KNN and PCA Notes

- Darshan R M

"Learn, Share, and Collaborate"

Darshan R.M.

Week - 8

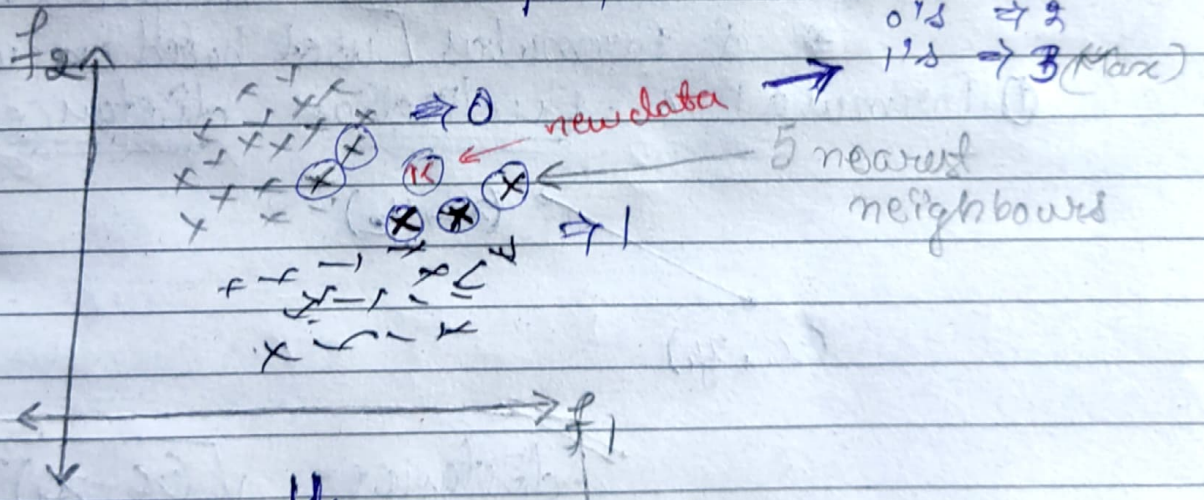
KNN and PCA

* K Nearest Neighbour [KNN]
  ↳ Can solve both classification and regression problem.

① classification:

* Consider,

f_1	f_2	y [Binary classification]
-	-	0
-	-	1



Training data

steps

① Initialize K value.
K > 0 ——— ∞
K = 1, 2, 3, 4, 5, ——— ∞
hyper parameter

Steps

① Initialize K value ← hyper parameter
Consider, K=5

② Find K Nearest Neighbours for the test data

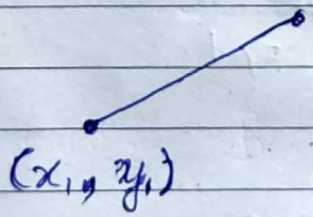
③ From those K=5, how many neighbours belong to 0 Category and 1 Category.

Max no of nearest neighboured Category is answer.

↓ But we need some distance value to be find to find nearest neighbours.

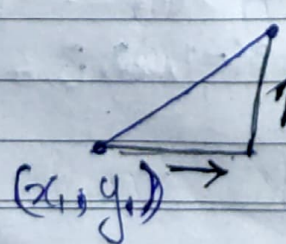
⇒ 2 Formulas [used based on usecase]

① Formula 1 ⇒ Euclidean distance.



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

② Formula 2 ⇒ Manhattan distance

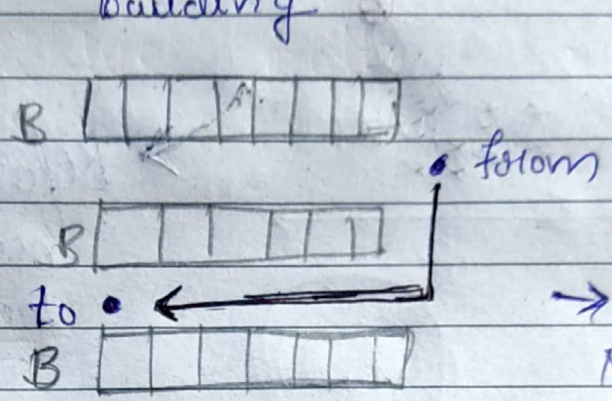


Calculate from sides not from hypothenuse

* where is this distance metrics is used?

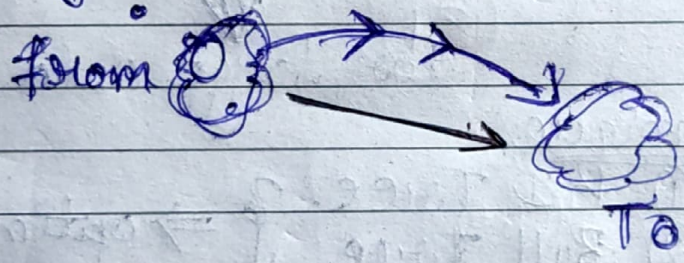
⇒ Ex:

① In america.



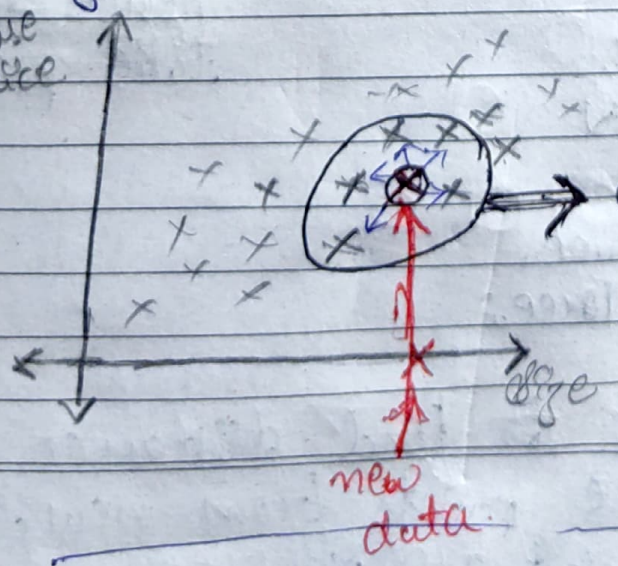
→ Can use Manhattan

② Flight.



