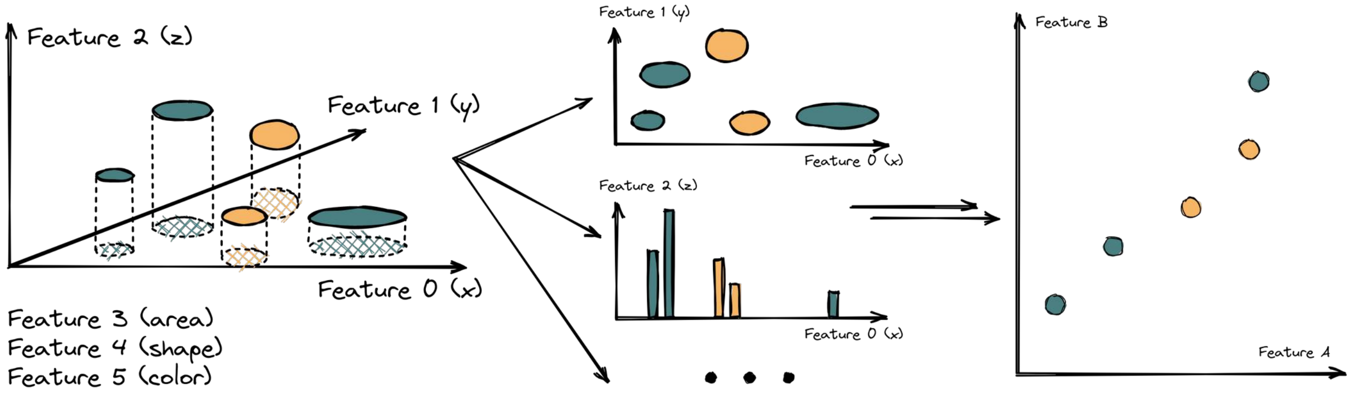
Dimensionality reduction



Week 7

Middlesex University Dubai; Winter22;
CST4050; Instructor: Dr.Ivan Reznikov

v1.2

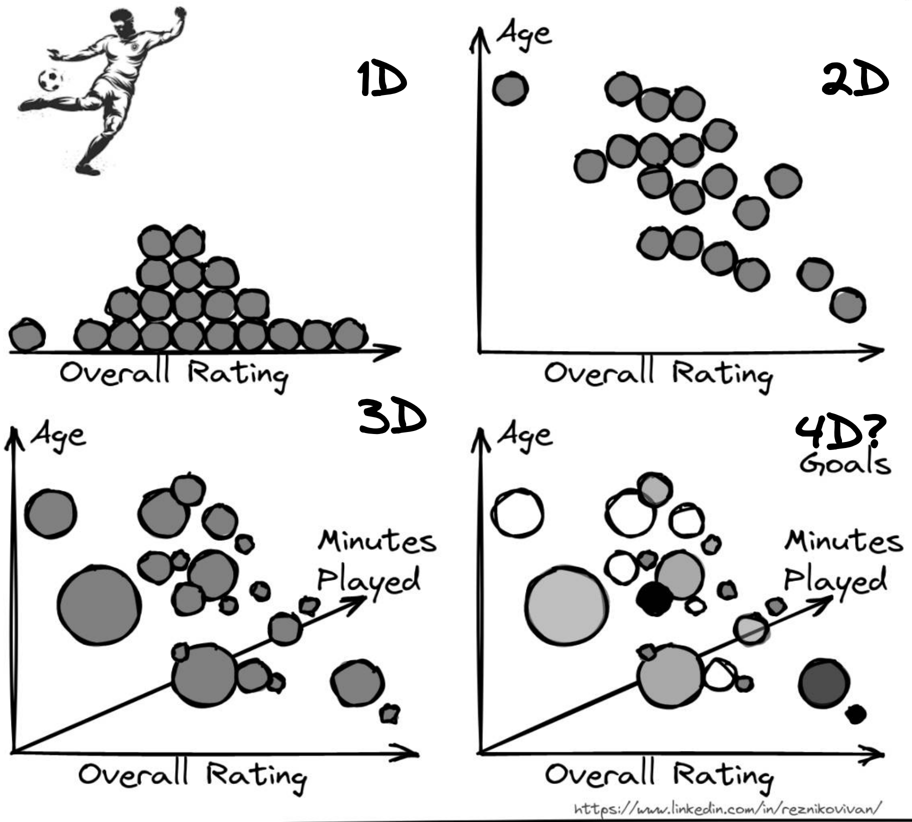
Plan

What is dimension?

Curse of dimensionality

PCA: step-by-step approach

What is a Dimension?



Overall Rating	Age	Minutes Played	Goals	...
40	37	360	0	0
50	30	900	1	2
55	36	360	0	0
55	31	180	0	3
60	35	180	2	0
60	31	90	3	1
60	28	180	6	1
60	24	90	3	0
65	33	180	0	0
65	30	360	2	0
65	27	90	0	0
65	23	540	2	4
70	34	90	1	2
70	29	90	3	1
70	22	180	0	0
75	25	180	3	0
75	21	90	1	3
80	27	90	3	1
85	20	360	4	1
90	18	90	7	1

Dimensions = features, attributes, variables, etc.

How many Dimension is a lot?

1. **Generate random train data:**

Size = 51, dimensions = 3, range (0,1)

```
In [3]: np_arr = np.random.rand(size,3)
        np_arr

Out[3]: array([[0.69646919, 0.28613933, 0.22685145],
               [0.55131477, 0.71946897, 0.42310646],
               [0.9807642 , 0.68482974, 0.4809319 ],
               [0.39211752, 0.34317802, 0.72904971],
               [0.43857224, 0.0596779 , 0.39804426],
               [0.73799541, 0.18249173, 0.17545176],
               [0.53155137, 0.53182759, 0.63440096],
               [0.84943179, 0.72445532, 0.61102351],
               [0.72244338, 0.32295891, 0.36178866],
               [0.22826323, 0.29371405, 0.63097612],
               [0.09210494, 0.43370117, 0.43086276],
               [0.4936851 , 0.42583029, 0.31226122],
               [0.42635131, 0.89338916, 0.94416002],
               [0.50183668, 0.62395295, 0.1156184 ],
               [0.31728548, 0.41482621, 0.86630916],
               [0.25045537, 0.48303426, 0.98555979],
               [0.51948512, 0.61289453, 0.12062867],
               [0.8263408 , 0.60306013, 0.54506801],
               [0.34276383, 0.30412079, 0.41702221],
```

