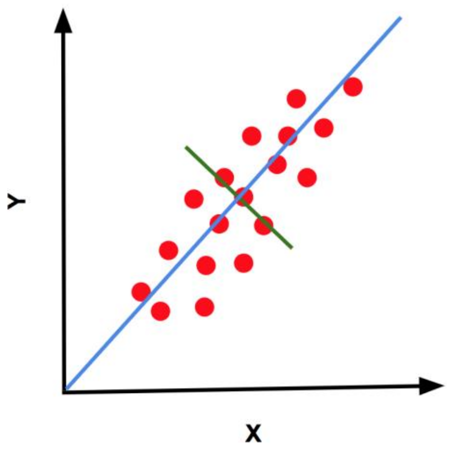


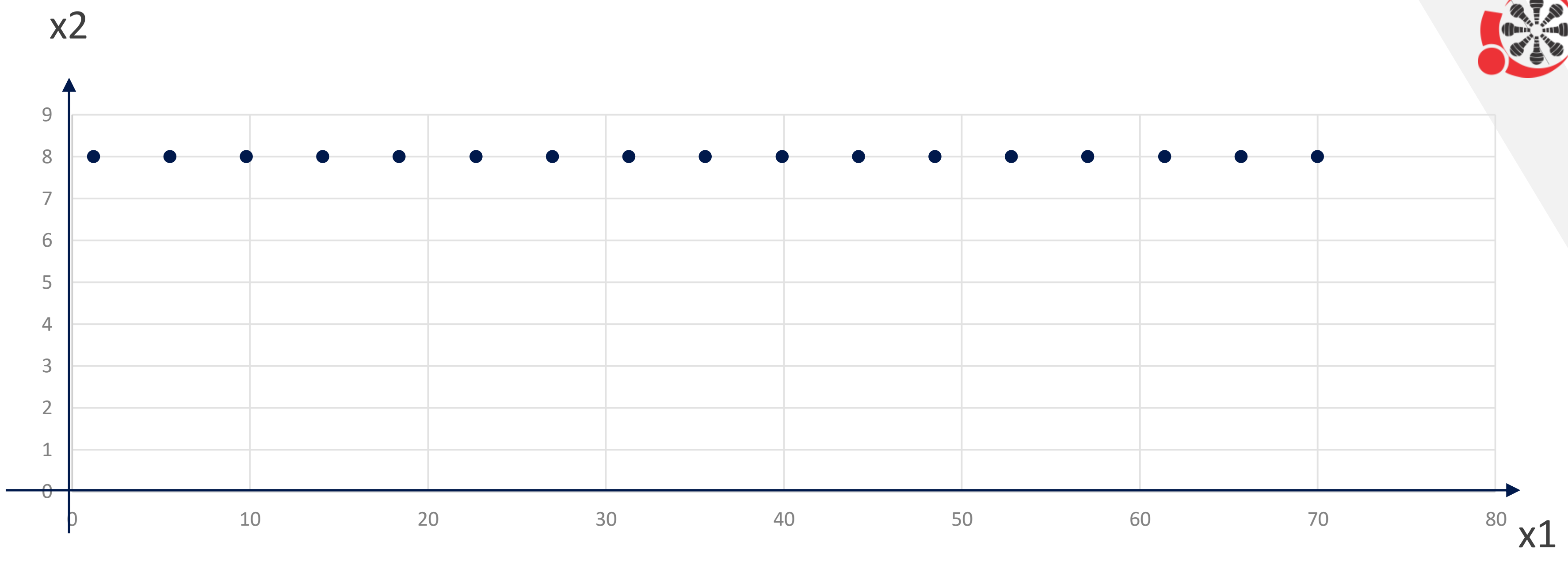
Principal Component Analysis

Feature Engineering

- Stepwise Regression
  - Backward Elimination
  - Forward Elimination
- Step AIC

Which Variable is relevant variable ?

