Article

An Introduction to Bayesian Knowledge Tracing with pyBKT

Okan Bulut [1,*], Jinnie Shin [2], Seyma N. Yildirim-Erbasli [3], Guher Gorgun [4] and Zachary A. Pardos [5]

[1] Centre for Research in Applied Measurement and Evaluation, University of Alberta, Edmonton, AB T6G 2G5, Canada
[2] College of Education, University of Florida, Gainesville, FL 32611, USA; jinnie.shin@coe.ufl.edu
[3] Faculty of Arts, Concordia University of Edmonton, Edmonton, AB T5B 4E4, Canada; yildirim.erbasli@concordia.ab.ca
[4] Measurement, Evaluation, and Data Science, University of Alberta, Edmonton, AB T6G 2G5, Canada; gorgun@ualberta.ca
[5] School of Education, University of California Berkeley, Berkeley, CA 94720, USA; pardos@berkeley.edu
[*] Correspondence: bulut@ualberta.ca

Abstract: This study aims to introduce Bayesian Knowledge Tracing (BKT), a probabilistic model used in educational data mining to estimate learners' knowledge states over time. It also provides a practical guide to estimating BKT models using the pyBKT library available in Python. The first section presents an overview of BKT by explaining its theoretical foundations and advantages in modeling individual learning processes. In the second section, we describe different variants of the standard BKT model based on item response theory (IRT). Next, we demonstrate the estimation of BKT with the pyBKT library in Python, outlining data pre-processing steps, parameter estimation, and model evaluation. Different cases of knowledge tracing tasks illustrate how BKT estimates learners' knowledge states and evaluates prediction accuracy. The results highlight the utility of BKT in capturing learners' knowledge states dynamically. We also show that the model parameters of BKT resemble the parameters from logistic IRT models.

Keywords: bayesian knowledge tracing; learner modeling; learning state; item response theory

Citation: Bulut, O.; Shin, J.; Yildirim-Erbasli, S.N.; Gorgun, G.; Pardos, Z.A. An Introduction to Bayesian Knowledge Tracing with pyBKT. *Psych* **2023**, *5*, 770–786. https://doi.org/10.3390/psych5030050

Academic Editor: Alexander Robitzsch

Received: 11 June 2023
Revised: 17 July 2023
Accepted: 20 July 2023
Published: 23 July 2023

1. Introduction

Learner modeling (also referred to as student modeling) refers to a collection of mathematical models utilized to estimate a learner's understanding of the subject matter being taught or their level of competency in other constructs such as effect, motivation, and meta-cognition [1,2]. A learner model can lay the foundation for adaptive learning systems such as intelligent tutoring systems (ITS) [3,4]. These models typically rely on data gathered through the learner's explicit or implicit interactions with the system to estimate the learner's current state of knowledge or trace their learning progress across multiple tasks. In addition, contextual information, such as the type of knowledge or skill being modeled and the purpose of the model, is considered to determine which modeling approach can be more suitable (for a more detailed discussion of learner models, see Harrison and Roberts [5] and Pelánek [1]).

There are multiple techniques available for modeling learners' understanding and learning progress. Among the widely recognized learner models, knowledge tracing particularly stands out based on its ability to trace learners' knowledge and predict their knowledge states [6–8]. Within adaptive learning systems, it is essential to construct a knowledge-tracing model for all learners interacting with the system, representing their proficiency levels in all skills taught in the course [9,10]. For example, an online learning platform offering training in various language skills (e.g., vocabulary, grammar, and conversation) can use a knowledge tracing model to track each learner's progress and proficiency. Suppose a learner excels in vocabulary but needs help with grammar. In that case, the system can identify their areas of weakness and provide targeted exercises, tutorials, or practice materials specifically focused on improving their grammar skills. This personalized approach allows the learner to receive tailored support and effectively address their individual learning needs, ultimately enhancing their language proficiency.

Although knowledge tracing was not initially introduced within the Bayesian framework [11], subsequent research showed that the computational requirements of knowledge tracing could be accurately satisfied by a Dynamic Bayesian Network, which has since become the standard representation of knowledge tracing known as Bayesian Knowledge Tracing (BKT) [12,13]. As a special case of a hidden Markov model with observable nodes, BKT is used to analyze learners' knowledge or skill progression over time and predict their future responses to items [9]. The main objective of BKT is to determine whether a learner has successfully gained proficiency (i.e., answered items or tasks correctly) in a set of skills represented as binary variables (one per skill). BKT has emerged as a successful model in this domain, yielding significant advancements [13,14].

In the remainder of this paper, we first briefly discuss BKT as a learning modeling approach and describe the variants of the standard BKT model. Next, we demonstrate the estimation of BKT models using the pyBKT library in Python. We use the Cognitive Tutor dataset [15] to illustrate different functions in the pyBKT library. With annotated code snippets, we demonstrate how BKT estimates learners' knowledge states and evaluates prediction accuracy. Additionally, we compare BKT with item response theory (IRT) in modeling response accuracy (i.e., answer correctness).

2. Bayesian Knowledge Tracing

BKT is a probabilistic model widely used in the field of educational data mining and learning analytics [16–18]. BKT offers a dynamic process of modeling the probability that a learner possesses knowledge of each skill throughout their learning journey [9,19]. As an HMM with observable nodes, the goal of BKT is to estimate the knowledge states of individual learners over time based on their interactions with educational tasks or assessments. In BKT, the learner's knowledge is defined as a latent variable while the learner's responses to items (i.e., their performance) are treated as observable variables [20]. The knowledge is represented as a binary variable, indicating whether the learner has learned the skill. The items are also scored dichotomously, with answers categorized as either correct or incorrect.

At its core, BKT leverages Bayesian inference to update and refine its estimates of a learner's knowledge state at a particular time step t as the learner responds to questions or completes learning activities (obs_t). The model includes two knowledge parameters for each learner: the probability of knowing a concept before encountering a task (i.e., prior knowledge; $P(L_0)$), and the probability of learning or acquiring knowledge from the task at time step t (i.e., learned knowledge; $P(L_t)$). The probability of transitioning from the not-known state to the known state after each answer is denoted as $P(T)$). In addition to knowledge parameters, the model also involves two performance parameters: the probability of making a mistake when applying a known skill (slip; $P(S)$) and the probability of answering an item correctly with a not-known skill (guess; $P(G)$) [21].

These four parameters in BKT are utilized to update the probability of learning, representing the likelihood of the learner's knowledge of a specific skill. More specifically, as the learner responds to the items, BKT updates $P(L_t)$ based on the accuracy of their response (correct or incorrect):

$$P(L_t|obs_t = 1) = \frac{P(L_t)(1 - P(S))}{P(L_t)(1 - P(S)) + (1 - P(L_t))P(G)} \tag{1}$$

$$P(L_t|obs_t = 0) = \frac{P(L_t)P(S)}{P(L_t)P(S) + (1 - P(L_t))(1 - P(G))} \tag{2}$$

As the learner transitions from one step (t) to the next ($t+1$), the updated prior for the following time step can be calculated by [22]:

$$P(L_{t+1}) = P(L_t|obs_t) + (1 - P(L_t|obs_t))P(T), \tag{3}$$

which suggests that the learner is likely to transition from a not-known state to a known state by learning from immediate feedback and any other instructional support; however, the probability of forgetting ($P(F)$) remains zero [22]. Figure 1 illustrates the workflow of the standard BKT model.

