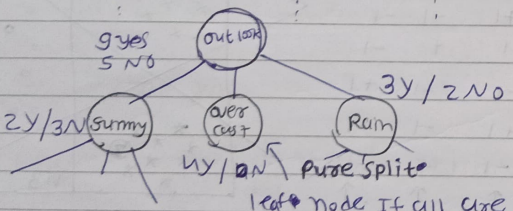
~ continue in DL 1 books

Decision Tree

⇒ use for Classification and Regression both
⇒ There are 2 technique in Decision Tree
ID3 ← → CART

a) Entropy and gini index ← Purity split
b) Information gain ← feature Decision split

Outlook	Tem	Humidity	Wind	Play
Sunny	HOT	High	False	No
Sunny	HOT	High	True	No
Overcast	HOT	High	Weak	yes
Rain	mild	High	W	yes
Rain	Cool	Normal	F	yes
Rain	Cool	Normal	T	no
Overcast	Cool	Normal	T	yes
Sunny	Mild	High	F	no
Rain	Cool	Normal	F	yes
⋮	⋮	⋮	⋮	⋮



leaf node If all are Y or No
⇒ If there is no Pure split we take another feature and split again untill leaf node

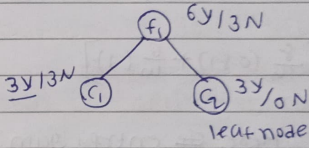
⇒ For Purity check ⇄ entropy
gini index
⇒ for feature selection → Information gain

Purity Check 1) Entropy

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

+ for yes
− for No

It can be more then 2 OP category

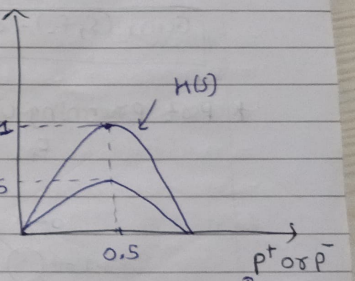


For C_1 $H(S) = -3/3 \log_2 3/3 - 0/3 \log_2 0/3$

$H(S) = 0$ ⇒ Pure split

For C_1 $H(C_1) = -3/6 \log(3/6) - 3/6 \log_2(3/6)$

$H(C_1) = 1$ ⇒ Impure split



2) gini Impurity

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

$$Gini = \sum_{i=1}^{n} (P^*)^2$$

Probability of + or −

GI = 1 − Gini

GI for Impure Split = $1 - ((P_+)^2 + (P_-)^2) = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right)$

GI = 0.5 for Impure

GI = 0 for Pure

⇒ we can use both Entropy and gini Impurity but If we has big dataset then for easy calculation GI is best.
(∵ Entropy contain log)

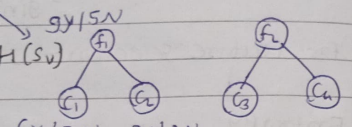
for feature selection Information gain

Samples in Child node | Entropy of child node

$$Gain(S, f_1) = H(S) - \sum_{level} \frac{|S_v|}{|S|} H(S_v)$$

↑ S of f_1 ↑ ↑ Total samples

Information gain for f_1 root node | Entropy of f_1 root node



$H(S) = H(f_1) = 0.94$

$H(C_1) = 0.81$

$H(C_2) = 1$

Gain = Parent's entropy − weighted entropy of child

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

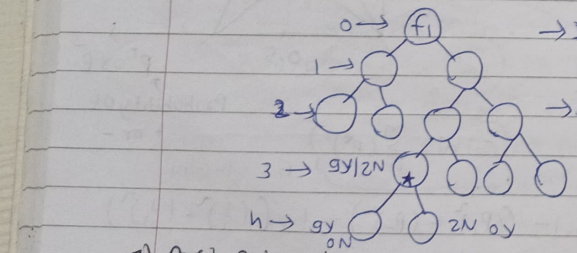
$Gain(S, f_1) = 0.049$

Here entropy gain in f_2 is greater so we select f_2

$Gain(S, f_2) = 0.052$

* Post Pruning and Pre-Pruning Decision trees

f_1	f_2	f_3	O/P



→ If we split in depth then we got over fitting

→ So to overcome with this we use some methods

1) Post Pruning
2) Pre Pruning

1) Post Pruning :

In Post Pruning we make decision tree and then Pruning(cut) it. using Parameters.

⇒ There are many parameters let we take max-depth here for * node we see that Probability of y is high so there is no need to further split. by doing this we can overcome with overfitting

⇒ max depth set as 3 level so there is no splitting after 3 level and there is no overfitting

2) Pre Pruning : we are applying hyper parameters tuning while making decision tree

⇒ set max-depth in some range of values [. . . .]

and define max-features [. . . . .]

⇒ now decision tree select and after applying all parameters find where we can get max accuracy

⇒ Post Pruning ← Small dataset

Pre pruning ← Bigger dataset

Disadvantage: time complexity but in test time it's low (only)

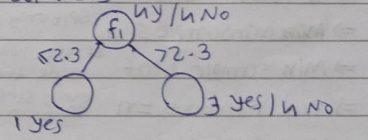
★ Decision Tree split for numerical

	f_1	O/P
x_i →	2.3	Yes
	3.6	Yes
	4	No
	5.2	No
	6.7	Yes
	8.9	No
	10.5	Yes
	14.2	No

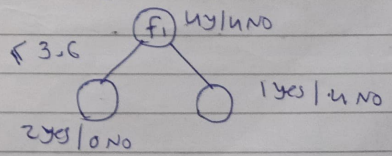
1) Sorting all values in ascending order

2) Threshold value = 2.3

$x_i \leq 2.3$



3) next threshold value is 3.6 and again split will happens

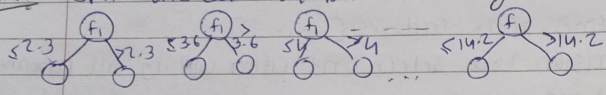


4) now for both tree we calculate Entropy and information gain and we select tree who has better entropy and information gain

5) This kind of division will happen for each x_i value (every value of f_1)

Step 1 Sort the data on basis of numerical column

Step 2 split the data on basis of every numerical values



Step 3 calculate entropy or information gain of each tree and select best tree on basis of entropy and IG.

~~Do~~

Step 4 Do it recursively ~~for~~ left and right nodes and finally we get decision tree.

⇒ Splitter = best but in random splits are random (info gain) ~~random~~ For best in numerical column we choose those split which has high ~~IG~~

⇒ ~~Max~~ MinSample split Split only when there is min n number of sample (rows)

⇒ Min sample Leaf If any node has min n sample ~~in~~ in their every child node then only split

⇒ Max Features = n every step choose n random features and split on it not split on every features

⇒ Max Leaf Nodes

⇒ Min Impurity Decrease (gini index or gini impurity) parents entropy − ~~entropy~~ weighted child is > min impurity decrease then only split

receiver operating characteristic curve — Area under curve

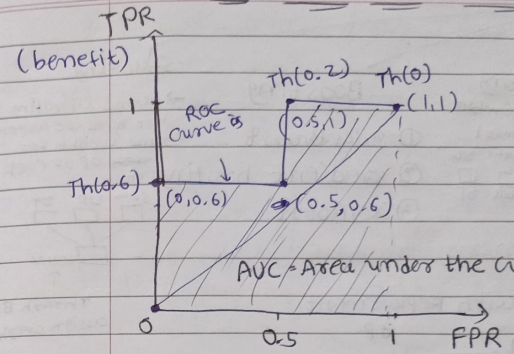
ROC And AUC Curve

Actual \ Pred	0	1
0	TN	FP
1	FN	TP

$TPR = \frac{TP}{TP+FN}$ (Pred +ve / Total +ve actual)

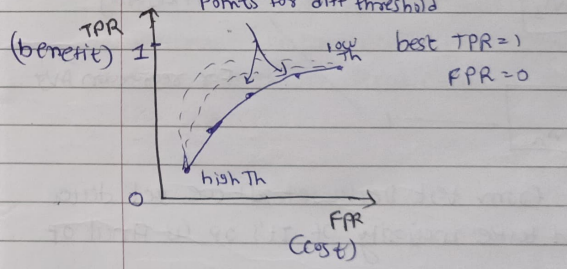
$FPR = \frac{FP}{FP+TN}$ (Pred +ve / Total −ve actual)

Cost ↑



⇒ on base of what we actually need (High TPR or low or high FPR or low) we can select threshold value for given problem

⇒ good model has always greater then 0.5 this area. If area of model less then ~~or~~ that then our model is dump model It guess randomly



Ensemble Techniques

Combining multiple models

BAGGING (Bootstrap aggregation) — same but give diff data while training
Ex random forest

Voting ensemble → Use model are diff algo
Ex: SVM LOR DT

Boosting
① Ada Boost
② gradient boosting model
③ XgBoost

Stacking → same as voting but here we assign some weights for each OP of each

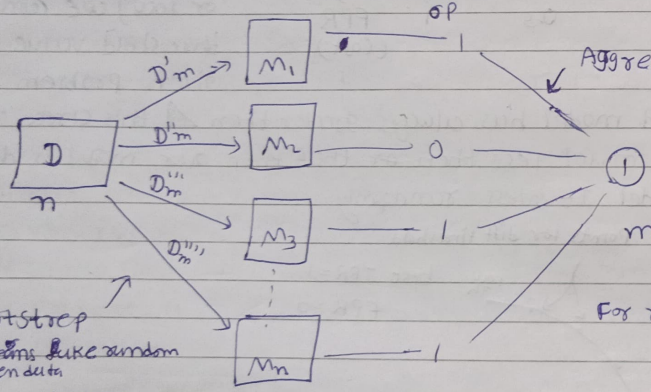
SVM LOR DT → KNN ↑ Through BP assign weights

