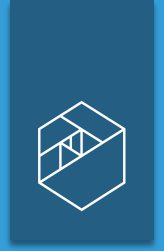
Dimensionality Reduction: SVD, Featuring PCA

Agenda

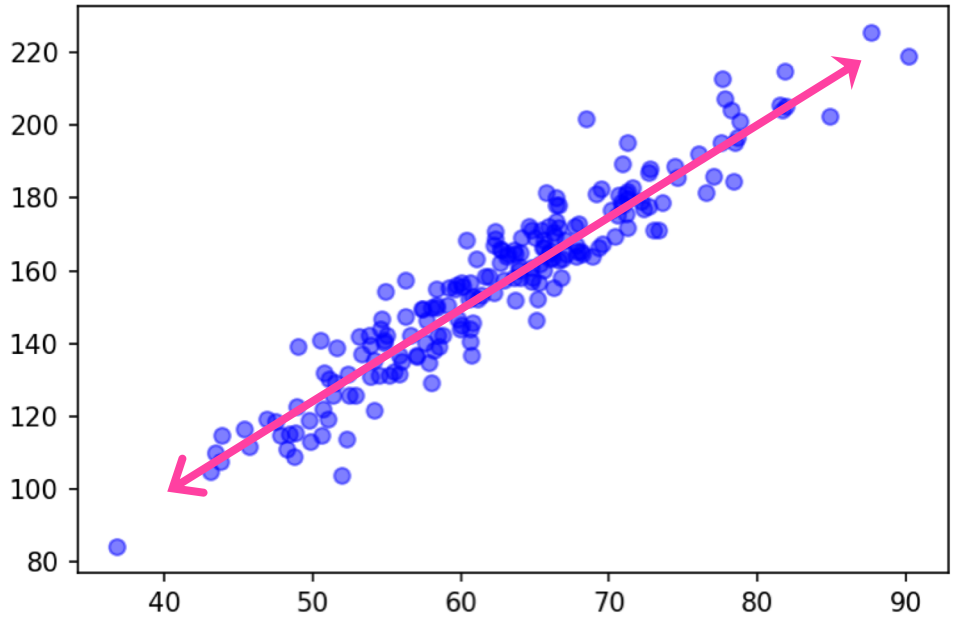
- Dimensionality reduction
  - Assumptions
- Matrix decomposition
  - In linear algebra
  - Singular value decomposition (SVD) in particular
  - Relation to dimensionality reduction
- Principal Component Analysis (PCA)
  - As a technique for dimensionality reduction / feature extraction
- Preview of applications



Dimensionality Reduction

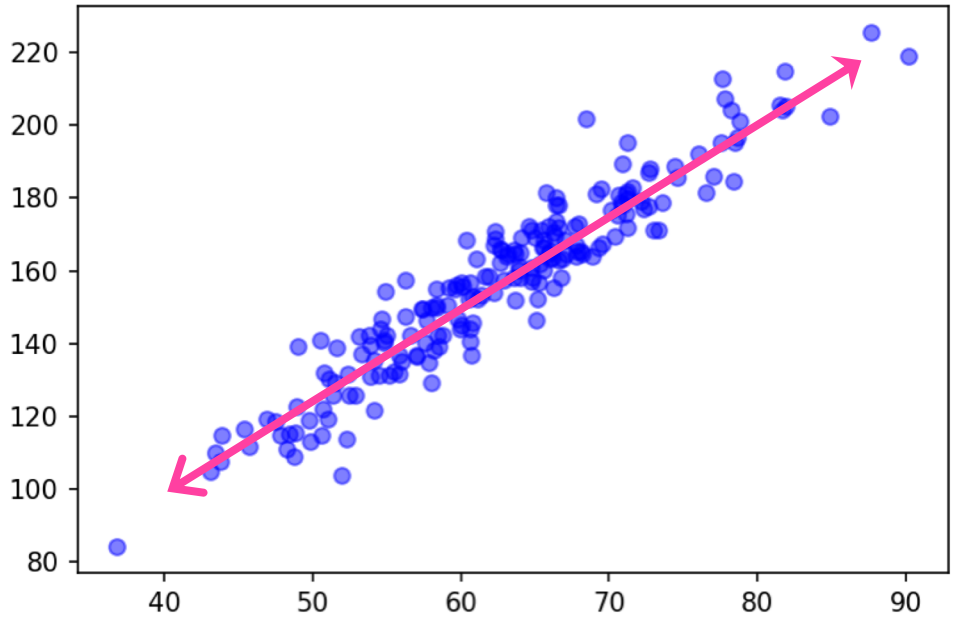
Assumptions for dimensionality reduction

- **For dimensionality reduction to make sense, we assume that our data mostly lie in a lower dimension**



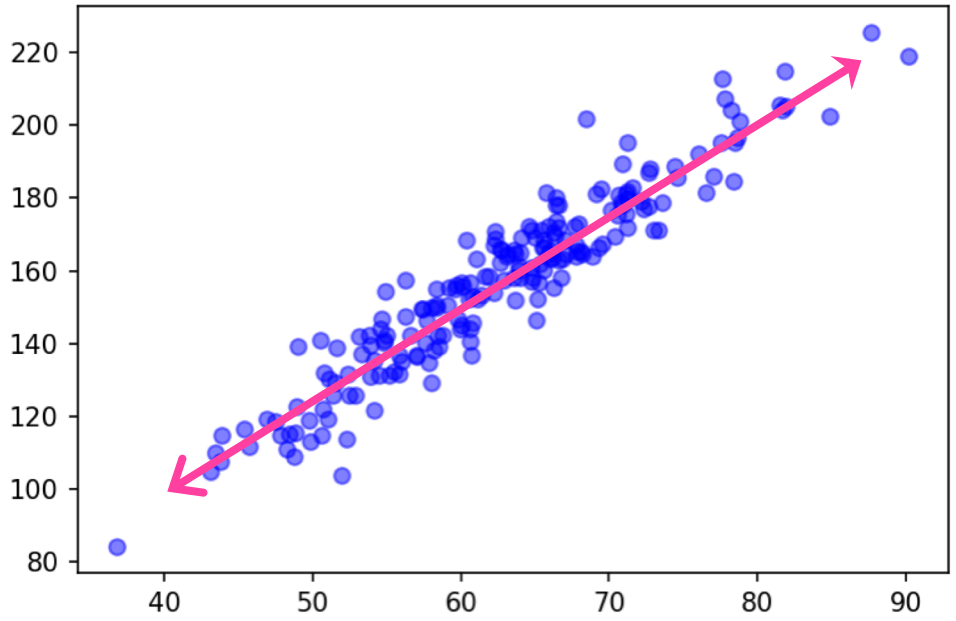
Assumptions for dimensionality reduction

- **For dimensionality reduction to make sense, we assume that our data mostly lie in a lower dimension**
- If our data is scattered randomly across all of its dimensions, we'd lose too much information by reducing dimensions



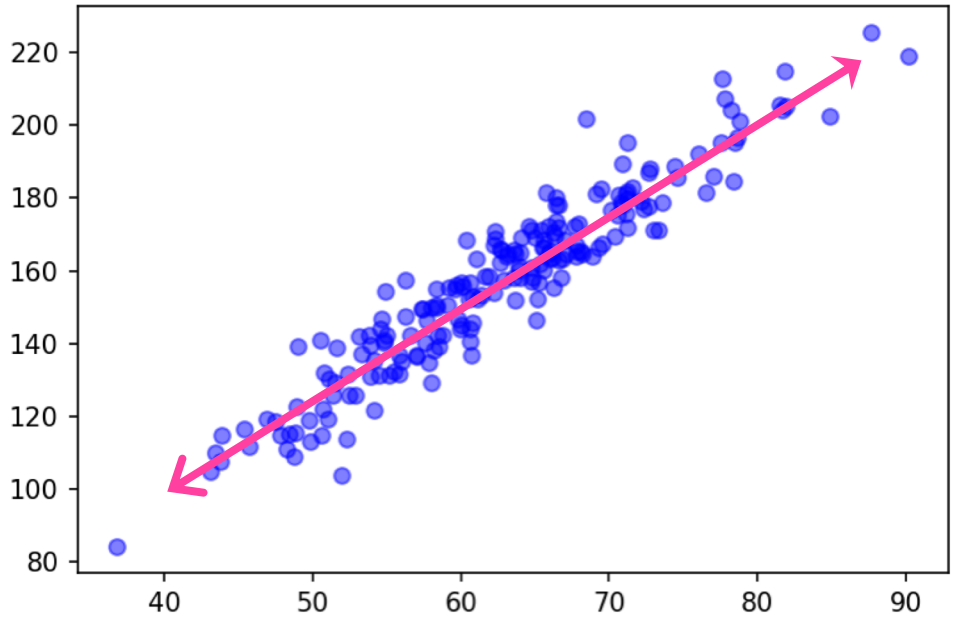
Assumptions for dimensionality reduction

- **For dimensionality reduction to make sense, we assume that our data mostly lie in a lower dimension**
- If our data is scattered randomly across all of its dimensions, we'd lose too much information by reducing dimensions
- But, if we think the data can reasonably be approximated by a lower dimension, then dimensionality reduction gives us a lot of positives