y	x1	x2
10.2	1.2	8
22.7	5.5	8
35.2	9.8	8
47.7	14.1	8
60.2	18.4	8
72.7	22.7	8
85.2	27	8
97.7	31.3	8
110.2	35.6	8
122.7	39.9	8
135.2	44.2	8
147.7	48.5	8
160.2	52.8	8
172.7	57.1	8
185.2	61.4	8
197.7	65.7	8
210.2	70	8

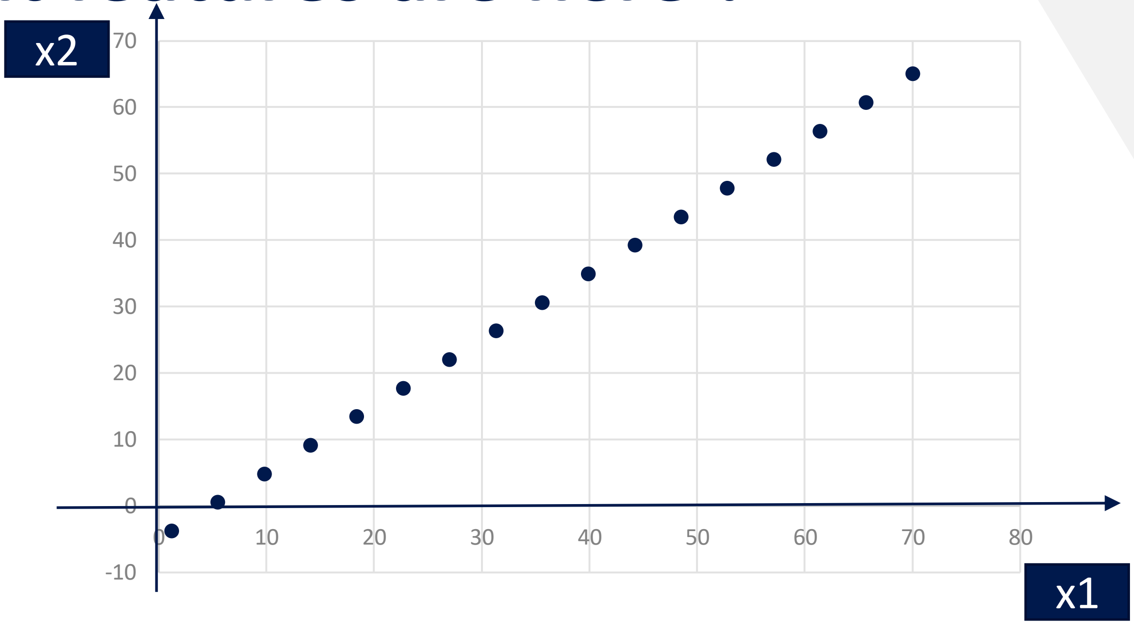


- All the variation in the explanatory variable is in x1, direction only
- This direction is called dominant **Principal Component** of explanatory variables
- The x_2 direction is **redundant**, since not variance along that axis

INNOMATICS RESEARCH LAB

How many relevant features are here ?

y	x1	x2
10.2	1.2	-3.8
22.7	5.5	0.5
35.2	9.8	4.8
47.7	14.1	9.1
60.2	18.4	13.4
72.7	22.7	17.7
85.2	27	22
97.7	31.3	26.3
110.2	35.6	30.6
122.7	39.9	34.9
135.2	44.2	39.2
147.7	48.5	43.5
160.2	52.8	47.8
172.7	57.1	52.1
185.2	61.4	56.4
197.7	65.7	60.7
210.2	70	65



- This seems variations are both in x_1 and x_2 and both are correlated

How many relevant features are here ?

y	x1	x2
10.2	1.2	-3.8
22.7	5.5	0.5
35.2	9.8	4.8
47.7	14.1	9.1
60.2	18.4	13.4
72.7	22.7	17.7
85.2	27	22
97.7	31.3	26.3
110.2	35.6	30.6
122.7	39.9	34.9
135.2	44.2	39.2
147.7	48.5	43.5
160.2	52.8	47.8
172.7	57.1	52.1
185.2	61.4	56.4
197.7	65.7	60.7
210.2	70	65

