Aristotle University of Thessaloniki

School of Electrical & Computer Engineering

P. Petrantonakis
S. Papadopoulos

Pattern Recognition & Machine Learning

Team 44

Christos Choutouridis [8997]
cchoutou@ece.auth.gr

Semester Assignment

Overview

1. **Parts A–C** cover probabilistic modeling, Bayesian decision theory, and k-NN classification.
2. **Part D** focuses on multi-class model comparison and performance analysis.
3. A **modular architecture** and a systematic **experimental methodology** are used throughout.

Part A

Maximum Likelihood Estimation

Maximum Likelihood Estimation

Probabilistic Model

- Data vectors are modeled probabilistically
- Gaussian assumption in $\mathbb{R}^2$
- Parameters unknown

$$p(\mathbf{x} \mid \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Multivariate Gaussian distribution

- Continuous probability density
- Shape controlled by covariance
- Orientation captures feature correlation

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

Maximum Likelihood Estimation

Maximum Likelihood Estimation - Mean

- Parameters estimated from data
- Maximum Likelihood principle
- Mean captures central tendency

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

Maximum Likelihood estimation — Covariance

- Covariance captures variance and correlation
- Determines shape and orientation
- Estimated from centered data

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^\top$$

Maximum Likelihood Estimation

Implementation

- Dataset loaded and split into classes
- Mean vectors and covariance matrices estimated via MLE
- Gaussian pdf evaluated on a 2D grid
- Dedicated functions used for estimation, density evaluation, and visualization

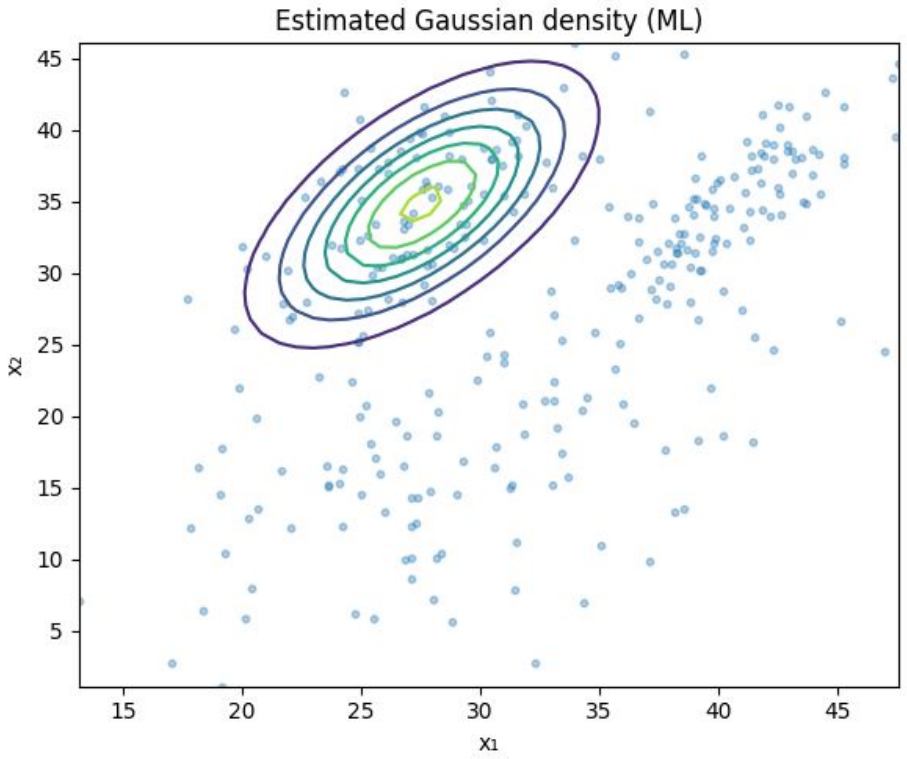
MLE implementation overview

- *mle_mean(), mle_covariance():* parameter estimation
- *estimate_gaussians_mle():* per-class Gaussian modeling
- *gaussian_pdf(), compute_gaussian_grid():* density evaluation
- *plot_gaussians_3d():* visualization of class-conditional densities

Maximum Likelihood Estimation

Estimated Gaussian density

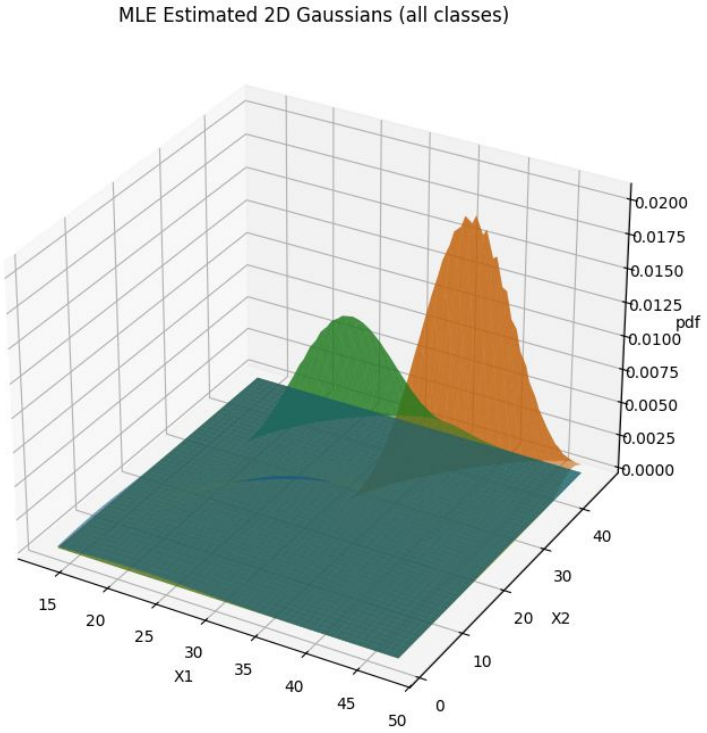
- Density evaluated on a 2D grid
- High-density regions align with data
- Model reflects data structure



