

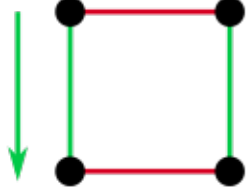
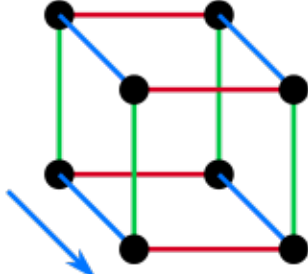
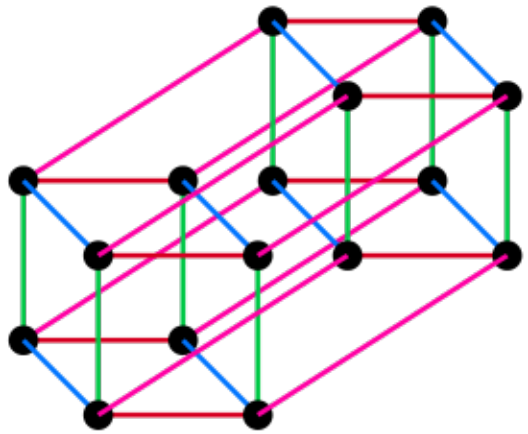



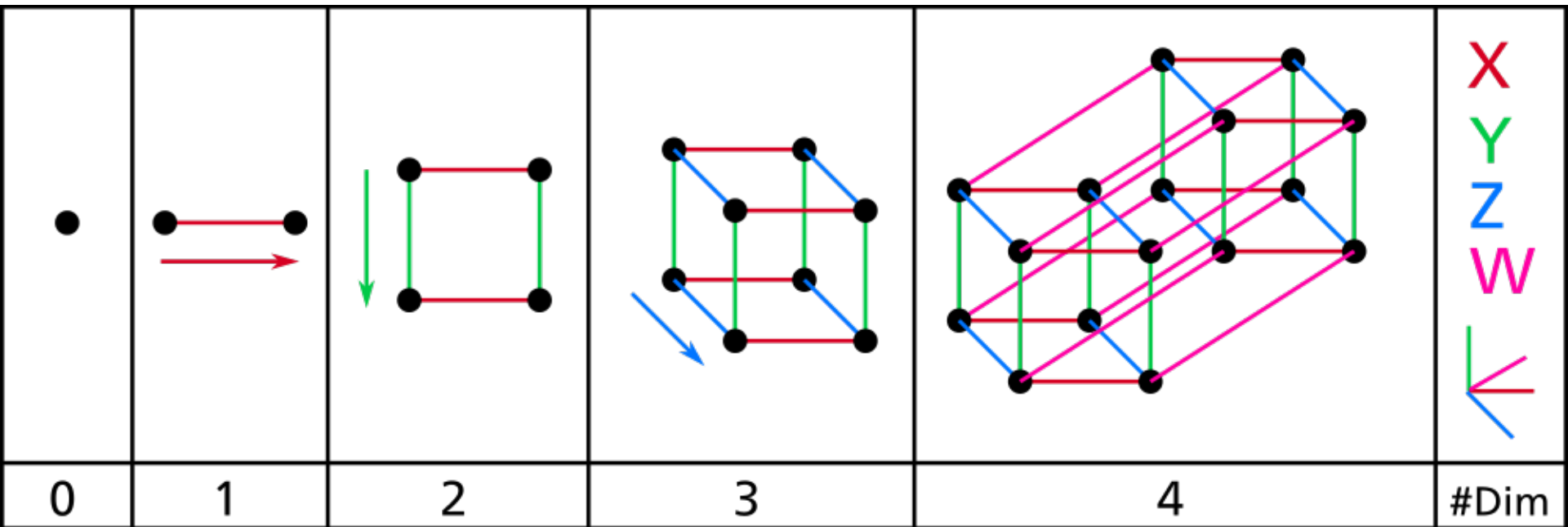
Dimensionality Reduction



Curse of Dimensionality

					<div><div>X</div><div>Y</div><div>Z</div><div>W</div></div>
0	1	2	3	4	#Dim

Curse of Dimensionality



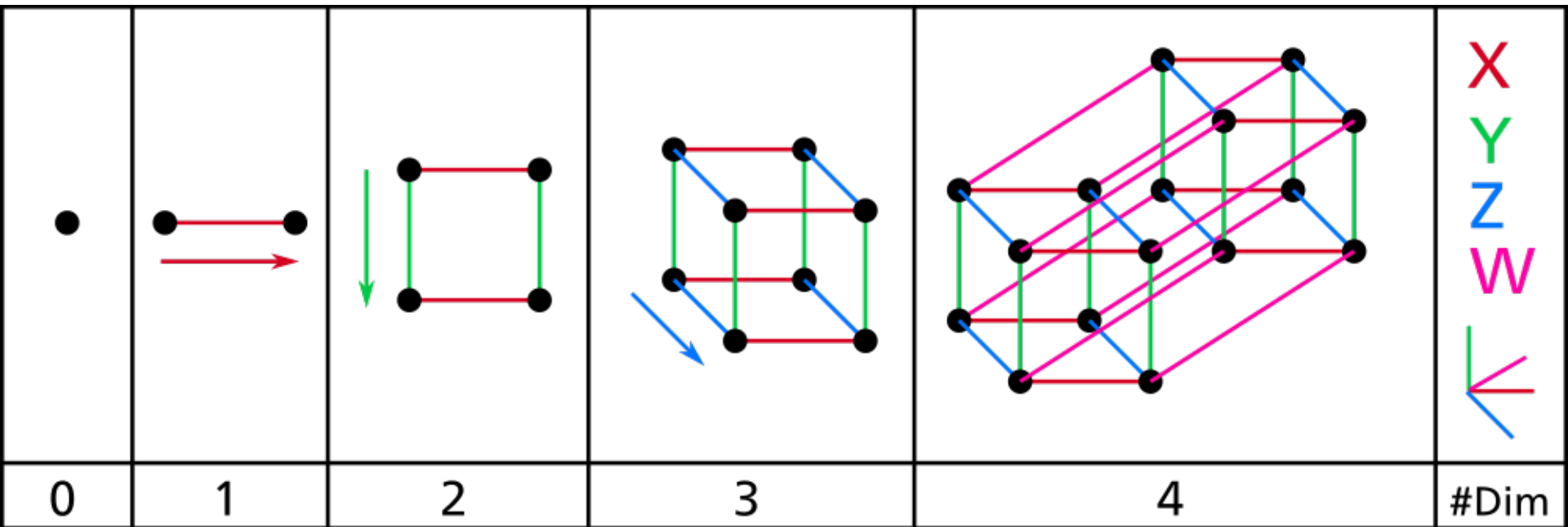
One dimension:

Small space

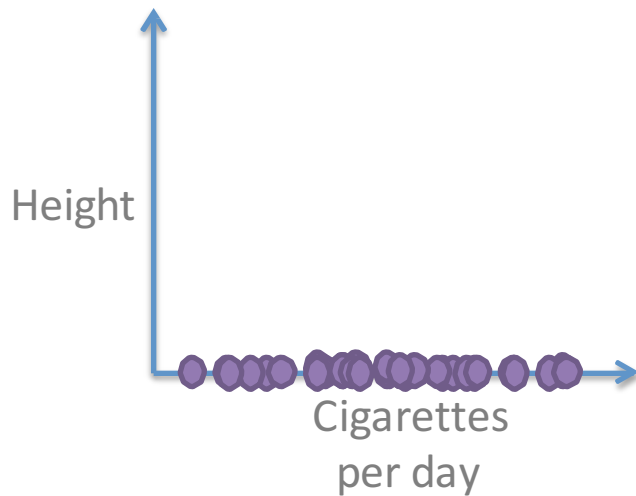
Being close quite probable



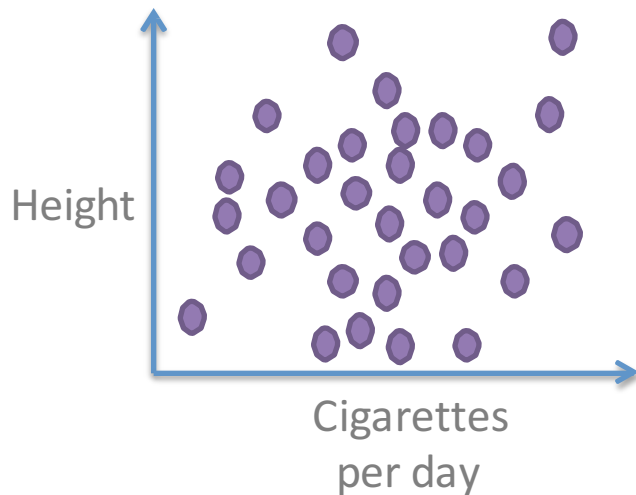
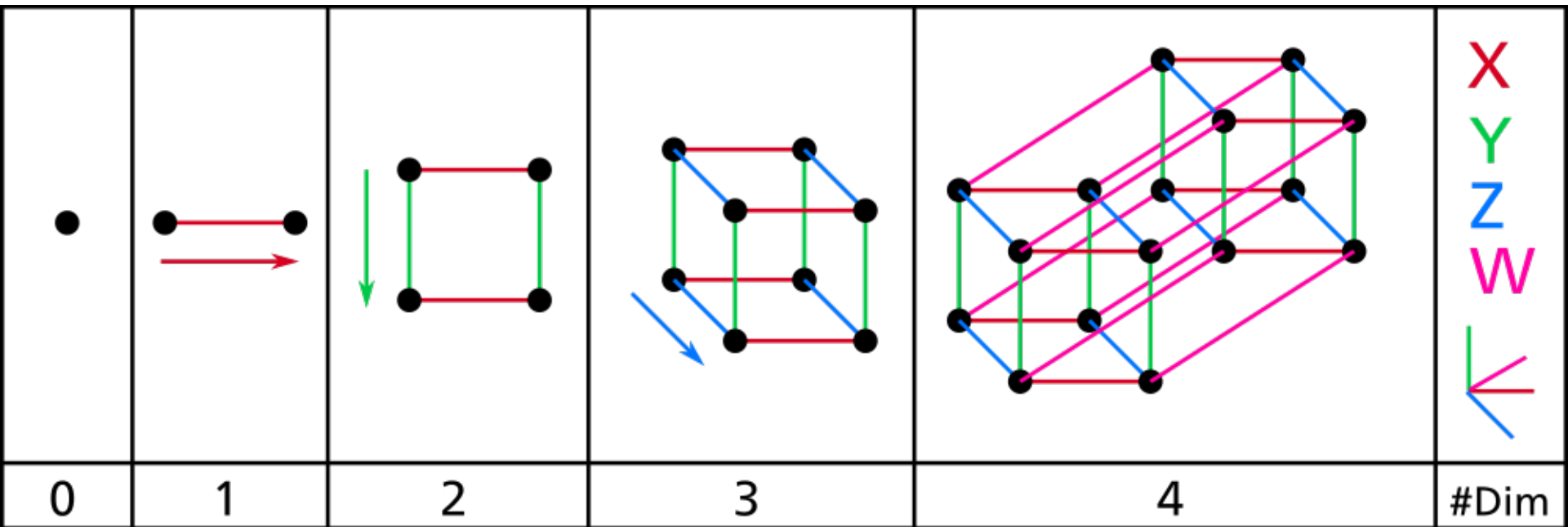
Curse of Dimensionality



Two dimensions



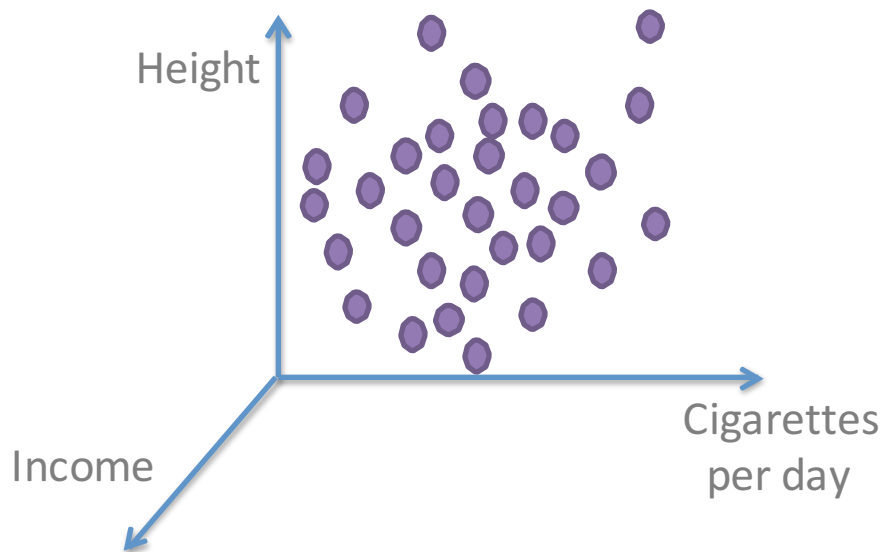
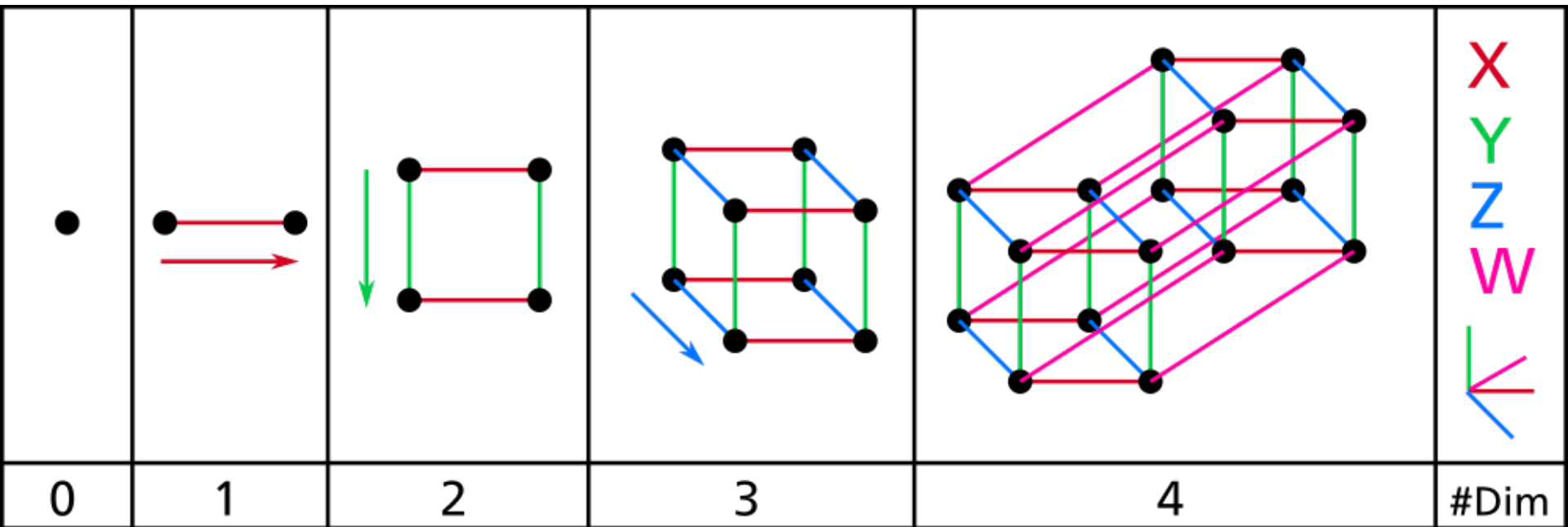
Curse of Dimensionality



Two dimensions:

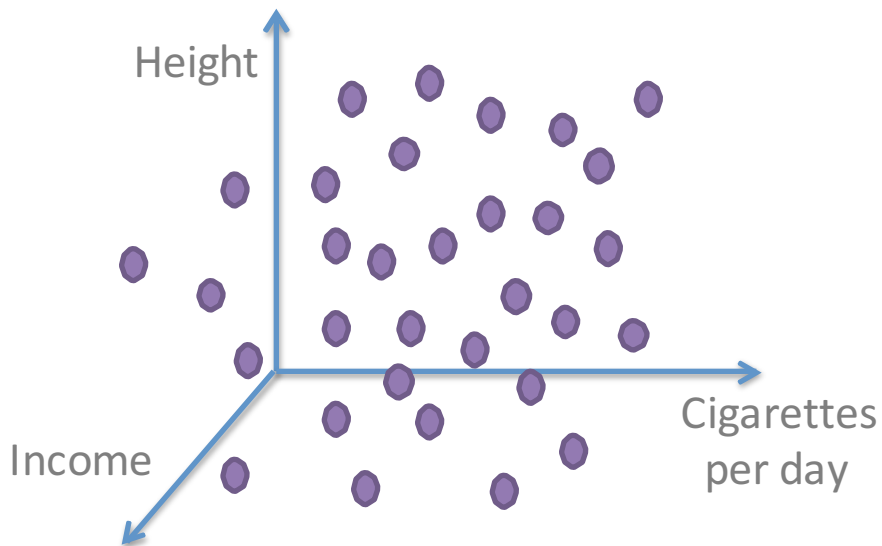
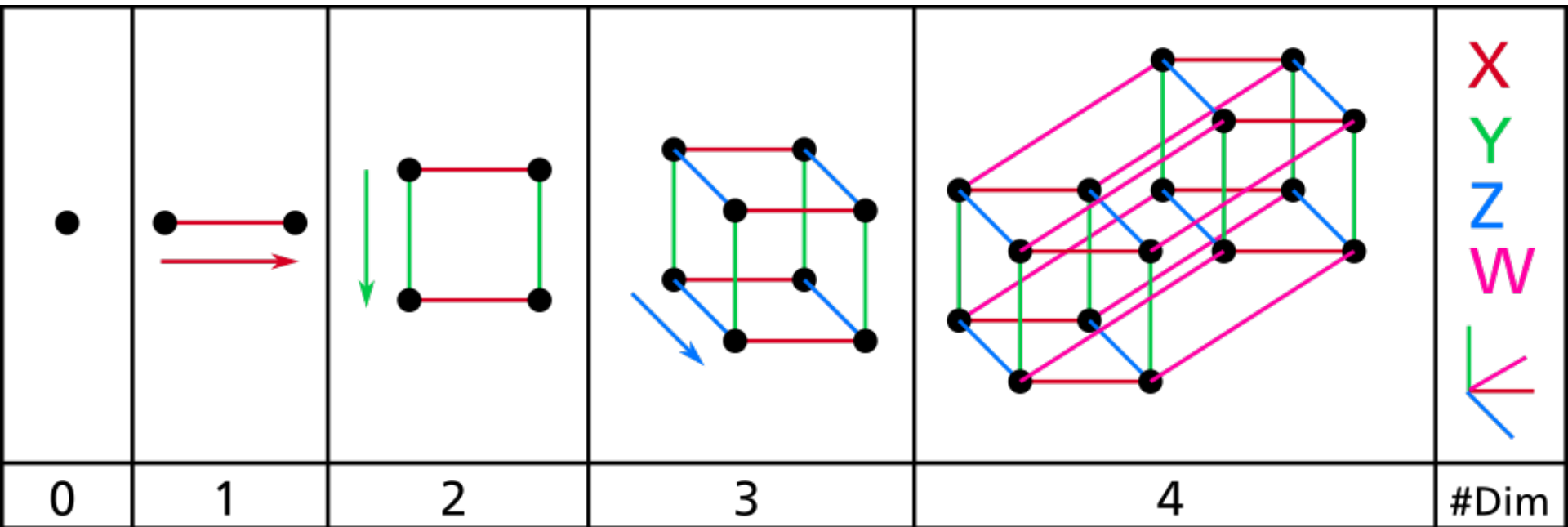
More space but still not so much
Being close not improbable

