https://www.pexels.com/photo/macbook-pro-beside-spiral-notebook-669616/

Member-only story

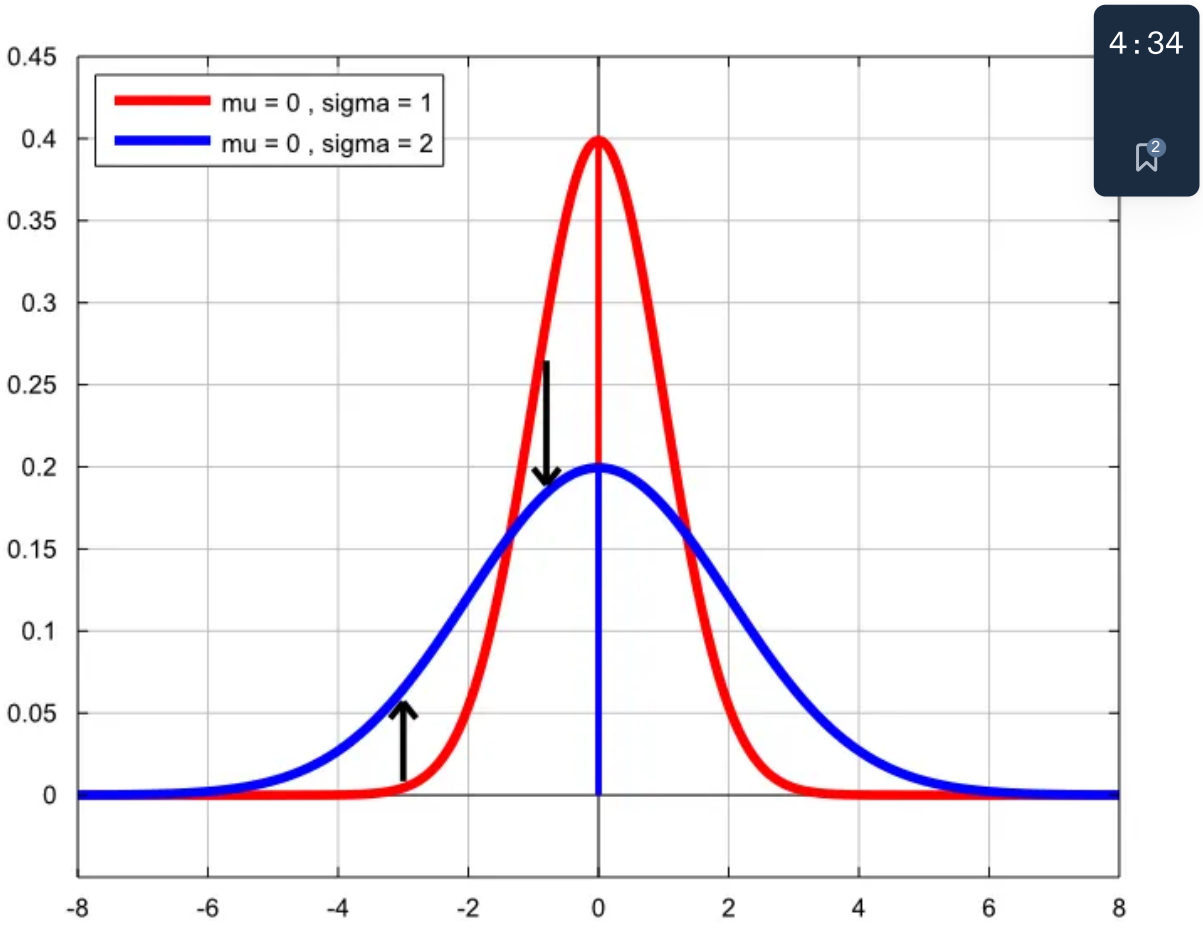
Gaussian Mixture Models Clustering Algorithm Explained

Cory Maklin · Follow
Published in Towards Data Science
8 min read · Jul 15, 2019

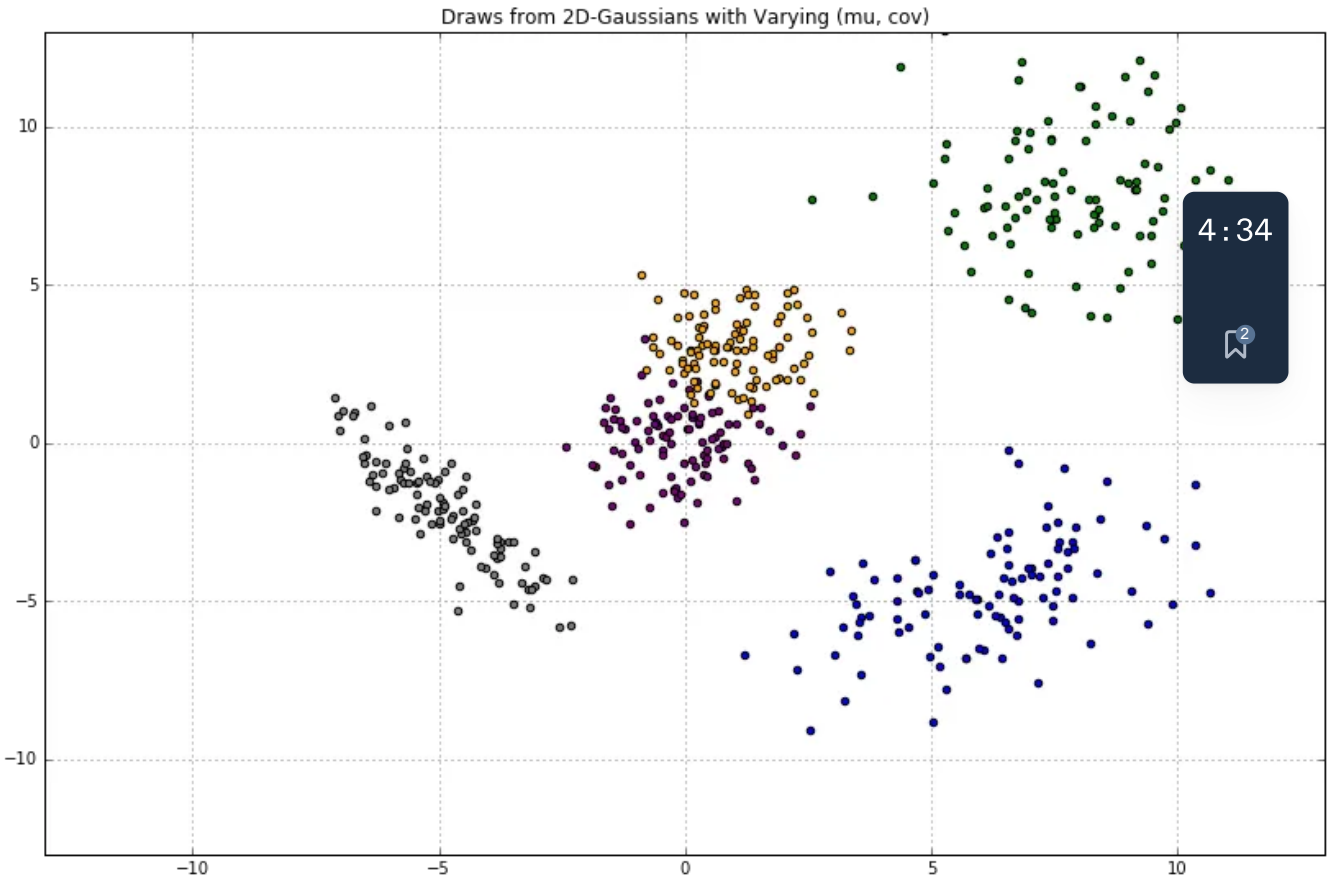
Listen Share More

Gaussian mixture models can be used to cluster unlabeled data in much the same way as k-means. There are, however, a couple of advantages to using Gaussian mixture models over k-means.

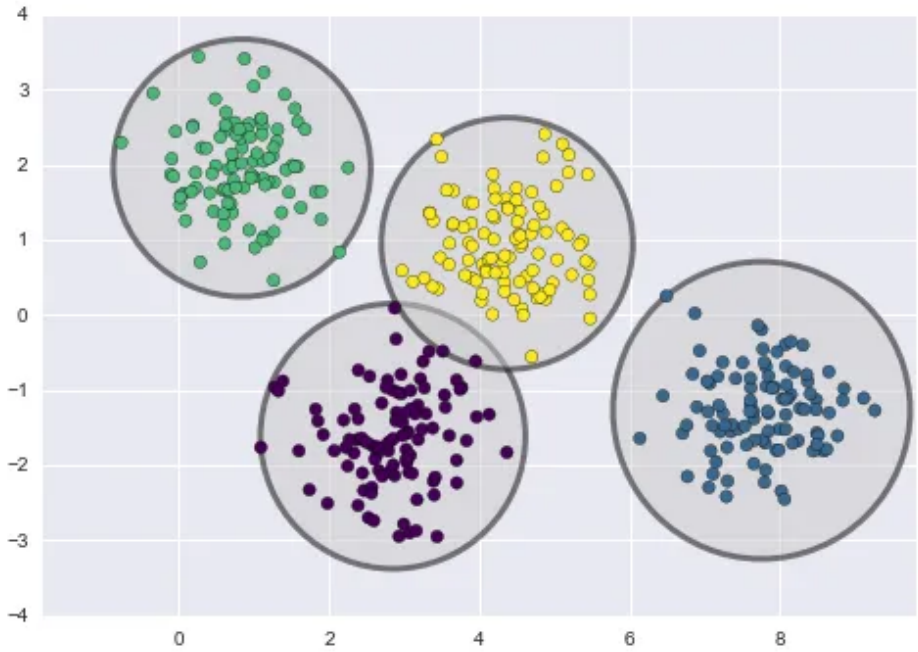
First and foremost, k-means does not account for variance. By variance, we are referring to the width of the bell shape curve.



