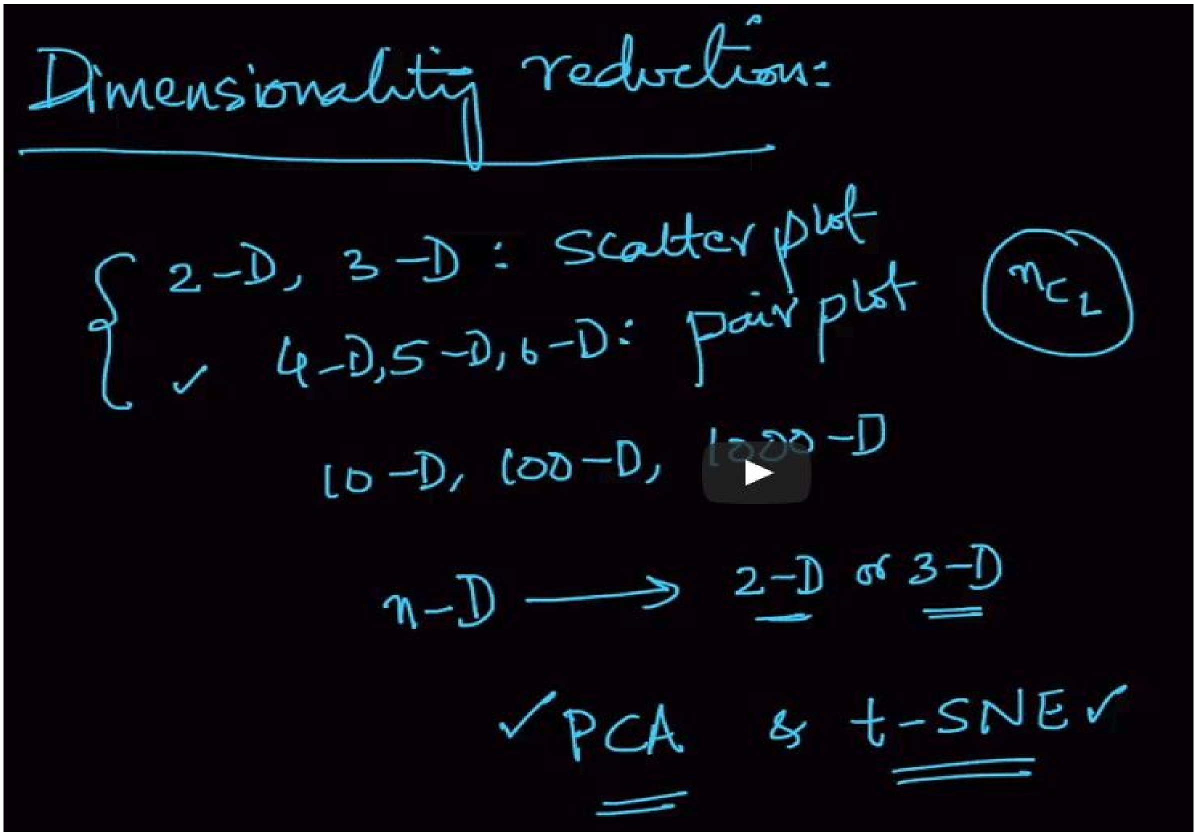
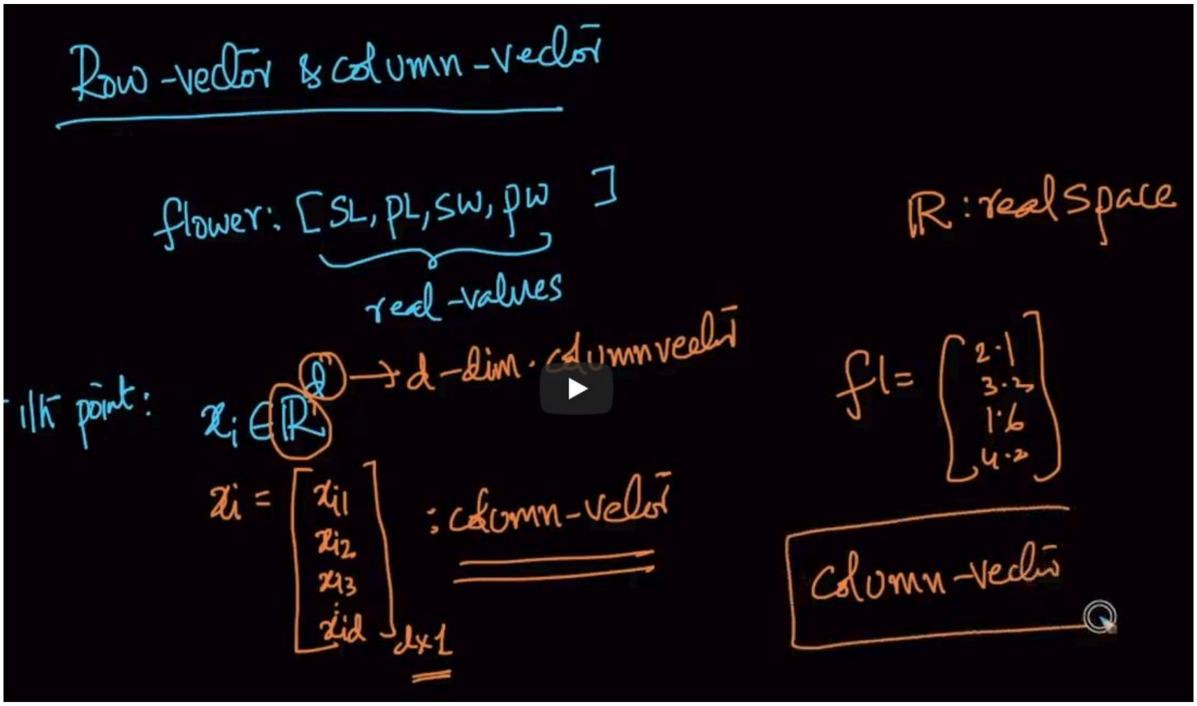
# DIMENSIONALITY REDUCTION




We've seen that 2-d,3-D data can be analyzed using scatterplots. 4-D,5-D can be dealt with pair plots(Iris) but if we've n-D data then we convert them into 2-D or 3-D using dimensionality reduction techniques like PCA & t-SNE.

## COLUMN VECTOR AND ROW VECTOR

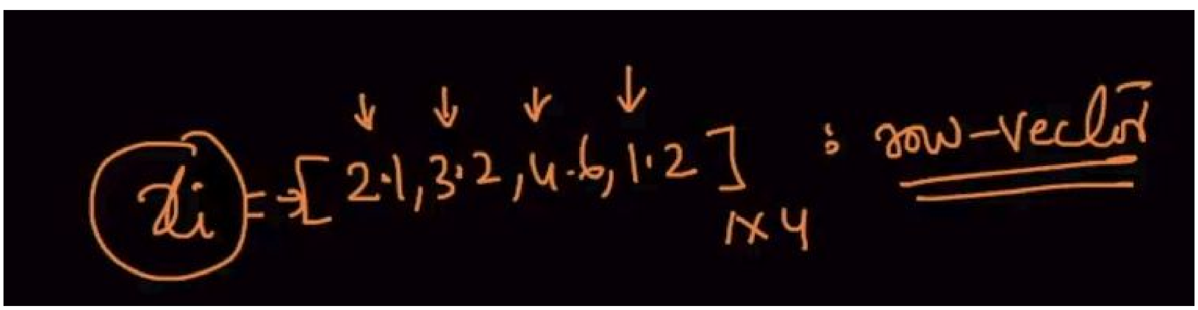



Let's take our flower dataset. It has 4 features and all have real-values (values like 1.56,5.89 etc)

Then suppose we are taking out the ith value then $x_i \in \mathbb{R}^d$ then x is our column vector with d dimensions and **one column** and **d-rows**

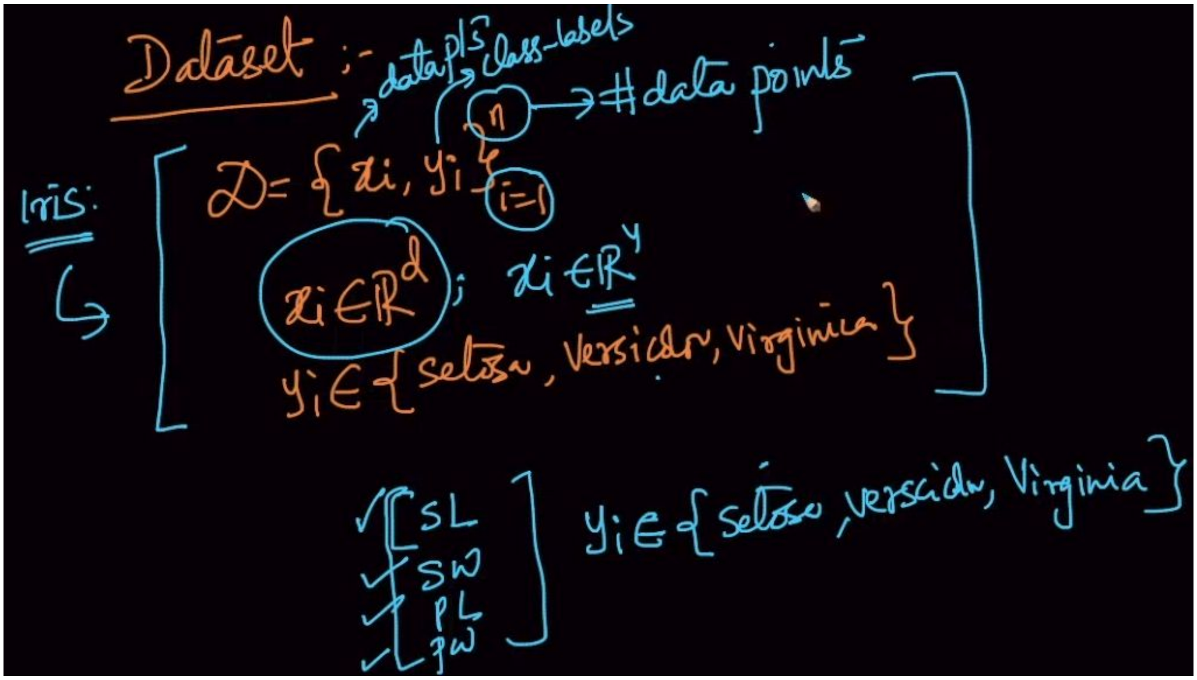
Note : If nothing is specified then we take our vector as column-vectors