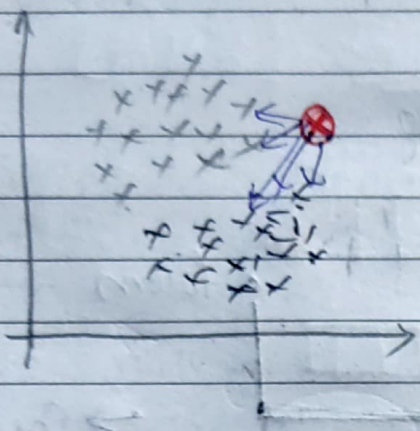
→ Can use Euclied an.

② Regression:



→ average of all the points to find out the o/p.

* we have to understand the variance.



Time Complexity ⇒ O(n)

Before selecting top5 nearest points do we have to calculate all the points distance from new points

⇓

For large data TC ↑↑

Technique.

① KD Tree. } ⇒ optimizer.
② Ball Tree. }

⇓

Creats Binary Tree → Search time decrease Hence, optimization takes place.

* Variants of KNN

① KD Tree.
② Ball Tree.

problems: Need to find distance of all the points from new data points to select top [K=5] points. T.C ⇒ O(n)

We have to ↓↓

① KD Tree

Consider,

f_1	f_2
7	2
5	4
9	6
2	3
4	7
8	1

$f_1 \Rightarrow$ 2, 4, [5], 7, 8, 9

$f_2 \Rightarrow$ 1, 2, [3, 4], 6, 7

$\frac{3+4}{2} = \frac{7}{2} = 3.5$

$\frac{7+5}{2} = 6$

① median $f_1 = 6$

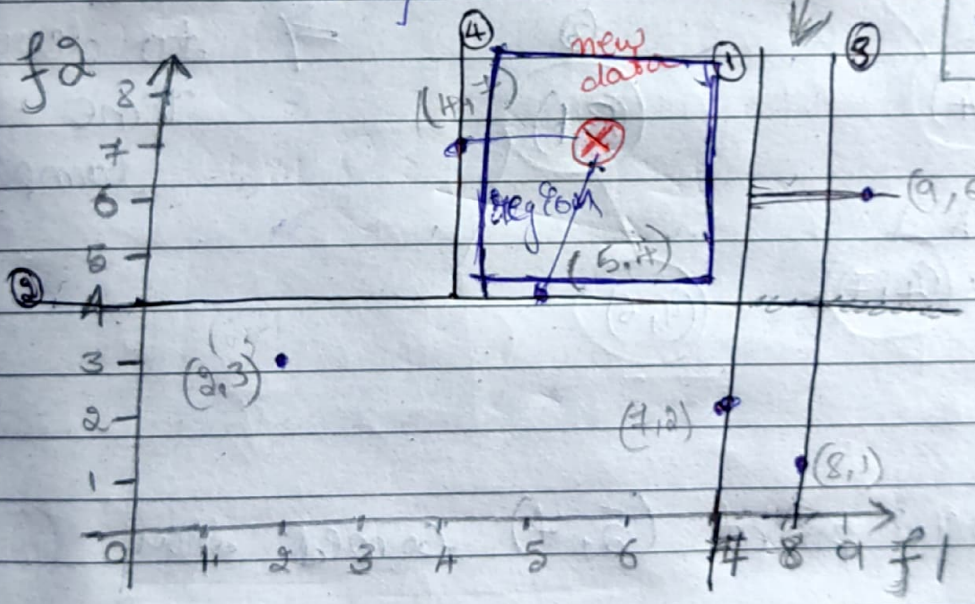
② median $f_2 = 3.5$

we can take nearest value and draw a line.

Consider f_1 as root

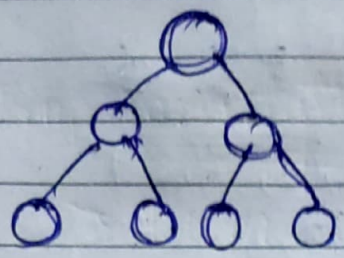
③ median for 8, 9 ⇒ consider 8

Continue this



Aim: we have to find the way to avoid checking distance for each and every point.

To do this we form a "KD Tree" which is in form of binary Tree!"

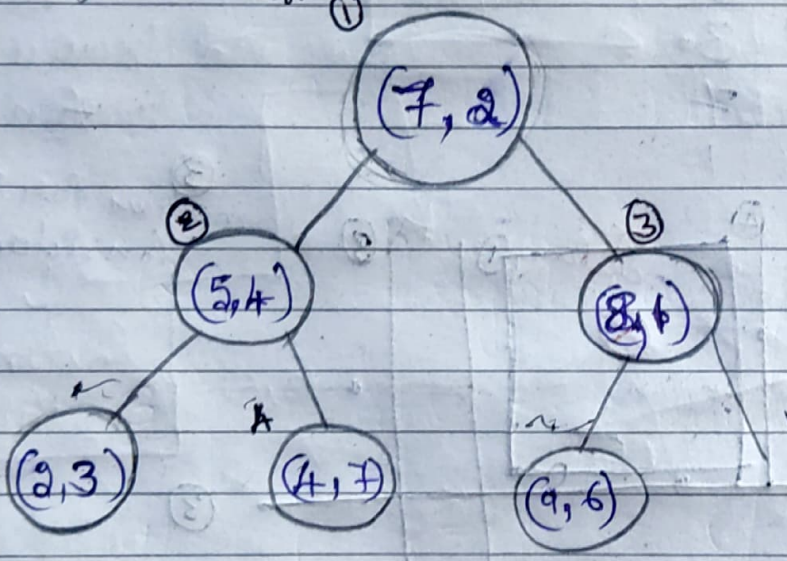


→ we can easily go and find the path because at each level data is divided into half.

