# Optimizing Customer Targeting Using Reinforcement Learning and Neural Networks for Adaptive Marketing Strategies

Mohammad Zubair Khan*
Katz School of Science and Health
Yeshiva University
New York, NY, USA
*mkhan10@mail.yu.edu

David Li†
Katz School of Science and Health
Yeshiva University
New York, NY, USA
†david.li@yu.edu

***Abstract*—Optimizing customer targeting in marketing is crucial for maximizing returns, yet traditional approaches often struggle to adapt to changing customer behaviors. This paper presents a reinforcement learning-based framework that models customers' spending inclinations as normal distributions to optimize targeting strategies. We introduce a novel mean-stat strategy, which leverages statistical confidence intervals to dynamically adjust exploration-exploitation trade-offs. Unlike traditional methods such as $\epsilon$-greedy and decaying $\epsilon$-greedy, which are used for comparison, the mean-stat strategy is mathematically proven to provide superior performance by efficiently identifying high-value customers. Additionally, a neural network-based approach using the cross-entropy method further enhances adaptability by learning complex patterns in customer behavior. Extensive simulations validate the effectiveness of these strategies, demonstrating the practical advantages of incorporating advanced learning techniques and mathematical rigor into marketing strategies.**

***Index Terms*—Reinforcement Learning, Customer Targeting, Marketing, Cross-Entropy Method, Neural Networks.**

## I. Introduction

In a competitive marketplace, identifying and targeting high-value customers is crucial for profitability. Traditional marketing approaches often fail to adapt dynamically to changing customer behaviors. Reinforcement learning (RL) offers a promising solution by simulating customer spending inclinations, modeled as normal distributions, and optimizing targeting strategies. This paper compares traditional RL methods with a novel mean-statistical strategy, which uses statistical confidence intervals, and a neural network-based approach for environments with complex reward distributions.

The simulation of customer behavior has been widely studied in fields such as e-commerce [1] and advertising [2]. However, its application in broader marketing strategies remains relatively unexplored. We address this gap by implementing and comparing several strategies aimed at optimizing customer targeting based on simulated data.

## II. Related Work

Customer simulations and reinforcement learning (RL) have shown significant promise for optimizing marketing strategies by predicting customer behavior and responses to promotional offers. The $\epsilon$-greedy strategy, which balances exploration and exploitation by selecting a random action with probability $\epsilon$ and the action with the highest estimated reward with probability $1 - \epsilon$, has been a widely adopted approach in RL research [3]. Variants such as the decaying $\epsilon$-greedy strategy have been introduced to reduce exploration over time, allowing the agent to exploit more as confidence in the learned policy increases [4].

In the realm of deep reinforcement learning, Mnih et al. [5] successfully utilized the $\epsilon$-greedy strategy as part of the exploration techniques for training deep Q-networks (DQN) on high-dimensional sensory inputs, such as raw pixel data from Atari games. The study demonstrated that the $\epsilon$-greedy strategy effectively balances learning by preventing premature convergence to suboptimal policies. Furthermore, Dann et al. [6] provided theoretical analysis for $\epsilon$-greedy with function approximation, establishing sample complexity bounds and regret guarantees in tasks where traditional exploration policies can struggle. Their results highlighted the simplicity and effectiveness of $\epsilon$-greedy in practical applications.

Recent work has also extended the $\epsilon$-greedy framework to adaptive and domain-specific scenarios. For example, Coelho-Magalhães et al. [7] employed a decaying $\epsilon$-greedy approach in a functional electrical stimulation-induced cycling controller. The decaying $\epsilon$ strategy allowed the RL agent to adaptively adjust stimulation patterns over time for real-time optimization of muscle activation.

While confidence-based exploration techniques like Upper Confidence Bound (UCB) and Thompson Sampling have been effective in certain RL scenarios, their computational requirements can be prohibitive in large-scale customer targeting problems [8]. Therefore, we propose the mean-statistical (mean-stat) strategy in this paper, leveraging statistical confidence intervals to dynamically adjust targeting probabilities. This strategy aims to offer a practical trade-off between computational efficiency and performance, particularly in high-dimensional environments. Additionally, neural network-based policies utilizing the cross-entropy method provide a robust framework for learning customer behaviors and refining targeting strategies [9], [10].