Assumptions for dimensionality reduction

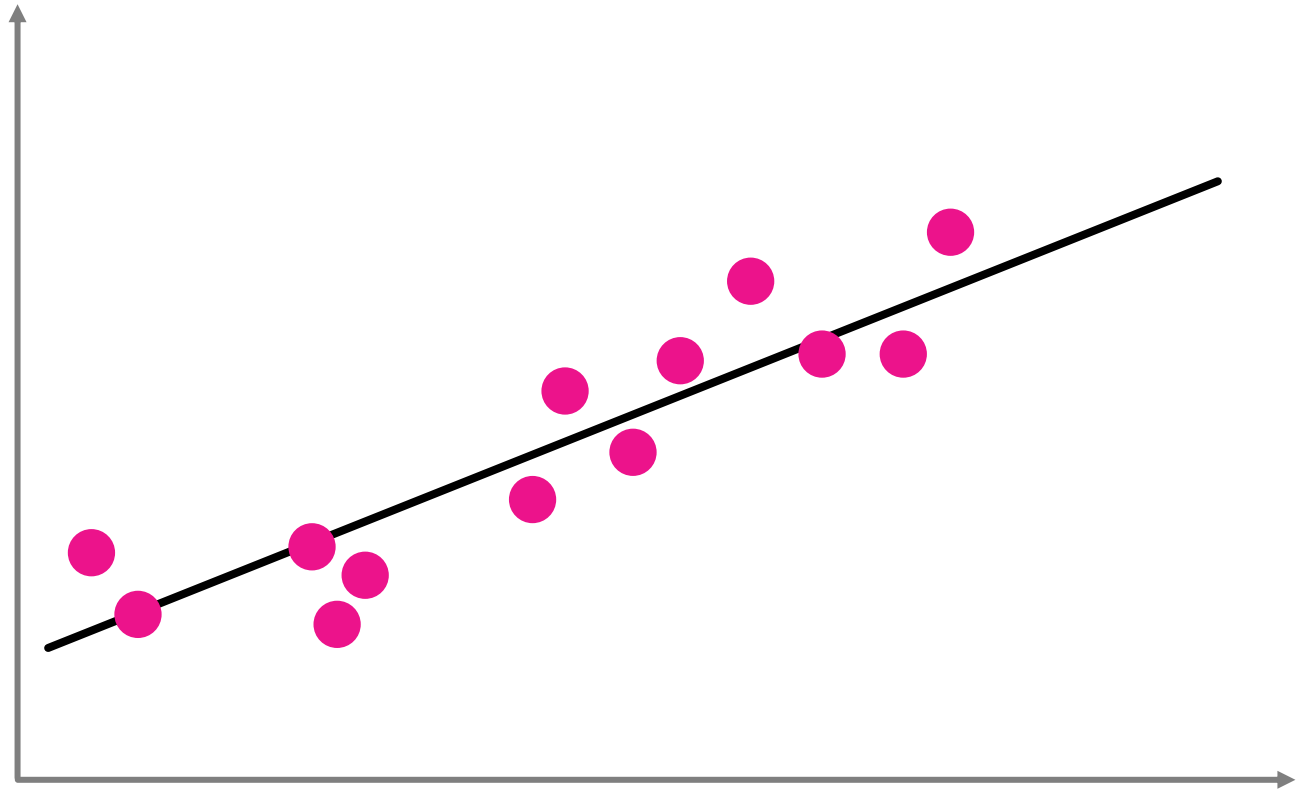
- For dimensionality reduction to make sense, we assume that our data mostly lie in a lower dimension
- If our data is scattered randomly across all of its dimensions, we'd lose too much information by reducing dimensions
- But, if we think the data can reasonably be approximated by a lower dimension, then dimensionality reduction gives us a lot of positives



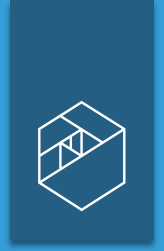
Check for understanding:

Given what we learned in the curse of dimensionality lecture, what do we think some positives of dimensionality reduction might be?

Dimensionality reduction is finding new axis



- The new, reduced axis could be encoding important information!
- We can learn about our data by getting rid of extraneous dimensions that are just noise
  - Some dimensions with little variance might be, e.g., measurement error (imprecise instruments)
  - If a dimension is low variance, not super helpful for modeling



Singular Value Decomposition

Matrix decomposition is just multiplication

- **Matrix decomposition** or **matrix factorization** is a broad term that encompasses lots of techniques

Matrix decomposition is just multiplication

- **Matrix decomposition** or **matrix factorization** is a broad term that encompasses lots of techniques
  - It just means turning one matrix into the product of several matrices in canonical form

Matrix decomposition is just multiplication

- **Matrix decomposition** or **matrix factorization** is a broad term that encompasses lots of techniques
  - It just means turning one matrix into the product of several matrices in canonical form
    - Canonical form means a unique, standard way of representing something in math
    - Putting things in canonical/standard forms helps us reason more easily because different subparts of the standard form have different meanings
    - We can more easily compare artifacts if they're all in the same canonical form

Matrix decomposition is just multiplication

- **Matrix decomposition** or **matrix factorization** is a broad term that encompasses lots of techniques
  - It just means turning one matrix into the product of several matrices in canonical form
    - Canonical form means a unique, standard way of representing something in math
    - Putting things in canonical/standard forms helps us reason more easily because different subparts of the standard form have different meanings
    - We can more easily compare artifacts if they're all in the same canonical form
  - The type of decomposition depends on the shape and type of our matrix, and our use case

Matrix decomposition is just multiplication

- **Matrix decomposition** or **matrix factorization** is a broad term that encompasses lots of techniques
  - It just means turning one matrix into the product of several matrices in canonical form
    - Canonical form means a unique, standard way of representing something in math
    - Putting things in canonical/standard forms helps us reason more easily because different subparts of the standard form have different meanings
    - We can more easily compare artifacts if they're all in the same canonical form
  - The type of decomposition depends on the shape and type of our matrix, and our use case
    - E.g. eigendecomposition requires a square matrix, but singular value decomposition works on rectangular matrices

Matrix decomposition is just multiplication

- **Matrix decomposition** or **matrix factorization** is a broad term that encompasses lots of techniques
  - It just means turning one matrix into the product of several matrices in canonical form
    - Canonical form means a unique, standard way of representing something in math
    - Putting things in canonical/standard forms helps us reason more easily because different subparts of the standard form have different meanings
    - We can more easily compare artifacts if they're all in the same canonical form
  - The type of decomposition depends on the shape and type of our matrix, and our use case
    - E.g. eigendecomposition requires a square matrix, but singular value decomposition works on rectangular matrices

If you remember nothing else, take away these points:

1) Decomposition is multiplying matrices in a standard form
2) Different subparts of the standard form have special meanings

Singular Value Decomposition (SVD)

- SVD is useful because it’s easy to compute and doesn’t require a square matrix

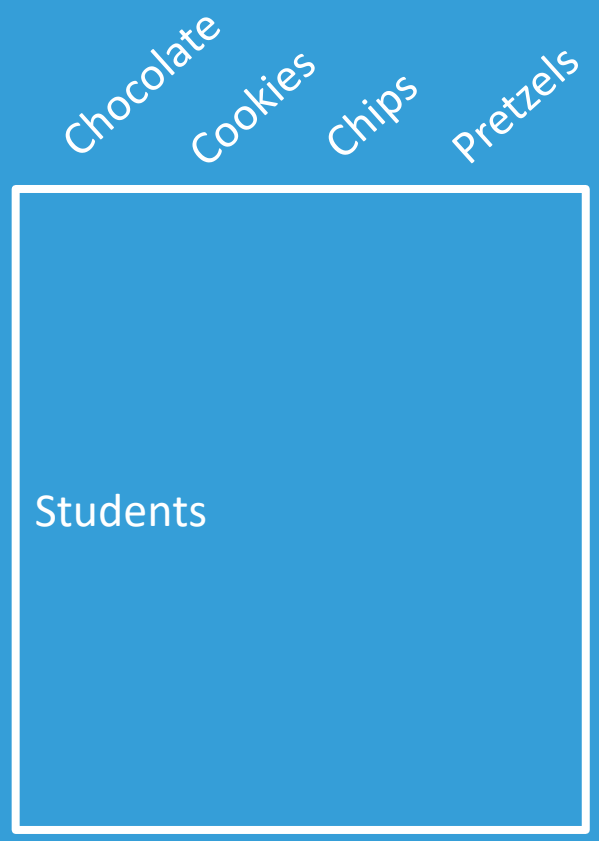
Singular Value Decomposition (SVD)

- SVD is useful because it's easy to compute and doesn't require a square matrix
  - The matrices we're working with are our data
  - Rows are observations
  - Columns are covariates or features (e.g. dollars spent on coffee per month, rating of dark chocolate)

Singular Value Decomposition (SVD)

- SVD is useful because it's easy to compute and doesn't require a square matrix
  - The matrices we're working with are our data
  - Rows are observations
  - Columns are covariates or features (e.g. dollars spent on coffee per month, rating of dark chocolate)
- SVD gives us categories in our data that we might not be able to see by looking at all of the data together! This will become more clear with an example

Singular Value Decomposition (SVD)



Singular Value Decomposition (SVD)

Data matrix:
Ratings of snack foods

Chocolate	Cookies	Chips	Pretzels
10	7	0	3
8	6	2	4
3	1	8	5

Students

Singular Value Decomposition (SVD)

Data matrix:
Ratings of snack foods

Chocolate	Cookies	Chips	Pretzels
10	7	0	3
8	6	2	4
3	1	8	5

Students

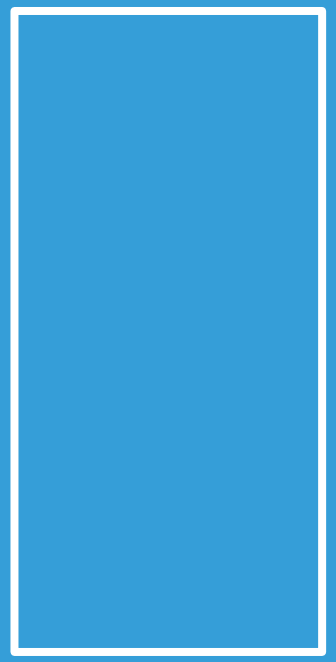
=

Singular Value Decomposition (SVD)

