Curse of Dimensionality

Essentially, as the number of dimensions (features) increases, the difficulty of working with the data grows exponentially. Here's a breakdown:

1. What it Means:

Imagine you have data points.

- In 1 dimension (a line), points are close or far.
- In 2 dimensions (a square), points spread out more.
- In 3 dimensions (a cube), they spread out even more.

As you keep adding dimensions, the "space" your data lives in becomes vast and mostly empty.

2. Why it Happens & Key Phenomena:

- **Data Sparsity:**
    - **Intuition:** Consider a line segment of length 1. If you have 10 data points uniformly spread, they are relatively close. Now consider a 1x1 square. To maintain the *same density* (points per unit volume), you'd need $10^2 = 100$ points. In a 1x1x1 cube, you'd need $10^3 = 1000$ points. For a hypercube with 10 dimensions, you'd need 1010 (10 billion) points to maintain the same density!
    - **Effect:** With a fixed amount of training data, the data becomes incredibly sparse in high dimensions. There are vast empty areas, making it hard to find neighbors or establish local patterns. Your data points effectively become isolated.
- **Distance Concentration:**
    - **Intuition:** In high dimensions, the distance between any two randomly chosen points tends to become almost the same. The difference between the maximum and minimum distances between points decreases relative to the minimum distance.
    - **Effect:** Algorithms that rely on distance measures (like k-Nearest Neighbors (k-NN), clustering algorithms like K-Means, or Support Vector Machines with certain kernels) become less meaningful. If all points are roughly equidistant, the concept of "nearest" neighbors loses its significance.
- **Combinatorial Explosion:** **The number of ways features can interact or combine grows exponentially, making it harder to find the truly relevant patterns.**

3. Consequences in Machine Learning:

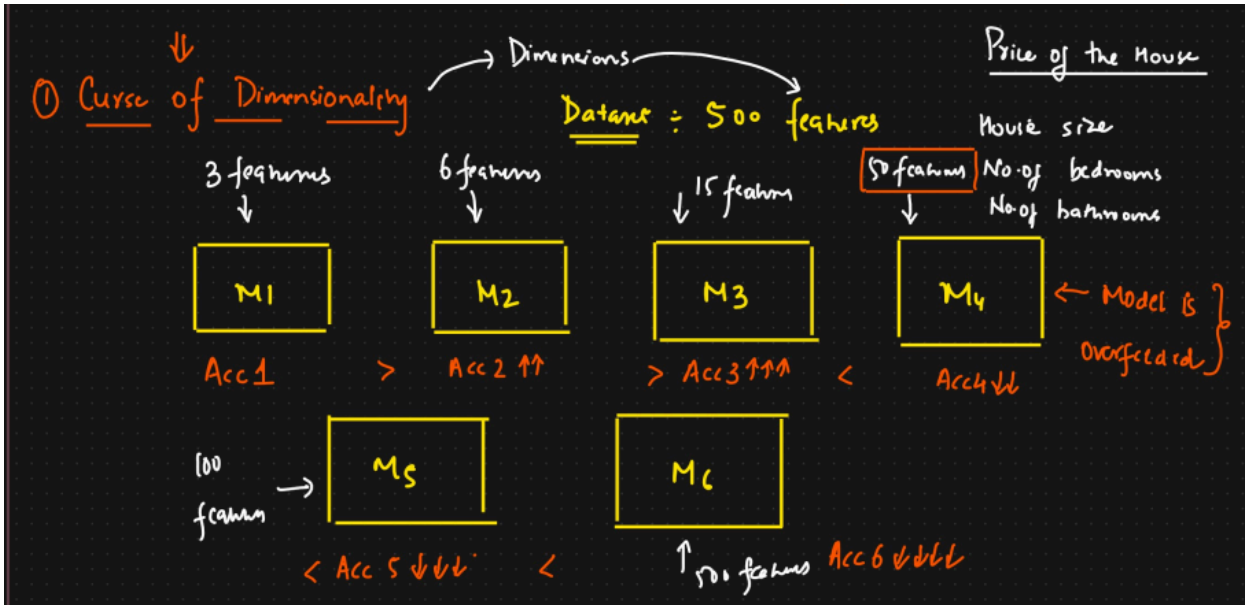
- **Increased Need for Data:** You need exponentially more data to generalize effectively and avoid spurious correlations as dimensions increase.
- **Overfitting:** Models have more flexibility to fit the training data perfectly, including noise, because there are so many dimensions to work with. This leads to poor performance on new, unseen data.

