DS5020 Final Project

Om Rastogi

December 2025

1 Part I — Predictive Intelligence: Boston Housing Dataset

Code: `https://github.com/omrastogi/ds5020_linal_and_prob/blob/main/project/1-Boston_Housing_Regression.ipynb`

1.1 Purpose and Mathematical Theme

This section demonstrates how linear algebra enables prediction through linear regression and how probability quantifies uncertainty using a Gaussian residual model. The objective is to predict median housing prices from structured town-level attributes and interpret the prediction error statistically.

1.2 Dataset and Features

The dataset contains 14 variables: 13 input features and one target. The target variable is `MEDV`.

Feature	Description
CRIM	Per capita crime rate by town
ZN	Proportion of residential land zoned for large lots
INDUS	Non-retail business acres per town
CHAS	Charles River dummy variable (1 = bounds river)
NOX	Nitric oxides concentration (ppm)
RM	Avg. number of rooms per dwelling
AGE	% of units built prior to 1940
DIS	Weighted distances to employment centres
RAD	Accessibility index to radial highways
TAX	Property-tax rate per $10,000
PTRATIO	Pupil-teacher ratio by town
B	$1000(B_k - 0.63)^2$ (based on Black population proportion)
LSTAT	% lower-status population

Table 1: Summary of Boston Housing Dataset Features

1.3 Exploratory Data Analysis (Extra Work)

Before modeling, we examined the empirical distributions of all variables using histogram + KDE plots. This step provided a visual basis for choosing appropriate missing-value strategies and transformation methods for different feature groups.

