

① Data ingestion

② EDA

③ Preprocessing | \Rightarrow feature engineering

④ Model building

⑤ Model evaluation

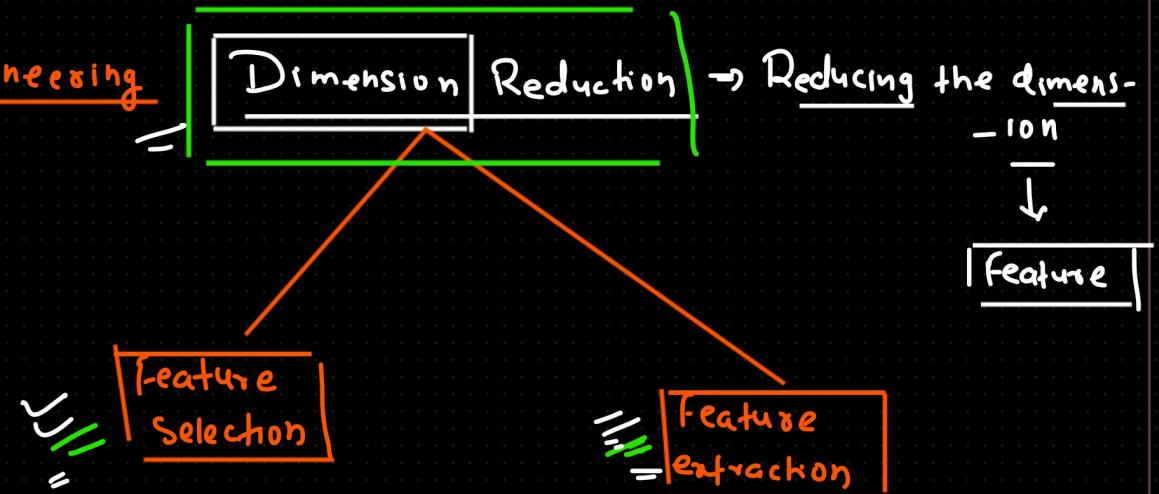
Acc to curse of dim.
"As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially."



Eg: Say, you dropped a coin on a 100-meter line. How do you find it? Simple, just walk on the line and search. But what if it's 100×100 sq. m. field? It's already getting tough, trying to search a (roughly) football ground for a single coin. But what if it's $100 \times 100 \times 100$ cu.m. space?! You know, football ground now has thirty-story height. Good luck finding a coin there! That, in essence, is "curse of dimensionality". In order to get rid of this using dimensionality reduction approaches like PCA we are basically trying reducing the dimensions in such a way that resultant (vector/set of dimensions) have essence of all the dimensions making it easier for ML algo to comprehend as volume or traversal space decreases.

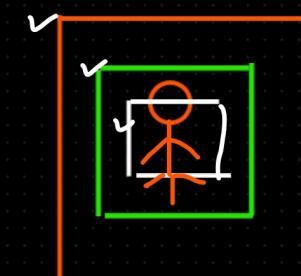
$$\left\{ \begin{array}{l} f_1 \dots f_{10} \rightarrow M_1 \Rightarrow 75 \cdot 1 \\ f_1 \dots f_{100} \rightarrow M_2 \Rightarrow 80 \cdot 1 \\ f_1 \dots f_{1000} \rightarrow M_3 \Rightarrow 72 \cdot 1 \end{array} \right. \begin{array}{l} \text{less} \\ \text{Optimal} \end{array}$$

Over \Rightarrow overfitting \Rightarrow sparsity
underfitting



Follow this beautiful article explaining this topic in most beautiful manner with set of examples:

[https://medium.com/flutter-community/curse-of-dimensionality-an-intuitive and-practical-explanation-with-examples-399af3e38e70#:~:text=Example%20they%20can%20move%20in.](https://medium.com/flutter-community/curse-of-dimensionality-an-intuitive-and-practical-explanation-with-examples-399af3e38e70#:~:text=Example%20they%20can%20move%20in.)



