Journal of Computer Science and Frontier Technologies
ISSN Print: 3104-4204
ISSN Online: 3104-4212
CD ISSN: 3104-4220
https://doi.org/10.63313/JCSFT.9108

A Multi-Model AI Pipeline for Student Risk Prediction, Profiling, Knowledge Tracing, and Resource Recommendation in Intelligent Learning Systems

Rui Jiang*. Yuan Jiang. Qiaoling Cheng and GuiHong Wu
Faculty of Artificial Intelligence, Baise University, Baise, Guengxi, 533000, China
Corresponding author: myzjyuan@bsuc.edu.cn

How to cite this paper: Jiang, R., Jiang, Y., Cheng, Q., & Wu, G. (2026). A multi-model AI pipeline for student risk prediction, profiling, knowledge tracing, and resource recommendation in intelligent learning systems. Journal of Computer Science and Frontier Technologies, 4(3), 59–71. ISSN Print: 3104-4204, ISSN Online: 3104-4212.
https://doi.org/10.63313/JCSFT.9108
Published: 2026-10-30

Copyright © 2026 by author(s) and Erytis Publishing Limited.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Abstract

How can a learning platform turn raw click streams and gradebook entries into timely, personalized support for students? This question motivated the work reported here. We describe an end-to-end AI training pipeline for an Intelligent Learning Assistance System (ILAS) that we built to support undergraduate Java programming instruction. Rather than betting on a single modeling paradigm, our pipeline brings together four complementary approaches: XGBoost for academic risk prediction, K-Means for student profiling, Bayesian Knowledge Tracing for knowledge-state estimation, and BERT for semantic encoding of learning resources. Working with behavioral data from 204 students across four classes, we extract a 23-dimensional feature vector from 11 heterogeneous Excel sheets. The XGBoost classifier reaches 90.2% accuracy and 0.993 AUC (one-vs-rest) in a three-class risk task. K-Means clustering, using 11 behavioral features, identifies four interpretable learner profiles (A through D) with a silhouette score of 0.398. We train 15 BKT models via the Baum-Welch EM algorithm and produce a 204 × 15 mastery matrix. A pre-trained bert-base-chinese model encodes 15 learning resources into 768-dimensional vectors using both CLS and mean pooling. Finally, a grid search over 45 weight combinations fuses BKT recall, BERT semantic similarity, and collaborative filtering into a recommendation strategy that achieves Precision@5 of 1.0 in offline evaluation. The entire pipeline completes in about 70 seconds on a GPU and outputs 16 serialized artifacts ready for production use.

Keywords

Intelligent Learning Assistance System; Learning Analytics; XGBoost Risk Prediction; K-Means Student Profiling; Bayesian Knowledge Tracing; BERT-based Resource Recommendation

1. Introduction

Over the last decade, learning analytics and educational data mining have moved from specialty concerns to mainstream infrastructure in online and blended education [1]. Yet the gap between the rhetoric of personalization and what most deployed systems actually deliver remains wide. A major reason is that the learning process is inherently multi-dimensional: we might care about whether a student is at risk of failing, what kind of learner she is, which concepts she has actually mastered, and which resources would best fill her current gaps. These different questions do not map cleanly onto a single model or loss function [2].

Designing a practically useful ILAS therefore forces us to think in terms of a small ecosystem of models, each optimized for a different slice of the problem. In our view, such a system needs at least four capabilities. First, it should flag at-risk students early enough that instructors can intervene. Second, it should discover latent learner profiles — not as rigid labels, but as orientation points for differentiated pedagogy. Third, it needs fine-grained knowledge tracing that follows mastery state across individual topics over time. Fourth, it needs content-based resource recommendation that matches materials to diagnosed gaps. The architecture described in this paper is our attempt to integrate all four into a single, reproducible training pipeline.

The four modeling approaches we combine are:

- XGBoost [3] for supervised academic risk prediction (Section 4)
- K-Means clustering [4] for unsupervised student persona discovery (Section 5)
- Bayesian Knowledge Tracing (BKT) [5] for cognitive knowledge state estimation (Section 6)
- BERT [6] for semantic encoding of learning resources (Section 7)

Our aim was to start from the Excel spreadsheets that instructors routinely export from a university learning management system, and to convert them into a suite of trained models without manual cleaning. The pipeline parses heterogeneous multi-level headers, extracts features, trains the four models, evaluates them, and serializes the resulting artifacts. A recommendation module (Section 9) then fuses the outputs of the three knowledge-aware models. Sections 3 and 4 cover data acquisition and feature engineering; Sections 5-8 present each model; Section 9 discusses recommendation fusion; Section 10 reports overall results and deployment artifacts; and Section 11 reflects on limitations and future work.

2. System Architecture Overview

Figure 1 shows the pipeline at a glance. It is mostly sequential because later stages depend on outputs from earlier ones. Feature engineering sits at the center: every model reads the same 23-dimensional feature matrix. The BKT and BERT components additionally feed the mastery matrix and resource vectors into the recommendation fusion module.

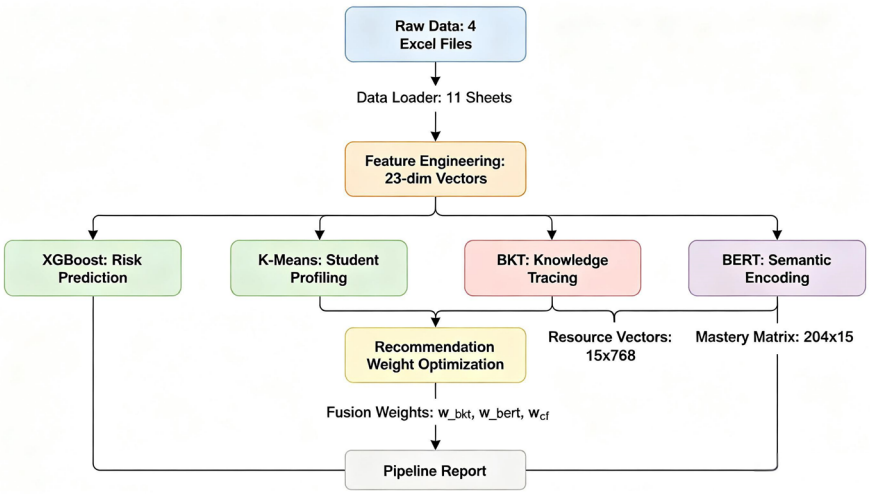


Figure 1. System Architecture of the ILAS AI Model Training Pipeline

3. Data Acquisition

3.1. Data Sources

The dataset comes from four Excel workbooks (.xlsx), each belonging to a different class of roughly 50 students enrolled in an Object-Oriented Programming (Java) course at a Chinese university. The files were exported directly from the institutional learning management platform and stored with filenames of the form {course_code}_{class_id}_{class_name}.xlsx. Each workbook contains 11 worksheets, and the structure of these worksheets is summarized in Table 1.

Table 1. Data sources and their header structures.

#	Sheet Name	Header Structure
1	Composite Score	Simple (1-level)
2	Learning Progress	Simple (1-level)
3	Chapter Access	Simple (1-level)
4	Task Completion	3-level
5	Video Watch	2-level
6	Discussion	Simple (1-level)
7	Homework	2-level
8	Exam	2-level
9	Quiz	2-level
10	Attendance	2-level
11	Weighted Scores	Simple (1-level)

3.2. Header Parsing Strategy

The worksheets do not share a uniform header structure. Some use a single header row, others use two-level hierarchical headers, and one uses three levels. We therefore built a parser that detects and handles all three cases programmatically, rather than manually normalizing the files. The three parsing modes are:

Simple (1-level). Rows where row index 2 contains the column headers and data begin at row index 3. This applies to Composite Score, Learning Progress, Chapter Access, Discussion, and Weighted Scores.

Two-level. The Video Watch, Homework, Exam, Quiz, and Attendance sheets use rows 2-3 as hierarchical headers. We forward-fill empty cells in the top-level header and concatenate the levels with underscores — for example, 'Chapter 1_Score'.

Three-level. Only the Task Completion sheet reaches this depth, with rows 2-4 forming a {chapter}_{section}_{metric} hierarchy. The same concatenation rule extends naturally to three levels.

After parsing the headers, we clean every sheet by removing trailing all-NaN rows and columns, disambiguating duplicate column names with numeric suffixes, and concatenating the four class files row-wise. We append source_file and class_name columns to preserve provenance. Several fields required special handling: fraction values such as '9/226', duration strings like '93.2 min', and timestamps in mixed formats. The result is a merged dataset of 204 students.

4. Feature Engineering

From the merged data we construct a 23-dimensional feature vector for each student, labeled F1 through F23. These features are drawn from nine of the eleven source sheets and capture several distinct facets of learning behavior. Figure 2 maps each feature group back to its source sheet.

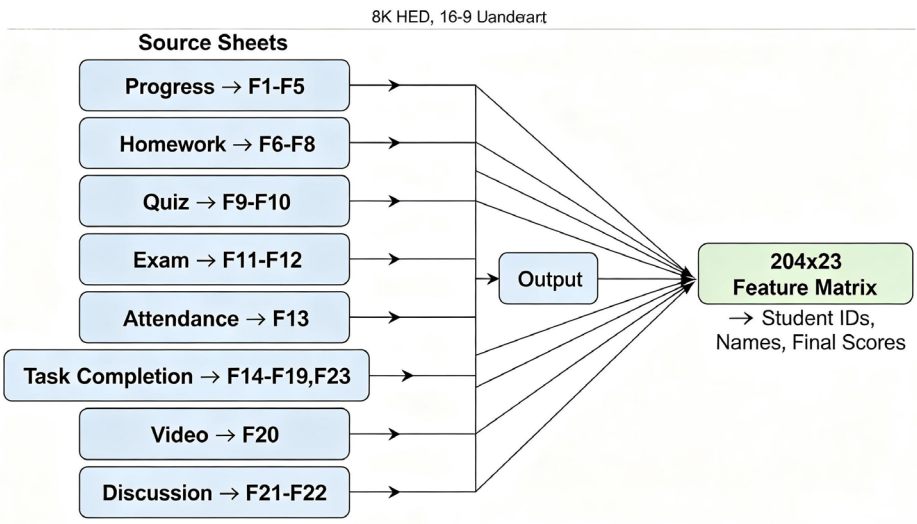


Figure 2. Feature Engineering Pipeline: From 11 Sheets to 23 Features

4.1. Learning Progress Features (F1-F5)

From the Learning Progress sheet we derive five features: F1 (task_completion_rate), the proportion of tasks completed; F2 (chapter_access_count), a raw count of chapter views; F3 (video_completion_rate), the ratio of video watched to total video

length; F4 (video_watch_duration_min), total viewing time in minutes; and F5 (discussion_count), the number of discussion posts.

4.2. Homework Features (F6-F8)

The Homework sheet provides per-assignment scores, submission times, and status fields. We aggregate these into F6 (hw_avg_score), the mean homework score; F7 (hw_completion_rate), the fraction of assignments submitted; and F8 (hw_late_submission_rate), computed by comparing each submission time to the median submission time for that assignment, with submissions more than 48 hours after the median counted as late.

4.3. Quiz and Exam Features (F9-F12)

Quizzes contribute F9 (quiz_avg_score) and F10 (quiz_completion_rate); exams contribute F11 (exam_avg_score) and F12 (exam_participation_rate). Missing exam scores are treated as zero, effectively marking the student as a non-participant.

4.4. Attendance Feature (F13)

F13 (attendance_rate) comes from the Attendance sheet. The platform distinguishes several status categories on the Chinese-language interface: signed in, absent, late, personal leave, sick leave, and not signed. We count only sessions with a valid status value in the denominator, and define attendance as the proportion of sessions where the student was marked as signed in.

4.5. Temporal Learning Behavior Features (F14-F19, F23)

Because the Task Completion sheet records a timestamp for every task submission, it is the richest source of temporal behavior. We extract several features from these timestamps. F14 (early_task_ratio) and F15 (late_task_ratio) compare submission dates to April 1 and June 1, 2025, revealing procrastination. F16 (morning_study_ratio) and F17 (night_study_ratio) count submissions during 8:00-12:00 and 20:00-24:00, respectively. F18 (study_frequency) is the number of distinct study dates divided by the 113-day semester. F19 (study_burst_index), defined as the maximum daily task count divided by the mean daily task count, captures cramming: high values indicate that a student compresses work into a few intense sessions. F23 (knowledge_coverage) counts the number of distinct task groups the student has engaged with, giving a rough measure of breadth.

4.6. Video and Discussion Features (F20-F22)

F20 (video_rewatch_ratio) comes from the Video Watch sheet: we count the proportion of videos whose rewatch ratio exceeds 1.0. The Discussion sheet yields F21 (discussion_reply_ratio) and F22 (received_likes), which capture the social dimension of engagement.

4.7. Feature Assembly

The output is a 204 × 23 feature matrix X, together with three aligned arrays — student_ids, student_names, and final_scores (the composite course score). Missing values are filled with zeros. This single matrix feeds every downstream model.

5. XGBoost Academic Risk Prediction

5.1. Task Definition

The first model addresses a practical question: which students are at risk of performing poorly, and how early can we identify them? We frame this as a three-class supervised classification task using the 23-dimensional feature vectors. Labels are derived from final scores: Low risk (≥75, label 0), Medium risk (60-75, label 1), and High risk (<60 or missing, label 2). Missing scores are treated as High risk, a conservative choice that biases the model toward intervention.

5.2. Model Configuration

We use the XGBoost implementation of gradient-boosted trees [3] with objective multi:softprob for three-class probability output. We set max_depth=5, learning_rate=0.05, and n_estimators=200. To control overfitting on a small dataset, we apply subsample=0.8 and colsample_bytree=0.8, set min_child_weight=3, and add L1 (reg_alpha=0.1) and L2 (reg_lambda=1.0) regularization. Early stopping with a patience of 20 rounds is applied, and training is GPU-accelerated via CUDA.

5.3. Training Procedure

We split the data 80/20 using stratified random sampling (random_state=42) so that the class distribution is preserved in both partitions. A StandardScaler is fitted on the training set only and applied to both splits. We report both held-out test performance and 5-fold stratified cross-validation on the training set.

5.4. Results

On the held-out test set the model reaches 90.24% accuracy and 0.993 AUC under the one-vs-rest scheme. Cross-validation yields 84.64% accuracy with a standard deviation of 9.31%; the relatively high variance is consistent with the modest sample size. The five most important features by gain are exam_participation_rate (0.119), hw_completion_rate (0.108), early_task_ratio (0.100), hw_avg_score (0.097), and task_completion_rate (0.082). The dominance of participation and completion features over raw ability proxies is striking: it suggests that behavioral engagement is a stronger predictor of success than prior achievement alone, in line with established work on student engagement [7]. Practically, this means the model can flag at-risk students before summative assessments reveal problems.

6. K-Means Student Profiling

6.1. Task Definition

Knowing that a student is at risk does not, by itself, tell us what kind of intervention would help. For that, we turn to K-Means clustering to discover latent learner profiles from behavioral data. We deliberately exclude score-related features (F6, F9, F11) so the clusters reflect how students learn rather than how well they currently perform. The 11 features used are task_completion_rate, chapter_access_count, video_watch_duration_min, discussion_count, hw_late_submission_rate, quiz_completion_rate, attendance_rate, morning_study_ratio, night_study_ratio, study_frequency, and study_burst_index.

6.2. Optimal K Selection

We run the Hartigan-Wong algorithm [4] with k-means++ initialization for K = 2 through 7, scoring each solution with the silhouette coefficient. The results are K=2: 0.443, K=3: 0.347, K=4: 0.398, K=5: 0.437, K=6: 0.436, K=7: 0.413. Although K=2 and K=5 score slightly higher, we choose K=4 because it maps cleanly onto the four intervention categories that instructors at our partner institution already use (A/B/C/D). K=2 collapses too much useful variation, and K=5 creates an extra cluster without a clear pedagogical meaning.

6.3. Profile Labeling

After clustering, we sort the four groups by mean final score and assign labels A through D with descriptive Chinese names. The resulting profiles are shown in Table 2.

Table 2. K-Means student profile characteristics.

Profile	Label	Students	Mean Score	Attendance	Task Completion
A	Excellent & Self-Disciplined	35	83.3	95.5%	95.5%
B	Hardworking, Needs Breakthrough	48	75.1	97.7%	33.6%
C	Fluctuating Potential	12	67.7	97.8%	18.2%
D	Requires Focused Attention	109	64.5	91.0%	1.1%

Several patterns in Table 2 are worth noting. Profile B is the most interesting from an intervention standpoint: these 48 students attend almost every class (97.7%) but complete only 33.6% of tasks. They are present but not self-directed — a pattern of untapped potential. Profile D, the largest group (109 of 204, or 53.4%), shows near-zero task completion (1.1%) and the lowest attendance (91.0%), indicating broad disengagement that likely requires institutional rather than classroom-level action. For visualization, we also run PCA with two components, which together explain 59.8% of the variance (PC1: 48.3%, PC2: 11.5%).

7. Bayesian Knowledge Tracing

7.1. Model Formulation

While the XGBoost and K-Means models operate at the student level, Bayesian Knowledge Tracing (BKT) [5] zooms in to the level of individual knowledge components. The model treats a student's mastery of a given knowledge point as a latent binary state that evolves over time. At each time step t, the student is either in a mastered state (Lt = 1) or an unmastered state (Lt = 0). What we observe, however, is not the state itself but a binary outcome Ot — whether the student answered a relevant assessment item correctly. The model is governed by four parameters:

- P(L0) — initial probability of mastery
- P(T) — probability of learning (transitioning from not-mastered to mastered)
- P(G) — probability of guessing correctly when not mastered
- P(S) — probability of slipping (answering incorrectly) when mastered

We adopt the standard no-forgetting assumption: P(L_{t+1} = 1 | L_t = 1) = 1. This is a simplification, but one that we consider defensible for a single-semester programming course where knowledge is largely cumulative [8]. Under this assumption, the emission probabilities take a simple form: a correct answer from an unmastered state occurs with probability P(G) (a guess), and an incorrect answer from a mastered state occurs with probability P(S) (a slip).

7.2. Training: Baum-Welch EM Algorithm

We estimate the parameters using the Baum-Welch algorithm [9], the standard EM procedure for Hidden Markov Models. Figure 3 illustrates the forward-backward computation at the heart of the E-step.

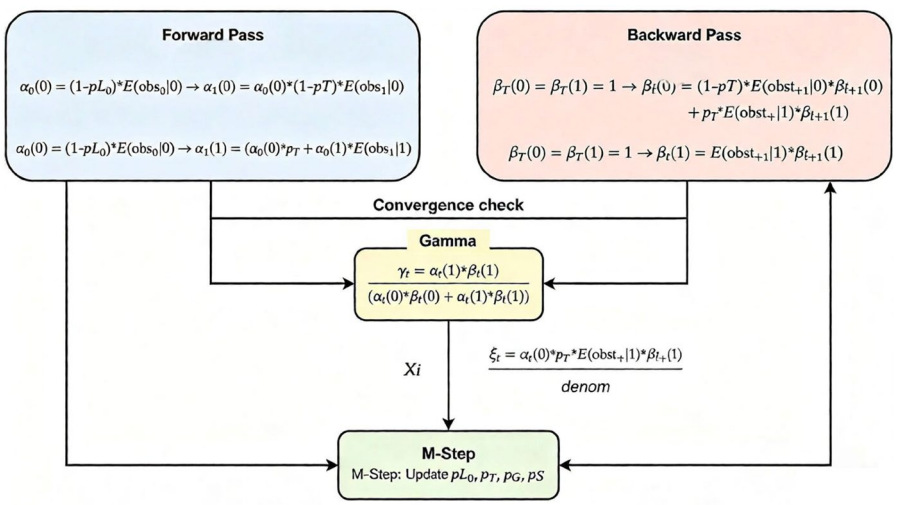


Figure 3. BKT Forward-Backward Algorithm for a Single Student's Response Sequence

In the E-step, the forward-backward algorithm computes two quantities for each student's response sequence: the smoothed posterior probability of mastery at each

time step, $\gamma t = P(Lt = 1 \mid O1:T)$, and the expected number of learn transitions, $\xi t = P(Lt = 0, Lt+1 = 1 \mid O1:T)$. The M-step then updates the parameters using the expected sufficient statistics aggregated across all students: P(L0) is set to the mean of γ0; P(T) to $\Sigma\xi t / \Sigma(1 - \gamma t)$; P(G) to $\Sigma(1 - \gamma t) \cdot I(Ot = 1) / \Sigma(1 - \gamma t)$; and P(S) to $\Sigma\gamma t \cdot I(Ot = 0) / \Sigma\gamma t$. We run the EM procedure for up to 100 iterations with a convergence tolerance of 10^{-4}.

7.3. Response Sequence Construction

We construct binary response sequences by thresholding quiz and homework scores at 60: correct (1) if the score is at least 60, incorrect (0) otherwise. Columns are mapped to knowledge points by keyword matching. For example, quiz columns whose Chinese names contain the terms for 'inheritance' or 'polymorphism' map to K08 (Inheritance & Polymorphism); homework columns labeled Experiment 1 through Experiment 15 map sequentially to K02-K14. The 15 knowledge points span the Java curriculum from language overview (K01) to advanced features (K15).

7.4. Results

The 15 BKT models converge to noticeably different mastery estimates. K08 (Inheritance & Polymorphism) reaches 0.999, while K09 (Inter-class Relationships) and K15 (Advanced Features) both sit at 0.300 — a value we interpret as evidence of sparse assessment coverage rather than genuine difficulty. K04 (Flow Control), at 0.637, appears to be a topic that students genuinely struggle with. Pelánek [10] has emphasized that BKT estimates are sensitive to the density and quality of assessment items, and our results confirm this caution: topics with thin coverage should not be over-interpreted. The full 204 × 15 mastery matrix is saved as mastery_matrix.csv and used directly by the recommendation module.

8. BERT Semantic Encoding

8.1. Task Definition

The fourth model addresses a different question: how semantically similar are two learning resources? To answer this, we encode each of the 15 resources (one per knowledge point) into a dense vector space where proximity corresponds to meaning. Each resource contains a title and a descriptive body; for example, the K07 resource covers classes, objects, constructors, encapsulation, access modifiers, and design patterns in roughly 120 tokens.

8.2. Encoding Pipeline

We use the bert-base-chinese checkpoint [6], a 12-layer Transformer with 768 hidden dimensions, 12 attention heads, and about 110 million parameters. We concatenate title and body, tokenize with a maximum length of 512 tokens, and extract two representations:

- CLS Pooling: The hidden state of the [CLS] token at the final layer serves as the sentence representation. This is the standard approach used in BERT for classification tasks [6].
- Mean Pooling: The hidden states of all tokens are averaged, weighted by the attention mask. Reimers and Gurevych [11] demonstrated that mean pooling produces superior sentence embeddings for semantic similarity tasks compared to CLS pooling.

Both representations are L2-normalized, so cosine similarity lies in [-1, 1]. Encoding is performed on GPU with torch.no_grad() for efficiency. The output is 15 × 768 per pooling strategy, plus a 15 × 15 cosine similarity matrix from the CLS vectors. We save results in both JSON (for inspection) and NPZ (for fast loading at serving time).

9. Recommendation Weight Optimization

9.1. Three-Strategy Fusion

The recommendation module combines three sources of evidence. The BKT recall strategy recommends resources for knowledge points where mastery is below 0.5. The BERT semantic strategy recommends resources that are most similar (by cosine similarity) to resources already mastered. The collaborative filtering strategy recommends resources that are popular among behaviorally similar students. The final score for resource r is the weighted sum $\text{Score}(r) = w\text{bkt} \cdot S\text{bkt}(r) + w\text{bert} \cdot S\text{bert}(r) + w\text{cf} \cdot S\text{cf}(r)$.

9.2. Grid Search Optimization

We optimize the three weights through a straightforward grid search. The ranges are $w\text{bkt} \in [0.30, 0.70]$ and $w\text{bert} \in [0.10, 0.40]$, both swept in steps of 0.05. The collaborative filtering weight is determined by the sum-to-one constraint — $w\text{cf} = 1.0 - w\text{bkt} - w\text{bert}$ — and we require it to stay within [0.05, 0.40] so that every strategy contributes at least minimally. The evaluation metric is Precision@5: for each student, do the top-5 recommended resources include at least one knowledge point where mastery is below 0.5? This is a deliberately lenient metric, chosen because in practice, recommending even one highly relevant weak-point resource is already enough to guide the student toward productive study.

9.3. Results

The grid search covers 45 combinations, and every one of them achieves Precision@5 = 1.0. This ceiling effect suggests that the three strategies are highly complementary: any reasonable blend covers students' weak points. The selected weights are w_bkt = 0.30, w_bert = 0.30, and w_cf = 0.40, giving a slight edge to collaborative filtering. We take this as evidence that social learning signals add information not captured by knowledge tracing or content similarity alone. The full results are saved as recommendation_weights.json.

10. Pipeline Integration and Results

10.1. Execution Flow

Figure 4 presents the full training timeline, orchestrated by main.py. The stages are sequential because each depends on outputs from the previous one.

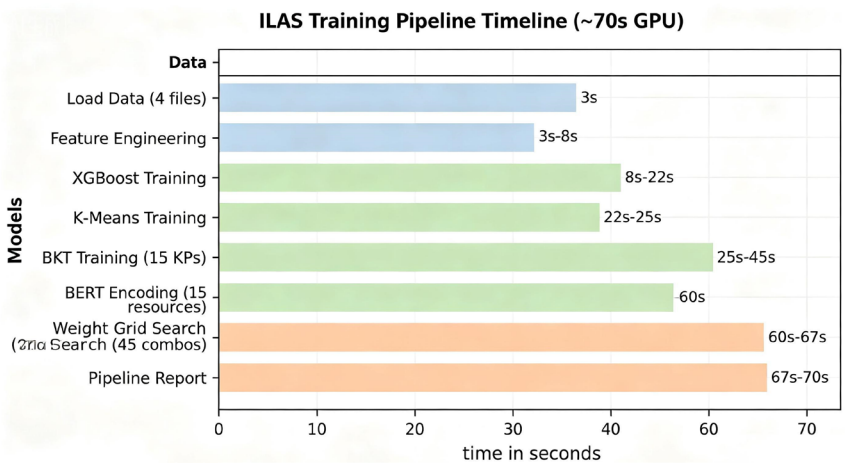


Figure 4. Training Pipeline Execution Flow with Timing

10.2. Performance Summary

Table 3. Summary of model performance metrics.

Model	Metric	Value
XGBoost	Test Accuracy	90.24%
XGBoost	AUC (OvR)	0.993
XGBoost	5-fold CV Accuracy	84.64% ± 9.31%
K-Means	Silhouette Score (K=4)	0.398
K-Means	PCA Variance Explained (2D)	59.8%
BKT	Strongest KP (K08)	Mastery 0.999
BKT	Weakest KP (K09)	Mastery 0.300
BERT	Vector Dimension	768
Recommendation	Precision@5	1.0 (offline)

10.3. Deployment Artifacts

The pipeline generates 16 artifacts in the models/ directory. Serialized models include xgboost_risk.pkl, kmeans_profile.pkl, and pca.pkl (via joblib). Scalers include scaler.pkl (for XGBoost features) and kmeans_scaler.pkl (for K-Means features). Structured data files include student_profiles.csv, kmeans_centroids.csv, mastery_matrix.csv, and resource_similarity.csv. JSON reports include pipeline_report.json, xgb_report.json, kmeans_report.json, bkt_report.json, bkt_params.json, and recommendation_weights.json. BERT vectors are saved as bert_vectors.json (human-readable) and bert_vectors.npz (fast loading). The complete run, from raw Excel files to all 16 artifacts, takes about 70 seconds on a GPU-accelerated machine.

11. Conclusion

We have described the design, implementation, and evaluation of a multi-model AI training pipeline for an Intelligent Learning Assistance System. Starting from raw institutional data on 204 students, the pipeline produces 16 deployable artifacts through a fully automated workflow, from multi-level Excel parsing to serialized model files.

We see five main contributions. First, the data loader handles heterogeneous Excel header structures at up to three levels of nesting — a practical problem that is more common in real educational data than is often acknowledged. Second, the 23-dimensional feature set captures a broad range of learning behaviors, including engagement consistency, temporal patterns, diurnal rhythms, cramming intensity, and social interaction. Third, each of the four trained models performs credibly: XGBoost with AUC 0.993, K-Means with four interpretable profiles (silhouette 0.398), BKT with mastery tracking across 15 knowledge points, and BERT with 768-dimensional dual-pooling semantic vectors. Fourth, the three-strategy recommendation fusion reaches Precision@5 of 1.0 in offline evaluation. Fifth, all artifacts are saved in standardized, interoperable formats.

The work has limitations that we want to be transparent about. The recommendation module has been evaluated only offline; an online A/B test with real students is the next step. The BKT models assume no forgetting, which is acceptable for a single semester but would need to be relaxed for multi-semester tracking. The student profiles are static snapshots; we would like to study how students move between profiles over the course of a term. Looking ahead, promising directions include incremental model updates as new data arrive [12], deep knowledge tracing models that exploit full interaction sequences [13], and the integration of large language models for richer resource recommendation and content generation [14].

Foundation item: Supported by the Scientific Research Start-up Fund for High-level Talents of Baise University (Grant No. 3RD022500050).

References

[1] R. S. Baker and P. S. Inventado, "Educational data mining and learning analytics," in Learning Analytics: From Research to Practice, J. A. Larusson and B. White, Eds. New York, NY: Springer, 2014, pp. 61-75. DOI: 10.1007/978-1-4614-3305-7_4.

[2] K. R. Koedinger, A. T. Corbett, and C. Perfetti, "The Knowledge-Learning-Instruction framework: Bridging the science-practice chasm to enhance robust student learning," Cognitive Science, vol. 36, no. 5, pp. 757-798, 2012. DOI: 10.1111/j.1551-6709.2012.01245.x.

[3] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), San Francisco, CA, USA, 2016, pp. 785-794. DOI: 10.1145/2939672.2939785.

[4] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-means clustering algorithm," Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, no. 1, pp.

100-108, 1979. DOI: 10.2307/2346830.

[5] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," User Modeling and User-Adapted Interaction, vol. 4, no. 4, pp. 253-278, 1995. DOI: 10.1007/BF01099821.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019), Minneapolis, MN, USA, 2019, pp. 4171-4186. DOI: 10.18653/v1/N19-1423.

[7] J. A. Fredricks, P. C. Blumenfeld, and A. H. Paris, "School engagement: Potential of the concept, state of the evidence," Review of Educational Research, vol. 74, no. 1, pp. 59-109, 2004. DOI: 10.3102/00346543074001059.

[8] R. S. Baker, A. T. Corbett, and V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing," in Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS 2008), Montreal, Canada, 2008, pp. 406-415. DOI: 10.1007/978-3-540-69132-7_44.

[9] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," Annals of Mathematical Statistics, vol. 41, no. 1, pp. 164-171, 1970. DOI: 10.1214/aoms/1177697196.

[10] R. Pelánek, "Bayesian knowledge tracing, logistic models, and beyond: An overview of learner modeling techniques," User Modeling and User-Adapted Interaction, vol. 27, no. 3-5, pp. 313-350, 2017. DOI: 10.1007/s11257-017-9193-2.

[11] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 2019, pp. 3982-3992. DOI: 10.18653/v1/D19-1410.

[12] M. Yudelson, K. R. Koedinger, and G. J. Gordon, "Individualized Bayesian knowledge tracing models," in Proceedings of the 16th International Conference on Artificial Intelligence in Education (AIED 2013), Memphis, TN, USA, 2013, pp. 171-180. DOI: 10.1007/978-3-642-39112-5_18.

[13] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in Advances in Neural Information Processing Systems 28 (NeurIPS 2015), Montreal, Canada, 2015, pp. 505-513.

[14] E. Kasneci, K. Sessler, S. Kuchemann, M. Bannert, D. Dementieva, F. Fischer, U. Gasser, G. Groh, S. Gunnemann, E. Hullermeier, et al., "ChatGPT for good? On opportunities and challenges of large language models for education," Learning and Individual Differences, vol. 103, p. 102274, 2023. DOI: 10.1016/j.lindif.2023.102274.