Curse of Dimensionality



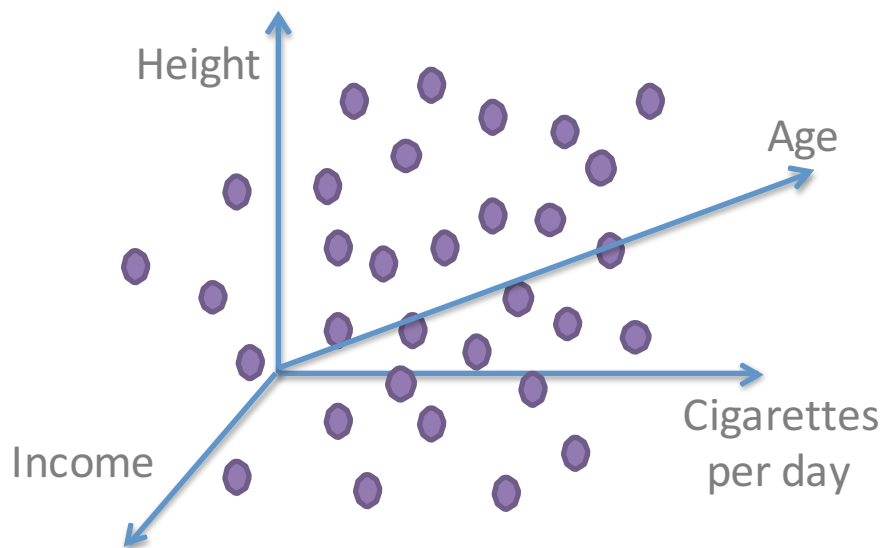
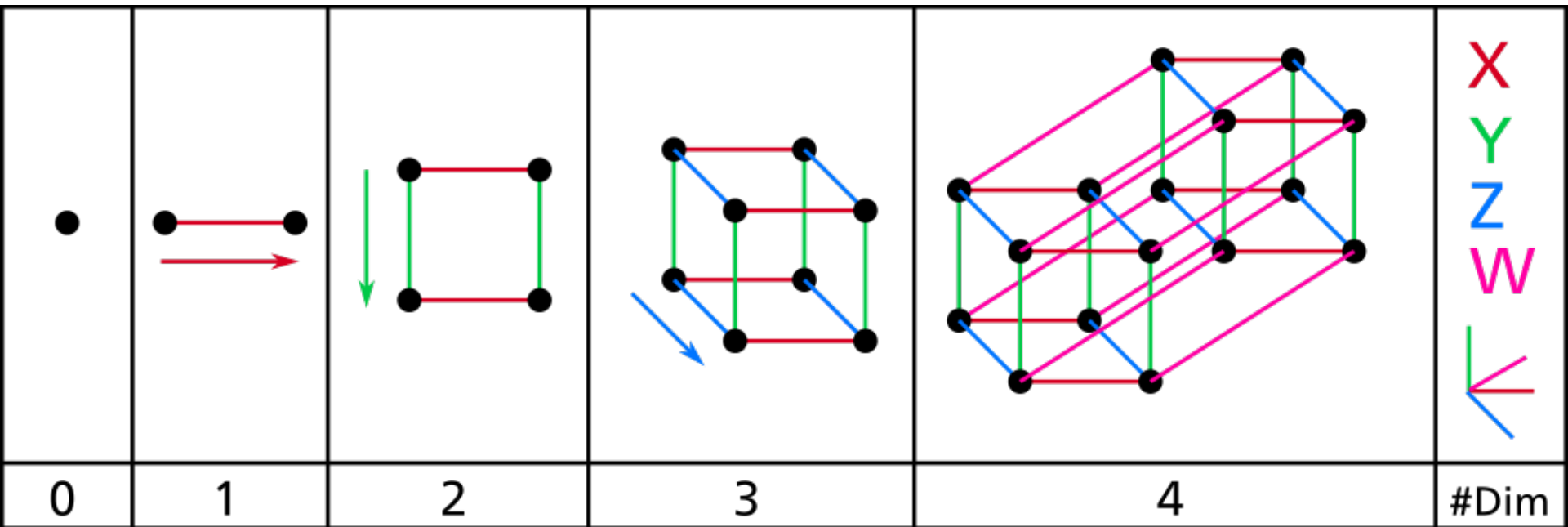
Three dimensions

Curse of Dimensionality



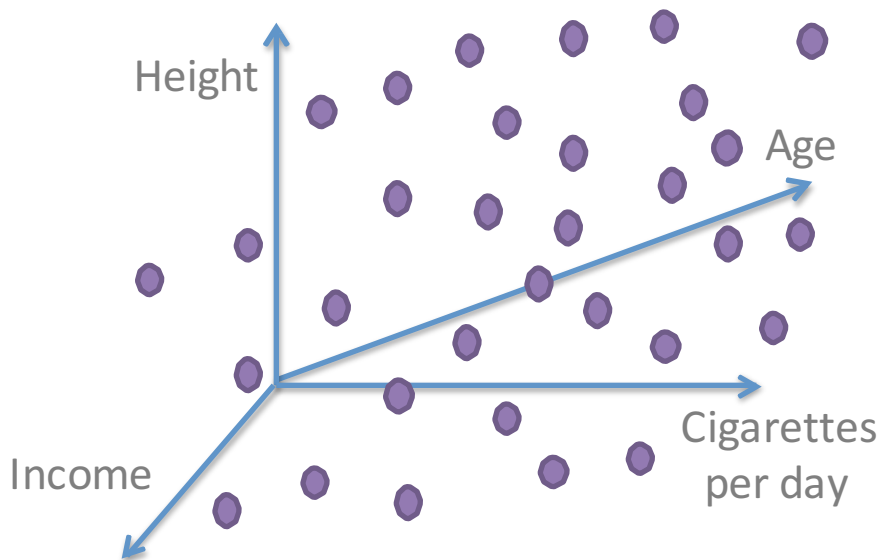
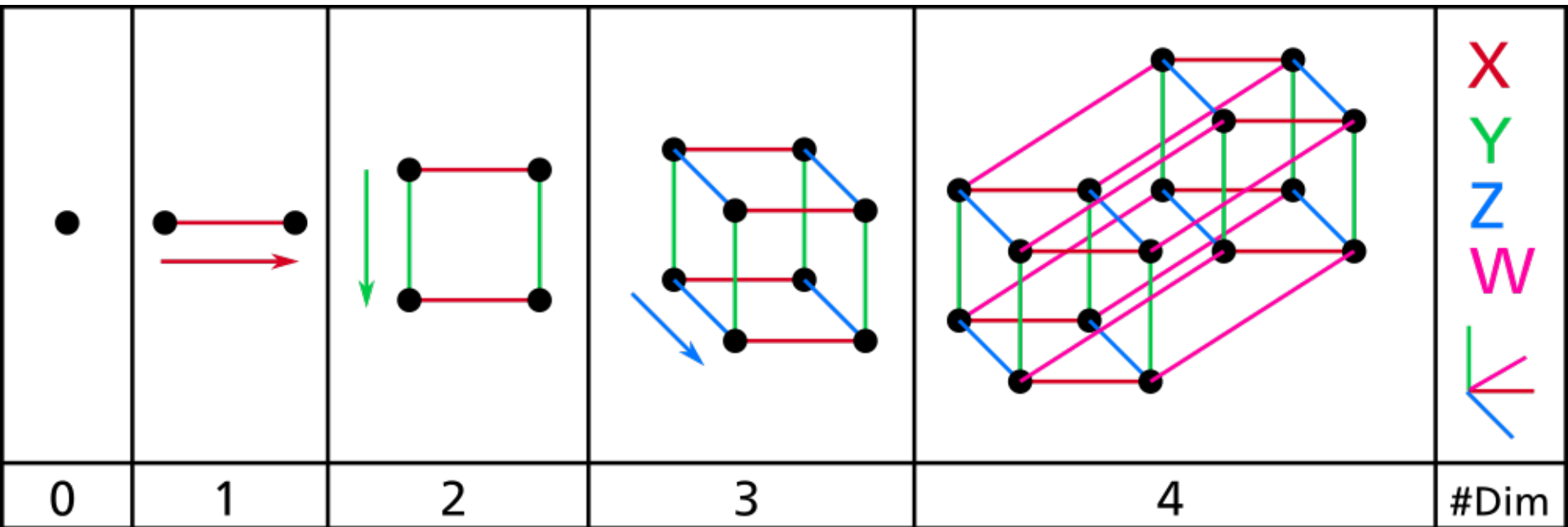
Three dimensions:
Much larger space
Being close less probable

Curse of Dimensionality



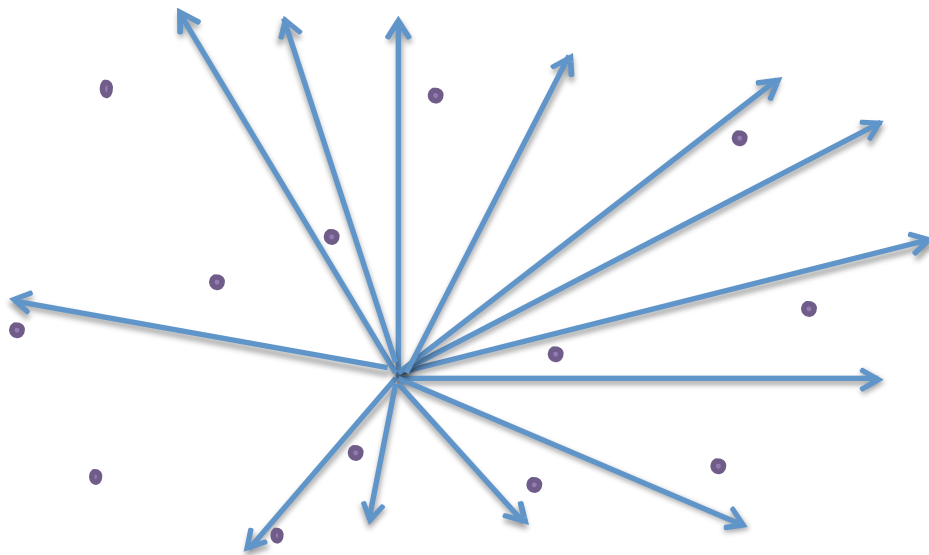
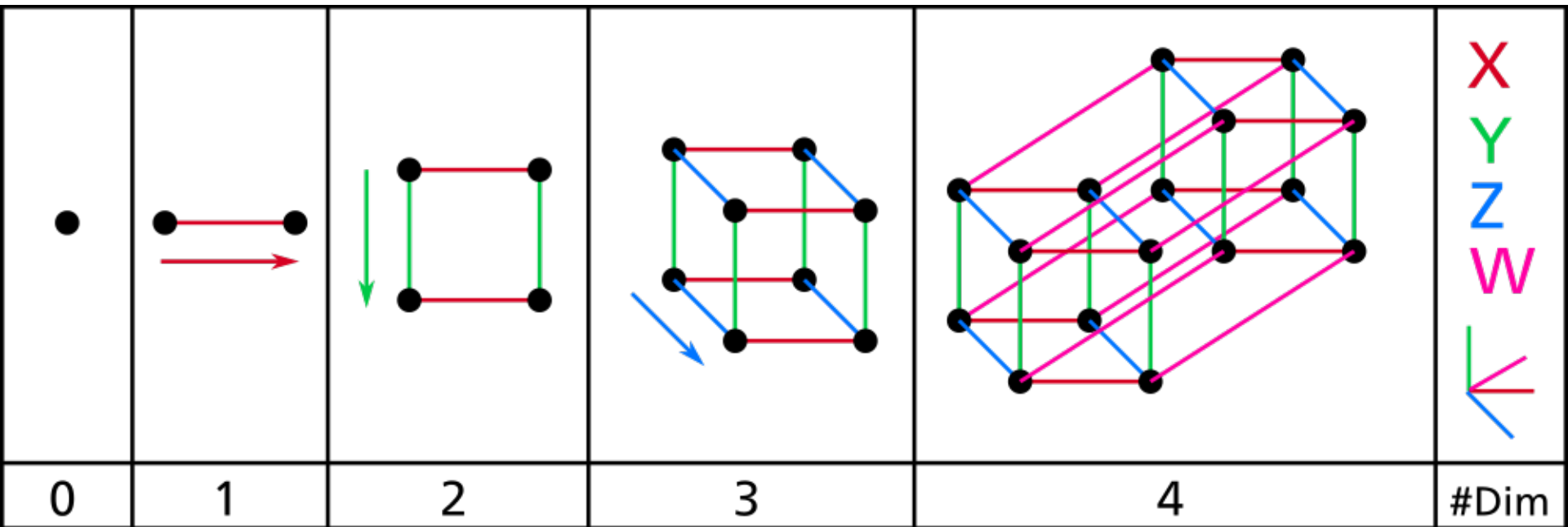
Four dimensions

Curse of Dimensionality



Four dimensions:
omg so much space
Being close quite improbable

Curse of Dimensionality



Thousand dimensions:

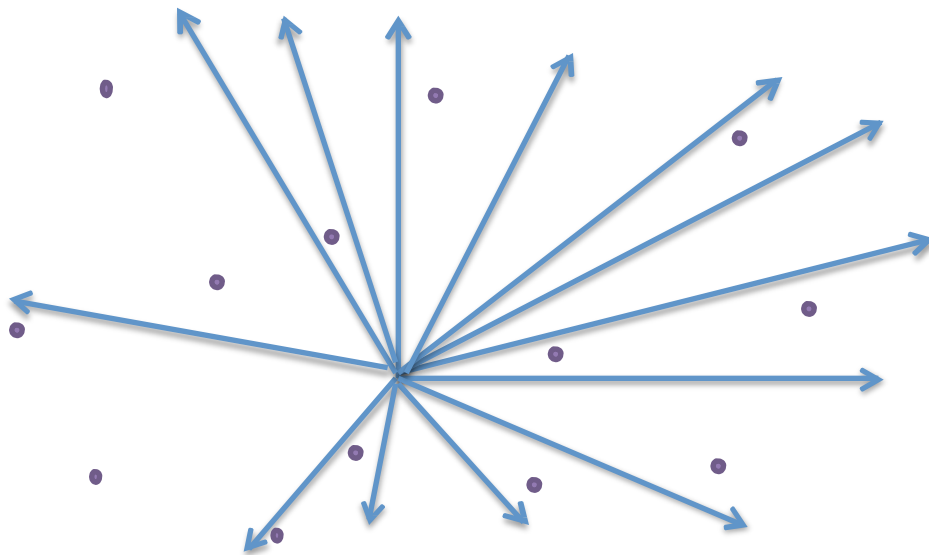
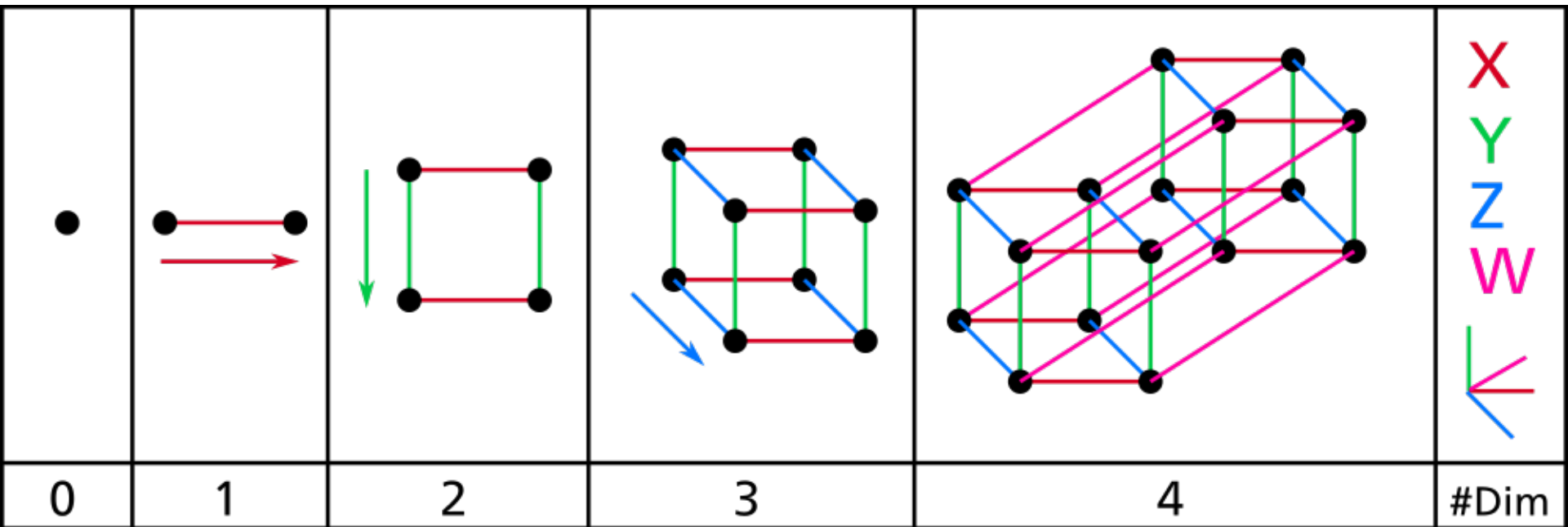
Helloooo... hellooo.. helloo...

Can anybody hear meee.. mee..

mee.. mee..

So alone....