1

Feature Selection \Rightarrow [Subset of the feature]

$$f_1 \dots f_{100} \Rightarrow \boxed{\text{Model}} \Rightarrow \underline{\text{Feature}}$$



50, 60, 70, 80

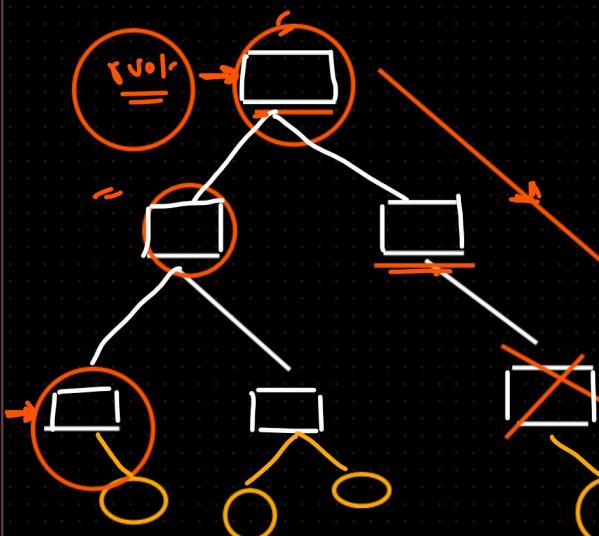
$$\underline{\underline{\underline{\underline{\underline{\underline{\underline{\{1,2,3,4,5,6,7\}}}}}}}}$$

$$\downarrow \\ \underline{\underline{\underline{\underline{\underline{\underline{\underline{\{1,2,3\}}}}}}}}$$

Sel
theory

Scikit learn

Below are the methods that can be used to perform feature selection:



In DT root level has high importance as it has largest info gain

Google feature importance method

① Filter method \Rightarrow Corr_{correlation}

Info gain \Rightarrow feature importance

② Wrapper method.

③ LASSO (Regularization based method)



VV Imp:

Google scikit learn feature selection.

Using this we can implement different feature selection approaches.

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_selection

Wafer fault Detection



= foot

= feature Selection

Can follow this blog as well for feature for feature selection

methods available in sklearn lib:

<https://towardsdatascience.com/5-feature-selection-method-from-scikit-learn-you-should-know-ed4d116e4172>

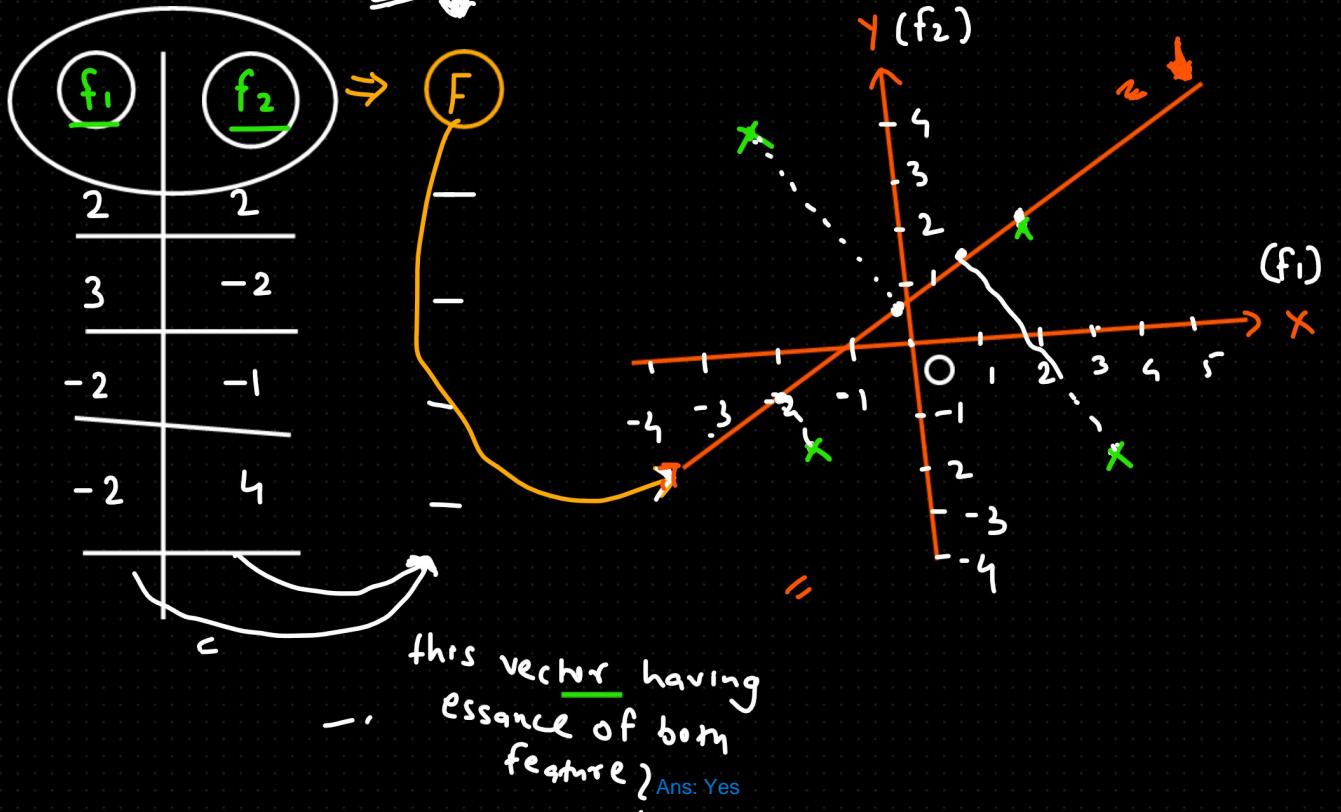
Please note that having more and more number of features will definitely help in building a model having high accuracy. But beyond some threshold on increasing number of features further will result into curse of dimensionality which will increase the volume and sparsity. To get rid of this we reduce number of features by taking a subset of features which are important(feature selection) or by reducing the dimension in such a way that the resultant reduced dimensions have essence of whole dataset(feature extraction)



