Implementación de aprendizaje automático para la selección óptima de anuncios en páginas web

Ing. César Gastón Cárdenas Córdova

3 de noviembre de 2020

Índice

Índice de figuras

1. Introducción

El aprendizaje automático, aplicado a la selección óptima de anuncios en páginas web, es una buena alternativa para mejorar los ratios de conversión o tasa de clics cuando se quieren promocionar nuevos productos o campañas[1]. Este hecho ha conllevado a que algunas compañías incluyan este tipo de servicios de optimización en su portafolio de soluciones, como por ejemplo Optimizely, Dynamic Yield y Frosmo.

El aprendizaje por refuerzo (reinforcement learning) es la base principal para el desarrollo de esta tecnología. Este consiste, en términos generales, en la programación de agentes inteligentes que exploren las posibles acciones, para resolver un problema determinado, y registren (aprendan) las recompensas o penalidades obtenidas en cada una. En otras palabras, el sistema experimenta a través de prueba y error para alcanzar el mayor beneficio posible.[2]

En el presente trabajo, se propone el diseño de un servicio Restful básico, que implemente el algoritmo Upper Confidence Bound (UCB), para optimizar la selección de anuncios en un caso práctico definido. Asimismo, se muestra cómo este método en general aporta mejores beneficios que una solución tradicional, donde se seleccionan los anuncios de forma arbitraria, y bajo qué condiciones es más óptimo. A pesar que existen otros algoritmos que es posible aplicar; como por ejemplo: Thompson sampling, Epsilon greedy policy, etc; escapa fuera del alcance la revisión de los mismos, debido a que no es parte del objetivo encontrar la mejor solución de aprendizaje por refuerzo que resuelva el problema. Por el contrario, lo que se busca es brindar una solución mínimo viable que motive al lector a implementar machine learning, si se encuentra en una situación similar al escenario planteado, conociendo a priori las circunstancias en las que puede obtener mejores resultados.

Así pues, para lograr el objetivo propuesto, en primera instancia se define un caso de negocio sobre la promoción de un nuevo producto, que se desea optimizar. Luego, se explica a grandes rasgos en qué consiste el problema del bandido multibrazo y las similitudes con el caso práctico planteado. En tercer lugar, se detalla cómo funciona el algoritmo UCB y los parámetros que utiliza para resolver problemas de aprendizaje por refuerzo. Una vez que se tiene el modelo base de la solución, se analiza el contexto del problema para identificar las funcionalidades y los componentes necesarios para la implementación. Por último, es necesario determinar escenarios críticos, para poner a prueba al sistema y verificar los resultados obtenidos en la optimización.

[1] Cfr. Arumughom 2019 [1]
[2] Cfr. Silva 2019 [2]

2. Caso práctico

Supongamos que una empresa, que cuenta con diez mil usuarios, desea promocionar la venta de un nuevo producto, colocando un anuncio al momento que el cliente ingresa al portal web de la compañía. Bajo este escenario pueden ocurrir dos posibles eventos:

1. El usuario se siente interesado por el anuncio y accede al mismo, haciendo clic en un botón.

2. El usuario no está interesado en el anuncio y lo cierra.

Asimismo, se han propuesto tres posibles anuncios para incentivar la compra del nuevo producto. Como no es posible determinar a priori cuál de ellos influirá más en el cliente, se decide mostrar dinamicamente uno u otro a cada usuario que ingresa al portal. La figura 1 muestra la representación del escenario con las tres posibles alternativas de anuncio y las opciones que puede elegir el usuario, comprar o cerrar.

Fig. 1: Interfaz de ejemplo del anuncio en la página web

Por las características del caso, podría ser una alternativa emplear pruebas A/B[3]. Sin embargo, la compañía no cuenta con suficiente tiempo para establecer un periodo de prueba y necesita obtener la mayor cantidad de ventas posibles. Debido a esta necesidad, el departamento de tecnología decide emplear aprendizaje por refuerzo, para que la selección de los anuncios sea más óptima y automaticamente se seleccione el mejor de ellos con el paso del tiempo.

[3] Cfr. Rojas 2019 [3]

3. Problema del bandido multibrazo

El problema del bandido multibrazo o MAB, por sus siglas en inglés «Multi-armed bandit», se plantea en la teoría de probabilidad estadística con el siguiente enunciado: "Supongamos que en un casino existe un número K de maquinas tragamonedas, donde la probabilidad de ganar en cada una de ellas es diferente y desconocida. ¿Con qué máquinas jugarías y en qué orden para máximizar la suma de recompensas obtenidas?"[4]

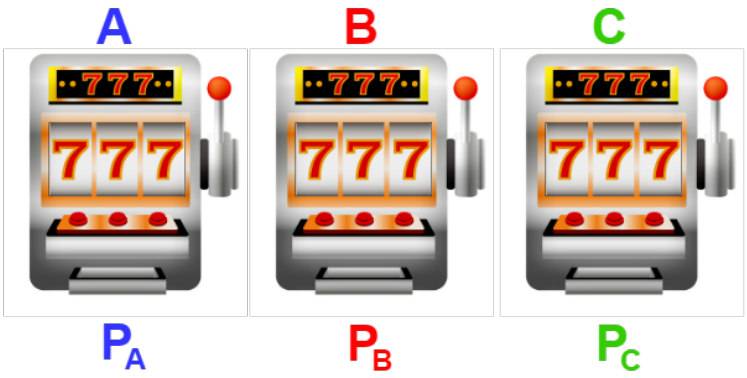

Fig. 2: Máquinas tragamonedas

Por ejemplo, si se tuvieran N fichas por jugar en las tres máquinas de la figura 2; cada una con probabilidad de éxito P_A, P_B y P_C; para obtener una ganancia X. Lo primero que podríamos pensar, para resolver el problema, es utilizar una fracción del total de fichas disponibles para tratar de determinar el porcentaje de éxito de cada máquina (exploración) y, luego, apostar el resto en la que se obtengan mejores resultados (explotación).

Como se puede apreciar, el problema descrito es muy similiar al caso práctico planteado en la sección anterior. Las máquinas tragamonedas estarían dadas por los anuncios que deseamos mostrar a los usuarios, mientras que las probabilidades de ganar el premio mayor en cada una de ellas sería equivalente al «ratio de conversión»[5].

La situación dilemática sobre cuánto explorar y explotar, para maximizar la ganancia, puede afrontarse con el aprendizaje por refuerzo. Esta, es una técnica de inteligencia artificial que consiste en la programación de agentes inteligentes[6] que exploren su entorno para identificar acciones que maximicen recompensas. Bajo este modelo, las primeras elecciones tomadas por el agente serán hasta cierto punto aleatorias. Pero, en cada caso, se

[4] Cfr. Wikipedia 2020 [4]

[5] Hace referencia al porcentaje de visitantes de un sitio web que realizan aquello que la página les invita a hacer. [5]

[6] Un agente inteligente es un ente con capacidad de percibir el entorno y tomar acciones aplicando un razonamiento o algoritmo determinado.

asignará una recompensa o penalidad; dependiendo de si fue una acción correcta o erronea, respectivamente. De esta forma, poco a poco se eligirán las que brindan mas beneficio, mientras se descarta el resto.[7]

Los algoritmos e implementaciones disponibles para simular este aprendizaje automático son muy variados. Por ejemplo, Markel Sanz Ausin en «Introducción al aprendizaje por refuerzo. Parte 1: el problema del bandido multibrazo.»[8] desarrolla ϵ-greedy policy con Python, Sourish Dey en «Leveraging Power of Reinforcement learning in Digital Marketing»[9] utiliza el UCB[10] también con Python, mientras que Christian Hubbs hace lo propio con lenguaje R en «Multi-Armed Bandits: UCB Algorithm».[11] Se puede ver que los lenguajes predilectos para hacer demostraciones rápidas son Python y R, pero es posible usar cualquier otro como Java, C#, Javascript, etc.