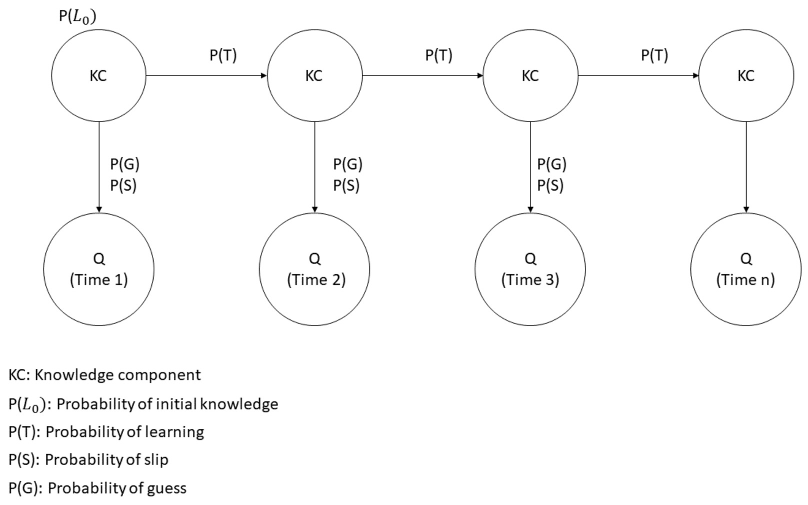

P(L_0)
KC
P(T)
KC
P(T)
KC
P(T)
KC
P(G)
P(S)
P(G)
P(S)
P(G)
P(S)
Q (Time 1)
Q (Time 2)
Q (Time 3)
Q (Time n)
KC: Knowledge component
P(L_0): Probability of initial knowledge
P(T): Probability of learning
P(S): Probability of slip
P(G): Probability of guess

Figure 1. The workflow of BKT.

BKT analyzes learners' responses to each item to track their knowledge states, updating the probabilities of prior and learned knowledge in an iterative process [23]. This enables BKT to improve its estimates as it gathers more data. The knowledge states estimated by BKT can predict a learner's future performance, identify areas of weakness or misconception, and provide interventions and support. By tracking a learner's latent knowledge continuously, BKT enables personalized adaptation of practice and learning experiences [24]. This dynamic assessment of knowledge can optimize the allocation of educational resources, ensuring learners can navigate challenging materials and receive support when necessary [25].

3. Variants of Bayesian Knowledge Tracing

To date, researchers have proposed different variants of the standard BKT model, including an item difficulty-based model [26], which addresses item performance variability within a skill by fitting distinct guess and slip values for each item, a learner-level individualization model [27], which shows the learned knowledge provides better predictive performance compared to prior knowledge, and a hybrid model [28], which incorporates regression to determine if a learner's response was a guess or a slip, taking contextual information into consideration. Furthermore, researchers combined BKT with other modeling frameworks to enhance its accuracy and interoperability (e.g., [29]). In the following sections, we describe two extensions of the standard BKT model worth mentioning.

3.1. IRT-BKT Model

In the field of psychometrics, researchers have extensively studied various test theories to model students' knowledge states within a test session. Test theories differ from KT in that they are primarily designed for tests where students' knowledge states are assumed to be static (i.e., students' knowledge remains unchanged as they respond to a set of questions) [30]. As a modern test theory, IRT stands out as another notable approach for modeling students' (current) knowledge states. IRT is a modeling framework used for constructing and analyzing learning and assessment data [31]. It provides a systematic approach for modeling the relationship between the characteristics of the items (e.g., difficulty, discrimination, and guessing) and the probability of learners responding to those items correctly. Within the IRT context, the probability of person j answering item i correctly, $P(Y_{ij})$, can be expressed as follows:

$$P(Y_{ij} = 1 | \theta_j, a_i, b_i, c_i) = c_i + (1 - c_i)\frac{e^{a_i(\theta_j - b_i)}}{1 + e^{a_i(\theta_j - b_i)}}, \tag{4}$$

where a_i, b_i, and c_i refer to the discrimination, difficulty, and guessing parameters of item i, respectively, and θ_j represents person j's latent trait (or ability) level on the construct being measured by the item. The IRT model shown in Equation (4) is known as the three-parameter logistic (3PL) model as it involves three item parameters (discrimination, difficulty, and guessing). Depending on the psychometric conceptualization of the construct being modeled, the 3PL model can be simplified by eliminating some parameters. For example, the one-parameter logistic (1PL) model only includes the difficulty (b) parameter, assuming no guessing and equal discrimination (e.g., $a = 1$) across all items.

Although IRT and BKT seem to offer distinct approaches to understanding learner knowledge and performance, both approaches aim to extract insights and understand patterns from the learner data obtained in educational settings. IRT is more tailored for cross-sectional data, aiming to analyze item responses to assess learners' latent traits, such as knowledge or ability, while BKT, specifically designed for longitudinal data, focuses on modeling and tracking learners' knowledge states over time [29,32]. Deonovic et al. [29] discussed the connection between BKT models and IRT models and highlighted how the limitations of each model can be mitigated by integrating concepts from the other. The authors also argued that there is a fundamental issue with both BKT and IRT models: their inability to account for the educational context within the modeling framework. BKT can estimate the knowledge state through the learning parameter, while IRT can estimate the level of acquired learning or the learner's proficiency through the ability parameter. However, neither model includes a specific component to represent the teaching or educational factors that can impact learners' interaction with the learning environment. Consequently, these models do not capture how differences in teaching may contribute to variations in learning outcomes within the IRT framework, or how the learning process itself is influenced within the BKT framework. Deonovic et al. [29] suggested the integration of BKT and IRT in order to combine the strengths of learning sciences and psychometrics.

In an effort to enhance estimation accuracy, Wang et al. [33] proposed a novel approach that combines IRT and BKT to model learning outcomes—Knowledge Tracing Model on Item Response Theory (IRT-BKT). By incorporating BKT, which utilizes the sequence of learner responses to trace the knowledge status, they aimed to effectively track and assess the learner's knowledge acquisition. In an earlier model, researchers employed IRT and BKT using Markov Chain Monte Carlo with maximum posterior probability in order to estimate the parameters of IRT and BKT (i.e., theta, discrimination, difficulty, learn, guess, and slip) [34]. In their model, first, IRT estimates the learner's ability for skill. Next, this information is combined with the estimated difficulty and discrimination level of each skill to calculate the probability of a learner already knowing a skill before practicing it. Second, skill-specific knowledge tracing probabilities are fitted to account for the likelihood of learning, guessing, and slipping. Instead of separately estimating item and ability parameters in IRT and learn, guess, and slip parameters in BKT, IRT-BKT combines both frameworks to estimate all parameters simultaneously [33]. Wang et al. [33] compared the original IRT and BKT models [34] and their proposed IRT-BKT model and found that the IRT-BKT model outperformed the original IRT and BKT models in terms of parameter accuracy.

