| Sr. No. | Title | Page No. | Sign./Remarks |
|---|---|---|---|
| | | | |

⇒ why multicollinearity bad?

$y = w_1x_1 + w_2x_2 + \dots + w_0$

→ In this eq we assume that $w_i$ is denote how y increase or decrease with respect to $x_i$ while all other $x_j$ ($j \in 1$ to $n$, $j \neq i$) is constant

→ Coefficient $w_i$ gives relation bet$^n$ y and $x_i$ while all other x is constant

Is multicollinearity bad
- No, If we want prediction only using linear regression
- yes, If we need to find relation bet$^n$ y & x by using w because here w is not important it has no meaning If there is multicollinearity

Types

Structured
by creating new feature
Ex one hot encoding

| City | A | B | C |
|---|---|---|---|
| A | | | |
| B | | | |
| C | | | |

need to remove one column

Database
Data itself contain multi collinearity

How to remove it → increase data
→ remove one correlated column
→ use Lasso, Ridge regression
→ use ~~PCA~~ Principle Component Regressor

⇒ multicollinearity more affect on Parametric ML algo Ex Linear, logistic regression, Nivebase, KNN algo where we take assuption that all input columns are independent

while non parametric Ex Tree base algo and algo where no assumption that input cols are indepent

⇒ Fit(), transform(), fit_transform()

Transformers: Standard scaler, MinMax scaler, PCA, Imputer ect.

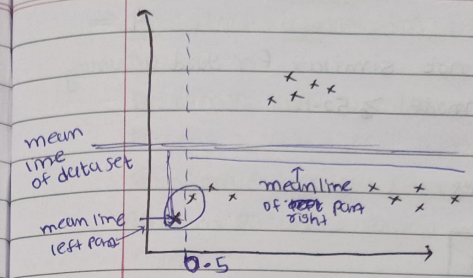
fit() is only compute Ex compute $\mu, \sigma$, compute none values using mean, median

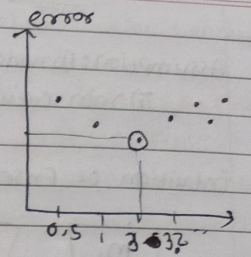
transform() transform data Ex convert each data point to $z = \frac{x-\mu}{\sigma}$

Imputer put values to nan values

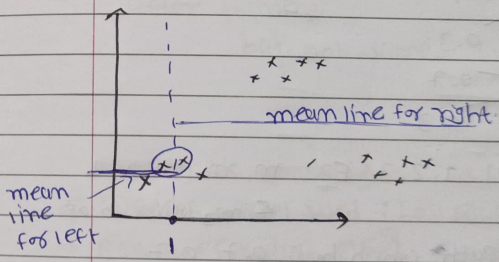
→ fit() Compute parameters and weights

# Regression Trees



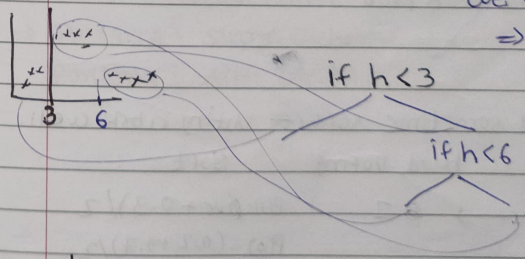

⇒ In this method every time we take two successive point and ~~for~~ draw line at mid of both and for every part (left and write) we take ~~mean~~ ~~MSE~~ SSE (some square error) and plot graph for every two successive point and consider those vertical line for which SSE is min.

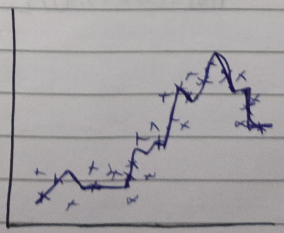



⇒ we do this for every two successive and ~~then~~ take that ! line which ~~less~~ min error