[7] Cfr. Sanz 2020a [6]
[8] Cfr. Sanz 2020b [6]
[9] Cfr. Sourish 2019 [7]
[10] Upper Confidence Bound
[11] Cfr. Hubbs 2020 [8]

4. Upper confidence bound

Upper confidence bound (UCB) o, en castellano, Límite de confianza superior es un algoritmo de aprendizaje por refuerzo determinista con un criterio de selección basado en el principio de optimismo frente a la incertidumbre. En otra palabras, esto significa que la política del algoritmo intentará explorar una opción menos conocida en cada elección, con el «optimismo» de que sea más beneficiosa que las opciones que ya conoce. Este proceso se repetirá hasta que el algoritmo determine cual de todas las que ha probado reiteradas veces ha tenido mayor éxito y descartará el resto.[12]

Por ejemplo, en la figura 3 se muestran los resultados que obtiene un agente al realizar 14 rondas de elecciones con el algoritmo UCB en un «bandido de 3 brazos»[13]. Cada marca roja indica que se obtuvo un resultado negativo, mientras que las marcas verdes representan recompensas y los números, asignados a cada círculo, los ordenes de cada «ronda»[14].

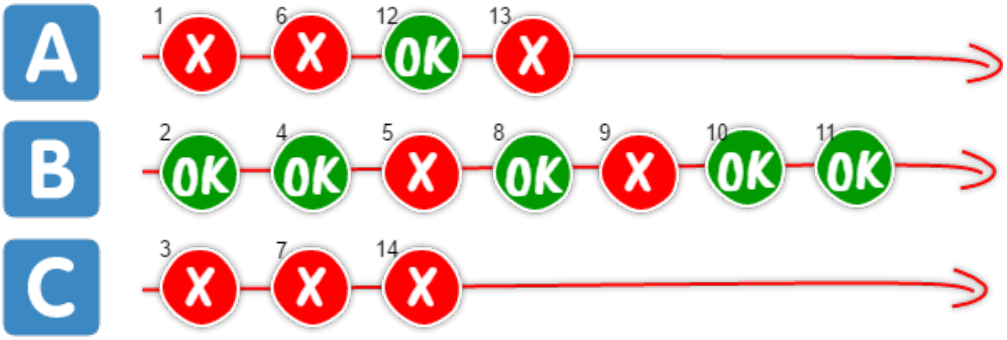

Fig. 3: Simulación de decisiones del UCB

Para entender cómo han ocurrido estás elecciones primero vamos a definir la matriz $n_{i,j}$, donde se van a contabilizar las veces en las que se elige cada opción i en la ronda j, para $j \in [1, N]$. Dado que en el ejemplo tenemos 14 muestras de las decisiones del algoritmo, se tomará ese valor para N. La siguiente matriz muestra cómo incrementa el conteo de cada opción elegida, a lo largo de las 14 muestras:

$$n_{i,j} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 3 & 4 & 4 \\ 0 & 1 & 1 & 2 & 3 & 3 & 3 & 4 & 5 & 6 & 7 & 7 & 7 & 7 \\ 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 3 \end{bmatrix}$$

Una vez que se tiene cada opción contabilizada, es necesario identificar cual es el beneficio probable que ofrecen en base a lo que se va explorando. Para representar esto, utilizaremos una matriz $P_{i,j}$ que indica el porcentaje de veces que se ha obtenido recompensas en la opción

[12] Cfr. Banditalgs 2016 [9]
[13] Hace referencia a un problema de bandido multibrazo con 3 brazos o 3 máquinas tragamonedas.
[14] En el contexto del problema del bandido multibrazo, cada ronda es el momento en que se debe elegir por una de las máquinas tragamonedas

i en la ronda j. Por ejemplo, en la siguiente matriz podemos ver que $P_{2,5} = 0{,}67$ porque la alternativa B fue seleccionada 3 veces y se obtuvo 2 recompensas ($\frac{2}{3} = 0{,}67$)

$$P_{i,j} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0{,}33 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0{,}67 & 0{,}67 & 0{,}67 & 0{,}75 & 0{,}6 & 0{,}67 & 0{,}71 & 0{,}71 & 0{,}71 & 0{,}71 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Si utilizaramos simplemente estas probabilidades, para determinar que decisiones tomar en cada caso, sería muy complicado saber cuando intentar explorar una nueva alternativa. Ya que, en las primeras elecciones $P_{2,j}$ obtiene altos porcentajes de recompensas y esto impediría que considere escoger A o C, basados unicamente en este criterio. Entonces, es necesario que el algoritmo tenga un «incentivo» para explorar opciones nuevas cada momento. El UCB resuelve este inconveniente utilizando la siguiente fórmula para calcular el indicador de selección:

$$\lambda_{i,j} = P_{i,j-1} + \sqrt{\frac{\gamma_{j-1}}{n_{i,j-1}}}$$

El valor de $\lambda_{i,j}$ es un indicador de confianza que representa qué tan beneficioso es seleccionar la alternativa i en la ronda j. Como se mencionó antes, $P_{i,j-1}$ sería la probabilidad de acierto en la elección anterior a j y $n_{i,j-1}$ la cantidad de veces que se eligió la alternativa i hasta el j anterior. De esta forma, al estar $n_{i,j-1}$ en el denominador de la función, el índice crecerá menos en la alternativa que se haya seleccionado mas veces que las otras. El parámetro $\gamma_{i,j}$ de la fórmula, es una función que crece de forma logarítmica a medida que se hacen más elecciones:

$$\gamma_j = 2 \ln j, \forall j \in [1, N]$$

$$\gamma_j = \begin{bmatrix} 0 & 1{,}39 & 2{,}20 & 2{,}78 & 3{,}22 & 3{,}58 & 3{,}89 & 4{,}16 & 4{,}39 & 4{,}61 & 4{,}80 & 4{,}97 & 5{,}13 & 5{,}29 \end{bmatrix}$$

Ahora, que se tienen identificados los datos necesarios para calcular el indicador de confianza, es posible estimar los valores que orientaron al algoritmo a tomar una u otra opción en cada ronda. Por ejemplo, los cálculos para $j = 6$ serían de la siguiente forma:

$$\lambda_{A,6} = P_{A,5} + \sqrt{\frac{\gamma_5}{n_{A,5}}} = 0 + \sqrt{\frac{3{,}22}{1}} = 1{,}7941$$

$$\lambda_{B,6} = P_{B,5} + \sqrt{\frac{\gamma_5}{n_{B,5}}} = 0{,}67 + \sqrt{\frac{3{,}22}{3}} = 1{,}7025$$

$$\lambda_{C,6} = P_{C,5} + \sqrt{\frac{\gamma_5}{n_{C,5}}} = 0 + \sqrt{\frac{3{,}22}{1}} = 1{,}7941$$

