Knowledge Tracing

Lukas Olson, Marjorie Ivy, Roshni Dhanasekar, and Varun Krishnaswamy

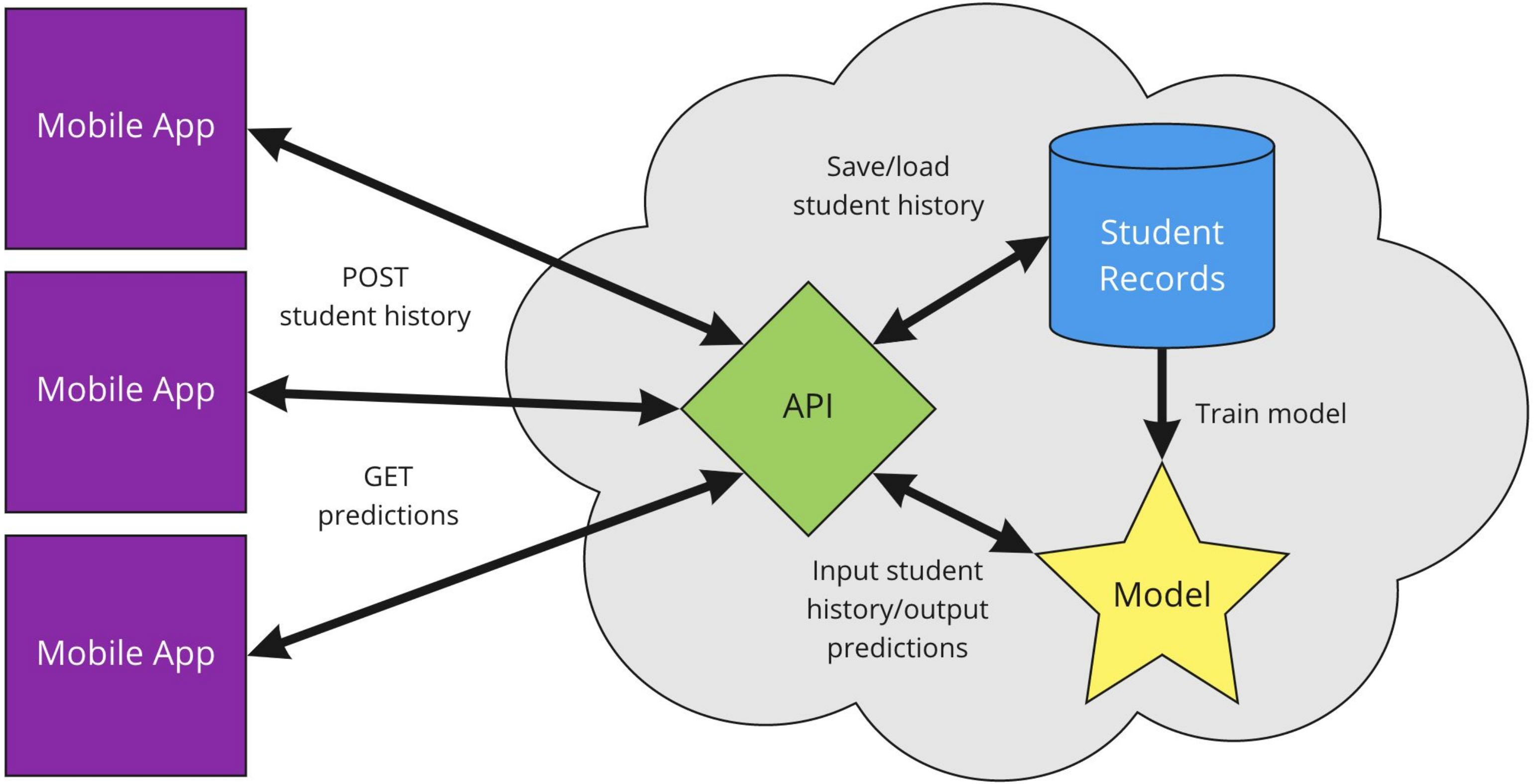
https://github.gatech.edu/VIP-ITS/Knowledge-Tracing

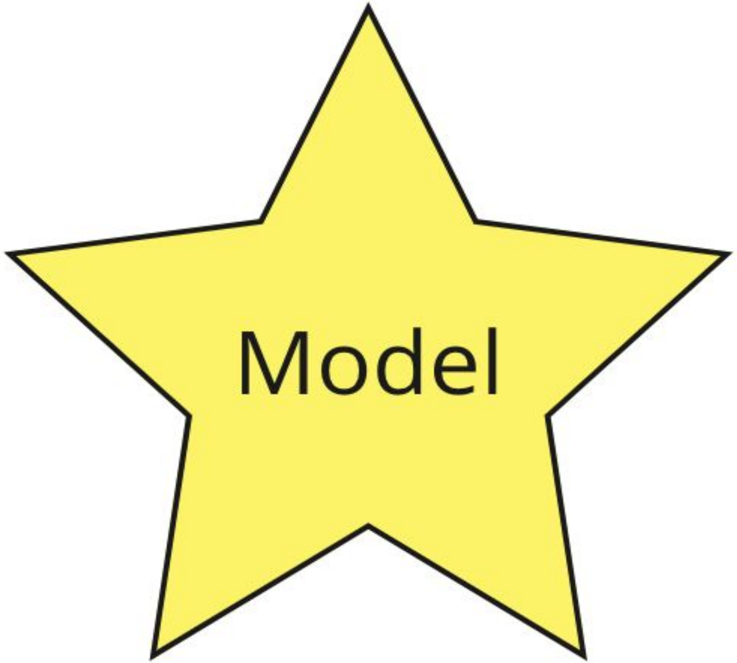
Motivation

Problem: Normal quiz apps, e.g. Quizlet, Kahoot, help you study on the go, but how do you prioritize what to study?

Solution: Knowledge tracing predicts what a student knows so we can recommend what they need to study.

