Journal of Computer Science

Original Research Paper

Reinforcement Learning in Financial Services: Modelling Payment Switching as a Multi-Armed Bandit Problem

[1]Ishaya Gambo, [2]Christopher Agbonkhese, [1]Segun Aina, [3]Mogboluwaga Tayo Otegbayo, [4]Johnson Bayo Adekunle, [1]Israel Odetola, [5]Omobola Gambo, [1]Tolulope Oluwadare and [1]Oluwatoni Odetola

[1]*Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria*
[2]*Department of Digital and Computational Studies, Bates College, Lewiston, USA*
[3]*Vitruvian Shield PT, LDA, Portugal*
[4]*Venture Garden Group, Ikeja, Lagos, Nigeria*
[5]*Department of Arts and Social Science Education, Lead City University, Nigeria*

Article history
Received: 07- 03- 2024
Revised: 17-05-2024
Accepted: 01-06 2024

Corresponding Author:
Ishaya Gambo
Department of Computer Science and Engineering, Awolowo University, Ile-Ife, Nigeria
Email: ipgambo@oauife.edu.ng

Abstract: The ever-evolving landscape of digital payments demands continuous innovation and self-improvement. This study addresses this imperative by simulating a model for payment routing, a crucial aspect of the digital payment ecosystem. To achieve this, industry professionals were interviewed to inform the approach, emphasizing data randomization for effective data collection. Using Python, a randomized dataset is created and three Reinforcement Learning (RL) algorithms are implemented and evaluated: Epsilon Greedy, Upper Confidence Bound (UCB), and Thompson Sampling. The paper adopts the Multi-Armed Bandit (MAB) framework to model payment routing as a resource allocation problem, offering a computational approach to real-world resource allocation dilemmas. Through simulation, we eliminate real-time transaction costs, allowing us to focus on algorithmic approaches without implications for customers, businesses, or payment providers. Among the RL algorithms studied, UCB emerges as the most effective in addressing this Multi-Armed Bandit problem, corroborating findings from prior research. This study suggests not only the potential of modeling real-world problems as MAB but also the superior performance of the UCB algorithm in solving RL problems. The paper underscores the need for increased focus on non-consumer-facing aspects of the financial services industry, emphasizing cross-disciplinary research to create infrastructure and software solutions. Researchers can extend this study by exploring MAB algorithms in various domains with options for system choices. The simulation-based approach offers a cost-effective means of testing system performance and hypotheses across a spectrum of industries, fostering innovation and progress.

Keywords: Multi-Armed Bandit Problem, Reinforcement Learning, Digital Payments, Transaction, Simulation

Introduction

The evolution of technology has brought about a profound transformation in the financial services sector on a global scale. Innovations such as mobile banking applications, automated teller machines, credit and debit cards, stock prediction, automated savings, e-commerce, AI-supported financial advisory, and embedded finance have revolutionized the industry. These digitization efforts have undoubtedly reshaped the landscape, providing unparalleled convenience and accessibility to consumers worldwide.

However, with the increasing prevalence of digital transactions (Alessio *et al*., 2022), the persistent occurrence of online transaction failures presents a significant challenge to the reliability and widespread adoption of digital financial solutions. This ongoing issue underscores the critical importance of prioritizing error reduction in the integration of technology to enhance financial service delivery, particularly considering the far-reaching global implications of failures within the financial services industry.

The multifaceted nature of transaction failures encompasses various factors, including system downtime, poor network connections, gateway timeouts, incorrect details, and declined transactions by card issuers, among others (Flutterwave, n.d.; Parikh, 2019; Stripe, 2023). These failures not only erode trust in digital payments but also restrict customer options, impeding the transition from traditional to digital modes of financial transactions. The impact extends to businesses that rely on receiving payments and to consumers seeking to make payments or transfer funds. Consequently, these failures can lead to businesses enforcing policies such as requiring customers to wait for transaction alerts before leaving or preferring cash payments, thereby leaving customers with limited alternatives. This highlights concerns about the reliability of online transactions, echoing sentiments expressed by Blackmon and Mwesigwa (2021), who identify reliability as a key barrier preventing many individuals from accessing the formal financial system.

In response to the complex nature of transaction failures, various strategies have been implemented for error detection and correction within digital financial services. Among these strategies, one of the most common is the display of pop-up error messages, which provide users with insights into the reasons behind transaction failures. For instance, error messages such as "Issuer/Switch Inoperative," which usually indicate challenges such as the inability to reach the issuer, the unresponsiveness of the issuer, or the failure of the payment processor to complete payment authorization, or "Timeout Transactions," are often displayed when failures occur, alerting users to underlying issues (Duò, 2022).

Going further, discerning between failures originating from the switch and those from the issuer is a crucial aspect of managing transaction failures. If the failure stems from the switch, it may require a change of the payment switch by the payment terminal or gateway. Conversely, if the failure lies with the issuer, customers may need to complete their transactions using an alternative method or try again later. Despite the importance of distinguishing between switch and issuer failures, this task can be challenging due to the coexistence of multiple switches in practice. Payment terminals and gateways can select which switch provider to utilize and some entities even have multiple switches available for processing payments. However, only one switch can be used per transaction at a time. This complexity underscores the challenges of managing transaction failures and emphasizes the importance of effective error detection and resolution mechanisms in digital financial services. This problem can be effectively modeled as a Multi-Armed Bandit (MAB) problem from a computing perspective.

In response to existing challenges within financial services on this issue, there has been a growing application of technology to address these issues. Some notable attempts include portfolio selection and optimization (ElectrifAi, 2020; Hambly *et al*., 2021), stock price forecasting (Sahu *et al*., 2023), credit score modeling (Lu *et al*., 2013), algorithmic trading, risk and credit rating assessment, asset pricing, insurance claims assessment and fraud detection (Bao *et al*., 2020). These applications utilize a diverse array of technologies, including software development, cloud computing, embedded systems, artificial intelligence, cybersecurity, deep learning, and machine learning (Fong *et al*., 2021).