Maximum Likelihood Estimation

3D visualization of estimated Gaussian densities

- Class-conditional Gaussian models estimated via MLE
- Probability density visualized as a surface over the feature space
- Highlights relative position, spread, and overlap between classes
- Used as a qualitative tool to support later classification analysis



Part B

Parzen Window Estimator

Parzen window

Parzen window estimator

- Density estimated by averaging kernel contributions
- Bandwidth h controls smoothing (bias–variance trade-off)
- We evaluate p(x) at the sample points

$$\hat{p}(x;h) = \frac{1}{Nh}\sum_{n=1}^{N} K\left(\frac{x - x_n}{h}\right)$$

Parzen window

Kernel functions

- Uniform (box) kernel
- Gaussian kernel
- Both integrate to 1 (valid pdf smoothing kernels)

Uniform Kernel

$$K(u) = \begin{cases} 1, & |u| \leq \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}$$

Uniform Kernel

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right)$$

Parzen window

Reference true pdf

- True pdf provided for evaluation
- Used to quantify the error and select h
- True likelihood computed at the sample points

$$p(x) = \mathcal{N}(x \mid \mu = 1, \sigma^2 = 4)$$

Bandwidth selection criterion

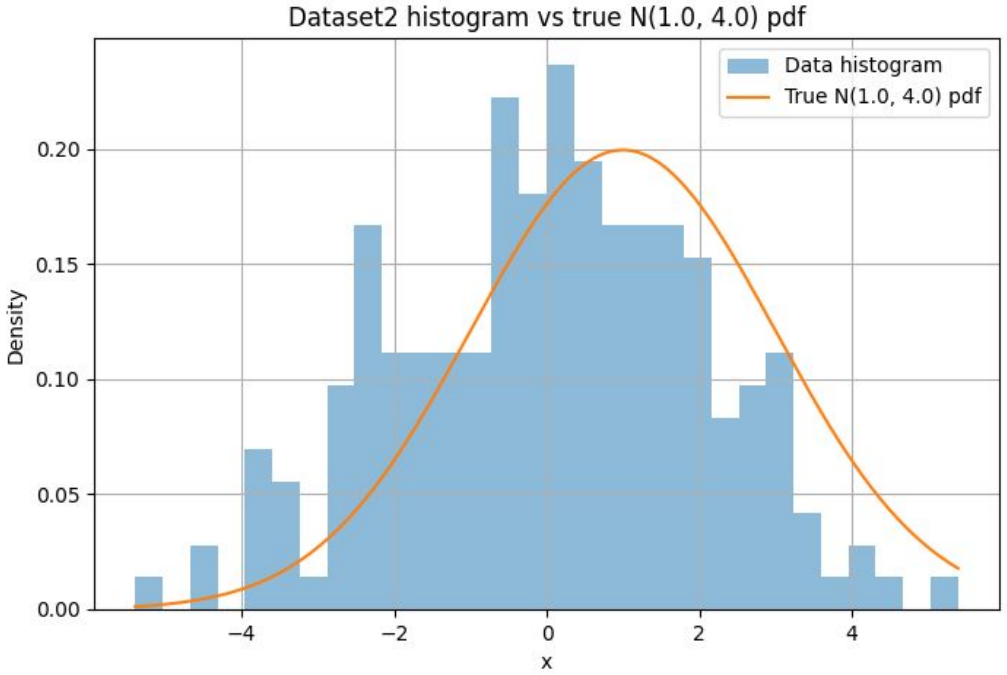
- Compute p(x_i; h) for each sample x_i
- Compare against the true pdf p(x_i)
- Select h that minimizes Mean Squared Error

$$\mathrm{MSE}(h) = \frac{1}{N} \sum_{i=1}^{N} \left(p(x_i) - \hat{p}(x_i; h) \right)^2$$

Parzen window

histogram vs true

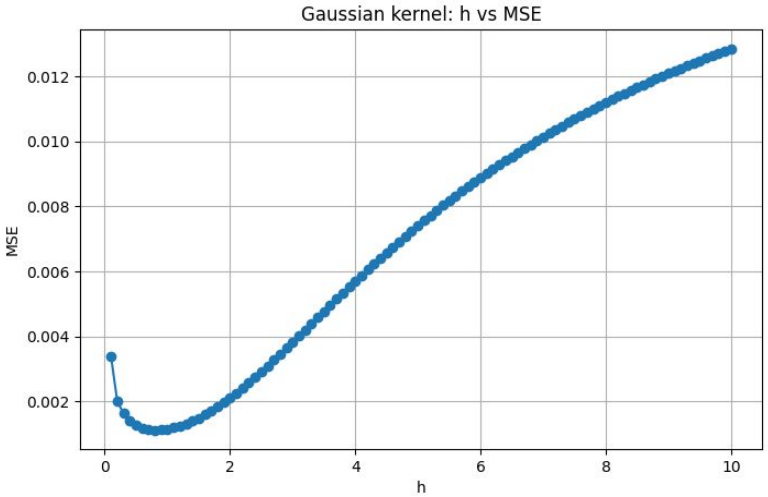
- Histogram provides a coarse density picture
- True N(1,4) pdf overlaid for sanity check
- Supports the Gaussian-like structure of Dataset 2



