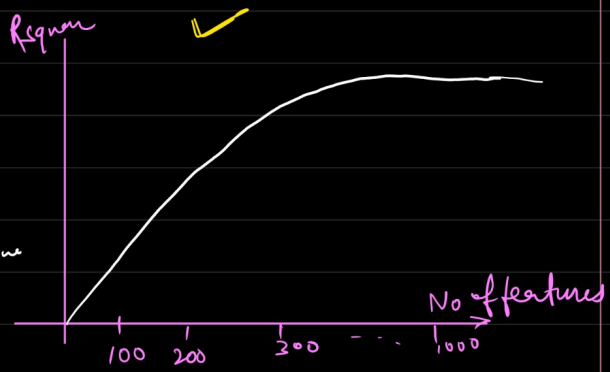
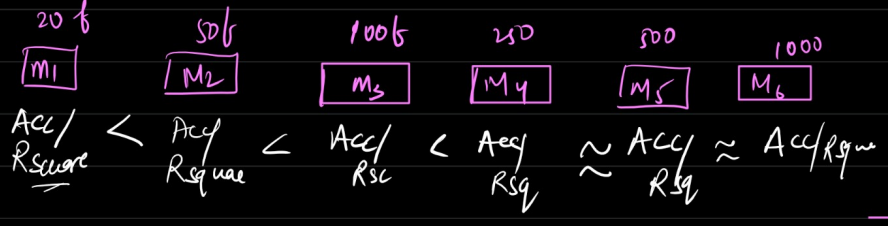
PCA (Principal Component analysis)





* With increase in no of features, after one point of the Acc/Rsquare stops increasing.

* Why ??

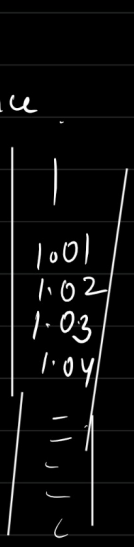
→ Multicollinearity ($f_1, f_2, f_3 \simeq f_4$)

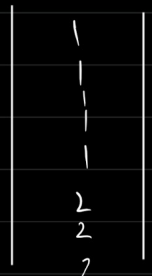
→ All the entries of features are exactly same.

→ two features are same

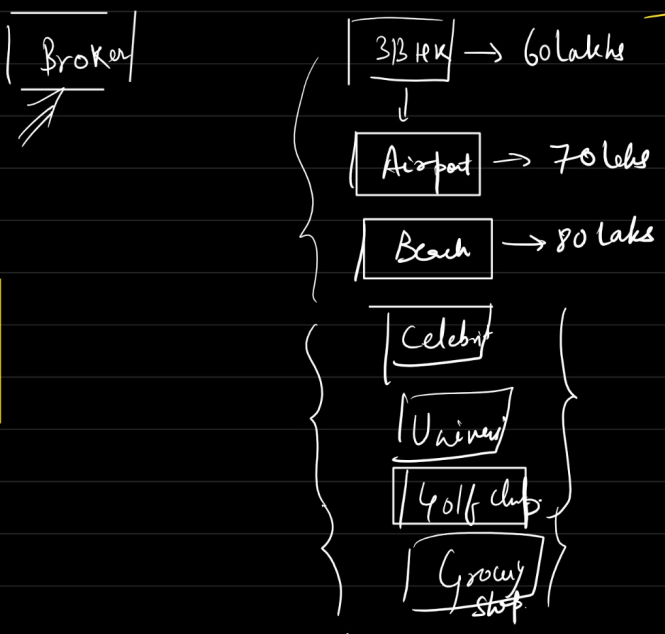
→ lots of duplicate entries. f_1

→ No variance.





* With increase in no of feature performance of model degrades.
  - Execution time
  - Evaluation



* Curse of dimensionality :-
  With increase in no of feature the performance of model degrades.

* To remove curse of Dimensionality :-
  1. Feature Selection
     → Correlation, VIF, P-value, Select k feature, I.G., Covariance, RFE
     - Disadvantage
       → dropping data
       → Time consuming.
  2. Feature Extraction
     ⇓
     PCA (Dimensionality reduction technique)

Why we should remove C.D ?