## III. Reinforcement Learning for Customer Targeting

We explore the application of reinforcement learning (RL) to optimize customer targeting in a market setting. The goal is to simulate customer behavior, where each customer has a hidden inclination to pay, represented as a normally distributed random variable. The RL agent aims to maximize cumulative rewards by selecting actions based on learned patterns of customer behavior.

### *A. Modeling Customer Behavior*

We represent each customer's propensity to spend as a stochastic process, modeled by a normal distribution parameterized by a mean $\mu$ and standard deviation $\sigma$, capturing the expected spending behavior and its associated variability. The environment comprises $K$ distinct customers, with each customer defined by its specific parameters $\mu_k$ and $\sigma_k$. The agent's objective is to learn the latent reward distributions associated with these customers and to select actions (target customers) that maximize the expected cumulative reward over time.

At each time step, the RL agent selects a customer based on the current policy. The reward is drawn from the customer's associated probability distribution. The objective of the agent is to maximize the cumulative reward over $N$ time steps, which translates into identifying and targeting high-value customers.

Fig. 1 illustrates the simulated reward distribution for 10 customers. Each customer exhibits a distinct spending pattern, and the RL agent uses these observations to inform its strategy.

## IV. Reinforcement Learning Strategies

### *A. Epsilon-Greedy Strategy*

The $\epsilon$-greedy strategy is a well-established method for managing the exploration-exploitation trade-off in reinforcement learning. The strategy operates by selecting the action with the highest estimated reward with probability $1 - \epsilon$, while choosing a random action with probability $\epsilon$. This approach ensures that the RL agent continues to explore the action space while also exploiting known good actions. Mnih et al. [5] utilized the $\epsilon$-greedy strategy in deep Q-learning for Atari games, illustrating its effectiveness in balancing the need for exploration to discover new policies with the desire to maximize cumulative rewards.

### *B. Decaying $\epsilon$-Greedy Strategy*

To improve the balance between exploration and exploitation, a decaying $\epsilon$-greedy strategy is employed. In this variant, the parameter $\epsilon$ decreases over time, allowing for more exploration during the early stages of learning and gradually shifting towards greater exploitation as the agent becomes more confident in its value estimates. This approach has been shown to improve learning outcomes in dynamic environments, such as in the work by Coelho-Magalhães et al. [7], where it was used to optimize muscle activation patterns in functional electrical stimulation cycling. As $\epsilon$ decays, the algorithm reduces randomness, leading to a more refined policy.

By integrating these $\epsilon$-greedy techniques and augmenting them with new strategies, such as the proposed mean-stat strategy, we aim to further enhance the agent's ability to learn from customer behaviors in a marketing context. This approach offers a framework for dynamically adjusting targeting probabilities based on observed reward uncertainties and statistical measures, addressing some of the limitations of conventional RL strategies.

### *C. Decay Schedule $1/t$-Greedy Strategy*

The $1/t$-Greedy [11] Strategy is a variant of the traditional $\epsilon$-greedy strategy where the exploration rate decays according to the schedule $\epsilon = 1/t$, where $t$ is the current time step or episode number. This decay schedule reduces the exploration rate more gradually as learning progresses, ensuring that the agent continues to explore for longer than in fixed or rapidly decaying schedules, thus balancing exploration and exploitation more effectively over time.

The benefits of the $1/t$-greedy strategy include:

- **Smooth Transition from Exploration to Exploitation**: Unlike fixed decay rates, the $1/t$ schedule provides a smoother transition from exploration to exploitation. This is particularly useful in environments where learning progresses slowly or the agent needs to adapt to changing dynamics.
- **Continuous Exploration**: By decaying at a slower rate than exponentially decreasing schedules, the $1/t$-greedy strategy ensures that the agent does not cease exploration entirely, maintaining a small but non-zero probability of exploring new actions even in later stages of learning.
- **Adaptation to Changing Environments**: In dynamic environments, where the optimal policy may shift over time, the $1/t$-greedy strategy allows for ongoing adaptation by keeping some level of exploration active, thus enabling the agent to adjust to new optimal strategies.

