

# Latent Class Model Analysis: A Computational Pipeline

Siyuan Tang | STATS 607 | December 2025

---

## A. Motivation

### Problem

Analyzing multivariate categorical data becomes intractable as the number of variables grows. With  $m = 20$  binary variables, there are  $2^{20} \approx 1$  million possible patterns. Latent class models solve this by assuming  $K \ll 2^m$  latent classes within which variables are conditionally independent.

### Key challenges:

1. How many classes ( $K$ ) exist?
2. How to fit models efficiently?

### Impact

Latent class analysis is used in social sciences, medicine, marketing, and psychology for clustering and segmentation. This project provides researchers with automated, validated, and computationally efficient tools.

---

## B. Project Description

### What I Built

For my methodology of latent class modeling, see `docs/methodology.pdf`.

#### 1. Core Pipeline ( `src/` )

- **Vectorized EM algorithm** for parameter estimation
- **BIC-based model selection** with smart parallelization
- **Synthetic data generation** for validation

#### 2. Simulation Studies ( `simulation/` )

- Monte Carlo validation:  $M = 50$  simulations  $\times$  5 sample sizes  $\times$  4  $K$  values

- Tests: BIC selection accuracy, parameter estimation errors, classification performance
- Metrics: Success rates, MAE/RMSE for  $\pi$  and  $\theta$ , confusion matrices

### 3. Visualization

- BIC curves, parameter error plots, confusion matrices, accuracy trends

## Course Concepts Used

**Vectorization:** Eliminated nested loops using NumPy broadcasting

**Parallel Computing:** Adaptive parallelization strategy

- Few  $K$  values  $\rightarrow$  parallelize initializations
- Many  $K$  values  $\rightarrow$  parallelize across  $K$

**Numerical Stability:** Log-space computations with log-sum-exp trick

**Simulation Studies:** Structured Monte Carlo framework for validation

**Progress tracking:** via tqdm

**Software Engineering:** Modular design, CLI interfaces, comprehensive documentation

---

## C. Results

### Main Pipeline Demo

**Synthetic data analysis:**

```
python main.py --generate-synthetic --n-samples 2000 --k-true 3
```

**Output:** BIC correctly selects  $K = 3$

Mixture Weights:

```
True  $\pi$ :      [0.3359 0.3342 0.3299]
Estimated  $\pi$ : [0.341  0.3298 0.3292]
Mean Absolute Error (MAE):  0.003426
Root Mean Squared Error (RMSE): 0.003939
```

Categorical Probabilities ( $\theta_{rkc}$ ):

```
Mean Absolute Error (MAE):  0.013091
Root Mean Squared Error (RMSE): 0.016334
```

### Per-Class $\theta$ Errors:

Class 0: MAE = 0.013441

Class 1: MAE = 0.012655

Class 2: MAE = 0.013177

✓ True parameters saved to: results/analysis\_true\_params.npz

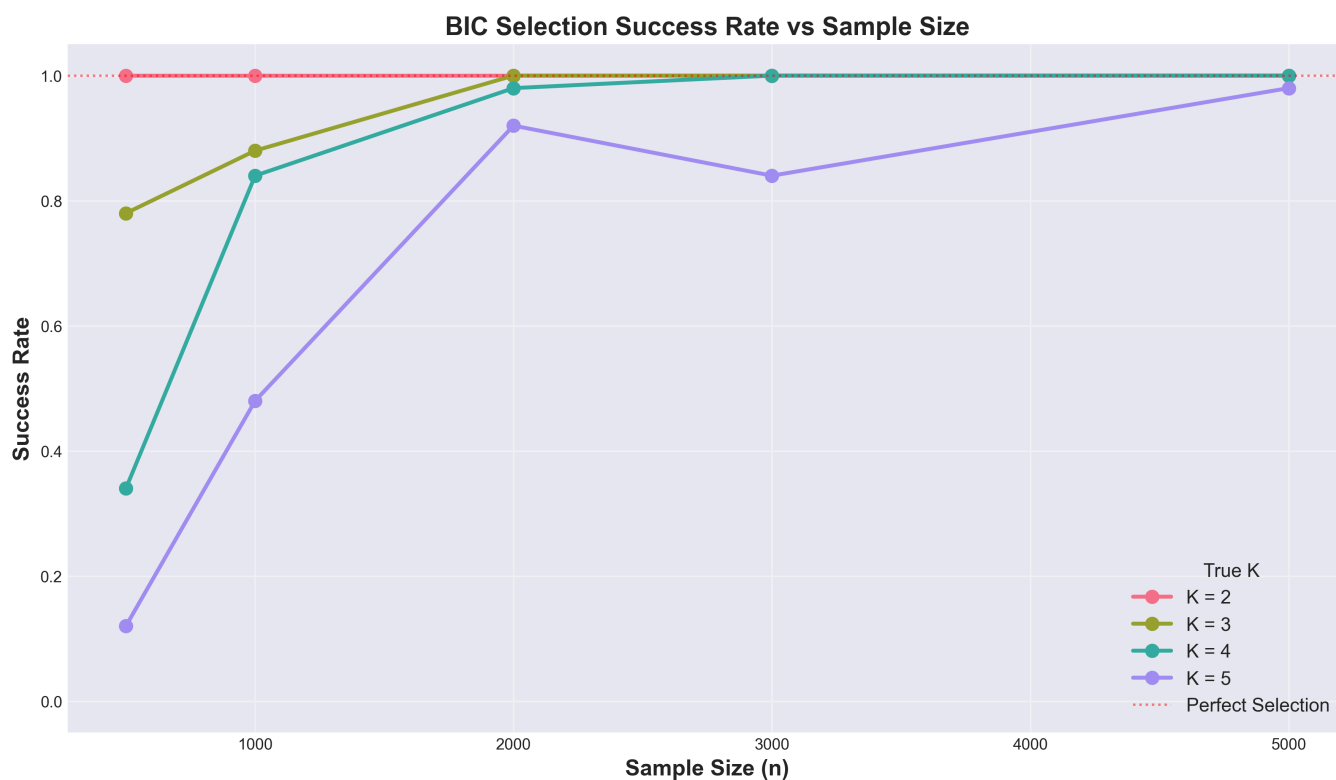
### User's dataset:

```
# Suppose the user's dataset is `data/mydata.csv`  
python main.py --data data/mydata.csv --output-prefix my_analysis
```

