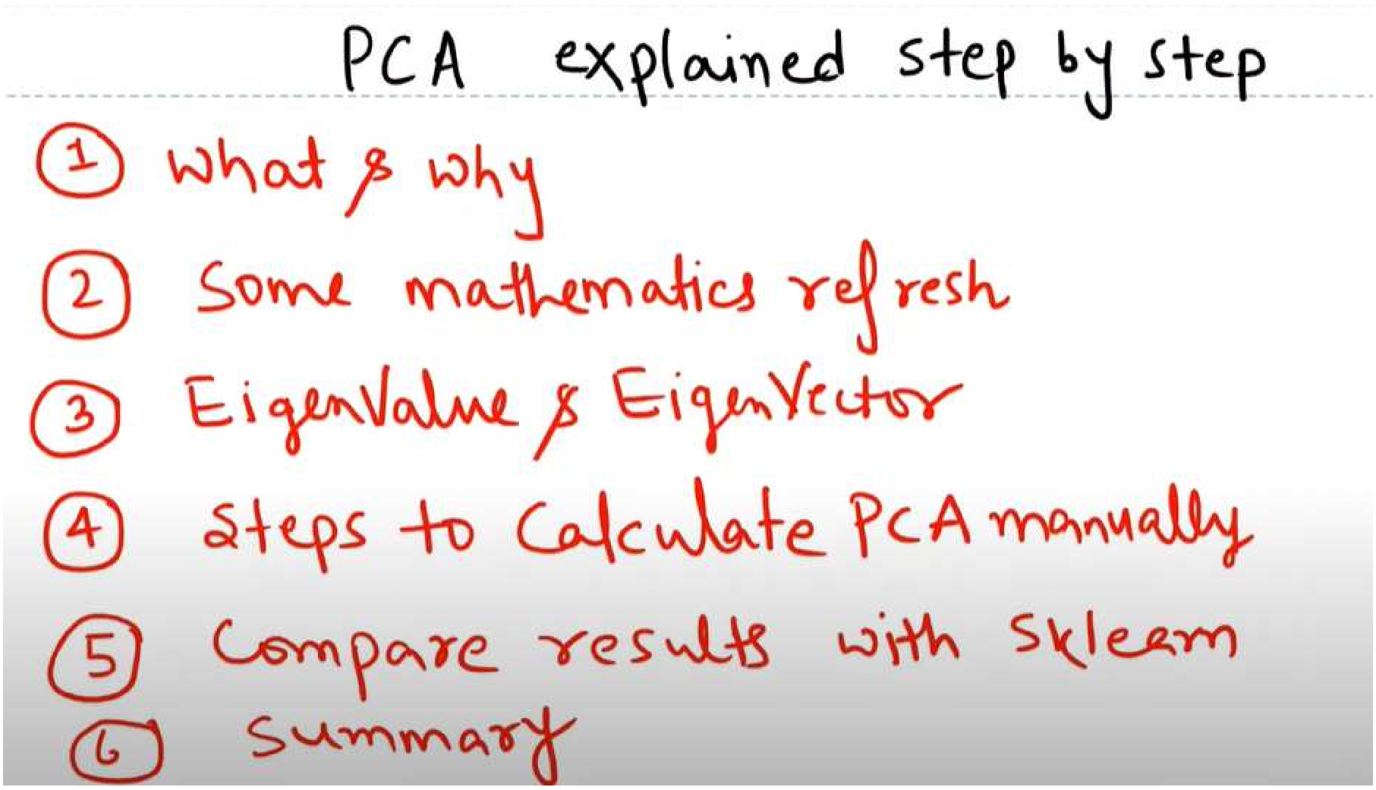
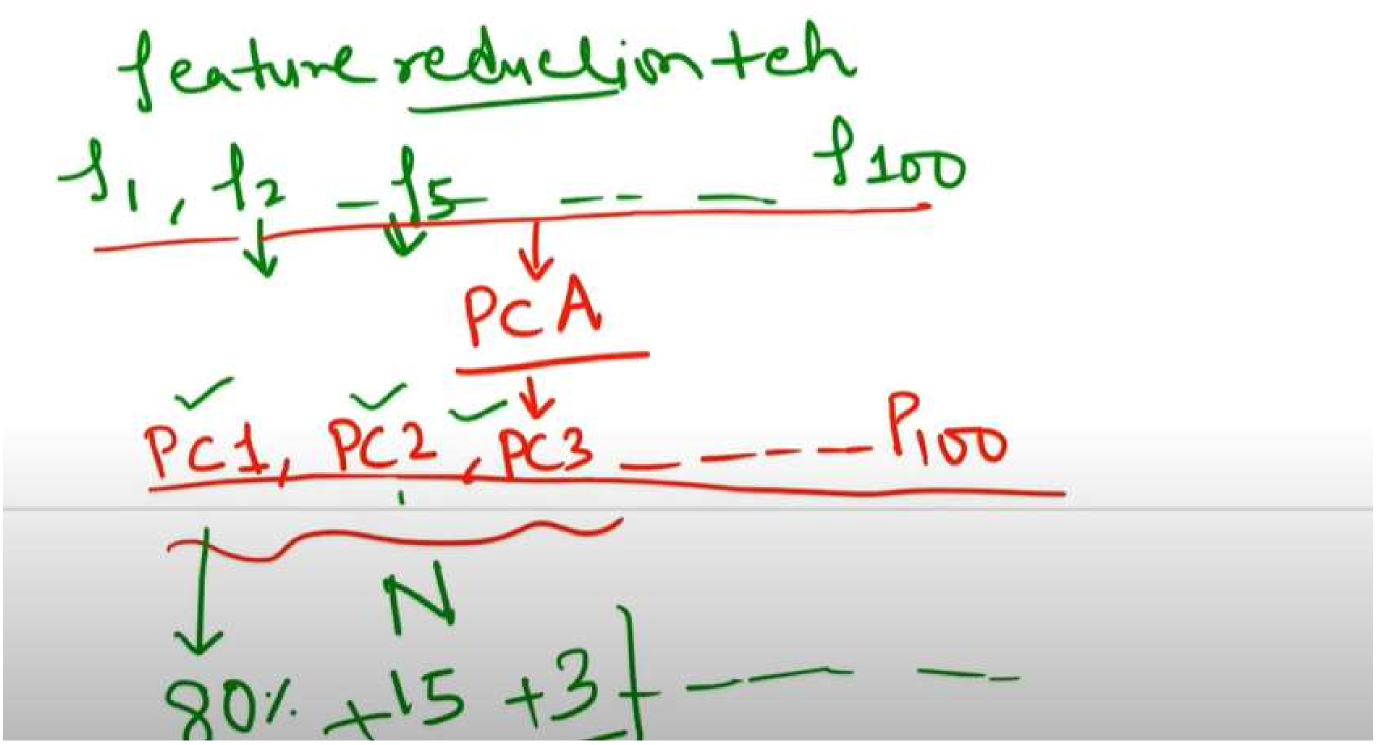