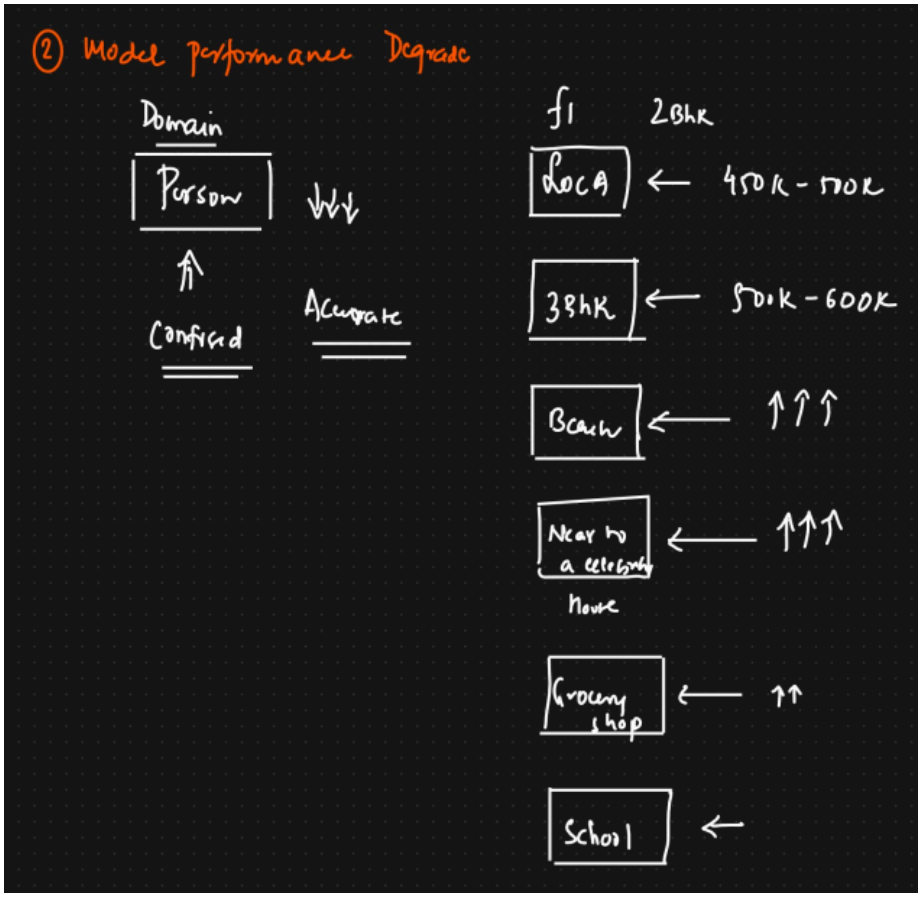
- **Computational Complexity:** Algorithms often take longer to run and require more memory as the number of dimensions increases (e.g., calculating distances, matrix operations).
- **Degraded Algorithm Performance:** Many algorithms, especially those relying on distance or density, perform poorly.
- **Redundancy and Irrelevance:** High dimensionality often comes with irrelevant features (that provide no useful information) or redundant features (that provide similar information to others), which can confuse models.

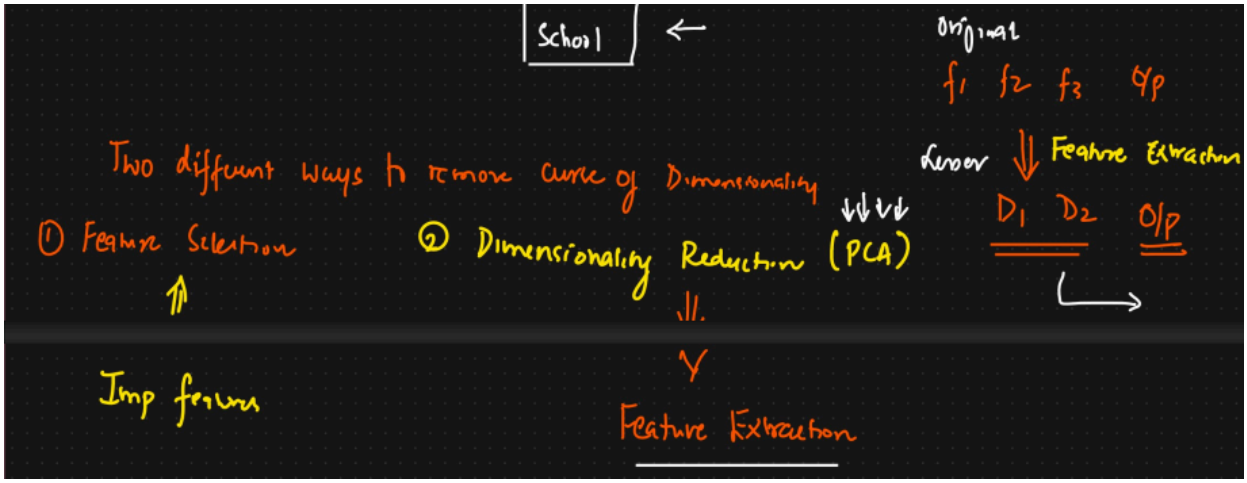
4. How to Combat the Curse of Dimensionality:

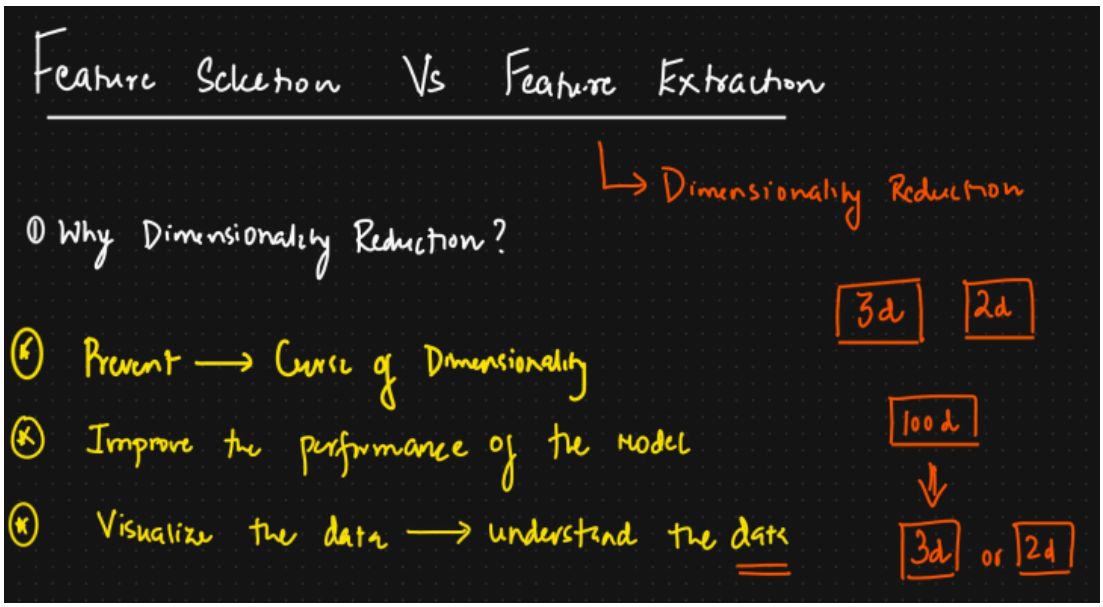
Since getting exponentially more data is often impossible, the common strategies involve reducing the effective number of dimensions:

- **Dimensionality Reduction Techniques:**
    - **Principal Component Analysis (PCA):** Creates new, uncorrelated features (principal components) that capture the **maximum variance**, allowing you to keep fewer dimensions. (Unsupervised)
    - **Linear Discriminant Analysis (LDA):** Similar to PCA but is supervised and tries to find dimensions that maximize separability between classes.
    - **t-Distributed Stochastic Neighbor Embedding (t-SNE) & Uniform Manifold Approximation and Projection (UMAP):** Non-linear techniques primarily used for visualization but can sometimes be used for dimensionality reduction.
- **Feature Selection:** Methods to select a subset of the original features that are most relevant to the task:
    - **Filter Methods:** Score features based on statistical properties (e.g., correlation with the target) independently of the model.
    - **Wrapper Methods:** Use a specific machine learning model to evaluate the usefulness of feature subsets.
    - **Embedded Methods:** Feature selection is built into the model training process (e.g., LASSO regression which uses L1 regularization to shrink some feature coefficients to zero).
- **Feature Engineering:** Creating new, potentially more informative features from the existing ones, possibly reducing the total number needed.
- **Regularization:** Techniques (like L1 - LASSO, L2 - Ridge) add penalties to the model's complexity, discouraging it from relying too heavily on any single feature or becoming overly complex, which helps prevent overfitting in high dimensions.
- **Domain Knowledge:** Using expertise about the problem domain to manually select relevant features or guide the feature engineering process.


① Curse of Dimensionality
Dimensions
Dataset : 500 features
Price of the House
House size
No. of bedrooms
No. of bathrooms
3 features
6 features
15 features
50 features
M1
M2
M3
M4
Model is overfitted
Acc 1 > Acc 2 ↑↑ > Acc 3 ↑↑↑ < Acc 4 ↓↓
100 features
M5
M6
< Acc 5 ↓↓↓ <
500 features
Acc 6 ↓↓↓↓


② Model performance Degrade
Domain
Person
↓↓↓
Confined
Accurate
f1
2BHK
Loca
450K - 550K
3BHK
500K - 600K
Beach
↑↑↑
Near to a celebrity house
↑↑↑
Grocery shop
↑↑
School


School
Original
f1 f2 f3 ... fp
Reduce
Feature Extraction
D1 D2 ... Dp
Two different ways to remove curse of Dimensionality
1) Feature Selection
2) Dimensionality Reduction (PCA)
Imp features
Feature Extraction


Feature Selection Vs Feature Extraction
Dimensionality Reduction
1) Why Dimensionality Reduction?
Prevent → Curse of Dimensionality
Improve the performance of the model
Visualize the data → understand the data
3d
2d
100 d
3d or 2d