Nevertheless, Chawla *et al*. (2021) underscore the limited focus on transaction failures and correction modes in the existing literature on online payment systems, highlighting it as a largely unexplored area. While machine and deep learning have received considerable attention, Reinforcement Learning (RL) remains relatively underexplored in specific domains. However, notable attempts have been made to apply RL in fields such as healthcare (Bastani and Bayati, 2020; Durand *et al*., 2018; Yauney and Shah, 2018) and recommendation systems (Zhou *et al*., 2017). Furthermore, Arthur *et al*. (2020) emphasize the potential applications of RL in risk management, market microstructure, and portfolio allocation within finance, indicating promising prospects for RL techniques in addressing financial challenges.

In this study, we adopt the RL approach due to its suitability for addressing the multi-armed bandit problem, a subset within the field of decision-making under uncertainty. The paper aims to fill the gap highlighted by Chawla *et al*. (2021) by investigating the potential of leveraging RL to enhance the success rates of digital transactions. Through this exploration, the goal is to contribute to a deeper understanding of transaction failures and to advance solutions for improving the reliability of digital financial services.

Related Work

At the intersection of finance, economics, and reinforcement learning, researchers have worked on modeling problems and developing computational solutions in each of these fields independently and in some cases collectively. MAB problems and algorithms have gotten significant attention from researchers, further justifying this research area.

Gifford *et al*. (1995) provide one of the earliest papers on payment switching as a part of payment networks. They provide a foundation for efficient payment networks and a relevant historical context, but their work has limited applicability as payments have evolved to include more players and are more sophisticated.

Cheng *et al*. (2016) address the complexity of payment routing for financial credit networks. The authors employ the game theory model and investigate the potential of algorithmic approaches to optimal payment routing. The

combination of theoretics and algorithmic design offers valuable insight into the challenge of efficient payment routing. However, as game theory is employed, the assumption of rational behavior may not be practical in real financial networks.

Comșa *et al*. (2019) compared the performance of RL algorithms in optimizing the management of radio resources, and packet scheduling. The authors compare ten RL algorithms, using similar values for these parameters; learning rate, discount factor, and exploration. The simulation performed validates the performance of RL over adaptive schedulers and its potential to improve 5G network provisioning. The problem and solution belonged to the RL field, but they did not fit the MAB framework. Therefore, the problem did not use any MAB algorithm.

Nagendra *et al*. (2017) conducted a study where they compared the efficacy of three RL algorithms Q-learning, value function approximation, and policy gradient actor-critic in addressing the cart-pole problem. Additionally, they explored an approach integrating the RL algorithm with a practical swing-up controller. The best-performing algorithm differed in a discrete and continuous space but had no performance variations when the swing-up was integrated. This makes a good case for the need to compare RL algorithms, but it focuses on a singular problem for RL algorithms; the cart-pole balancing problem.

Mahajan and Teneketzis (2008) take a mathematical approach to explaining MAB problems including features of classical MAB problems, computational issues, and variations of MAB problems. While this approach provides a problem definition and modeling, it does not account for the factors that come into play in real-world applications as many assumptions are made in mathematical modeling.

Bouneffouf *et al*. (2020) conducted a survey examining the practical applications of both MAB and contextual bandits. Their aim was to underscore the significance of these techniques in addressing real-world challenges. Their work creates a taxonomy of MAB application domains which include healthcare, finance, recommender systems, and anomaly detection. As it reports the outcome of a survey, it does not detail the application domains or how domain-specific systems can leverage RL.

Bouneffouf and Féraud (2016) consider the scenario where the agent is aware of the reward function for each arm but is unaware of the distribution. The reward function aimed to improve the efficiency of the financial system in conducting transactions (Gambo *et al*., 2023). The authors develop a new algorithm which they name the Adjusted Upper Confidence Bound algorithm (AUCB) that is adjusted to fit the problem scenario and compare it with the regular Upper Confidence Bound algorithm (UCB) and five other MAB algorithms. The AUCB outperformed all others across sigmoid, Gaussian, and decreasing reward functions but is only relevant for scenarios that fit the context of the solution.

Zhou *et al*. (2019) formulate the route selection problem for transportation as an MAB problem, with the varying routes to a destination as the possible "arms" that can be selected. By viewing vehicles as servers and sensors, they apply the UCB algorithm and MAB algorithm to find the shortest route for a vehicle that has a fixed start location and destination. By using the shortest path algorithm as the benchmark, the UCB algorithm had a lower mean regret value and improved arrival time by 4.8%. While it validates the potential of MAB algorithms in routing problems, its domain specificity means their solution cannot be directly applied in another problem domain such as humanities, healthcare, or finance.

Arthur *et al*. (2020) highlight the applicability of RL in finance and economics, particularly MAB problems. Their work highlights the theoretics in RL problem formulation, leveraging mathematical notations to model how RL can be used in economic modeling, risk management, and game theory. However, their work does not point to the applicability of RL in payments or provide a code implementation of the mathematical models.

Huo and Fu (2017) modeled portfolio selection as a stochastic multiarmed bandit problem, exploring the exploration-exploitation dilemma in selecting portfolios. They highlight how MAB is a suitable model for decision-making in an uncertainty. They introduce risk awareness in their research, combining asset filtering, an MAB policy, and risk minimization in their algorithm to achieve a risk-return balance. Their research highlights the potential of MAB and RL solutions in financial services but is limited by domain specificity.

The next section describes the approach taken to designing the model and the algorithms for payment switch selection. From requirement gathering to simulation and evaluation.

Materials and Methods

The simulation of the model was conducted in a basic Multi-Armed Bandit (MAB) problem environment. Since the model does not leverage GPU capabilities, its hardware requirements are minimal. The simulation was run on a laptop or desktop computer with at least 2GB of RAM and a processor with a speed of 2.0GHz or higher. The modeling and simulation were done within a code editor, making the choice of operating system flexible. Any operating system that can accommodate a code editor, such as Windows, Linux, or macOS, is sufficient. For this research, however, the simulation was conducted in a Windows environment. The model does not require an internet connection or virtual machine, and it operates

efficiently under these conditions without high-end computing resources.

Requirement Gathering Phase

The requirement-gathering phase involves the identification of the features required for the development of the model. The interview approach was selected owing to the nascent and sensitive nature of digital payments. To identify the relevant features in the dataset, payment experts in Nigeria were interviewed.

In practice, payment success requires the connectivity of APIs provided by varying companies in the payment chain. However, these APIs are usually paid for and require licensing in some cases. These were pointed out by industry experts, making modeling and simulation the best approach to addressing this problem.