De estas tres alternativas el UCB eligirá la mayor, que en este caso podrían ser tanto la opción A como la opción C. Asimismo, se puede observar que en B, a pesar que el porcentaje de acierto es mayor, no llegó a superar en puntaje a las otras alternativas debido a ya fue elegida 3 veces en otras oportunidades.

j	Indicador UCB		
	$\lambda_{A,j}$	$\lambda_{B,j}$	$\lambda_{C,j}$
1	-	-	-
2	-	-	-
3	-	-	-
4	1.4823	2.4823	1.4823
5	1.6651	2.1774	1.6651
6	1.7941	1.7025	1.7941
7	1.3386	1.7596	1.8930
8	1.3950	1.8056	1.3949
9	1.4420	1.7697	1.4420
10	1.4823	1.5375	1.4823
11	1.5174	1.5428	1.5174
12	1.5485	1.5420	1.5485
13	1.6204	1.5569	1.5764
14	1.3825	1.5703	1.6015

Tab. 1: Indicador de criterio de selección UCB

En las primeras elecciones, el UCB intentará explorar todas las opciones al menos una vez, antes de empezar a tomar el valor de $\lambda_{i,j}$. Esto puede verse en la tabla 1, donde se encuentran, marcados en color verde, los indicadores que representan el nivel de confianza máximo en cada una de las 14 rondas del caso de ejemplo, planteado al inicio de esta sección.

En este apartado se ha logrado explicar a grandes razgos cómo funciona el UCB para encontrar el «brazo» óptimo, en un problema de bandido multibrazo. A pesar que la implementación del algoritmo no es compleja y puede ser replicada en cualquier lenguaje de programación, en este trabajo se empleó el paquete UCB de Kurth Ericson para Node.Js, disponible en NPMjs.

5. Implementación

En esta sección, se describe la propuesta para implementar aprendizaje automático, con el algoritmo UCB, en la selección óptima de anuncios. Para esto, se parte del supuesto que la aplicación web de la compañía ya está desarrollada con todas las funcionalidades para ofrecer y vender productos. Asimismo, los anuncios que van a mostrarse, que en este caso vendrían a ser archivos de imágenes, también ya han sido diseñados y están listos para usarse en la página web.

En la figura 4 se muestra el proceso que se sigue al momento de mostrar el banner publicitario al usuario. La secuencia consta de 7 pasos, que se detallan a continuación:

1. El usuario ingresa al portal web para ver los productos.
2. Al cargar la página se invoca al servicio para la selección óptima del anuncio.
3. El servicio utilizará el algorimo UCB para determinar el anuncio más adecuado, de acuerdo a los indicadores de éxito/fracaso que tiene registrado.
4. Se carga la imagen del anuncio seleccionado por el servicio.
5. El usuario al ver el anuncio tiene dos opciones, cerrarlo o ir a la ventana de compra.
6. Para cualquiera de los dos casos la web invocará a un método que registre la selección del usuario.
7. El servicio registra las recompensas y penalidades para actualizar los indicadores del algoritmo UCB.

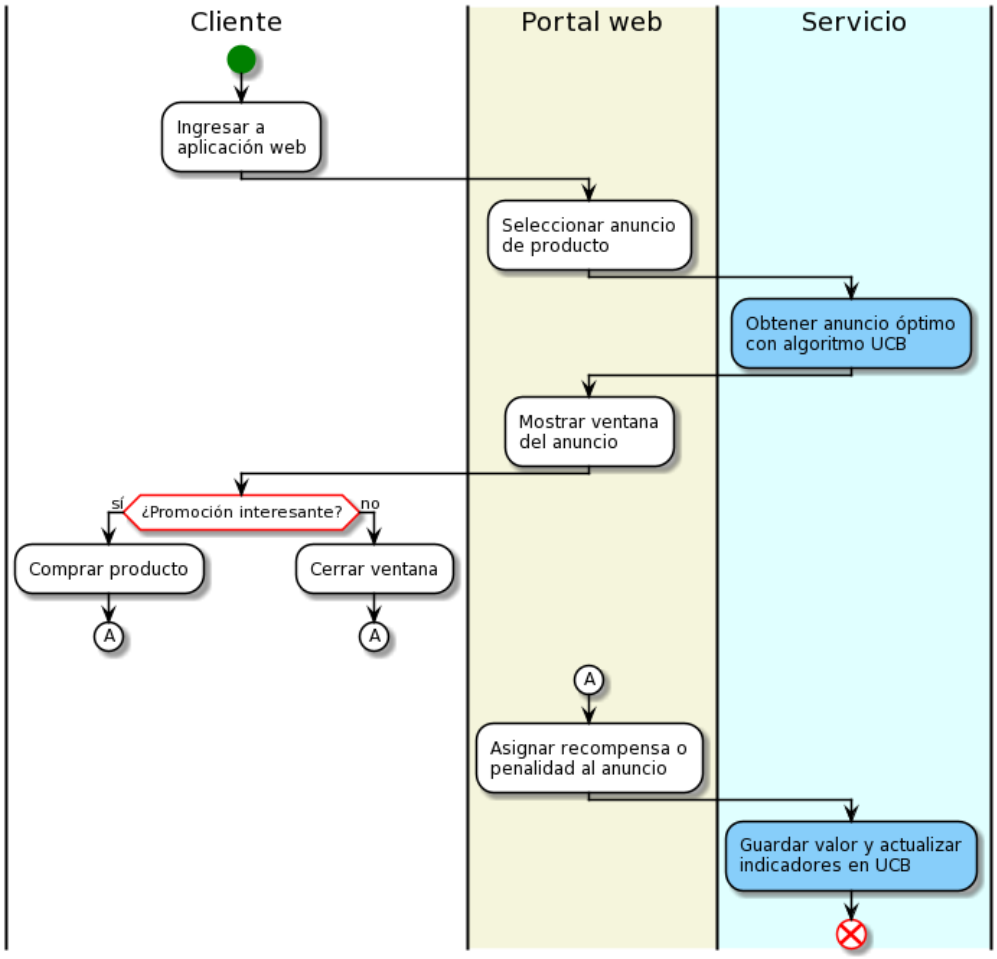

Fig. 4: Flujo de proceso de la funcionalidad

Las tareas marcadas en celeste del gráfico 4 representan las tareas que debe llevar a cabo el agente inteligente. Como se puede observar, quien hace uso del servicio no es el cliente directamente, sino el portal web. Por tanto, este vendría hacer el actor para los dos casos de uso definidos, tal como muestra la figura 5.

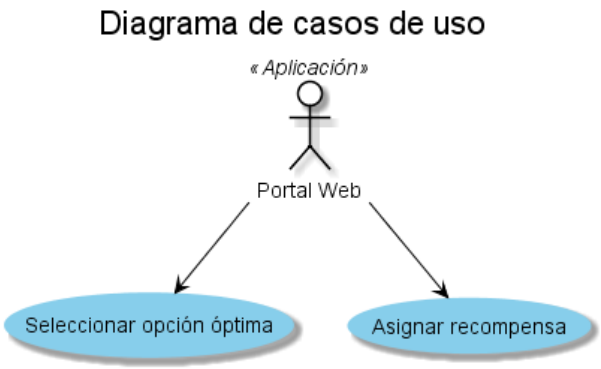

Fig. 5: Casos de uso del servicio

Para cumplir con los requerimientos identificados, se necesitarán basicamente tres componentes. En primer lugar, el módulo con la implementación del algoritmo UCB, el servicio que expondrá las funcionalidades a la aplicación cliente y, finalmente, el componente de persistencia; para guardar el estado de los indicadores del algoritmo. En el siguiente gráfico se puede apreciar la interacción de cada uno de estos componentes.

