

# Machine learning

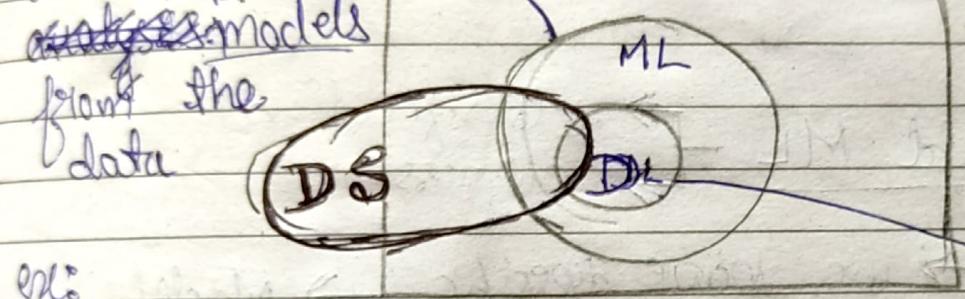
Darshan R.M.

Week - 13

## Machine learning Part - 1

\* AI vs ML vs DL vs DS

It provides statistical  
to learn, analyze,  
visualize and  
develop predictive  
~~analysis~~ models  
from the  
data



Smart application  
that can perform  
its own task  
without any  
human intervention  
ex: self driving  
Car,  
Robots

Amazon  
Netflix → Recommodation  
System

Deep learning

- Mimic Human being

[Multi layered neural  
networks]

ex: object detection,  
image recognition,  
chatbot, chat GPT

Recommommendation  
System

## Mathematics

- ① Linear algebra.
- ② Statistics.
- ③ Probability.
- ④ Calculus.

## \* Types of ML

- ① Supervised ML.
- ② Unsupervised ML.
- ③ Semisupervised ML.
- ④ Reinforcement.

① Supervised ML  $\xrightarrow{\text{classification}}$   $\xrightarrow{\text{regression}}$

Dataset  $\rightarrow$  we have specific output  $\rightarrow$  Model trained

do  $\downarrow$  predict  
o/p

(Independent feature)

(Classification input)

in <sup>o</sup> . Play hours.	study hours	o/p (dependent feature)
8	2	Fail
7	3	Fail
3	6	Pass

o/p (dependent feature)  
Pass/ Fail

8

2

Fail

7

3

Fail

3

6

Pass

No of outputs are fixed.

~~class~~ Categorical feature

$\rightarrow$  classification

if 2 categories binary feature

( $\theta_0 + \theta_1 x$ ) Regression

O/p  $\Rightarrow$  Continuous feature

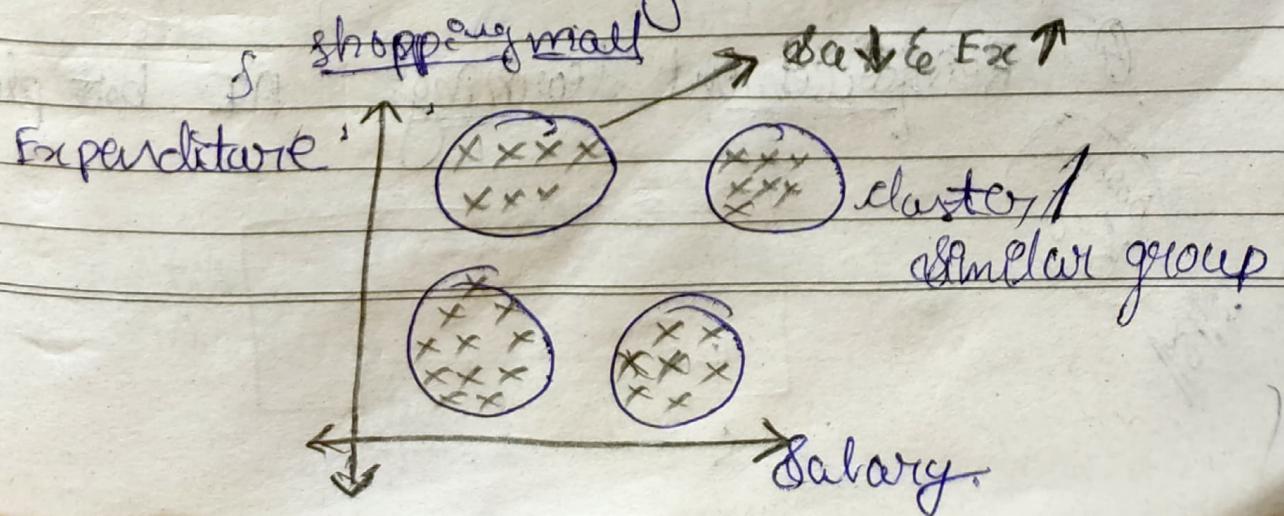
C2CB

Size of house	No. of Rooms	O/P Price	Continuous

## ② Unsupervised ML:

Dataset  $\rightarrow$  clusters / similar groups

Ex: Customer segmentation



To launch product  $\rightarrow$  Based on clusters

20% dis count  $\leftarrow$  Target the audience

\* In Unsupervised  $\rightarrow$  No off as such

we try to create clusters.

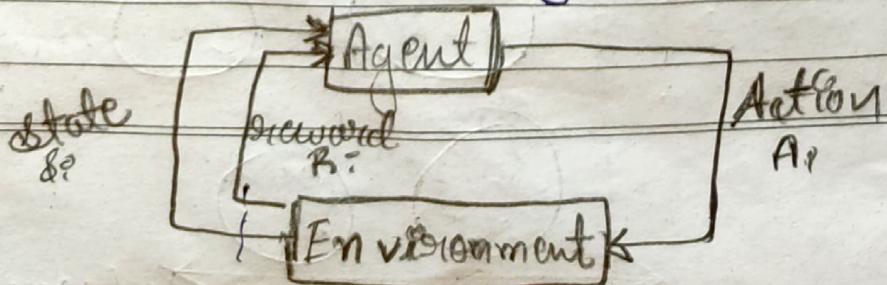
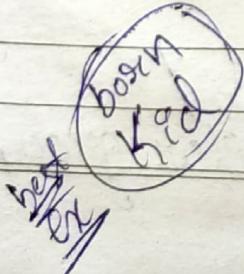
\* after we can apply supervised learning also

③ Semi-supervised : Supervised + Unsupervised

e.g. Netflix has so many models based on region.

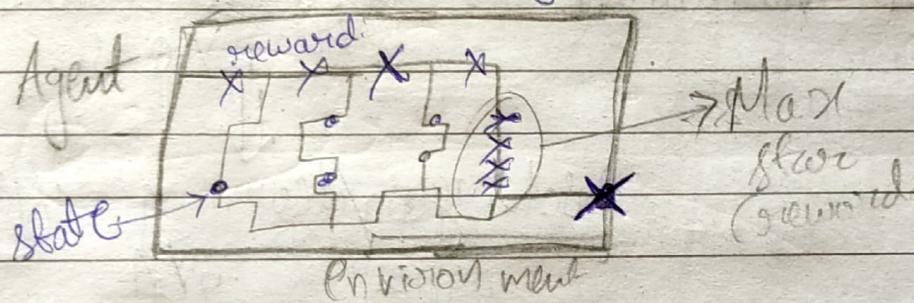
first it apply clustering  $\xrightarrow{\text{then}}$  supervised to predict output