Architecture

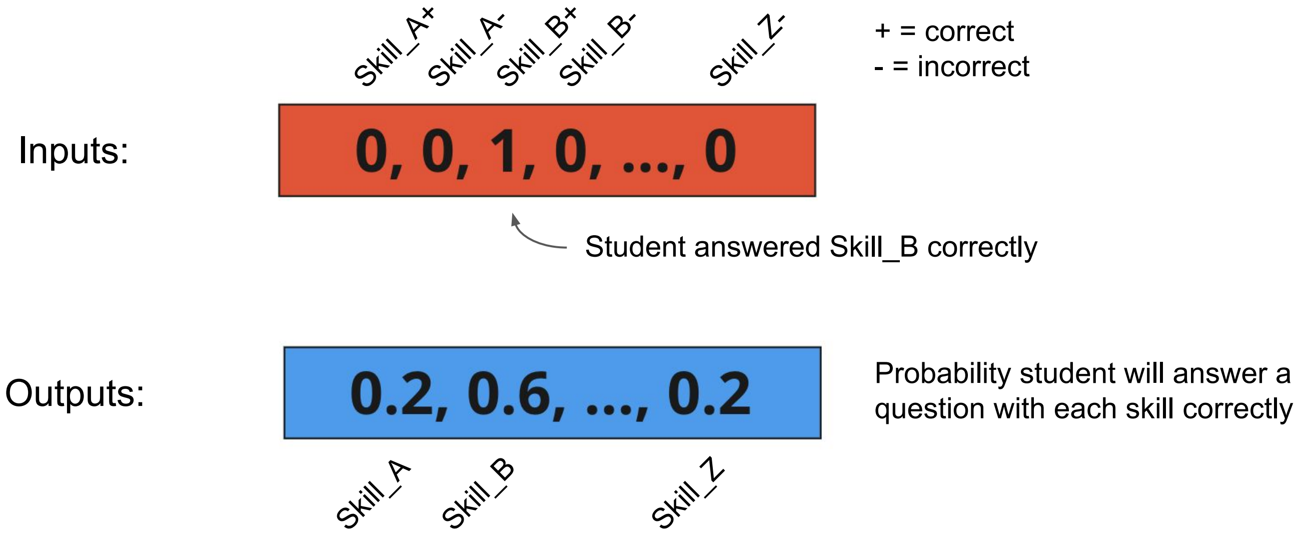




Model

Deep Knowledge Tracing

Predict the probability that a student will answer a question correctly given their previous answers to other questions.



Skill_A+ Skill_A- Skill_B+ Skill_B- Skill_Z-

+ = correct
- = incorrect

Inputs: 0, 0, 1, 0, ..., 0

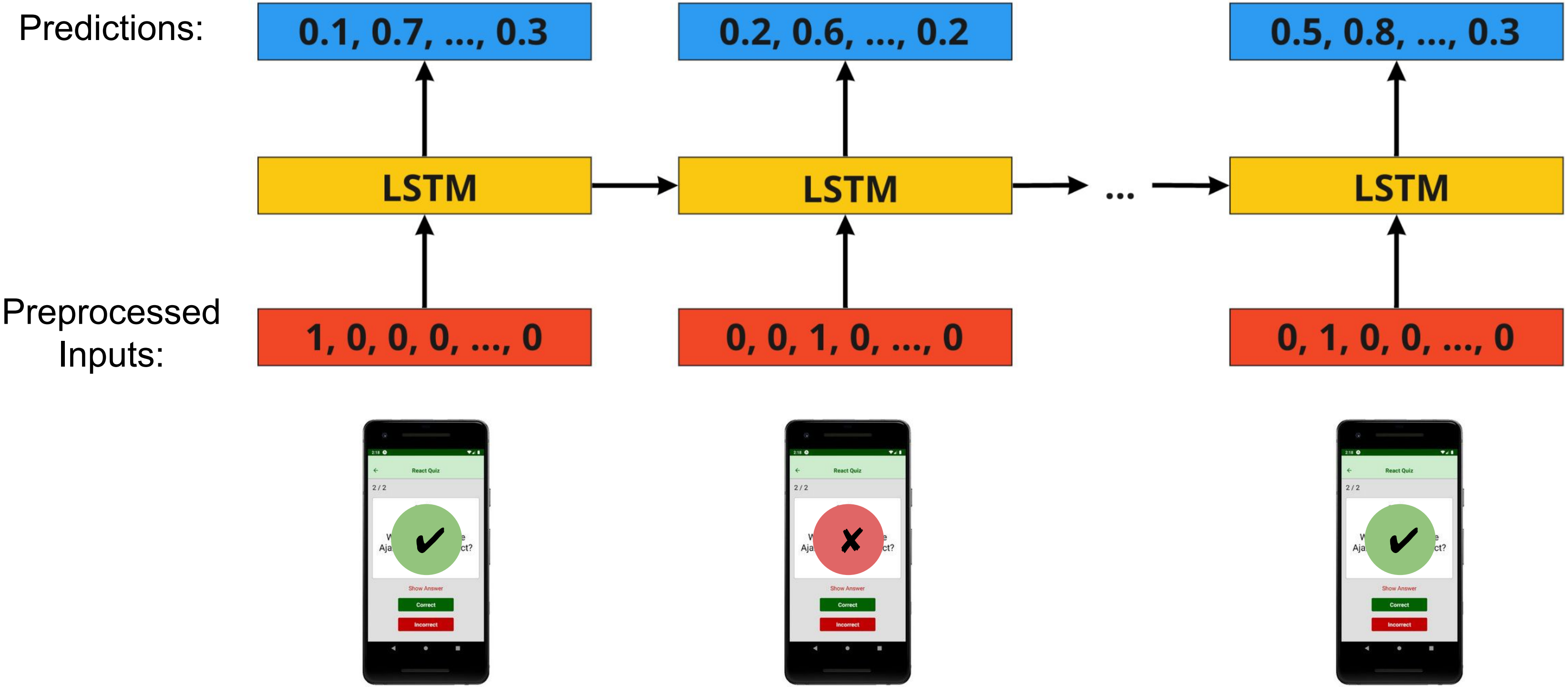
Student answered Skill_B correctly

Outputs: 0.2, 0.6, ..., 0.2

Skill_A Skill_B Skill_Z

Probability student will answer a question with each skill correctly

Deep Knowledge Tracing



Predictions:

0.1, 0.7, ..., 0.3

0.2, 0.6, ..., 0.2

0.5, 0.8, ..., 0.3

LSTM

LSTM

...

LSTM

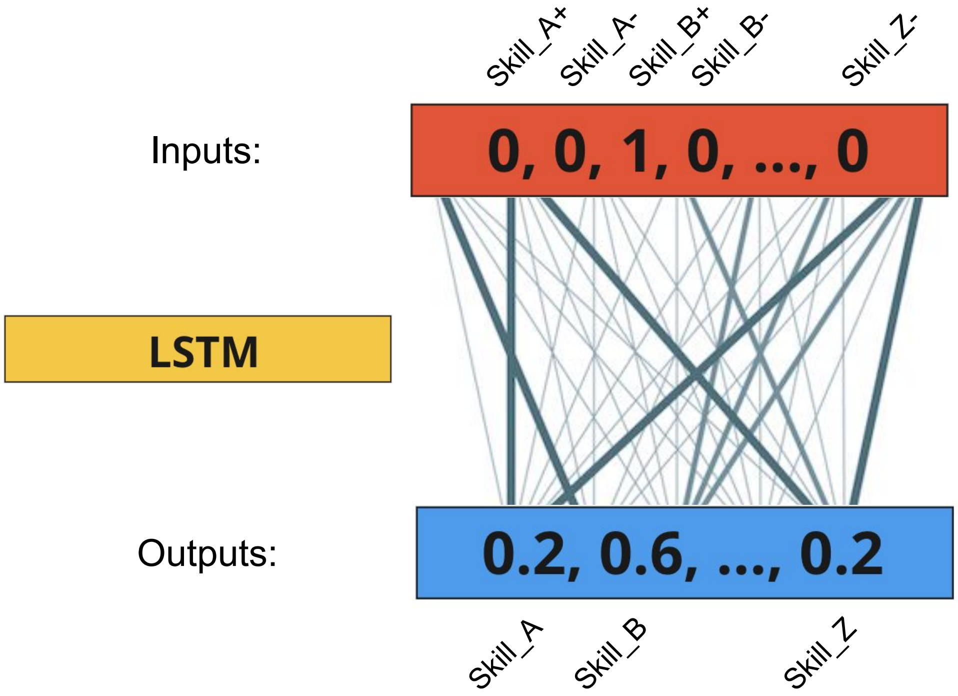
Preprocessed Inputs:

1, 0, 0, 0, ..., 0

0, 0, 1, 0, ..., 0

0, 1, 0, 0, ..., 0

Deep Knowledge Tracing



Inputs: 0, 0, 1, 0, ..., 0

LSTM

Outputs: 0.2, 0.6, ..., 0.2

+ = correct
- = incorrect

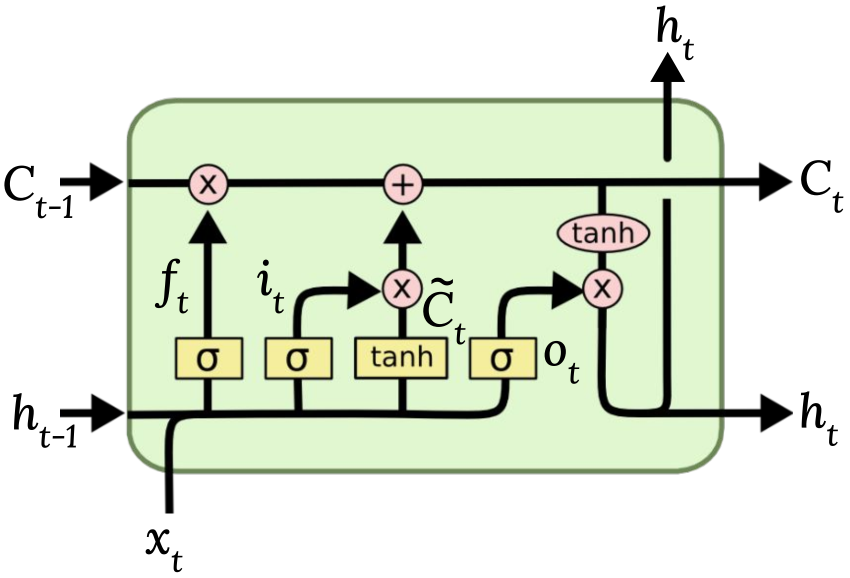
Weights w_i

Probability student will answer a question with each skill correctly

Deep Knowledge Tracing

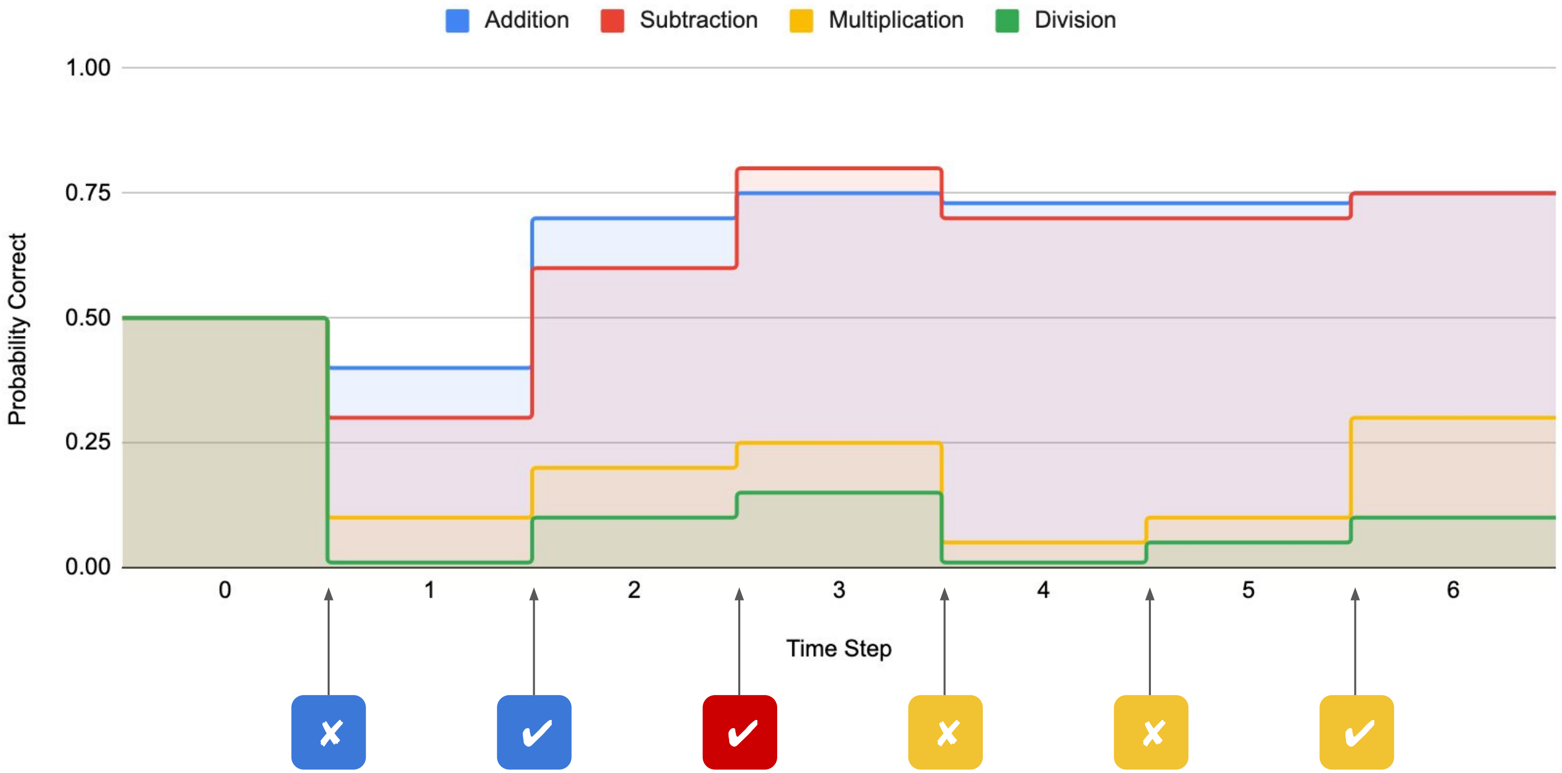
LSTM

Model: A stack of Long Short-Term Memory cells that remember past data. Trained by backpropagation.