Data matrix:
Ratings of snack foods

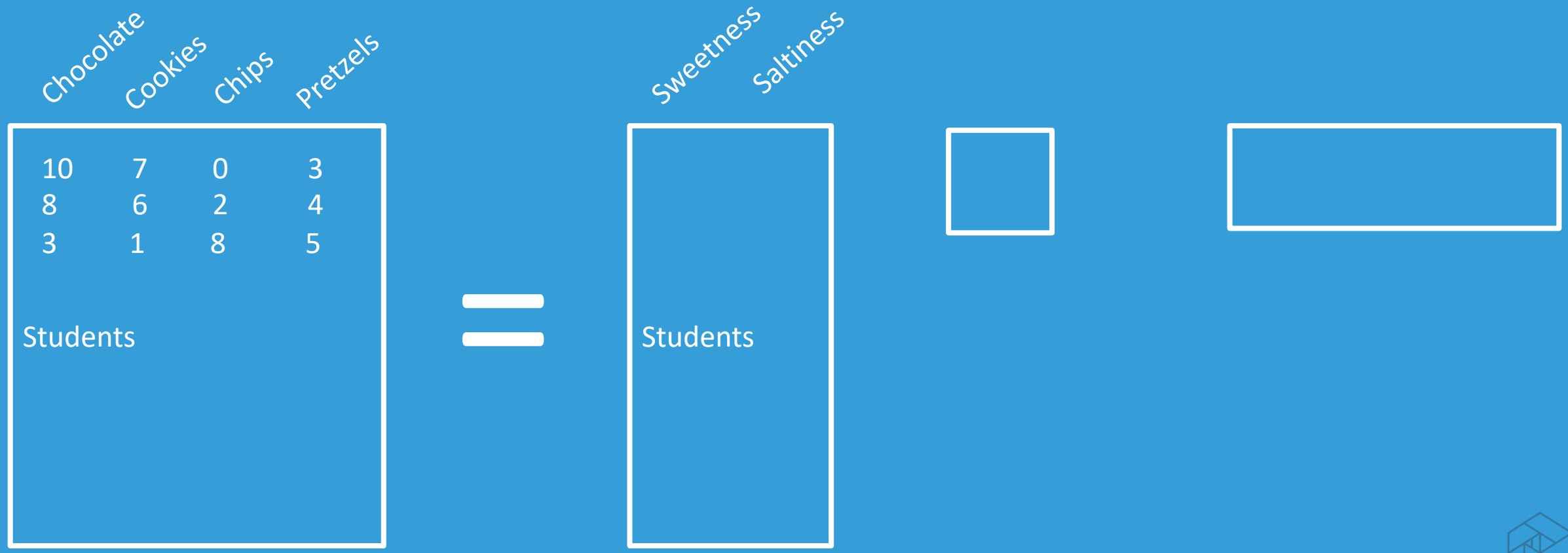
Chocolate	Cookies	Chips	Pretzels
10	7	0	3
8	6	2	4
3	1	8	5

Students



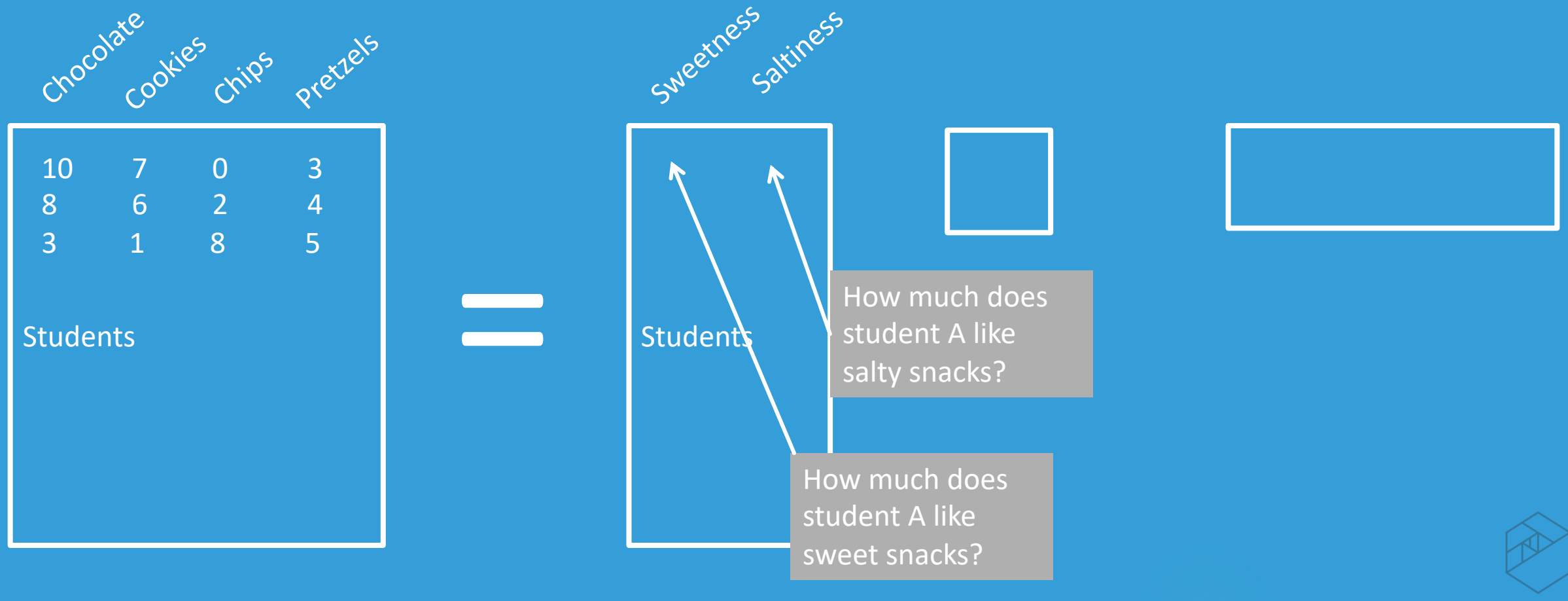
Singular Value Decomposition (SVD)

Data matrix:
Ratings of snack foods



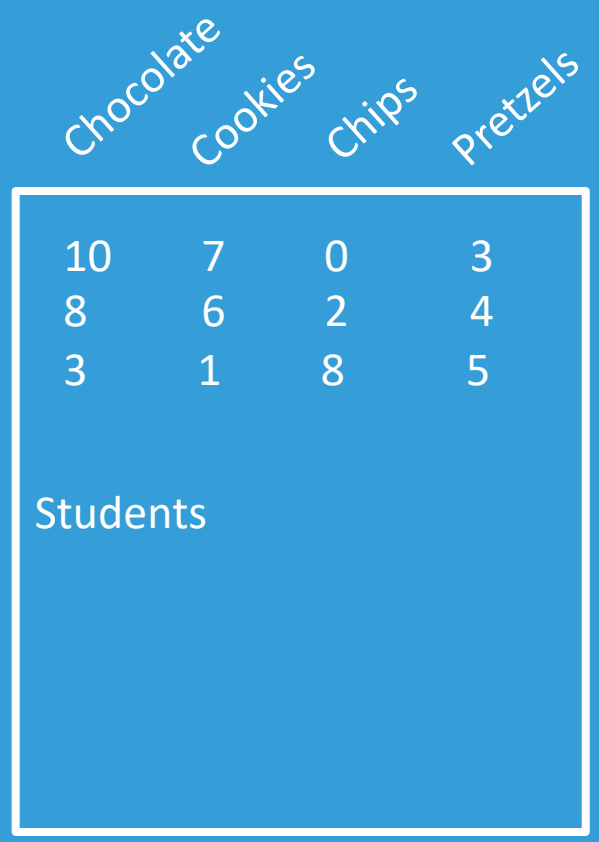
Singular Value Decomposition (SVD)

Data matrix:
Ratings of snack foods

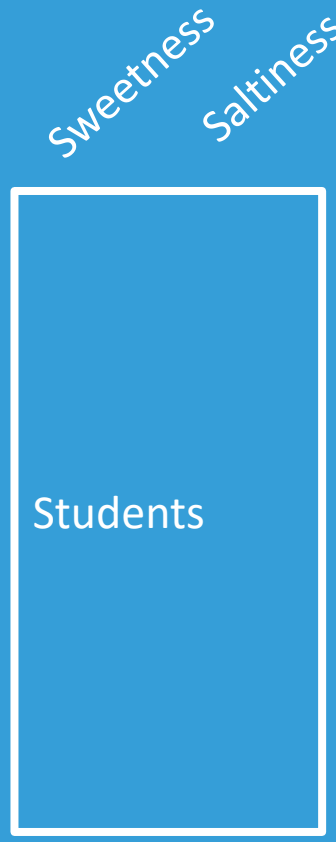


Singular Value Decomposition (SVD)

Data matrix:
Ratings of snack foods



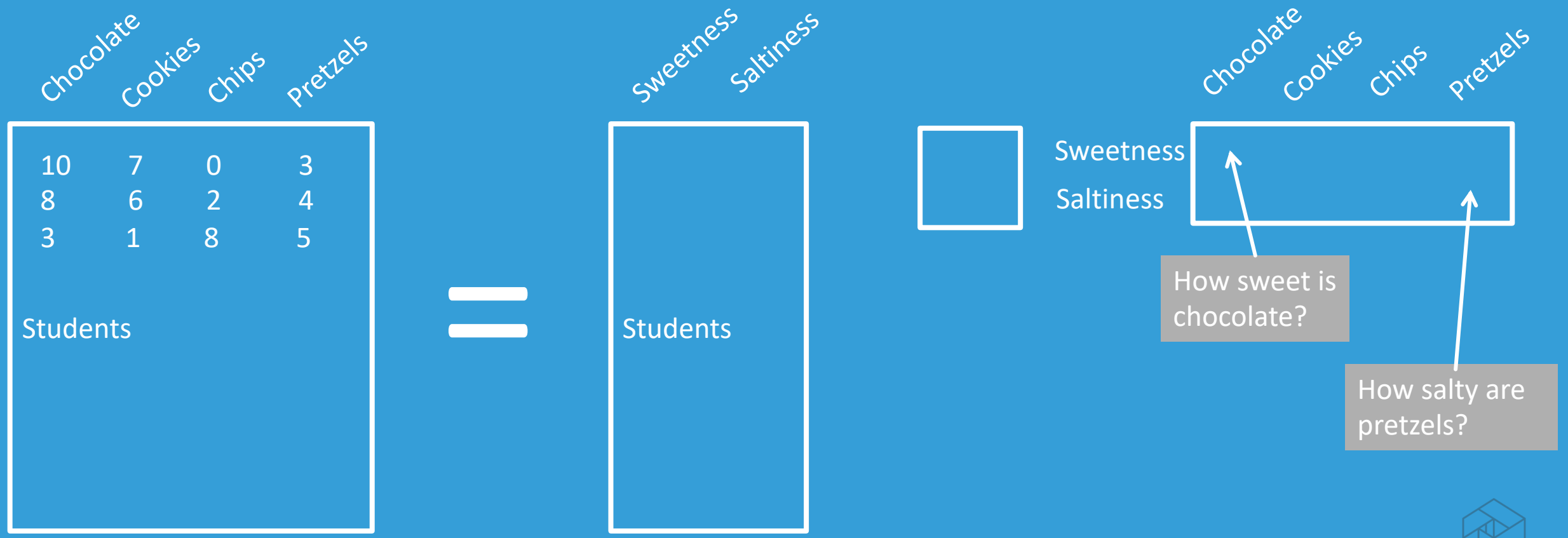
=



On the diagonal, the strength of sweetness and saltiness (variance)