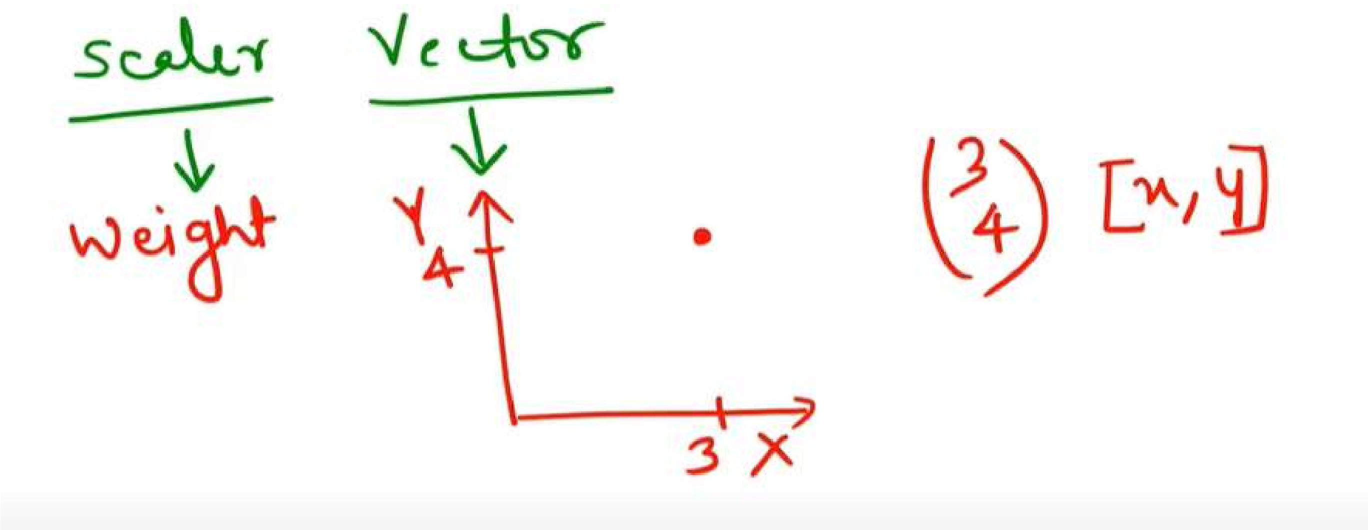
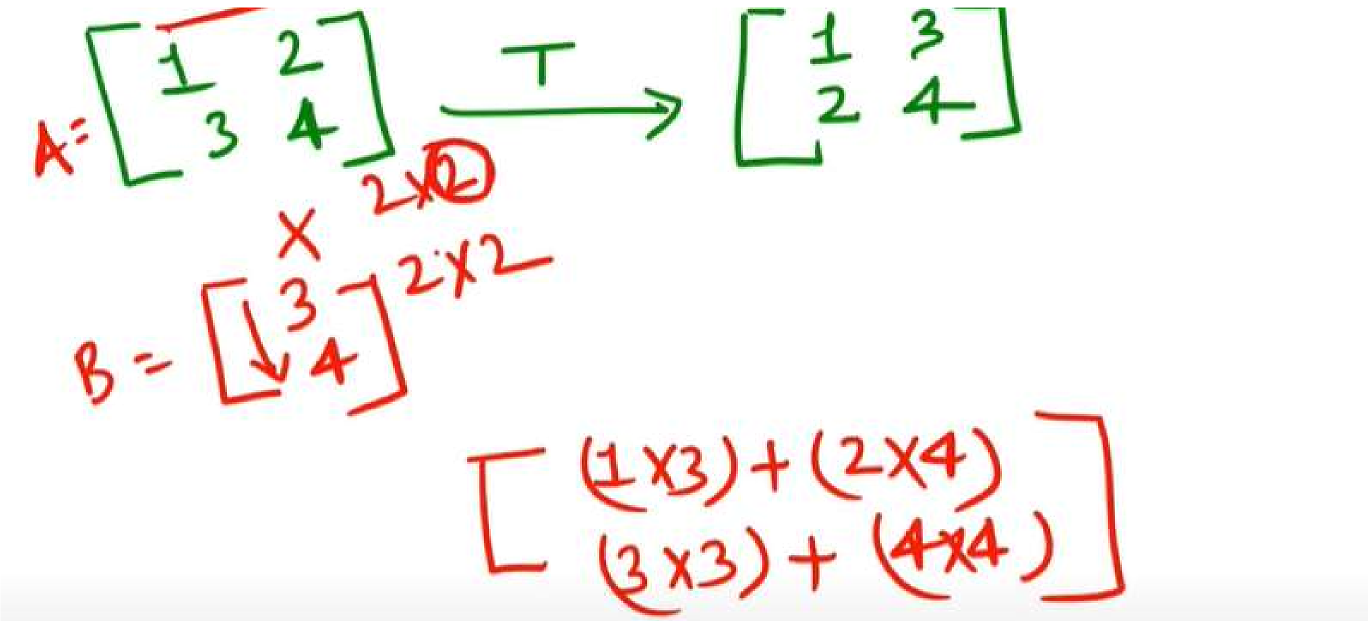




