



DATA SCIENCE &
ARTIFICIAL INTELLIGENCE

Music Recommendation System using Q-Learning

Roshanak Behrouz



Problem statement

System Overview

Develop a music recommendation system that learns to suggest music genres a user will enjoy, by leveraging reinforcement learning, specifically Q-Learning.



Maximize User Satisfaction

The system aims to maximize user satisfaction by recommending relevant genres that align with user preferences and listening history.



Minimize Negative Experiences

The system aims to minimize negative user experiences by avoiding unwanted recommendations through intelligent learning.

Translate the problem into a RL framework



Agent

MusicRecommendationAgent

The intelligent system that learns to make music recommendations through interaction with the environment.



Environment

Simulated user feedback system that provides rewards based on recommendation quality and user preferences.

Reinforcement Learning Components



State

Current music genre representing the user's present listening context or preference.



Action

Recommending a specific music genre from the available genre options to the user.

Reinforcement Learning Components



Reward

Generic User: Numerical feedback from simulated user indicating recommendation quality based on user preferences and satisfaction.

User Profile: The rewards can be customized based on a profile, allowing for different reward sensitivities for each genre, making the simulation more realistic and personalized.



Reinforcement Learning Components



Policy

Epsilon-greedy during training for exploration/exploitation balance, then **greedy** based on learned Q-table for optimal decisions.



Q-Value/Q-Table

Stores expected future rewards for all state-action pairs, enabling the agent to make informed decisions based on learned experience.

Why Q-Learning?

Model-Free Learning

Q-Learning does not require a pre-existing model of the environment. It learns directly from interactions, which is beneficial here since accurately modeling complex user behavior can be challenging.

Off-Policy Learning

Q-Learning is an off-policy algorithm, meaning it can learn the optimal policy while following a different exploration policy. This allows the agent to explore different recommendations to discover better strategies without necessarily having to act optimally during training.

Discrete State and Action Space

The music recommendation problem has a discrete and relatively small set of states (genres) and actions (recommending genres). Q-Learning performs well in such environments, where a Q-table can effectively map state-action pairs to values.



How is Q-Learning Implemented?

Initialization and Q-Table

- **Agent initialized**
- **Q-Table is created**
- **Dimensions:** (#genres × #genres)
- **Rows:** Current state
- **Columns:** Actions
- **Cells:** Each cell $Q(s,a)$ in the table represents the estimated **expected future cumulative reward** the agent can get if it takes action A while in state S.
- **Initial Values:** All cells start at 0.0 and are updated through reinforcement learning based on user feedback and rewards.

From / To Recommend	Pop	Rock	Jazz	Classical	HipHop
Pop	0.0	0.0	0.0	0.0	0.0
Rock	0.0	0.0	0.0	0.0	0.0
Jazz	0.0	0.0	0.0	0.0	0.0
Classical	0.0	0.0	0.0	0.0	0.0
HipHop	0.0	0.0	0.0	0.0	0.0

Updating knowledge (Q-Value Update Rule)

$$Q(s, a) \leftarrow Q(s, a) + \alpha [R + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

Symbol	Meaning
s	Current state
a	Action taken from state s
r	Reward received after taking action a
s'	Next state after taking action a
a'	All possible actions in next state s'
α	Learning rate (controls how much we update Q)
γ	Discount factor (importance of future rewards)
$Q(s, a)$	Current estimate of Q-value
$\max_{a'} Q(s', a')$	Estimated maximum future reward from next state s'

Example

Step 1: Old Q-Value

```
old_Q('Rock', 'Pop') = Q[1,0] =  
1.0
```

Step 2: Max Future Value

```
max_a' Q('Pop', a') = max([0.2,  
0.0, 0.0, 0.0, 0.0]) = 0.2
```

Step 3: TD Target

```
TD Target = R +  $\gamma$  × max_future  
= 10 + (0.9 × 0.2) = 10.18
```

Step 4: TD Error

```
TD Error = 10.18 - 1.0 = 9.18  
(The "surprise" factor)
```

Step 5: Update Term

```
Update =  $\alpha$  × TD Error  
= 0.1 × 9.18 = 0.918
```

Step 6: New Q-Value

```
new_Q = old_Q + Update  
= 1.0 + 0.918 = 1.918
```

Final Result

Q('Rock', 'Pop'): 1.0 → 1.918

The agent learned that recommending Pop after Rock leads to better outcomes than previously expected!

Action Selection (Epsilon-Greedy Policy)



Exploration

ϵ

With a probability of **epsilon**, the agent explores by picking a random genre. This helps discover new, potentially better, recommendation strategies.



