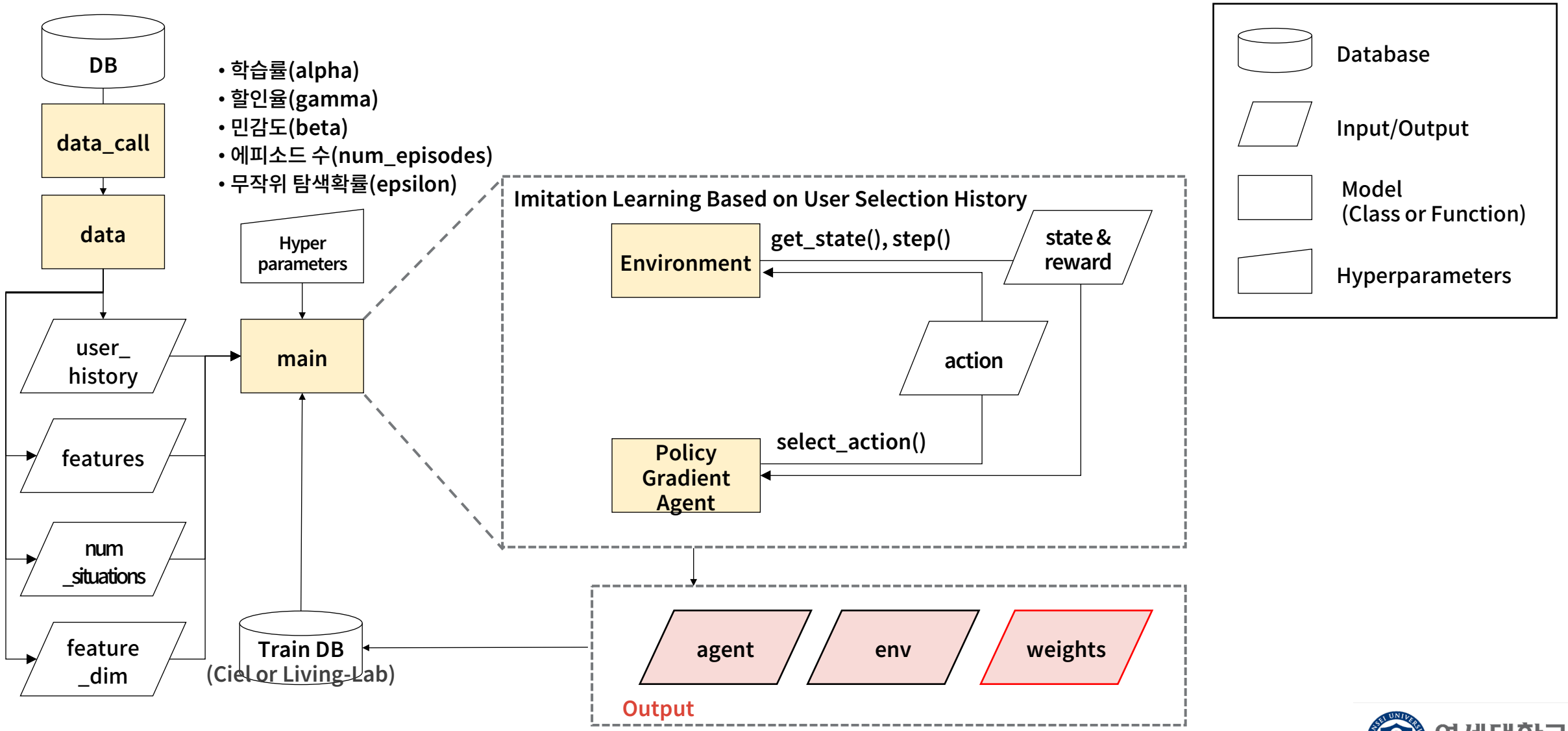
자율주행 수요응답 대중교통 이용자 이용패턴 및 선호경로 분석 (English)