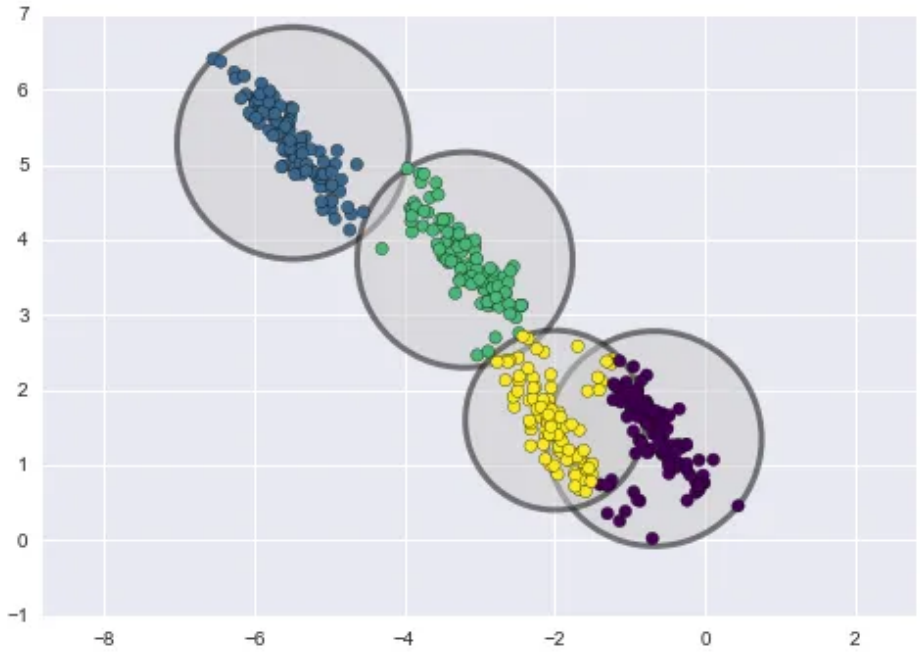
In two dimensions, variance (covariance to be exact) determines the shape of the distribution.



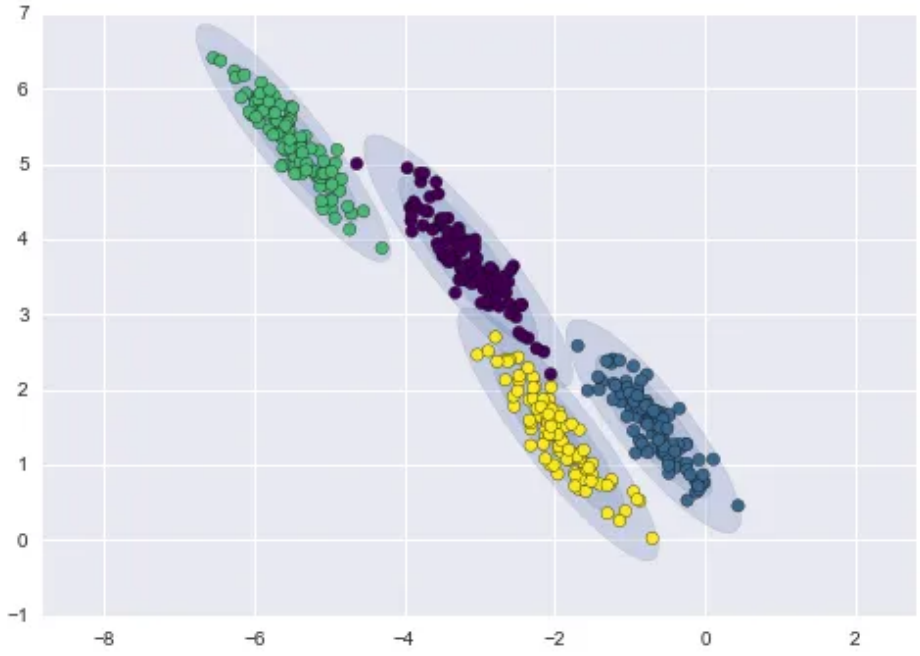
One way to think about the *k*-means model is that it places a circle (or, in higher dimensions, a hyper-sphere) at the center of each cluster, with a radius defined by the most distant point in the cluster.



This works fine for when your data is circular. However, when your data takes on different shape, you end up with something like this.



In contrast, Gaussian mixture models can handle even very oblong clusters.



The second difference between k-means and Gaussian mixture models is that the former performs hard classification whereas the latter performs soft classification. In other words, k-means tells us what data point belong to which cluster but won't provide us with the probabilities that a given data point belongs to each of the possible clusters.

In calling the `predict` function, the model will assign every data point to one of the clusters.

```
gmm.predict(X)
```

```
array([2, 1, 0, 1, 2, 2, 3, 0, 1, 1, 3, 1, 0, 1, 2, 0, 0, 2, 3, 3,
       0, 3, 3, 0, 2, 0, 3, 0, 1, 1, 0, 1, 1, 1, 1, 1, 3, 2, 0, 3,
       3, 3, 1, 3, 1, 2, 3, 2, 1, 2, 2, 3, 1, 3, 1, 2, 1, 0, 1, 3,
       1, 2, 1, 3, 0, 3, 1, 3, 3, 1, 3, 0, 2, 1, 2, 0, 2, 2, 1, 0,
       1, 1, 0, 2, 1, 3, 3, 0, 2, 2, 0, 3, 1, 2, 1, 2, 0, 2, 2, 0, 1, 0,
       3, 3, 2, 1, 2, 0, 1, 2, 2, 0, 3, 2, 3, 2, 2, 2, 2, 3, 2, 3, 1, 3,
       3, 2, 1, 3, 3, 1, 0, 1, 1, 3, 0, 3, 0, 3, 1, 0, 1, 1, 1, 0, 1, 0,
       2, 3, 1, 3, 2, 0, 1, 0, 0, 2, 0, 3, 3, 0, 2, 0, 0, 1, 2, 0, 3, 1,
       2, 2, 0, 3, 2, 0, 3, 3, 0, 0, 0, 0, 2, 1, 0, 3, 0, 0, 3, 3, 3, 0,
       3, 1, 0, 3, 2, 3, 0, 1, 3, 1, 0, 1, 0, 3, 0, 0, 1, 3, 3, 2, 2, 0,
       1, 2, 2, 3, 2, 3, 0, 1, 1, 0, 0, 1, 0, 2, 3, 0, 2, 3, 1, 3, 2, 0,
       2, 1, 1, 1, 1, 3, 3, 1, 0, 3, 2, 0, 3, 3, 3, 2, 2, 1, 0, 0, 3, 2,
       1, 3, 0, 1, 0, 2, 2, 3, 3, 0, 2, 2, 2, 0, 1, 1, 2, 2, 0, 2, 2, 2,
       1, 3, 1, 0, 2, 2, 1, 1, 1, 2, 2, 0, 1, 3])
```

4:34

On the other hand, we can call the `predict_proba` function to return the probabilities that a data point belongs to each of the K clusters.

```
gmm.predict_proba(X)
```



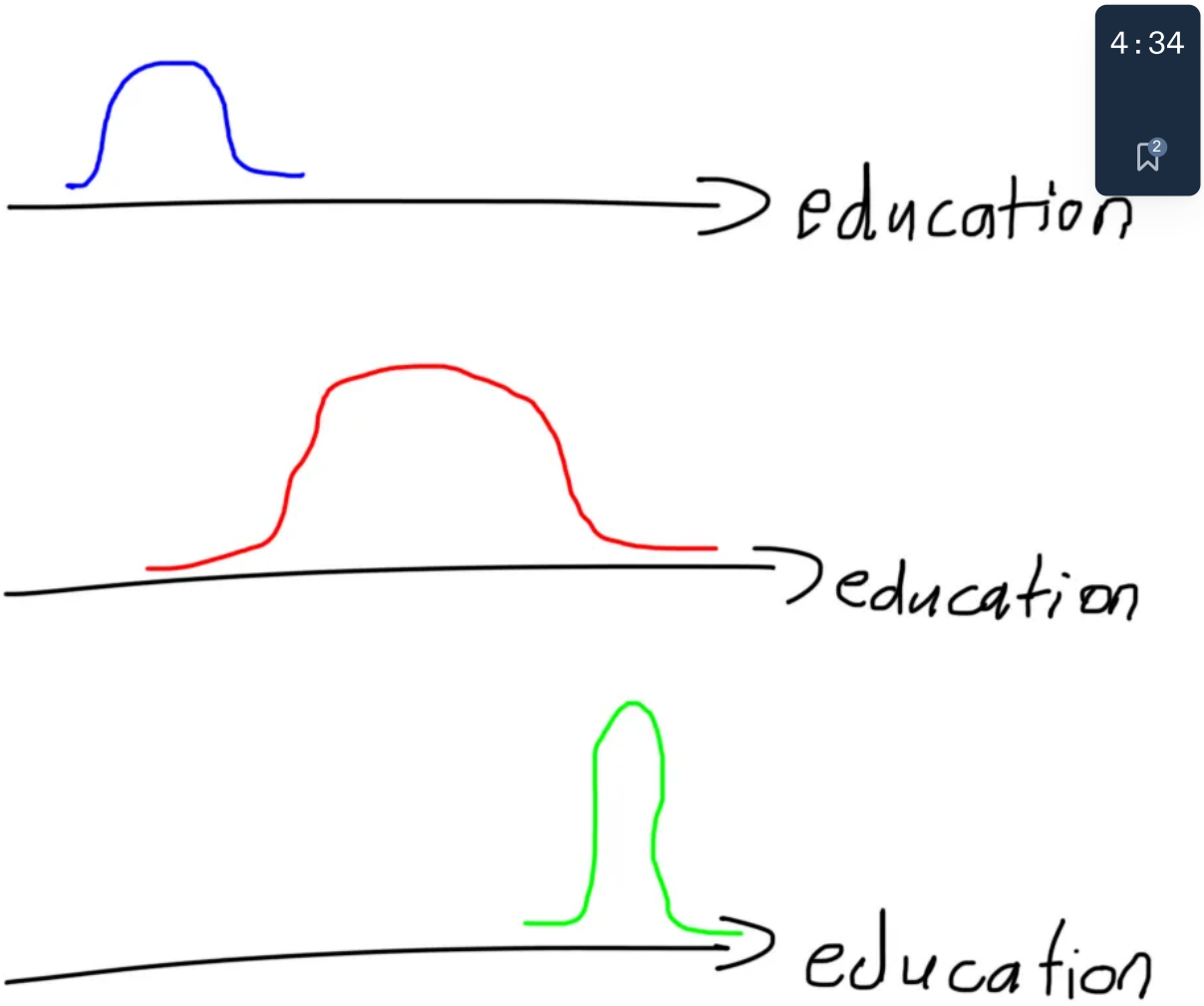
```
[1.000, 0.000, 0.000, 0.000],
...,
[1.000, 0.000, 0.000, 0.000],
[0.000, 1.000, 0.000, 0.000],
[0.000, 0.000, 0.000, 1.000]])
```

Gaussian Mixture Models At A Glance

As the name implies, a Gaussian mixture model involves the mixture (i.e. superposition) of multiple Gaussian distributions. For the sake of explanation, suppose we had three distributions made up of samples from three distinct classes.