Mathematically, the decay schedule can be expressed as:

$$\epsilon_t = \frac{1}{t}$$

where $\epsilon_t$ represents the exploration rate at time step $t$. As $t$ approaches infinity, $\epsilon_t$ approaches zero, leading the agent to focus almost entirely on exploiting the best-known action. This gradual decay helps prevent overfitting to specific actions by maintaining diversity in the agent's action choices throughout the learning process.

The $1/t$-greedy strategy is especially suitable for problems with a large number of states or actions, where extensive exploration is necessary to find optimal policies. However, the slow decay may result in longer convergence times, making it more appropriate for tasks where long-term learning is feasible.

### *D. Mean-Statistical Strategy*

The mean-statistical strategy leverages confidence intervals around the estimated means of each customer's reward. It

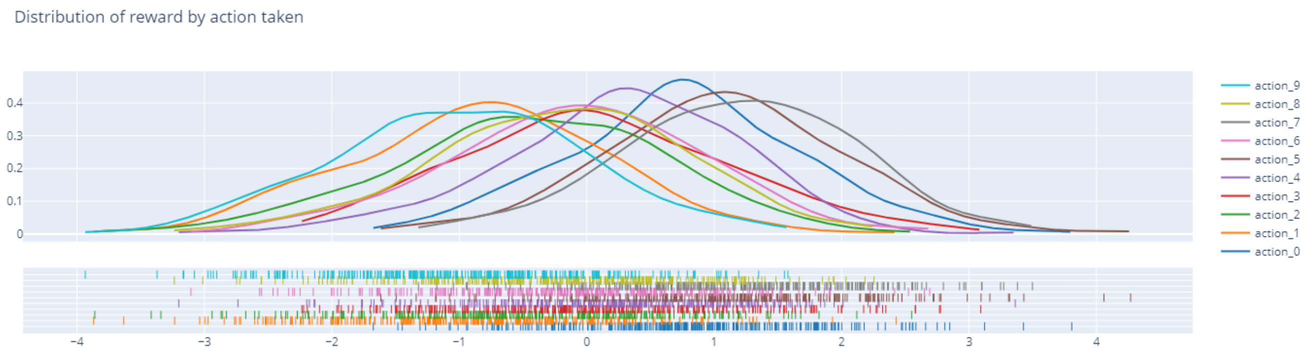

Fig. 1. Reward distribution simulated for 10 customers with random selection, showing distinct spending patterns.

checks if the upper bound for any customer is lower than the lower bound for any customer, if so, that customer is not targeted any further. By using statistical significance, it adjusts the probability of selecting each customer, passing the remaining customer's upper bounds through a softmax calculation. This method dynamically updates customer selection based on observed reward uncertainty, using:

$$\mu_{ui} = \mu_i + z_{\alpha/2}\frac{\sigma_i}{\sqrt{n_i}}$$

$$P(i,t) = \frac{e^{\mu_{ui}}\mathbb{1}_{\{i\notin(k_{te}\cup k_{tu})\}} + \mathbb{1}_{\{i\in k_{tu}\}}\frac{\sum_{j=1;j\notin(k_{te}\cup k_{tu})}^{N} e^{\mu_{uj}}}{N-|(k_{te}\cup k_{tu})|}}{\sum_{j=1;j\notin(k_{te}\cup k_{tu})}^{N} e^{\mu_{uj}} + |k_{tu}|\frac{\sum_{j=1;j\notin(k_{te}\cup k_{tu})}^{N} e^{\mu_{j}}}{N-|(k_{te}\cup k_{tu})|}} \tag{1}$$

with $\mathbb{1}_{\{.\}}$ as indicator function, $z_{\alpha/2}$ represents the z-score (t-score if less than 30 samples) for a $95\%$ confidence interval, $N$ is the number of customers, $k_{te}$ is the set of customers the strategy is sure about their mean are lower under the statistical significance till the time step $t$, $k_{tu}$ is the set of all unexplored customers, $|k_t|$ is the cardinality of the set, $\sigma_i$ is the standard error, $\mu_i$ is the mean estimate, and $n_i$ is the number of times customer $i$ has been selected. For all the customers in $k_t$ the probability is zero.