⇓

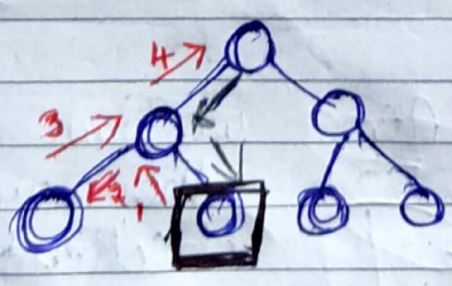
split are done based on the "median"

refer graph below ↖ split



⇒ Done due to decrease time complexity

* if new point (5, 7) comes then nearest

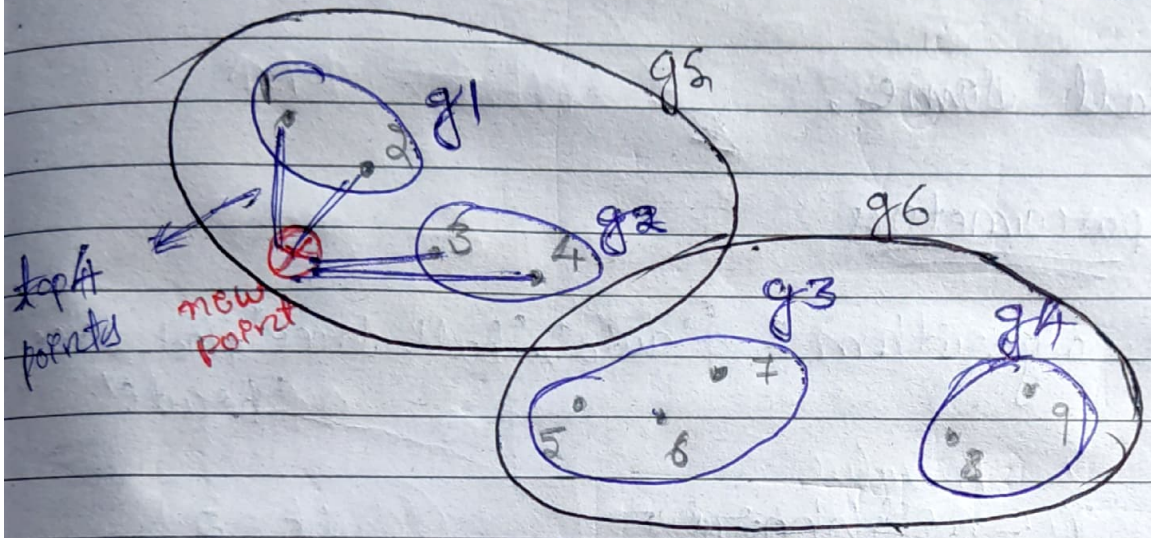


For nearest point → traverse back

[Backtracing is used]

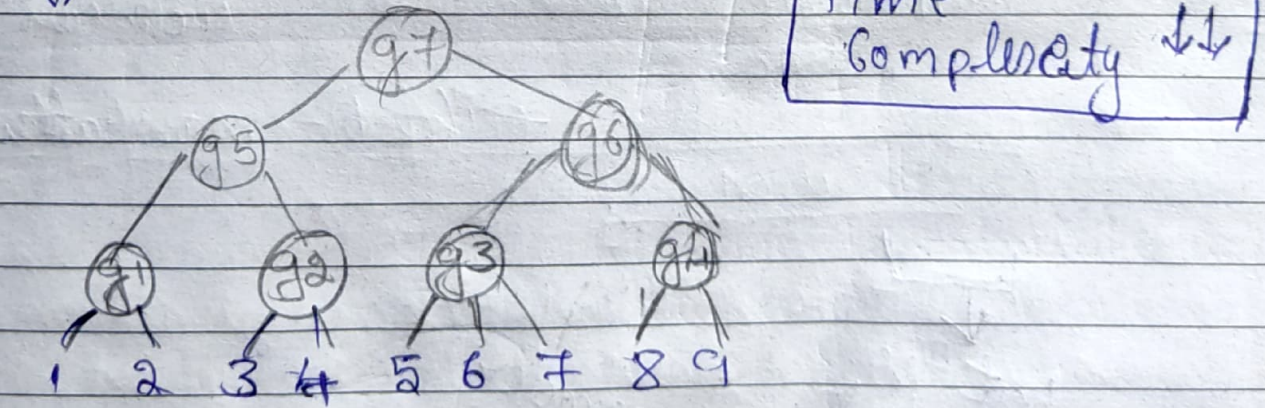
② Ball Tree :

↳ better than KD tree.
[as back tracking is used in KD Tree]



Data

① Group the nearest items.



Time Complexity ↓↓

② group the nearest groups.

* Implementation

> from sklearn.neighbors import KNeighborsRegressor / KNeighborsClassifier

* rest all same..

parameters

① algorithm : {'auto', 'ball_tree', 'kd_tree', 'brute'}

② n_neighbors = ___ (default = 5)
K value

③ weights = {'uniform', 'distance'}

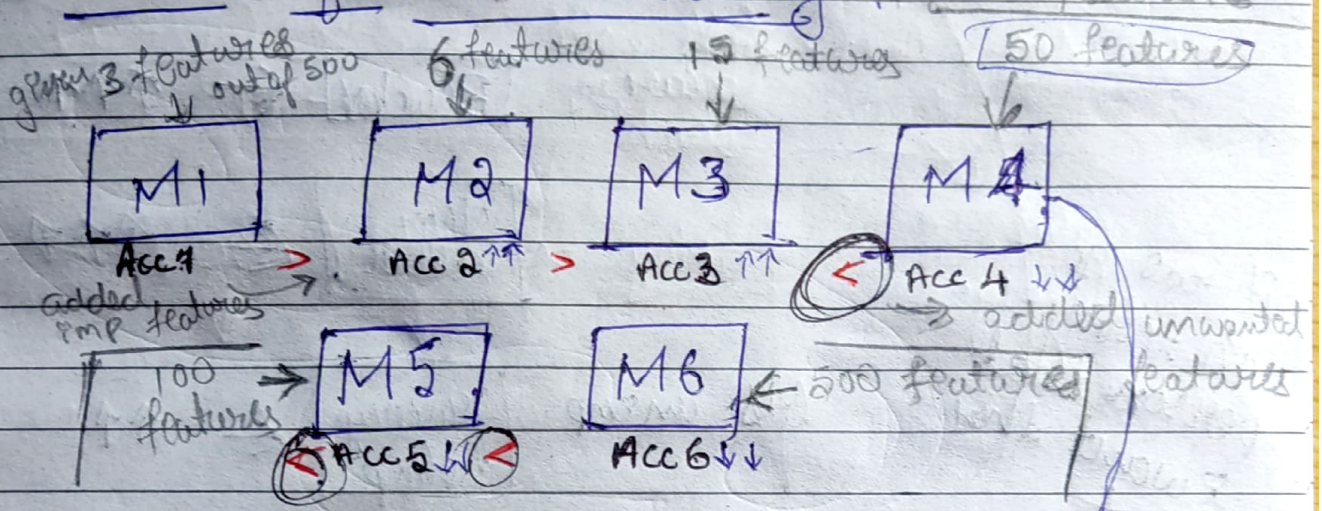
④ p = ___ (default = 2)
- p → Minkowski metric
- euclidean distance
- * if (p=1) → manhattan distance