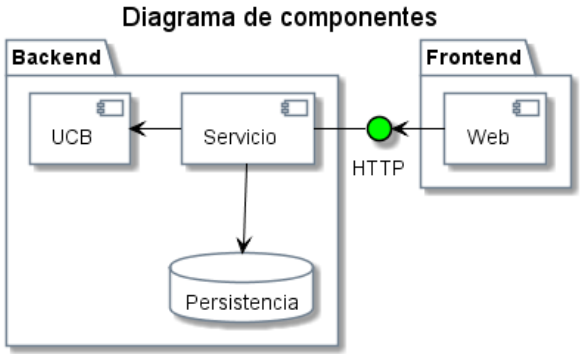

Fig. 6: Diseño de componentes propuesto

- **UCB:** Para este componente se propone la utilización del paquete UCB para Node.js.
- **Servicio:** Es el encargado de exponer los métodos del agente y de interactuar con el componente UCB. En este caso se plantea el uso del framework web Fastify para Node.js, aunque es posible usar cualquier otro disponible para ese entorno de ejecución, como por ejemplo: Express, Koa, etc.
- **Persistencia:** Es el encargado de guardar y actualizar los niveles de confianza de cada brazo en todo momento, esto para evitar que se pierdan los valores en caso de un reinicio del servicio o un redespliegue. En este caso se optó por usar MongoDB, pero también puede emplearse otra base de datos basada en documentos o SQL.

Hasta este momento, se tienen definidos los casos de uso y los componentes necesarios para implementarlos. Ahora, se detalla cómo estos intervienen en la secuencia pasos necesarios para llevar a cabo la optimización.

- **Inicialización:** Cuando el servicio se reinicia por un despliegue o caída, es necesario verificar si existen parámetros del algoritmo UCB en base de datos, para asignarlos al momento de instanciar el objeto. Esto con el fin de que el sistema continue el aprendizaje donde se quedó. Obviamente, la primera vez que inicie el sistema no existirá información previa y empezará a aprender desde cero.

- **Obtener anuncio:** Para obtener el anuncio óptimo, se emplea el método select del paquete UCB. Este devuelve un número que hace referencia al índice del brazo que se está seleccionando. Como se tienen tres anuncios diferentes A, B y C; los posibles valores (brazos) que se podrán retornar serán 0, 1 y 2. Luego, la parte front se encargará de interpretar este valor y mostrar la imagen del anuncio correspondiente.

- **Asignar recompensa:** Cada vez que se muestra un anuncio, el cliente tiene dos opciones, comprar el producto o cerrar la ventana, según sea el caso se asignará una recompensa positiva o negativa, respectivamente. Para esto, el servicio empleará el método reward del paquete UCB, que se encargará de actualizar los indicadores de éxito de cada opción y retornalos. Asimismo, este recibe como parámetros el número de brazo, que representa el indicador del anuncio que se está evaluando, y un número binario, donde 0 indica que el usuario cerró la ventana y 1 que accedió a comprar el producto. Cuando se tenga la respuesta del método, esta se guardará en base de datos, para que pueda ser consultada posteriormente en caso de un reinicio o redespliegue del servicio.

La representación gráfica de estos procesos puede visualizarse en la figura 7.

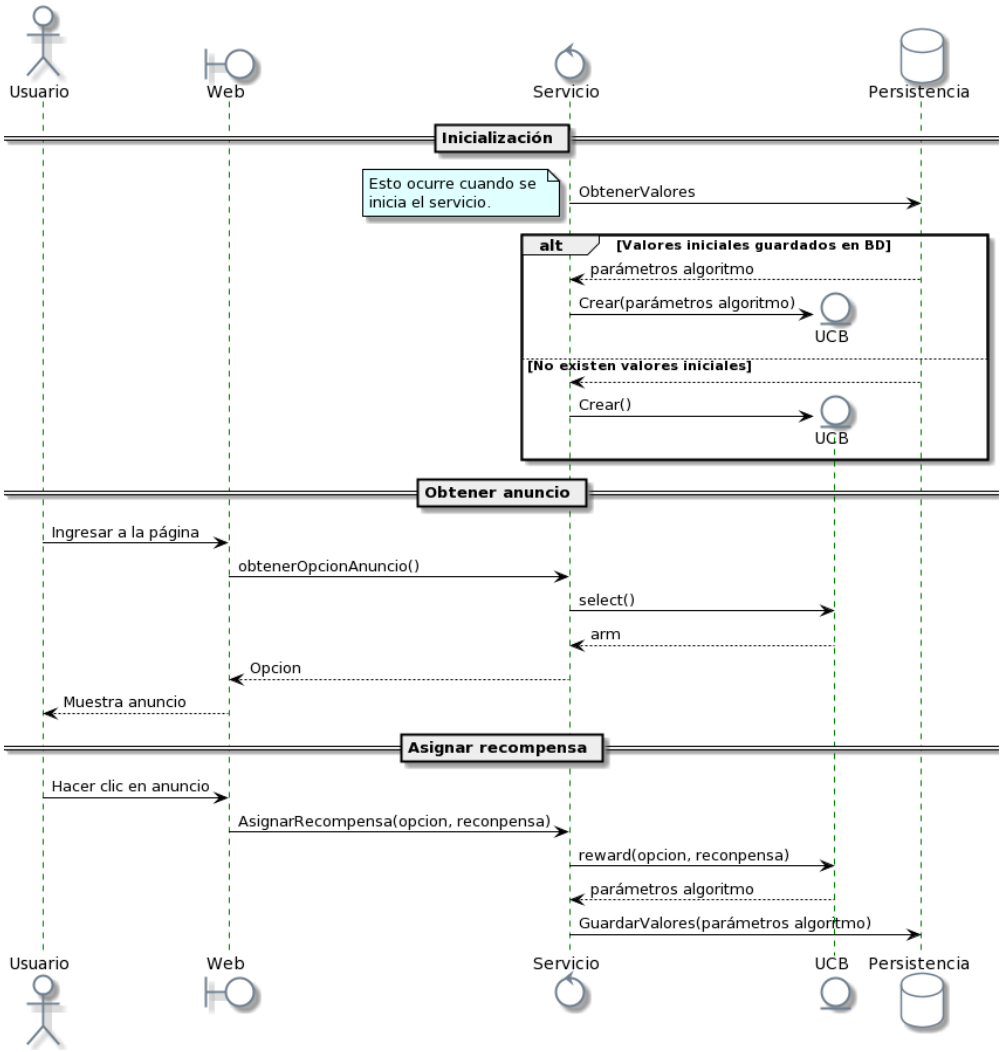

Fig. 7: Implementación de paquete UCB para Node.js

Esta implementación propuesta, se encuentra dispobible en un repositorio Github, para servir de base en desarrollos similares, con otros lenguajes de programación, o para extender la funcionalidad a problemas mas complejos.

6. Pruebas

En este apartado, se definen los escenarios de prueba que permitan evaluar el comportamiento de la solución bajo diferentes contextos. Para esto se han considerado como casos más críticos las situaciones en las que los anuncios tienen ratios de conversión muy cercanos o distantes unos de otros. Es decir, si la mayor cantidad de conversiones se dan en mas de uno de forma similar, se esperaría que las selecciones, aplicando la política UCB, estén continuamente entre ellos y solo eventualmente se intente explorar el de menor ratio. Por otro lado, el caso opuesto a este sería que todos los anuncios tengan porcentajes de conversión muy variados entre sí, haciendo que pueda ser más fácil determinar el óptimo.