In this example through feature extraction we are reducing 2 dimension data(f_1 and f_2) into 1 dimension by projecting them on vector F such that this vector F have essence of both the features(f_1 and f_2)

Dimension reduction or feature extraction can be used to:

1. To get rid off the curse of dimensionality problem.
2. To visualize higher dimension dataset. For Eg; For visualizing dataset with 4 features we can first convert into 3 dimension using dimensionality reduction approaches such as PCA followed by visualizing resultant 3D dataset.



Following are some of the latest techniques used for dimensionality reduction or feature extraction:

- 1 PCA
 - 2 t-SNE
 - 3 LDA
- } State of art
(State of art means latest)

- 1 Standardized the Data
- 2 Covariance matrix
- 3 Eigen value and eigen vector
- 4 Principle Component

PCA

These are the 4 steps in which PCA is implemented

Step 1: Scaling of the data

$$\begin{aligned} \text{Min-Max} &\Rightarrow [0-1] \\ \text{Standard Scalar} &\Rightarrow \\ Z\text{-Score} & \end{aligned}$$

Standardization of data can be done using any of the below approaches:

\downarrow This is the general formula for Min Max scaling where X represents the data point.
Also, we may customize this formula according to our need.

$$\frac{\text{Min} - \text{Max}}{X - \text{Min}} = \frac{X - \text{Min}}{\text{Max} - \text{Min}}$$

In min-max based approach we suppress the dataset under min and max range such that all the data points after scaling lie in the interval [0,1]

$$\frac{[a_1, a_2, a_3, a_4, \dots, a_n]}{\downarrow \quad \quad \quad \downarrow}$$

Min Max

In this example
 $a_1 == \text{min}$
 $a_n == \text{max}$

scaling min data point a_1

$$= \frac{M/n - M/m}{M - m} = 0$$

scaling max datapoint a_n

$$= \frac{M/m - M/n}{M - m} = 1$$



This plot shows how a unscaled dataset get scaled under 0 and 1 after applying min max scaling