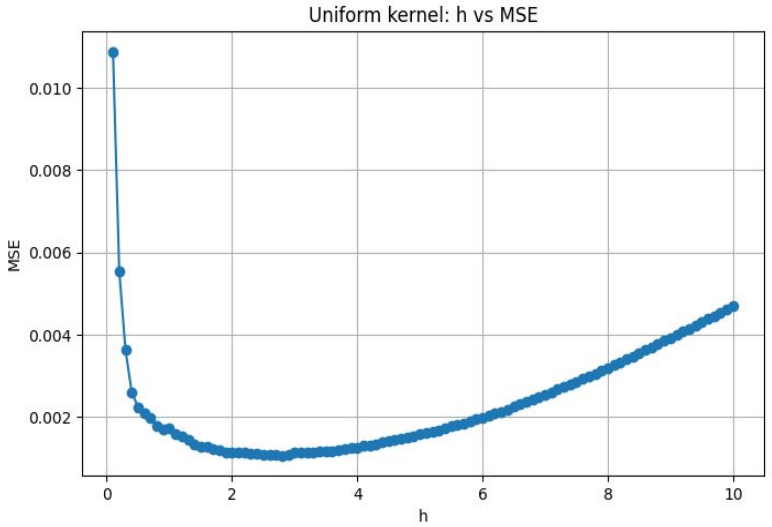
Parzen window

Plot: h vs MSE (Uniform & Gaussian)

Gaussian kernel: best h ≈ 0.8



Uniform kernel: best h ≈ 2.8



Parzen window

Implementation

- *parzen_estimate_1d()*:
  compute p̂(x_eval) using (1 / (N·h)) · Σ K((x_eval − x_n)/h)
- *evaluate_parzen()*:
  evaluate p̂(x_i) at each sample x_i in the dataset
- *true_normal_pdf_1d():*
  compute the reference N(μ, var) pdf at the same points
- *scan_bandwidths_parzen()*:
  loop over h, compute estimated pdf, compare to true pdf, store MSE(h)
- *plot_histogram_with_pdf()*, *plot_h_vs_error()*:
  generate the final figures for reporting

Part C

k-Nearest Neighbors Classifier

k-NN for Pattern Recognition (Concept)

k-Nearest Neighbors (k-NN) is a **non-parametric**, **instance-based** classifier.

- No explicit training phase
- Classification based on **local neighborhood**
- Suitable for **low-dimensional pattern recognition**
- Decision boundaries adapt to data geometry

In this assignment, k-NN is used for **binary 2D pattern classification.**

Distance Measure

We use the **Euclidean distance** to measure similarity between patterns.

Distance definition

$$d(\mathbf{x}, \mathbf{x}_i) = \sqrt{\sum_{j=1}^{d} (x_j - x_{i,j})^2}$$

- **x**: query sample
- $\mathbf{x}_i$: training sample
- D = 2 (2D patterns)

Neighborhood Selection

For each test sample:

- Compute distances to all training samples
- Sort distances in ascending order
- Select the **k nearest neighbors**

Distance definition

$$\mathcal{N}_k(\mathbf{x}) = \text{indices of the } k \text{ smallest distances}$$

This step defines the **local region** influencing the decision.

Probabilistic Classification Rule

k-NN estimates **class probabilities** using neighbor frequencies.

Class probability estimate:

$$P(y = c \mid \mathbf{x}) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x})} 1(y_i = c)$$

- Probabilities **sum to 1**
- Supports **any number of classes**
- Final label:

$$\hat{y} = \arg\max_c P(y = c \mid \mathbf{x})$$

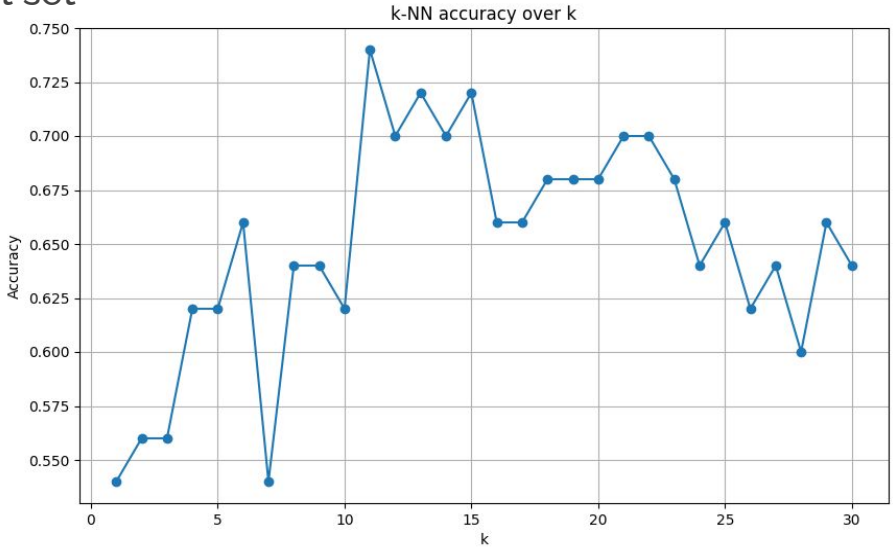
Model Selection: Accuracy vs k

To select the optimal number of neighbors:

