Machine Learning for Smarter Innovation

Week 1: Foundations & Clustering

Discovering Innovation Patterns with ML

BSc Course in AI-Enhanced Innovation

Prerequisites & What You Need

Setting You Up for Success

What You Need to Know

- Basic Python (variables, loops, functions)
- High school math (averages, distances)
- How to use Jupyter notebooks
- Basic data concepts (tables, rows, columns)

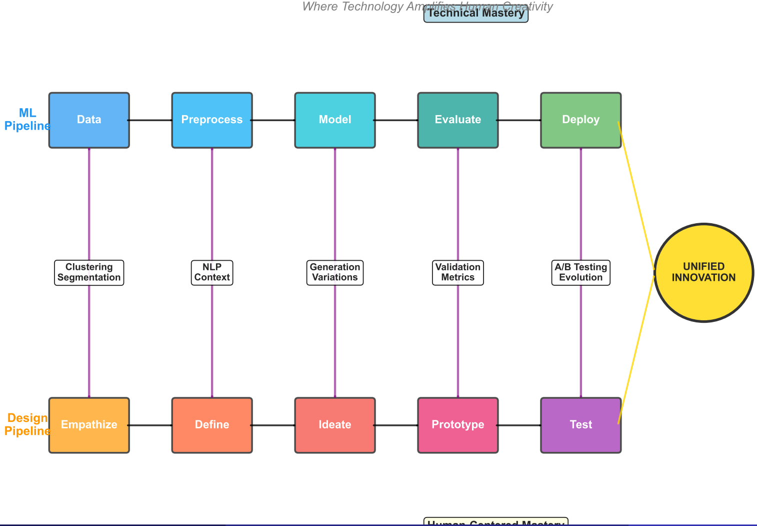
What We'll Provide

- All code templates
- Step-by-step instructions
- Visual explanations
- Practice datasets

No machine learning experience required!

The Unified Innovation Pipeline

Where Technology Amplifies Human Creativity



PART 1

Foundation & Context

What we'll explore:

- Why traditional design hits limits
- How ML amplifies human insight
- The dual pipeline approach
- Your learning journey ahead

Setting the stage for transformation

By the end of Part 1, you will be able to:

- **Understand** the limitations of traditional innovation approaches
- **Recognize** how ML enhances human creativity
- **Explain** the dual pipeline methodology
- **Navigate** the 10-week learning journey
- **Identify** Week 1's role in the overall course

Success Criteria

- Can articulate 3+ traditional design limitations
- Can describe ML's value proposition
- Can map ML pipeline to design pipeline
- Understand clustering's role in innovation

PART 1

Foundation & Context

Understanding the Innovation Challenge

The Innovation Challenge

Why Traditional Design Needs AI Enhancement

Traditional Design Limits

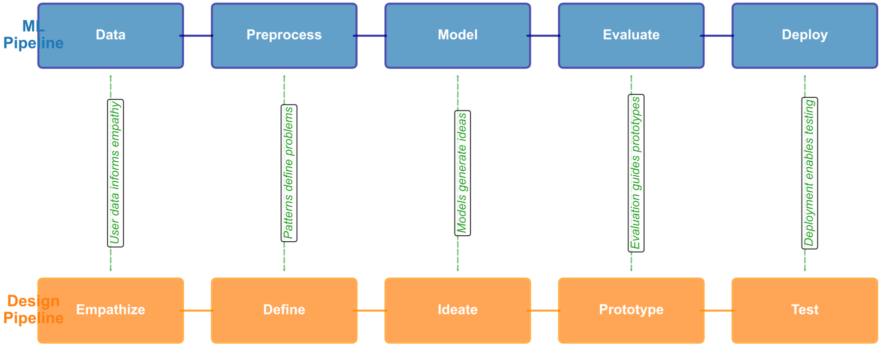
- **Scale**: Can analyze 50 ideas, not 50,000
- **Speed**: Months for insights
- **Bias**: Designer's perspective dominates
- **Patterns**: Miss hidden connections
- **Iteration**: Slow feedback loops

AI-Enhanced Innovation

- **Scale**: Analyze millions of data points
- **Speed**: Real-time insights
- **Objectivity**: Data-driven discovery
- **Patterns**: Find non-obvious relationships
- **Iteration**: Continuous learning

The Promise: 100x more insights, 10x faster innovation

The Convergence: ML Meets Design Thinking



Each connection point creates innovation opportunities

The Dual Pipeline (Continued)

Understanding Both Worlds

ML Pipeline

Data → Preprocess → Model → Evaluate → Deploy

- Collect innovation data
- Clean and transform
- Train algorithms
- Validate accuracy
- Scale to production

Design Pipeline

Empathize → Define → Ideate → Prototype → Test

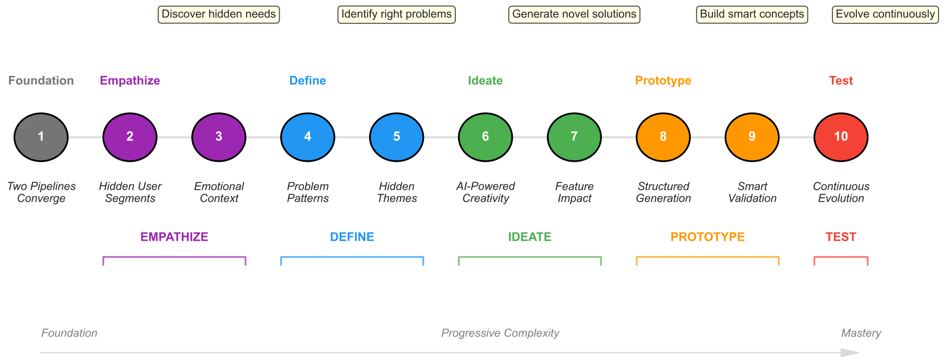
- Understand innovation needs
- Frame problems
- Generate solutions
- Build concepts
- Validate innovation impact

Integration = Innovation at Scale

Your Innovation Journey

10 Weeks to Understanding AI-Powered Design

10-Week Innovation Journey



Your Innovation Journey (Continued)

What You'll Learn in Each Stage

Stage	Weeks	Innovation Unlocked
Discover	1-2	Find hidden innovation opportunities
Define	3-4	Identify the right problems to solve
Ideate	5-6	Generate novel solutions with AI
Prototype	7-8	Build smart, adaptive concepts
Test	9-10	Evolve through continuous learning

This Week: Clustering for Innovation Pattern Discovery

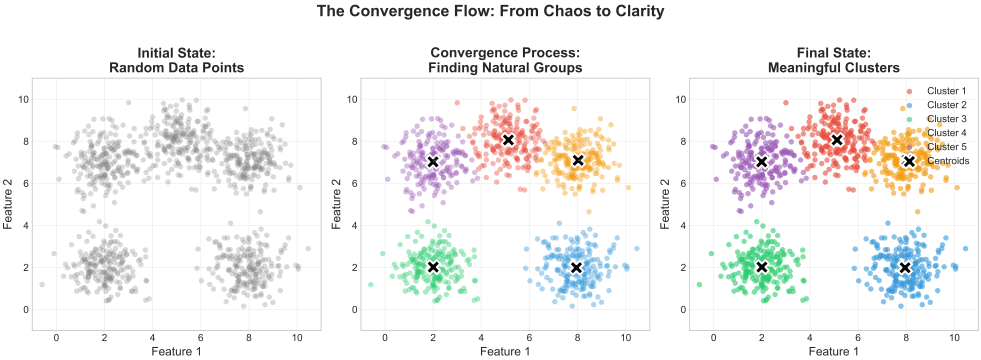
What We'll Learn:

- How clustering reveals innovation categories
- K-means algorithm fundamentals
- Finding the optimal number of clusters
- Quality metrics for validation
- Advanced clustering techniques

Design Applications:

- Create innovation archetypes
- Map innovation evolution paths
- Identify opportunities systematically
- Prioritize design efforts
- Scale analysis to thousands of ideas

Goal: Transform scattered ideas into innovation patterns



The Convergence Flow: Order from Chaos

Watch 5000 innovation ideas self-organize into meaningful patterns

Check Your Understanding - Part 1

Quick Knowledge Check

True or False?

