Introduction to Machine Learning

Module 7: Unsupervised learning

Module Checklist:

- ❏ Overview of unsupervised learning
  - ❏ Intuition
  - ❏ Pros and cons
- ❏ Clustering algorithms
  - ❏ K-means clustering
  - ❏ Principal component analysis (PCA)

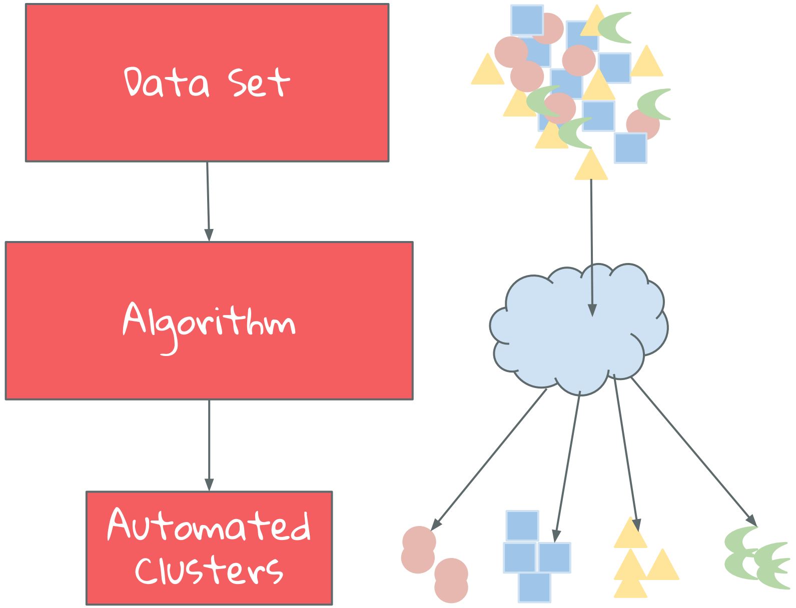
In the first half of this course, we learned a lot about supervised learning algorithms. Now, we turn to **unsupervised learning algorithms**: what are they, when are they useful, and what role do they play in the future of machine learning?

Recap: Supervised v. Unsupervised Learning

Supervised Learning	Unsupervised Learning
• For every x, there is a y • Goal is to predict y using x • Most methods used in practice are supervised.	• For every x, there is no y • Goal is not prediction, but to investigate x. • Unsupervised methods read data first and then suggest which classification scheme(s) might apply.

Unsupervised learning

Many unsupervised learning algorithms involve identifying patterns.

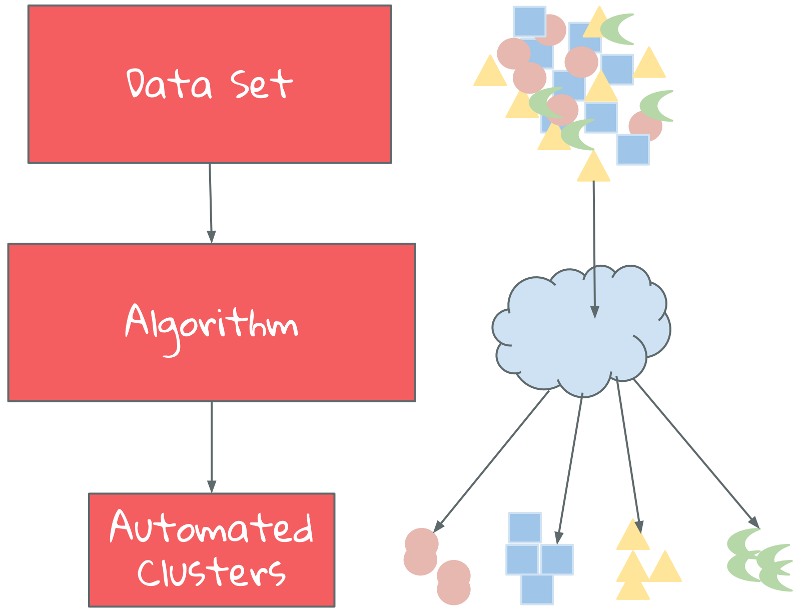


Unsupervised learning involves ***identifying patterns***.

This is perhaps the most basic human task - even babies can do it!

Unsupervised learning

Many unsupervised learning algorithms involve identifying patterns.



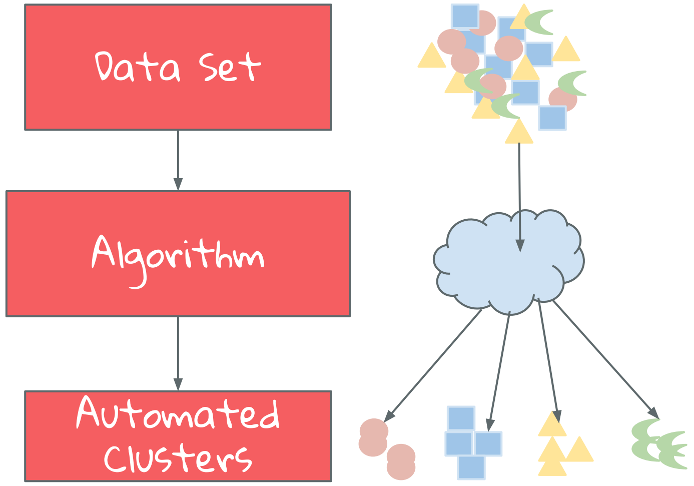
It turns out that what's extremely intuitive for human babies to do **is pretty difficult to have computers do well.**

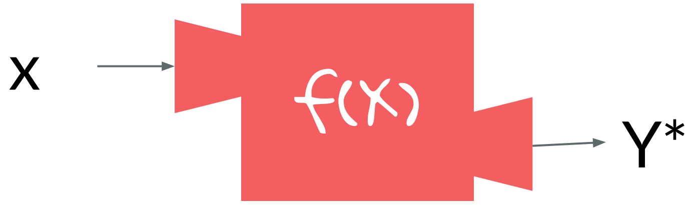
How is this model framework different than what we've already seen?

Task

There is no f(x).

Instead of trying to choose the best f(x) to map x to the true Y, we just want to understand raw patterns in x.





How is this model framework different than what we've already seen?

Task	There is no f(x).	Instead of trying to choose the best f(x) to map x to the true Y, we just want to understand raw patterns in x.
Learning Methodology	There is no loss function.	We do not learn from the data by trying to reduce our loss to 0.
Performance	Difficult to evaluate performance	Evaluating the performance of clustering methods is notoriously difficult.

Unsupervised algorithms still have a task, and involve feature engineering and selection.

Task	What is the problem we want our model to solve?
What is the unsupervised approach we plan to use?	For example, is hierarchical clustering better than k-means for this task?
Feature engineering & selection	How do we decide what features to include in our model?

Unsupervised algorithms are still machine learning algorithms, which means they actively learn from the data.

Learning Methodology	Clustering algorithms are unsupervised. How does that affect the learning process?
How does our ML model learn?	Overview of how the model teaches itself.
Optimization process	How do unsupervised learning models optimize if there is no loss function?

Unsupervised algorithms are unfortunately very difficult to evaluate for performance.

Performance — How do we evaluate the performance of an unsupervised algorithm?

With supervised learning we had a clear goal: predict our outcome variable with high accuracy. ***Evaluating the performance of clustering methods is difficult. Often, we rely on holistic evaluations of our clustering algorithm.***

Unlike with supervised learning, we cannot check the performance of our model by comparing loss function. Clustering quality is ***subjective***, and largely ***dependent on the assumptions*** made at the outset.

Pros and cons of using an unsupervised approach

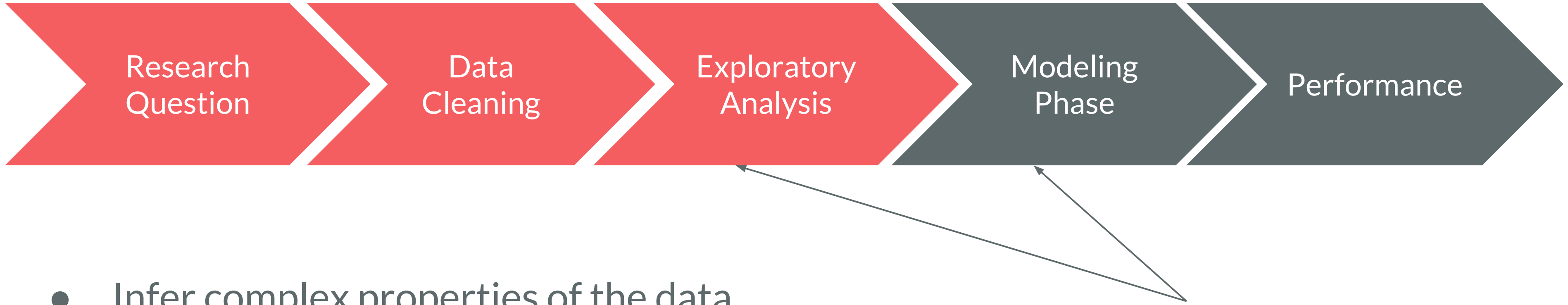
When would I turn to unsupervised learning?

- You have extremely high dimensional data (i.e., many features) that you want to investigate
- You have a research question but no labelled outcome feature
    - This is true for many datasets
- You want to detect any relationships or patterns in your data
    - E.g. customer behavior data
- You don't have time to dive deep into defining an outcome
    - Use unsupervised learning as first exploratory step

As the amount of data in the world grows, we will increasingly turn to unsupervised learning methods.

Unsupervised Learning

Unsupervised learning is an important tool, often used as part of your exploratory analysis of the data.



- Infer complex properties of the data (such as sub groups)
- Discover interesting and informative methods of visualization
- Often used in exploratory phases of data analysis

You can often leverage an unsupervised algorithm to aid feature selection during exploratory analysis, before using a supervised algorithm during the modeling phase.

Unsupervised Learning

The future of machine learning is unsupervised learning.

Supervised learning is the icing on the cake

Unsupervised learning is the cake itself

Humans learn mostly through unsupervised learning: we absorb vast amounts of data from our surroundings without needing a label.

To reach true machine intelligence (i.e., a machine that thinks and learns for itself), ML needs to get better at **unsupervised** learning - it should learn without us having to feed it labels or explicit instructions.

We will have only scratched the surface in this class.