Exploitation

$1 - \epsilon$

With a probability of **1 - epsilon**, the agent exploits its current knowledge by choosing the genre that has the highest Q-value for the current state.

Adaptive Learning Behavior

The epsilon value gradually decreases over time, causing the agent to explore less and exploit more as it gains experience and confidence in its learned Q-values.

$$\epsilon_{t+1} = \epsilon_t \times \text{decay_rate}$$

Reward

Two Reward Calculation Approaches

The system implements two distinct reward calculation methods: Generic User rewards based on pure probability, and Specific User Profile rewards based on personalized preferences.



Generic User

The reward calculation is entirely probabilistic and is **not tied to the specific genre recommended**. The agent learns a general policy that tries to maximize rewards from a user whose reactions are largely unpredictable and not based on specific genre preferences.

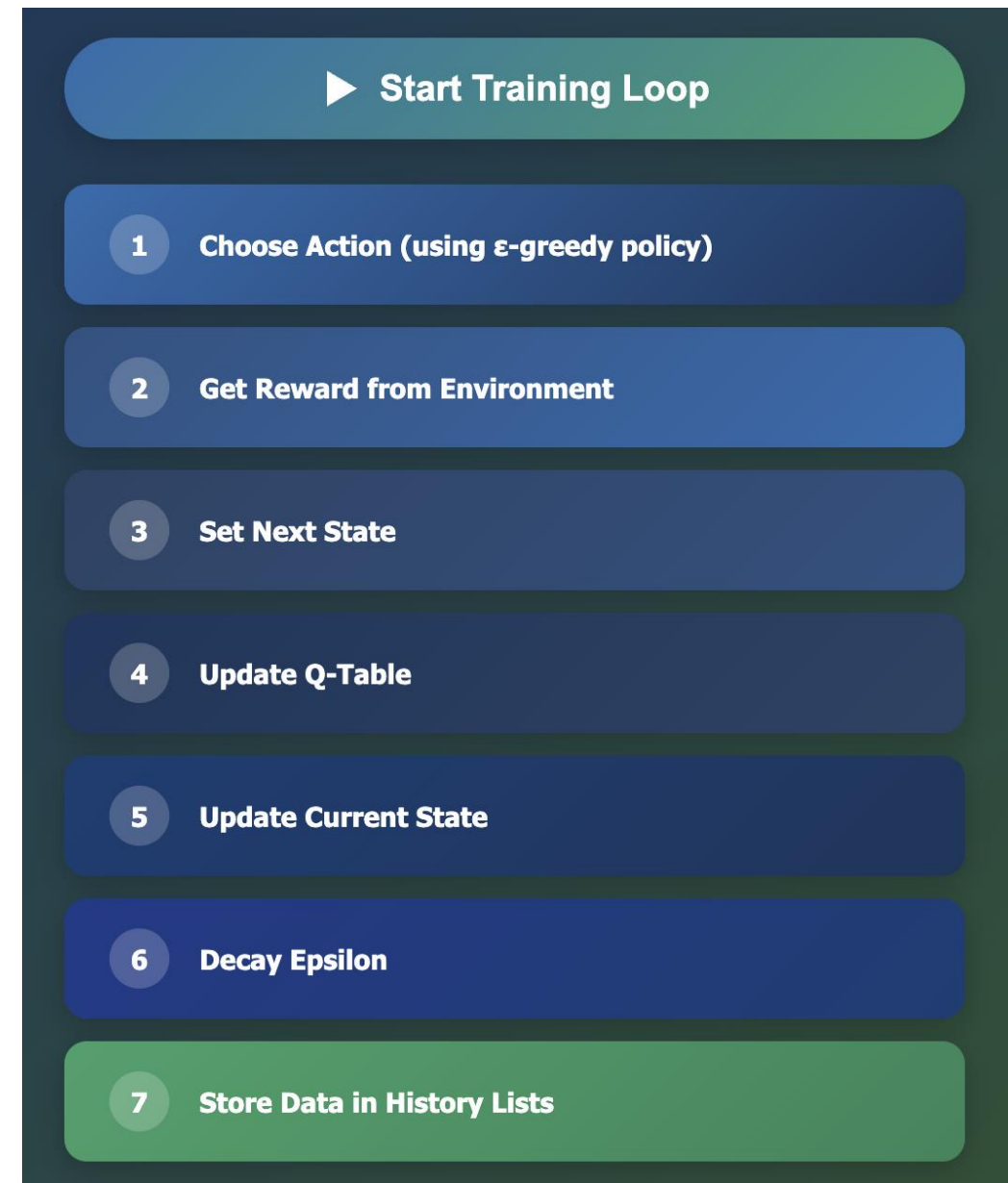


Specific User Profiles

The reward calculation becomes **personalized based on that user's specific preferences** for the recommended genre. The system retrieves user preferences and calculates probabilistic rewards accordingly.