A Reinforcement learning: AI bot playing

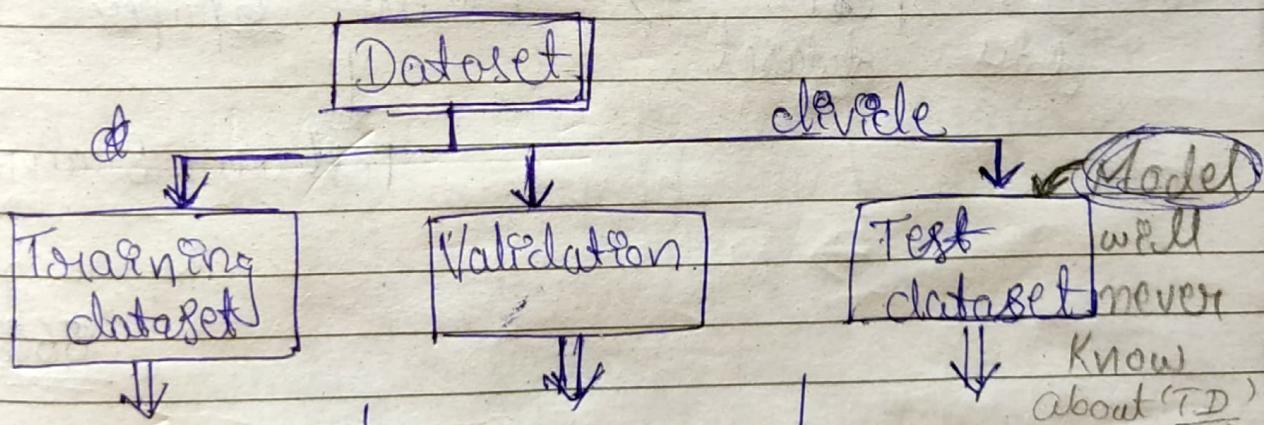


Reinforcement learning is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward.

Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.



### \* Train, Test, and Validation



We will Train the model

Hyperparameter Tuning of the model.

We tweak some parameters to get more accuracy

Test the model

How model will work

accuracy % ]

Q&A

Exam

Training dataset  $\Rightarrow$  Books  
 $\downarrow$   
using Q/A  $\Rightarrow$  Train

Validation  $\Rightarrow$  different books and  
Q/A

$\downarrow$   
Solving/learning different  
types of Q/A  $\Rightarrow$  tuning  
parameters

Model should  
not know  
about test  
data dataset  $\Rightarrow$  Exam paper

$\downarrow$   
Test our Brain

we get  
"86%"

tendency of model  
to make systematic  
errors.

tendency of model  
to make random  
errors.

## \* Bias, Variance, Overfitting and Underfitting

\* Model performance  $\Rightarrow$  Accuracy  $\uparrow \rightarrow$  high

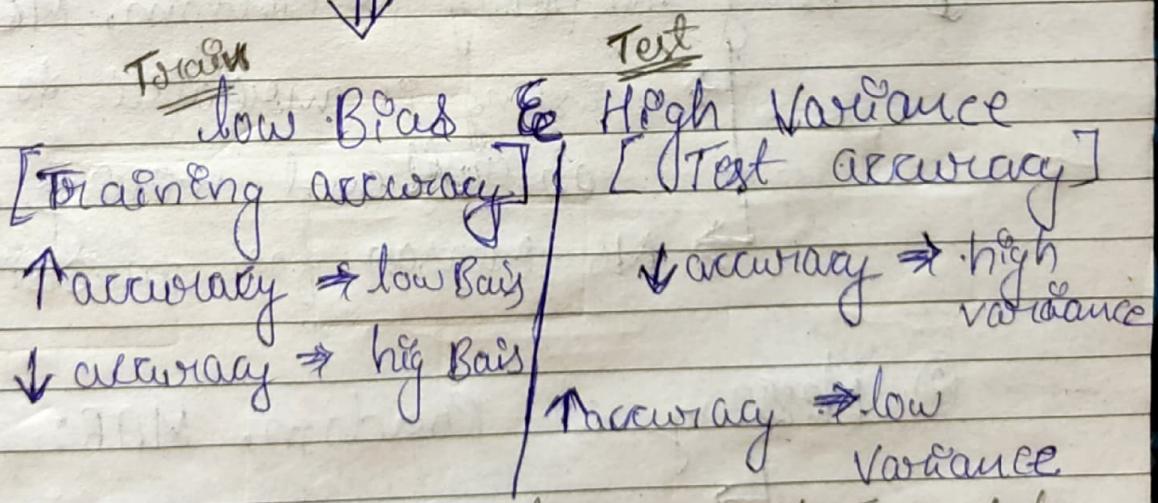
### ① Overfitting and underfitting

#### Dataset

~~Book~~  $\rightarrow$  Train  $\rightarrow$  Model is Trained  $\nrightarrow$  Accuracy  $\uparrow \xrightarrow{\text{extra}} 95\%$

exam  $\rightarrow$  Test  $\rightarrow$  Model is Tested  $\downarrow$

Train is overfitted  $\rightarrow$  Overfitting: Accuracy  $\downarrow \rightarrow 65\%$   
to Book



Model not properly trained with Train data

② Train  $\rightarrow$  accuracy  $\downarrow \Rightarrow 55\%$

Test  $\rightarrow$  accuracy  $\downarrow \Rightarrow 50\%$

High Bias & High Variance  
(More error)

Underfitting

### ③ Generalized model

Train  $\Rightarrow$  Accuracy ↑↑  $\rightarrow 85\%$   
Test  $\Rightarrow$  Accuracy ↑↑  $\rightarrow 86\%$

out of all ~~am~~  
should be. low Bias and low Variance

### \* Feature engineering \* Handling missing values

Survey in Google form some fields are not filled.

3 mechanism why missing value occur

#### ① Missing Completely at Random, MCAR:

- missing is unrelated to both observed and missing data.
- No reason for missing.

#### ② Missing at Random, MAR:

- probability of value being missing depends only on the observed data
- missing value are related to observed data.

### ③ Missing data not at random, MNAR:

- % of missing values depend on the value of missing data itself.
- Missingness is not random.  
dependent on unobserved / unmeasured factors.

\* > df.isnull() gives True if value is null

> df.isnull().sum() column null value  
Count =

> # delete rows or data point

df.dropna() → row wise

\* deleteria: loss of data

df.dropna(axis=1) → Column wise

\* Imputation missing values

① Mean value computation → for normal distribution

> df['new']=df['age'].fillna(df['age'].mean())

② Median value imputation  $\Rightarrow$  than normal distribution other distribution

> `df['age'].median() = df['age'].fillna(df['age'].median())`

③ Mode value imputation  $\Rightarrow$  for Categorical value.

> `df['embarked'].mode()`

> `df[---].unique()` # possible values

> `df.dtypes`

> mode  $\Rightarrow$  `df[df['---'].notna()]['---'].mode()`

> `df.fillna(mode)`

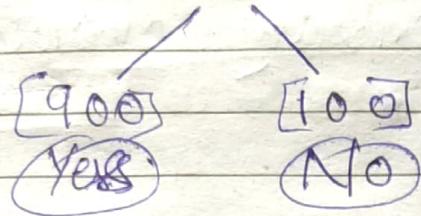
## \* Handling imbalance dataset

[Dataset]

classification  $\rightarrow$  o/p in Categories

↓  
out come then  
Binary classification

1000 dataset



ratio =  $\frac{9}{1}$  → Imbalanced dataset

↓  
Model → will be biased to  
Yes Category

How to handle imbalanced datasets

- ① UpSampling.
- ② DownSampling. (bad)