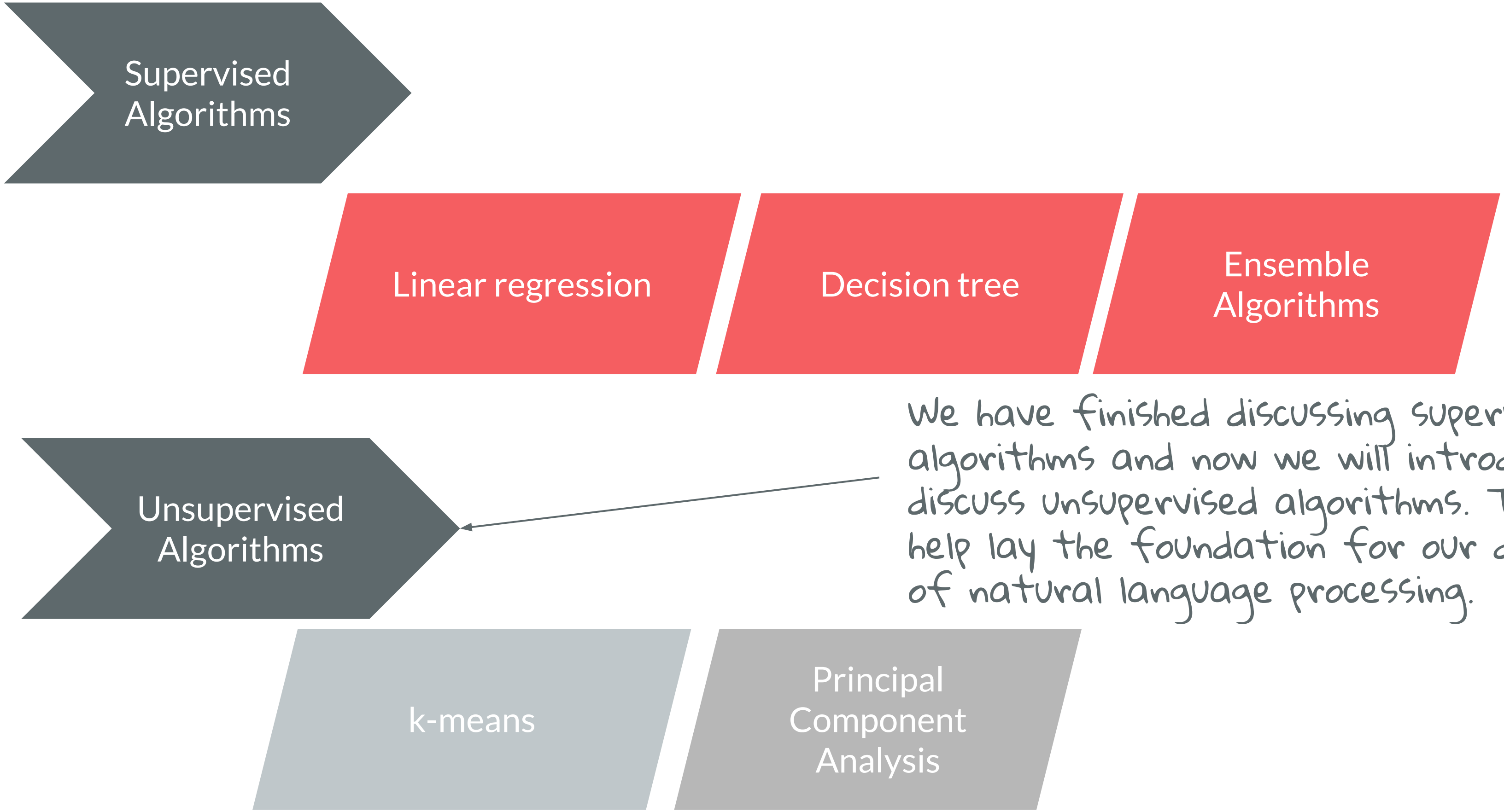
With that said, let’s turn to our first unsupervised algorithms: **clustering algorithms**.

Clustering Algorithms

1. K-means clustering
2. Principal component analysis

Central concept: identify similar sub-groups within the data.

Where are we?



We have finished discussing supervised algorithms and now we will introduce and discuss unsupervised algorithms. This will help lay the foundation for our discussion of natural language processing.

Unsupervised Algorithms
K-means clustering
Principal Component Analysis

Let’s start with K-means clustering, an algorithm that splits data into k clusters along features that you select.

1. K-means clustering

K Means Clustering: model cheat sheet

Pros

- Easy to represent physically
- Does not assume any underlying distribution (e.g. no normal distribution assumption like in linear regression)
- Produces intuitive groupings
- Can work in many dimensions

Cons

- Time-consuming to find the optimal number of clusters
- Time-consuming feature engineering (features must be numeric and normalized)

Assumptions

- Assumes existence of underlying groupings

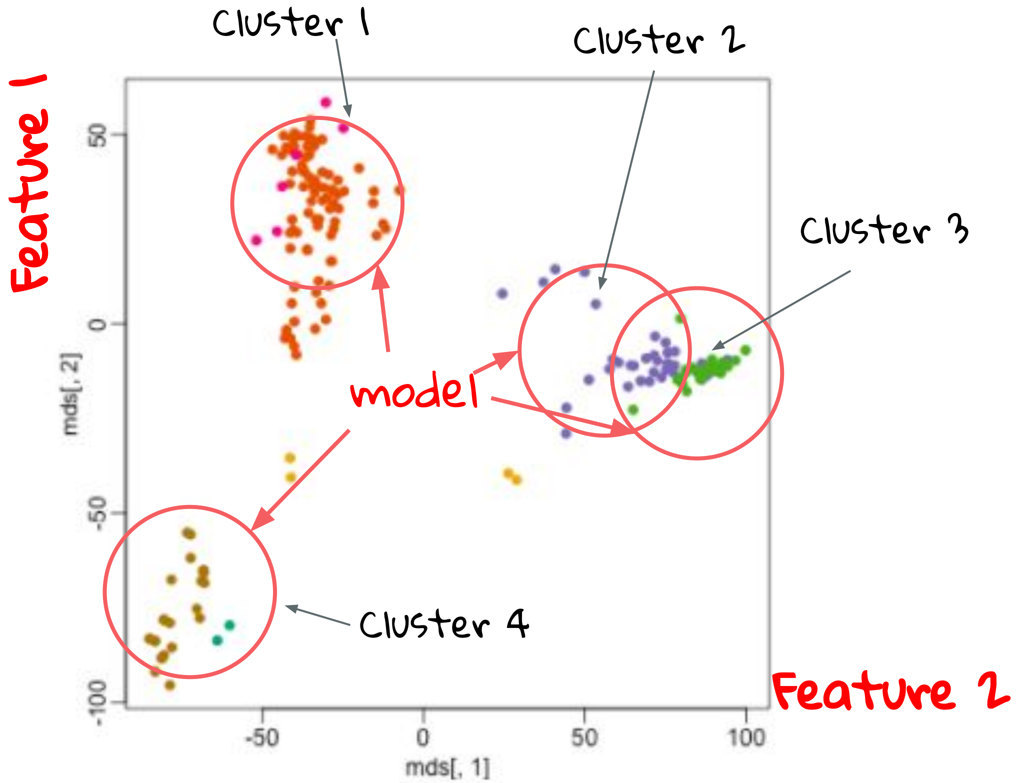
Clustering Task

Clustering is a powerful unsupervised algorithm that detects naturally occurring patterns in the data.

Clustering splits data in order to find out how observations are similar on a number of different features.

We are not predicting a true Y.

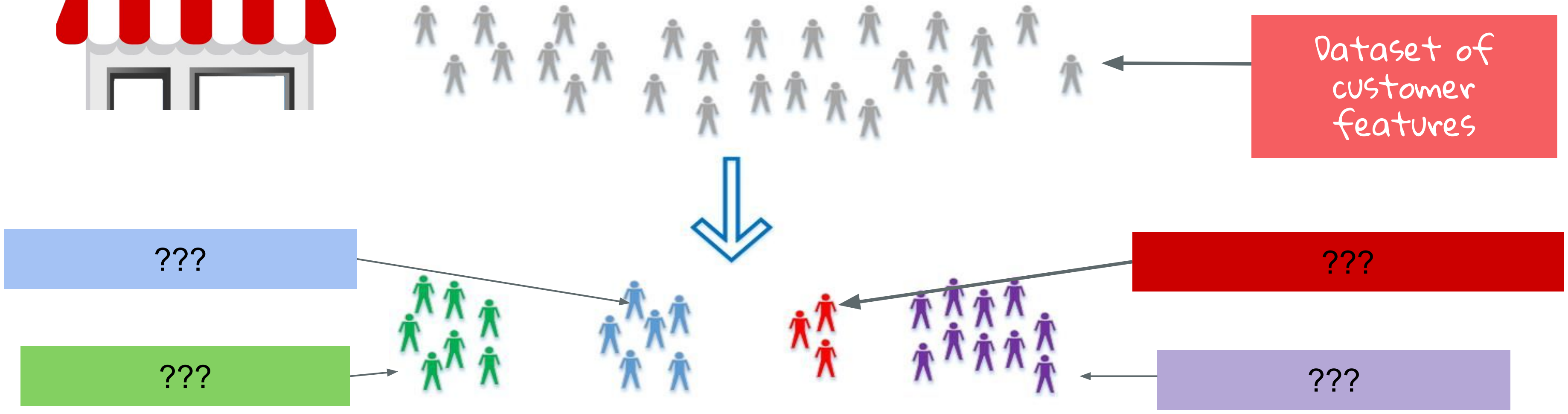
The clusters are the model. We decide the number of clusters, represented as K.



Clustering Task

Defining Groups

Imagine that you are the owner of an online clothing store. You want to segment your customers by their buying habits.



Clustering Task → Defining Groups

Imagine that you are the owner of an online clothing store. You want to segment your customers by their buying habits.

NOTE: As the owner of the store, you ***think*** that some customers are bargain-hunters while others are price-insensitive; some come in every week while others drop in during the holidays.

But you don't actually know definitively which ones are which.

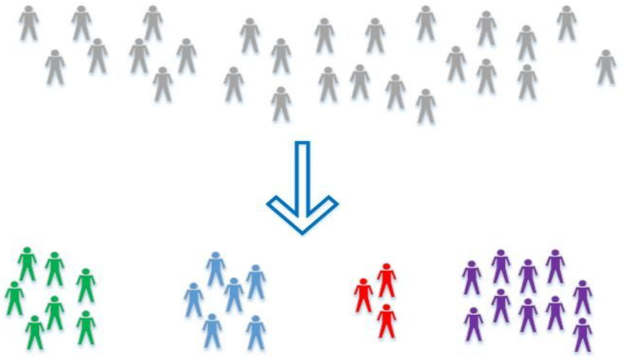
You **could** take a supervised approach to this problem, and try to **predict** how much a customer will spend, or how frequently a customer will drop in. But here, we're just trying to **see what the data will tell us**. This is what makes clustering an unsupervised problem.

Clustering Task

Since we have an online website for our store, we have valuable data about how our consumers behave online.

Dataset of customer features

customer_id	Number of visits	Avg_amt_spent	traffic type	%_of_visits_during_sales
	X1	X2	X3	X4
1237482	5	$92	organic	20%
1213345	50	$35	Email_sale	100%
2323764	20	$200	Email_new_collection	10%
2326734	1	$40	organic	100%



What features will be relevant to our clustering task?