- If we create new variable

$$X_1 = x_1 + x_2$$
$$X_2 = x_1 - x_2$$

In terms of new variable X_1 and X_2 it clear that there is only one true relevant feature

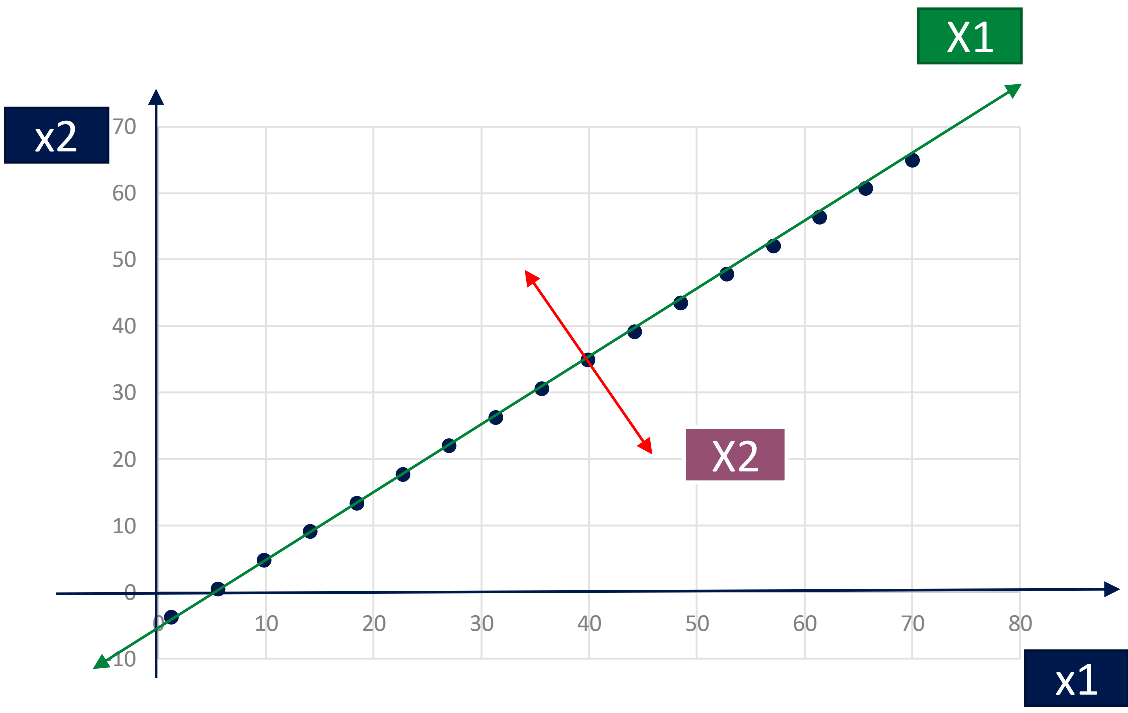
y	X1	X2
10.2	-2.6	5
22.7	6	5
35.2	14.6	5
47.7	23.2	5
60.2	31.8	5
72.7	40.4	5
85.2	49	5
97.7	57.6	5
110.2	66.2	5
122.7	74.8	5
135.2	83.4	5
147.7	92	5
160.2	100.6	5
172.7	109.2	5
185.2	117.8	5
197.7	126.4	5
210.2	135	5

Transformed variables: Interpretation

- The transformation we did is equivalent to viewing from the data points from a rotated co-ordinate system.