best is selected using hyperparameter tuning

* Principal Component analysis [PCA]

[Dimensionality Reduction]

① Curse of Dimensionality:



Dataset : 500 features

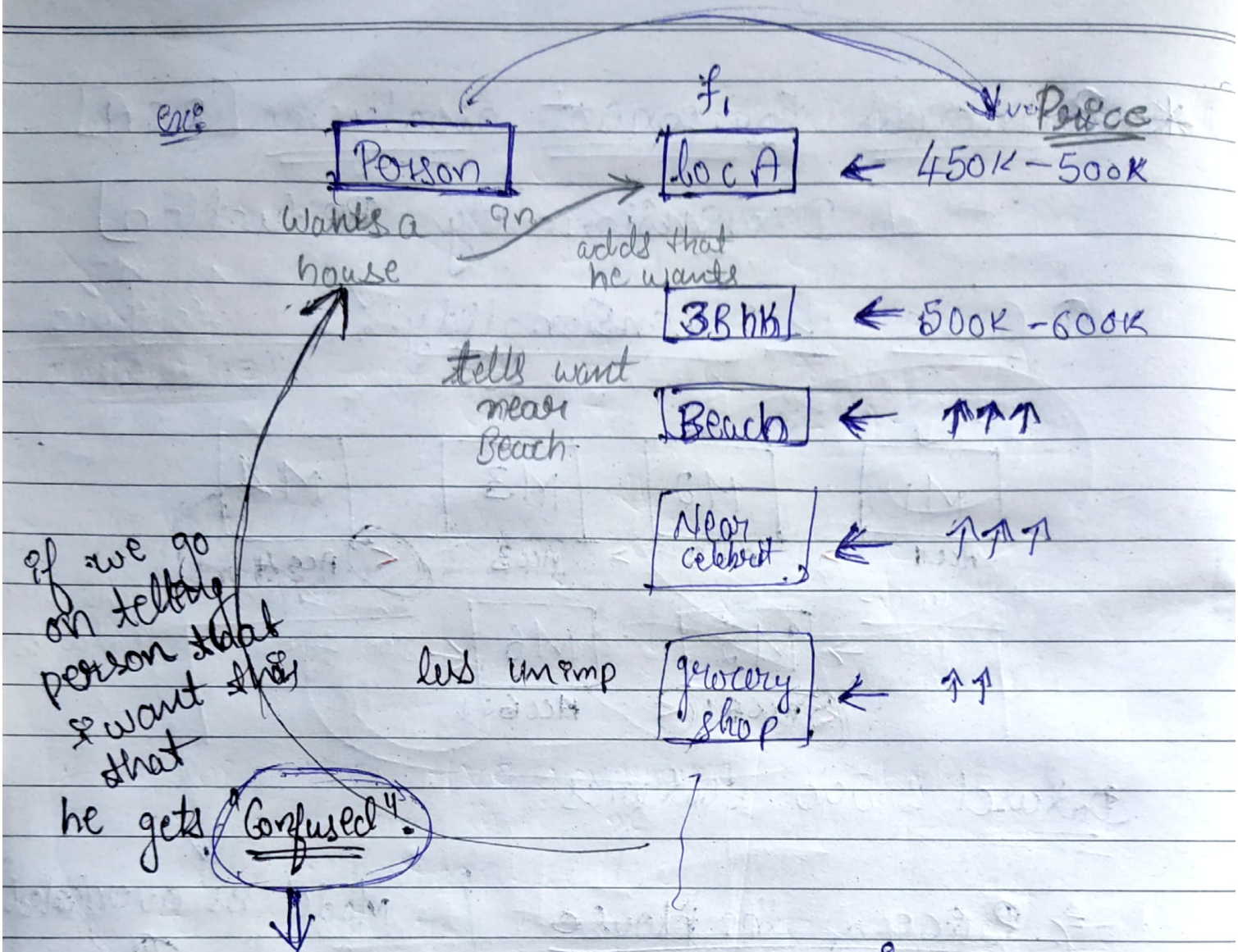
⇒ Price of the House

House size
No of bedrooms
No of bathrooms

Model is overfitted

As our model will learn from feature if we give any feature it will learn. so even if we give unwanted feature.

② Model performance degrade



* Two different ways to remove curse of dimensionality

① Feature selection. ⇒ Take imp features and train model

② ~~Feature~~ dimensionality reduction.

⇓

Feature extraction

f_1 f_2 f_3 $\rightarrow$ O/P

extract ⇓

D_1 D_2 $\rightarrow$ O/P

lesser dimension

* Feature selection v/s Feature extraction

↳ Dimensionality Reduction

* why dimensionality reduction?

⇒ Prevent → Curse of dimensionality.

* Improve performance of model. ↓ more dimension the formula will big

* Visualize the data ↓

⇓ ↳ understand the data

Max we can see '3d'.

* (Feature selection)

↳ used to select imp feature which help in o/p prediction.