Training Loop

- The training loop runs for a specified number of episodes to train the Q-learning agent.
- In each episode, the agent selects a genre based on the current state using an ϵ -greedy strategy.
- After taking an action, it receives a reward.
- The state is then updated, and the exploration rate is decayed over time to encourage more exploitation.
- Rewards, actions, and epsilon values are logged for evaluation.



 Next Iteration

Policy Formation

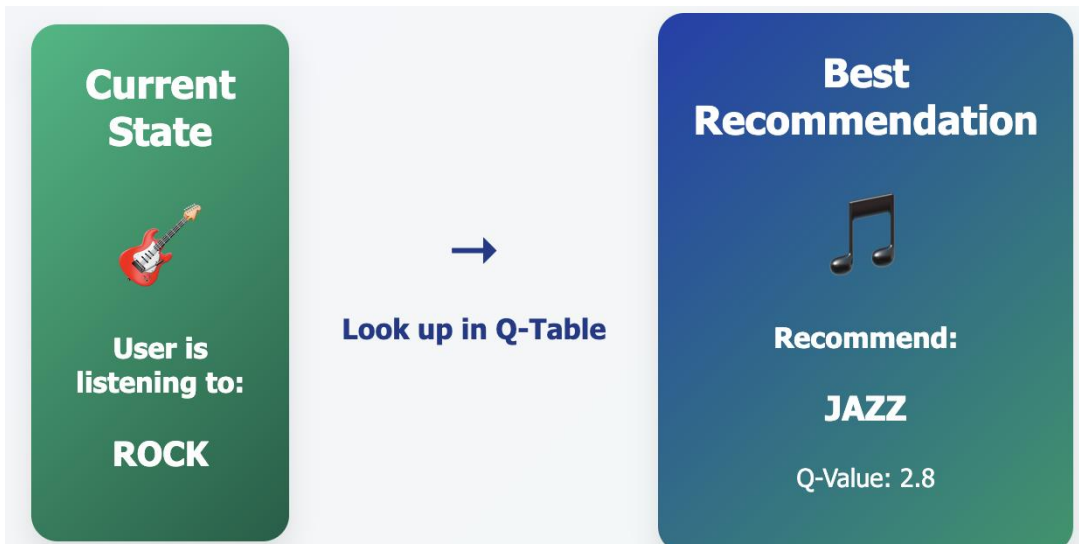
 Policy Formation	
Action	Agent selects a genre to recommend using epsilon-greedy strategy
Reward	User feedback received after recommendation (positive/negative)
Q-table Update	Q-values adjusted based on received reward and learning rate
Episodes	Many cycles of action-reward-update create learning experience
Optimal Policy	Best genre recommendations implicitly stored in final Q-table

Through this continuous cycle of action, reward, and Q-table updates over many episodes, the agent gradually learns an optimal "policy." This policy is implicitly stored in the Q-table, indicating the best genre to recommend for any given current genre state.

Making Recommendations

Once the training is complete and the Q-table has converged (i.e., the Q-values have stabilized), the agent has learned its "optimal policy":