**Row-vector**




This is a row-vector with 1 row and 4 columns

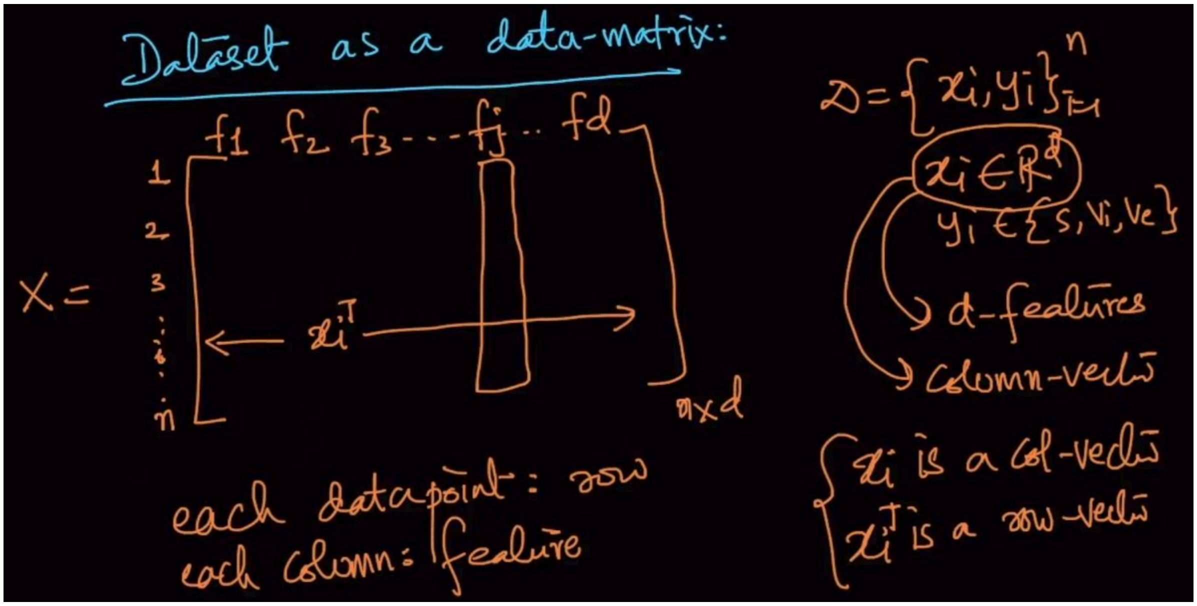
## How to represent a dataset ?




A dataset has generally two variables: data points ($x_i$) and class-labels($y_i$) where n- No. of data points
Here our dataset (Iris) has 4 features, therefore, it is represented as $x_i \in \mathbb{R}^4$ and
y $\in \{setosa, versicolor, virginica\}$

So, this is how a dataset is represented. There are other ways as well

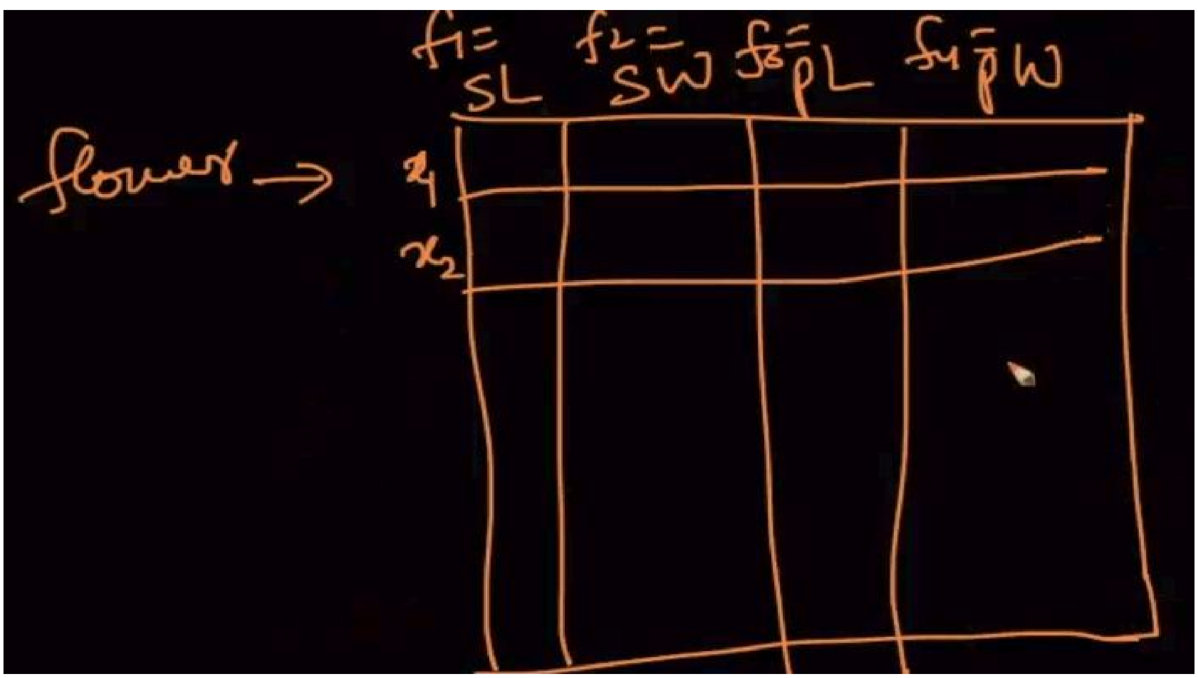
# DATASET REPRESENTATION USING MATRIX




X is an n*d matrix where n is number of data points(rows) and d is dimensions .

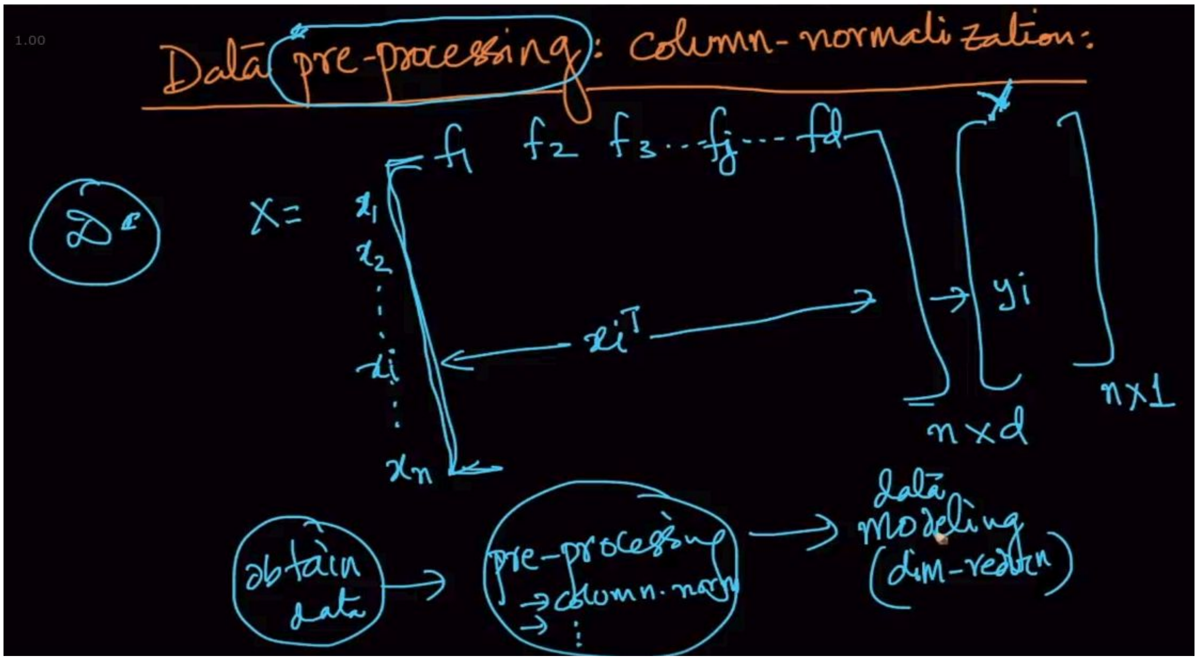
Note: Here an ith point is represented as $x_i^T$ where T is transpose as $x_i$ is a column vector.

Every column is a feature here.

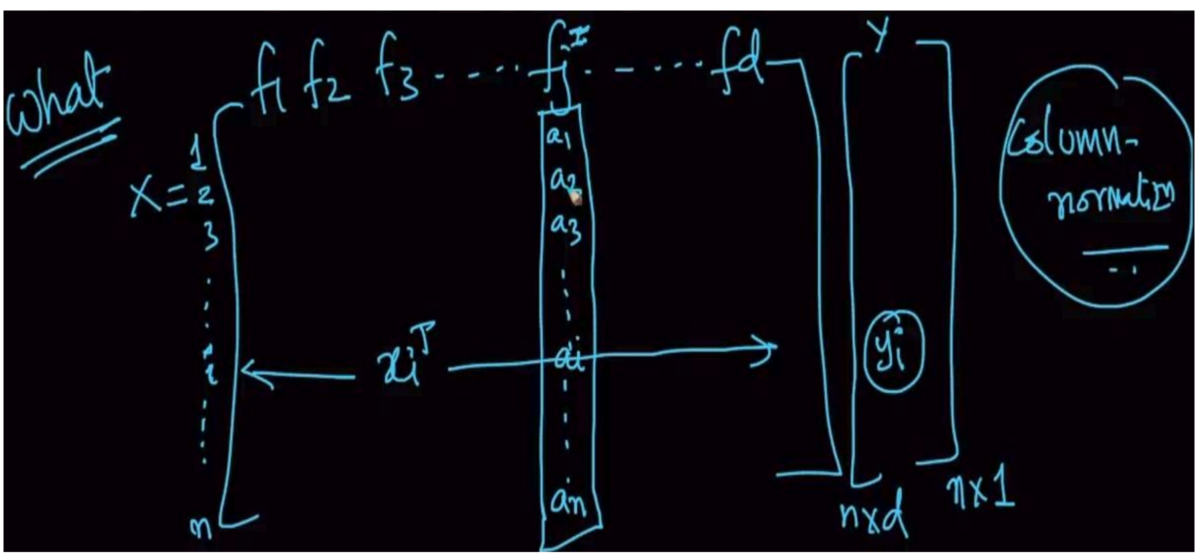



y is a (1*d) column vector which tells us about the target values.
This is how it is represented for Iris dataset where the f is feature and x (data points) are rows