Singular Value Decomposition (SVD)

Data matrix:
Ratings of snack foods

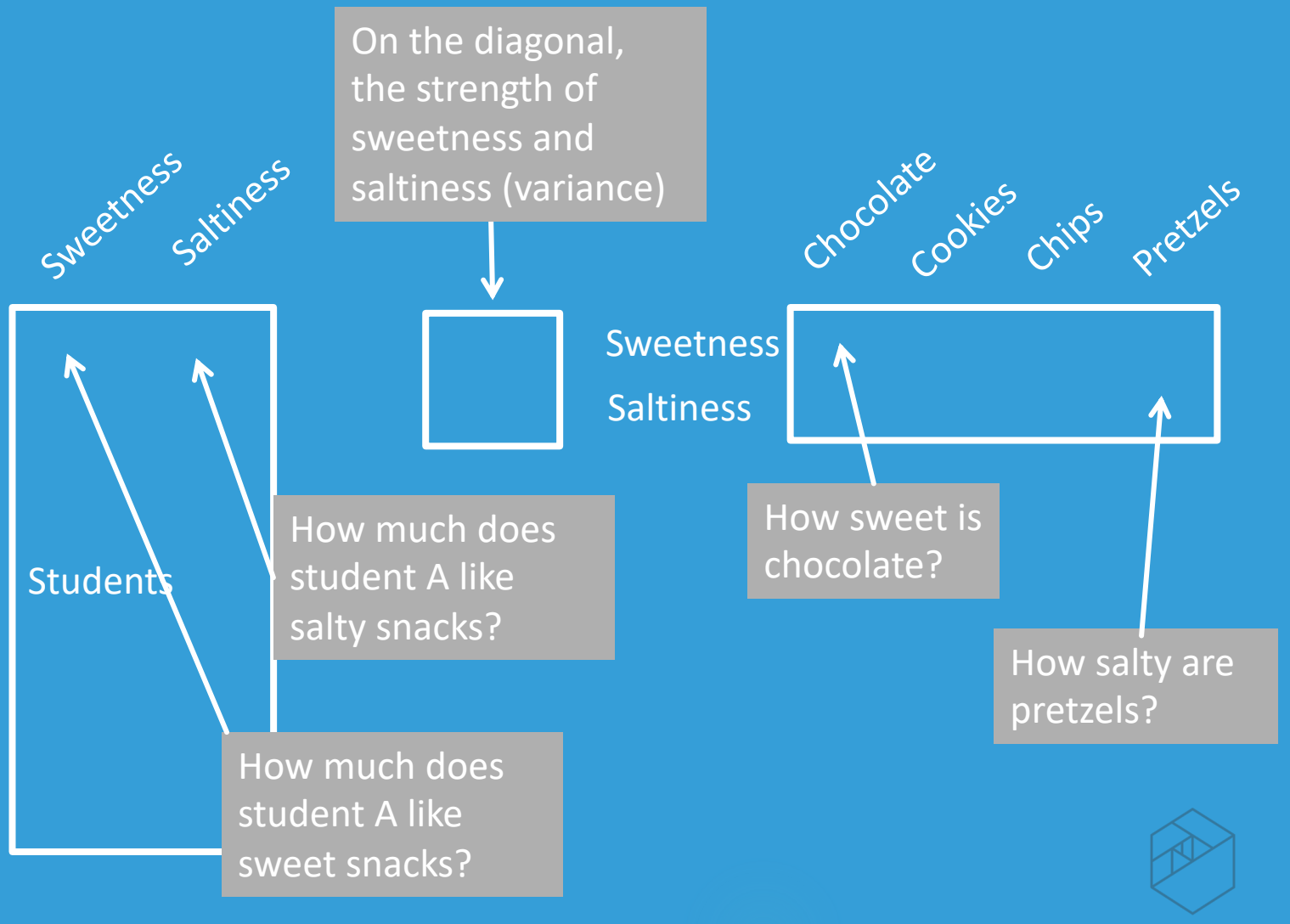


Singular Value Decomposition (SVD)

Data matrix:
Ratings of snack foods

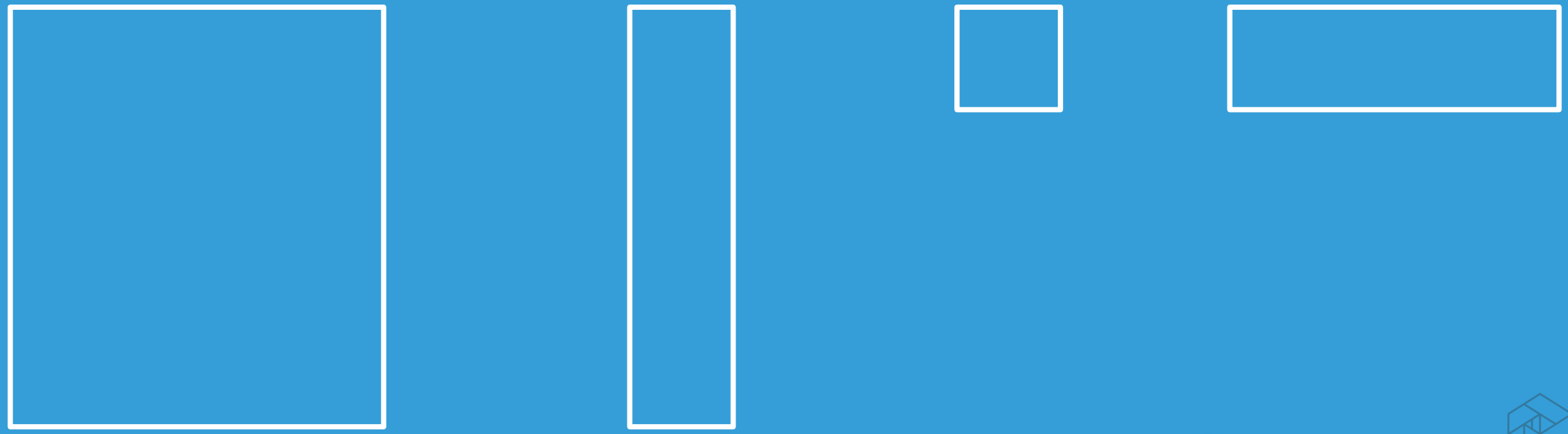
	Chocolate	Cookies	Chips	Pretzels
Students	10	7	0	3
	8	6	2	4
	3	1	8	5

=



Singular Value Decomposition (SVD)

$$A_{m \times n} = U_{m \times r} \Sigma_{r \times r} V'_{n \times r}$$



Singular Value Decomposition (SVD)

$$A_{m \times n} = U_{m \times r} \Sigma_{r \times r} V'_{n \times r}$$

- **A:** Our data matrix, where rows are observations and columns are covariates/features
- **U:** Left singular vectors, or decomposing each of our observations into how well they're represented by a category
- **Σ:** Diagonal matrix (all off-diagonal entries are 0) of singular values, or the strengths of our categories
- **V:** Right singular vectors, or decomposing our covariates into categories

- It's always possible to do this decomposition, and **the decomposition is unique for each data matrix A**

Using Σ for dimensionality reduction

- Turns out, ratings of chocolate and cookies are very highly correlated for most students

Using Σ for dimensionality reduction

- Turns out, ratings of chocolate and cookies are very highly correlated for most students
- Similarly, students who like/dislike chips tend to have the same feeling about pretzels

Using Σ for dimensionality reduction

- Turns out, ratings of chocolate and cookies are very highly correlated for most students
- Similarly, students who like/dislike chips tend to have the same feeling about pretzels
  - We don’t need all four categories of snacks, we can just use two: (1) sweet, (2) salty
  - SVD gave us these categories that we couldn’t even see in our raw data!

Using Σ for dimensionality reduction

- Turns out, ratings of chocolate and cookies are very highly correlated for most students
- Similarly, students who like/dislike chips tend to have the same feeling about pretzels
  - We don't need all four categories of snacks, we can just use two: (1) sweet, (2) salty
  - SVD gave us these categories that we couldn't even see in our raw data!
- **Σ encodes the strength/weight of each category**

Using Σ for dimensionality reduction

- Turns out, ratings of chocolate and cookies are very highly correlated for most students
- Similarly, students who like/dislike chips tend to have the same feeling about pretzels
  - We don't need all four categories of snacks, we can just use two: (1) sweet, (2) salty
  - SVD gave us these categories that we couldn't even see in our raw data!
- **Σ encodes the strength/weight of each category**
  - Categories are in order from highest to lowest across the diagonal from top left to bottom right

Using Σ for dimensionality reduction

- Turns out, ratings of chocolate and cookies are very highly correlated for most students
- Similarly, students who like/dislike chips tend to have the same feeling about pretzels
  - We don't need all four categories of snacks, we can just use two: (1) sweet, (2) salty
  - SVD gave us these categories that we couldn't even see in our raw data!
- **Σ encodes the strength/weight of each category**
  - Categories are in order from highest to lowest across the diagonal from top left to bottom right

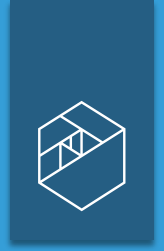
- We can reduce dimensionality by dropping categories with very low strengths (set them to zero)
  - This will drop the rightmost columns of U and the last rows of V'
  - Multiply what's left to get the new data matrix A