$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$	Input Gate
$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$	Forget Gate
$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$	Output Gate
$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$	Cell Update
$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$	Cell State
$h_t = \tanh(C_t) * o_t$	Hidden State

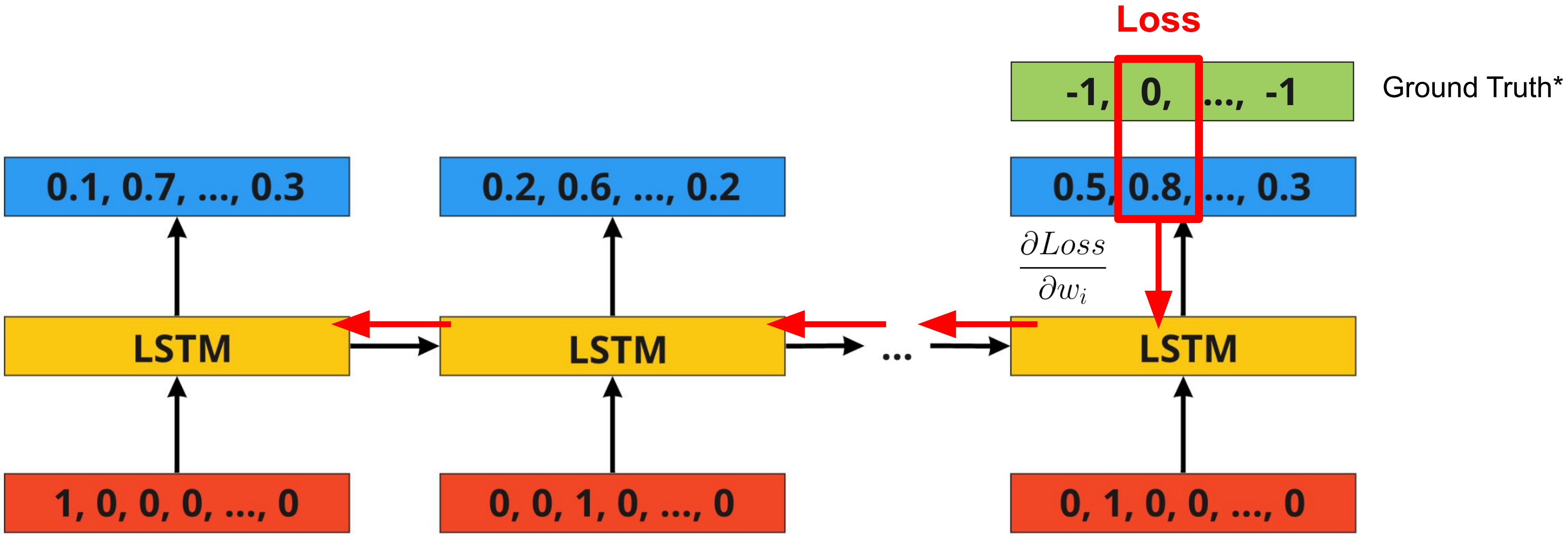
Deep Knowledge Tracing



Deep Knowledge Tracing

Backpropagation

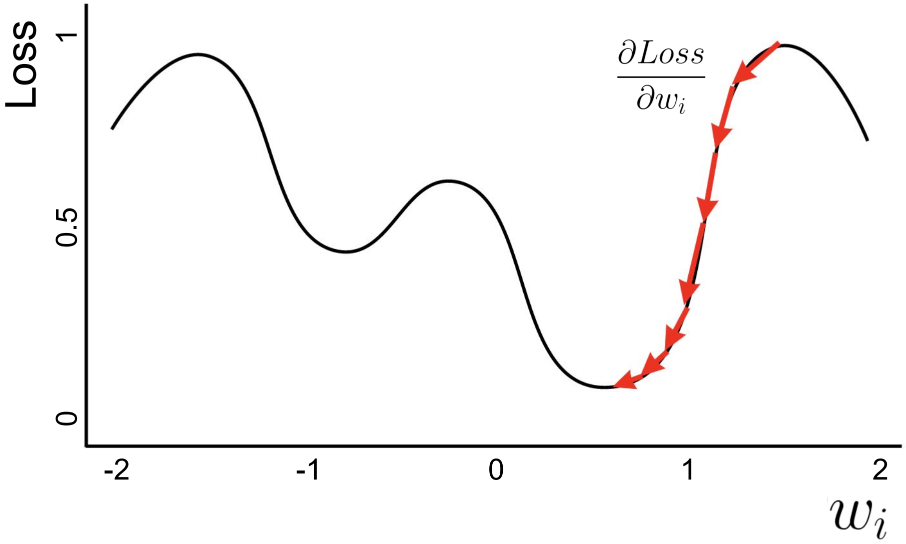
$$\text{Loss} = -\frac{1}{\substack{\text{output}\\\text{size}}} \sum_{i=1}^{\substack{\text{output}\\\text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

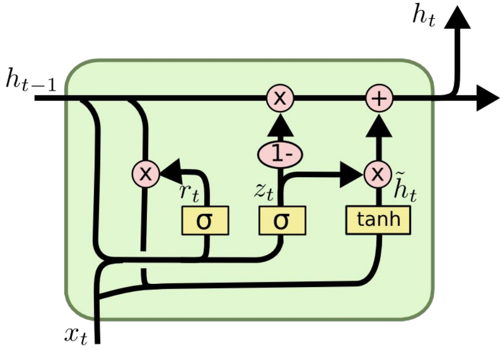


* -1 values are masked (ignored) because that skill was not assessed at that time step