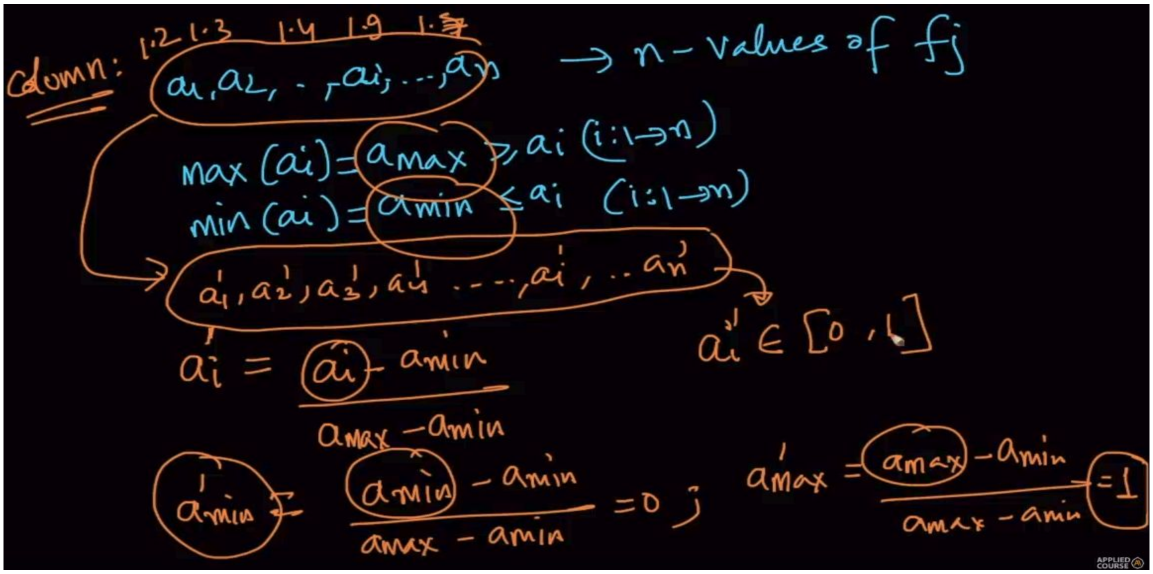
## DATA PREPROCESSING : COLUMN NORMALIZATION




Data preprocessing is an important step. Before data modeling or running ML algorithms we've to make The dataset such that the ML algos perform better there. Column normalization is one data preprocessing step done before Dimensionality Reduction.

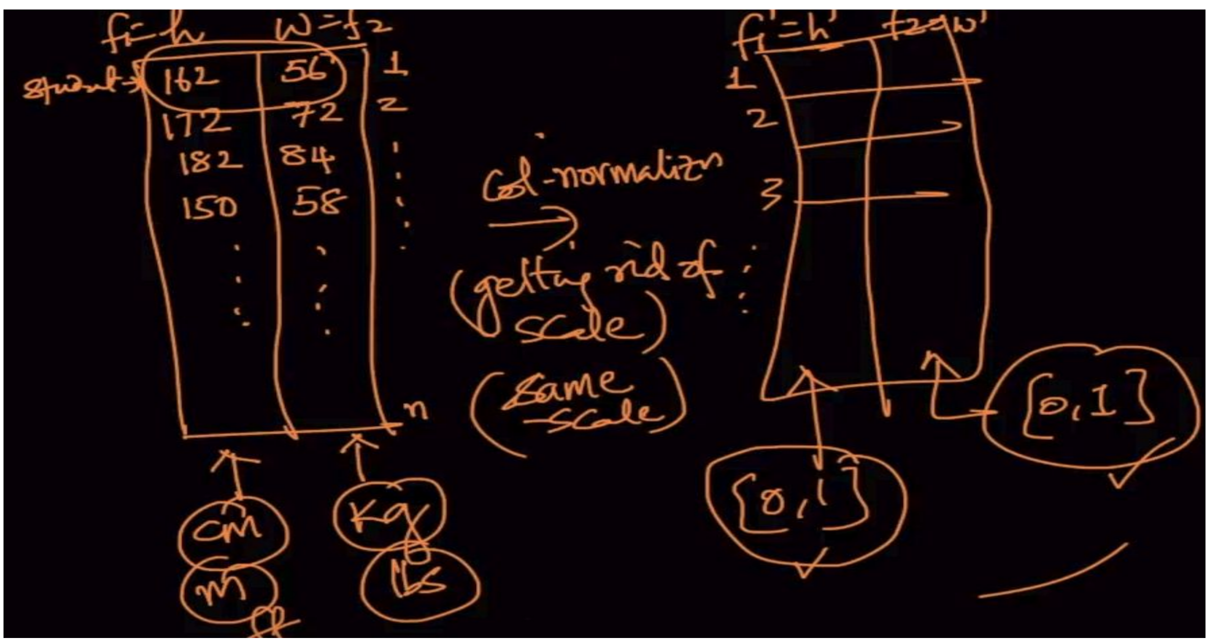



Here , we select a all the columns one-by-one to normalize them.



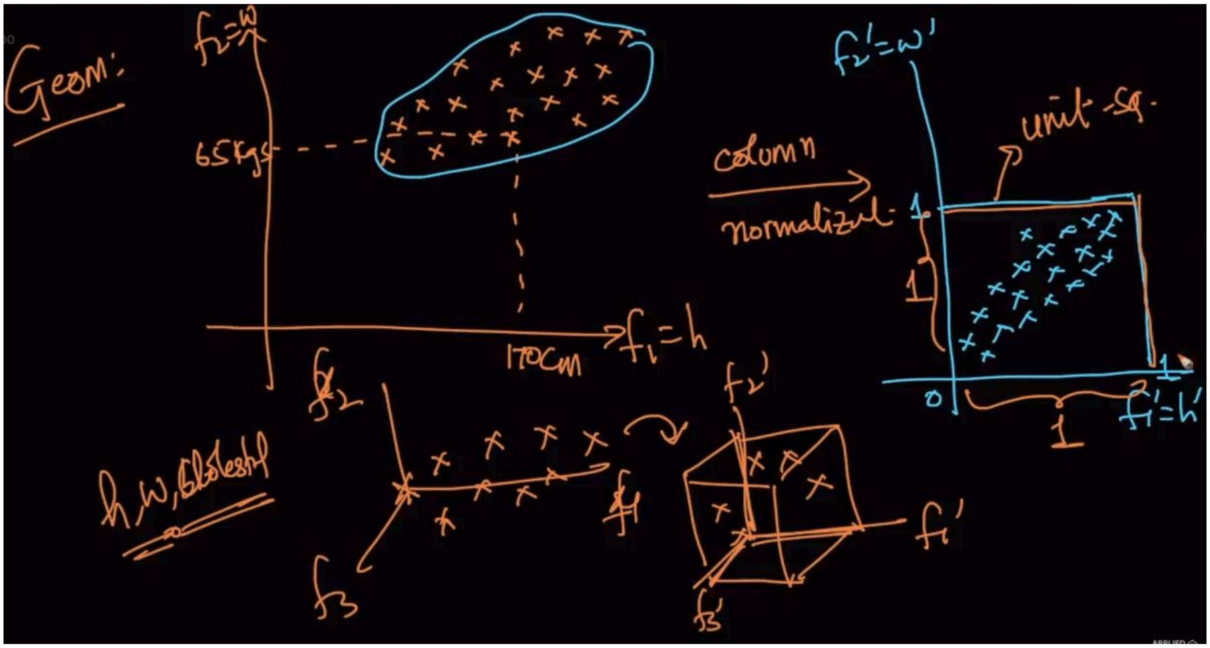

We select a feature (column) and then get the max and min values. After that we perform the above calculations to normalize them i.e make an $a_i'$ such that $a_i' \in [0, 1]$. This operation is column normalization

**Why are we doing this step?**



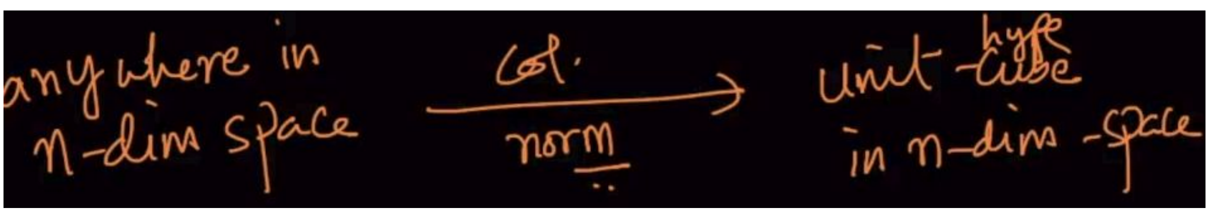

Here, the data is in cm and kgs but if it'd been lbs in place of kgs it would've nearly been doubled so we get rid of that problem by scaling in the same scale i.e [0,1]

**GEOMETRIC INTUITION**

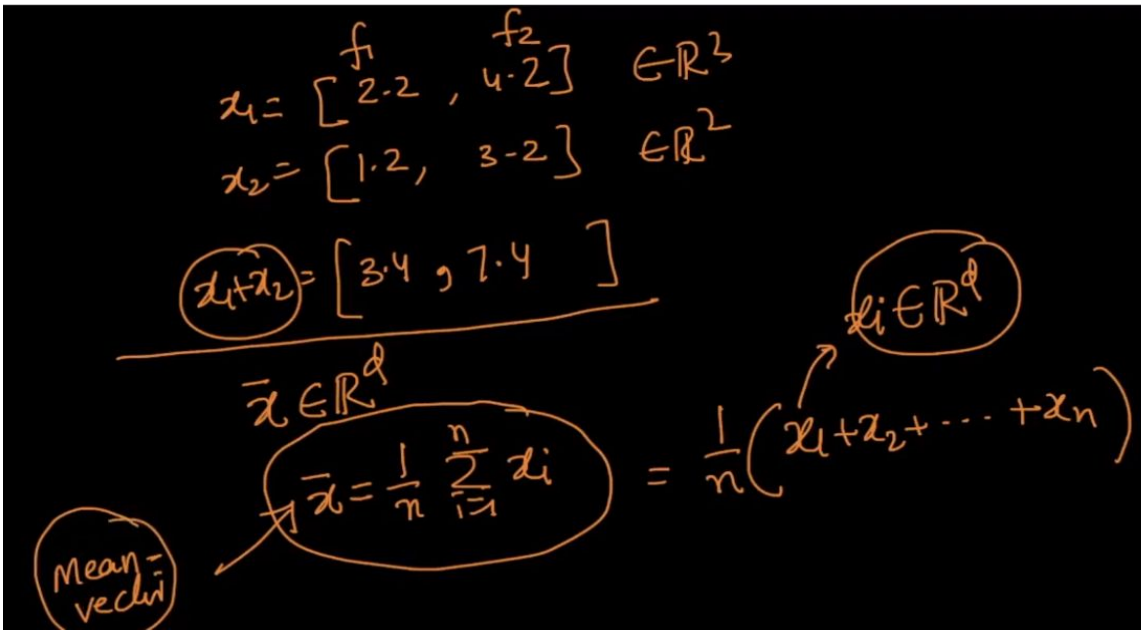



We plotted the data points normally. By column normalization, we scale them and squish into a unit square (1*1) so that they lie in the same scale. We are not changing how is the data aligned or positions of data points just scaling it.