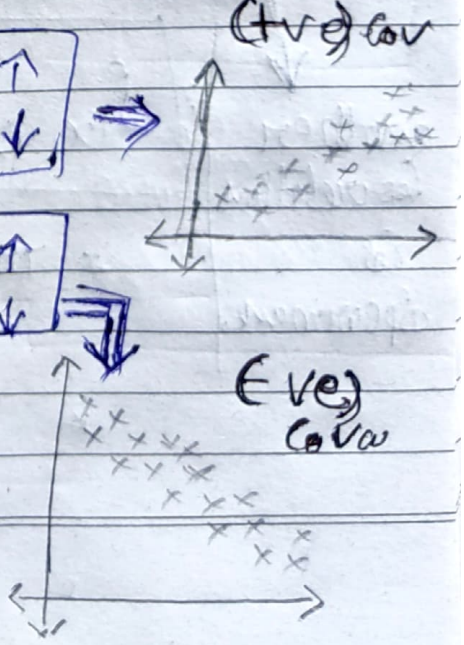
Ex:

x	y
-	-
-	-
-	-
-	-
-	-

i/p | o/p

if $x\uparrow$ $y\uparrow$ / $x\downarrow$ $y\downarrow$ ⇒ (+ve) cov

$x\downarrow$ $y\uparrow$ / $x\uparrow$ $y\downarrow$ ⇒ (-ve) cov



if there is a linear relationship it means that our 'x' is imp feature to predict 'y'.

* We can quantify the relationship mathematically.

⇒ ~~Cov(x,y) = $\sum_{i=1}^{n}$ ... y~~

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

⇓

May be +ve / -ve. / 0 → no relationship

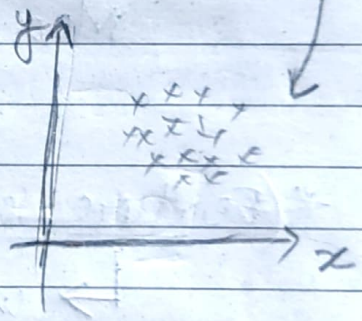
+ve → linear relationship

-ve → inverse linear relationship

no value range

so we use

* If we get +ve covariance we can say that x is very imp. feature to predict y.



* Pearson Correlation

another correlation is Spearman.

$\rho_{(x,y)} = \frac{Cov(x, y)}{\sigma_x \sigma_y}$ o/p ⇒ [-1, 1]

* the more towards the value of +1 the more +ve correlated it is.

* — 11 — (-ve) then — 11 — (-ve).

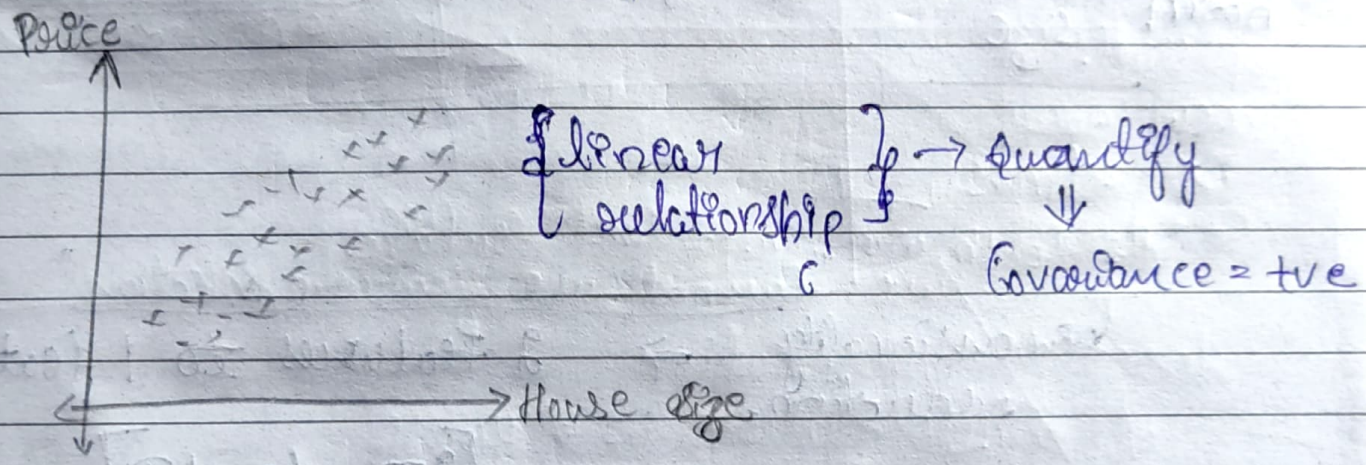
* if value = 0 then no relationship.

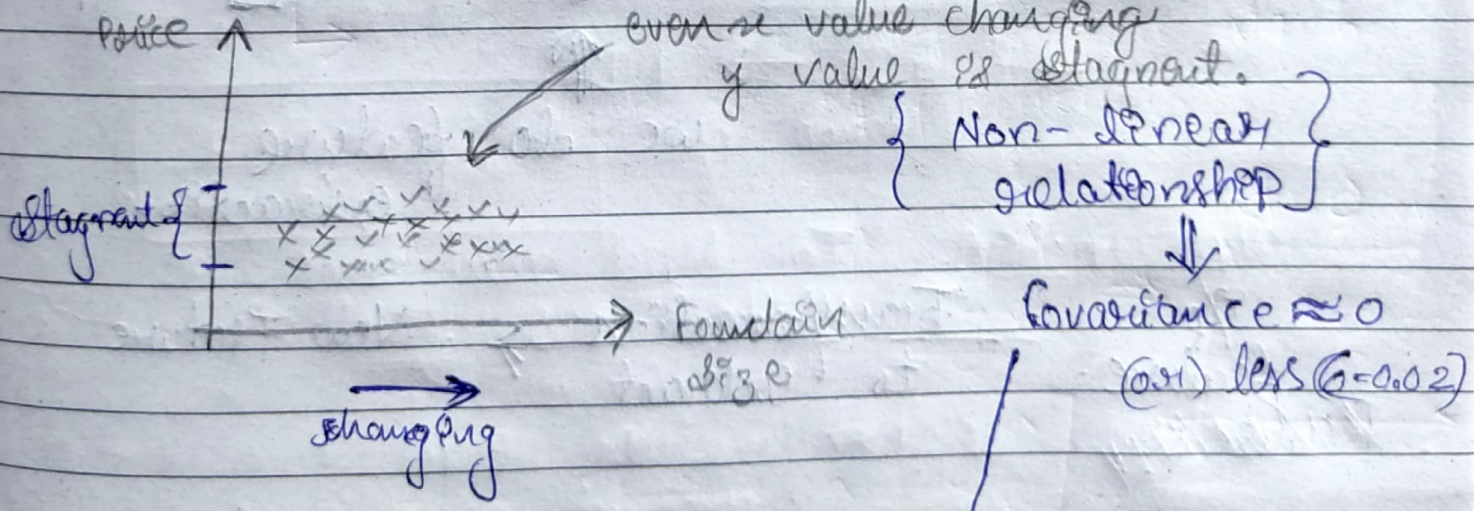
Common sense that Fountain donot affect Price.