SND
Standardization \Rightarrow
$$\frac{x - \mu}{\sigma}$$

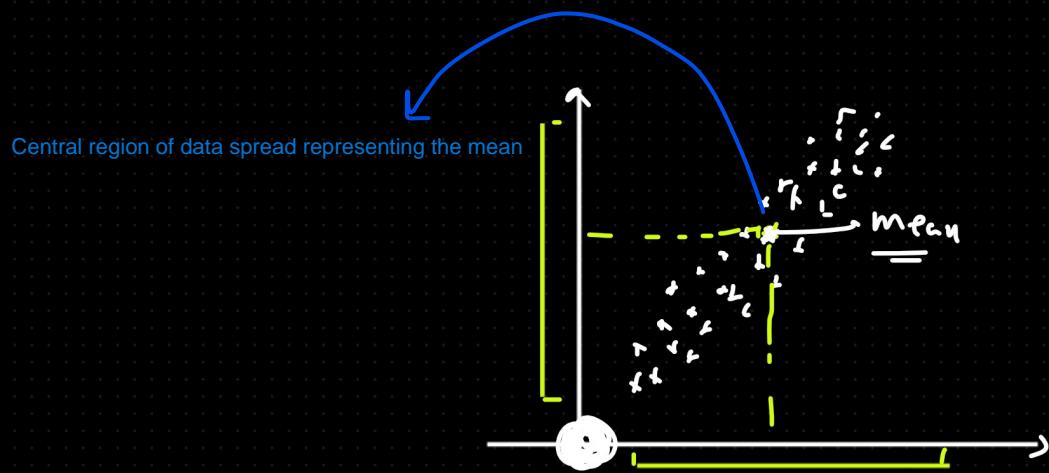
Z-Score

$$\left\{ \begin{array}{l} \mu = \text{Mean} \\ \sigma = \text{std} \end{array} \right\}$$

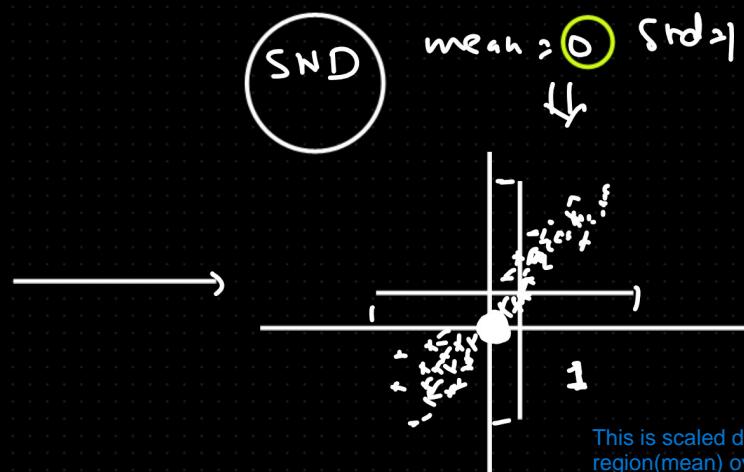


In Standard scaler or Z score based approach we perform standardization on the dataset (with mean and variance not equals to 0 and 1 respectively) in such a way that the resultant standardized dataset have mean equal to zero and standard deviation = 1

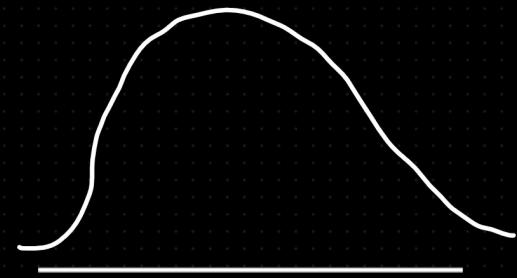
Data \rightarrow standardization \Rightarrow Standard Data
 $\left\{ \begin{array}{l} \mu = 0 \\ \sigma = 1 \end{array} \right\}$



This is the dataset with mean and variance not equals to 0 and 1 respectively



This is scaled dataset with central region(mean) over origin and closely or tightly packed(variance =1)



Step 2:

Variance \Rightarrow

$$\bar{x} \text{ } 2, 8, 10, 12, 6, 5, 13, 14$$

Variance is the spread of data around the mean

Spread of the data around the mean

Mean

$$\bar{x}$$
 deviation $\Rightarrow (\text{Mean} - \text{Df})^2$

mean of deviation

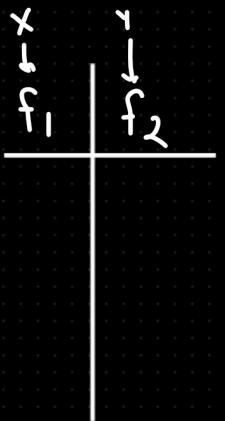
Follow this flow to remember the Variance formula in easy way

$$\left[\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2 \right]$$

Formula of variance

Variance \rightarrow

Cov-variance



X, X

$$\left[\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y}) \right] \leftarrow \frac{\text{Cov}(X, Y)}{\text{Cov}(X, X)}$$

$$\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(x_i - \bar{x}) \quad \frac{\text{Cov}(X, X)}{\text{Cov}(X, X)}$$