Same for 3-D or n-D data.

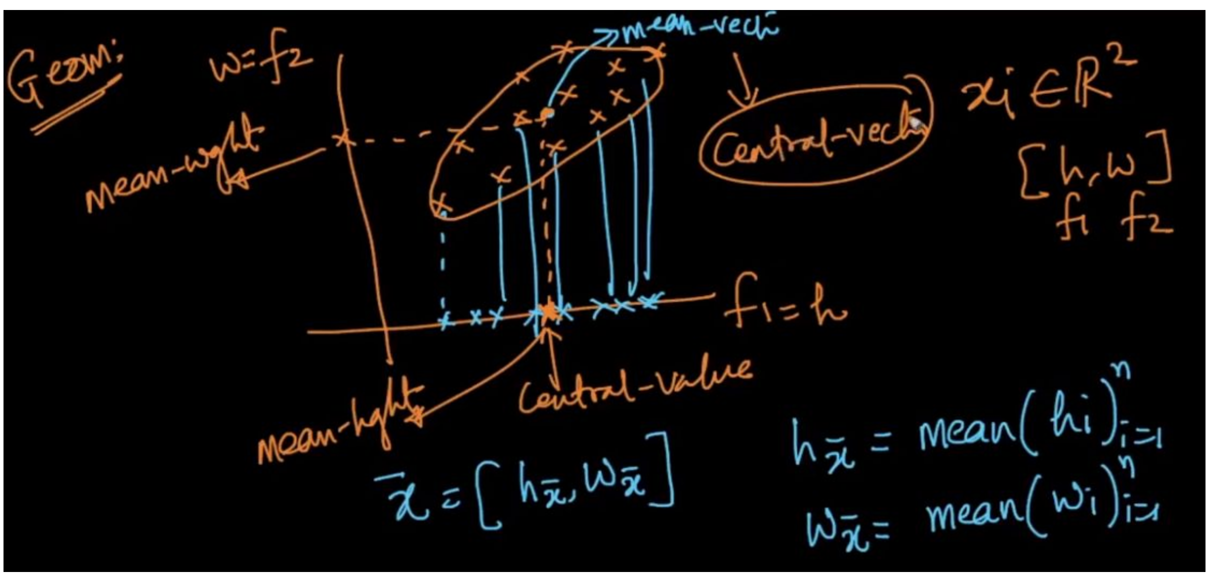


**MEAN VECTOR**



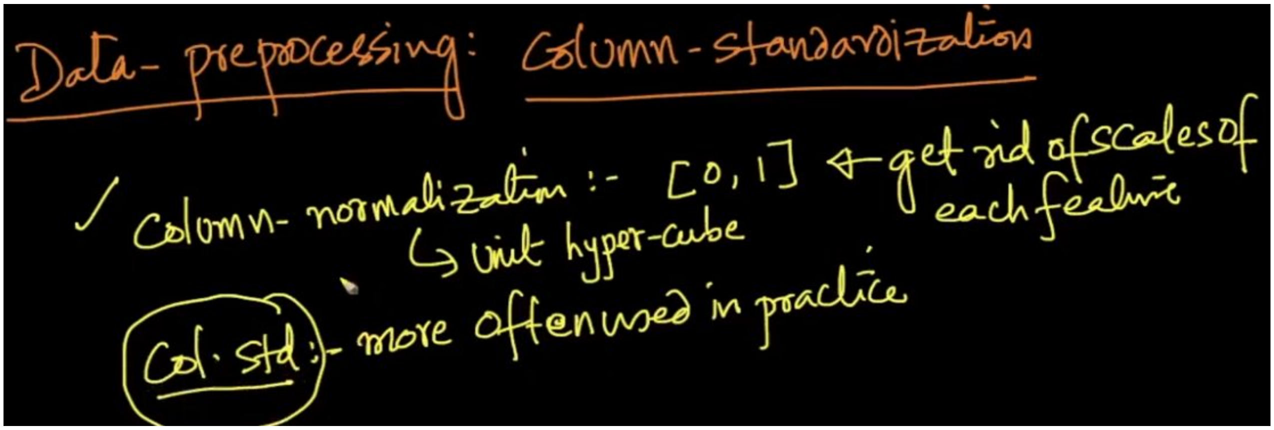

Here, x1 and x2 are two different vectors $2\,dimensions$ then component wise addition is done as shown and mean is calculated just like that.

**Geometric Intuition**

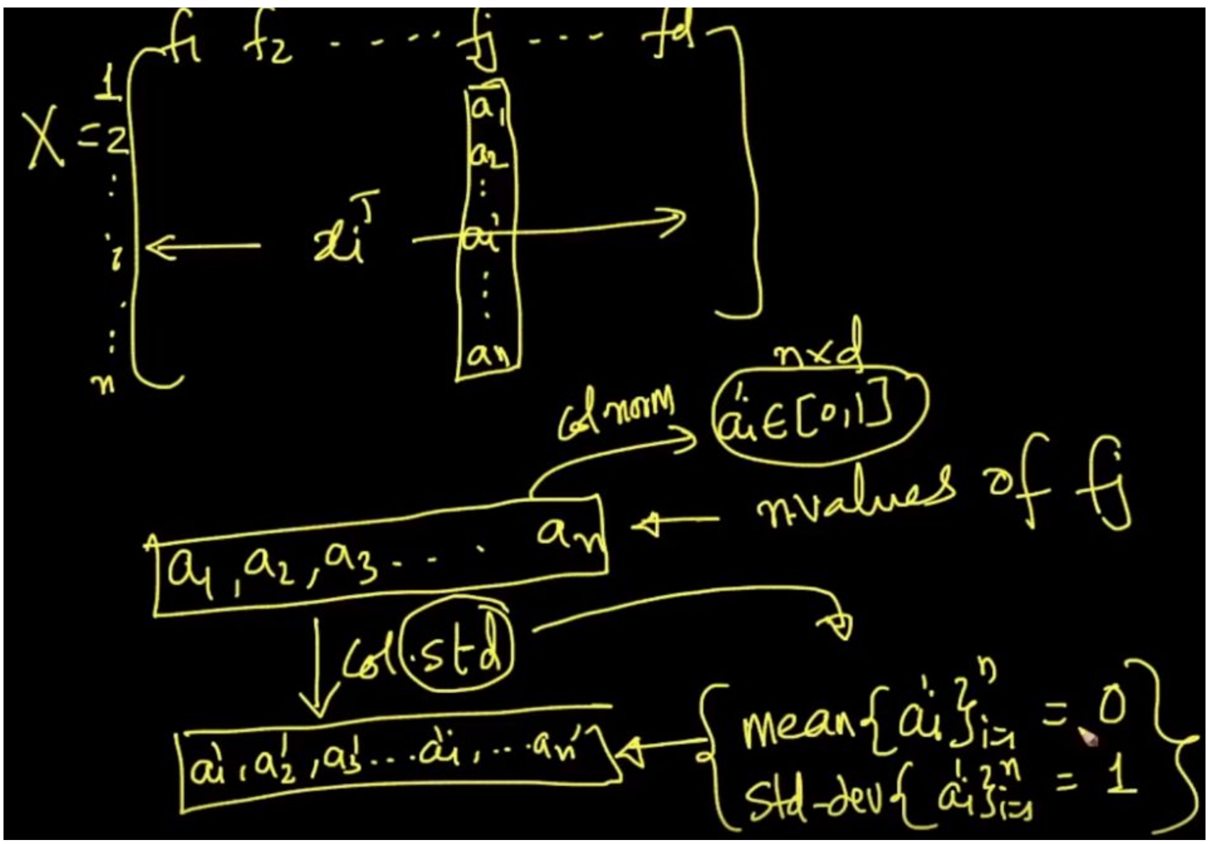



Mean vector ($\bar{x}$) is just the plotted vector the mean-height and mean-weight as shown above.Just like Mean is central value of all the values Mean vector is central vector of all the vectors.

## DATA PREPROCESSING : COLUMN STANDARDIZATION

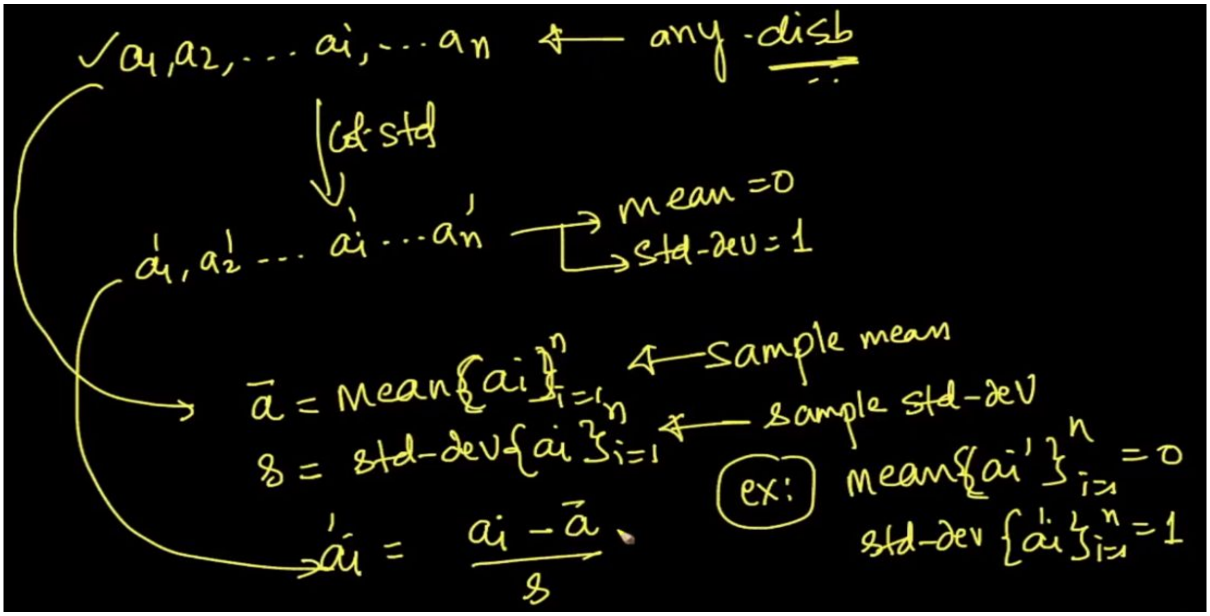



Column standardization is used more than column normalization because it can be used for distributions and much more statistical operations.