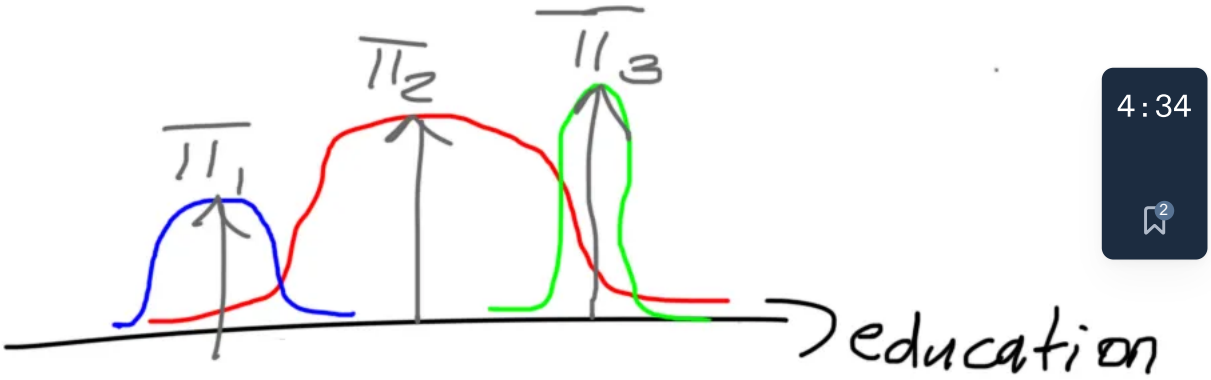
The blue Gaussian represents the level of education of people that make up the lower class. The red Gaussian represents the level of education of people that make up the

middle class, and the green Gaussian represents the level of education of people that make up the upper class.



Not knowing what samples came from which class, our goal will be to use Gaussian Mixture Models to assign the data points to the appropriate cluster.

After training the model, we'd ideally end up with three distributions on the same axis. Then, depending on the level of education of a given sample (where it is located on the axis), we'd place it in one of the three categories.



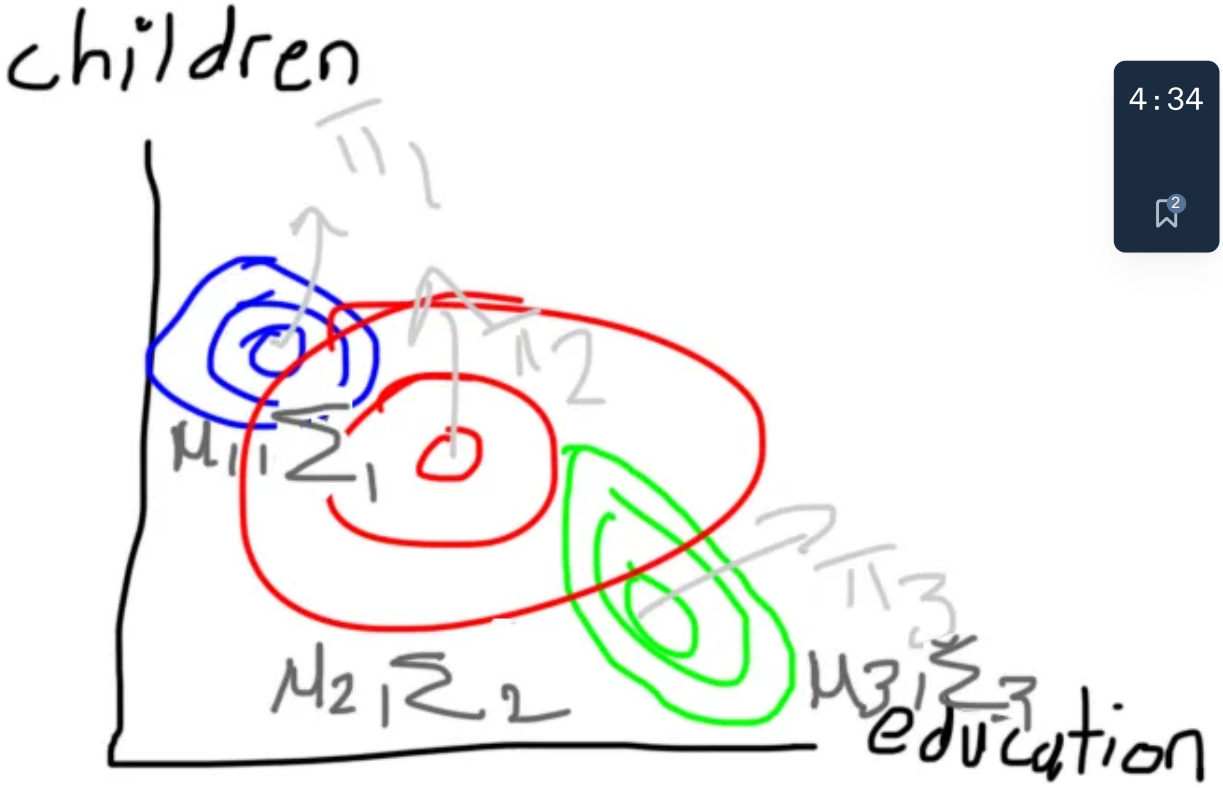
Every distribution is multiplied by a weight π to account for the fact that we do not have an equal number of samples from each category. In other words, we might only have included 1000 people from the upper class and 100,000 people from the middle class.

Since, we're dealing with probabilities, the weights should add to 1, when summed.

$$\pi = [\underset{\pi_1}{0.31}, \underset{\pi_2}{0.48}, \underset{\pi_3}{0.21}]$$

$$\sum_{K=1}^{K} \pi_K = 1$$

If we decided to add another dimension such as the number of children, then, it might look something like this.



Gaussian Mixture Model Algorithm

For those of you who aren't mathematically inclined, I apologize in advance as the next section is quite heavy.

Let's suppose we wanted to know what is the **likelihood that the ith sample came from Gaussian *k*.** We can express this as:

$$p(z_i = k \mid \theta)$$

Where theta represents the mean, covariance and weight for each Gaussian.

$$\theta = \{\mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3, \pi_1, \pi_2, \pi_3\}$$

You may also come across the equation written as π. This is not to be confuse
the weight associated with each Gaussian (confusing I know).

$$p(z_i = k \mid \theta) = \pi_k$$

Next, we express the **likelihood of observing a data point given that it came from Gaussian K** as:

$$p(x_i \mid z_i = k, \mu_k, \Sigma_k)$$

The latter is sometimes written as follows (I believe the ***N*** comes from ***N***ormal Distribution):

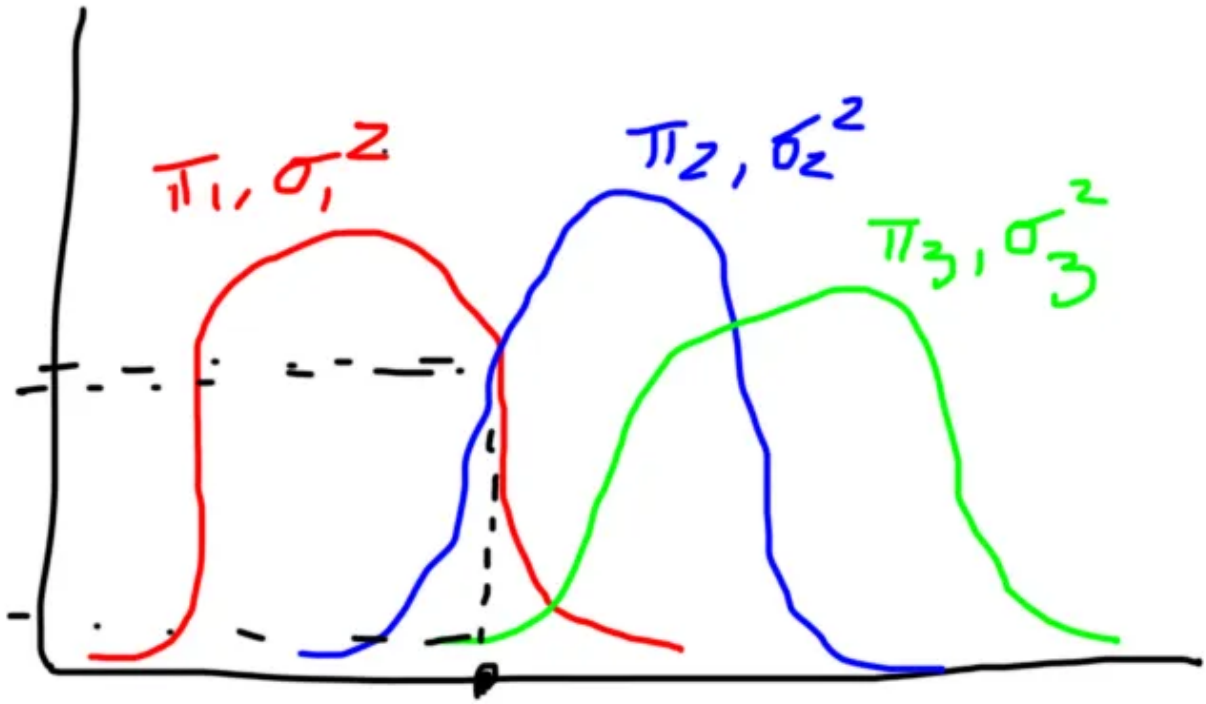
$$p(x_i \mid z_i = k, \mu_k, \Sigma_k) = N(x_i \mid \mu_k, \Sigma_k)$$

Suppose we had a Gaussian distribution where the horizontal axis is the different IQ scores an individual could possibly get, lowest through highest. We can find out how likely it is for an individual to have an IQ of 120 by drawing a vertical line from the position along the x-axis to the curve and then looking at the corresponding value on the y-axis. The value of y at any point is equal to the equation above.