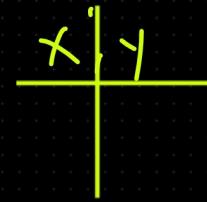


$$= \left[\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 \right]$$

Relationship b/w variance and covariance w.r.t a random variable X
Variance of X == Co variance of(X,X)

Covariance is commutative in nature

$$\text{Cov}(x, y) = \text{Cov}(y, x)$$



$$= \begin{bmatrix} \text{Cov}(x, x) & \text{Cov}(x, y) \\ \text{Cov}(y, x) & \text{Cov}(y, y) \end{bmatrix} \equiv \Rightarrow \text{Covariance matrix of } x, y \text{ feature}$$

↓
I Std mean = 0 ;
↓
 $\Rightarrow \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})$
mean mean

Why we do Standardization?

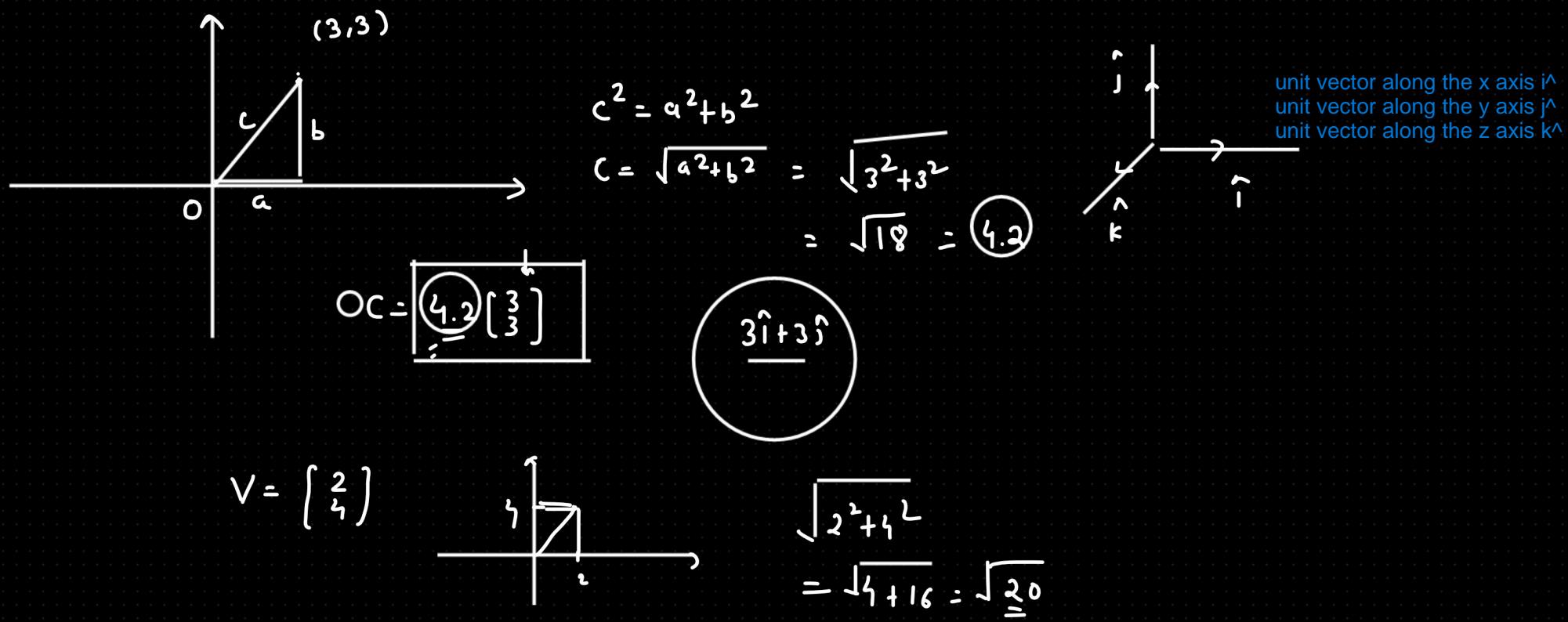
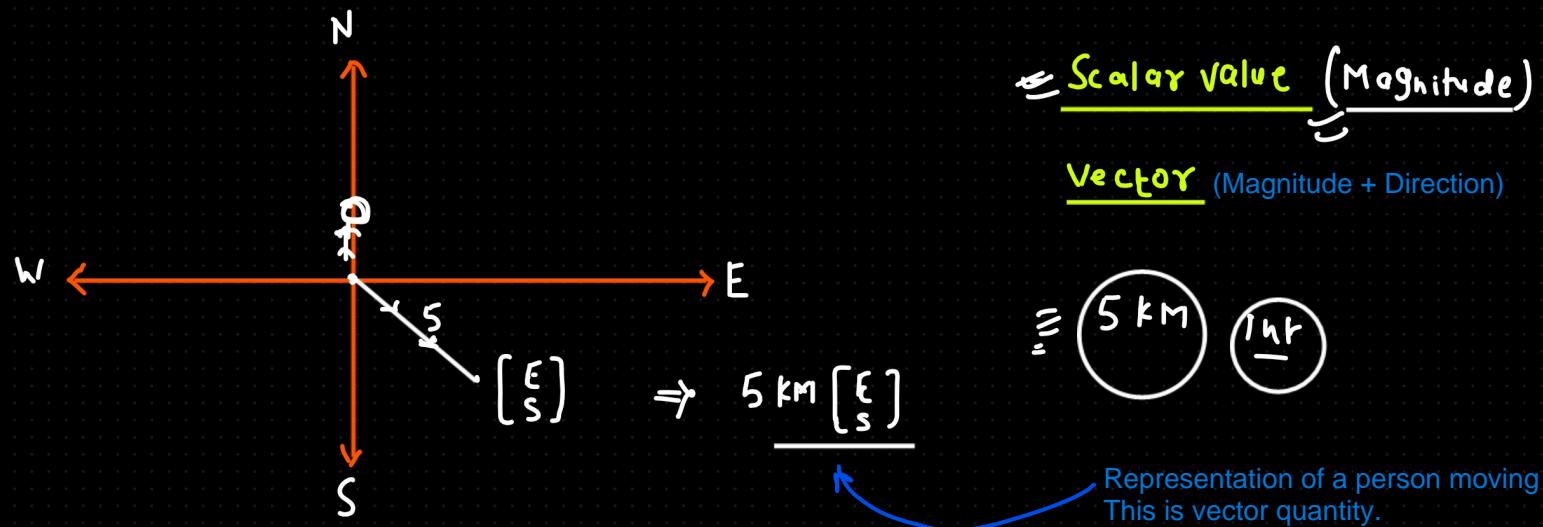
Since in the Standardization step we are making the mean 0. So, our cov / Var formula is also getting updated(since mean = 0) that is less calculation. This is why we are doing the Standardization which is kind of optimizing the model learning phase.