1. To improve performance of model.
2. Visualise the data → for insights
3. Prevents from overfitting
4. Better interpretation
5. To remove C.D

$f_1\ f_2 \ - - - \ f_{1000}$

↓

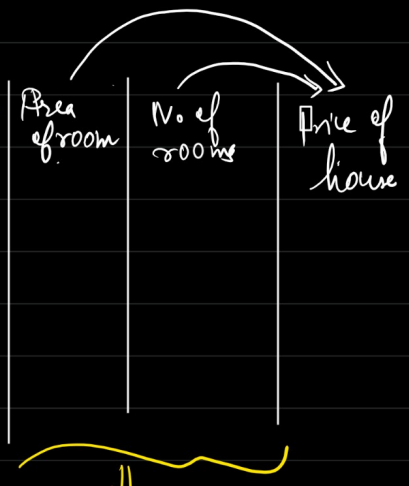
10000

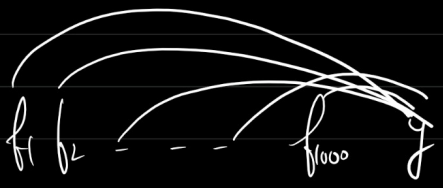
↓

3d, 2d.

* Feature Selection

→ Corr, Cov.





* Feature Extraction

↓

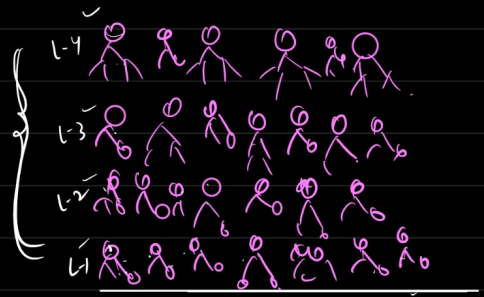
data transformation to extract new feature which represents both the feature.

(Domain understanding) House Area | Price

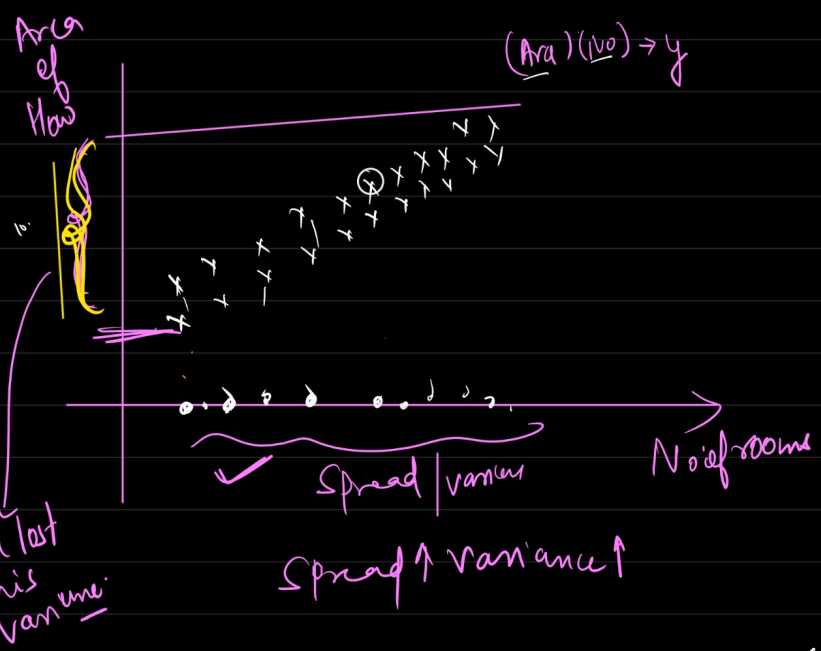
Let say # of bathroom | distance from airport | Distance from university