1. Clustering requires labeled data (F)
2. ML can process more data than humans (T)
3. Design thinking has 5 stages (T)
4. Clustering finds hidden patterns (T)

Can You Explain?

- What is the dual pipeline approach?
- Why combine ML with design thinking?
- What problem does clustering solve?

Ready for Part 2? Let's dive into the technical details!

We've seen the challenge:
Thousands of innovation ideas with hidden connections

Traditional approach:
Manual segmentation based on demographics

The ML solution:
Let the data reveal its own natural groups

Enter: Clustering Algorithms

PART 2

Technical Core

What we'll learn:

- K-means clustering algorithm
- Finding optimal K with elbow method
- Distance metrics and quality measures
- Advanced techniques (DBSCAN, Hierarchical)
- Feature importance analysis

Learning the basics step by step

Part 2: Learning Objectives

Technical Skills You'll Develop

By the end of Part 2, you will understand:

- **How** K-means clustering works
- **What** the elbow method shows us
- **Why** we measure distances
- **How to check** if clusters are good
- **Differences** between algorithms
- **When to use** each method

Practical Skills

- Use K-means step by step
- Understand quality scores
- Pick the right algorithm
- Adjust settings properly
- Work with different patterns
- Prepare data for analysis

PART 2

Technical Core

Machine Learning Algorithms & Implementation

The Innovation Classification Problem

5000 Ideas - How Do They Connect?

The Pain

Current Reality:

- One-size-fits-all solutions
- Generic innovation categories
- Missed opportunities
- Unhappy edge cases

The Cost:

- Most innovations get misclassified
- Features with low adoption rates
- Inefficient resource allocation

The Question

What if we could...

- Find natural innovation clusters?
- Discover innovation patterns?
- Innovate at scale?
- Identify opportunity gaps?

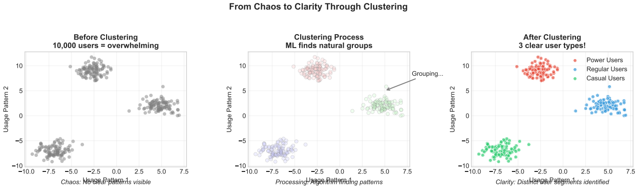
We can!

Solution: Clustering

Clustering Finds:

- Natural groupings (like sorting laundry by color)
- Similar approaches (things that work the same way)
- Hidden patterns (connections you didn't see before)
- Innovation relationships (which ideas go together)

Key Insight:
Things that look similar often belong in the same group
(Just like organizing books by topic on a shelf)

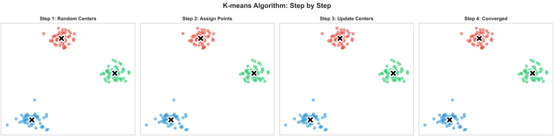


The Process:

1. Choose K *(number of clusters)*
2. Place K random centroids *(initial group centers)*
3. Assign points to nearest centroid *(by calculating distances)*
4. Update centroids *(move to cluster mean)*
5. Repeat until convergence *(no changes occur)*

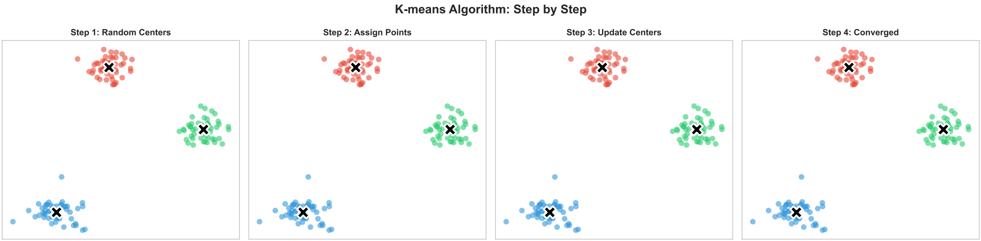
Strengths:

- Fast and scalable
- Easy to understand
- Works well for spherical clusters



K-Means in Action

Step-by-Step Convergence



Iteration 1 $\rightarrow$ Iteration 3 $\rightarrow$ Iteration 5 $\rightarrow$ **Converged**

The Goldilocks Problem

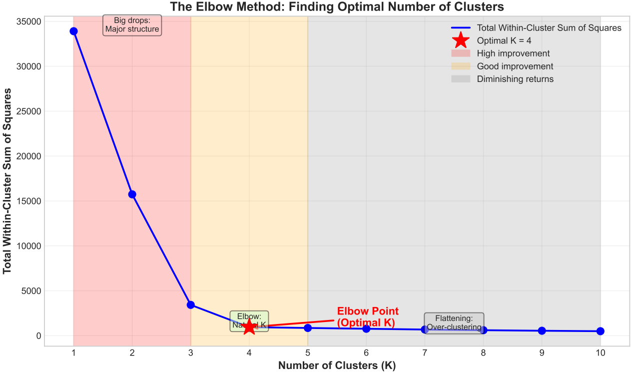
Too Few vs. Too Many Groups

Too Few (K	Just Right (K	Too Many (K
Oversimplification	**Optimal Balance**	**Analysis Paralysis**
• Mixed segments • Lost nuance • Generic solutions	• Clear segments • Actionable insights • Manageable complexity	• Overfitting • Tiny segments • Impossible to act on

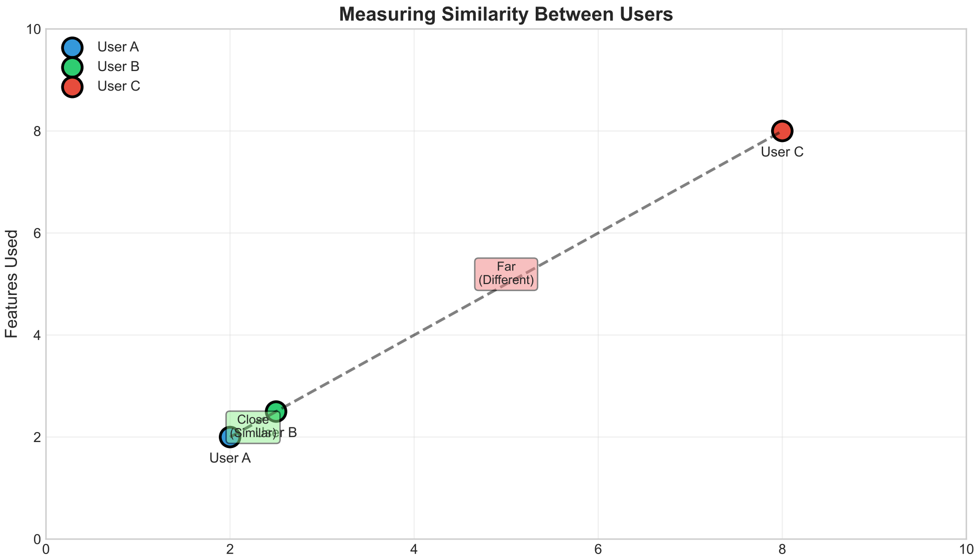
How do we find the sweet spot?