If we'd like to know the likelihood of observing the sample i while taking into account all the different distributions, we simply sum the likelihoods of observing the sample given that it came from each of the possible Gaussian.

$$p(x_i|\theta) = \sum_{k=1}^{3} p(x_i|z_i=k,\theta)p(z_i=k|\theta)$$

Said differently, we take one sample (row) from our dataset, look at a single feature (i.e. level of education), plot its position on the x-axis and sum the corresponding y values (likelihood) for each distribution.



In order to extend this to all samples in our dataset. We assume the likelihood of observing one sample is independent from all the others and then we can simply multiply them.

$$L(\theta) = p(X|\theta) = \prod_{i=1}^{N} \sum_{k=1}^{3} p(x_i|z_i=k,\theta)\, p(z_i=k|\theta)$$

We can rewrite the equation using the nomenclature we saw previously as follows:

$$L(\theta) = p(x|\theta) = \prod_{i=1}^{N} \sum_{k=1}^{3} N(x_i | \mu_k, \Sigma_k)$$

More often than not, we take the log of the likelihood because the multiplication of two numbers inside of a log is equal to the sum of the logs of its constituents, and it's easier to add numbers than to multiply them.

$$\text{Rule 1:} \quad \log_b (M \cdot N) = \log_b M + \log_b N$$

$$\log(p(x|\theta)) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{3} N(x_i | \mu_k, \Sigma_k) \pi_k \right)$$

Expectation Maximization (EM) Algorithm

We have yet to address the fact that we need the parameters of each Gaussian (i.e. variance, mean and weight) in order to cluster our data but we need to know which sample belongs to what Gaussian in order to estimate those very same parameters.

This is where *e*xpectation maximization comes in to play. At a high level, the expectation maximization algorithm can be described as follows:

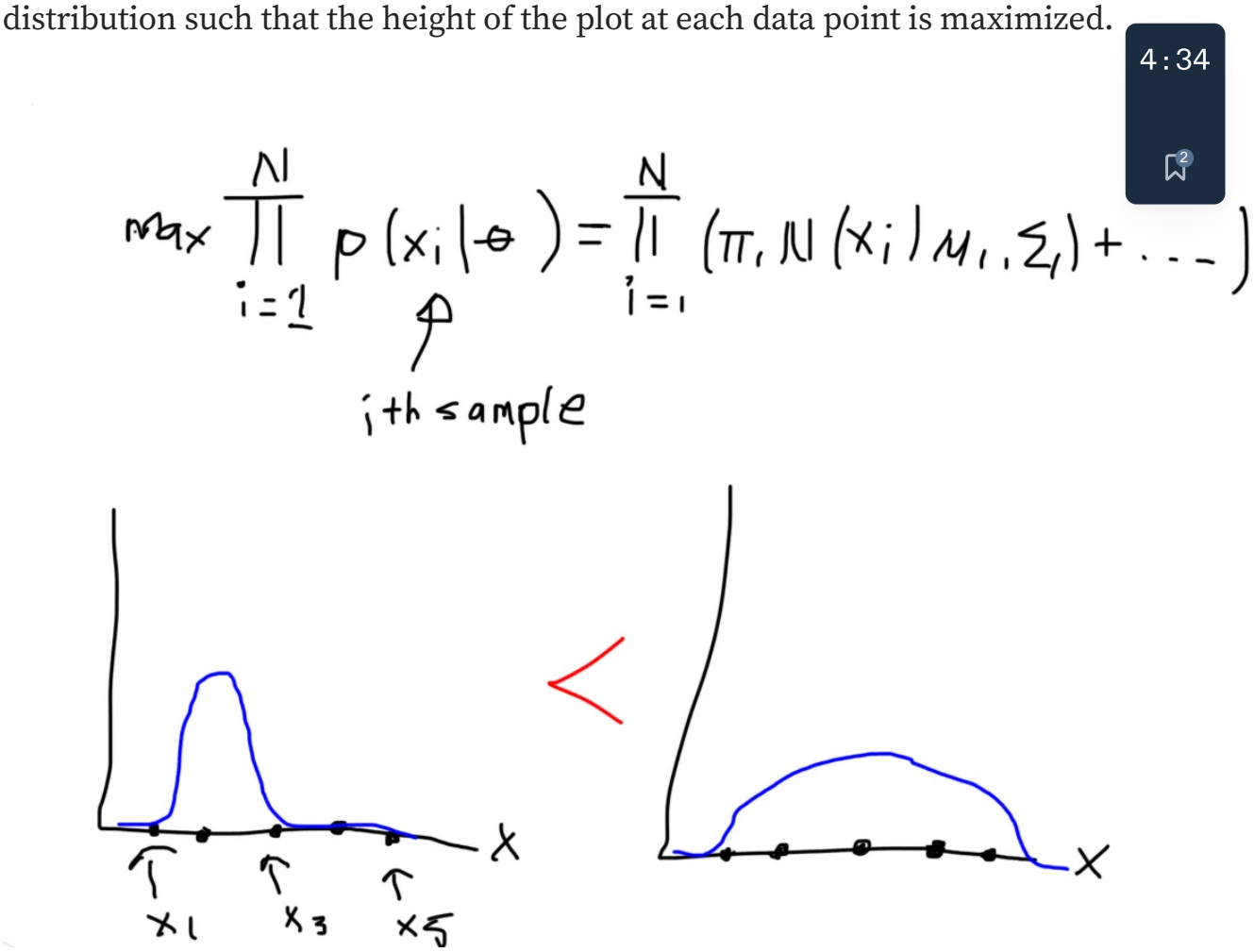
1. Start with random Gaussian parameters (θ)
2. Repeat the following until we converge:

a) **Expectation Step**: Compute p(zi = k | xi, θ). In other words, does sample *i* look like it came from cluster k?

b) **Maximization Step**: Update the Gaussian parameters (θ) to fit points assigned to them.

Maximization Step

In the maximization step, we want to maximize the likelihood that each sample came from the distribution. Recall, that the likelihood is the height of the curve at a point along the x-axis. Therefore, we want to modify the variance and mean of the distribution such that the height of the plot at each data point is maximized.



This brings up the question, “How should we go about selecting the optimal values for the variance and mean.” Using the general form of *expectation maximization*, we can derive a set of equations for the mean, variance and weight.

$$\max_{\theta} \sum_{i=1}^{N} E_{q(z)} \log p(x_i, z_i | \theta)$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{3} q(z_i = k) \log \left(\frac{1}{N} \exp\left(- \frac{(x_i - \mu_k)^2}{2\sigma^2} \right) \right)$$

$$= \sum_{i=1}^{N} \sum_{k=1}^{3} q(z_i = k) \left(\log \frac{\pi_k}{N} - \frac{(x_i - \mu_k)^2}{2\sigma^2} \right)$$

$$\frac{\partial}{\partial \mu_k} = \sum_{i=1}^{N} q(z_i = k) \left(0 - \frac{2(x_i - \mu_k)(-1)}{2\sigma^2} \right)$$

$$0 = \sum_{i=1}^{N} q(z_i = k)(x_i) - \sum_{i=1}^{N} q(z_i = k) \mu_k$$

$$\mu_k = \frac{\sum_{i=1}^{N} q(z_i = k) x_i}{\sum_{i=1}^{N} q(z_i = K)}$$

We can follow the same process to obtain the equations for the covariance and weight.

$$\mu_c = \frac{\sum_{i=1}^{N} p(z_i = k) x_i}{\sum_{i=1}^{N} p(z_i = k)}$$

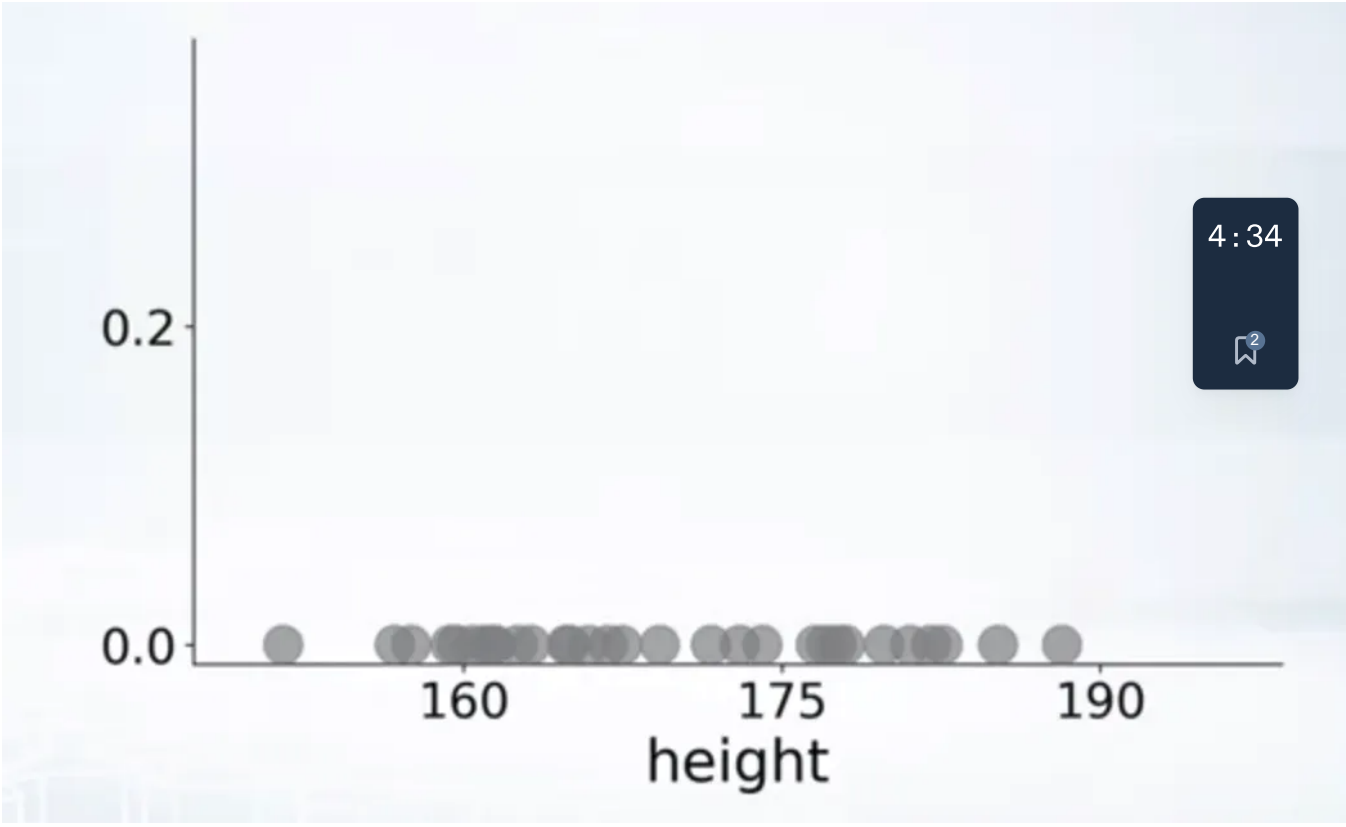
$$\sigma_c^2 = \frac{\sum_{i=1}^{N} (x_i - \mu_c)^2 \, p(z_i = k)}{\sum_{i=1}^{N} p(z_i = k)}$$