Random forest

* Bagging

Raw sample with Replacement

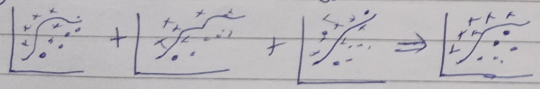
m<n



Bootstraping means take random sample from given data

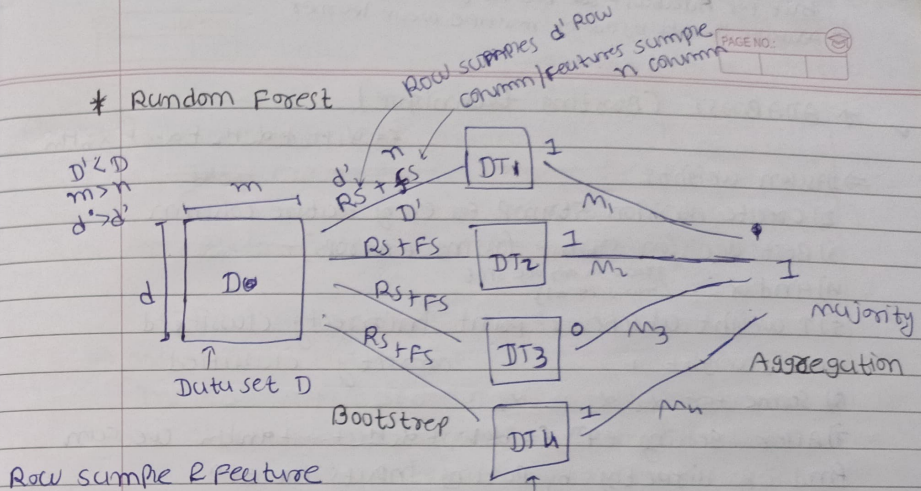
⇒ After training is done from test Data set If we get data to those models and take majority of it's OP as final OP

⇒ Why Ensemble Techniques work



⇒ voting & boosting can have diff weights but Bagging has same

* Random Forest



Decision Trees has low bias and high variance

⇒ Here we combine Decision trees so high variance convert into low variance

⇒ In regression Problem we take O.P as Avg or mean or median of all O.Ps

⇒ In case of Bagging we take equal importance of each vote but for Adaboost we not
⇒ Adaboost is made by adding multiple weak learner

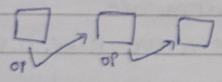
PAGE NO.:

✱ ADABOOST (Boosting technique)

$f = \alpha_1 H_1 + \alpha_2 H_2 + \alpha_3 + \alpha_n H_n$

⇒ 1) Assign weights
2) Create decision stump for every feature column
3) Best decision stump (using entropy) ↑ Tree with only one split (max depth = 1)
4) Find α
5) ↑ weight whichever point incorrectly classified and ↓ weight " " correctly classified
6) Same for $\alpha_2, \alpha_3 \dots \alpha_n$
7) After getting eqn $f = \alpha_1 H_1 + \alpha_2 H_2 + \dots + \alpha_n H_n$ we can find OP directly by putting inputs in eqn

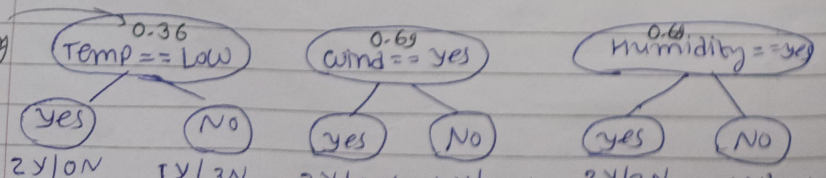
1) weak learner (accuracy is just over 50%)
2) weight
3) Dependency (give 1st op into 2nd, 2nd op to 3rd ....)



Temp	Wind	Humidity	Play	sample weight	Updated weight	Normalized weight	range (+ve)
High	No	yes	N	1/6	0.07	0.09	[0 - 0.09]
low	yes	no	y	1/6	0.07	0.09	[0.09 - 0.18]
High	yes	yes	N	1/6	0.07	0.09	[0.18 - 0.27]
High	No	yes	y	1/6	0.37	0.54	[0.27 - 0.81]
Low	yes	yes	y	1/6	0.07	0.09	[0.81 - 90]
High	yes	No	N	1/6	0.07	0.09	[0.90 - 1]

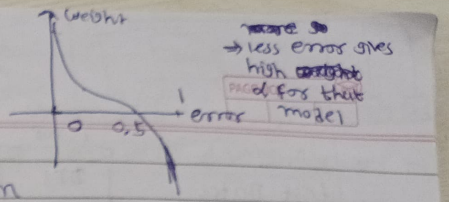
(wrongly classified)

Entropy (select which has less entropy)



⇒ Now we select column which has low entropy as a weak learner and it's called base learner

$f = \alpha_1 H_1 + \alpha_2 H_2 + \alpha_3 H_3$



⇒ So we select Temp column
Total error = 1/6 (∵ node no has 1 yes and weight of that yes 1/6)

Performance of stump = $\frac{1}{2} \log_e \left(\frac{1 - TE}{TE} \right)$

$\alpha_1 = \frac{1}{2} \log_e (5) = 0.8$

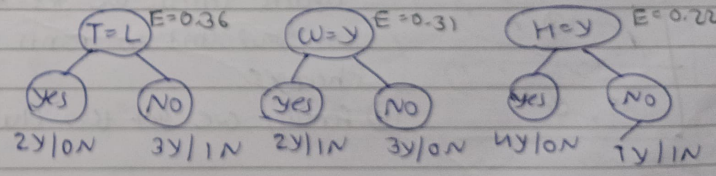
⇒ now update the weight
Increase weights whose guess is incorrect and decreases all others weight.

increase weight $\times e^{\text{perf of stump}} = \frac{1}{6} \times e^{0.8} = 0.37$

decrease = $\frac{1}{6} \times e^{-0.8} = 0.07$

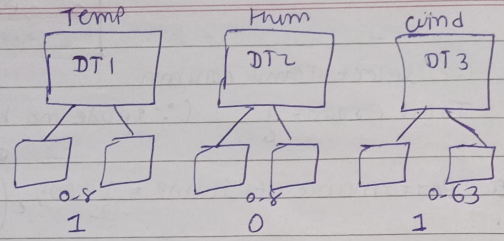
⇒ select random datas from original 1st data set (using probability)

T	W	H	P	sample weight (Parent dataset)
H	y	N	N	1/6
H	N	y	y	1/6
H	N	y	y	1/6
L	y	N	y	1/6
H	N	y	y	1/6
L	y	y	y	1/6



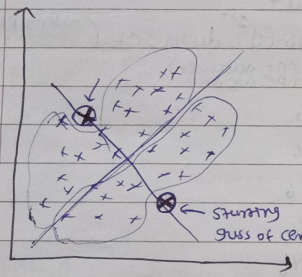
⇒ now we select Humidity column (∵ low entropy)
Total error = 1/6
$\alpha_2 = 0.8$
⇒ Update weights and same for this also

⇒ let assume we final get $\alpha_3 = 0.63$ so we get our final model $F = 0.8 H_1^{\text{Temp}} + 0.8 H_2^{\text{Humi}} + 0.63 H_3^{\text{Wind}}$



★ K means Clustering (unsupervised Technique)

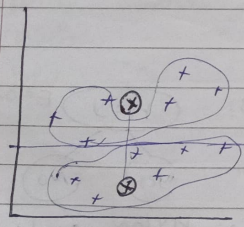
⇒ K means Clustering finds the simmilarity between the points and group them into cluster



① K values ⇒ Centroids K=2
(If K=2 then 2 clusters)

② initialize two centroid randomly

③ select group & avg and update centroid to new avg value



④ repeat ~~step~~ step ③ again and again untill we get fix number of group and no points movement changes

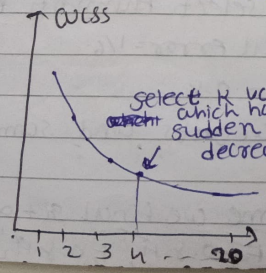
⑤ finally we get two cluster or group

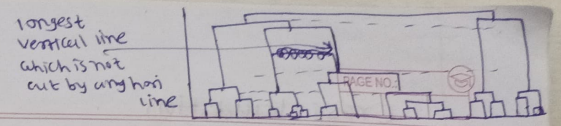
Elbow method

K = 1 to 20

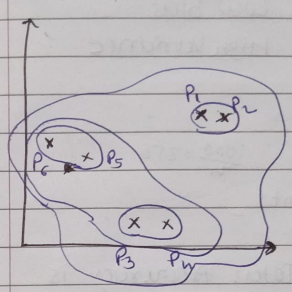
⇒ wcss is square sum of each points to its nearest centroid

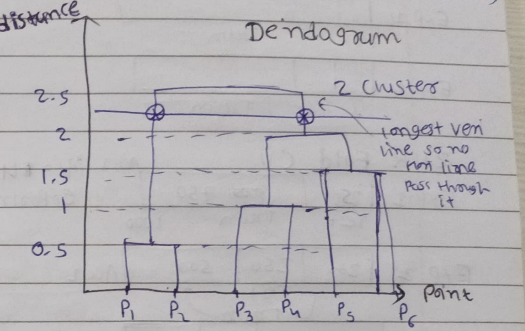
wcss = within cluster sum of square





★ Hierarchical Clustering intuition (unsupervised ml)





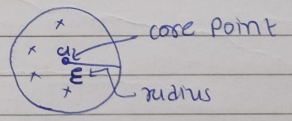
⇒ This method use to know how many number of cluster should be use in order to classify points

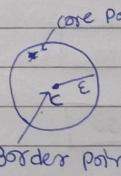
★ DBSCAN (Density-Based Spatial Clustering of Applications with Noise) Property

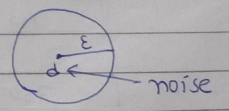
Core Point : atleast MinPts should be in ε range of core point

Border point : does not satisfy min point condition but atleast one core point present in range of Border point

Noise Point : no point present







Advantages : Is great at separating ~~high~~ clusters of high density versus low density cluster. & great with handling outliers

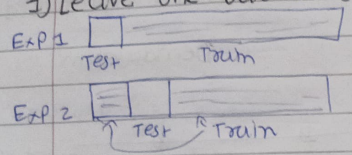
Disadvantage : does not work well when clusters of varying density

⇒ not work well with clusters of similar density

⇒ struggles with high dimention data

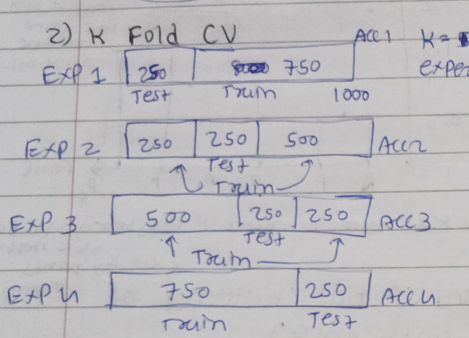
Cross Validation

1) Leave one out cross validation (LOOCV)