Owing to the sensitivity of the data in the payments industry, no public dataset on payment transactions exists. Hence, the a need to create a randomized dataset to address the problem. The data required included the feedback codes for switch/issuer inoperative error and payment success; 91 and 00, transaction amounts, and card network. We used the provided list of all possible error codes in payments as described by Stripe (2023). While the error codes and card networks are fixed, the transaction amount can be any value acceptable by payment systems. In this study, the range of 500 and 100,000 is used for the transaction amount.

Modeling and Design Specification

This phase involved the development of a skeletal framework for how the system will operate. The models developed in this phase are based on the requirements gathered in the previous phase. The system design provides a blueprint of the system that can be developed into a working solution (Gambo and Agbonkhese, 2024) using suitable programming languages. The unified object-oriented modeling framework was used to model the adjustment the system makes to the existing payments system and the workings of the algorithms in the model.

The sequence diagram, shown in Fig. (1), illustrates the interactions occurring in the payment process and the flow of payments when a switch selection exists in the routing process. When payment is initiated, the routing system selects a switch with which to route a payment. Based on the transaction feedback received in return, it can select the switch for the next transaction, learning on the go. This shows the difficulty of balancing exploration and exploitation in the MAB problem.

The class diagram depicted in Fig. (2) offers a clear high-level view of the interactions existing in the system modeled in this study. The base class for all the MAB algorithms; the Bandit algorithm, generalizes on all the algorithms, having all the common attributes of each the number of arms and total rewards for each arm. Each of the three algorithms is presented as a subclass, with its own separate features. The Bandit algorithm class is associated with both the Streamlit app and the data class. The data class abstracts the randomized dataset and the streamlit app abstracts the streamlit interface used to simulate the transactions.

Dataset

The dataset, though randomized, accounts for specific scenarios that could occur in a payment routing process. Following the interviews in the requirement phase, four switches were settled for in the dataset. Hence, the routing is to occur among 4 switches. The scenarios accounted for include: Three switches fail, one succeeds; Two switches fail, two succeed; One switch fails, three succeed and all switches succeed. These scenarios are plausible in a real-world payment process and are accounted for. The case of all switches failing is ignored as there is no routing to happen in this case and it is a nearly impossible scenario. The row items under the switches are randomized using 00 for success and 91 for failure, following the standard error codes as seen in Stripe (2023).

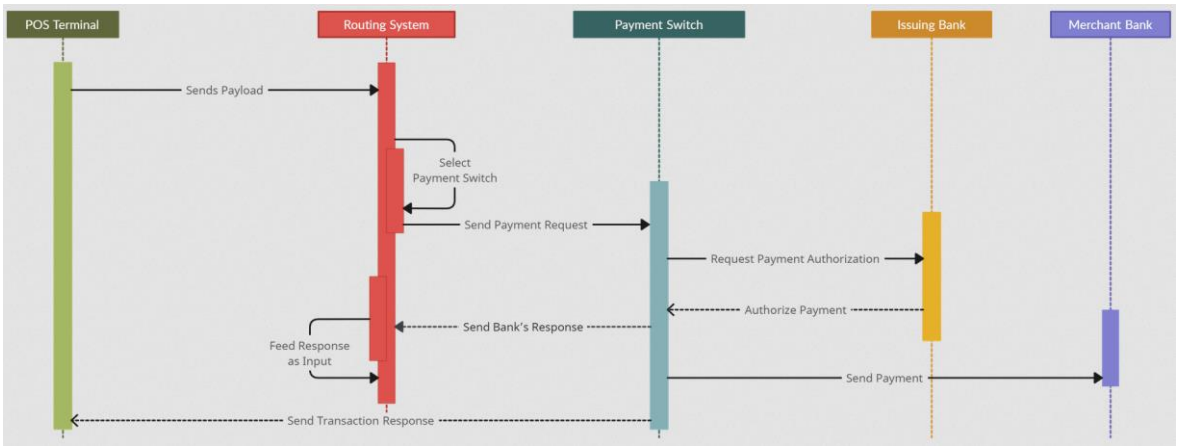

Fig 1: Sequence Diagram of expected payment routing flow

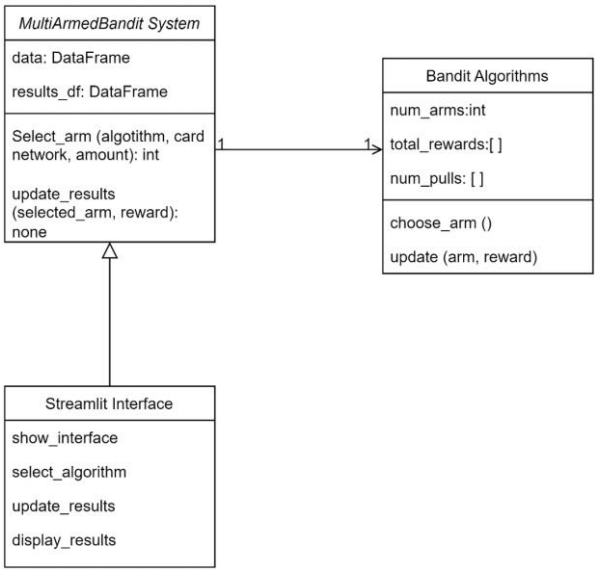

Fig 2: Class Diagram showing classes and their methods

The dataset also contains the card network and transaction amount columns. The card network is randomly distributed but follows a weighted distribution. Today, three card networks active in Nigeria are Interswitch's Verve, Mastercard, and VISA. The weighting followed the data from Statista (2023) highlighting that 18% of Cardholders had a VISA card, 28% MasterCard and 54% Verve. Hence the weights 0.18, 0.28, and 0.54 are allocated to each of the card networks.

The transaction amounts are randomly generated but within the range of 500 and 100,000 as this is a fair bandwidth, as industry experts interviewed highlighted that the largest volume of digital transactions falls within the range. The data randomizations are all done with Python code. The language's random module, NumPy library, and Pandas library are leveraged to achieve this.

The dataset of 4,000 rows is used in training the RL algorithms to learn from the dataset. The selections made in the simulation are benchmarked with a test dataset that is unknown to the algorithms. An 80-20 split between training and validation sets is used in this study.