2. **Generate target data:**

Size = 51, dimensions = 1

count(0) = 26, count(1) = 25

3. **Build 10 intervals (sections):**

Group data in intervals using 0.1 window

4. **Build “naive classifier“:**

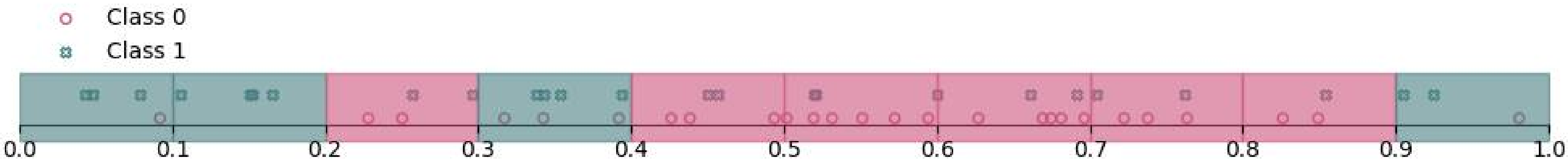
default_forecast_value = 0

Logic: the most number of points will set the class for the interval. If equal number of 0/1 values: class is set to default_forecast_value

How many Dimension is a lot?

1 Dimension:

Misclassified points: 17 out of 51
Empty sections: 0



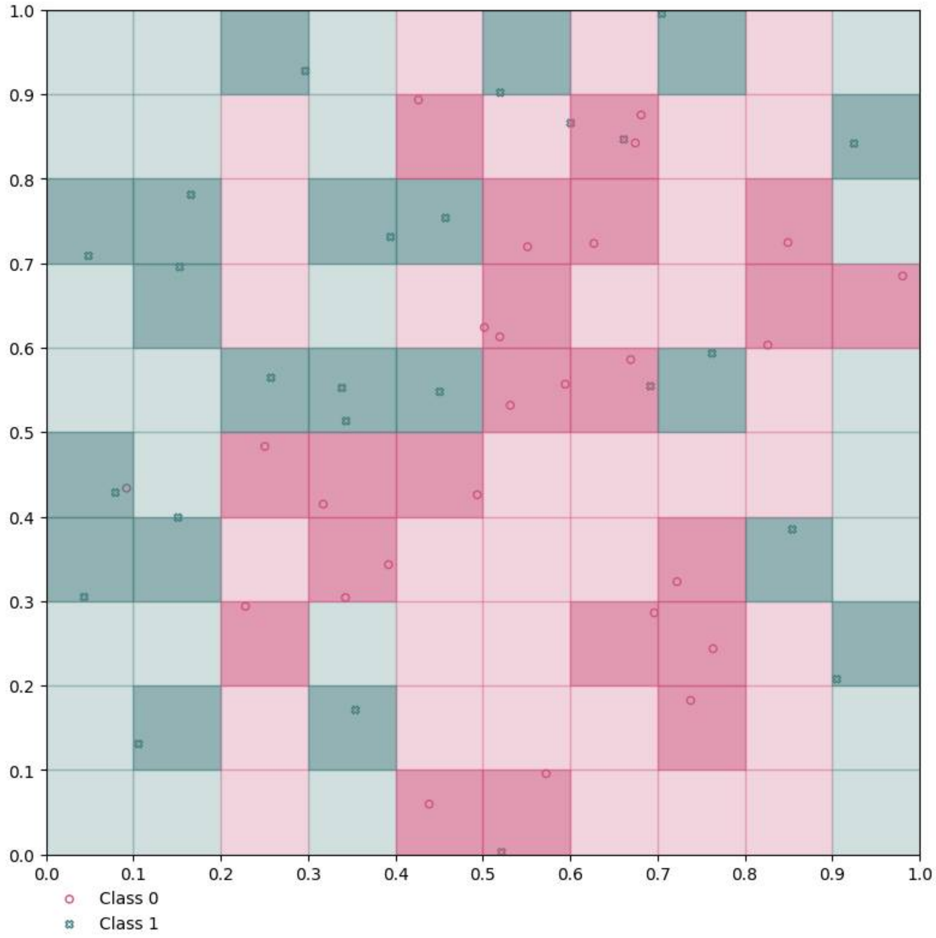
How many Dimension is a lot?

2 Dimensions:

Misclassified points: 5/51
Empty sections: 59

Is our classifier doing better?
No!!

The data is already too sparse.

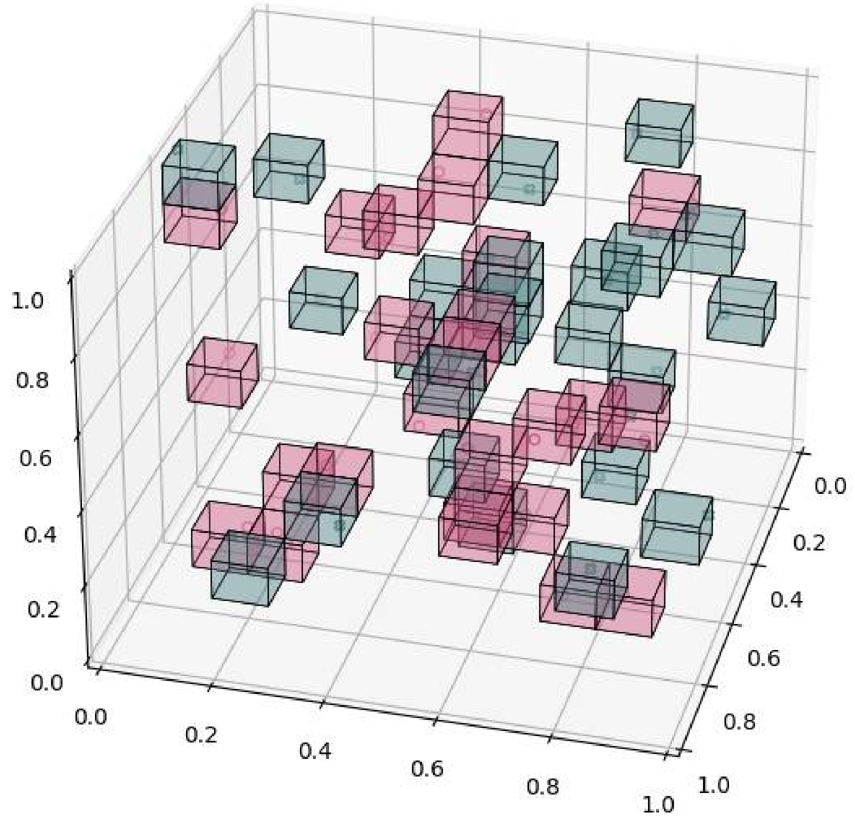


How many Dimension is a lot?