## Simulation Results

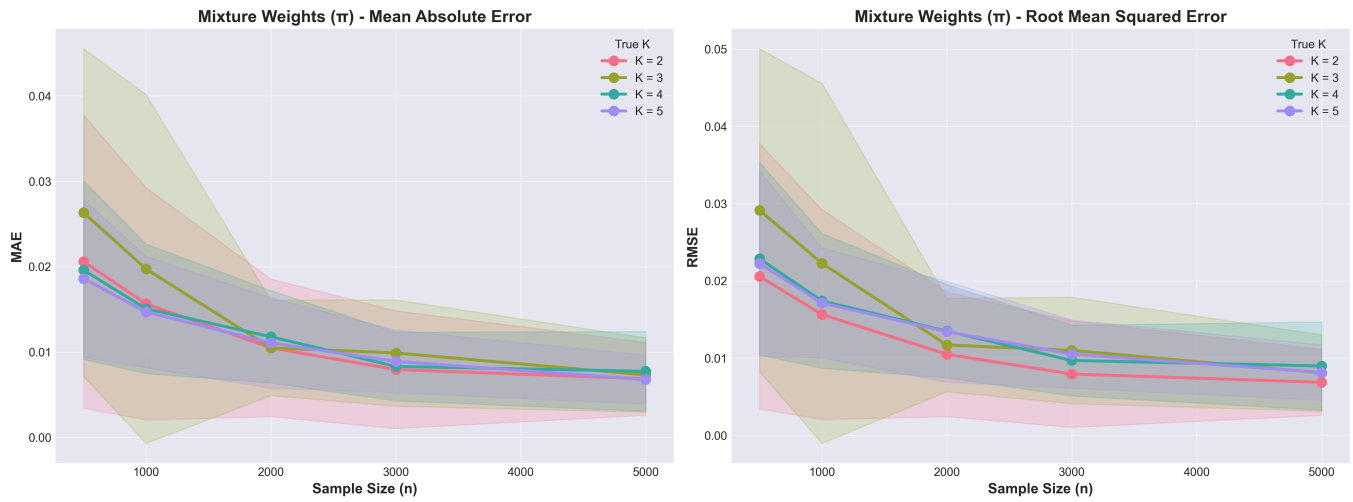
### 1. BIC Selection Accuracy

See [simulation/results/figures/bic\\_success\\_rates\\_combined.png](#).



### 2. Parameter Estimation

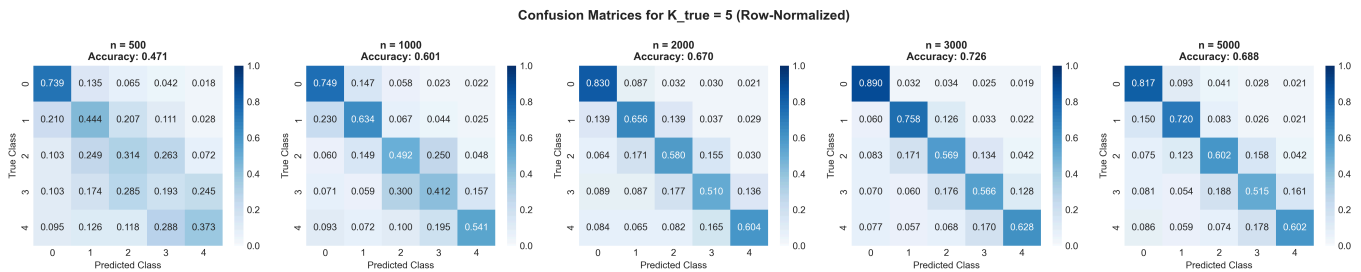
See [simulation/results/figures/pi\\_estimation\\_errors.png](#).