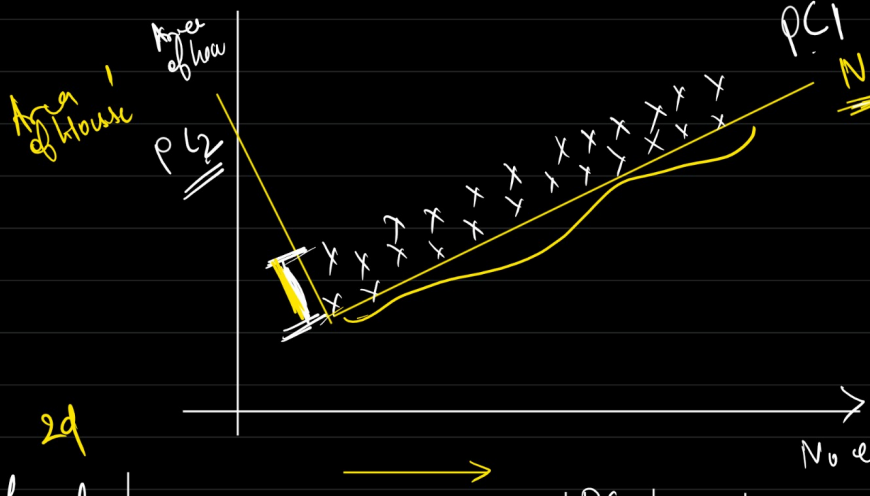
$f_1\ f_2 \ - - - \ f_{1000}$

* PCA

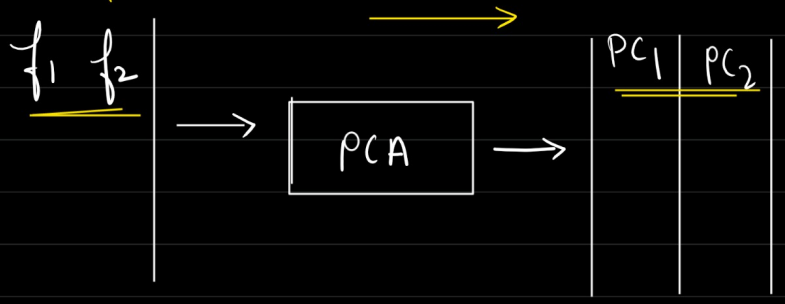


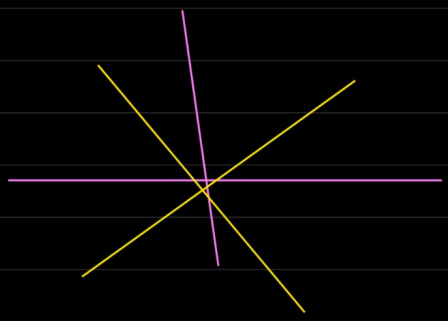
→ Variance / Spread is more



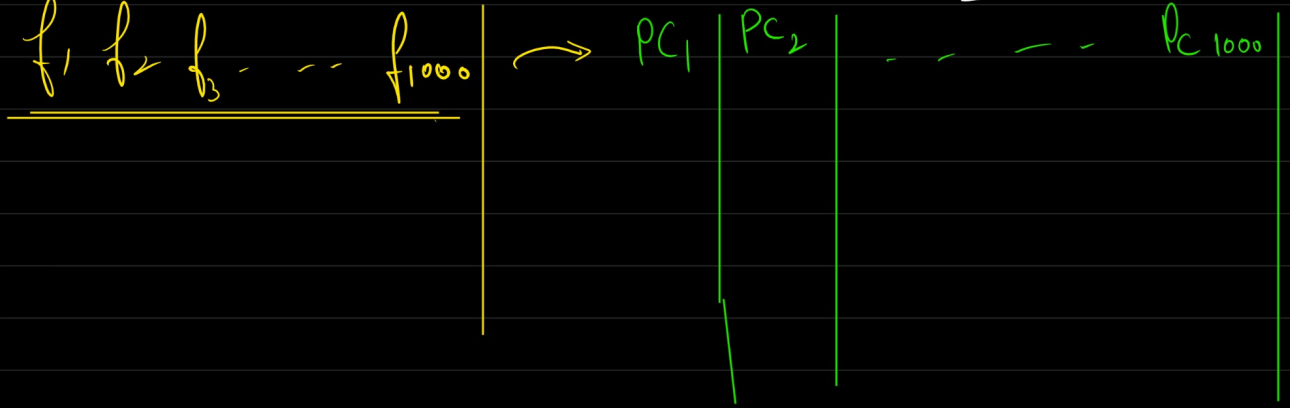


2d → 1d → (No of rooms only)
(without loosing much of variance / data)





As PC_1 has maximum variance, you will select only PC_1 for model training.