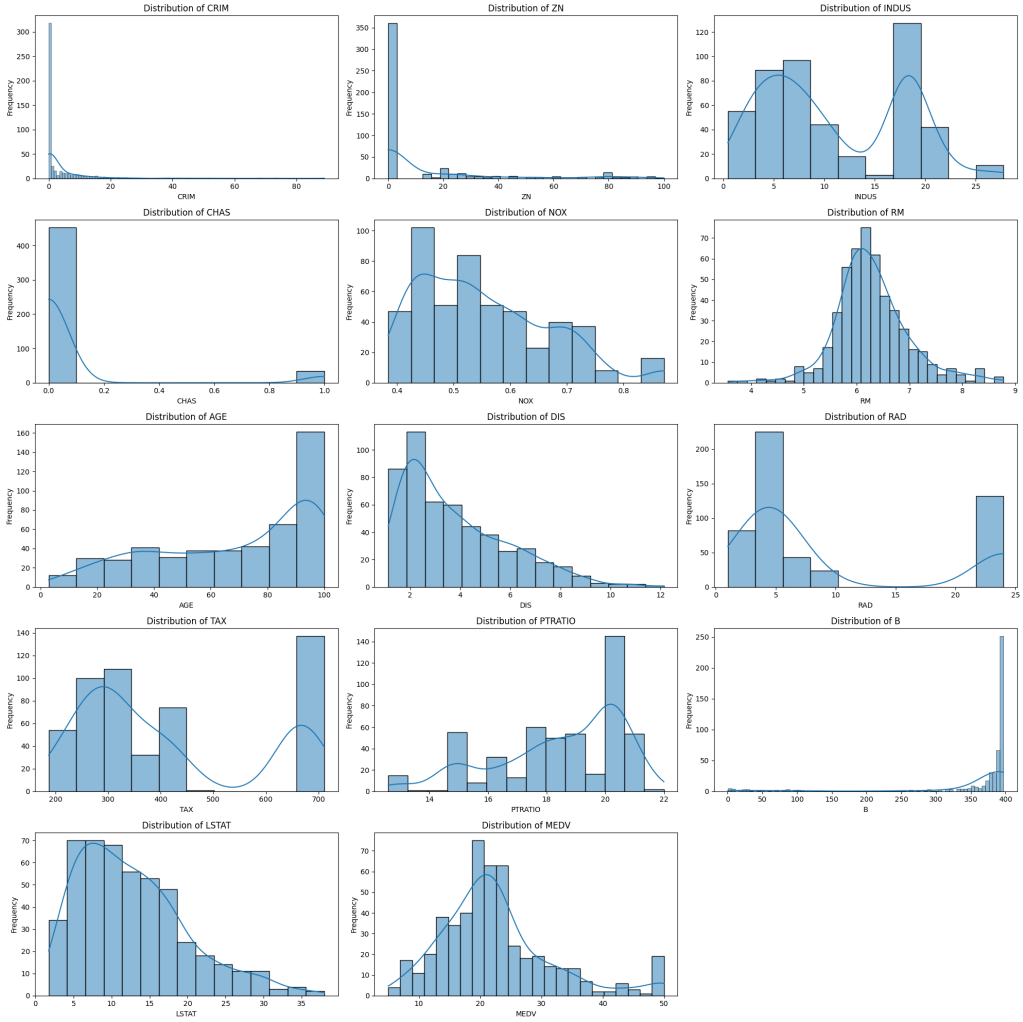

Figure 1: Distribution of all Boston Housing variables (histogram + KDE). This exploratory analysis motivated feature-specific imputation and transformation choices.

1.4 Handling Missing Values and Feature Transformations

- **Left-skewed:** `CRIM, ZN, INDUS, LSTAT, DIS, NOX`
- **Right-skewed:** `B, PTRATIO, AGE`
- **Near-normal:** `RM`
- **Bimodal:** `RAD, TAX` (kept as numeric + 2-cluster KMeans labels)
- **Binary:** `CHAS`

The preprocessing choices were:

- **Yeo–Johnson + StandardScaler** for skewed features
- **StandardScaler** for `RM`
- **RobustScaler** for raw `RAD, TAX`
- **Passthrough** for `CHAS` and KMeans labels

Adding KMeans labels for `RAD` and `TAX` provides a compact categorical proxy for their discrete structure while retaining the original numeric information.

1.5 Train–Test Split and Target Scaling

The data was split 80–20 into train and test sets. Features were transformed using the fitted preprocessing pipeline, and the target was standardized for numerical stability and consistent residual analysis. Let X_{train} and y_{train} denote the processed training inputs and targets.

1.6 Closed-Form Linear Regression

For numerical stability, we compute:

$$\hat{w} = X_{\text{train}}^{+} y_{\text{train}}.$$

Test predictions and residuals are:

$$\hat{y}_{\text{test}} = X_{\text{test}} \hat{w}, \qquad r_{\text{test}} = y_{\text{test}} - \hat{y}_{\text{test}}.$$

1.7 Gaussian Residual Findings

Residuals were summarized as:

$$\epsilon_{\text{train}} \sim \mathcal{N}(0.013,\ 0.480^2), \qquad \epsilon_{\text{test}} \sim \mathcal{N}(0.215,\ 0.521^2).$$

The positive test mean (0.215) indicates mild underprediction on the test set. The standard deviations (≈ 0.48 train, ≈ 0.52 test) represent the typical prediction error magnitude on the scaled target, with a slight increase on test reflecting generalization noise.

1.8 Model Evaluation Visualizations

1.8.1 True vs. Predicted Prices

We visualize predictive alignment using a true-versus-predicted scatter plot (scaled targets). A tighter clustering around the $y = x$ reference line indicates stronger predictive performance.