Finding the Elbow:

- Plot inertia vs K
- Look for the "elbow"
- Balance between:
  - Too few: Mixed groups
  - Too many: Overfitting



Optimal K = 5
Best trade-off between simplicity and accuracy


Measuring Similarity Between Users
User A
User B
User C
Features Used
Far
(Different)
Close
(Similar)
User A
User B
User C
0
2
4
6
8
10
0
2
4
6
8
10

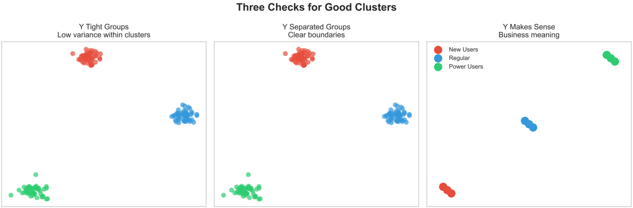
Silhouette Score:

- Ranges from -1 to +1 *(mathematical measure)*
- Higher = better separation *(clearer groups)*
- Our score: **0.73** *(good clustering!)*

What it measures:

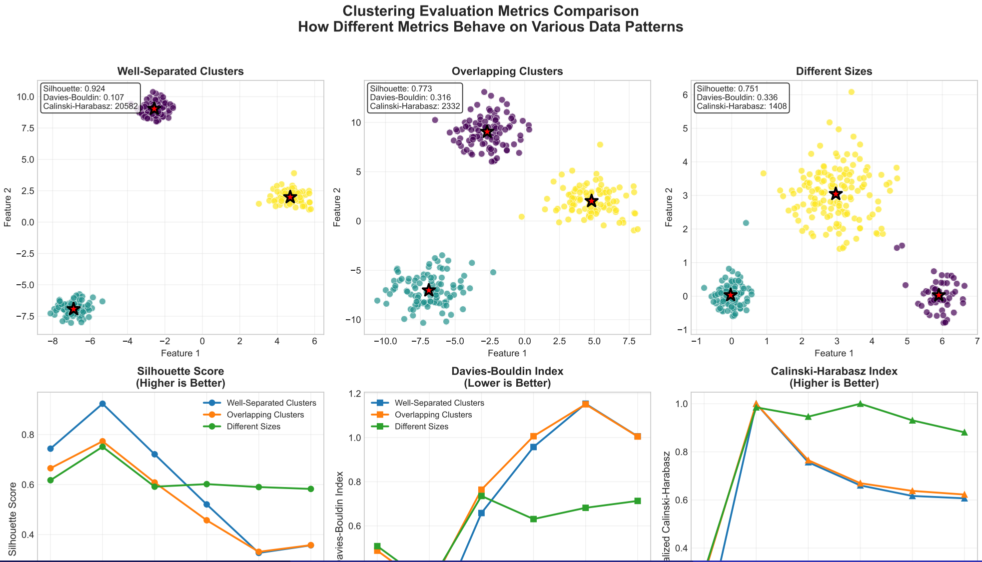
- Within-cluster cohesion
- Between-cluster separation
- Overall cluster validity

0.73 = Strong clusters!



Comparing Evaluation Metrics

Different Metrics for Different Data Patterns



K-Means Assumes Spherical Clusters

But what about:

- Innovations connected through technology stacks
- Domain-specific innovation clusters
- Evolution patterns (incremental, disruptive)
- Outliers and noise points

K-Means Forces Round Pegs into Round Holes

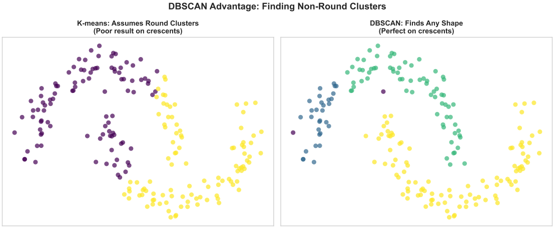
Solution: Density-Based Clustering

DBSCAN Advantages:

- No need to specify K *(finds groups automatically)*
- Finds arbitrary shapes *(not just circles)*
- Identifies outliers *(points that don't belong)*
- Handles noise well *(robust to random points)*

Perfect for:

- Non-spherical patterns
- Varying densities
- Outlier detection
- Exploratory analysis



DBSCAN: Understanding Parameters

Two Simple Settings Control Everything

Epsilon (Distance)

What it does:
Sets the maximum distance to consider points as neighbors

Think of it as:
How far can points be apart and still be friends?

Too small: Many tiny clusters
Too large: Everything merges

MinPts (Density)

What it does:
Minimum neighbors needed to form a dense region

Think of it as:
How many friends make a group?

Too small: Noise becomes clusters
Too large: Small clusters vanish

Rule of thumb: MinPts $= 2 \times$ dimensions

Clustering Algorithm Comparison

Technical Characteristics at a Glance

Algorithm	Speed	Shape	Outliers	Params	Best For
K-Means	Fast $O(nkt)$	Spherical clusters	Sensitive	K only	Quick segments
DBSCAN	Medium $O(n \log n)$	Any shape	Robust (detects)	eps, MinPts	Complex shapes
Hierarchical	Slow $O(n^2)$	Any shape	Moderate	Distance threshold	Multi-level analysis
GMM	Medium $O(nkt)$	Elliptical clusters	Moderate	K, covariance	Overlapping groups

Each algorithm has its strengths - choose wisely!

When to Use Each Algorithm

Practical Decision Guide

K-Means

Perfect when:

- Speed is critical
- Clusters are roughly equal size
- You know K in advance
- Data has spherical patterns

Hierarchical

Perfect when:

- Need multiple granularities
- Want to visualize relationships
- Small to medium datasets
- Exploring data structure

DBSCAN

Perfect when:

- Clusters have irregular shapes
- Outliers need identification
- Density varies across data
- You don't know K

GMM

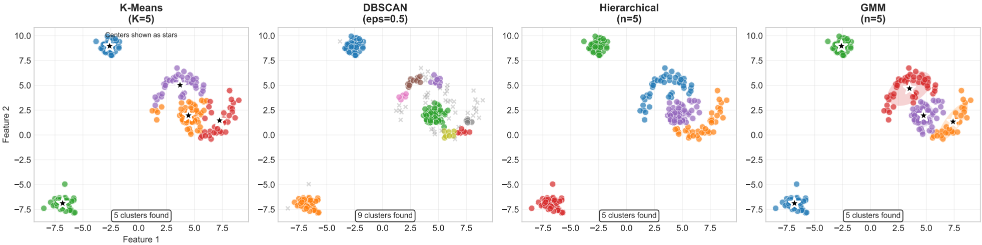
Perfect when:

- Groups overlap
- Need probability scores
- Elliptical cluster shapes
- Soft assignments needed

Algorithm Visual Comparison

Same Data, Different Approaches

Clustering Algorithms Visual Comparison
Same Data, Different Approaches



K-Means (K=5)

- Fast and scalable
- Spherical clusters
- Fixed K required
- Sensitive to outliers

Best for: Quick segmentation with known cluster count

Complexity: O(nkt)

DBSCAN (eps=0.5)

- Finds arbitrary shapes
- Identifies outliers
- No K needed
- Sensitive to parameters

Best for: Anomaly detection and irregular patterns

Complexity: O(n log n)

Hierarchical (n=5)

- Dendrogram output
- No K needed initially
- Interpretable
- Computationally expensive

Best for: Taxonomies and exploring relationships

Complexity: $O(n^2)$

GMM (n=5)

- Soft assignments
- Elliptical clusters
- Probabilistic
- Assumes Gaussian distribution

Best for: Overlapping groups and uncertainty modeling

Complexity: O(nkt)

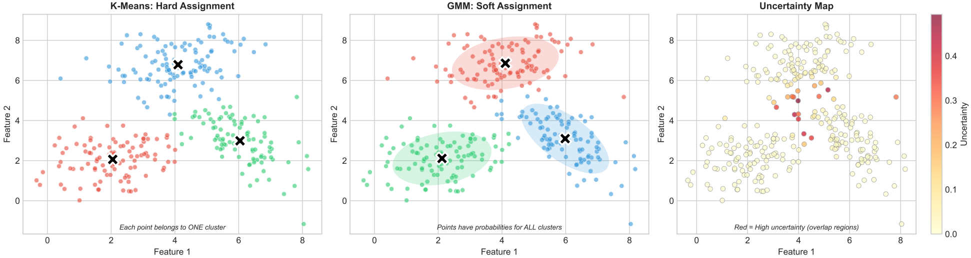
Dataset: Mix of 3 Gaussian blobs and 2 moon-shaped clusters (250 points total)

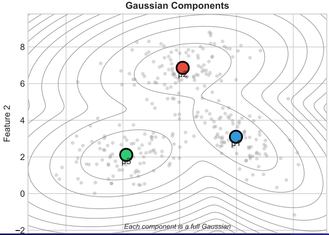
Gaussian Mixture Models (GMM)