3.2. KT-IDEM Model

Pardos and Heffernan [26] sought to explore the effectiveness of extending knowledge tracing to encompass item difficulty and thus enhance prediction accuracy, called Knowledge Tracing Item Difficulty Effect Mode (KT-IDEM). In their study, they also aimed to address the potential performance limitations arising from the increased complexity of feature generation across different datasets. Thus, they sought to minimize modifications to the knowledge tracing while incorporating item difficulty. By adopting this approach, they could have a balance between enhancing the model's capabilities by including item difficulty and maintaining the simplicity and integrity of the original knowledge tracing framework. In their study, Pardos and Heffernan [26] explored how the addition of question-level difficulty can be incorporated into the guess and slip parameters. The intuitive notion is that skills associated with a high guess rate (guess parameter) can be considered easier. Conversely, skills characterized by a low guess rate and a higher rate of mistakes (slip parameter) can be considered more challenging. Based on this understanding, the study suggested that question difficulty can be effectively captured by manipulating the guess and slip parameters. In their experimental setup, the researchers trained and tested both knowledge tracing and KT-IDEM models using the same cross-validation data from the ASSISTments platform. The evaluation of the models was conducted using the Area Under the Curve (AUC) metric. The results strongly favored KT-IDEM over KT. The superior performance of KT-IDEM, as indicated by the higher AUC scores, highlighted the effectiveness and improved predictive capability of the KT-IDEM model compared to the standard KT model.

4. Estimating BKT Models with pyBKT

4.1. Data

We demonstrate the implementation of pyBKT and its comparison to IRT with the Cognitive Tutor dataset [15] focused on students' math abilities in middle and high school. The dataset includes students' interactions and the correctness of their attempts in a set of items associated with 12 knowledge components in mathematics. The knowledge components are "Calculate unit rate", "Calculate part in proportion with fractions", "Plot whole number", "Plot terminating proper fraction", "Plot imperfect radical", "Plot non-terminating improper fraction", "Plot pi", "Plot decimal-thousandths", "Finding the intersection, SIF", "Finding the intersection, Mixed", and "Finding the intersection, GLF". A group of 587 unique students' interactions with the tutoring system are logged and analyzed for this demonstration. Table 1 shows the number of items and students for each knowledge component included in the Cognitive Tutor data set.

Table 1. Overview of the Cognitive Tutor dataset.

Knowledge Components	Number of Items	Number of Students
Calculate part in proportion with fractions	112	453
Calculate the total in proportion with fractions	88	451
Calculate unit rate	200	470
Finding the intersection, GLF	22	72
Finding the intersection, Mixed	16	74
Finding the intersection, SIF	14	71
Plot decimal - thousandths	14	251
Plot imperfect radical	21	253
Plot non-terminating improper fraction	13	255
Plot pi	10	253
Plot terminating proper fraction	31	256
Plot whole number	9	256

4.2. The pyBKT Library

Badrinath et al. [22] developed pyPKT as an accessible, easy-to-implement, and computationally efficient Python library to facilitate the estimation of various BKT models. The pyBKT library is built on HMM where the model is composed of an observable node of whether a learner has mastered a skill (i.e., represented with the learner's dichotomous response sequences) and four hidden nodes; probability of initial knowledge (i.e., $P(L_0)$), transition probability (i.e., probability of learning; $P(L)$), probability of slip (i.e., $P(S)$), and probability of guessing (i.e., $P(G)$). pyBKT uses the expectation-maximization algorithm to learn hidden nodes from the dichotomous action sequences and estimates the probability of a learner mastering a skill (i.e., knowledge component; KC) at time t.

pyBKT requires a Python version of 3.5 or higher and can be run on all operating systems. The pyBKT library's internal data helper processes the response-per-row data file (.csv or .tab) to create an internal pyBKT data format Badrinath et al. [22]. The estimation error asymptote at 50 students for learn and slip parameters whereas the estimation error asymptote at 100 students when prior and guess parameters are added to the model. The sequence length is also an important factor for model estimation accuracy. A sequence length of at least 15 is needed per skill for model accuracy. That is at least 50 students for learn and slip parameters or 100 students for prior and guess parameters with at least 15 sequences are needed for accurate model estimation (for more information, see Badrinath et al. [22]). The pyBKT library requirements include numpy, sklearn, pandas, scikit-learn, scipy, and joblib libraries.

The pyBKT library involves functions for a wide range of tasks, including data generation, model estimation, prediction, cross-validation, parameter initialization, and evaluation [22]. Furthermore, pyBKT includes many BKT variants such as *Standard Knowledge Tracing*, *Knowledge Tracing and Forgets Models* (i.e., forgets), *Item Difficulty Effect Model* (i.e., multigs), *Prior per Student Model* (i.e., multiprior), *Item Order Effect Model* (i.e., multipair), and *Item Learning Effect Model* (i.e., multilearn). The pyBKT's Github page (https://github.com/CAHLR/pyBKT; accessed on 19 July 2023) includes a quick start tutorial and several code examples for estimating various BKT models. The pyBKT library is compatible with Windows, Mac OS X, and Linux operating systems.

4.3. Case Study 1: Estimating the Standard BKT Model

In the following case study, we demonstrate how pyBKT can be used for estimating the standard BKT model with the Cognitive Tutor dataset. In a standard BKT model, at least four variables should exist in the dataset:

1. Learner ID
2. Question ID
3. Skill name
4. Learners' dichotomous responses to questions

First, we begin with the installation of pyBKT and then import it with other libraries necessary for the implementation of our case studies (see Listing 1).

Listing 1. Installing necessary Python libraries.

```
# Installing necessary Python libraries
pip install pyBKT
from pyBKT.models import Model
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

Next, we read the file for the Cognitive Tutor dataset in Python. The accepted input formats for pyBKT are Pandas data frames and data files of type .csv (comma-separated values) or .tsv (tab-separated values). We can import the dataset as a Pandas data frame and then use it in the pyBKT functions, or we can directly refer to the file location (Listing 2).

Listing 2. Importing the Cognitive Tutor dataset.

```
# The Cognitive Tutor dataset can be downloaded from:
# https://raw.githubusercontent.com/CAHLR/pyBKT-examples/master/data/ct.csv

# Reading the Cognitive Tutor dataset
df = pd.read_csv('ct.csv', encoding='latin')
```

The *Model* function fits the BKT model to all skills available in the Cognitive Tutor dataset. However, we can also train the model for a single skill or a set of skills. In the Cognitive Tutor dataset, there are 12 math skills, indicated as *KC(default* (i.e., knowledge components). Assume that a teacher is interested in understanding students' mastery of generating plots for various purposes, which encompasses knowledge components such as *plot terminating improper fraction*, *plot non-terminating improper fraction*, *plot whole number*, *plot decimal - thousandths*, and *plot pi*. We can use regex to simply match the knowledge components that include the word *plot* for selecting the skills that we want to use for training BKT. We use skills = ".*Plot.*" to select plot-related knowledge components and fit a standard BKT model, without considering other model specifications such as *forgets*, *multigs*, *multiprior*, *multipair*, *multilearn*. In Listing 3, we initialize a BKT model and fit the model for skills including *Plot*.

Listing 3. Training BKT for a specific set of skills.

```
# Initialize the model and set the seed for replicability purposes
model = Model(seed = 42, num_fits = 1)

# Train a simple BKT model on plot-related knowledge components (with file location)
model.fit(data_path = 'ct.csv', skills = ".*Plot.*")
# Or, train the model using the Pandas dataframe
model.fit(data = df, skills = ".*Plot.*")
```

Alternatively, we can train a standard BKT model on a unique math skill available in the Cognitive Tutor dataset using *skills*. For example, specifying skills = "Plot imperfect radical" would train the BKT model on a particular skill (i.e., plotting imperfect radical). In addition, we can train the model on each unique skill in the dataset using a loop, see Listing 4:

Listing 4. Training BKT for each unique skill.

```
skill_list = list(df['KC(Default)'].unique())
for s in skill_list:
  model.fit(data_path = 'ct.csv', skills = s)
```

To model learner data using the pyBKT library, the dataset must consist of column names familiar to pyBKT (e.g., order_id, skill_name, and correct). In the above code snippet, we did not need to change the column names because pyBKT can automatically infer column name mappings in the Cognitive Tutor dataset. However, if the dataset includes unfamiliar column names, then the user may need to specify a mapping from the dataset's column names to pyBKT's expected column names. This process is referred to as the model defaults. Assume that the Cognitive Tutor dataset included three columns: person, skill_math, and answer representing the order ID, skill names, and correctness, respectively. For these columns, we could specify the default column names (see Listing 5) to lookup in the dataset and use them in the model training process as follows:

Listing 5. Mapping the existing column names to expected pyBKT columns.

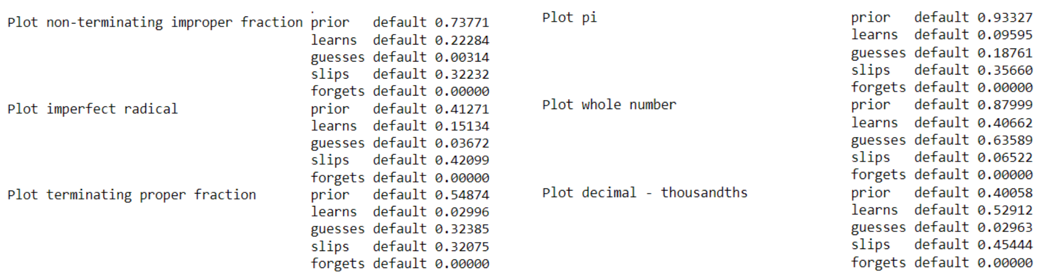
```
# Update the column mappings (i.e., defaults)
defaults = {'order_id': 'person', 'skill_name': 'skill_math', 'correct': 'answer'}

# Fit the BKT model using the updated defaults
model.fit(data = df, defaults = defaults)
```

pyBKT offers a variety of features for prediction and evaluation. We can use the *params* function to obtain fitted parameters, including the probability of prior knowledge, the probability of guessing, the probability of slipping, and the probability of learning the skills. We can extract fitted parameters by using Listing 6. Figure 2 shows the pyBKT output with the estimated parameters for each skill. In the output, "prior" refers to the prior probability of knowing, "learns" is the probability of transitioning to the knowing state given not known, "guesses" is the probability of guessing correctly given the not-knowing state, and "slips" is the probability of picking incorrect answer given the knowing state. Note that all "forgets" values in the output (i.e., the probability of transitioning to the not knowing state given known) are zero because the standard BKT model assumes no forgetting.

Listing 6. Parameters for unknown states.

```python
# Print estimated model parameters
model.params()
```



```
Plot non-terminating improper fraction prior   default 0.73771
                                       learns  default 0.22284
                                       guesses default 0.00314
                                       slips   default 0.32232
                                       forgets default 0.00000
Plot imperfect radical                 prior   default 0.41271
                                       learns  default 0.15134
                                       guesses default 0.03672
                                       slips   default 0.42099
                                       forgets default 0.00000
Plot terminating proper fraction       prior   default 0.54874
                                       learns  default 0.02996
                                       guesses default 0.32385
                                       slips   default 0.32075
                                       forgets default 0.00000
Plot pi                                prior   default 0.93327
                                       learns  default 0.09595
                                       guesses default 0.18761
                                       slips   default 0.35660
                                       forgets default 0.00000
Plot whole number                      prior   default 0.87999
                                       learns  default 0.40662
                                       guesses default 0.63589
                                       slips   default 0.06522
                                       forgets default 0.00000
Plot decimal - thousandths             prior   default 0.40058
                                       learns  default 0.52912
                                       guesses default 0.02963
                                       slips   default 0.45444
                                       forgets default 0.00000
```

Figure 2. Parameters obtained from the standard BKT model

Furthermore, we can evaluate the prediction accuracy of the fitted BKT model using root-mean-square error (RMSE), which is the default evaluation metric. Alternatively, we can use metric=['auc'] to obtain the area under a curve (AUC) metric for the trained model (see Listing 7).

Listing 7. Model evaluation.

```python
model.evaluate(data_path = 'ct.csv', metric=['rmse', 'accuracy', 'auc'])
```

On the pyBKT's GitHub page (https://github.com/CAHLR/pyBKT; accessed on 19 July 2023), the authors of the pyBKT library also demonstrate how to use a custom accuracy function for model evaluation as given in Listing 8:

Listing 8. Custom evaluation metrics.

```python
# Function for Mean Absolute Error (MAE)
def mae(true_vals, pred_vals):
  return np.mean(np.abs(true_vals - pred_vals))

model.evaluate(data_path = 'ct.csv', metric = mae)
```

We found an overall accuracy of 67%, with RMSE, AUC, and MAE values of 0.45, 0.74, and 0.41, respectively for the skills related to generating plots. This, however, represents model evaluation for the whole dataset as we did not split the data into training and test sets. Therefore, we can employ *k*-fold cross-validation as given in Listing 9 to obtain a more reliable evaluation of the performance of the BKT model.

Listing 9. Three-fold cross-validation.

```python
model.crossvalidate(data_path = 'ct.csv', folds = 3)
```

After running a three-fold model cross-validation, we obtain the RMSE values of 0.48, 0.44, 0.50, 0.47, 0.29, and 0.48 for *plot non-terminating improper fraction*, *plot imperfect radical*, *plot terminating improper fraction*, *plot pi*, *plot whole number*, and *plot decimal- thousandths*, respectively (see Figure 3).

We can also use different variants of the standard BKT model (see Listing 10), such as *forgets*, which relaxes the assumption that learners do not forget the content or skill they mastered, *multigs*, which fits different guess and slip parameters for each class, or *multilearn* which fits a different learn rate if *forgets* is used.

In the above code snippet, we assume that students may forget the knowledge component mastered, the BKT model is allowed to have different guess and slip parameters, and the model is estimated with a different learn rate. For this model, we found an RMSE value of 0.39, an accuracy of 78%, and an AUC value of 0.85. These findings suggest that model fit improved when we allowed students to forget, used different guess and slip parameters, and enabled different learn rates.

skill	rmse
Plot non-terminating improper fraction	0.48260
Plot imperfect radical	0.43936
Plot terminating proper fraction	0.49667
Plot pi	0.47264
Plot whole number	0.28965
Plot decimal - thousandths	0.47810

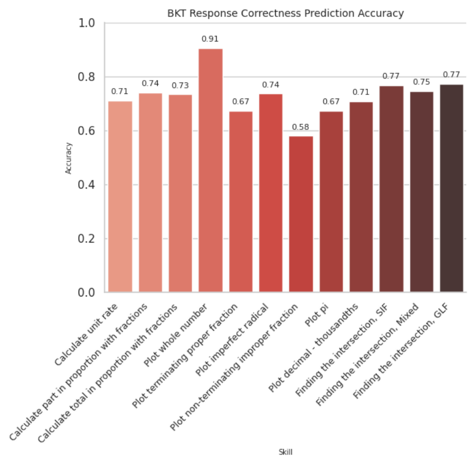
Figure 3. RMSE values estimated for plot generation skills.

Listing 10. Training BKT variants.
```
model.fit(data_path = 'ct.csv', multilearn = True, forgets = True, multigs = True)
```

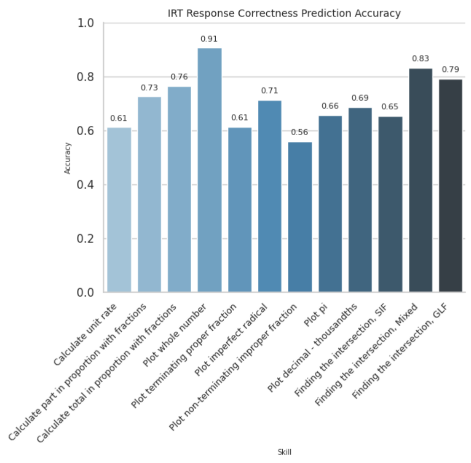
4.4. Case Study 2: Comparing IRT and BKT in Modeling Response Accuracy

In the second case study, we will compare the performance of BKT and IRT models (see Listing 11). A suitable task for this comparison is to predict the response of a learner to a given item. In this task, the models can be evaluated based on their accuracy in predicting whether the learner will answer the items correctly or incorrectly. By comparing the prediction accuracy of IRT and BKT models on the same dataset, we can assess their performance in capturing learner knowledge and predicting their responses (see Listing A1 in Appendix A for the *IRT* function for estimating the 1PL IRT model).

The results indicated that both models exhibited a promising performance in predicting the correctness of learners' responses, with each model estimating the results separately for each knowledge component (see Figure 4). Comparing the average classification performance of the BKT and 1PL IRT models, we found that the BKT model achieved a slightly higher average performance of 0.73, while the 1PL IRT model achieved an average performance of 0.71. Notably, both models demonstrated their strengths and weaknesses across different knowledge components. The "Plot non-terminating improper fraction" category emerged as the worst performing component, while the "Plot whole number" component exhibited the highest accuracy, with both models achieving close to 90% accuracy. These findings emphasize the potential of both the IRT and BKT models in effectively modeling learners' responses and providing valuable insights into knowledge component performance in educational settings.



(**a**) BKT prediction results



(**b**) IRT prediction results

Figure 4. Prediction results by skill.

Listing 11. Knowledge prediction with BKT and IRT.

```
def main(skill='Plot imperfect radical'):
    train_data = df[df['KC(Default)'] == skill]
    train_data = train_data[['Anon Student Id', 'Problem Name', 'Correct First Attempt']]
    train_data.columns = ['user_id', "question_id", "is_correct"]

    labels, levels = pd.factorize(train_data['user_id'])
    train_data['user_id'] = labels
    user_dic = dict(zip(levels, list(range(len(levels)))))
    labels, levels = pd.factorize(train_data['question_id'])
    train_data['question_id'] = labels
    question_dic = dict(zip(levels, list(range(len(levels)))))

    # Use 30% of the full dataset for testing and 30% of test data for validation
    train_data, test_data = train_test_split(train_data, test_size=0.3)
    val_data, test_data = train_test_split(test_data, test_size=0.3)

    train_data = train_data.reset_index()
    train_data = train_data.drop(columns=['index'])
    test_data = test_data.reset_index()
    test_data = test_data.drop(columns=['index'])
    val_data = val_data.reset_index()
    val_data = val_data.drop(columns=['index'])

    lr = 1e-2
    num_iteration = 50
    irt_model = IRT(lr, num_iteration, len(user_dic), len(question_dic))
    theta, beta, val_acc_lst, neg_lld_lst, val_lld_lst = irt_model.irt(train_data, val_data
     )

    print("Validation accuracy: {}".format(val_acc_lst[-1]))
    test_acc = irt_model.evaluate(test_data)
    print("Test accuracy: {}".format(test_acc))
    return test_acc, user_dic, question_dic, theta, beta

skill_list = list(df['KC(Default)'].unique()); skill_acc = []
for s in skill_list:
    # Train the model on the data
    test_acc, user_dic, question_dic, theta, beta = main(s)
    skill_acc.append(test_acc)
```

4.5. Case Study 3: Associations between IRT and BKT Parameters

In our final case study, we compare the parameters of IRT (1PL) and BKT. In IRT, the parameters (difficulty, or b, and ability, or θ) are specific to individual items and learners, respectively, and are estimated separately. In BKT, the parameters (initial knowledge, learning rate, guess probability, and slip probability) are estimated globally for each skill or concept. Furthermore, IRT focuses on modeling individual learner abilities and item difficulties, providing a continuous measure of learner proficiency. BKT, on the other hand, focuses on modeling learning as a dynamic process and can capture changes in knowledge over time. Overall, the IRT and BKT models have distinct parameterizations and capture different aspects of learning. IRT primarily focuses on estimating ability and item difficulty, while BKT models the dynamics of learning and incorporates parameters related to initial knowledge, learning rate, guess probability, and slip probability. The choice between these models depends on the research or educational context and the specific learning aspects of interest.

In the IRT model, item difficulty refers to the level of item challenge or the probability of a correct response from individuals with average ability. In the BKT model, the parameters of guess and slip represent the probability of guessing the correct answer and the probability of slipping, respectively. The difficulty parameter in IRT can be defined based on these BKT parameters, by re-parameterizing the guess parameter [29]. The following code snippet shows how to compare the 1PL IRT and BKT model parameters by converting BKT parameters to IRT parameters and then calculating the correlation between the IRT parameters and the transformed BKT parameters (Listing 12).

Listing 12. Python code example linking BKT and IRT parameters.

```python
correal_result = []
for s in list(set(df['KC(Default)'])):
  #s = "Calculate part in proportion with fractions"
  model = Model(seed = 42, num_fits = 1)
  model.fit(data_path = 'ct.csv', skills = s, multigs = True, forgets=True, multilearn =
     True)
  #print(model.params())
  acc = model.evaluate(data_path = 'ct.csv', metric = 'accuracy')

  learn_params= {}
  for i, j in dict(model.params()['value']).items():
    if i[-2] == 'learns':
      #print(i)
      learn_params[i[-1]] = j
    else:
      None

  forget_params= {}
  for i, j in dict(model.params()['value']).items():
    if i[-2] == 'forgets':
      #print(i)
      forget_params[i[-1]] = j
    else:
      None

  guess_params= {}
  for i, j in dict(model.params()['value']).items():
    if i[-2] == 'guesses':
      #print(i)
      guess_params[i[-1]] = j
    else:
      None

  slip_params= {}
  for i, j in dict(model.params()['value']).items():
    if i[-2] == 'slips':
      #print(i)
      slip_params[i[-1]] = j
    else:
      None

  test_acc, user_dic, question_dic, theta, beta = main(s)

  # Function for converting BKT parameters to IRT parameters
  def bkt_to_irt(q_id):
      # Extract BKT parameters
      p_guess = guess_params[q_id]

      # Convert BKT parameters to IRT parameters
      difficulty = np.log(p_guess)
      # Return IRT parameters
      irt_params = {

          'difficulty': difficulty
      }
      return~irt_params

  beta_bkt = []
  for q_id in list(question_dic.keys()):
    irt_parameters = bkt_to_irt(q_id)
    #print(irt_parameters)
    beta_bkt.append(irt_parameters['difficulty'])
  print(s)
  print(pd.DataFrame(zip(beta_bkt, beta)).corr().abs())
  correal_result.append(pd.DataFrame(zip(beta_bkt, beta)).corr().abs())
```

The results shown in Table 2 indicate the Pearson correlation calculated between the IRT difficulty parameters and the reconstructed difficulty parameters from BKT $\log(\pi_{\phi kp})$, which represents the forget parameter ϕ for the specific problem p in the knowledge component k. The results suggest that most of the difficulty parameters from IRT (b_{kp}) and the converted difficulty parameters from BKT ($\log(\pi_{\phi kp})$) generally were highly correlated, indicating the close alignment between the IRT and BKT models.

Table 2. Correlation between item difficulty parameters from the IRT and BKT models.

Knowledge Components	Number of Problems	Correlation
Calculate part in proportion with fractions	112	0.809
Calculate total in proportion with fractions	88	0.821
Calculate unit rate	200	0.808
Finding the intersection, GLF	22	0.550
Finding the intersection, Mixed	16	0.937
Finding the intersection, SIF	14	0.879
Plot decimal - thousandths	14	0.911
Plot imperfect radical	21	0.904
Plot pi	10	0.919
Plot terminating proper fraction	31	0.931
Plot whole number	9	0.797

5. Discussion

Personalized learning applications such as ITS and adaptive e-learning systems have gained increasing importance in recent years due to their potential to revolutionize education and learning processes. ITS has the ability to adapt to individual learning needs and styles by analyzing learners' performance history and predicting their future performance. ITS also generates vast amounts of data on learner interactions, performance, and progress. Researchers have proposed various modeling techniques, such as knowledge tracing [11], to leverage the data collected within ITS and gain valuable insights into the learning process. As a probabilistic framework, BKT harnesses learner data to make inferences about the learner's mastery of specific concepts or skills. This modeling technique uses utilizes Bayesian inference to update the probabilities of the learner's knowledge state based on binary response data (correct/incorrect answers). By analyzing the learner's current knowledge state, the model can estimate the likelihood of correctly answering future items or tasks. BKT can be used to model each learner's mastery of the knowledge in order to adapt to the difficulty of the material being presented, allowing for personalized and individualized learning experiences [27].

BKT differs from existing test theories, such as Cognitive Diagnostic Modeling (CDM) designed to measure students' proficiency levels in specific cognitive processes or components based on their performance in a test [30,35]. CDM explicitly models the presence or absence of specific cognitive skills that contribute to students' performance and categorizes students into different skill profiles. Similarly, IRT aims to quantify students' proficiency levels on a continuous scale based on their responses to a set of items associated with one or more latent traits. Unlike CDM and IRT, BKT utilizes Bayesian inference to estimate the probability of students' knowledge states at any given time. In other words, BKT updates students' knowledge estimates as new evidence (i.e., responses) becomes available, allowing for dynamic tracking of their learning progress. Thus, BKT is more suitable for tracking a student's knowledge state over time using an intelligent tutoring system, whereas IRT and CDM are often used for measuring a student's ability at a specific time point using computer-based or computerized adaptive tests.

Despite the growing popularity of BKT as a learner modeling technique, the use of BKT in practice has remained limited due to the lack of accessible and easy-to-use software to implement BKT models. In this paper, we present a practical guide to the estimation of BKT models using the pyBKT library in Python [22]. As an open-source software program, pyBKT enables researchers to estimate the standard BKT model and its variants while also allowing for new models to be proposed. With empirical data from Carnegie Learning's Cognitive Tutor programs, we demonstrate how BKT models can be used to model students' learning states in various math skills. First, we demonstrate the estimation of the standard BKT model and describe the key functions for extracting model parameters and evaluating model accuracy. Second, we compare the performance of the BKT and 1PL IRT models in modeling response accuracy (i.e., answer correctness). In the final case study, we examine the association between the difficulty parameters from IRT and the guess parameter from BKT. In addition to the built-in functions in pyBKT, we also provide custom functions for estimating the IRT models and transforming the BKT parameters.

Limitations and Future Research

This study has some limitations worth mentioning. For instance, this study focused specifically on the estimation of standard BKT models using the pyBKT library. However, recent research suggests that the application of more advanced modeling techniques (e.g., deep learning) to knowledge tracing can provide new directions for modeling complex learner data. Piech et al. [23] introduced Deep Knowledge Tracing (DKT) that trains a recurrent neural network on learner interactions to model learners' latent knowledge states and predict their future performance. Research suggests that DKT can provide substantial improvements over BKT, especially when large amounts of training data are available. For instance, Gorgun and Bulut [36] used the ASSISTment 2009–2010 skill builder dataset to compare the performance of Bayesian and deep knowledge tracing models in the presence of disengaged responses. The authors reported that DKT outperformed BKT in terms of model accuracy, before and after removing disengaged responses in the dataset. For researchers interested in using DKT models, we recommend the pyKT library [37] in Python (see the pyKT website for more details, https://pykt.org; accessed on 19 July 2023).

Another limitation of this study is that our case studies only involve dichotomously scored items (i.e., correct or incorrect). This is mainly because the pyBKT library can process response accuracy in three categories: -1 (no response), 0 (incorrect), or 1 (correct). However, KT models, including BKT, can be extended to polytomous items with partial credit, such as 0 (incorrect), 1 (partially correct), and 2 (fully correct). For instance, Wang and Heffernan [38] used Bayes Net Toolbox for Matlab (https://github.com/bayesnet/bnt; accessed on 19 July 2023) to estimate a BKT model by making the correctness continuous. Similarly, Ghosh et al. [39] proposed a KT method for predicting the exact option students select in multiple-choice items. These methodologies enable the application of BKT to other types of items, such as short-answer items, essays, and Likert-scale items, thereby extending the capabilities of today's intelligent tutoring and adaptive learning systems.

6. Conclusions

In conclusion, BKT is a powerful technique that plays a crucial role in modeling and understanding students' learning processes. By capturing the dynamics of student knowledge acquisition over time, BKT enables educators and researchers to gain insights into individual student progress and tailor instructional strategies accordingly. Its probabilistic nature allows for the estimation of the probability of a student's mastery of a concept, providing valuable information for personalized learning and targeted interventions.

The pyBKT library further enhances the utility of BKT by offering a user-friendly and efficient framework for estimating BKT models with ease. By leveraging the computational power of Python, pyBKT simplifies the implementation process, allowing researchers and educators to focus on analyzing educational data rather than grappling with complex coding. With pyBKT, users can seamlessly estimate BKT parameters, such as initial knowledge, learning and slip rates, and use these estimates to make accurate predictions about student performance. The library also supports model selection and comparison, enabling researchers to fine-tune their BKT models based on data-driven insights.

By combining the power of BKT with the convenience of the pyBKT library, educators and researchers can unlock new possibilities in the field of educational data mining. This combination empowers them to uncover patterns, identify struggling students, design personalized interventions, and ultimately improve learning outcomes. As educators and researchers continue to harness the potential of big data technologies, BKT and tools similar to pyBKT will remain indispensable in shaping the future of education.

Author Contributions: Conceptualization, O.B.; methodology, O.B., J.S., S.N.Y.-E. and G.G.; software, Z.A.P. and J.S.; formal analysis, O.B., J.S. and G.G.; writing—original draft preparation, O.B., J.S., S.N.Y.-E. and G.G.; visualization, J.S. and G.G.; supervision, O.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: A publicly available dataset was analyzed in this study. This dataset can be found here (accessed on 19 July 2023): https://raw.githubusercontent.com/CAHLR/pyBKT-examples/master/data/ct.csv. The Python codes used in this study can be downloaded from https://osf.io/u4ykg/ (accessed on 19 July 2023). The original version of the IRT estimation function used in this study is available at https://github.com/jasonli8408/Student-Diagnostic-Question-Prediction-Model/blob/main/final_project/part_a/item_response.py (accessed on 19 July 2023).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AUC	Area under the curve
BKT	Bayesian knowledge tracing
DKT	Deep Knowledge Tracing
HMM	Hidden Markov Model
IRT	Item Response Theory
IRT-BKT	Knowledge Tracing Model on Item Response Theory
ITS	Intelligent tutoring systems
KT-IDEM	Knowledge Tracing Item Difficulty Effect model
MAE	Mean absolute error
RMSE	Root-mean-square error
1PL	One-parameter logistic model
3PL	Three-parameter logistic model

Appendix A

Listing A1. Python example.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import~train_test_split

class IRT:
    def __init__(self, lr, iterations, user_size, question_size):
        self.lr = lr
        self.iterations = iterations
        self.user_size = user_size
        self.question_size = question_size
        self.theta = np.zeros(user_size)
        self.beta = np.zeros(question_size)
        self.val_acc_lst = []
        self.neg_lld_lst = []
        self.val_lld_lst = []

    def sigmoid(self, x):
        """ Apply sigmoid function. """
        return np.exp(x) / (1 + np.exp(x))

    def neg_log_likelihood(self, data):
        """ Compute the negative log-likelihood. """
        log_lklihood = 0.
        for ind in np.arange(len(data["is_correct"])):
            i = data["user_id"][ind]
            j = data["question_id"][ind]
            cij = data["is_correct"][ind]

            theta_i = self.theta[i]
            beta_j = self.beta[j]
            diff = theta_i - beta_j
            log_lklihood += cij * diff - np.log(1 + np.exp(diff))
        return~-log_lklihood

    def update_theta_beta(self, data):
        """ Update theta and beta using gradient descent. """
        diff_theta_beta = np.expand_dims(self.theta, axis=1) - np.expand_dims(self.beta, axis=0)
        sig = self.sigmoid(diff_theta_beta)

        grad_theta = np.zeros_like(diff_theta_beta)
        grad_beta = np.zeros_like(diff_theta_beta)

        for ind in np.arange(len(data["is_correct"])):
            i = data["user_id"][ind]
            j = data["question_id"][ind]
            cij = data["is_correct"][ind]

            grad_theta[i, j] = cij - sig[i, j]
            grad_beta[i, j] = sig[i, j] -~cij

        self.theta = self.theta + self.lr * np.sum(grad_theta, axis=1)
        self.beta = self.beta + self.lr * np.sum(grad_beta, axis=0)

    def evaluate(self, data):
        """ Evaluate the model given data and return the accuracy. """
        pred = []
        for i, q in enumerate(data["question_id"]):
            u = data["user_id"][i]
            x = (self.theta[u] - self.beta[q]).sum()
            p_a = self.sigmoid(x)
            pred.append(p_a >= 0.5)
        return np.sum((data["is_correct"] == np.array(pred))) / len(data["is_correct"])

    def irt(self, train_data, val_data):
        for i in range(self.iterations):
            neg_lld = self.neg_log_likelihood(train_data)
            score = self.evaluate(val_data)
            self.val_acc_lst.append(score)
            self.neg_lld_lst.append(neg_lld)
            self.val_lld_lst.append(self.neg_log_likelihood(val_data))
            print("NLLK: {} \t Score: {}".format(neg_lld, score))
            self.update_theta_beta(train_data)

        return self.theta, self.beta, self.val_acc_lst, self.neg_lld_lst, self.val_lld_lst
```

References

1. Pelánek, R. Bayesian knowledge tracing, logistic models, and beyond: An overview of learner modeling techniques. *User Model. User-Adapt. Interact.* **2017**, *27*, 313–350. [CrossRef]
2. Aleven, V.; Koedinger, K.R. Knowledge component (KC) approaches to learner modeling. *Des. Recomm. Intell. Tutoring Syst.* **2013**, *1*, 165–182.
3. Chrysafiadi, K.; Virvou, M. Student modeling approaches: A literature review for the last decade. *Expert Syst. Appl.* **2013**, *40*, 4715–4729. [CrossRef]
4. Woolf, B.P. Student Modeling. In *Advances in Intelligent Tutoring Systems*; Nkambou, R., Bourdeau, J., Mizoguchi, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 267–279.
5. Harrison, B.; Roberts, D. A review of student modeling techniques in intelligent tutoring systems. In Proceedings of the 8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Stanford, CA, USA, 8–12 October 2012; Swanson, R., Vaish, R., Orkin, J., Niehaus, J., Godwin, J.A., Guarino, S., Youngblood, G.M., Eds.; The AAAI Press: Washington, DC, USA, 2012; pp. 61–66. [CrossRef]
6. Am, E.H.; Hidayah, I.; Kusumawardani, S.S. A Literature Review of Knowledge Tracing for Student Modeling: Research Trends, Models, Datasets, and Challenges. *J. Inf. Technol. Comput. Sci.* **2021**, *6*, 344. [CrossRef]
7. Xiong, X.; Zhao, S.; Van Inwegen, E.G.; Beck, J.E. Going deeper with deep knowledge tracing. In Proceedings of the International Conference on Educational Data Mining (EDM), Raleigh, NC, USA, 29 June–2 July 2016.
8. Minn, S.; Yu, Y.; Desmarais, M.C.; Zhu, F.; Vie, J.J. Deep knowledge tracing and dynamic student classification for knowledge tracing. In Proceedings of the 2018 IEEE International Conference on data Mining (ICDM), Singapore, 17–20 November 2018; pp. 1182–1187.
9. Sapountzi, A.; Bhulai, S.; Cornelisz, I.; van Klaveren, C. Dynamic models for knowledge tracing & prediction of future performance. In Proceedings of the Seventh International Conference on Data Analytics, Athens, Greece, 18–22 November 2019.
10. Carlon, M.K.J.; Cross, J.S. Knowledge tracing for adaptive learning in a metacognitive tutor. *Open Educ. Stud.* **2022**, *4*, 206–224. [CrossRef]
11. Corbett, A.T.; Anderson, J.R. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Model. User-Adapt. Interact.* **1994**, *4*, 253–278. [CrossRef]
12. Pardos, Z.; Bergner, Y.; Seaton, D.; Pritchard, D. Adapting bayesian knowledge tracing to a massive open online course in edx. In Proceedings of the Educational Data Mining 2013, Memphis, TN, USA, 6–9 July 2013; Citeseer: Princeton, NJ, USA, 2013.

13. Hawkins, W.J.; Heffernan, N.T.; Baker, R.S. Learning Bayesian knowledge tracing parameters with a knowledge heuristic and empirical probabilities. In Proceedings of the Intelligent Tutoring Systems: 12th International Conference, ITS 2014, Honolulu, HI, USA, 5–9 June 2014; Proceedings 12; Springer: Berlin/Heidelberg, Germany, 2014; pp. 150–155.
14. Vaisakh, K.; Ravi, A.; Aakash, K.; Sai, A.; Bhaskar, J. SkillCrest: A Skill Assessment System Using Deep Knowledge Tracing and User Feedback Sentiment Analysis. In Proceedings of the 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 1–3 October 2021; pp. 1–7.
15. Ritter, S.; Anderson, J.R.; Koedinger, K.R.; Corbett, A. Cognitive Tutor: Applied research in mathematics education. *Psychon. Bull. Rev.* **2007**, *14*, 249–255. [CrossRef] [PubMed]
16. Sao Pedro, M.; Baker, R.; Gobert, J. Incorporating scaffolding and tutor context into bayesian knowledge tracing to predict inquiry skill acquisition. In Proceedings of the Educational Data Mining 2013, Memphis, TN, USA, 6–9 July 2013; Citeseer: Princeton, NJ, USA, 2013.
17. Mao, Y. Deep Learning vs. Bayesian Knowledge Tracing: Student Models for Interventions. *J. Educ. Data Min.* **2018**, *10*, 28–54.
18. David, Y.B.; Segal, A.; Gal, Y. Sequencing educational content in classrooms using Bayesian knowledge tracing. In Proceedings of the Sixth International Conference on Learning Analytics & Knowledge, Edinburgh, UK, 25–29 April 2016; pp. 354–363.
19. Baker, R.S.D.; Corbett, A.T.; Gowda, S.M.; Wagner, A.Z.; MacLaren, B.A.; Kauffman, L.R.; Mitchell, A.P.; Giguere, S. Contextual slip and prediction of student performance after use of an intelligent tutor. In Proceedings of the User Modeling, Adaptation, and Personalization: 18th International Conference, UMAP 2010, Big Island, HI, USA, 20–24 June 2010; Proceedings 18; Springer: Berlin/Heidelberg, Germany, 2010; pp. 52–63.
20. Naeem, M.; Tidswell, A.; Magdy, Y. Bayesian Knowledge Tracing for Assessment Results Analysis. In Proceedings of the 2021 17th International Computer Engineering Conference (ICENCO), Cairo, Egypt, 29–30 December 2021; pp. 30–34.
21. Qiu, Y.; Qi, Y.; Lu, H.; Pardos, Z.A.; Heffernan, N.T. Does Time Matter? Modeling the Effect of Time with Bayesian Knowledge Tracing. In Proceedings of the EDM, Eindhoven, The Netherlands, 6–8 July 2011; pp. 139–148.
22. Badrinath, A.; Wang, F.; Pardos, Z. pyBKT: An accessible Python library of Bayesian Knowledge Tracing Models. In Proceedings of the 14th International Conference on Educational Data Mining, Virtual Event, 29 June–2 July 2021; pp. 468–474.
23. Piech, C.; Bassen, J.; Huang, J.; Ganguli, S.; Sahami, M.; Guibas, L.J.; Sohl-Dickstein, J. Deep Knowledge Tracing. In Proceedings of the Advances in Neural Information Processing Systems, NIPS 2015, Montreal, QC, Canada, 7–12 December 2015; Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2015; Volume 28.
24. Corbett, A. Cognitive computer tutors: Solving the two-sigma problem. In Proceedings of the User Modeling 2001: 8th International Conference, UM 2001, Sonthofen, Germany, 13–17 July 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 137–147.
25. Käser, T.; Klingler, S.; Schwing, A.G.; Gross, M. Dynamic Bayesian networks for student modeling. *IEEE Trans. Learn. Technol.* **2017**, *10*, 450–462. [CrossRef]
26. Pardos, Z.A.; Heffernan, N.T. KT-IDEM: Introducing item difficulty to the knowledge tracing model. In Proceedings of the User Modeling, Adaption and Personalization: 19th International Conference, UMAP 2011, Girona, Spain, 11–15 July 2011; Proceedings 19; Springer: Berlin/Heidelberg, Germany, 2011; pp. 243–254.
27. Yudelson, M.V.; Koedinger, K.R.; Gordon, G.J. Individualized bayesian knowledge tracing models. In Proceedings of the Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, 9–13 July 2013; Proceedings 16; Springer: Berlin/Heidelberg, Germany, 2013; pp. 171–180.
28. Baker, R.S.d.; Corbett, A.T.; Aleven, V. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In Proceedings of the Intelligent Tutoring Systems: 9th International Conference, ITS 2008, Montreal, QC, Canada, 23–27 June 2008; Proceedings 9; Springer: Berlin/Heidelberg, Germany, 2008; pp. 406–415.
29. Deonovic, B.; Yudelson, M.; Bolsinova, M.; Attali, M.; Maris, G. Learning meets assessment: On the relation between item response theory and Bayesian knowledge tracing. *Behaviormetrika* **2018**, *45*, 457–474. [CrossRef]
30. Wang, F.; Huang, Z.; Liu, Q.; Chen, E.; Yin, Y.; Ma, J.; Wang, S. Dynamic Cognitive Diagnosis: An Educational Priors-Enhanced Deep Knowledge Tracing Perspective. *IEEE Trans. Learn. Technol.* **2023**, *16*, 306–323. [CrossRef]
31. Kang, T.; Cohen, A.S.; Sung, H.J. IRT model selection methods for polytomous items. In Proceedings of the Annual Meeting of the National Council on Measurement in Education, Montreal, QC, Canada, 11–15 April 2005.
32. Khajah, M.M.; Huang, Y.; González-Brenes, J.P.; Mozer, M.C.; Brusilovsky, P. Integrating knowledge tracing and item response theory: A tale of two frameworks. In Proceedings of the CEUR Workshop Proceedings, Crete, Greece, 27 May 2014; University of Pittsburgh: Pittsburg, PA, USA, 2014; Volume 1181, pp. 7–15.
33. Wang, S.; Han, Y.; Wu, W.; Hu, Z. Modeling student learning outcomes in studying programming language course. In Proceedings of the 2017 Seventh International Conference on Information Science and Technology (ICIST), Da Nang, Vietnam, 16–19 April 2017; pp. 263–270.
34. Xu, Y.; Mostow, J. Using item response theory to refine knowledge tracing. In Proceedings of the Educational Data Mining 2013, Memphis, TN, USA, 6–9 July 2013.
35. Deonovic, B.; Chopade, P.; Yudelson, M.; de la Torre, J.; von Davier, A.A. Application of cognitive diagnostic models to learning and assessment systems. *Handbook of Diagnostic Classification Models: Models and Model Extensions, Applications, Software Packages*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 437–460.

36. Gorgun, G.; Bulut, O. Considering Disengaged Responses in Bayesian and Deep Knowledge Tracing. In Proceedings of the Artificial Intelligence in Education, Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners' and Doctoral Consortium, Durham, UK, 27–31 July 2022; Rodrigo, M.M., Matsuda, N., Cristea, A.I., Dimitrova, V., Eds.; Springer: Cham, Switzerland, 2022; pp. 591–594.
37. Liu, Z.; Liu, Q.; Chen, J.; Huang, S.; Tang, J.; Luo, W. pyKT: A Python Library to Benchmark Deep Learning based Knowledge Tracing Models. *arXiv* **2023**, arXiv:cs.LG/2206.11460.
38. Wang, Y.; Heffernan, N. Extending knowledge tracing to allow partial credit: Using continuous versus binary nodes. In Proceedings of the Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, 9–13 July 2013; Proceedings 16; Springer: Berlin/Heidelberg, Germany, 2013; pp. 181–188.
39. Ghosh, A.; Raspat, J.; Lan, A. Option Tracing: Beyond Correctness Analysis in Knowledge Tracing. *arXiv* **2021**, arXiv.2104.09043. [CrossRef]