

**ARTIFICIAL INTELLIGENCE PROJECT ON**

**RL BASED SYSTEM FOR ASSISTING CAB DRIVERS**

**SUBMITTED BY**

**AKHILA AVULDAPURAM (00916387)**

**ROOPA BRUNDA JAMPANI(00879573)**

**VARUN CHOWDARY BANDLAMUDI (00885862)**

**TABLE OF CONTENTS:**

|  |  |
| --- | --- |
| INTRODUCTION | **2** |
| APPROACH | **2** |
| PROJECT OBJECTIVE | **2** |
| COMPONENTS | **3** |
| ASSUMPTIONS | **4** |
| CODE EXPLANATION | **5** |
| RESULTS | **8** |
| CONCLUSION | **10** |

# **INTRODUCTION:**

Most drivers in this cutthroat industry receive a respectable volume of rides from clients during the day. But many drivers lament that even if revenue is up, their earnings are not growing, especially in light of the recent rises in gas and energy prices. It is crucial for drivers to make the proper decisions to optimize their daily profits by selecting the right rides. We are utilizing reinforcement learning in this project to help cab drivers increase their income. In the dynamic realm of urban transportation, enhancing the decision-making abilities of taxi drivers and streamlining routes can significantly boost customer happiness and productivity. The Cab Driver Assistance System employs reinforcement learning, a kind of artificial intelligence, to provide intelligent decision support to cab drivers.

# **APPROACH:**

The objective of this project is to develop an environment and an RL agent capable of selecting the optimal request. Deep Q-learning (DQN) is required for our agent's training.

**OBJECTIVES:**

* By applying reinforcement learning techniques, you can make the system learn and suggest the best routes based on historical data, user preferences, and traffic conditions at the moment. Route planning will be ideal as a result of this.
* Dynamic Decision Support: Provide taxi drivers access to a real-time decision-making tool that accounts for a variety of factors, including changing customer demand, road congestion, and weather.
* Adaptive Learning: Build a system that continuously learns from user input and the actions of taxi drivers, adapting its strategies to the ever-changing urban environment.

**GOALS:**

* **Create an environment:**

The "environment class" is the "Env.py" file; every method (function) in the class serves a particular function.

* **Build an Agent:**
* Create an agent that uses DQN to learn which request is the best. We are free to select the hyperparameters of our choosing, such as the learning rate, discount factor, and epsilon (decay rate). Plot the Q-Values of a small sample of state-action pairings along episodes to see if the Q-Values are convergent.
* Training is solely dependent on our selection of the epsilon-function. If decays quickly, our model won't be able to explore much, and the Q-values will converge early but to levels that are not ideal. Our model will converge slowly if it decays gradually.

**COMPONENTS:**

* The environment model serves as a realistic training ground for reinforcement learning algorithms by simulating the city's traffic patterns, roadways, and potential impediments.
* Reinforcement learning techniques can be used to help the system discover the best rules via trial and error. Examples of these algorithms are Q-learning and Deep Q Network (DQN).
* UI: Create a user-friendly UI so that taxi drivers may get recommendations in real time and decide what to do. Additionally, for ongoing learning, the interface ought to make user feedback easier.

# **ASSUMPTIONS:**

The taxis run on electricity. It may operate continuously for 30 days, or 24 by 30 hours.

After that, it must refuel. If the cab driver is finishing up at that point in his journey, he will stop to recharge. Thus, the terminal condition is reached as soon as the taxi driver completes 24\*30 hours, regardless of how many rides they complete in a month.

* Only five spots in the city are suitable for the cab to operate.
* Every choice is made once every hour. For this project, minutes and seconds won't be considered. For instance, a cab driver may receive a request at 3 PM, 4 PM, and so forth. Only during these periods may he choose which of the requests to accept. At 3:30 PM, a request cannot be made.
* Traffic conditions determine how long it takes to travel between locations, which is measured in integer hours only. Additionally, traffic is influenced by the day of the week and the hour of the day.

**BENEFITS:**

* Efficiency: Provide the best routes to take in order to reduce travel time and fuel consumption.
* Customer satisfaction: Cut down on wait times and deliver dependable, prompt service to improve the entire customer experience.
* Adaptability: The system changes with time to keep up with shifting city dynamics and traffic patterns.

**CODE EXPLANATION:**

**Environment Class:**

The goal of a reinforcement learning challenge is to teach an agent how to interact with its surroundings. By doing actions, the agent reaches several situations, or states. Rewards for actions may be both good and bad.

Here, the agent's sole goal is to optimize its overall reward throughout the duration of an episode. Everything that occurs in the environment between the initial state and the last, or terminal, state is included in this episode. We encourage the agent to gain experience to understand how to execute the optimal actions. This is the approach or guideline.

This is the "environment class"; every function or method within the class serves a certain function. Set the hyperparameters to their initial values.

m =5, number of locations ranges from 0...m-1

t =24, number of hours, ranges from 0....t-1

d =30, number of days, ranges from 0 ... d-1

C =5, Cost Per hour for fuel and other costs

R =9, Reward per hour revenue from a passenger

For feeding the NN, convert the state into a vector. This process turns a specified state into a vector format. The vector is of size m + t + d.

We have 2 architectures of DQN,

We pass only State as input.

We pass State and Action as input.

Architecture 1 (which takes only State as input) outperforms Architecture 2 since you only need to execute the NN once for each state because we obtain Q (s, a) for every operation. Proceed with the action where Q (s, a) is at its maximum.

**Next State function:**

Takes state and action as input and returns next state with considering below conditions.