Low bias
High variance

2) K Fold CV



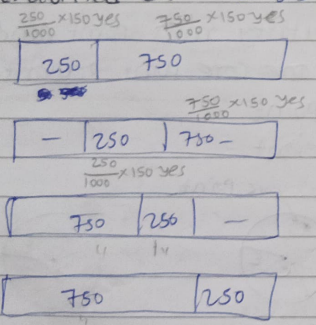
K = 4 experiments

$\frac{1000}{4} = 250$

Total Accuracy is mean of all accuracy

3) Stratified CV

150 yes 850 No



⇒ every time the test data and train data contain same ratio of numbers of instance

4) Time series CV

	I/P	O/P
Day 1		
Day 2	Day 1 D2 D3 D4 D5	D6
Day 3		
Day 4	D1 D2 D3 D4 D5 D6	D7
Day 6		
Day 7	D1 D2 D3 D4 D5 D6 D7	D8

let we need Day 6-7 OP

⋮ ⋮

Baye's Theorem

$P(A/B) = \frac{P(A \cap B)}{P(B)}$ Probability of A when B is given

$P(A/B) = \frac{P(B/A)\ P(A)}{P(B)}$

* Naive Baye's classifier

$$P(y/x_1, x_2, \dots x_n) = \frac{P(x_1/y)\ P(x_2/y) \cdots P(x_n/y)\ P(y)}{P(x_1)\ P(x_2) \cdots P(x_n)}$$

$$= \frac{P(y) \prod_{i=1}^{n} P(x_i/y)}{P \prod_{i=1}^{n} P(x_i)}$$

$$P(y/x_1, x_2, \dots x_n) \propto P(y) \prod_{i=1}^{n} P(x_i/y)$$

Example

Outlook	Yes	No	P(y)	P(N)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temp	Yes	No	P(y)	P(N)
HOT	2	2	2/9	2/5
MILD	4	2	4/9	2/5
COOL	3	1	3/9	1/5
Total	9	5	100%	100%

	P(y)	P(N)
Yes	9	9/14
No	5	5/14
Total	14	100%

P(y) & P(N)

Assume Today(sunny, HOT) ret find Prob

$P(yes/today) = \frac{P(sunny/yes)\ P(HOT/yes)\ P(yes)}{P(Today)}$ ← const

$P(yes/Today) = 2/9 \cdot 2/9 \cdot 9/14$

$= 0.031$

$P(No/Today) = 3/5 \cdot 2/5 \cdot 5/14 = 0.08571$

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

$P(No/Today) = 1 - 0.27$

$= 0.73$

So, Today person will Not play

How to apply Naive Baye's classifier on Text data (NLP)

f_1	f_2	f_3	f_4	O/P
The	food	Delicious	Bad	
1	1	1	0	1
1	1	0	1	0
0	1	0	1	0
0	1	1	0	1
0	0	0	1	0

In sentence many words $[x_1, x_2, x_3 \dots x_n]$

$P(y=yes/sentence) = P(y=yes/x_1, x_2 \dots x_n)$

For test sentence

$\propto P(y) \prod_{i=1}^{n=3} P(x_i/y=yes) \cdot P$

Bad is not included

$\propto 2/5 \times (1/2)\ (2/4)\ (2/4)$

$P(y)$ $P(x_1/y=yes)$ $P(x_2/y=yes)$ $P(x_3/y=yes)$

$= 1/10$

$= 0.1$

$P(y=No/x_1, x_2, x_4)$

→ For Numerical data

Height	weight	Gender
172	150	M
180	180	M
165	165	M
139	110	F
145	150	F
160	130	F

Find Prob of being male given {H=185, W=170}

we have to Find $P(M/H=185, W=170)$ & $P(F/H=185, W=170)$ and which ever is large (∵ we ignore denominator term so need to calculate both Prob)

$P(F/H=185, W=170) = P(H=185/M)\ P(W=170/M)\ P(M)$ (P(M) = 1/2)

or Binomial or Multinomial or Poisson

assume normal distribution so we find μ, σ of all males H & W and by Putting X = H or W of given que. and find Prob $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

★ SVM Kernels

For degree d polynomials the polynomial kernal is

$$K(x,y) = (x^T y + c)^d$$

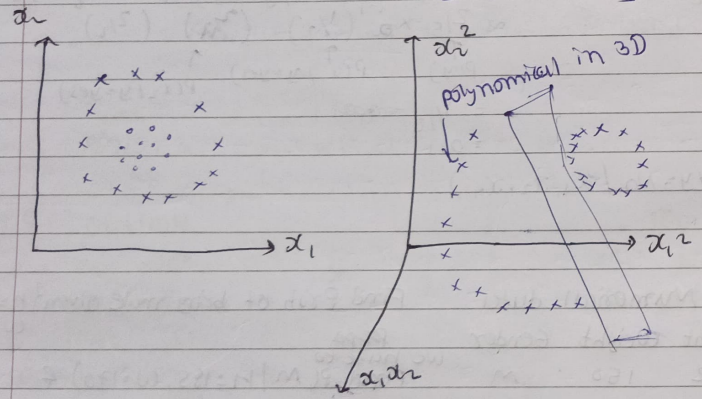
When c=0 kernal is called homogeneous

Features		o/p
x_1	x_2	y

$$f(x_1, x_2) = (X^T X + c)^d$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot [x_1 \; x_2]$$

$$= \begin{bmatrix} x_1^2 & x_1 x_2 \\ x_1 x_2 & x_2^2 \end{bmatrix}$$



★ GRadient Boosting

base model + R_2

pred of DT 1 =

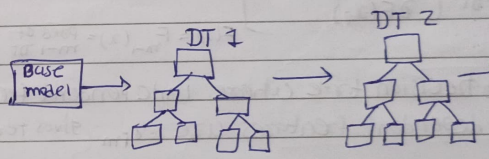
For Base model residual

For 2nd DT residual

Degree	exp	salary	base model gives $\hat{y}$	R_1	R_2	y_{DT1}	y_{DT2}
BE	2	50K	75	-25	-23		52
Masters	3	70K	75	-5	-3		72
masters	5	80K	75	5	3		77
PHD	6	100K	75	25	20		95

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

Base model give O/P 75 for all inputs



Step 1) Base model

2) Compute Residual, Error, Pseudo Residual

3) Construct DT

input Degree, exp & OP R_1 (not salary)

$$F(x) = h_0(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots + \alpha_n h_n(x)$$

O/P from ⇒ Base model + DT 1 + DT 2

hyper parameter 1, hyper parameter 2

Algoritham Input: $\{x_i, y_i\}$ (1), $L(y, F(x))$ (2) Loss fun, No of Trees (3)

Step 1) Initialize model with constant value

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

Loss function $= \frac{1}{2}(y - \hat{y})^2$

we have to find $\gamma = \hat{y}$ so that loss is min

$F(x) = f_0(x) + f_1(x) + f_2(x) + \dots$ (1) (2)

Step 2) Iterate $M = 1$ to M:

i) Compute so-called Pseudo-residuals

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right] \quad \text{for } i = 1, \dots, n$$

i^{th} row, m^{th} DT

$F(x) = F_{m-1}(x) =$ Pred of $m-1$ DT

$\text{Loss} = \frac{1}{2}(y_i - \hat{y}_i)^2$

$r_{im} = \frac{-2}{2}(y_i - \hat{y}_i)(-1) = y_i - \hat{y}_i$

$\text{Loss} = \sum \frac{1}{2}(y_i - \gamma)^2$ ii) now, Create Decision tree where independent features are x_i and dependent features are r_{im}, gives terminal regions $R_{jm}, j = 1, 2, 3, \dots, n_j$

have to find γ so loss min

$\frac{dL}{d\gamma} = \sum (y_i - \gamma) = 0$

$\gamma = \frac{\sum y_i}{n}$

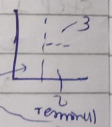
For loss = mse γ = mean

iii) compute multiplier γ_m by solving this eqn.

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x) + \gamma h_m(x_i))$$

Find γ such that γ_m is min

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma)$$

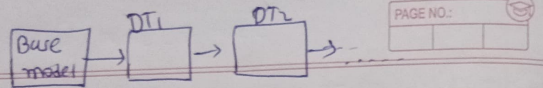


iv) update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

old value; value from DT

Step 3) output $F_m(x)$ $f_0(x) + f_1(x) + \dots + f_{m-1}(x)$



☆ XGBoost

Classification

o/p = 0

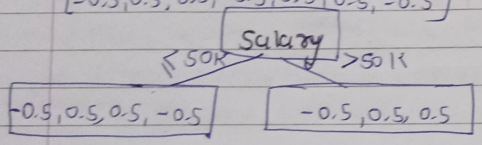
Salary	Credit	Approval	Residual	New Probabi	☆ residual
≤50K	Bad	0	0-Pr = -0.5	0.6	-0.6 (0-0.6)
≤50K	Good	1	1-Pr = 0.5	0.4	0.6 (1-0.4)
≤50K	G	1	0.5	0.3	0.7 (1-0.3)
>50K	B	0	-0.5	0.7	-0.7
>50K	G	1	0.5	0.2	0.8
>50K	Normal	1	0.5	0.1	0.9
≤50K	N	0	-0.5		

o/p is ~~either~~ either 0 or 1 and since this is classification problem probability of 0 or 1 Pr = 0.5

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

Base model

Pr = 0.5



⇒ For xgboost we need to classify into ~~binary~~ two child If there are two then more category in salary we need to classify in to two child only (OR credit)

$$\text{Similarity weight} = \frac{[\Sigma(\text{Residuel})]^2}{\Sigma[Pr(1-Pr)+\lambda]}$$

In this problem let $\lambda = 0$ (hyper parameter)

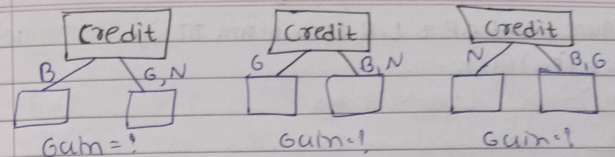
Similarity weight for left side (≤50K) $= \frac{[-0.5+0.5+0.5-0.5]^2}{[0.5(0.5)+0.5(0.5)+0.5(0.5)+0.5(0.5)]} = \frac{0}{1} = 0$

Similarity weight for right side (>50K) $= \frac{(-0.5+0.5+0.5)^2}{[0.5(0.5)+0.5(0.5)+0.5(0.5)]} = \frac{0.25}{0.75} = \frac{1}{3}$