In the beginning, the strategy starts by assigning equal probability to each of the customers. As soon as the number of samples crosses 2 for any of the customers, the probability calculation becomes as shown in Equation (1). This equation ensures that the sum is always 1, thus making the policy interpretable as a probability.

This has been implemented in the accompanying code, detailed in Appendix A.

### *E. Neural Network-Based Approach*

Instead of directly feeding the mean upper bounds to softmax, we extend the RL framework by incorporating a neural network-based policy using the cross-entropy method. The neural network is tasked with predicting the optimal customer based on observed data, here the observations are the output confidence intervals from our mean stat strategy for the reward estimates i.e. mean estimate upper bound as well as lower bounds for each customer, and the number of times a customer was selected. This approach allows the RL agent to adapt its strategy dynamically by learning patterns in the reward distributions of different customers.

This gives an elastic edge to the pre-existing knowledge of mean estimates. As the neural network can learn the right patterns of producing policy in the early stages to maximize the returns.

The neural network architecture is designed to efficiently process the input data and produce a decision policy that maximizes the cumulative reward. The architecture is as follows:

The neural network consists of:

- **Input**: A vector representing the upper and lower bounds of the estimated reward for each customer, along with the number of samples for each customer. This representation captures both the confidence in the reward estimates (via the bounds) and the frequency of sampling, which helps the network understand which customers have been sufficiently explored.
- **Architecture**: A single hidden layer with 128 neurons, using the ReLU (Rectified Linear Unit) activation function. The ReLU function is chosen because it helps the network learn non-linear relationships effectively while avoiding the vanishing gradient problem, which is common in deep networks. The hidden layer allows the model to capture complex interactions in the input data, improving the quality of the action predictions.
- **Output**: A softmax layer that provides a probability distribution over the customers, determining which customer to target at each time step. The softmax function ensures that the output values sum to one, making them interpretable as probabilities. The network selects the customer with the highest probability as the target for the current time step.

The network is trained using the cross-entropy loss function:

$$L = -\sum_{i=1}^{n} y_i \log(p_i) \tag{2}$$

where $L$ is the loss function (cross-entropy), $y_i$ is the true label (indicating the action that yields the highest reward), and $p_i$ is the predicted probability from the neural network for action $i$. This loss function measures the difference between the predicted probability distribution and the true action distribution, guiding the training process to reduce the prediction error.

Training the network involves several steps:

- **Data Collection**: The agent interacts with the environment, collecting episodes consisting of state, action, and reward tuples. Each episode provides a sequence of customer-related statistics as states, customer selections, and the corresponding rewards received.
- **Batch Processing**: The collected episodes are divided into batches, and the network is trained on these batches using gradient descent. The batch size is typically set to a small number, such as 16, to ensure stable updates while still allowing the network to generalize from diverse data points.
- **Elite Episode Selection**: The cross-entropy method filters out episodes with low rewards, retaining only the top-performing episodes (e.g., the top 30% based on the total reward). These "elite" episodes are used to train the network, ensuring that the model focuses on learning from the most successful strategies.
- **Network Update**: The network parameters are updated using backpropagation and an optimization algorithm such as Adam. The optimizer adjusts the weights to minimize the cross-entropy loss, improving the model's predictions over successive training iterations.
- **Exploration and Exploitation Balance**: During training, the network is encouraged to explore various customers by adjusting its action selection policy, as the mean estimate intervals are very large. This exploration is gradually reduced as the model becomes more confident in its predictions, allowing it to focus on exploiting the most rewarding top customers.

By incorporating the cross-entropy method, the neural network-based approach enhances the RL agent's ability to refine its targeting policy continuously. It leverages past interactions to learn patterns in the reward structure, leading to more efficient identification of high-paying customers compared to traditional strategies. Additionally, the training continues until the network starts outperforming the mean stat strategy.