Deep Knowledge Tracing

Backpropagation





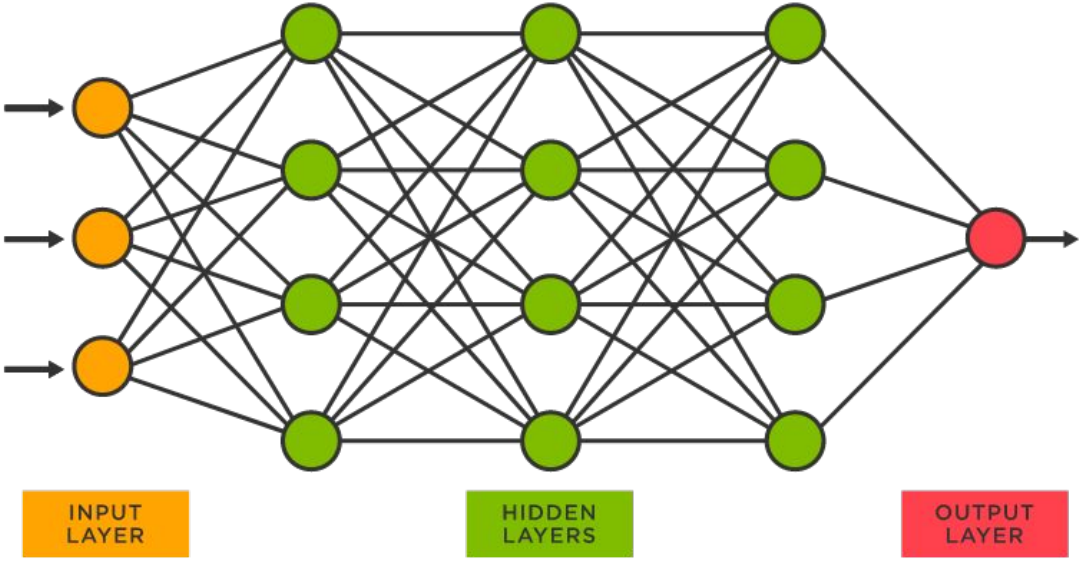
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Modeling - Hyperparameters Defined



Batch Size:

- Number of training examples used with one interaction of the neural network

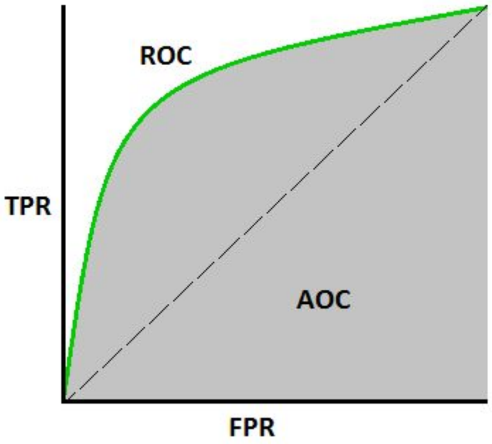
Number of Layers:

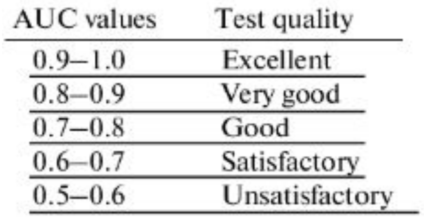
- number of hidden layers
  - Ex. image: 3

Hidden Size:

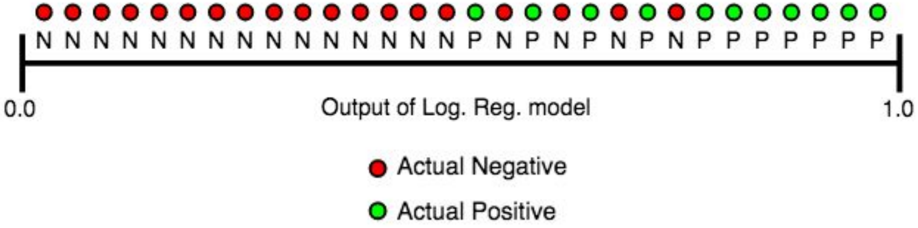
- Number of neurons within each hidden layer
  - Ex. image: 4

Modeling - Evaluation Metric





AUC values	Test quality
0.9–1.0	Excellent
0.8–0.9	Very good
0.7–0.8	Good
0.6–0.7	Satisfactory
0.5–0.6	Unsatisfactory


N N N N N N N N N N N N N N N P N P N P N P N P P P P P P P
0.0
Output of Log. Reg. model
1.0
● Actual Negative
● Actual Positive

TPR = True Positive Rate

$$\text{TPR /Recall / Sensitivity} = \frac{TP}{TP + FN}$$

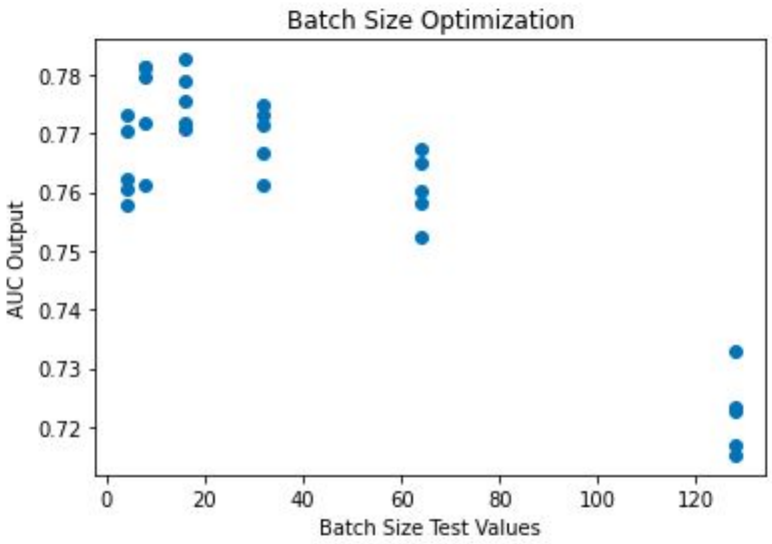
FPR = False Positive Rate

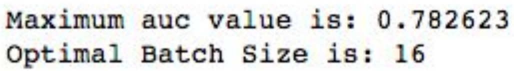
$$\text{Specificity} = \frac{TN}{TN + FP}$$