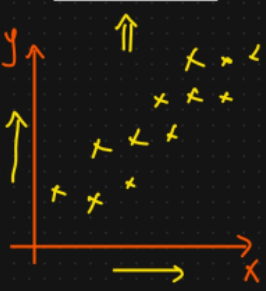
Feature Selection

Ip	Op
X	→ y
–	–
–	–
–	–
–	–

+ve ⇒

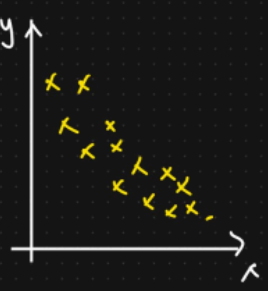
X↑	y↑
X↓	y↓

⇑

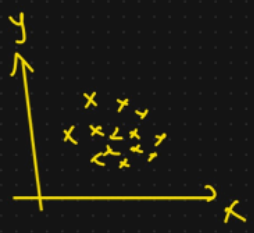


X↓	y↑
X↑	y↓

← -ve



No Relationship between X & Y



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

= +ve

= -ve

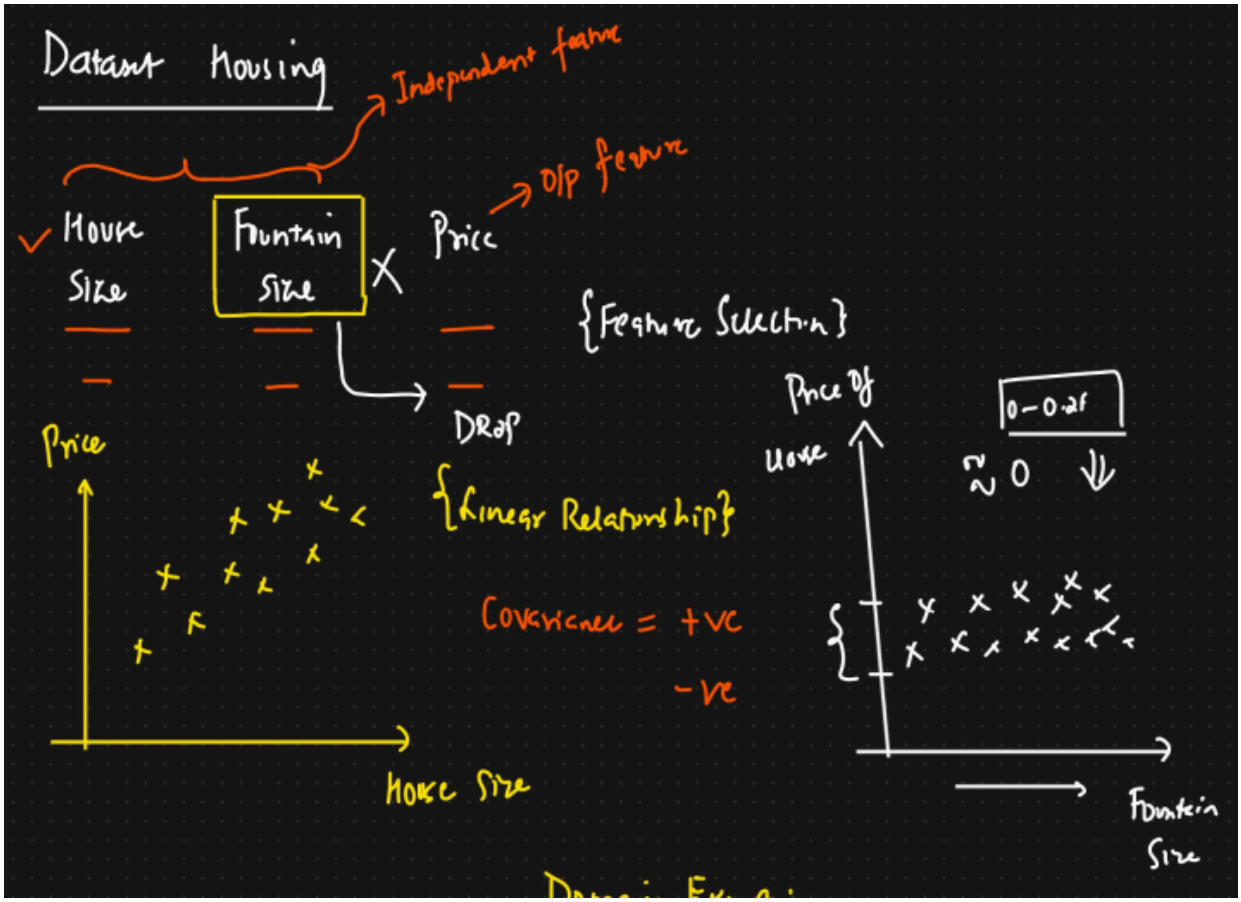
≈ 0

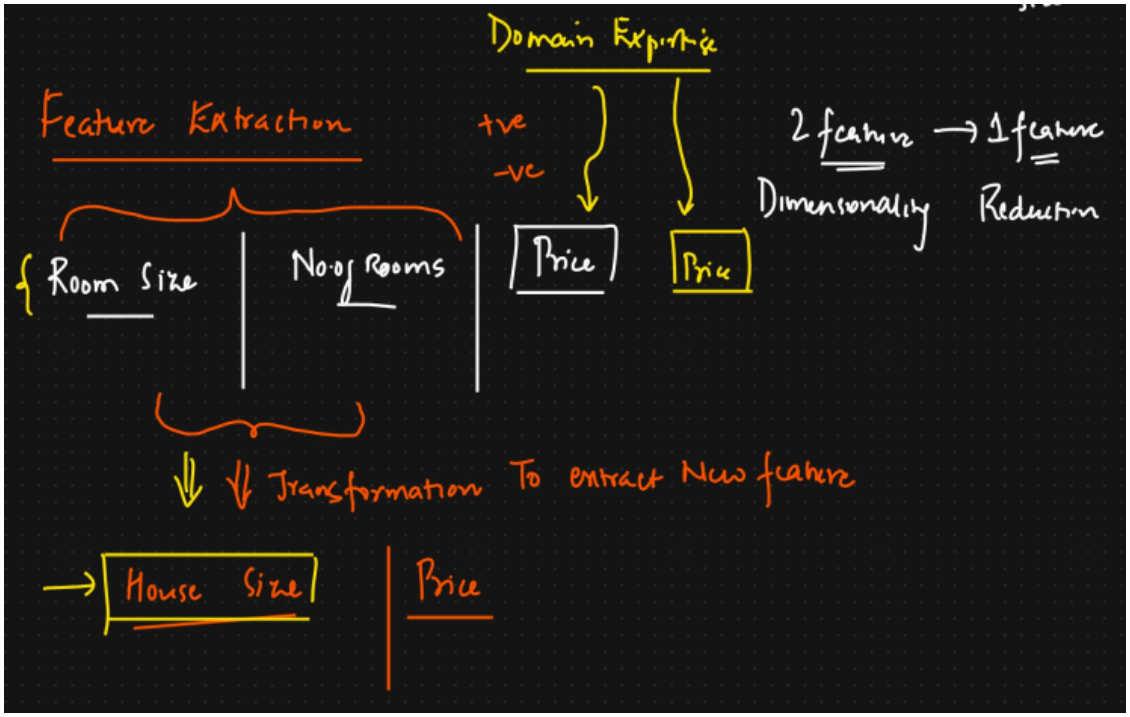
≈ 0 {No Relationship}

-ve correlated

$$\text{Pearson Correlation} = \frac{Cov(x,y)}{\sigma_x * \sigma_y} = \boxed{-1 \text{ to } 1}$$

The more toward the value of ±1 the

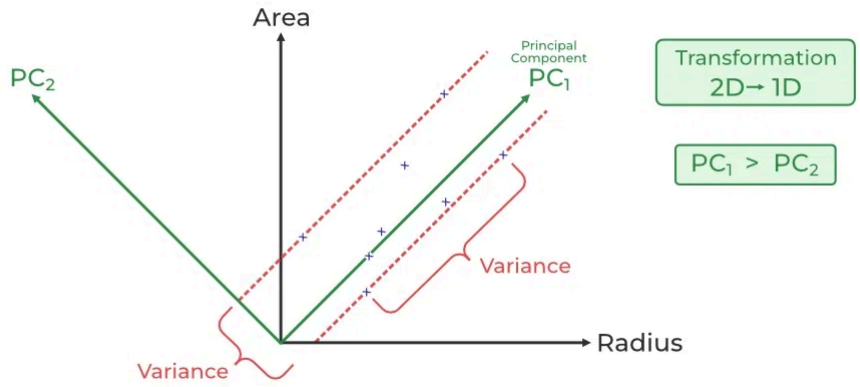

Dataset Housing
Independent feature
House Size
Fountain Size
Price
o/p feature
{Feature Selection}
DROP
Price
House Size
{Linear Relationship}
Covariance = +ve
-ve
Price of House
0-0.2
≈ 0
Fountain Size


Domain Expertise
Feature Extraction
+ve
-ve
2 feature → 1 feature
Dimensionality Reduction
Room Size
No. of Rooms
Price
Price
Transformation To extract New feature
House Size
Price

Principal Component Analysis (PCA)

1. Introduction: What is PCA?

- PCA is an **unsupervised machine learning algorithm** primarily used for **dimensionality reduction**.
- It transforms a dataset with many variables (features) into a smaller set of new variables, called **principal components (PCs)**, while trying to retain most of the original dataset's information (variance).
- It does this by identifying the directions (principal components) in the data along which the variation (spread) is maximal.
- PCA helps in simplifying complexity, improving model performance (by reducing overfitting and computational cost), and enabling visualization of high-dimensional data.