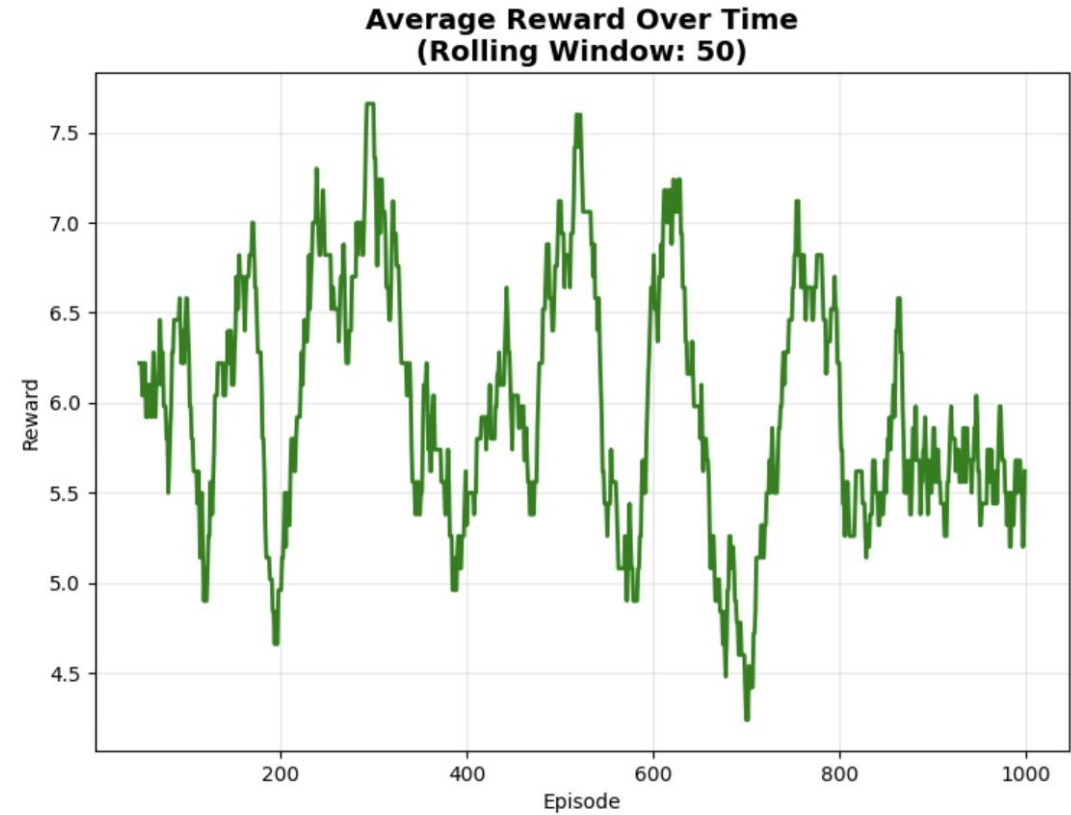
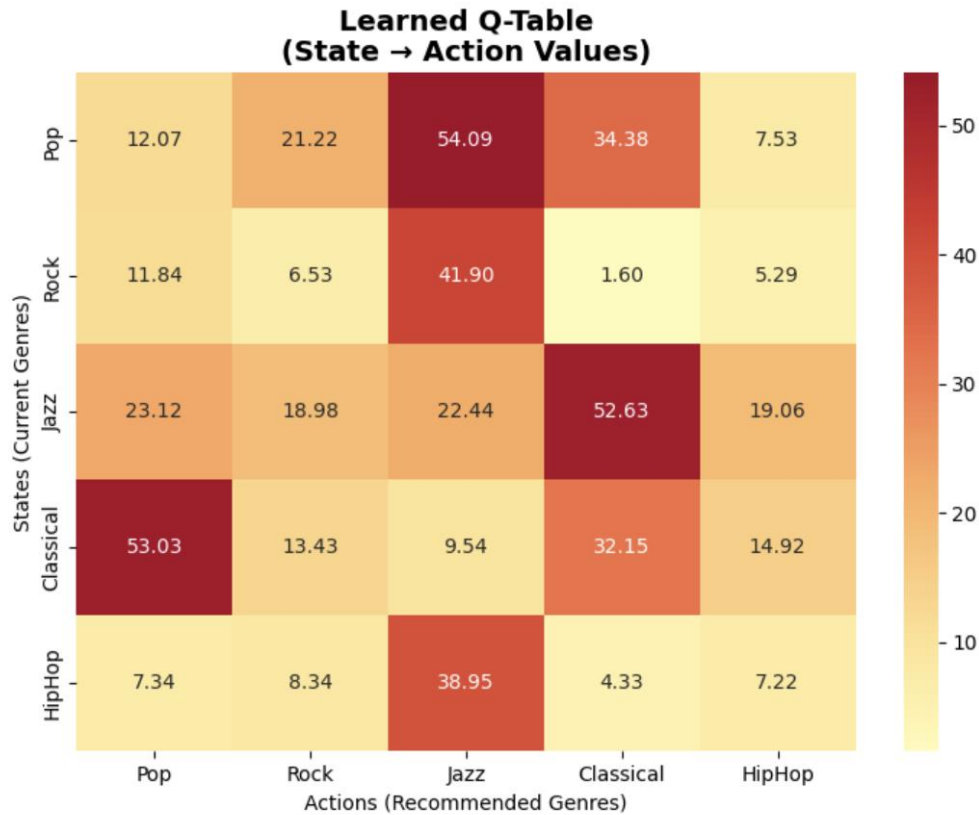
- To make a recommendation for a given "current genre" (state), the agent simply looks at the row in the Q-table corresponding to that genre.
- It then selects the action (recommended genre) that has the **highest Q-value** in that row. This is the action the agent has learned will lead to the greatest expected cumulative future reward.