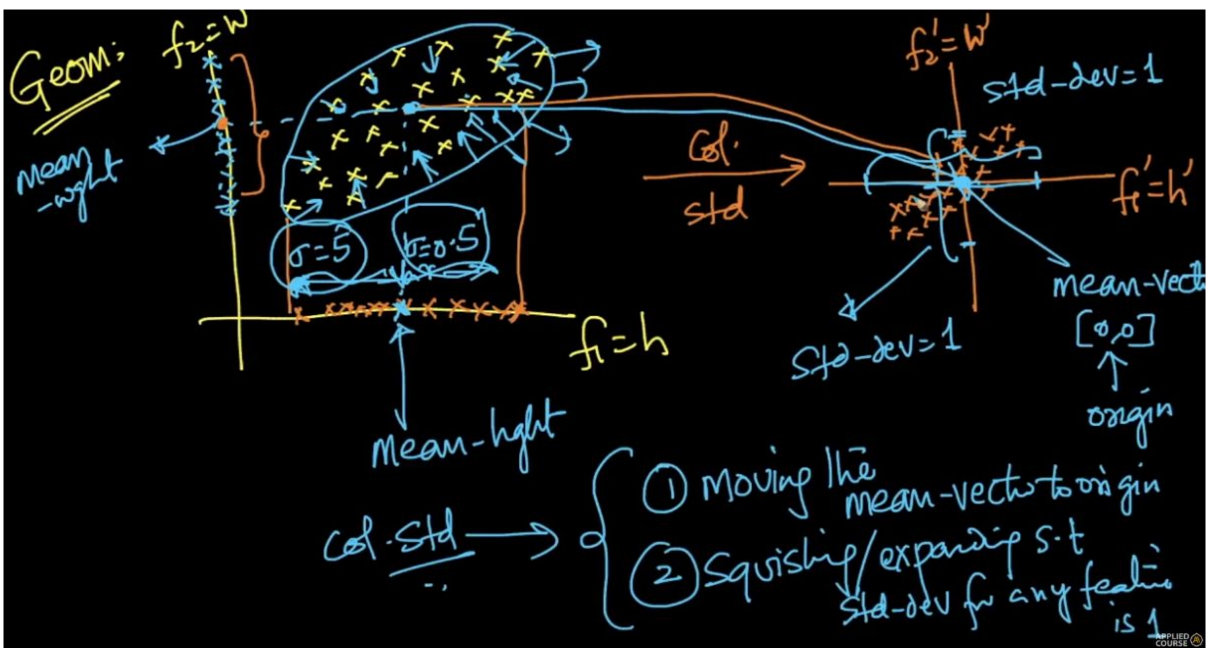

In column standardization, the columns/features are standardized such that the mean becomes 0 and std.dev = 1

**How to convert the array for standardization**



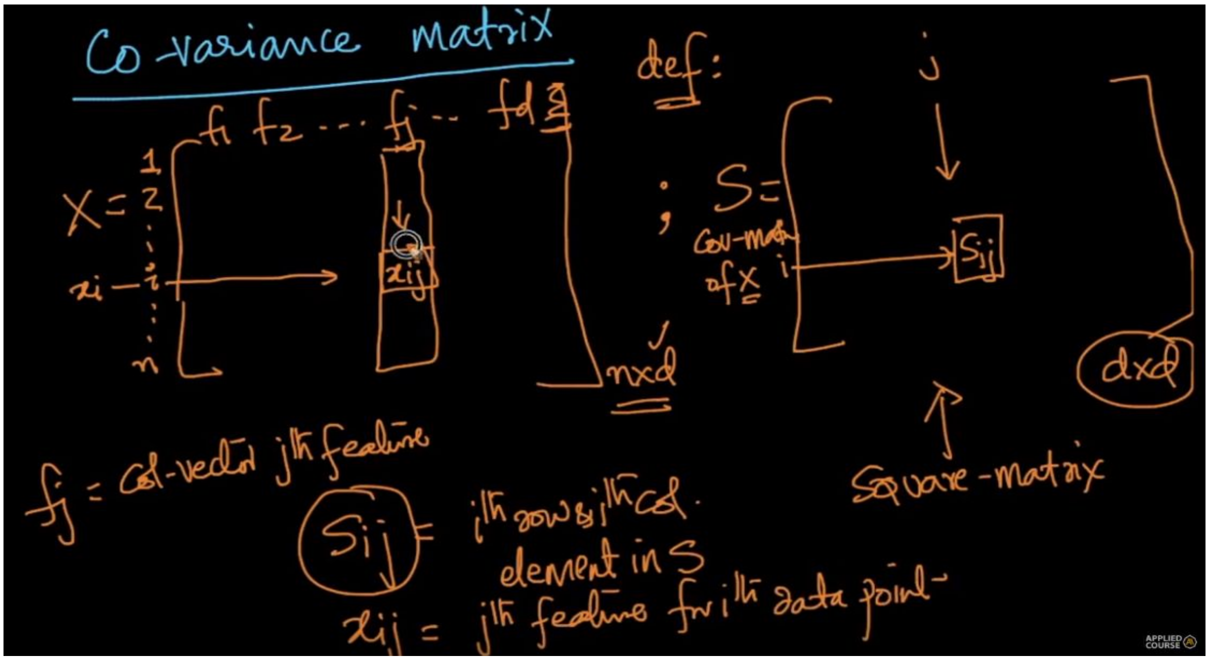

Calculate the mean($\overline{a}$) and std-dev(s) of the given feature $a_i$ and then $a_i' = \frac{a_i - \overline{a}}{s}$ and therefore the mean of $a'_i = 0$ and std-dev =1

**Geometric Intuition**

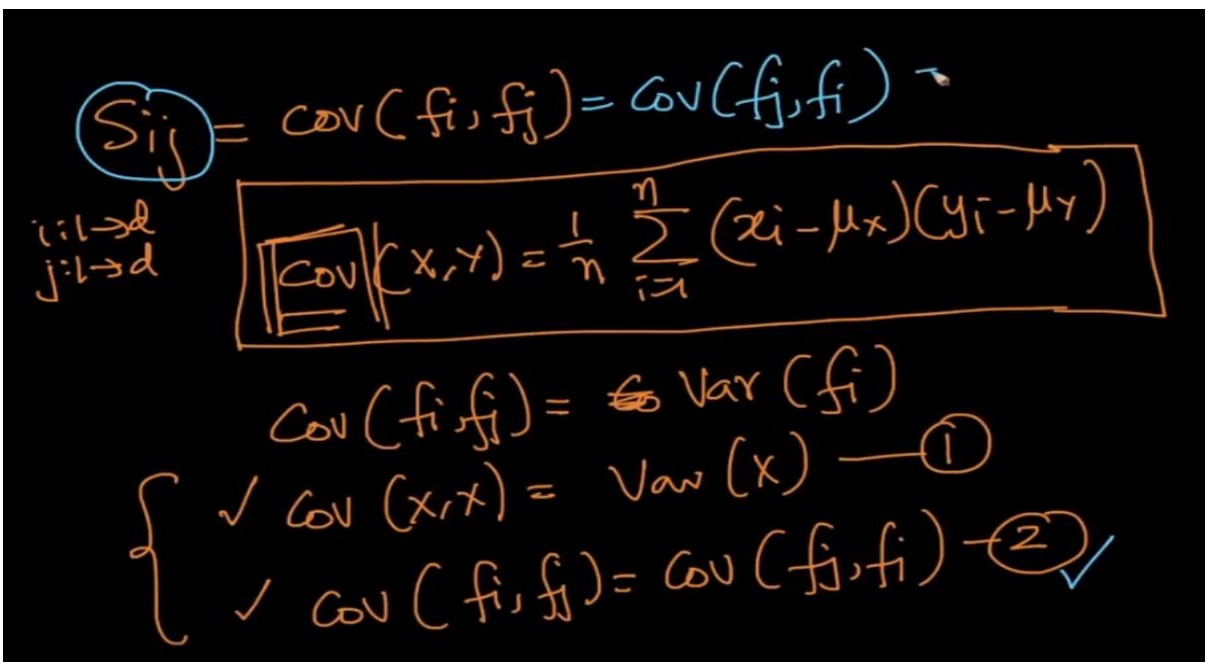



So basically what we are doing the mean vector is [0,0] and gets to the origin and we squish/expand the spread/std-dev is 1 as shown above