2. Goal & Applications:

- **Dimensionality Reduction:** Reduce the number of features (variables) in a dataset, especially when dealing with the "curse of dimensionality" (where too many features degrade model performance). This is useful when many original variables are correlated and contain redundant information.
- **Data Visualization:** Reduce high-dimensional data down to 2 or 3 principal components (PC1, PC2, PC3) which can then be plotted to visualize patterns or clusters.
- **Noise Reduction:** By focusing on components with the most variance, PCA can filter out noise which often resides in components with low variance.
- **Information Compression:** Used to compress data (like images or signals) by storing only the most significant principal components, allowing reconstruction with minimal information loss.
- **Feature Extraction:** Creates new, uncorrelated features (the PCs) from the original set, which can be used as input for other machine learning models.
- **Exploratory Data Analysis (EDA):** Helps understand the underlying structure and variance within the data.

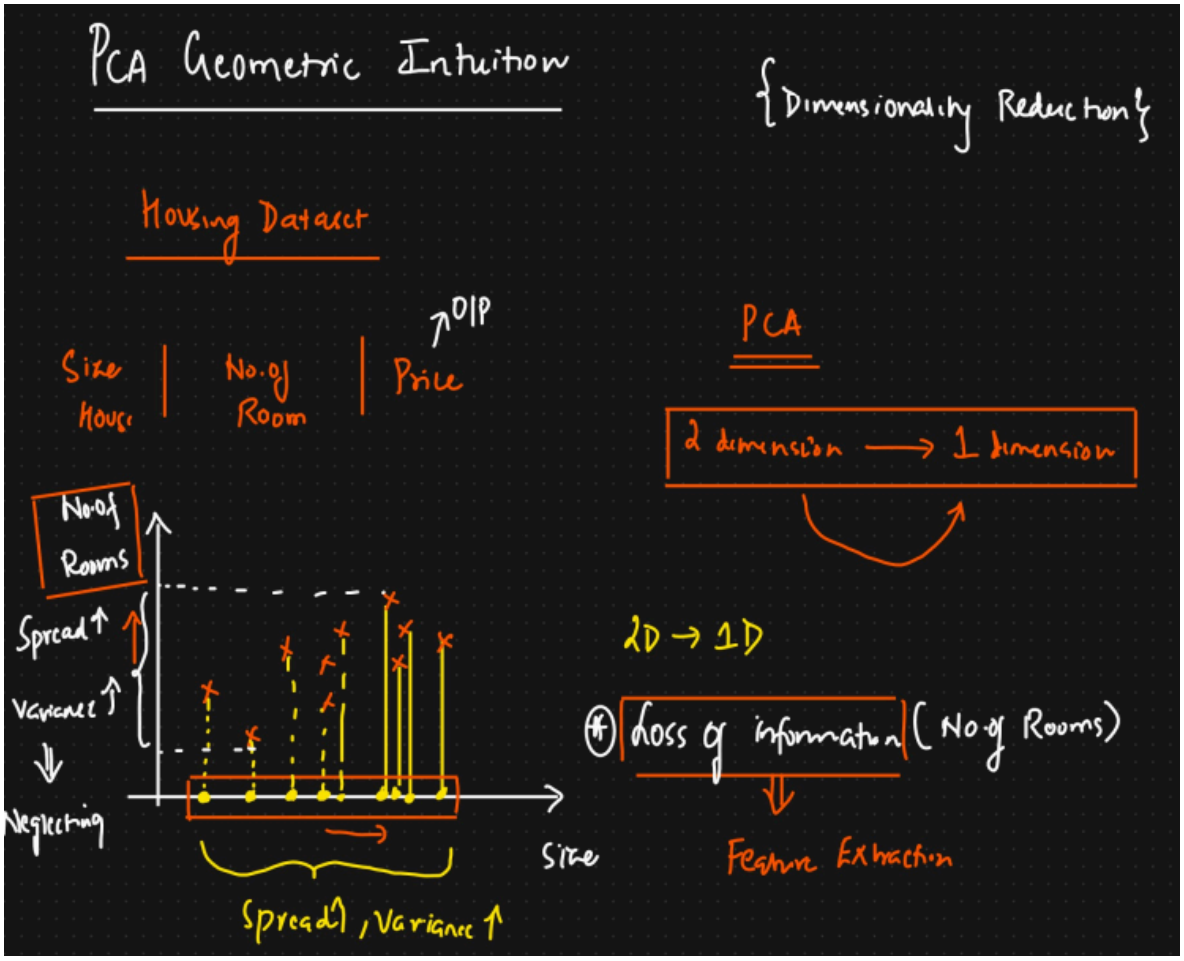
- **Specific Uses:** Finance (risk management, portfolio analysis), healthcare (analyzing patient data), image recognition (face recognition), bioinformatics.