Streamlit Simulation

The need for an interface to test the workability of using MAB algorithms in payment routing gives credence to the use of Streamlit to create an interface that simplifies the payment data relevant to the system; transaction amount and card network. Added to this interface is a selection pane with the three algorithms allowing for the selection of different algorithms while leaving transaction amount and card network unchanged to see the decision the algorithm will make. The code implementation is seen in Fig. (3).

As Fig. (3) reflects, the code creates a Streamlit web interface with options to select between epsilon greedy, Thompson Sampling, and UCB algorithms. Depending on the selected algorithm, it displays simulation results such as total rewards and mean regret.

Implementation and Evaluation

In this study, we chose the upper confidence bound algorithm, epsilon greedy algorithm, and Thompson Sampling algorithm as the algorithms to use.

The ε-Greedy algorithm is an MAB algorithm that attempts to balance exploration and exploitation by allocating some rounds of selection to exploration, i.e., random selection of arms and the rest to exploitation. Each arm in an environment has an estimate of its reward $Q_t(a)$ at a time t. The ε parameter defines the rate of exploration and the action A_t follows that Eq. (1):

$$A_t = \begin{cases} argmax\, Q_t(a), with\ probability\ 1 - \varepsilon \\ random\ arm\ (a), with\ probability\ \varepsilon \end{cases} \quad (1)$$

The ε parameter is used to define the exploration-exploitation trade-off. Usually between 0 and 1, higher ε values favor exploration and lower values favor exploitation, defining how the algorithm adapts to the problem (Mohan, 2014). While its simplicity and computational efficiency are worthwhile, the ε-Greedy algorithm may struggle in environments where the reward distribution changes rapidly (Sutton, 2018).

```python
import streamlit as st
import numpy as np
import pandas as pd

# Import multi-armed bandit algorithms (epsilon-greedy, Thompson Sampling, UCB)
from Epsilon_Greedy_alg import epsilon_greedy
from UCB_alg import ucb
from Thompson_Sampling_alg import thompson_sampling
#from Thompson_aalg import thompson_sampling

# Load pre-existing data
data = pd.read_csv("randomized_dataset.csv")  # Load your data as shown in the previous response

st.title("Multi-Armed Bandit App")

# User input for transaction amount and card network
amount = st.number_input("Transaction Amount", 500, 100000, step=100)
card_network = st.selectbox("Card Network", ["Mastercard", "Visa", "Verve"])

# Define the selected algorithm (e.g., via a radio button)
algorithm = st.radio("Select Algorithm", ["Epsilon Greedy", "Thompson Sampling", "UCB"])

# Function to select an arm based on the chosen algorithm
def select_arm(algorithm):
    # Convert data to a Pandas DataFrame
    data_df = pd.DataFrame(data) #, columns=["Switch1", "Switch2", "Switch3", "Switch4", ]
    # Create a new row of data with the given amount and card network
    new_row = np.zeros(4)
    if card_network == "Mastercard":
        new_row[0] = amount
    elif card_network == "Visa":
        new_row[1] = amount
    elif card_network == "Verve":
        new_row[2] = amount

    # Append the new data to the existing data
    new_data = np.vstack([data_df, new_row])

    if algorithm == "Epsilon-Greedy":
        rewards, _ = epsilon_greedy(new_data.iloc[:, :-2], epsilon=0.1)
    elif algorithm == "Thompson Sampling":
        rewards, _ = thompson_sampling(new_data.iloc[:, :-2])
    elif algorithm == "UCB":
        rewards, _ = ucb(new_data.iloc[:, :-2])

    selected_arm = np.argmax(rewards)
    return selected_arm

if st.button("Select Arm"):
    selected_arm = select_arm(algorithm)
    st.success(f"Selected Arm: {selected_arm}")
```

Fig. 3: Streamlit code for web interface

As its name implies, the UCB algorithm uses the UCB of the estimated reward of an arm in its attempt to balance exploration and exploitation. Usually based on Hoeffding's inequality, the algorithm estimates the upper bound of the reward of each arm and selects the arm having the highest upper board. At time step *t*, the UCB for an arm *a* is defined in Eq. (2):

$$A_t = arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right] \quad (2)$$

In Eq. (2), $Q_t(a)$ represents the estimated reward of arm *a*, *c* denotes the degree of exploration and $N_t(a)$ indicates the number of times arm a has been selected. The algorithm's approach to the exploration-exploitation trade-off involves selecting arms having higher uncertainty in the estimates of the reward, thus encouraging exploration while favoring arms with likely higher rewards (Zhang, 2019). While the UCB algorithm is better at adapting to changing rewards than ε-Greedy, it is more complex to implement.

The Thompson Sampling algorithm uses a Bayesian method to address the MAB problem. Thompson Sampling keeps a probability distribution (Beta distribution) over the arms and uses the sampling from the probability distributions to make a decision. At time step *t*, the algorithm selects the arm having the highest sampled value from the distribution samples.

Operating a "belief" system, Thompson Sampling balances the trade-off by sampling the arms by their probability distribution, making it effective in environments having complex reward structures (Russo *et al*., 2018). However, it can require computational more resources than other algorithms especially when many arms are involved.

The algorithms were implemented in code using Python in separate files before they were imported for use by Streamlit and evaluation. Owing to their disposition as algorithms that attempt to address the exploration-exploitation dilemma and specifically, the MAB problem, there are similarities between them. These similarities are highlighted in the class diagram in Fig. (2). The code implementations for these algorithms are presented in Figs. (4-6).

The Python code in Fig. (4) defines the class 'epsilon greedy' for the epsilon greedy algorithm in multi-armed bandit problems, initializing parameters and selecting arms accordingly. In Fig. (5), the Python class "Thompson sampling" implements Thompson Sampling for multi-armed bandit problems, initializing parameters and sampling from Beta distributions. Updates occur based on observed rewards. Moreover, in Fig. (6), the Python class "UCB" implements the UCB algorithm for multi-armed bandit problems, initializing parameters and selecting arms based on UCB values. Updates are performed based on observed rewards.

Overall, the three algorithms are encapsulated within Python classes, each designed to initialize parameters, select arms, and update values based on observed rewards. This implementation facilitates experimentation and analysis of various strategies for optimizing decision-making processes in scenarios with uncertainty and limited information (Olivier and Lihong, 2011; Auer and Ortner, 2010).