- Green = X1
- Red = X2

- The dominant principal component is X1 axis



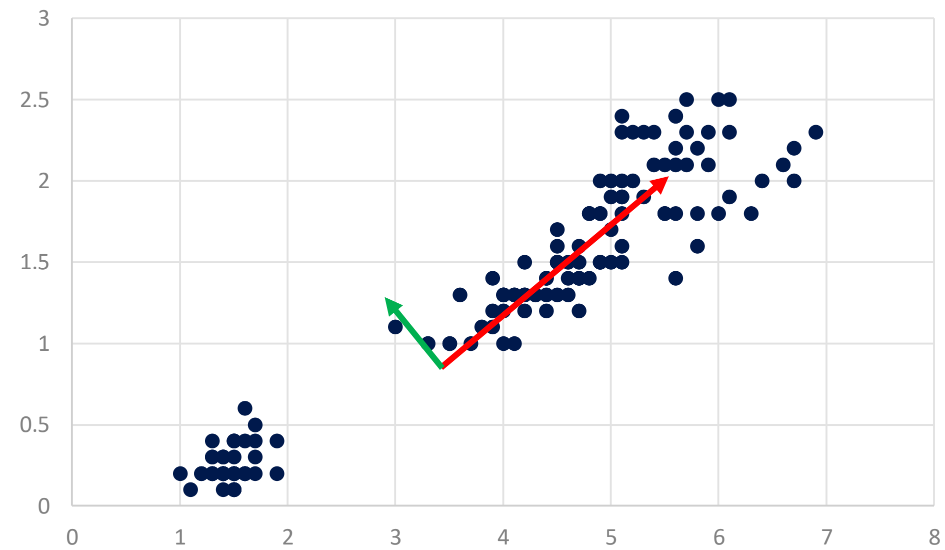
Principal Component Analysis

- PCA is the method which allows you to identify the "directions" in which most of the variations in the data is present.

- Equivalently, it can be thought as method to identify the "directions " along which there is least variations (or least useful information). Identifying this would allow us to drop this irrelevant direction in our regression/model building.

Principal Component directions

y	x1	x2
5.1	1.4	0.2
4.9	1.4	0.2
4.7	1.3	0.2
4.6	1.5	0.2
5	1.4	0.2
5.4	1.7	0.4
4.6	1.4	0.3
5	1.5	0.2
4.4	1.4	0.2
4.9	1.5	0.1
5.4	1.5	0.2
4.8	1.6	0.2
4.8	1.4	0.1
4.3	1.1	0.1
5.8	1.2	0.2
5.7	1.5	0.4
5.4	1.3	0.4
5.1	1.4	0.3
5.7	1.7	0.3
5.1	1.5	0.3
5.4	1.7	0.2




```
data = pd.read_csv('./data/sample_data.csv')
data.head()
```

```
X = data[['x1','x2']]
```

```
sns.relplot('x1','x2',data=X,aspect=2.5)
plt.show()
```

PCA Methodology

x1	x2
1.4	0.2
1.4	0.2
1.3	0.2
1.5	0.2
1.4	0.2
1.7	0.4
1.4	0.3
1.5	0.2
1.4	0.2
1.5	0.1
1.5	0.2
1.6	0.2
1.4	0.1
1.1	0.1
1.2	0.2
1.5	0.4
1.3	0.4
1.4	0.3
1.7	0.3
1.5	0.3
1.7	0.2

Starts with analyzing covariance matrix of features

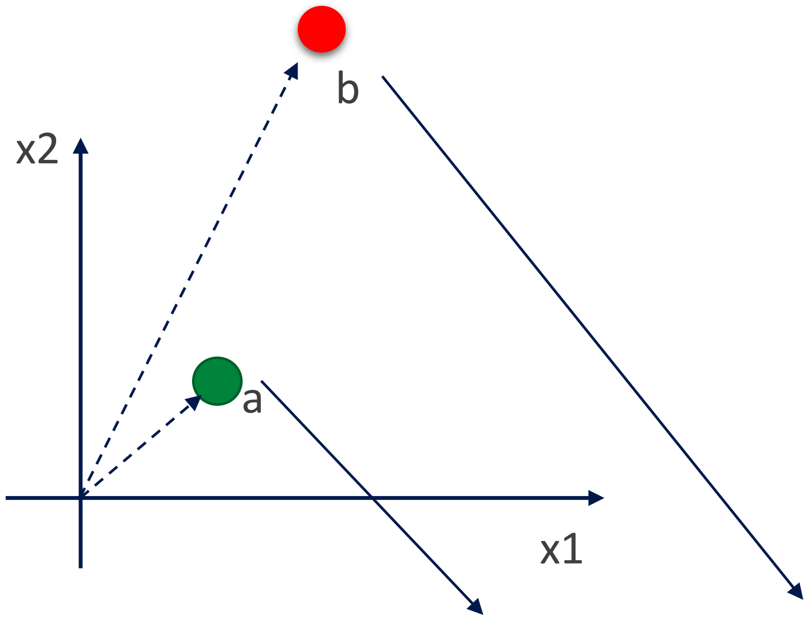
$$cov = \begin{bmatrix} cov(x_1, x_1) & cov(x_1, x_2) \\ cov(x_1, x_2) & cov(x_2, x_2) \end{bmatrix}$$

```
X.cov()
```