- Evaluate k in [1, 30]
- Compute classification accuracy on the test set

Accuracy definition

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} 1(\hat{y}_i = y_i)$$



Best performance achieved at k = 11

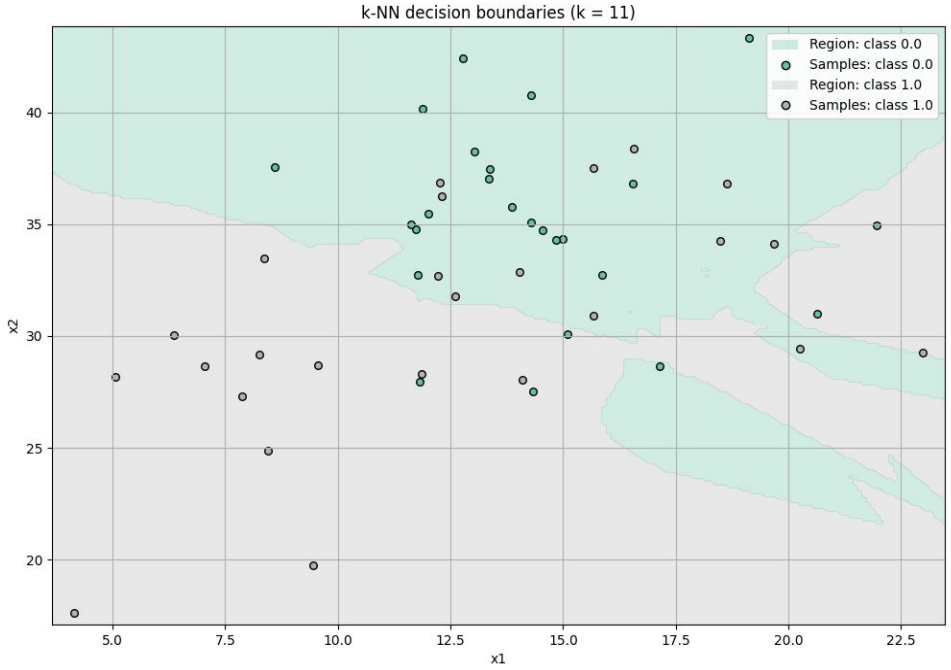
Decision Boundaries (Best k)

Using the optimal value k=11:

- A dense 2D grid is generated
- Each grid point is classified using k-NN

Observations:

- A Small k → noisy, unstable boundaries
- Large k → smoother but biased decisions
- Optimal k balances bias–variance trade-off



Implementation details

Core functions implemented:

- *eucl()*:
  Euclidean distance computation
- *neighbors()*:
  k nearest neighbors selection
- *predict()*:
  Probabilistic classification
- *evaluate_over_k()*:
  Accuracy evaluation for multiple k
- *plot_decision_boundaries_2d()*:
  2D decision region visualization

Part D

Find the best Classifier

Find the Best Classifier

Problem statement & dataset

- Multiclass classification problem 5 classes
- Input: high-dimensional (224) feature vectors
- Labeled training set and unlabeled test set
- Goal: maximize classification accuracy and generalization

	HK	HL	HM	HN	HO	HP	HQ	
4	-00.088081	-00.1358	1.1888	0.59024	0.13113	0.94097	1	
3	-00.21821	0.063382	-00.35871	-00.039761	0.75732	0.49834	2	
7	0.58943	0.219	0.8678	0.13884	1.0808	-00.54227	5	
6	-00.31295	-00.36409	-00.043227	1.1712	-00.13544	-00.077792	2	
4	-00.30453	0.3072	1.0471	0.92149	0.73302	0.67078	3	
5	-00.082145	0.12385	0.23552	0.39436	1.0049	-00.21823	4	
9	-00.1544	-00.27032	0.19659	0.084938	0.93795	0.64476	3	
3	-00.30806	-01.1012	0.24436	-00.17289	0.77315	-00.13703	5	
2	-00.14617	-00.08408	0.31663	0.591	0.33166	0.096781	2	
2	0.28557	0.10348	-00.12998	1.1834	-00.20695	0.42816	1	
5	-00.15857	0.26869	0.11591	-00.42673	-00.20697	-00.50059	5	

Methodology

Overview

- Preprocessing: scaling and PCA-based feature representations
- Models: distance-based, linear, kernel-based, and ensemble classifiers
- Evaluation: stratified validation accuracy and confusion matrices

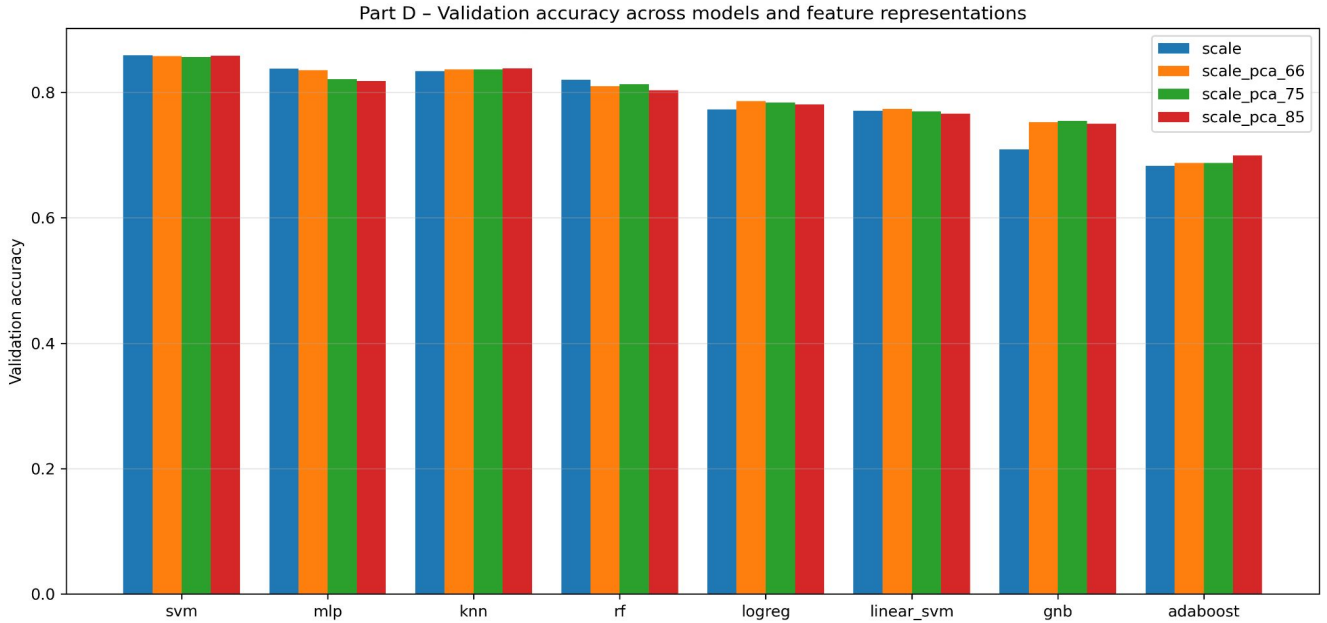
Experimental organization (Sequencer)