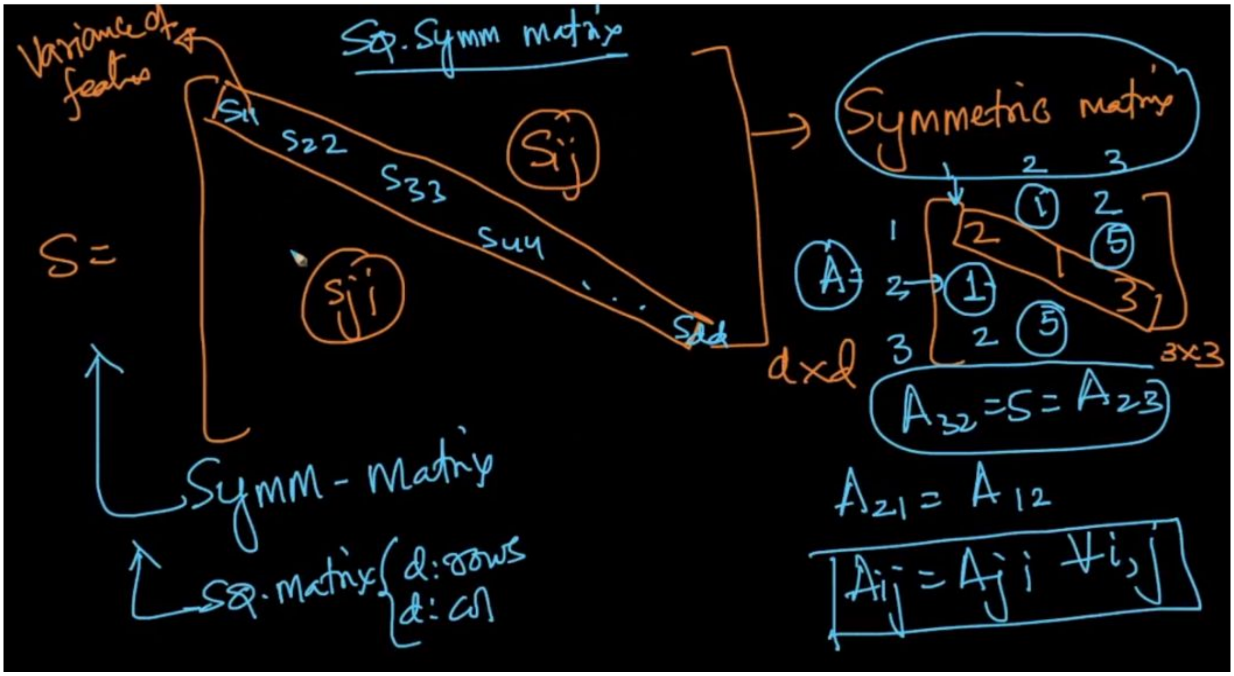
## Co-variance of a data Matrix




We defined a square matrix S of n*n dimensions. $S_{ij}$ & $x_{ij}$ are defined properly above.

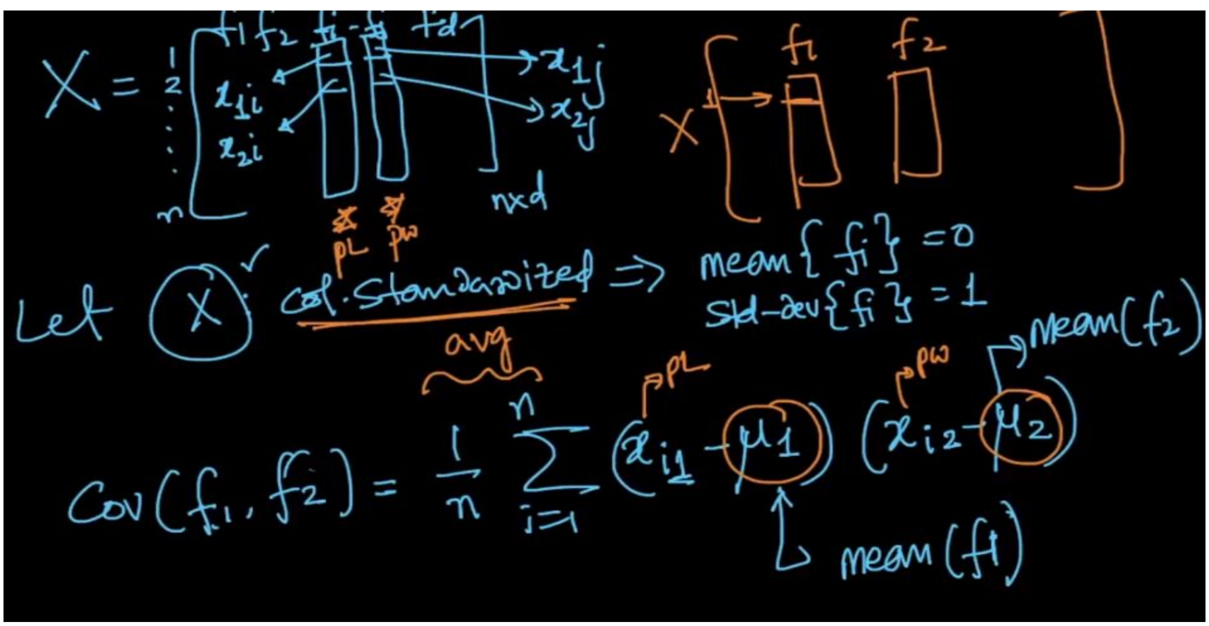



This is the covariance of $S_{ij}$ which is explained below and the properties are mentioned above.

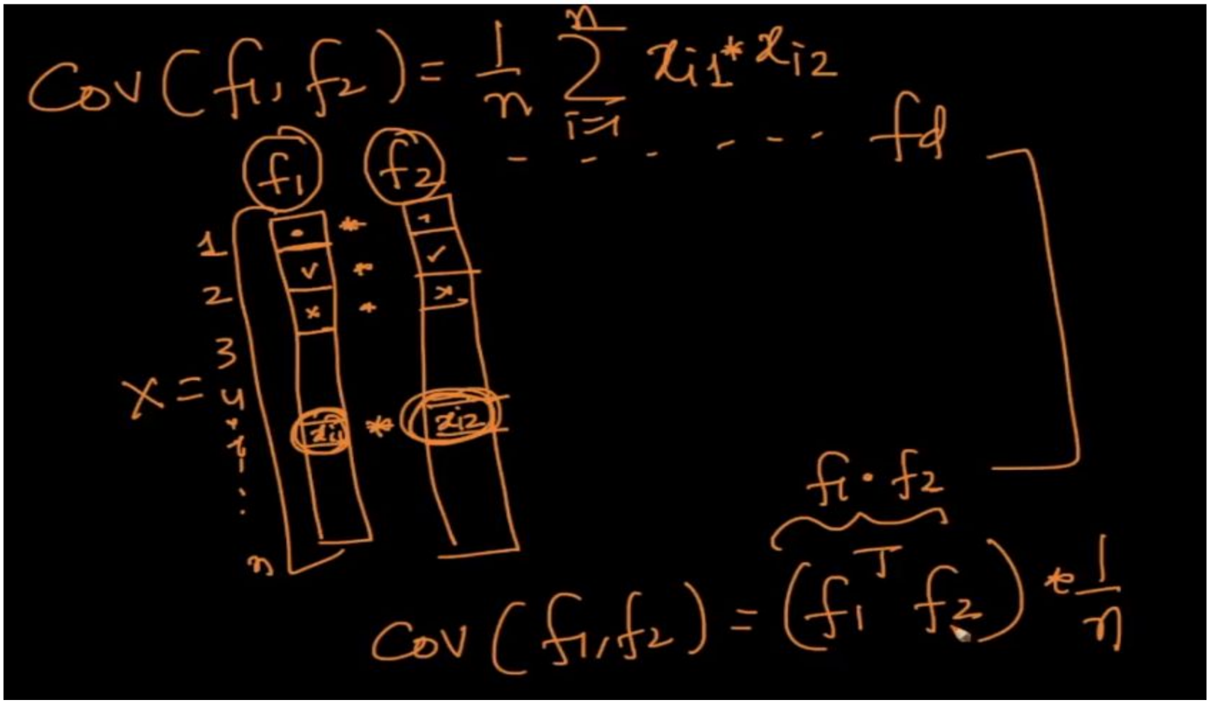



If $A_{ij} = Aji \ \ \forall i, j$ then it is a symmetric matrix as mentioned above. Our matrix S is also a symmetric matrix as well as a square matrix . Diagonal elements represents variance since Cov(X,X) =Var(X)

And the other elements represents covariance

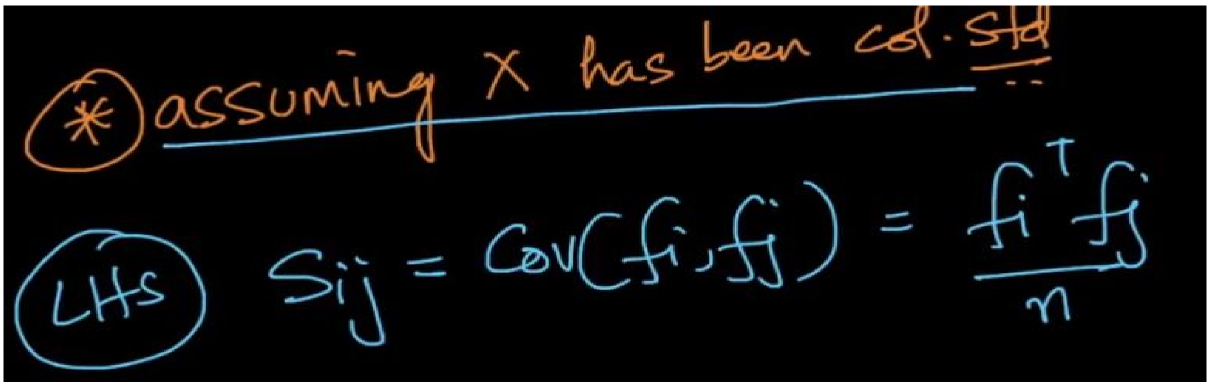


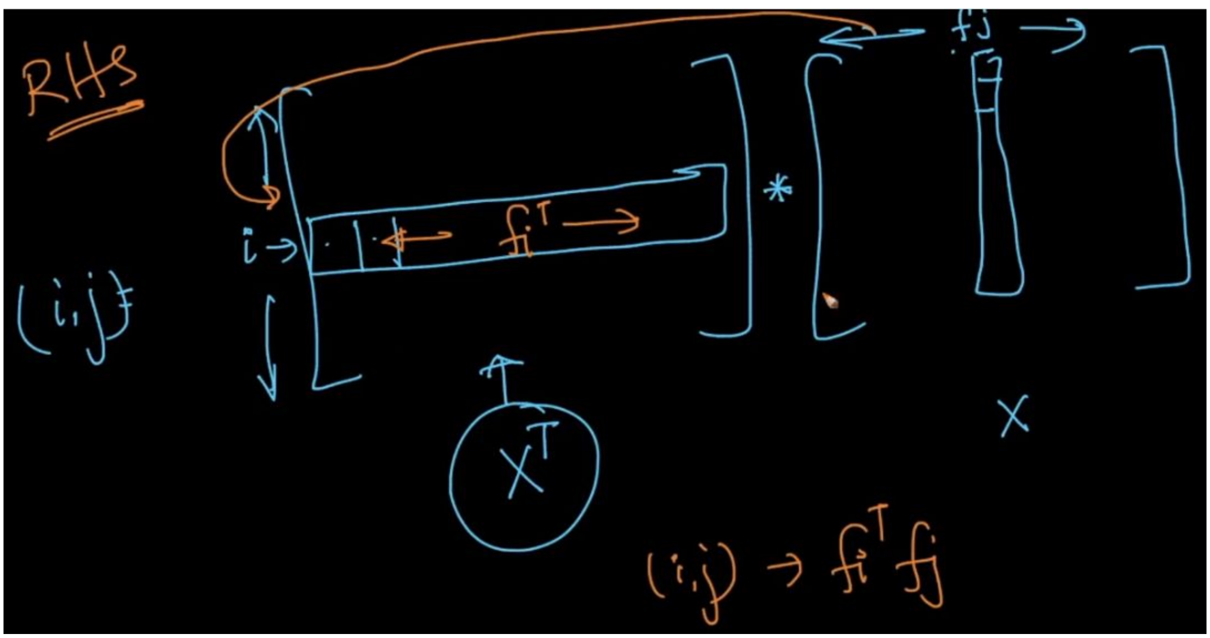
Covariance of two features ,f1 and f2. First we column standardize the X therefore their means becomes 0. Therefore, cov(f1,f2) = $\frac{1}{n}\sum_{i=1}^{n} x_{i1} * x_{i2}$