⇒ now for every ~~two~~ part left of line ! and right of ! line we do same step

⇒ when point less then threshold we stop process for that part



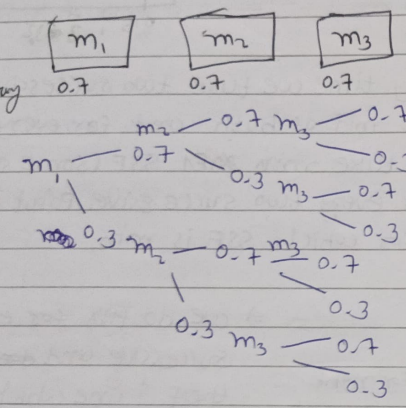



Regression Curve

# Voting Ensemble

Assumption: i) model should be not similar for good accuracy
ii) min accuracy of every model ≥ 50%

intuition of Ensemble techniques

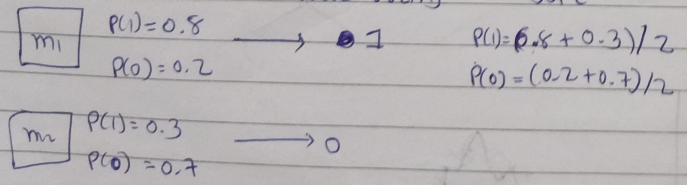



op is max op from model 1, 2, 3 Ex: $m_1, m_2$ gives 1 & $m_3$ gives 0 op

then we choose those path which has 0.7, 0.7 and find max prob

Hard & soft voting

⇒ Some time hard and sometime soft voting work well.




⇒ Voting & boosting can have diff weight while Bagging has same.

# Bagging

⇒ Bootstrap aggregation : for same model diff data training

Bagging: row sample with replacement bootstrap = True
Pasting : " " without " bootstrap = False
Random subspaces : Column sampling with or without replacement
Random Patches : Row and column sampling with or without

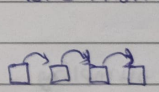
OOB (out of Box) : There are some ~36% data exist which is not feed into any of model

→ so we can use that data in testing arr also
oob_score = True inside bag = BaggingClassifier(... )
after training
bag.oob_score_ give accuracy of that OOB data

Bagging Vs Boosting

⇒ Type of model used : Low Bias high vari. (Bagging) High bias low variance (Boosting)

⇒ Bagging = Parallel
Boosting = sequential



⇒ Bagging = same weightage
Boosting = diff weights

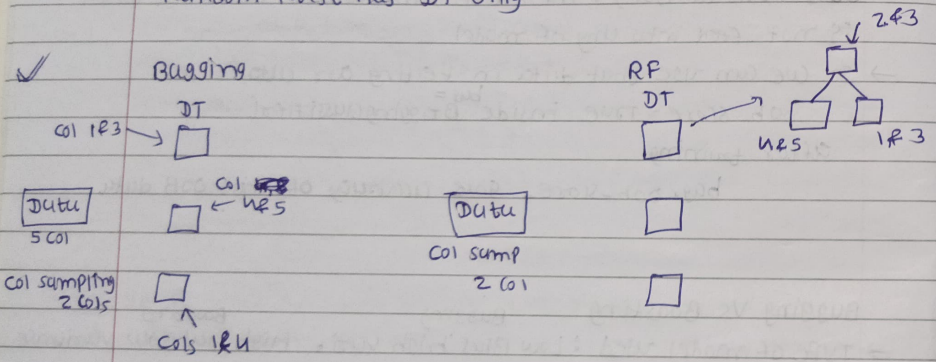
# Random forest

⇒ In Random forest noisy data distribute among all the DT and not feed in one DT so impact of noisy data distribute over trees not store in one DT

⇒ Difference between Random Forest and Bagging:

Bagging can have → DT, any also
CNN
SVM

Random forest has DT only




⇒ Bagging : Tree level sampling
before every tree columns are fixed

⇒ RF : node level sampling
in every tree, every node has select Random columns.

# Feature Importance using Random Forest and Decision Trees

⇒ For DT:

(sum of importance of every node with column C1) / (Imp of all cols)

gives feature importance of C1

node importance = IG = information gain of that node

Σ Feature imp = 1

⇒ impurity based feature importance can be misleading for high cardinality features (many unique values)

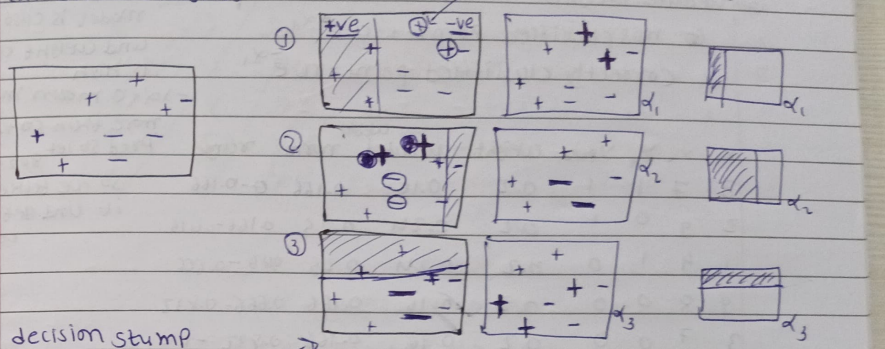
in this kind of case use sklearn.inspection.permutation_importance