연세대학교
YONSEI UNIVERSITY

Environments

- **Python 3.11.1**
- **Package List**
  - pandas (version 2.2.2)
  - numpy (version 2.0.0)
  - random(Python built-in packages)
  - pickle (Python built-in packages)

Framework


DB
data_call
data
• 학습률(alpha)
• 할인율(gamma)
• 민감도(beta)
• 에피소드 수(num_episodes)
• 무작위 탐색확률(epsilon)
Hyper parameters
user_ history
features
num _situations
feature _dim
main
Train DB
(Ciel or Living-Lab)
Imitation Learning Based on User Selection History
Environment
get_state(), step()
state & reward
action
Policy Gradient Agent
select_action()
agent
env
weights
Output
Database
Input/Output
Model
(Class or Function)
Hyperparameters

data_call & data

- **A module that retrieves historical data from the database based on user ID (individual tests not conducted)**
- **Data is retrieved by entering the user ID (id) into the user history DB (user_history_df) and the post vehicle dispatch information DB (features_df)**
  - In the initial study, a survey was conducted on 500 living lab participants, and based on this, the user_history_df and features_df were constructed for development (For further explanation of survey, please refer to attachment file)
  - Strictly speaking, 'Linc' and 'License' denote attributes of the user rather than the vehicle. However, since this model is constructed based on survey data, which allows for the incorporation of user attributes into the learning process, they have been included. Considering the feasibility of utilizing user information and its effectiveness in learning, these variables are scheduled for removal in a future enhancement of the dispatch algorithm.

data_call

구분	자료명	자료형태	설명
Input	path	[string]	Domain of DB
Output	user_history_df(DB)	[DataFrame] (예시자료: user_history.csv)	user ID(id), service record sequence(situation), alternative chosen by user(choice)
	features_df(DB)	[DataFrame] (예시자료: features.csv)	user ID(id), service record sequence(situation), available alternative(alternative; 2 = 거절을 의미), alternative features(access, wait, ivt, egress, constant, Linc*, license*; Rejected alternatives have all information set to 0, except for the constant)

data_call & data

- **A module that retrieves historical data from the database based on user ID (individual tests not conducted)**
- **Data is retrieved by entering the user ID (id) into the user history DB (user_history_df) and the post vehicle dispatch information DB (features_df)**
  - In the initial study, a survey was conducted on 500 living lab participants, and based on this, the user_history_df and features_df were constructed for development (For further explanation of survey, please refer to attachment file)

data

구분	자료명	자료형태	설명
Input	id	[Int] (ID)	user ID – Subject to change depending on the DB environment
	path	[string]	Domain of DB
	user	[Dictionary]	{user id :{user_history_df : [DataFrame], features_df : [DataFrame]}}
Output	user_history	[List]	List of alternative chosen by user (Example : [1, 2, 0, 1, 0, 1, 2])
	features	[Array]	Alternative features by service record sequence
	num_situations	[Int]	Number of recored serquence
	feature_dim	[Int]	Number of features columns(Referred to as the number of alternative features)