Soft Clustering for Overlapping Innovation Categories

Gaussian Mixture Models (GMM): Soft Clustering for Innovation

Beyond Hard Boundaries: Probabilistic Innovation Classification





GMM vs K-means

GMM Advantages:
- Soft assignments (probabilities)
- Captures cluster shape (elliptical)
- Handles overlapping clusters
- Provides uncertainty estimates
- Models data generation process

K-means Advantages:
- Faster computation
- Simpler interpretation
- Less parameters
- More stable results
- Works well for spherical clusters

When to use GMM:
- Overlapping innovation categories
- Need probability scores
- Non-spherical clusters
- Uncertainty quantification needed

Innovation Category Probabilities

Innovation	Tech	Service	Social
AI Assistant	0.85	0.10	0.05
Sharing Platform	0.30	0.45	0.25
Green Energy	0.60	0.15	0.25
Digital Health	0.40	0.50	0.10

GMM provides probability of belonging to each category

Fixed K Gives One View

But real relationships are hierarchical:

- Organization: Company → Department → Team → Individual
- Geography: Country → Region → City → Neighborhood
- Products: Category → Subcategory → Brand → SKU
- Innovations: All → Categories → Sub-types → Specific solutions

K-means: Pick 5 groups and that's it

What if we need flexibility?

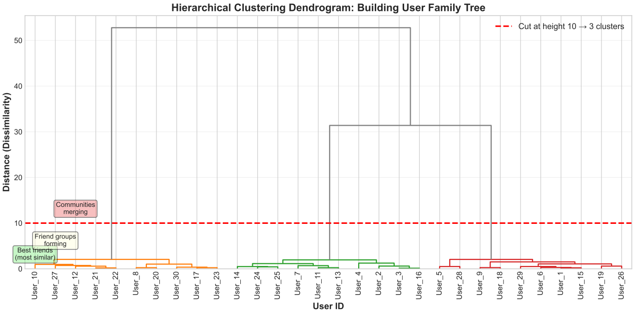
Solution: See the full hierarchy, cut where needed

Dendrogram Benefits:

- Shows cluster hierarchy
- Multiple granularities
- Natural relationships
- No preset K needed

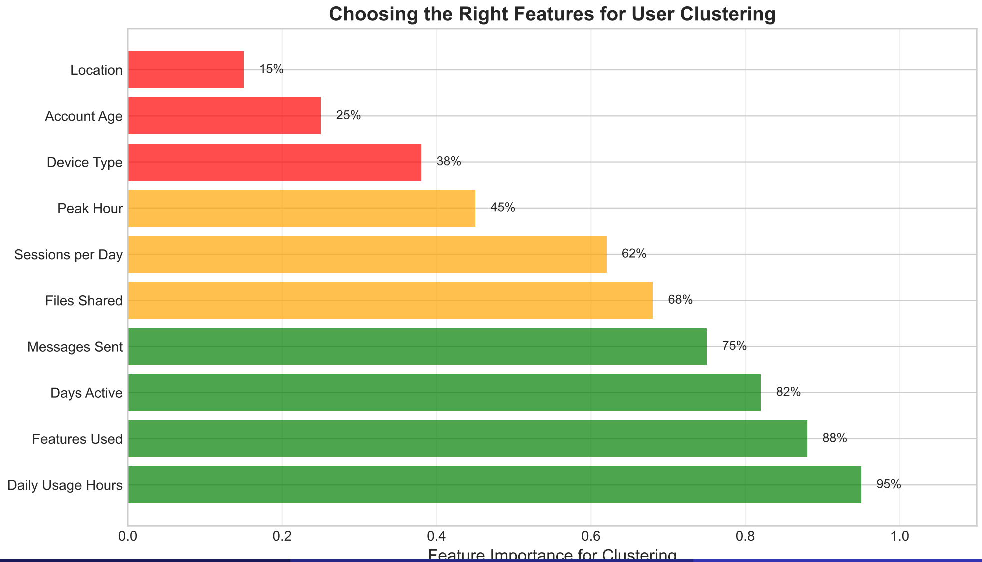
Cut the tree at any level:

- High cut = Few clusters
- Low cut = Many clusters
- Choose based on needs



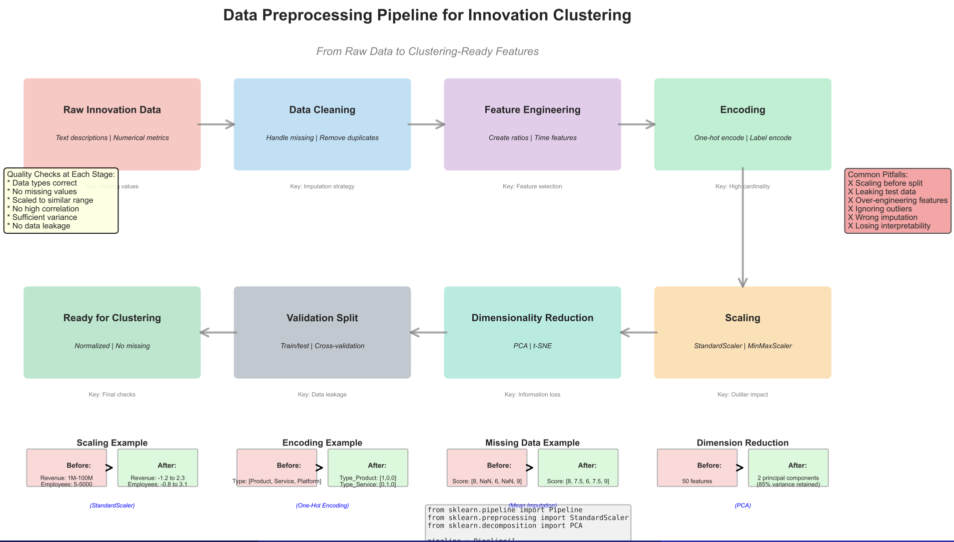
What Drives the Clusters?

Feature Importance Analysis



Data Preprocessing Pipeline

From Raw Data to Clustering-Ready Features



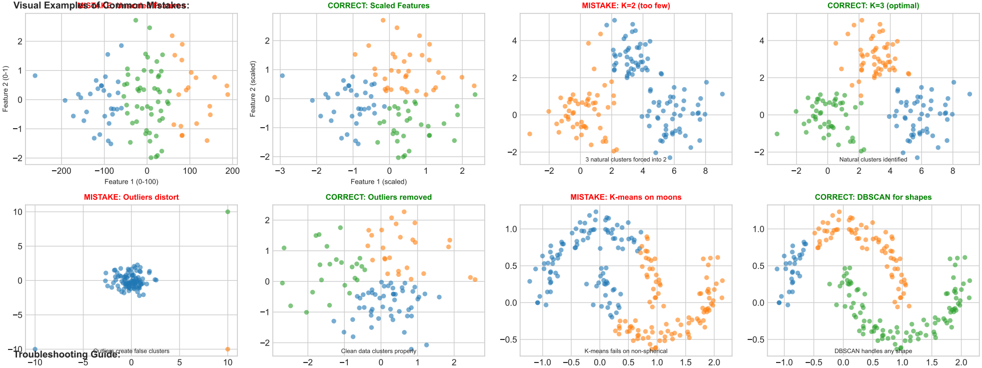
Common Mistakes & Troubleshooting

Learn from These Pitfalls

Common Clustering Mistakes & Troubleshooting Guide

Learn from These Mistakes to Master Clustering

Visual Examples of Common Mistakes:



Troubleshooting Guide:

Problem	Symptoms	Solution	Prevention
Poor separation	*Low silhouette score*	Try different K or algorithm	Use elbow method
	Different runs = different clusters	Set random_state, increase n_init	Use deterministic init
	Takes too long to converge	Reduce features, subsample data	Use MiniBatch K-means

GOLDEN RULES:
1. Always scale your features
2. Visualize before clustering
3. Try multiple algorithms
4. Validate with domain knowledge
5. Check cluster stability