Using Σ for dimensionality reduction

- **So, how many categories do we keep?**

Using Σ for dimensionality reduction

- **So, how many categories do we keep?**
- Depends, but good rule of thumb is ~70-80% of variance explained
- Note that, in practice, we can't usually interpret the singular values like we did here
  - That is, we can't really see that the first component is sweetness and the second is saltiness
  - Similar to how in clustering we usually can't put a category on each cluster (e.g. high spenders)

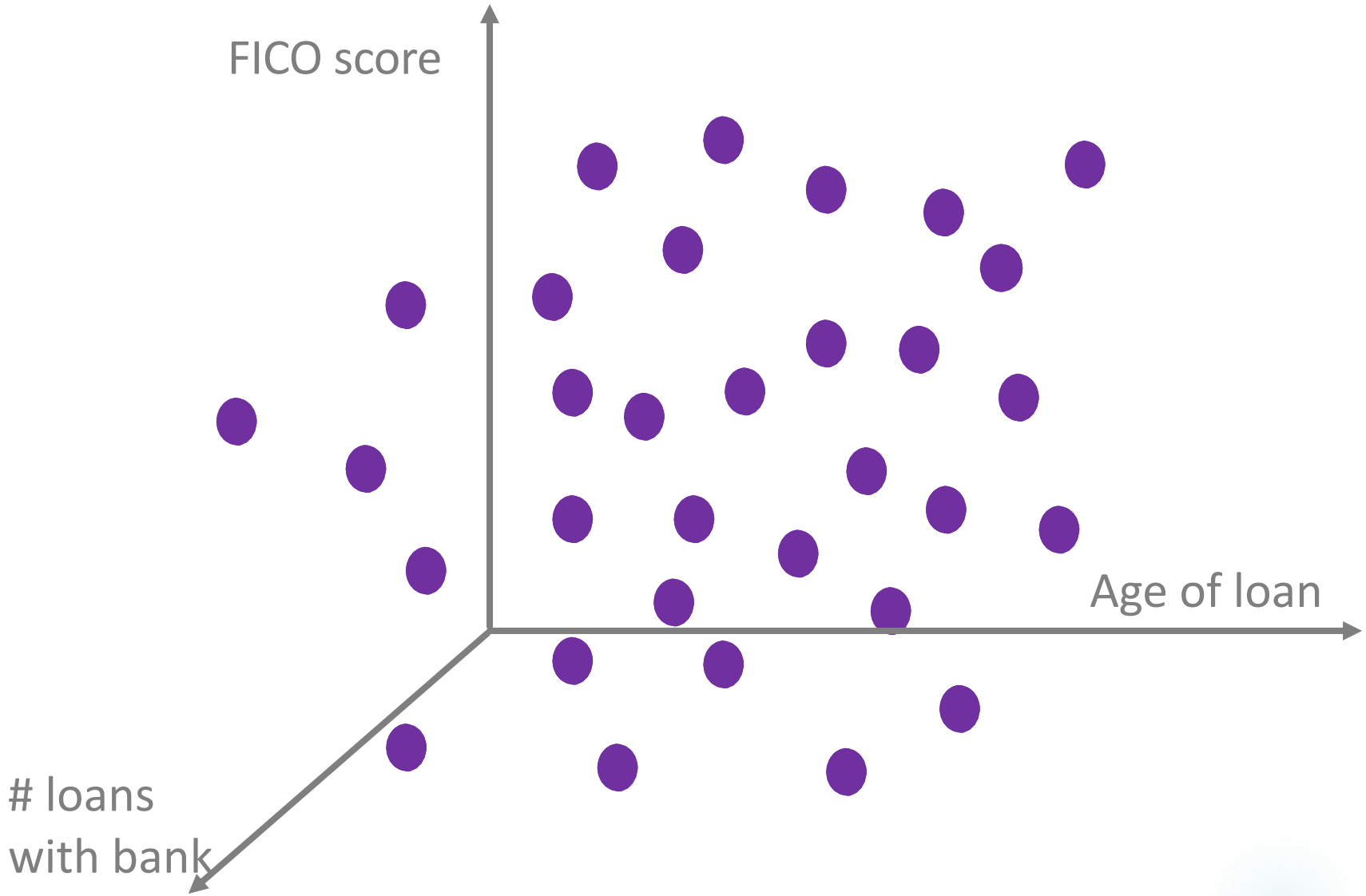


Feature Selection vs. Feature Extraction

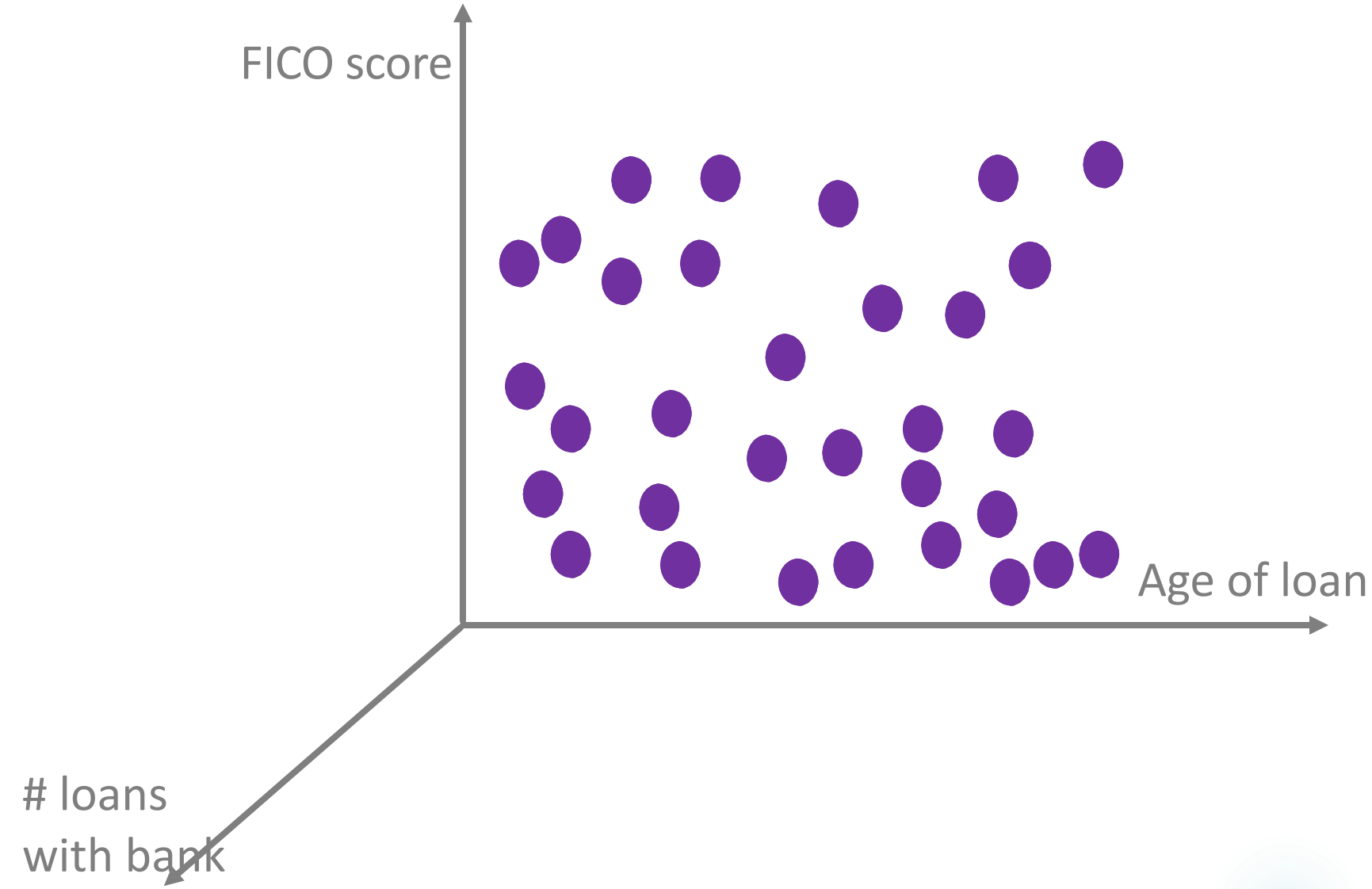
Feature **selection** vs feature **extraction**

- Feature selection involves removing features that aren't helpful
  - They may not be predictive of our dependent variable
  - They may not have a lot of variation
- Feature extraction uses information from all features, but creates artificial new features that are composites
  - Uses information from all features
  - May put more weight on certain features

