

Algoritmo para Ajustar Markups Dinámicamente

Abstracto

Durante tres semanas he estado trabajando sobre este proyecto donde he implementado, ajustado y evaluado un algoritmo que ajusta dinámicamente los markups de un distribuidor hotelero. Con un objetivo final de optimizar las peticiones en ventas y maximizar beneficios. A partir de una lectura exhaustiva del código original y migración de PHP a Python para poder trabajar con ello, se han desarrollado varios generadores de datos diarios realistas. Estos generadores incorporan una función que se basa en la probabilidad de compra, el cual se ve afectada por la comisión de una agencia, histórico de ventas, estacionalidad y localización del hotel.

Además, he entrenado varios modelos de machine learning sobre un dataset de 100000 registros donde se valora la atractividad del hotel, dependiendo del mes y distancia al centro urbano. Entre los cuales he integrado el mejor modelo pero con la condición de que no afectara con mucha agresividad sobre la lógica principal.

A continuación, a partir de los gráficos y estadísticos he visto que hay una pequeña mejoría cuando hay muchas peticiones y pocas ventas. El simulador y entorno donde he realizado las pruebas me han facilitado ver que había una pequeña perdida dentro de la función calcularValor.

Metodología

Para mejorar dicho algoritmo tuve que seguir varios pasos estructurados, que incluyeron el análisis del código existente, la generación de datos, el diseño del simulador, el entrenamiento y la incorporación de modelos de Machine Learning.

1. Análisis del Código Existente

Durante estos primeros lleve a cabo una lectura detallada del código para entender el funcionamiento interno del algoritmo y cómo se recopilan los datos. Sin embargo, había

conceptos que no estaban bien documentados y datos que no se definían con claridad. Por este motivo, estuve analizando el código línea a línea para poder entender las diferentes menciones a conceptos. Esto dificultó la compresión inicial de los datos y la lógica que aplica la empresa.

2. Construcción del Simulador

Una vez analizado los datos, empecé a pasar la lógica del código de php a python usando VS Code como IDE para poder realizar mis pruebas. En primer lugar, programé varios generadores de datos que simulan las peticiones, ventas y precios realizadas cada día por las diferentes agencias. El proceso de generar los datos, se basaba inicialmente en:

- Establecer precios netos por proveedor y hotel.
- Simular las ventas diarias en base a markups iniciales.
- Generar las peticiones por agencia, respetando relaciones reales con hoteles y proveedores.

Sin embargo, estos fueron ajustados para que actuaran con más realismo, añadiéndoles más variables como:

- Diferencia entre la comisión media de la agencia y el markup
- Diferencia entre markups diarios
- Diferencia entre peticiones diarios
- Mes de compra

El simulador principal está basado en una lógica secuencial diaria de hasta un año, donde se usan los datos de días anteriores para ajustar los precios y simular el comportamiento de las agencias sobre los cambios diarios. Para ello, se usaron valores predeterminados en los dos primeros días para permitir la convergencia del sistema.

3. Desarrollo del Algoritmo de Ajuste

El algoritmo principal de ajuste de descuentos se basa en:

- Construir un peso entre cliente proveedor

- Calcular las comisiones medias históricas
- Determinar descuentos iniciales cuando no hay datos
- Diferencia entre el beneficio de hoy y ayer
- Ajustar con una función calcularValor las situaciones de alto interés y pocas ventas
 - Basada en una transformación logarítmica para suavizar las situaciones como muchas peticiones y baja conversión.
- Ajustar en base a la atractividad predecida por un modelo de machine learning
- Controlar la agresividad del algoritmo mediante límites

4. Entrenamiento del modelo

Para poder añadir variables como la temporada del año o ubicación del hotel, use un generador de 100000 registros simulando el atractivo de cada hotel según estos parámetros:

- Distancia al centro urbano
- Si es centrico
- Mes del año

Entrené varios modelos como RandomForestRegressor, HistGradientBoostingRegressor y LGBMRegressor, seleccionando finalmente LGBMRegressor por su rendimiento ($R^2 = .$). El valor que predice el modelo se usa como una señal que ajusta levemente los markups ya ajustados.