3 Dimensions:

Misclassified points: 1 out of 51
Empty sections: 951

Though our naive classifier can correctly set 0/1 classes to 50 out of 51 points, it’s pretty useless.



How many Dimension is a lot?

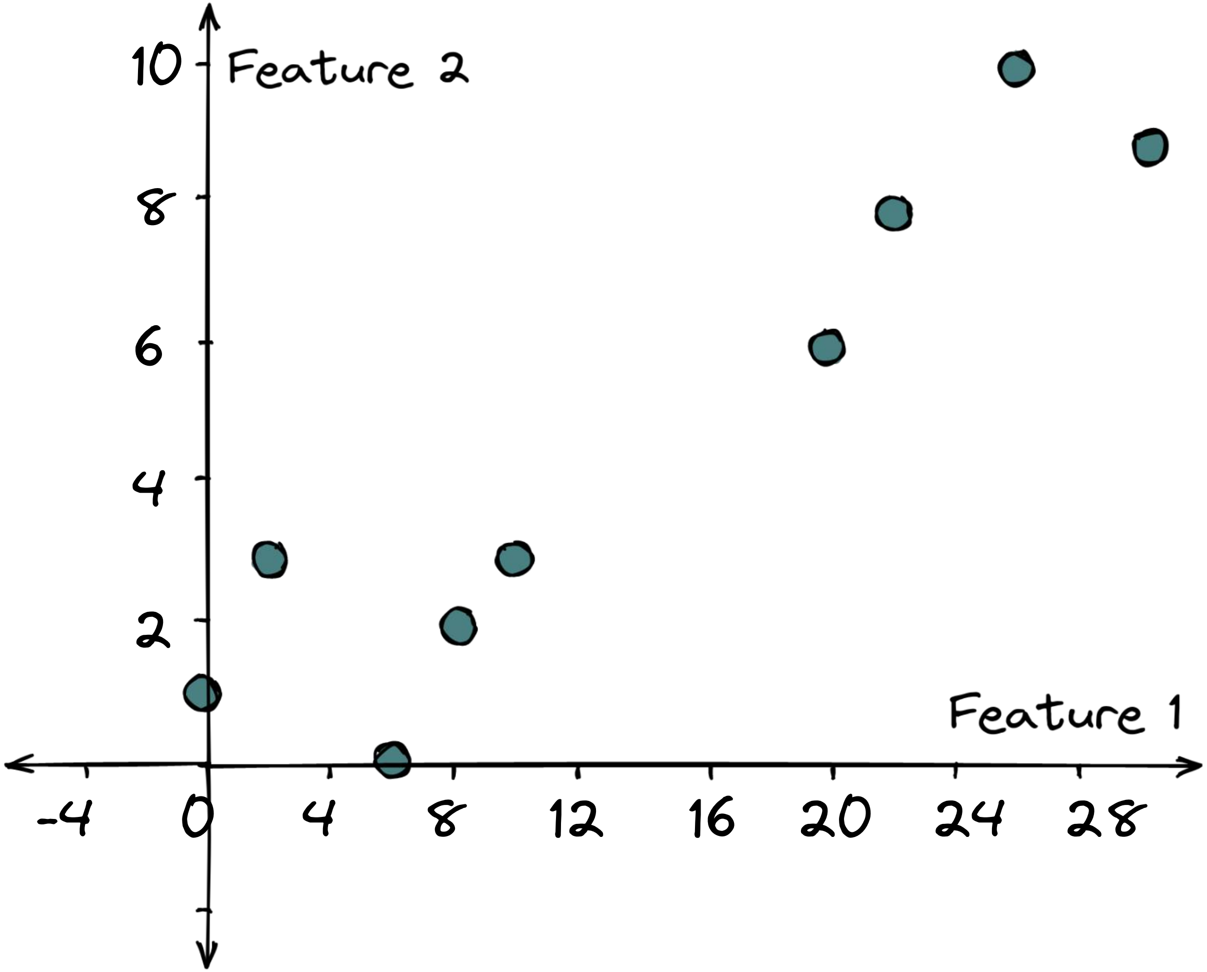
51 data points:
1 feature → 51 points per 10 intervals = 51/10 = **5.1** points per "box".
2 features → 51 points per 10^2 intervals – **0.51** points per section.
3 features result in a density $51/10^3$ = 51/1000 = **0.051** points per interval.

With more data, it becomes easier to separate it. We've almost perfectly separated 51 points using just 3 dimensions.

The results will be different if we use smaller interval ranges, but no matter what, it's always possible to separate N+1 points using N-dimensions.