- Modular pipeline for preprocessing, training, and evaluation
- Systematic exploration of (preprocess, model) combinations
- Clear separation of:
  - baseline investigation
  - problem analysis
  - tuning and final training

Baseline validation results

- Multiple models evaluated under identical conditions
- Kernel-based and instance-based methods perform best
- Simpler probabilistic models underperform

- SVM achieves the highest val. accuracy
- k-NN & MLP follow closely
- Small performance differences



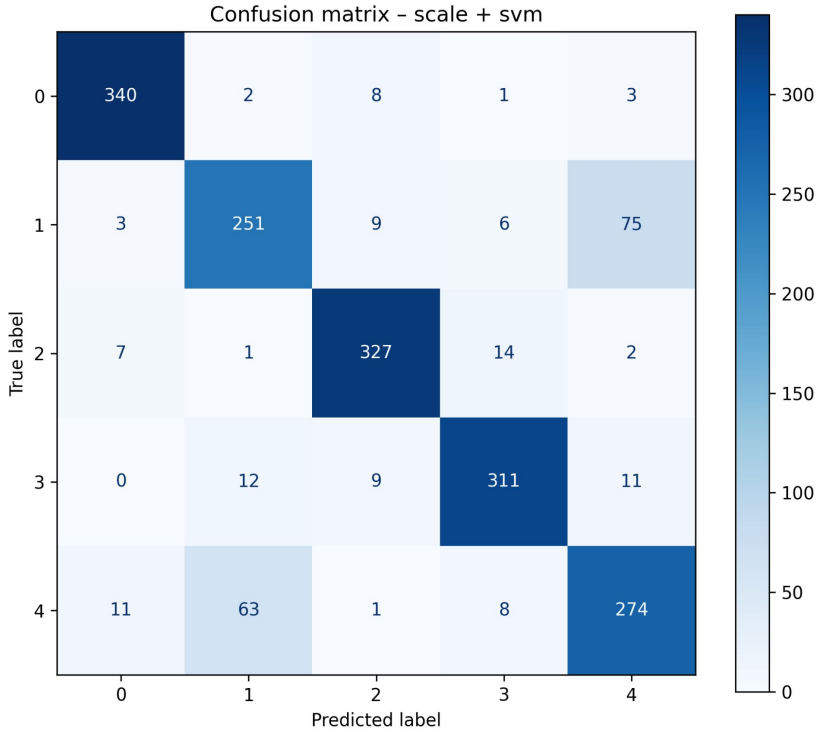
Problem Investigation

- Strong diagonal dominance, but systematic confusions remain
- Specific class pairs are repeatedly misclassified
- Overall accuracy hides structured class overlap

Classification report (best config):

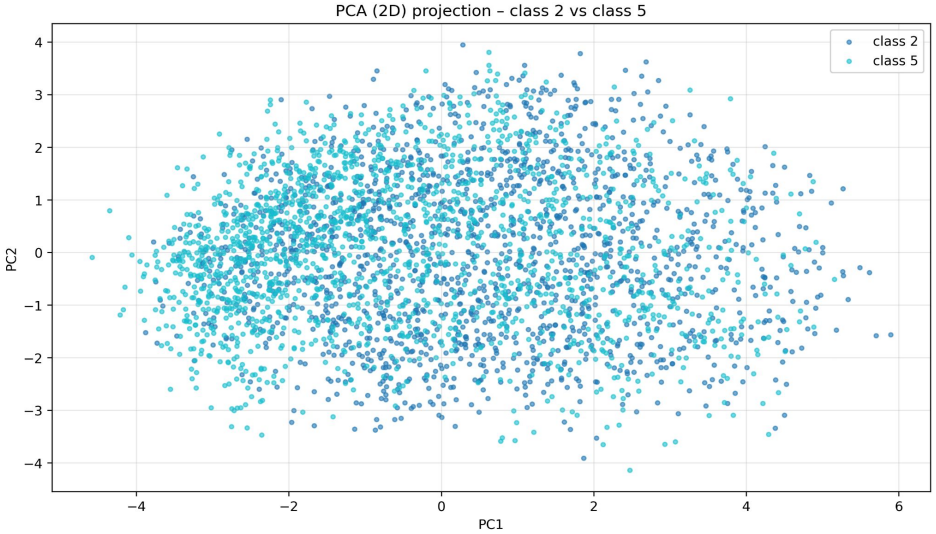
	precision	recall	f1-score	support
1	0.94	0.96	0.95	354
2	0.76	0.73	0.75	344
3	0.92	0.93	0.93	351
4	0.91	0.91	0.91	343
5	0.75	0.77	0.76	357
accuracy			0.86	1749
macro avg	0.86	0.86	0.86	1749
weighted avg	0.86	0.86	0.86	1749