$$\text{FPR} = 1 - \text{Specificity}$$

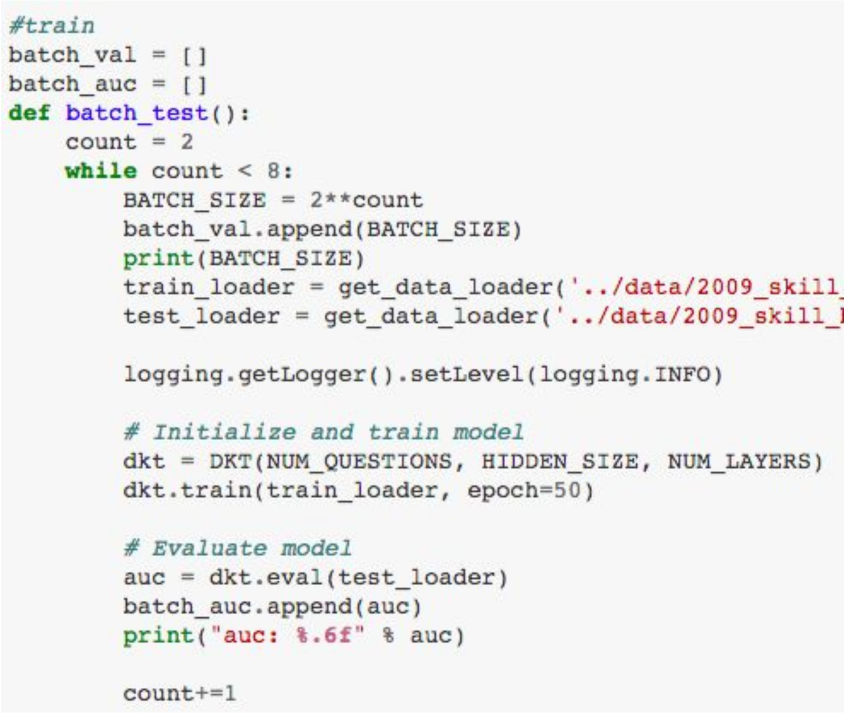
$$= \frac{FP}{TN + FP}$$

Modeling - Results



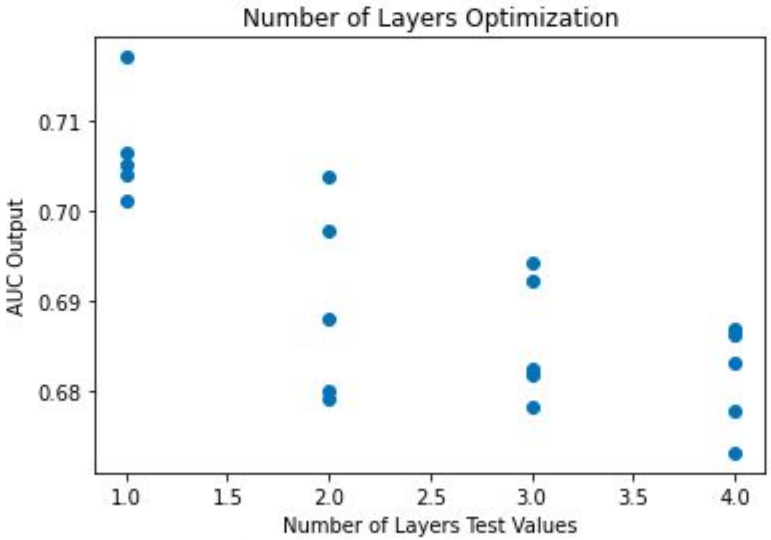


```
Maximum auc value is: 0.782623
Optimal Batch Size is: 16
```

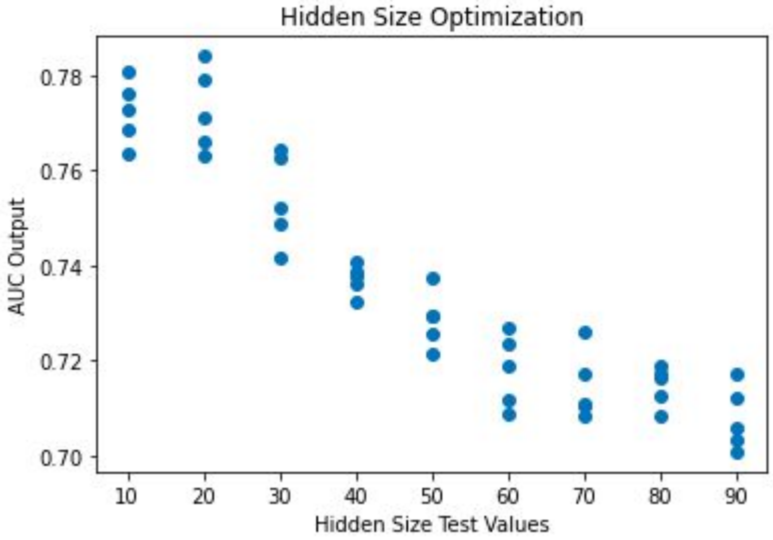


```
#train
batch_val = []
batch_auc = []
def batch_test():
    count = 2
    while count < 8:
        BATCH_SIZE = 2**count
        batch_val.append(BATCH_SIZE)
        print(BATCH_SIZE)
        train_loader = get_data_loader('../data/2009_skill_
        test_loader = get_data_loader('../data/2009_skill_l

        logging.getLogger().setLevel(logging.INFO)

        # Initialize and train model
        dkt = DKT(NUM_QUESTIONS, HIDDEN_SIZE, NUM_LAYERS)
        dkt.train(train_loader, epoch=50)

        # Evaluate model
        auc = dkt.eval(test_loader)
        batch_auc.append(auc)
        print("auc: %.6f" % auc)

        count+=1
```

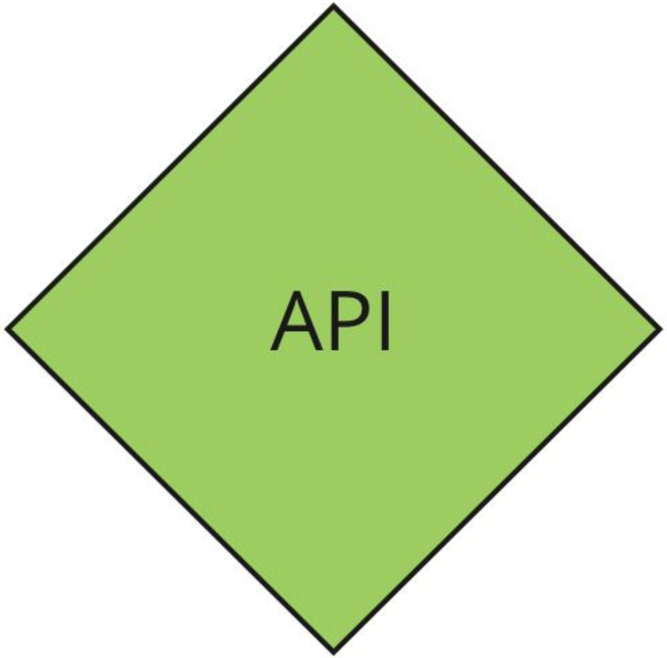
Modeling - Results con.



```
Maximum auc value is: 0.716986
Optimal Number of Layers is: 1
```

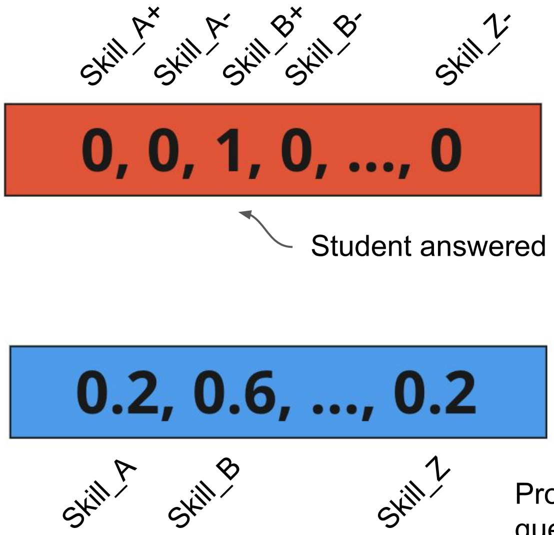


```
Maximum auc value is: 0.784084
Optimal Hidden Size is: 20
```