Clustering Task

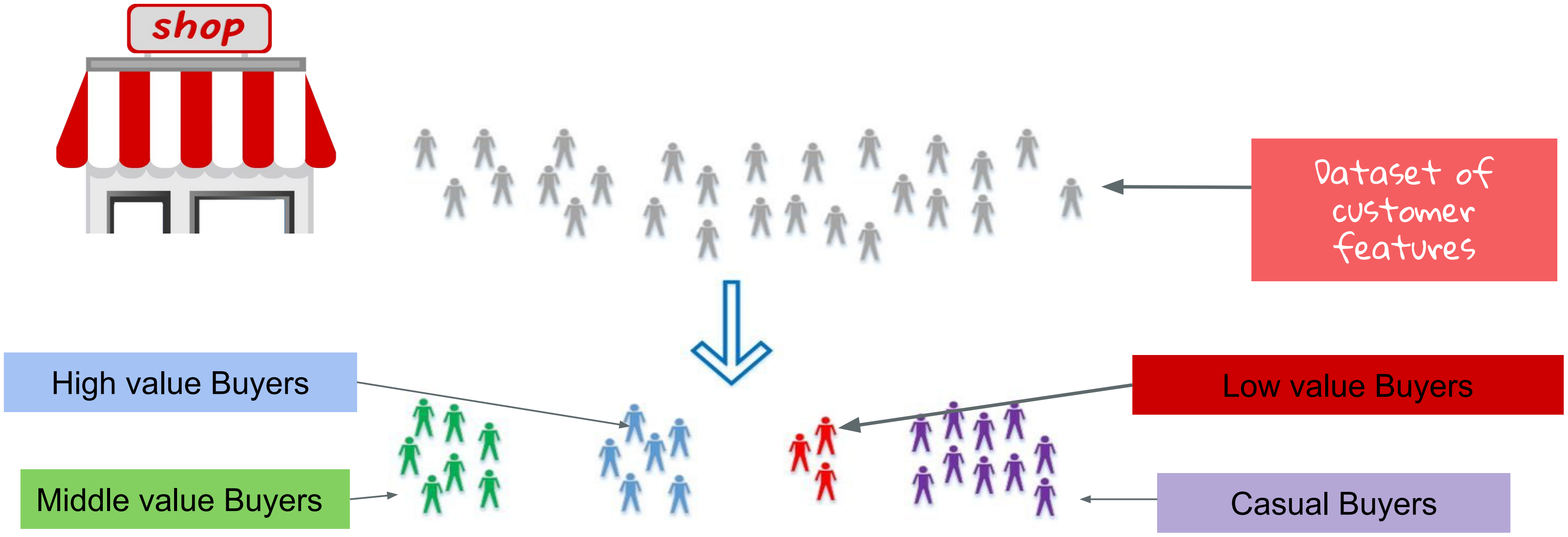
We select features that will determine how clusters are formed in our algorithm.

customer_id	Number of visits	Avg_amt_spent	traffic type	%_of_visits_during_sales
	X1	X2	X3	X4
1237482	5	$92	organic	20%
1213345	50	$35	Email_sale	100%
2323764	20	$200	Email_new_collection	10%
2326734	1	$40	organic	100%

Since we want to segment customers by their buying habits, we probably want to form clusters using the features “number of visits” and “average amount spent.” Let’s start with these.

Clustering Task → Defining Groups

Using number of visits and amount spent, we can see groups below:

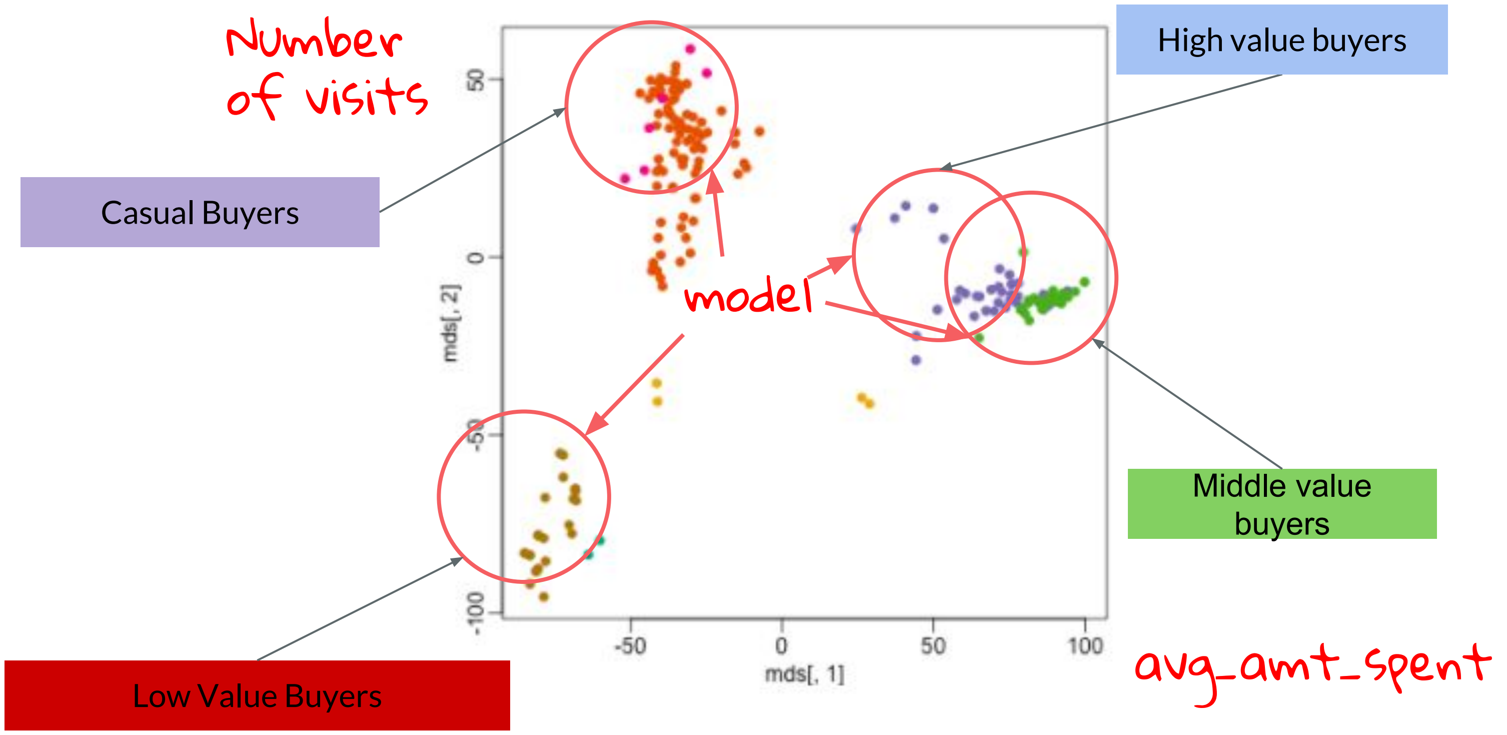


Task

Defining groups

We can demonstrate clustering using two features in two dimensional space!

We use *avg_price_clicked_on* and *number_of_visits* to cluster our customers.



Task

Defining groups

Using two features, we can say something about our customers.

Once we cluster using the features we select, we can say something about the value of our customers.

High value buyers	Middle value buyers	Low Value Buyers	Casual Buyers
• Not price-sensitive • Frequent buyers	• Price-sensitive • Infrequent buyers	• Not price-sensitive • Infrequent buyers	• Price-sensitive • Frequent buyers

Task

Feature Engineering & Selection

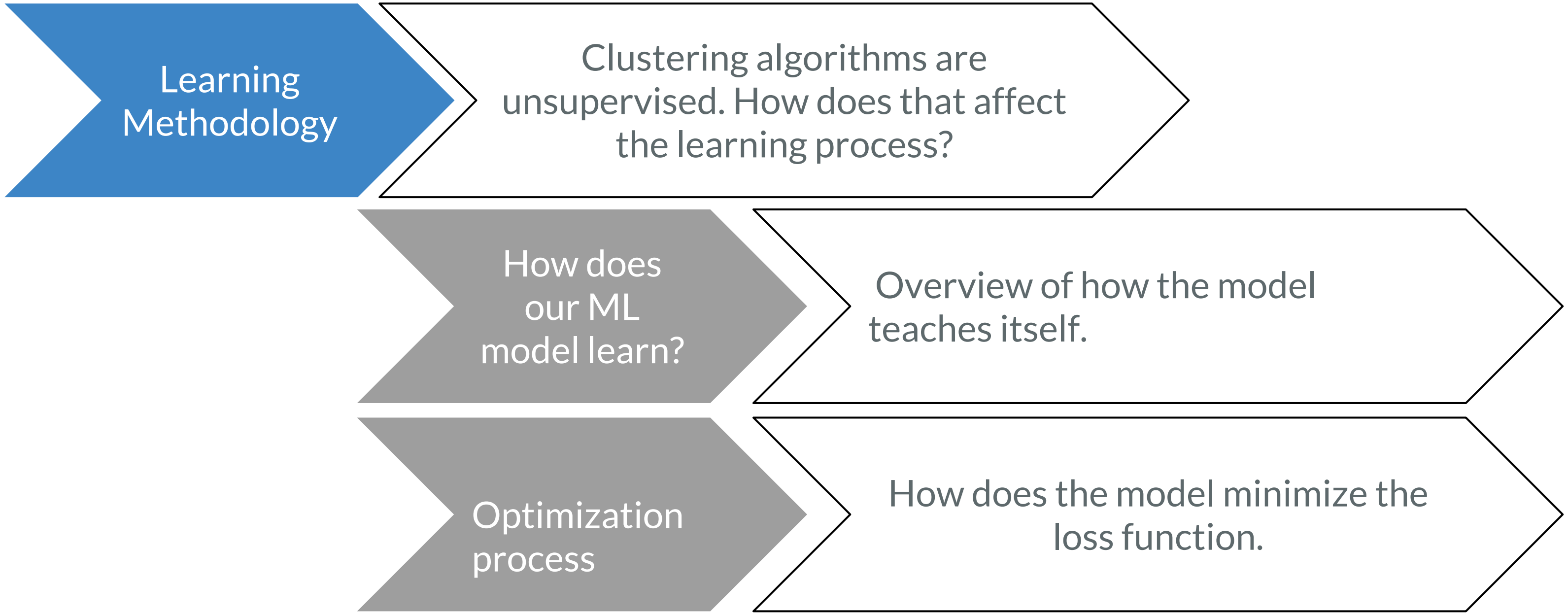
Feature selection is just as important in unsupervised as in supervised algorithms.

In this simple example, we used 2 features to cluster, and produced 4 different clusters (K=4).

But we can include as many features as we want, and determine how many clusters to produce. We will return to this point later, but first, let’s find out how this algorithm learns.

How does our k-mean algorithm learn from the data to group similar observations together?

Unsupervised algorithms are still machine learning algorithm, which means they actively learn from the data!


Learning Methodology

Clustering algorithms are unsupervised. How does that affect the learning process?

How does our ML model learn?

Overview of how the model teaches itself.