3. How PCA Works: Step-by-Step

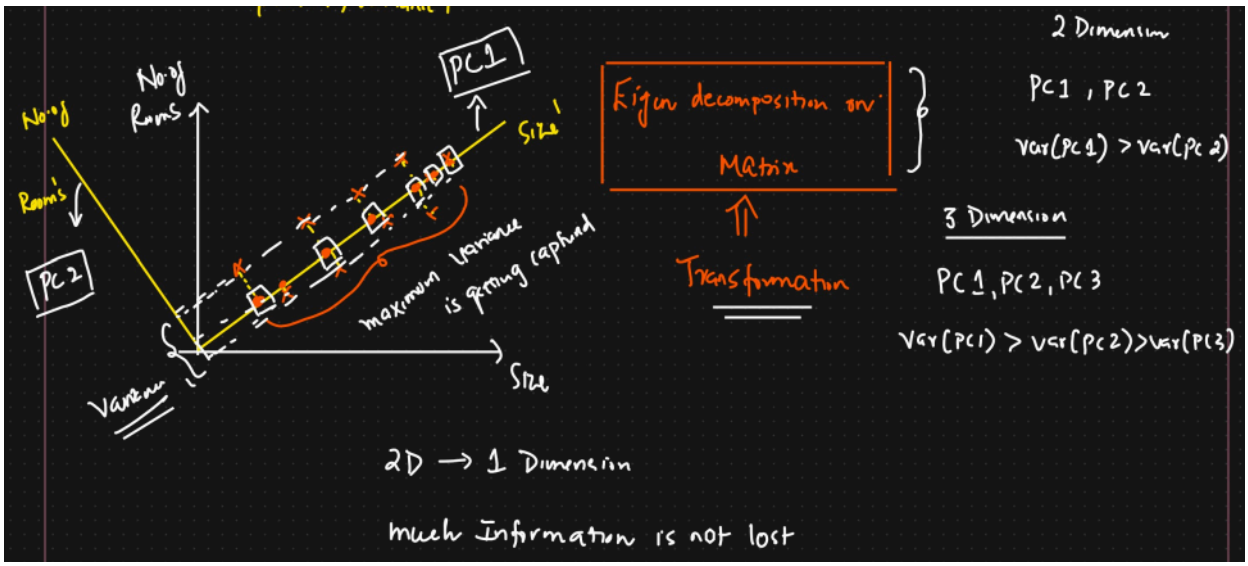
PCA transforms the data into a new coordinate system defined by the principal components. The core idea is to find the axes of **maximum variance**.

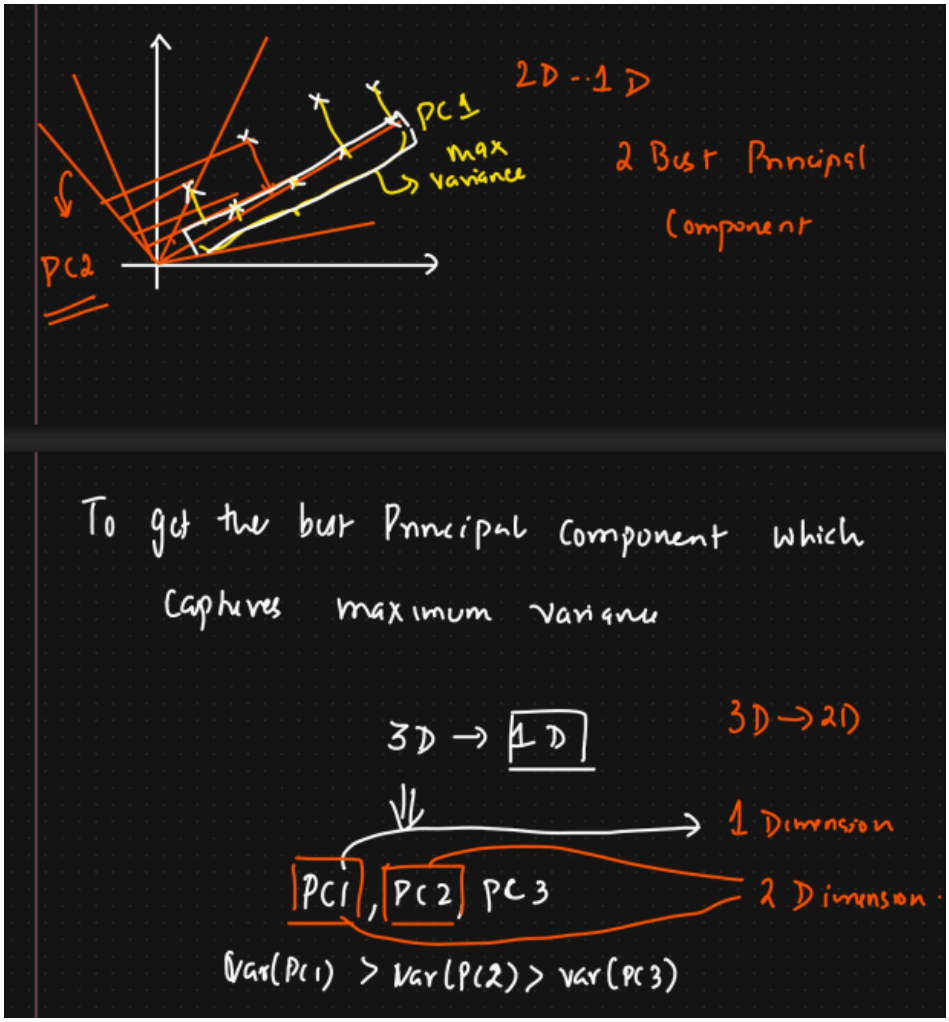
- **Step 1: Standardization**
  - **Why:** PCA is sensitive to the scale of the original variables. Variables with larger ranges would disproportionately influence the PCs. Standardization ensures all variables contribute equally.
  - **How:** Transform the data so that each feature has a mean of 0 and a standard deviation of 1. For each value x in a feature column, the standardized value x_{std} is calculated as: $x_{std} = \frac{x-\mu}{\sigma}$ where μ is the mean of the feature column and σ is its standard deviation.
- **Step 2: Covariance Matrix Computation**
  - **Why:** To understand how different variables in the input data vary with respect to each other (i.e., their linear relationship or correlation). Highly correlated variables suggest redundancy.
  - **How:** Calculate the covariance matrix for the standardized data. This is a symmetric square matrix (p x p, where p is the number of original features).
    - The diagonal elements represent the variance of each feature.
    - The off-diagonal elements represent the covariance between pairs of features. A positive covariance indicates features tend to increase together, negative indicates one increases as the other decreases, and near-zero indicates little linear relationship.
- **Step 3: Eigendecomposition of the Covariance Matrix**
  - **Why:** To find the principal components and their magnitudes. The eigenvectors of the covariance matrix represent the directions of maximum variance in the data (these are the principal components), and the corresponding eigenvalues represent the magnitude of the variance along those directions.
  - **How:** Calculate the eigenvectors and eigenvalues of the covariance matrix.
    - **Eigenvectors (v):** These vectors define the directions of the new axes (principal components). They are unit vectors and are orthogonal (perpendicular) to each other, meaning the principal components are uncorrelated.
    - **Eigenvalues (λ):** These are scalars indicating how much variance is captured by each eigenvector (principal component). A higher eigenvalue means the corresponding principal component captures more variance from the dataset.
- **Step 4: Sort Eigenvectors and Select Principal Components**
  - **Why:** To reduce dimensionality by keeping only the most significant components.