	x1	x2
x1	3.113179	1.296387
x2	1.296387	0.582414

The covariance matrix contains information about both correlation and the "special direction" of maximal variance

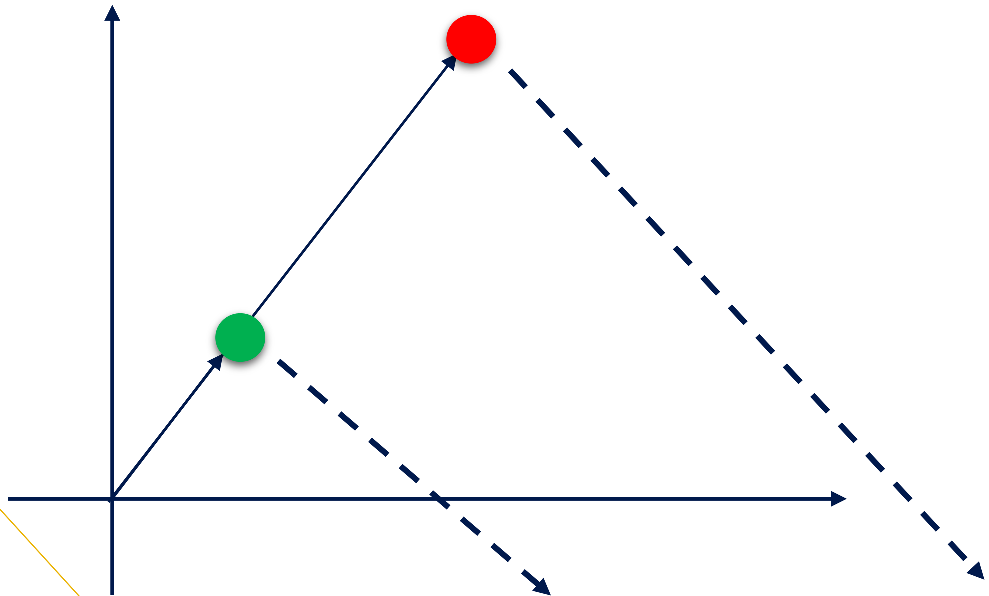
Matrix as a transformation on a vector



$$\begin{bmatrix} 3.11 & 1.29 \\ 1.29 & 0.58 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4.4 \\ 1.87 \end{bmatrix}$$

Matrix as a transformation on a vector

- Special Vectors



$$\begin{bmatrix} 3.11 & 1.29 \\ 1.29 & 0.58 \end{bmatrix} \begin{bmatrix} 0.9215 \\ 0.3882 \end{bmatrix} = \begin{bmatrix} 3.3722 \\ 1.4208 \end{bmatrix} = 3.66 \begin{bmatrix} 3.3722 \\ 1.4208 \end{bmatrix}$$

For given matrix there are special directions, along which its effect only to stretch (without rotation). Such direction is called **Eigen direction or Eigen vectors**

Eigen vectors mathematics

- The eigenvectors and eigenvalues of matrix **A** are defined to be the nonzero **x** and λ values that solve

- $A\,X = \lambda\,X$ (A is just stretching)

- For a n-dim square matrix, there are atmost n eigen-vectors and eigen-values.