WARNING SIGNS:
* Silhouette < 0.3
* Clusters change each run
* Single point clusters
* All points in one cluster

SUCCESS INDICATORS:
* Silhouette > 0.5
* Stable across runs
* Balanced cluster sizes
* Makes business sense

Parameter Tuning Guidelines

Recommended Ranges and Best Practices

Clustering Parameter Tuning Guidelines

Recommended Ranges, Methods, and Best Practices

K-Means

Parameter	Range	Default	Tuning Method
n_clusters (K)	`2-10`	3-5	*Elbow/Silhouette*
init	`['k-means++', 'random']`	k-means++	*Always k-means++*
n_init	`10-100`	10	*More for stability*
max_iter	`100-1000`	300	*Increase if no convergence*
tol	`1e-6 to 1e-2`	1e-4	*Smaller for precision*

DBSCAN

Parameter	Range	Default	Tuning Method
eps	`0.01-2.0`	0.5	*k-distance plot*
min_samples	`3-20`	2*dims	*Domain knowledge*
metric	`['euclidean', 'manhattan']`	euclidean	*Data dependent*
algorithm	`['auto', 'ball_tree']`	auto	*Auto is fine*
leaf_size	`10-50`	30	*Memory vs speed*

GMM

Parameter	Range	Default	Tuning Method
n_components	`2-10`	3-5	*BIC/AIC*
covariance_type	`['full', 'diag', 'spherical']`	full	*Start full, simplify*
max_iter	`50-500`	100	*Monitor convergence*
n_init	`1-10`	1	*More for stability*
init_params	`['kmeans', 'random']`	kmeans	*kmeans faster*

Tuning Strategies

Grid Search

Pros: Exhaustive, Reproducible, Simple

Cons: Slow, Curse of dimensionality

Use when: Small parameter space

Random Search

Pros: Faster, Better for many params, Parallelizable

Cons: May miss optimum, Not reproducible

Use when: Large parameter space

Bayesian Opt

Pros: Efficient, Learns from history, Fewer iterations

Cons: Complex, Overhead for simple problems

Use when: Expensive evaluations

Validation Metrics

Metric	Range	Interpretation	Use For
Silhouette	[-1, 1]	Higher is better	General quality
Davies-Bouldin	[0, ∞)	Lower is better	Cluster separation
Calinski-Harabasz	[0, ∞)	Higher is better	Dense clusters
Inertia	[0, ∞)	Lower is better	K-means only
BIC/AIC	(-∞, ∞)	Lower is better	GMM selection

Tuning Best Practices

1. Start with defaults, then tune
2. Use cross-validation when possible
3. Consider computational budget
4. Log all experiments
5. Visualize parameter effects
6. Use domain knowledge
7. Check stability across runs
8. Don't overfit to metrics

IMPORTANT:
No metric is perfect!
Always validate with:
• Visual inspection
• Domain expertise
• Business goals

Check Your Understanding - Part 2

Technical Concepts Review

Quick Quiz

1. K in K-means stands for:
   - ☐ Kernel
   - ✓ Number of clusters
   - ☐ Constant
2. DBSCAN finds:
   - ☐ Only circles
   - ✓ Any shape clusters
   - ☐ Exactly K groups

Can You Calculate?

If Silhouette Score = 0.75:

- Is this good? Yes!
- Range is [-1, 1]
- Higher = better separation

Remember:

- Elbow method finds optimal K
- Scale your data first!

Great job! Now let's apply these concepts!

We've learned the technical tools:

Clustering, metrics, quality measures

But clusters are just numbers...

Until we connect them to innovation opportunities

Let's transform data into innovation insights

Each cluster represents innovation opportunities and patterns

PART 3

Innovation Pattern Analysis

What we'll create:

- Data-driven innovation archetypes
- Innovation pattern maps per category
- Cluster-specific journeys
- Opportunity heat maps
- Design priority matrices

Where ML reveals innovation patterns

Part 3: Learning Objectives

Innovation Applications You'll Explore

By the end of Part 3, you will be able to:

- **Create** innovation archetypes
- **Map** innovation patterns
- **Design** opportunity matrices
- **Analyze** innovation lifecycles
- **Build** ecosystem maps
- **Prioritize** innovation efforts

Design Outcomes

- Innovation taxonomy framework
- Cluster-based strategies
- Data-driven prioritization
- Opportunity identification
- Pattern recognition skills
- Ecosystem understanding

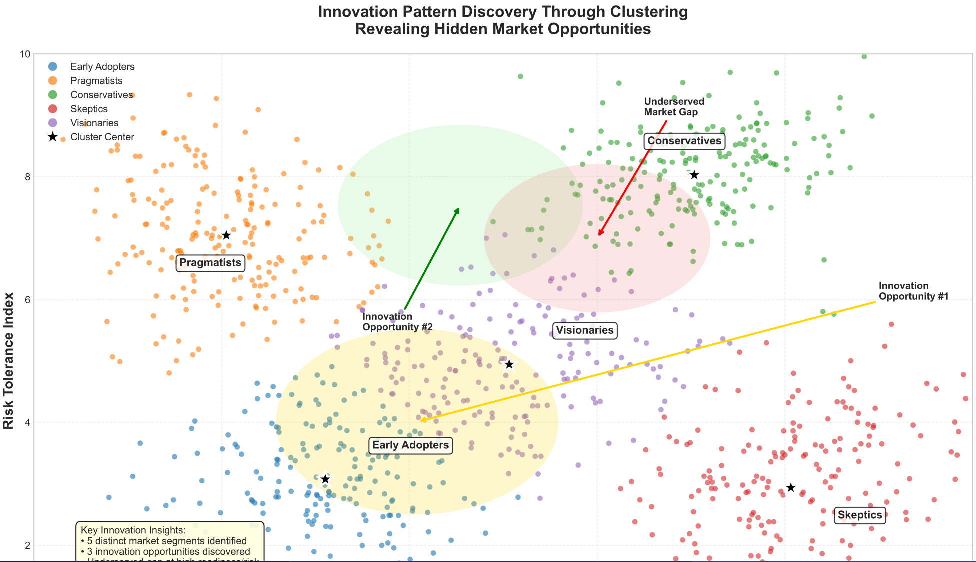
PART 3

Design Integration

Bridging Technology & Human Experience

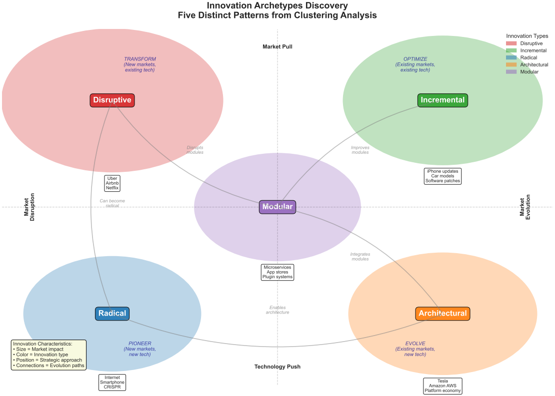
From Data Points to Innovation Insights

Bridging the Technical-Human Gap



AI-Generated Innovation Archetypes

Data-Driven Character Development and Pattern Mapping

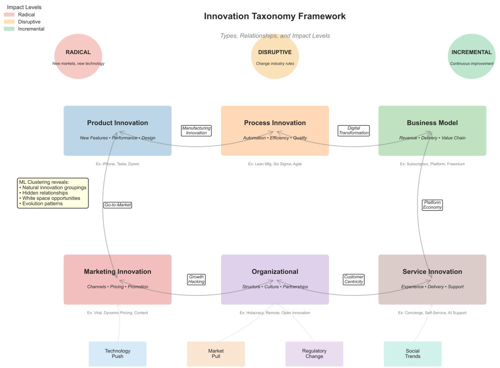


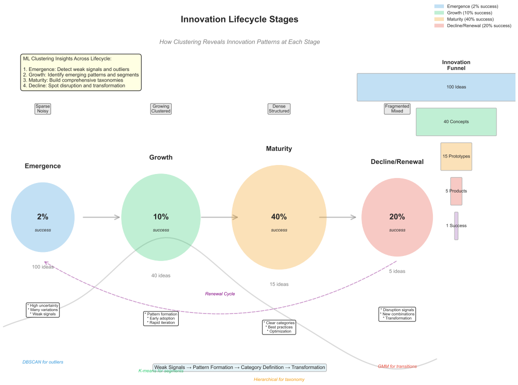
Innovation Types:

- Disruptive Innovation
- Incremental Improvement
- Platform-Based

Pattern Insights:

- Each cluster's unique traits
- Innovation velocity differences





Framework Levels:

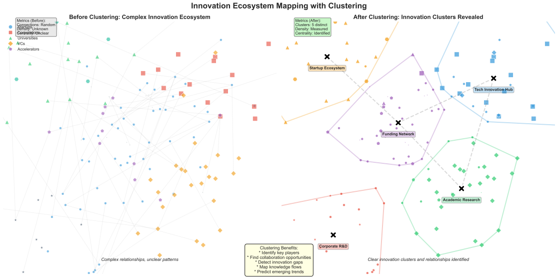
- Types & relationships
- Impact measurements
- Strategic positioning

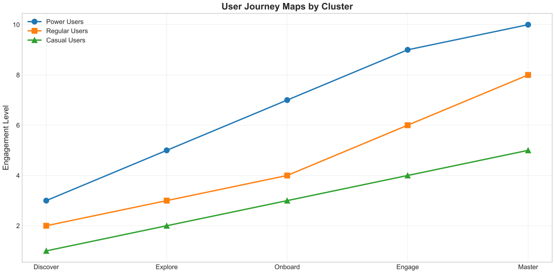
Lifecycle Stages:

- Ideation & discovery
- Development & testing
- Launch & scaling

Innovation Ecosystem & Journey Mapping

From Networks to Evolution Paths





Ecosystem Elements:

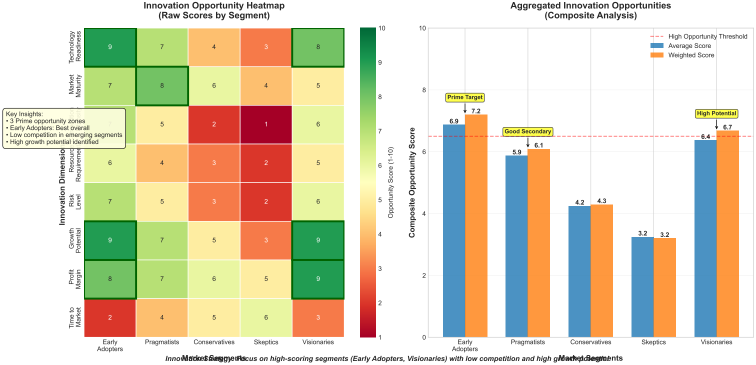
- Network connections
- Stakeholder clusters
- Value flows

Evolution Paths:

- Different speeds
- Varying trajectories
- Unique milestones

Innovation Opportunities by Cluster

Where Each Category Has Potential



Key Findings:

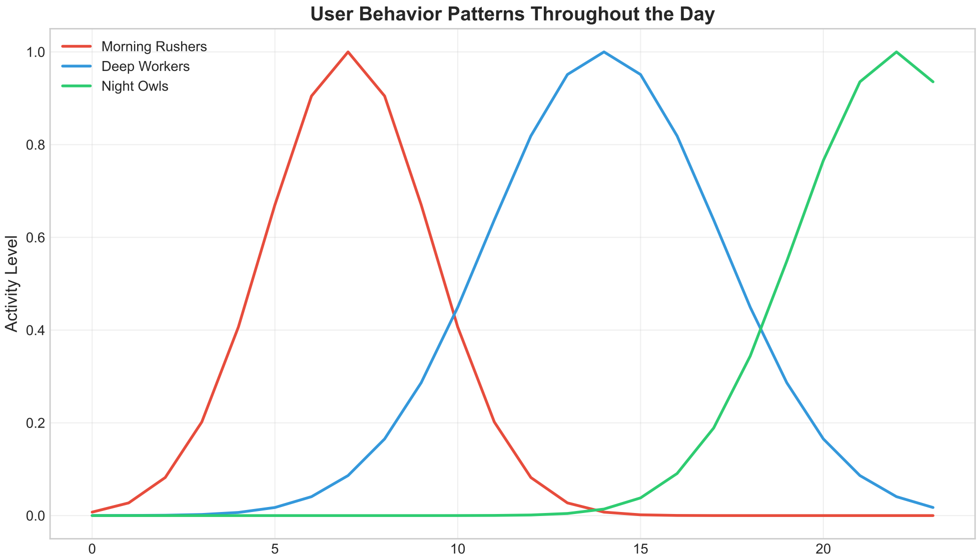
- Emerging tech: Early stage
- Disruptive: Scalability
- Incremental: Integration
- Platform-based: Network effects

Design implication:

One solution won't fit all!

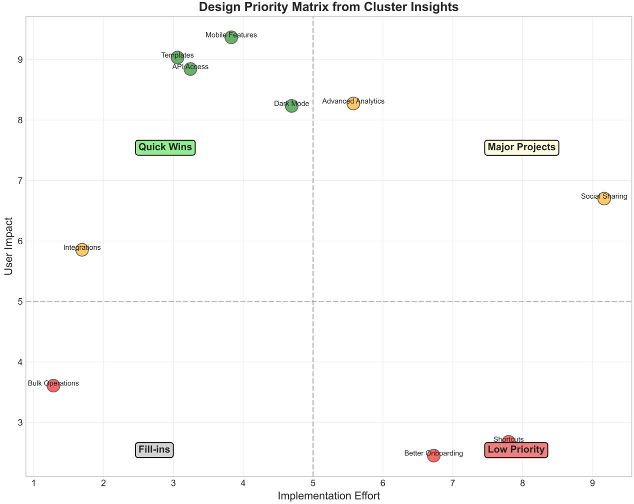
Innovation Patterns Revealed

What Clusters Tell Us About Evolution



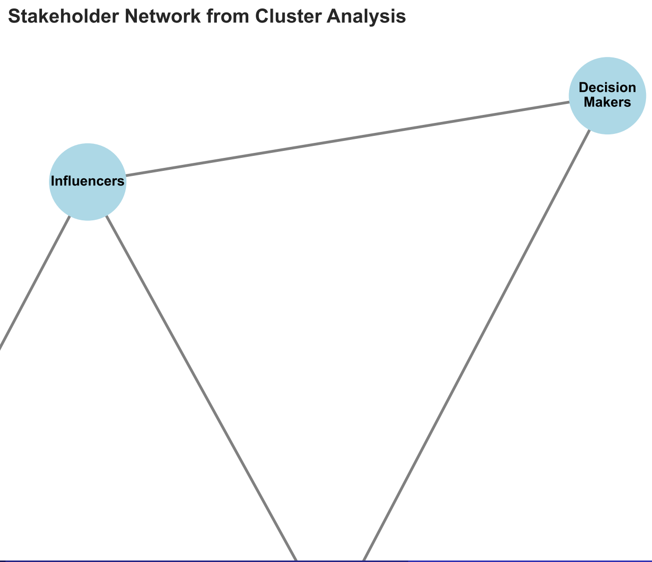
Design Priority Matrix

Where to Focus Your Efforts



Priority Quadrants:

- **High Impact + High Effort**
  Strategic initiatives
- **High Impact + Low Effort**
  Quick wins
- **Low Impact + Low Effort**
  Fill-ins
- **Low Impact + High Effort**
  Avoid


Stakeholder Network from Cluster Analysis
Decision Makers
Influencers

Match the Application

Match algorithm to use case:

1. Customer segmentation → K-means
2. Finding outliers → DBSCAN
3. Creating taxonomy → Hierarchical
4. Overlapping groups → GMM

Design Thinking

How does clustering help in:

- **Empathize**: Find user groups
- **Define**: Identify patterns
- **Ideate**: Discover opportunities
- **Prototype**: Target solutions
- **Test**: Validate segments

Excellent! Ready to practice with real data?