* Consider housing dataset

Independent

House size	Fountain size	o/p feature Price
—	—	—
—	—	—
—	—	—



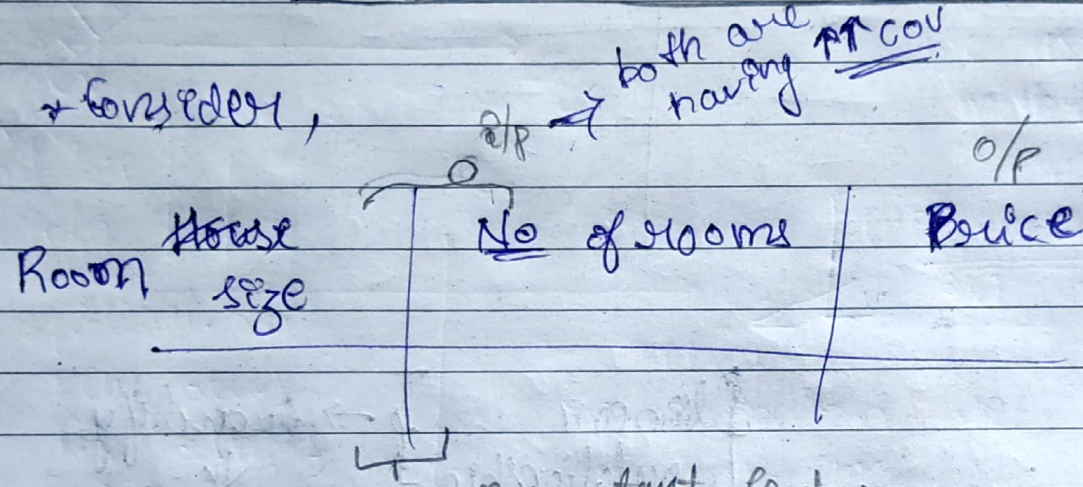


As it's a non important feature and we can drop that feature.

* (Feature Extraction:)

↳ we are extracting new feature from the existing feature. Hence, we reduce no dimensions.

* Consider,



As both are important feature we cannot use feature selection

dimensionality reduction. ⇒ 2 features to 1 feature.

[2D to 1D]

↑

we do feature extraction

i/p feature. → Transformation to extract → New feature

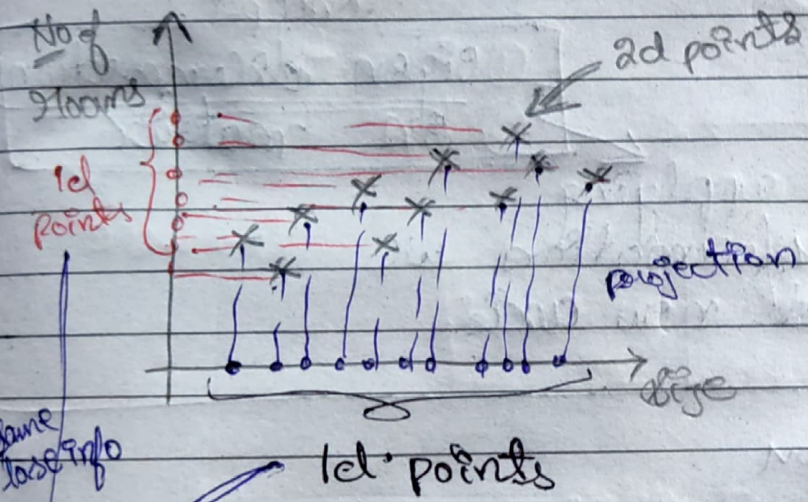
*

*(PCA) By Geometric Intuition

↳ used for dimensionality reduction.

→ Consider

size house	No of rooms	o/p price
i/p		



using PCD we need to reduce from [2d → 1d]

① Feature selection,
② Feature extraction.

area b/w first and last data point is a 'spread.' → spread ↑ then Variance ↑

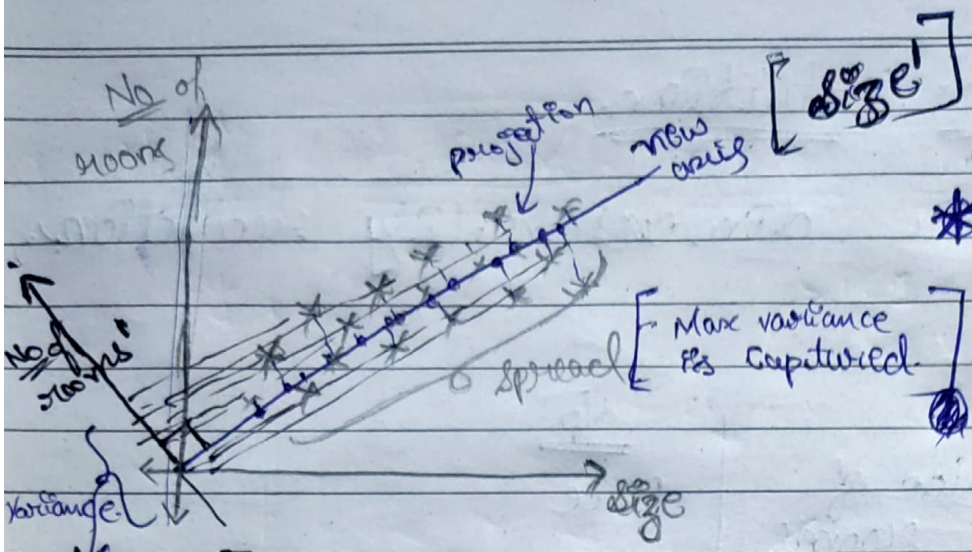
disadvantage of this approach where directly project is that

[lost of information] in this direct projection.

we losing No of rooms info we getting only size info.