Eigen vectors & PCA

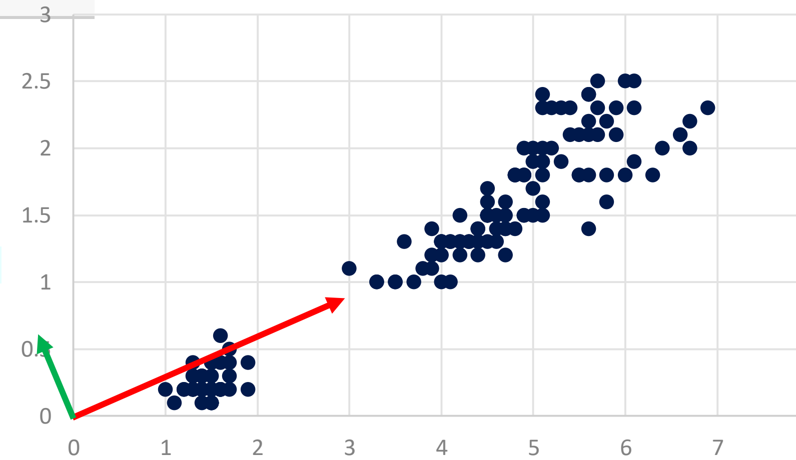
- Eigenvectors are the principal component directions
- Eigenvalues are the magnitude of stretch
- Eigenvalues represent the magnitude of variance of those directions

Eigen values and Eigen vectors

```python
eigvalue, eigvector = np.linalg.eig(X.cov())
print('INFO: Eigenvectos = \n',eigvector)
print('\nINFO: Eigenvalues =',eigvalue)
```

```
INFO: Eigenvectos =
 [[ 0.92154695 -0.38826694]
 [ 0.38826694  0.92154695]]

INFO: Eigenvalues = [3.65937449 0.03621925]
```



- $E_1 = \begin{bmatrix} 0.9215 \\ 0.3882 \end{bmatrix} \Rightarrow \lambda_1 = 3.6593$
- $E_2 = \begin{bmatrix} -0.3882 \\ 0.9215 \end{bmatrix} \Rightarrow \lambda_2 = 0.0362$

Dominant
Principal Component

Eigen vector and Eigen values

- Remember, in the other example when we transformed the data points from original variable x_1, x_2 into new transformed variables, X1 and X2, we could reduce the dimensions ?

- The matrix of eigen-vectors as a whole also allows you to transform each one of our data-points into new variables X_1 & X_2

Transform into Principal Component

- Any record in our data set (x_1, x_2)
- When multiplied by the matrix of eigenvector, we get the new coordinates in the rotated principal component axis.

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0.9215 & -0.3882 \\ 0.3882 & 0.9215 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0.9215 * x_1 - 0.3882 * x_2 \\ 0.3882 * x_1 + 0.9215 * x_2 \end{bmatrix}$$

x1	x2
1.4	0.2
1.4	0.2
1.3	0.2
1.5	0.2
1.4	0.2
1.7	0.4
1.4	0.3
1.5	0.2
1.4	0.2
1.5	0.1
1.5	0.2
1.6	0.2
1.4	0.1
1.1	0.1
1.2	0.2
1.5	0.4
1.3	0.4
1.4	0.3
1.7	0.3
1.5	0.3
1.7	0.2

X1	X2
1.367819	-0.35926
1.367819	-0.35926
1.275664	-0.32044
1.459974	-0.39809
1.367819	-0.35926
1.721937	-0.29144
1.406646	-0.26711
1.459974	-0.39809
1.367819	-0.35926
1.421147	-0.49025
1.459974	-0.39809
1.552129	-0.43692
1.328992	-0.45142
1.052528	-0.33494
1.18351	-0.28161
1.537627	-0.21378
1.353318	-0.13613
1.406646	-0.26711
1.68311	-0.38359
1.498801	-0.30594
1.644283	-0.47574

```
x_arr = X.values # converting into array
```