Sm for parent salary $= \frac{0.25}{7(0.25)} = 0.14$

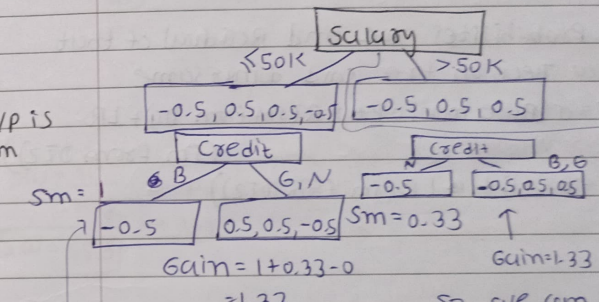
Gain = Sm Left Child + Sm right Child − Sm of parent

Gain = 0 + 0.33 − 0.14 = 0.21



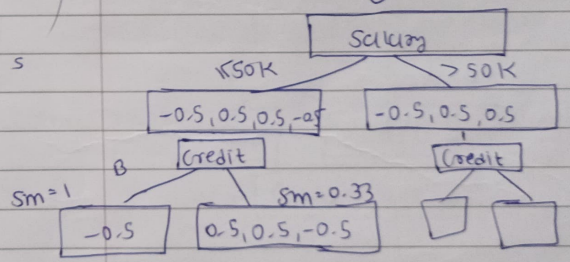
Let assume salary give highest gain among salary and 3 Credit Tree so salary is best way of constructing

now we are doing split w.r.t Credit



o/p is sm

So we can select any of ~~two~~ one



Post Pruning

Cover value = Pr(1−Pr)

In this problem Pr = 0.5

so Cover value = 0.25

If Gain is less then Cover value then we cut that branch

Base model output = log(odds)

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

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

Here we traverse tree according to Input data and find similarity weight of ~~root~~ leaf node

For train data (550, Bad)

Probs $P_r = \sigma(\text{Base model o.p} + L.R\ (\text{sm from DT1}))$ let LR = 0.1

$= \sigma(0 + (0.1)(1))$

$= \sigma(0.1)$

$= \frac{1}{1+e^{-0.1}} = 0.6$

For (550, 6) assume $P_r = 0.4$

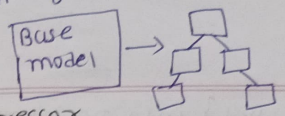
:

After finding all Probabilities we find Residual of that and creating new Decision tree and doing same

now $P_r = \sigma(\text{Base model o.p} + LR(\text{sm from DT1}) + LR(\text{sm from DT2}))$

$P_r = \sigma(0 + 0.1(1) + LR(\text{sm from DT2}))$

o/p = avg = 51K



★ Regressor

Exp	Gap	Salary	Residual 1	o/p =	Residual 2
2	yes	40K	-11 (40-51)	46	-4 (40-46)
2-5	yes	42K	-9 (42-51)	46	-4 (42-46)
3	NO	52K	1 (52-51)	53.5	-1.5 (52-53.5)
4	NO	60K	9	62	-2 (60-62)
4.5	yes	62K	11	63	-1 (62-63)

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

Exp — sw = $\frac{1}{5+1} = 0.166$

≤2 : -11 — $sw = \frac{11^2}{1+1}$ (let $\lambda = 1$) — sw = 65.5

>2 : -9, 9, 11, 1 — $sw = \frac{(-9+1+9+11)^2}{4+1} = 28.8$

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

Gain = 65.5 + 28.8 − 0.16

Gain = 93.84

Exp — sm = 0.166

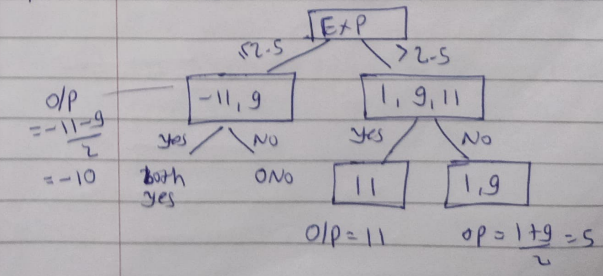
≤2.5 : -11, -9 — sm = 133.33

>2.5 : 1, 9, 11 — sm = 110.25

Gain = 143.42

similarly we find Gain for all and choose max gain split

let assume above tree (≤2.5, >2.5) is best



o/p = $\frac{-11-9}{2}$ = -10

o/p = 11

o/p = $\frac{1+9}{2}$ = 5

Now, we pass the data {2, yes} into Decision Tree and get value of output

$O/P_1 = 51 + (0.5)(-10) = 46$

Base model ↑ LR ↑ OP from DT1

After finding O/P_1 column we find Residual 2 and make another Tree DT2

$O/P_2 = 51 + LR\,DT_1 + LR\,DT_2$

Base model ↑ OP from DT_1 ↑ OP from DT_2

⇒ There is hyper parameter γ has some value node which has Information Gain < γ we Post Prun (cut that sub Tree)

Tensors

★ 0-D Tensor / Scalar

Ex 2

★ 1D Tensor / vector No of Axis = Rank = dimension

Ex [1, 2, 3, 4, 5] ⟶ 1D Tensor

Collection of 0D Tensor / Scalar → 4D Vector

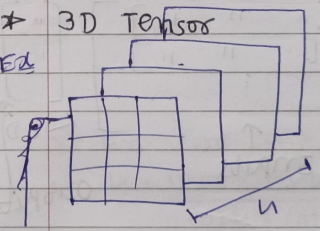
1D array / array

★ 2D Tensor / matrices

Ex $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ Collection of 1D Tensor / vector

★ 3D Tensor

Ex

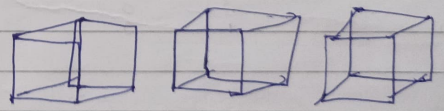


4×3×3

3D

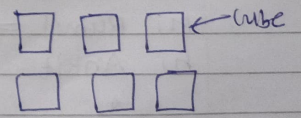
Collection of 2D Tensor

★ 4D Tensor



Collection of 3D Tensor

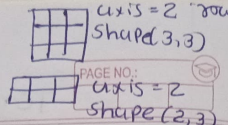
★ 5D Tensor



Collection of 4D Tensor

⇒ In machine learning There are mostly 0-5D Tensor are used

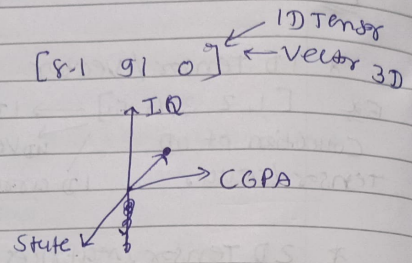
No of axis = Rank = No of dimension



Ex of 1D Tensors

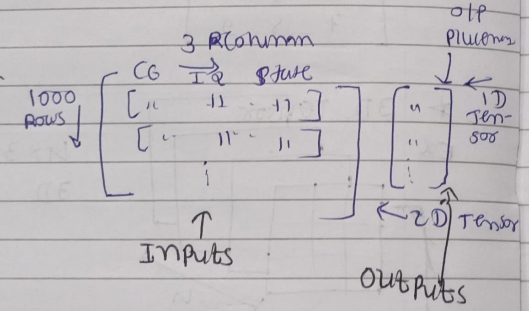
	CGPA	IQ	state	Placement
1000 Rows ↓	8.1	91	WB 0	1
	7.2	102	KR 1	0
	?			

each Row is 1D Tensor



Ex of 2D Tensors

	CG	IQ	State	Placem
1000 Rows ↓				



Ex of 3D Tensors

NLP = natural Language Processing

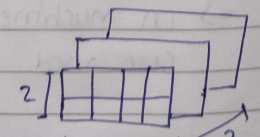
	Hi	Nitish	Rahul	Ankit
Hi Nitesh ⇒	1	0	0	0
	0	1	0	0
Hi Rahul ⇒	1	0	0	0
	0	0	1	0
Hi Ankit				

$$\left[\,[[1,0,0,0],[0,1,0,0]]\;,\;[[1,0,0,0],[0,0,1,0]]\;,\;[[1,0,0,0],[0,0,0,1]]\,\right]^{*}$$

3 × 2 × 4

Time series data

	Highest	lowest
Day 1	—	—
Day 2	—	—
⋮	⋮	⋮
Day 365		



For 1 year [365, 2] ← 2D Tensor

For 10 year we have 10 2D Tensor

Time axis

10 × 365 × 2

3D Tensor

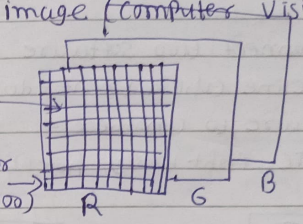
conda.bat deactivate → command for return into Base (root) environment

conda activate envname : enter into {envname} environment

Spider Code run F5

Example of 4D Tensor

image (Computer Vision)



For 1 Image (3, 1200, 800) ← 3D Tensor

for 50 images (50, 3, 1200, 800) ← 4D Tensor

Example of 5D Tensor

Videos = Collections of images

let assume we have video of 60 sec

30 fps

480P (480 × 720 × 3 Channels)

(4, 1800, 480, 720, 3)

4 videos ↑ 1800 = 60×30 images for 1 video ↑ 480, 720 Pixels ↑ 3 Colors

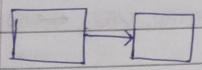
If we store these 4 videos in float 32 then we need

$$\frac{4 \times 1800 \times 480 \times 720 \times 3 \times 32}{1024 \times 1024 \times 1024} = 27\ GB$$

4 Videos take very large space so we use Video Codec formate which allocate memory less than float 32 and compress it.

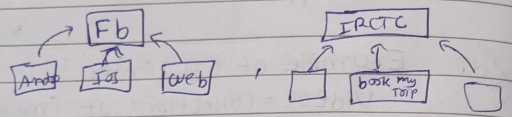
Ex MKV, mp4

★ API (Application Programming interface)

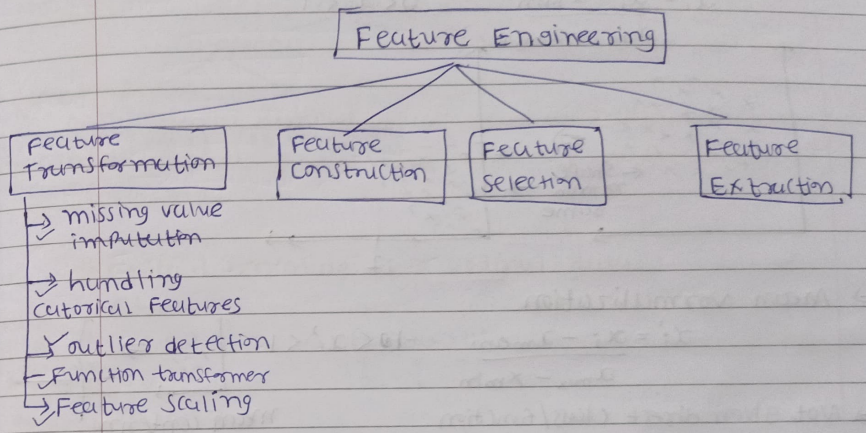


⇒ API is connect two software
⇒ API is pipeline which transfer data from one software to another
⇒ use: IRCTC, flight booking, hotel booking...