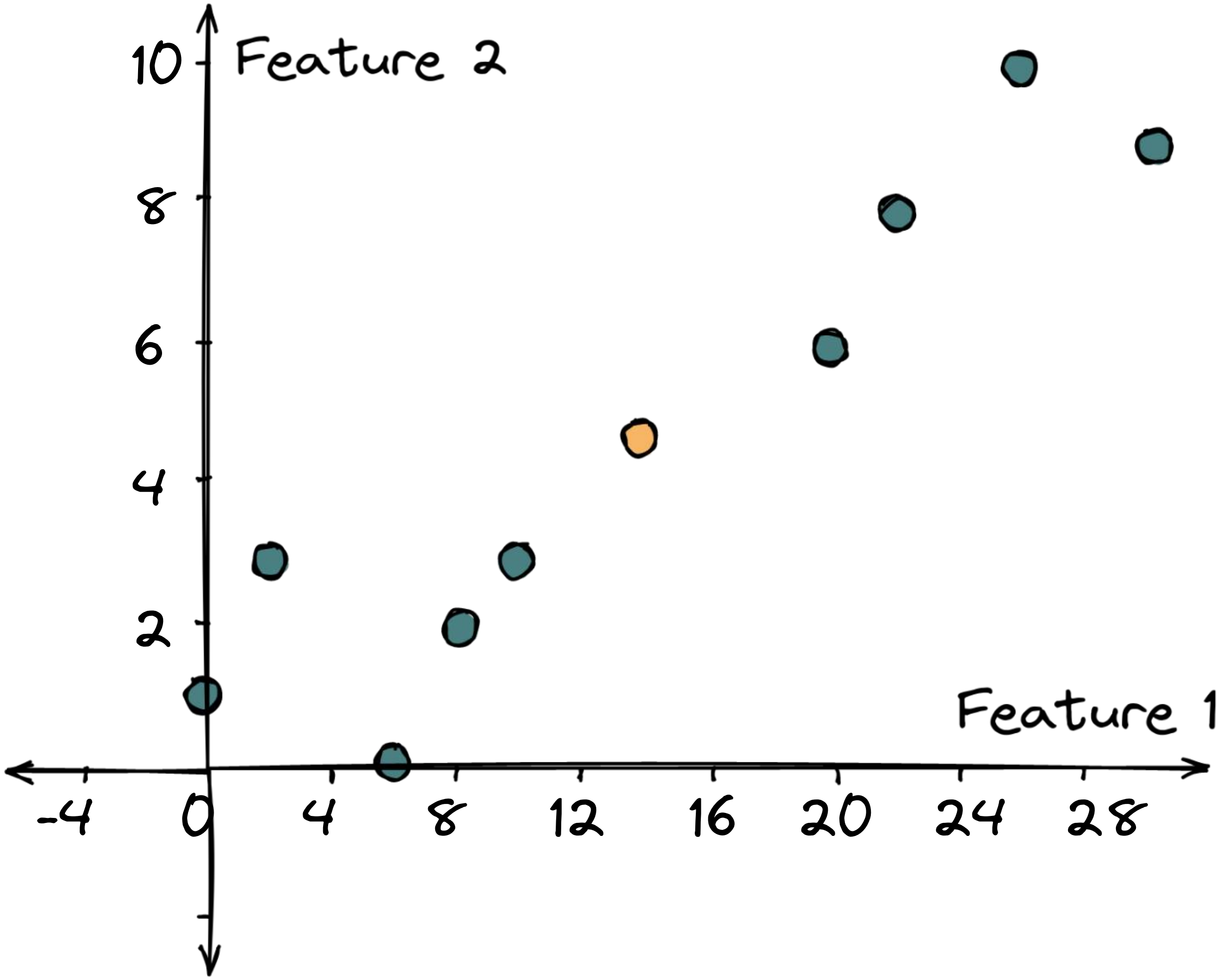
In our case, it seems 2 dimensions are already too much.

