Trabajo Práctico 1 — Checkpoint 2

[75.06/95.58] Organización de Datos
Segundo cuatrimestre de 2023

Grupo 15

Ayala, Tomás Gabriel - tayala@fi.uba.ar - 105336
Giacobbe, Juan Ignacio - jgiacobbe@fi.uba.ar - 109866
Olaran, Sebastian - solaran@fi.uba.ar - 109410

Docente corrector:
Pereira, Francisco

Índice

1. Introduccion

Para este checkpoint 2, con el objetivo de aplicar el análisis de datos previamente hecho en el checkpoint 1, es armar un árbol de decisión. Para empezar, para que el árbol pueda interpretar el dataset obtenido en el checkpoint anterior tuvimos un nuevo preprocesamiento de los datasets, realizando cambios en los mismos. Para empezar, a algunas de las columnas categóricas (hotel, meal, market segmnet, distribution channel, customer type y deposit type) les aplicamos One Hot Encoding para que puedan ser procesadas por el árbol. Sin embargo, no todas las variables fueron transformadas de esta manera, debido a que generaríamos una cantidad muy grande de columnas, aumentando así el costo computacional de una manera drástica para nuestros modelos. Para evitar eso, utilizamos Ordinal Encoding para las variables country, assigned room type, reserved room type y arrival date month. Con estos cambios empezamos a construir nuestros árboles.

2. Construcción del modelo

Nuestros árboles fueron construídos con una previa optimización de hiperparámetros mediante el algoritmo de Random Search, y además los entrenamos de manera supervisada, es decir, usamos el dataset de train(para el cual ya sabíamos los resultados correctos para la variable target). Fuimos viendo que, para los conjuntos de entrenamiento y validación, el árbol presentaba unas métricas bastante buenas(casi siempre nos daba para ambos conjuntos un F1 score de más de 0.80). Sin embargo, a la hora de realizar nuestras predicciones sobre el dataset de test, y posteriormente subirlo a la competencia de Kaggle, observamos que nuestros árboles no pasaban del 0.50 para la métrica buscada, a pesar de haber obtenido un alto valor para nuestros conjuntos de entrenamiento. Esto se debió a que nuestros árboles de decisión presentaron un grave problema de sobreajuste (overfitting).

Al graficar nuestros árboles pudimos ver que tenían una tendencia a ramificarse"excesivamente. Esto se debía a la cantidad de features que se estaban utilizando para entrenar el modelo(para estos primeros árboles habíamos decidido usar todas las variables del dataset como feature para entrenar los modelos). Buscando solucionar el problema, volvimos a revisar los datos y el análisis obtenidos a partir del checkpoint 1. La idea era encontrar una forma de solucionar nuestro problema de overfitting y reducir el ruido en nuestras predicciones. Esto nos permitió dar un salto en la calidad de nuestra predicciones: de pasar de menos 0.50 a más de 0.70 de F1 score en la competencia de Kaggle. Las features seleccionadas dentro de este grupo del modelo son

- hotel.
- Lead time.
- Arrival date month.
- Stays in weeekend nights and stays in week nights.
- adults, children and babies
- country
- Customer type
- market segment
- is repeated guest, previous cancellations y previous booking not cancelled
- reserved room type, assigned room type
- booking changes, days in the waiting list y adr
- required car parking spaces y total of special requests

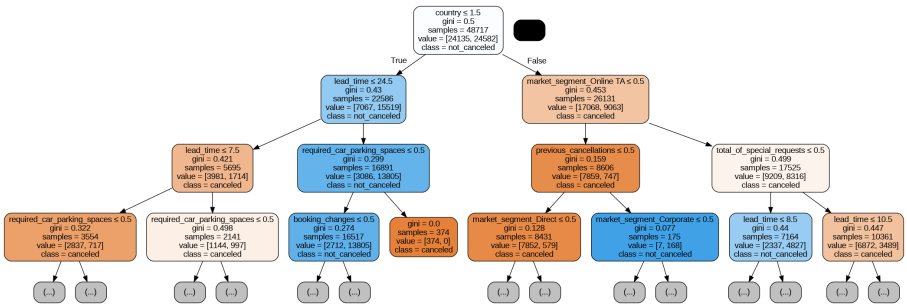
ACLARACIÓN: por cuestiones de tiempo, decidimos tomar al árbol que nos dió 0.78 de F1 score en la competencia(denominado "quinto árbol.en nuestro notebook), y sobre ese mismo realizamos el análisis que se nos pide en el trabajo práctico. Esto se debió a que el árbol con mejor performance(que nos dió un 0.80 de F1 score lo obtuvimos en fechas muy cercanas a la entrega).

Para la construcción de nuestro mejor predictor no solo se utilizaron las features anteriormente mecionadas, sino que también hubo un proceso de optimización de hiperparámetros que nos permitieron llegar a nuestro mejor arbol(con mejor performance). Para empezar, para la optimización de hiperparámetros utilizamos el algoritmo de Random Search, debido a que nos permitía abarcar una buena cantidad de hiperparámetros y tenía un costo computacional mucho más bajo que el algoritmo de Grid Search(este algoritmo era mucho más costoso computacionalmente). Los parámetros que estuvimos investigando fueron el criterion (en general alternando entre entropia y gini), min sample leaf, min samples split, el max depth y el cpp alpha.

Para ello tambien utilizamos k-fold cross validation. Estuvimos experimentando dentro de los rangos 5 a 12 pero observamos que el que dio los mejores resultados fue utilizar 10 folds. Utilizamos el rango de 5 a 12 folds a la hora de realizar los algoritmos porque queríamos reducir el costo computacional(a más folds, mayor coste).