# Adaboost

Stage wise additive method

⇒ different weights

upsampling wrong classified point




decision stump (decision tree with depth = 1)

can use any model

Ex: logistic regression, SVM

(KNN not used ∵ sample base...)

$$h(x) = \text{sign}\left(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)\right)$$

| $x_1$ | $x_2$ | y | $y_{pred}$ | weight ($y_m$) |
|---|---|---|---|---|
| 3 | 7 | 1 | 1 | 0.2 |
| 2 | 9 | 0 | 0 | 0.2 |
| 1 | 4 | 1 | 0 | 0.2 |
| 9 | 8 | 0 | 0 | 0.2 |
| 3 | 7 | 0 | 0 | 0.2 |

wrong pred

★ $x_1 > 5$, $x_2 \gtrless 10$, $x_1 > 2$, $x_1 > 1$, $x_2 > 9$ ...

Decision stump

Choose DS which has high IG

let that DT has model $m_1$

⇒ now find weight of that DS

weight = $\alpha_1$

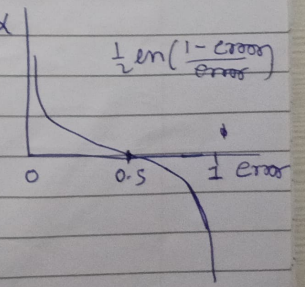
depends on error rate

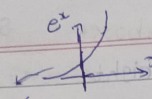
error for $m_1$ = 0.2 + 0.2

error = 0.4

$$\alpha_1 = \frac{1}{2} \ln\left(\frac{1-0.4}{0.4}\right)$$

$$\boxed{\alpha_1 = 0.2}$$





If model has good then α of that model is also high and weight change is high

→ α<0 mean model gives more than 50% wrong Pred so let 80% wrong so we take ~ of it and get 80% correct

→ update weights

misclassified Point = $we^{\alpha_1}$

Correctly classified point = $we^{-\alpha_1}$

| $x_1$ | $x_2$ | $y_{in}$ | $y_{pred}$ | weight | updated weight | norm | range |
|---|---|---|---|---|---|---|---|
| 3 | 7 | 1 | 1 | 0.2 | 0.16 | 0.166 | 0-0.166 |
| 2 | 9 | 0 | 1 | 0.2 | 0.24 | 0.25 | 0.166-0.416 |
| 1 | 4 | 1 | 0 | 0.2 | 0.24 | 0.25 | 0.416-0.666 |
| 9 | 8 | 0 | 0 | 0.2 | 0.16 | 0.166 | 0.666-0.832 |
| 3 | 7 | 0 | 0 | 0.2 | 0.16 | 0.166 | 0.832-1 |
| | | | | | 0.96 | 1 | |

5 random number bet$^n$ 0-1

let random numbers are 0.13, 0.43, 0.62, 0.3, 0.11

so we take 1, 3, 3, 2, 1

row for next model and same step as ★ to till now.

Final op of any input is $\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x) + \dots$

⇒ we can assign Learning Rate

LR · α

So change in weight is small

So update of Amplitude will be small and reduce overfitting.

## Gradient boosting

CG

| salary | Pred1 | Pred2 |
|---|---|---|
| 8 | 3 | 4.8 |
| 7 | 4 | 4.8 |
| 6 | 8 | 4.8 |
| 9 | 6 | 4.8 |
| 5 | 3 | 4.8 |

Pred2 = -1.8

# Gradient Boosting

here overfitting hence consider ex

| iq | CG | lpa | base Pred1 | res1 | Pred2 | res 2 | Pred 3 |
|---|---|---|---|---|---|---|---|
| 90 | 8 | 3 | 4.8 | -1.8 | -1.8 | -1.62 | -1.62 |
| 100 | 7 | 4 | 4.8 | -0.8 | -0.8 | -0.72 | -0.72 |
| 110 | 6 | 8 | 4.8 | 3.2 | 3.2 | 2.88 | 2.88 |
| 120 | 9 | 6 | 4.8 | 1.2 | 1.2 | 1.08 | 1.08 |
| 80 | 5 | 3 | 4.8 | -1.8 | -1.8 | -1.62 | -1.62 |

Pred1 - 2Pred2 - 2Pred3 ...