→ df['target'].value\_counts()

sklearn.utils import resample  
with replacement

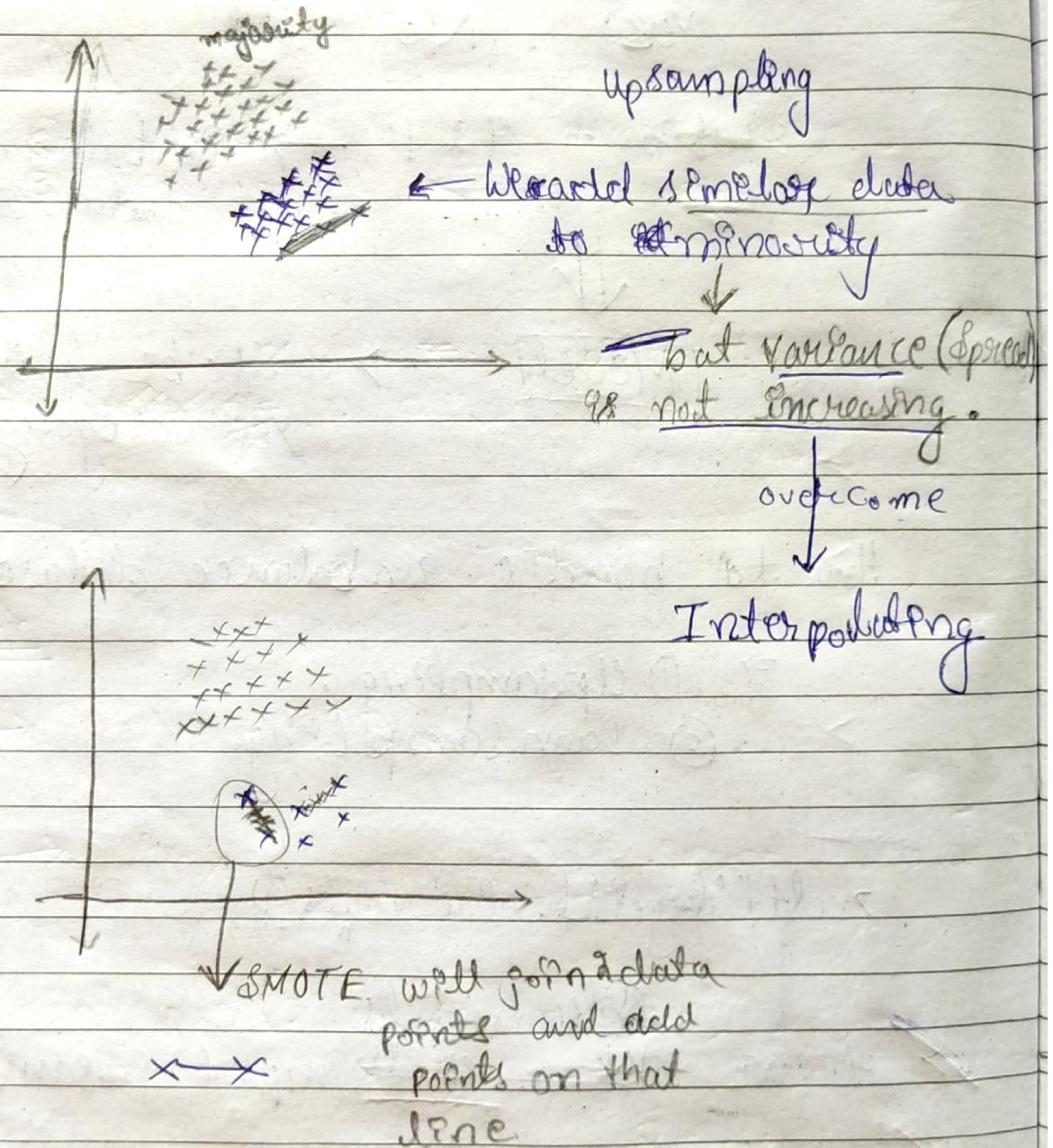
df['new'] = resample(df\_majority, replace=True,  
n\_samples=df['target'].sum(),  
random\_state=42)  
↳ set seed value

df.shape  
(100, 3)



## SMOTE

↳ Synthetic minority over sampling Technique



> follow 8k learn. datasets import  
make classification

used to create  
binary / multiple  
class dataset

df1 df2  
 ↓ ↓  
 >  $x, y = \text{make_classification}(\text{n\_samples} = 1000,$   
 $\text{n\_redundant} = 0, \text{n\_features} = 2,$   
 $\text{n\_clusters\_per\_class} = 1, \text{weights} = [0.9],$   
 $\text{random\_state} = 12)$

+ pyp install imblawn  
 \* over sampling  
 > from imblawn import SMOTE

~~over sample = SMOTE()~~  
 ~~$x, y = \text{oversample. fit\_resample}($~~   
~~↑ ↑~~  
~~df1 df2~~  
~~final df [ 'f1', 'f2' ]]~~  
~~final df [ 'target' ] )~~  
 ✓  
 concat  
 ↓  
 plot

> plt.scatter(df['f1'], df['f2'], c=df['target'])

## \* Data interpolation.

→ estimating unknown values within a dataset based on the known values.

① Linear interpolation.

② Cubic interpolation. [scipy]

③ Polynomial interpolation.

① Linear

$$x_{\text{new}} = np.linspace(1, 5, 10)$$

~~np.interp(x\_new, x, y)~~

(5) 10 data points

Known

Unknown

② Cubic

$$y = x^3$$

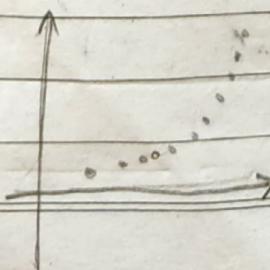
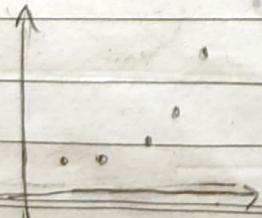
Interplot ID

from: scipy.interpolate import interp1d

→  $f = \text{interp1d}(x, y, \text{kind}='cubic')$

→  $x_{\text{new}} = np.linspace(1, 5, 10)$

$y_{\text{interp}} = f(x_{\text{new}})$



$$y = x^3$$

③ Polynomial.

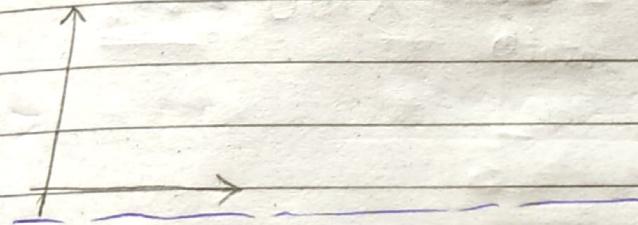
$$y = x^2$$

Polynomial

$p = \text{np.polyfit}(x, y, 2)$

$\text{linspace}(1, 5, 10)$

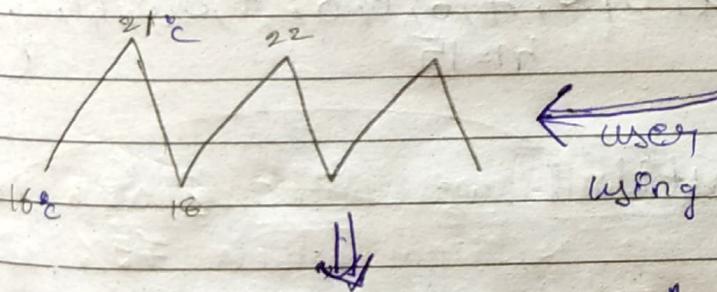
$y\text{-interp.} = \text{np.polyval}(p, x_{\text{new}})$



\* Use Case: (House) [pamisonic]

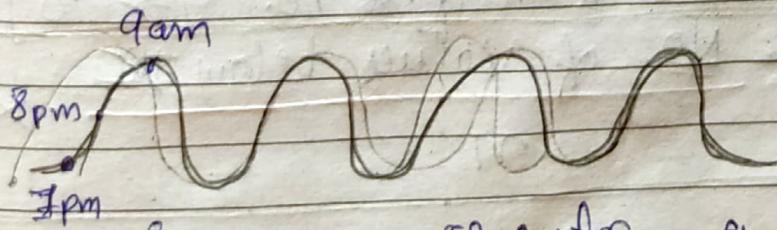
AI model provides  $\rightarrow$  steep profile

AC's  $\rightarrow$  previous data.



Electricity

loss of electricity waste  
as sudden pumping



So we create a sine wave instead of bump up

## \* Percentiles and Quartiles

(\*) Percentage: 1, 2, 3, 4, 5, 6

\* what % of the numbers that are odd

$$\Rightarrow \% \text{ of the odd} = \frac{3}{\text{total } 6} \times 100 = \underline{\underline{50\%}}$$

## \* Percentiles

defn A percentile is a value below which a certain percentage of percentage of observations (or) data points lie.