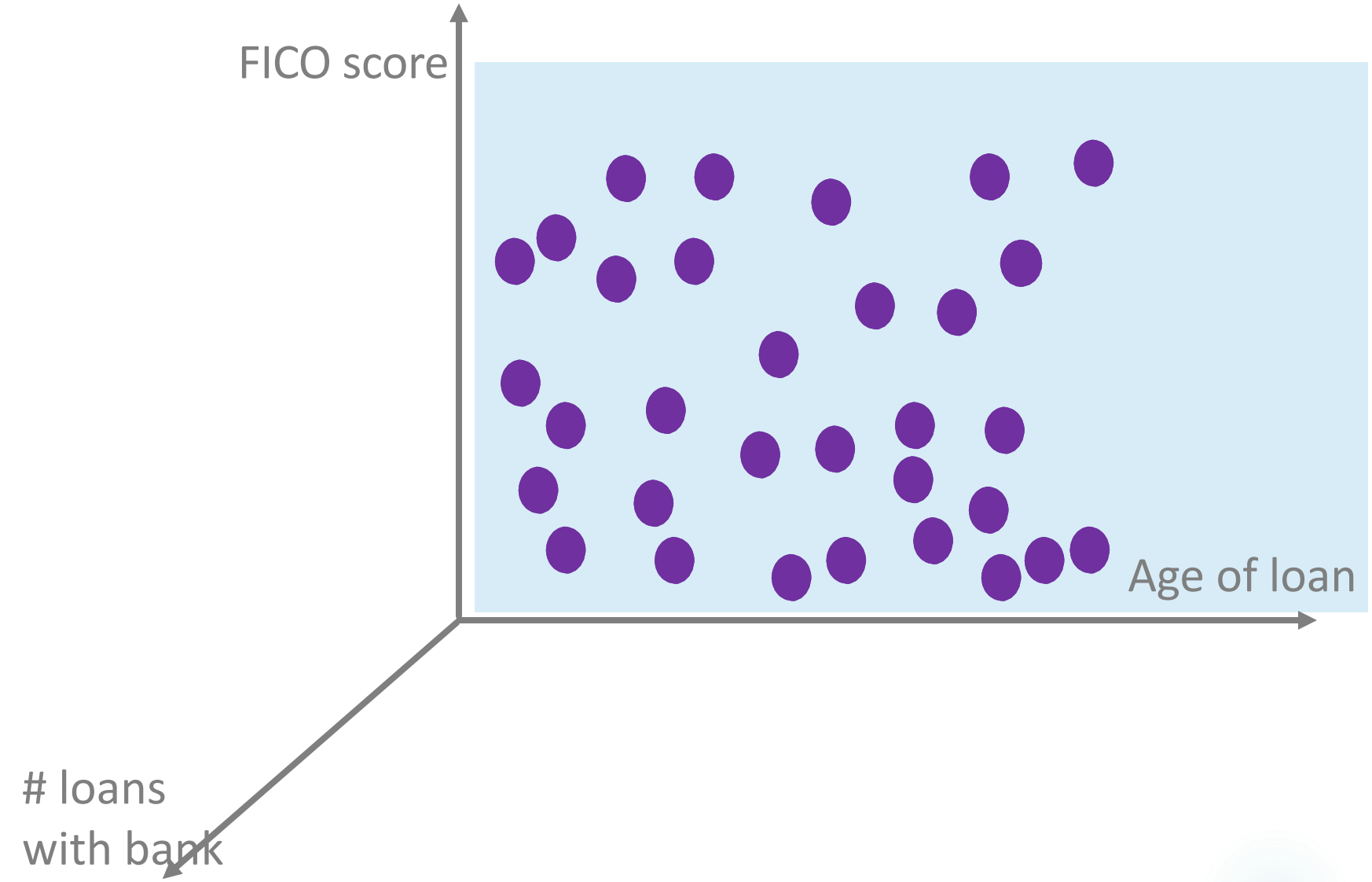
3D → 2D Feature **Selection**



3D → 2D Feature **Selection**



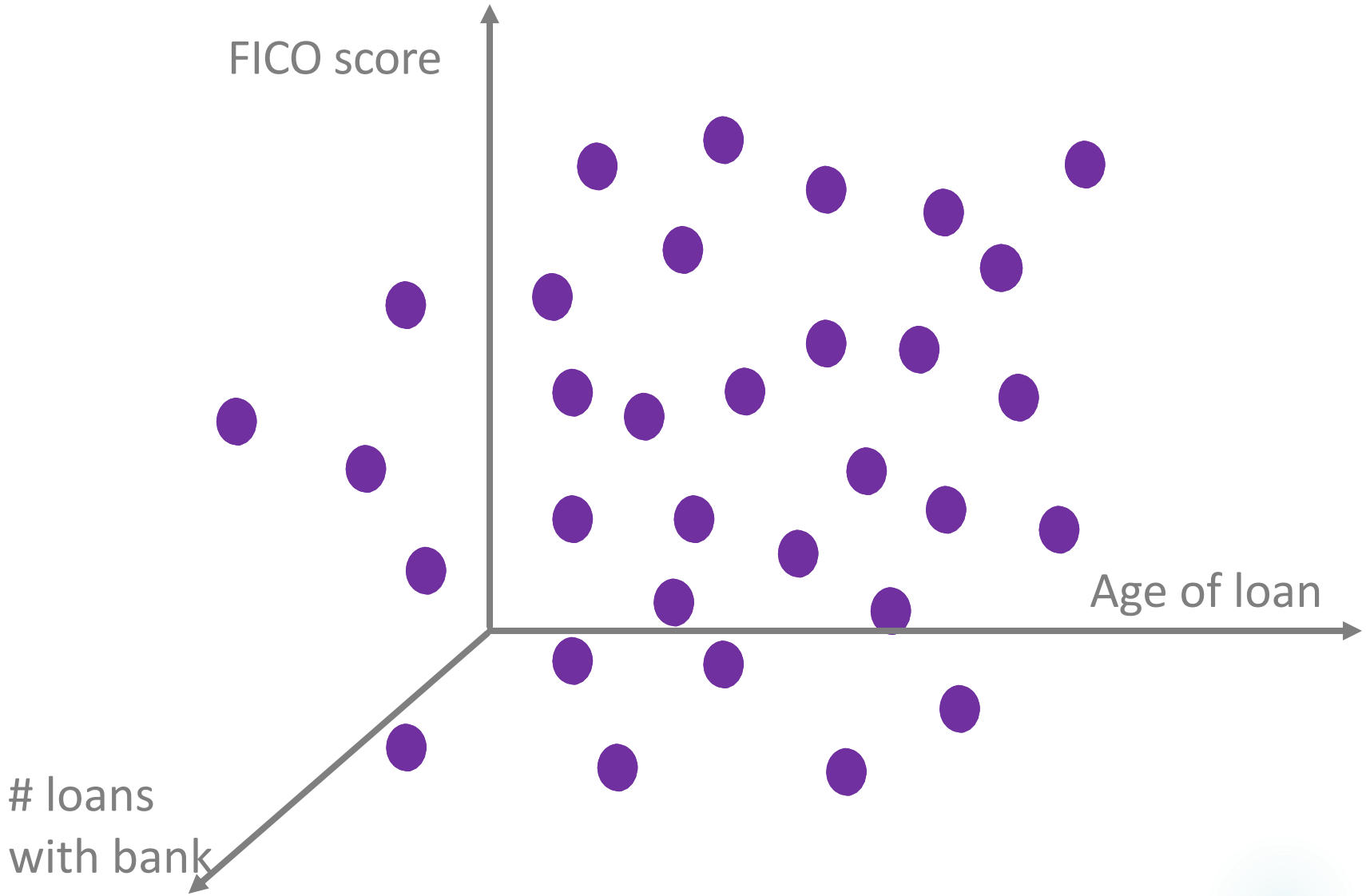
3D → 2D Feature **Selection**



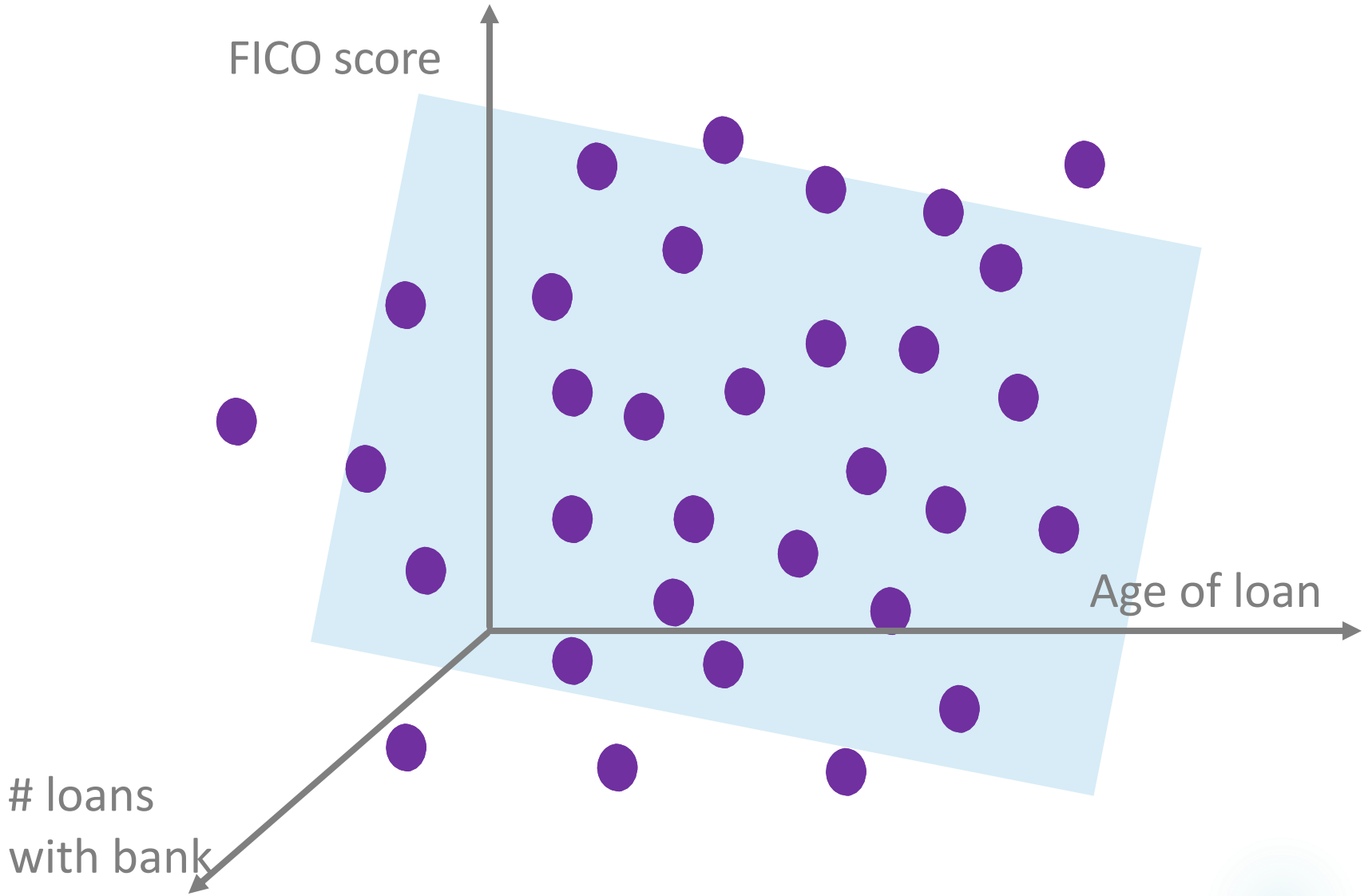
Feature selection

- Feature selection is an art
  - Try fitting with some features, remove some features, re-fit and compare
  - Regularization
  - Feature importance scores (only for some types of models)