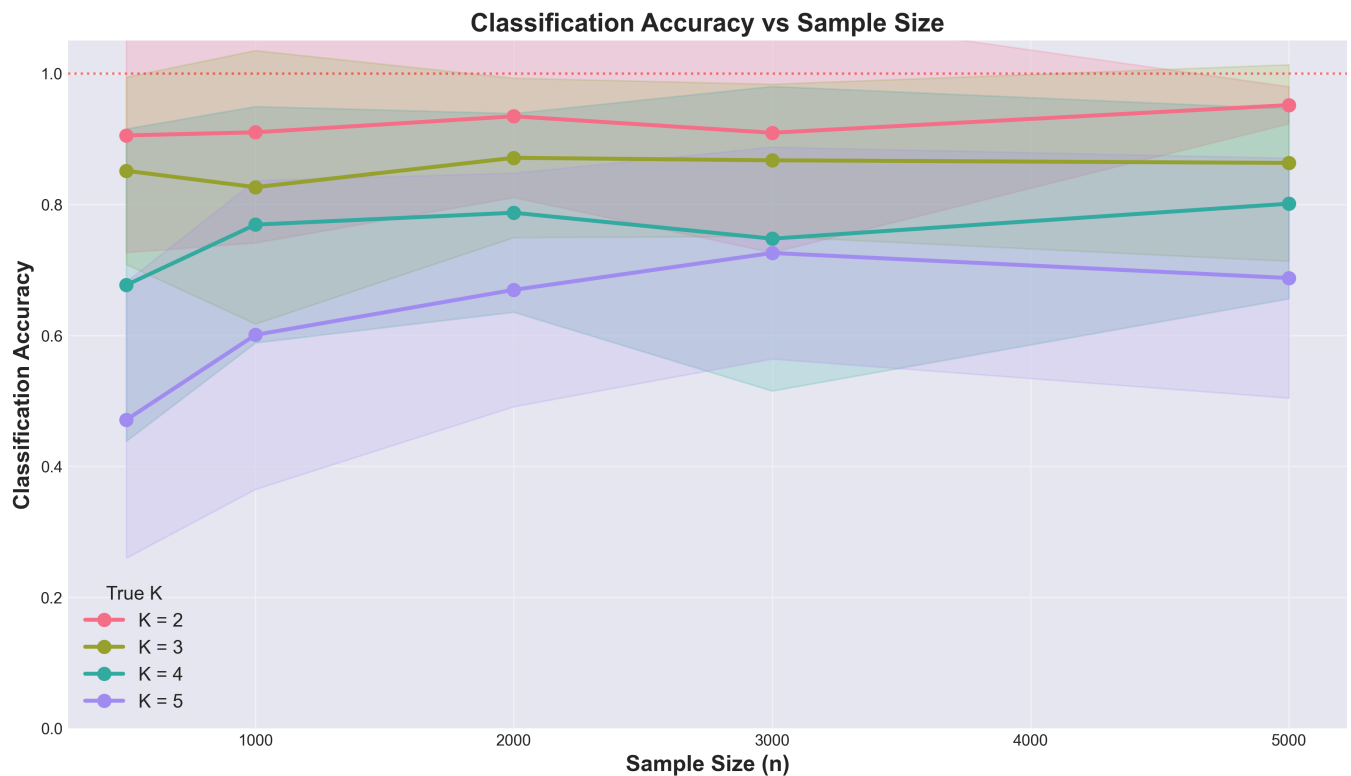
And [simulation/results/figures/theta\\_estimation\\_errors.png](#) . (Not shown here)

### 3. Classification Performance

See [simulation/results/figures/confusion\\_matrices\\_K{2,3,4,5}.png](#) . (Only [simulation/results/figures/confusion\\_matrices\\_K5.png](#) is shown here)



See [simulation/results/figures/classification\\_accuracy\\_vs\\_n.png](#) .



#### 4. Computation Time

See `simulation/results/figures/bic_computation_time.png` . (Not shown here)

---

## D. Lessons Learned

### Challenges

#### 1. Numerical Instability

- Problem: Probability products can cause underflow
- Solution: Log-space + log-sum-exp trick

#### 2. Label Switching

- Problem: EM produces arbitrary class labels
- Solution: Post-hoc sorting by mixture weights

#### 3. Parallelization Trade-offs

- Problem: Naive parallelization → memory issues and slow
- Solution: Adaptive strategy based on  $K$  range size

## How My Approach Changed

**Before STATS 607:**

- Nested loops
- Sequential processing only
- Ad-hoc scripts

**After STATS 607:**

- Vectorization and parallelization
- Modular and reusable code
- Comprehensive validation