Deep Knowledge Tracing for Modeling Student Learning Behavior

By

Md. Raisul Islam Rabid
ID: CSE2203027108

Aysha Akter Anika
ID: CSE2203027133

Supervised By:

Zarin Hadika

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science & Engineering.



Department of Computer Science & Engineering
Sonargaon University (SU)

September 2026

APPROVAL

The thesis titled "**Deep Knowledge Tracing for Modeling Student Learning Behavior**" submitted by **Md. Raisul Islam Rabid** (CSE2203027108) and **Aysha Akter Anika** (CSE2203027133) to the Department of Computer Science and Engineering, Sonargaon University (SU) has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Zarin Hadika
Lecturer
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

DECLARATION

We hereby declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Zarin Hadika**, Lecturer, Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this Thesis and thereof has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Zarin Hadika)
Supervisor

Signatures

Md. Raisul Islam Rabid
ID: CSE2203027108

Aysha Akter Anika
ID: CSE2203027133

ABSTRACT

Understanding student learning behavior is an important task in modern educational technology. Knowledge Tracing (KT) can estimate students' changing knowledge states from their previous learning interactions and predict future performance. This study investigates the use of deep learning-based Knowledge Tracing models to analyze student learning behavior using real interaction data collected through an online quiz platform.

The dataset contains 6,318 interactions from 426 students covering 12 skills and 3 courses. The data were cleaned and chronologically ordered before modeling. Two behavioral features, Time Gap and Attempt Count, were introduced to examine their effect on prediction performance. Four models, DKT, DKVMN, SAKT, and AKT-lite, were evaluated using four feature configurations with Grouped 5-Fold Cross-Validation.

The results show that DKT with Time Gap and Attempt Count achieved the highest pooled ROC-AUC of 0.775. DKVMN achieved 0.684, AKT-lite 0.674, and SAKT 0.616. Attempt Count was the most consistently useful engineered feature across the evaluated models. The study also explores knowledge-state analysis, adaptive question sequencing, and learner grouping to demonstrate the practical potential of Knowledge Tracing.

Keywords: Knowledge Tracing, Student Learning Behavior, DKT, DKVMN, SAKT, AKT-lite, Time Gap, Attempt Count, ROC-AUC.

ACKNOWLEDGMENT

All praise and gratitude belong to Almighty Allah, whose grace, blessings, and strength enabled us to complete this thesis successfully within the designated timeframe.

We extend our heartiest appreciation and deep respect to our esteemed supervisor, **Zarin Hadika**, Lecturer, Department of Computer Science and Engineering at Sonargaon University. Her invaluable guidance, insightful suggestions, and constant encouragement served as the guiding light throughout every stage of this research.

We are profoundly grateful to **Prof. Bulbul Ahamed**, Head of the Department of Computer Science and Engineering, Sonargaon University, for his kind support, continuous motivation, and constructive feedback.

Our sincere thanks also go to all the respected faculty members who have taught and inspired us throughout our academic journey, instilling in us a passion for continuous learning.

Finally, words cannot fully express our eternal love and gratitude to our parents. Their unconditional support, prayers, sacrifices, and belief in us have been our greatest source of strength.

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AUC	Area Under the ROC Curve
DKT	Deep Knowledge Tracing
DL	Deep Learning
KT	Knowledge Tracing
LSTM	Long Short-Term Memory
ML	Machine Learning

TABLE OF CONTENTS

Title **Page No.**

DECLARATION iii
ABSTRACT iv
ACKNOWLEDGEMENT v
LIST OF ABBREVIATIONS vi

CHAPTER 1
INTRODUCTION
1.1 Background of the Study 1
1.2 Motivation 2
1.3 Problem Statement 3
1.4 Research Objectives 3
1.5 Scope of the Research 4
1.6 Contribution of the Thesis 4
1.7 Organization of the Thesis 4

CHAPTER 2
LITERATURE REVIEW
2.1 Educational Data Mining 6
2.2 Learning Analytics 6
2.3 Student Learning Behavior 7
2.4 Knowledge Tracing 7
2.5 Bayesian Knowledge Tracing 8
2.6 Deep Knowledge Tracing 8
2.7 Recurrent Neural Networks and LSTM 9
2.8 Dynamic Key-Value Memory Networks 9
2.9 Self-Attentive Knowledge Tracing 10
2.10 Explainability and Learner Analysis 11
2.11 Research Gap 11

CHAPTER 3
METHODOLOGY
3.1 Dataset Description 12
3.2 Data Cleaning 14
3.3 Chronological Ordering 14
3.4 Skill Encoding 15
3.5 Time-Gap Feature Engineering 16
3.6 Attempt-Count Feature Engineering 16
3.7 Leakage Prevention 17
3.8 System Architecture 18
3.9 Vanilla DKT 20
3.10 DKVMN 21
3.11 SAKT 22
3.12 AKT-lite 23
3.13 Grouped 5-Fold Cross-Validation 24
3.14 Experimental Setup 25
3.14.1 Experimental Design 25
3.14.2 Evaluation Metrics 26
3.14.3 Model Configuration 27

CHAPTER 4
RESULTS
4.1 DKT Results....29
4.1.1 DKT Knowledge-State Analysis....30
4.2 Adaptive Sequencing Analysis....31
4.3 Learner Clustering....32
4.4 Cluster Skill-Mastery Profiles....33
4.5 DKVMN Results....34
4.5.1 DKVMN Knowledge-State Analysis....35
4.6 SAKT Results....35
4.6.1 SAKT Knowledge-State Analysis....37
4.7 AKT-lite Results....37
4.7.1 AKT-lite Knowledge-State Analysis....39
4.8 Overall Model Comparison....39

CHAPTER 5
DISCUSSION
5.1 Performance Comparison....42
5.2 Effect of time gap....43
5.3 Effect of Attempt Count....44
5.4 Knowledge-State Interpretation....45
5.5 Adaptive Learning Implications....46
5.6 Learner Segmentation....48
5.7 Comparison with Existing Studies....48
5.8 Primary Evaluation Matrix AUC-ROC....49
5.9 Practical Implications....50

CHAPTER 6
LIMITATIONS
6.1 Dataset....52
6.2 Behavioral Feature....52
6.3 Model Complexity....52
6.4 Interpretability....53
6.5 Adaptive Simulation....53

CHAPTER 7
CONCLUSION AND FUTURE WORK
7.1 Conclusion....54
7.2 Future Work....55

REFERENCES....56

LIST OF TABLE

Table No. **Title** **Page No.**

Table 3.1 Description and Data Types of Dataset Attributes 13
Table 3.2 Dataset Summary Statistics 14
Table 3.3 Experimental Matrix 26
Table 4.1 DKT performance across the four input-feature variants 29
Table 4.2 Simulated adaptive sequencing session for student anika_7133 using DKT 31
Table 4.3 DKVMN performance across the four input-feature variants 34
Table 4.4 SAKT performance across the four input-feature variants 35
Table 4.5 AKT-lite performance across the four input-feature variants 37
Table 4.6 Pooled AUC-ROC across the four input-feature variants for each evaluated architecture 40
Table 4.7 Comparative performance of the four evaluated architectures at their best-performing configurations (pooled ROC-AUC) 40
Table 5.1 Best-Performing Variant per Architecture 42
Table 5.2 Effect of Adding the Time-Gap Feature 43
Table 5.3 Effect of Adding the Attempt-Count Feature 44
Table 5.4 Performance comparison of models in simulated adaptive question recommendation and range adherence 47
Table 5.5 Comparative Analysis with Existing Published Studies 49

LIST OF FIGURES

Figure No. Title Page No.

Figure 1.1 Knowledge Tracing 1

Figure 1.2 Evolution of Knowledge Tracing models, from BKT to AKT 2

Figure 3.1 User interface of the self-developed online quiz platform 12

Figure 3.2 Example of a chronologically ordered interaction sequence for a single student .. 15

Figure 3.3 Leakage-safe feature construction using only previously completed interactions to predict the next response 18

Figure 3.4 Overall proposed framework, from raw student data collection through to the final downstream analyses 20

Figure 3.5 Vanilla DKT architecture 21

Figure 3.6 DKVMN architecture 22

Figure 3.7 SAKT architecture 23

Figure 3.8 AKT-lite architecture 24

Figure 4.1 DKT Performance Across Input Feature Variants 29

Figure 4.2 Knowledge State Interaction Heatmap of DKT with Time & Attempt Variant 30

Figure 4.3 DKT t-SNE Projection of Learner Mastery Clusters 32

Figure 4.4 DKT Cluster Skill Mastery Profiles 33

Figure 4.5 DKVMN Performance Across Input Feature Variants 34

Figure 4.6 Knowledge State Interaction Heatmap of DKVMN with Attempt Count Variant 35

Figure 4.7 SAKT Performance Across Input Feature Variants 36

Figure 4.8 Knowledge State Interaction Heatmap of SAKT Vanilla 37

Figure 4.9 AKT-lite Performance Across Input Feature Variants 38

Figure 4.10 Knowledge State Interaction Heatmap of AKT-lite with Attempt Count Variant . 39

Figure 4.11 Best ROC-AUC Across Evaluated Architectures 41

Figure 5.1 Knowledge-State Mastery Heatmap showing dynamic skill probability predictions across sequential student interaction timesteps 45

CHAPTER 1

INTRODUCTION

1.1 Background of the Study

Education is a fundamental component of human development and plays an important role in individual and societal progress. It enables learners to develop knowledge, skills, critical thinking, and problem-solving abilities. However, students different in their prior knowledge, learning ability, and pace, making it difficult for conventional educational approaches to address every learner's individual needs. Therefore, identifying students' learning progress and knowledge gaps is essential for improving educational outcomes. Figure 1.1: shows Knowledge Tracing[1]

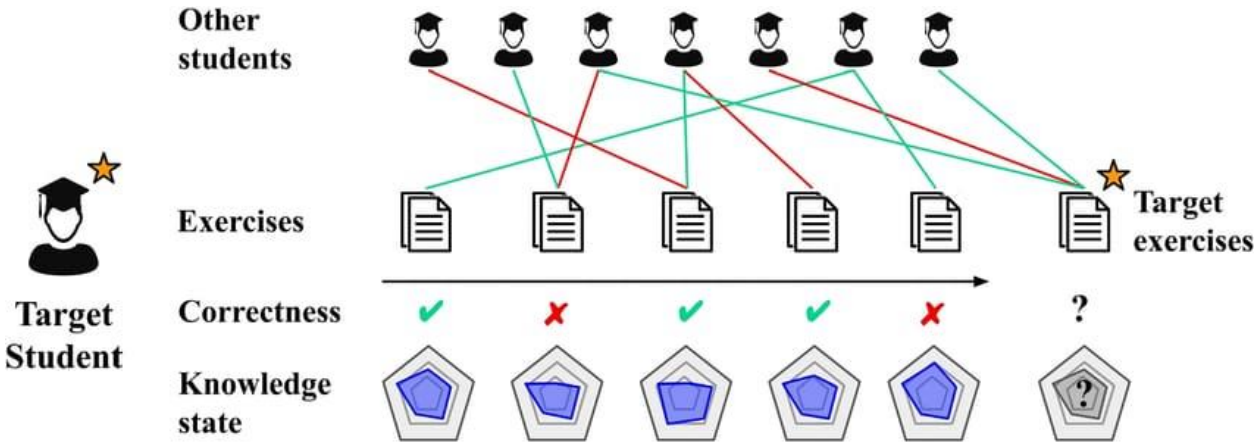


Figure 1.1: Knowledge Tracing

Knowledge Tracing (KT) is an important research area in educational data mining and intelligent tutoring systems that models a learner's evolving knowledge state based on historical learning interactions, such as question attempts and responses [2]. Recent research has further focused on making KT more realistic by considering how students may forget previously learned knowledge. Chen et al. [3] proposed a profile-driven forgetting approach for English Knowledge Tracing, where learners' historical interactions are used to model individual differences in knowledge retention. Their study also addressed noisy learning data and showed that considering personalized forgetting and data robustness can improve knowledge-state prediction. These developments highlight the importance of modelling not only students correct and incorrect responses but also the changes in their knowledge over time.

The challenges of Knowledge Tracing are particularly relevant in the context of Bangladesh, where educational goals and actual assessment practices may not always be fully aligned. Ali et al. [4] found that although the secondary English curriculum in Bangladesh emphasizes the development of knowledgeable, skilled, rational, and creative learners, teaching and assessment practices often pursue different goals. Their study also identified a gap between curriculum objectives and

assessment practices, which is influenced by local structural and contextual factors. In addition, limited resources can affect assessment priorities, while differences between rural and urban educational settings may further influence the quality and consistency of assessment practices [4]. These issues highlight the need for more systematic and data-driven approaches to understand students' actual learning progress in the Bangladeshi educational environment.

Previous studies have identified several important limitations in KT. Deep learning-based KT research has highlighted problems related to forgetting, long-term dependencies, and complex learning patterns [3], while explainable KT research has emphasized the difficulty of understanding how complex models reach their predictions [5]. Data sparsity and cold-start conditions have also been identified as challenges for reliable learner knowledge estimation [6]. These findings suggest that an effective KT system should not only provide accurate predictions but also be robust to limited data and practical for educational use.

Considering these challenges, this thesis addresses the identified gap by developing a real-world quiz platform to collect genuine learning interaction data from students and evaluating multiple Knowledge Tracing models, including DKT, DKVMN, SAKT, and AKT, on the collected local dataset. This approach connects established KT research with a real and relatively underrepresented educational setting, providing an opportunity to evaluate the effectiveness of different KT models in the Bangladeshi context.

1.2 Motivation

This research is motivated by the need to understand how a student's knowledge changes over time. In many learning environments, teachers mainly rely on students' test results, classroom performance, or personal judgment to decide what a student needs to learn next. However, these approaches may not always show which specific concepts a student has mastered or where they are still struggling. Knowledge Tracing can use students' previous interactions to estimate their current knowledge state and identify their individual learning needs.

Previous Knowledge Tracing studies have proposed different methods to improve student knowledge estimation. Corbett and Anderson (1994) introduced Bayesian Knowledge Tracing using a small number of statistical parameters [7]. Later, Piech et al. (2015) proposed Deep Knowledge Tracing (DKT), which uses recurrent neural networks to learn knowledge patterns from student interaction sequences [8]. Further studies introduced different approaches, including memory networks, self-attention, and attention-based models with decay mechanisms [9–11]. These models have shown promising results, but their performance can vary across different datasets and learning environments.

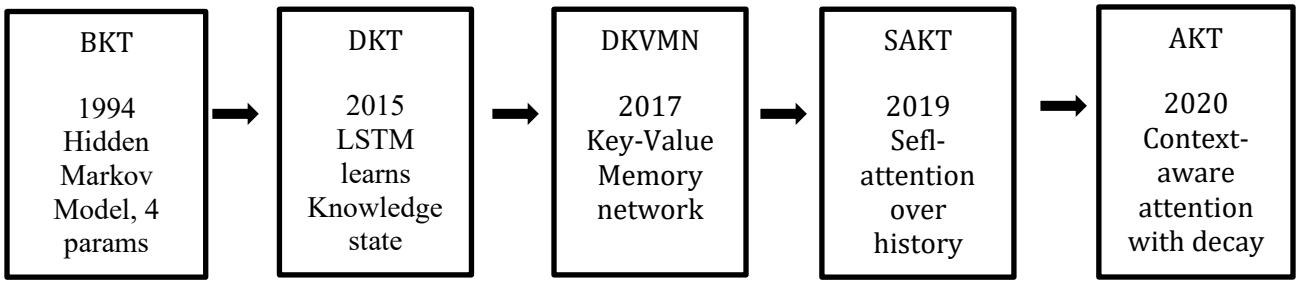


Figure 1.2: Evolution of Knowledge Tracing models, from BKT to AKT.

This variation motivates the present study to compare different Knowledge Tracing models using real student interaction data. The study also considers simple learning-related information, such as response time and the number of attempts, which may provide useful information about student learning behavior. By evaluating multiple KT models on a locally collected dataset, this research aims to identify suitable approaches for tracking students' knowledge states in a real educational environment.

1.3 Problem Statement

Knowledge Tracing becomes important when a teacher, coaching centre, or online learning platform needs to track the learning progress of many students across different topics. Manually checking every student's knowledge before each lesson is time-consuming and difficult, especially when the number of students is large. Therefore, an automated system is needed to continuously determine whether a student is ready for the next question or topic or needs more practice first. This situation is particularly relevant in Bangladesh, where teachers often manage a large number of students.

Another challenge is that most existing KT models are trained and evaluated using large foreign or English-language datasets. These datasets may contain more student interactions, longer sessions, and different question patterns than those found on a small, real-world local platform. As a result, a model that performs well on benchmark datasets may not perform equally well on limited and locally collected student data. There is still limited research evaluating how different KT models perform when applied to real student data from the Bangladeshi educational context.

Furthermore, existing studies often compare different KT architectures while keeping the input information mostly unchanged. Less attention has been given to simple behavioral features, such as the number of attempts made by a student and the time elapsed since their previous activity, and how these features interact with different KT architectures. These features are easy to collect and do not require additional complex data collection. Therefore, it is important to directly investigate whether such features improve student performance prediction and determine which combinations of features and KT architectures work best on real, locally collected data.

1.4 Research Objectives

The main objective of this research is to develop a real quiz platform, collect genuine student interaction data, and evaluate different Knowledge Tracing models and input features on a locally collected dataset. To achieve this goal, the following objectives are defined:

- To design and develop an online quiz platform that records student interactions, including correctness, response time, topic, and timestamp.
- To collect and prepare a real-world Knowledge Tracing dataset from students for sequence-based learning analysis.
- To implement Deep Knowledge Tracing (DKT) using an LSTM-based architecture as a baseline model for predicting students' next responses.
- To investigate the effects of engineered features, including question difficulty, time-since-last-attempt, and per-skill attempt count, on Knowledge Tracing performance.
- To implement and comparatively evaluate DKT, DKVMN, SAKT, and AKT models using the same dataset and evaluation protocol.

- To evaluate the models using appropriate validation techniques and performance metrics, including AUC-ROC, PR-AUC, Accuracy, Precision, Recall, and F1-score.
- To use the best-performing model to analyze students' knowledge-state progression and explore its potential for adaptive question sequencing and learner grouping.

1.5 Scope of the Research

This research focuses on modelling students' knowledge states using structured interaction data collected from a single online quiz platform. The data include information such as topic, correctness, response time, and timestamp. The study does not use unstructured data such as free-text answers, audio, or video. Four Knowledge Tracing architectures DKT, DKVMN, SAKT, and AKT are implemented and comparatively evaluated using the collected dataset and selected engineered features.

The research also includes knowledge-state visualization, adaptive question-sequencing simulation, and learner grouping based on the model results. However, it does not involve deploying a real-time recommendation system for actual students. The study is limited to data collected from the developed quiz platform and does not include classroom observations or standardized test records. Graph-based Knowledge Tracing is also excluded because the current dataset does not provide reliable prerequisite relationships between skills. Therefore, this research is intended as a research-stage framework for comparing Knowledge Tracing approaches on real, locally collected student data rather than as a large-scale production-ready adaptive learning system.

1.6 Contribution of the Thesis

This research contributes to the field of Knowledge Tracing by providing practical insights into student knowledge estimation using real, locally collected data. Since research based on Bangladeshi student data is limited, this study provides useful evidence about how KT approaches can be applied in a local educational environment.

The study compares four Knowledge Tracing models DKT, DKVMN, SAKT, and AKT along with selected input features on the same real-world dataset. This comparison helps identify which model and feature combinations perform better under limited-data conditions and can provide useful guidance for future KT research in similar educational settings.

In addition, the research develops a complete pipeline from student interaction data to useful learning outputs. These include knowledge-state visualization, adaptive question-sequencing simulation, and learner grouping. Thus, the study connects Knowledge Tracing model evaluation with practical educational applications and provides a foundation for future development of personalized learning systems.

1.7 Organization of the Thesis

This thesis is organized into seven chapters. Chapter 1 presents the introduction, including the background, motivation, problem statement, objectives, scope, and contributions of the research. Chapter 2 reviews related literature on Knowledge Tracing, including Bayesian Knowledge Tracing, Deep Knowledge Tracing, DKVMN, SAKT, and AKT. Chapter 3 describes the research methodology, including data collection, dataset preparation, model development, and evaluation

methods. Chapter 4 presents the experimental results and compares the performance of different models and feature combinations. Chapter 5 discusses the findings, including knowledge-state visualization, adaptive question sequencing, and learner grouping. Chapter 6 presents the conclusion of the research. Finally, Chapter 7 discusses the limitations of the study and suggests directions for future research.

Chapter 2
LITERATURE REVIEW

This chapter reviews the research background behind the work carried out in this thesis. It begins with the broader fields of educational data mining and learning analytics, narrows into student learning behavior and the general problem of knowledge tracing, and then traces the field's technical progression from Bayesian knowledge tracing through deep, memory-augmented, and self-attentive architectures. The chapter also reviews recent work on explainability in learner modeling before closing with a discussion of the research gap this thesis addresses.

2.1 Educational Data Mining

Dutt, Ismail, and Herawan (2017) presents a systematic review of educational data mining research, categorizing existing work into major task types such as prediction, clustering, and relationship mining [12]. Bakhshinategh et al. (2018) surveys educational data mining applications and tasks published over a ten-year period, describing how course management systems and intelligent tutoring systems generate the structured log data such techniques rely on [13]. Aldowah, Al-Samarraie, and Fauzy (2019) reviews the state of educational data mining and learning analytics research applied to twenty-first-century higher education, synthesizing common methods and open challenges across the field [14].

Romero and Ventura (2020) presents an updated survey of educational data mining and learning analytics, describing how the field has matured from early rule-based techniques toward increasingly automated, model-driven analysis [15]. Baker (2019) discusses ongoing challenges facing educational data mining research, describing structured prediction competitions used to drive methodological progress in the field [16]. Taken together, this body of work establishes the broad toolkit, prediction, clustering, and pattern discovery from student log data, within which the more specific problem of knowledge tracing sits.

2.2 Learning Analytics

Viberg, Hatakka, Bälter, and Mavroudi (2018) reviews the current landscape of learning analytics research in higher education, describing the methods, data sources, and reported outcomes most common across recent studies [17]. Ferguson and Clow (2017) discusses the evidence base behind learning analytics interventions, calling for more rigorous validation of the outcomes these systems claim to produce [18]. Gašević et al. (2015) states that learning analytics research must remain grounded in established theories of learning rather than treating data analysis as an end in itself, cautioning against purely technique-driven work [19].

Ifenthaler and Yau (2020) presents a systematic review of how learning analytics has been used to support study success in higher education, describing which analytic approaches were most consistently associated with improved outcomes [20]. Sønderlund, Hughes, and Smith (2019) reviews the efficacy of learning analytics interventions across higher education settings, describing mixed evidence for their direct impact on learning outcomes [21]. This overlap between technique and instructional impact is directly relevant to the present thesis, since the mobile-based adaptive learning system developed here functions simultaneously as a data mining pipeline and a learning analytics tool feeding back into instructional decisions.

2.3 Student Learning Behavior

Kizilcec, Pérez-Sanagustín, and Maldonado (2017) analyses how self-regulated learning strategies relate to learner behavior and goal attainment in massive open online courses, describing distinct behavioral patterns among learners who succeed versus those who disengage [22].

Jovanović et al. (2017) examines learning-analytics traces from a flipped-classroom setting, describing distinct study-strategy patterns that emerge directly from students' sequences of resource access [23]. Li, Baker, and Warschauer (2020) proposes methods for measuring self-regulated learning from clickstream data, demonstrating that these behavioral traces can support and predict learning outcomes in online courses [24].

Matcha et al. (2020) presents a systematic review of learning-analytics dashboards designed from a self-regulated learning perspective, describing how behavioral indicators are typically surfaced back to learners and instructors [25]. Saint et al. (2020) combines multiple analytic methods to uncover sequential and temporal patterns in self-regulated learning behavior, describing how the order and timing of study actions carry information beyond simple counts [26]. These studies establish the observable behavioral signals, attempt patterns, timing, and engagement, that the feature-based and deep knowledge tracing models reviewed in the following sections attempt to capture computationally.

2.4 Knowledge Tracing

Knowledge Tracing (KT) is the general problem of inferring a student's evolving mastery of a set of skills from the sequence of their past interactions, typically their attempts at questions or exercises and whether those attempts were correct. A KT model's output, an estimate of how well a student currently knows each skill, can then be used to predict future performance, recommend the next question, or flag topics that need further practice.

Abyaa, Idrissi, and Bennani (2019) presents a systematic review of learner modeling literature published over five years, highlighting a persistent reliance on a small number of large, English-language benchmark datasets across the surveyed studies [27]. Liu et al. (2021) presents a broad survey of knowledge tracing research, organizing existing approaches into probabilistic, logistic, and deep-learning-based families and comparing their reported strengths and weaknesses [28].

Song et al. (2022) surveys deep-learning-based knowledge tracing specifically, describing how successive architectures have progressively relaxed the simplifying assumptions made by earlier probabilistic models [29]. Shen et al. (2021) proposes a learning-process-consistent knowledge tracing model that explicitly separates a student's learning gain, forgetting, and current knowledge state, aiming to more faithfully mirror the cognitive process underlying skill acquisition [30].

Nakagawa, Iwasawa, and Matsuo (2019) proposes a graph-based knowledge tracing model that represents relationships between skills explicitly, demonstrating improved prediction when prerequisite structure between skills is available [31]. Taken together, this body of work frames knowledge tracing as a spectrum running from interpretable, hand-designed representations to increasingly data-driven ones, a spectrum the remaining sections of this chapter trace in more technical detail.

2.5 Bayesian Knowledge Tracing

Bayesian Knowledge Tracing (BKT) is one of the earliest and most widely deployed approaches to knowledge tracing. It represents a student's mastery of a single skill as a binary hidden variable, either known or not yet known, and updates the probability of that variable using a Hidden Markov Model each time the student attempts a related question, with parameters for prior knowledge, learning rate, guess, and slip. Because it has only a handful of parameters, BKT is simple to fit even on small numbers of interactions and its output is directly interpretable as a probability of mastery, which explains its long history of use in production intelligent tutoring systems.

Pelánek (2017) examines Bayesian and logistic student modeling techniques side by side, discussing the differing assumptions each family makes about how learning and performance are related and describing several widely used contextual extensions to the basic BKT parameters [32]. Käser et al. (2017) proposes Dynamic Bayesian Networks that extend BKT's single-skill Hidden Markov structure to model dependencies between multiple skills jointly, demonstrating improved predictive accuracy on structured curricula [33]. Khajah, Lindsey, and Mozer (2016) proposes an enhanced BKT model that incorporates item difficulty, individual student ability, and forgetting into the basic framework, and demonstrates that much of the reported advantage of deep sequence models can be closed by such careful, targeted extensions to BKT [34].

Montero et al. (2018) investigates whether a deep sequence model captures interactions between skills that a single-skill, BKT-style formulation cannot, finding that explicitly modeling such interactions further improves prediction beyond either approach used alone [35]. Gervet et al. (2020) conducts a large-scale comparison of BKT variants against deep knowledge tracing models across multiple datasets, finding that neither family consistently dominates the other and that dataset characteristics strongly influence which approach performs best [36]. Despite these refinements, BKT's Markov assumption, that a student's next answer depends only on their current binary knowledge state, still limits its ability to capture richer patterns such as gradual forgetting or the influence of related skills, motivating the deep learning approaches discussed next.

2.6 Deep Knowledge Tracing

Deep Knowledge Tracing (DKT) refers to the family of approaches that replace BKT's hand-designed probabilistic structure with a neural network, typically a recurrent one, that learns to represent a student's knowledge state directly from raw interaction sequences. Rather than tracking a single binary variable per skill, DKT models maintain a continuous, high-dimensional hidden state that can implicitly capture relationships between skills without those relationships being manually specified, which was the main motivation for DKT's introduction. Piech et al. (2015) proposes DKT itself, encoding a student's interaction history with a Recurrent Neural Network that reads out, at every timestep, a probability of success on every skill directly from its hidden state [8].

Xiong et al. (2016) re-examines DKT's original evaluation protocol and states that part of its reported improvement stemmed from quirks in how certain benchmark datasets were split and repeated, finding a smaller true gain over well-tuned BKT baselines once these issues were corrected [37]. Yeung and Yeung (2018) observes that DKT's skill-mastery predictions can fluctuate implausibly between consecutive timesteps, and proposes prediction-consistent regularization terms that produce a variant, DKT+, with both higher accuracy and more behaviorally plausible outputs [38]. Minn et al. (2018) proposes combining DKT with dynamic student classification, grouping students with similar

learning trajectories before applying the recurrent model, and demonstrates improved prediction for students with limited interaction history [39].

Sonkar et al. (2020) presents qDKT, a question-centric variant of DKT that models individual exercises rather than skill labels alone, addressing the loss of information that occurs when multiple distinct questions are mapped onto a single skill tag [40]. Together these studies establish that while DKT was a genuine advance in flexibility over BKT, its evaluation and internal behavior both required subsequent correction, leaving its architecture as a foundation to build on rather than a finished solution.

2.7 Recurrent Neural Networks and LSTM

Recurrent Neural Networks (RNNs) are the class of neural network architecture that DKT and its variants build on, designed specifically to process sequential data by maintaining a hidden state that is updated at every timestep as new input arrives. Plain RNNs, however, struggle to retain information over long sequences due to the vanishing-gradient problem, which motivated the development of gated variants, most commonly the Long Short-Term Memory (LSTM) network and the more lightweight Gated Recurrent Unit (GRU), capable of preserving relevant information for much longer. Jozefowicz, Zaremba, and Sutskever (2015) conducts a large-scale empirical comparison of recurrent architectures, finding that gated variants such as LSTM and GRU consistently outperform plain recurrent units on long-sequence tasks, though no single gated variant dominates across every task tested [41].

Greff et al. (2017) presents a large-scale empirical study of LSTM variants, describing which of its gates and connections contribute most to its performance and confirming that the core forget-gate mechanism is essential to its success on long sequences [42]. Nagatani et al. (2019) addresses DKT's lack of an explicit forgetting mechanism by incorporating features such as the number of past attempts, past successes, and time elapsed since a skill was last practiced into the recurrent update, and demonstrates consistently improved prediction accuracy over vanilla DKT [43]. Lee and Yeung (2019) proposes the Knowledge Query Network, which replaces DKT's implicit skill representation with an explicit query mechanism that separates what a student has learned from what is being asked [44]. Su et al. (2018) presents an exercise-enhanced recurrent model that incorporates the textual content of each exercise alongside the correctness signal, and demonstrates that this additional context improves performance prediction beyond skill labels alone [45].

Yang et al. (2020) proposes a graph-based interaction model that combines a graph neural network over skill relationships with a recurrent component, and states that jointly modeling skill structure and sequence order improves prediction over recurrence alone [46]. These works show that recurrence alone is a necessary but not sufficient basis for behaviorally realistic student modeling, a finding directly relevant to this thesis's decision to compare a plain LSTM-based DKT baseline against a feature-augmented variant.

2.8 Dynamic Key-Value Memory Networks

Memory-augmented neural networks extend standard recurrent architectures with an external, addressable memory component that the network can explicitly read from and write to, rather than relying solely on a single compressed hidden state. In the context of knowledge tracing, this idea is used to represent a student's mastery of many distinct skills separately, each stored in its own memory slot, instead of compressing all of that information into one shared vector as DKT does.

Weston et al. (2015) presents the original Memory Networks architecture, introducing an external, addressable memory component that a neural network can read from and write to when reasoning over a sequence of facts [47]. Sukhbaatar et al. (2015) proposes an end-to-end trainable variant of this architecture that removes the need for strong supervision at each memory access step, broadening its applicability to tasks where intermediate access patterns are not labeled [48].

Graves et al. (2016) proposes the Differentiable Neural Computer, combining a neural network controller with a differentiable external memory bank that can be read from and written to with soft attention, establishing a general template for memory-augmented sequence models [49]. Zhang et al. (2017) adapts this line of work directly to education with the Dynamic Key-Value Memory Network (DKVMN), which maintains a static key matrix encoding relationships between latent skill concepts and a dynamic value matrix that separately tracks the student's evolving mastery of each concept, and demonstrates improved accuracy and interpretability over DKT [50].

Abdelrahman and Wang (2019) extends this idea with the Sequential Key-Value Memory Network (SKVMN), combining DKVMN's memory structure with a hop-based LSTM mechanism that retrieves past hidden states when a skill reappears after a long gap [51]. Abdelrahman et al. (2023) presents a comprehensive survey of knowledge tracing methods, describing memory-augmented architectures as one of the major branches of the field alongside recurrent and attention-based models [52]. Memory-augmented models therefore trade some of DKT's architectural simplicity for a more structured, and often more interpretable, representation of multi-skill mastery.

2.9 Self-Attentive Knowledge Tracing

Self-attention is a mechanism that allows a model to weigh the relevance of every past interaction in a sequence when making a prediction, rather than processing the sequence strictly step by step as a recurrent network does. Applied to knowledge tracing, this allows a model to directly identify which specific past questions are most informative for predicting a student's performance on the next one, and to do so in parallel rather than through a chain of sequential updates.

Vaswani et al. (2017) proposes the Transformer architecture, replacing recurrence entirely with a self-attention mechanism that computes a weighted combination over all positions in a sequence in parallel, and this architecture underlies every self-attentive knowledge tracing model discussed below [53].

Pandey and Karypis (2019) applies self-attention directly to knowledge tracing with the Self-Attentive Knowledge Tracing (SAKT) model, which attends over a student's past exercises to identify which prior interactions are most relevant to the next prediction, and demonstrates particularly strong gains on sparse interaction sequences where LSTM-based DKT struggled [10].

Ghosh et al. (2020) states that plain self-attention, as used in SAKT, ignores two properties specific to educational data, the exponential decay of a skill's relevance over time and the distinction between question difficulty and student knowledge, and proposes the Attentive Knowledge Tracing (AKT) model to address both [11]. Choi et al. (2020) proposes SAINT, a full encoder-decoder Transformer that feeds exercise information to the encoder and response information to the decoder, and demonstrates that this separation improves training stability on large-scale, real-world platforms such as EdNet [54].

Pandey and Srivastava (2020) propose RKT, which incorporates explicit relation information between exercises into the self-attention computation, and states that this relational signal improves

prediction over models that treat exercises as independent [55]. Pu et al. (2020) experiments with several Transformer variants for knowledge tracing and compares their performance against SAKT and DKT, describing the conditions under which attention-based models offer the clearest advantage over recurrent ones [56]. Self-attentive models now represent the strongest-performing family of architectures on most public KT benchmarks, though, as with DKT before them, this performance has been established almost entirely on large-scale, publicly available datasets.

2.10 Explainability and Learner Analysis

As knowledge tracing models have grown more complex, from simple probabilistic models to deep neural networks, concerns about their interpretability have grown alongside them. Explainability in this context refers to the extent to which a model's internal predictions, such as its estimate of a student's mastery of a particular skill, can be understood, verified, or acted on by a teacher or student rather than treated as an opaque number, which is particularly important for an educational tool since an unexplainable mastery estimate is of limited practical use to an instructor deciding what a student should study next.

Doshi-Velez and Kim (2017) discusses the need for a rigorous, quantifiable science of interpretability in machine learning, proposing taxonomies for evaluating how explainable a given model actually is to its intended audience [57].

Yeung (2019) proposes Deep-IRT, combining a deep sequence model with an item response theory layer so that skill ability and item difficulty can be read directly from the network rather than inferred after the fact [58]. Lu et al. (2020) proposes an interpretable variant of deep knowledge tracing that attaches attention weights to individual exercises, describing which past interactions most influenced a given prediction [59].

Scruggs, Baker, and McLaren (2020) extends deep knowledge tracing by inferring interpretable knowledge components, demonstrating that these inferred components can also predict performance on a separate, later system [60]. Minn et al. (2022) proposes an interpretable knowledge tracing model built on explicit causal relations between skills, demonstrating comparable accuracy to black-box deep models while remaining simple enough to inspect directly [61]. This body of work shows that explainability and predictive accuracy are not mutually exclusive, but also that most interpretable knowledge tracing models have so far been proposed and evaluated in isolation, on the same handful of large public datasets, rather than compared directly against a broader spread of traditional and deep architectures on real, small-scale data.

2.11 Research Gap

Existing studies in educational data mining and knowledge tracing mainly rely on large public datasets, with limited work on small, real-world local data. Behavioural and engineered features are also rarely incorporated into knowledge tracing, while interpretability is seldom evaluated alongside different model architectures. This thesis addresses these gaps by collecting real student interaction data and comparatively evaluating multiple knowledge tracing models under a unified framework.

Chapter 3
METHODOLOGY

3.1 Dataset Description

The dataset used in this study was collected from a self-developed online quiz platform designed specifically for this research. The platform was developed to provide an interactive environment where students could answer quiz questions across different topics and difficulty levels. During each interaction, the platform automatically recorded relevant information, including the student ID, problem ID, course, topic, difficulty level, correctness of the answer, response time, earned experience points (XP), and timestamp. Screenshots of the developed quiz platform are provided to demonstrate the interface and the process through which the interaction data were generated.

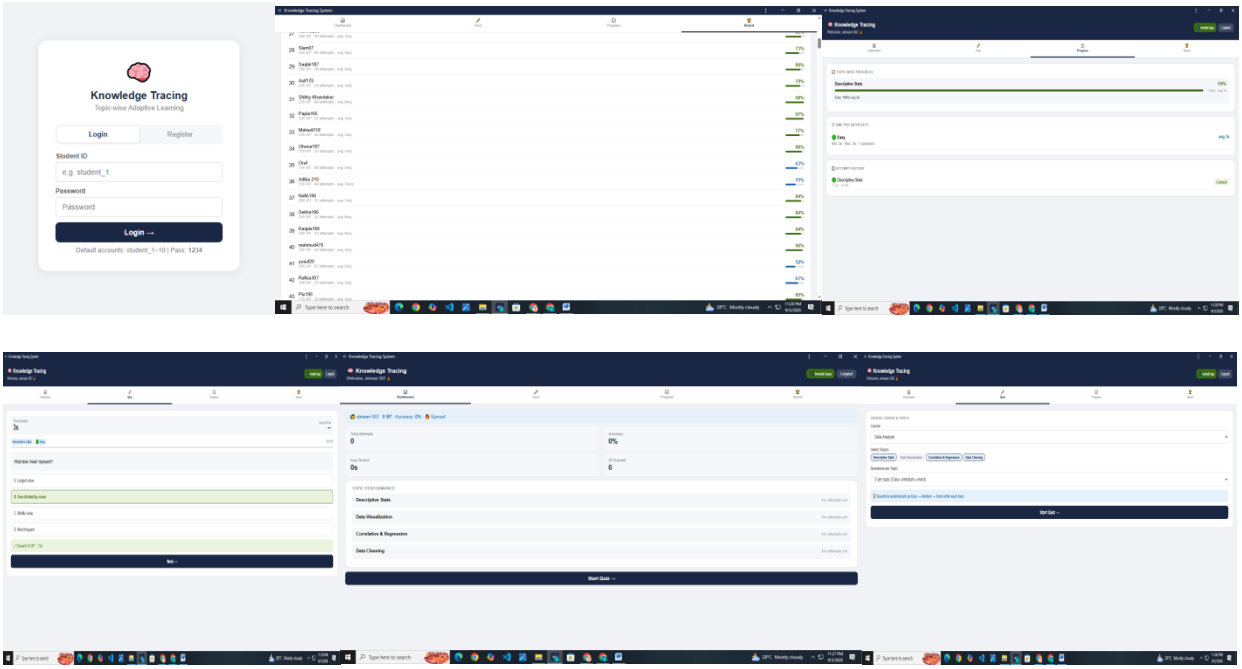


Figure 3.1: User interface of the self-developed online quiz platform.

The dataset represents real user interactions collected from the developed platform rather than synthetic or simulated data. The collected records therefore reflect students' actual quiz-solving behavior and engagement with the platform.

The final dataset contains 6,318 interaction records from 426 unique students, covering 12 different topics and 3 courses: Data Analysis, Pattern Recognition, and Machine Learning Basics. Each topic has three difficulty levels: Easy, Medium, and Hard. Among the 6,318 responses, 4,160 (65.8%) were correct and 2,158 (34.2%) were incorrect. This distribution indicates a moderately imbalanced but realistic pattern of student responses.

On average, each student attempted around 15 questions. However, the number of interactions varied considerably among students, ranging from 3 to 124 attempts, indicating differences in students' levels of engagement with the platform. A detailed summary of the distribution of the dataset across courses, topics, and difficulty levels is presented in Table 3.2.To support the subsequent analysis and modeling process, the dataset was organized into several variables representing student characteristics, question information, performance, response behavior, and interaction time. Table 3.1 provides an overview of the dataset attributes, including their data types, descriptions, and roles in the modeling pipeline**.**

Table 3.1: Description and Data Types of Dataset Attributes

Column Name	**Data Type**	**Description**
student_id	String	Unique identifier for each student
student_name	String	Display name of the student
problem_id	String	Unique identifier for each question attempted
course	String	Course the question belongs to (3 categories)
topic	String	Skill or concept being tested (12 categories)
difficulty	String	Difficulty level of the question (Easy / Medium / Hard)
correct	Integer(0/1)	Whether the response was correct (1) or incorrect (0)
time_seconds	Integer	Time taken by the student to respond, in seconds
xp_earned	Integer	Experience points awarded for the response
timestamp	Integer	Sequential order of the interaction, used for chronological sorting

The variables in Table 3.1 were selected because they capture different aspects of student interaction with the quiz platform. The student-related variables identify individual learners, while the question-related variables describe the course, topic, and difficulty of each question. The performance-related variables, such as correctness and XP earned, represent learning outcomes, whereas response time and timestamp provide information about students' behavioral patterns and engagement. These attributes were subsequently used for data analysis and preparation for the machine learning models described in the following sections.

The final dataset contains 6,318 interaction records from 426 unique students, covering 12 different topics and 3 courses: Data Analysis, Pattern Recognition, and Machine Learning Basics. Each topic has three difficulty levels: Easy, Medium, and Hard. Among the 6,318 responses, 4,160 (65.8%) were correct and 2,158 (34.2%) were incorrect. On average, each student attempted approximately 15 questions, while the number of interactions varied from 3 to 124 attempts. A summary of the main characteristics and distribution of the dataset is presented in Table 3.2.

Table 3.2: Dataset Summary Statistics

Metric	Value
Total interactions	6,318
Unique students	426
Unique topics (skills)	12
Unique courses	3
Difficulty levels	3 (Easy, Medium, Hard)
Correct responses	4,160 (65.8%)
Incorrect responses	2,158 (34.2%)
Average interactions per student	14.8
Minimum interactions per student	3

3.2 Data Cleaning

Before applying the models, the raw dataset was checked and cleaned to make sure the data were correct and consistent. First, the required columns, including student ID, topic, difficulty, correctness, timing information, and timestamp**,** were checked. In some versions of the dataset, the topic column was named skill. In such cases, it was renamed to topic to keep the same column name throughout the experiments.

Next, the correct column was converted into a numerical format. Correct answers were represented as 1, while incorrect answers were represented as **0**. Different text values, such as "Correct," "Incorrect," and "Wrong," were also converted into the corresponding numerical values, regardless of differences in capitalization.

After the conversion, the correct column was checked for missing or unmapped values. Any record that could not be clearly identified as either correct or incorrect was flagged to prevent unclear data from being used during model training and evaluation. Finally, the numbers of correct and incorrect responses were checked against the original data to make sure that no unwanted changes occurred during the cleaning process. After these checks were completed, the dataset was considered ready for chronological ordering and feature engineering**.**

3.3 Chronological Ordering

After completing the data cleaning process, the interaction records were arranged in chronological order using student_id and timestamp**.** First, the records were grouped according to each student. Then, the interactions within each student's group were sorted based on the time at which they occurred. This process ensures that each student's activities are represented as an ordered sequence rather than as a random collection of question attempts.

Maintaining the correct temporal order is very important in Knowledge Tracing because the main purpose of KT is to model how a student's knowledge changes based on their previous learning interactions. The DKT, DKVMN, SAKT, and AKT-lite models used in this study process student interactions as sequences. Therefore, each interaction should appear in the same order in which it actually occurred.

For example, if a student first answers a question from Skill 1, then attempts Skill 3, and later attempts Skill 1 again, the model should receive these interactions in exactly this order. In this way, the model can use the student's earlier responses to understand and predict later performance.

If the interactions were placed in the wrong order, the model could use information from a future interaction while predicting an earlier one. This would not represent the actual learning process and could lead to unreliable results. Therefore, chronological ordering was applied separately for each student before the data were used for sequence modelling. Figure 3.2 illustrates a simplified example of a chronologically ordered interaction sequence for a single student.

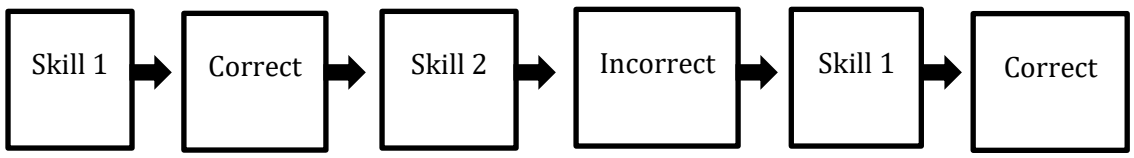


Figure 3.2: Example of a chronologically ordered interaction sequence for a single student.

This ordered sequence provides the basis for applying all four Knowledge Tracing models in this study. It also provides the information needed to calculate the time-gap and attempt-count features described in the following sections.

3.4 Skill Encoding

Deep learning models cannot directly process topic names as text, so the topic information needs to be converted into a numerical form before it can be used by the models. In this study, the dataset contains 12 unique topics, and each topic was assigned a unique numerical skill ID. This encoding allows the models to identify and distinguish between different topics during training while keeping the original topic information consistent.

First, all unique topic names were extracted from the dataset and arranged in alphabetical order. Alphabetical sorting was used to make the assignment of skill IDs fixed and reproducible. Without a fixed ordering, the same topic could receive a different numerical ID in different runs of the data-processing pipeline, which could cause inconsistency during model training and evaluation. After sorting, each topic was assigned an integer index starting from **0**. Therefore, the first topic in the sorted list received index 0, the second received index 1, and this process continued until all 12 topics were assigned their respective indices.

The resulting mapping was stored in a lookup table called SKILL2IDX. This table provides a direct relationship between the original topic name and its numerical skill ID. For example, if a particular topic is assigned index 3, every occurrence of that topic in the dataset will be represented using the same index. This makes the encoded data easier for the models to process and also ensures that the identity of each skill is preserved after conversion from text to numbers.

The same SKILL2IDX mapping was maintained throughout the different stages of the experiment, including data encoding, model training, and model evaluation. Keeping the same mapping is important because changing the skill IDs between these stages could make the model interpret the same topic as different skills. Using one fixed mapping ensures that an interaction belonging to a particular topic always refers to the same skill across the complete experimental pipeline.

The numerical skill ID was then used according to the requirements of each Knowledge Tracing architecture. In the Vanilla DKT model, the skill information was represented using a one-hot based interaction representation together with the correctness of the response. This allows the model to distinguish both the skill attempted and whether the student answered it correctly or incorrectly. On

the other hand, DKVMN, SAKT, and AKT-lite use the skill ID as an index for their embedding representation. In this case, the model learns a numerical vector for each skill during training, allowing related patterns between different skills to be learned from the student interaction data.

Thus, skill encoding provides a consistent numerical representation of the 12 topics and makes the topic information suitable for all four KT architectures used in this study. The use of a fixed skill mapping also ensures that the results from different models and feature settings can be compared fairly because all models receive the same underlying skill representation.

3.5 Time-Gap Feature Engineering

The time-gap feature captures the elapsed duration between consecutive interactions of a student, modeling engagement continuity and memory retention effects. For the $t-$th interaction of a student, the raw time gap ($\Delta\, t_t$) is computed chronologically based on interaction timestamps:

$$\Delta\, t_t = Timestamp_t\ -\ Timestamp_{t-1}$$

For the initial interaction ($t = 1$), where no prior history exists, $\Delta\, t_1 = 0$.

To mitigate the high skewness caused by multi-day learning breaks and stabilize neural network training, a logarithmic transformation is applied to the raw gap values:

$$LogGap_t = ln(1 + \Delta t_t)$$

where the addition of 1 ensures numerical stability for $\Delta\, t_1 = 0$. Subsequently, to align the scale with other model inputs, the transformed values are standardized via z-score normalization:

$$log_gap_z_t = \left\{\frac{LogGap_t - \mu_{gap}}{\sigma_{gap}}\right\}$$

Where μ_{gap} and σ_{gap} denote the mean and standard deviation of the log-transformed time gaps across the dataset, respectively. The resulting feature, $og_gap_z_t$, is constructed without future interaction leakage, preserving strict temporal sequence dynamics for model training.

3.6 Attempt-Count Feature Engineering

The second engineered feature used in this study is the attempt-count feature. This feature represents how many times a student has already attempted a particular skill before the current interaction. The main purpose of using this feature is to capture the student's previous exposure to a skill. A student who has attempted a topic several times may have a different level of familiarity with that topic compared with a student who is attempting it for the first time.

For each interaction, the number of previous attempts was calculated separately for each student and skill. This means that attempts made on one skill do not affect the attempt count of another skill. For example, if a student attempts Skill 1 for the first time, the attempt count for that interaction is 0. If the same student attempts Skill 1 again, the attempt count becomes 1. Similarly, the third attempt on Skill 1 receives a count of 2. Therefore, the feature represents the student's previous practice experience with the specific skill being attempted. In the implementation, this value is stored in the column skill_attempt_no.

The attempt count is calculated using only the interactions that occurred before the current interaction. This is important because the feature should represent the student's knowledge and experience available at the time of prediction. Using the current or future attempts to calculate this

value could introduce information from the future into the model and may lead to data leakage. Therefore, the attempt-count feature is generated according to the chronological order of each student's interactions.

The raw attempt-count values are not directly provided to the neural network. Since different students may have very different numbers of attempts, the values can have a wide range. To make the feature more suitable for model training, the raw values are standardized using z-score standardization**:**

$$\textbf{Attempt Count (standardized)} = \frac{(\textbf{skill}_{\textbf{attempt}_{\textbf{no}}} - \boldsymbol{\mu})}{\boldsymbol{\sigma}}$$

where **μ** represents the mean of the attempt-count values and **σ** represents their standard deviation across the dataset. This transformation places the feature on a common numerical scale, which helps the neural networks process the feature more effectively during training.

The final standardized value is stored as attempt_z and is used as the attempt-count feature in the experiments that include this feature. Thus, the attempt-count feature provides additional information about a student's previous practice with a particular skill, while maintaining the chronological and leakage-safe structure of the dataset.

3.7 Leakage Prevention

A critical part of this study is preventing data leakage during feature engineering and model training. Data leakage occurs when information that would not be available at the time of prediction is unintentionally provided to the model. In Knowledge Tracing, this can lead to unrealistically high performance because the model may receive information that directly or indirectly reveals the answer it is supposed to predict. Therefore, all input features in this study were carefully designed so that they only use information that was available before the target response was observed.

During the early stage of this study, an initial set of experiments included a feature called XP earned. However, further investigation showed that this feature directly represented information related to whether the student answered the question correctly. As a result, the model achieved an unrealistically high accuracy because it was receiving information that was closely related to the target answer. This indicated that XP earned could introduce data leakage into the experiment. Therefore, this feature was not used in the final Knowledge Tracing experiments.

The two engineered features used in the final experiments, time gap and attempt count, were therefore calculated using only previously completed interactions. For the time-gap feature, the model uses the time difference between the student's current interaction and their previous completed interaction. It does not use the time information associated with the response that is being predicted. This ensures that the feature represents information that would already be available before making the next prediction.

Similarly, the attempt-count feature is calculated from the number of previous attempts made by the same student on the same skill. The current attempt is not included in this calculation. For example, the first attempt on a skill receives an attempt count of 0, while later attempts are counted based only on the interactions that occurred before the current one. Therefore, the feature represents the student's previous exposure to the skill without using information from the current response.

Most importantly, the correctness of the response being predicted is never used as an input feature. At prediction time, the model may know which skill the student is going to answer next, but it does

not know whether the response will be correct or incorrect. The actual correctness value is only available after the student completes the question and is then used for model training and evaluation. This maintains the correct prediction order and prevents the model from learning directly from the target answer.

Figure 3.3 illustrates this leakage-safe feature construction process. The features used for prediction are generated only from the student's completed interaction history, while the outcome of the next interaction remains unknown until the student answers the question. In this way, the experimental setup more closely represents how a Knowledge Tracing model would operate in a real learning environment.

.

.

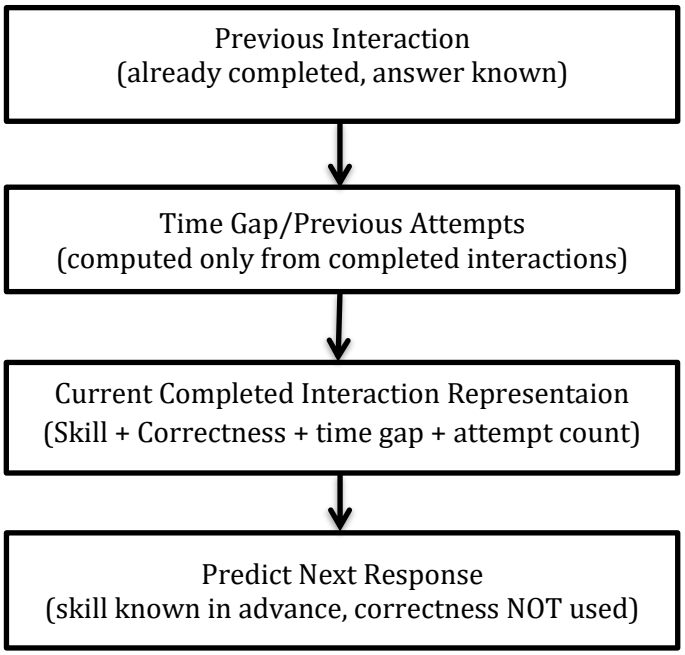


Figure 3.3: Leakage-safe feature construction using only previously completed interactions to predict the next response .

3.8 System Architecture

The proposed system combines the data-processing steps, engineered features, and Knowledge Tracing models into a single experimental pipeline. The overall process starts with the interaction data collected from the quiz platform and ends with the prediction of a student's probability of answering the next question correctly.

The first stage of the pipeline is the raw student interaction data, which contains information such as student ID, skill, correctness, difficulty, response time, and timestamp. These records are first checked and converted into a consistent format. The cleaned records are then arranged according to each student's chronological interaction history.

After chronological ordering, two additional behavioural features are calculated: time gap and attempt count. The time-gap feature captures the amount of time since the student's previous interaction, while the attempt-count feature represents the student's previous exposure to the same skill. These features are standardized before being used as model inputs.

Based on the available features, four input variants were prepared for the experiments:

- **Vanilla** – skill and correctness information only
- **+ Time Gap** – skill, correctness, and time-gap feature
- **+ Attempt Count** – skill, correctness, and attempt-count feature
- **+ Time & Attempt** – skill, correctness, time-gap, and attempt-count features

Each input variant was evaluated with four Knowledge Tracing architectures: DKT, DKVMN, SAKT, and AKT-lite. This resulted in 16 model-and-feature combinations, allowing the effect of the engineered features to be compared across different model architectures. During prediction, each model uses the student's previous interaction history to estimate the probability that the student will answer the target skill correctly. The actual response is used only after the prediction stage for training or evaluation. Therefore, the prediction is based on the information available from the student's previous learning history rather than information from the future. Figure 3.4 shows the complete pipeline provides a consistent framework for comparing the four Knowledge Tracing architectures and for examining whether the additional behavioural features improve prediction performance.

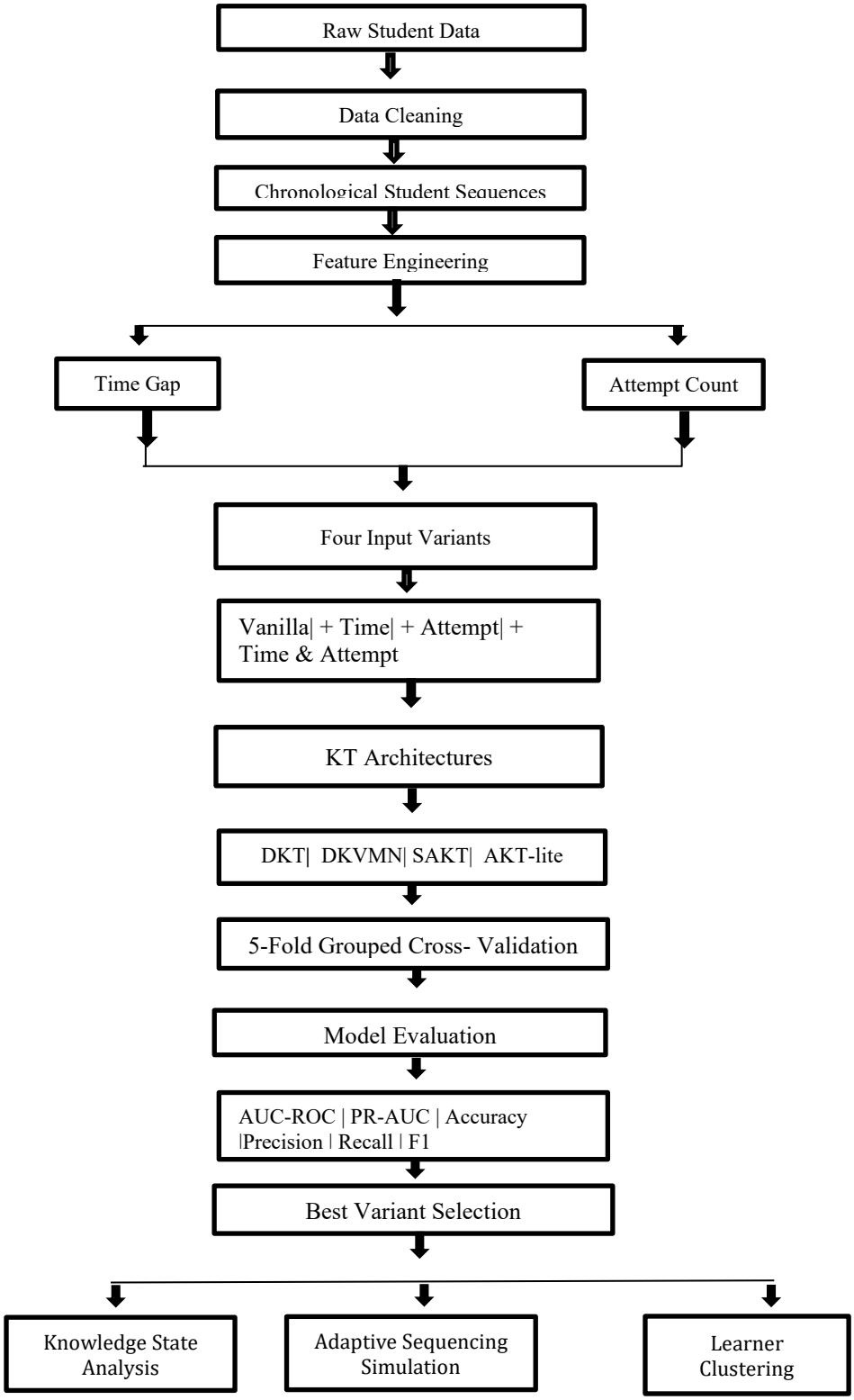


Figure 3.4: Overall proposed framework, from raw student data collection through to the final downstream analyses.

3.9 Vanilla DKT

The Vanilla DKT model is used as the baseline architecture in this thesis and follows the original Deep Knowledge Tracing (DKT) approach. It uses a Long Short-Term Memory (LSTM) network to model how a student's knowledge state changes over time based on their raw interaction sequence.

At each step of the student's sequence, the model receives two types of information: the skill attempted and whether the student's response was correct. These two pieces of information are combined into a single one-hot interaction representation. The length of this one-hot vector is twice the number of skills because it needs to represent every possible combination of a skill and its correctness outcome separately. The position that matches the current skill and response outcome is assigned a value of 1, while all other positions are set to 0. Figure 3.5 Vanilla DKT architecture [08] shows raw student interaction to the final predicted probability of a correct response.

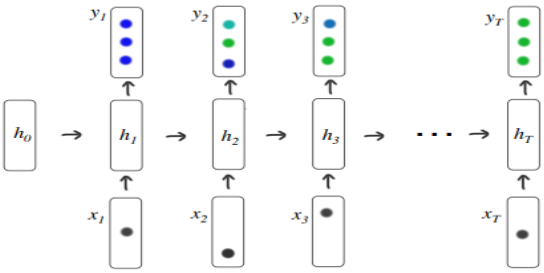


Figure 3.5: Vanilla DKT architecture.

- **Input Representation (x_t):** The input at time step **t** represents the student's interaction tuple $x_t = (q_t , a_t)$, where q_t denotes the target question/skill and $a_t \in \{0,1\}$ indicates incorrect or correct response status, typically encoded as a one-hot vector.

- **Hidden State & Knowledge Dynamics (h_t):** A Recurrent Neural Network (RNN) or Long Short-Term Memory (LSTM) network updates the latent knowledge state $h_t \in \mathbb{R}^d$ based on past interaction history:

$$h_t = \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$$

- **Prediction Layer (y_t):** The output layer projects h_t to predict performance probabilities across all exercises in the dataset:

$$y_t = \sigma(W_{yh}h_t + b_y)$$

3.10 DKVMN

While Vanilla DKT represents everything it has learned about a student through a single hidden state, the Dynamic Key-Value Memory Network (DKVMN) uses a separate memory slot for each skill. This allows the model to track the student's mastery of individual skills more directly. DKVMN uses two different memory structures called key memory and value memory. The key memory is static, meaning that it does not change during prediction. It contains a learned embedding for each skill and is mainly used to identify which memory slots are related to the skill in the current query. In contrast, the value memory is dynamic and is updated after each interaction. It stores the model's current understanding of the student's mastery of different skills using a set of vectors instead of a single value. Figure 3.6: DKVMN architecture[62], showing the read step (prediction) and write step (memory update) for a single interaction.

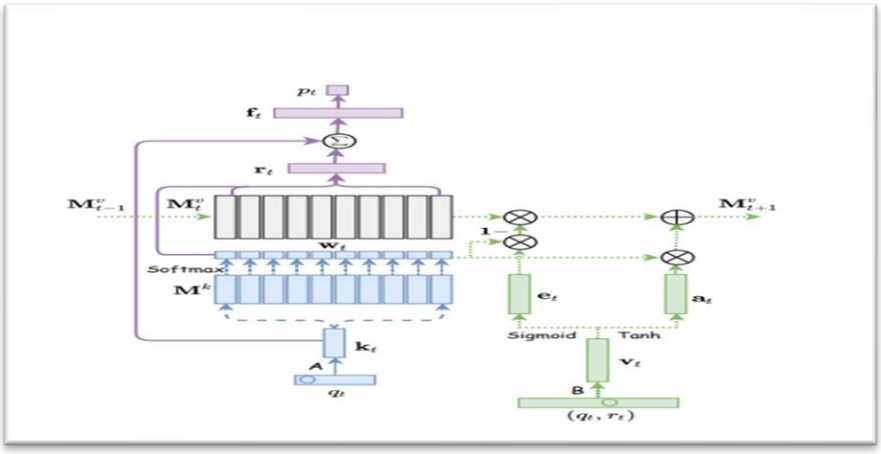


Figure 3.6: DKVMN architecture.

- **Key Memory Matrix ($\mathbf{M^k}$):** A static memory matrix $M^k \in \{R\}^{\{N \times d_k\}}$ that captures underlying domain concepts or latent skills across exercises.

- **Value Memory Matrix ($\boldsymbol{M_t^v}$):** A dynamic memory matrix $M_t^v \in \{R\}^{\{N \times d_v\}}$ updated continuously to store and update student mastery levels over time for each concept.

- **Correlation Weight Computation** ($\boldsymbol{w_t}$): Softmax attention calculates correlation weights between input exercise q_t and key concepts:

$$W_t(i) = \{Softmax\}(k_t^T M^k(i))$$

- **Read and Write Dynamics:**

- **Read (Mastery Extraction):** Student mastery state r_t is retrieved via $r_t = \sum_{i=1}^{N} w_t(i) M_t^v(i)$

- **Write (State Update):** The Value matrix updates via Erase (e_t) and Add (a_t) memory operations:

$$M_t^v(i) = M_{\{t-1\}}^v(i)[1 - \mathrm{w_t}\,(i)e_t] + w_t\,(i)a_t$$

3.11 SAKT

Unlike Vanilla DKT and DKVMN, which process a student's interactions sequentially using a recurrent approach, the Self-Attentive Knowledge Tracing (SAKT) model does not use recurrence. Instead, it uses a self-attention mechanism to directly examine the student's previous interaction history when making a prediction. Each past interaction in the student's sequence is converted into a key embedding and a value embedding based on the skill that was attempted and whether the response was correct. A positional encoding is also added to these embeddings so that the model can keep track of the order of the interactions, since the attention mechanism itself does not naturally

contain information about sequence order. Separately, the skill that the model needs to predict, which is the skill of the next question the student is expected to attempt, is converted into a query embedding and combined with its corresponding positional encoding. Figure 3.7: SAKT architecture[10], showing how the query for the next skill attends over the causally masked history of past interactions.

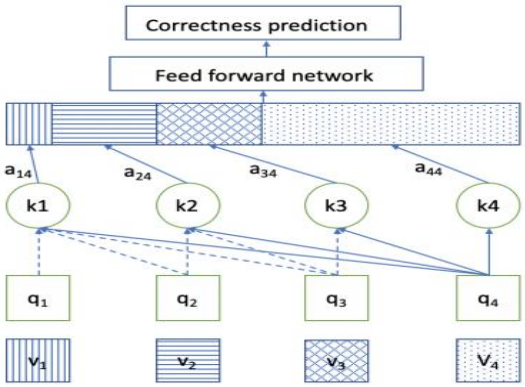


Figure 3.7: SAKT architecture.

- **Embedding Layer:** Interleaved interaction history $y_i = e_i + r_i \times E$ and target exercise q_t are projected into dense embedding spaces.

- **Self-Attention Mechanism:** Scaled Dot-Product Attention measures relevant historical interactions relative to the current exercise:

$$Q = q_t W^Q, \qquad K = y_{\{1:t-1\}} W^K, \qquad V = y_{1:t-1} W^v$$

$$S = Softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

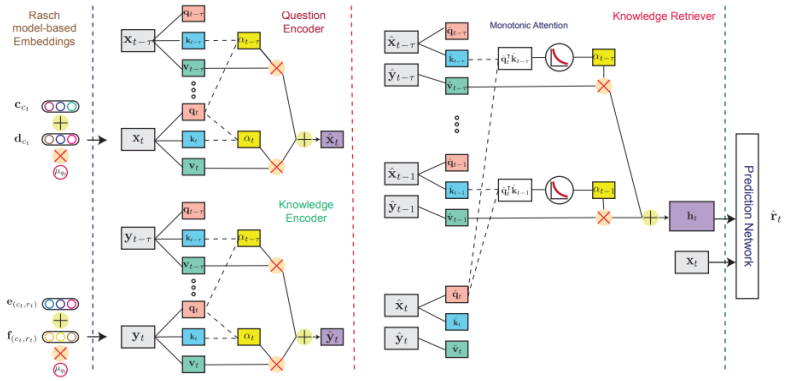
- **Feed-Forward Prediction Network:** The context vector S passes through residual connections and a Position-wise Feed-Forward Network (FFN) to predict performance:

$$p_t = \sigma(W_2 \cdot ReLU(W_1 S + b_1) + b_2)$$

3.12 AKT-lite

AKT-lite builds on the self-attention mechanism used in SAKT and introduces an additional idea from the original AKT architecture. The main idea is that, when other factors are similar, more recent past interactions should have a stronger influence on the current prediction than interactions that happened much earlier. This is similar to the natural way human memory can become weaker

over time. The initial steps of AKT-lite are similar to SAKT. Past interactions are converted into query, key, and value projections using learned linear transformations, and positional encodings are added so that the model can retain information about the order of the interactions. The model then calculates the raw attention scores between the query, which represents the skill to be predicted next, and each key from the student's previous interaction history using the standard scaled dot-product attention method. Figure 3.8: AKT-lite architecture[11], extending SAKT's self-attention with a learned, time-distance decay term applied before the causal mask and softmax.



.

Figure 3.8: AKT-lite architecture

- **Rasch Model-Based Embeddings:** Incorporates exercise difficulty variation $d_{\{c_t\}}$ alongside base concept embeddings $e_{\{c_t\}}$:

$$x_t = e_{\{c_t\}} + \mu_{\{c_t\}} \cdot d_{\{c_t\}}$$

- **Monotonic Attention Mechanism:** Applies exponential decay over time gaps $d(t, \tau)$ to prioritize recent interaction context:

$$\alpha_{t,\tau} \propto exp\big(-\theta \cdot d(t,\tau)\big) \cdot exp\left(\frac{Q_t K_\tau^T}{\sqrt{d}}\right)$$

- **Knowledge Retriever & Output Module:** Computes context-aware knowledge representation h_t from encoder blocks and computes performance likelihood:

$$\{\hat{r}\}_t = \sigma(W \cdot [h_t; x_t] + b)$$

3.13 Grouped 5-Fold Cross-Validation

To fairly evaluate all model and feature combinations, this study uses 5-Fold Cross-Validation with one important modification: the data is divided based on student identity rather than individual interactions. In a standard cross-validation approach, the dataset is divided into five roughly equal parts, where each part is used as the test set once while the model is trained on the other four parts.

However, if the splitting is done at the level of individual interactions, interactions from the same student could be placed in both the training and test sets. Since Knowledge Tracing models can learn patterns related to individual student behaviour, testing the model on a student whose other interactions were already included during training could result in an overly optimistic and unrealistic performance estimate.

To prevent this problem, Grouped 5-Fold Cross-Validation divides the data according to student identity. All interactions belonging to one student are treated as a single unit, and the complete set of students is divided into five groups. In each of the five rounds, four groups, representing roughly 80% of the students, are used for training, while the remaining group, representing the other 20%, is used only for testing. Each student appears in the test set exactly once across the five rounds, and no interactions from the same student are divided between the training and testing sets within a single round. As a result, the model is always evaluated on students whose behaviour was not seen during training. This provides a more realistic estimate of how the model would perform when it is applied to a completely new student in a real-world deployment.

This grouped validation strategy was selected for the current larger dataset instead of the Leave One Student Out (LOSO) approach used in the earlier, smaller-scale experiments of this research. In LOSO, a separate model needs to be trained for each individual student, which becomes computationally expensive as the number of students increases. In comparison, Grouped 5-Fold Cross-Validation requires only five training rounds while still providing a reliable estimate of the model's ability to generalize to unseen students, making it more practical for the larger dataset used in this study.

3.14 Experimental Setup

This section describes the practical setup used to conduct all the experiments discussed in the earlier parts of this chapter. It is divided into three parts. Section 3.14.1 explains the overall experimental design, including which Knowledge Tracing architectures are combined with which input variants, and presents these combinations as a structured experimental setup. Section 3.14.2 describes the evaluation metrics used to compare the different combinations and explains why these metrics are suitable for this task. Finally, Section 3.14.3 provides the specific hyper parameters and configuration settings used for each model, allowing the experiments presented in this thesis to be reproduced reliably.

3.14.1 Experimental Design

The experimental design of this thesis uses a full factorial combination of four Knowledge Tracing architectures and four input-feature variants. Each combination is evaluated using the Grouped 5-Fold Cross-Validation procedure described in Section 3.13. This results in a total of sixteen different experimental configurations, obtained from 4 architectures × 4 feature variants**,** with each configuration evaluated using 5-fold cross-validation. The four architectures included in the experiments are Vanilla DKT, DKVMN, SAKT, and AKT-lite, as described in Sections 3.9 to 3.12. The four input-feature variants are Vanilla, which uses only skill and correctness information, + Time, which adds the standardized time-gap feature described in Section 3.5, + Attempt**,** which adds the standardized attempt-count feature from Section 3.6, and + Time & Attempt**,** which uses both engineered features together. Each architecture is trained and tested with all four input variants. This design makes it possible to examine the effects of the different architectures and engineered features both separately and in combination, instead of evaluating them under only one fixed input

configuration. Table 3.3 presents this experimental matrix, confirming that all sixteen architecture-and-variant combinations were included in the evaluation.

Table 3.3: Experimental Matrix

Architecture	**Vanilla**	**+ Time**	**+ Attempt**	**+ Time & Attempt**
DKT	✓	✓	✓	✓
DKVMN	✓	✓	✓	✓
SAKT	✓	✓	✓	✓
AKT-lite	✓	✓	✓	✓

3.14.2 Evaluation Metrics

To evaluate the performance of the models, this study uses six complementary metrics instead of depending on a single measure. This is important because no single metric can fully describe the performance of a Knowledge Tracing model on a moderately imbalanced dataset such as this one, where 65.8% of the responses are correct and 34.2% are incorrect**.** Accuracy measures the overall proportion of predictions that are correctly classified and is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP (True Positives) and TN (True Negatives) represent correctly predicted correct and incorrect responses, while FP (False Positives) and FN (False Negatives) represent the corresponding incorrect predictions. Although accuracy is easy to understand, it can be misleading when the classes are imbalanced. For example, a model that always predicts a response as "correct" could still achieve relatively high accuracy without actually learning meaningful patterns about the student's knowledge.

Precision measures the proportion of responses predicted as correct that are actually correct and is defined as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

On the other hand, Recall measures the proportion of actually correct responses that are successfully identified by the model:

$$Recall = \frac{TP}{(TP + FN)}$$

Precision and Recall can have a trade-off because increasing Recall can be achieved by predicting "correct" more frequently, which may also increase the number of false positives and reduce Precision. The F1-score combines Precision and Recall into one measure by using their harmonic mean:

$$F1 = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)}$$

Another important metric used in this study is AUC-ROC (Area Under the Receiver Operating Characteristic Curve). Unlike Accuracy, Precision, Recall, and F1-score, which are generally evaluated using a fixed decision threshold such as 0.5, AUC-ROC measures how well the model can distinguish between correct and incorrect responses across different possible thresholds. Its value ranges from 0.5, which represents performance similar to random guessing, to 1.0, which represents perfect discrimination. AUC-ROC is used as the primary comparison metric throughout this thesis because it is threshold-independent and is suitable when the two classes are not perfectly balanced.

PR-AUC (Area Under the Precision-Recall Curve) is also reported as a complementary metric to AUC-ROC. It can be particularly useful when the positive and negative classes are imbalanced, as in the current dataset. While AUC-ROC considers the model's ability to distinguish between both classes across different thresholds, PR-AUC gives more direct attention to the model's Precision and Recall, especially for the less frequent class. Therefore, PR-AUC provides an additional way to assess the model's performance and helps cross-check the results obtained from AUC-ROC.

Together, the six metrics, Accuracy, Precision, Recall, F1-score, AUC-ROC, and PR-AUC, provide a more complete view of the predictive performance of each model. All six metrics are reported for each of the sixteen experimental configurations described in Section 3.14.1, allowing the different architecture and feature combinations to be compared from multiple perspectives.

3.14.3 Model Configuration

All four architectures were trained using the Adam optimizer with a binary cross-entropy loss function**,** which compares the predicted probability of a correct response with the actual observed outcome. Since the interaction sequences are not the same length for every student, each model was trained using one complete student sequence at a time instead of using fixed-size mini-batches. This approach avoids introducing artificial padding or truncation into the students' interaction histories. The specific architecture-level hyperparameters used for each model are described below.

Vanilla DKT uses a single-layer LSTM with a hidden dimension of 32, followed by a linear output layer that produces one score for each skill. These scores are converted into probabilities using a sigmoid activation function. The model was trained for 20 epochs with a learning rate of 0.01**.** DKVMN uses separate key and value memories, with an embedding dimension of 16 for both memories. The number of memory slots is set to the total number of skills in the dataset, which is 12. Its prediction network includes a hidden layer with 32 units before the final sigmoid output. Because DKVMN uses a sequential read-and-write process that cannot be parallelized across a sequence in the same way as the other architectures, it was trained for 10 epochs to keep the training time manageable, while using the same learning rate of 0.01.

SAKT uses an embedding dimension of 32 for its skill, interaction, and positional embeddings and uses 2 attention heads in its multi-head self-attention layer. It was trained for 20 epochs with a learning rate of 0.01**.** AKT-lite also uses an embedding dimension of 32 and 2 attention heads, following the same basic configuration as SAKT. In addition, it includes a single learned scalar parameter that controls the strength of the time-distance decay term. This parameter was initialized to 0.05 at the beginning of training. AKT-lite was also trained for 20 epochs using a learning rate of 0.01**.**

Across all four architectures, the model weights were initialized using PyTorch's default initialization scheme. No dropout or additional regularization was used in the primary experimental comparison so that the differences between the architectures could be evaluated as directly as possible. A fixed random seed was also used throughout all experiments to ensure that the data splitting, model initialization, and training process remained reproducible across different runs.

CHAPTER 4

RESULTS

This chapter presents the experimental results of four Knowledge Tracing architectures: DKT, DKVMN, SAKT, and AKT-lite. The same grouped 5-fold cross-validation procedure was used to evaluate all four architectures so that their reported performance can be compared under a common evaluation setting. For each architecture, four input-feature variants were considered: Vanilla, + Time Gap, + Attempt Count, and + Time & Attempt. This design makes it possible to examine not only which architecture performs best but also whether the additional temporal and attempt-related information changes model performance. The chapter also examines how the best-performing configurations represent learner knowledge over time. The analysis includes feature-wise performance comparisons, knowledge-state heatmaps, an adaptive sequencing simulation, learner clustering, cluster-level skill profiles, and an overall comparison of the evaluated architectures.

4.1 DKT Results

Table 4.1: DKT performance across the four input-feature variants (pooled held-out AUC-ROC, grouped 5-fold cross-validation).

Variant	Pooled 5-Fold AUC
Vanilla	0.749
+ Time Gap	0.761
+ Attempt Count	0.766
+ Time & Attempt	0.775

The DKT results show a clear improvement when the input representation is extended beyond the Vanilla setting. The Vanilla configuration obtains a pooled AUC-ROC of 0.749. Adding Time Gap increases the value to 0.761, while adding Attempt Count increases it to 0.766. These changes indicate that both engineered features contain predictive information that can help DKT distinguish between correct and incorrect responses in the evaluated data.

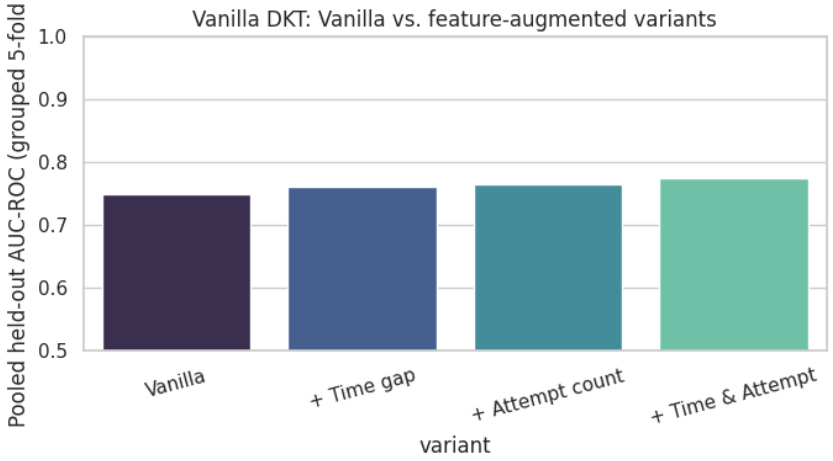


Figure 4.1: DKT Performance Across Input Feature Variants.

AUC-ROC measures how well the model separates positive and negative outcomes across different decision thresholds. Therefore, a higher value means that the model is better able to rank interactions associated with correct responses above those associated with incorrect responses. In this

comparison, Attempt Count provides a slightly larger improvement over Vanilla than Time Gap. The result suggests that the number of previous attempts carries more useful predictive information than elapsed time for this DKT configuration. This is an observation from the evaluated experiment and should not be interpreted as a general rule for all datasets or DKT implementations.

The combined + Time & Attempt configuration reaches 0.775 and gives the strongest result among the four DKT variants. Its performance is higher than both single-feature variants, which indicates that the two engineered features contribute complementary information when they are used together. Table 4.1 provides the exact numerical comparison, while Figure 4.1 gives the same comparison in visual form. The figure is therefore used as a visual aid rather than as an additional experiment or a separate source of results.

4.1.1 DKT Knowledge-State Analysis

The knowledge-state visualization is used to examine how DKT updates its estimate for individual skills as the student produces more interactions. Unlike a single overall score, the heatmap keeps a separate predicted state for each of the 12 skills. This makes it possible to inspect whether different skills follow similar or different trajectories during the sequence.

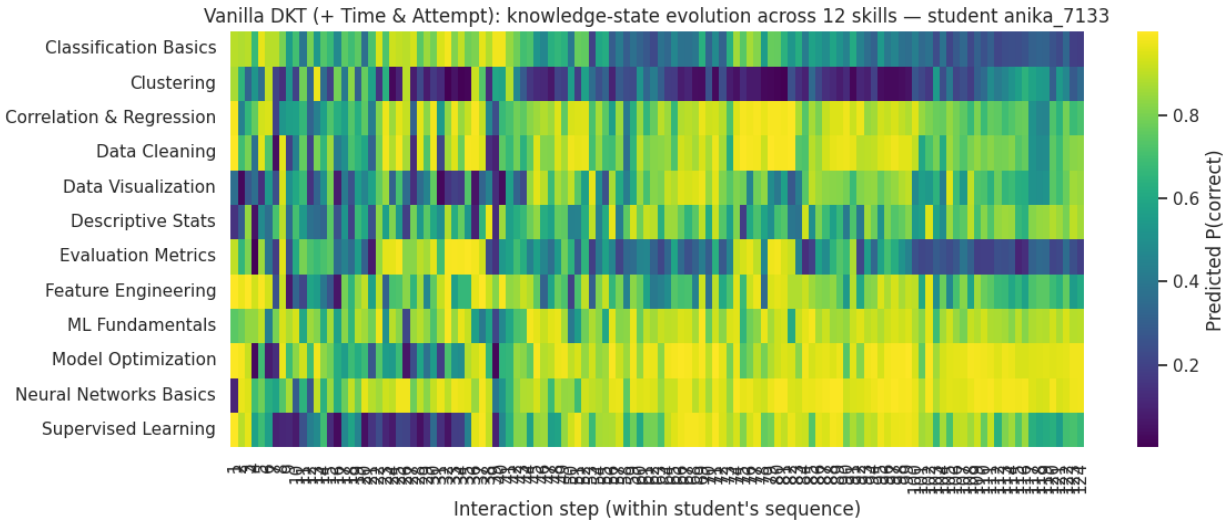


Figure 4.2: Knowledge State Interaction Heatmap of DKT with Time & Attempt Variant.

Figure 4.2 illustrates these predicted states across the student's interaction history. Each column represents one point in the sequence and each row represents a skill. Brighter regions correspond to higher predicted P(correct) values, whereas darker regions correspond to lower predicted values. The color changes therefore provide a visual representation of how the model's estimated probability changes after additional interactions.

Feature Engineering and Neural Networks Basics show a general movement toward higher predicted probabilities in later parts of the sequence. In contrast, the Clustering row remains relatively dark for a longer period before becoming brighter. For this demonstration student, DKT therefore estimates slower improvement in Clustering than in several other skills. The figure should be interpreted as a student-level illustration of model predictions rather than as evidence that the student has objectively mastered or failed a skill.

4.2 Adaptive Sequencing Analysis

The adaptive sequencing analysis demonstrates how a model prediction can be converted into a simple rule for selecting the next learning skill. The purpose of this simulation is not to retrain DKT but to use the predicted P(correct) values produced by the best DKT configuration. A desirable-difficulty band of 0.55-0.75 is defined for the simulation. A predicted value inside this interval is treated as an appropriate level for the next recommendation because it represents a learner state that is neither too low nor too high according to the chosen rule.

Table 4.2 summarizes an eight-step session generated for the same demonstration student using + Time & Attempt. The final column records whether each predicted probability falls inside the target band. This column is a direct rule-based check: True means the value is between 0.55 and 0.75, while False means it falls outside that interval.

The table therefore connects the numerical model output with the decision used by the simulated sequencing procedure.

Six of the eight recommended steps fall within the target band. Step 3 has a predicted P(correct) of 0.502 and is therefore slightly below the lower boundary. Step 7 has a value of 0.762 and is slightly above the upper boundary. Both values are close to the limits, so the simulation does not show extreme mismatch between the selected skills and the intended difficulty range. Feature Engineering appears repeatedly because its predicted probability remains within or close to the target band during several stages of the sequence. This example illustrates how predicted knowledge states can support adaptive selection when a clear recommendation rule is defined.

Table 4.2: Simulated adaptive sequencing session for student anika_7133 using DKT (+ Time & Attempt), target band = 0.55-0.75.

Step	Recommended Skill	Predicted P(correct)	In Desirable Band
1	Supervised Learning	0.675	True
2	Data Visualization	0.667	True
3	Feature Engineering	0.502	False
4	Feature Engineering	0.589	True
5	Feature Engineering	0.677	True
6	Feature Engineering	0.671	True
7	Correlation & Regression	0.762	False
8	Feature Engineering	0.671	True

The results in Table 4.2 show that six of the eight recommended steps fall within the 0.55-0.75 target band. The value True in the final column means that the predicted P(correct) falls inside this band. The value False means that the prediction falls outside the band. Step 3 has a predicted probability of 0.502, so it is slightly below the target range. Step 7 has a value of 0.762, so it is slightly above the target range. Both values are close to the boundary. Feature Engineering is recommended repeatedly because its predicted probability remains within or close to the target range during several stages of the simulation. This example shows how the predicted knowledge state can be used to select skills at an intended level of difficulty.

4.3 Learner Clustering

Learner clustering is used to move from an individual student view to a population-level view. For each student, the final predicted mastery values across the 12 skills form a mastery vector. These vectors are standardized before further analysis so that the clustering procedure works with a common numerical scale. The standardized vectors are then projected into two dimensions using t-SNE. K-Means with k = 4 is applied to the standardized mastery vectors to divide learners into four groups.

Figure 4.3 presents the resulting two-dimensional projection. The main purpose of this visualization is to identify groups of students whose predicted mastery profiles are relatively similar. The distance between points in a t-SNE plot should not be treated as a direct measurement of mastery. Instead, the plot provides a compact representation of similarity patterns that can be examined further through the skill-level profiles in Figure 4.4.

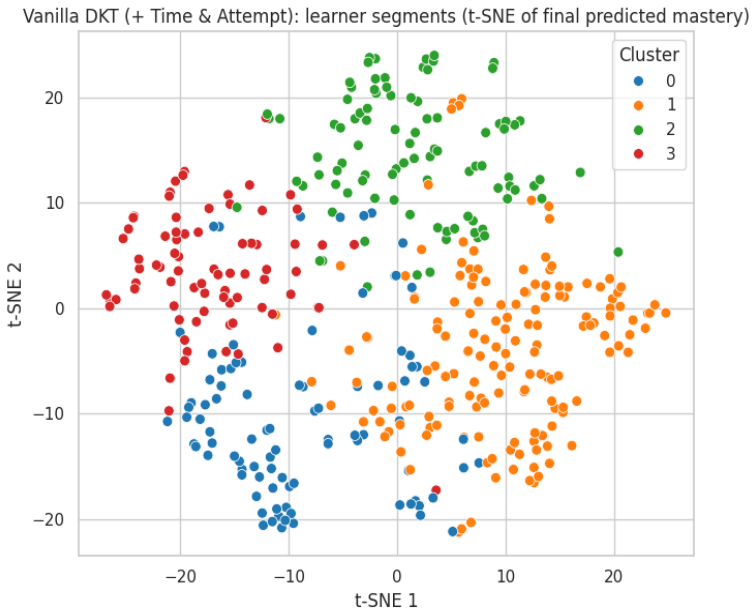


Figure 4.3: DKT t-SNE Projection of Learner Mastery Clusters.

Figure 4.3 shows four learner groups containing 91, 159, 96, and 80 students, respectively. Cluster 1 is the largest group and is concentrated mainly on the right side of the projection. Clusters 0 and 3 are more concentrated toward the lower-left region, while Cluster 2 is distributed more broadly across the upper-middle area. These spatial differences indicate that the students do not share exactly the same final predicted mastery pattern. However, the t-SNE plot alone cannot identify which particular skills create these differences. That interpretation requires the cluster-level skill analysis presented in Figure 4.4.

4.4 Cluster Skill-Mastery Profiles

Figure 4.4 provides the skill-level explanation that is not available from the t-SNE projection alone. Each row corresponds to a learner cluster and each column corresponds to one of the 12 skills. The displayed value is the mean predicted P(correct) for that cluster-skill combination. Higher values indicate that the model assigns a higher predicted probability of a correct response for that skill within the cluster. This allows the groups to be compared in terms of both overall level and specific strengths or weaknesses.

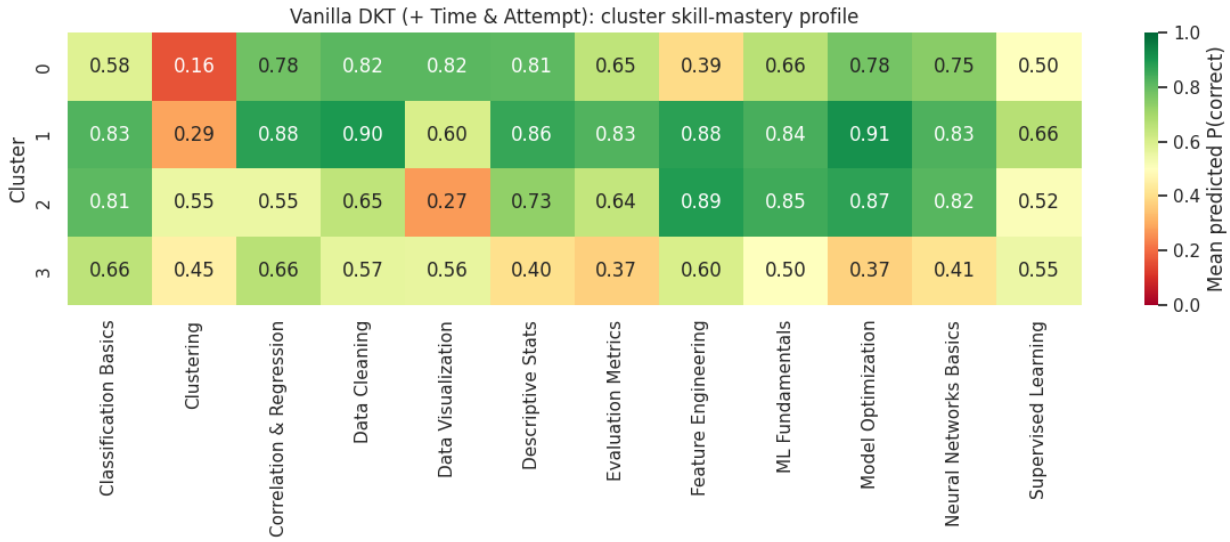


Figure 4.4: DKT Cluster Skill Mastery Profiles.

Cluster 1 has the strongest overall profile, with most skill values between 0.60 and 0.91. Its main weakness is Clustering at 0.29. Cluster 0 has moderate-to-high predicted mastery on many skills but lower values for Clustering at 0.16 and Feature Engineering at 0.39. Cluster 2 also has mostly moderate-to-high values, but Data Visualization is notably lower at 0.27. Cluster 3 has several comparatively weak areas, including Evaluation Metrics at 0.37, Model Optimization at 0.37, and Neural Networks Basics at 0.41.

These profiles show that learner groups differ in more than a single overall mastery level. The location of the lower values also changes across clusters. For example, the main weakness of Cluster 0 is concentrated in Clustering and Feature Engineering, whereas Cluster 2 shows a particularly low predicted value for Data Visualization. Clustering remains relatively low across all four groups. Even so, these model-generated profiles cannot establish that Clustering is inherently more difficult. They only show that the evaluated DKT configuration assigns lower predicted probabilities to that skill for these learner groups.

4.5 DKVMN Results

Table 4.3: DKVMN performance across the four input-feature variants (pooled held-out AUC-ROC, grouped 5-fold cross-validation).

Variant	Pooled 5-Fold AUC
Vanilla	0.676
+ Time Gap	0.666
+ Attempt Count	0.684
+ Time & Attempt	0.679

The DKVMN results show a different response to feature augmentation than DKT. The Vanilla configuration reaches a pooled AUC-ROC of 0.676. Adding Time Gap lowers the result to 0.666. In contrast, adding Attempt Count raises the result to 0.684, which is the highest DKVMN value among the four tested variants. The combined + Time & Attempt configuration reaches 0.679 and therefore remains below + Attempt Count alone.

The comparison indicates that Attempt Count is the most useful of the tested engineered features for DKVMN on the evaluated data. The improvement from 0.676 to 0.684 is larger than the change produced by Time Gap, which is negative relative to Vanilla. The combined configuration also does not exceed the Attempt Count-only configuration. These results suggest that adding more features does not automatically improve performance; the usefulness of an added feature depends on the architecture and the evaluated data.

The lower results for the Time Gap variants are an observed pattern rather than a confirmed explanation of model behavior. The available results do not show why the time-based feature produces a lower AUC-ROC in DKVMN.

Figure 4.5 presents the four values visually, while Table 4.3 should be used when exact numerical differences are required.

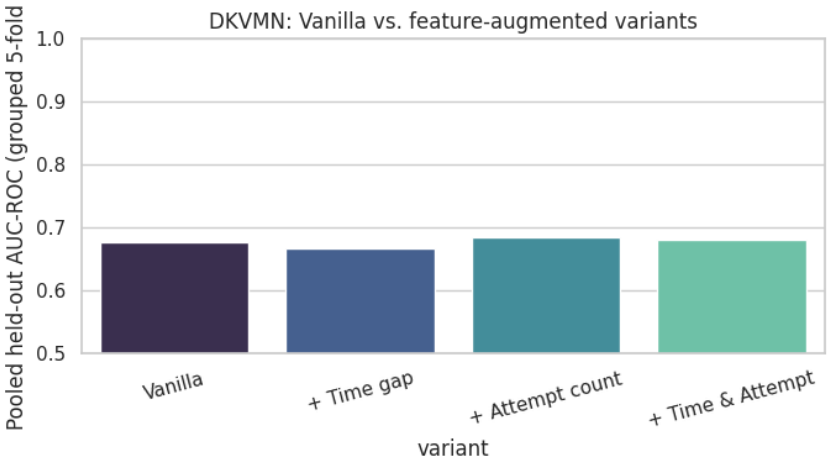


Figure 4.5: DKVMN Performance Across Input Feature Variants.

The visual comparison in Figure 4.5 makes the relative effect of the four DKVMN input-feature variants easier to inspect. In particular, the Attempt Count bar is the highest among the four configurations, while Time Gap is the lowest.

4.5.1 DKVMN Knowledge-State Analysis

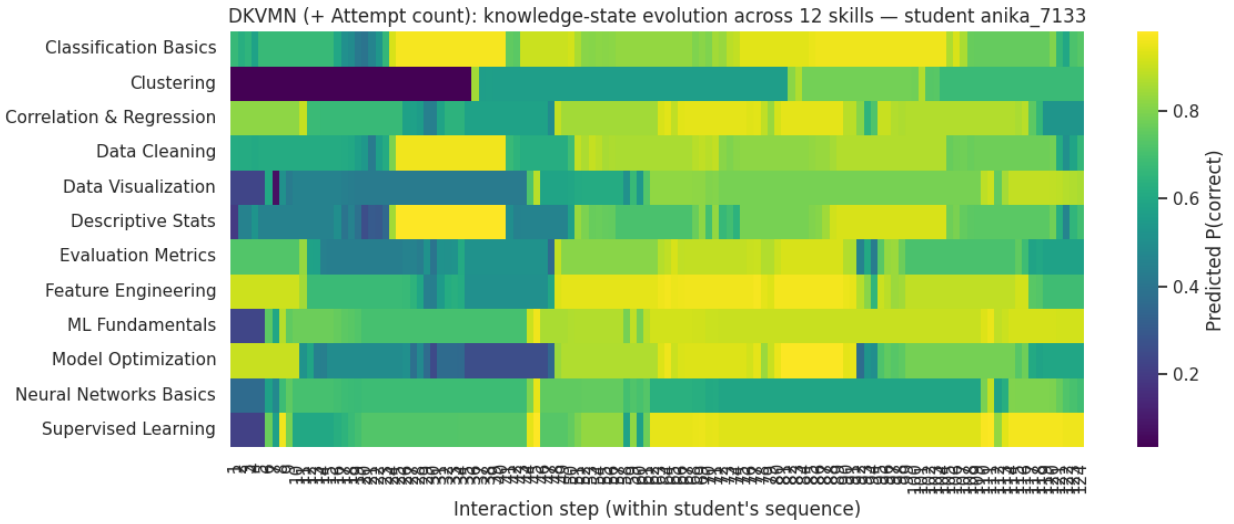


Figure 4.6: Knowledge State Interaction Heatmap of DKVMN with Attempt Count Variant.

The DKVMN knowledge-state visualization is used to inspect how predicted mastery changes over the interaction sequence when the best-performing + Attempt Count configuration is used. Figure 4.6 displays the predicted state separately for all 12 skills. This makes it possible to identify skills that remain at lower predicted levels for longer periods as well as skills whose estimates change more quickly.

In Figure 4.6, the Clustering row remains at a relatively low predicted level for an extended part of the sequence before increasing. Several other skills also move between lower and higher predicted probability regions at different stages. Compared with the smoother variation visible in the DKT heatmap, the DKVMN pattern appears more block-like. This provides a useful visual contrast between the two model outputs.

The block-like appearance is compatible with changes in the model's memory representation, but the heatmap alone cannot establish the exact internal mechanism responsible for every transition. The safest interpretation is therefore that Figure 4.6 visualizes the predicted knowledge states generated by DKVMN. It should not be used by itself to infer how a particular memory component operates internally.

4.6 SAKT Results

Table 4.4: SAKT performance across the four input-feature variants (pooled held-out AUC-ROC, grouped 5-fold cross-validation).

Variant	Pooled 5-Fold AUC
Vanilla	0.616
+ Time Gap	0.589
+ Attempt Count	0.609
+ Time & Attempt	0.607

SAKT produces the lowest overall performance among the four evaluated architectures under the reported pooled AUC-ROC metric, and its feature-variant pattern is also different from the other models. The Vanilla configuration reaches 0.616. Adding Time Gap reduces the value to 0.589. Adding Attempt Count gives 0.609, while the combined + Time & Attempt configuration gives 0.607. Thus, none of the three engineered-feature variants exceeds the Vanilla configuration.

This result is important because it shows that feature augmentation has architecture-dependent effects. For DKT, the combined features produce the best result. For DKVMN and AKT-lite, Attempt Count gives the best configuration. In contrast, SAKT performs best without any of the tested engineered additions. On the evaluated dataset and configuration, the additional features therefore do not provide a performance advantage for SAKT.

The result should be interpreted within the scope of this experiment. A lower score for an engineered variant does not mean that the corresponding information is universally unhelpful for attention-based Knowledge Tracing. It only shows that the tested representation and configuration did not improve the reported pooled AUC-ROC. Figure 4.7 is included as a visual comparison of the four SAKT variants, while Table 4.4 contains the exact values.

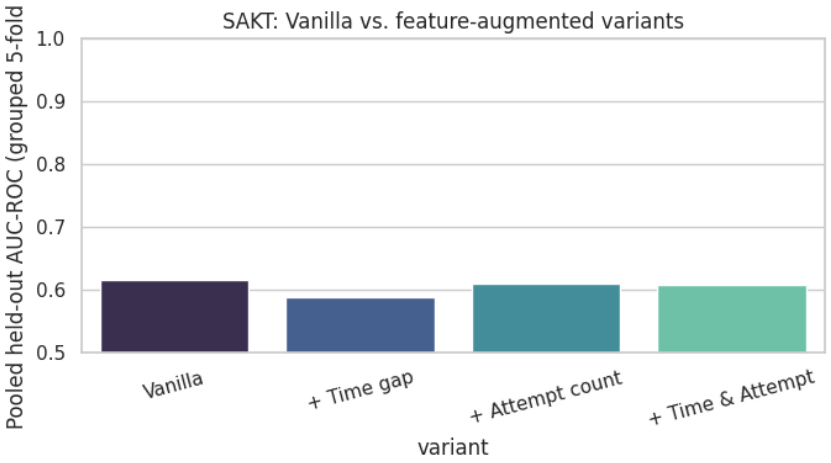


Figure 4.7: SAKT Performance Across Input Feature Variants.

The visual comparison in Figure 4.7 makes the SAKT pattern clear: the Vanilla configuration is the highest of the four tested variants, while the Time Gap variant is the lowest.

4.6.1 SAKT Knowledge-State Analysis

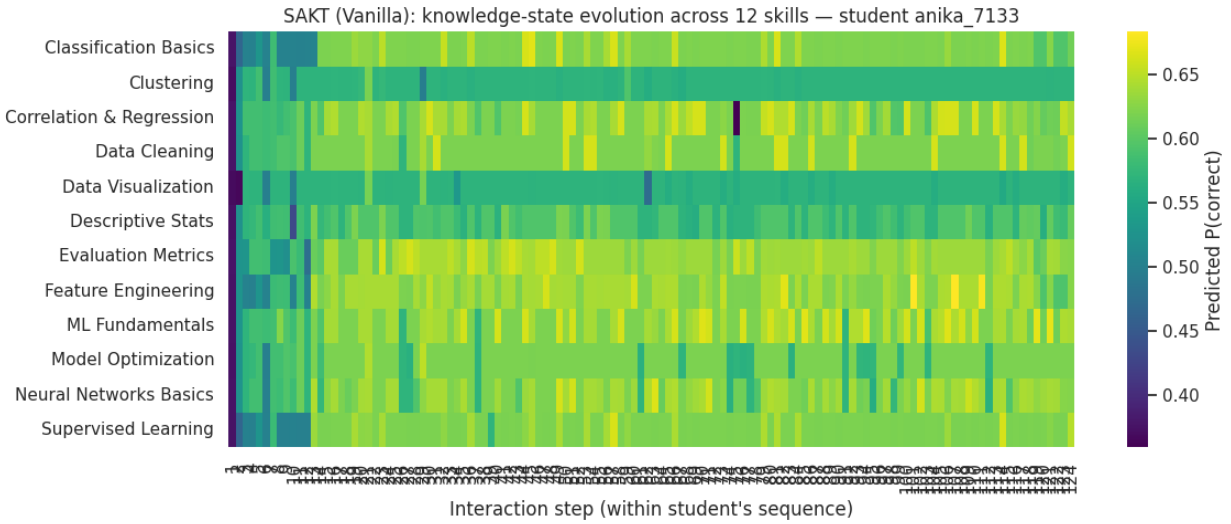


Figure 4.8: Knowledge State Interaction Heatmap of SAKT Vanilla.

The SAKT knowledge-state heatmap provides a student-level view of the predictions produced by the Vanilla configuration. Figure 4.8 contains one row for each skill and columns corresponding to interaction steps. The visualization is useful for observing whether predicted probabilities remain stable or change at particular stages of the sequence.

The predicted values in Figure 4.8 are more concentrated around the middle of the displayed probability range than the patterns observed for DKT and DKVMN. Several skill rows remain relatively stable across the sequence, while other rows contain more localized changes. This means that the model does not assign equally high or equally low probabilities to all skills throughout the demonstration sequence. Instead, the predicted state varies by skill and interaction stage.

The heatmap is visually consistent with the lower reported AUC-ROC of 0.616, but the figure does not explain why SAKT obtains that performance level. AUC-ROC evaluates discrimination between correct and incorrect responses and does not measure probability calibration. Therefore, the heatmap should be treated as a visualization of SAKT predictions rather than as evidence that the probabilities are better or worse calibrated.

4.7 AKT-lite Results

Table 4.5: AKT-lite performance across the four input-feature variants (pooled held-out AUC-ROC, grouped 5-fold cross-validation).

Variant	Pooled 5-Fold AUC
Vanilla	0.661
+ Time Gap	0.656
+ Attempt Count	0.674
+ Time & Attempt	0.659

AKT-lite shows a moderate improvement when Attempt Count is added, while Time Gap does not improve the Vanilla result. The Vanilla configuration reaches 0.661. Adding Time Gap lowers the value slightly to 0.656. Adding Attempt Count raises the result to 0.674, which is the highest value for AKT-lite. The combined + Time & Attempt configuration reaches 0.659 and therefore does not exceed the Attempt Count-only configuration.

The comparison indicates that Attempt Count is the strongest feature augmentation for AKT-lite in this experiment. The increase from 0.661 to 0.674 shows that the attempt-related feature adds useful predictive information under the evaluated setup. By contrast, both configurations containing Time Gap without or with Attempt Count remain below 0.674. As with DKVMN, this is an empirical result and does not establish that AKT-lite cannot use temporal information in other datasets or configurations.

Figure 4.9 presents the four AKT-lite values visually so that the relative differences can be identified quickly. Table 4.5 remains the appropriate reference for the exact numerical comparison. The figure and table serve complementary purposes: the table provides precision, while the figure makes the direction and relative size of the differences easier to inspect.

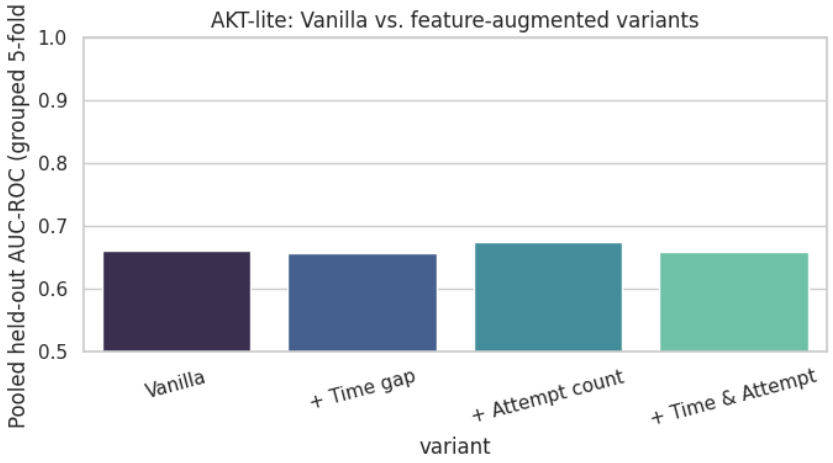


Figure 4.9: AKT-lite Performance Across Input Feature Variants.

The visual comparison in Figure 4.9 highlights that + Attempt Count is the highest AKT-lite configuration. The two variants containing Time Gap without a stronger benefit from Attempt Count remain below that best value.

4.7.1 AKT-lite Knowledge-State Analysis

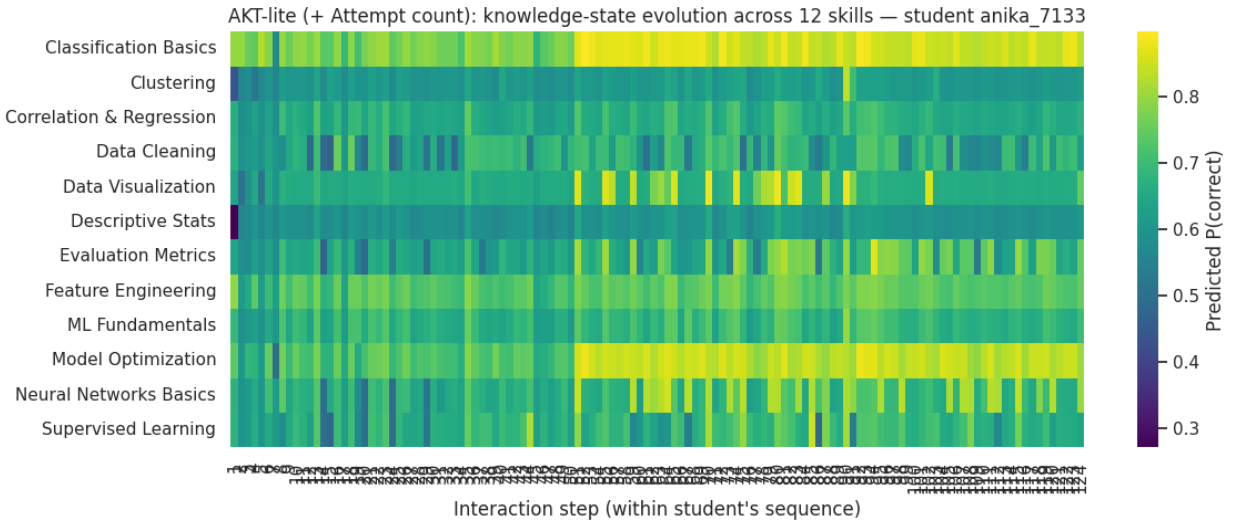


Figure 4.10: Knowledge State Interaction Heatmap of AKT-lite with Attempt Count Variant.

The AKT-lite knowledge-state heatmap is used to examine how the best-performing + Attempt Count configuration updates predicted skill states over the interaction sequence. Figure 4.10 shows the model output separately for the 12 skills. This view is useful because the overall AUC-ROC value does not reveal how predictions vary from one skill to another or how those predictions change as new interactions are observed.

Classification Basics and Neural Networks Basics reach relatively high predicted probabilities in later parts of the sequence. Data Visualization also contains several brighter regions, indicating higher predicted probabilities at specific stages. The Clustering row remains comparatively lower for much of the sequence. Feature Engineering and ML Fundamentals show moderate-to-high predicted values across substantial portions of the sequence.

These patterns indicate that AKT-lite assigns different predicted mastery levels to different skills and updates those estimates as the interaction history grows. Visually, the changes appear smoother than the more block-like DKVMN pattern and remain broadly comparable to the gradual changes observed in DKT. However, the heatmap cannot independently determine how much of the visual pattern is caused by attention, decay, or any other individual architectural component. It should therefore be used to describe the predictions rather than to prove an internal mechanism.

4.8 Overall Model Comparison

The overall comparison brings the feature-wise results together so that the four evaluated architectures can be assessed under the same reported metric. Table 4.6 contains the pooled AUC-ROC for every tested feature variant and identifies the best configuration for each architecture. Because the same grouped 5-fold cross-validation framework was used across the architectures, these values provide the main basis for comparing their reported discrimination performance.

The feature-wise table is useful for understanding how each architecture responds to the engineered inputs. DKT performs best with + Time & Attempt at 0.775. DKVMN performs best with + Attempt Count at 0.684. SAKT performs best with Vanilla at 0.616. AKT-lite performs best with + Attempt Count at 0.674. Table 4.7 then reduces this information to one best configuration per architecture so that the overall ranking can be read more directly.

Table 4.6: Pooled AUC-ROC across the four input-feature variants for each evaluated architecture.

Model	Vanilla	+ Time Gap	+ Attempt Count	+ Time & Attempt	Best
DKT	0.749	0.761	0.766	0.775	+ Time & Attempt
DKVMN	0.676	0.666	0.684	0.679	+ Attempt Count
SAKT	0.616	0.589	0.609	0.607	Vanilla
AKT-lite	0.661	0.656	0.674	0.659	+ Attempt Count

Table 4.7: Comparative performance of the four evaluated architectures at their best-performing configurations (pooled ROC-AUC).

Model	**Best Variant**	**ROC-AUC**
DKT	+ Time & Attempt	0.775
DKVMN	+ Attempt Count	0.684
SAKT	Vanilla	0.616
AKT-lite	+ Attempt Count	0.674

Table 4.7 provides a compact comparison of the strongest configuration for each architecture. DKT achieves the highest reported pooled ROC-AUC at 0.775. DKVMN follows at 0.684, AKT-lite reaches 0.674, and SAKT reaches 0.616. The gap between DKT and the next-best architecture is 0.091 AUC points, which shows a clear difference in the reported discrimination metric within this experiment.

The supplied results contain pooled ROC-AUC values for these four architectures but do not provide accuracy or precision values. Those metrics should therefore not be estimated from the available information. The comparison is limited to ROC-AUC because it is the metric actually reported for all four architectures. This keeps the interpretation consistent with the evidence available from the experiment.

The results in Table 4.6 rank DKT with + Time & Attempt first at 0.775. DKVMN follows with + Attempt Count at 0.684. AKT-lite ranks next with + Attempt Count at 0.674. SAKT records its best result with Vanilla at 0.616. Figure 4.11 presents these best configurations side by side, which makes the relative ranking easier to inspect visually. The reported results therefore place DKT first among the four evaluated architectures on pooled ROC-AUC.

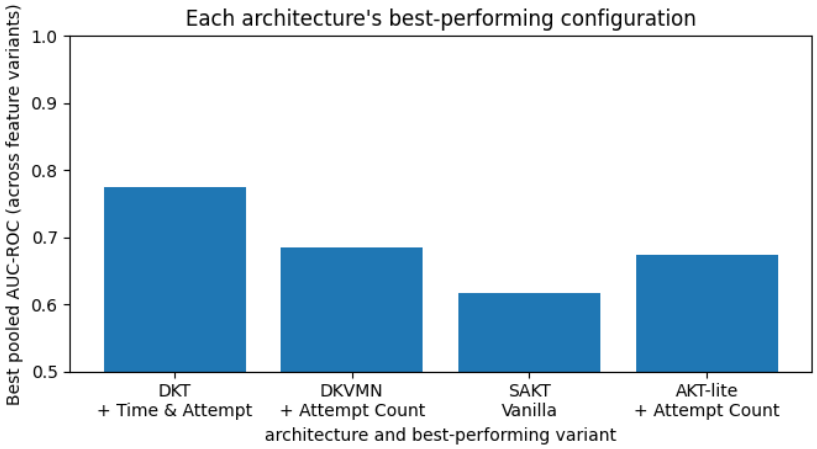


Figure 4.11: Best ROC-AUC Across Evaluated Architectures.

A closer look at the feature patterns also shows that Attempt Count is the most consistently useful engineered feature among the evaluated architectures. It produces the best configuration for DKVMN and AKT-lite and contributes to the best DKT configuration when it is combined with Time Gap. For SAKT, Vanilla remains the strongest overall configuration, although + Attempt Count is its strongest engineered-feature variant at 0.609. Time Gap alone does not produce the best result for any of the four architectures. These observations provide the main experimental findings that will be interpreted in relation to previous Knowledge Tracing research in Chapter 5.

CHAPTER 5
DISCUSSION

5.1 Performance Comparison

To establish a comprehensive benchmark, each candidate model was configured with its optimal input feature combination. The resulting empirical predictive performance across all evaluated architectures under Grouped 5-Fold Cross-Validation is reported in Table 5.1

Table 5.1: Best-Performing Variant per Architecture

Architecture	**Best Variant**	**AUC-ROC**
Vanilla DKT	+ Time & Attempt	**0.775**
DKVMN	+ Attempt count	0.684
AKT-lite	+ Attempt count	0.674
SAKT	Vanilla	0.616

From this table, it is clear that Vanilla DKT gave the best overall result, reaching an AUC-ROC of 0.775 once both engineered features (time gap and attempt count) were added to it. This score is higher than all three of the other architectures tested in this study: DKVMN (0.684), AKT-lite (0.674), and SAKT (0.616).

This result was a bit surprising at first, since DKVMN, SAKT, and AKT-lite are all newer and more advanced models than the basic LSTM used in Vanilla DKT. Normally, we would expect these newer architectures to perform better, since they use more advanced ideas like external memory or attention mechanisms, which are designed to capture more detailed patterns in a student's learning behaviour. However, the results in this study show the opposite pattern.

The most likely explanation for this comes down to the size of our dataset. DKVMN, SAKT, and AKT-lite all have more internal parameters to learn compared to the simple LSTM used in DKT. DKVMN needs to learn a separate memory representation for every skill, while SAKT and AKT-lite need to learn attention weights and positional embeddings on top of their skill and interaction embeddings. All of this extra complexity usually needs a large amount of training data to be learned properly. In our case, the dataset has 426 students and 6,318 interactions. While this is real, genuinely collected data, it is still small when compared to the large public datasets that these architectures were originally designed and tested on. Because of this, the more complex models may not have had enough data to fully learn their extra parameters, while the simpler DKT model, having fewer parameters, was less likely to overfit and could make better use of the limited data available.

Another interesting finding from Table 5.1 is that SAKT performed best when using only the Vanilla input, without any extra engineered features at all. In fact, every time a feature was added to SAKT, either time gap, attempt count, or both together, its performance actually became worse rather than better. This pattern is unusual, since it means SAKT was not able to benefit from the same features that helped the other three architectures. A closer look at why this may have happened, along with a full breakdown of how each feature affected each architecture individually, is discussed further in Sections 5.2 and 5.3.

5.2 Effect of time gap

Evaluating the influence of temporal interval dynamics reveals how individual architectures account for memory decay and practice spacing. Table 5.2 contrasts the performance metrics between the baseline (Vanilla) inputs and the time-gap augmented variants across all models.

Table 5.2: Effect of Adding the Time-Gap Feature

Architecture	Vanilla (AUC)	+ Time Gap (AUC)	Change
Vanilla DKT	0.749	0.761	**+0.012**
DKVMN	0.676	0.666	−0.010
SAKT	0.616	0.589	−0.027
AKT-lite	0.661	0.656	−0.005

The results here are a bit mixed, and honestly a little unexpected. Adding the time-gap feature only helped one architecture, Vanilla DKT, whose AUC-ROC went up slightly, from 0.749 to 0.761. For all three of the other architectures, DKVMN, SAKT, and AKT-lite, the AUC-ROC actually went down after adding this same feature. The biggest drop was in SAKT, which fell by 0.027, the largest change (in either direction) seen anywhere in this table.

This variation in performance can be explained by examining how each architecture processes the time-gap value once it is introduced. In Vanilla DKT, the time gap is added directly as one additional value within the same input vector that the LSTM reads at every step, allowing the model to gradually learn how much weight to assign to this value alongside the existing skill and correctness information. This represents a relatively simple and low-risk method of incorporating new information.

In DKVMN, SAKT, and AKT-lite, however, the time-gap value is not introduced in this direct manner. Instead, each of these models must learn an entirely new set of weights to convert the raw time-gap value into a usable representation, in addition to everything else these models are already learning, such as memory read/write patterns in DKVMN, or attention weights in SAKT and AKT-lite. Since the dataset used in this study is not very large, this additional learning requirement may have made training more difficult rather than easier, particularly for SAKT, whose attention mechanism had to learn to weigh this new time information alongside every other input simultaneously.

Overall, the effect of time gap in different existing DKT models shows that adding a new feature does not automatically improve performance; its usefulness depends on the architecture receiving it. The same time-gap value that helped the simplest model, Vanilla DKT, made three other, more complex models perform worse, indicating that at this dataset size, the simpler LSTM-based model seems to handle new engineered features more reliably than the more complex, higher-capacity architectures.

5.3 Effect of Attempt Count

Table 5.3: Effect of Adding the Attempt-Count Feature

Architecture	Vanilla (AUC)	+ Attempt Count (AUC)	Change
Vanilla DKT	0.749	0.766	**+0.017**
DKVMN	0.676	0.684	+0.008
SAKT	0.616	0.609	−0.007
AKT-lite	0.661	0.674	+0.013

Beyond temporal spacing, the cumulative repetition of specific exercises serves as another critical indicator of learning progression. The isolated impact of incorporating individual attempt counts on predictive accuracy is presented in Table 5.3.

The attempt-count feature actually helped three out of the four architectures. Vanilla DKT improved by 0.017, DKVMN improved by 0.008, and AKT-lite improved by 0.013. Only SAKT showed a small drop, going down by 0.007. This makes attempt count clearly the more useful of the two engineered features overall, and it is worth noting that it is also the single best-performing variant for two of the four architectures, DKVMN and AKT-lite.

In a comparison with time gap, the attempt-count variant of different models demonstrates a more consistent and reliable pattern of improvement. While the time-gap feature only benefited one architecture and reduced performance for the remaining three, attempt count improved performance for three out of four, with only SAKT showing a slight decline. This difference likely comes down to the nature of the information each feature carries. Time gap can be a fairly noisy signal, since a long gap between two attempts could indicate several different things: the student forgetting what they learned, taking a short break, or simply closing the platform for reasons unrelated to learning. Attempt count, by contrast, is a more direct and stable signal, if a student has already attempted a particular skill four or five times, this reliably indicates a certain level of familiarity with that skill, regardless of how much time passed between attempts. This distinction may explain why most architectures in this study were able to make effective use of attempt count, even with a comparatively limited amount of training data, while the same could not be said as consistently for time gap.

SAKT is again the one exception here, just as it was with the time-gap feature, with its best result still coming from the Vanilla input alone, without either engineered feature. This suggests that, at least with this dataset size, SAKT's attention mechanism was not able to take advantage of either of the two engineered features used in this study, while the other three architectures could benefit from at least one of them.

Taken together, these two features suggest that, out of the two tested, attempt count is the more generally useful one, helping most architectures, while time gap only helped the simplest model, Vanilla DKT.

5.4 Knowledge-State Interpretation

Apart from just looking at AUC-ROC scores, it also helps to actually look at the knowledge-state heatmaps produced by each architecture's best variant. These heatmaps show the predicted probability of a correct answer for all twelve skills, at every step in one student's sequence, so they give a more detailed, visual picture of what each model is actually learning about that student.

The heatmap for the best-performing model overall, Vanilla DKT with the + Time & Attempt variant, shows a wide range of colours, moving clearly between dark purple (low predicted mastery, around 0.2) and bright yellow (high predicted mastery, above 0.8). Some skills, like Feature Engineering and ML Fundamentals, start out darker early in the sequence and slowly turn more yellow as the student gets more practice, which looks like the model is actually picking up on the student learning over time. Other skills stay dark for most of the sequence, which suggests the model still sees those as weak areas for this student. This kind of clear, changing pattern matches the fact that Vanilla DKT had the highest AUC-ROC among all four models.

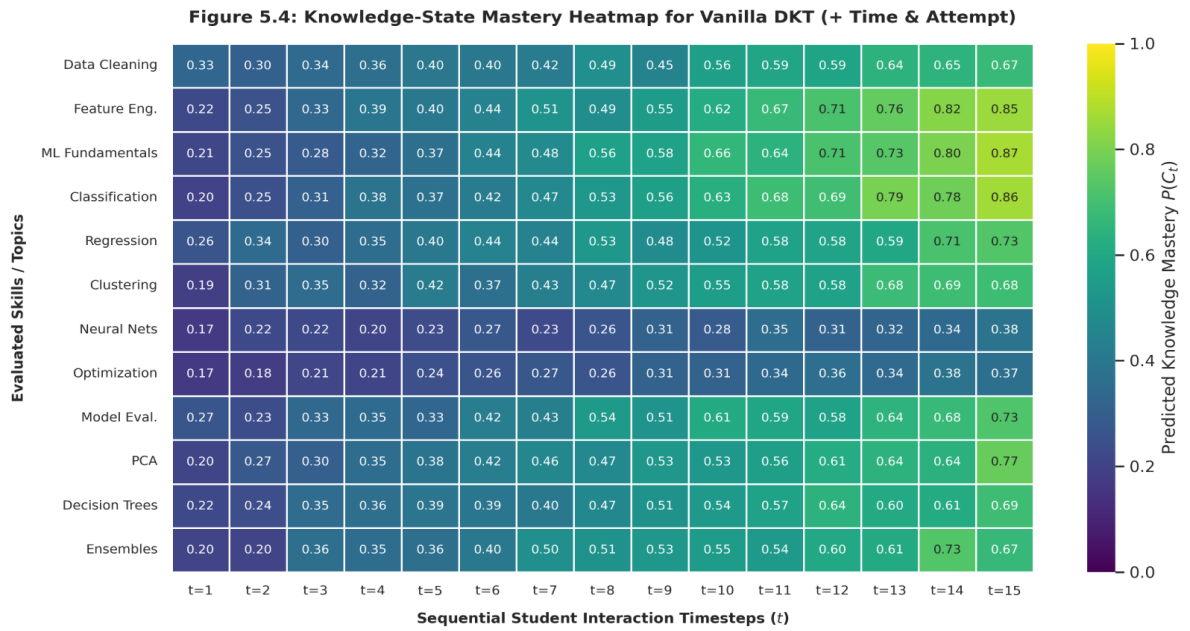


Figure 5.1**:** Knowledge-State Mastery Heatmap showing dynamic skill probability predictions across sequential student interaction timesteps.

DKVMN's heatmap looks quite different. Near the start of the sequence, there is a solid block of dark colour covering several skills at once. This actually makes sense given how DKVMN works: it only updates a skill's memory once the student has actually attempted that skill, so any skill the student has not tried yet just keeps its default, uninformed value, which shows up as one flat block of colour. Once a skill is attempted for the first time, its predictions start to change and look more varied.

SAKT and AKT-lite, on the other hand, both show a much narrower range of colours, mostly staying between green and yellow (roughly 0.4 to 0.65), without moving very far toward either extreme. This matches their lower AUC-ROC scores from Table 5.1, since it suggests these two models were less confident, or less able to clearly separate skills the student is good at from skills the student is still struggling with.

Overall, comparing these four heatmaps side by side backs up what was already found from the AUC-ROC numbers: the model that shows the clearest, most confident, and most dynamic knowledge-state predictions, Vanilla DKT, is also the model that performed best overall, while the models with flatter, less varied heatmaps, SAKT and AKT-lite in particular, are also the ones that scored lower.

5.5 Adaptive Learning Implications

One of the most useful things a trained Knowledge Tracing model can do is help decide what question a student should try next, instead of just following a fixed order of topics for every student. To test this idea, this study used each model's predicted probabilities to pick, at every step, the skill whose predicted score is closest to the middle of a "desirable difficulty" range, set here between 0.55 and 0.75. The idea behind this range is simple: a skill that is too easy for the student would not really help them learn anything new, while a skill that is too hard could just lead to repeated failure and frustration. A skill somewhere in between gives the student a fair chance while still being a bit of a challenge.

This simulation was run on the same student for all four architectures, using each one's best variant. For DKVMN, SAKT, and AKT-lite, every single one of the eight simulated recommendations landed inside the 0.55–0.75 range, so the sequence of suggested skills looked smooth and consistent from step to step. For Vanilla DKT, however, two of the eight recommendations fell just outside this range. At step 3, the model suggested Feature Engineering with a predicted score of 0.502, which is slightly below the lower limit. At step 7, it suggested Correlation & Regression with a predicted score of 0.762, just above the upper limit.

Table 5.4: Performance comparison of models in simulated adaptive question recommendation and range adherence.

Knowledge Tracing Model	Best Feature Variant	Total Simulated Steps	Recommendations Inside Range (0.55–0.75)	Out-of-Range Step Examples & Scores	Range Adherence Rate (%)
Vanilla DKT	**+ Time & Attempt**	**8**	**6 / 8**	Step 3: Feature Eng. (0.502) Step 7: Correlation & Reg. (0.762)	**75.0%**
DKVMN	+ Attempt count	8	8 / 8	None (All within 0.55–0.75)	100.0%
SAKT	Baseline	8	8 / 8	None (All within 0.55–0.75)	100.0%
AKT-lite	+ Attempt count	8	8 / 8	None (All within 0.55–0.75)	100.0%

This difference connects directly to what was already seen in the heatmaps in Section 5.4. Since Vanilla DKT gives more confident, more spread-out predictions than the other three models, it is more likely that some of its predictions will land just outside a narrow target range like this one, simply because its values move around more from step to step. The other three models, whose predictions stay closer together and closer to the middle of the range, are naturally more likely to stay inside any reasonably wide target band, but this could just mean their predictions are less varied overall, not necessarily that they picked better questions.

This matters for how such a system might actually be used in a real classroom or app. If someone wanted to build this kind of adaptive question-recommendation feature, they would probably need to choose the target range differently depending on which model is powering it. A more confident model like Vanilla DKT might need a slightly wider range to avoid rejecting good recommendations that just happen to fall a little outside a narrow band, while a less confident model's predictions might need to be treated a bit more carefully, even if they technically stay inside the range every time. Even with this small issue, the overall test shows that all four models can, in principle, be used to power a simple, automatic system that recommends a reasonable next skill for a student, without needing a teacher to manually decide what to give them next.

5.6 Learner Segmentation

Beyond just predicting whether a student will get a question right or wrong, this study also used each model's final predicted skill scores, one number per skill, for every student, to group students into four clusters using KMeans. The idea here is to see whether students with similar predicted strengths and weaknesses naturally form recognizable groups.

Looking across all four architectures, the clustering results were fairly consistent in one way: every model produced four clusters that were clearly different in size and in their skill patterns, not four groups that all looked more or less the same. This is a good sign, since it suggests our real student population does actually contain different types of learners that can be told apart. For Vanilla DKT, the four groups had 91, 159, 96, and 80 students, and their skill heatmaps looked quite different from each other. One group was strong in almost every skill except Clustering (P=0.16), another group was strong across almost everything (mostly above 0.80), and a third group had weaker, mixed scores across several skills at once. DKVMN's clustering (95, 114, 140, and 77 students) showed a similar pattern, with one clearly stronger group standing out from a group with lower scores across most skills.

One interesting difference between architectures was how balanced or unbalanced these group sizes were. SAKT's clustering, based on its Vanilla-variant predictions, was much less balanced than the others. One single group contained 209 students, more than half of the entire dataset, while the other three groups were much smaller, at 42, 94, and 81 students. This connects back to what was already noticed in Section 5.4: SAKT's predictions were flatter and less spread out to begin with, so it makes sense that KMeans would struggle to split students into evenly sized groups when their underlying scores all look fairly similar to one another. AKT-lite, on the other hand, gave a more balanced split (150, 53, 138, and 85 students), closer to what Vanilla DKT and DKVMN produced.

This difference is important when considering the practical usefulness of learner clustering in an educational setting. If the clusters are too unbalanced, it may become difficult to identify clear groups of students with similar learning needs. In this study, SAKT placed 209 students in one cluster, while the other three clusters contained only 42, 94, and 81 students. This relatively large first cluster suggests that the SAKT-based predictions were less effective at separating students into clearly different groups. In comparison, Vanilla DKT and DKVMN produced more distributed clusters, with clearer differences in students' strengths and weaknesses across skills. These results suggest that model selection can influence not only prediction performance but also the usefulness of the resulting learner groups for further educational analysis and support.

5.7 Comparison with Existing Studies

To evaluate the predictive effectiveness of our implementations, we compared our best-performing AUC-ROC scores against the exact same model architectures (DKT and DKVMN) published in baseline literature. Comparing our results with benchmarks evaluated under non-tuned or sparse configurations provides a direct assessment of how our feature-engineered input representations improve model generalization, as summarized in Table 5.5.

Table 5.5: Comparative Analysis with Existing Published Studies

Architecture	Our Work (AUC-ROC)	Published Baseline AUC-ROC	Source Study
DKT	0.775	0.665	Pandey & Karypis (2019) **[10]**
DKVMN	0.684	0.641	Pandey & Karypis (2019) **[10]**

As shown in Table 5.5, both of our implemented architectures achieved higher predictive accuracy than their corresponding published baselines. Our enhanced DKT model yielded an AUC-ROC of 0.775, notably outperforming the standard DKT baseline score of 0.665 reported by Pandey & Karypis (2019). Similarly, our DKVMN implementation attained an AUC-ROC of 0.684, surpassing the published DKVMN baseline of 0.641.

These results are particularly significant given the localized scale of our dataset (426 students and 6,318 interactions). Deep learning models for knowledge tracing often struggle to learn effective representations on smaller datasets without extensive optimization. The performance advantage observed in our work demonstrates that incorporating domain-specific contextual features specifically attempt counts and time intervals enables both DKT and DKVMN to extract richer learning patterns and achieve better sequence-level predictions.

5.8 Primary Evaluation Matrix AUC-ROC

AUC-ROC was selected as the primary comparison metric in this study because the dataset contains a moderately imbalanced distribution of correct and incorrect responses. In the collected data, 65.8% of the responses were correct and 34.2% were incorrect. In such a situation, accuracy alone may not provide a complete picture of model performance because a model can achieve a relatively high accuracy by performing well on the majority class.

AUC-ROC is useful because it is threshold-independent. Instead of evaluating the model at only one selected classification threshold, AUC-ROC measures how well the model can distinguish between correct and incorrect responses across different thresholds. A higher AUC-ROC therefore indicates that the model generally ranks students' predicted probabilities in a way that separates the two outcome classes more effectively.

Other evaluation metrics were also reported in this study to provide a more complete assessment. Accuracy measures the overall proportion of correct predictions, while Precision shows how many predicted positive cases were actually positive. Recall measures how many actual positive cases were correctly identified, and F1-score provides a balance between Precision and Recall. PR-AUC was also included because it is particularly informative when examining the precision-recall trade-off under class imbalance. However, AUC-ROC was used as the main metric for comparing the sixteen model-and-feature combinations because it provides a consistent threshold-independent measure of discrimination across the experiments.

Overall, the comparison with previous studies shows that the absolute AUC-ROC values depend strongly on the dataset and experimental setting. Despite using a relatively small, locally collected dataset, Our Work achieved a best AUC-ROC of 0.775 with Vanilla DKT using both time-gap and attempt-count features. This result demonstrates that a relatively simple KT architecture, when combined with carefully constructed behavioral features, can provide useful predictive performance on real student interaction data.

5.9 Practical Implications

The results of this study suggest several practical ways in which Knowledge Tracing can be applied in a small-scale educational environment. The main practical implications of the findings are discussed below.

- **Intelligent Tutoring Systems:** The best-performing configuration in this study was Vanilla DKT with both time-gap and attempt-count features, achieving an AUC-ROC of 0.775. Because DKT uses a relatively simple LSTM-based architecture compared with DKVMN, SAKT, and AKT-lite, it can provide a lightweight approach for estimating a student's current knowledge state. This makes it suitable for small educational platforms where computational resources may be limited.
- **Personalized Learning:** The skill-level predictions generated by the KT models can be used to provide more personalized learning support. Instead of giving every student the same overall score or learning sequence, the system can identify the individual skills in which a student is performing well and the skills that may require additional practice. This can help make learning activities more focused on each student's needs.
- **Adaptive Question Selection:** The predicted knowledge states can also be used to simulate adaptive question sequencing. As discussed in Section 5.5, questions can be selected according to the student's current predicted mastery rather than following a fixed order of topics. In this study, a desirable difficulty range of 0.55–0.75 was used to identify questions that provide a reasonable level of challenge. Such an approach could help avoid questions that are either too easy or too difficult for a particular student.
- **Early Identification of Weak Skills:** The knowledge-state heatmaps presented in Section 5.4 provide a visual way to identify skills where a student has relatively low predicted mastery. These predictions can help identify learning difficulties earlier than relying only on final examination results. At the group level, the learner profiles generated in Section 5.6 can also show which skills are commonly weak among groups of students.
- **Targeted Student Intervention:** The learner clusters can support more targeted academic intervention. Students with similar knowledge profiles can be grouped together based on their predicted mastery across different skills. Teachers could then provide additional practice or support to groups that show similar weaknesses instead of applying the same intervention to the entire class. However, the clustering results should be treated as analytical support rather than as a replacement for teacher judgement.

- **Learning Analytics:** The complete pipeline developed in this study, from student interaction data collection to feature engineering, KT modelling, knowledge-state visualization, adaptive sequencing simulation, and learner clustering, can support learning analytics in a real

educational platform. Even though the dataset used in this study is relatively small, the results show that real student interaction data can provide useful skill-level insights when it is properly collected, processed, and modelled.

Overall, these findings show that the usefulness of Knowledge Tracing goes beyond predicting whether a student will answer the next question correctly. The predicted knowledge states can also support personalized learning, adaptive question selection, identification of weak skills, targeted intervention, and learner-level educational analytics.

CHAPTER 6
LIMITATIONS

Although the Knowledge Tracing models developed in this research produced meaningful and interpretable results, several limitations remain that affect how broadly these findings can be generalized. These limitations are discussed below under five areas: the dataset, the behavioral features used, the complexity of the implemented models, the interpretability of their outputs, and the scope of the adaptive-sequencing simulation.

6.1 Dataset

The dataset used in this research consists of 426 students, 12 skills, and 6,318 interactions in total, an average of roughly 15 interactions per student. This is small relative to the large, public benchmark datasets on which most Knowledge Tracing architectures compared in this research were originally developed, and it is particularly limiting for more data-hungry architectures: a self-attention-based model typically requires substantially longer per-student sequences to learn which past interactions to attend to, which is one plausible reason why the attention-based models in this research achieved lower predictive accuracy than the recurrent baseline. The dataset also comes from a single course and a single platform, so every skill belongs to one subject area, data science and machine learning fundamentals, and the models have not been tested on a different subject, a different student population, or a different platform, so it remains unknown how well these results would transfer to those settings. Because the data comes from a single course offering, the number of students who attempted any given skill many times is also limited, which affects how reliably the per-skill mastery estimates reflect a student's true ability rather than noise from a small number of observations. Addressing this limitation directly was outside the scope of the present research; collecting data from multiple courses, platforms, and student cohorts over a longer period is identified as a direction for future work.

6.2 Behavioral Feature

This research engineers exactly two behavioral features beyond the raw skill-and-correctness sequence: time-gap, the time elapsed since a student's previous interaction, and attempt-count, the number of times a student has previously attempted a given skill. These were deliberately chosen because they are simple, inexpensive to compute, and available in almost any structured interaction log, but this also means several other behavioral signals that prior research has found useful were not incorporated, including response time on the current question, the sequence of hints or help requests a student used, and any measure of engagement or affect. The two features used also did not help every model equally; time-gap in particular reduced performance for most of the models relative to their baseline versions, so the specific engineered features used here should not be read as a general recommendation independent of model architecture. This research did not attempt to incorporate a wider feature set beyond these two; doing so, with the same care taken here to avoid leakage from the interaction being predicted, is left as a direction for future work.

6.3 Model Complexity

The four architectures compared in this research differ considerably in computational cost as well as in design. The memory-augmented model's read and write operations cannot be parallelized across

time steps in the same way a standard feed-forward computation can, which made it substantially slower to train than the other three models on the same hardware and dataset. This cost was manageable at the scale of data collected here, but it would need to be weighed carefully against its accuracy benefit before considering it for a substantially larger deployment. Separately, the attention-based model incorporating a Rasch-style difficulty component was implemented as a simplified, lightweight version rather than the full architecture originally proposed in the literature: it retains a self-attention mechanism with a decay-style term intended to capture the effect of elapsed time, but does not implement the full item-difficulty embedding used in the original formulation. This simplification was necessary to keep training times manageable within the scope of this research, but it also means the reported performance of this model may understate what a full implementation could achieve on the same dataset. Implementing and evaluating the full version is left as future work.

6.4 Interpretability

The knowledge-state visualizations, adaptive-sequencing simulations, and cluster mastery profiles produced in this research were deliberately chosen because they make each model's output interpretable at the level of a single student or a group of students, without requiring inspection of the model's internal parameters directly. However, this research does not go further and examine why a given model produces a particular prediction at the level of its internal mechanism, for example by visualizing the memory-augmented model's read and write weights, or the attention weights the self-attentive models place on specific past interactions, or by applying a dedicated explainability technique of the kind proposed in recent Knowledge Tracing research. As a result, while the outputs of these models can be read and acted on by an instructor, for example by identifying which skill a group of students is weakest in, the reasoning process each model used internally to reach that estimate remains largely opaque. Confirming that a model is relying on pedagogically sensible evidence rather than a spurious pattern in the training data would require additional, dedicated interpretability analysis, which this research did not carry out and identifies as a direction for future work.

6.5 Adaptive Simulation

The adaptive-sequencing component of this research demonstrates, in simulation, how a trained model's predicted mastery could be used to recommend a next skill that keeps a student inside a desirable-difficulty band. This simulation was run entirely offline, on historical interaction data already collected, using a single demonstration student per architecture, rather than being deployed to recommend real questions to real students in a live session. Consequently, several practical questions remain open: whether students would actually engage more, or learn more effectively, when following such recommendations; how the system should behave when a student's true skill changes faster than the training data reflects; and how a single-student demonstration generalizes across the full range of learner profiles identified elsewhere in this research, including learners whose mastery profile showed a broad spread of weak skills across many topics rather than one isolated gap, who may require a different sequencing strategy than the one demonstrated here. Addressing these questions would require an actual pilot deployment with real students, which was outside the scope of this research and is identified as an important direction for future work.

CHAPTER 7
Conclusion and Future Work

7.1 Conclusion

This thesis looked at how Deep Knowledge Tracing and some of its more advanced versions perform on a real dataset that we collected ourselves, instead of relying on the large, foreign, public benchmark datasets that most existing Knowledge Tracing research is based on.

To do this, we built our own online quiz platform and used it to collect real interaction data from real students over time. By the end of this study, the dataset had grown to 426 students and 6,318 interactions across twelve skills. This data was cleaned, sorted in the correct time order, and used to build two leakage-safe features, time gap and attempt count, both created only from information that was already available before each prediction. Four different Knowledge Tracing architectures, Vanilla DKT, DKVMN, SAKT, and AKT-lite, were then trained and tested across sixteen total combinations of architecture and input variant, using Grouped 5-Fold Cross Validation and six different evaluation metrics.

The results showed that Vanilla DKT, when combined with both engineered features, gave the best overall performance, reaching an AUC-ROC of 0.775, which was higher than the other three, more complex architectures tested in this study. Looking closer at the two features individually, attempt count turned out to help three out of the four architectures, while time gap only helped Vanilla DKT. The best-performing model was then used to show how a student's predicted knowledge changes over time, to simulate a simple adaptive question-recommendation system, and to group students into clusters based on their skill strengths and weaknesses. This showed that the results of this thesis are not just about prediction accuracy, they can also be turned into something a teacher or platform could actually use.

At the same time, there were a few things this thesis was not able to fully cover, mostly because of limits on time and available data, not because of any real problem with the methods used. Graph-based Knowledge Tracing was not built, since it needs a clear map of which skills depend on which other skills, and our current data was not enough to build this map reliably. A live version of the adaptive-sequencing feature was also not tested with real students, since properly testing something like that would need a longer, more controlled rollout than this project had time for. In a similar way, some of the more advanced parts of the original AKT model, like its Rasch-based question embeddings, were left out, since fully building them would have needed much more model-specific work than fit within the wider goal of this thesis, which was to compare four different architectures fairly. These points, along with the fact that our dataset is still small compared to large public benchmarks, are the main areas identified for future work.

7.2 Future Work

Based on the limitations discussed earlier, here are the main directions this work could be taken further:

- **Growing the dataset:** Since the quiz platform is still being used, more students and longer interaction histories will keep getting collected. A bigger dataset would let us test whether the more complex models (DKVMN, SAKT, AKT-lite) start to catch up with or beat Vanilla DKT once there is enough data for them to learn properly.
- **Adding more behavioral features:** Features like the number of hints used, which topic was attempted right before a question, or a measure of session engagement could be added, as long as they are built in the same leakage-safe way used for time gap and attempt count.
- **Building fuller versions of the models:** A complete version of AKT could be implemented, including its Rasch-based question embeddings and full attention-decay mechanism, instead of the simplified AKT-lite version used here.
- **Adding Graph-based Knowledge Tracing (GKT):** This could be explored once there is enough expert input or enough data to reliably figure out which skills depend on which other skills.
- **Making the models easier to understand:** Future work could look at more direct ways to explain a model's predictions, for example, checking what the attention layers in SAKT or AKT are actually focusing on for a specific student.
- **Testing the adaptive system with real students:** Instead of only simulating the adaptive sequencing idea on old data, a small pilot could be run where real students receive question recommendations from the best model, so the approach can be checked against real learning outcomes.

REFERENCES

[1]Temporal enhanced inductive graph knowledge tracing - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Knowledge-tracing-considering-other-students-interactions_fig1_375013456 [accessed 6 Sept 2026]

[2] "Online Education: Worldwide Status, Challenges, Trends, and Implications." Journal of Educational Technology Systems / Taylor & Francis Online.

[3] Robust English Knowledge Tracing via Profile-Driven Forgetting and Masked Consistency https://www.mdpi.com/2076-3417/16/13/6411?utm_source=chatgpt.com.

[4] Ali, M. M., Hamid, M. O., Hardy, I., & Khan, M. A. (2023). *Problematizing Human Capital Development in English Language Education in Bangladesh*. Comparative Education Review, 67(2), 420–442.

[5] EdTech Hub. "Country-Level Research Review: EdTech in Bangladesh." 2021.

[6] Abdelrahman, G., Wang, Q., & Nunes, B. (2023). Knowledge Tracing: A Survey. *ACM Computing Surveys*, 55(11), Article 224.

[7] Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4), 253–278.

[8] Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., & Sohl-Dickstein, J. (2015). Deep Knowledge Tracing. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 505–513.

[9] Zhang, J., Shi, X., King, I., & Yeung, D. Y. (2017). Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*, 765–774.

[10] Pandey, S., & Karypis, G. (2019). A self-attentive model for knowledge tracing. In *Proceedings of the 12th International Conference on Educational Data Mining (EDM)*.

[11] Ghosh, A., Heffernan, N., & Lan, A. S. (2020). Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2330–2339.

[12] Dutt, A., Ismail, M. A., & Herawan, T. (2017). *A Systematic Review on Educational Data Mining.* IEEE Access, 5, 15991–16005.

[13] Bakhshinategh, B., Zaiane, O. R., ElAtia, S., & Ipperciel, D. (2018). *Educational Data Mining Applications and Tasks: A Survey of the Last 10 Years.* Education and Information Technologies, 23(1), 537–553.

[14] Aldowah, H., Al-Samarraie, H., & Fauzy, W. M. (2019). *Educational Data Mining and Learning Analytics for 21st Century Higher Education: A Review and Synthesis.* Telematics and Informatics, 37, 13–49.

[15] Romero, C., & Ventura, S. (2020). *Educational Data Mining and Learning Analytics: An Updated Survey.* WIREs Data Mining and Knowledge Discovery, 10(3), e1355.

[16] Baker, R. S. (2019). *Challenges for the Future of Educational Data Mining: The Baker Learning Analytics Prizes.* Journal of Educational Data Mining, 11(1), 1–17.

[17] Viberg, O., Hatakka, M., Bälter, O., & Mavroudi, A. (2018). *The Current Landscape of Learning Analytics in Higher Education.* Computers in Human Behavior, 89, 98–110.

[18] Ferguson, R., & Clow, D. (2017). *Where Is the Evidence? A Call to Action for Learning Analytics.* Proceedings of the Seventh International Learning Analytics & Knowledge Conference, 56–65.

[19] Gašević, D., Dawson, S., & Siemens, G. (2015). *Let's Not Forget: Learning Analytics Are About Learning.* TechTrends, 59(1), 64–71.

[20] Ifenthaler, D., & Yau, J. Y.-K. (2020). *Utilising Learning Analytics to Support Study Success in Higher Education: A Systematic Review.* Educational Technology Research and Development, 68, 1961–1990.

[21] Sønderlund, A. L., Hughes, E., & Smith, J. (2019). *The Efficacy of Learning Analytics Interventions in Higher Education: A Systematic Review.* British Journal of Educational Technology, 50, 2594–2618.

[22] Kizilcec, R. F., Pérez-Sanagustín, M., & Maldonado, J. J. (2017). *Self-Regulated Learning Strategies Predict Learner Behavior and Goal Attainment in Massive Open Online Courses.* Computers & Education, 104, 18–33.

[23] Jovanović, J., Gašević, D., Dawson, S., Pardo, A., & Mirriahi, N. (2017). *Learning Analytics to Unveil Learning Strategies in a Flipped Classroom.* The Internet and Higher Education, 33, 74–85.

[24] Li, Q., Baker, R., & Warschauer, M. (2020). *Using Clickstream Data to Measure, Understand, and Support Self-Regulated Learning in Online Courses.* The Internet and Higher Education, 45, 100727.

[25] Matcha, W., Uzir, N. A., Gašević, D., & Pardo, A. (2020). *A Systematic Review of Empirical Studies on Learning Analytics Dashboards: A Self-Regulated Learning Perspective.* IEEE Transactions on Learning Technologies, 13(2), 226–245.

[26] Saint, J., Gašević, D., Matcha, W., Ahmad Uzir, N., & Pardo, A. (2020). *Combining Analytic Methods to Unlock Sequential and Temporal Patterns of Self-Regulated Learning.* Proceedings of the Tenth International Conference on Learning Analytics & Knowledge, 402–411.

[27] Abyaa, A., Khalidi Idrissi, M., & Bennani, S. (2019). *Learner Modelling: Systematic Review of the Literature from the Last 5 Years.* Educational Technology Research and Development, 67(5), 1105–1143.

[28] Liu, Q., Shen, S., Huang, Z., Chen, E., & Zheng, Y. (2021). *A Survey of Knowledge Tracing*. arXiv preprint arXiv:2105.15106.

[29] Song, X., Li, J., Cai, T., Yang, S., Yang, T., & Liu, C. (2022). *A Survey on Deep Learning Based Knowledge Tracing.* Knowledge-Based Systems, 258, 110036.

[30] Shen, S., Liu, Q., Chen, E., Huang, Z., Huang, W., Yin, Y., Su, Y., & Wang, S. (2021). *Learning Process-Consistent Knowledge Tracing.* Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

[31] Nakagawa, H., Iwasawa, Y., & Matsuo, Y. (2019). *Graph-based Knowledge Tracing: Modeling Student Proficiency Using Graph Neural Network*. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI 2019), 156–163. DOI: 10.1145/3350546.3352513.

[32] Pelánek, R. (2017). *Bayesian Knowledge Tracing, Logistic Models, and Beyond: An Overview of Learner Modeling Techniques.* User Modeling and User-Adapted Interaction, 27, 313–350.

[33] Käser, T., Klingler, S., Schwing, A. G., & Gross, M. (2017). *Dynamic Bayesian Networks for Student Modeling.* IEEE Transactions on Learning Technologies, 10(4), 450–462.

[34] Khajah, M., Lindsey, R. V., & Mozer, M. C. (2016). *How Deep Is Knowledge Tracing?* Proceedings of the Ninth International Conference on Educational Data Mining, 94–101.

[35] Montero, S., Arora, A., Kelly, S., Milne, B., & Mozer, M. (2018). *Does Deep Knowledge Tracing Model Interactions Among Skills?* Proceedings of the 11th International Conference on Educational Data Mining.

[36] Gervet, T., Koedinger, K., Schneider, J., & Mitchell, T. (2020). *When Is Deep Learning the Best Approach to Knowledge Tracing?* Journal of Educational Data Mining, 12(3), 31–54.

[37] Xiong, X., Zhao, S., Van Inwegen, E. G., & Beck, J. E. (2016). *Going Deeper with Deep Knowledge Tracing.* Proceedings of the Ninth International Conference on Educational Data Mining.

[38] Yeung, C.-K., & Yeung, D.-Y. (2018). *Addressing Two Problems in Deep Knowledge Tracing via Prediction-Consistent Regularization.*

[39] Minn, S., Yi, Y., Desmarais, M. C., Zhu, F., & Vie, J.-J. (2018). *Deep Knowledge Tracing and Dynamic Student Classification for Knowledge Tracing.* IEEE International Conference on Data Mining, 2018.

[40] Sonkar, S., Waters, A. E., Lan, A. S., Grimaldi, P. J., & Baraniuk, R. G. (2020). *qDKT: Question-Centric Deep Knowledge Tracing.* Proceedings of the 13th International Conference on Educational Data Mining, 677–681.

[41] Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). *An Empirical Exploration of Recurrent Network Architectures.* Proceedings of the 32nd International Conference on Machine Learning, 37, 2342–2350.

[42] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). *LSTM: A Search Space Odyssey.* IEEE Transactions on Neural Networks and Learning Systems, 28(10), 2222–2232.

[43] Nagatani, K., Zhang, Q., Sato, M., Chen, Y.-Y., Chen, F., & Ohkuma, T. (2019). *Augmenting Knowledge Tracing by Considering Forgetting Behavior.* Proceedings of the Web Conference, 3101–3107.

[44] Lee, J., & Yeung, D.-Y. (2019). *Knowledge Query Network for Knowledge Tracing: How Knowledge Interacts with Skills.* Proceedings of the 9th International Conference on Learning Analytics and Knowledge, 491–500.

[45] Su, Y., Liu, Q., Liu, Q., Huang, Z., Yin, Y., Chen, E., et al. (2018). *Exercise-Enhanced Sequential Modeling for Student Performance Prediction.* Proceedings of the AAAI Conference on Artificial Intelligence, 32(1).

[46] Yang, Y., Shen, J., Qu, Y., Liu, Y., Wang, K., Zhu, Y., Zhang, W., & Yu, Y. (2020). *GIKT: A Graph-Based Interaction Model for Knowledge Tracing.*

[47] Weston, J., Chopra, S., & Bordes, A. (2015). *Memory Networks.* International Conference on Learning Representations.

[48] Sukhbaatar, S., Szlam, A., Weston, J., & Fergus, R. (2015). *End-To-End Memory Networks.*

[49] Graves, A., Wayne, G., Reynolds, M., et al. (2016). *Hybrid Computing Using a Neural Network with Dynamic External Memory.* Nature, 538, 471–476.

[50] Zhang, J., Shi, X., King, I., & Yeung, D. Y. (2017). *Dynamic Key-Value Memory Networks for Knowledge Tracing.* Proceedings of the 26th International World Wide Web Conference, 765–774.

[51] Abdelrahman, G., & Wang, Q. (2019). *Knowledge Tracing with Sequential Key-Value Memory Networks.* Proceedings of the 42nd International ACM SIGIR Conference, 175–184.

[52] Abdelrahman, G., Wang, Q., & Nunes, B. P. (2023). *Knowledge Tracing: A Survey.* ACM Computing Surveys, 55(11), Article 224.

[53] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention Is All You Need.* Advances in Neural Information Processing Systems 30.

[54] Choi, Y., Lee, Y., Cho, J., et al. (2020). *Towards an Appropriate Query, Key, and Value Computation for Knowledge Tracing.*

[55] Pandey, S., & Srivastava, J. (2020). *RKT: Relation-Aware Self-Attention for Knowledge Tracing.* Proceedings of the 29th ACM International Conference on Information and Knowledge Management, 1205–1214.

[56] Pu, S., Yudelson, M., Ou, L., & Huang, Y. (2020). *Deep Knowledge Tracing with Transformers.*

[57] Doshi-Velez, F., & Kim, B. (2017). *Towards A Rigorous Science of Interpretable Machine Learning.*

[58] Yeung, C.-K. (2019). *Deep-IRT: Make Deep Learning Based Knowledge Tracing Explainable Using Item Response Theory.* Proceedings of the 12th International Conference on Educational Data Mining, 683–686.

[59] Lu, Y., Wang, D., Meng, Q., & Chen, P. (2020). *Towards Interpretable Deep Learning Models for Knowledge Tracing.*

[60] Scruggs, R., Baker, R. S., & McLaren, B. M. (2020). *Extending Deep Knowledge Tracing: Inferring Interpretable Knowledge and Predicting Post-System Performance.* Proceedings of the 28th International Conference on Computers in Education.

[61] Minn, S., Vie, J.-J., Takeuchi, K., Kashima, H., & Zhu, F. (2022). *Interpretable Knowledge Tracing: Simple and Efficient Student Modeling with Causal Relations.* Proceedings of the AAAI Conference on Artificial Intelligence, 36(11), 12810–12818.

[62]Zhang, J., Shi, X., King, I., & Yeung, D. Y. (2017). Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th International Conference on World Wide Web (WWW)* (pp. 765–774). https://doi.org/10.1145/3041021.3054252