4 : 34

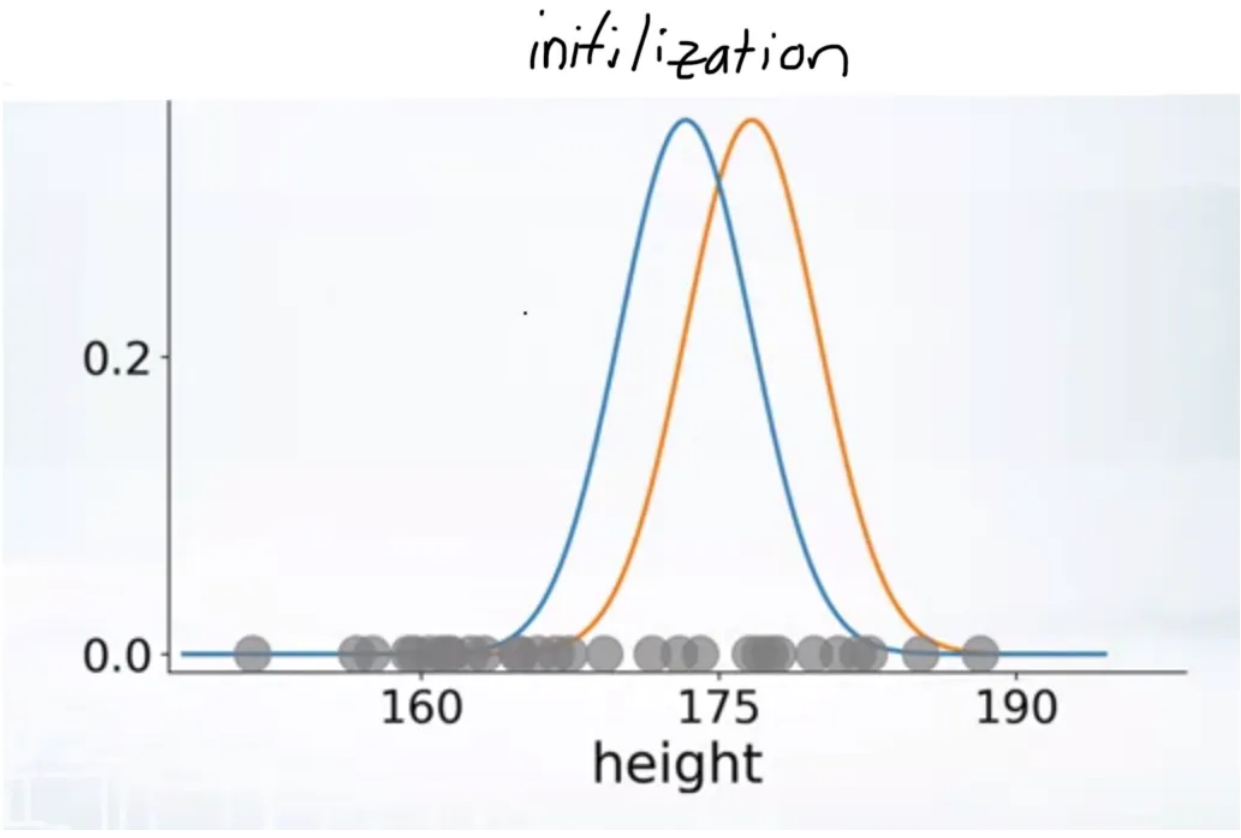
$$\pi_c = \frac{\sum_{i=1}^{N} p(z_i = k)}{N}$$

Once we have the equations, we simply apply them during the maximization step. That is to say, we plugin in the numbers in each equation to determine the best mean, covariance, weight, and then set the Gaussian parameters accordingly.

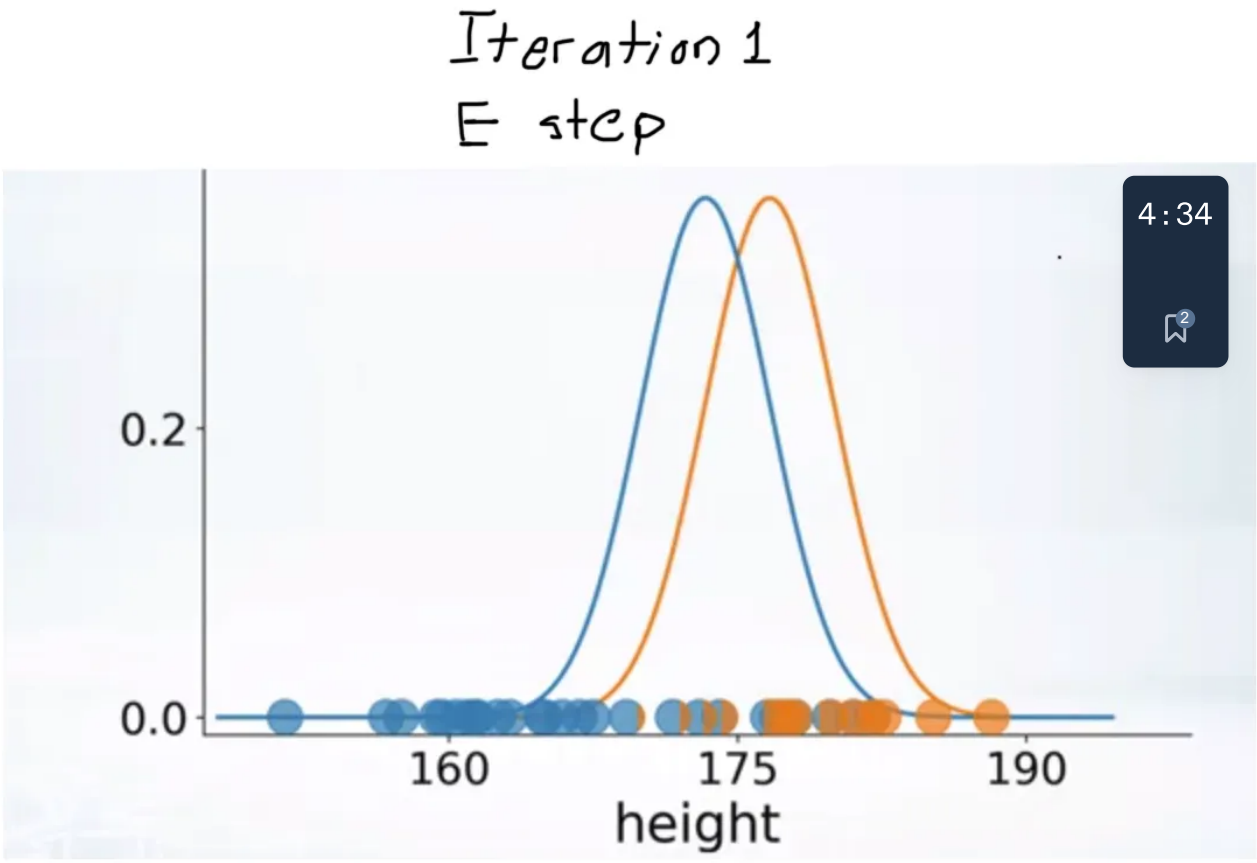
Let's take a look at the math in action. Initially, we don't know what points are associated with which distribution.