The algorithms consider the number of arms possible and the possible rewards. For the ε-Greedy algorithm, the value for epsilon, a number 0> ε >1 is passed as a parameter in its function.

In RL, a range of evaluation metrics exist to assess the performance of the learning process of an agent. However, different metrics are prioritized for different RL problems, the metrics that align with the learning outcomes and the task at hand are usually selected. For MAB problems, the important metrics are as shown in Table 1.

```python
import numpy as np
import pandas as pd

def epsilon_greedy(data, epsilon):
    num_arms = 4  # Number of arms (switches)
    num_rounds = len(data)

    rewards = np.zeros(num_arms)
    arm_counts = np.zeros(num_arms)

    regret = []

    for t in range(num_rounds):
        if np.random.rand() < epsilon:
            # Explore: Choose a random arm
            arm = np.random.randint(num_arms)
        else:
            # Exploit: Choose the arm with the highest estimated reward
            arm = np.argmax(rewards)

        reward = data.iloc[t, arm]
        arm_counts[arm] += 1

        # Map "91" (failure) to 0 and "00" (success) to 1
        if reward == "91":
            reward = 0
        elif reward == "00":
            reward = 1

        rewards[arm] += (reward - rewards[arm]) / arm_counts[arm]

        # Calculate regret
        optimal_reward = max(data.iloc[t, :num_arms])
        regret.append(optimal_reward - reward)

    return rewards, regret
```

Fig 4: Python code for epsilon greedy algorithm

```python
import numpy as np
import pandas as pd

def thompson_sampling(data):
    num_arms = 4
    num_rounds = len(data)

    alpha = np.ones(num_arms)
    beta = np.ones(num_arms)

    rewards = np.zeros(num_arms)
    regret = []

    for t in range(num_rounds):
        sampled_means = np.random.beta(alpha, beta)
        arm = np.argmax(sampled_means)

        reward = data.iloc[t, arm]

        # Map "91" (failure) to 0 and "00" (success) to 1
        if reward == "91":
            reward = 0
        elif reward == "00":
            reward = 1

        alpha[arm] += reward
        beta[arm] += 1 - reward

        rewards[arm] += reward

        # Calculate regret
        optimal_reward = max(data.iloc[t, :num_arms])  # Consider only the arm columns
        regret.append(optimal_reward - reward)

    return rewards, regret
```

Fig. 5: Python code for Thompson sampling algorithm

Table. 1: Important metrics used for MAB problems

SN	Metrics	Remarks
1	Total cumulative regret	The total cumulative regret is the sum of the gaps between the rewards that the bandit algorithm received and the rewards that the optimal arm would have given. It shows how much the algorithm "regrets" it had chosen differently throughout the experiment
2	Mean cumulative regret	Mean cumulative regret is the average regret per round or time step. It provides a normalized view of how well the bandit algorithm is performing over time. Lower mean cumulative regret indicates better performance
3	Total reward	Total reward represents the cumulative sum of rewards earned by the bandit algorithm throughout the experiment. Maximizing this metric is a primary goal of the bandit algorithm
4	F1-score	This metric serves as a balance between precision and recall, serving as a harmonic mean. It is mostly in binary classification but is a prominent metric in the computations that involve machine learning models
5	Accuracy	It measures the extent of the correctness of the predictions made. In this study, it assesses the number of times each algorithm correctly predicts the right switch in comparison with the dataset

```python
UCB_alg.py > ucb

def ucb(data):
    num_arms = 4
    num_rounds = len(data)

    rewards = np.zeros(num_arms)
    arm_counts = np.zeros(num_arms)
    regret = []

    for t in range(num_rounds):
        if 0 in arm_counts:
            # If any arm hasn't been pulled yet, select it
            arm = np.where(arm_counts == 0)[0][0]
        else:
            # Calculate the upper confidence bound for each arm
            ucb_values = rewards + np.sqrt(2 * np.log(t) / arm_counts)

            # Choose the arm with the highest UCB value
            arm = np.argmax(ucb_values)

        reward = data.iloc[t, arm]

        # Map "91" (failure) to 0 and "00" (success) to 1
        if reward == "91":
            reward = 0
        elif reward == "00":
            reward = 1

        arm_counts[arm] += 1
        rewards[arm] += (reward - rewards[arm]) / arm_counts[arm]

        # Calculate regret
        optimal_reward = max(data.iloc[t, :num_arms])
        regret.append(optimal_reward - reward)

    return rewards, regret
```

Fig. 6: Python code for UCB algorithm

Results and Discussion

Streamlit is used to simulate a payment transaction via a web interface, where both transaction amount and card network are selected. This web interface is presented in Fig. (7). In a real-world scenario, more card information is required to authenticate the transaction such as CVV and card expiry date. However, these are not required for the simulation. Also, on the web interface, is a list of the three algorithms. This allows for the selection of an MAB when a transaction is initiated. This way, each of the three algorithms can be tested for the arm (switch) they select when benchmarked against the same transaction amount and card network.

The three algorithms performed differently when benchmarked against the randomized dataset. As it is an RL problem, the benchmark metrics differ slightly from regular ML problems. In Fig. (8), the total reward for each algorithm is presented visually. It highlights the number of times each algorithm made the right decision by selecting the best switch. Thus, earning it a reward. The dataset was passed through each algorithm and this allowed for the generation of the reward count for each of them.

The three algorithms performed differently when benchmarked against the randomized dataset. As it is an RL problem, the benchmark metrics differ slightly from regular ML problems. In Fig. (8), the total reward for each algorithm is presented visually. It highlights the number of times each algorithm made the right decision by selecting the best switch. Thus, earning it a reward. The dataset was passed through each algorithm and this allowed for the generation of the reward count for each of them.

The three algorithms performed differently when benchmarked against the randomized dataset. As it is an RL problem, the benchmark metrics differ slightly from regular ML problems. In Fig. (8), the total reward for each algorithm is presented visually. It highlights the number of times each algorithm made the right decision by selecting the best switch. Thus, earning it a reward. The dataset was passed through each algorithm and this allowed for the generation of the reward count for each of them.