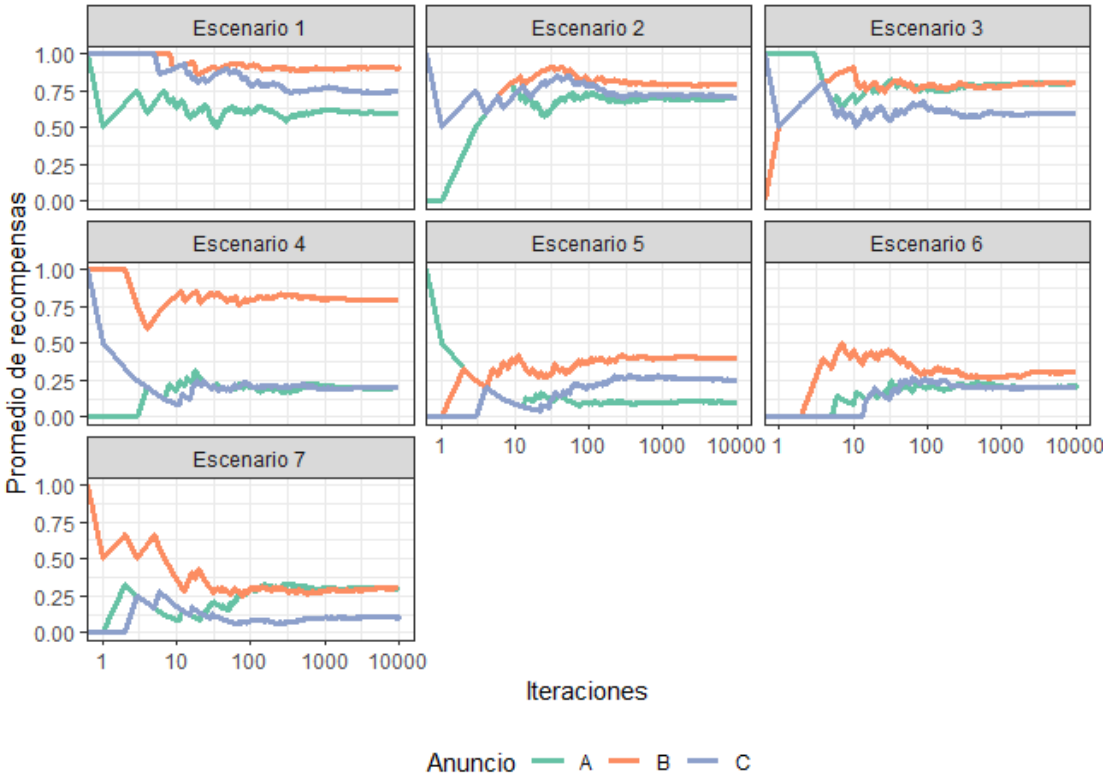

Fig. 8: Comparación de recompensas promedio por escenario

Para la creación de cada uno de estos escenarios, se consideró la simulación de 10,000 elecciones por anuncio con diferentes probabilidades de éxito para obtener recompensas. Por ejemplo, el gráfico 8 muestra el comportamiento del promedio de recompensas por anuncio a lo largo de las 10,000 rondas o iteraciones. En el escenario 1 y 5 se puede apreciar que la efectividad de cada alternativa está separada casi proporcionalmente, alrededor 0.15 entre uno y otro. Por otro lado, en el caso 4 la opción B tiene un rendimiento de

recompensas predominante frente las demás. En el resto de escenarios los promedios son mas cercanos, incluso en determinadas rondas los valores llegan hacer casi identicos.

Bajo estos siete casos se pondrán a prueba las elecciones de anuncios, aplicando los criterios de selección aleatoria y UCB. Para esto, se han generado 7 archivos de cada uno, donde se realizan $10,000$ selecciones utilizando ambas políticas. En la tabla de la ilustración 9 se muestran las primeras 5 rondas de las elecciones de prueba aleatorias. Las columnas A, B y C contienen valores de 1 ó 0, que indican si el anuncio debe retornar recompensa o no, respectivamente, en esa iteración. Además, la quinta columna (Selected) representa la opción seleccionada con 0, 1 y 2; haciendo referencia a los anuncios A, B y C, repectivamente. Esto quiere decir, por ejemplo, que en la quinta iteración se decidió elegir la azar la alternativa C, obteniendo beneficio positivo en ese caso. Asimismo, en las columnas siguientes se presenta el conteo total de selecciones y recompensas por anuncio en cada ronda.

Iteration	A	B	C	Selected	CountA	CountB	CountC	PickA	PickB	PickC
0	1	1	1	1	0	1	0	0	1	0
1	0	1	1	2	0	1	1	0	1	1
2	1	1	1	2	0	1	2	0	1	2
3	1	1	1	0	1	1	2	1	1	2
4	0	1	1	0	2	1	2	1	1	2
5	1	1	1	2	2	1	3	1	1	3

Fig. 9: Datos de prueba

La estructura presentada aplica también para los casos de prueba de la política UCB. Todos ellos pueden ser consultados en el siguiente enlace.

7. Resultados

En este apartado se prensentan los diferentes resultados obtenidos en cada escenario, divididos en tres secciones.

En la primera parte, se verifica si el algoritmo UCB logró identificar la opción más óptima, elegiendola más veces, y se compara esto frente a las elecciones arbitrarias. Luego, se analizan los beneficios[15] conseguidos en cada criterio de selección, para determinar las condiciones en las que es más probable maximizar la ganancia. Finalmente, en la última parte se examina el comportamiento de los aspectos anteriores a lo largo de las 10,000 iteraciones, para tratar de identificar las rondas mínimas necesarias que permitan hallar el mejor anuncio y, al mismo tiempo, obtener un margen de beneficio aceptable en comparación a una política aleatoria.

7.1. Determinación del mejor anuncio

Como era de esperarse, en general la solución logró identificar los anuncios que proporcionan mayor recompensa en cada caso y los seleccionó con más frecuencia. En el escenario 4 se obtuvo una de las situaciones ideales, ya que se eligió casi el 100 % de las veces la opción B, que es más eficiente para conseguir recompensas.

Los escenarios 3 y 7, donde las opciones A y B son las mejores con eficiencia similar, podrían considerarse más críticos porque ponen a prueba la capacidad del algoritmo para identificar la mejor alternativa. En ambos, el anuncio A obtuvo una cantidad de elecciones mayor, como puede verse en el gráfico 10. Sin embargo, mientras en el primero se tiene una diferencia de aproximademante 500 en las cantidades, en el segundo esto es casi el triple. Este comportamiento es llamativo porque, a pesar que ambos casos tienen situaciones similares, no se consiguen resultados equivalentes.

Con respecto a las selecciones aleatorias, se puede ver que la distribución es casi equitativa entre todos los anuncios, provocando pérdida de oportunidad al elegir opciones deficientes demasidas veces.

[15] Cantidad de recompensas totales.