API

API (Marjorie)



+ = correct
- = incorrect

Inputs: 0, 0, 1, 0, ..., 0 POST /update_database

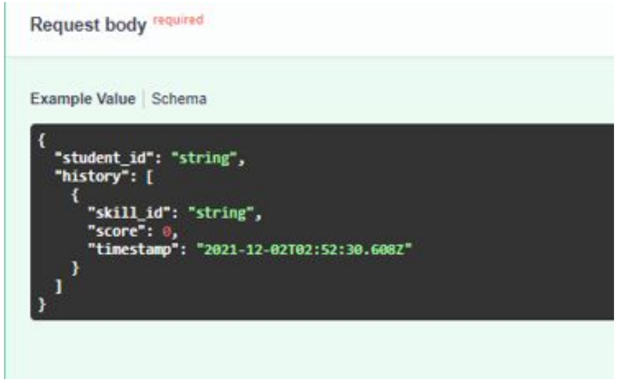
Student answered Skill_B correctly

Outputs: 0.2, 0.6, ..., 0.2 GET /predict

Probability student will answer a question with each skill correctly

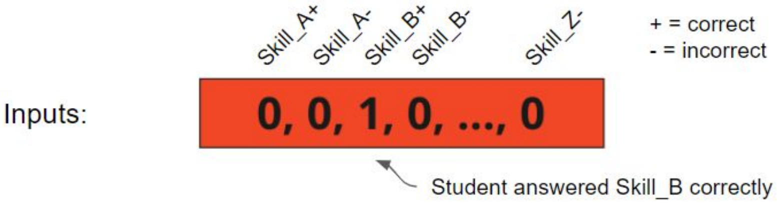
API - POST/update_database

```
“student_id” : “a1b2c3d4”,
“history”: [
    {
        “timestamp”:
“2021-10-03T10:33:54.073001+00:00”,
        “skill_id”: “skill_A”,
        “score”: 0
    },
    {
        “timestamp”:
“2021-10-04T10:33:54.073001+00:00”,
        “skill_id”: “skill_C”,
        “score”: 0
    },
    {
        “timestamp”:
“2021-10-05T10:33:54.073001+00:00”,
        “skill_id”: “skill_B”,
        “score”: 1
    }
  ]
```

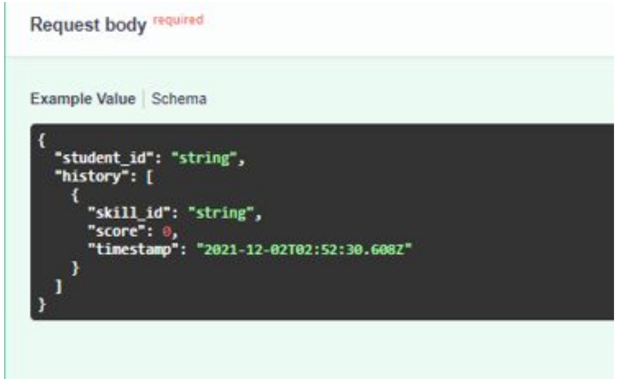


Request body required

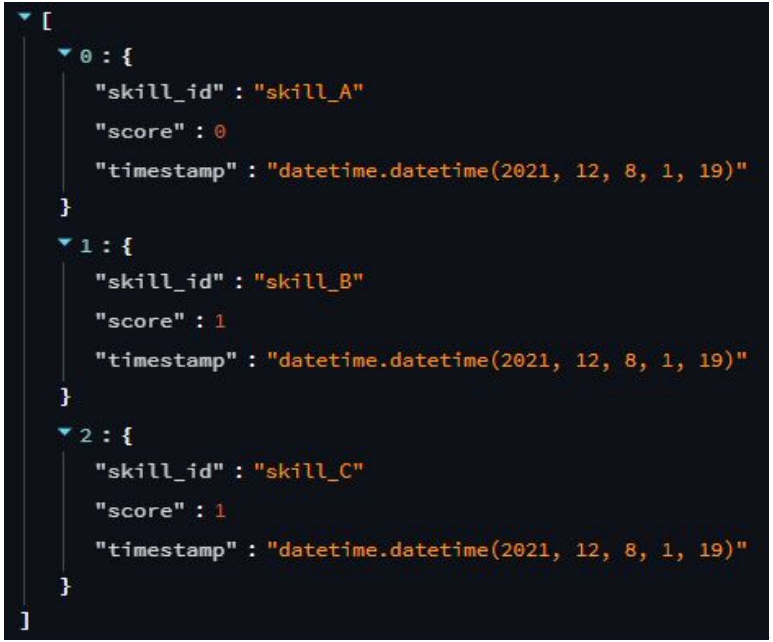
```
{
  "student_id": "string",
  "history": [
    {
      "skill_id": "string",
      "score": 0,
      "timestamp": "2021-12-02T02:52:30.608Z"
    }
  ]
}
```



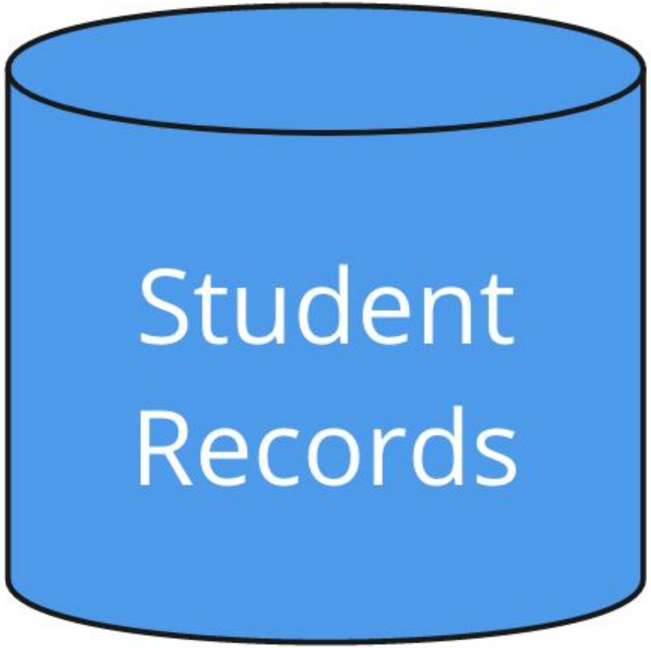
API - POST/update_database

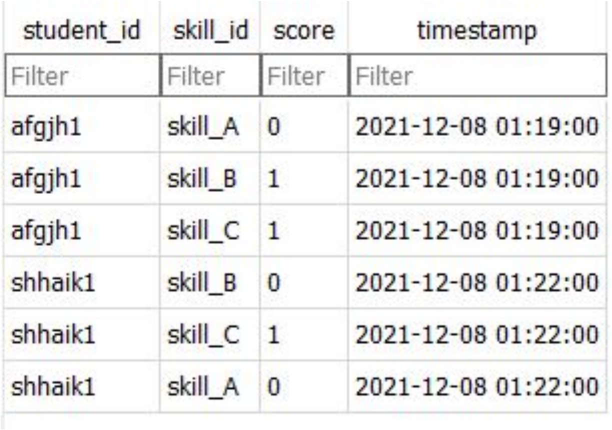


```
{
  "student_id": "string",
  "history": [
    {
      "skill_id": "string",
      "score": 0,
      "timestamp": "2021-12-02T02:52:30.608Z"
    }
  ]
}
```