PCA. Step1: Plot Data

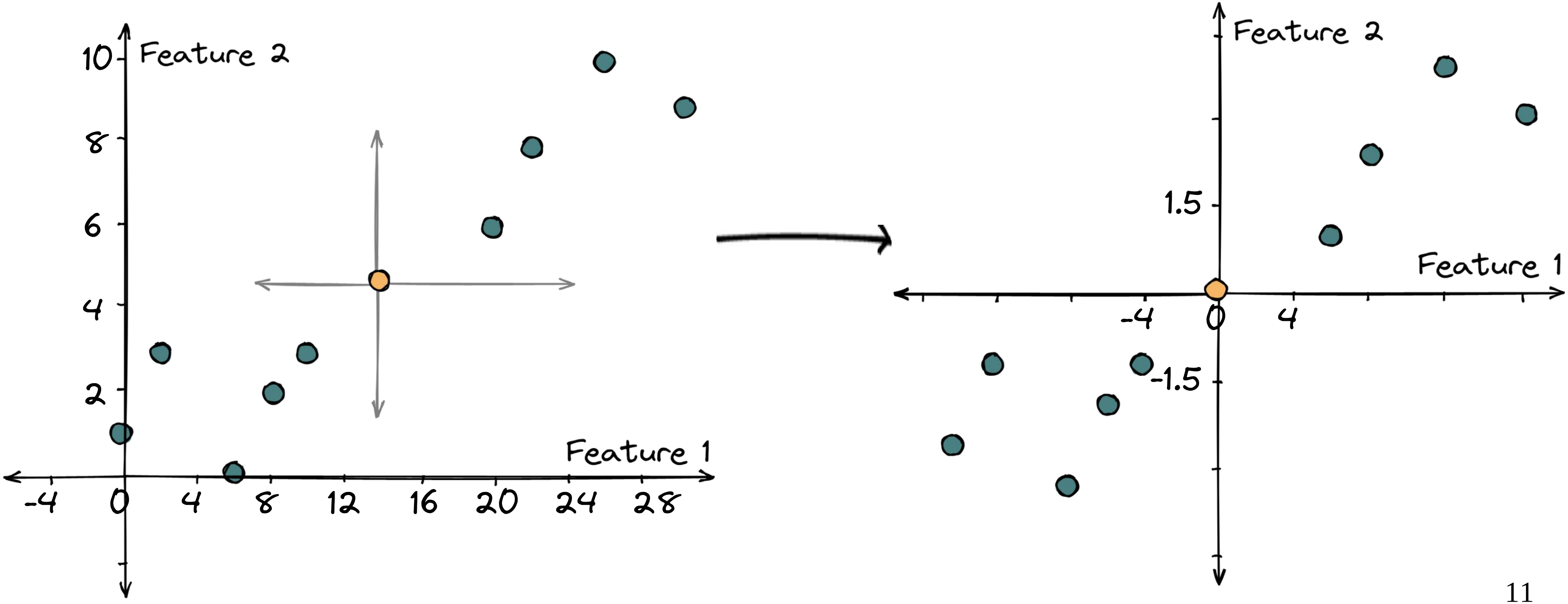


Feature 1	Feature 2
0	1
2	3
6	0
8	2
10	3
20	6
22	8
26	10
30	9

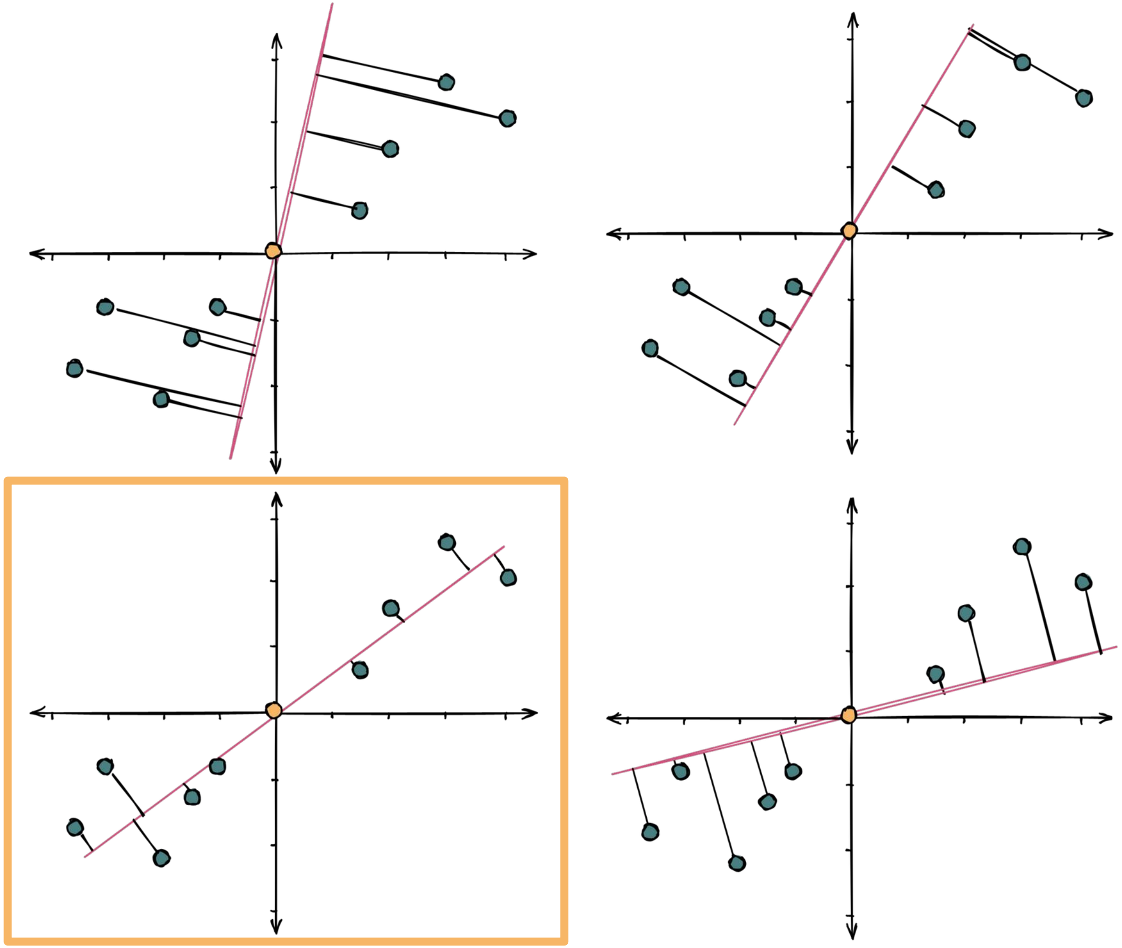
PCA. Step2: Plot Center of Data



PCA. Step3: Recenter Data



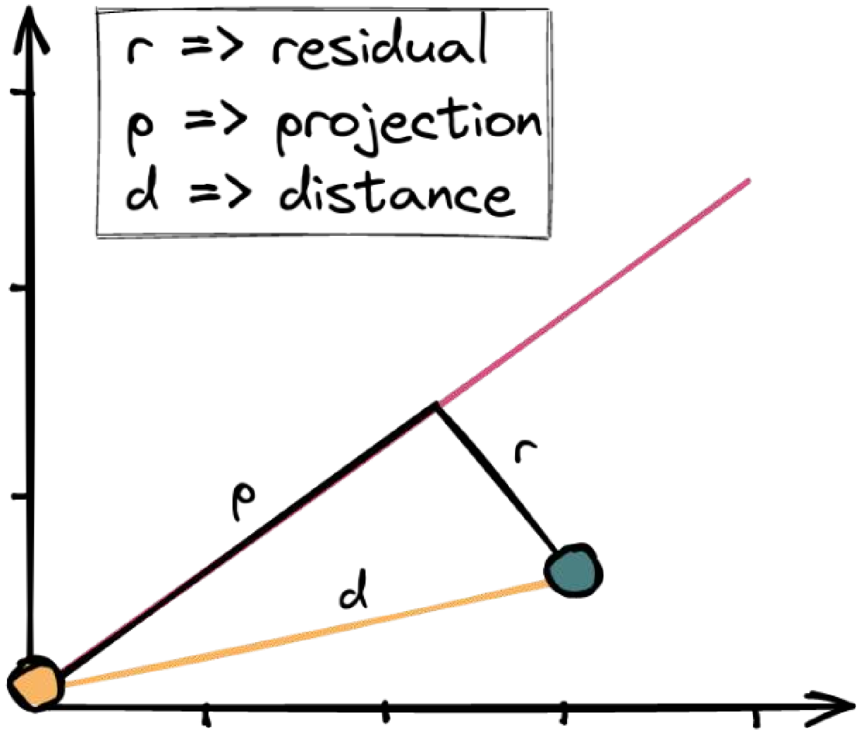