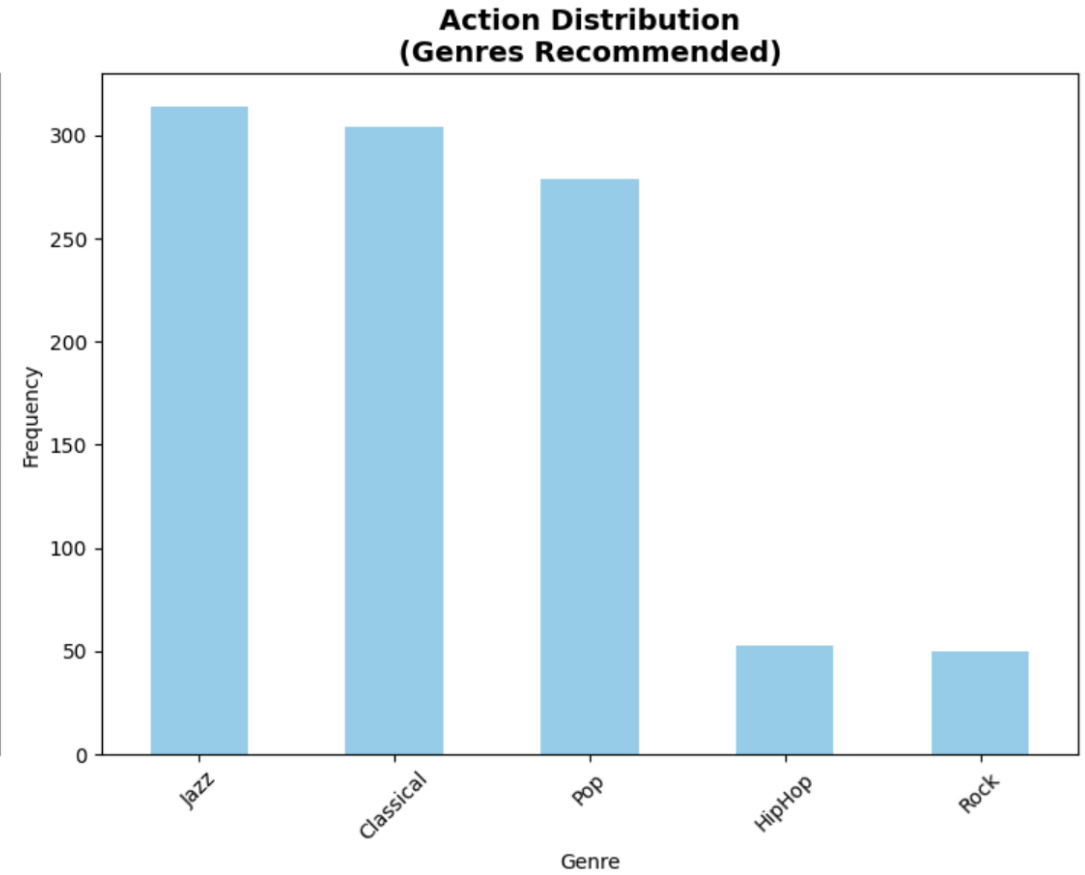
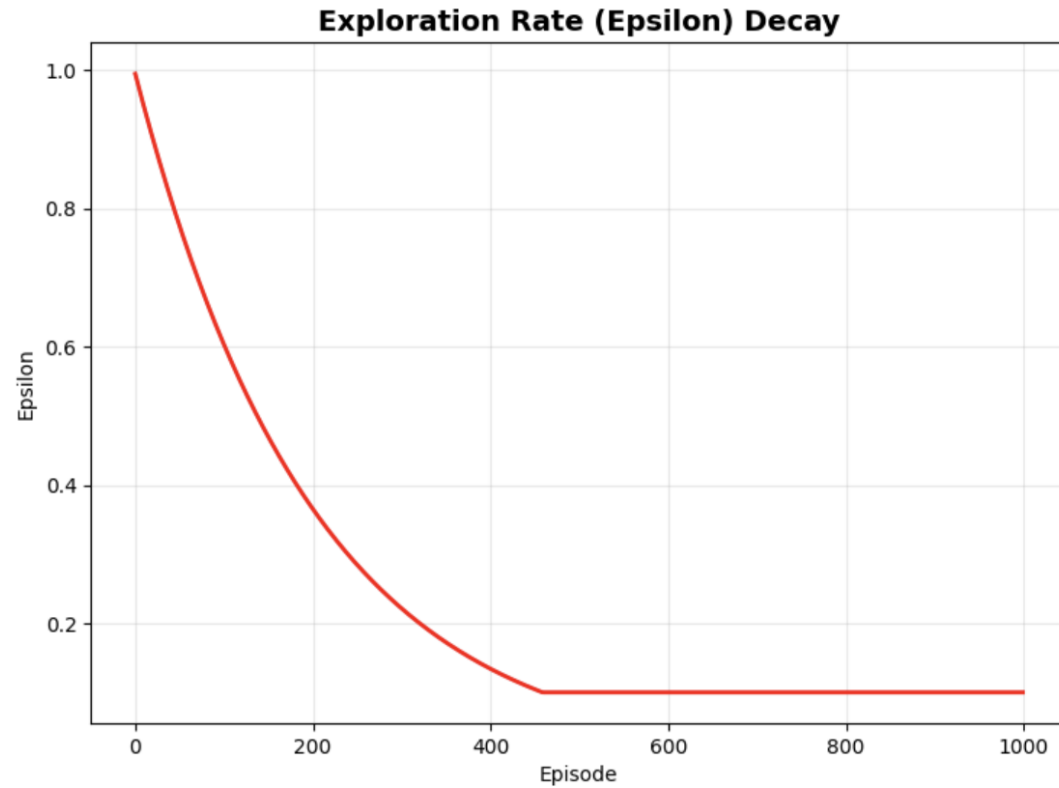
Q-Table: Music Recommendation System