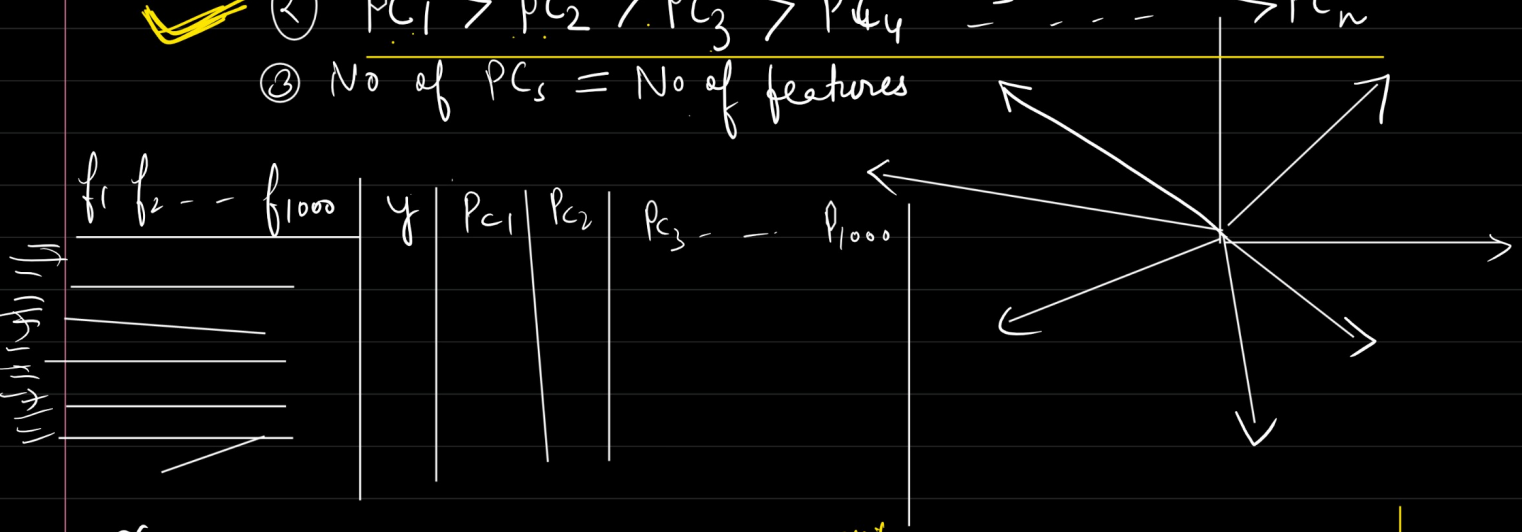


Two characteristics

① All of these PC's will be $\perp^r$ to each other.

② $PC_1 > PC_2 > PC_3 > PC_4 = \cdots > PC_n$

③ No of PC's = No of features



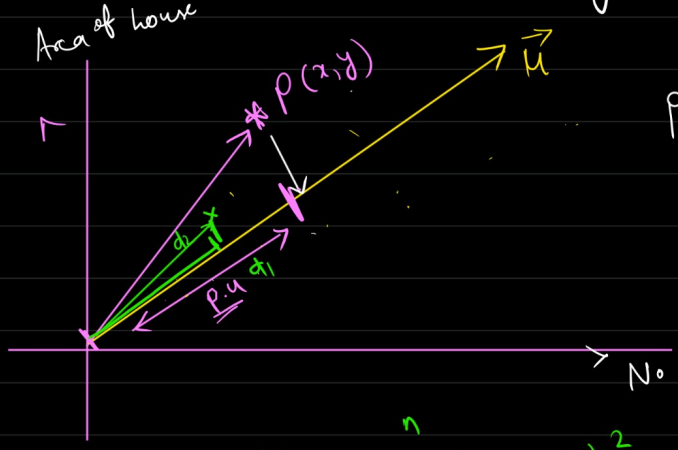
$\frac{r \times 1000}{F} = \frac{r \times 1000}{PC_4}$

100%

4d → 3f

PC_1	PC_2	PC_3	PC_4
70%	20%	9%	1%

* Axis transformation ⇒ Eigen decomposition of Covariance matrix.



Proj of p on $u = \frac{p \cdot u}{\|u\|} \to 1$

proj of p on $u = p \cdot u$

$d_0, d_1, d_2, d_3 \cdots d_n$

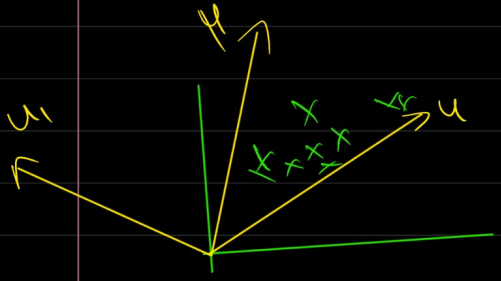
⇓

You want that unit vector where variance/spread is maximum.

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

⇓

Aim: To find that unit vector which captures maximum variance after projection



$$\text{Var} = \frac{\sum (x_i - \bar{x})^2}{n}$$

$$\Downarrow$$

$$\text{Covariance} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{n}$$

Covariance matrix

	X_1	X_2
X_1	$Cov(X_1, X_1)$	$Cov(X_1, X_2)$
X_2	$Cov(X_2, X_1)$	$Cov(X_2, X_2)$

$$\begin{bmatrix} Var\,X_1 & Cov(X_1, X_2) \\ Cov(X_1, X_2) & Var\,X_2 \end{bmatrix}$$

$$\Downarrow$$

Covariance matrix or Variance / Covariance matrix.