PCA. Step4a: Find the Best Fit Line

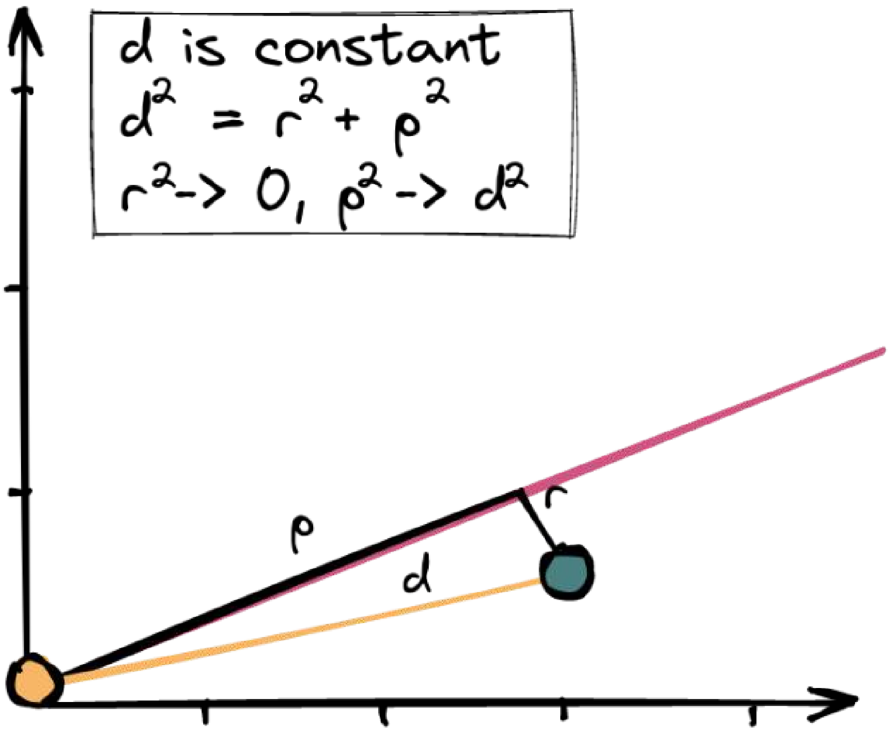


We look for the best fitting line. The classic way is to use the least square method – we look for the line that produces the least sum of squared errors.