Curse of Dimensionality



Thousand dimensions:

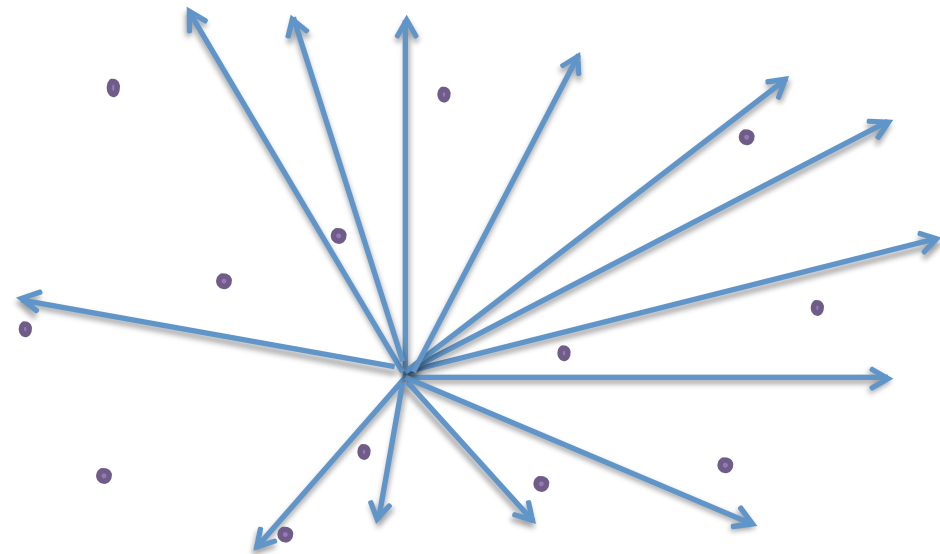
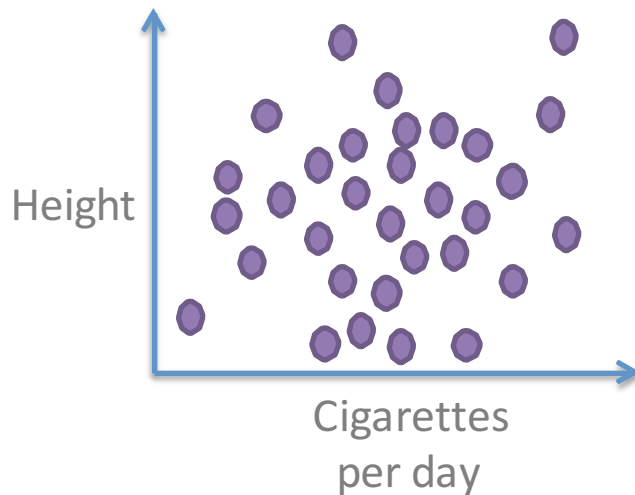
I specified you with such high resolution, with so much detail, that you don't look like anybody else anymore. You're unique.

Curse of Dimensionality

Demonstration of how space grows
(ipython notebook)

Curse of Dimensionality

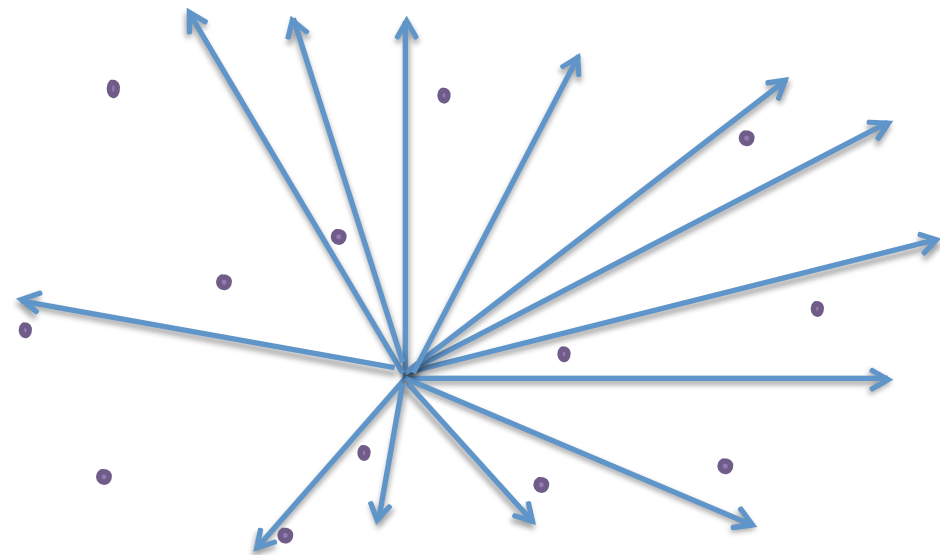
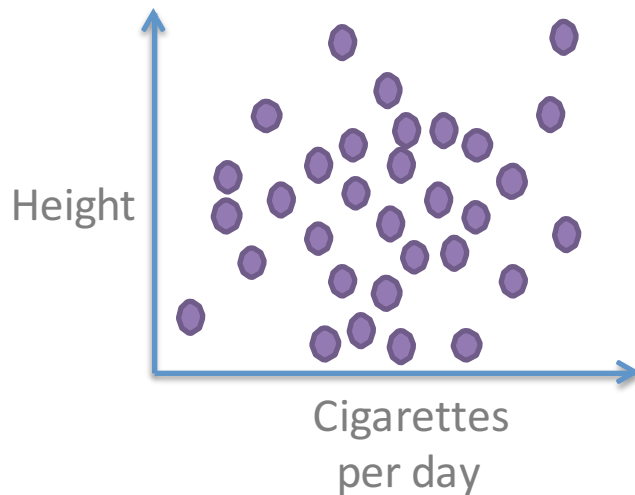
Classification, clustering and other analysis methods become exponentially difficult with increasing dimensions.



Curse of Dimensionality

Classification, clustering and other analysis methods become exponentially difficult with increasing dimensions.

To understand how to divide that huge space, we need a whole lot more data (usually much more than we do or can have).

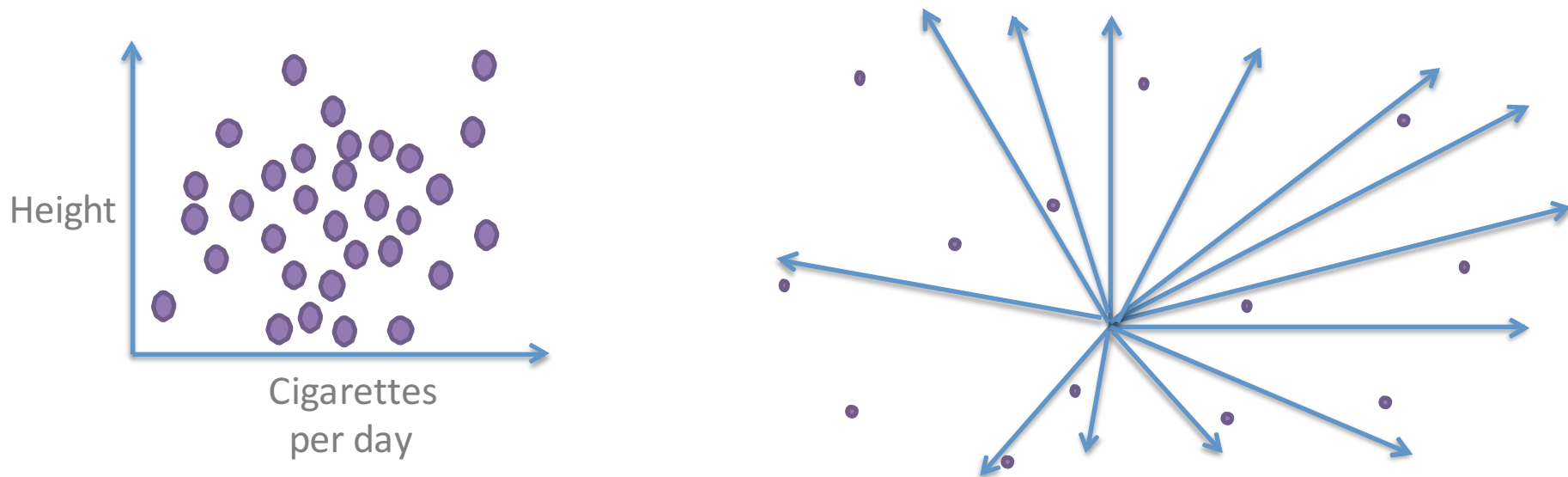


Dimensionality Reduction

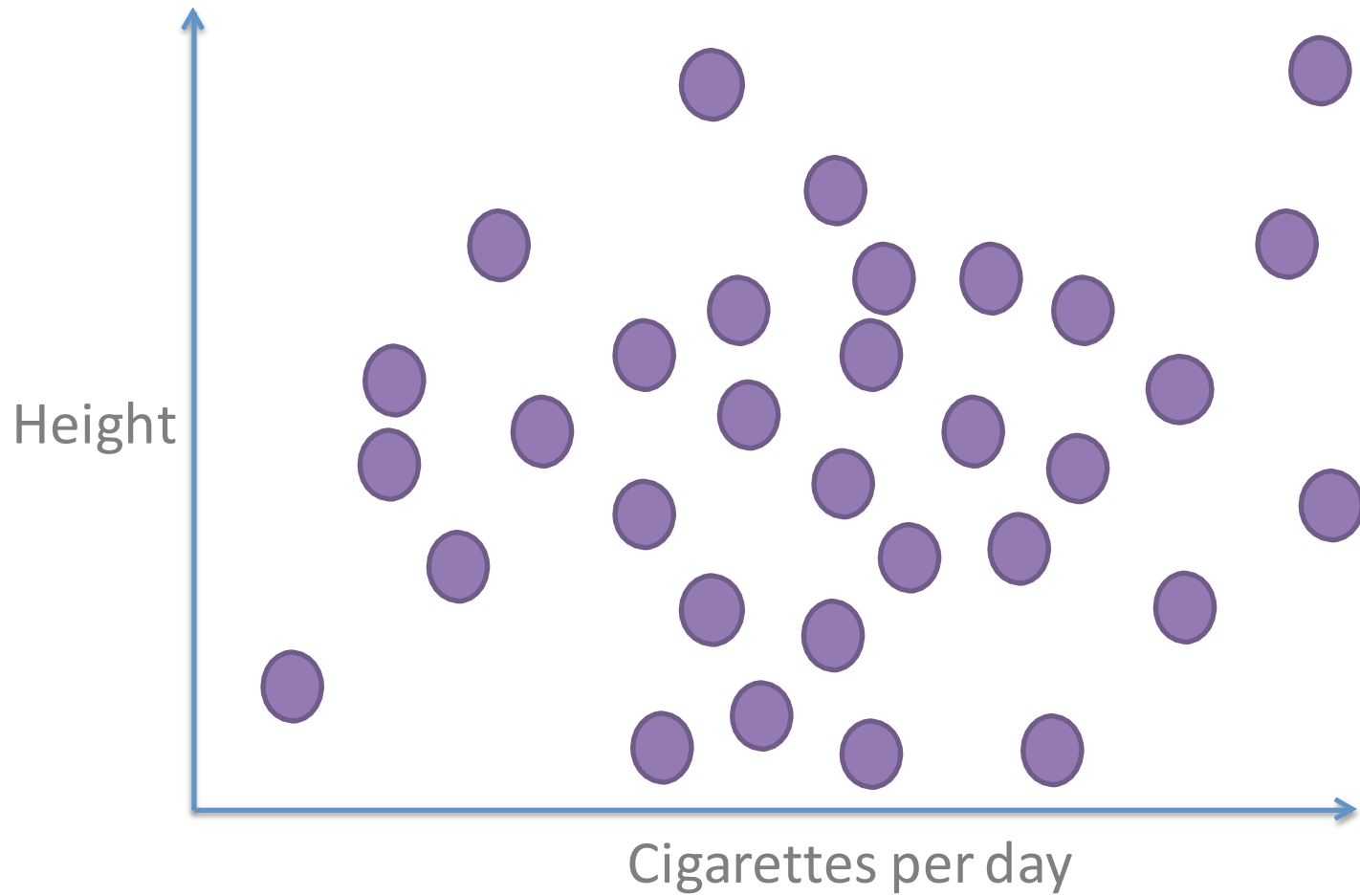
Lots of features, lots of data is best. But what if you don't have the luxury of ginormous amounts of data?

Not all features provide the same amount of information.

We can reduce the dimensions (compress the data) without necessarily losing too much information.