Driver refuses to request.

Cab is already at pick up point.

Cab is not at the pickup point.

**Reward function:**

Assessment requires to determine what action is to be taken to minimize loss and maximize benefits. The reward, r (s, a), in our system for taking an action a ∈ A

at a given state s ∈ Sis computed as follows.

**Cab Driver DQN Agent:**

In Agent class we need to work on below functions are

Assigning hyperparameters

Creating a neural-network model.

Define epsilon-greedy strategy.

Appends the recent experience state, action, reward, new state to the memory.

Build the DQN model using the updated input and output batch.

**Hyperparameters:**

We can tweak these parameters for better performance. self.discount\_factor = 0.95

self.learning\_rate = 0.01

self.epsilon = 1

self.epsilon\_max = 1

self.epsilon\_decay = -0.0001

self.epsilon\_min = 0.00001

**Neural Network Model:**

Using keras we build a sequential model by adding dense layers.

In order to improve learning with a relu activation function, we added hidden layers after supplying the state as input at the first layer using a relu nonlinear activation function.

Here, we are using Adam optimizer which uses epsilon greedy policy and learning rate to improve the weights and bias to minimize mean square error.

**Build the DQN model:**

Appends the recent experience state, action, reward, new state to the memory with updated input and output batch.

**EVALUATION:**

To find a strategy that maximizes long-term cumulative benefits, the model updates its tactics continually.

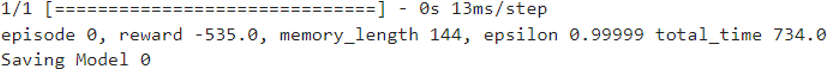
Below two are the performance metrices for our model.

Q-Value convergence.

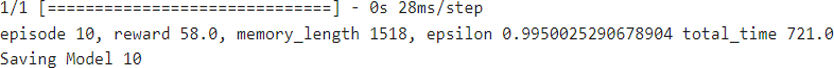
Rewards per episode.

**RESULTS:**

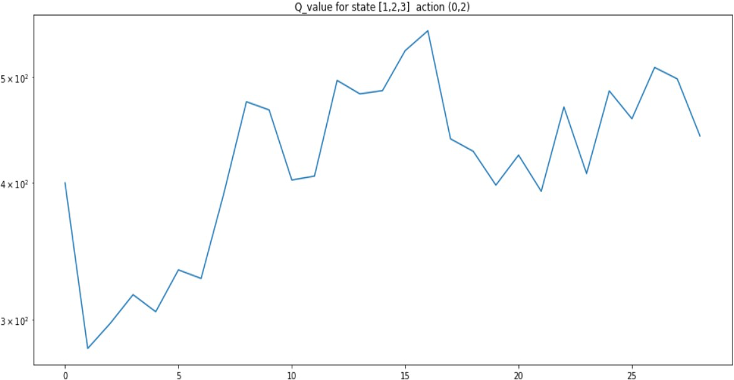
First, because our agent lacked experience, it did not learn as much, hence the incentives were not as good.



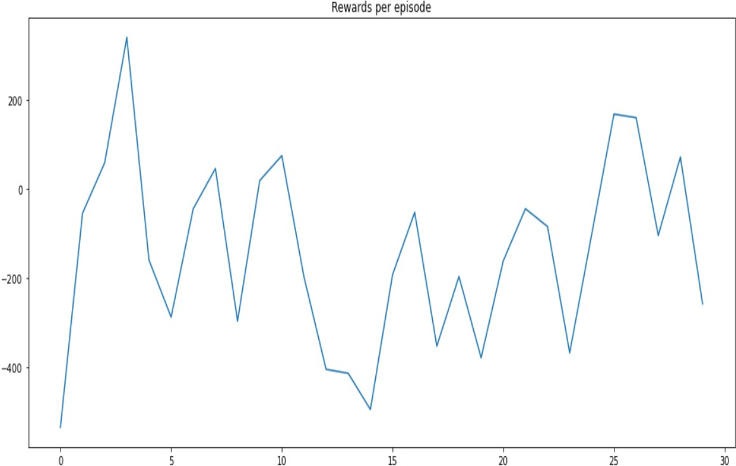
After some episodes, our agent got the positive rewards, as the episode increases the rewards for the agent also increases.



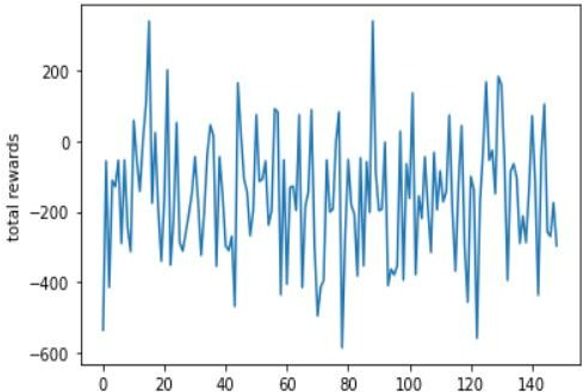
**Q-VALUE CONVERGENCE:**



**REWARDS PER EPISODE:**



**TOTAL REWARDS:**



**CONCLUSION:**

An RL-Based system agent that may improve taxi drivers' decision-making and help them optimize their profits has been developed by us using the Deep Q-Learning Network. We have shown Q-value convergence and incentives per episode to help you better understand the model's success. With each additional ride added to the system, the overall reward that is accrued increases. With the help of the Cab Driver Assistance System, which uses reinforcement learning, taxi drivers can now navigate cities more effectively and provide better services to customers by arming themselves with intelligent tools.