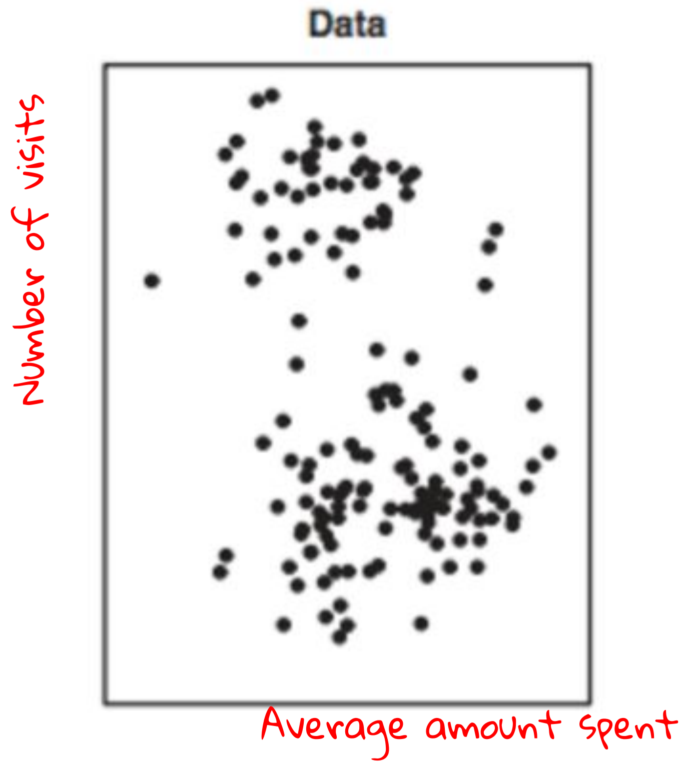
Optimization process

How does the model minimize the loss function.

Learning Methodology

How does our ML model learn?

We start with a simple scatter plot of number of visits against average amount spent.



How do we take this scatter plot and segment observations into K distinct groups?

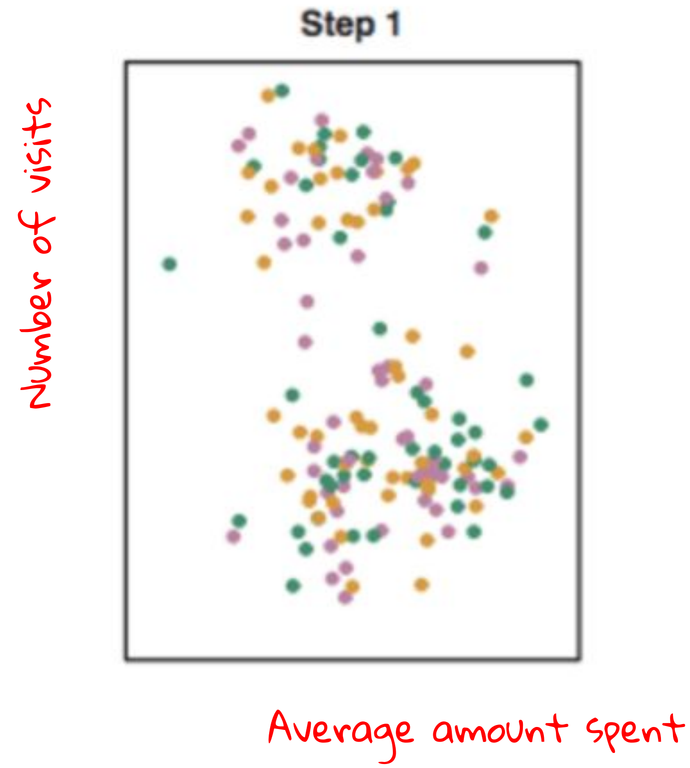
For now, let's say K = 3.

Source: Intro to Statistical Learning with Applications in R.pdf

Learning Methodology

How does our ML model learn?

Step 1: Randomly assign each customer to a cluster



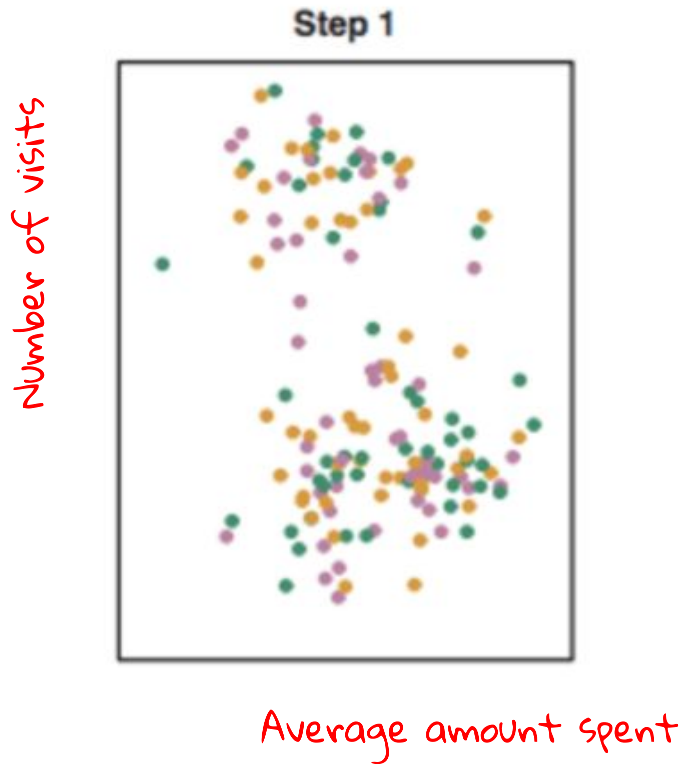
- Data points have **randomly** been assigned into pink, yellow, and green groupings

There are 3 colors (groups) here. Why?

Learning Methodology

How does our ML model learn?

Step 1: Randomly assign each customer to a cluster



- Data points have **randomly** been assigned into pink, yellow, and green groupings

There are 3 colors (groups) here because we decided K=3.

Learning Methodology

Why were our data points randomly assigned to three groupings? We set K, or number of groups, equal to 3

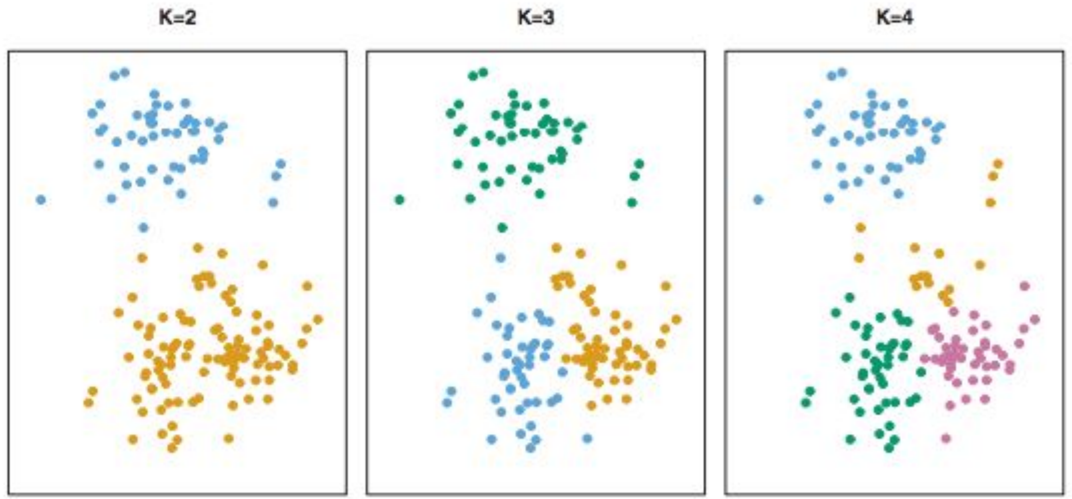
The number of clusters (k) is an example of a *hyperparameter.*

Hyperparameters are set by the researcher (you!), not determined by the model.

Hyperparameters	Higher level settings of a model that are fixed before training begins.

Learning Methodology

We can set our hyperparameter K to be 2,3,4...n.



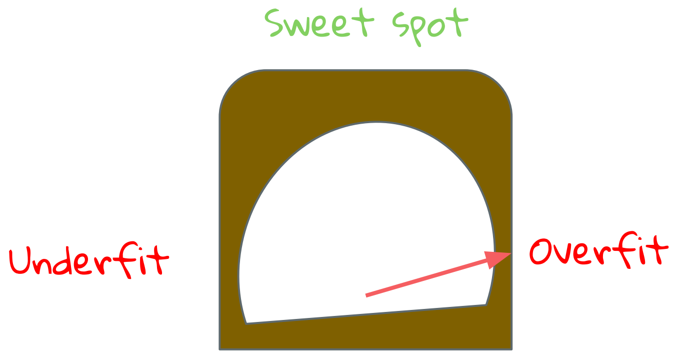
In our example we specified 3 clusters, but we can tell the model **whatever "K" we want.**

What should k be? A very large number of clusters will cause overfitting (for example, if you set n equal to the number of observations you will have a cluster for every observation!) This is not very useful for making sense of subsets of our data.

Performance

Recall overfitting vs. overfitting? If K is too large, we can have overfitting; too small and we may be underfitting.

If we are underfitting, we may be missing natural subsets of similar customers. If we are overfitting, we may have too many clusters, so our model does not generalize well to unseen data.

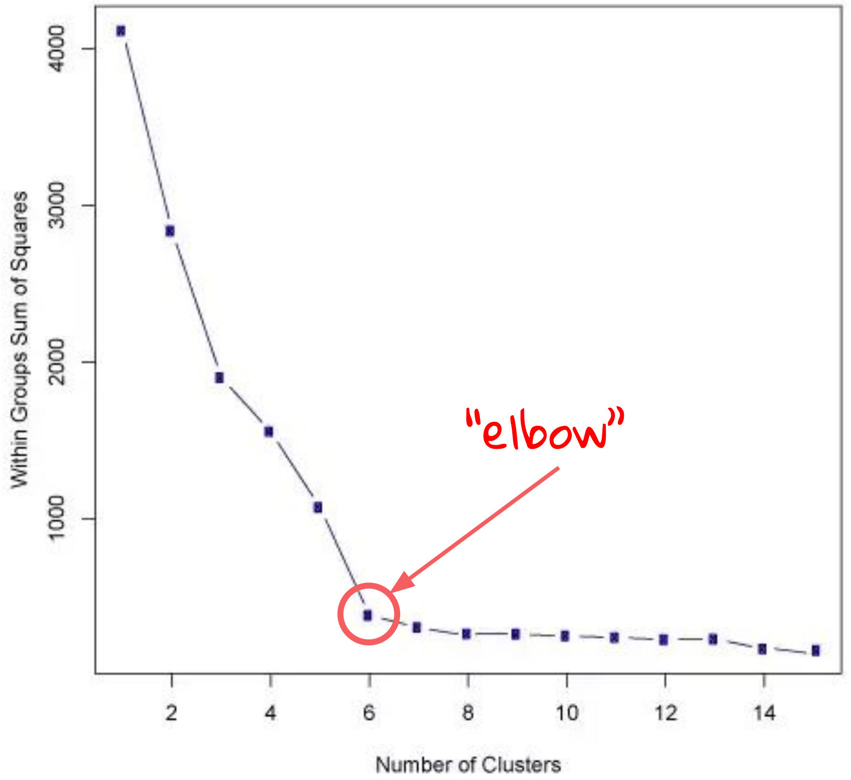


An elbow plot helps us avoid underfitting by showing how model error decreases with the number of clusters.