⇒ Pred 1 of base model is avg of O/P column lpa and then we train model1 with

I/P = CG, iq & O/P = res 1

m1 and model1 gives op Pred2

So actual op from Pred2 can be predicted by

op base model + op of model1(Pred 2)

(Pred1)

⇒ To reduce overfitting we introduce Learning rate to Pred op

So actual output = $m_0 + \alpha m_2$

(op of model1 = base (Pred1); op from model2 = (Pred2))

Pred

⇒ after that find res 2 = actual op − actual pred

lpa

for $\alpha = 0.1$ we calculated res2 is in above table

$3 - [4.8 + 0.1(-1.8)] = -1.62$

m2 ⇒ for model 2 ,

Train on I/P = iq, CG O/P = res 2

and model2 gives op Pred 3

actual op = $m_0 + \alpha m_1 + \alpha_2 m_2$

Pred from m2 ; op $m_1$ ; op $m_2$

Pred

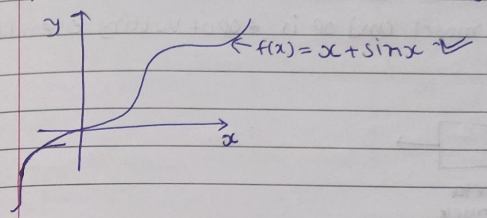
$$F(x) = h_0(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x) + \dots$$

Final op ; hyper para ; hyper

⇒ In Boosting we use DT as function and each fun is act like small part of final op function (Additive modeling)

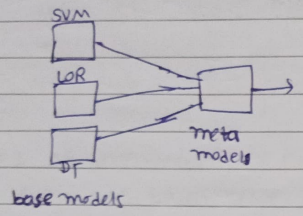
## ★ Additive Modelling

Additive modelling function has sum of small functions



# Stacking

⇒ Same as voting but we add new ML model which take input as op of other model and op is ip of voting essembel

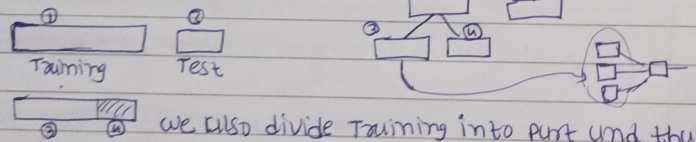



⇒ Stacking we have problem:

If base model are like DT then its give op ~~which~~ which contain overfitting and meta model also become overfited model

→ Solution:

1) Hold out method: Bending

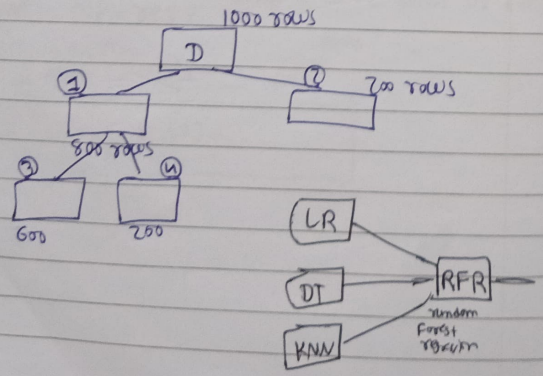



we also divide Training into part and that part is test for meta model

~~2. Cross validation: Bagging~~

| CG | iq | Package |
|---|---|---|
| 7 | 70 | 4 |
| 9 | 101 | 5 |
| 8 | 120 | 6 |
| - | - | - |

1000 rows




Step 1: Train 3 base models on ③

Step 2: using This 3 model and pred on ④

| | Package | LR_Pred | DT_Pred | KNN_Pred |
|---|---|---|---|---|
| dataset → | | | | |

shape = (200 rows, 4 cols)

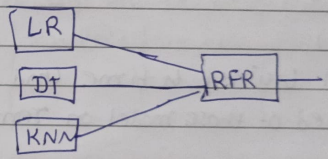
Step 3: Train meta model on step2's dataset

IP = Pred of 3 model
OP = Package (given)

~~op of base models trains ② gives to op of RFR data while training of meta model~~