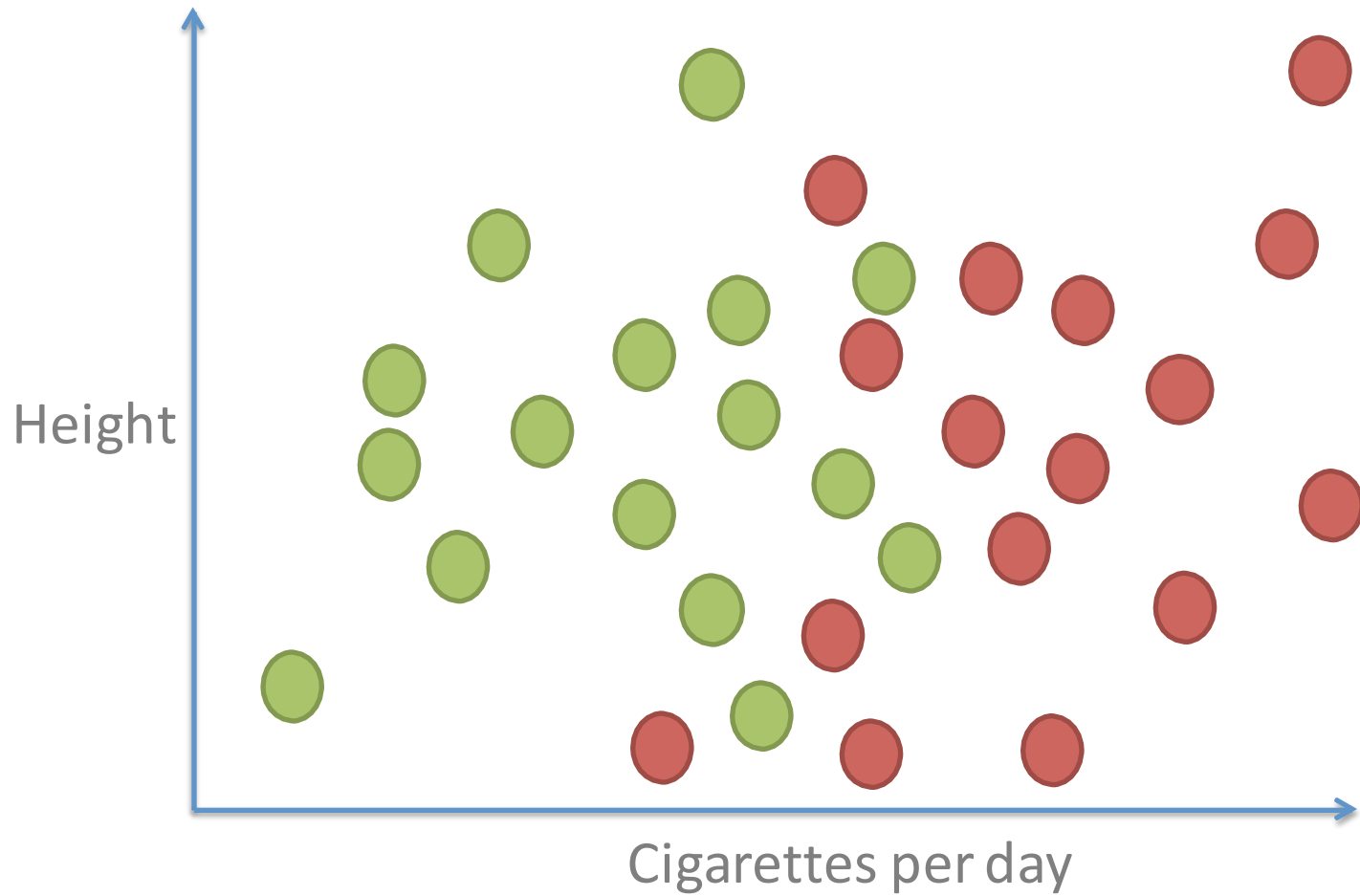


Feature Selection



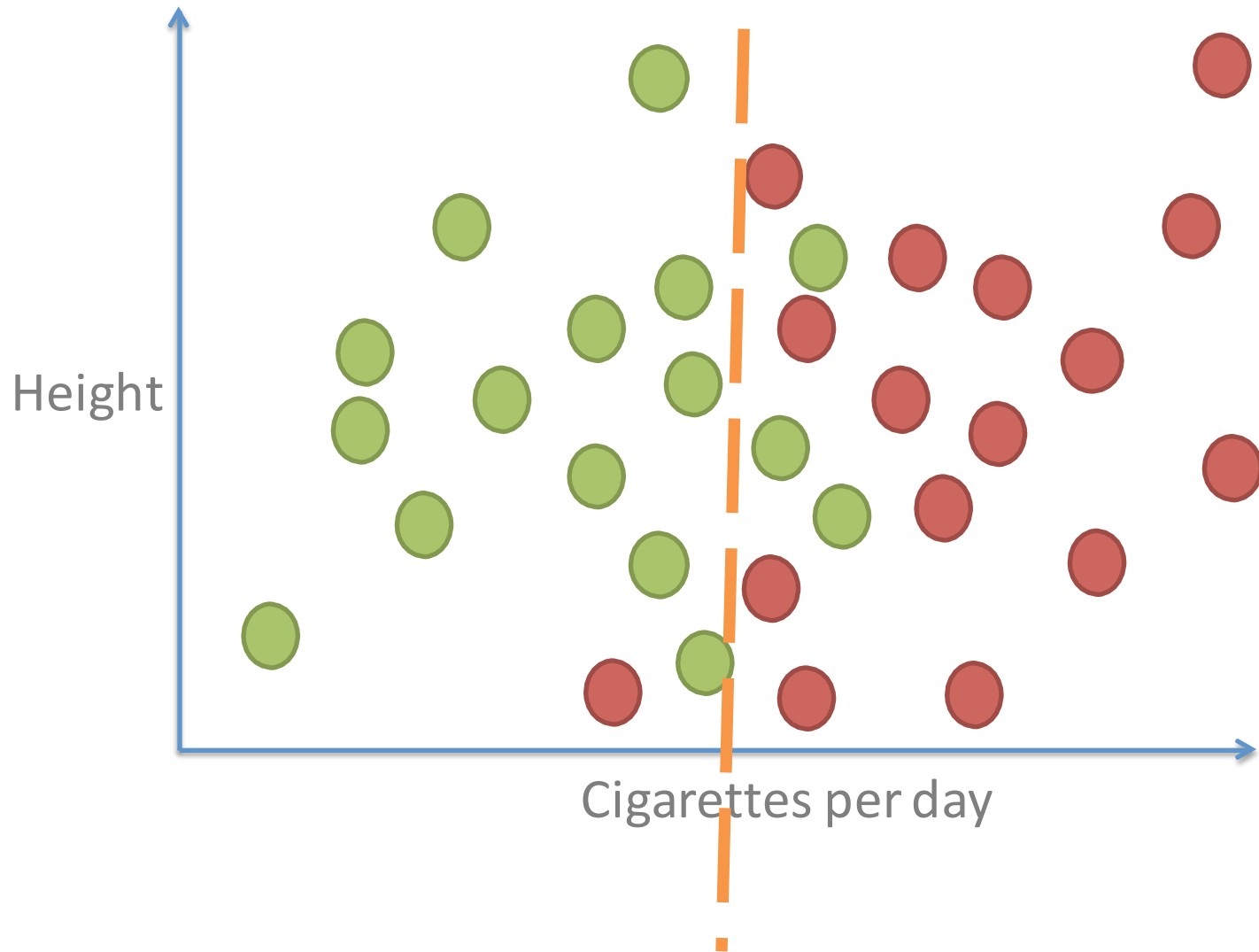
Feature Selection

Healthy / Heart Disease



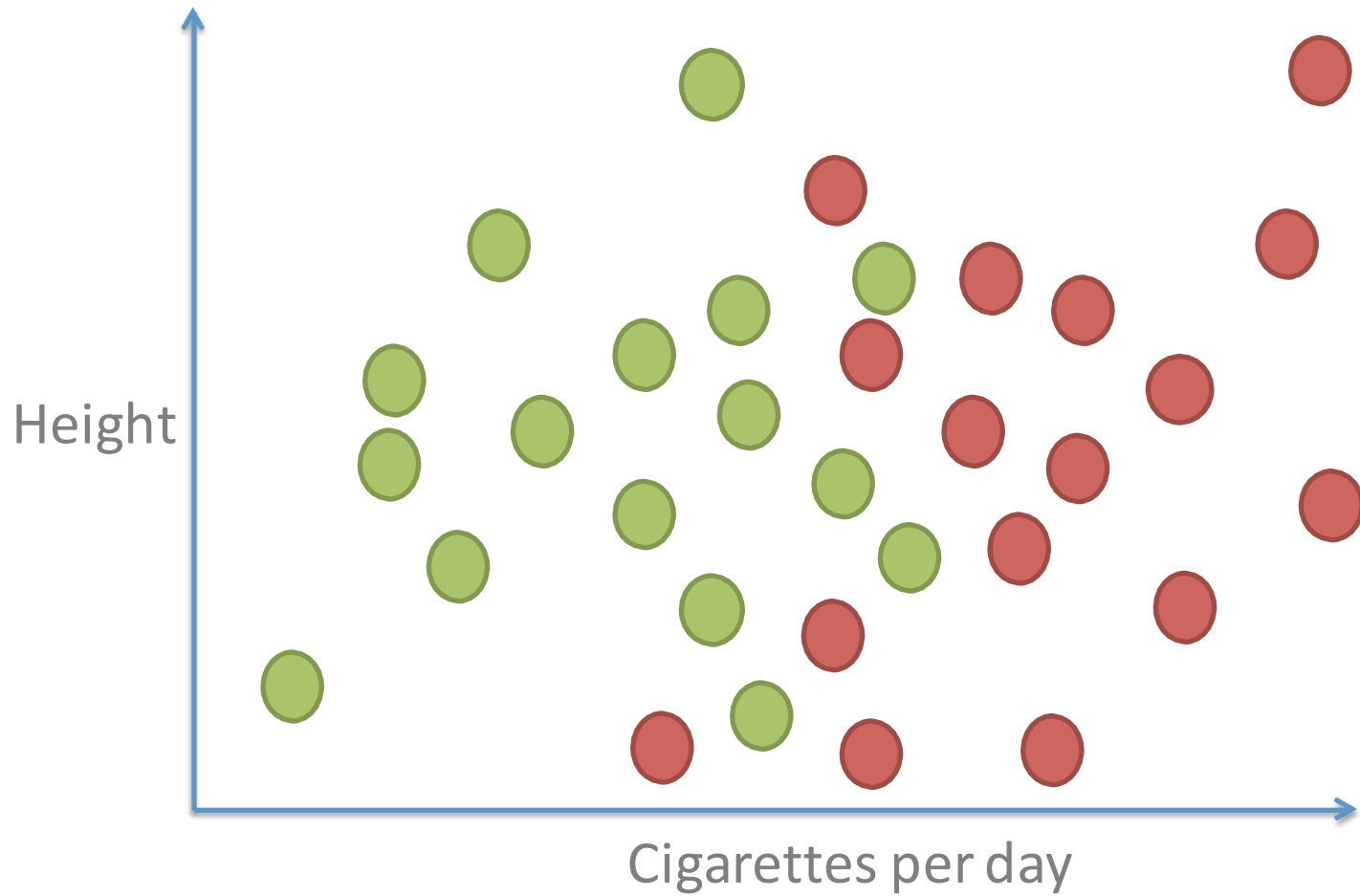
Feature Selection

Healthy / Heart Disease



Feature Selection

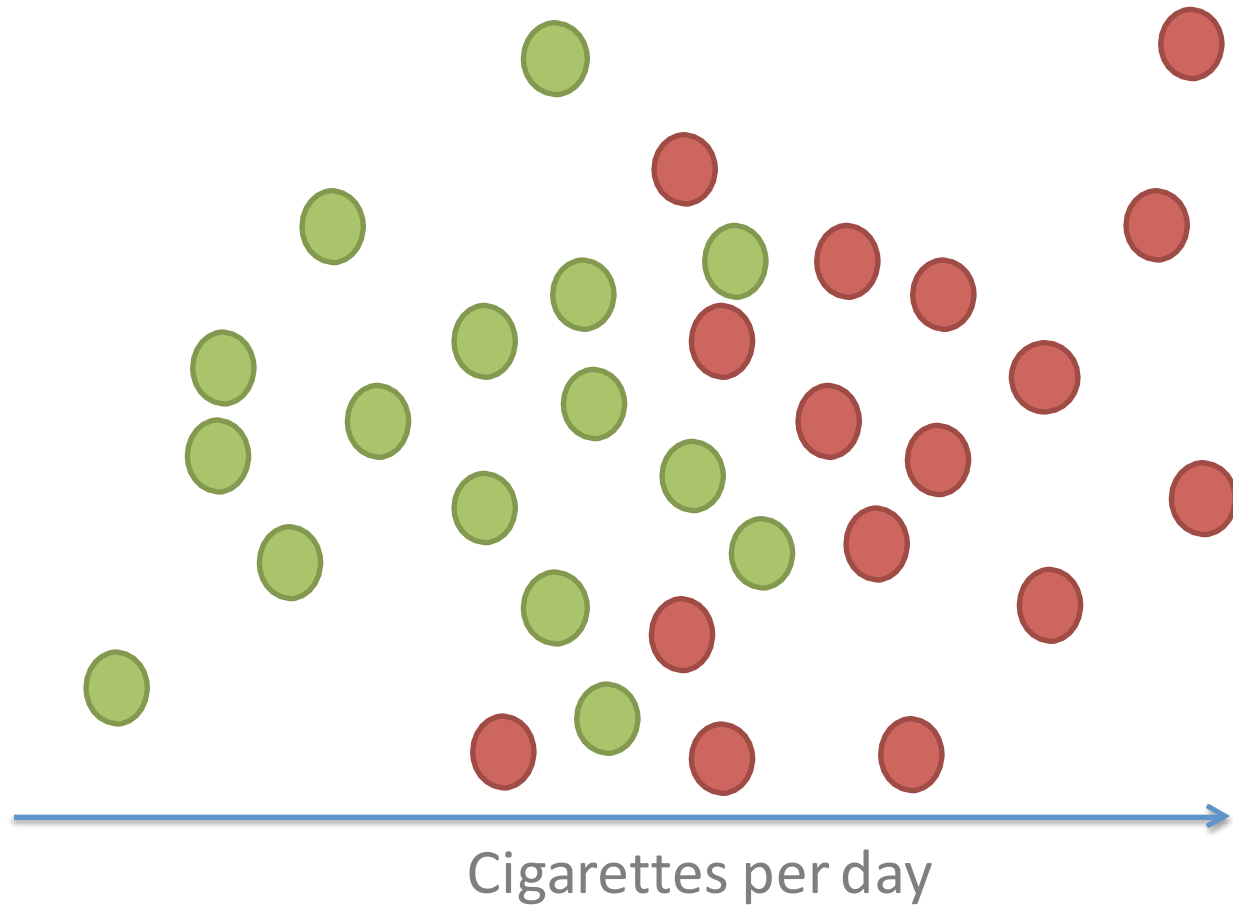
Healthy / Heart Disease



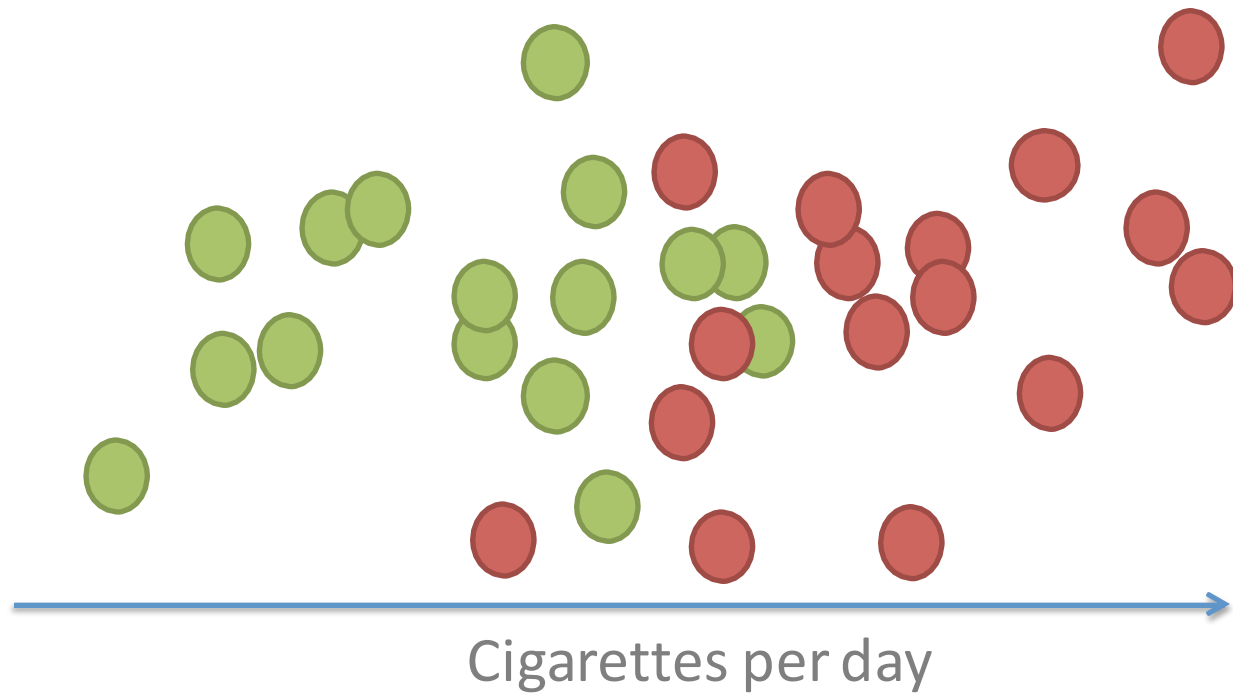
Feature Selection

Healthy / Heart Disease

Height

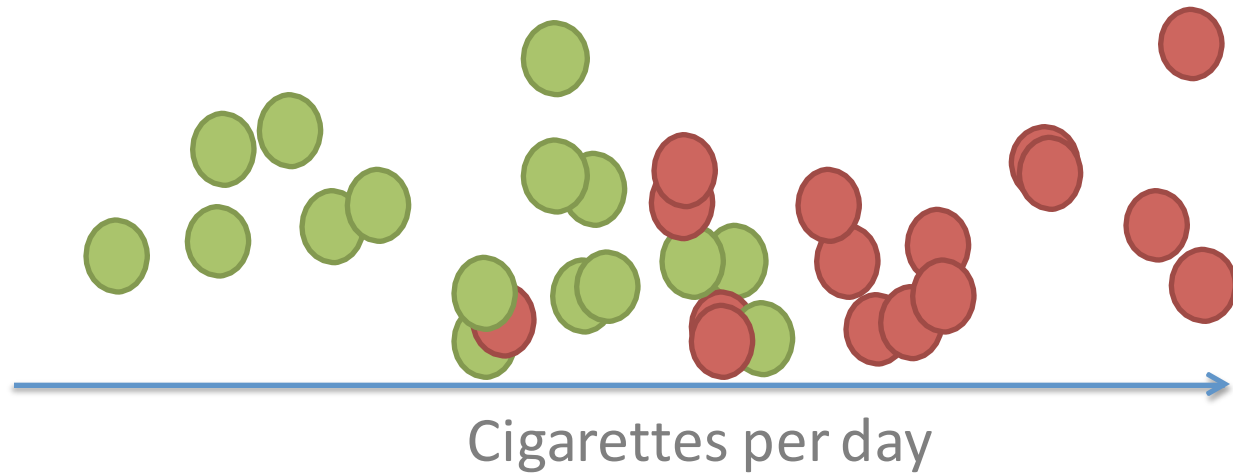


Healthy / Heart Disease



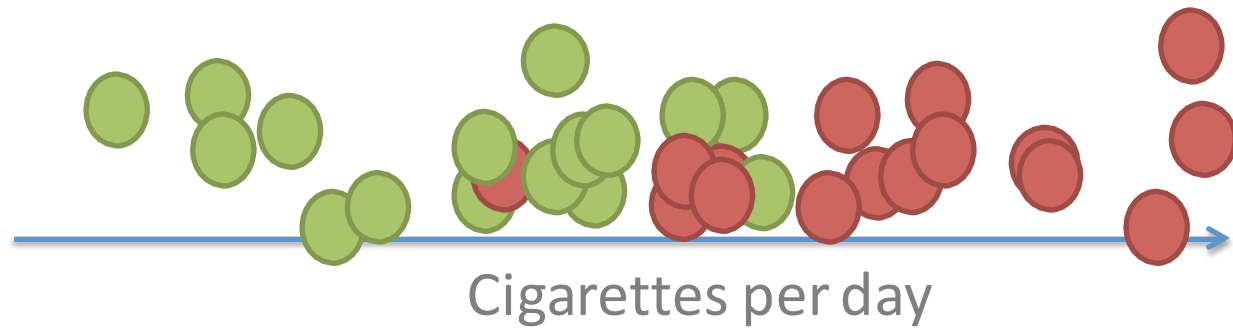
Feature Selection

Healthy / Heart Disease



Feature Selection

Healthy / Heart Disease



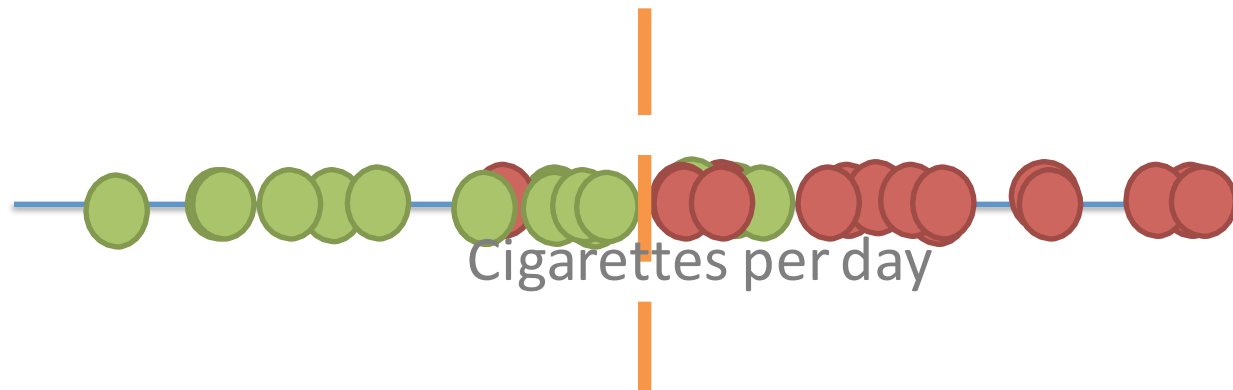
Healthy / Heart Disease