How do we know what the optimal number of clusters should be?

An Elbow Plot visualizes how the error decreases as K increases.

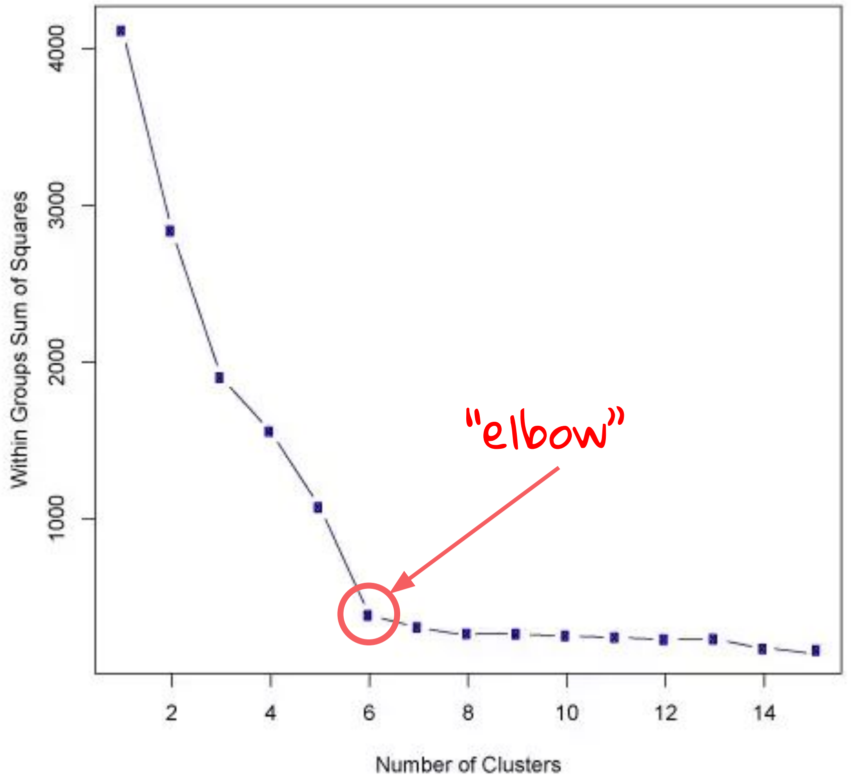
This plot is useful because it visualizes the ***trade-off*** between overfitting and underfitting. We need to find a balance, called the "elbow point".



An elbow plot helps us avoid underfitting by showing how model error decreases with the number of clusters.

*If K == n (number of observations), then distance = 0 (each observation its own cluster)! **This, as we know, is a case of overfitting.***

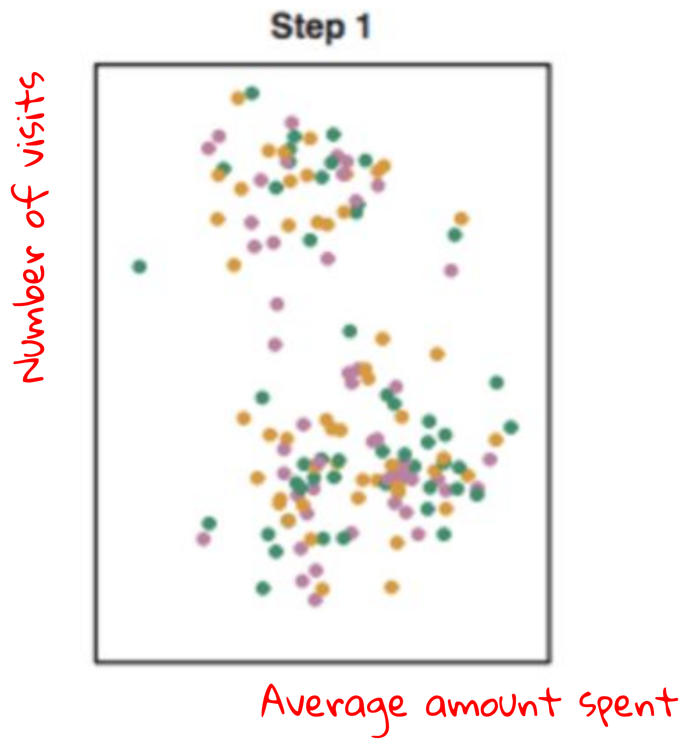
In the graph to the right, we should choose K=6. This appears to be the number of clusters where the **biggest gains in reducing error have already been made.**



Learning Methodology

How does our ML model learn?

To start with, let's set k=3. This means we want to find 3 clusters.



We suspect these 3 clusters will correspond to:

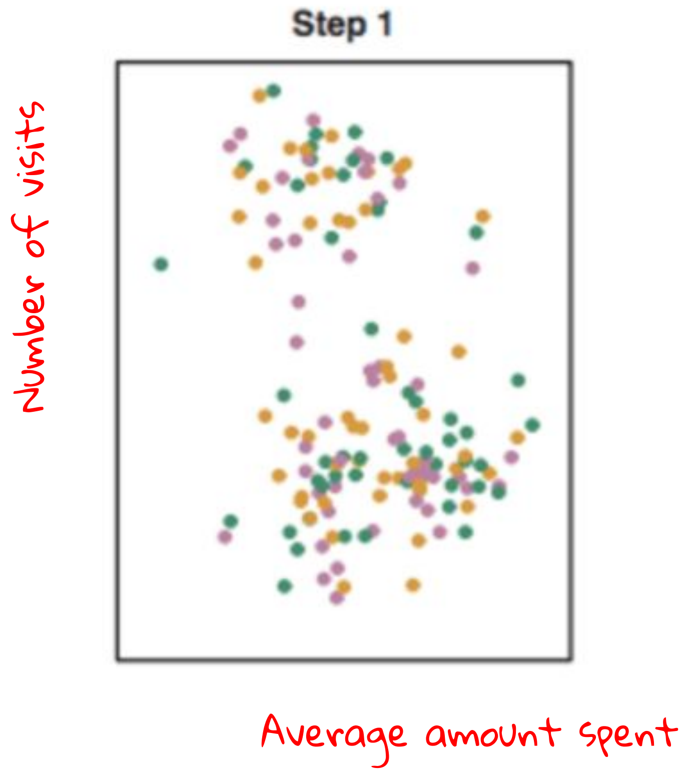
1. Budget customers that shop with us frequently
2. High end customers that shop with us frequently
3. Customers that shop with us infrequently

However, it is possible and in fact likely that there are naturally more than 3 subsets of customers.

Learning Methodology

How does our ML model learn?

Is our initial random allocation of observations to clusters useful?



Remember our first step was to randomly assign each customer to a cluster.

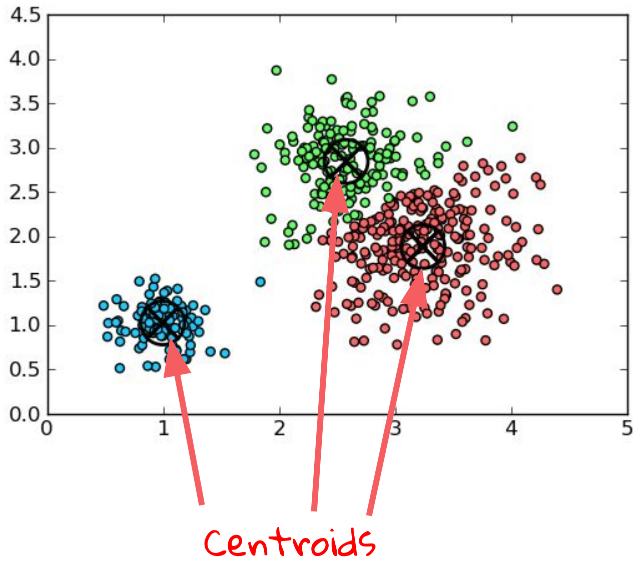
Clearly, our initial random assignment to clusters is poor. **How poor? And how do we improve?**

Learning Methodology

How does the model learn?

To evaluate how poor our clusters are, we can use **the distance between an observation and its cluster's centroid**.

A centroid is the center point of a cluster.



Learning Methodology

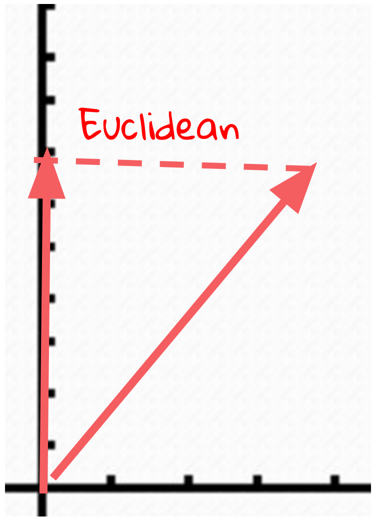
How does the model learn?

Review! How do we calculate distance between points?

- The “distance” here is the Euclidian distance (or spatial distance) where the distance between two vectors u and v with n elements is:

$$d = \sqrt{\sum_n (u_i - v_i)^2}$$

- In this example, the difference between customer c6 (7,2) and the cluster center (4,7) would be sqrt[(7-4)^2 + (2-7)^2] = 5.8.

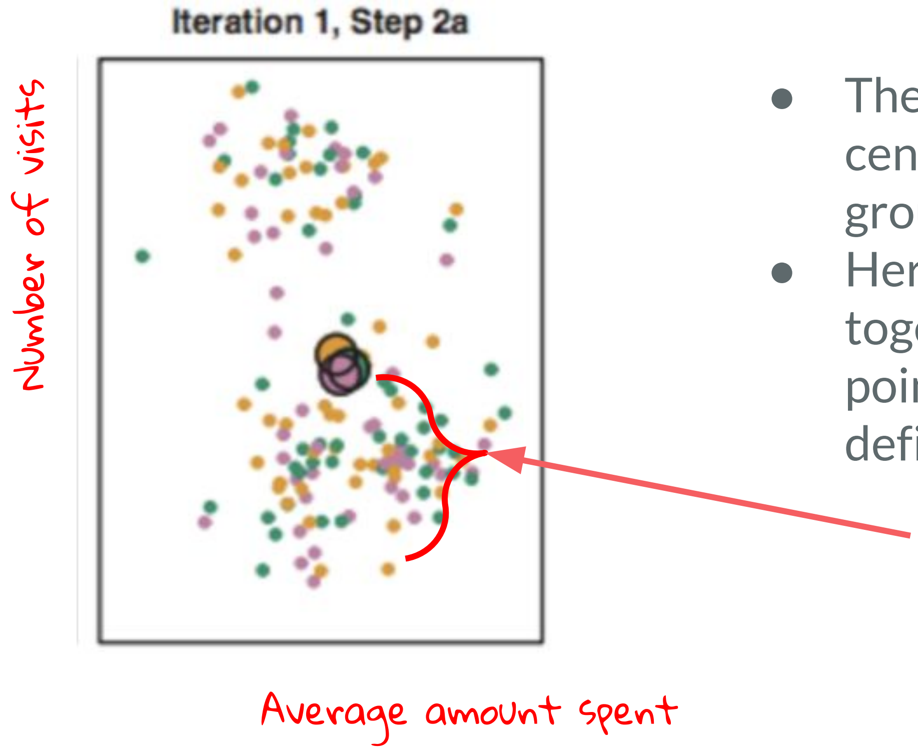


Important note: Clustering does not take categorical features as inputs, only continuous. Distance between categorical points would not be meaningful.