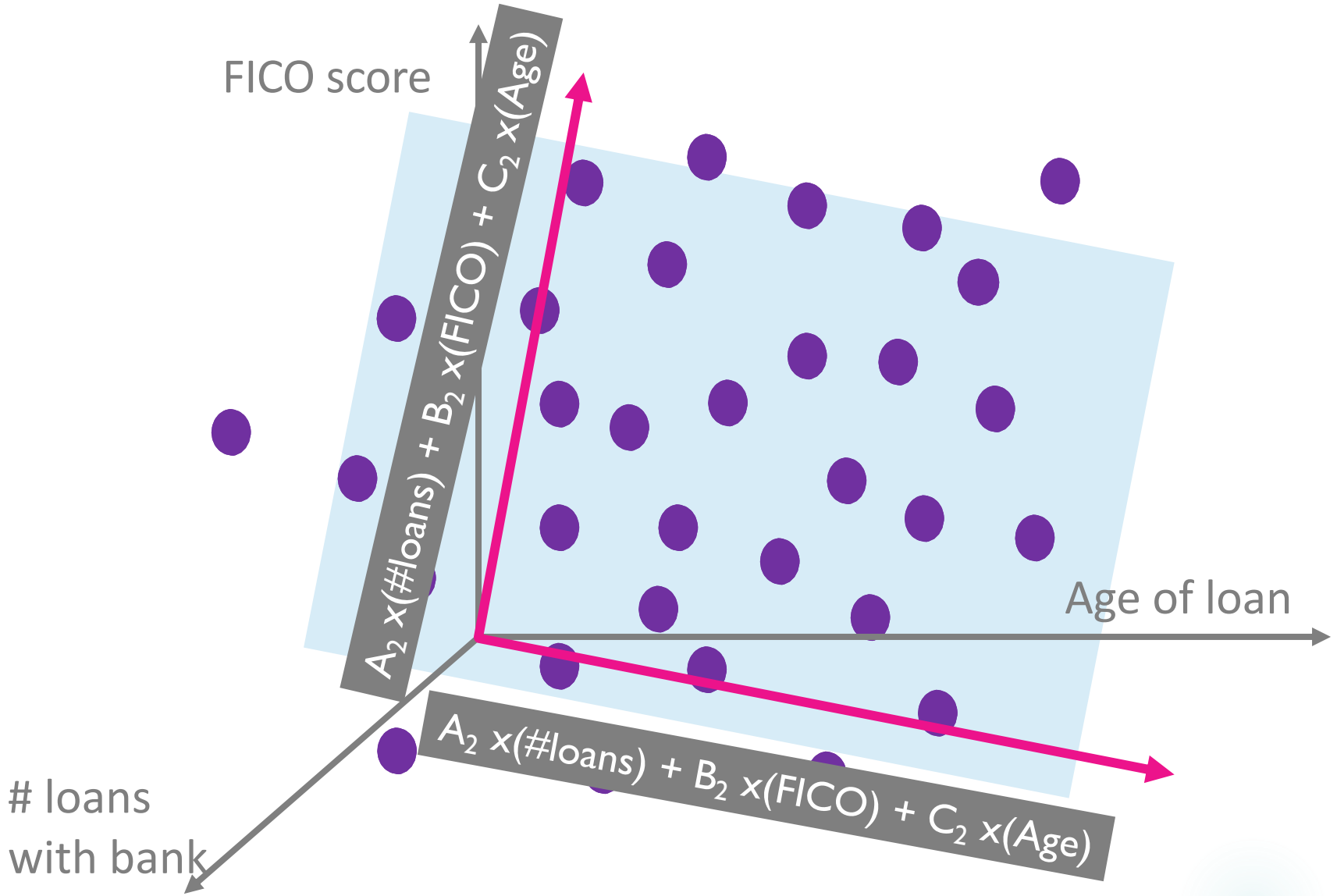
3D → 2D Feature **Extraction** (e.g. PCA)



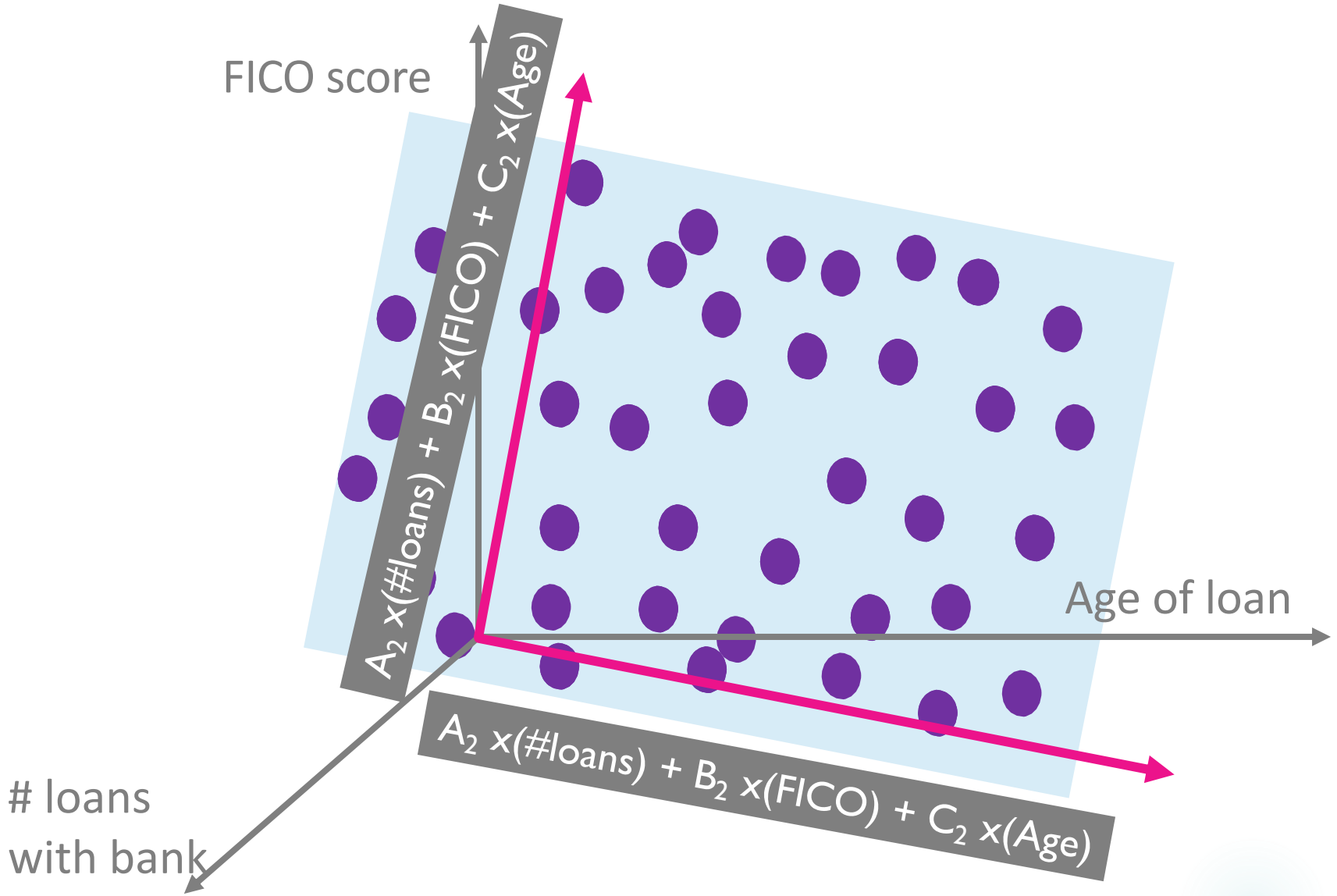
3D → 2D Feature **Extraction** (e.g. PCA)

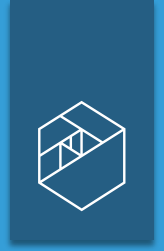


3D → 2D Feature **Extraction** (e.g. PCA)



3D → 2D Feature **Extraction** (e.g. PCA)



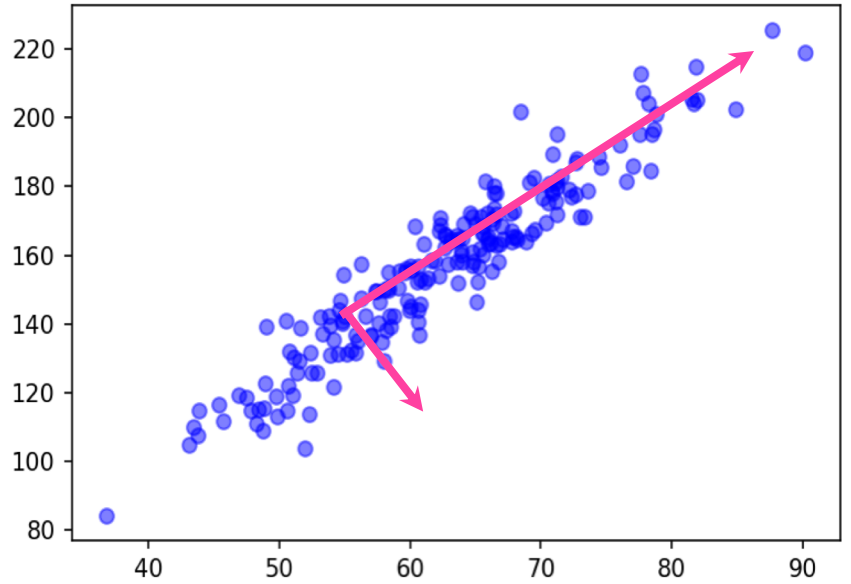


Principal Components Analysis (PCA)