Sorted values 1 2 3 4 5

$$x = 2, 2, 3, 3, 4, 6, 6, 6, 7, 8, 8, 9, 9, 10, 11, 12 \\ n=15$$

75 percentile: we will find a value below which 75% of entire distribution lies

① Percentile Rank of 10.  $\boxed{x=10}$

$$\text{percentile} = \frac{\text{No of Values below } \boxed{x}}{n} \times 100$$

$$= \frac{12}{15} \times 100 = \underline{\underline{80 \text{ percentile rank}}}$$

$\rightarrow$  indicates that 80% of entire distribution falls below the value 10.

Q2 What value exist at percentile 25.

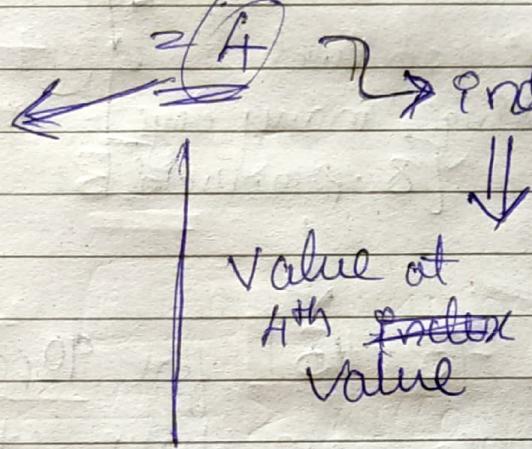
$$\Rightarrow \boxed{\text{Value} = \frac{\text{Percentile}}{100} \times (n+1)}$$

$$= \frac{25}{100} \times 16 \xrightarrow{n=15} \text{total no. of values.}$$

if  $4.2 / 4.5$

$4.7$

then average  $\left( \frac{4^{\text{th}} + 5^{\text{th}}}{2} \right)$

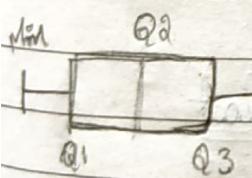


Q2. 4

### (\*) Quartiles

Q1  $\rightarrow$  25 percentile

[Median]



Q2  $\rightarrow$  50 percentile [Median]

Box plot

Q3  $\rightarrow$  75 percentile

E-commerce

E-commerce

Products

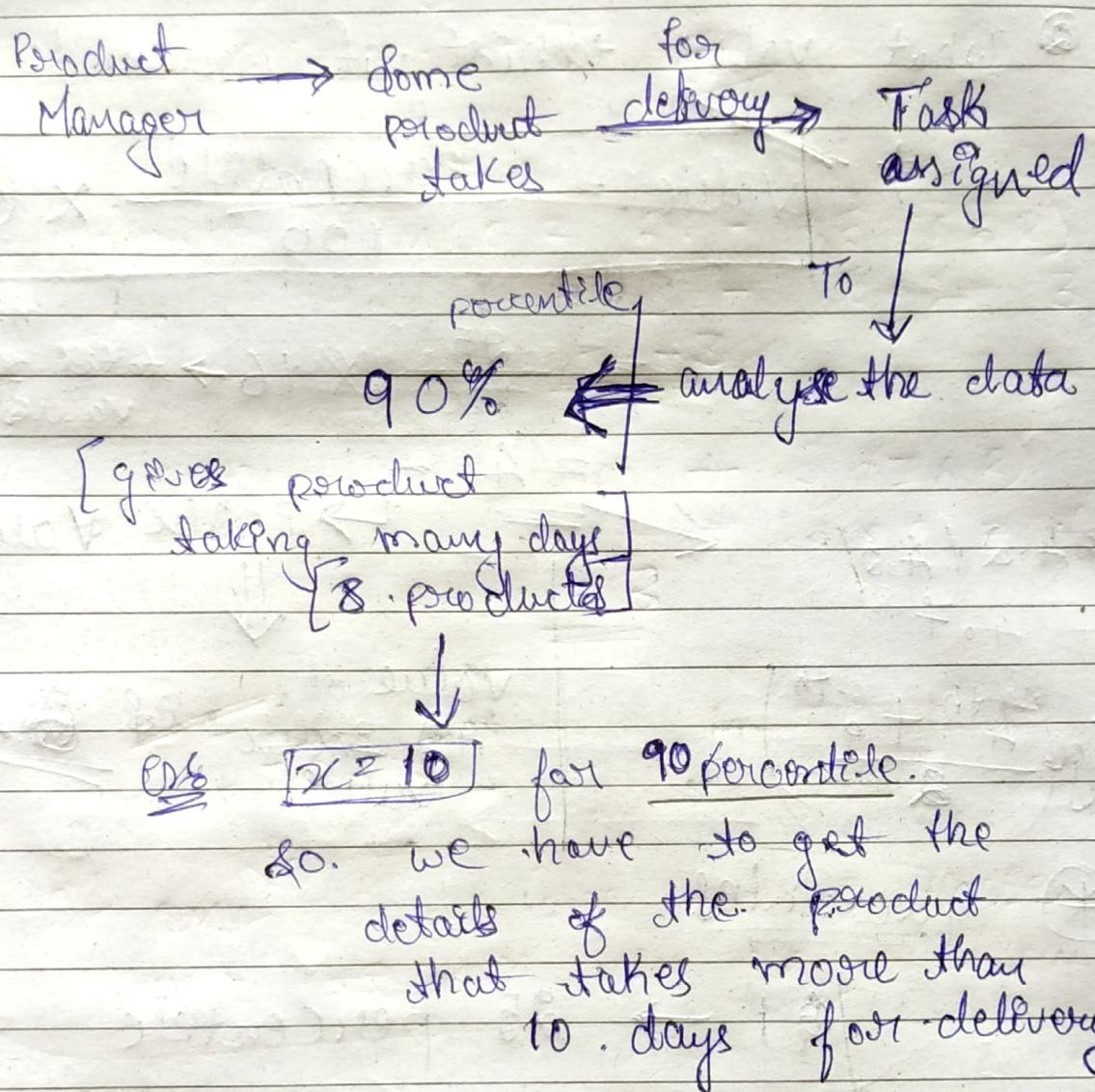
Delivery time

Location

=

2  
3  
1

=  
=



\* 5 number summary and Box plot  
 ↳ To find outliers.

- (i) Minimum
- (ii) First Quartile (25 percentile): Q1
- (iii) Second Quartile (50 percentile): Q2 (median)
- (iv) Third Quartile (75 percentile): Q3
- (v) Maximum

Removing the outlier.

Q3      20      Q3  
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17  
 $x = \{ 1, 2, 2, 2, 3, 3, 4, 5, 5, 5, 6, 6, 6, 6, 7, 8, 8,$   
 $9, 12, 19 \}$

→ this value is very big in dataset.

$$m = 19$$

[lower Fence  $\longleftrightarrow$  Higher Fence]

if any element

← lower fence

→ higher fence

are considered as 'outlier'

IQR  $\Rightarrow$  Inter Quartile Range.

$$Q3 - Q1$$

$$\text{Lower Fence} = Q1 - 1.5 \cdot (\text{IQR})$$

$$\text{Higher Fence} = Q3 + 1.5 \cdot (\text{IQR})$$

$$Q1 = 25 \text{ percentile} = \frac{25}{100} \times (19 + 1)$$

$$= 5$$

→ 5<sup>th</sup> ~~first~~ value

$$\boxed{Q1 = 3}$$

$$Q3 = 75 \text{ percentile} = \frac{75}{100} \times (19 + 1)$$

$$= 15$$

→ 15<sup>th</sup> ~~order~~ value

$$\boxed{Q3 = 7}$$

$$IQR = Q_3 - Q_1 = 7 - 3 = \underline{\underline{4}}$$

$$\therefore \text{lower fence} = Q_1 - 1.5(IQR) \\ = 3 - 1.5 \times 4 = 3 - 6 \\ = \underline{\underline{-3}}$$

$$\text{Upper fence} = Q_3 + 1.5(IQR) \\ = 7 + 1.5 \times 4 = 7 + 6 \\ = \underline{\underline{13}}$$

$$[-3, 13]$$

If ~~any~~ points come out of ~~these~~ range ~~then~~ then they are called outliers.