Learning Methodology

How does the model learn?

We calculate a centroid for each cluster. Our goal is to minimize the distance from centroid to any observation.



- The centroids are calculated as the center of their randomly assigned group.
- Here, centroids are really close together, and distance to the outer points is very large -- we can definitely do better!

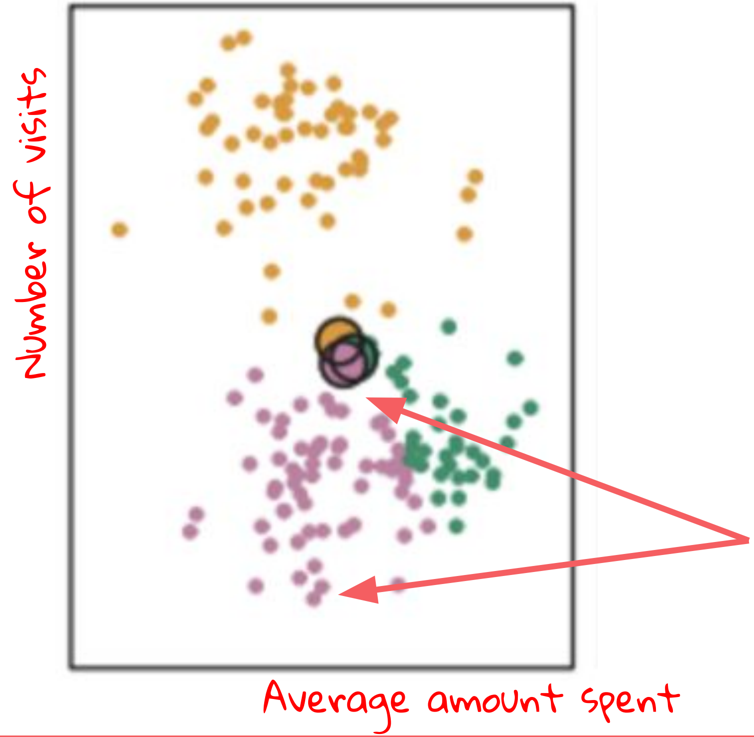
Goal is to minimize distance from centroid to any of the observations in its cluster

Learning Methodology

How does the model learn?

Step 2b: Re-assign each observation to the centroid it is nearest

Iteration 1, Step 2b



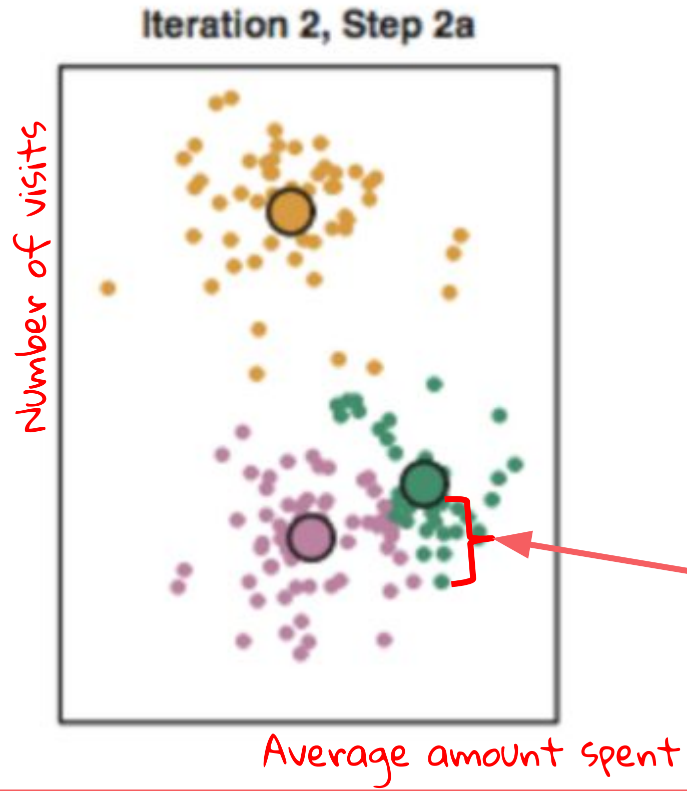
- Now we re-assign each observation to the cluster group of the nearest centroid. Now our clusters are starting to look better!

Of the three centroids, this observation is closest to the pink, so we assign it to the pink cluster

Learning Methodology

How does the model learn?

Iteration 2 Step 2a: Repeat centroid calculations!



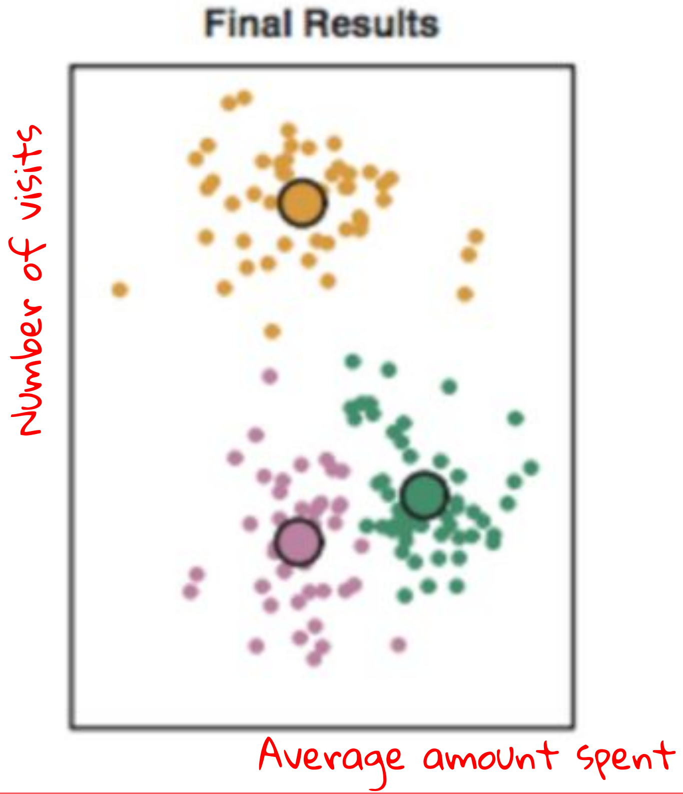
- Here we go again! Our recalculated centroid are now further apart, and we can see the distance minimization between observations and centroids

Overall distances are much smaller -- we're getting closer!

Learning Methodology

Optimization process

Iteration 2 Step 2b: Re-assign clusters based on nearest centroid



- This time around, fewer observations changed clusters.
- We keep repeating the centroid calculation / cluster re-assignments **until nothing moves anymore** - the model is complete!

Now that we know the mechanics of the K-means algorithm, let's think through an example that you can recreate in the Kiva data.

How do we cluster types of loans requested on the Kiva website?

Clustering Exercise 1

How would we group these loans into groups most similar in characteristics?

Clustering task:

Loan 4:
$25
1 borrower
Funded in 1 day

Loan 3:
$50
3 borrowers
Funded in 30 days

Loan 1:
$25
1 borrower
Funded in 1 day

Loan 2:
$1000
2 borrowers
Funded in 30 days

Loan 5:
$2000
5 borrower
Funded in 1 day

Loan 11:
$3500
3 borrowers
Funded in 40 days

Loan 6:
$1000
1 borrower
Funded in 1 day

Loan 7:
$75
1 borrower
Funded in 60 days

Loan 8:
$25
1 borrower
Funded in 1 day

Loan 9:
$50
1 borrower
Funded in 1 day

Loan 10:
$25
1 borrower
Funded in 1 day

Clustering Exercise 1

We can use the features "time to fund" and "size of loan"

Loan 9:
$500
4 borrower
Funded in 2 days

Loan 5:
$2000
5 borrower
Funded in 1 day

Loan 6:
$1000
1 borrower
Funded in 1 day

Big loans many borrowers funded quickly

Loan 11:
$3500
3 borrowers
Funded in 40 days

Loan 2:
$1000
2 borrowers
Funded in 30 days

Loan 3:
$50
3 borrowers
Funded in 30 days

Loan 7:
$75
1 borrower
Funded in 60 days

Long time to fund

Loan 1:
$25
1 borrower
Funded in 1 day

Loan 4:
$30
2 borrower
Funded in 1 day

Loan 8:
$25
1 borrower
Funded in 1 day

Loan 10:
$25
2 borrower
Funded in 1 day

Small loans few borrowers funded quickly

Clustering Task

Feature Engineering & Selection

What if we want to use more than two features to cluster?

We want to group similar loans using:

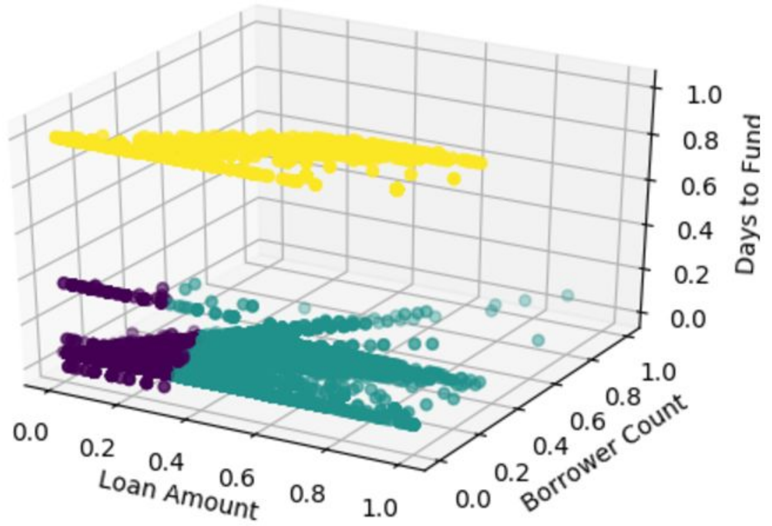
- Loan Amount - How much $ did the borrower request?
- Borrower Count - Is this a request from a single borrower or a group of borrowers?
- Time to Fund - How long did it take for the request to be funded on the site?

Task

We can extend our clustering to higher dimensions by adding more features.

So far, our clusters have been visualized in 2D for simplicity.

But this can be extended to ***n*** dimensions. The plot to the right uses 3 features to cluster, this results in a 3D visualization.



The true utility of clustering comes from making sense of high dimensional data.