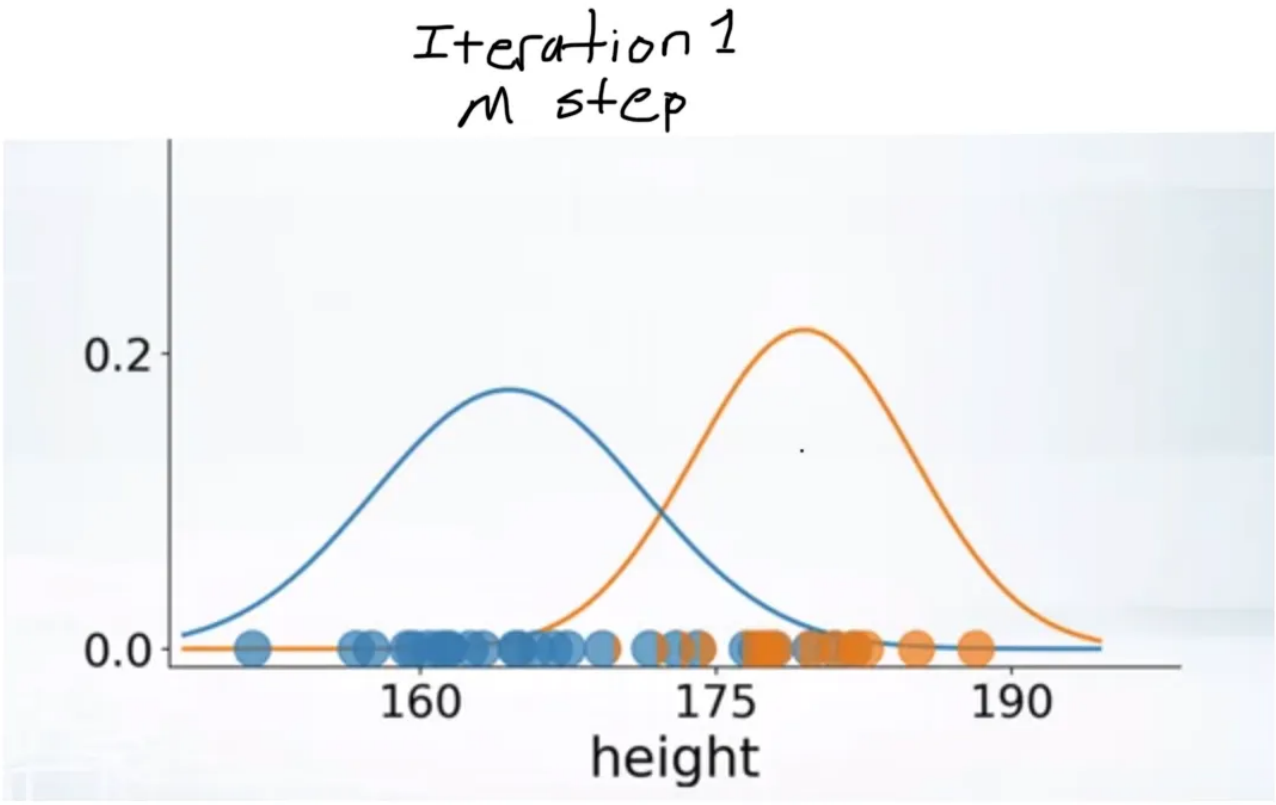


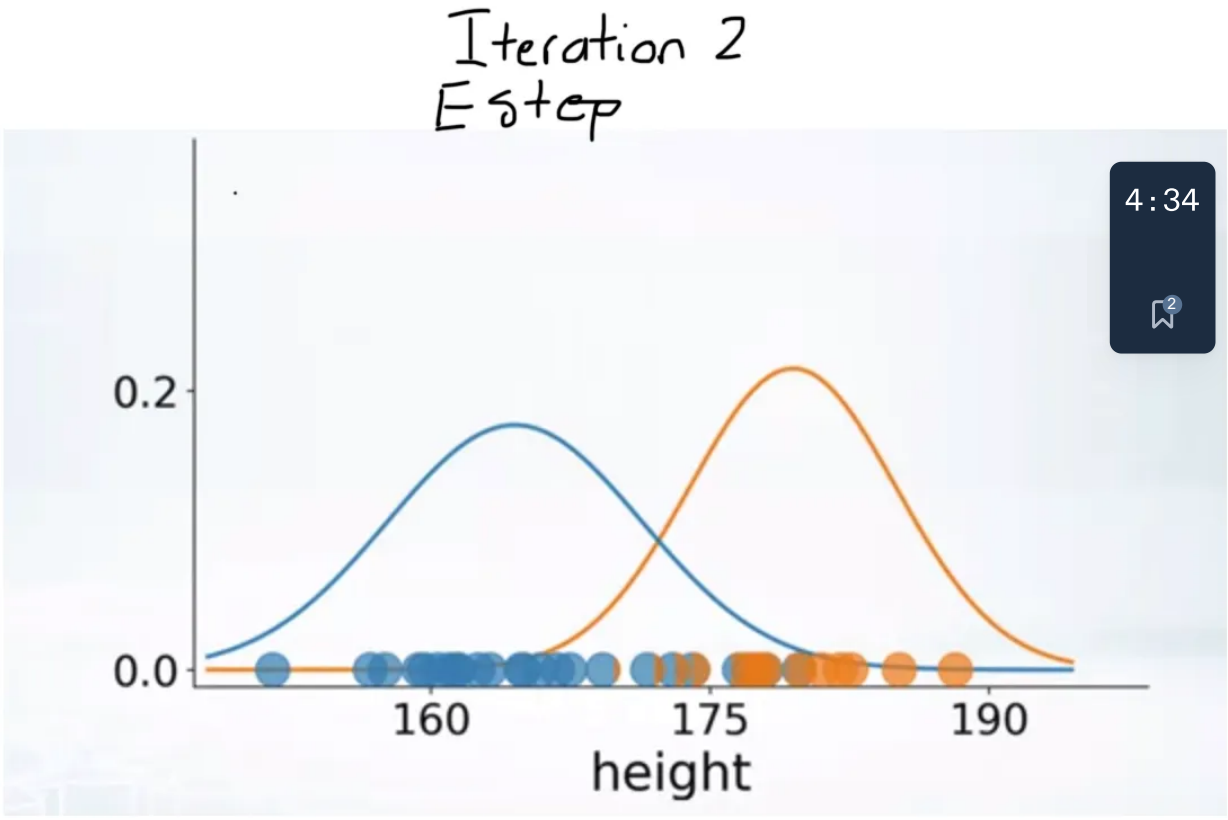
We start off with K Gaussians (in this case, K=2) with random mean, variance and weight.

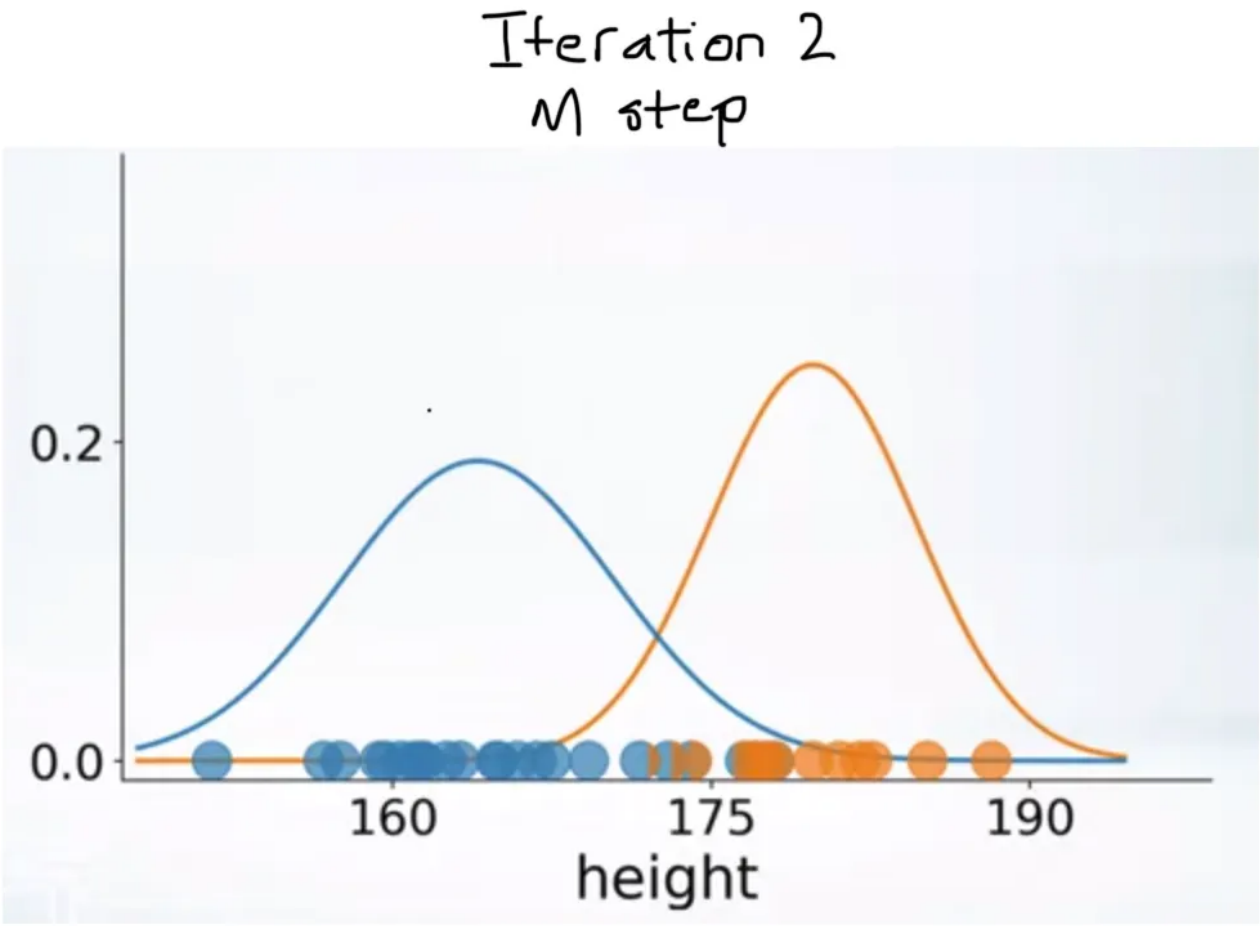


Then, we repeat the expectation and maximization steps until there is little to no change in theta (θ).


Iteration 1
E step
4:34
0.2
0.0
160
175
190
height


Iteration 1
M step
0.2
0.0
160
175
190
height





It's worth noting, that the algorithm is susceptible to local maxima.