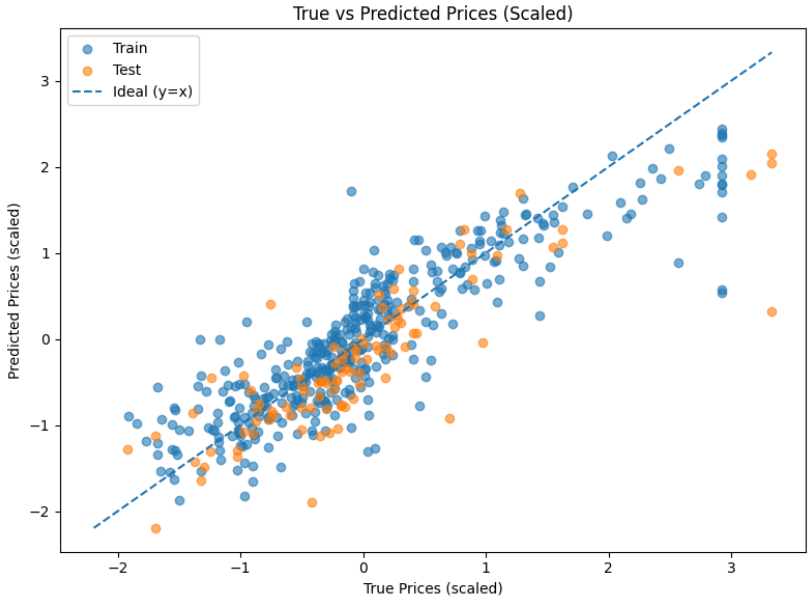

Figure 2: True vs. predicted prices (train and test, scaled). The dashed line denotes the ideal $y = x$ relationship.

1.8.2 Residual Distributions

We further visualize uncertainty by plotting the residual histograms for training and test sets.

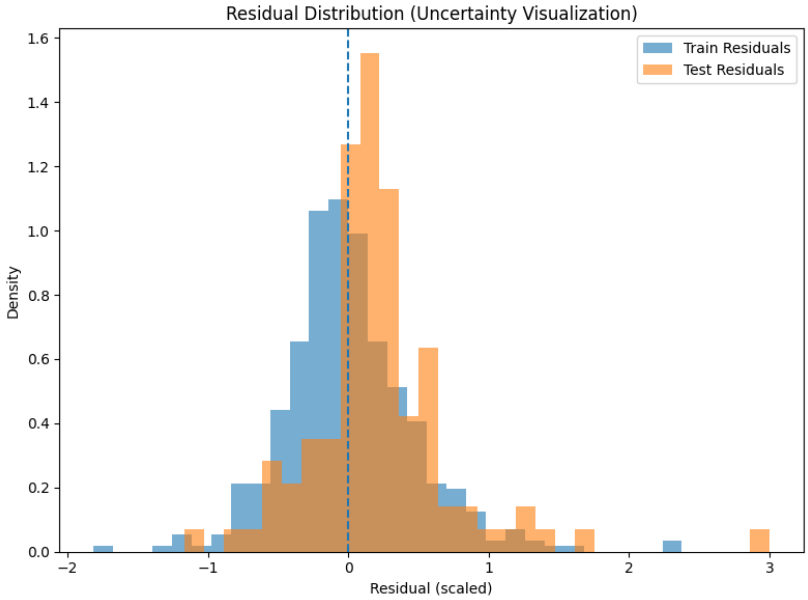

Figure 3: Residual distributions (scaled). This plot serves as an uncertainty visualization aligned with the Gaussian residual assumption.

2 Part II: Perceptual Intelligence (Digits Dataset)

Code: `https://github.com/omrastogi/ds5020_linal_and_prob/blob/main/project/2-Perceptual_Intelligence_(Digits_Dataset).ipynb`

2.1 Dataset

The Digits dataset contains grayscale handwritten digit images of size 8×8, flattened into 64-dimensional vectors:

$$X \in \mathbb{R}^{n \times 64}, \qquad y \in \{0, 1, \ldots, 9\}^n.$$

Eigendigits

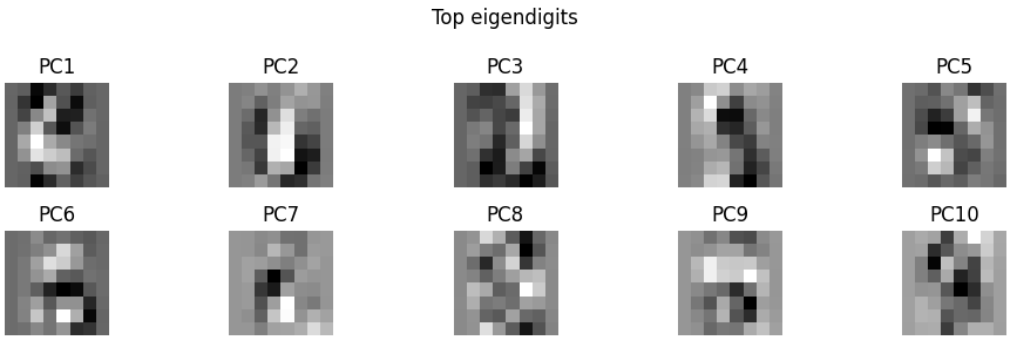

Figure 4: Top 10 eigendigits derived from the covariance eigenvectors.

2.2 Methodology

The PCA pipeline followed standard linear-algebra steps:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i, \qquad X_c = X - \mu,$$

$$C = \frac{1}{n-1} X_c^\top X_c \in \mathbb{R}^{64\times 64}, \qquad Cv_i = \lambda_i v_i,$$

where λ_i and v_i are sorted in descending order of explained variance. Using the top k eigenvectors $W_k = [v_1, \ldots, v_k]$,

$$Z = X_c W_k, \qquad \hat{X} = Z W_k^\top + \mu.$$

Reconstruction vs. k

Figure 5: Reconstructions for increasing k. Higher k restores finer digit details.

Probabilistic Classification

Naïve Bayes was trained on PCA features $z \in \mathbb{R}^k$:

$$\hat{y} = \arg\max_c P(c \mid z).$$

Number of PCs k	Naïve Bayes Accuracy
1	40.0%
5	83.7%
10	89.7%
20	93.3%
30	95.3%
40	95.1%
50	93.3%
64	89.3%

Table 2: Naïve Bayes accuracy on PCA-reduced features.

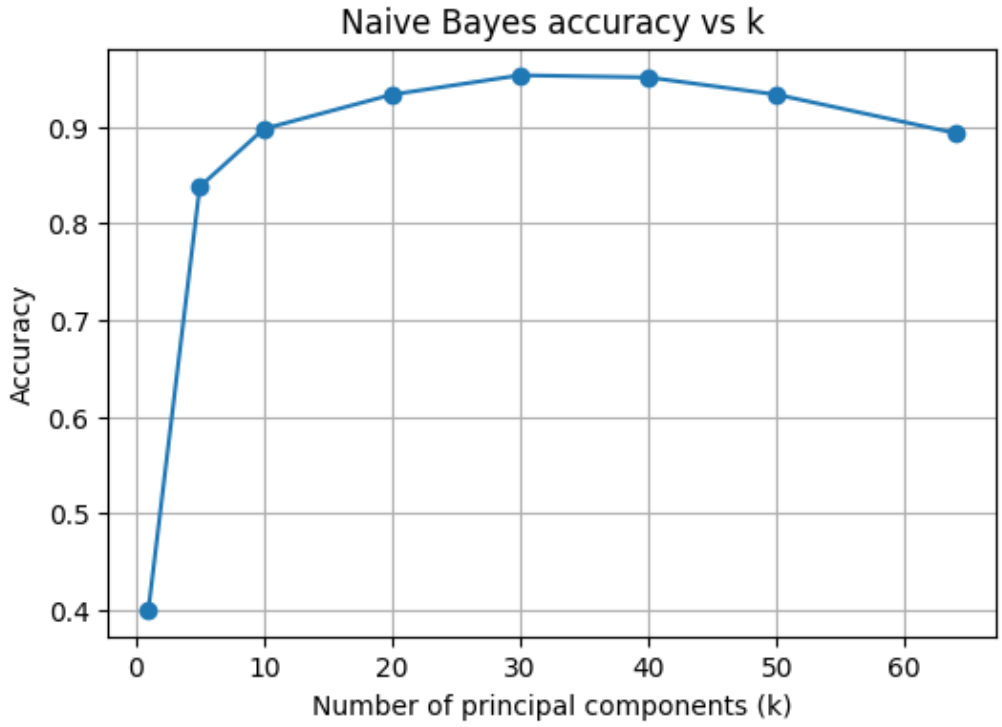

Figure 6: Accuracy vs. PCA dimensionality k.