Ex:



★ Feature Engineering

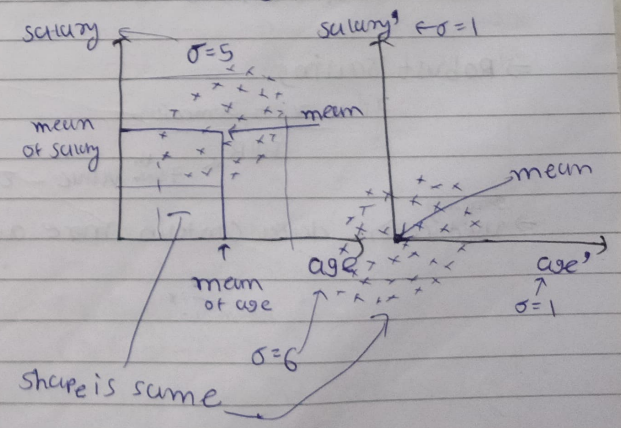


* Feature scaling - Standardization

⇒ ~~Standardization~~

Age(x_i)
27 ← x_1
15 ← x_2
33 ← x_3
63
90
:
500 values

$$x_i' = \frac{x_i - \bar{x}}{\sigma}$$ ← mean, ← SD



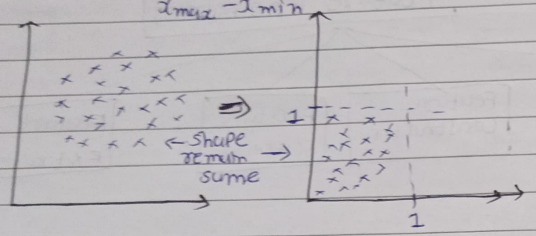
when to use
⇒ K means
⇒ K nearest neighbours
⇒ PCA
⇒ Artificial neural network
⇒ Gradient decent

PAGE NO.:

* Features Scaling - Normalization

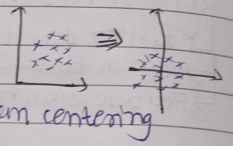
⇒ Min-Max Scaling Normalization

$$x_i' = \frac{x_i - x_{min}}{x_{max} - x_{min}} \qquad 0 < x_i' < 1$$



⇒ Mean Normalization

$$x_i' = \frac{x_i - x_{mean}}{x_{max} - x_{min}} \qquad -1 < x_i' < 1$$



⇒ Not given direct class/function in SKlearn

⇒ used when we need centered data

⇒ Max Abs Scaling

$$x_i' = \frac{x_i}{|x_{max}|}$$

SKlearn : MaxAbs Scaler

⇒ used when we have sparse data (data which contain more zeros)

⇒ Robust Scaling

$$x_i' = \frac{x_i - x_{median}}{IQR}$$

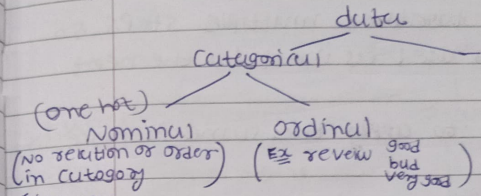
IQR ← 75th % value − 25th % value

⇒ generally used when data contain more outliers

Label encoding: work same as ordinal encoder but use when O/P Label(y) are given as category

PAGE NO.:

* Encoding Categorical variables



⇒ ordinal Encoding (For ordinal data)

Education		PG > UG > HS
High school	HS → 0	↑ ↑ ↑
under graduate	UG → 1	2 1 0
Post gra..	PG → 2	
	PG → 2	
	UG → 1	
	HS → 0	

⇒ one HOT Encoding (For nominal data)

Color	Target	Color_Y	Color_B	Color_R	Target
Y	0	1	0	0	0
RY	1	1	0	0	1
B	1	0	1	0	1
Y	1	1	0	0	1
R	1	0	0	1	1
Y	0	1	0	0	0

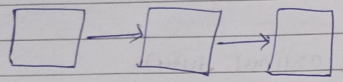
[1, 0, 0] → yellow
[0, 1, 0] → blue
[0, 0, 1] → Red

Multicolinearity : There should be not relation between Independent variables.
→ so we remove one of column

★ Pipelines

⇒ Pipelines which chains together multiple steps so that O/P of each step is used as input to the next step.

⇒ pipeline makes it easy to apply the same preprocessing to train and test



⇒ Transformation is used for linear Regression, Logistic regression

⇒ Decision Tree have no affect on Trans?.

★ Function Transformer

⇒ using transformer we can convert our data into normal distribution

SKlearn

- Function Transformer
  - log trans
  - Recipro
  - sq / sqrt
  - Custom
- Power Transformer
  - Box-Cox
  - yeo Johnson
- Quantile Transfor..

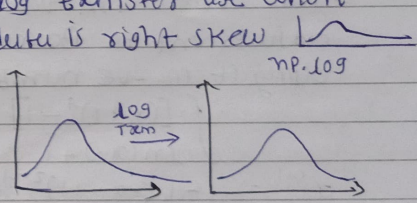
How to find data is normal or not?

use: sns.distPlot OR Pd.Skew() OR QQ Plot

log transformers

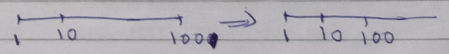
Age	
21	$\log_{10} 21$
35	$\log_{10} 35$
80	

⇒ log transfer use when data is right skew



⇒ when data contains zero values then we use np.log1p which add 1 to data and then take log.

$\log(1+x)$



Reciprocal Transforms

$\frac{1}{x}$ ⇒ convert small values to big and big to small

sq Transforms x^2 use for ⇒ left skewed

sqrt $\sqrt{x}$

★ Power Transformer

This is general transform for all like $\frac{1}{x}, x^2, x^3, \sqrt{x}$

Box cox Transform

$$x_i^{(\lambda)} = \begin{cases} x_i^{\lambda} - 1, & \text{if } \lambda \neq 0, \\ \ln(x_i), & \text{if } \lambda = 0 \end{cases}$$

⇒ The exponent here is variable called λ that cover the rung of -5 to 5 and in this process of searching, we examine all value of λ, finally we choose optimal value (Resulting in the best appro to a normal distri) for your variable

⇒ Applicable only ~~for~~ for $x_i > 0, \forall i$

Yeo-Johnson Transform

⇒ This Transformation is ~~somehow~~ somewhat of an adjustment to Boxcox transformation, by which we can apply it to -ve numbers.

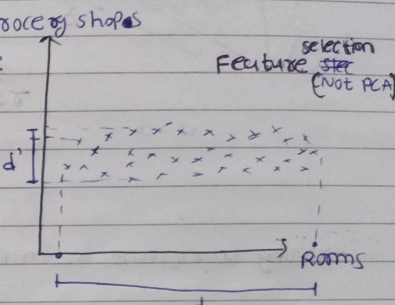
$$x_i^{(\lambda)} = \begin{cases} [(x_i+1)^{\lambda} - 1] / \lambda & \text{if } \lambda \neq 0, x_i \geq 0 \\ \ln(x_i) + 1 & \text{if } \lambda = 0, x_i \geq 0 \\ -[(-x_i+1)^{2-\lambda} - 1] / (2-\lambda) & \text{if } \lambda \neq 2, x_i < 0 \\ -\ln(-x_i+1) & \text{if } \lambda = 2, x_i < 0 \end{cases}$$

★ Principle component Analysis (PCA) Features extraction algorithm

⇒ PCA used to convert higher dimentional data into lower dimentions. Benefits: faster execution of algorithams & easy to visualize

⇒ Geometric intuition:

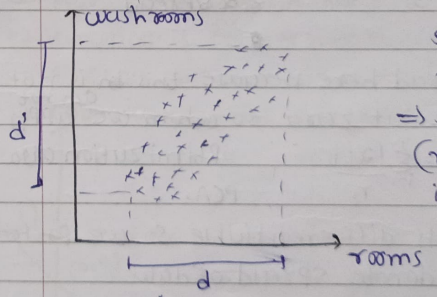
No of Rooms	No of grocery shop	Price
3	2	60
4	6	130
5	6	180
2	10	90



⇒ d > d' so we select axis which has more variance

Here Rooms has more variance

No of rooms	No of washrooms	Price



$d \approx d'$

⇒ Here rooms and washrooms are equally important feature so here feature selection is not important

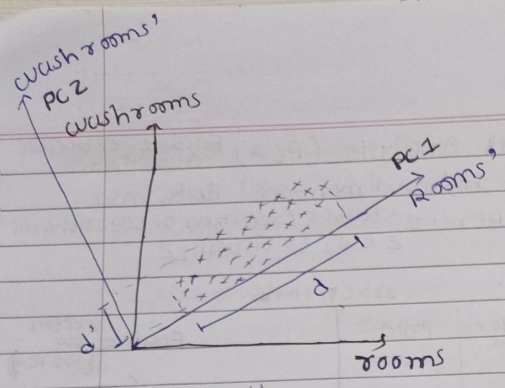
⇒ So we can not decide (mathamatical also) which is important Feature

⇒ In this kind of case Feature extraction is used

⇒ since both features are equally important we create new feature called size which tells us about rooms and washrooms.

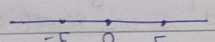
⇒ So, we use size instead of rooms & wrooms. this is what PCA does

⇒ PCA makes new set of features from existing features and select subset from new features which is more important



⇒ Here we choose PC 1 because It has more variance

⇒ why variance is important



mean = $\frac{-5+0+5}{3}$

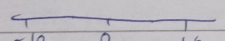
mean = 0

$\sigma^2 = 50/3$

$\sigma = \sqrt{50/3}$

variance $\sigma^2 = \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}$

variance = $\frac{25+0+25}{3}$



mean = 0

$\sigma^2 = 200/3$

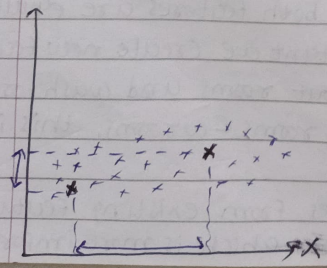
$\sigma = 2\sqrt{50/3}$

σ^2 is not spread but $\sigma^2 \propto$ Spread

⇒ we are not taking spread here because this is not differentiable at zero so we can not apply optimization also in PCA.