The resulted line is the first <u>principle component</u> (PC) => PC1

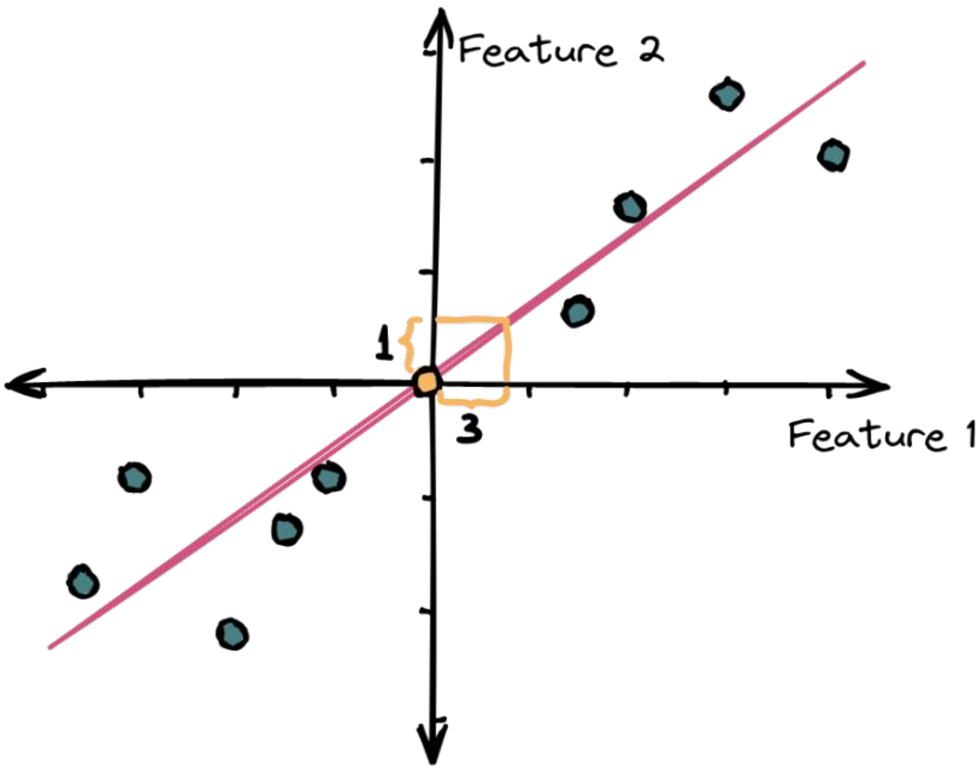
PCA. Step4b: Find the Best Fit Line

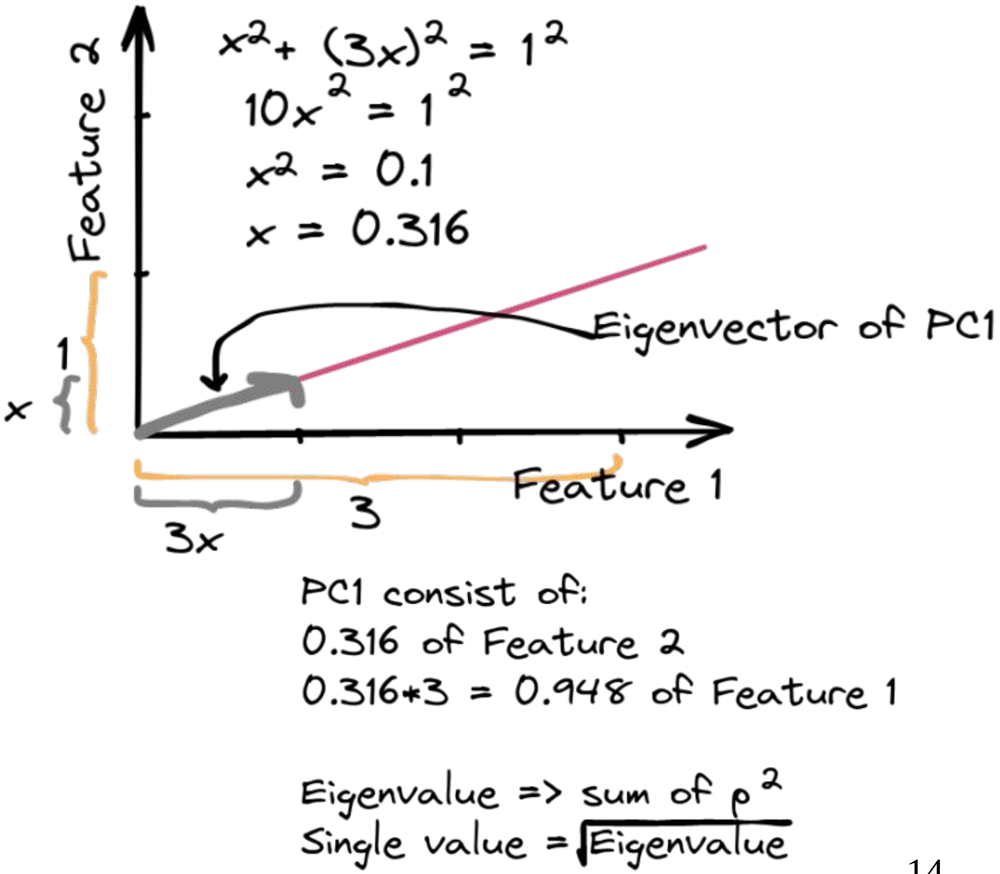




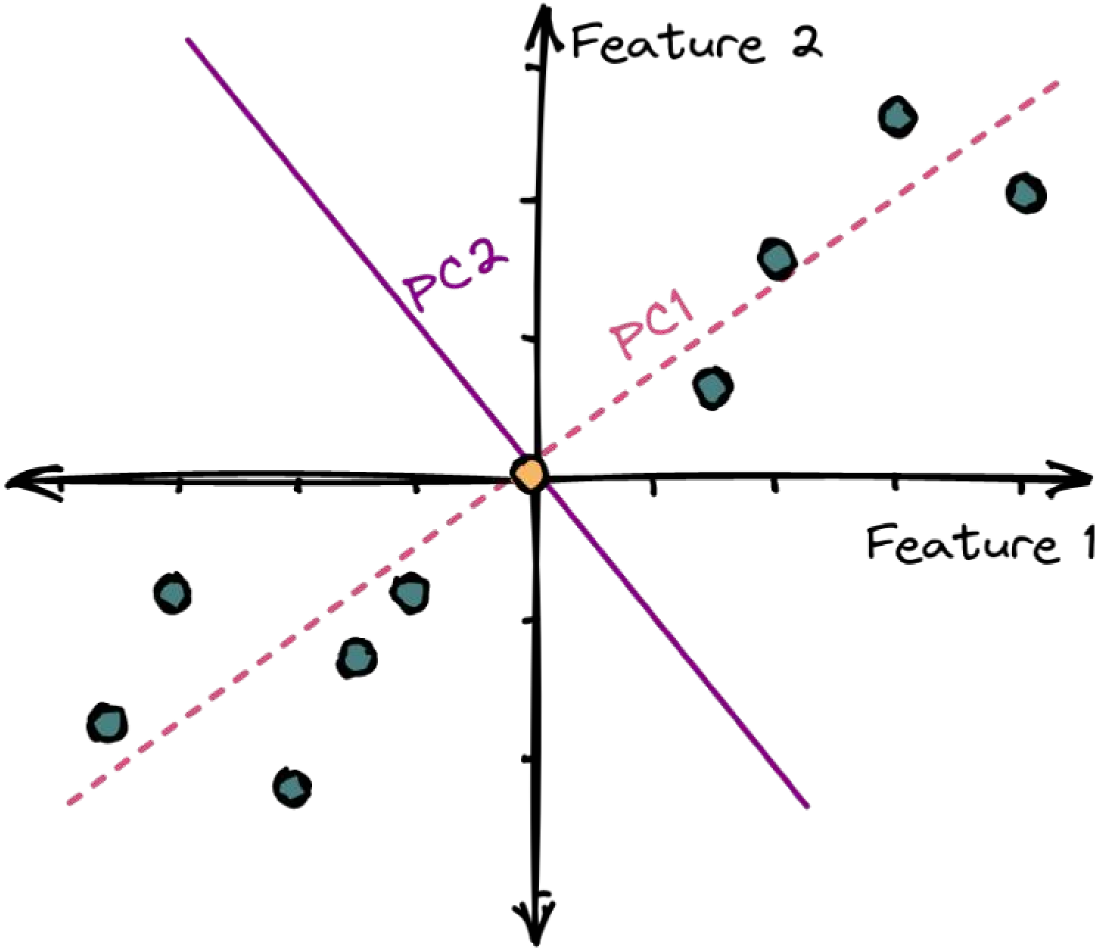
Instead of looking for minimal **sum(r_i^2)**, we can look for maximum **sum(p_i^2)**

PCA. Step5: Eigenvector & Eigenvalue



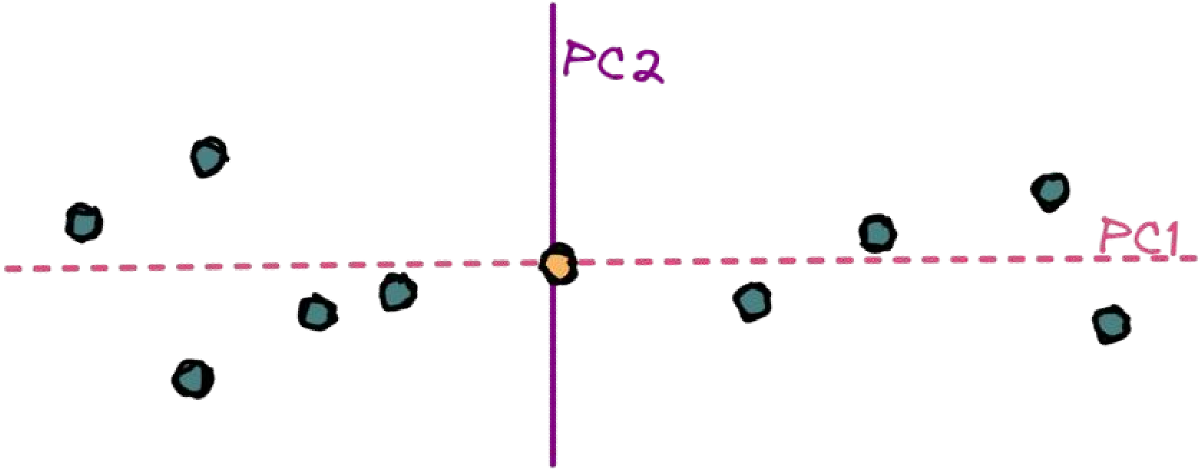


PCA. Step6: Principal Component 2



Second principle component (**PC2**) can be found as a perpendicular to **PC1**

PCA. Step7: Variations



$$\frac{\text{sum of } p^2 \text{ (PC1)}}{n-1} = \text{(variation for PC1)} = 130$$