2.3 Findings

- The top eigendigits capture dominant strokes and intensity patterns, acting as compact visual primitives.
- Reconstruction quality improves steadily with k, showing that principal components preserve meaningful structure.
- Classification peaks at moderate k, indicating PCA reduces redundancy while retaining discriminative information.

2.4 Limitations

- PCA is linear and may miss nonlinear digit manifolds.
- Extremely small k under-represents class detail; very large k can reintroduce noise.

3 Part III: Personalized Intelligence (MovieLens Dataset)

Code: `https://github.com/omrastogi/ds5020_linal_and_prob/blob/main/project/3-Personalized_Intelligence_(MovieLens_Dataset).ipynb` **Notebook:** `https://colab.research.google.com/drive/1NRp6QeTJJC7kPD0w3LotwVSphQdN8S`

3.1 Dataset and Preprocessing Strategy

MovieLens ratings are provided as triples:

$$(userId,\ movieId,\ rating).$$

The full user–movie matrix is highly sparse. To keep computation manageable and improve density, I filtered to the top N users and top N movies by rating counts, yielding a smaller, denser submatrix for SVD experiments.

3.2 Matrix Construction

From the training subset, I constructed:

$$R_{train} \in \mathbb{R}^{m \times n},$$

where m is the number of retained users and n is the number of retained movies. Entries in R_{train} represent observed ratings, while missing entries correspond to unrated user–movie pairs.

Because standard SVD requires a fully numeric matrix, I filled missing values using a simple and interpretable baseline:

$$R_{train}(i, j) \leftarrow \bar{r}_i,$$

where $\bar{r}_i$ denotes the mean rating given by user i in the training data. If a rare row-level edge case arises after filtering, a global mean is used as fallback. This imputation acts as a neutral prior that preserves individual rating bias without adding artificial cross-user structure.

3.3 Truncated SVD for Latent Preference Learning

The core mathematical model assumes that preferences are low-rank:

$$R_{train} \approx U_k \Sigma_k V_k^\top, \quad k \ll \min(m, n).$$

Here:

- $U_k \in \mathbb{R}^{m \times k}$ encodes **latent user preferences**,
- $V_k \in \mathbb{R}^{n \times k}$ encodes **latent movie attributes**,
- $\Sigma_k \in \mathbb{R}^{k \times k}$ scales the importance of each latent dimension.

Rather than computing full SVD, I used **TruncatedSVD** (randomized SVD), which is significantly faster for this setting. This aligns with the practical recommender assumption that only the strongest latent factors are needed to capture meaningful structure.

3.4 Predicting Missing Ratings

Once U_k, Σ_k, and V_k are obtained, the predicted rating matrix is:

$$\hat{R} = U_k \Sigma_k V_k^\top.$$

This yields a dense matrix where:

- observed entries approximate the training ratings,
- previously missing entries become **predicted ratings**.

Thus, "predict missing ratings" refers to estimating ratings for user–movie pairs that were NaN in the original pivoted matrix (i.e., unrated interactions).

3.5 Model Selection Using Validation RMSE

While training reconstruction error is guaranteed to decrease as k increases, this is not a reliable indicator of recommendation quality. Therefore, I selected k using a validation-driven metric:

$$\text{RMSE} = \sqrt{\frac{1}{|\mathcal{D}_{val}|} \sum_{(i,j,r)\in\mathcal{D}_{val}} \left(r - \hat{R}_{ij}\right)^2}.$$

I evaluated RMSE over a range of k values to observe the trade-off between underfitting and overfitting.