    - **How:** Sort the eigenvectors in descending order based on their corresponding eigenvalues. Decide how many principal components (k) to keep. Common methods include:
        - **Eigenvalue Criterion:** Keep components with eigenvalues greater than 1 (Kaiser's criterion).
        - **Proportion of Variance Explained:** Keep enough components to explain a certain percentage of the total variance (e.g., 90%, 95%, 99%). The proportion of variance explained by a PC is its eigenvalue divided by the sum of all eigenvalues.
        - **Scree Plot:** Plot the eigenvalues (sorted) against the component number. Keep the components before the "elbow" point where the eigenvalues start to level off.
- **Step 5: Transform the Data**
    - **Why:** To project the original data onto the new, lower-dimensional subspace defined by the selected principal components.
    - **How:** Create a projection matrix using the selected top k eigenvectors. Multiply the original standardized data matrix by this projection matrix to obtain the new dataset with k dimensions (the principal component scores). Each data point in the new dataset represents the projection of the original data point onto the principal component axes.

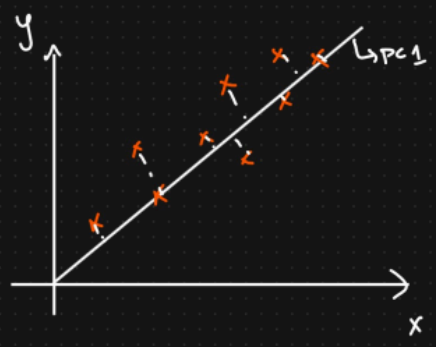


If we directly project the points on the X axis then there will be loss of information about no. of rooms.




PC1
max variance
PC2
2D → 1D
2 Best Principal Component
To get the best Principal Component which captures maximum variance
3D → 1D
3D → 2D
1 Dimension
PC1
PC2
PC3
2 Dimension
Var(PC1) > Var(PC2) > Var(PC3)

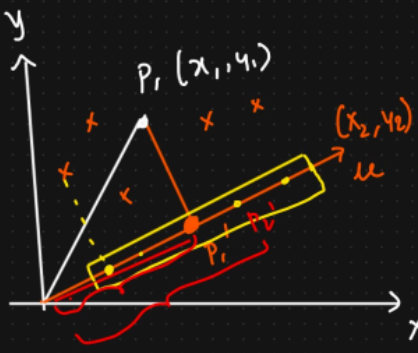
Maths Intuition behind PCA Algorithm



2D → 1D

① Projection ✓

② Cost function → Variance ✓



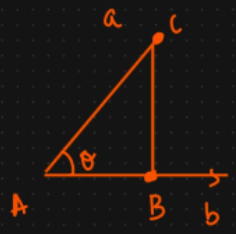
$$Proj_{P_1} u = \frac{P_1 \cdot u}{\|u\|}$$

$\|u\| = 1 \rightarrow$ unit vector

$$Proj_{P_1} u = P_1 \cdot u \Rightarrow \text{Scalar value}$$

$\Downarrow$

P_1'



$$P_0', P_1', P_2', P_3', P_4', \ldots P_n'$$

$\Downarrow$

Scalar values

$\Downarrow$

Variance

$P_0', P_1', P_2', P_3', P_4', \cdots P_n'$

$\Downarrow$

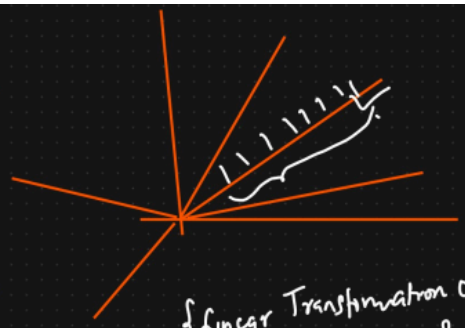
$x_0', x_1', x_2', x_3', x_4' \cdots x_n'$

Max Variance $= \sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}$ { Goal : Find the best unit vector which captures maximum variance }.

$\Downarrow$

Cost function

$\Downarrow$ Eigen vectors And $\Downarrow$ Eigen values.

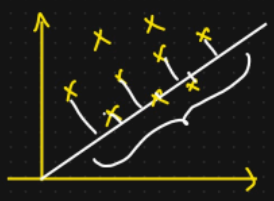


1. Covariance Matrix between features
2. Eigen vectors and Eigen values will found out from this covariance matrix { Linear Transformation of matrix }

$$Av = \lambda v$$

3. Eigen vector $\longrightarrow$ Eigen value $\rightarrow$ magnitude of the Eigen vector $\longrightarrow$ Capture the maximum variance

Eigen vectors And Eigen values [Linear Transformation]



[Eigen decomposition of covariance Matrix]

⇓

Eigen vector & Eigen values

$$\begin{bmatrix} \quad \end{bmatrix} * [v] = \lambda * v$$

↓

Eigen value

$$A * v = \lambda * v$$

Eigen vector → Maximum magnitude

⇓

Principal Component

⇓

Max Var

Eigen vector → Max Magnitude

⇓

Max Eigen vector

⇓

Best Principal Component → PC1

Steps to calculate Eigen value and vectors

① Covariance of features

[X , Y] → X'

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

2x2

$$A = \begin{array}{c|cc} & X & Y \\ \hline X & Var(X) & Cov(X,Y) \\ Y & Cov(Y,X) & Var(Y) \end{array}$$