What is Scaler & Vector



Matrix Transpose $ Multipication



Eigen Value & Eigen Vector

$$AV = \lambda V$$

$\lambda \rightarrow$ Value, $V \rightarrow$ Vector

$$\begin{bmatrix} 3 & 6 \\ 5 & 4 \end{bmatrix} \times V = \lambda \times V$$

$$\begin{bmatrix} 3 & 6 \\ 5 & 4 \end{bmatrix} \times V = \lambda \times V \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$V(M - \lambda I) = 0$$

$$\begin{bmatrix} 3 & 6 \\ 5 & 4 \end{bmatrix} \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

$$\begin{bmatrix} 3 & 6 \\ 5 & 4 \end{bmatrix} \times V = \lambda \times V \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$V(M - \lambda I) = 0$$

$$\begin{bmatrix} 3 & 6 \\ 5 & 4 \end{bmatrix} \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

$$\begin{bmatrix} 3-\lambda & 6 \\ 5-0 & 4-\lambda \end{bmatrix} = 0$$

$$(3-\lambda)(4-\lambda) - 30 = 0$$

$$\lambda = 9, \lambda = -2$$

Steps for Developing PCA

→	Physics	Maths
S1	a	b
S2	c	d

Step1 → Define Data

step2 → Make Data mean

Double-click (or enter) to edit

Double-click (or enter) to edit

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from numpy.linalg import eig
```