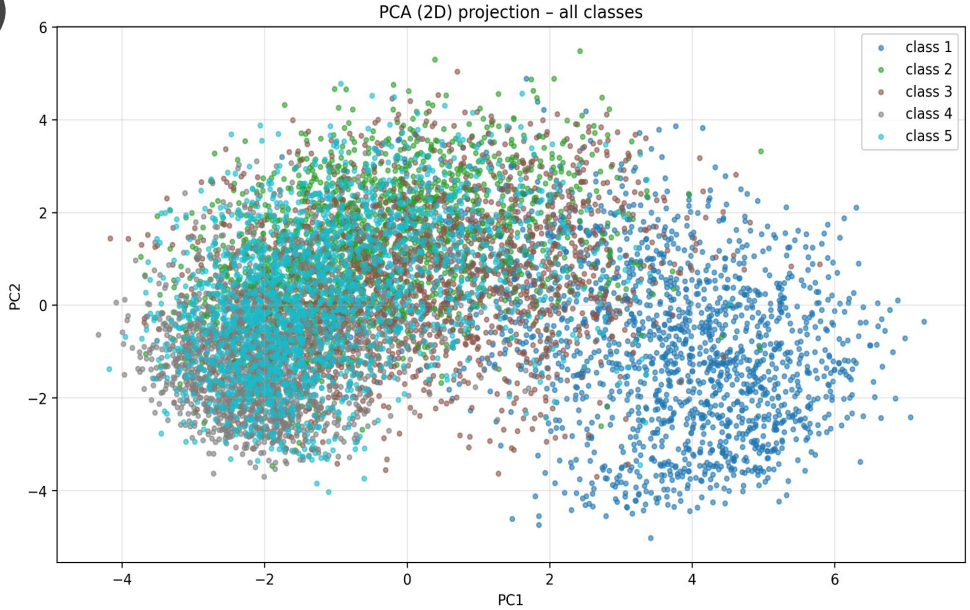
"We are doing well — but not equally well for all classes."



Problem Investigation (PCA)

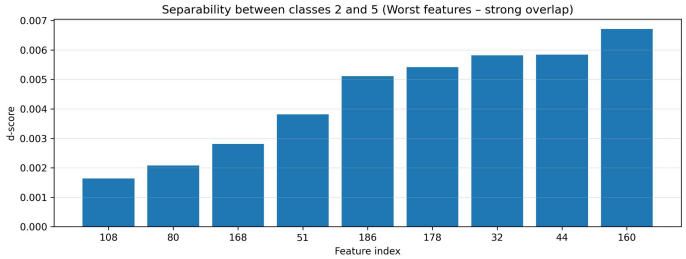
- Significant overlap between multiple classes
- No clear linear separation in low-dimensional projections
- Indicates intrinsic difficulty of the dataset



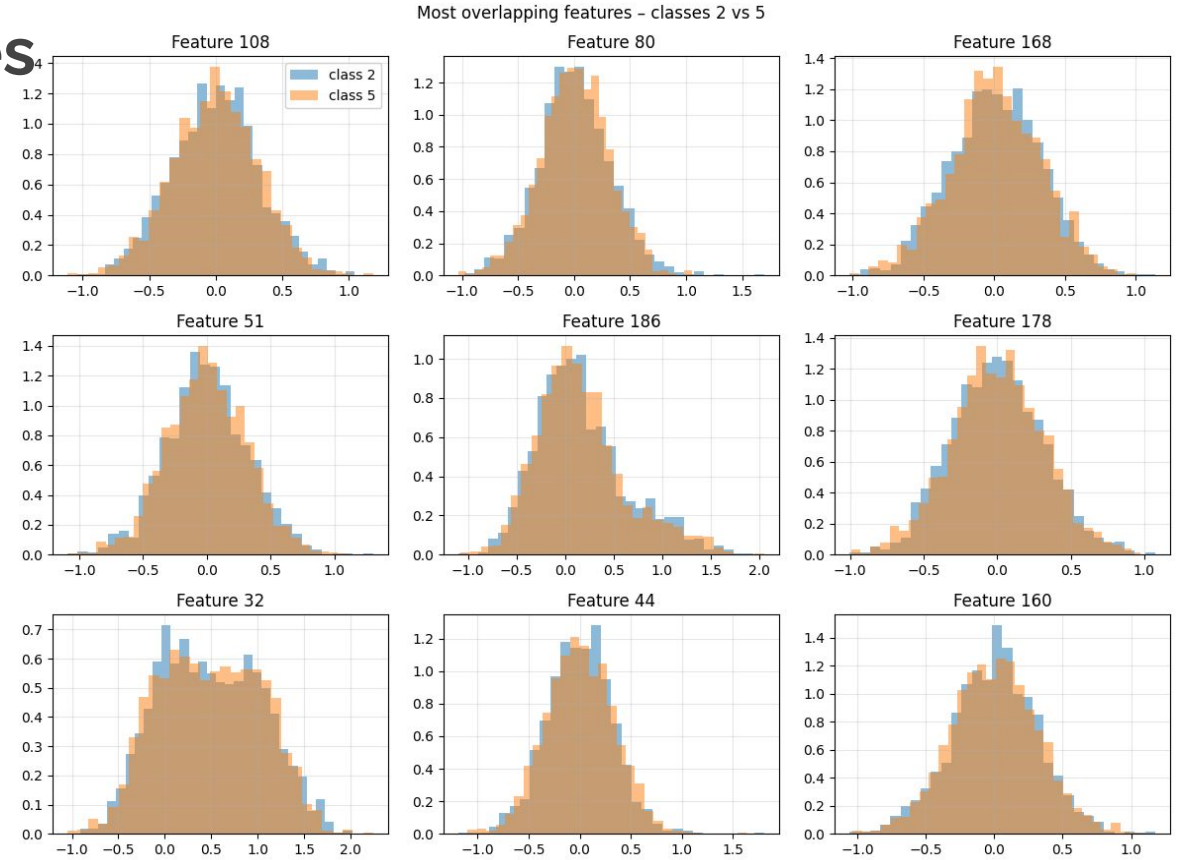


- Severe overlap between the two most confused classes
- Explains confusion patterns seen in the confusion matrices