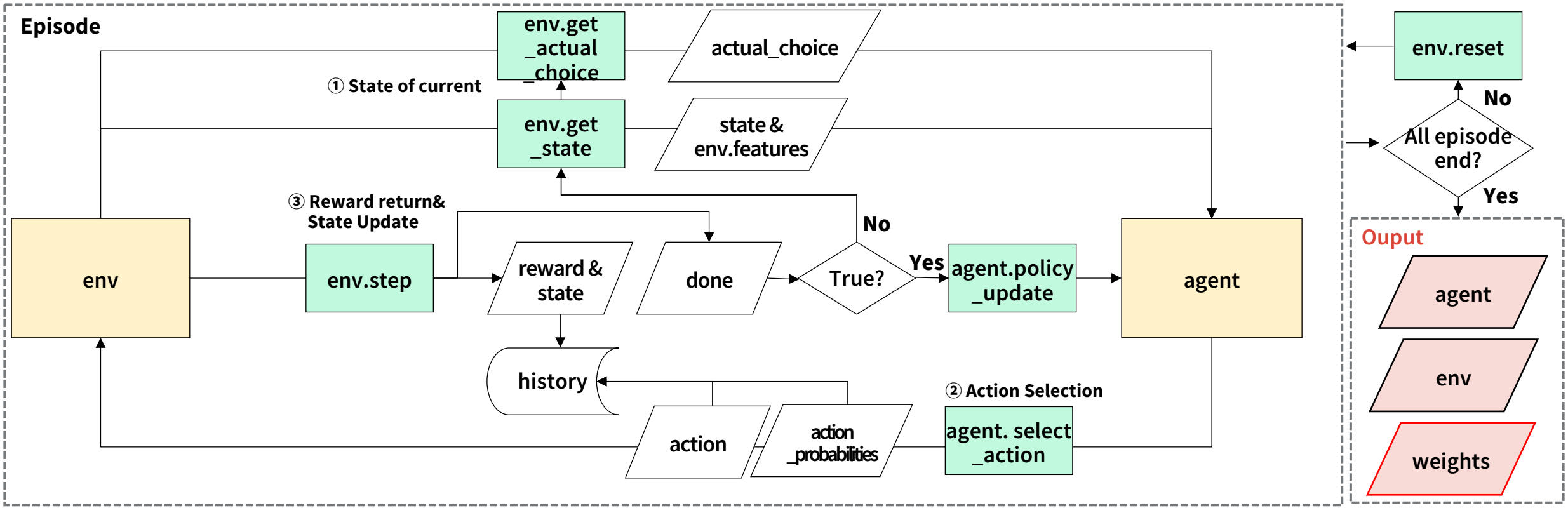
main[1/2]

- **Function to perform reinforcement learning based on the retrieved data**
  - For each user ID, an Environment (env) and a PolicyGradientAgent (agent) object are created
  - Training is conducted using arbitrarily selected hyperparameters, and the total rewards for each episode are recorded to assess the convergence of the reinforcement learning process

구분	자료명	자료형태	설명
Input	user_history	[List]	
	features	[Array]	
	feature_dim	[Int]	
	learned_weights	[Array] 1Xfeature_dim	previously learned Agent's policy function weights (if the previously trained weight exist, or not None)
Output	agent(trained_agent)	[Instance]	Trained Agent
	env(trained_env)	[Instance]	Learning Environment(MDP)
	agent.weights(weights)	[Array] 1Xfeature_dim	Trained Weights per user ID

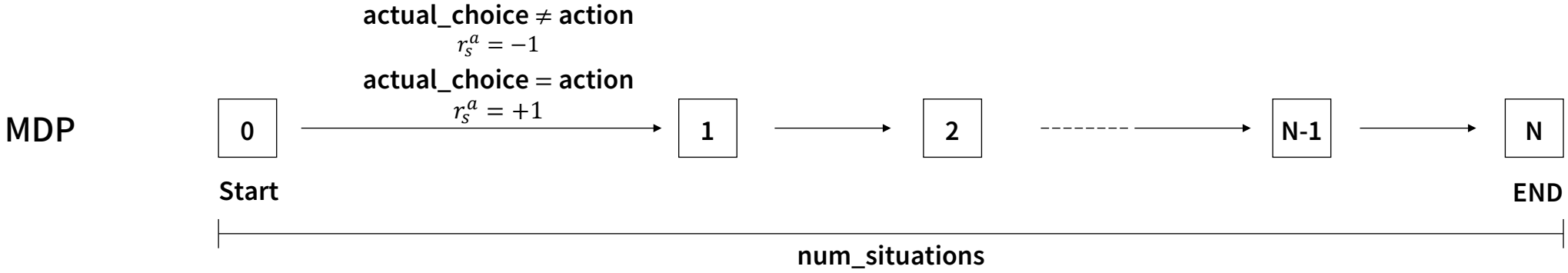
main[2/2]

- **Until the episode termination point is reached, the local instances of env and agent within the function repeat following structure;**
  - State of current → Action Selection → Reward return and State Update → State of current