→ How to avoid ?

⇒ PCA



* we do some transformation on x and y axis.

⇓

we use some mathematical concept.

eigen decomposition on matrix

after applying transformation we get new axis.

small loss of info

spread on new axis is almost correct

+

loss of information of other feature is very less.

* 2D → 1D
[much info is not lost]

aim in PCA: To find new axis line called as 'PC1' & 'PC2' ↪ other new axis
Max variance

PC1 ⇒ Capture large amount of variance → why as it tells spread/ capture of data.

PC2 ⇒ next large amount of variance

if more dimensions then it continues

* If 2 Dimension we get ↓

PC1, PC2

⇓

Var(PC1) > Var(PC2)

* if 3 dimension we get ↓

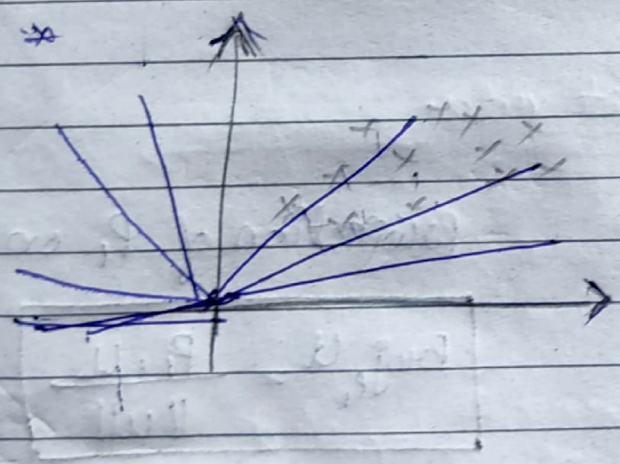
PC1, PC2 & PC3

⇓

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

---

*



2D → 1D

⇒ PCA will find out 2 best principal Component lines

which captures max variance.

Goal: To get the best principal Component ~~line~~ which Captures maximum variance.

* Ex. 3D

⇓

PC1, PC2, PC3

↓

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

⟹

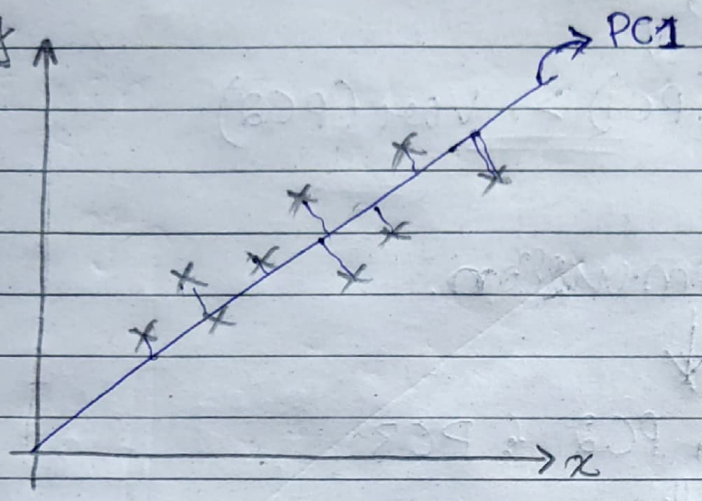
3D → 1D

↳ Take PC1

3D → 2D

↳ Take PC1 & PC2

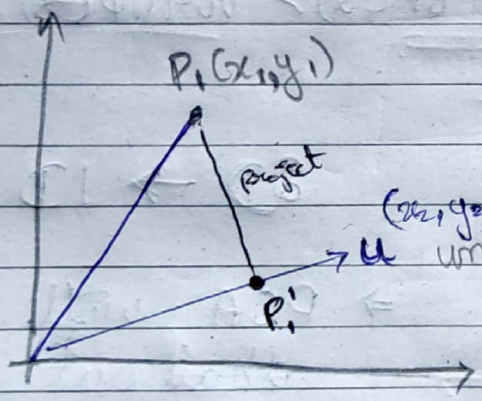
* Math intuition behind PCA algorithm



2D → 1D

[Max variance Captured]

① Projection.
② Cost Function ↓
Variance



Projection of P_1 on u

$$\text{Proj}_{P_1} u = \frac{P_1 \cdot u}{\|u\|}$$

∴ For unit vector, $\|u\| = 1$

$$\Rightarrow \text{Proj}_{P_1} u = P_1 \cdot u$$

→ Scalar value. [only magnitude]

Dot product of P_1 and u is P_1'

$\Rightarrow$ P_1'

For other points projection.

$[P_0', P_1', P_2', P_3', P_4', \dots\dots P_n']$ — P_n' tells about distance from origin.

⇓

Scalar value

⇓

easy to Calculate Variance.

* $P_0', P_1', P_2', P_3', \dots, P_n'$

⇓ notation change to understand

$x_0', x_1', x_2', \dots\dots, x_n'$

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

Maximize ←

{ goal: To find the best unit vector which captures max variance? }

⇓

→ Cost function.

To Find

Eigen decomposition

* Eigen vectors and eigen values.

① Covariance matrix b/w features.

② Eigen vectors and eigen values will found out from this Covariance matrix

③ Eigen vector → eigen value.