Separability of classes



- Some features strongly discriminate between classes
- Several features exhibit almost complete overlap
- Classification difficulty is feature-driven



Problem Investigation

Interpretation

- Performance ceiling is caused by data overlap
- Classifier choice is not the primary limitation
- Better models alone cannot resolve intrinsic ambiguity

Tuning Strategy

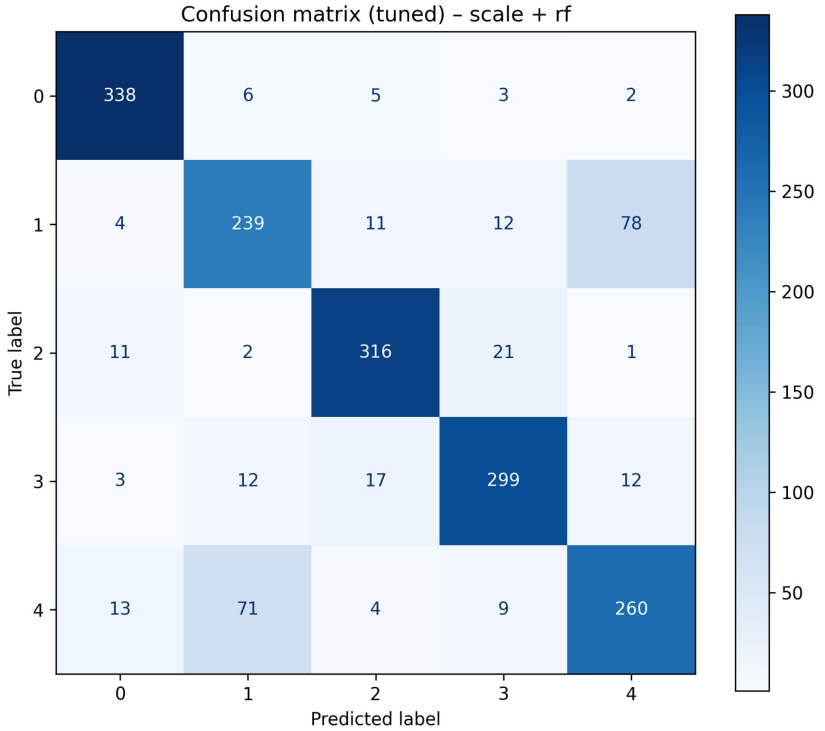
- Focus on the strongest baseline models
- Tune hyperparameters to reach the performance ceiling
- Expect incremental, not dramatic, improvements

Random Forest (tuned)

- Moderate improvements in overall balance
- Marginally enhances class-wise metrics
- Does not resolve the ambiguity

Classification report:

	precision	recall	f1-score	support
1	0.92	0.95	0.93	354
2	0.72	0.69	0.71	344
3	0.90	0.90	0.90	351
4	0.87	0.87	0.87	343
5	0.74	0.73	0.73	357
accuracy			0.83	1749
macro avg	0.83	0.83	0.83	1749
weighted avg	0.83	0.83	0.83	1749

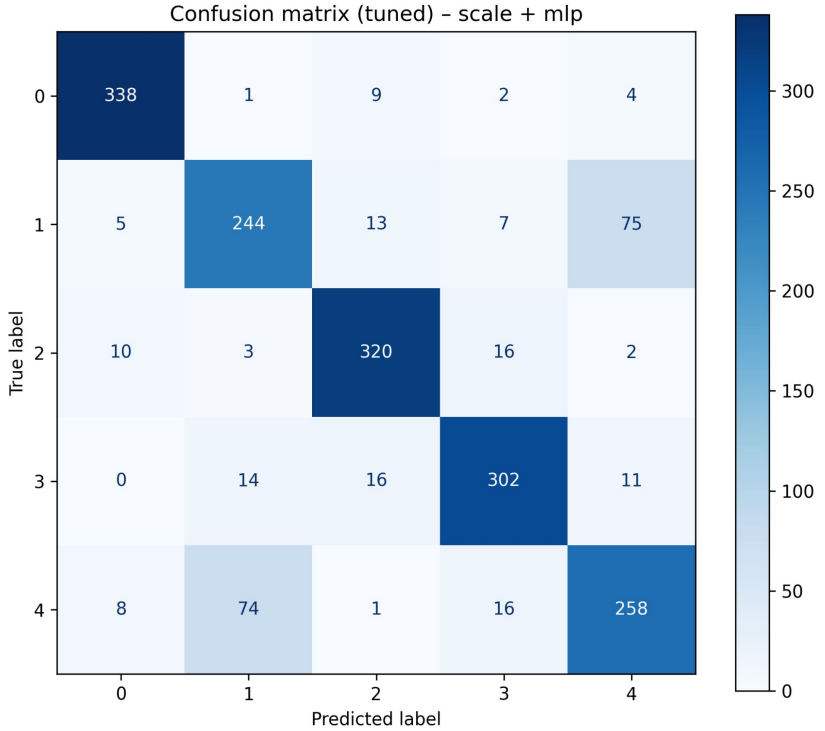


MLP Classifier (tuned)

- Moderate improvements in overall balance
- Marginally enhances class-wise metrics
- Does not resolve the ambiguity