$$\frac{\text{sum of } p^2 \text{ (PC2)}}{n-1} = \text{(variation for PC2)} = 1.6$$

$$\text{PC1 variation} = 130/(130+1.6) = 0.987 = 98.7\%$$

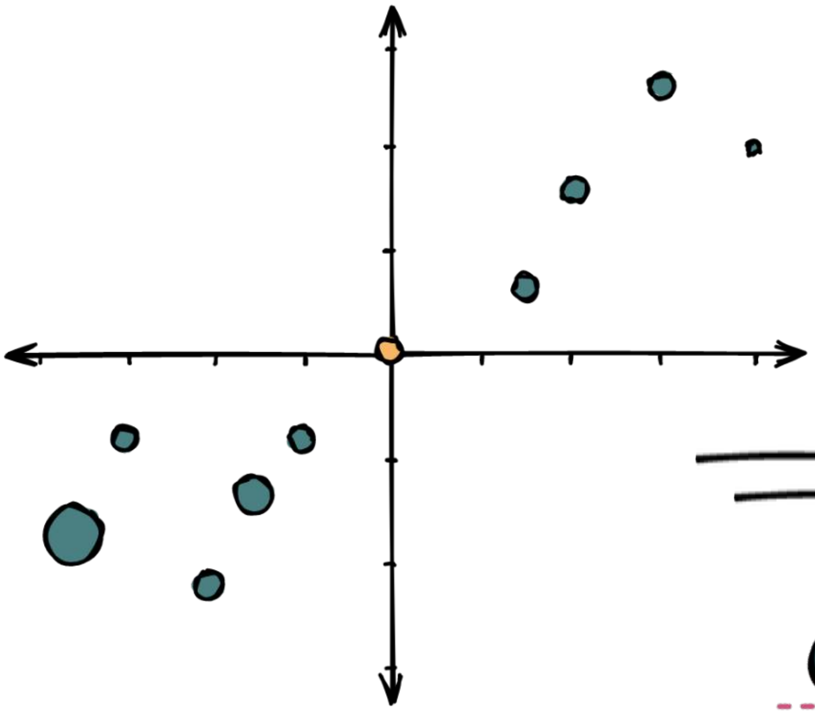
At this moment, we don't need Feature1 and Feature2 anymore.

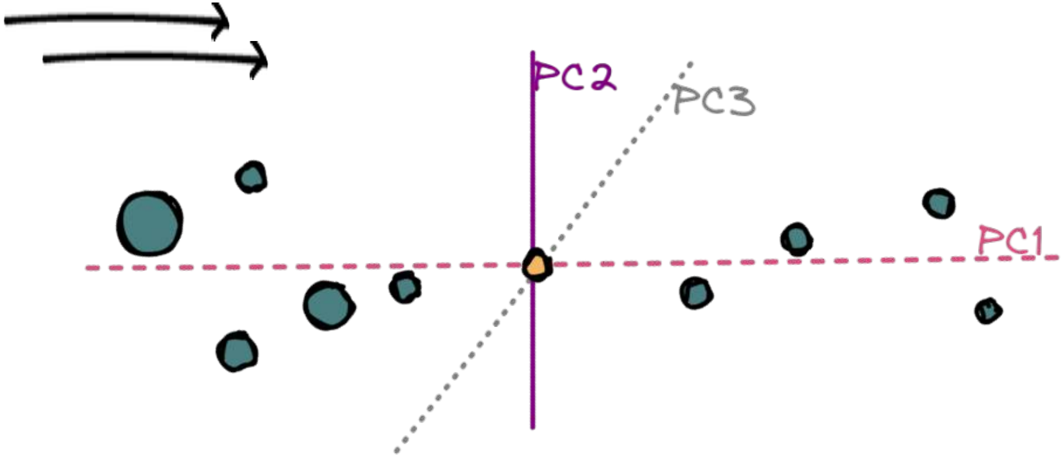
By rotating PC1-PC2 we result in the plot that uses principal components as axes.

The PC variations allow us to understand the importance of principal components in explaining data.

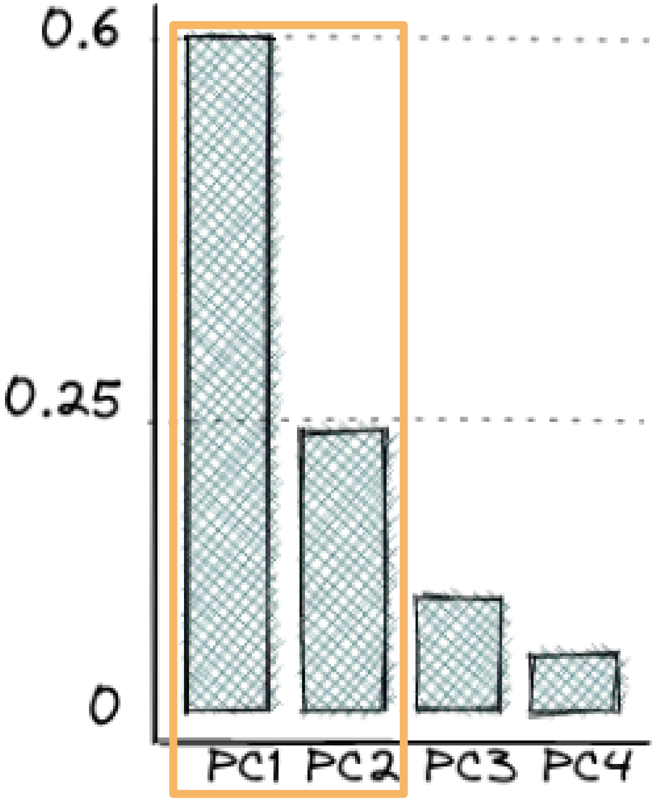
PCA. Step8: PC3, PC4, PCn, etc

For every next PC we build the perpendicular line to the PCs present. It's hard to plot the PCs after PC3, so we'll need to imagine.





PCA. Step9: PC Importance



Imagine you've built 4 PCs.
The resulted variations are:
PC1 = 0.6
PC2 = 0.25
PC3 = 0.1
PC4 = 0.05

Makes sense to leave only PC1 and PC2, as they represent 85% of the variation. Now, instead of 4+ features we've reduced their number to 2.