Resultados

Para poder evaluar la efectividad del cambio en la función calcularValor y variables añadidas, realice múltiples simulaciones de 365 días, registrando los datos diarios de ventas y beneficios.

1. Evolución de los Descuentos

Los prints de evolución diaria de los markups con el ajuste logarítmico muestran que los cambios son progresivos, con ajustes más agresivos en situaciones de baja conversión.

- peso-3.gz

```
Agencia002: {2: 1.9, 4: 2.5, 5: 2.67, 8: -2.21, 11: 2.77, 12: 0.65, 14: 2.86, 15: 1.5,
```

- peso-4.gz

```
Agencia002: {2: 1.99959, 4: 2.5, 5: 2.67, 8: -2.20687, 11: 2.77024, 12: 0.65, 14: 2.86,
```

2. Relación Peticiones – Ventas – Beneficio

Uno de los indicadores más importantes dentro del algoritmo es el beneficio por petición (BP). El problema estaba durante los escenarios con pocas ventas y alto interés, que provocaba que el BP fuera extremadamente bajo, teniendo una casi nula reacción sobre el algoritmo.

- calcularValor() Antiguo, Dia 4

```
0.00012964301519398144  
0.00012964301519398144  
0.00012964301519398144  
8.642867679598763e-05
```

- calcularValor() Logaritmico, Dia 4

```
0.014279188985971457  
0.012494290362725025  
0.008924493116232161  
0.00024493116232161
```

Como podemos apreciar el BP se volvió más interpretable y balanceado, respondiendo el algoritmo de forma más coherente ante desequilibrios. Con esto conseguimos incentivar que se vendan hoteles con muchas peticiones y pocas ventas.

3. Efecto del Modelo Predictivo de Atractividad

El modelo de machine learning es más conservador sobre los markups en hoteles con predicciones altas (8–10), mientras que en hoteles considerados poco atractivos (3–5), se bajan los markups para facilitar la venta y compensar la mala ubicación temporada. El ajuste debe ser entre este rango (-0.15, 0.15) y actúa como un “fine-tuning” estratégico del algoritmo principal. Los modelos usados para entrenar el dataset datosDatasets.csv

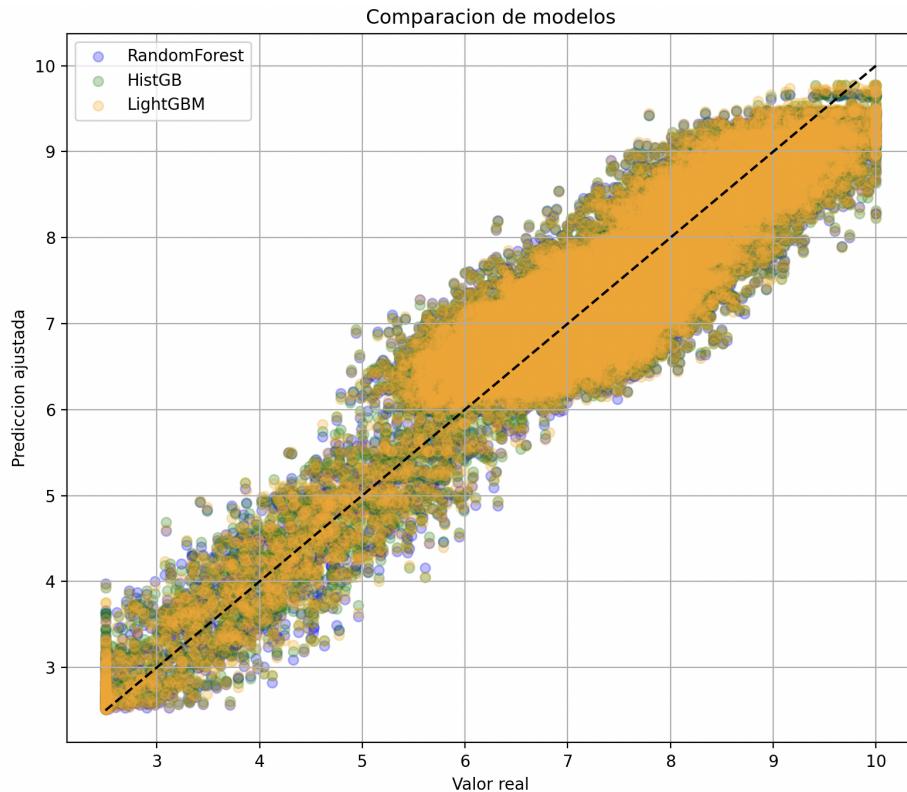
```
hotel_id,dia_del_ano,distancia,es_centrico,mes,target
9239,313,3.61,1,11,3.68
580,178,2.28,1,6,9.29
1160,214,3.44,1,8,3.24
```