The effectiveness of the neural network-based strategy is further evaluated through simulations, where it is shown to outperform simpler methods like $\epsilon$-greedy and complex ones like mean-stat strategies in scenarios with complex reward distributions.

## V. Mathematical Foundations of Customer Targeting Strategies

### A. Estimating Expected Rewards

For each customer $i$, let their observed average spending inclination be denoted by $\mu_i$. Let there be a total $N$ steps taken to obtain rewards from $n$ customers using competing strategies like decay schedule-$1/t$ and mean-stat.

### B. For schedule-$1/t$

The expected reward from decay schedule-$1/t$ can be estimated using:

$$E_{1/t}(N) = \frac{1}{N}\sum_{t=1}^{N}\left((1-\frac{1}{t})\mu_{i_{max}} + \frac{1}{t}\sum_{j\neq i_{max}}^{n}\mu_j(\frac{1}{n-1})\right) \tag{3}$$

### C. For Mean-Statistical Strategy

In the mean-stat strategy, the expected return can be estimated using:

$$E_{ms}(N) = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{n}P(i,t)\mu_{it} \tag{4}$$

The strategy explores uniformly early on ($P(i,t) = \frac{1}{n}$ when $n_{ti} < 2$ for all customers $i$ at time step $t$), then shifts towards higher-reward customers as the estimates improve. Assuming this strategy samples twice from each customer in the first $2n$ steps then. For $t < 2n$, the probability of choosing any customer is uniform:

$$P(i,t) = \frac{1}{n}$$

For $t > 2n$, the probabilities are adjusted based on the softmax of the estimated rewards as mentioned in Equation (1)

This method ensures that the strategy initially explores all customers equally and then gradually shifts to favor the customers with higher estimated rewards. While ignoring the low returns customers with confidence.

### D. Comparison of Strategies

Here are the trends in the chances of finding the optimal customer as time progresses for different strategies:

- **Random strategy**:

$$\frac{1}{n} \longrightarrow \frac{1}{n} \tag{5}$$

- $\epsilon$ **strategy**:

$$e \longrightarrow e \tag{6}$$

- $\epsilon$**-greedy strategy**:

$$\frac{1}{n} \longrightarrow e \tag{7}$$

- $\frac{1}{t}$ **strategy**:

$$\frac{1}{n} \longrightarrow \frac{1}{t} \times \frac{1}{n-1} \tag{8}$$

- **Mean-stat strategy**:
$$\frac{1}{n} \longrightarrow \frac{e^{\mu_{i,t}}}{\sum_{j=1}^{n-|k_{te}|} e^{\mu_{j,t}}} \tag{9}$$
where $|k_{te}|$ is the count of customers whose estimated confidence interval falls below any of the other customers at a time step $t$.

This shows that, for all the other strategies, except for mean-stat, the probability of finding the optimal customer either remains constant or decreases. But only in the mean-stat strategy does the chances of finding the optimal customer if not already found increase from $1/n$ to a bigger value as the denominator keeps shrinking from Equation (9).

## VI. Statistical Estimation and Confidence Intervals

### A. Estimating Standard Deviation

Given $n$ random samples, the sample mean $\bar{x}$ is calculated as:
$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$
The sample standard deviation is given by:
$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$
This estimate follows a chi-squared distribution with $n-1$ degrees of freedom:
$$\frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1}$$
From this, we can compute the confidence interval of the standard deviation for the mean:
$$\left(\frac{(n-1)s^2}{\chi^2_{\alpha/2,n-1}}, \frac{(n-1)s^2}{\chi^2_{1-\alpha/2,n-1}}\right)$$
As we acquire more samples, the interval shrinks, making the estimates more certain over time. Knowing to use Chi-squared distribution helps pin down the interval. By incorporating neural networks we hope the model learns hidden distributions of the space which makes the deep reinforcement learning approach stand in advantage against traditional pure statistics-based models such as mean-stat that we have discussed in this paper.