Healthy / Heart Disease

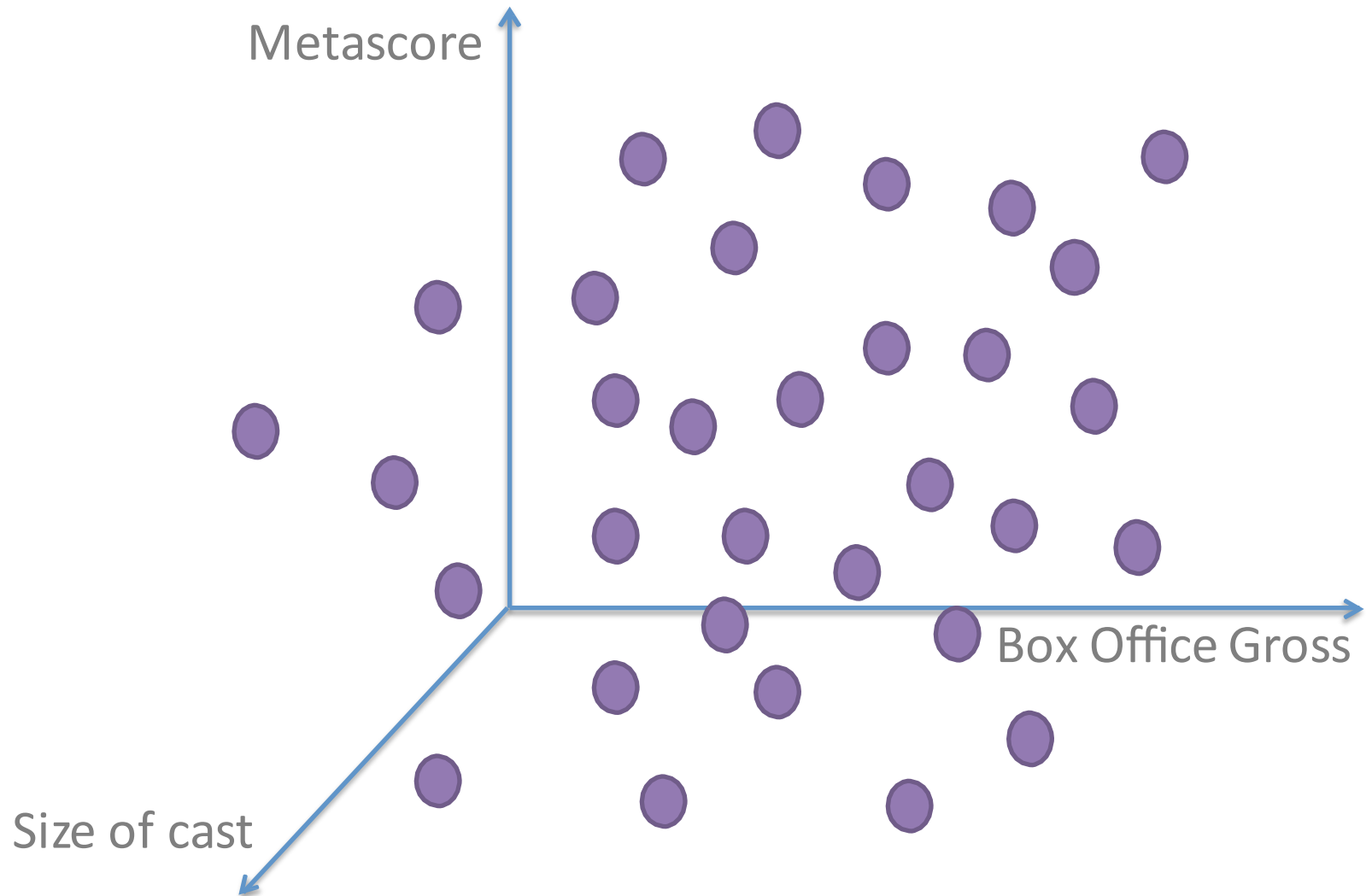


Feature Selection

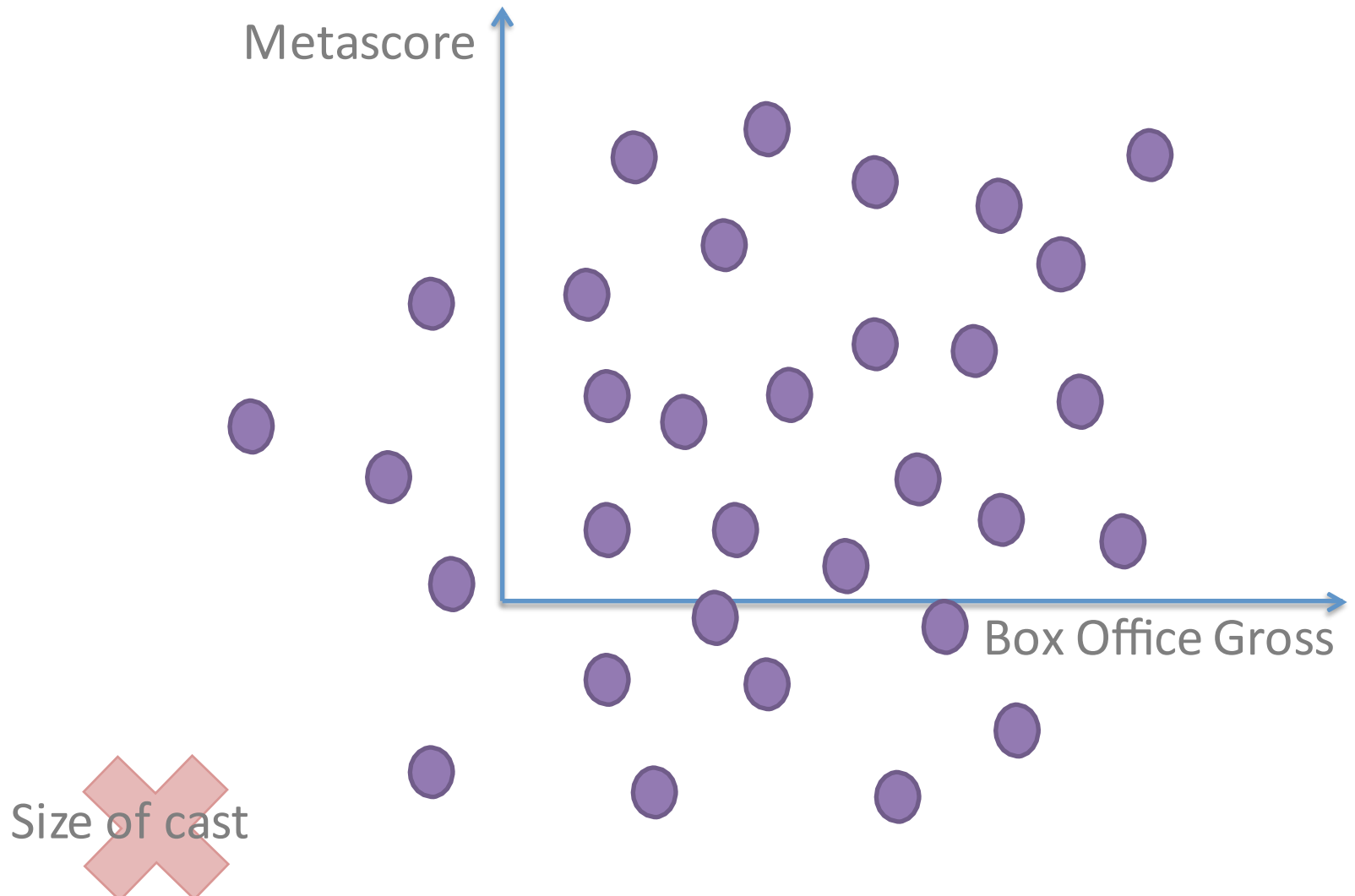
Healthy / Heart Disease



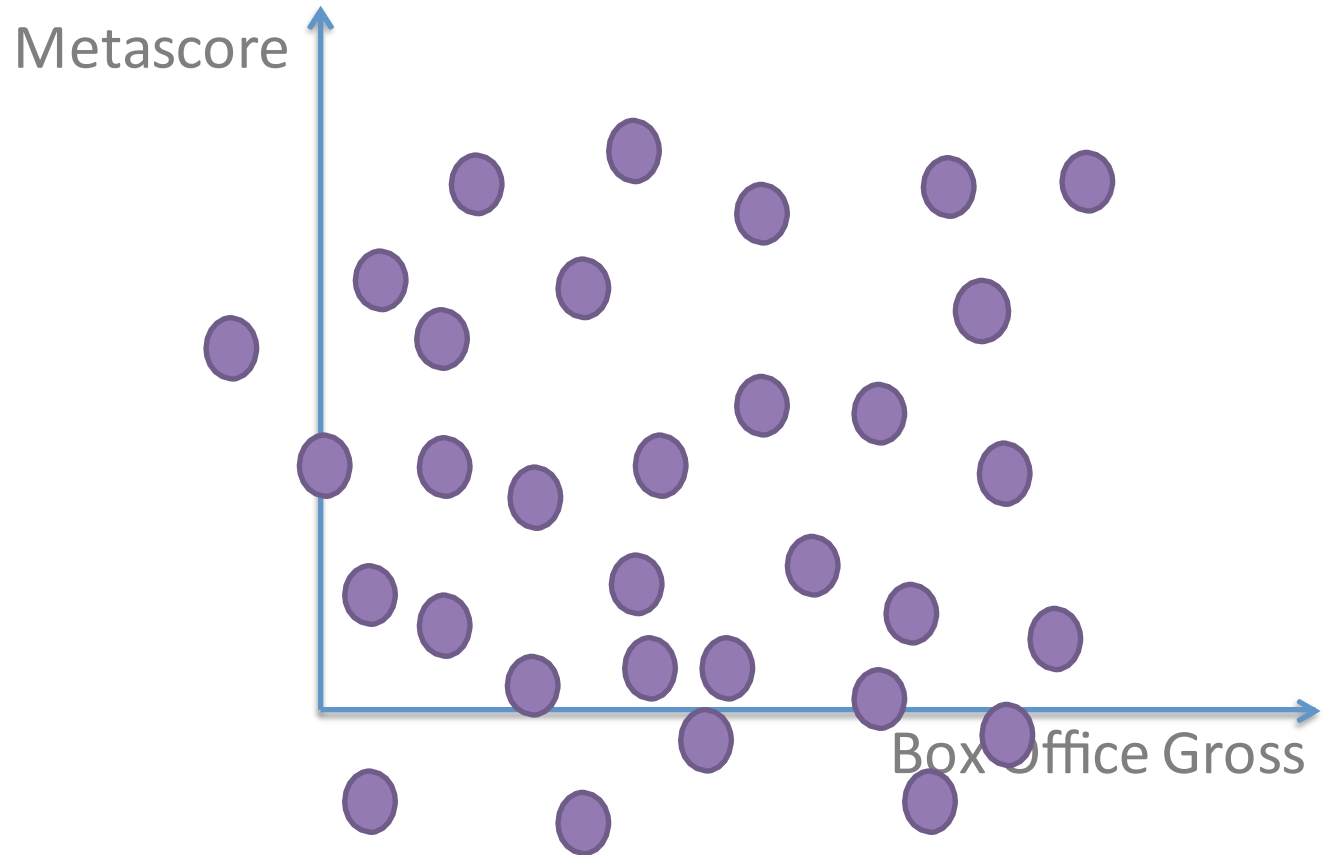
Feature Selection



Feature Selection

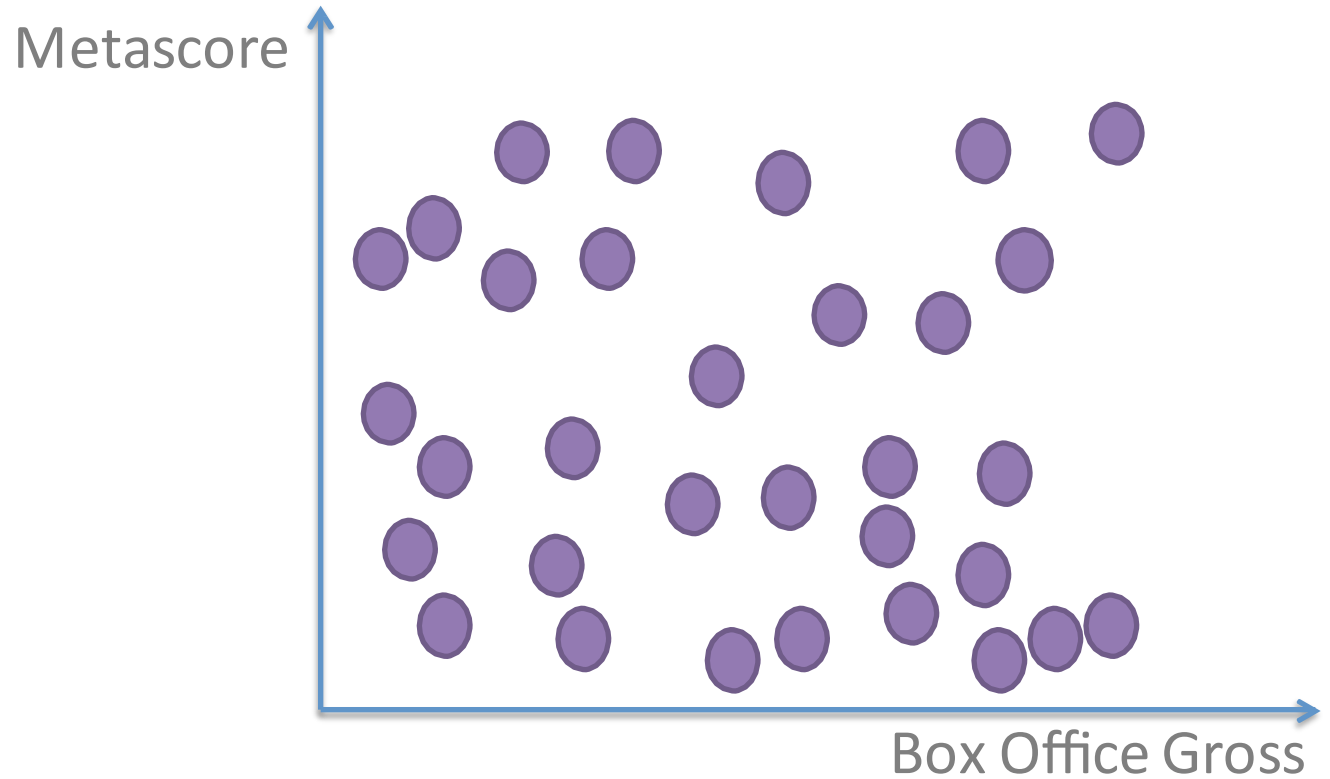


Feature Selection



Size of cast

Feature Selection



Feature Selection

Common Sense

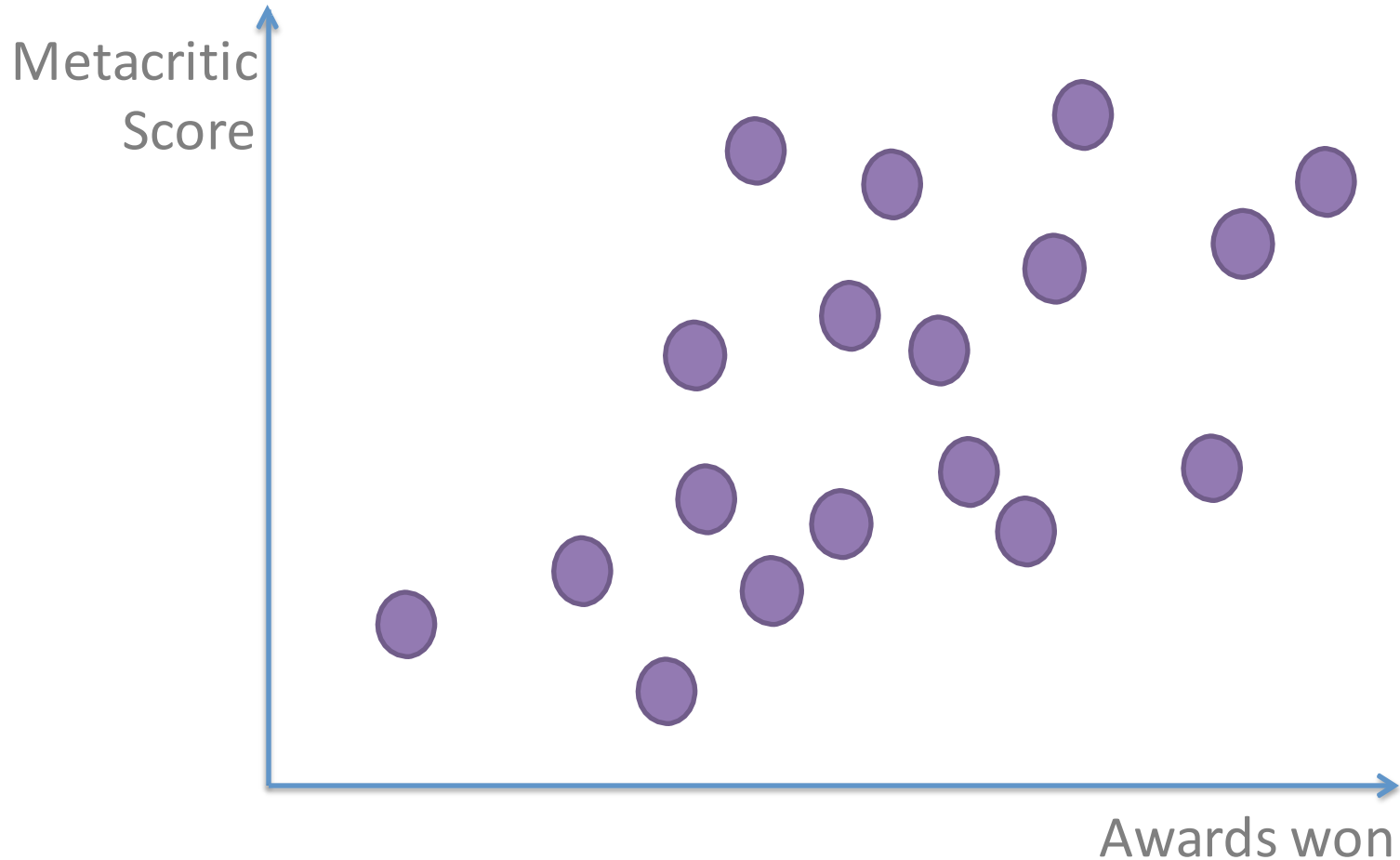
Experiments (remove a feature, fit again, evaluate results)

Regularization (in regression)

Feature scores (χ^2 , F-test, etc. : `SelectKBest()`)

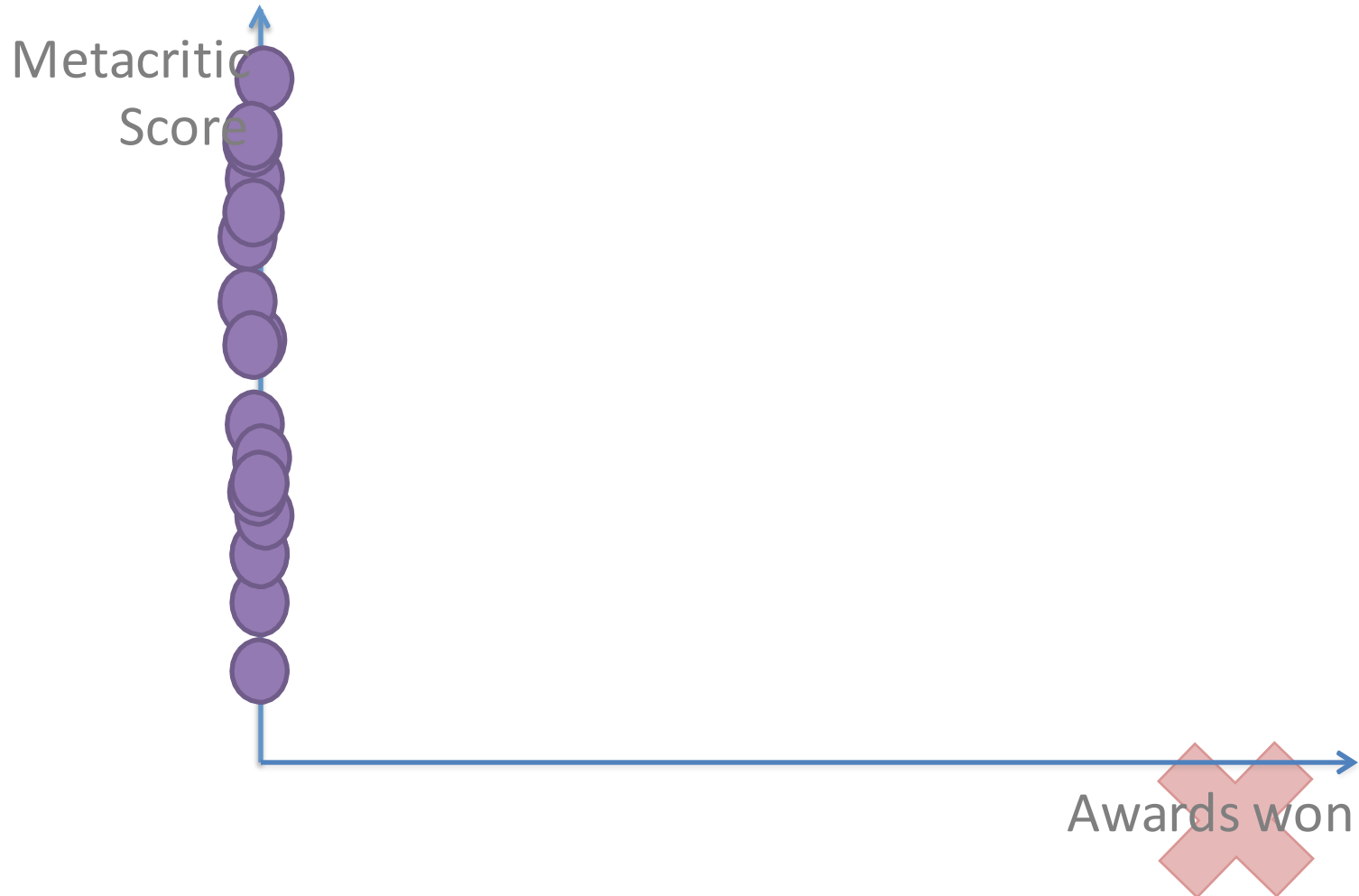
Feature Extraction

Do I have to choose the dimensions among existing features?