Outliers in our dataset ~~are~~ = 29 (as  $29 > 13$ )

\* 5 number summary [Box plot] (excluding outlier)

$$\text{Minimum} = 1$$

$$Q_1 = 3$$

$$(\text{median}) Q_2 = 5$$

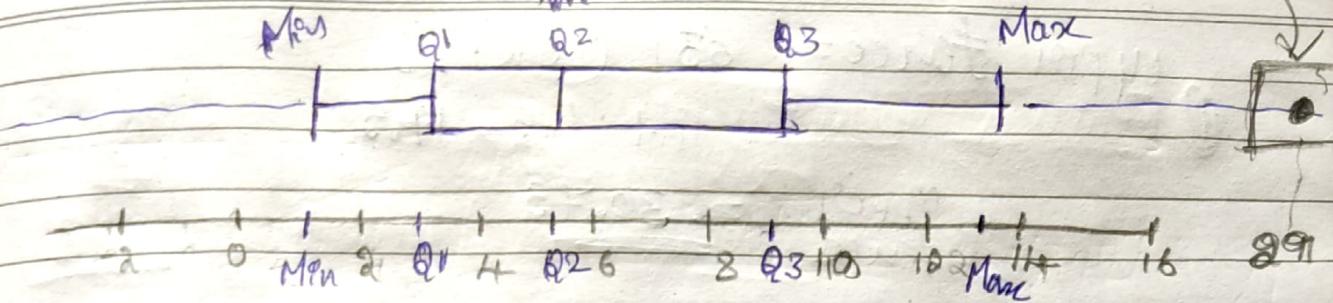
$$Q_3 = 7$$

$$\text{Maximum} = 9$$

$$IQR = 9.5 \approx 10^{\text{th}} \text{ element}$$

Outlier

median



### Assignment

$$\textcircled{1} \quad Y = \{-13, -18, -5, -6, 3, 4, 5, 6, 7, 7, 8, 10, 15, 55\}$$

$n = 14$

$$\Rightarrow Q_1 = 25^{\text{th}} \text{ percentile} = \frac{25}{100} \times (14 + 1)$$

$$= \underline{\underline{3.75}}$$

$$Q_1 = \frac{(3^{\text{rd}} + 4^{\text{th}})}{2} = \frac{(-5) + (-6)}{2} = \frac{-11}{2}$$

$$Q_1 = \underline{\underline{-5.5}}$$

$$\Rightarrow Q_3 = 75^{\text{th}} \text{ percentile} = \frac{75}{100} \times (14 + 1)$$

$$= \underline{\underline{11.25}}$$

$$Q_3 = \frac{11^{\text{th}} + 12^{\text{th}}}{2} = \frac{8 + 10}{2} = \underline{\underline{9}}$$

$$\text{IQR} = Q_3 - Q_1 = 9 - (-5.5)$$

$$= \underline{\underline{14.5}}$$

$$\text{lower fence} = Q_1 - [Q_3 + IQR] / 1.5$$

$$= -5.5 - \frac{14.5}{1.5} (14.5)$$

$$\text{lower fence} = \underline{\underline{-27.25}}$$

$$\text{upper fence} = Q_3 + (IQR)1.5$$

$$= 9 + 10.5 \times 14.5$$

$$\text{upper fence} = \underline{\underline{30.75}}$$

$$[-27.25, 30.75]$$

$\approx 5.5\%$  outliers

Box plot

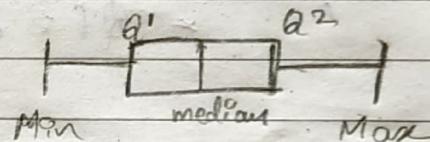
$$\text{Min} = -13$$

$$Q_1 = -5.5$$

$$Q_2 = 5.5$$

$$Q_3 = 9$$

$$\text{Max} = 51$$



Handling outliers

men, Q1, med, Q3, max = np.quantile (list, [0, 0.25, 0.5, 0.75, 1])

➢ plt.boxplot (list)

➢ sns.boxplot (list)

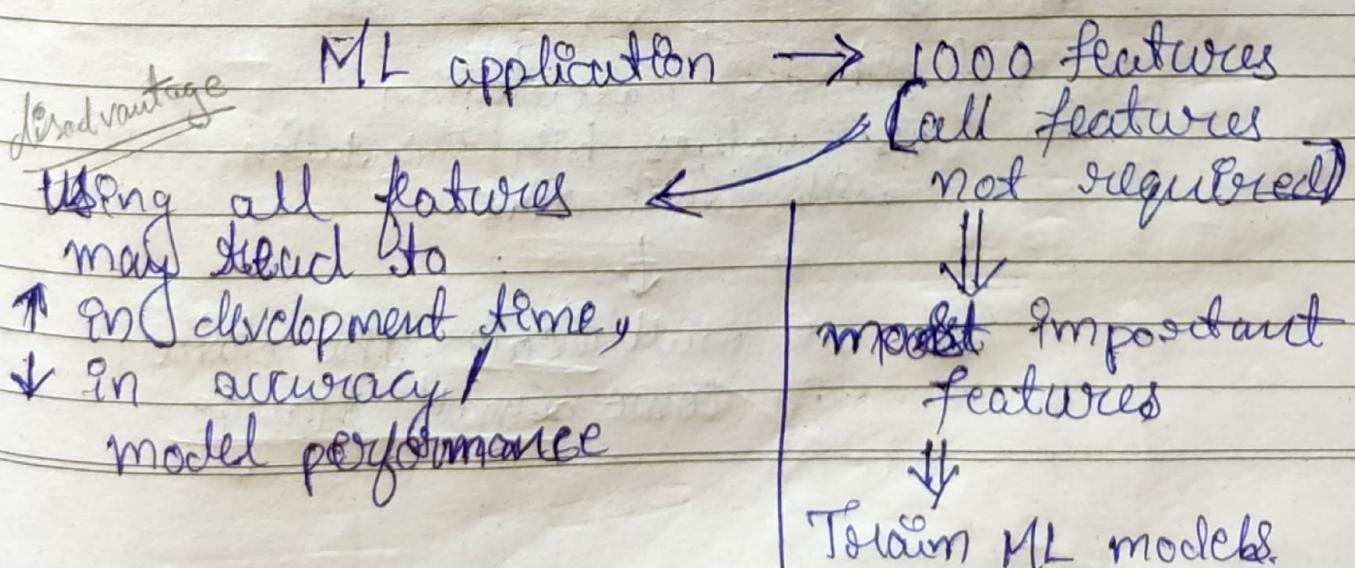
## Feature selection

- Filter method,
- Wrapper method,
- Embedded methods

Study after <sup>few</sup> ML algo

## Feature extraction

def: Feature extraction is a process of selecting and extracting the most important features from raw data.