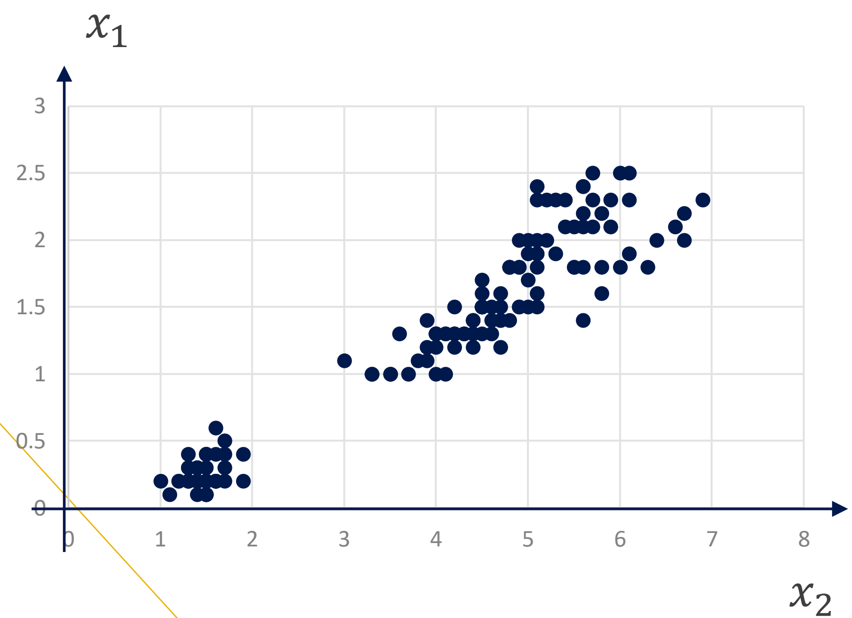
```
# None, 2  = (None, 2 ) * (2,2)
X_pca = np.dot(x_arr,eigvector) # performing dot product
```

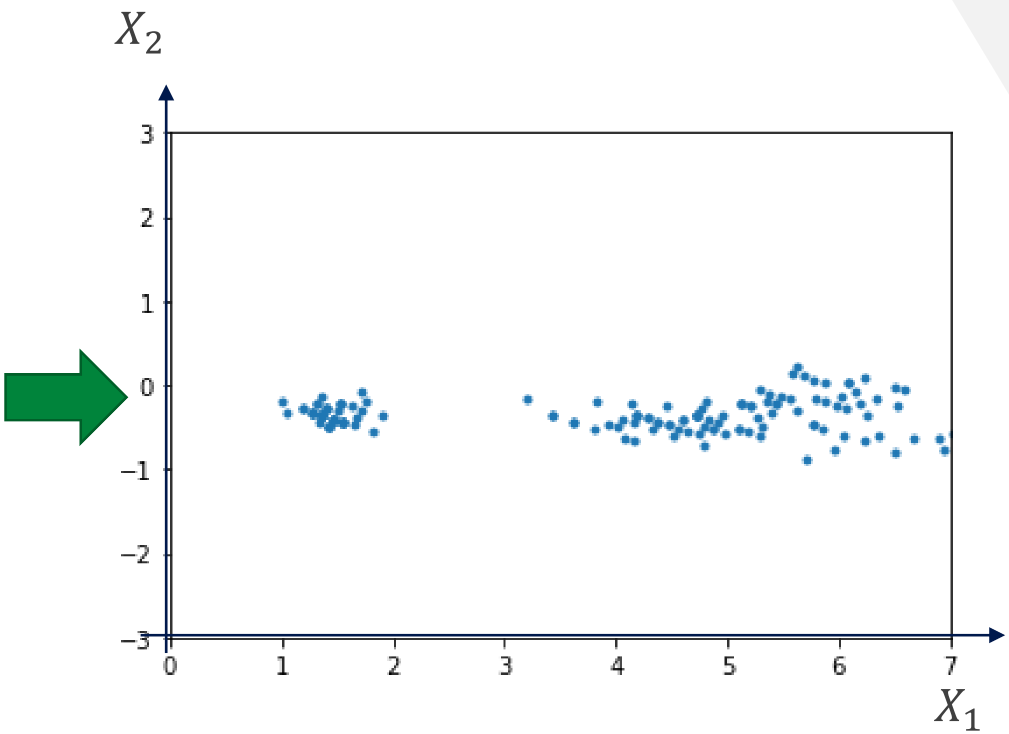
```
X_pca_df = pd.DataFrame(X_pca,columns=['x1','x2'])
X_pca_df.head()
```