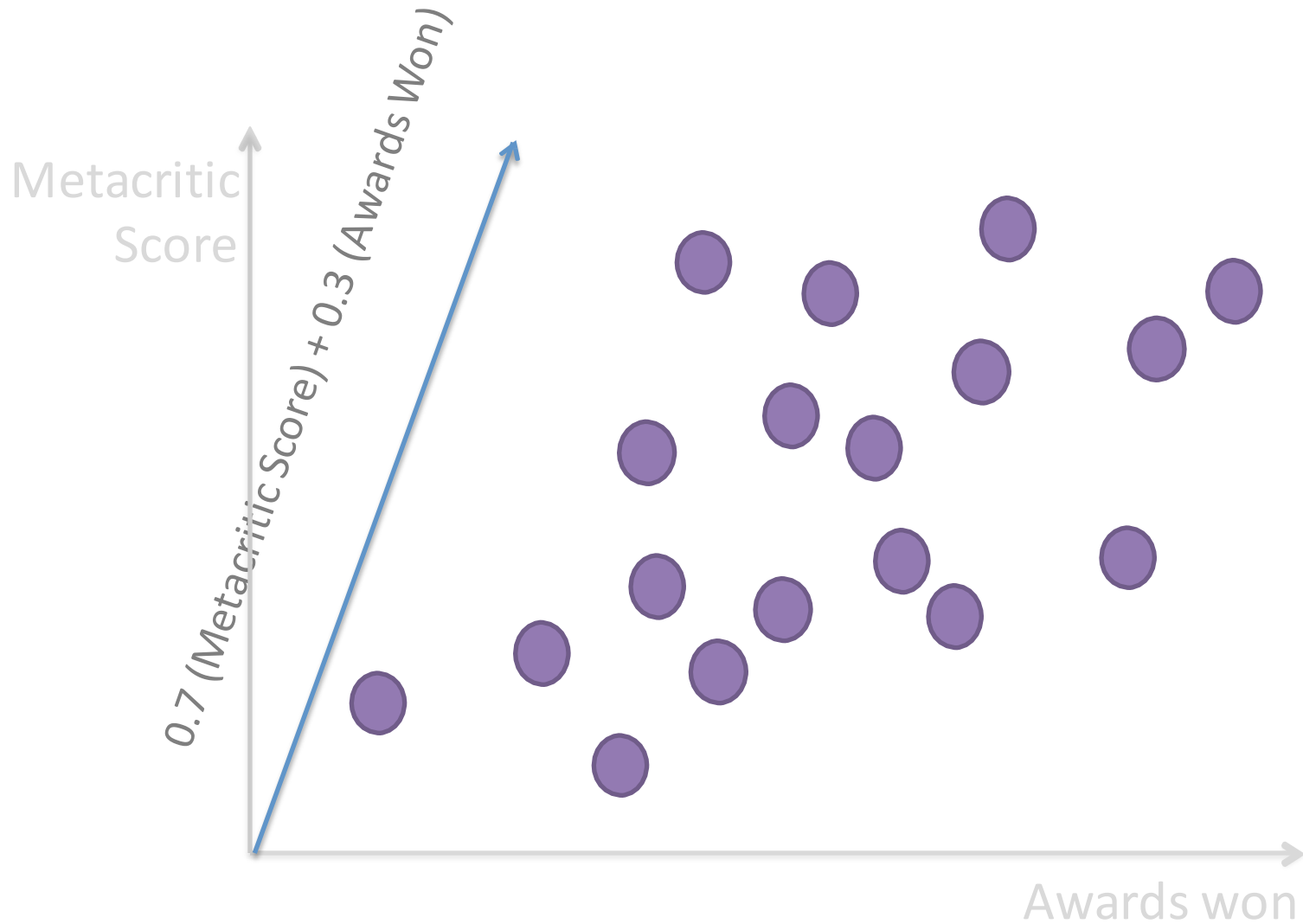


Feature Extraction

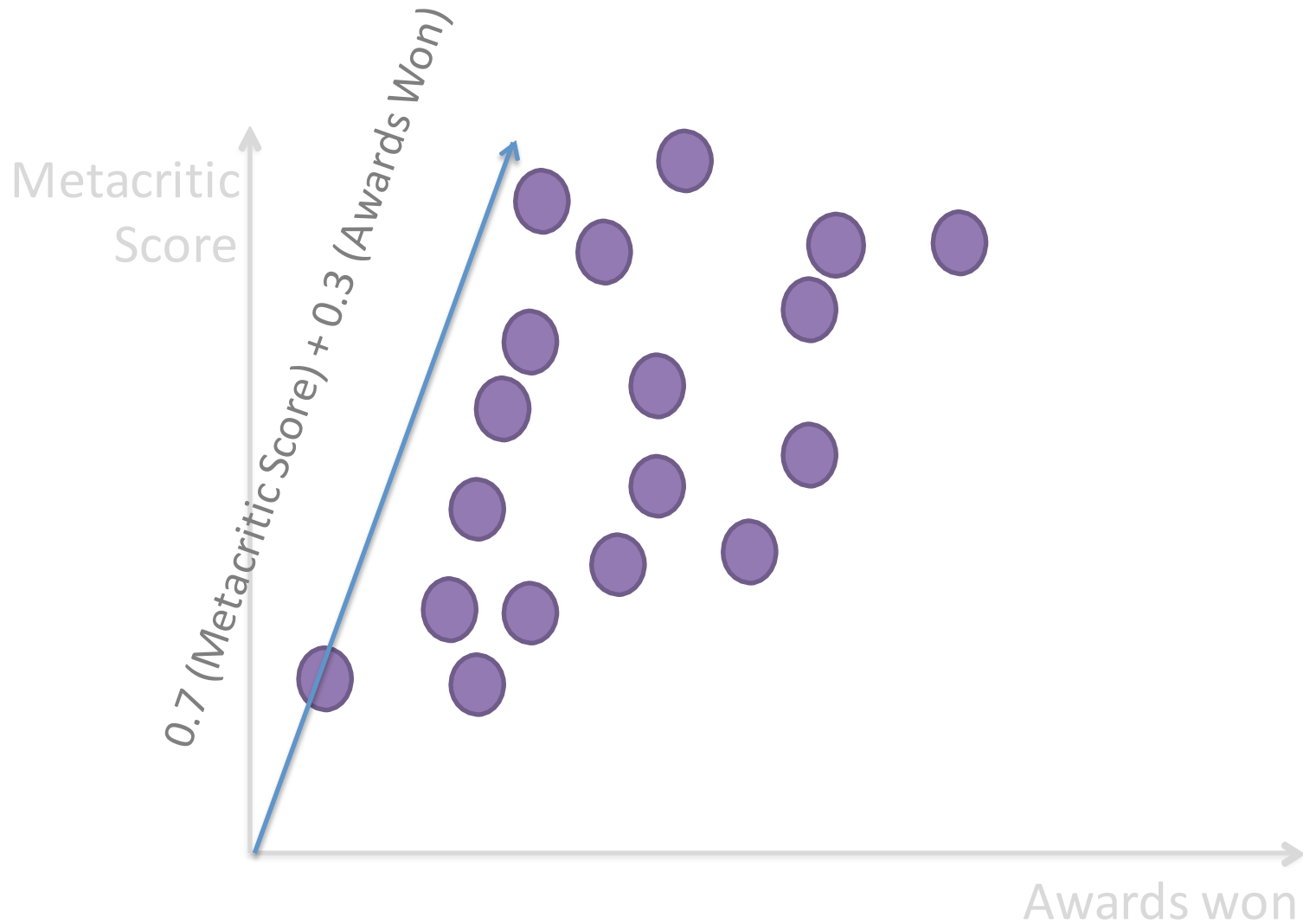
Do I have to choose the dimensions among existing features?