Se generó un dataset de 100.000 registros y se dividió en un 80 % para entrenamiento y un 20 % para validación para una evaluación objetiva del modelo. Se evaluaron distintos modelos de regresión supervisada, entre ellos:

- RandomForestRegressor
- HistGradientBoostingRegressor
- LGBMRegressor

Tras ajustar parámetros y comparar su rendimiento en el conjunto de validación, se seleccionó finalmente LGBMRegressor. Este modelo consiguió un coeficiente de 0.8763, mostrando una buena capacidad para capturar la variabilidad del objetivo.

- Grafica comparacion de modelos



- Resultados de los modelos

```
Mejor modelo: LightGBM (R2 = 0.8763)
Modelo guardado en /Users/nikiliao/Desktop/Projects/Hotel Pricing Model
o/descuentos_optimizador/datos/pkl/fijos/modeloFinal.pkl
```

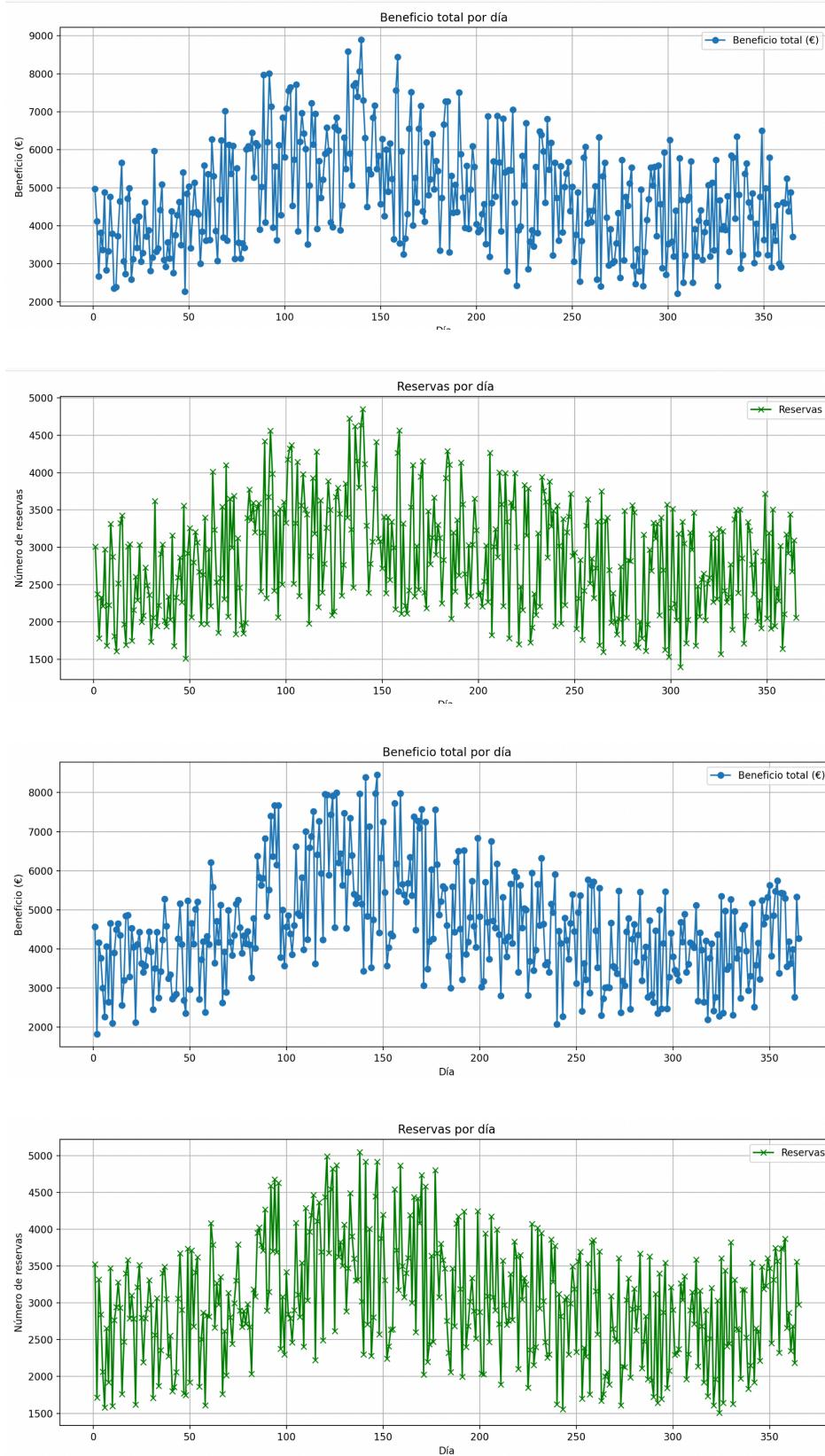
Resumen comparativo de metricas por modelo:

Modelo	MAE	RMSE	R2
RandomForest	0.392	0.492	0.8762
HistGB	0.392	0.492	0.8763
LightGBM	0.392	0.492	0.8763

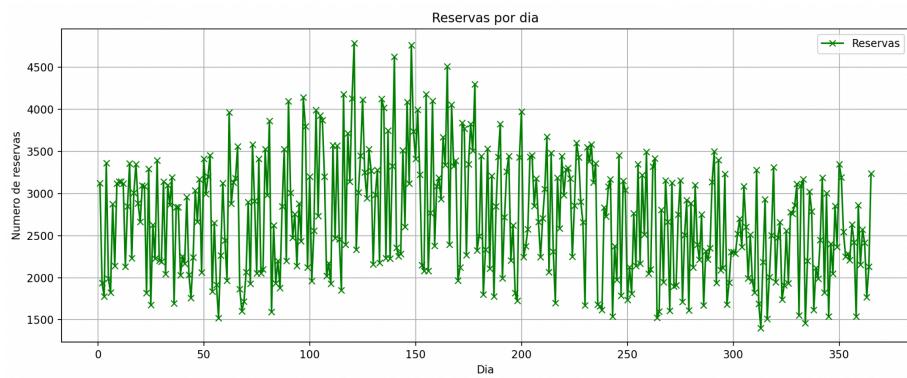
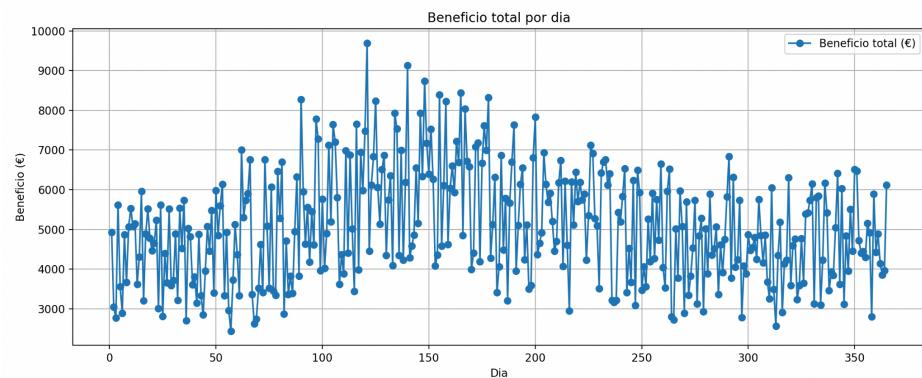
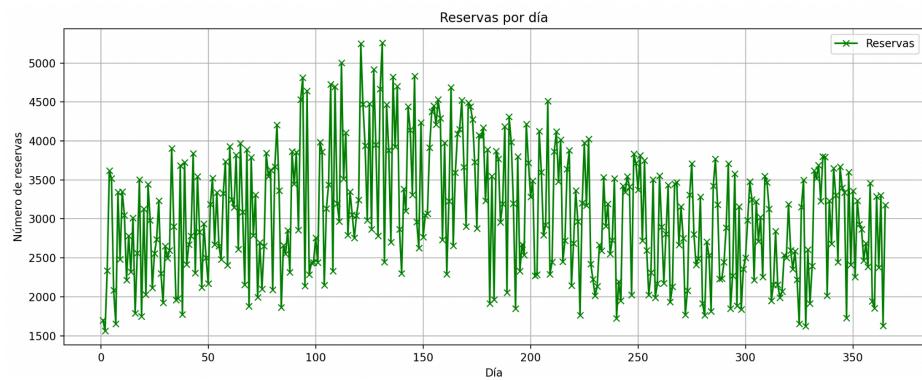
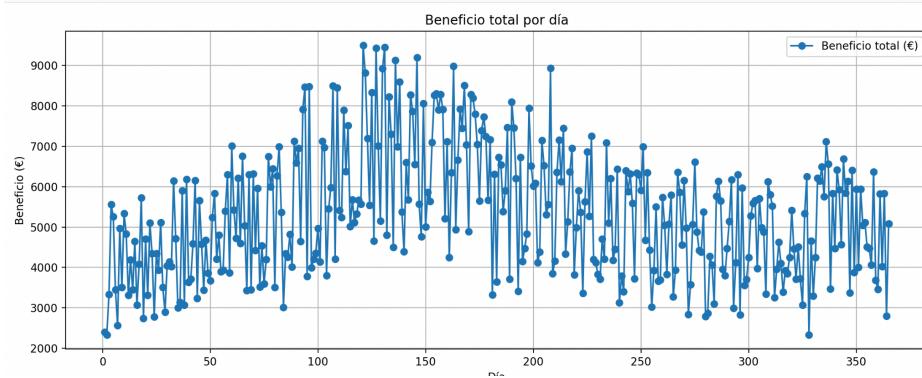
4. Comprobaciones Visuales y Métricas

Después de realizar el simulador de 365 días varias veces, el simulador registra los beneficios y ventas diarias. Posteriormente, usando matplotlib nos permite ver los gráficos y analizar con claridad el comportamiento de cada variante del algoritmo. Los gráficos muestran que los tres algoritmos ajustan correctamente los markups, con resultados positivos y realistas. Sin embargo, podemos ver que con los dos últimos algoritmos se obtiene un beneficio global mayor, y las ventas son inferiores. La diferencia entre el segundo y tercer algoritmo es mínima y habría que realizar más simulaciones, pero el problema es que una simulación del tercer algoritmo tarda unas 10-12 horas en completarse.

