

The Pattern Finder

Finding hidden patterns — with no labels at all

Day 03 · Week 1

Unlocks  Pattern Finder

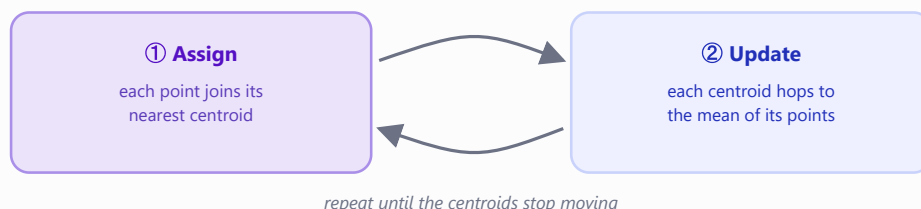
1 Mission briefing

A crate lands in the Lab with no label — thousands of measurements and **nobody** to say what they mean. No answer key, no teacher. Find the hidden structure yourself. *Previously:* you trained the Prediction Machine to name species from labeled examples. Today the labels are gone. This is **unsupervised learning**, and by the end of the day it powers your Studio's newest module.

- ☐ Tell **supervised** (answer key) from **unsupervised** (none) apart
- ☐ Watch it **rediscover the penguin species** with zero labels
- ☐ Build a photo **color-compressor** with k-means
- ☐ Run **k-means** to discover groups no one named
- ☐ Use the **elbow method** to guess how many groups exist
- ☐ Squash 4-D and 64-D data into pictures with **PCA & t-SNE**

2 Field guide the concepts

Unsupervised learning finds structure with no labels to copy. The workhorse is **k-means**. Picture three pizza shops opening in a new city with no map of customers. They play a two-move game, over and over: every customer orders from the **nearest** shop, then each shop moves to the **center** of the customers it just served. Repeat and the shops drift into the middle of three natural neighborhoods. Swap "shop" for **centroid** and "customer" for data point and that's the whole algorithm — *k* is how many shops you place.



k-means alternates **assign** ↔ **update** until it settles. Standardize features first so neither dominates.

Seeing high dimensions. A color is just a 3-D point (its R, G, B) — so k-means on a photo's pixels finds its palette and repaints it with a few colors (**compression**). And when data has 4 or 64 dimensions, **PCA** casts the maximum-variance 2-D "shadow", while **t-SNE** untangles curvier structure into a map you can actually look at.

A precise reading

K-means minimizes squared distances to centroids and may settle at a local solution. Cluster numbers are arbitrary, and clusters need not equal real species. The elbow is a heuristic. PCA preserves variance, not all information; t-SNE emphasizes local neighborhoods, so distances between islands and island sizes are not reliable measures of the original data.

3 Lab missions check each off in the notebook

- 1 **Let scikit-learn do the clustering** ☐

`KMeans(n_clusters=3, n_init=10)` on two standardized measurements.
Expect: three clean blobs with red-X centroids.
- 2 **The reveal: did it find the species?** ☐

Cross-tabulate the clusters against the *true* species.
Expect: each cluster lines up mostly with one species — with zero labels.
- 3 **Find the elbow** ☐

Record `.inertia_` for $k = 1 \dots 8$ and plot it.
Expect: the curve drops steeply, then bends around $k = 3$.
- 4 **Compress the photo to 8 colors** ☐

Fit k-means on a pixel sample, then `.predict` every pixel.
Expect: the repaint still looks like the photo — in only 8 colors.
- 5 **How few colors can you get away with?** ☐

Repaint at $k = 3, 8, 16$ and compare.
Expect: $k = 3$ loses subtle shades; 8 vs 16 barely differ.
- 6 **Flatten 4-D penguins onto a 2-D page** ☐

`PCA(n_components=2)` on all four measurements.
Expect: the 2-D shadow keeps well over 80% of the variance.
- 7 **Draw the map** ☐

Scatter the t-SNE embedding of 8×8 digit images, colored by digit.
Expect: roughly ten islands — one per digit, unprompted.

4 Cheat sheet clustering & dimensionality reduction

<code>KMeans(n_clusters=k, n_init=10).fit(X)</code>	Run assign/update until centroids settle
<code>km.fit_predict(X)</code>	Fit and return a cluster number per point
<code>km.labels_ · km.cluster_centers_</code>	Each point's cluster · the centroid positions
<code>km.inertia_</code>	Tightness of clusters (lower = tighter) — for the elbow
<code>km.predict(pixels)</code>	Label new points using a fitted model
<code>pd.crosstab(clusters, truth)</code>	Compare clusters to real labels in a grid
<code>StandardScaler().fit_transform(X)</code>	Rescale so no feature dominates the distance
<code>PCA(n_components=2).fit_transform(X)</code>	Best 2-D shadow; <code>.explained_variance_ratio_</code>
<code>TSNE(n_components=2, perplexity=30)</code>	Untangle high-D data into a 2-D map

5 Unlock your Studio module

Pattern Finder

Put your name on your work and pick a default number of colors; the unlock cell saves a small config token. It must write exactly:

```
ai_studio/models/pattern_finder.json
```

The clustering runs *live* in the app — upload any photo and it finds the dominant colors:

```
$ python ai_studio/app.py
```

6 Mini-glossary

unsupervised learning

Finding structure in data that has no labels to check against.

centroid

A cluster's center — the mean of the points assigned to it.

elbow method

Pick k where the inertia curve stops dropping sharply — its "bend".

t-SNE

A nonlinear map that pulls similar points together to reveal structure.

cluster

A group of points that resemble each other more than the rest.

inertia

How tightly points hug their centroid; smaller means tighter clusters.

PCA

Dimensionality reduction that keeps the most-informative straight-line "shadow".

7 Show & tell

1. k-means never saw a label — how well did its **3 clusters** match the real species? Show your crosstab.
2. Show your photo at **3, 8, and 16 colors**. Where's the sweet spot, and what falls apart at $k = 3$?
3. Your elbow bent at **$k = 3$** . If a mystery dataset bent at $k = 5$, what would that tell you?