Figure (9) indicates the mean regret for the algorithm. Worth mentioning, that regret is a common feature in RL problem formulation and solution presentation. The aim of RL solutions is to minimize regret and maximize rewards. This means that having lower regrets and higher rewards positions an algorithm as a better performer for the problem at hand. Regret can be viewed in two ways, mean regret and cumulative regret. The mean regret shown in Fig. (9) reflects the average difference between the returns achieved through the use of the three algorithms and the returns that

DOI: 10.3844/jcssp.2024.1519.1529

would have been obtained had the optimal strategy been selected retrospectively. Essentially, it quantifies the opportunity cost or loss incurred due to suboptimal decision-making over a period of time (Li *et al*., 2021). The cumulative regret for the algorithms is presented in Fig. (10). As Fig. (10) reflects, the cumulative regret is not only a sum of regrets but a highlight of the trend in regret from the first round to the last round.

We applied the f1-score and accuracy metrics, which are widely used for evaluating ML models, to measure the performance of the algorithms. These metrics are also suitable for RL. In this study, these metrics helped to a benchmark for comparing the efficiency of the algorithms implemented in a form that can be compared with traditional algorithms. A snapshot of the values obtained is presented in Table 2.

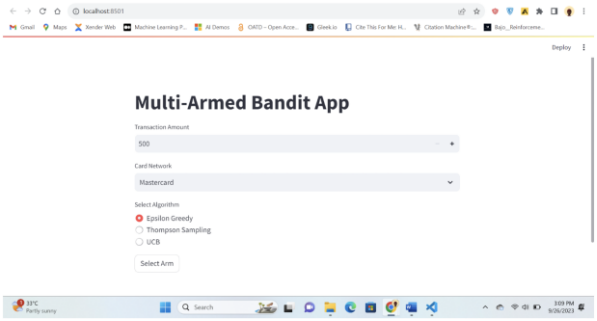

Fig. 7: Streamlit simulation interface

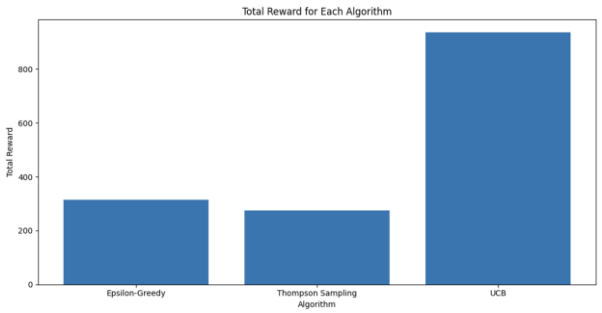

Fig. 8: Total reward chart for the algorithms

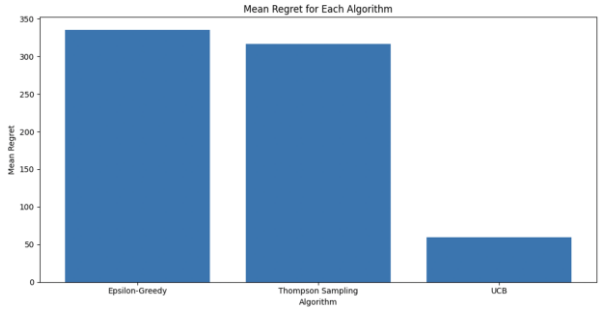

Fig. 9: Mean regret for the algorithms

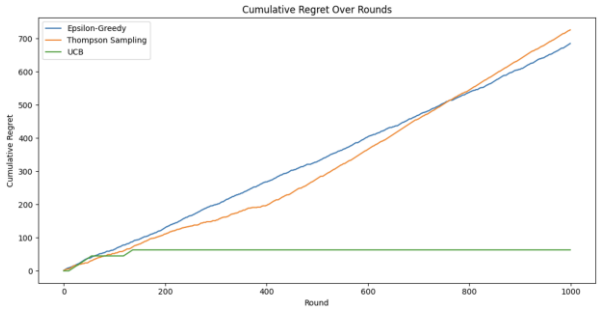

Fig. 10: Cumulative regret for the algorithms

Table 2: Accuracy and f1-score for each algorithm

Algorithm	Accuracy	F1-score
Epsilon greedy	0.315	0.478
Thompson sampling	0.274	0.428
UCB	0.937	0.966

Epsilon-Greedy Algorithm approach to balancing exploration and exploitation is influenced by the value of ε selected. As observed by Mohan (2014), smaller values favor exploitation. Owing to the number of arms in the environment, a value of 0.1 was chosen. Assessing cumulative regret in Fig. (10), the ε-Greedy algorithm sees a slow but upward slope, indicative of exploration, as the algorithm attempts to understand the environment and the reward functions.

However, between round 400 and 600, the cumulative regret has a steady and upward trend, exceeding that of Thompson Sampling by round 600. This highlights the problem of the ε-Greedy algorithm with exploitation. It continues to exploit an arm that already proved to be a good result and could stay with it without exploring other arms that could have higher rewards.

The total reward chart in Fig (8) shows the ε-Greedy algorithm slightly edges the Thompson algorithm in its sum of rewards. Looking at the cumulative regret chart in Fig. (10) helps give an explanation for this small but significant difference. ε-Greedy retained a lower regret up until a point between round 600 and 800 before exceeding the Thompson Sampling algorithm at round 800. The ε-Greedy mostly retained a lower regret and higher reward compared to the Thompson Sampling algorithm but there is no certainty that this would not have significantly changed if the number of rounds was increased by 50 or 100%.

The mean regret chart in Fig. (9) highlights a higher mean regret for the ε-Greedy algorithm which is only slightly higher than the Thompson Sampling algorithm. This trend somewhat conflicts with cumulative regret. However, the Thompson sampling algorithm maintained an upward yet steady increase in its cumulative regret, ensuring that its mean regret would be lower than that of the ε-Greedy algorithm which saw a rapid increase in its cumulative regret.

Across the regret and reward metrics, the UCB algorithm retained impressive results. The cumulative regret in Fig. (10) stands out as it highlights the algorithm's approach to the

exploration-exploitation dilemma in favor of exploration and arms with likely higher rewards. As a better adapter to changing rewards, the UCB algorithm initially has an upward trend in its cumulative regret which starts to steady then sharply increase before remaining steady till the end. This learning is impressive as it means the algorithm learns quickly and is less likely to choose the wrong switch among the available options.