## ① Feature Scaling

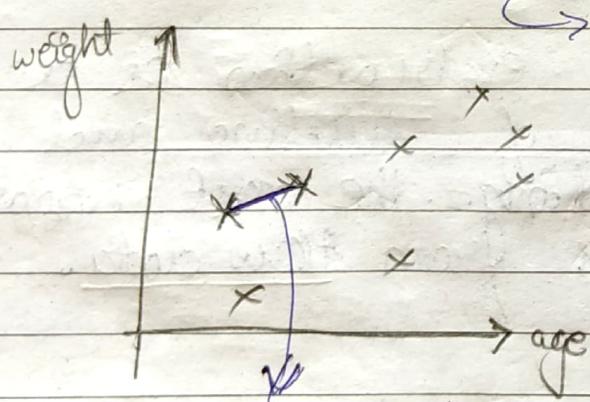
Age	weight	height	BMI
22	70	140 cm	
28	75	160 cm	
35	80	155 cm	

(years) (kg) (cms)

all features are calculated in different units

(KNN) →

- \* There are some ML algo which works on distance matrix.



distance calculation  
as values big Calculation  
will take time

So we will scale the data  
in some small range

ML technique

(i) Standardization  $\rightarrow [-3, +3]$

Z-score  $Z = \frac{x - \bar{x}}{\sigma}$

data (convert)

$$\left\{ \frac{32 - \bar{x}}{\sigma}, \frac{28 - \bar{x}}{\sigma} \right\}$$
$$\left\{ \frac{35 - \bar{x}}{\sigma} \right\}$$

standard normal distribution

$\text{SND} \rightarrow [\mu = 0, \sigma = 1]$

deep learning - image data

(ii) Normalization [Min-Max Scaler]

↳ transfer value b/w 0 and 1  
[0, 1]

~~Age~~

24

25

32

45

50

58

~~Age'~~  $\left\{ \begin{array}{l} \text{Min} \\ \text{Max} \\ \text{Scalor} \end{array} \right\}$

[0, 1] values

Transformation

0.0  
0.9

$$x_{\text{scaled}} = \frac{x_i - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

### (iii). Unit vector

↳ Magnitude of 1

$$\vec{x} = (3, 4)$$

Magnitude  
A/c to Pythagorean theorem,

$$\|\vec{x}\| = \sqrt{3^2 + 4^2}$$

$$= \underline{\underline{5}}$$

Unit vector,  $\hat{\mu} = \left( \frac{3}{\|\vec{x}\|}, \frac{4}{\|\vec{x}\|} \right)$

Unit vector  $\hat{\mu} = \underline{\underline{\frac{3}{5}, \frac{4}{5}}}$

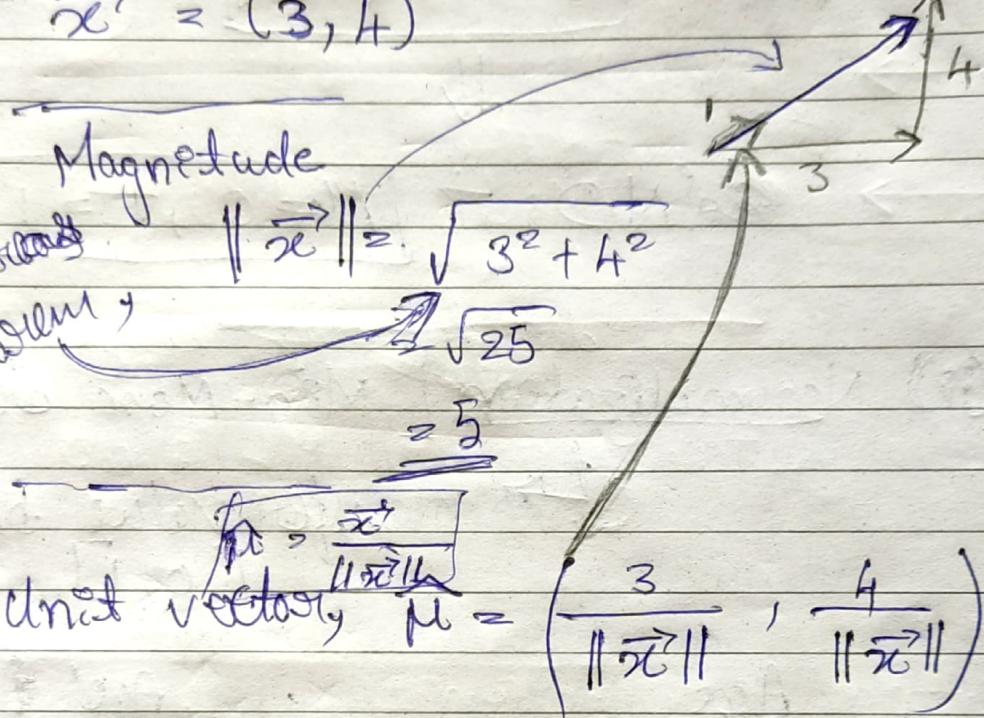
$$\|\hat{\mu}\| = \sqrt{\left(\frac{3}{5}\right)^2 + \left(\frac{4}{5}\right)^2}$$

$$= \sqrt{\frac{9}{25} + \frac{16}{25}}$$

$$= \sqrt{\frac{25}{25}} = \underline{\underline{1}}$$

Magnitude of unit vector is always  $\underline{\underline{1}}$

$\|\hat{\mu}\| = \underline{\underline{1}}$



## ② Feature selection:

→ we just pick the most important features.

500 features → take Top 10 features

↓ then

ML model Train.

To pick top 10 features...

### ③ Filter method:

~~Ex:~~ Correlation

### ④ Embedded method:

### ⑤ Wrapper method:

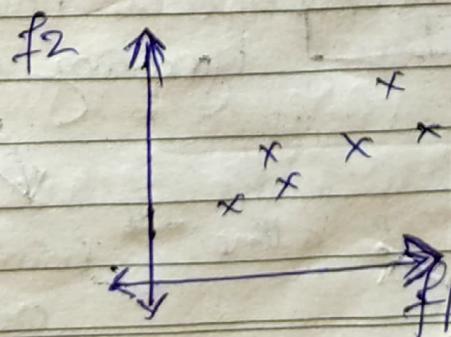
## ③ PCA [Principal Component Analysis]

part of unsupervised learning

( $f_1$   $f_2$ )

of

2 Dimension [2D]  
point.)



2D Convert → 1D  
using  
PCA

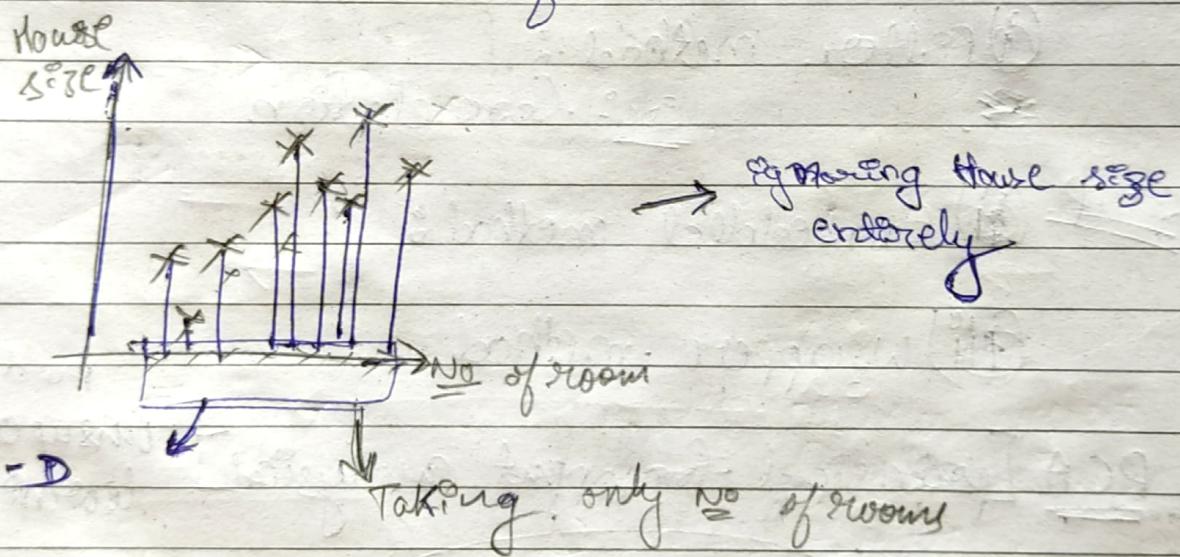
Here, there  
will be  
some loss of  
data