$$\Rightarrow \frac{1}{N} \sum_{i=1}^N (x_i)(y_i)$$

$$\frac{x_1 y_1 + x_2 y_2 + x_3 y_3 + \dots + x_n y_n}{N}$$

Considering X and Y are 2 vectors then the numerator is dot product of these 2 vectors.

Dot Product - of two
vector



Step 3:

Eigen value & eigen vector

We are using cov matrix since it is square matrix that will be used for calculating eigen vector

$$AV = \lambda V$$

A = Square Matrix

V = Vector

λ = Scalar value

Please note matrix can have multiple rows and columns but vector can have multiple rows and only 1 column.

If we multiplying a square Matrix A with vector V and if we get a new vector that is λ times of vector V , then vector V is called eigen vector of Matrix A and λ is called the eigen value

Also,

$= \left\{ \begin{array}{l} \text{if we multiply square matrix } A \text{ by vector } V \\ \text{the new vector } (\lambda V) \text{ does not change the direction. than only if it is eigen vector} \end{array} \right.$

Example 1:

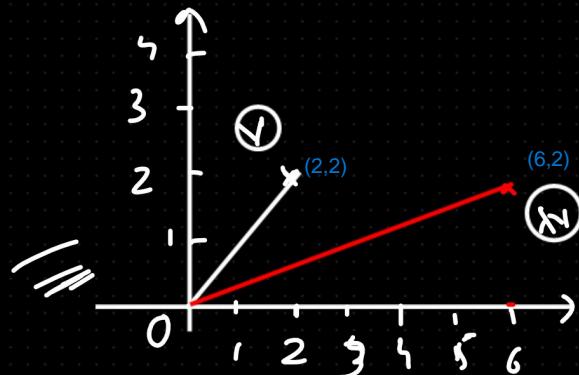
$$A = \begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$AV = \lambda V$$

$$\begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 2 \\ 2 \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} 1 \times 2 + 2 \times 2 \\ 1 \times 2 + 2 \times 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 2+4 \\ 2+0 \end{bmatrix} \Rightarrow \begin{bmatrix} 6 \\ 2 \end{bmatrix}$$



From above plot we can say that both the vectors are having different direction since they are not coinciding over each other. If they were coinciding over each other or on the same line then we would have said that both are having the same direction but different magnitude where respectively direction would have represented eigen vector whereas, respectively magnitude would have represented the eigen value.

$$AV = \begin{bmatrix} 6 \\ 2 \end{bmatrix}$$

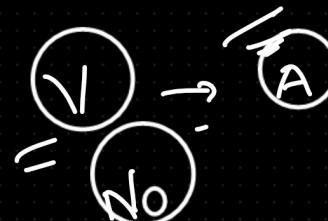
$$\begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \end{bmatrix}$$

unit movement along x axis = $i^{\wedge} = 6$

unit movement along y axis = $j^{\wedge} = 2$

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

$$A \begin{pmatrix} V \\ V \end{pmatrix} = \lambda \begin{pmatrix} V \\ V \end{pmatrix}$$



It is clear that direction has changed after A is multiplied with V. If both were equal then we would have said that same direction. Same can be also visualized through graphical representation

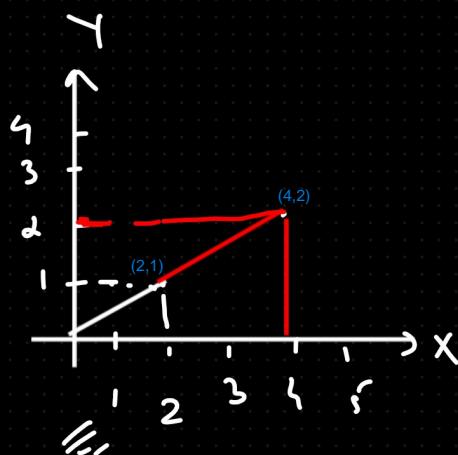
In this example $AV=\lambda V$ but since direction is changing V cannot be considered as Eigen value and λ cannot be considered as Eigen value.

Example 2:

$$\begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \times 2 + 2 \times 1 \\ 1 \times 2 + 0 \times 1 \end{bmatrix} = \begin{bmatrix} 2+2 \\ 2+0 \end{bmatrix} \Rightarrow \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

λV

$$| A \times V = \lambda \times V |$$



$$\begin{bmatrix} 1 & 2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

$\lambda = \text{Eigen value}$

Are these vectors eigen vectors of matrix A ?

Here since both the vectors are coinciding over each other or are on the same line so we will say that both are having the same direction.

| Principle Component |

In this example since $AV=\lambda V$ and direction is same for vector V in the process so V will act as Eigen vector and λ will be considered as Eigen value.

Eigen value tells us how much the eigen vector changes in size when we multiply it(Eigen vector) with ^{square} matrix.