```
Marks =np.array([[3,4],[2,8],[6,9]])
print(Marks)
```

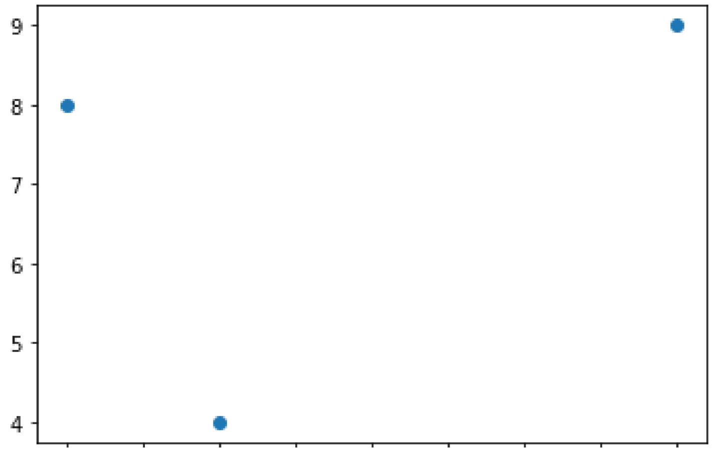
```
[[3 4]
 [2 8]
 [6 9]]
```

```
Marks_df= pd.DataFrame(Marks,columns=["Physics","Maths"])
Marks_df
```

	Physics	Maths
0	3	4
1	2	8
2	6	9

```
plt.scatter(Marks_df["Physics"],Marks_df["Maths"])
```

```
<matplotlib.collections.PathCollection at 0x7f6ba68dc950>
```




```
#making data  mean Centric
Meanbycolumn=np.mean(Marks.T,axis=1)
print(Meanbycolumn)
```

```
[3.66666667 7.        ]
```

```
Scaled_Data = Marks- Meanbycolumn
Scaled_Data
```

```
array([[-0.66666667, -3.        ],
       [-1.66666667,  1.        ],
       [ 2.33333333,  2.        ]])
```

```
Marks
```

```
array([[3, 4],
       [2, 8],
       [6, 9]])
```

```
print(Marks_df["Physics"].mean())
print(Marks_df["Maths"].mean())
```

```
3.6666666666666665
7.0
```

```
#Find Covariance matrix of above scaled data
Cov_mat= np.cov(Scaled_Data.T)
Cov_mat
```

```
array([[4.33333333, 2.5       ],
       [2.5       , 7.        ]])
```

find the Eigen Value and Eigen Vector of the above Covariance matrix

step4 → find λ_1, λ_2

V_1, V_2

Double-click (or enter) to edit

```
Eval, Evec =eig(Cov_mat)
print(Eval)
print(Evec)
```

```
[2.83333333 8.5       ]
[[-0.85749293 -0.51449576]
 [ 0.51449576 -0.85749293]]
```

Get Original Data Projected to principal Components as new axis

```
Projected_data = Evec.T.dot(Scaled_Data.T)
print(Projected_data.T)
```

```
[[-9.71825316e-01  2.91547595e+00]
 [ 1.94365063e+00  1.11022302e-16]
 [-9.71825316e-01 -2.91547595e+00]]
```

```
from sklearn.decomposition import PCA
```

```
pca= PCA(n_components=2)
pca.fit_transform(Marks)
```

```
array([[ 2.91547595e+00, -9.71825316e-01],
       [-7.37588530e-16,  1.94365063e+00],
       [-2.91547595e+00, -9.71825316e-01]])
```

variance explanation ratio by each PCA

```
pca.explained_variance_ratio_
```

```
array([0.75, 0.25])
```

```
PCDF=pd.DataFrame(data=pca.fit_transform(Marks),columns=['PC1','PC2'])
PCDF
```

	PC1	PC2
0	2.915476e+00	-0.971825
1	-7.375885e-16	1.943651
2	-2.915476e+00	-0.971825

Double-click (or enter) to edit