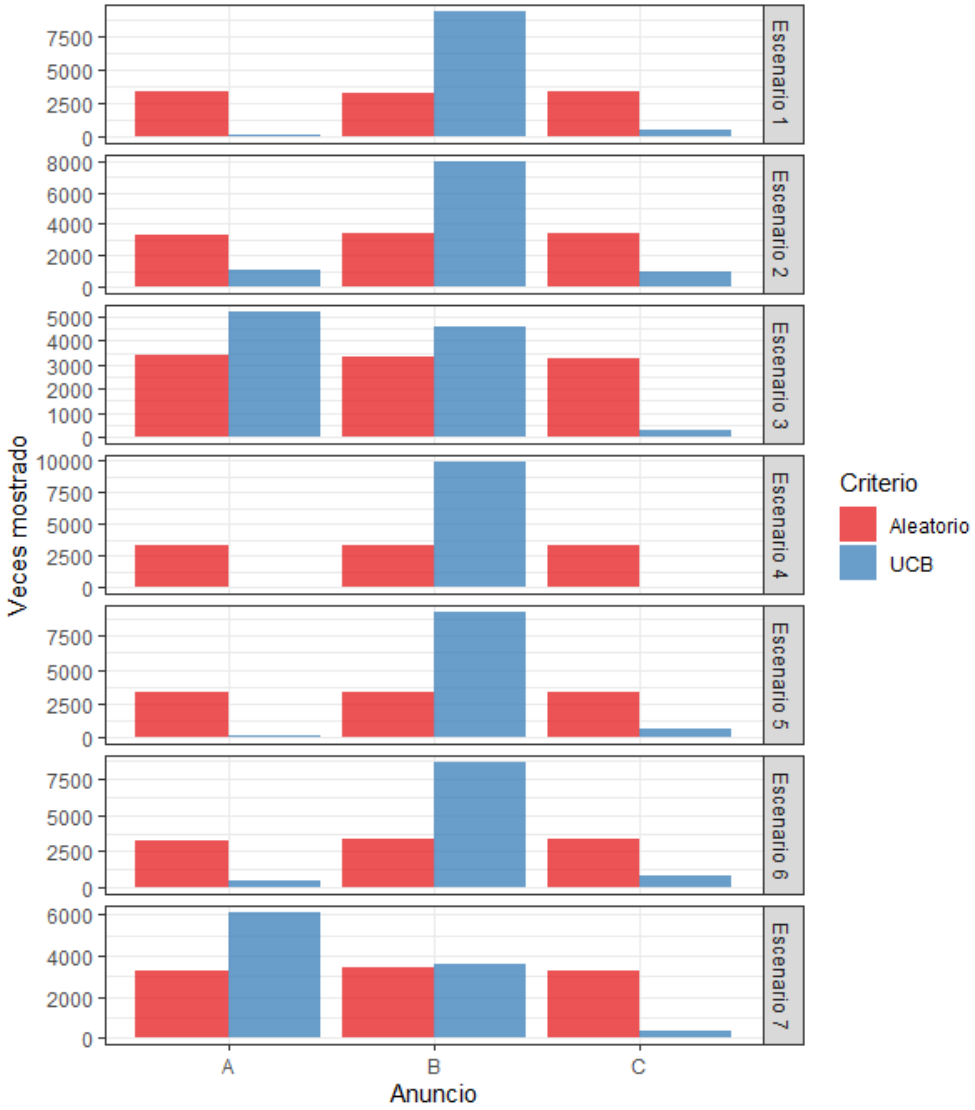

Fig. 10: Comparación de selección de anuncios usando UCB y aleatoriedad

7.2. Constraste de recompensas por criterio de selección

En esta sección se analizan las diferencias en los beneficios proporcionados por cada estrategia de selección. En la figura 11, se puede apreciar el comportamiento de las recompensas por criterio para cada escenario. En la mayoría de ellos se ha conseguido una mejora de alrededor de 500 conversiones mas, que indicarían que la optimización está funcionando realmente. Sin embargo, también se puede notar que en los casos 1, 5 y sobre

todo en el 4 el UCB ha logrado obtener una ventaja considerablemente mayor, casi 1,500 en los primeros y 3,800 en el último.

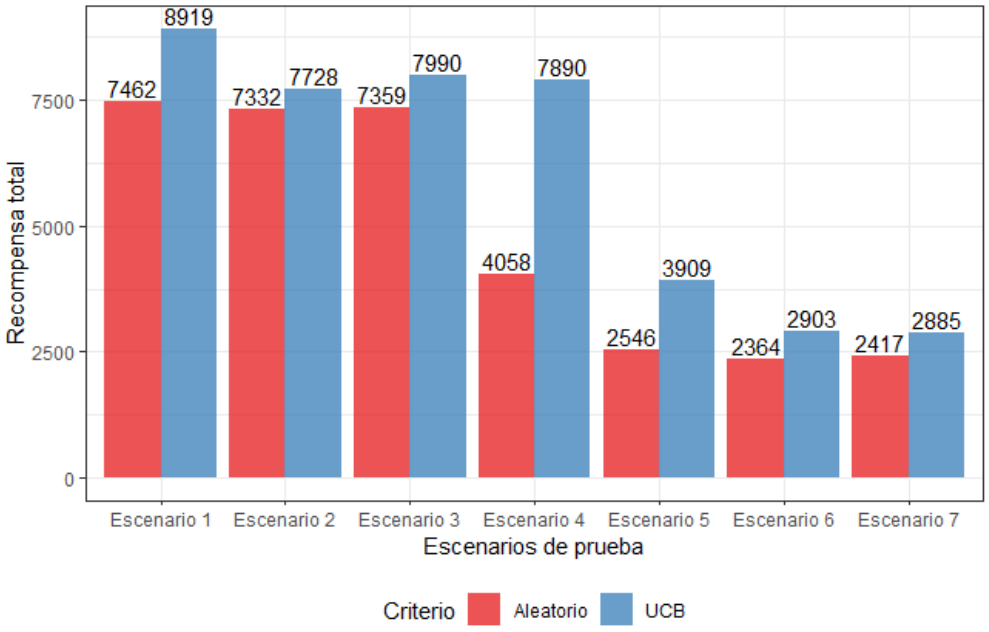

Fig. 11: Comparación de beneficios usando UCB y aleatoriedad

Con esto se podría confirmar que la mejora funciona satisfactoriamente. Pero, a pesar de eso, otro punto a tener en cuenta son las condiciones que provocaron que el beneficio sea mayor o menor en determinadas circunstancias. Con la finalidad de descubrir este aspecto, se ha elaborado la tabla de la figura 12, donde se presentan diferentes factores de cada escenario, como por ejemplo: el ratio de conversión promedio de los anuncios, el ratio máximo de ellos, la desviación estándar, la conversión obtenida de cada una de las estrategias y la diferencia entre ambas.

Escenario	Media	Máximo	Desviación estándar	Conversión UCB	Conversión Aleatorio	Mejora
Escenario 1	0.7490000	0.9037	0.1528836	0.8917	0.7520	0.1397
Escenario 2	0.7353333	0.8002	0.0561890	0.7805	0.7333	0.0472
Escenario 3	0.7362667	0.8032	0.1142904	0.7981	0.7379	0.0602
Escenario 4	0.4003667	0.8021	0.3479118	0.7963	0.4025	0.3938
Escenario 5	0.2471000	0.3976	0.1491683	0.3850	0.2446	0.1404
Escenario 6	0.2320667	0.3006	0.0594796	0.2803	0.2300	0.0503
Escenario 7	0.2308000	0.2965	0.1123267	0.2872	0.2252	0.0620

Fig. 12: Tabla de resultados obtenidos en cada escenario

Es probable que alguno de estos factores pueda estar directamente relacionado con los ratios de conversión o con la mejora resultante. Para corroborar este punto, se puede calcular la matriz de correlación de todos ellos, para medir el grado de correspondencia que tienen entre si. Estos valores pueden variar de -1 a 0 (grado de relación inversa) o de 0 a 1 (relación directa).