	X_1	X_2	X_3
X_1	$Var(X_1)$	$Cov(X_1, X_2)$	$Cov(X_1, X_3)$
X_2	$Cov(X_2, X_1)$	$Var(X_2)$	$Cov(X_2, X_3)$
X_3	$Cov(X_3, X_1)$	$Cov(X_3, X_2)$	$Var(X_3)$

$$\Downarrow$$

Whole idea is to capture spread.

Theorem :→ If you decompose a covariance matrix of features, then you will get eigen value and Eigen vector and eigen vector with highest magnitude of eigen value capture the maximum variance/spread

→ 2nd highest magnitude → 2nd highest var

→ 3rd highest matrix → 3 highest var

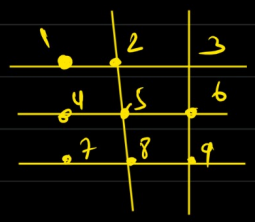
→ 4th " " → 4th highest.

$$PC_1 > PC_2 > PC_3$$

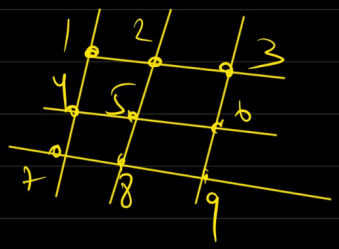
* Linear transformation of a matrix.

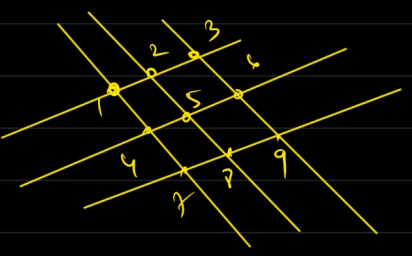
Analogy

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$



linear transformation





* **Linear transformation**
  - ↳ A matrix transformation that brings changes in the coordinate system.

$$A\vec{V} = \lambda \cdot \vec{V}$$

A → matrix, $\vec{V}$ → Eigenvector, λ → Eigenvalue, Eigen vector.

(When we use covariance matrix as A, Eigen decomposition of covariance matrix → Eigen value, Eigen vector)

Linear transformation pointers

→ Many vectors were changing both magnitude & direction.

→ Few vectors changed only magnitude & not the direction.

⇓

these vectors are called as **Eigen vector.**

$$\begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix}$$

$(1,0) \longrightarrow (3,0)$ direction same, magnitude changed

$(-1,-1) \longrightarrow (-4,-2)$,,

* Eigen values are change in magnitude for eigen vector

$$A\vec{V} = \lambda\vec{V}$$

A → Covariance matrix, λ → Scalar

* The Vector which only changes magnitude and not direction will be equals to dimension of matrix / no of feature.

No of feature = No of PC.

$Av = \lambda v$ → Vector.

↳ Stretch / shrink

Steps to Calculate Eigen value / Eigen vector (PC's)

Step-1 Standardise the data (make the data mean centred)

Step-2 Covariance matrix (df.cov())

Step-3 Eigen decomposition np.linalg

$Av = \lambda v$

λ → eigen value

v → eigen vector.

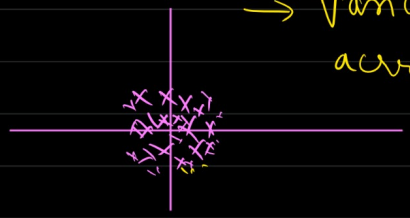
1000 f → 1000 PC's

$PC_1 > PC_2 > \dots > PC_n$

* Max var will be captured by first few principal comp.

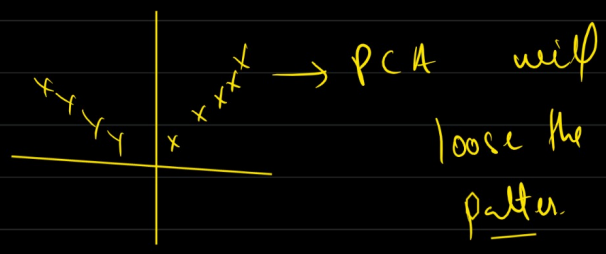
* PC's are ⊥r to each other

* Disadvantage of PCA



→ Variance spread across all axis is same.

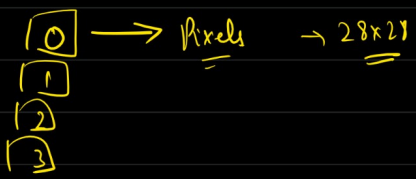
→ PCA will fail



→ PCA will loose the pattern.

* MNIST

0 — 9



→ Pixels → 28×28