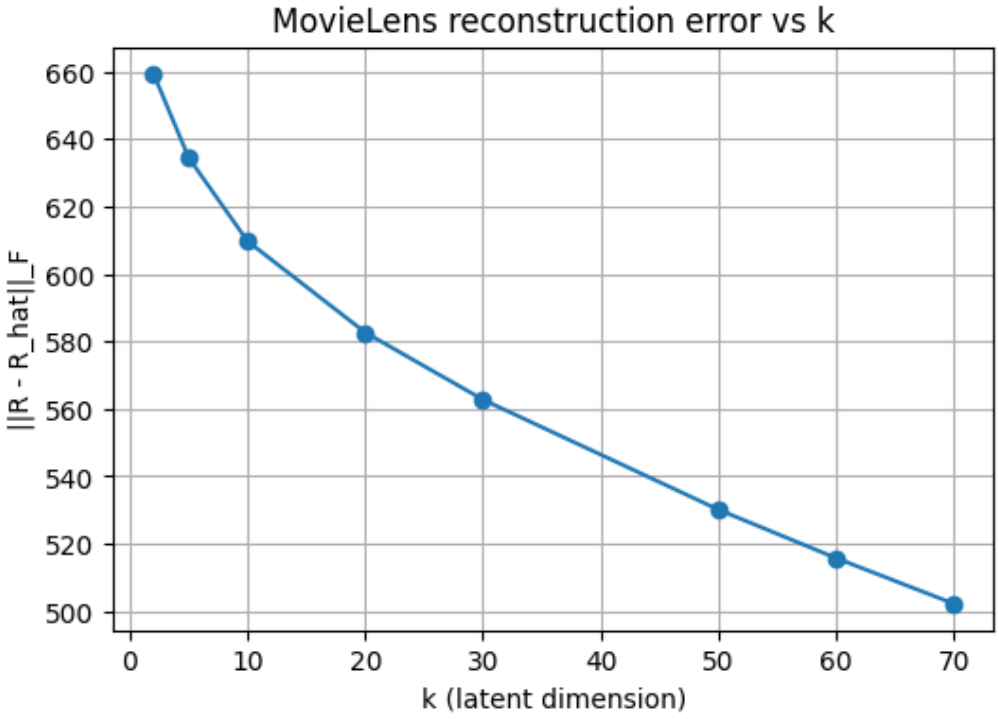

Figure 7: Validation RMSE vs. latent dimension k. The optimal k is the value that minimizes RMSE.

3.6 Findings and Interpretation

The validation curve showed that **small k values achieved the lowest RMSE**, while larger k values degraded performance. This behavior is consistent with the following interpretation:

- **Low-rank structure is real:** A small number of latent dimensions appears sufficient to capture major preference patterns in this subset.
- **Capacity vs. generalization:** Increasing k improves reconstruction of the filled training matrix, but begins fitting noise and imputation artifacts rather than true preference signals.
- **Why validation is essential:** Without a held-out evaluation set, one could incorrectly conclude that larger k is always better due to monotonically decreasing training reconstruction error.

In other words, this part explicitly demonstrates a core principle of personalization: **the best representation is not the one that reconstructs past behavior most perfectly, but the one that generalizes best to unseen interactions.**

3.7 Latent Space Visualization (Optional)

To interpret learned structure geometrically, I optionally visualized users or movies in 2D by setting $k = 2$. This yields low-dimensional embeddings that can reveal preference clusters or outliers.