One Dataset independent feature

No of rooms	House size	Price
-	-	-
-	-	-
-	-	-
-	-	-

### Feature selection.

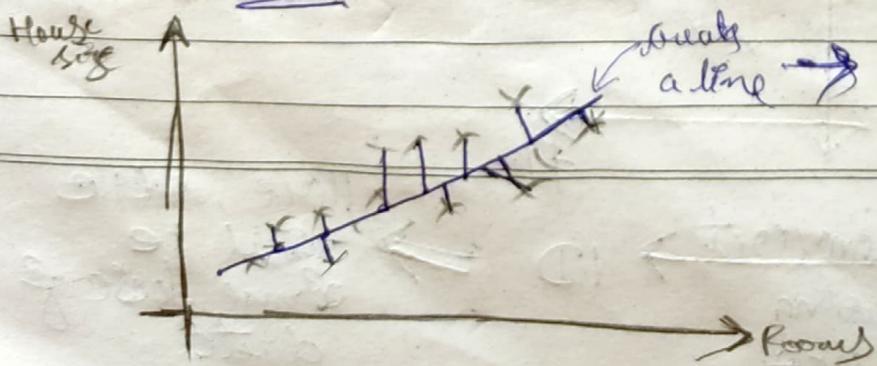
select top 1 feature



\* There will  
"loss of information"

Here, we are not  
ignoring a feature  
completely that  
is advantage

### But, in PCA



→ only "some  
loss of info"

1000 D  $\rightarrow$  2D

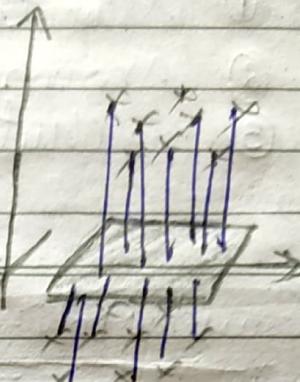
[Can be done]



we cannot  
points in ~~1000D~~  
only

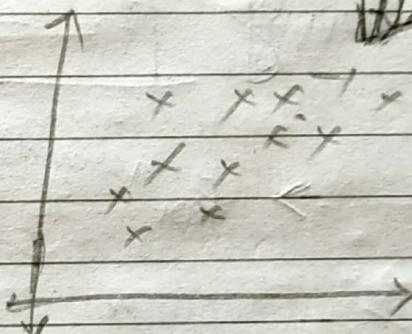
3D  $\rightarrow$  2D

3D



Projection

2D



\* Higher

dimension

$\xrightarrow{\text{to}}$

lower

dimension

## \* Feature Scaling

- ① Standardization
- ② Normalization - Min Max Scaler
- ③ Unit Vector

### ① Standardization

apply 
$$z = \frac{x - \mu}{\sigma}$$

np.mean()  
np.std()

> for  $x$  in test(df[ $\_$ ]):

$$z\text{-score} = (x - \text{mean}) / \text{std}$$

normalize.append(z-score)

from sklearn.preprocessing import StandardScaler

calculate  
rescaled  
values

scaler = StandardScaler()

> scaler.fit(df[['feature $_1$ ']])

form new  
data

transform  
the feature

(001)

> scaler.transform(df[[ $\_$ ]])

?scaler.fit\_transform(df[[ $\_$ ]])

## ② Normalization - MinMaxScaler

> from sklearn.preprocessing import  
    MinMaxScaler

> min\_max = MinMaxScaler()

> min\_max.fit\_transform(df[[-, -1]])

for any new data > min\_max.transform(df[[-, -1]])

## ③ Unit vector

> from sklearn.preprocessing import  
    normalize

> normalize(df[[-, -1]])

## Data Encoding

- ① Nominal / one hot encoding.
- ② Label and ordinal Encoding.
- ③ Target guided ordinal Encoding.

why? ex:

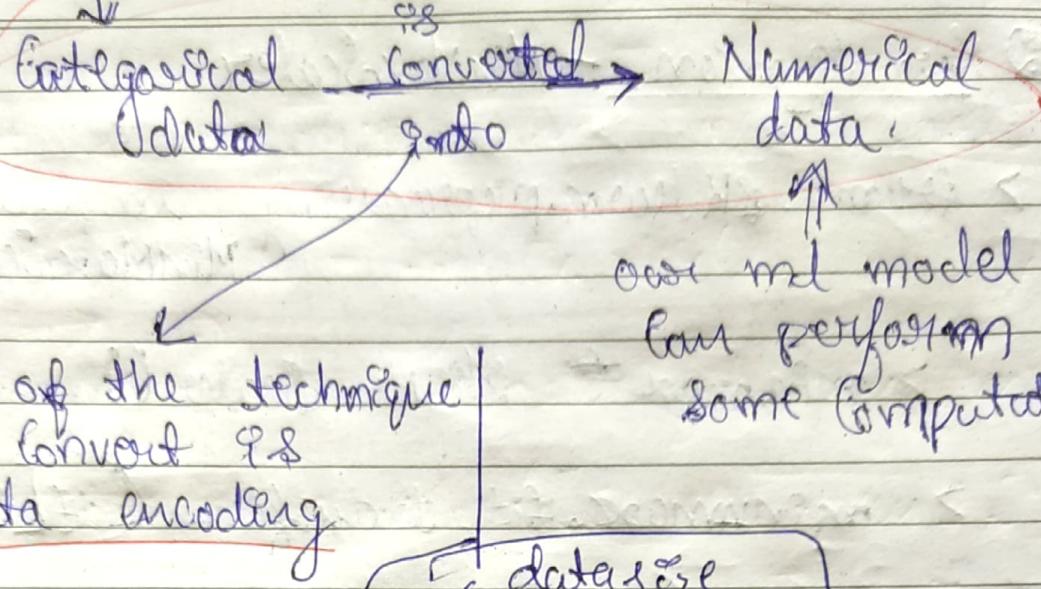
Exp.	Degree	Salary
B.E		ok.
B.Tech		
Master		

our model cannot understand this values but only numerical value can be understood

Categorical variables

Data encoding → Numerical Value  
Model can perform computation

our model cannot understand  
categorical data



our ml model  
can perform  
some computation

### ① Nominal encoding (OHE OneHot Encoder) encoding

$\Rightarrow$  one hot encoding (OHE) [Red green Blue]

represented as a binary vector	Ex:	Color	Red	green	Blue
Categorical		Red	[1 0 0]		
		Green		[0 1 0]	
		Blue			[0 0 1]

### \* disadvantage

① if we have 1000 features  
then 1000 new features are  
created.

② Sparse matrix  $\Rightarrow$  overfitting  
[low bias]  
[high variance]

## Scikit learn

> imports

from sklearn.preprocessing import

OneHotEncoder

> encoder = OneHotEncoder()

Encoded:

> encoder.fit\_transform([df['color']]).

alphabet | blue | green | red | toarray()

>

Encoded\_df = pd.DataFrame(encoder.fit\_transform([df['color']]),  
columns=encoder.get\_feature\_names\_out())

New  
data >

encoder.transform([[ 'blue']]).toarray()

→ [ 0. 1. 0 0 ]

> final\_df = pd.concat([df, encoded])

② Labels (or) cardinal encoding:

↳ Assigning a unique label to each category in the variables

may be based on according  
alphabet (or) frequency

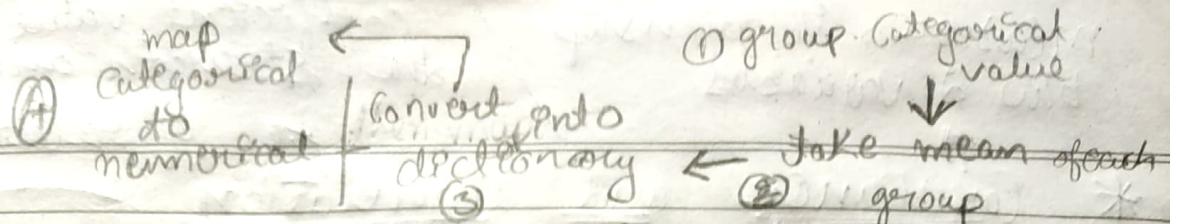
- > `from sklearn.preprocessing import LabelEncoder`
  - > ~~encoder~~ encoder = LabelEncoder()
  - > encoded = encoder.fit\_transform(`df['L-1']`)
  - > encoder.transform([`['red']`, `['green']`, `['blue']`])
- different values for 1 feature
- $\begin{bmatrix} [1 \rightarrow], [2 \rightarrow], [3 \rightarrow] \end{bmatrix}$
- $\begin{bmatrix} [1 \rightarrow], [2 \rightarrow], [3 \rightarrow] \end{bmatrix}$
- multiple values for 1 feature.