Current Genre / Recommend	Pop	Rock	Jazz	Classical	HipHop
Pop	1.2	0.8	1.5	0.9	1.1
Rock	1.9	0.3	2.8	1.4	2.1
Jazz	1.6	2.2	0.5	2.5	0.7
Classical	0.8	1.1	2.3	0.4	0.6
HipHop	2.0	1.7	1.3	0.9	0.8

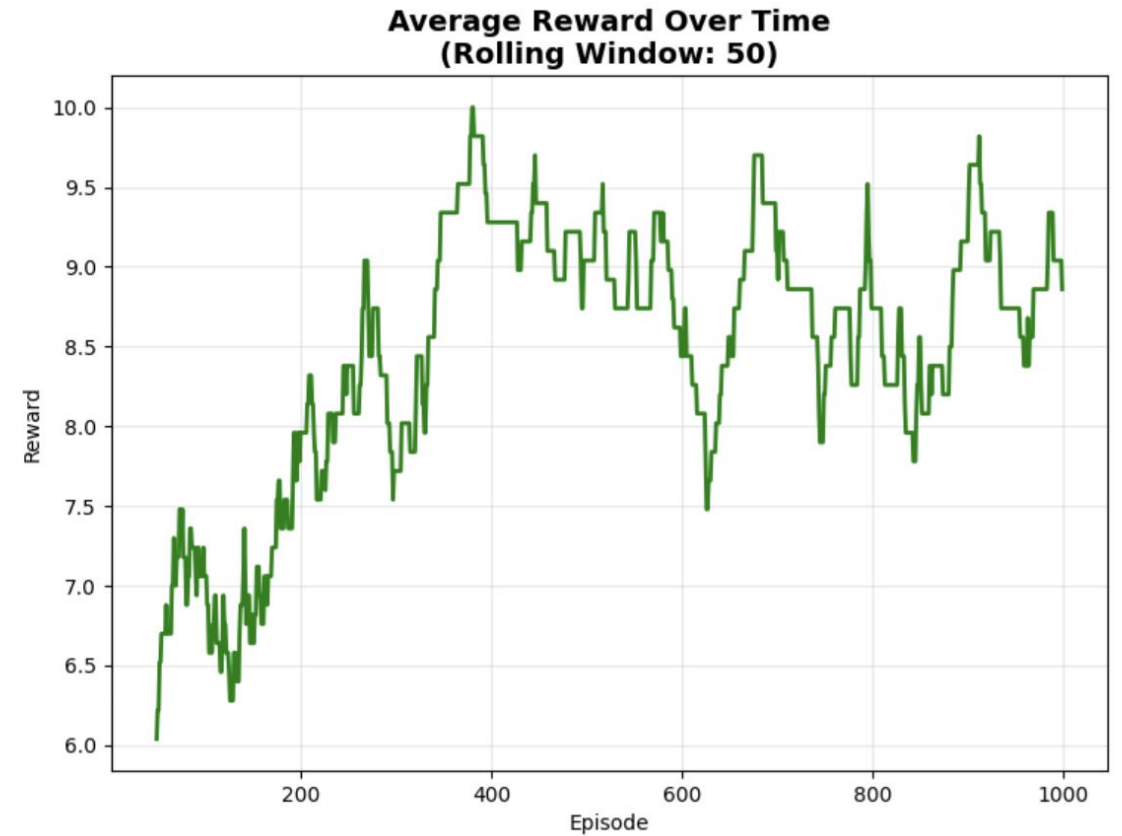
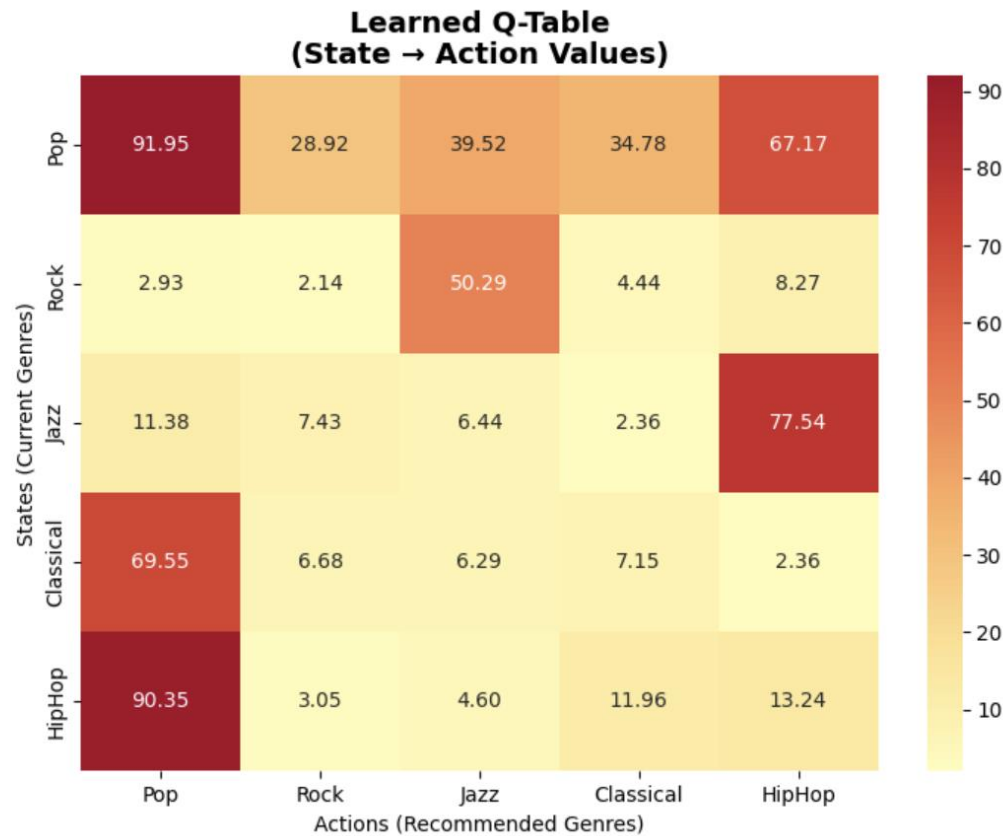
Results (Generic users)