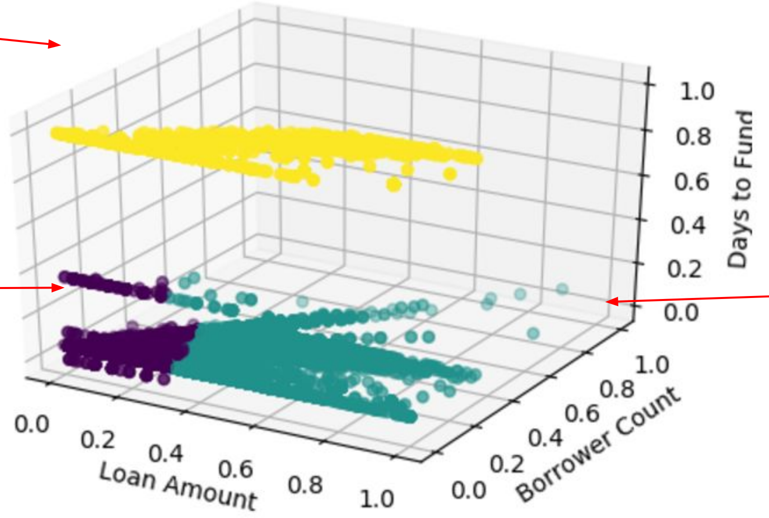
Clustering Task

Feature Engineering & Selection

Example of clustering using 3 features: time to fund, the number of borrowers and the loan amount.

Last-minute funded loans

Small loans with few borrowers that fund quickly



Clustering Features:

- Time to Fund
- Borrower Count
- Loan Amount

Big loans that fund quickly

Fascinating! If you were the Kiva CEO, what would you do with this information?

We can change up how many features we use to cluster our data, but we can also **choose the number of clusters (k)** we want to see in our data.

Exercise 2

In this example, we see three separate clusters...

Loan 9:
$500
4 borrower
Funded in 2 days

Loan 5:
$2000
5 borrower
Funded in 1 day

Loan 6:
$1000
1 borrower
Funded in 1 day

Loan 11:
$3500
3 borrowers
Funded in 40 days

Loan 2:
$1000
2 borrowers
Funded in 30 days

Loan 3:
$50
3 borrowers
Funded in 30 days

Loan 7:
$75
1 borrower
Funded in 60 days

Loan 1:
$25
1 borrower
Funded in 1 day

Loan 8:
$25
1 borrower
Funded in 1 day

Loan 4:
$30
2 borrower
Funded in 1 day

Loan 10:
$25
2 borrower
Funded in 1 day

Exercise 2

What if we thought there were 4 distinct groupings?

Loan 9:
$500
4 borrower
Funded in 2 days

Loan 5:
$2000
5 borrower
Funded in 1 day

Loan 6:
$1000
1 borrower
Funded in 1 day

Big loans many borrowers funded quickly

Loan 11:
$3500
3 borrowers
Funded in 40 days

Loan 2:
$1000
2 borrowers
Funded in 30 days

Big Loans Long time to fund

Loan 3:
$50
3 borrowers
Funded in 30 days

Loan 7:
$75
1 borrower
Funded in 60 days

Small Loans Long time to fund

Loan 1:
$25
1 borrower
Funded in 1 day

Loan 4:
$30
2 borrower
Funded in 1 day

Loan 8:
$25
1 borrower
Funded in 1 day

Loan 10:
$25
2 borrower
Funded in 1 day

Small loans few borrowers funded quickly

K-means clustering is useful when we want to identify groups in our data based on similarities across specific features.

But what if you have 250 features? It's difficult to wrap your head around clusters based on this many features.

Principal Component Analysis can help! PCA is an unsupervised algorithm that helps determine which features have **the most explanatory power, and are therefore the most important to include**.

2. Principal Component Analysis

What is Principal Component Analysis?

PCA finds which features are most correlated in a dataset, and removes them, leaving you with **the most "important" features** - i.e., the "principal components."

PCA is helpful for **feature selection** and **engineering**

What is Principal Component Analysis?

Consider the following mini dataset:

cumulative_gpa	last_year_gpa	test_scores	attendance	tutoringYN
3.55	3.65	89%	91%	Y

If you wanted to choose a single feature from this dataset to represent the entire dataset, you would want to select the one that contains **the most information**. Here, the feature "*cumulative_gpa*" intuitively contains more information than "*last_years_gpa*."

This is the fundamental idea of PCA: select the features that contain the most information.

PCA Task | Dimension reduction

PCA is a dimension reduction technique.

This is why PCA is sometimes called a "*dimension reduction*" technique: by seeing which features are important, we can remove the unimportant features. Recall the difference between **low** and **high** dimension data:

low

# rooms	House price
1	32,000
3	100,000
4	232,000
2	50,000
...	...

high

x	y	z	a	b	...
John	31	M	21st St.	CH	...
Jane	42	F	3rd Ave	KE	...

*PCA tells us **which features** are important and informative when we have very high dimensional data!*

PCA Task

Dimension reduction

PCA is more rigorous than picking and choosing features yourself

When faced with a gigantic, high-dimensional dataset, you could pick and choose the features **you** think are important.

But this can be inefficient, and even worse, can lead to us missing out on potentially important data! PCA can provide a rigorous way of determining which features are important.

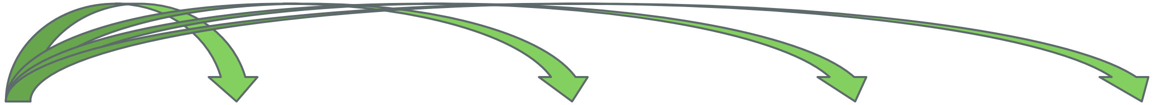
studentid	cumulative_gpa	last_year_gpa	test_scores	attendance	tutoringYN	
1	3.55	3.65	89%	91%	Y	...
2	2.76	2.50	73%	90%	Y	...
...	...	...	...	...	...	...

Learning Methodology

How does the model learn?

How does PCA work?

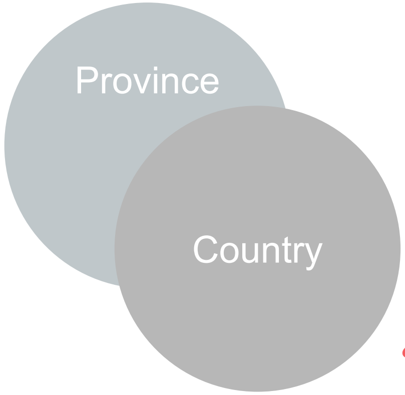
So how do we determine which features contain the **most information**? We measure **variation** (or, how *different* a feature is from another) as a proxy for **information**.



studentid	cumulative_gpa	last_year_gpa	test_scores	attendance	tutoringYN
1	3.55	3.65	89%	91%	Y
2	2.76	2.50	73%	90%	Y
...	...	...	...	...	...

How does variation relate to information?

Why does measuring variation make sense? Recall our discussion of multicollinearity back in Module 4, Linear Regression. When two features are correlated, it doesn't make sense to include them both in a model, because *you only need one to capture the information of both.*



In PCA, we use a similar concept to **detect and remove** redundant and non-informative features. If a feature contributes very little **variation**, it can be removed.

See Module 4, Linear Regression

Learning Methodology

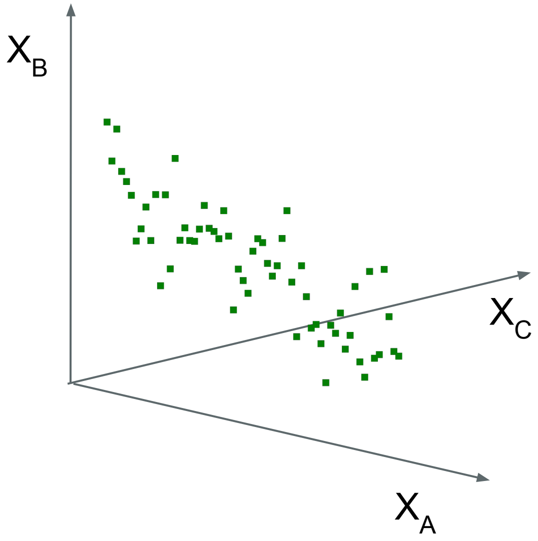
How does the model learn?

Let's step through the algorithm...

Assume a simplified dataset with three features: X_{a}, X_b and X_c.

1. We must first **standardize** the data. Let the new standardized features be X_A, X_B and X_C.

For each datapoint x for each feature X, subtract the mean of X and divide by the standard deviation of X.



Why? Because we will be measuring variation. If we do not standardize data, we can convert one feature from km to cm, and cause that feature's variance to increase. Standardization ensures that our data variance is independent of whatever transformations we might use.

[1] For more on why we need to standardize data, read here.

Learning Methodology

How does the model learn?

Finding Principal Component #1

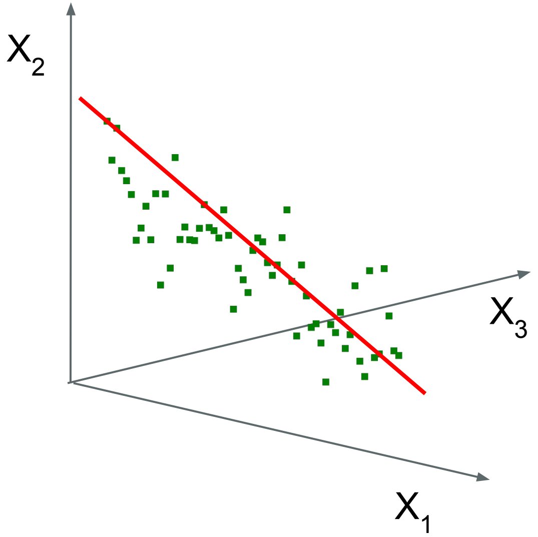
2. To find the first Principal Component, find **the direction of maximum variance**. This is Principal Component #1 (PC1)!

Formally, PC1 is the linear combination of the features…

$$PC1 = c_1(X_A) + c_2(X_B) + c_3(X_C)$$

… that has the **largest variance**.

In our simplified example, we eyeball the direction in which the data varies most.



Finding Principal Component #2

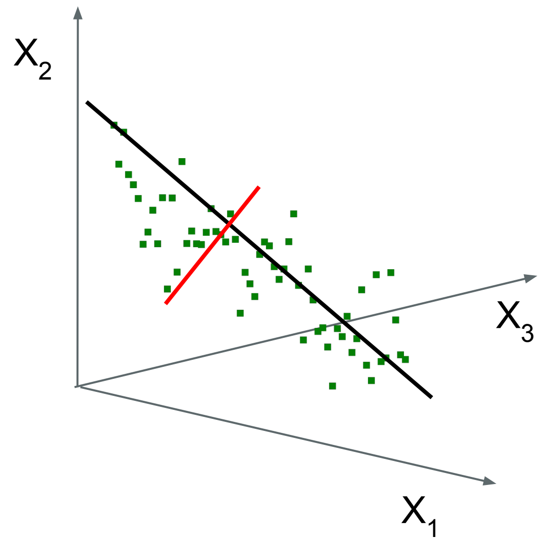
3. To find the second Principal Component, find **the second highest direction of variance**. This is Principal Component #2 (PC2)!

Formally, PC2 is the linear combination of the features…

$$PC2 = c_4(X_A) + c_5(X_B) + c_6(X_C)$$

… that has the second largest variance **AND is uncorrelated with PC1**.

In our simplified example, we can roughly eyeball the direction in which the variance is second highest.



Learning Methodology

How does the model learn?