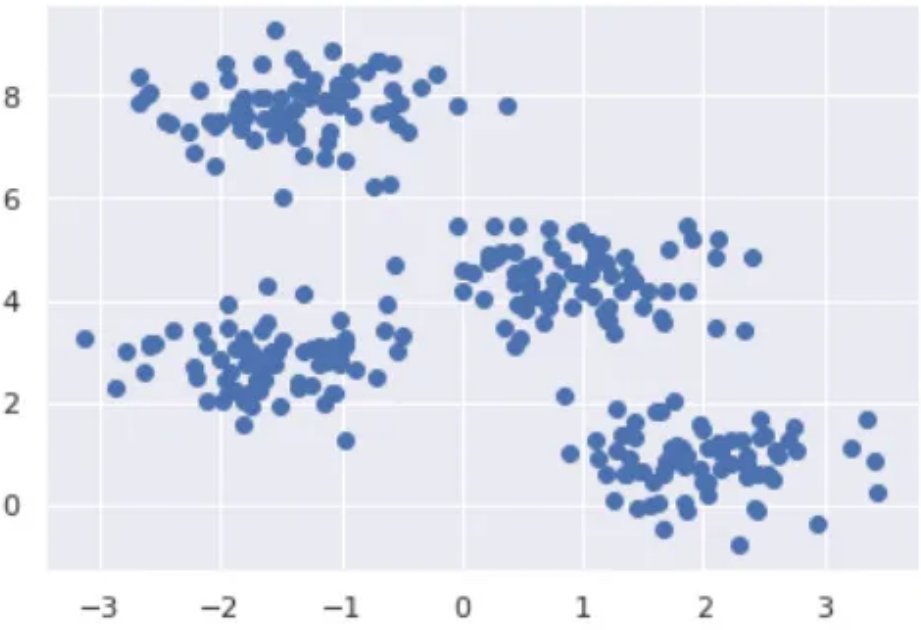
Code

Now, that we have a grasp of how Gaussian mixture models work, let's take a look at how we could go about implementing them. To start, import the following libraries.

```
import numpy as np
from sklearn.datasets.samples_generator import make_blobs
from sklearn.mixture import GaussianMixture
from matplotlib import pyplot as plt
import seaborn as sns
sns.set()
```


We randomly generate 4 clusters.

```
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60,
random_state=0)
plt.scatter(X[:,0], X[:,1])
```

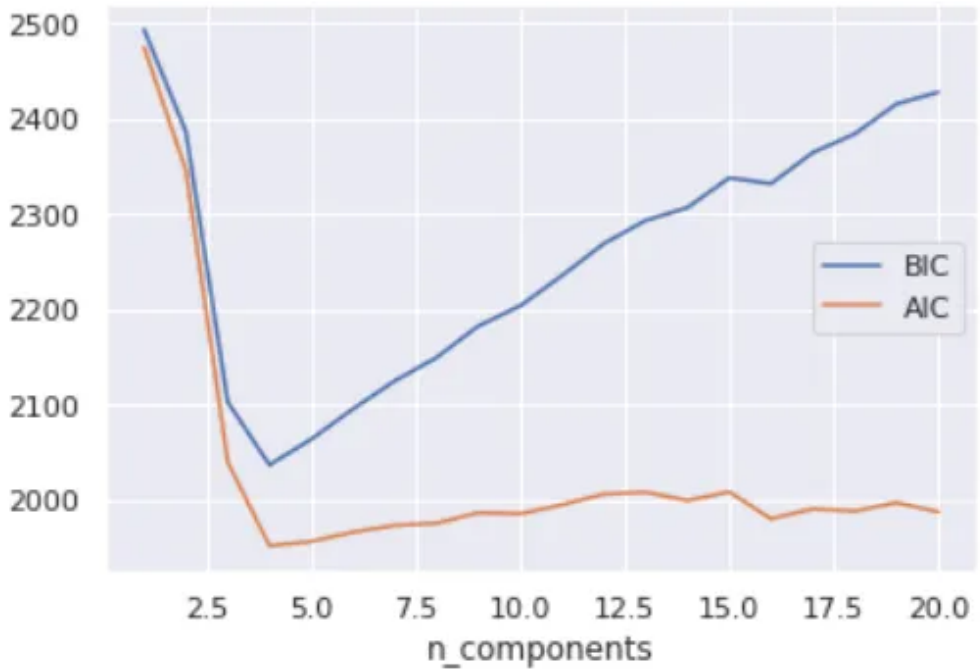


The optimal number of clusters (K) is the value that minimizes the Akaike information criterion (AIC) or the Bayesian information criterion (BIC).

```
n_components = np.arange(1, 21)
models = [GaussianMixture(n, covariance_type='full',
random_state=0).fit(X) for n in n_components]

plt.plot(n_components, [m.bic(X) for m in models], label='BIC')
plt.plot(n_components, [m.aic(X) for m in models], label='AIC')
```

```
plt.legend(loc='best')
plt.xlabel('n_components');
```

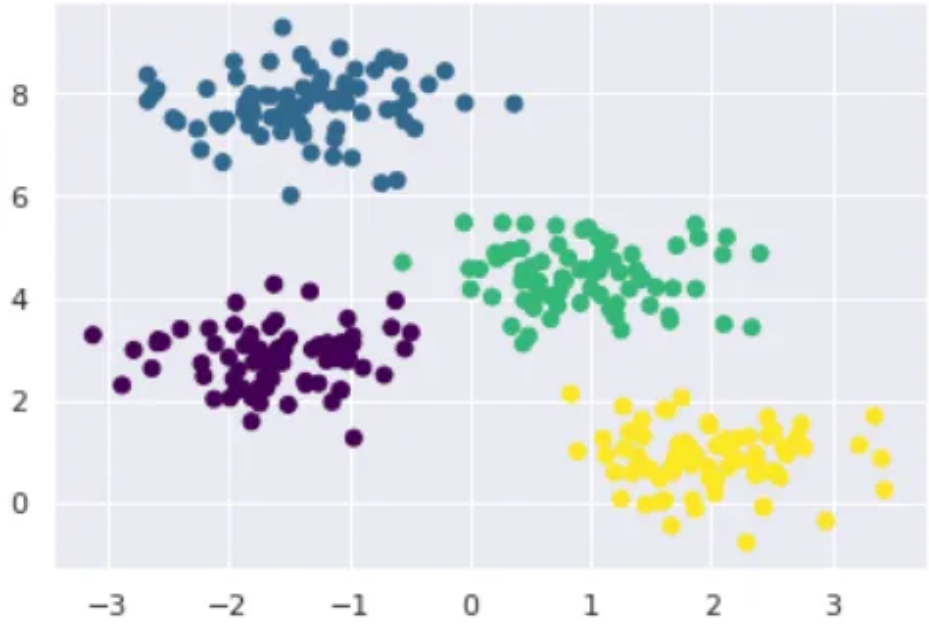


We train our model using the optimal number of clusters (in this case, 4).

```
gmm = GaussianMixture(n_components=4)
gmm.fit(X)
```

We use the `predict` method to obtain a list of points and their respective clusters.

```
labels = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis');
```



4 : 34

Final Thoughts

Unlike k-means, Gaussian mixture models account for variance and returns the probability that a data point belongs to each of the K clusters.

Machine Learning | Data Science | Artificial Intelligence | Programming | Technology

Follow

Written by Cory Maklin

3.8K Followers · Writer for Towards Data Science

Problem Solver • QuantumBlack • Read my articles for free: https://corymaklin.substack.com

More from Cory Maklin and Towards Data Science

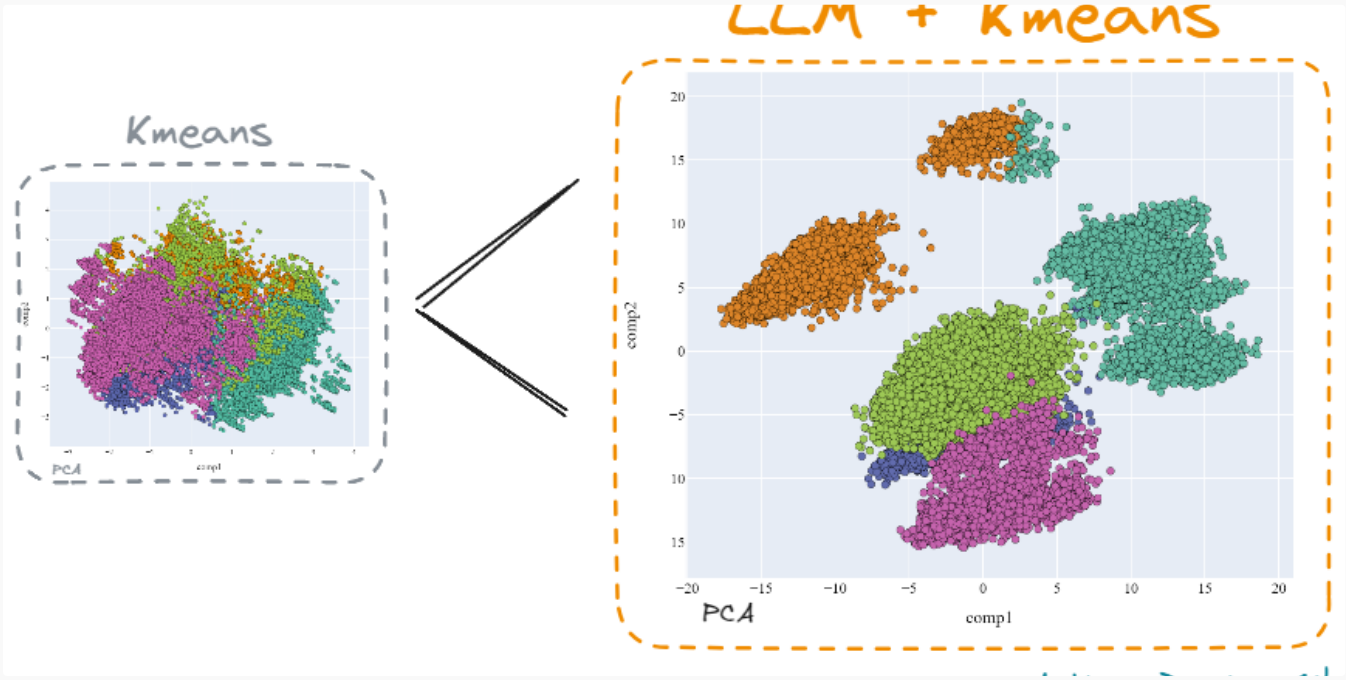
Cory Maklin

The Star Schema Is Dead Long Live Wide Tables

If you do a quick search for books data engineer must read, you'll find two books: 1. Designing Data-Intensive Applications 2. The Data...

· 2 min read · May 29

49 11



Damian Gil in Towards Data Science

Mastering Customer Segmentation with LLM

Unlock advanced customer segmentation techniques using LLMs, and improve your clustering models with advanced techniques

23 min read · Sep 26

2.5K 22

4 : 34

Giuseppe Scalamogna in Towards Data Science

New ChatGPT Prompt Engineering Technique: Program Simulation

A potentially novel technique for turning a ChatGPT prompt into a mini-app.