En el gráfico 13 se muestra una parte la matriz de correlación, representada de forma triangular debido a que se obviaron las relaciones inversas de los factores y las relaciones entre si mismos. Además, se encuentran marcados en circulos las correspondencias que son significativamente más relevantes, por tener un valor mayo a 0.6.

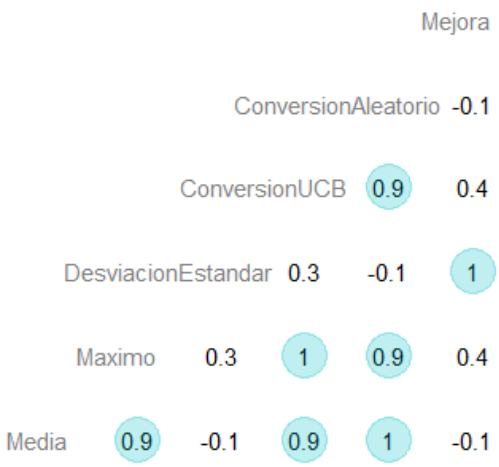

Fig. 13: Nivel de correlación entre las variables de los escenarios

Se puede observar que, mientras la media de conversión de anuncios o el máximo sean mayores, se incrementarán los beneficios sin importar que método de selección se utilice. Por otro lado, a medida que la desviación estándar aumenta; es decir, que los ratios de conversión estén mas alejados de la media; será más beneficioso emplear aprendizaje por refuerzo, en comparación a una estrategia arbitraria, para optimizar la selección de anuncios, debido a que la diferencia de las recompensas se incrementará.

7.3. Comportamiento de selecciones por iteración

Como se mencionó anteriormente, para las pruebas se consideraron 10,000 simulaciones, debido a que ese fue el total de clientes que se estableció en el caso práctico. Sin embargo, cabe cuestionar cuáles hubieran sido los resultados con una cantidad inferior de los mismos. Para responder esta interrogante, se optó por comparar el comportamiento del promedio de las recompensas, para cada estrategia de selección, a lo largo de las 10,000 rondas, como muestra la imagen 14.

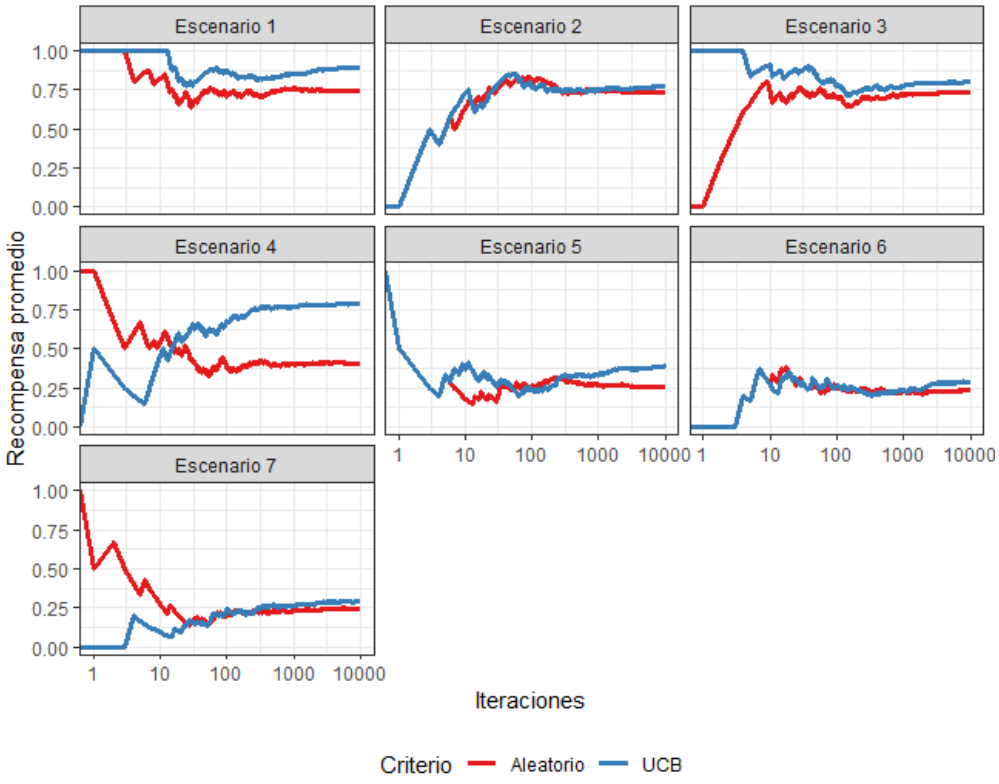

Fig. 14: Promedio de recompensas por iteración

En el gráfico se puede ver cómo, para la mayoría de los escenarios, las recompensas se estabilizan entre las rondas 100 y 300. Además, se aprecia que, a partir de ese instante en los casos 5 y 7, se empiezan a distanciar los beneficios de ambos criterios de selección. Por otro lado, los escenarios 2 y 6, donde la desviación estándar es menor ($\approx$ 0,06), se requirieron mas de 1000 rondas antes de que el UCB empiece a diferenciarse de la estrategia arbitraria. Un comportamiento opuesto ocurre con el caso 4, que tiene la desviación típica mas alta ($\approx$ 0,35), donde la diferenciación ocurrió alrededor de la trigésima iteración.

Adicionalmente, es posible evaluar el comportamiento de las selecciones de cada anuncio a lo largo de las rondas y ver si se corresponde con lo visto en las recompensas. Para esto, en la figura 15, se representan graficamente los porcentajes de selección de cada una de las alternativas, en las iteraciones para los diferentes escenarios. En este, se puede destacar que en general la identificación de la opción más eficiente ocurre también entre las iteraciones 100 y 300. Uno de los casos excepcionales es el escenario 3, donde las alternativas A y B son igualmente óptimas y se evidencia que ambas se seleccionaron cerca del 50 % de las veces hasta las rondas finales. Asimismo, en el caso 7, que tiene la misma característica de contar con más de una opción eficiente, no ocurre eso y la elección de A aumenta a partir de la iteración 300 hasta llegar aproximadamente al 62 %, mientras que B baja y se mantiene cerca del 33 %.

Un aspecto curioso que ocurre con el escenario 2 es que, a pesar que se identifica tempranamente la alternativa B como la mejor, las recompensas finales llegan a ser muy similares a las que se obtienen utilizando un criterio aleatorio. Esto se debe a que las otras opciones, al tener porcentajes de conversión muy cercanos[16], llegan a ser casi igual de eficientes que la óptima.