### ③ Cardinal encoding



Encoder ~~data~~ data that have interconsec. order (ex) ranking.

- > `from sklearn.preprocessing import CardinalEncoder`
  - > `card-Encoder = CardinalEncoder(`
  - > `Categories = ['small', 'medium', 'large']`
  - > tell the order of ranking to be given
  - > `card-Encoder.fit_transform(df['size'])`
  - >
- "large is great" so highest rank



### ④ Target guided ordinal encoding:

Encode based on their relationship with the target variable.

→ useful when: large number of unique categories.

→ Categories in categorical variable → replace categorical variable by numerical value based on mean/median of target variable

→ Create: monotonic relationship.  
→ improves predictive power.

>  $mp = df.groupby(['city'])['price'].mean().to_dict()$

City	feature
london	150.0
newyork	190.0

>  $df['city\_encod'] = df['city'].map(mp)$

>

measures relationship  
b/w 2 variables

coefficient value tells  
the strength and direction  
of relationship

## \* Covariance and Correlation

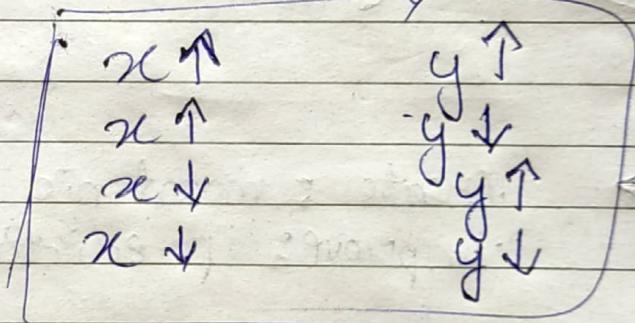
→ imp EDA & Feature engineering.

x	y
2	3
4	5
6	7
8	9

### ① Covariance

degree of association

{ relationship between }  
x and y



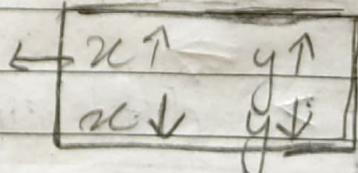
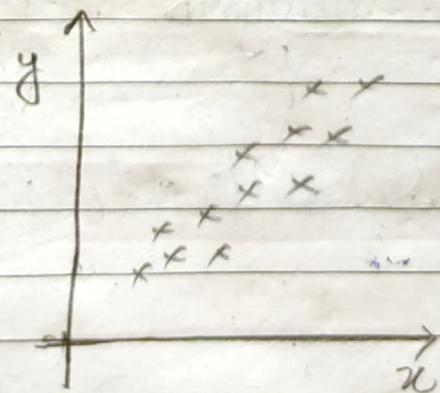
size

location

price

o/p

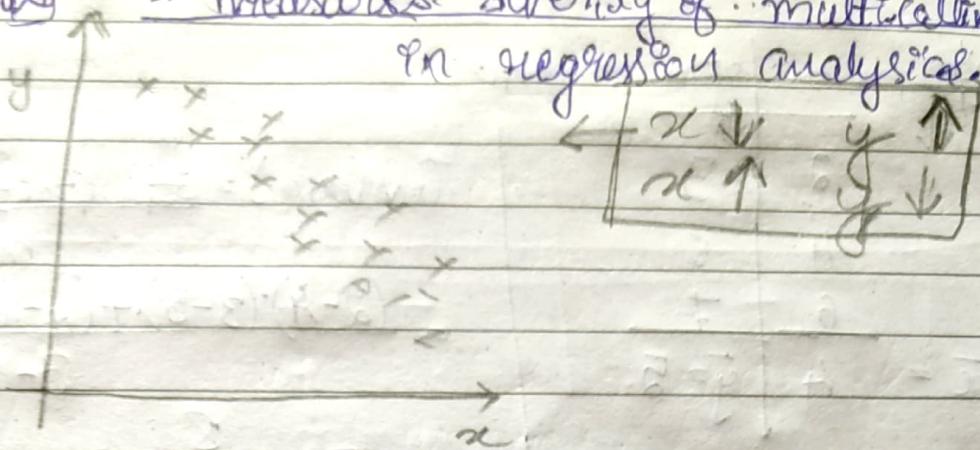
% [Independent]



VIF  $\rightarrow$  Variance inflation factor. ( $VIF \rightarrow$   $\uparrow$  in level of multicollinearity)

$VIF = 1 \rightarrow$  no multicollinearity

measures severity of multicollinearity in regression analysis.



Covariance. ( $x, y$ )

$$(05) \text{ Cov}(x, y) = \sum_{i=1}^n \frac{(x_i - \bar{x})(y_i - \bar{y})}{(n-1)}$$

Variance ( $x$ )

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

Variance tells the spread of data.

Variance ( $x$ ) = covariance ( $x, x$ )

how  $x$  is related to itself.

$$\begin{bmatrix} x \uparrow & y \uparrow \\ x \downarrow & y \downarrow \end{bmatrix} \Rightarrow +ve \text{ cov}$$

tells that  $x$  and  $y$  are positively correlated.

$$\begin{bmatrix} x \uparrow & y \downarrow \\ x \downarrow & y \uparrow \end{bmatrix} \Rightarrow -ve \text{ cov}$$

$\bar{x}$	$y$	$n=3$
2	3	
4	5	
6	7	
4	5	$\bar{y}=5$

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

$$= \frac{(2-4)(3-5) + (4-4)(5-5) + (6-4)(7-5)}{3-1}$$

$$= \frac{8}{2}$$

$$\text{Cov}(x,y) = 4 \quad [\text{+ve}]$$

↓  
we can tell that  $x$  and  $y$  are positively correlated

### Advantage

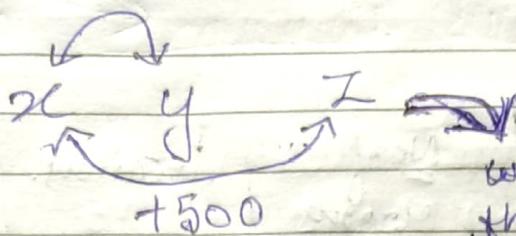
- ① Relationship b/w  $x$  and  $y$   
may be +ve/-ve

### Disadvantage

- ② we don't have a limit/range for the values of covariance.

↓  
To overcome [range problem]  
we have Spearman's correlation  
and Pearson's correlation coefficient  
(range → [-1, +1])

+100



## ② Pearson Correlation coefficient [-1 to +1]

$$\rho_{(x,y)} = \frac{\text{Cov}(x,y)}{\sigma_x \sigma_y}$$

standard deviation

Range  $\Rightarrow [-1, +1]$

- \* The more the value towards +ve the more positively correlated they are.
- \* The more the value towards -ve the more negatively correlated they are.

use case:

Dataset: 1000 features [ML models]

(Taking all feature and training is not so important.)

<u>Ex:</u>	Haunted house	size of the house	No of rooms	location.	No of people staying
	-	-	-	-	-

To predict price  
No of people staying  
 will not help      not correlated to price

So, we do Feature selection  
 where we select only features  
 which are "highly correlated"  
 to ("price") output.  
 and  
 less correlated can be dropped.  
 Feature.

③ Spearman rank correlation → better than Pearson

↳ we consider rank

$$\rho_s = \frac{\text{Cov}(R(x), R(y))}{\sqrt{R(x)} \sqrt{R(y)}}$$

<u>Op Price</u>	<u>x</u>	<u>y</u>	<u>R(x)</u>	<u>R(y)</u>
-	1	3	5	5
-	3	4	4	4
-	5	6	3	3
-	7	8	2	1
-	0	7	6.	2
-	8	1	1	6

Highest

Rank 1 → assigned to highest  
absolute value

\* python → Covariance and Correlation

> df.cov()

> df.covr(method = 'pearson')

> df.covr(method = 'spearmann')

diagonal  
elements will  
always be 1.