$$A \rightarrow v$$

$$\lambda v$$



$$Av = \lambda v$$

$$Av - \lambda v = 0$$

$$(A - \lambda I)v = 0$$

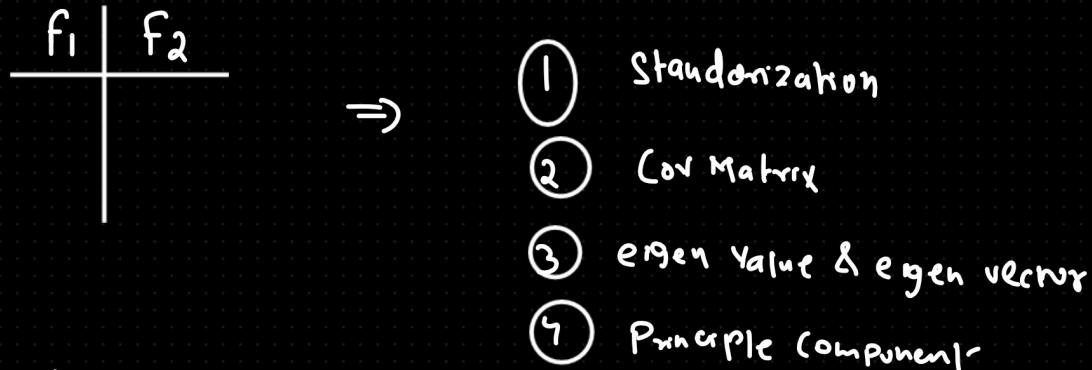
$$\text{Det } |A - \lambda I|$$

I = Identity matrix

We can say that this is the Eigen equation

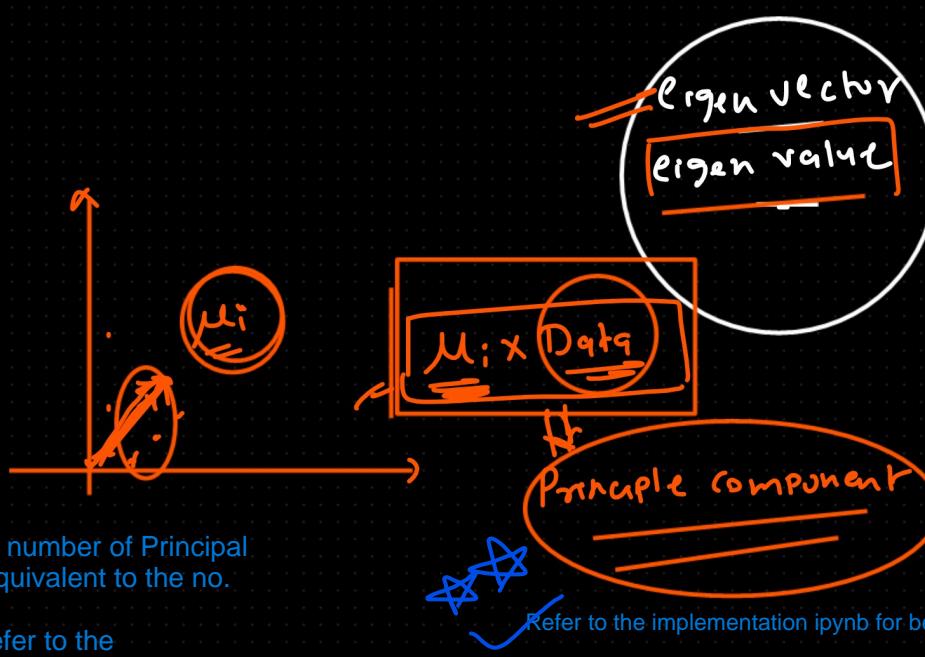
Please note that here we have multiplied scalar lambda with Identity matrix so that to convert it into matrix, later which can undergo like term subtraction with square covariance matrix A

Also, note that number of elements in eigen vector=number of elements in eigen values=no. I/P feature.
Please refer to the ipynb implementation for the better understanding.



Step 4:

u_i is the eigen vector which when multiplied with the datapoints(obtained after scaling) gives principal component which will project all the datapoints on the same vector resulting in dimensionality reduction.



This implies that the total number of Principal component axis will be equivalent to the no. of features in input scaled dataset. Please refer to the implementation ipynb for better understanding.

Important: Refer to implementation ipynb for better understanding

While training the model we will be providing the PCA transformed dataset and while prediction we will first of all transform new data on which prediction needs to be made into PCA transformed data and then use the existing learning (trained over PCA transformed train dataset) to carry out the prediction.

Means for any new data set that is not the part of the test split of existing dataset then we need to prepare a data pipeline that we convert this new data in PCA transformed data which will be then passed to the model for carrying out the prediction.

Again please note PCA is applicable only for numerical columns and not for categorical columns. Even if we are transforming categorical col into numeric using then it will generate sparse matrix(0s and 1s) will not be taken forward for PCA

Apply PCA on numerical col and encoding on categorical col and combine to generate the final dataset.

In Feature Selection we are taking the subset of the features whereas, in Feature Extraction we are producing vector having essence of whole dataset with all the features