## VII. Experimental Setup and Results

### A. Simulation Setup

We simulate a marketing scenario with 10 customers, each with a distinct normal distribution of spending inclinations. The strategies are run for 3,000 steps, and the cumulative average rewards and action distributions are recorded.

### B. Cumulative Reward

Fig. 2 shows the cumulative average reward for each strategy. The mean-stat and neural network strategies significantly outperform $\epsilon$-greedy methods and decay schedule-$1/t$ by efficiently identifying high-paying customers.

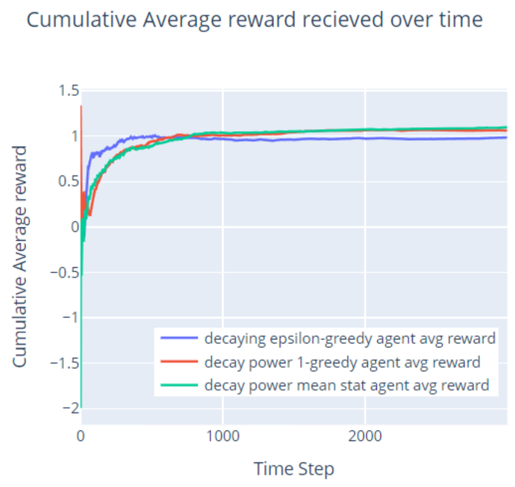

Fig. 2. Cumulative average reward for different strategies.

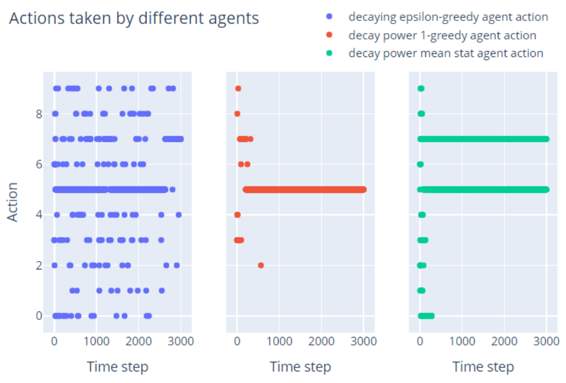

Fig. 3. Distribution of targeting actions over time.

### C. Action Distribution

Fig. 3 illustrates the distribution of targeting actions taken by each strategy. The mean-stat and neural network methods quickly converge on the most lucrative customers, while decaying $\epsilon$-greedy strategies exhibit a more uniform distribution.

## VIII. Discussion and Conclusion

The results demonstrate that the mean-stat strategy outperforms random, $\epsilon$-greedy methods and decay schedule-$1/t$ method by using statistical confidence intervals to guide decisions. The neural network approach, although computationally more intensive, provides greater flexibility in adapting to complex reward distributions.

As shown in Fig. 1, customers 5 and 7 are the most rewarding to target. In Fig. 3 we can see how our strategy ends up optimally using resources to smartly only target the top rewarding customers and therefore, generate the most reward as seen in the Fig. 2.

The trends in Fig. 2 show that our strategy initially experiences losses, as indicated in Fig. 3, while it attempts to learn effectively to optimize performance in the long run. By adding neural networks to this approach, the strategy is able to account for unknown distribution more smartly as it has seen and learned the policy that is effective across a wide variety of distributed simulations.

These strategies provide a strong foundation for customer targeting in marketing, where understanding customer behavior and adapting marketing strategies dynamically is crucial. Future work will involve extending this framework to incorporate more complex customer behavior models and exploring additional RL-based strategies.

All these simulations revealed that the faster a strategy stops exploring the more chances are it is not going to find the optimal customer. Whereas as we see from (9) that is not the case with the mean-stat strategy.

The primary advantage of the mean-stat strategy is its robustness in balancing exploration and exploitation, making it well-suited for real-world applications such as customer targeting in restaurants or other similar setups.

## IX. Real-World Application: Restaurant Targeting

In a real-world scenario, restaurants can use the mean-stat strategy to identify high-paying customers by offering promotions selectively. Over time, the strategy converges on customers with higher spending potential, allowing the restaurant to maximize returns on promotional efforts.