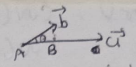
mean abs deviation (MAD) = $\sum_{i=1}^{n} \frac{|x_i|}{n}$

⇒ while square function is differentiable so we prefer σ^2 instead of MAD. to denote spread of data

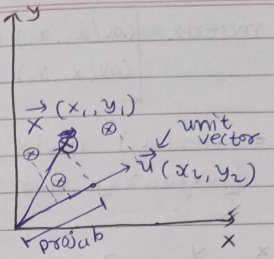


Ex of Photographer

$proj_u b = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|}$



$AB = proj_a b$

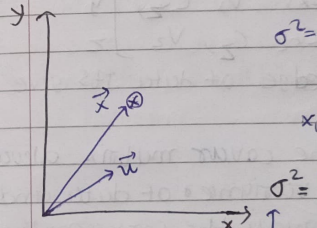


$proj_u b = \frac{\vec{u} \cdot \vec{x}}{|\vec{u}|} = \vec{u} \cdot \vec{x} = u^T x$

$= [x_1 \; y_1] \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$

$= x_1 x_2 + y_1 y_2$ ← scaler

Here we choose $\vec{u}$ unit vector such that variance σ^2 is max



$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i' - \bar{x}')^2$

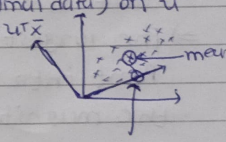
i^{th} Point which Project on $\vec{u}$

Projection of mean (from original data) on $\vec{u}$

$x_i' = u^T x_i$

$\bar{x}' = u^T \bar{x}$

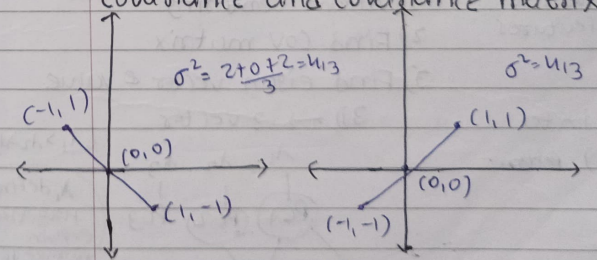
$\sigma^2 = \sum_{i=1}^{n} \frac{(u^T x_i - u^T \bar{x})^2}{n}$



⇒ PCA select $\vec{u}$ such that σ^2 is maximum

⇒ Variance is only measured in 1D. while coveriance give us 2D relations. (relation between x & y)

covariance and covariance matrix



Covar = $\frac{(1)(-1) + (0)(0) + (1)(-1)}{3}$

Covar = 2/3

$= -2/3$

⇒ covar is statistical quntity which represented how two data are change or behave on each other not give intense because cov change if scale of data change so we we correlation

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

pearson corelation = $\frac{cov(x, y)}{\sigma_x \cdot \sigma_y}$ ← SD = -1 to 1

apply this linear transformation of vector V

$$\begin{bmatrix} \quad \end{bmatrix} \cdot V = \lambda \cdot V$$

λ ↑ eigen value

x_1	x_2

$\text{cov}(x_1, x_1) = \text{var}(x_1)$

$\text{cov}(x_2, x_1) = \text{cov}(x_1, x_2)$

$$\text{cov matrix} = \begin{bmatrix} \text{cov}(x_1,x_1) & \text{cov}(x_2,x_2) \\ \text{cov}(x_1,x_2) & \text{cov}(x_2,x_2) \end{bmatrix}$$

$$= \begin{bmatrix} \text{var}(x_1) & \text{cov}(x_1,x_2) \\ \text{cov}(x_1,x_2) & \text{var}(x_2) \end{bmatrix}$$

For 3 column

$x_1\ x_2\ x_3$ — x y z

$$\text{cov matrix} = \begin{bmatrix} V_x & C_{xy} & C_{xz} \\ C_{xy} & V_y & C_{zy} \\ C_{xz} & C_{zy} & V_z \end{bmatrix} \begin{matrix} x \\ y \\ z \end{matrix}$$

⇒ CoV matrix give complet knowledge of data Its give spread and orientation.

⇒ The largest eigen vector of the covar matrix always point into direction of largest variance of data and the magnitude of this vector equals the corresponding eigen value.

⇒ The second largest eigen vector is always orthogonal to the largest eigen vector and points into direction of the second largest spread of the data

⇒ Step by step solution

f_1	f_2	f_3	target

3 Column of features

1) mean centering
2) Find CoV matrix
3) Find eigen vector & value

3D → 3 vector

$\lambda_1, \lambda_2, \lambda_3$ $(\lambda_1 > \lambda_2 > \lambda_3)$

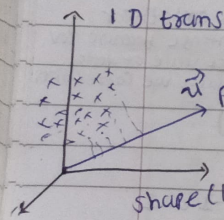
PC1 PC2 PC3

λ_1 define how that λ alone explain variance of the Data

CoV of PC1 > PC2 > PC3

4) trans form points into 1D or 2D (here)

1D transformation



$\vec{u}$ PC1

$u^T \cdot x_i, \forall i$

shape (1,3) (1000,3)

PC1	target

dot Product ↓ result

$(1000,3) \cdot (3,1) \rightarrow (1000,1)$

⇒ eigen value define how that particular one principle component define or explain variance of whole data

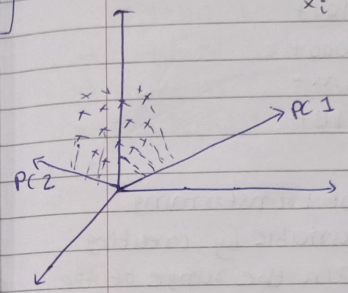
⇒ λ is also called explain variance

⇒ $\frac{\lambda_i}{\sum \lambda_i} \times 100$ How many percent

2D transformation

$(1000,3) \cdot (3,2) \rightarrow (1000,2)$

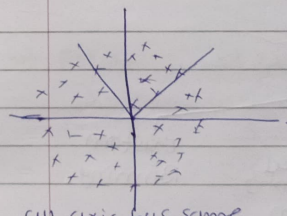
x_i u



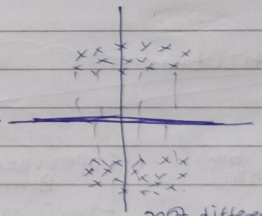
PC 1	PC 2	Target

⇒ generally take that much PCA so 90% variance is explain by those PCAs

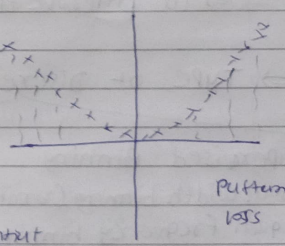
⇒ when PCA does not work



all axis has same variance



not different bet^n two point from upper and lower



Pattern loss

Binning (Discretization) and Binarization

⇒ Encoding Numerical Features

No of downloads → bins

23 → 100+

173457 → 1000+

9927 → 1M+

1B+

✱ Discretization (Binning)

Discretization is the process of transforming continuous variables into discrete variable by creating set of contiguous intervels that span the range of the variable's values. Discretization is also called binning where bin is an alternative name of interval

why use Discretization

⇒ To handel outliers

⇒ To improve the value spread

⇒ Type of Discretization

- Unsupervised Binning
  - equal width binning (uniform) (bins)
  - equal frequency binning (quantile binning)
  - K Mean binning
- supervised Binning
  - Decision tree binning
- custom Binning

i) equal width binning / uniform binning

$$\frac{max - min}{bins} =$$

⇒ Can handel outliers

⇒ No change in spread of data

ii) Equal frequency / quantile Binning

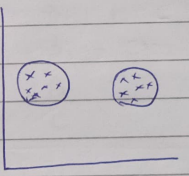
⇒ If intervals is 10 then each interval contain 10% of total observations

⇒ width of each intervals are may different

⇒ make value spread uniform

⇒ can handel outliers

iii) K means binning



⇒ algo used is same as K mean Clustering

⇒ This binning is used when data is spread in form of clusters

⇒ Encoding and Discretized variable

SKlearn

→ KBinsDiscretizer()

- bins=?
- strategy
  - uniform
  - quantile
  - K means
- encoding
  - ordinal
  - one hot encoding

iv) custom / Domain Based Binning

base on your Domain Knowledge you can do binning

[0-18] → Kids

[18-60] →

[60-80] →

★ Binarization (special case of binning)

⇒ Ex For black and white pixel 0-127.5 = 0

127.5-255 = 1

⇒ Ex salary < 6L then 0

> 6L then 1

Mix Data

⇒ Ex 1 cabin

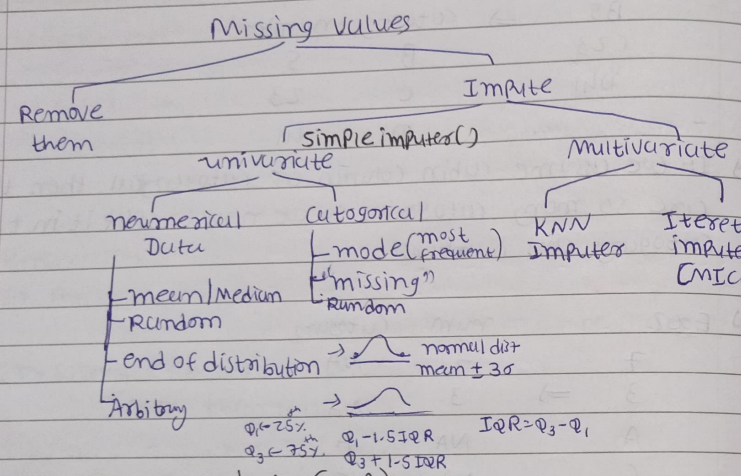
B5	⇒	category	numbers
C23		B	5
D41		C	23
		D	41

⇒ If we assume cabin column as categorical then there are so many categories so we need to split it in to category and numbers

⇒ Ex 2

		num	Category
7		7	NA
3	⇒	3	NA
A		NA	A
C		NA	C

Handling Missing Data



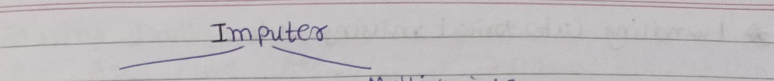
⇒ Complete case analysis (CCA)

⇒ CCA also called "list-wise deletion" of cases consist in discarding observations (raw) where values in any of the variables (columns) are missing

⇒ complete case analysis means analyzing only those observation for which there is information in all the variables in data set

Assumption For CCA: → missing completly random
→ < 5% data missing

⇒ You have to ensure that before and after data remving distribution of data is same.