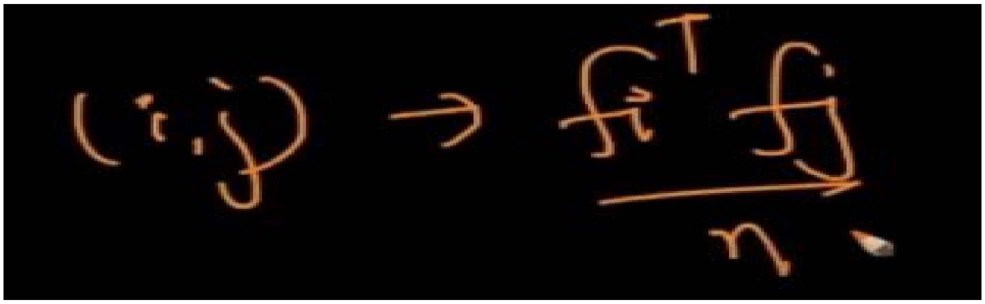

As it is seen , Cov is nothing but = $(f_1^T f_2) * \frac{1}{n}$ i.e f1.f2 where T is transpose.

We need to Prove that S = $\frac{1}{n}(X^T)$(X) which is d*d.
After col.std . LHS is the above and we know that.

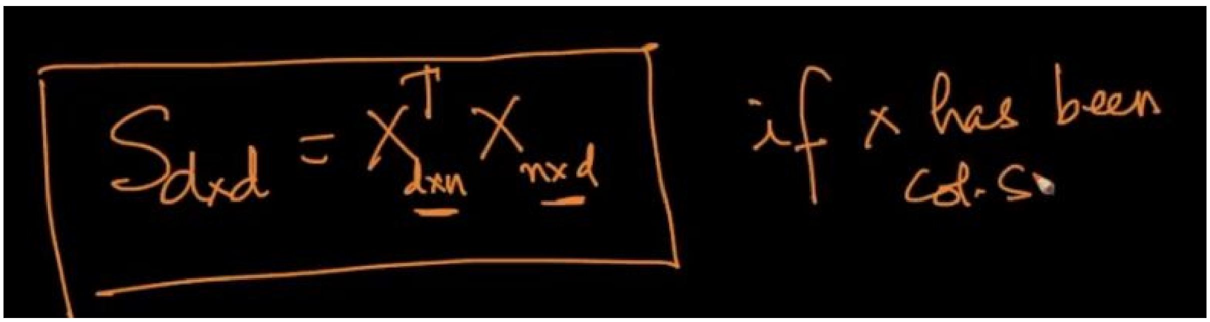


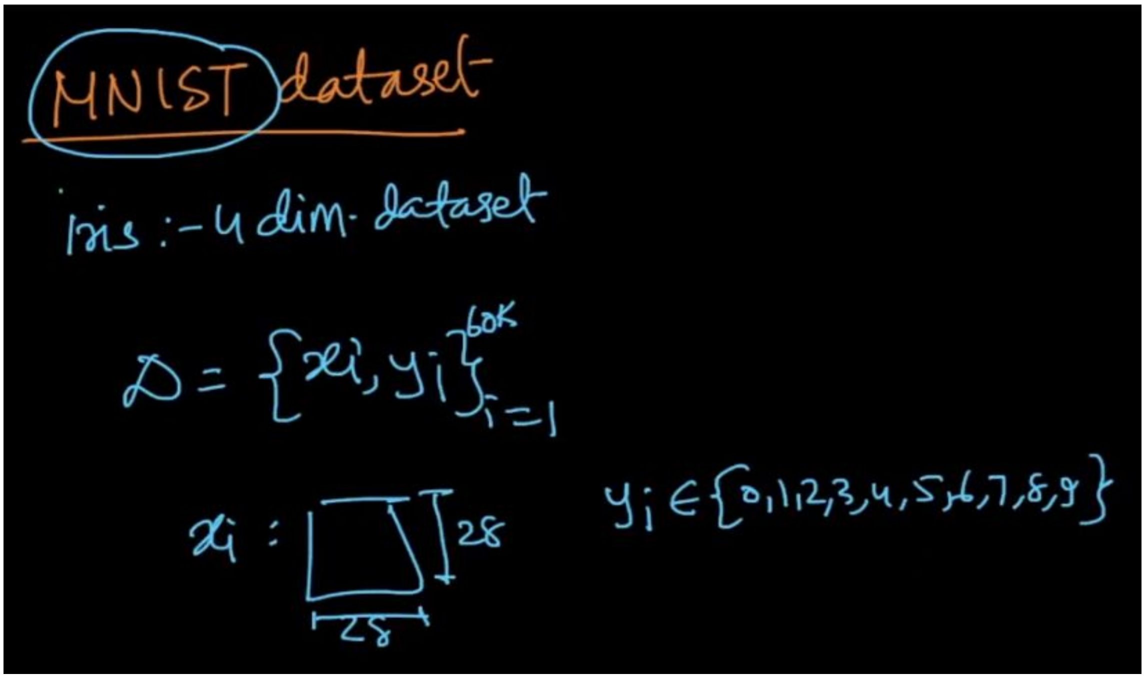

Here we are doing $X^T * X$ for ith row of $X^T$ and jth column of X. These i and j are nothing but the features $f_i$ and $f_j$ .



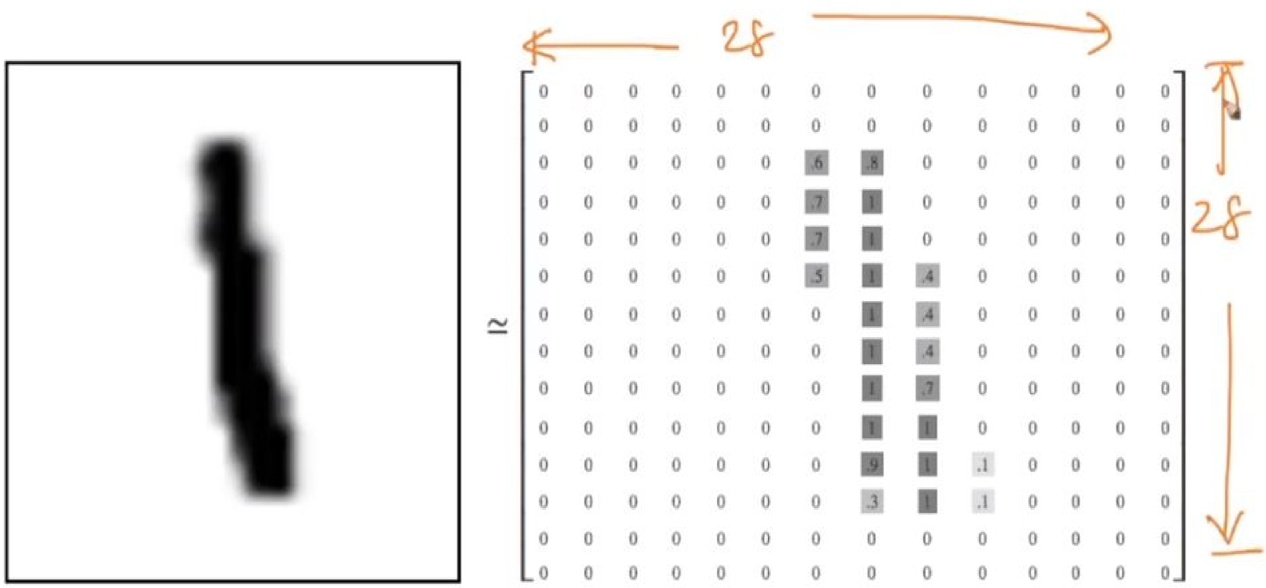

Therefore it is proved that LHS = RHS



## MNIST DATASET : Explanation

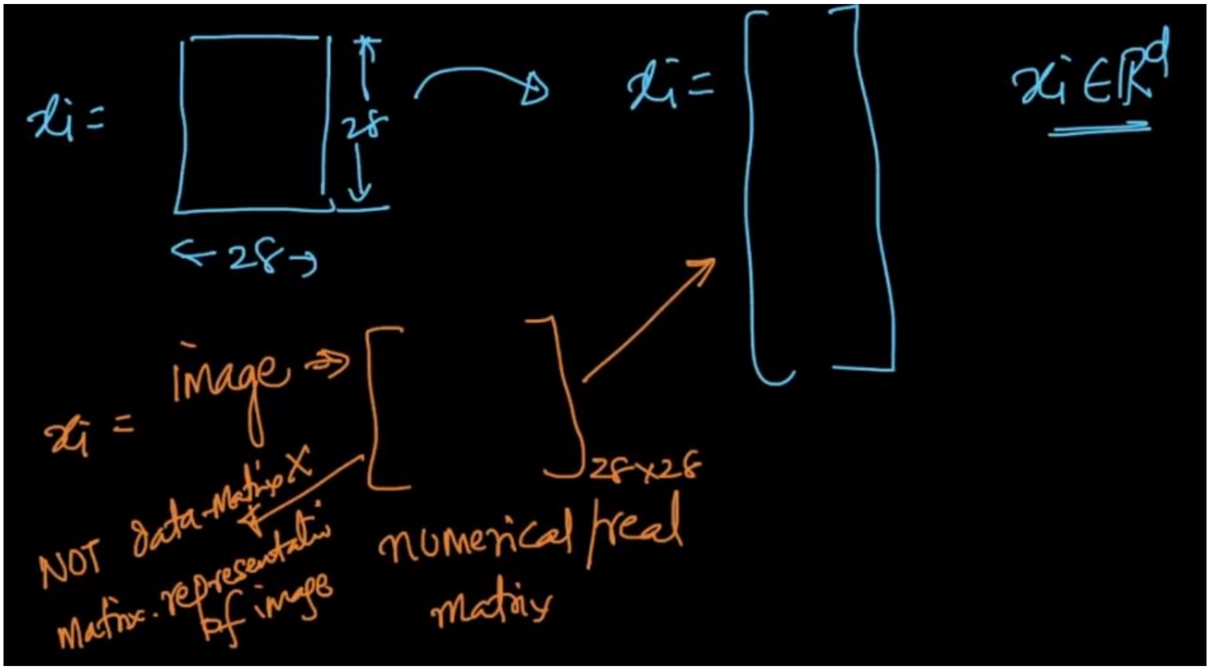



Here, in the MNIST dataset X is our 28*28 dataset. y is our label/target variable