The UCB algorithms have higher total reward and low mean regret seen in Figs. (8-9) which aligns with its cumulative reward values. This impressive results from UCB are reinforced in the F1 and accuracy results seen in Table 2. On a scale of 0 and 1 for both metrics, UCB tends closer to 1 with 0.966 and 0.937 F1-score and accuracy, respectively. In practice, this high accuracy might not be attainable due to the dynamic nature of systems when deployed for use. However, this high threshold indicates that its performance in practice will be within reasonable ranges.

The ε-Greedy algorithm outperforms the Thompson algorithm on both metrics, though with smaller margins than the UCB's. This highlights the possibility of improving the ε-Greedy algorithm to perform better. This could mean adjusting the value of the ε parameter to favor exploration by opting for higher values of ε.

Evidently, the UCB algorithm significantly outperforms the other two algorithms. The approach to balancing exploration and exploitation works well in environments with uncertainty, aligning with the environment of a switch routing system. While ε-Greedy performs better than Thompson Sampling across most metrics, the slightly lower mean regret score means the ε-Greedy had a higher regret per round on average.

Conclusion

It is possible to infer that switch selection in the payment process can introduce an automated routing system between the terminal and switches to improve the payment process and address transaction failures due to switch failures. This would improve the payment flow and reduce losses for consumers and businesses.

Payment processors can improve payment success by implementing similar solutions that fit their own organizational context, as variations exist in the internal payment flow of many processors. However, they can build upon the approach taken in this study to develop custom switch selection solutions that attempt to eliminate the switch/issuer inoperative error and better serve the customers and businesses who use their services.

This study exposes the urgent need for the financial services industry to be under the radar of researchers, from a non-consumer-facing perspective. Researchers can consider improving the existing model by considering API deployment, as an API will be capable of communicating with APIs of payment switches and payment terminals. This will require partnering with industry players for free access to their APIs.

Notably, there is a dearth of research in the development of infrastructure and internal software for financial services. Researchers can explore these gaps from a cross-disciplinary perspective. This is because it is a heavily regulated industry that cuts across law, International Trade, technology, and finance. Multifaceted research would deliver infrastructure for the industry that not only works but also fits the industry context. As Shafiq *et al*. (2024) suggest, handling conflicts in stakeholder goals using machine learning approaches can provide a useful framework for addressing such interdisciplinary challenges.

There are many situations where the explore-exploit dilemma exists, such as stock portfolio selection, medical trials, and recommender systems. These problems can be modeled as MAB problems and have MAB algorithms applied to address them. Researchers can explore MAB algorithms in the development of automated systems where options exist for the system to choose from. In environments where funding is limited, modeling and simulations provide an opportunity to test the performance of systems without incurring significant costs.

In conclusion, this study underscores the critical need for ongoing research into the financial services industry's infrastructure from a non-consumer-facing perspective. By building on the foundation laid in this study, future research can significantly advance the development of automated routing systems and custom switch selection solutions. Additionally, interdisciplinary approaches and the exploration of MAB algorithms hold promise for addressing complex problems within the financial services sector and beyond.

Acknowledgment

The authors acknowledge the support of TETFund and Centre of Excellence Obafemi Awolowo University, Ile-Ife in carrying out the research.

Funding Information

The authors gratefully acknowledge the financial support provided by TETFund and the Centre of Excellence, Obafemi Awolowo University, Ile-Ife, which made this research possible.

Author’s Contributions

All authors equally contributed to this work.

Ethics

The authors affirmed the originality of this article, containing unpublished materials. The corresponding author confirms that all co-authors have thoroughly reviewed and approved the manuscript and there are no ethical concerns associated with the research.

References

Alessio, B., Ahmed, F., Eitan, G., Nii Amaah, O.-A., & Edem, S. (2022). *The Future of Payments in Africa*. McKinsey & Company.

Arthur, C., Romuald, É., & Carl, R. (2020). *Reinforcement Learning in Economics and Finance*. arXiv.

Auer, P., & Ortner, R. (2010). UCB Revisited: Improved Regret Bounds for the Stochastic Multi-Armed Bandit Problem. *Periodica Mathematica Hungarica*, *61*(1), 55–65. https://doi.org/10.1007/s10998-010-3055-6

Bao, Y., Hilary, G., & Ke, B. (2022). Artificial Intelligence and Fraud Detection. In V. Babich, J. R. Birge, & G. Hilary (Eds.), *Innovative Technology at the Interface of Finance and Operations* (Vol. 11, pp. 223–247). Springer. https://doi.org/10.1007/978-3-030-75729-8_8

Bastani, H., & Bayati, M. (2020). Online Decision-Making with High-Dimensional Covariates. *Operations Research*, *68*(1), 276–294. https://doi.org/10.1287/opre.2019.1902

Blackmon, W., & Mwesigwa, B. (2021). *Measuring Fees and Transparency in Nigeria's Digital Financial Services: Using an Audit Study to Determine Reliability, Compliance, and Transparency of Provider Costs*.

Bouneffouf, D., & Féraud, R. (2016). Multi-Armed Bandit Problem with Known Trend. *Neurocomputing*, *205*, 16–21. https://doi.org/10.1016/j.neucom.2016.02.052

Bouneffouf, D., Rish, I., & Aggarwal, C. (2020). Survey on Applications of Multi-Armed and Contextual Bandits. *2020 IEEE Congress on Evolutionary Computation (CEC)*, 1–8. https://doi.org/10.1109/cec48606.2020.9185782

Chawla, A., Manjhi, G., & Gaurav, B. (2021). *Implications of Banking Regulations on Online Payment Failures*. Munich Personal RePEc Archive. https://mpra.ub.uni-muenchen.de/105285/1/MPRA_paper_105285.pdf

Cheng, F., Liu, J., Amin, K., & Wellman, M. P. (2016). Strategic Payment Routing in Financial Credit Networks. *Proceedings of the 2016 ACM Conference on Economics and Computation*, 721–738. https://doi.org/10.1145/2940716.2940738

Comșa, I.-S., Zhang, S., Aydin, M., Kuonen, P., Trestian, R., & Ghinea, G. (2019). A Comparison of Reinforcement Learning Algorithms in Fairness-Oriented OFDMA Schedulers. *Information*, *10*(10), 315. https://doi.org/10.3390/info10100315

Duò, M. (2022). *Complete List of Credit Card Declined Codes in 2023*. Kinsta. https://kinsta.com/blog/list-of-credit-card-declined-codes/