A method of feature extraction

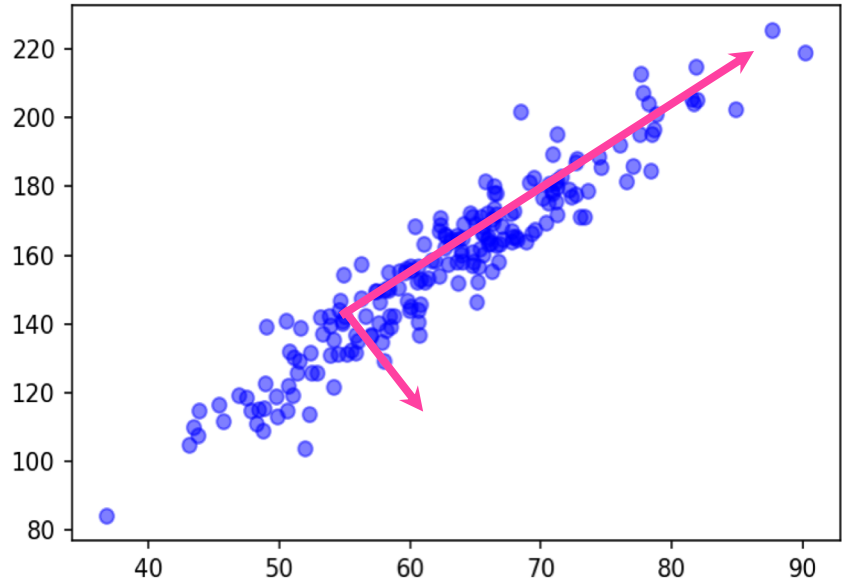
Principal Components Analysis (PCA)

- PCA is an unsupervised technique because we don't make reference to our labels/Y



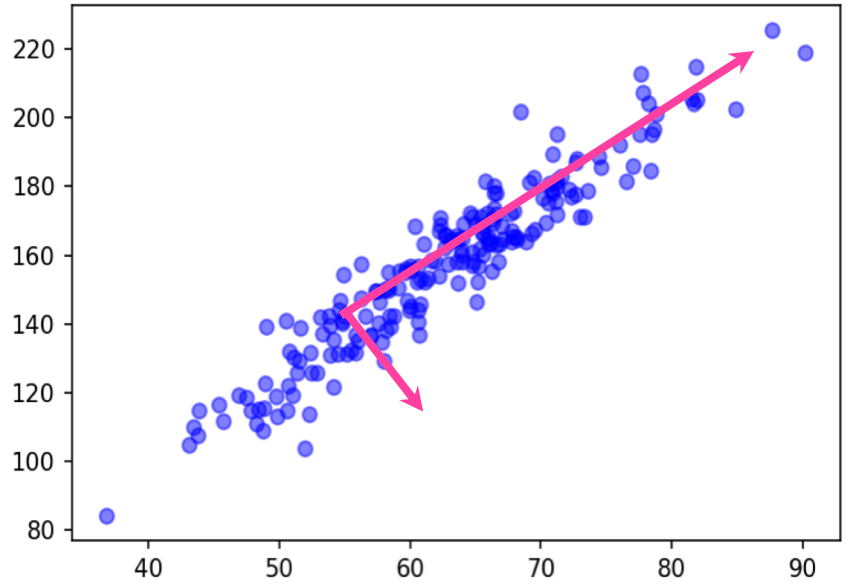
Principal Components Analysis (PCA)

- PCA is an unsupervised technique because we don't make reference to our labels/Y
- It's a technique for representing the direction in our data of most variation



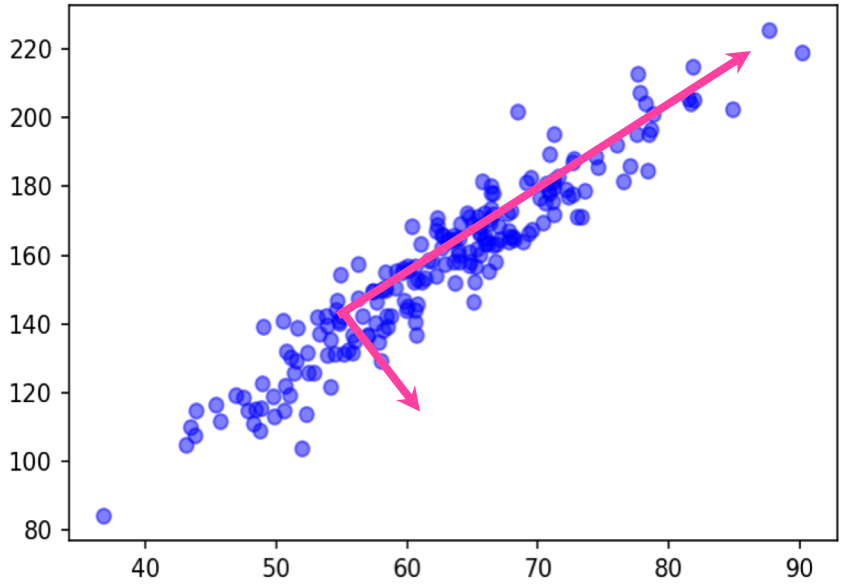
Principal Components Analysis (PCA)

- PCA is an unsupervised technique because we don't make reference to our labels/Y
- It's a technique for representing the direction in our data of most variation
- We care about the direction of maximal variation because that represents the differences in our observations, and will help us in our clustering/classification tasks



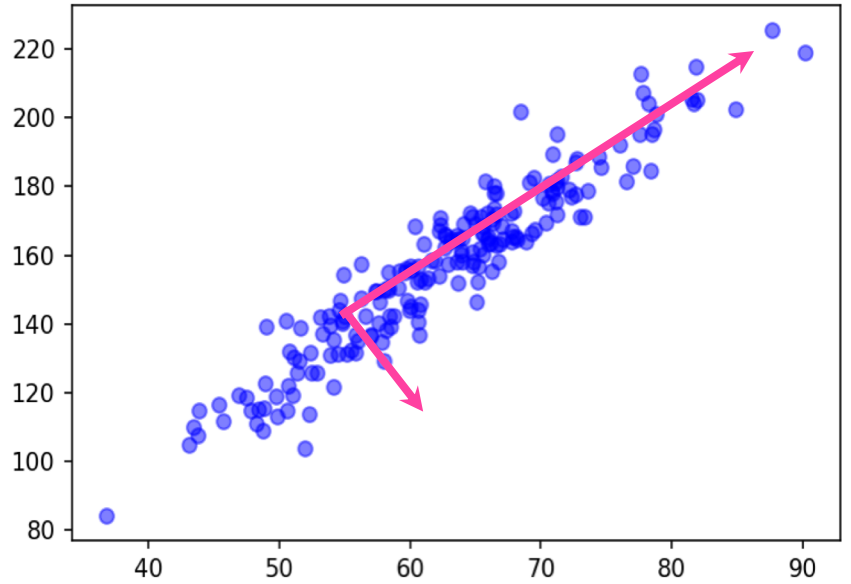
Principal Components Analysis (PCA)

- PCA is an unsupervised technique because we don't make reference to our labels/Y
- It's a technique for representing the direction in our data of most variation
- We care about the direction of maximal variation because that represents the differences in our observations, and will help us in our clustering/classification tasks
- In practice, we use PCA to reduce dimensionality by dropping the components that explain the least variance



Principal Components Analysis (PCA)

- PCA is an unsupervised technique because we don't make reference to our labels/Y
- It's a technique for representing the direction in our data of most variation
- We care about the direction of maximal variation because that represents the differences in our observations, and will help us in our clustering/classification tasks
- In practice, we use PCA to reduce dimensionality by dropping the components that explain the least variance
- Behind the scenes, many PCA implementations are doing SVD



How do we find the principal components?

- Remember, we're not using labels (Y) in our PCA calculation (even if they exist)

How do we find the principal components?

- Remember, we're not using labels (Y) in our PCA calculation (even if they exist)
- Each principal component is a linear combination of all variables, with a weight on each (the loading)

How do we find the principal components?

- Remember, we're not using labels (Y) in our PCA calculation (even if they exist)
- Each principal component is a linear combination of all variables, with a weight on each (the loading)
  - Begin by centering all columns/variables (mean 0)
  - Solve the maximization problem of finding a linear combination of Xs with max variance
  - Continue to find linear combinations with the next-highest variance, but where the component is uncorrelated with all components that came before

How do we find the principal components?

- Remember, we're not using labels (Y) in our PCA calculation (even if they exist)
- Each principal component is a linear combination of all variables, with a weight on each (the loading)
  - Begin by centering all columns/variables (mean 0)
  - Solve the maximization problem of finding a linear combination of Xs with max variance
  - Continue to find linear combinations with the next-highest variance, but where the component is uncorrelated with all components that came before
- Turns out, being uncorrelated is the same thing as being orthogonal (perpendicular)

How do we find the principal components?

- Remember, we're not using labels (Y) in our PCA calculation (even if they exist)
- Each principal component is a linear combination of all variables, with a weight on each (the loading)
  - Begin by centering all columns/variables (mean 0)
  - Solve the maximization problem of finding a linear combination of Xs with max variance
  - Continue to find linear combinations with the next-highest variance, but where the component is uncorrelated with all components that came before
- Turns out, being uncorrelated is the same thing as being orthogonal (perpendicular)
- For PCA, start with a variance/covariance matrix of our data: X'X

How do we find the principal components?

- Remember, we're not using labels (Y) in our PCA calculation (even if they exist)
- Each principal component is a linear combination of all variables, with a weight on each (the loading)
  - Begin by centering all columns/variables (mean 0)
  - Solve the maximization problem of finding a linear combination of Xs with max variance
  - Continue to find linear combinations with the next-highest variance, but where the component is uncorrelated with all components that came before
- Turns out, being uncorrelated is the same thing as being orthogonal (perpendicular)
- For PCA, start with a variance/covariance matrix of our data: X'X
  - Eigenvectors = directions
  - Eigenvalues = variances

PCA is not a panacea for high dimensions

- PCA is looking for linear relationships between variables; if they're related in some different way, PCA won't help us
- **Standardize your data, otherwise PCA will just pick out variables that are on larger scales (higher variance)**
- Since we're not using Y, the variance PCA picks up on may not be as meaningful for our use case
- Components are not easy to interpret, so you lose interpretability

Applications of dimensionality reduction

- NLP
  - Topic modeling: axes are topics and documents cluster around axes
  - Recommender systems
- Visualization
- Reduce computational complexity by reducing amount of data we're storing and processing
- Data compression with little loss (e.g. image compression)