Finding Principal Component #3

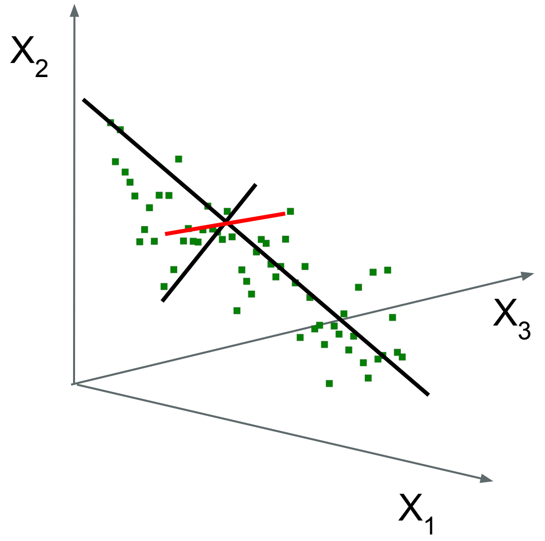
4. To find the third and final Principal Component, find **the third highest direction of variance**. This is Principal Component #3 (PC3)!

Formally, PC3 is the linear combination of the features...

$$PC3 = c_7(X_A) + c_8(X_B) + c_9(X_C)$$

... that has the third largest variance **AND is uncorrelated with both PC1 and PC2**.

In our simplified example, we can roughly eyeball the direction in which the variance is third highest.



Learning Methodology

How does the model learn?

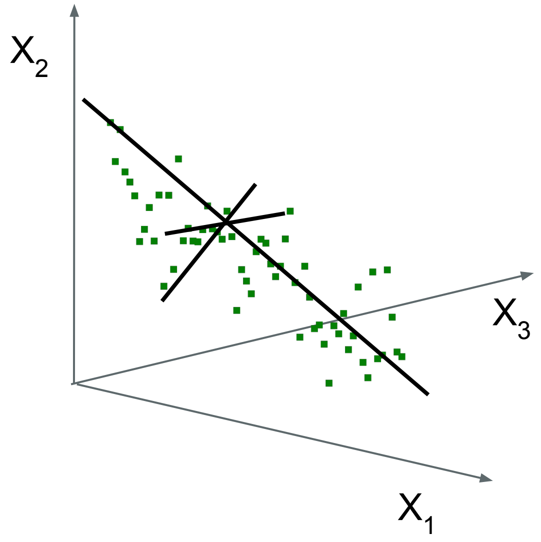
How many principal components can you have? How many do we want?

For the sake of completeness, we calculated all the possible principal components (there can be as many principal components as features.)

However, as our main aim is dimensionality reduction, we will keep the top principal components.

How many should we keep?

Recall our discussion of the elbow plot in k-means clustering. We use similar logic to see how many principal components we want.



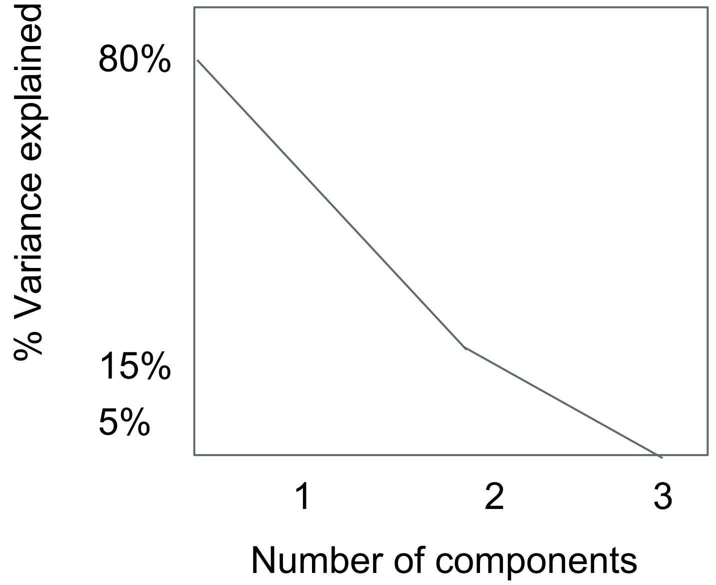
Learning Methodology

How does the model learn?

How many principal components can you have? How many do we want?

If we choose to keep only PC1, we will be able to capture 80% of the original data's information. Pretty good!

If we choose to keep both PC1 and PC2, we will be able to capture 95% of the original data's information. Even better!

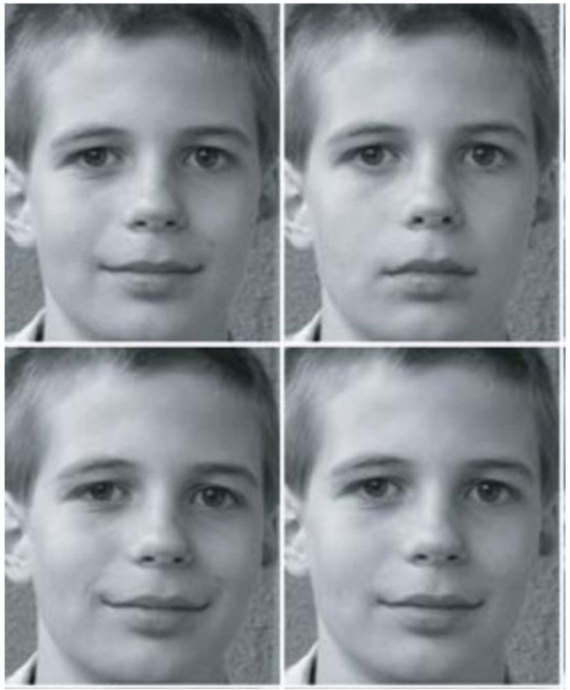


Note that if we chose to keep PC1, PC2, and PC3, we would capture 100% of the original data's information. However, we wouldn't choose to do this, as our main intention was dimension reduction.

For simplicity, our example had a dataset that comprised only 3 features. In reality, you probably wouldn’t apply PCA to such a simple dataset. The true utility of PCA comes from when we apply it to datasets with hundreds or thousands of features.

Consider, for example, **image processing**, where every pixel of an image constitutes a feature.

For each of these 321 × 261 pixel images, every pixel is a feature, resulting in 321 * 261 = **83,781 features** for only 4 observations.

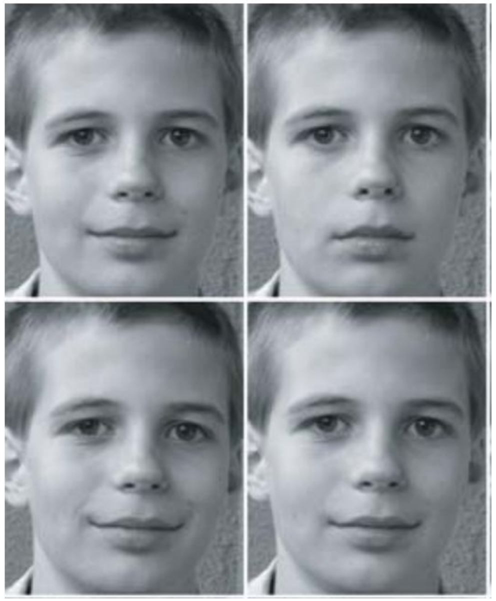


This is a very high-dimensional dataset, but we can use PCA to simplify it.

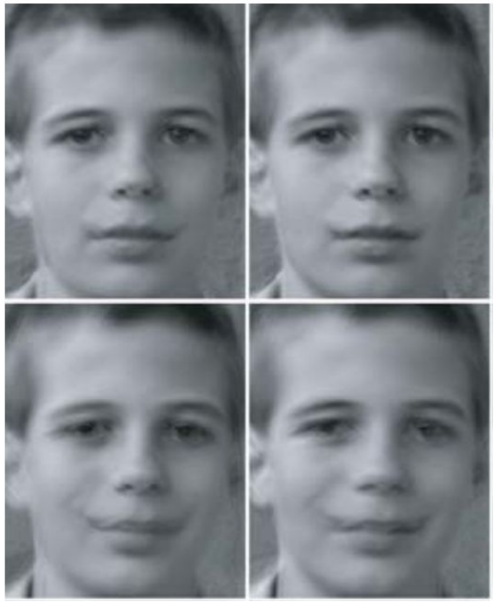
Source: http://people.ciirc.cvut.cz/~hlavac/TeachPresEn/11ImageProc/15PCA.pdf

With PCA, we are able to reduce a dataset with 83,781 features and 4 observations to a dataset with 4 principal components and 4 observations.

Compare the original images...

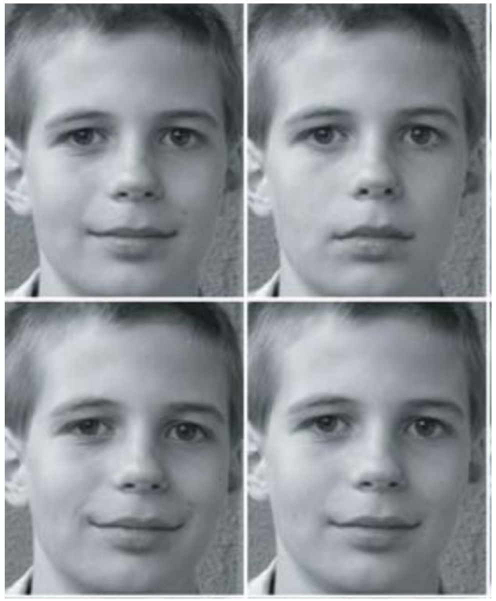


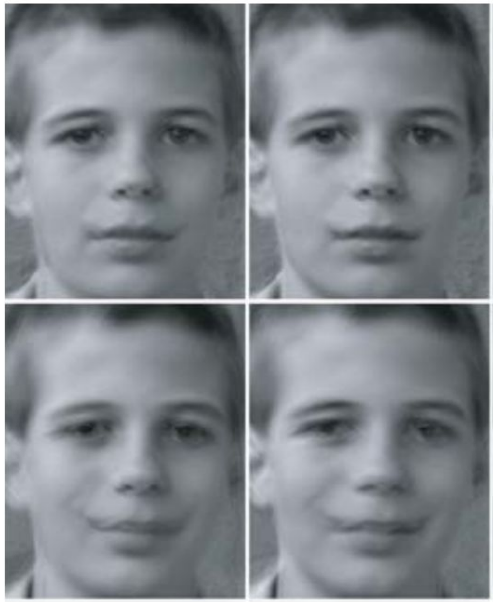
... to images recreated using 4 principal components



PCA is able to retain most of the **information** from the original dataset, while condensing the data quite a bit.

Source: http://people.ciirc.cvut.cz/~hlavac/TeachPresEn/11ImageProc/15PCA.pdf





End of theory

Advanced resources

Want to take this further? Here are some resources we recommend:

- Textbooks
  - Unsupervised learning, Introduction to Statistical Analysis, Chapter 10.2
  - Hierarchical clustering, Introduction to Statistical Analysis, Chapter 10.3.2
- Web resources
  - Clustering optimization using Silhouette plots
  - More applications of k-means: Anomaly detection