$Cov(X,X) = Var(X)$

$Cov(Y,Y) = Var(Y)$

	X	Y	Z
X			
Y			
Z			

$\lambda_1, \lambda_2, \lambda_3$

f_1 & f_2

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

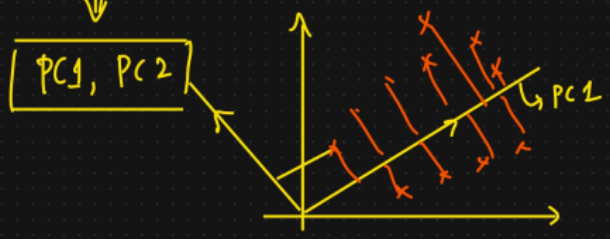
λ_1 λ_2

λ_1, λ_2 → Eigen values

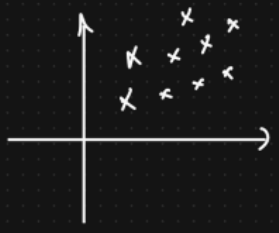
λ_1 ⇓ PC1 . λ_2 ⇓ PC2

⇓

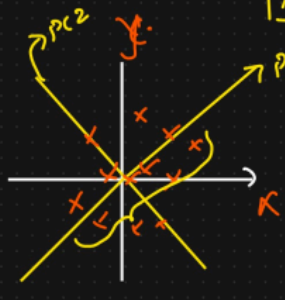
PC1, PC2



(1)



2D → 1D



λ_1 ⇒ magnitude of the Eigen vector

① Standardize the data

② Covariance Matrix of X & Y

$$A = \begin{array}{c|cc} & X & Y \\ \hline X & Var(x) & Cov(x,y) \\ Y & Cov(y,x) & Var(y) \end{array} \quad 2 \times 2$$

③ Find out Eigen vectors And value

$$A v = \lambda v$$

$[\lambda_1, \lambda_2]$ => Eigen values.

$\lambda_1 \rightarrow PC1$, $\lambda_2 \rightarrow PC2$

3D → 2D

$\lambda_1, \lambda_2, \lambda_3$

$\lambda_1 \rightarrow PC1$, $\lambda_2 \rightarrow PC2$, $\lambda_3 \rightarrow PC3$

PC1, PC2 → [1D]

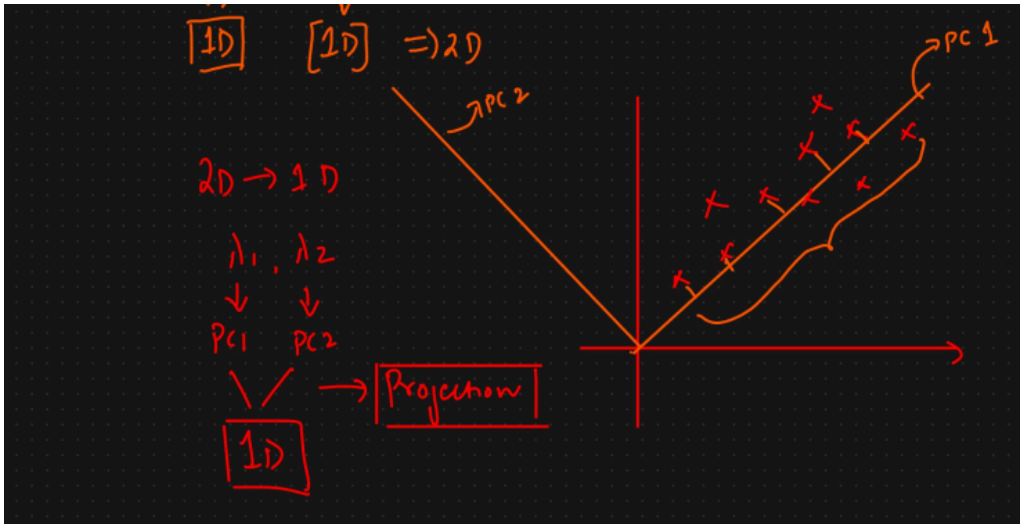
PC3 → [1D]

=> 2D

3D → 1D

$\lambda_1, \lambda_2, \lambda_3$

[1D]



4. Key Concepts Summary

- **Principal Components (PCs):** New, uncorrelated variables that are linear combinations of the original variables. They represent directions of maximum variance in the data and are orthogonal to each other. PC1 captures the most variance, PC2 the second most, etc.
- **Variance:** A measure of the spread or dispersion of data points along a particular direction. PCA seeks directions that maximize this.
- **Covariance:** A measure of the joint variability of two random variables. Used to understand linear relationships between features.
- **Eigenvectors:** Directions (vectors) in space that remain unchanged in direction when a linear transformation (represented by the covariance matrix in PCA) is applied. They define the principal component axes.
- **Eigenvalues:** Scalars representing the factor by which an eigenvector is stretched or shrunk by the linear transformation. In PCA, they quantify the amount of variance captured by each principal component (eigenvector).

5. Assumptions and Limitations

- **Linearity:** PCA assumes that the principal components are linear combinations of the original features and that the underlying relationships are linear. It may not perform well if the data has strong non-linear structures (Kernel PCA can sometimes address this).
- **Scale Sensitivity:** As mentioned, PCA is highly sensitive to the scale of the data. Standardization is crucial.
- **Outlier Sensitivity:** Outliers can significantly skew the results because PCA tries to maximize variance. Consider outlier detection and removal before applying PCA.
- **Correlation Assumption:** PCA works best when features are correlated. If features are already uncorrelated, PCA won't achieve significant dimensionality reduction.

- **Interpretability:** While the principal components capture maximum variance, they are linear combinations of original features and may not always have a clear real-world interpretation.
- **Information Loss:** Dimensionality reduction inevitably involves some information loss, though PCA aims to minimize this by retaining components with the most variance.
- **Missing Values:** Standard PCA implementations typically require complete data (no missing values). Missing values need to be handled (e.g., through imputation) beforehand.