Environment

- **Markov Decision Process (MDP) and Reward Environment**
- **Provides the state environment (MDP) for learning, and when the PolicyGradientAgent takes an action, a corresponding reward is granted**
  - if the agent correctly matches the actual historical record for the given state, it receives +1; if not -1 is applied
  - The episode continues until the final dispatch record is reached (transition probability$P_{ss'}^{a}$ is equal to 1)



구분	자료명	자료형태	설명
Input	user_history	[List]	
	features	[Array]	
Output	actual_choice	[List]	Actual choice per sequence and user id
	state	[Int]	Agent state on MDP
	reward	[Int]	Reward by action of Agent
	done	[Int]	Indicator of Episode termination(If terminated, last number of state given)

PolicyGradientAgent

- **The policy function that determines the agent's action is a logit-based softmax function**
- **Based on Policy gradient theorem, the policy parameters (weights) are updated to improve the policy, encouraging the agent to choose actions with a higher probability of obtaining greater expected rewards**
- **Formula)**
  - Policy function

$$\pi_\theta(a_t \mid s_t) = \frac{\exp(\beta \cdot Q_\theta(s_t, a_t))}{\sum_{a'} \exp(\beta \cdot Q_\theta(s_t, a_t'))}$$

$$Q_\theta(s_t, a_t) = \phi(s_t, a_t)^T \theta$$

- β: temperature parameter
- Q_θ : value function parameterized by weight θ
- $\phi(s, a)$: information (features) of the alternative corresponding to action a in state s

  - Policy improvement

$$\theta \leftarrow \theta + \alpha G_t \nabla_\theta \ln \pi_\theta(a_t \mid s_t)$$

$$\nabla_\theta \ln \pi_\theta(a_t \mid s_t) = \phi(s_t, a_t) - \mathbb{E}_{a_t' \sim \pi_\theta}[\phi(s_t, a_t')]$$

- G_t: Expected Return

구분	자료명	자료형태	설명
Input	feature_dim	[Int]	
	learned_weights	[Array] 1Xfeature_dim	
	history	[List]	action, state, reward, action probabilities at time t
Output	action	[Int]	Chosen alternative by agent on given state
	action probabilities	[List]	Probability of action by agent on given state

user

- **Dictionary object including historical data ('user_history_df', 'features_df'), environment (env), and trained agent (agent) instances, based on user ID**
- **Included**
  - user_history_df
  - features_df
  - env
  - agent
  - accuracy : Accuracy based on training data (It shows low accuracy, and train and test sets are not separated due to limited survey data; future updates are planned)
  - weights

weights.csv

- **Feature-specific weights of the action-value function for each user ID**
- **In this training, the agent selects the alternative with the highest action-value function, which is analogous to a discrete choice logit model based on utility maximization**
  - The higher the weighted sum score of an alternative's features, the better the alternative is considered
  - Due to limitations in the estimation data, some variables exhibit positive weights(which will be addressed in future living lab implementations)
  - In order to incorporate the trained weight data into the simulation, it is necessary to arbitrarily assign virtual IDs to the real-time demand input data and match the corresponding weights
  - Although the Linc and License features were used during training, they are difficult to apply in real-world scenarios and have been removed from the given weight.csv file.

구분	컬럼	설명
user ID	Id	user ID [1~500; int]
Weight	access	Weight of walking access time from pickup location [min]
	wait	Weight of waiting time at pickup location[min]
	ivt	Weight of in-vehicle time[min]
	egress	Weight of egress time(Walking time from drop-off location to final destination) [min]

Limitation and Future Update

- **Due to limitations in the estimation data, some variables exhibit positive weights (which will be addressed in future living lab implementations)**
  - This survey data was collected from 500 living lab residents regarding six hypothetical scenarios, and in order to improve the accuracy of the learning outcomes, it is necessary to obtain dispatch history data
  - Additionally, this learning result is intended to imitate user preferences and acceptance behaviors based on accumulated usage experience, and therefore does not intentionally guarantee state independence
  - In other words, when applying this model to survey data that lack cumulative usage experience, there are inherent limitations in the estimation