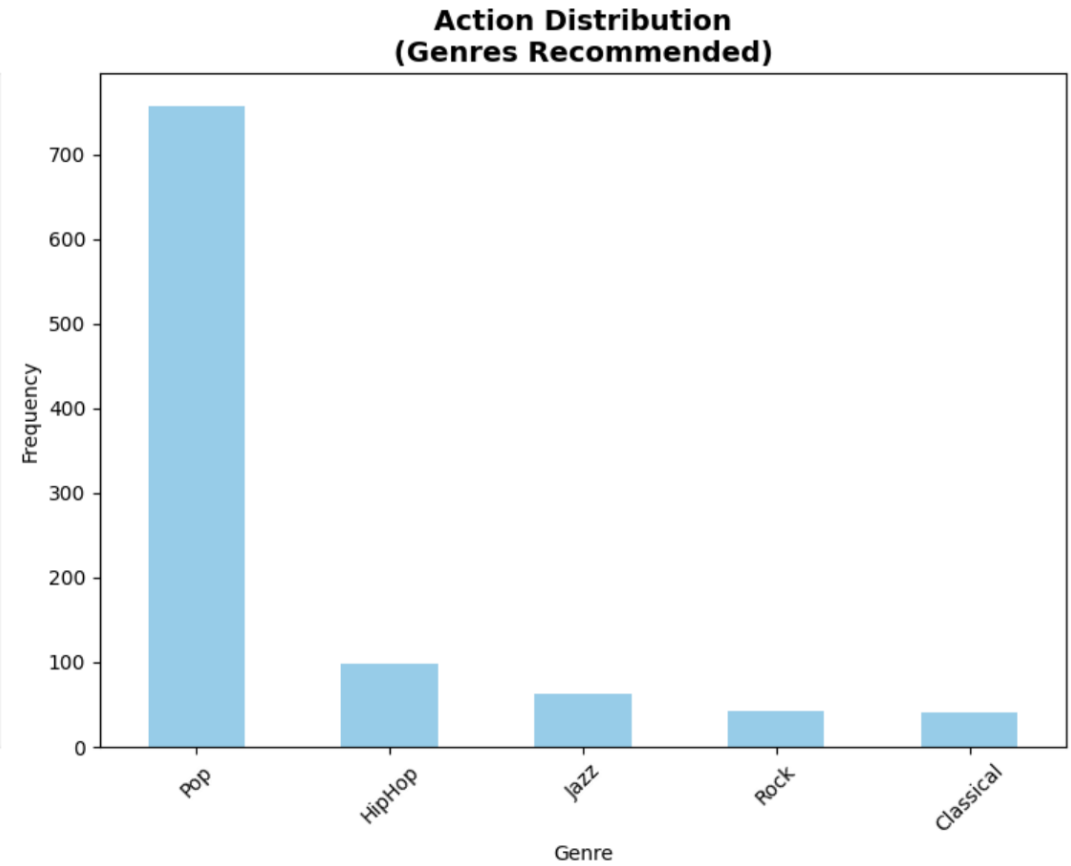
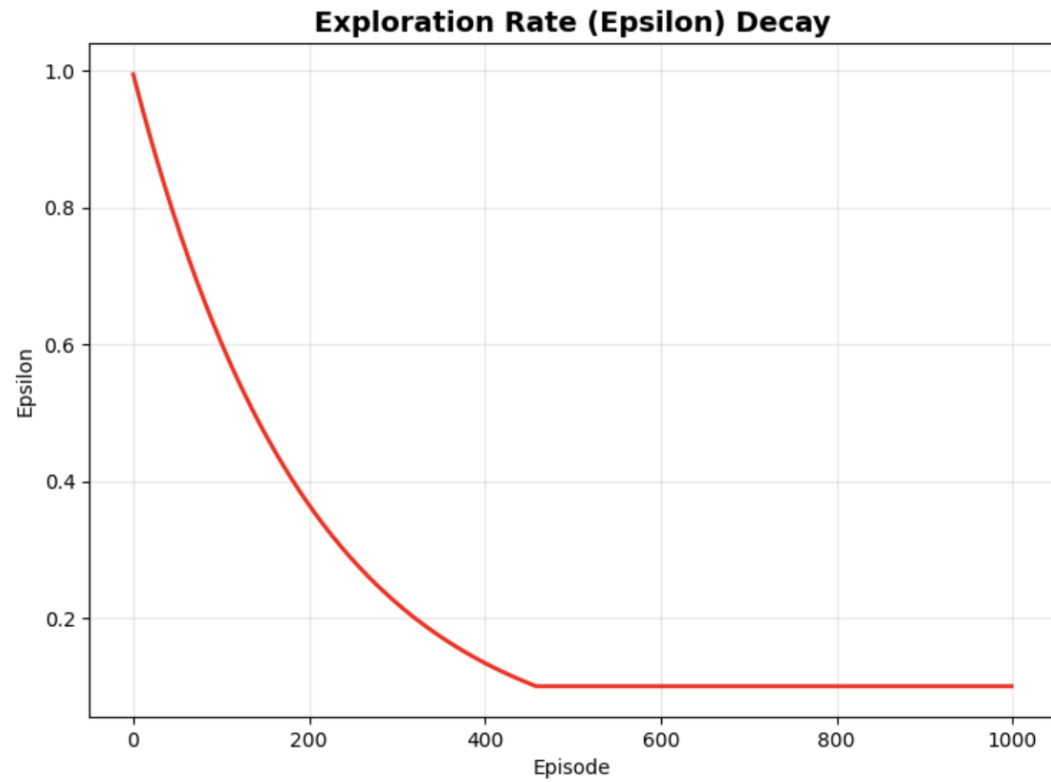
Results (Generic users)



Results(Pop lover agent)



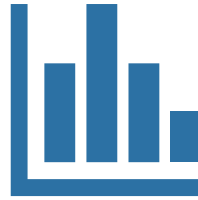
Results(Pop lover agent)



AI Assist



Adding comments



Graphs



Design



Thank you