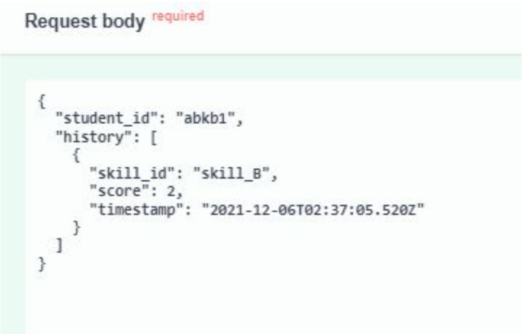
```
[
  0 : {
    "skill_id" : "skill_A"
    "score" : 0
    "timestamp" : "datetime.datetime(2021, 12, 8, 1, 19)"
  }
  1 : {
    "skill_id" : "skill_B"
    "score" : 1
    "timestamp" : "datetime.datetime(2021, 12, 8, 1, 19)"
  }
  2 : {
    "skill_id" : "skill_C"
    "score" : 1
    "timestamp" : "datetime.datetime(2021, 12, 8, 1, 19)"
  }
]
```





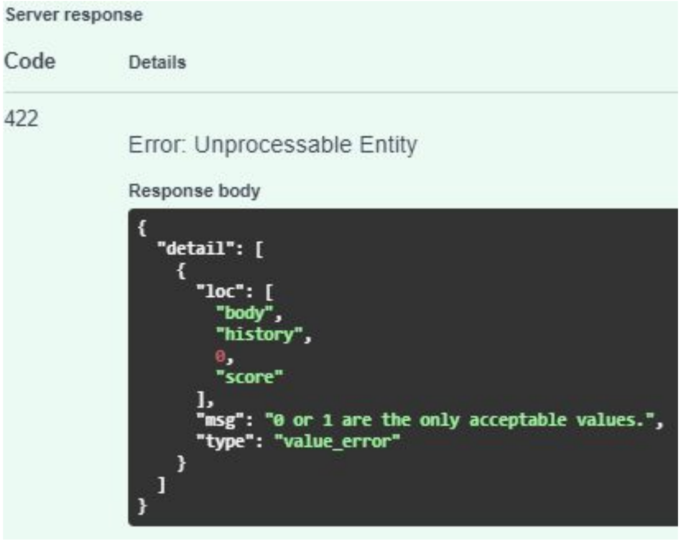
student_id	skill_id	score	timestamp
Filter	Filter	Filter	Filter
afgjh1	skill_A	0	2021-12-08 01:19:00
afgjh1	skill_B	1	2021-12-08 01:19:00
afgjh1	skill_C	1	2021-12-08 01:19:00
shhaik1	skill_B	0	2021-12-08 01:22:00
shhaik1	skill_C	1	2021-12-08 01:22:00
shhaik1	skill_A	0	2021-12-08 01:22:00

API - Validation of Values



Request body

```
{
  "student_id": "abkb1",
  "history": [
    {
      "skill_id": "skill_B",
      "score": 2,
      "timestamp": "2021-12-06T02:37:05.520Z"
    }
  ]
}
```



Server response

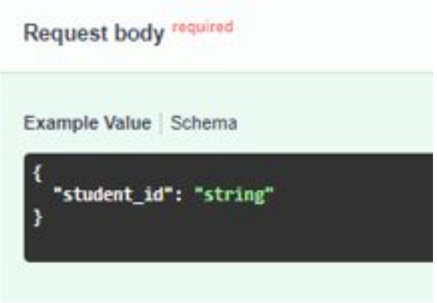
Code	Details
422	Error: Unprocessable Entity

Response body

```
{
  "detail": [
    {
      "loc": [
        "body",
        "history",
        0,
        "score"
      ],
      "msg": "0 or 1 are the only acceptable values.",
      "type": "value_error"
    }
  ]
}
```

API - GET/predict

```
“student_id”: “a1b2c3d4”,

“predictions”: {

    “skill_A”: 0.9,

    “skill_B”: 0.1,

    “skill_C”: 0.54,

}
```

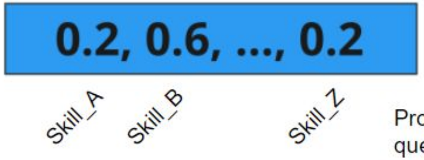


Request body required

Example Value | Schema

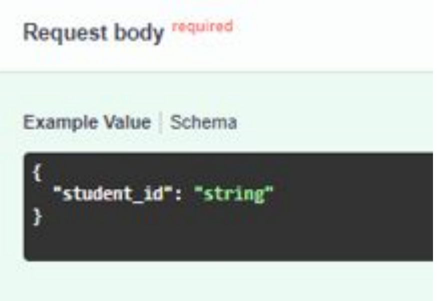
```
{
  "student_id": "string"
}
```

Outputs:

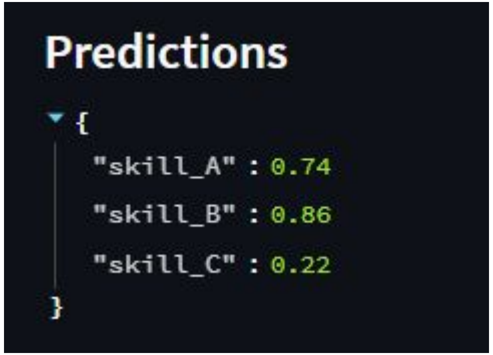


Probability student will answer a question with each skill correctly

API - GET/predict



```
{
  "student_id": "string"
}
```


Predictions

```
▾ {
    "skill_A" : 0.74
    "skill_B" : 0.86
    "skill_C" : 0.22
  }
```

Future Work

- Train model on actual ITS data
- Integrate preprocessing and model with API
- Deploy the API in Production
- Evaluate more recent modeling methods (SAKT, AKT, etc)
- Use reinforcement learning for recommendations