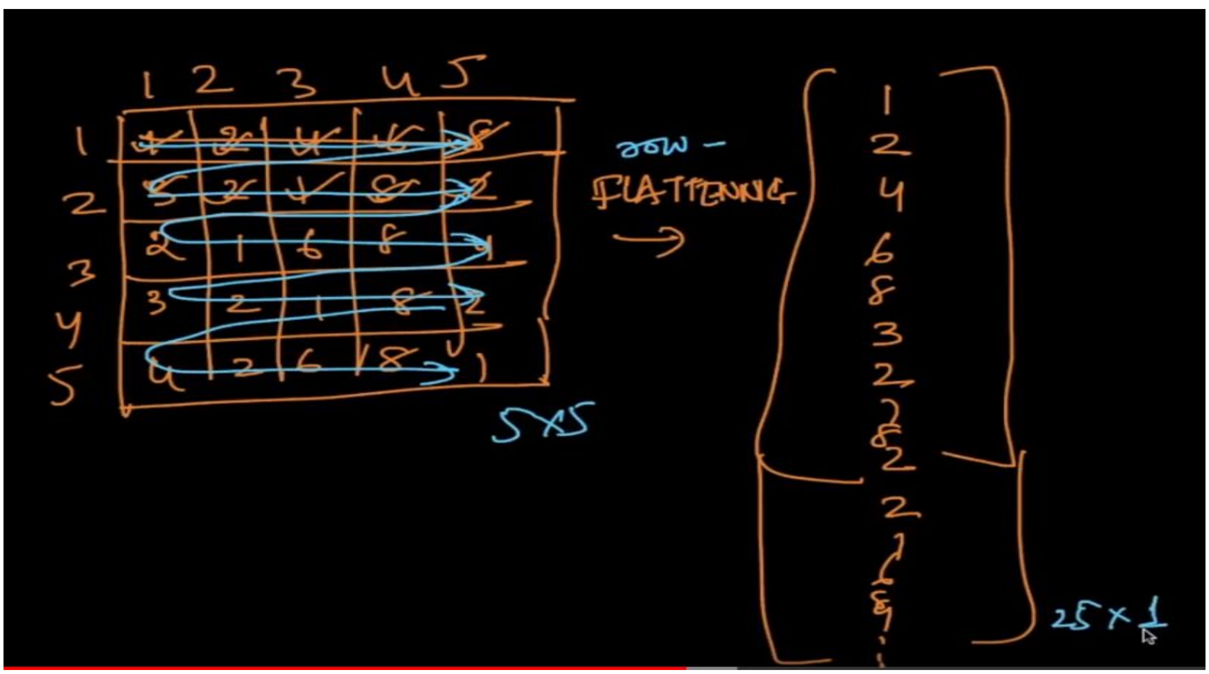



Where 1 is represented like this in a 28*28 matrix.

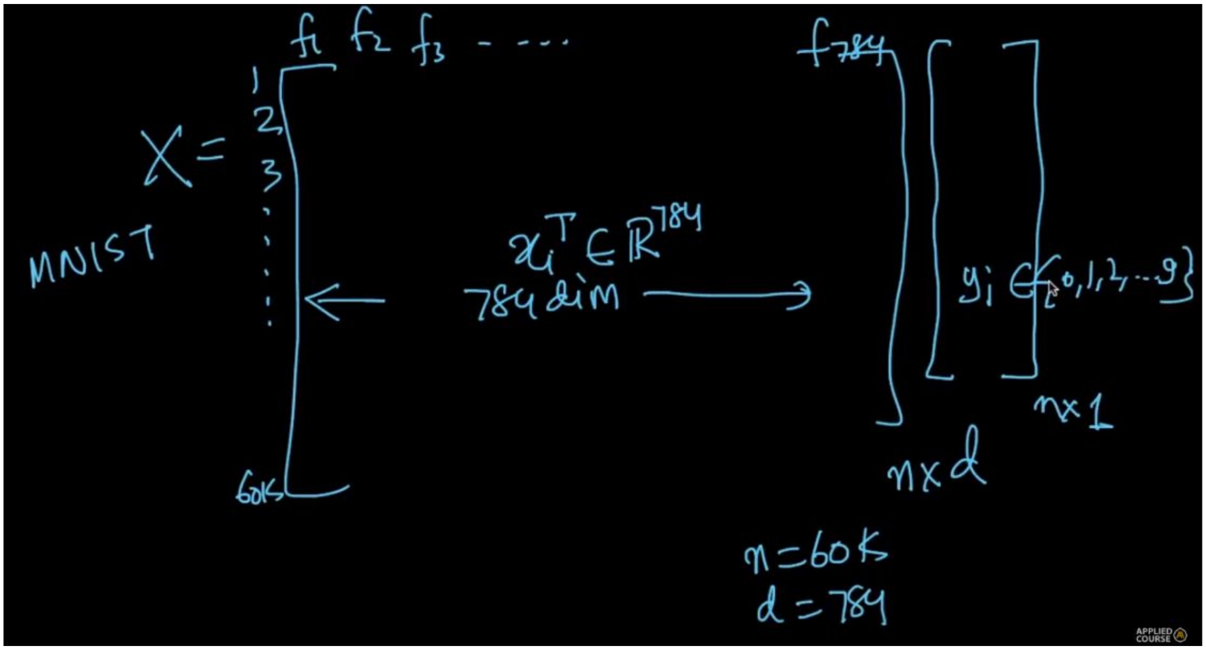
Now we need to convert this image matrix into a vector by using something called flattening



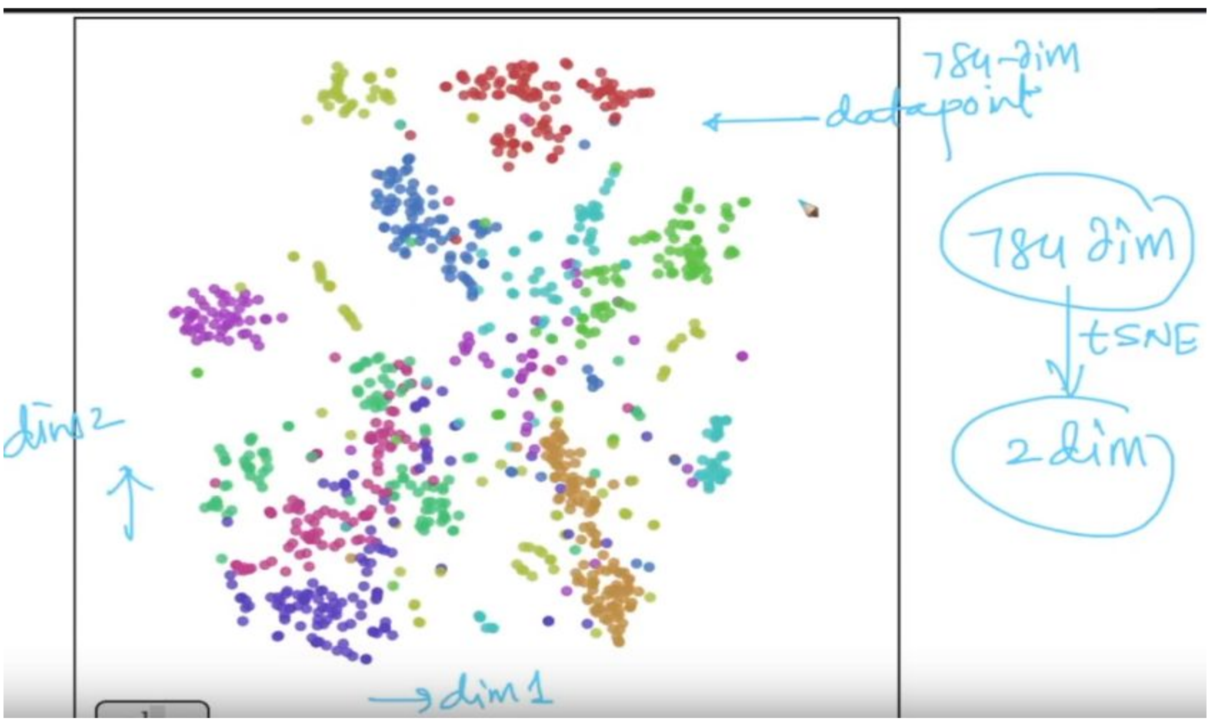

$x_i$ is just one data point and the 28*28 matrix is just 1 data point $x_i$ .LEt's see how to convert this matrix into a col vector




Let's say we've a 5*5 matrix as shown we flatten it by accessing each row and putting them in a single vector as shown .This is called row flattening which makes it 25 * 1.Our dataset has 28*28 therefore it becomes 784*1 vector and it represents the features of a single data point $x_i$




This is the representation of MNIST dataset, With 60k inputs/numbers and 784 dimensions




We can't visualize 784 dimensions therefore we use a dimensionality reduction technique called t-SNE to convert it into 2-d as shown.

[Visualizing MNIST]