9 min read · Sep 3

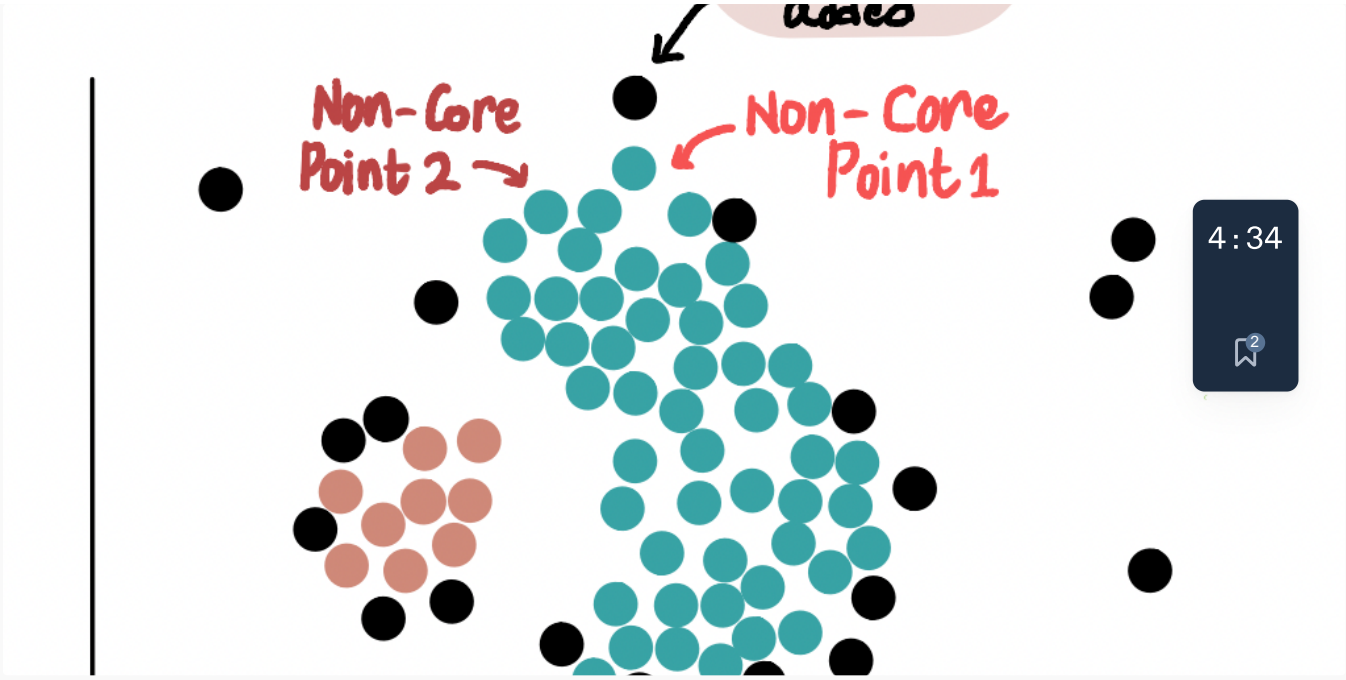
1.7K 17

Cory Maklin in Towards Data Science

ARIMA Model Python Example—Time Series Forecasting

An example of how to perform time series forecasting by building an ARIMA model in Python.

· 9 min read · May 25, 2019



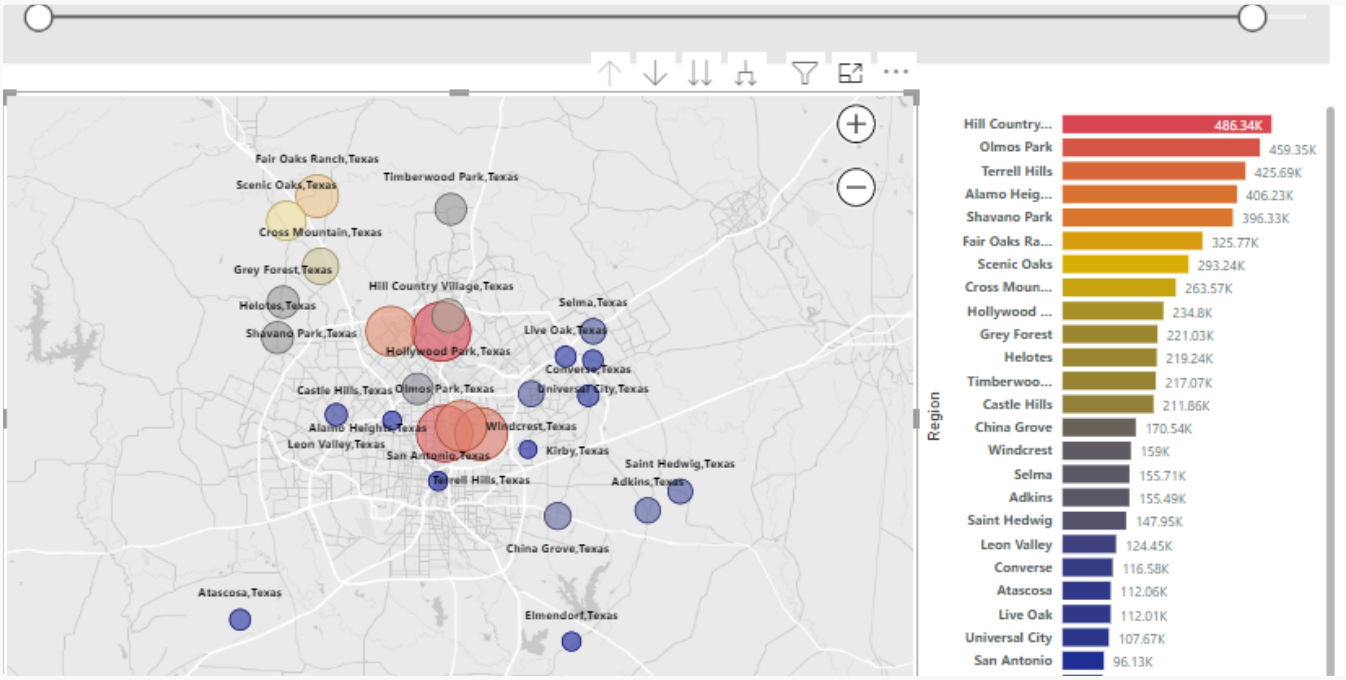
Shreya Rao in Towards Data Science

DBSCAN Clustering: Break It Down For Me

An accessible introduction to a powerful algorithm

6 min read · Nov 29, 2022

405 4



Rayan Yassminh www.linkedin.com/in/rayan-yassminh

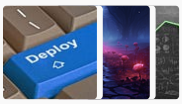
Clustering Time Series : Analysis of Housing Prices in Bexar County, TX from 1996 to 2020

Clustering time series is a powerful technique utilized to group similar time series data based on their underlying patterns and...

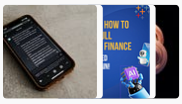
9 min read · May 5

1

Lists



Predictive Modeling w/ Python
20 stories · 456 saves



ChatGPT prompts
24 stories · 472 saves

ChatGPT
21 stories · 183 saves

AI Regulation
6 stories · 142 saves

YashwanthReddyGoduguchintha

Hierarchical clustering

Hierarchical clustering:-

3 min read · Apr 8

1

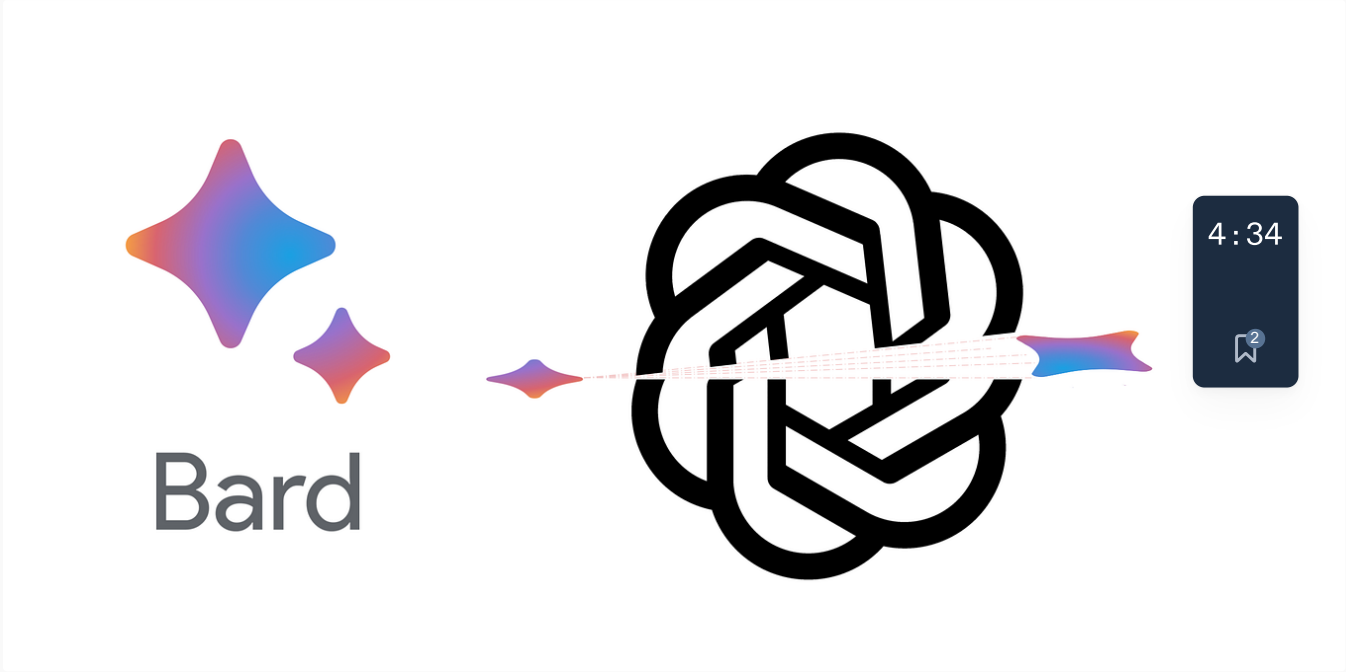
Faridah Yusuf

Difference between K means and Hierarchical Clustering

In this article we are going to be looking at the differences between k-means clustering and hierarchical clustering.

2 min read · Jun 1

10



AL Anany

The ChatGPT Hype Is Over—Now Watch How Google Will Kill ChatGPT.

It never happens instantly. The business game is longer than you know.

· 6 min read · Sep 1

11.8K 370

Yannis Poulakis

Is Silhouette the Right Clustering Evaluation Metric for You?

While popular, simple experiments prove that Silhouette index may not be the index to evaluate every clustering solution.

4 min read · Jul 4

17

4 : 34