Using the mean-stat strategy, a restaurant can sample customers by offering initial promotions, then analyze the subsequent spending behavior. Based on the spending patterns, the restaurant can adjust its marketing strategy, focusing on high-value customers while still exploring other customers.

## X. Conclusion

This paper introduces innovative strategies for optimizing customer targeting in marketing using reinforcement learning. We proposed the mean-stat strategy, which leverages statistical confidence intervals to dynamically guide the exploration-exploitation trade-off, and a neural network-based approach utilizing the cross-entropy method for more adaptive decision-making. Through extensive simulations, we demonstrated that both the mean-stat and neural network-based methods consistently outperform traditional strategies such as $\epsilon$-greedy and decay schedule-$1/t$ by more effectively identifying high-value customers.

The mean-stat strategy effectively balances exploration and exploitation, converging on the most rewarding customers while maintaining the flexibility to adapt to changing observed dynamics. The neural network approach further enhances this adaptability by learning complex patterns in customer behavior, resulting in a more refined targeting policy in diverse reward distributions.

Our findings indicate that ceasing exploration faster increases the risk of suboptimal outcomes. The mean-stat strategy's ability to continue exploration based on statistical significance ensures a greater probability of identifying the optimal customers over time, as evidenced by the evolving action distributions and cumulative reward trends in our experiments.

Future work will focus on expanding the framework to encompass more sophisticated models of customer behavior, integrating additional real-world data sources, and exploring alternative reinforcement learning methods to further improve targeting effectiveness. These enhancements could help adapt the strategies to complex, real-world marketing scenarios, such as personalized promotions in various industries.

Ultimately, the approaches presented in this paper offer a practical and scalable foundation for dynamic customer targeting, where understanding customer behavior and continuously adapting marketing strategies are critical for maximizing returns.

## References

[1] W. Wang, B. Li, X. Luo, and X. Wang, "Deep reinforcement learning for sequential targeting," *Management Science*, vol. 69, no. 9, pp. 5439–5460, 2023. [Online]. Available: https://doi.org/10.1287/mnsc.2022.4621

[2] G. Shani, D. Heckerman, and R. I. Brafman, "An mdp-based recommender system," *Journal of Machine Learning Research*, vol. 6, pp. 1265–1295, 2005.

[3] W. C. J. C. H., "Learning from delayed rewards," *PhD thesis University of Cambridge, England*, 1989. [Online]. Available: https://cir.nii.ac.jp/crid/1570291224641056384

[4] B. Andrew and S. Richard S, "Reinforcement learning: An introduction," 2018.

[5] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[6] C. Dann, Y. Mansour, M. Mohri, A. Sekhari, and K. Sridharan, "Guarantees for epsilon-greedy reinforcement learning with function approximation," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 4666–4689. [Online]. Available: https://proceedings.mlr.press/v162/dann22a.html

[7] T. Coelho-Magalhães, C. Azevedo Coste, and H. Resende-Martins, "A novel functional electrical stimulation-induced cycling controller using reinforcement learning to optimize online muscle activation pattern," *Sensors*, vol. 22, no. 23, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/23/9126

[8] W. R. THOMPSON, "ON THE LIKELIHOOD THAT ONE UNKNOWN PROBABILITY EXCEEDS ANOTHER IN VIEW OF THE EVIDENCE OF TWO SAMPLES," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, 12 1933. [Online]. Available: https://doi.org/10.1093/biomet/25.3-4.285

[9] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and Computing in Applied Probability*, vol. 1, no. 2, pp. 127–190, 1999.

[10] P.-T. de Boer, D. Kroese, S. Mannor, and R. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, 2005. [Online]. Available: https://EconPapers.repec.org/RePEc:spr:annopr:v:134:y:2005:i:1:p:19-67:10.1007/s10479-005-5724-z

[11] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

## Appendix A
## Python Code

This appendix includes Python code snippets used to implement the various strategies. The full implementation, along with additional details, can be found in the GitHub repository: https://github.com/.