⇒ can do only help of that column ~~or~~ no need other columns ⇒

⇒ Arbitrary value Imputer:

~~Replace~~ fill missing data with arbitrary random values like 1, 99, 1000, Missing, 0, -1, 5, 8

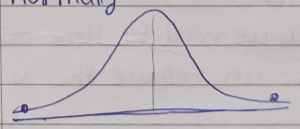
⇒ by giving random values we create difference bet^n where data is available and where not.

⇒ when data is not missing randomly and we cant use CCA then we can use Arbitrary imputer

⇒ disAdv: Change distri, covariance or corr with other columns change, data distri of particular col ↓ variance change

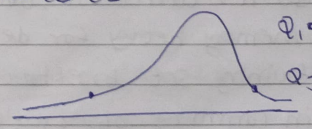
⇒ End of distribution (same like Arbitrary value imputer)
replace value with a or b

If normally



a → mean + 3σ
b → mean - 3σ

else skewed



a → Q1 - 1.5 IQR
b → Q3 + 1.5 IQR
IQR = Q3 - Q1

disAdv: same as Arbitrary value & mean/median

⇒ when data is not missing at random

⇒ Mean/Median
Adv: simple
DisAdv: → Chang distribution → outliers comes in picture (missing value = mean/median) → Covariance or corr with other column change

→ use only when data missing completely at random & < 5% data missing

★ Handling Categorical missing data

⇒ most Frequent value imputation (mode)
fill NA with most frequence value in that column
Assumption : missing completly at random (MCAR)
mode should be much more w.r.t others
missing < 5%

⇒ missing category imputer (same like arbitrary here can't use value so we use 'missing' to indicate that this is missing)
fill NA with 'missing' so that machine learning also know that those data are missing

★ Random Imputation (Numerical & Cato)

⇒ select random number from data and fill it with it
⇒ It can be applied for both numerical and categorical
⇒ distribution and variance is not change much
⇒ covariance matrix is change and this change is random so it's affect on covariance matrix
⇒ memory heavy for deployment, as we need to store original training set to extract value from an replace the NA in comming observation
⇒ well suited for linear models as it does not distort the distribution, regardless of % of NA

★ Missing Indicator

⇒ Ex

Age	Fare	Age_NA
18	32	F
20	35	F
19	32	F
NA	35	T

⇒ adding Age_NA can help a model to differentiate missing values row so model performence is increase
⇒ This method is not work every time but its work sometime

★ KNN Imputer

⇒ 1) find K nearest neighbour
2) find the value

S.No	Feature 1	F2	F3	F4
1	33	—	67	21
2	—	45	68	12
3	23	51	71	18
4	40	—	81	—

to find value in S.No 2 let consider S.No 2 & 3

$$d = \sqrt{\frac{\omega}{3/3}\left[(45-51)^2+(67-71)^2+(12-18)^2\right]}$$

$$\frac{\text{Total number of coordinates}}{\text{number of present coordinates}}$$

let S.No 2 & 4

$$d = \sqrt{\frac{\omega}{3/1}\left[(68-81)^2\right]}$$

For row no 2 d from point 1 is $d_1 = 11.29$

from point 3 $d_3 = 9$

$d_4 = 5.19$

$d_5 = 14.79$

Let consider K=2 so for K=2 we ~~could~~ avg closest 2 point for 1st column for value of 2nd row 1st column value

$$\frac{23+40}{2} = 31.5$$

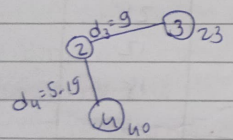
If K=3 then $\frac{23+40+33}{3} = 96/3$

Adv: more accurate

Disadv: more calculation

training set is on server so memory ~~is~~ problem

→ For uniform weight $w_2 = \frac{23+40}{2}$



→ For distance weight

$$\frac{\left(\frac{1}{9} \times 23\right) + \left(\frac{1}{5.19} \times 40\right)}{2}$$

something like that

MCAR : missing completly at random

MAR : missing value can fill by other column (There is relation)

MNAR: " " can not " "

★ Multivariate Imputation by Chained Equations for missing value (MICE Algorithum)

(Iterative Imputer)

Assumption: MAR

(MAR gives best result)

Adv: Accurate

DisAdv: slow

training set is on server

R&D	Administration	Marketing
8	15	30
NaN	5	20
15	10	41
12	NaN	26
2	15	NaN

Step 1 : Fill all NaN with mean of that column

R&D	Adm	Mar
★ 8	15	30
9.25	5	20
15	10	41
12	11.25	26
2	15	29.25

Step 2: Remove all column 1 missing values here 9.25

Step 3: and Predict missing values (left to right)

of column 1 using other columns

R&D	Adm	Mar
8	15	30
NaN	5	20
15	10	41
12	11.25	26
2	15	29

y

x

x = training data input

y = training data op

& predict on ▭

R&D	Adm	Mar
8	15	30
23.111	5	20
15	10	41
12	11.25	26
2	15	29

Step 4: Remove column 2 missing value and Predict using other column do same for all column so after iteration 1 we get below matrix

Iteration 0			Iteration 1			Diff		
RED	Adm	Mar	RED	Adm	Mar	RED	Ad	Mar
8	15	30	8	15	30	0	0	0
9.25	5	20	23.14	5	20	13.89	0	0
15	10	41	15	10	41	0	0	0
12	11.25	26	12	1206	26	0	-0.19	0
2	15	39.25	2	15	31.56	0	0	2.31

doing same process with Iteration 1 as talking ★ until convergence occures.

Outliers

⇒ outliers are mostly affect on weight base algo like Linear, Logistic, Adaboost, Deep learning

⇒ not much impact on Tree base ⇒ random forest, XGBoost gradient boosting

How to treat outliers

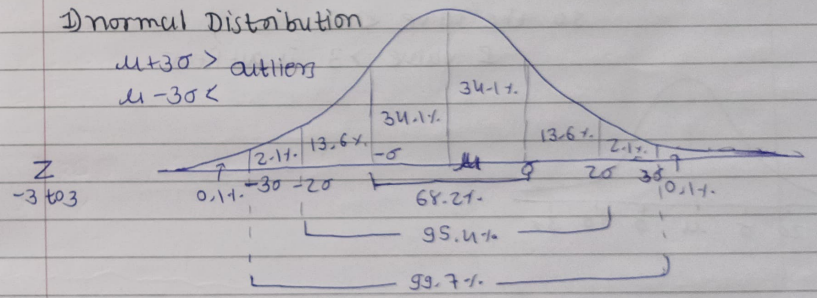
- remove outliers → Trimming
  - Disadv: → thin
  - adv: → fast
- Capping (outlier values is $\mu+3\sigma$ or $\mu-3\sigma$ for normal (which is near))
- missing value
- Discretization

How to detect outliers

1) normal Distribution

$\mu+3\sigma >$ outliers

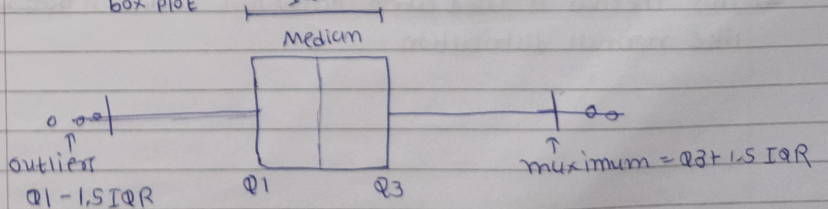
$\mu-3\sigma <$



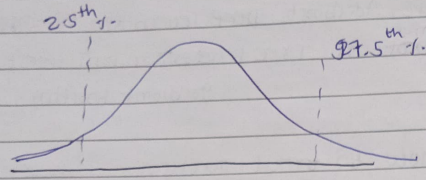
Z
-3 to 3

2) Skew Distribution

box plot — IQR - interquartile range



3) Other Distribution



How to deal with outliers

i) outlier removal using z-score (For like Normal distr)

Capping: assign outlier as $\mu+3\sigma$ or $\mu-3\mu$ whichever is close

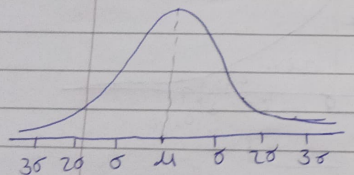
z score

$$x_i' = \frac{x_i - \mu}{\sigma}$$

For like normal distr $\mu \approx 0$

so the value < -3

& value > 3 is outliers

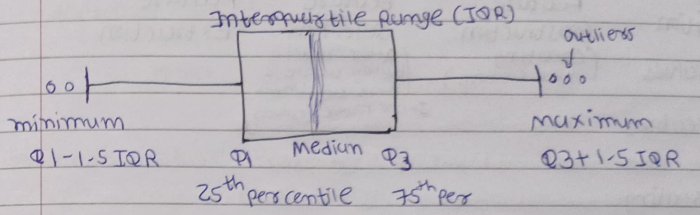


Trimming: remove those row which has outliers

⇒ This method is used when distribution of data is like normal distribution

ii) outlier removing using IQR method

⇒ This method is used when data distribution is skewed



iii) outlier removing using % percentile method

⇒ we can also choose any values like 1th percentile & 99th per.. and can do capping & trimming

Feature Engineering

- Feature Transformation
  - missing values handling
  - Categorical Features
  - Outliers
  - Feature scaling
- Feature construction (creating new features)
- Feature selection (after model building we see is there any benifit if we remove features)
- Feature Extraction

⇒ we have completed Feature transformation now we covers Feature construction where we convert features to new features

★ Feature construction

There is no fix process to do base on domain knowledge only you can do Feature construction

⇒ Ex

Sibsp / Parentchild ⇒ Family type

0 → alone

1 → small

2 → big

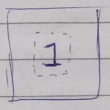
⇒ Feature splitting

Ex Name

Mr AnKit ⇒ Mr | AnKit

Curse of dimensionality

⇒ Performes of model is increasing as we add features but after certain feature If we add more then performer is decreases. Ex



mnist

⇒ more feature : Performes decreases computation

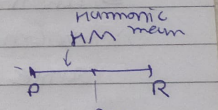
⇒ so we need to do dimentionality Reduction

- Feature selection
  - Forward selection
  - backward elemination
- feature extraction
  - PCA
  - LDA
  - tsne

★ F1 score

F1 score lies near to low values

$$F1 = \frac{2PR}{P+R}$$



Ex P = R = 80

$AM = \frac{80+80}{2} = 80$

$F1\ score = \frac{2 \times 80 \times 80}{160} = 80$

P = 60 R = 100

AM = 80

$F1\ score = \frac{2 \times 6 \times 100}{160} = 75$

⇒ Multiclass Precision & Recall

Actual \ Predicted	Dog	Cat	Rabbit	Total
Dog	25	5	10	40
Cat	0	30	4	34
Rabbit	4	10	20	34
Total	29	45	34	

Precision: what proportion of predicted Positive is truly Positive?