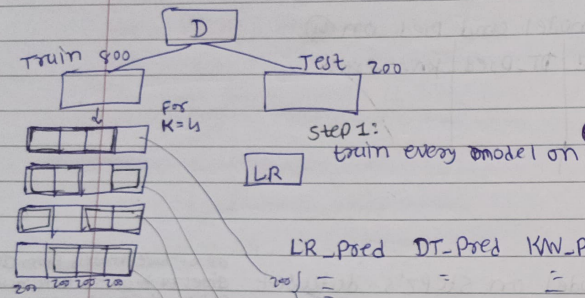
Step 4: For testing this model

take ② data and pred on ③ model




⇒ by this method we can overcome overfitting (∵ while training of meta model base models does not ~~for~~ has training on ④)

ii) K-Fold approach - Stacking




step 1: train every model on 600 data Pred on 200 data

Shape = (800, 4)

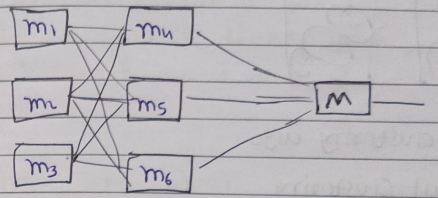
⇒ Here every model trained 4 time and above dataset contain Pred of those model on remaining 200 data in each time

Step 2: using above data train meta model

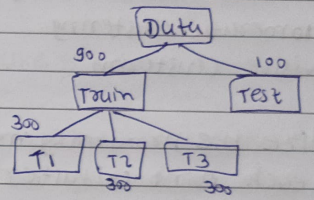
Step 3: For base model
Train base models on Train data (800) and you get your base models

Step 4: now train this model  on Test (200) data

★ Multi-layer stacking




can have multi layers also




Step 1:
Train $m_1, m_2, m_3$ on T1 and Pred on T2 and we got

| | $m_1$ Pred | $m_2$ Pred | $m_3$ Pred | Package |
|---|---|---|---|---|
| Shape (300,4) | | | | |

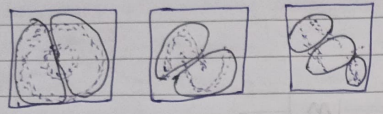
Step 2: Train $m_4, m_5, m_6$ on This dataset and pred on T3

| | $m_4$ Pred | $m_5$ Pred | $m_6$ Pred | Package |
|---|---|---|---|---|
| Shape (300,4) | | | | |

Step 3: now train meta model M on This data (300,4)

# Hierachical Clustering intution

⇒ K means clustering not gives good result on



So we need other clustering algo

⇒ Types of Hierachical Clustering
1) Agglomeratic clustering
2) Divisive Clustering

Algo: 1) initialize the proximity matrix
2) make each point a cluster
3) inside a loop
   a) merge the 2 closest clusters
   b) Update the proximity matrix
4) Until only one cluster is left

1)

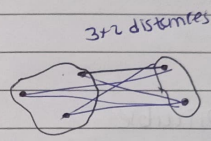
| | $P_1$ | $P_2$ | $P_3$ | $P_4$ | $P_5$ |
|---|---|---|---|---|---|
| $P_1$ | 0 | – | – | – | – |
| $P_2$ | – | 0 | – | – | – |
| $P_3$ | – | – | 0 | – | – |
| $P_4$ | – | – | – | 0 | – |
| $P_5$ | – | – | – | – | 0 |

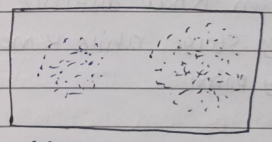
→

| | $P_1$ | $P_2$ | $C_1$ | $P_5$ |
|---|---|---|---|---|
| $P_1$ | 0 | – | – | – |
| $P_2$ | – | 0 | – | – |
| $C_1$ | ? | ? | 0 | ? |
| $P_5$ | – | – | – | 0 |

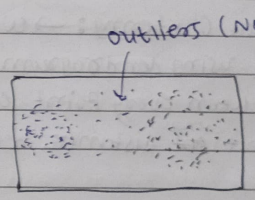
⇒ Type of Agglomeratic clustering
1) Min (single-link)
2) Max (complete-Link)
3) Average
4) ward

1) single Link (min)