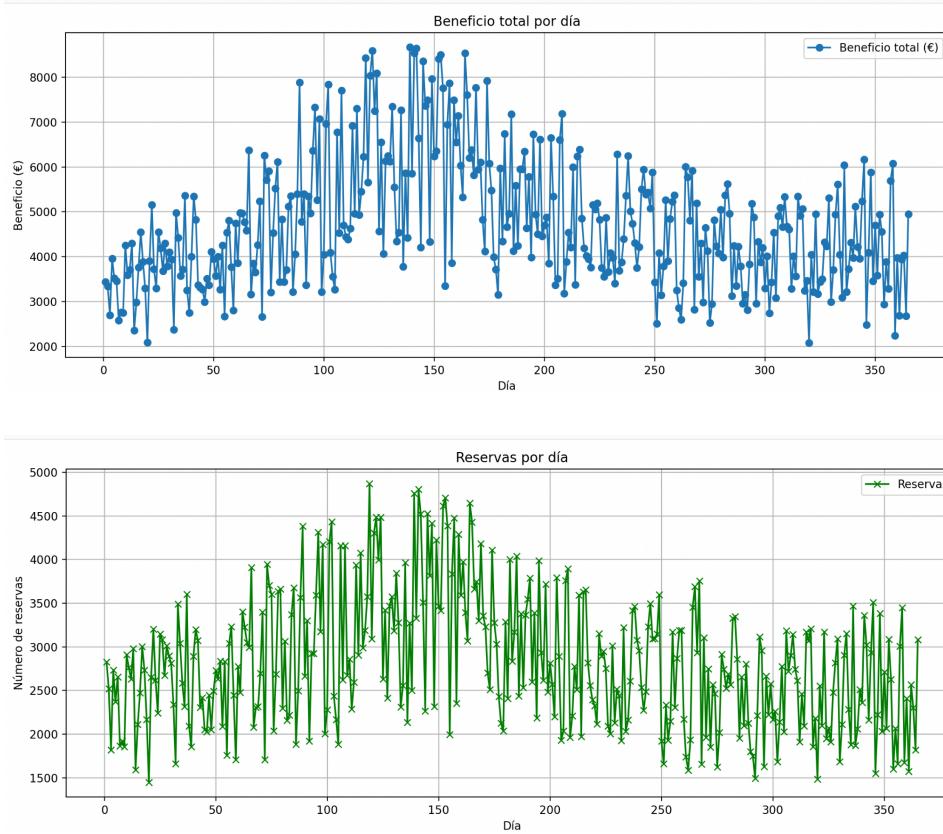
- Beneficio y ventas diarias del algoritmo (normal)



- Beneficio y ventas diarias del algoritmo (logarítmico)



- Beneficio y ventas diarias del algoritmo (algorítmico y machine learning)



- Estadísticos del algoritmo (normal)

Ventas total acumulado: 1,050,566.00
 Ventas media por dia: 2,878.26
 Beneficio total acumulado: 1,651,689.41
 Beneficio medio por dia: 4,525.18

Ventas total acumulado: 1,069,153.00
 Ventas media por día: 2,929.19
 Beneficio total acumulado: 1,694,705.06
 Beneficio medio por día: 4,643.03

- Estadísticos del algoritmo (logarítmico)

Ventas total acumulado: 1,048,891.00
 Ventas media por día: 2,873.67
 Beneficio total acumulado: 1,934,637.38
 Beneficio medio por día: 5,300.38

Ventas total acumulado: 990,103.00
Ventas media por dia: 2,712.61
Beneficio total acumulado: 1,847,432.03
Beneficio medio por dia: 5,061.46

- Estadísticos del algoritmo (algorítmico y machine learning)

Ventas total acumulado: 1,028,874.00
Ventas media por día: 2,818.83
Beneficio total acumulado: 1,721,996.92
Beneficio medio por día: 4,717.80

Conclusiones

Puedo concluir que el nuevo algoritmo ajusta adecuadamente los markups y ayuda a suavizar los extremos. El simulador me ha facilitado el entendimiento del algoritmo y la realización de pruebas exhaustivas. Los resultados obtenidos son muy prometedores pero habría que evaluarlos en un entorno real con datos reales. Así se podría verificar si la adaptación recomendada verdaderamente maximiza los beneficios. Además, este algoritmo está calibrado sobre unos datos generados artificialmente y habría que determinar si requiere de ajustes adicionales, ya que los rangos de ventas, peticiones y probabilidades de conversión no son al 100% fieles a la realidad ya que hay otros muchos factores.

Referencias

<https://www.airdna.co/blog/vacation-rental-metrics-booking-lead-time>