	x1	x2
0	1.367819	-0.359264
1	1.367819	-0.359264
2	1.275664	-0.320438

Var(X1) = 3.65   Var(X2) = 0.0362

Data transformed into basis of Principal Component

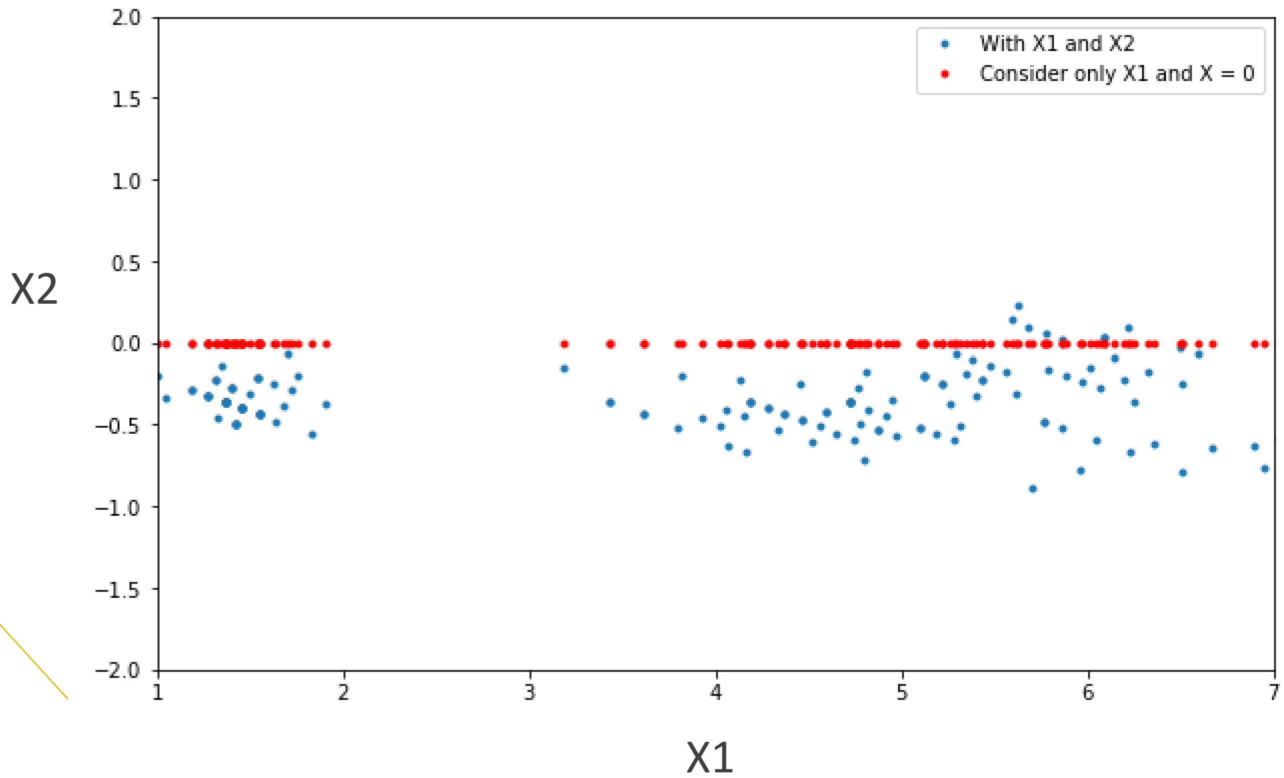




- X2 variable has small variance, we can now drop it by setting it to zero

Dimensionality Reduction

- X_2 has been set to zero



So, now instead of doing regression for x_1, x_2. PCA allows to do regression only with X_1.
This is point of doing PCA. It allows us to ignore variable with low variance

Reference