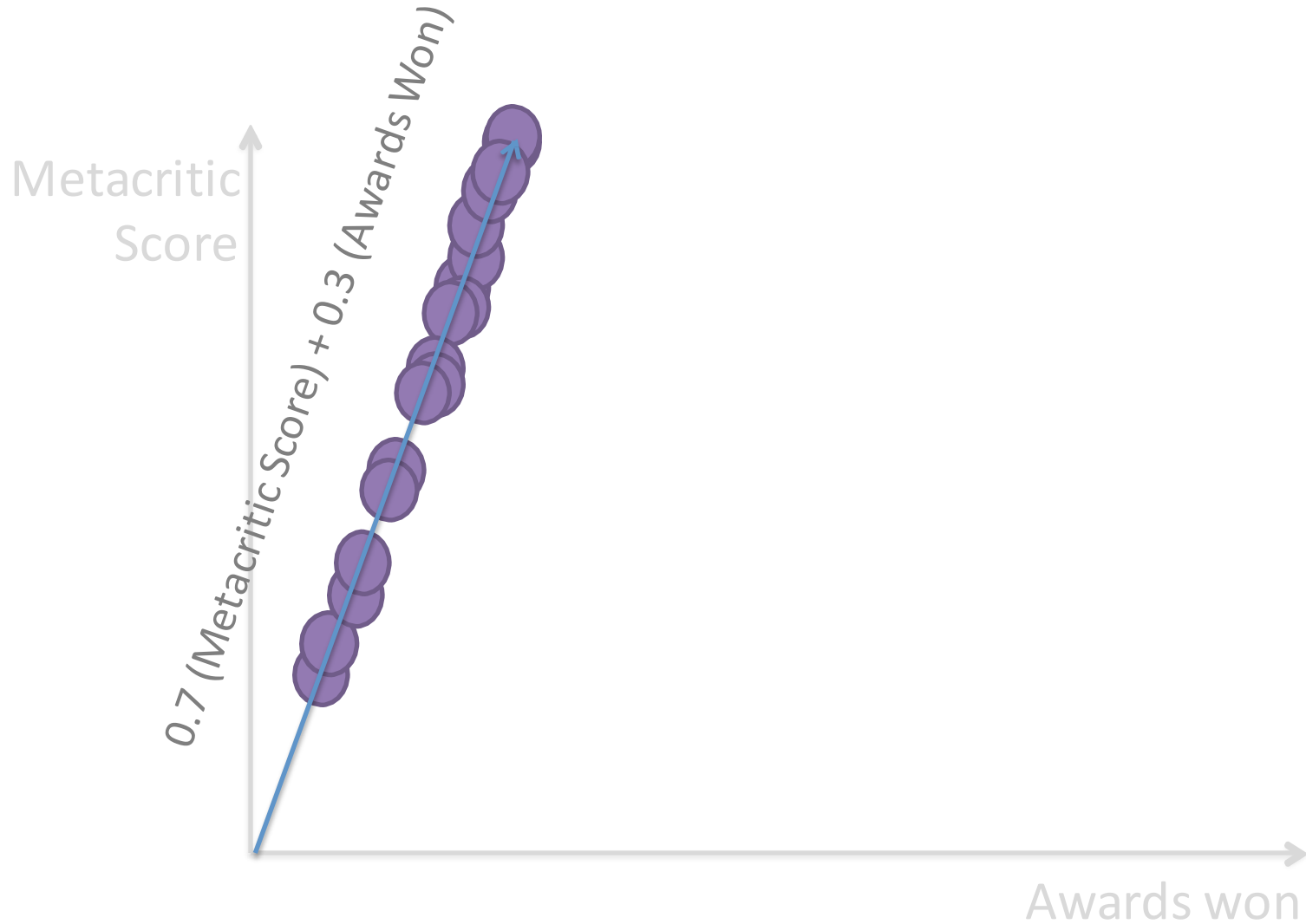
Feature Extraction



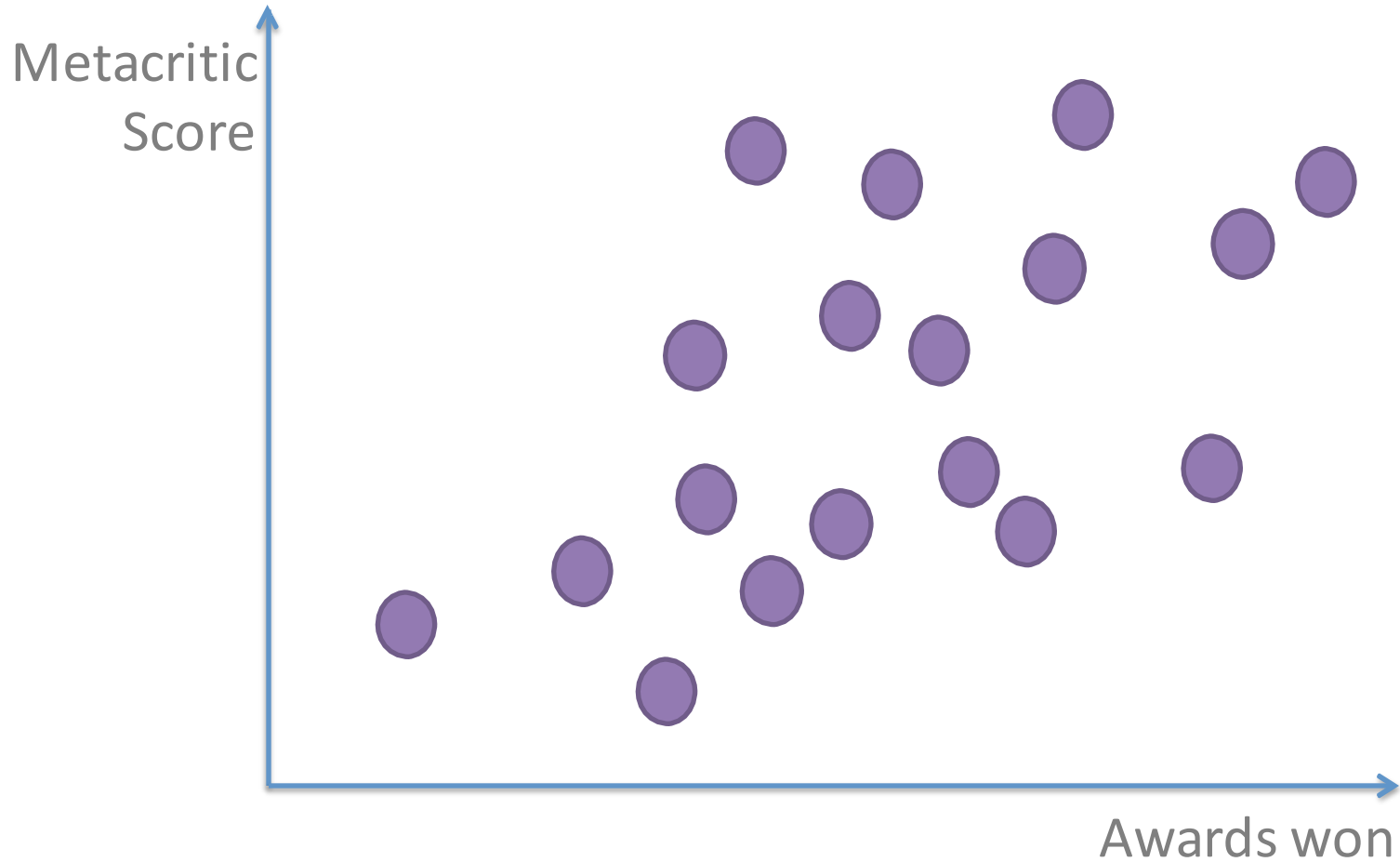
Feature Extraction



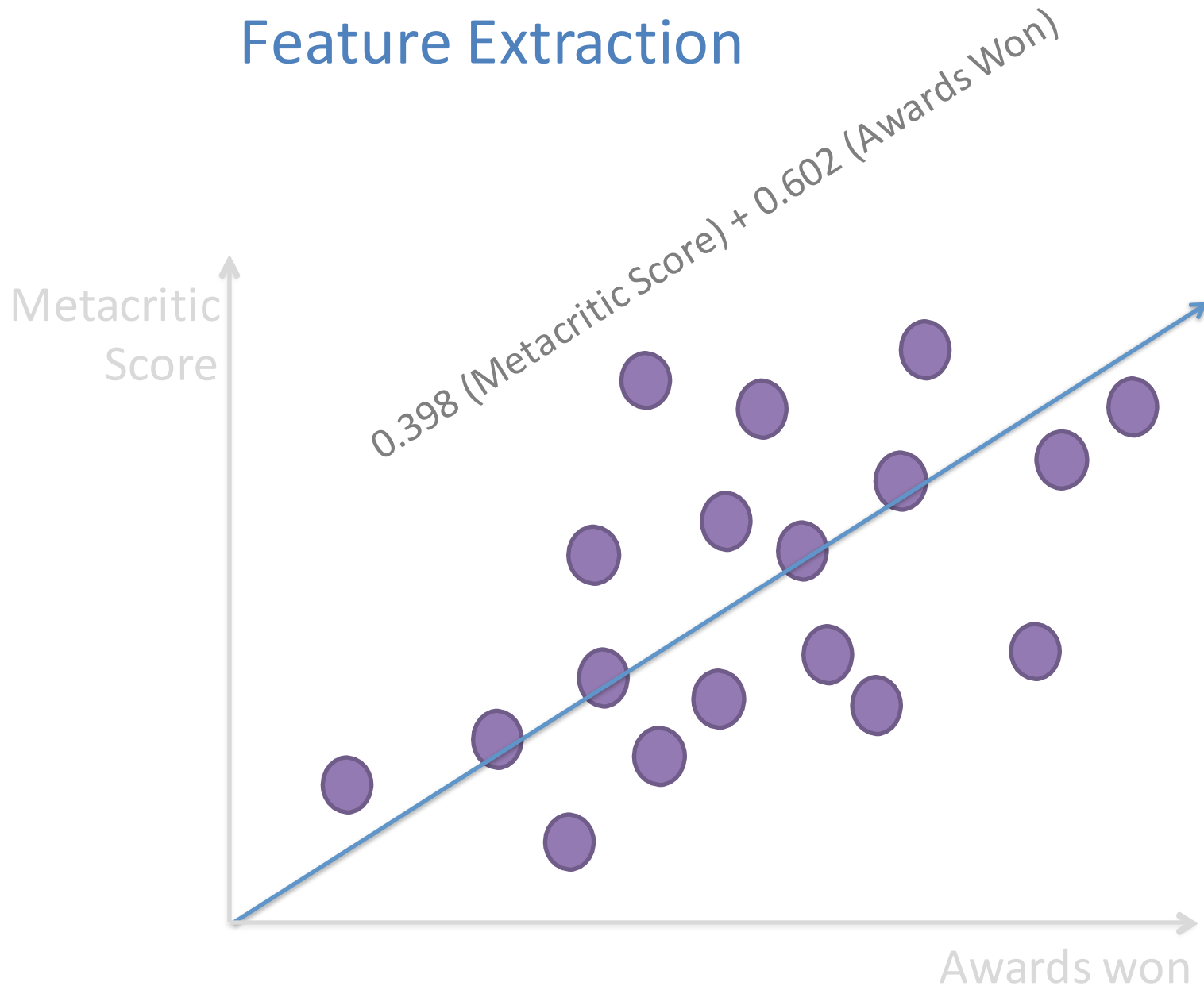
Feature Extraction



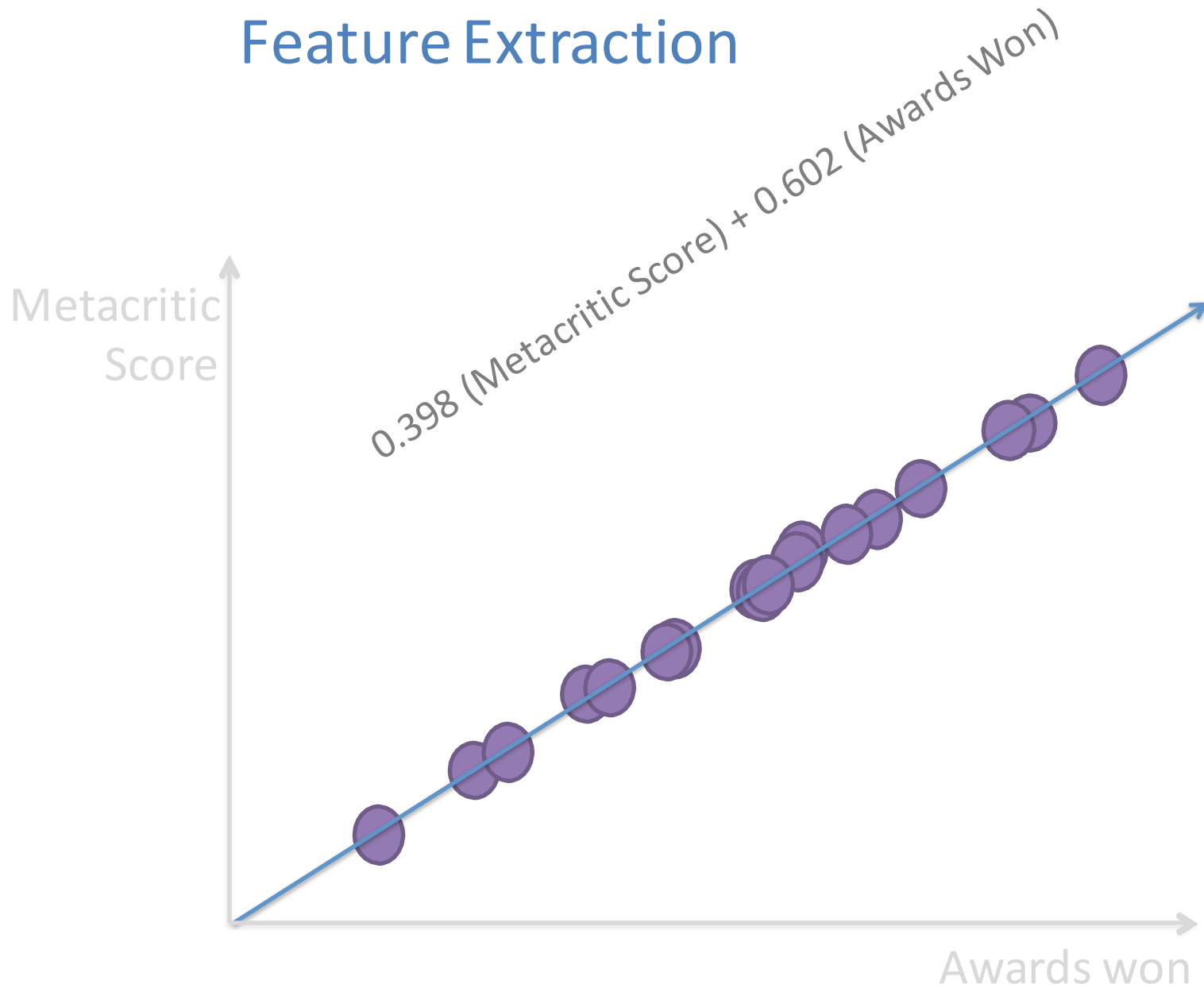
Feature Extraction



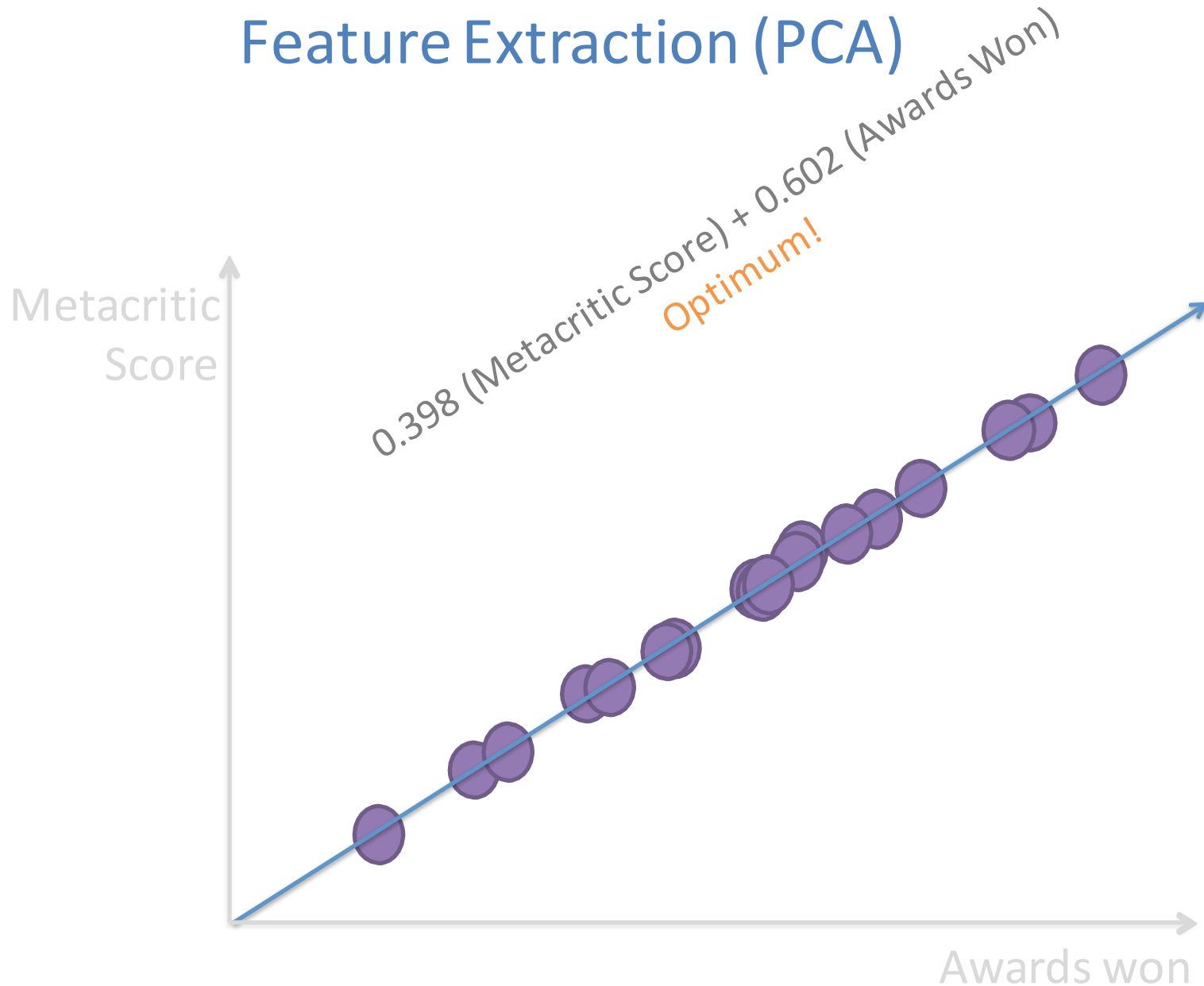
Feature Extraction



Feature Extraction



Feature Extraction (PCA)



Feature Extraction

Advantage: You retain more information

Disadvantage: You lose interpretability

Feature Extraction

Advantage: You retain more information

Disadvantage: You lose interpretability

2D

$$\text{Oscar_or_not} = \text{logit}(\beta_1(\text{Metascore}) + \beta_2(\text{Awards}))$$

Feature Extraction

Advantage: You retain more information

Disadvantage: You lose interpretability

2D

$$\text{Oscar_or_not} = \text{logit}(\beta_1(\text{Metascore}) + \beta_2(\text{Awards}))$$

Feature selection 1D

$$\text{Oscar_or_not} = \text{logit}(\beta_1(\text{Metascore}))$$

Feature Extraction

Advantage: You retain more information

Disadvantage: You lose interpretability

2D

$$\text{Oscar_or_not} = \text{logit}(\beta_1(\text{Metascore}) + \beta_2(\text{Awards}))$$

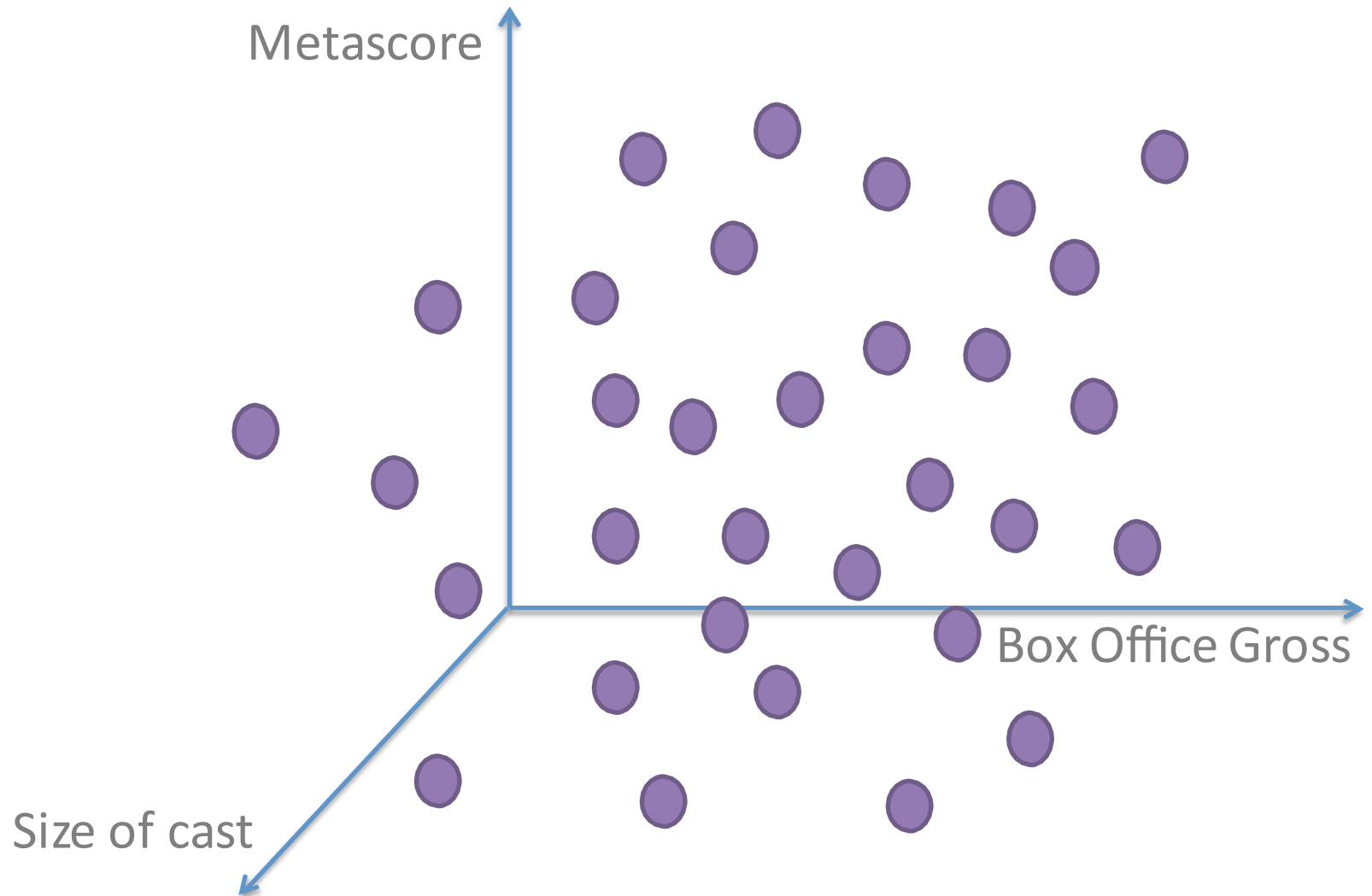
Feature selection 1D

$$\text{Oscar_or_not} = \text{logit}(\beta_1(\text{Metascore}))$$

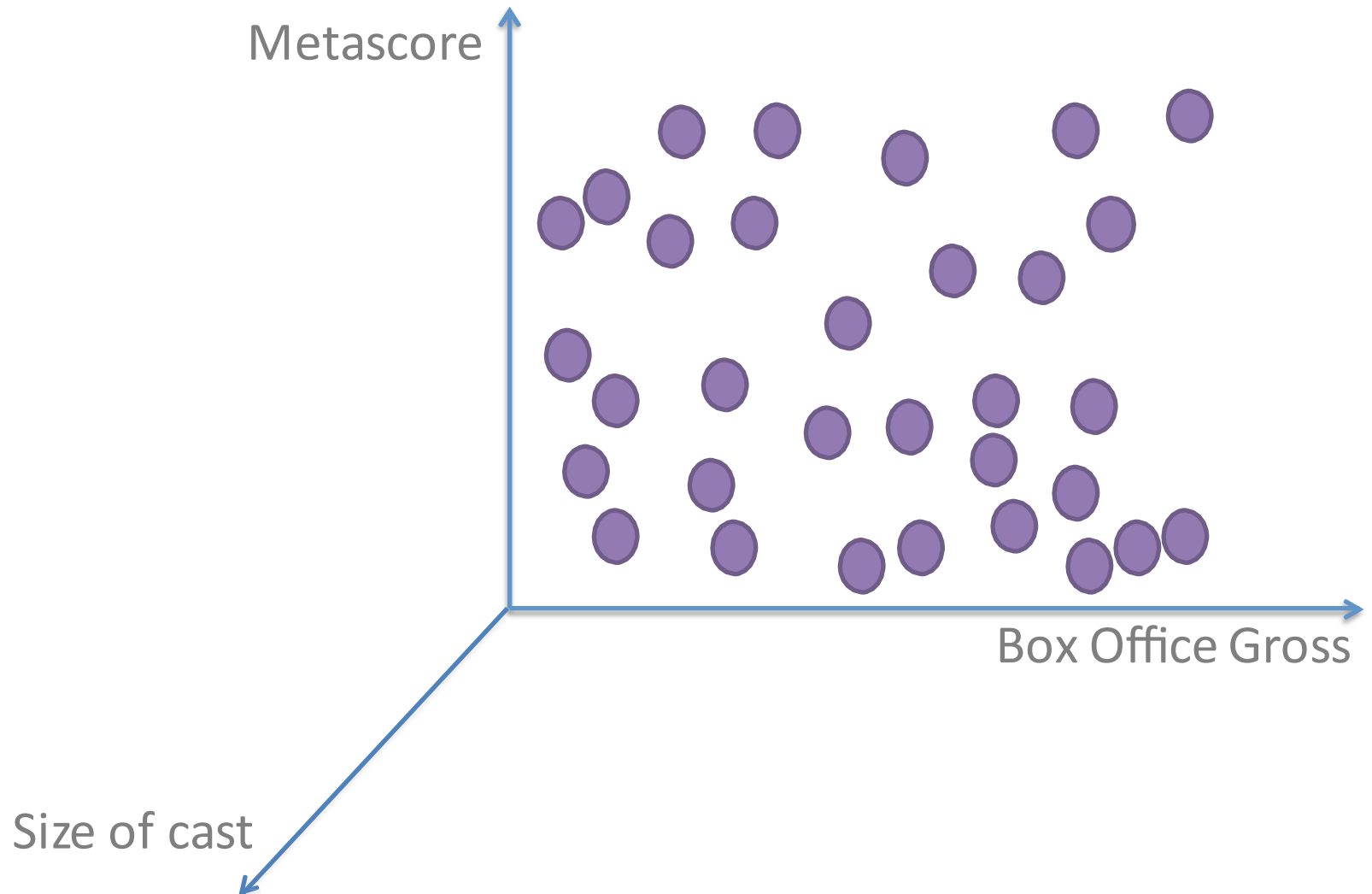
Feature extraction 1D

$$\text{Oscar_or_not} = \text{logit}(\beta_1(0.4 * \text{Metascore} + 0.6 * \text{Awards}))$$