[16] Esto puede corroborarse en la figura 8, donde se evidencia promedios de recompensa muy cercanos entre los 3 anuncios a partir de la ronda 100

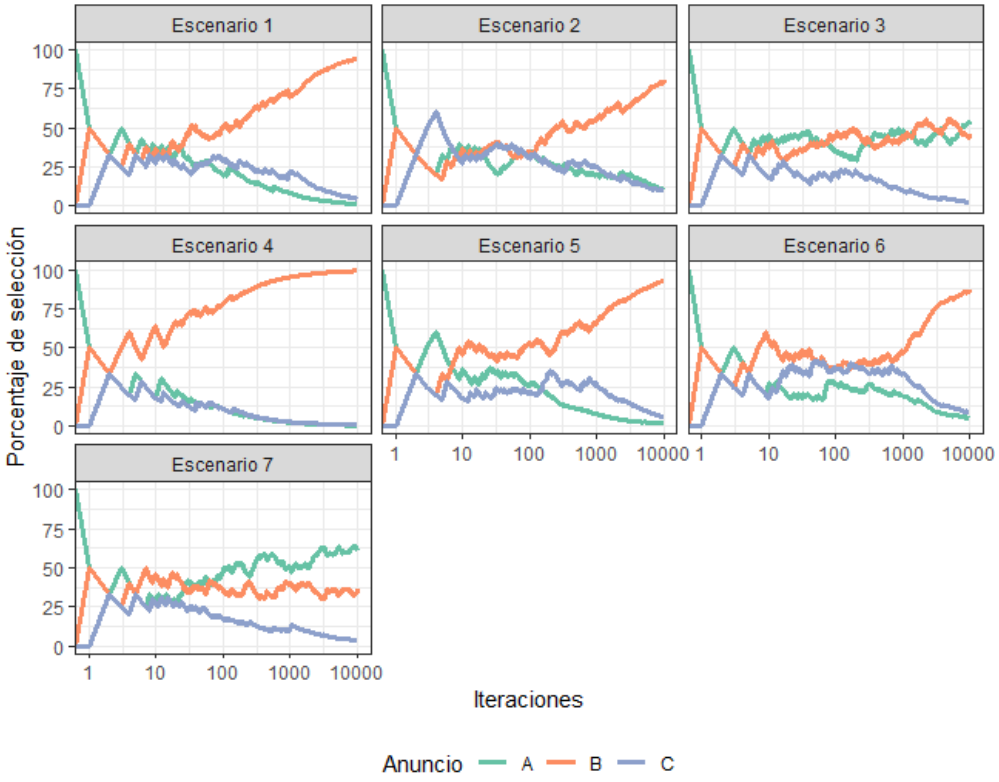

Fig. 15: Porcentaje de seleccion de anuncios por iteración

8. Conclusiones

El objetivo principal de este trabajo, es proponer una solución de aprendizaje automático para resolver un caso práctico del problema MAB[17] de 3 brazos. Para esto, se planteó el diseño básico de un servicio que permita a cualquier página web, donde se requiera promocionar un producto, hacer uso del aprendizaje por refuerzo para seleccionar el mejor anuncio y, de esta forma, mejorar las ventas en comparación a una selección aleatoria.

Para el desarrollo del servicio se utilizaron las tecnologías Node.js con Fastify y MongoDB. Sin embargo, es posible emplear cualquier otro framework web, lenguaje de programación o base de datos que convenga según el caso. Ya que, como se ha explicado, el algoritmo UCB tiene una implementación sencilla y es facilmente replicable. También, podría optarse por utilizar algún otro paquete con un algoritmo diferente de aprendizaje por refuerzo, y obtener beneficios similares o mayores.

Los resultados obtenidos en las pruebas han sido favorables en comparación a una estrategia arbitraria. Ya que, mientras que esta se aproxima al promedio de los ratios de conversión, el UCB tiende al máximo de ellos, permitiendo la optimización. Además, gracias a que el aprendizaje se da en tiempo real, se evita la inversión de tiempo exclusivo para exploración, que normalmente se requeriría en pruebas A/B.

Asimismo, es necesario considerar que existen ciertas circunstancias en las que el UCB no será capaz de obtener resultados superlativos. Como se logró identificar en los escenarios de prueba 2, 3, 6 y 7; cuando las alternativas de elección tienen ratios de conversión cercanos entre si, los beneficios que se obtienen son muy similares a los de una estrategia aleatoria. Para evitar esto en la práctica, es ideal que las opciones o anuncios sean totalmente diferentes unos de otros, para que las conversiones sean variadas. En segundo lugar, otro punto que pudo evidenciarse es que en general el algoritmo necesitó al menos entre 100 y 300 rondas para determinar el mejor aviso publicitario. Como el sistema aprende «sobre la marcha», es probable que esto quede a medias si se tienen interacciones por debajo de ese intervalo, provocando que los resultados no sean los deseados.

El aprendizaje por refuerzo, aplicado a la optimización de anuncios en páginas web, es capaz de aportar grandes beneficios a la conversión en una promoción o campaña. Además, su desarrollo e implementación es relativamente rápido, si no existen restricciones de infraestructura podría tomar alrededor de dos semanas. Finalmente, implantar la solución es la única forma de saber con certeza si se obtendrán o no mejoras; debido a que la efectividad del algoritmo depende mucho de la cantidad de clientes y la probabilidad de éxito de los anuncios, que dificilmente pueda conocerse a priori.

[17] Haciendo referencia al problema del bandido multibrazo (en inglés Multi-armed bandit)

Referencias

[1] D. Arumughom, "Multi-armed bandit algorithms for website optimization," https://browsee.io/blog/bandit-algorithms-for-website-optimization/, Noviembre 2019, [Online; accedido 23-Mayo-2020].

[2] M. Silva, "Aprendizaje por refuerzo: Introducción al mundo del rl," https://medium.com/aprendizaje-por-refuerzo-introducci%C3%B3n-al-mundo-del/aprendizaje-por-refuerzo-introducci%C3%B3n-al-mundo-del-rl-1fcfbaa1c87, Abril 2019, [Online; accedido 02-Mayo-2020].

[3] J. Rojas, "A/b testing: una buena forma de medir el éxito de tus campañas," https://velogig.com/a-b-testing-mide-tus-acciones-de-marketing/, Mayo 2019, [Online; accedido 09-Abril-2020].

[4] Wikipedia, "Multi-armed bandit," https://en.wikipedia.org/wiki/Multi-armed_bandit, Octubre 2020, [Online; accedido 17-Octubre-2020].

[5] Workana, "¿qué es el ratio de conversión?" https://www.workana.com/i/glosario/ratio-de-conversion/, Octubre 2020, [Online; accedido 17-Octubre-2020].

[6] M. Sanz, "Introducción al aprendizaje por refuerzo. parte 1: el problema del bandido multibrazo," https://medium.com/@markelsanz14/introducci%C3%B3n-al-aprendizaje-por-refuerzo-parte-1-el-problema-del-bandido-multibrazo-afe05c0c372e, Marzo 2020, [Online; accedido 09-Abril-2020].

[7] D. Sourish, "Leveraging power of reinforcement learning in digital marketing," https://towardsdatascience.com/leveraging-power-of-reinforcement-learning-in-digital-marketing-d373c88a39ab, Junio 2019, [Online; accedido 10-Agosto-2020].

[8] C. Hubbs, "Multi-armed bandits: Ucb algorithm," https://towardsdatascience.com/multi-armed-bandits-ucb-algorithm-fa7861417d8c, Enero 2020, [Online; accedido 09-Abril-2020].

[9] banditalgs.com, "The upper confidence bound algorithm," https://banditalgs.com/2016/09/18/the-upper-confidence-bound-algorithm/, Setiembre 2016, [Online; accedido 09-Abril-2020].