Classification report:

	precision	recall	f1-score	support
1	0.94	0.95	0.95	354
2	0.73	0.71	0.72	344
3	0.89	0.91	0.90	351
4	0.88	0.88	0.88	343
5	0.74	0.72	0.73	357
accuracy			0.84	1749
macro avg	0.83	0.84	0.83	1749
weighted avg	0.83	0.84	0.84	1749

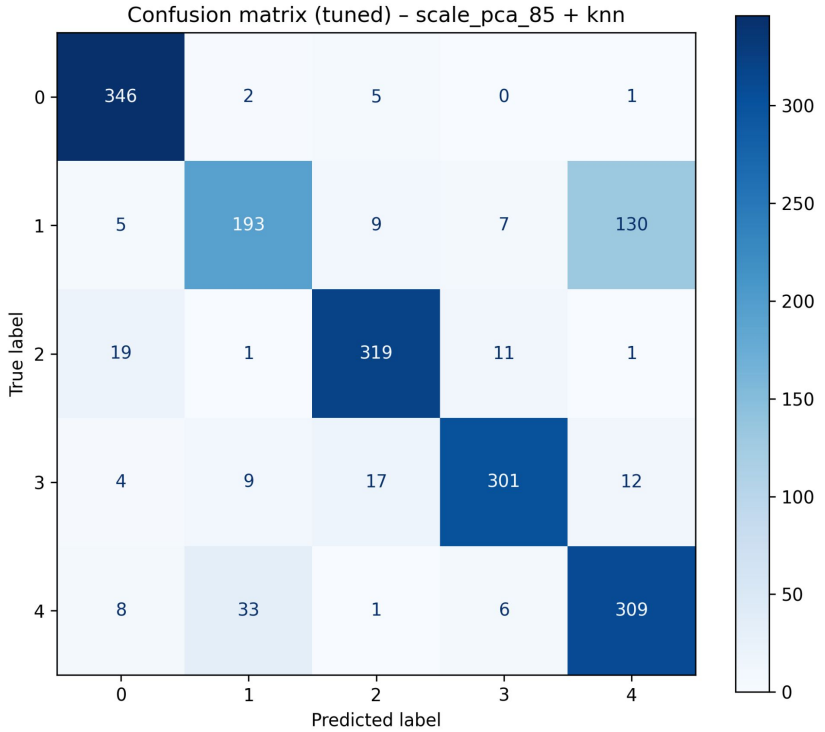


kNN Classifier (tuned)

- Performance ranking remains largely unchanged
- Recall for class 2 drops significantly
- Recall for class 5 increases

Classification report:

	precision	recall	f1-score	support
1	0.91	0.98	0.94	354
2	0.81	0.56	0.66	344
3	0.91	0.91	0.91	351
4	0.93	0.88	0.90	343
5	0.68	0.87	0.76	357
accuracy			0.84	1749
macro avg	0.85	0.84	0.84	1749
weighted avg	0.85	0.84	0.84	1749



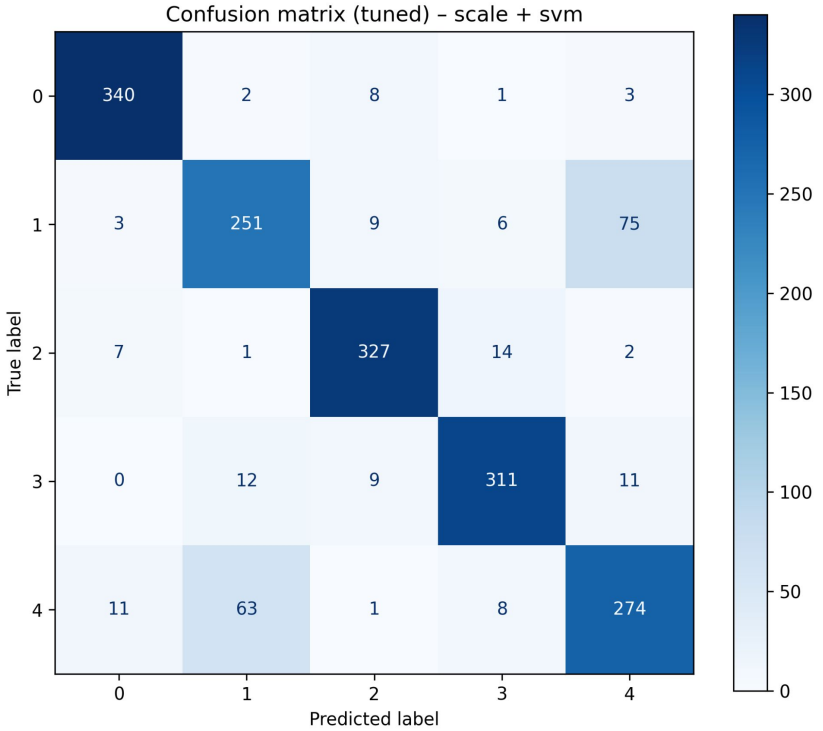
SVC (RBF) Classifier (tuned)

- **Best trade-off** between overall accuracy and class-wise balance.
- Improved discrimination without excessive overfitting
- Confusion between classes 2 and 5 persists

Classification report:

	precision	recall	f1-score	support
1	0.94	0.96	0.95	354
2	0.76	0.73	0.75	344
3	0.92	0.93	0.93	351
4	0.91	0.91	0.91	343
5	0.75	0.77	0.76	357
accuracy			0.86	1749
macro avg	0.86	0.86	0.86	1749
weighted avg	0.86	0.86	0.86	1749

“Nominated Model”



Final model & Conclusions

Final model

- Best-performing model: scale + SVM
- Balanced performance across all classes
- Used for final test-set prediction and submission

Conclusions

- Systematic experimental methodology is critical
- Visualization reveals limitations hidden by accuracy
- Data characteristics define achievable performance
- Further gains require better features or additional data

Thank you.