[Magnitude of Eigen vector]

↓

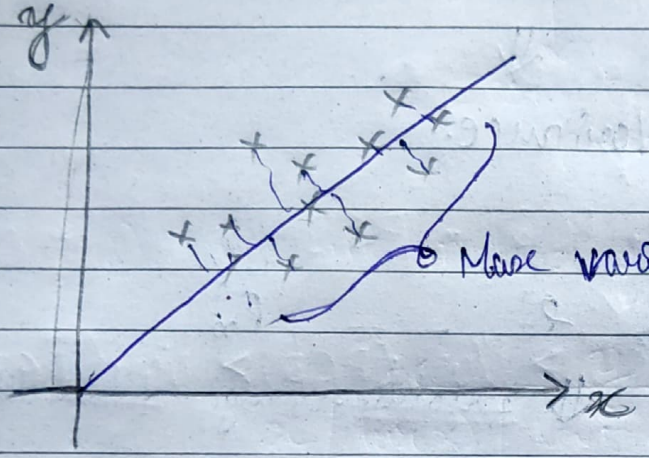
Capture the maximum Variance

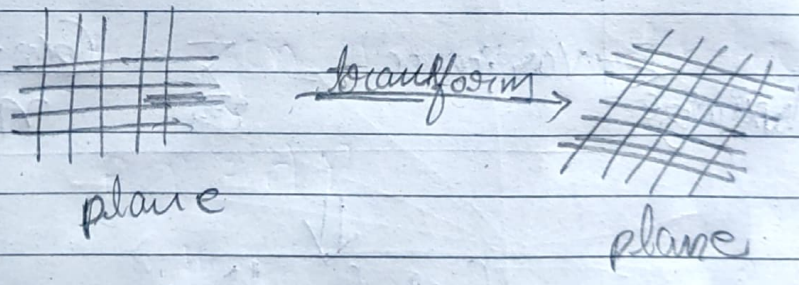
we find Eigen values & vector → Maximum magnitude ⇒ Principal component analysis ⇒ Max Variance

$$Av = \lambda v$$ {linear Transformation of matrix}

Eigen decomposition of variance matrix

we find Eigen values and vectors





* Eigen vector ⇒ Max magnitude ⇒ Max eigen vector

↓

Best principal component line ⇒ PC1

Steps to Calculate the eigen value and eigen vectors

(1) Covariance of features

(x, y) → x'

Z

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

$$Cov(x, x) = Var(x)$$

Consider, A

	x	y
x	Var(x)	Cov(x, y)
y	Cov(x, y)	Var(y)

	x	y	z
x	Var(x)	Cov(x, y)	Cov(x, z)
y	Cov(x, y)	Var(y)	Cov(y, z)
z	Cov(x, z)	Cov(y, z)	Var(z)

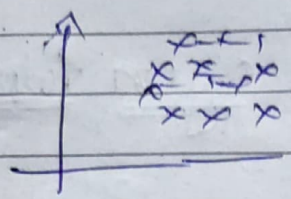
f1 & f2

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

eigen values: $\lambda_1 > \lambda_2$

λ_1 → PC1, λ_2 → PC2

Revise

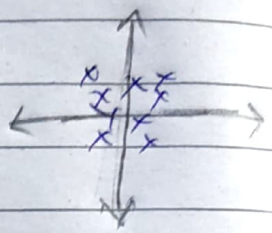


2D → 1D

① Standardize the data.

↓

Zero centered points.

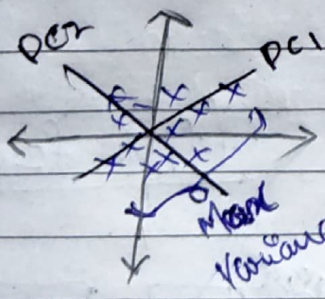


② Covariance matrix of x and y.

	x	y
x	Cov(x)	Cov(x,y)
y	Cov(y,x)	Cov(x)

③ Find out Eigen vectors and values

$$A v = \lambda v$$



λ_1 , λ_2 ⟹ eigen values

↓ PC1 ↓ PC2

If 3D → 2D

λ_1 , λ_2 , λ_3

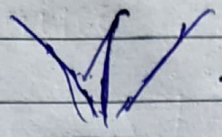
↓ PC1 ↓ PC2 ↓ PC3

1D 1D ⇒ 2D

if 3D → 1D

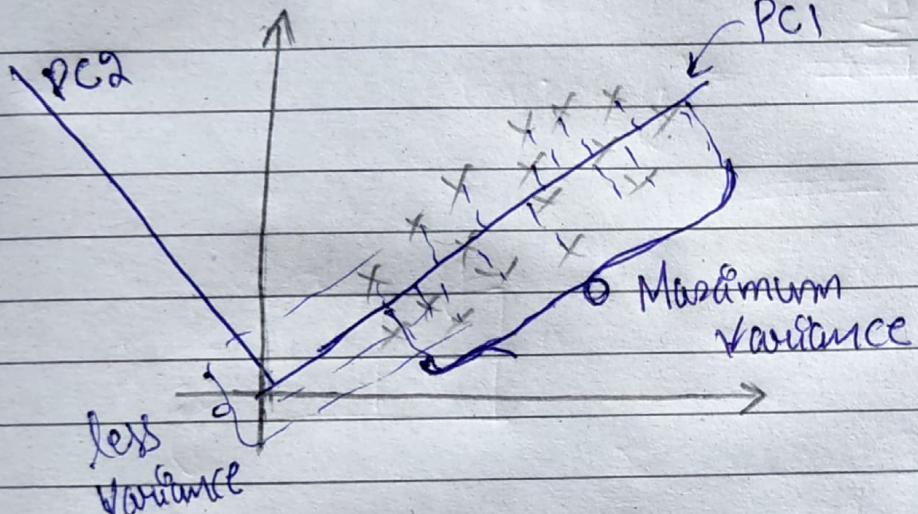
PC1 > PC2 > PC3

$\lambda_1, \lambda_2, \lambda_3$



→ projection

1D



* Implementation

eigen decomposition

> from sklearn.decomposition import PCA

* To visualize atleast we have to reduce dimensions to 3d.

> pca = PCA (n_components = 3)

> x_train = pca.fit_transform(x_train)

3d ← 4d

eigen vectors → pca.components_

> pca.explained_variance_ratio_ → Variance Captured