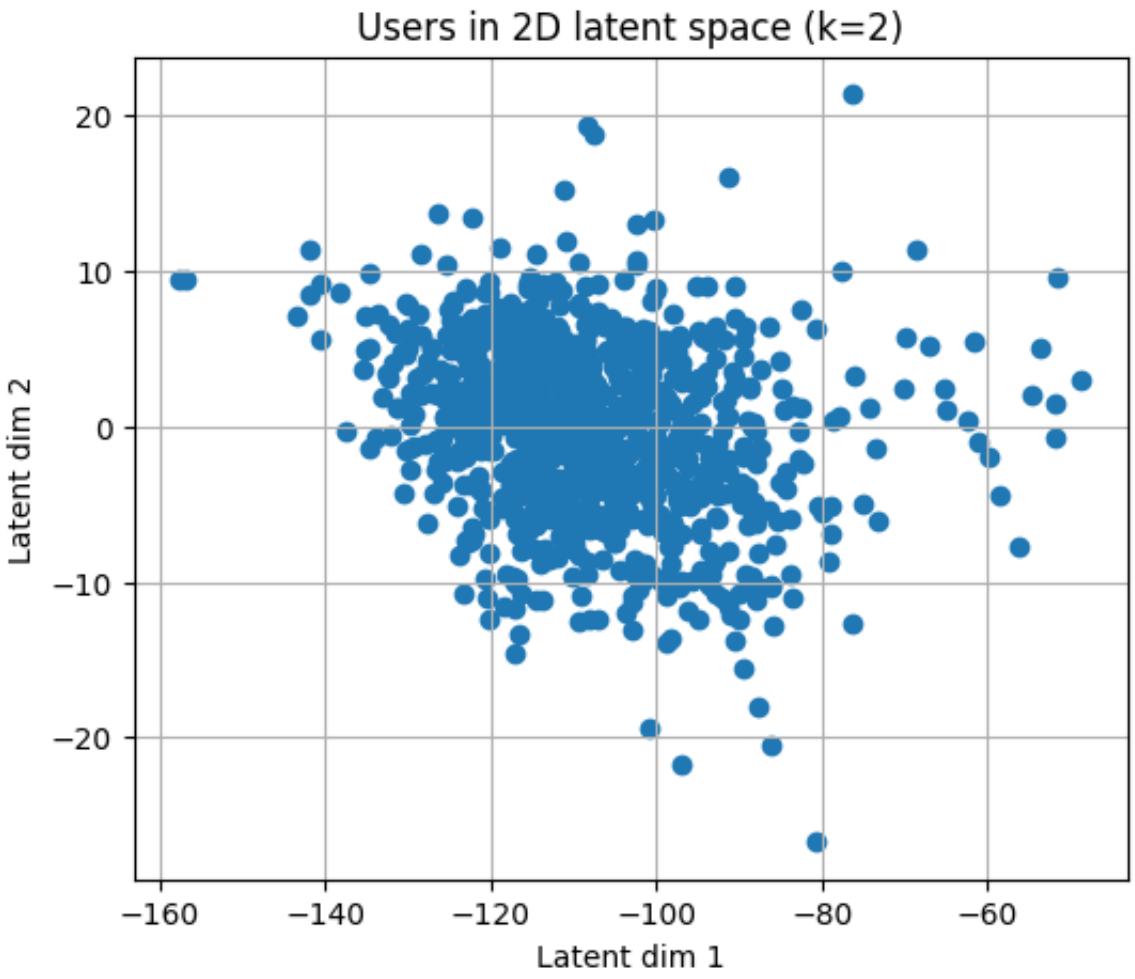

Figure 8: Users embedded into a 2D latent space inferred by truncated SVD.

3.8 Limitations of the Approach

This notebook implementation focuses on clarity and mathematical interpretability, but it comes with known limitations:

- **Cold-start problem:** Pure SVD cannot recommend for entirely new users or new movies without additional mechanisms or retraining.
- **Imputation bias:** Filling missing entries with user means is simple and fast, but can influence learned factors, especially at large k.
- **Subset dependence:** The optimal k and error behavior may change depending on the density and composition of the chosen user–movie subset.

These limitations reflect why production-grade recommenders often extend this foundation using bias terms, iterative matrix factorization (SGD/ALS), metadata-driven hybrids, or deep models.