Durand, A., Achilleos, C., Iacovides, D., Strati, K., Mitsis, G. D., & Pineau, J. (2018). Contextual Bandits for Adapting Treatment in A Mouse Model of De Novo Carcinogenesis'. *Machine Learning for Healthcare Conference*, 1–15. https://doi.org/http://proceedings.mlr.press/v85/durand18a/durand18a.pdf

ElectrifAi. (2020). *Portfolio Optimization with Machine Learning*. ElectrifAi Blog. https://www.electrifai.com/blog/portfolio-optimization-with-machine-learning

Flutterwave. (n.d.). *Common bank transaction errors*. Flutterwave. https://flutterwave.com/gb/support/general/common-bank-transaction-errors

Fong, D. (2021). *Seven Technologies Shaping the Future of Fintech*. McKinsey & Company. https://www.mckinsey.com/cn/our-insights/our-insights/seven-technologies-shaping-the-future-of-fintech

Gambo, I., & Agbonkhese, C. (2024). Software Design Specification Proposal of a Diagnostic Decision Support System for Clinical Low Back Pain. *Procedia Computer Science*, *231*, 243–250. https://doi.org/10.1016/j.procs.2023.12.199

Gambo, I., Ogundare, T., Magreola, M., & Ajayi, T. (2023). Refining Blockchain-Powered Industrial Internet-of-Things Architecture for Improved Performance. *2023 Fifth International Conference on Blockchain Computing and Applications (BCCA)*, 122–129. https://doi.org/10.1109/bcca58897.2023.10338910

Gifford, D. K., Stewart, L. C., Payne, A. C., & Treese, G. W. (1995). Payment Switches for Open Networks. *Digest of Papers. COMPCON'95. Technologies for the Information Superhighway*, 26–31. https://doi.org/10.1109/cmpcon.1995.512359

Hambly, B., Xu, R., & Yang, H. (2021). *Recent Advances in Reinforcement Learning in Finance*. *33*(3), 437–503.

Huo, X., & Fu, F. (2017). Risk-Aware Multi-Armed Bandit Problem with Application to Portfolio Selection. *Royal Society Open Science*, *4*(11), 171377. https://doi.org/10.1098/rsos.171377

Li, H., Yang, X., & Li, D. (2021). A Reinforcement Learning Approach for Portfolio Optimization with Mean Regret. *Expert Systems with Applications*, *171*, 114533.

Lu, H., Liyan, H., & Hongwei, Z. (2013). Credit Scoring Model Hybridizing Artificial Intelligence with Logistic Regression. *Journal of Networks*, *8*(1), 253–261. https://doi.org/10.4304/jnw.8.1.253-261

Mahajan, A., & Teneketzis, D. (2008). Multi-Armed Bandit Problems. In A. O. Hero, D. A. Castañón, D. Cochran, & K. Kastella (Eds.), *Foundations and Applications of Sensor Management* (1st ed., Vol. 5, pp. 121–151). Springer US. https://doi.org/10.1007/978-0-387-49819-5_6

Mohan, S. (2014). *A brief overview of the Multi-Armed Bandit in Reinforcement Learning*. Analytics Vidhya.

Nagendra, S., Podila, N., Ugarakhod, R., & George, K. (2017). Comparison of Reinforcement Learning Algorithms Applied to the Cart-Pole Problem. *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 26–32. https://doi.org/10.1109/icacci.2017.8125811

Olivier, C., & Lihong, L. (2011). An Empirical Evaluation of Thompson Sampling. *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, 2249–2257. https://doi.org/doi/10.5555/2986459.2986710

Parikh, S. (2019). *Why Do Online Payments Fail and how to handle? Razorpay Blog*. Razorpay. https://razorpay.com/blog/online-payments-failure-reasons/

Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., & Wen, Z. (2018). A Tutorial on Thompson Sampling. *Foundations and Trends® in Machine Learning*, *11*(1), 1–96. https://doi.org/10.1561/2200000070

Sahu, S. K., Mokhade, A., & Bokde, N. D. (2023). An Overview of Machine Learning, Deep Learning, and Reinforcement Learning-Based Techniques in Quantitative Finance: Recent Progress and Challenges. *Applied Sciences*, *13*(3), 1956. https://doi.org/10.3390/app13031956

Shafiq, S., Gambo, I., Asghar, M. A., Ibrahim, J., & Safdar, F. (2024). A Suggestive Framework for Handling Conflicts in Stakeholder's Emotional Goals Using Machine Learning Approach. *2024 International Conference on Engineering & Computing Technologies (ICECT)*, 1–7. https://doi.org/10.1109/icect61618.2024.10581078

Statista. (2023). *Visa, Mastercard market share in Nigeria 2022*. Statista. https://www.statista.com/statistics/1175891/distribution-of-credit-and-debit-cards-scheme-in-nigeria/

Stripe. (2023). *A complete list of card decline codes—and what each one means. Stripe Resources-Payments.* https://stripe.com/resources/more/a-complete-list-of-decline-codes

Sutton, R. S. (2018). Reinforcement learning: An Introduction. *A Bradford Book*, 1–13.

Yauney, G., & Shah, P. (2018). Reinforcement Learning with Action-Derived Rewards for Chemotherapy and Clinical Trial Dosing Regimen Selection. *Machine Learning for Healthcare Conference*, 161–226. https://doi.org/10.48550/arXiv.1811.09867

Zhang, J. (2019). *Reinforcement Learning — Multi-Arm Bandit Implementation Comparison between ϵ-greedy and UCB*. Towards Data Science. https://towardsdatascience.com/reinforcement-learning-multi-arm-bandit-implementation-5399ef67b24b

Zhou, J., Lai, X., & Chow, J. Y. J. (2019). Multi-Armed Bandit On-Time Arrival Algorithms for Sequential Reliable Route Selection under Uncertainty. *Transportation Research Record: Journal of the Transportation Research Board*, *2673*(10), 673–682. https://doi.org/10.1177/0361198119850457

Zhou, Q., Zhang, X., Xu, J., & Liang, B. (2017). Large-Scale Bandit Approaches for Recommender Systems. In D. Liu, S. Xie, Y. Li, D. Zhao, & E. S. M. El-Alfy (Eds.), *Neural Information Processing* (Vol. 10634, pp. 811–821). Springer. https://doi.org/10.1007/978-3-319-70087-8_83