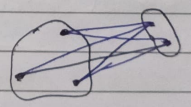



work well

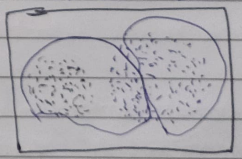



does not

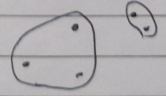
2) Complete link (max)



⇒ Complete link work well on outliers or noisy data
⇒ but not work well on

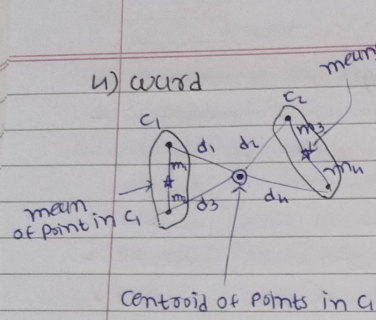


3) Grop average



$$\frac{\sum d_i}{6} \quad i=1,2,\ldots 6$$

4) ward




$$\text{dist}_{C_1 \to C_2} = d_1^2 + d_2^2 + d_3^2 + d_4^2 - m_1^2 - m_2^2 - m_3^2 - m_4^2$$

Variance minimize

Adv of Hierarchical Clustering: → widely applicable
→ by using dendrogram we can know distance between every point at any stage while K mean can't ~~get~~ know distance bet^n any point

disAdv: → can not use on big data sets because distance matrix takes large memory

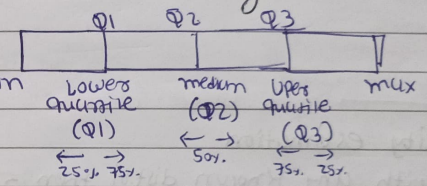
Statistics — Descriptive, Inferential

★ Coefficient of variation (CV): represented amount of variability in a dataset

$$CV = \frac{\sigma}{\mu} \times 100$$

Can compare spread of to diff column we use CV

★ 5 number summary in statistics




★ Correlation and causation
⇒ The phase "Correlation does not ~~prove~~ imply causation" means that two variables are associated with each other it does not mean that one causes the other
⇒ Eg: more icecream sale more murder ~~reason~~ reason was weather

★ Random Variable
⇒ Algebric Variable
x is unknown value $x + 5 = 10$

$$\boxed{x = 5}$$

⇒ Random variable: a random variable is a set of possible value from a random experiment.
Toss Experiment
$X = \{H, T\}$
↑ sample space values that can have RV.

★ Parametric density estimation

⇒ is method to estimate prob density function (PDF) of random variable by assuming that underlying distribution belongs to specific parametric family of prob distri such as normal, expo, poisson distri.

⇒ We have sample of μ & σ

by using assumption of particular distribution find Probability of all point and make graph

★ Non-Parametric density estimation

⇒ If distri not match with any known distri then use...

⇒ It is statistical technique used to estimate PDF without making any assumption about underlying distribution.

⇒ It is called non-Para density estimator because it doesn't required any assumption or predefine distri function.

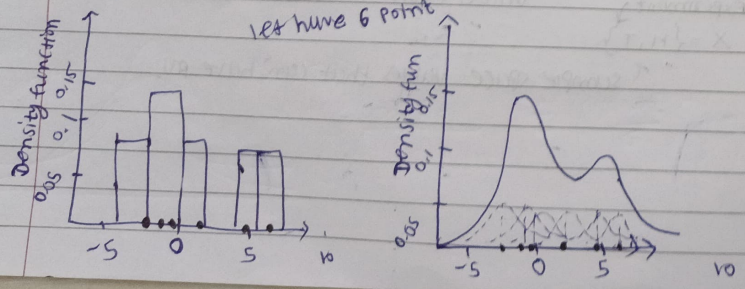
⇒ This technique involves construct estimate of PDF using the available data. This done by creating Kernal density estimate.

adv: can find PDF of any kind of data

dis adv: Computationaly expensive

not give good result with less data.

KDE: Kernal density fun to smooth out the data and create a continuous estimate of the underlying density function.