You've learned:

- The clustering algorithms
- How to validate quality
- Design applications

Now let's see it in action

How these techniques work in practice
to find patterns in data

PART 4

Summary & Practice

What we'll do:

- See real-world success patterns
- Consolidate key learnings
- Practice with exercises
- Preview next week
- Explore resources

From learning to doing

PART 4

Summary & Practice

Real-World Applications & Next Steps

Academic Applications:

- Student performance analysis *(grouping by learning styles)*
- Research paper categorization *(organizing by topics)*
- Course recommendation systems *(matching students to courses)*
- Exam question classification *(grouping by difficulty)*

Benefits You'll See:

- Better understanding of patterns
- Faster data analysis
- More accurate groupings
- Clearer insights from data

Key Takeaways
What We've Learned

Technical Skills

- K-means clustering algorithm
- Choosing optimal K with elbow method
- Silhouette scores for validation
- DBSCAN for complex shapes
- Hierarchical clustering

Design Applications

- Data-driven innovation archetypes
- Segment-specific journeys
- Opportunity identification
- Priority matrices
- Scaled innovation analysis

Clustering transforms data into actionable innovation insights

Implementation Checklist

Ensuring Successful Clustering Projects

Data Preparation

- ☐ Collect relevant features
- ☐ Handle missing values
- ☐ Standardize/normalize data
- ☐ Remove outliers if needed
- ☐ Feature engineering complete
- ☐ Data quality verified

Quality Assurance

- ☐ Silhouette score ¿ 0.5
- ☐ Cluster sizes balanced
- ☐ Visual inspection done
- ☐ Stability tested
- ☐ Business sense verified
- ☐ Edge cases handled

Algorithm Selection

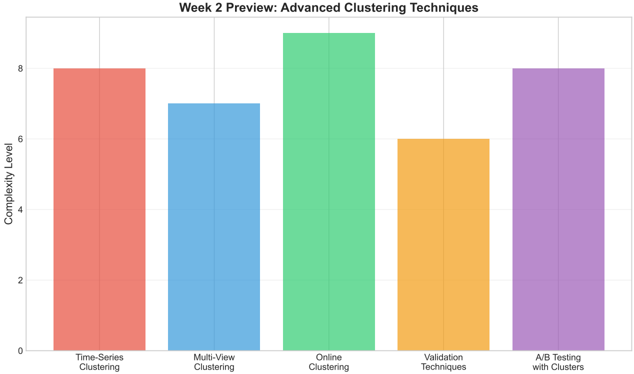
- ☐ Choose distance metric
- ☐ Select clustering method
- ☐ Determine optimal K
- ☐ Validate with metrics

Common Pitfalls

- × Forgetting to scale features
- × Wrong distance metric
- × Forcing unnatural K
- × Ignoring outliers

Next Week: Advanced Clustering

Going Deeper into Innovation Patterns



Week 2 Topics:

- Density-based clustering
- Gaussian mixture models
- Clustering validation
- Feature engineering
- Real-time clustering

Design Focus:

- Dynamic innovation tracking
- Evolving innovation landscapes
- Predictive opportunity analysis
- Micro-innovation detection

Technical Resources

Papers:

- MacQueen, J. (1967). K-means
- Ester et al. (1996). DBSCAN
- Rousseeuw (1987). Silhouettes

Tools:

- scikit-learn clustering
- Orange data mining
- KNIME analytics

Design Resources

Books:

- "Design Thinking" - Tim Brown
- "Sprint" - Jake Knapp
- "Lean UX" - Jeff Gothelf

Applications:

- Miro (journey mapping)
- Figma (archetype creation)
- Optimal Workshop

Questions? Let's discuss!

Glossary of Technical Terms

Key Concepts Quick Reference

Clustering Algorithms:

- **K-Means**: Partitions data into K predefined clusters
- **DBSCAN**: Density-based spatial clustering
- **Hierarchical**: Builds cluster tree (dendrogram)
- **GMM**: Gaussian Mixture Models, soft clustering

Key Parameters:

- **K**: Number of clusters
- **eps**: Neighborhood radius (DBSCAN)
- **min_samples**: Minimum points for density
- **n_init**: Number of random initializations

Evaluation Metrics:

- **Silhouette**: Cluster cohesion vs separation [-1,1]
- **Inertia**: Sum of squared distances to centroids
- **Davies-Bouldin**: Ratio of within to between distances
- **Calinski-Harabasz**: Ratio of dispersions

Innovation Terms:

- **Empathy Mapping**: Understanding user perspectives
- **Pain Points**: User problems/frustrations
- **User Archetypes**: Representative user groups
- **Innovation Ecosystem**: Connected stakeholders

Implementation Checklist

Your Step-by-Step Guide to Success

Data Preparation:

- ☐ Collect innovation feedback data
- ☐ Clean and remove duplicates
- ☐ Handle missing values
- ☐ Normalize/standardize features
- ☐ Create feature vectors

Algorithm Selection:

- ☐ Analyze data distribution
- ☐ Choose appropriate algorithm
- ☐ Set initial parameters
- ☐ Prepare validation strategy

Implementation:

- ☐ Run clustering algorithm
- ☐ Calculate evaluation metrics
- ☐ Visualize results (PCA/t-SNE)
- ☐ Validate with domain experts
- ☐ Iterate and refine

Innovation Application:

- ☐ Map clusters to user personas
- ☐ Identify innovation opportunities
- ☐ Create targeted solutions
- ☐ Design prototype features
- ☐ Test with user groups

Ready? Start with data preparation and work your way down!

What K-means tries to minimize:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} w_{ij} ||x_i - \mu_j||^2$$

In simple terms: Make points close to their group centers
Where:

- n = number of data points
- k = number of clusters
- $w_{ij} = 1$ if x_i belongs to cluster j, 0 otherwise
- μ_j = centroid of cluster j

Update Rules:

1. Assignment: $c^{(i)} = \arg\min_j ||x^{(i)} - \mu_j||^2$
2. Update: $\mu_j = \frac{1}{|S_j|} \sum_{i \in S_j} x^{(i)}$

Euclidean Distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Manhattan Distance:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Minkowski Distance:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

Cosine Similarity:

$$\cos(\theta) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

Jaccard Distance:

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

Mahalanobis Distance:

$$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}$$

Silhouette Score for point i:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Where:

- $a(i)$ = average distance to points in same cluster
- $b(i)$ = average distance to points in nearest neighbor cluster

Interpretation:

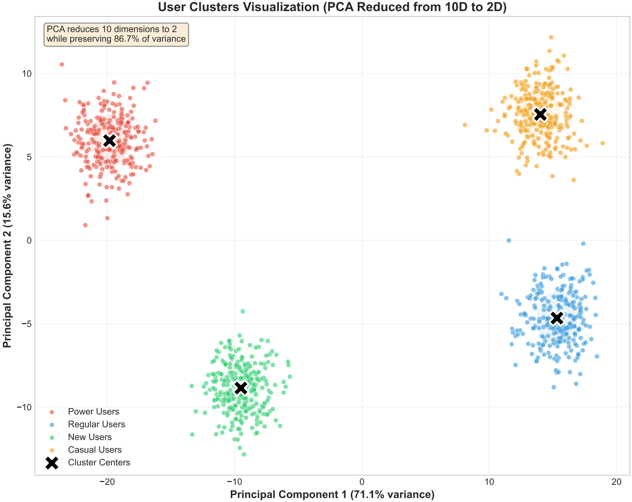
- $s(i) \approx 1$: Well clustered
- $s(i) \approx 0$: On border between clusters
- $s(i) \approx -1$: Misclassified

Overall Score:

$$S = \frac{1}{n}\sum_{i=1}^{n} s(i)$$

Appendix: Visualizing High-Dimensional Data

Making Complex Data Viewable in 2D



PCA Process:

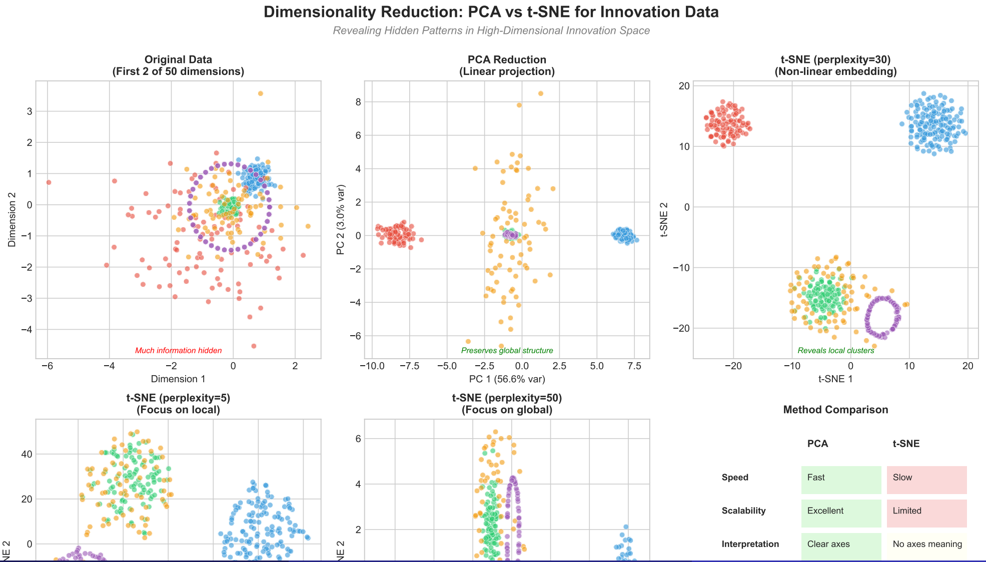
1. Standardize data
2. Compute covariance matrix
3. Find eigenvectors/values
4. Select top 2 components
5. Transform data

Variance Explained:

- PC1: 45.2%
- PC2: 28.7%
- Total: 73.9%

Dimensionality Reduction: PCA vs t-SNE

Revealing Hidden Patterns in High-Dimensional Innovation Space



Key Parameters:

- ϵ (eps): Maximum distance between points
- MinPts: Minimum points to form dense region

Point Classification:

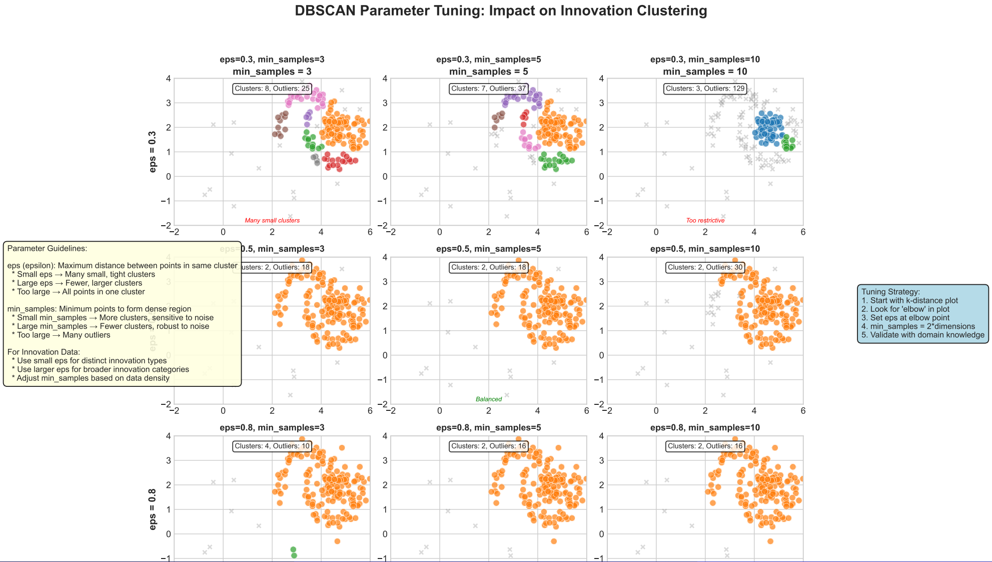
- **Core point**: Has $\geq$ MinPts within ϵ
- **Border point**: Within ϵ of core point
- **Noise point**: Neither core nor border

Algorithm Steps:

1. Find all core points
2. Form clusters from core points within ϵ
3. Assign border points to clusters
4. Mark remaining as noise

DBSCAN Parameter Tuning

Impact of eps and min_samples on Clustering Results



Appendix: Python Implementation

Ready-to-Use Code Snippets

K-Means Clustering

```
from sklearn.cluster import KMeans
import numpy as np

# Generate sample data
X = np.random.randn(1000, 2)

# Configure and fit K-means
kmeans = KMeans(
    n_clusters=3,
    random_state=42,
    n_init=10
)
labels = kmeans.fit_predict(X)

# Get cluster centers
centers = kmeans.cluster_centers_

# Calculate inertia
inertia = kmeans.inertia_
```

DBSCAN Clustering

```
from sklearn.cluster import DBSCAN

# Configure DBSCAN
dbscan = DBSCAN(
    eps=0.3,
    min_samples=5,
    metric='euclidean'
)

# Fit and predict
labels = dbscan.fit_predict(X)

# Analyze results
outliers = labels == -1
n_clusters = len(set(labels)) - 1

print(f"Clusters: {n_clusters}")
print(f"Outliers: {sum(outliers)}")
print(f"Coverage: {100*(1-sum(outliers)/len(labels)):.1f}%")
```

Appendix: Evaluation Metrics Code

Measuring Clustering Quality

Silhouette Analysis

```
from sklearn.metrics import silhouette_score
from sklearn.metrics import silhouette_samples

# Calculate overall score
score = silhouette_score(X, labels)
print(f"Silhouette Score: {score:.3f}")

# Per-sample scores
sample_scores = silhouette_samples(X, labels)

# Per-cluster average
for i in range(n_clusters):
    cluster_scores = sample_scores[labels == i]
    avg = cluster_scores.mean()
    print(f"Cluster {i}: {avg:.3f}")
```

Finding Optimal K

```
# Elbow method
inertias = []
silhouettes = []
K_range = range(2, 10)

for k in K_range:
    km = KMeans(n_clusters=k,
                random_state=42)
    labels = km.fit_predict(X)

    inertias.append(km.inertia_)
    silhouettes.append(
        silhouette_score(X, labels)
    )

# Find elbow point
# Plot inertias vs K
# Choose K at elbow
```

Appendix: Implementation Guidelines

Practical Considerations

Data Preparation

- Standardize features
- Handle missing values
- Remove outliers (if needed)
- Feature selection/engineering
- Consider scaling methods

Validation Methods

- Silhouette score
- Davies-Bouldin index
- Calinski-Harabasz score
- Visual inspection
- Domain expert review

Algorithm Selection

- K-means: Spherical, similar size
- DBSCAN: Arbitrary shapes
- Hierarchical: Nested structure
- GMM: Overlapping clusters

Common Pitfalls

- Not scaling features
- Wrong distance metric
- Ignoring outliers
- Over-clustering
- Forcing clusters

Glossary of Technical Terms

Key Concepts Reference

Algorithms:

- **K-means**: Partitions data into K spherical clusters
- **DBSCAN**: Density-based clustering for arbitrary shapes
- **GMM**: Gaussian Mixture Models for soft clustering
- **Hierarchical**: Tree-based clustering approach

Metrics:

- **Silhouette**: Measures cluster separation (-1 to 1)
- **Inertia**: Sum of squared distances to centroids
- **Davies-Bouldin**: Ratio of within to between cluster distance

Concepts:

- **Centroid**: Center point of a cluster
- **Elbow Method**: Technique to find optimal K
- **Outlier**: Data point not belonging to any cluster
- **Convergence**: When algorithm stops improving

Preprocessing:

- **Standardization**: Zero mean, unit variance
- **Normalization**: Scale to [0,1] range
- **PCA**: Principal Component Analysis
- **t-SNE**: t-distributed Stochastic Neighbor Embedding