$P_{Dog} = \frac{25}{29} = 0.86$

$P_{cat} = \frac{30}{45} = 0.66$

$P_R = \frac{20}{34} = 0.58$

Macro Precision = avg of all precision

$\text{macro } P = \frac{0.86 + 0.66 + 0.58}{3} = 0.70$

weighted Precision = In this we multiple all precision value with its weight

$\text{weight of class} = \frac{\text{actual Number of that class in data}}{\text{total Data}}$

$\text{weighted } P = \frac{40}{108} \times 0.86 + \frac{34}{108} \times 0.66 + \frac{84}{108} \times 0.58 = 0.71$

⇒ when all classes have same number then use macro otherwise weighted

same for recall also & F1 score

$F1_D = \frac{2 P_D R_D}{P_D + R_D}$

$F1_C = \frac{2 P_C R_C}{P_C + R_C}$

$F1_R = \frac{2 P_R R_R}{P_R + R_L}$

can find macro & weight F1 score

Projects

★ Recommender system

- content base — recommend on base of similarity of content (tags)
- collaborative filtering — recommend content on base of similarity of users
- Popularity base
- Hybrid — combine both

project Flow

1 Data → 2 Preprocessing → 3 Model → 4 website → deploy

Car Prediction

⇒ .str.isnumeric() str is nume or not
⇒ .unique() / .nunique() gives num of unique categories
gives all unique values
⇒ backup = car.copy()
copy dataset into backup
⇒ a.drop('col1', inplace=True, axis=1)
drop col1
⇒ a = pd.read_csv('file.csv')
⇒ data.columns | data.index (Row)
list of all column's name
⇒ a['col1'].value_counts()
gives count of unique values
⇒ series.str.split(' ').str.get(0)
split string with ' ' and (Not show error if not found)
gives 0th element in every row
⇒ series.str.replace(',','')
⇒ car[~car['fuel type'].isna()]
excluding those rows
⇒ series.str.slice(0,3)
gives first 3 element in list
⇒ .join(' ') / ' '.join(list)
Join list with ' '
⇒ df['brand'].replace([re1], 'uncommon') (first, every name in col which is in list replace)

⇒ car.reset_index(drop=True)
⇒ car.to_csv('clean_data')
Save data frame in to csv

⇒ flights.groupby('year').sum().reset_index()

Flights

	year	month	passenger
0	1949	jan	112
1	1949	may	98
2	1950	April	105

	year	passengers
0	1049	1120
1	1050	1180
2	1051	1250
3	1052	1800

House Price

⇒ data.isna().sum()
missing values in every row
⇒ data.drop(columns=['a','b'])
data.drop('col1', axis=1)
⇒ try:
...
except:
⇒ data['col'].apply(fun)
apply function on column
⇒ x.strip() (remove if a is a list: x.rstrip('a'))
remove leading and trailing whitespace
x.strip('abc')
remove abc in x
⇒ lambda x: x² if x in array1 else x
x² If x is in array1 else return x

Book Recommendation System

⇒ books = pd.read_csv('Book.csv', sep=";", error_bad_lines=False)
⇒ books.rename(columns={'a':'b', 'x':'y'}) (dictionary; rename a to b, x to y)
rename the columns of dataset
⇒ ratings['user_id'].isin(y) (np.where(data['days'].isin(['sun','saturday']),1,0))
True: If that user_id is in y
else False
⇒ data.groupby('title').count() ← type = pandas data frame

title	f1	f2
"1	2	5
"2	1	6

← occurence of feature f2 in title1

data.groupby('title')['f1'].count() ← type: pandas series
title is index and f1 is series
whenever do reset_index() then it is become pandas data frame with index 1,2,3... and series Book name & f1

title	f1

⇒ data.drop_duplicates(['col_1', 'col_2'])
drop row which has same col1 → col2 pair

⇒ data.pivot_table(columns='col', index='ind', values='val')

col →
index ↓ (values)

data.fillna(0)

⇒ from scipy.sparse import csr_matrix
csr_matrix(data) ← type = scipy.sparse.csr.csr_matrix
↖ avoid 0th's calculations used when sparse matrix
⇒ data.iloc[2,:] row of df at index 2 series
⇒ data.iloc[2:,:].values numpy.ndarray shape = (n,1)
⇒ data.iloc[2,:].values.reshape(-1,1) shape = (1,n)
⇒ data.iloc[:,:-1]
⇒ distance, suggetions = model.kneighbours(row, no of neighbours you want) (array of)
array of dis, array of suggetions
⇒ data.index[0] index of row 0th
⇒ data.index all the rows index

☆ Laptop Price Predictor

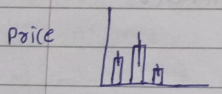
⇒ df['col'].astype('int32') Change the type of col to int32

⇒ sns.displot(df['Price']) 

⇒ df['col'].value_counts().plot(kind='bar')

df.corr()['col1'] corr of col1 to others col

⇒ sns.barplot(x=df['col1'], y=df['col2'])



⇒ df['col'].apply(lambda x: x**2)

⇒ df['col'].str.split(' ')

Pandas Series and that column series contain list of element

⇒ df['col'].str.split(' ', expand=True)

Pandas dataframe contain series (column) of seperated element (with ' ')

Numpy (Numeric Python)

⇒ a = np.array([1,2,3]) size = 3 [1,2,3]

a = a[:, np.newaxis]

size = (3,1) [[1] [2] [3]]

row column

a = a[np.newaxis, :]

size = (1,3) [[1 2 3]]

row column

⇒ x = np.array([[1,2,3],[4,5,6]])

np.ravel(x) [1,2,3,4,5,6]

flute the array x.ravel()

⇒ a = np.array({1,2,2})

{1,2} numpy array

⇒ x[1:,::2]

From row 1 to last & all the column with gap of 2

⇒ x.sum(axis=) 0 row 1 column

⇒ x.sum() Sum of all array

⇒ np.count_nonzero(x)

number of nonzero ele in x

⇒ np.hstack((arr1, arr2))

Combine 2 array horizontally

⇒ np.concatenate((arrays), axis=1)

arr1 = (20,1)

arr2 = (20,1) conc to (20,2)

✓ ⇒ x = [[1,2], [3,4], [5,8]] then x[y=3,1] = (x[y=3][:,1])

numpy array which has y=3 and take it's 1st column (index) all rows

take only those data

df = pd.DataFrame()

Pandas

⇒ df = pd.DataFrame(dict1) ⇒ pd.sort_values('col1')
df.to_csv('friends.csv')
⇒ df.index = ['1', '2', '3'] ⇒ df.sample(5, replace=False, axis=0)
change index to given name ⇒ df['col1'].sum() sum of that series
⇒ pd.Series(list) ⇒ df['col1'].values convert any form DF
Pandas.core.series.Series ⇒ df = pd.DataFrame(np.random.rand(334,5), index=np.arange(334))
⇒ df.dtype data type of each column
⇒ df.to_numpy()
convert df to numpy array
⇒ df1 = df
df1 is pointing to df
df1 is not copy of df
For copy df1 = df.copy()

⇒ df.loc[df['A'] < 5]
loc used when col name
of row name
⇒ df.iloc[1:3, 2:7] row 1→3 & col 2→7
df.iloc[[0,5], [1,2]] row = 0, 5 col = 1, 2

⇒ dropna bydefault how='any' axis=0 (row)
df.dropna()
any = drop If any ele is na
all = drop only If all ele na

⇒ drop_duplicates(subset=['name'])
drop dup in column name
keep = last , False
keep last dup
bydefault first drop all dup

⇒ list ⇒ Dictionaries
→ l = m.copy() l=m change → dic.get('name')
l If we change m not show error while name
→ l.insert(1, num) insert num at position 1 is not found it return none
→ l.extend(m) l → append m while dic['name'] throw error If not found
→ dic = {'name': 'Kurum', 'age': 19}
key value
dic.keys() dict_keys['name', 'age']
dic.items()
dict_items [('name', 'Kurum'), ('age', 19)]

⇒ dic.update(new_dic) add
→ dic.clear() empty
→ dic.pop('name') give value of name remove name
→ del dic['name']
→ def fun(**name) arg take as dict
print(name['fname'], name['mname'])
name(fname = "Buchum", mname = 'Hi')

→ empty_dic = dict() or {}

⇒ sets
→ hi = set() emptyset

⇒ Tuples Tuples can't be change
→ tup = (1,) need comu for single ele
→ tup3 = tup + tup1 This is possible because here we are not change in tupule but making new
→ def fun(* name)
Tuple argument

Plots

⇒ sns.scatterplot(tips['bill'], tips['tip'], hue=['sex'], style=tips['smoker'], size=tips['family_size'])
⇒ sns.barplot(titanic['Pclass'], titanic['age'], hue=titanic['sex']
⇒ sns.boxplot(titanic['sex'], titanic['Age'], hue=titanic['survived'])
⇒ sns.distplot(titanic['Age']) density pdf Area Soft version of histogram
⇒ Pd.crosstab(titanic['Pclass'], titanic['survived'])

Pclass	Survived 0	1
1	80	136
2	97	87
3	372	119

⇒ sns.heatmap( ) & clustermap( )
⇒ titanic.groupby('Pclass').mean()['survived']
⇒ sns.PairPlot(iris, hue='species')
⇒ sns.lineplot(new['year'], new['Passengers'])
⇒ sns.scatterplot(df['Alcohol'], df['malic acid'], hue=df['class'], palette=color_dict)
color_dict={1:'red', 3:'green', 2:'blue'}
⇒ df['col'].plot.density(color='red')
multiplot
⇒ plt.hist(Pd['col2'].values, bin=100)
⇒ Sns.KdePlot(data['Age']) same distplote KDE Kernel density estimation curve=1
, bw_adjust=5) In PDF (Prob density function)