⇒ For every point consider as mean and draw gaussian distribution or Kernal. and all all graph

⇒ mean = each point } For Kernal (can be any Ex gaussian

σ = bandwidth

↑ hyper para

```
from SKlearn.neighbour import KernalDensity
model = KernalDensity (bandwidth=3, Kernal='gaussian')
model.fit (x_train)
```

exponential, linear and more

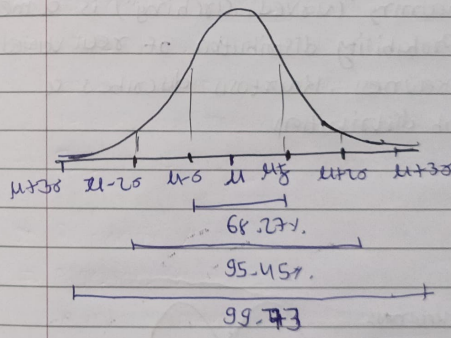
OR

using Charm

```
sns.Kdeplot (sample, bw_adjust=5)
```

↑ x

★ Empirical Rule



momentum 1st → mean
2nd → std
3rd → skewness
4th → kurtosis

PAGE NO.:

★ Non-gaussian distribution

Continous non-gaussian distribution
- uniform distri
- Log-normal distri
- Pareto distri
- T-distri
- Chi-square distri

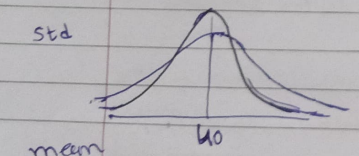
Discrite non-gaussian distri

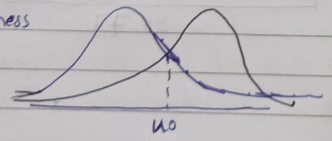
⇒ non-gaussian distri —transformation→ gaussian distri

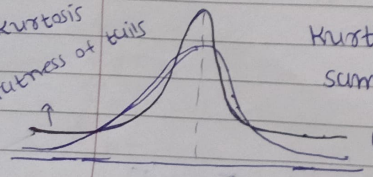
★ Kurtosis

⇒ Kurtosis is 4th statistical moment. in probability theory and statistic, kurtosis (meaning "curved, arching") is a measure of the "tailedness" of probability distribution of real valued random variable. Like skewness, kurtosis describes a particular aspects of prob distribution

2007 → 100 matches
2008 → 100 matches




Kurtosis:

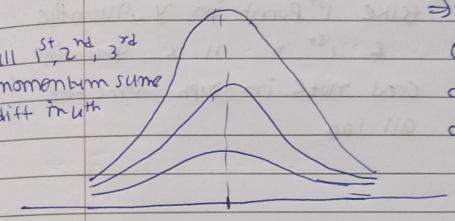
$$\text{sample} \left\{ \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i}^{n} \left( \frac{x_i - \bar{x}}{\sigma} \right)^4 \right\} - \frac{3(n-1)^2}{(n-2)(n-3)}$$

⇒ practical use-case of kurtosis

→ In finance kurtosis risk refers to risk associated with possibility of extreme outcomes or "fat tails" in the distribution of returns of particular asset or portfolio.

→ If distribution has high kurtosis means there is higher likelihood of extreme event occurring either positive or negative, compared to normal distribution.

⇒ Excess kurtosis & Types




⇒ Excess kurtosis is measurement of how much more peaked or flute distribution is compared to normal distribution, which is considered to have kurtosis of 0 it is calculated by substracting 3 from sample kurtosis coefficient.

Types: Leptokurtic
Mesokurtic
Platykurtic

Leptokurtic: (sample kurtosis − 3 > 0)

A distribution with positive excess kurtosis called leptokurtic "Lepto" means "slender" in term of shape. It has a fatter tails. This indicates that there are more extreme values

Platy
~~Meso~~kurtic < 0

negative excess kurtosis and fewer outliers

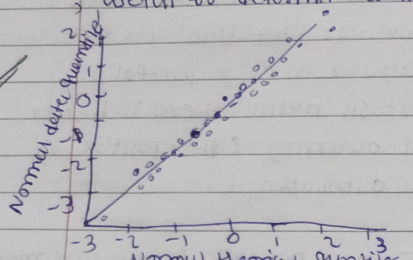
Mesokurtic = 0

~~similar~~ normal distri (regardless of any μ, σ)

PAGE NO.:

★ Quantile - Quantile Plot (Q-Q Plot)

useful to determin whether set of data follows normal distn




$x \rightarrow$ sort

$\hookrightarrow$ quantiles $\rightarrow$ 100

$1^{st}, 2^{nd}, 3^{rd} \dots 100^{th}$ quantiles

percentage

take $1^{st}$ point in y_quantile

& $1^{st}$ " " x_quantile

and note in graph same for all 100

PAGE NO.: