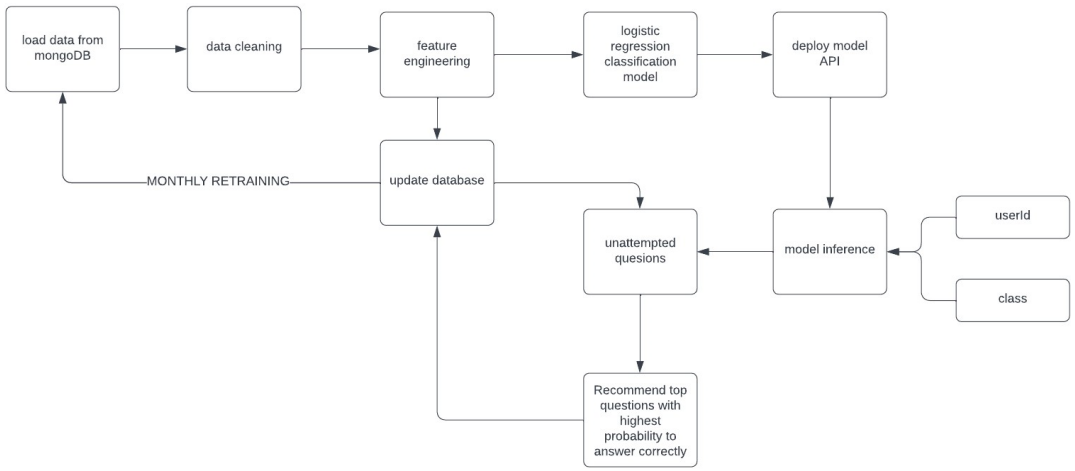
Recommender System Development



Model workflow for sequential question recommendation

Knowledge tracing is a method of modeling learner's knowledge over time to predict their future performance. This is critical in adaptive learning systems and intelligent tutoring systems which deliver personalized content based on the learner's knowledge state. Logistic regression is one commonly used algorithm in knowledge tracing.

Here's an introduction to the steps for applying logistic regression to build sequential knowledge tracing models:

1. **Data Collection**: The first step is to gather and prepare the data. You'll typically need data on the sequences of problems or tasks each student worked on, whether they succeeded or failed, and any relevant features about the students or tasks (such as previous performance, difficulty level of tasks, etc.)
2. **Feature Engineering**: Once you have the raw data, the next step is to engineer appropriate features that could be indicative of a learner's knowledge. For example, you might have binary features indicating whether a student answered correctly on previous questions, or numerical features indicating the average score over some

number of recent problems. It's often beneficial to include temporal information as well.

3. **Model Training**: The next step is to train your logistic regression model. In the simplest case, this involves a binary outcome variable (did the student get the next question right or not), and your chosen feature variables. Logistic regression models the log-odds of the probability of an event happening (in this case, answering the question correctly) as a linear combination of the input features.
4. **Model Validation**: After the model has been trained, you must validate the model's performance. This typically involves splitting your dataset into a training set and a validation set. The model is trained on the training set and then its performance is evaluated on the validation set. Common metrics for evaluation include the AUC-ROC, log loss, accuracy, and others. It's critical to perform this step to ensure your model is generalizable and not overfitting to your training data.
5. **Model Iteration**: After validating your model, you may need to go back to steps 2-4 multiple times. Perhaps you find new features to engineer, or need to change how you're pre-processing your data. This is an iterative process that involves continually refining your model.
6. **Model Deployment**: Once you're satisfied with your model's performance, the next step is deployment. This involves integrating your model into the e-learning platform so that it can make real-time predictions. In the context of knowledge tracing, these predictions would likely be used to adapt the learning experience in real time, such as selecting the next most appropriate task for the student.
7. **Model Monitoring and Maintenance**: After deployment, continue to monitor your model's performance to ensure it is working as expected. As you collect more data, you can also retrain your model to ensure it's leveraging the most recent data.

Literature Review: Bayesian Knowledge Tracing for E-Learning

Bayesian Knowledge Tracing (BKT) is a widely used model in the field of adaptive learning and intelligent tutoring systems for tracing the evolving knowledge state of students (Zhang et al., 2017). The primary goal of BKT is to personalize the practice sequence and optimize the learning process for individual students (Zhang et al., 2017). However, existing methods such as BKT and Deep Knowledge Tracing (DKT) have

limitations in accurately modeling a student's knowledge state and identifying specific concepts that the student is proficient in or unfamiliar with (Zhang et al., 2017).

To address these limitations, researchers have proposed various approaches and models in the field of knowledge tracing. One such model is the Dynamic Key-Value Memory Networks (DKVMN) introduced by (Zhang et al., 2017). DKVMN is designed to exploit the relationships between underlying concepts and directly output a student's mastery level of each concept (Zhang et al., 2017). The model consists of a static matrix called "key" that stores the knowledge concepts and a dynamic matrix called "value" that stores and updates the mastery levels of corresponding concepts (Zhang et al., 2017). Experimental results have shown that DKVMN outperforms existing models in a range of knowledge tracing datasets (Zhang et al., 2017).

A literature review by provides an overview of the research trends, models, datasets, and challenges in knowledge tracing for student modeling (Am et al., 2021). The review highlights that there are two main approaches used in knowledge tracing: probabilistic and deep learning (Am et al., 2021). BKT is the most widely used model in the probabilistic approach, while DKT is popular in the deep learning approach (Am et al., 2021). The review also mentions that the ASSISTments 2009-2010 dataset is frequently used for evaluating probabilistic and deep learning approaches in knowledge tracing (Am et al., 2021). The review suggests the need for additional studies to explore and develop new models in knowledge tracing (Am et al., 2021).

In addition to the models and approaches, there are also efforts to develop accessible tools and libraries for implementing Bayesian Knowledge Tracing. introduces pyBKT, an accessible and computationally efficient library of model extensions for Bayesian Knowledge Tracing (Badrinath, 2021). The library provides data generation, fitting, prediction, and cross-validation routines, making knowledge tracing more accessible to researchers and practitioners (Badrinath, 2021).

Furthermore, the connection between Bayesian Knowledge Tracing and Item Response Theory (IRT) is explored in a study by (Deonovic et al., 2018). The study highlights the fundamental connection between these two models and emphasizes the role of education in modeling learner data (Deonovic et al., 2018). The stationary distribution of the latent variable and the observed response variable in Bayesian Knowledge Tracing are related to an IRT model, indicating the importance of considering educational factors in knowledge tracing models (Deonovic et al., 2018).

Several studies have also examined the application of Bayesian Knowledge Tracing in specific educational contexts. For example, analyze the use of BKT in assessing the learning curve of students in an introductory algorithms discipline (Raposo et al., 2019). The results show the promise of BKT in designing intelligent tutors that respond to individual student performance and recommend appropriate content during learning (Raposo et al., 2019).

Overall, Bayesian Knowledge Tracing is a widely used model in the field of e-learning and adaptive learning systems. Researchers have proposed various extensions and approaches to improve the accuracy and effectiveness of knowledge tracing. The development of accessible tools and libraries, as well as the exploration of connections with other models, further contribute to the advancement of knowledge tracing in e-learning. Future research should focus on developing new models and exploring the integration of machine learning techniques to enhance the efficiency of adaptive learning systems (Pliakos et al., 2019).

There are several papers where logistic regression was applied in knowledge tracing.

One such paper is by (Baker & Inventado, 2014), where they mention that logistic regression, specifically the Performance Factors Assessment (PFA) approach proposed by Pavlik et al. in 2009, can be effective for cases where multiple skills are relevant to a problem or problem step at the same time. Logistic regression models, such as PFA, predict students' mastery of knowledge concepts by analyzing the relationship among factors that have an impact on students' answering accuracy.

Another paper by Ceccarelli et al. (2022) discusses the use of logistic regression models in knowledge tracing. They mention that logistic regression models, based on item response theory (IRT), are commonly used to predict students' mastery of knowledge concepts by analyzing the relationship among factors that impact students' answering accuracy.

In a study by (Yeung & Yeung, 2018), logistic regression is mentioned as one of the approaches used to solve the knowledge tracing problem. They highlight that logistic regression models, such as the one used in performance factors analysis (PFA), have been developed as an alternative to Bayesian knowledge tracing (BKT) based on the hidden Markov model (HMM).

Furthermore, logistic regression is mentioned in a literature review by (Am et al., 2021), where they discuss the modeling of knowledge and skills in knowledge tracing. They mention that logistic models, including Bayesian knowledge tracing and logistic regression models, have been considered in the literature.

Other papers also mention the use of logistic regression in knowledge tracing. For example, Minn et al. (2018) mention that logistic regression is one of the techniques explored in the field of intelligent tutoring systems for knowledge tracing. Sarsa et al. (2021) mention logistic regression as one of the baseline models used for comparison in their study on deep learning models for knowledge tracing. Rodrigues et al. (2022) mention logistic regression as one of the classic knowledge tracing models, along with Bayesian knowledge tracing.

In summary, logistic regression has been applied in knowledge tracing as an alternative to Bayesian knowledge tracing and as a way to model the relationship between factors impacting students' answering accuracy. It has been used in various studies and is considered one of the classic models in the field of knowledge tracing.

Python script for recommender system model (codes on GitHub)

The Python code snippet provided is a recommendation system for an e-learning platform that uses a trained logistic regression model to predict which questions a student is likely to answer correctly, given their past performance. The system uses MongoDB as its database.

Let's break down the code block by block:

1. **Imports and Connection to MongoDB Database**: The script first imports necessary modules and then connects to the MongoDB client using the connection string stored in an environment variable. It also connects to a specific database 'afrilearn'.

```
from bson import ObjectId
import pandas as pd
import pickle
import random
import os
from pymongo import MongoClient

cluster = os.environ['RECODB_KEY']
```

```
client = MongoClient(cluster)
db = client.afrilearn
```

1. **Function: get_prob_correct()**: This function accepts a trained model and a set of unattempted questions. It drops the 'next_attempt' column from the DataFrame of questions and calculates the probability of correctly answering each question using the model's `predict_proba` method. It then returns these probabilities.

```
def get_prob_correct(model,unattempted_questions):
    X = unattempted_questions.drop(['next_attempt'],axis=1)
    prob_correct = model.predict_proba(X)[:,1]
    return prob_correct
```

1. **Function: get_recommendations()**: This function recommends questions for a user based on their encoded user ID, the type of recommendation (either 'lesson' or 'subject'), and other parameters.
    - The function begins by mapping a course ID to a class label using a predefined dictionary.
    - If a user ID is provided, the function then loads a pickled dictionary of label encoders and extracts the specific encoder for the user ID.
    - Using the encoded user ID, it retrieves from the database a DataFrame of questions that the user has not yet attempted. If there are no unattempted questions, it instead retrieves all questions associated with the user.
    - The function then loads a pickled dictionary of classifiers and selects the one associated with the class label.
    - It calculates the probability of the user correctly answering each unattempted question and adds these probabilities as a new column in the DataFrame.
    - Depending on the type of recommendation, the function then recommends questions based on lesson ID or subject name, or simply recommends the questions that the user is most likely to answer correctly.
    - If no user ID is provided, the function randomly selects questions from a pickled DataFrame.

```
def get_recommendations(courseId,n_questions,userId=None,rec_type=None,lessonId=None,subje
ct_name=None):
    ...
    ...
    return recommended_questions
```

In summary, this script provides a recommendation system for an e-learning platform, recommending questions to users based on their predicted probability of correctly answering each question. It interfaces with a MongoDB database and uses pickled logistic regression classifiers and label encoders to make its predictions and recommendations.

Python script for model deployment (code on GitHub)

This Python script is for a web application built using the Flask framework. The application is designed to interact with a recommendation system (which seems to be defined in a separate Python module named `recommender`) to suggest questions for a user in an e-learning platform. Here's the breakdown of what each part of the script does:

1. **Importing Required Libraries and Modules**:
    - `Flask` is the web framework used to build the application.
    - `render_template` is used to render HTML templates.
    - `request` is used to handle HTTP requests.
    - `jsonify` is used to send JSON responses.
    - `CORS` (Cross Origin Resource Sharing) is used to enable cross-origin requests.
    - `ObjectId` is imported from `bson` to handle MongoDB ObjectIds.
    - `get_recommendations` is a function imported from the `recommender` module to get question recommendations for users.

```
from flask import Flask, render_template, request, jsonify
from flask_cors import CORS
from bson import ObjectId
from recommender import get_recommendations
```

2. **Creating the Flask Application and Enabling CORS**:
    - The `Flask` application is created with the name `app`.
    - `CORS` is applied to the Flask application, allowing any domain to interact with this application.

```
app = Flask(__name__)
CORS(app, resources={r"*":{"origins":"*"}})
```

3. **Defining Application Routes and Their Functions**:

    a. **Main Page**:

    - When users access the root URL (`'/'`), they are served with an HTML page named `'questions.html'`.

```
@app.route('/')
def main():
    return render_template('questions.html')
```

    b. **Recommendation System Endpoint**:

    - Users can either send a `GET` or `POST` request to the `/recommend` endpoint.
    - If a `GET` request is received, the application sends back a JSON response indicating that a `POST` request is needed.
    - If a `POST` request is received, the application expects certain parameters in the JSON payload (e.g., `'userId'`, `'class_name'`, `'n_questions'`, `'rec_type'`, etc.).
    - It uses these parameters to call the `get_recommendations` function, which presumably returns a list of recommended questions.
    - The returned questions are then sent back to the client.

```
@app.route('/recommend', methods=['GET','POST'])
def reco_system():
    ...
    ...
    return questions
```

c. **Submit Endpoint**:

- This is another endpoint, `/submit`, which can handle both `GET` and `POST` requests.
- The application expects data from a form submission, uses this data to call the `get_recommendations` function and then returns the recommended questions or redirects the user back to the `'index.html'` or `'questions.html'` page with an appropriate message.

```
@app.route('/submit', methods=['POST', 'GET'])
def submit():
    ...
    ...
    return questions
```

4. **Running the Flask Application**:

- Finally, the script checks if it is the main module that is being run and, if so, starts the Flask development server.

```
if __name__ == '__main__':
    app.debug = False
    app.run()
```

In summary, this script is creating a simple web application using Flask to serve as a front-end for a question recommendation system in an e-learning platform. The application has three routes: the main page, a recommendation endpoint, and a submit endpoint for handling question recommendations based on different HTTP request methods and payloads.

Python script for data preprocessing (code on GitHub)

This Python script appears to be part of a machine learning pipeline to predict whether a user will attempt a particular question next, based on synthetic user interaction data with an e-learning platform. The script uses the Pandas library to manipulate data, PyMongo to interact with a MongoDB database, and Scikit-learn for building a Logistic Regression model. Below is the detailed explanation of each part of the script:

1. **Importing Required Libraries and Modules**:
    - `pandas`, `pymongo`, `sklearn`, `random`, and `os` modules are imported.

```
import pandas as pd
from pymongo import MongoClient
from sklearn.preprocessing import LabelEncoder
...
import os
```

2. **Database Connection Setup**:
    - The script retrieves a connection string for MongoDB from an environment variable `MAINDB_KEY` and establishes a connection with a MongoDB database named `afrilearn`.

```
main_cluster = os.environ['MAINDB_KEY']
client = MongoClient(main_cluster)
db = client.afrilearn
```

3. `get_classes` **Function**:
    - This function takes a DataFrame `df` as input, and merges it with a mapping of lessons to classes from a CSV file, returning the merged DataFrame.

```
def get_classes(df):
    ...
    return merged
```

4. `get_user_data` **Function**:
    - This function takes `courseId` as a parameter and retrieves relevant user activity and question data from the MongoDB database. It returns a list of `userIds` who are enrolled in the specified course, a DataFrame `questions_df` containing questions related to that course, and the raw questions DataFrame `questions`.

```
def get_user_data(courseId):
    ...
    return userIds, questions_df, questions
```

5. `get_synthetic_data` **Function**:

   - This function takes a `class_name` as input and simulates synthetic user interaction data with questions, including the number of attempts per question, success in each attempt, and whether the user will attempt the question next. It returns a DataFrame of this synthetic data.

```
def get_synthetic_data(class_name:str) -> pd.DataFrame:
    ...
    return responses_df
```

6. `label_transforms` **Function**:

   - This function takes a DataFrame `responses_df` and applies label encoding to several columns (e.g., 'difficulty', 'subjectId', 'userId', 'questionId'). It returns the transformed DataFrame and the label encoders used.

```
def label_transforms(responses_df):
    ...
    return responses_df, le_difficulty, le_subjectId, le_userId, le_questionId
```

7. `get_training_data` **Function**:

   - This function takes a DataFrame `responses_df` and prepares it for training a machine learning model. It applies label transformations, splits the data into features (`X`) and target (`Y`), and further splits these into training and testing sets.

```
def get_training_data(responses_df):
    ...
    return x_train, x_test, y_train, y_test
```

8. `classification` **Function**:

   - This function takes a DataFrame `responses_df`, retrieves the training and testing data, trains a Logistic Regression classifier using the training data, and evaluates its accuracy on the testing data. It returns the trained classifier and the accuracy score.

```
def classification(responses_df):
    ...
    return classifier, accuracy
```

In summary, this script appears to be simulating user interactions with an e-learning platform to create a dataset. It then processes this data, transforming it into a format suitable for machine learning, and trains a Logistic Regression model to predict whether a user will attempt a particular question next based on their interaction history. The script is organized as a set of functions, each responsible for a specific part of the data processing, feature engineering, and model training pipeline.

References

Am, E., Hidayah, I., Kusumawardani, S. (2021). A Literature Review Of Knowledge Tracing For Student Modeling : Research Trends, Models, Datasets, and Challenges. JITeCS, 2(6). https://doi.org/10.25126/jitecs.202162344

Am, E., Hidayah, I., Kusumawardani, S. (2021). A Literature Review Of Knowledge Tracing For Student Modeling : Research Trends, Models, Datasets, and Challenges. JITeCS, 2(6). https://doi.org/10.25126/jitecs.202162344

Aslam, S., Jilani, A., Sultana, J., Almutairi, L. (2021). Feature Evaluation Of Emerging E-learning Systems Using Machine Learning: An Extensive Survey. IEEE Access, (9), 69573-69587. https://doi.org/10.1109/access.2021.3077663

Badrinath, A. (2021). Pybkt: An Accessible Python Library Of Bayesian Knowledge Tracing Models.. https://doi.org/10.48550/arxiv.2105.00385

Baker, R., Inventado, P. (2014). Educational Data Mining and Learning Analytics., 61-75. https://doi.org/10.1007/978-1-4614-3305-7_4

Ceccarelli, M., Su, W., Huang, T., Shi, J. (2022). Ntm-based Skill-aware Knowledge Tracing For Conjunctive Skills. Computational Intelligence and Neuroscience, (2022), 1-16. https://doi.org/10.1155/2022/9153697

Deonovic, B., Yudelson, M., Bolsinova, M., Attali, M., Maris, G. (2018). Learning Meets Assessment. Behaviormetrika, 2(45), 457-474. https://doi.org/10.1007/s41237-018-0070-z

Halpern, D., Tubridy, S., Wang, H., Gasser, C., Popp, P., Davachi, L., … & Gureckis, T. (2018). Knowledge Tracing Using the Brain.. https://doi.org/10.31234/osf.io/fmj48

Liu, H., Zhang, T., Li, F., Gu, Y. (2021). Tracking Knowledge Structures and Proficiencies Of Students With Learning Transfer. IEEE Access, (9), 55413-55421.

https://doi.org/10.1109/access.2020.3032141
Minn, S., Yu, Y., Desmarais, M., Zhu, F., Vie, J. (2018). Deep Knowledge Tracing and Dynamic Student Classification For Knowledge Tracing..
https://doi.org/10.1109/icdm.2018.00156
Pliakos, K., Joo, S., Park, J., Cornillie, F., Vens, C., Noortgate, W. (2019). Integrating Machine Learning Into Item Response Theory For Addressing the Cold Start Problem In Adaptive Learning Systems. Computers & Education, (137), 91-103.
https://doi.org/10.1016/j.compedu.2019.04.009
Raposo, A., Maranhao, D., Neto, C. (2019). Analise Do Modelo Bkt Na Avaliacão Da Curva De Aprendizagem De Alunos De Algoritmos..
https://doi.org/10.5753/cbie.sbie.2019.479
Rodrigues, T., Souza, J., Bernardino, H., Baker, R. (2022). Towards Interpretability Of Attention-based Knowledge Tracing Models.. https://doi.org/10.5753/sbie.2022.224685
Sarsa, S., Leinonen, J., Hellas, A. (2021). Empirical Evaluation Of Deep Learning Models For Knowledge Tracing: Of Hyperparameters and Metrics On Performance And Replicability.. https://doi.org/10.48550/arxiv.2112.15072
Yeung, C., Yeung, D. (2018). Addressing Two Problems In Deep Knowledge Tracing Via Prediction-consistent Regularization.. https://doi.org/10.1145/3231644.3231647
Zhang, J., Shi, X., King, I., Yeung, D. (2017). Dynamic Key-value Memory Networks For Knowledge Tracing.. https://doi.org/10.1145/3038912.3052580
Zhang, Q., Yang, D., Fang, P., Liu, N., Zhang, L. (2020). Develop Academic Question Recommender Based On Bayesian Network For Personalizing Student’s Practice. Int. J. Emerg. Technol. Learn., 18(15), 4. https://doi.org/10.3991/ijet.v15i18.11594