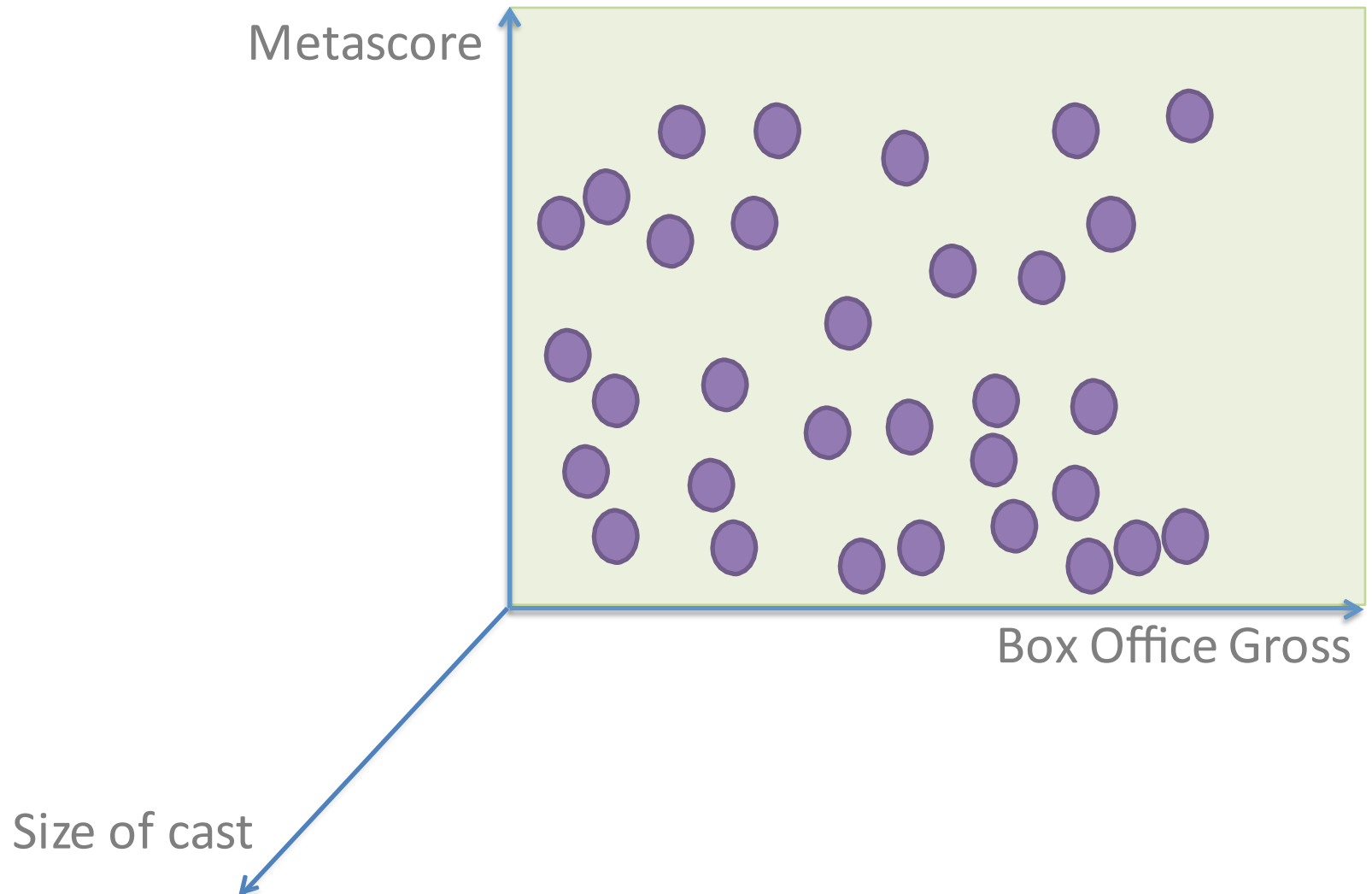
3D → 2D Feature Selection



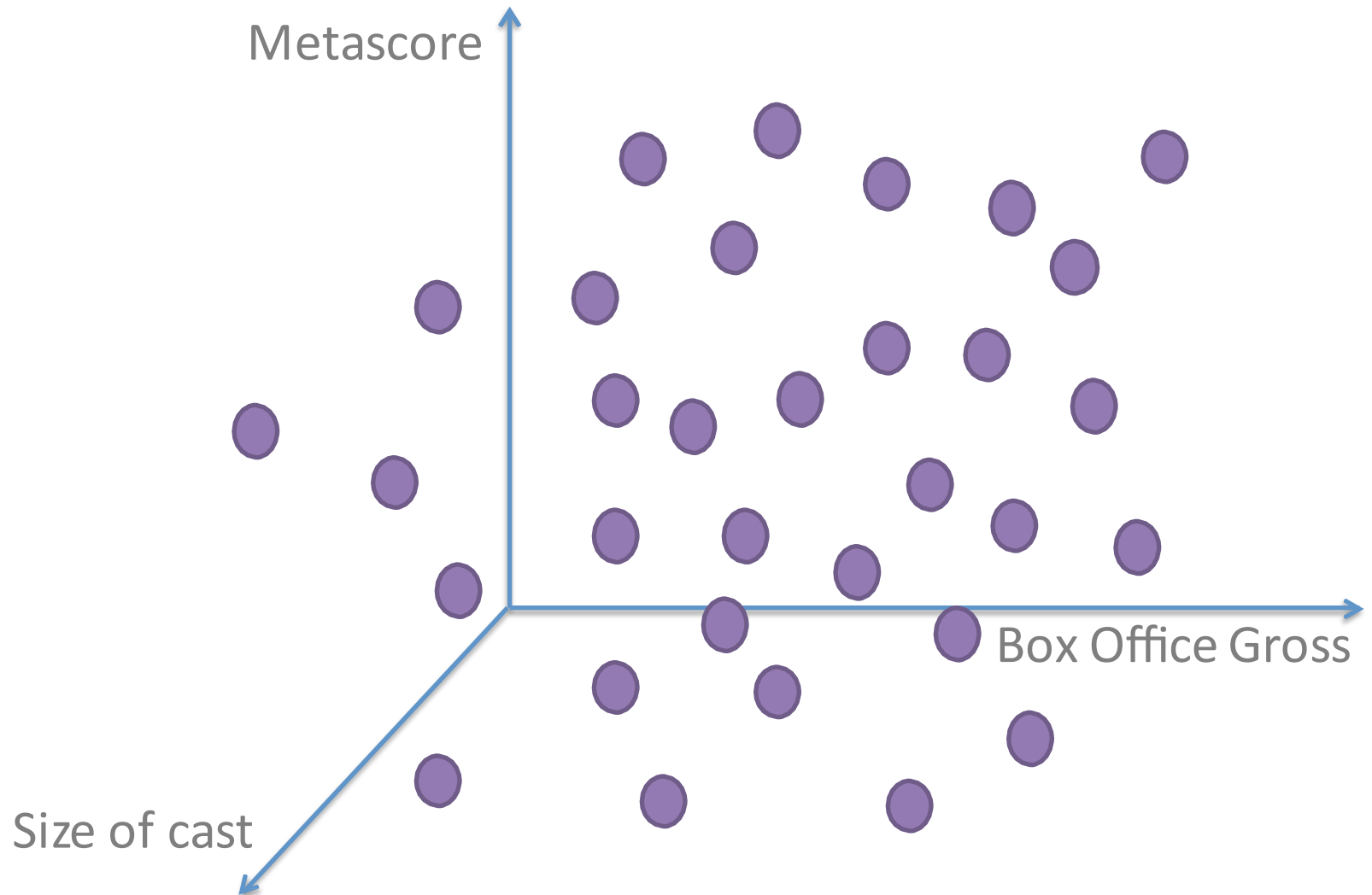
3D → 2D Feature Selection



3D → 2D Feature Selection

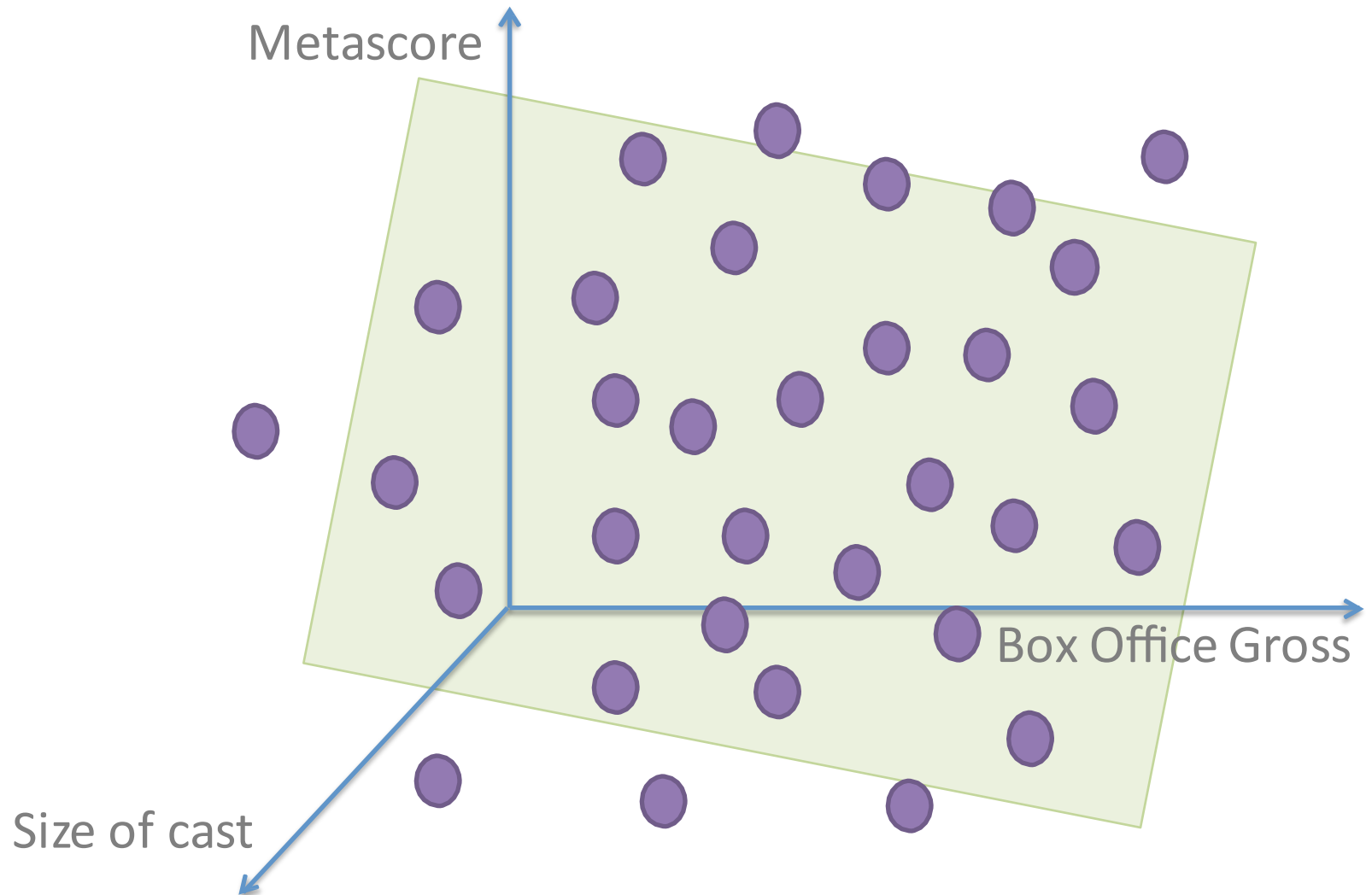


3D → 2D Feature Extraction (PCA)



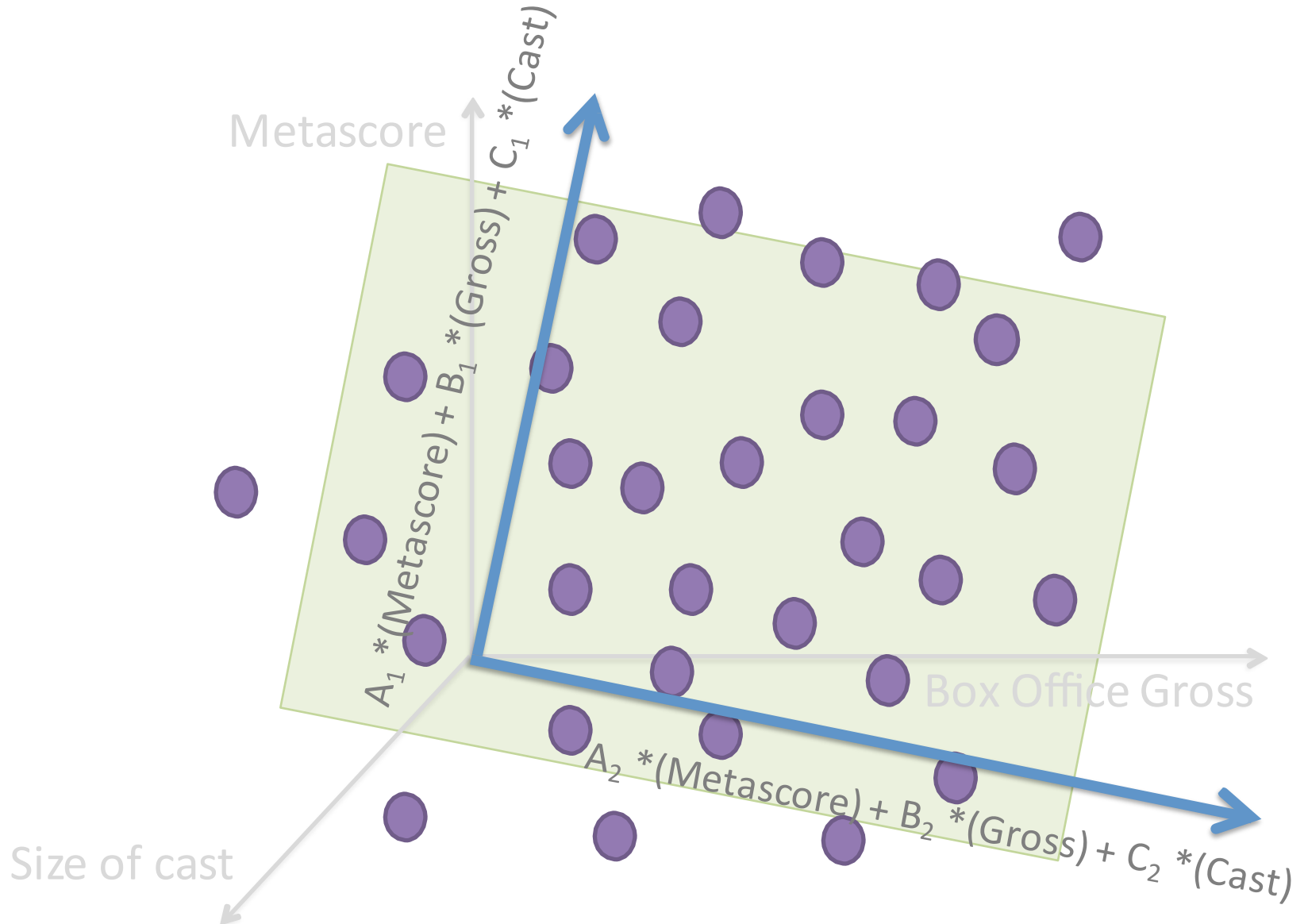
3D → 2D Feature Extraction (PCA)

Optimum plane



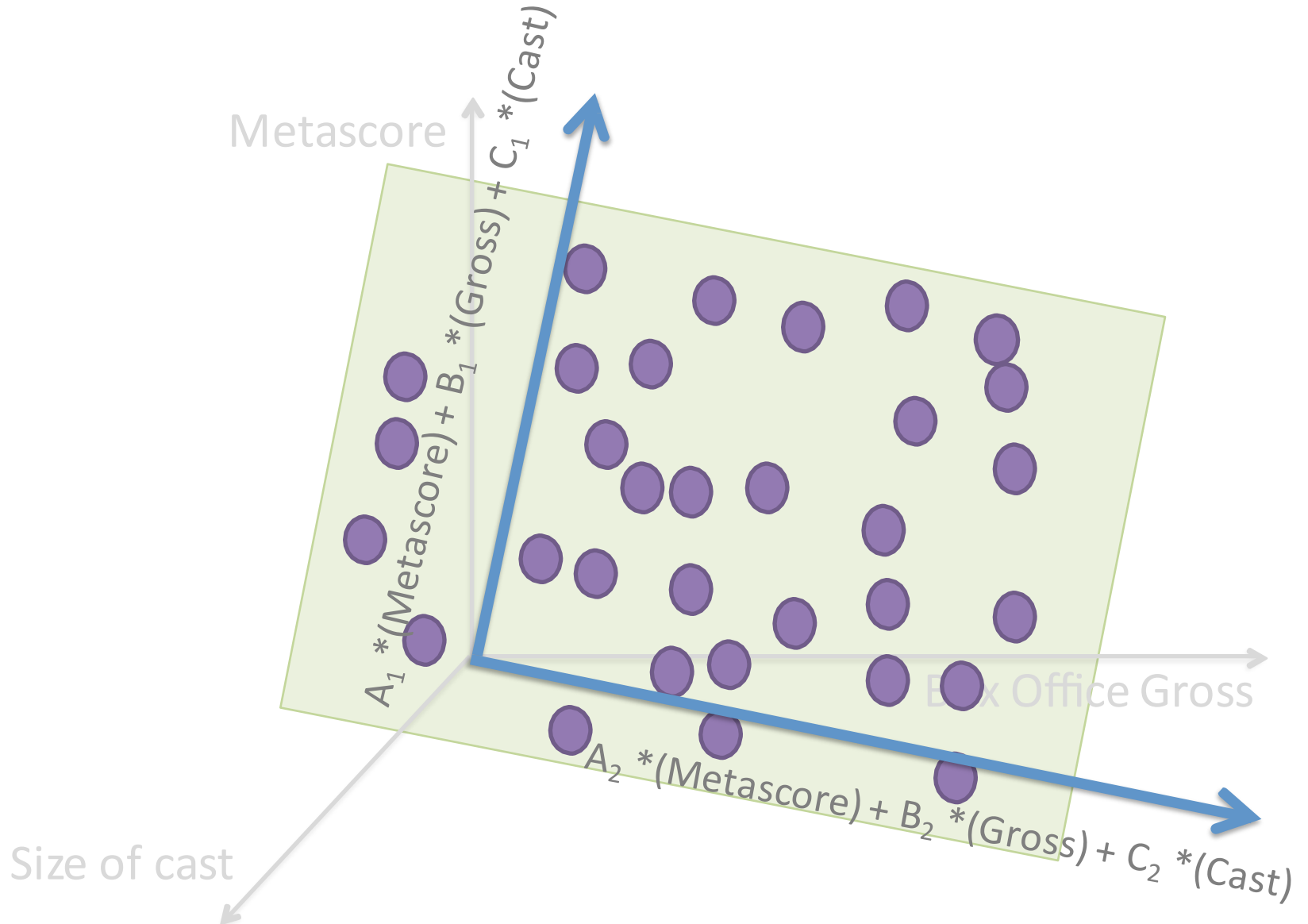
3D → 2D Feature Extraction (PCA)

Optimum plane



3D → 2D Feature Extraction (PCA)

Optimum plane



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

```
reducer = PCA( n_components = 20 )  
reduced_X = reducer.fit_transform(X)
```

How and why to use PCA

Improving your clustering

Improving your classification
(alternative to feature selection)

Visualizing high dimensional data in 2D or 3D

Data compression with little loss

PCA Math

Vectors defining the reduced hyperplane are eigenvectors of the covariance matrix of the features.

Singular Value Decomposition (SVD) is a related decomposition that can be used to solve the PCA problem as well, and with better numeric properties.

(See notebook.)