Debido a que en la competencia de Kaggle se nos estuvo evaluando con la metrica F1, utilizamos esta misma técnica para poder optimizar los hiperparámetros. Desde el modelo inicial al modelo final hubo grandes saltos en las métricas. Comenzando por el modelo inicial con una métrica de 0.36998 hasta nuestro mejor predictor de 0.78963 en la competencia.

A continuación podemos observar una imagen de los primeros 3 niveles de nuestro arbol mas significativo.



Podemos notar y desarrollar las principales reglas:

1. Si country es menor o igual a 1.50 Esta es la regla principal que inicia el proceso de toma de decisiones en el árbol. Se evalúa el atributo country, que representar un código de país. Si el valor de country es menor o igual a 1.50, el árbol procede a la siguiente regla.

2. Si lead_time es menor o igual a 24.50 En esta regla, el árbol evalúa el atributo lead_time, que representa la cantidad de días de antelación con la que se realizó la reserva. Si el valor de lead_time es menor o igual a 24.50, lo que sugiere que la reserva se hizo con poca anticipación, el árbol sigue por esta rama.

3. Si total_of_special_requests es menor o igual a 0.50 Aquí se considera el atributo total_of_special_requests, que representa el número de solicitudes especiales realizadas por el cliente. Si el valor de total_of_special_requests es menor o igual a 0.50, el árbol continúa evaluando la siguiente regla.

4. Si booking_changes es menor o igual a 0.50 En esta regla, el árbol observa el atributo booking_changes, que representa la cantidad de cambios realizados en la reserva. Si el valor de booking_changes es menor o igual a 0.50, lo que indica pocos cambios en la reserva, el árbol sigue por esta ruta.

5. Si lead_time es menor o igual a 210.50 La última regla en esta secuencia considera nuevamente el atributo lead_time, pero ahora con un valor de corte diferente. Si lead_time es menor o igual a 210.50, el árbol continúa por esta rama.

Estas reglas representan una parte fundamental del proceso de toma de decisiones del modelo basado en el árbol de decisión. Cada una de ellas evalúa características específicas de la reserva, como el país de origen, el tiempo de antelación, las solicitudes especiales y los cambios en la reserva. La secuencia de estas reglas guía el flujo de decisiones del modelo y finalmente lleva a una decisión o predicción sobre si un cliente realizará o no una reserva en el hotel. Estas reglas proporcionan una visión valiosa de cómo el modelo interpreta y utiliza la información para tomar decisiones.

3. Cuadro de Resultados

A continuacion vemos un cuadro de resultados de 3 nuestros modelos mas representativos

Modelo	F1-Test	Precision Test	Recall Test	Accuracy	Kaggle
Peor predictor	0.85299	0.84556	0.86055	0.85032	0.46045
Arbol de cambio de features	0.82690	0.82757	0.82622	0.82545	0.75041
Mejor Predictor	0.84871	0.84884	0.84881	0.84915	0.80227

Como hemos aludido anteriormente, nuestros modelos antes de que se reduzcan la cantidad de features que le estabamos dando al modelo para que estabamos aprendiendo, nos encontramos con un gran problema de overfitting. Esto se refleja en las buenas metricas en el set de test pero una mala puntuacion en el score de kaggle. Tambien se puede ver el salto en el score al poder ver como se reduce el ruido que se genera en el arbol. Y luego el salto que se puede dar con una optimizacion de los hiper paramentros.

4. Matriz de Confusión

A continuacion podemos ver la matriz de confusión de nuestro mejor predictor

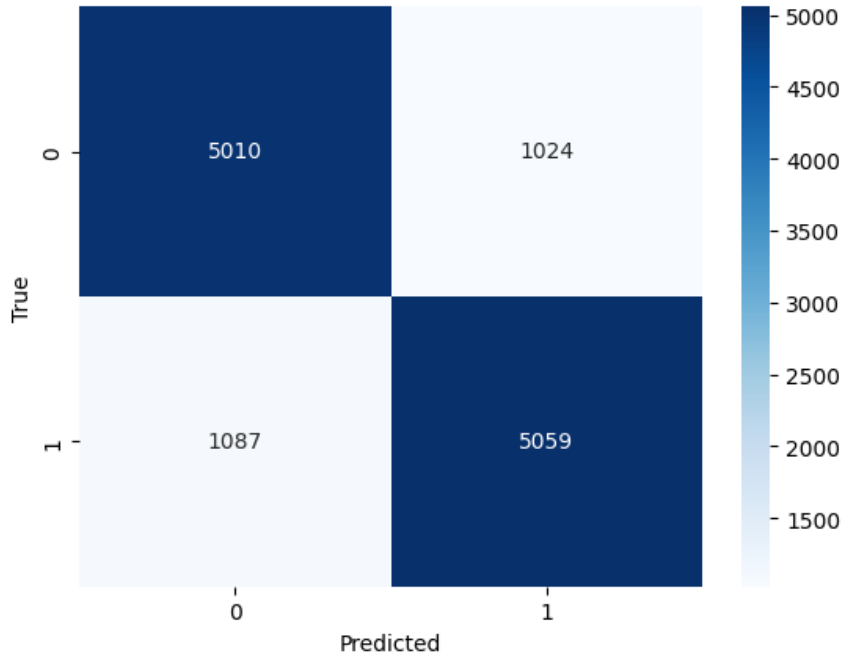


Figura 1: Matriz de prediccion del dataset real del train-test split y las predicciones del arbol entrenados.

Como podemos observar, y como da a entender nuestro score, cerca de un 80 por ciento de la predicciones son correctas, no podemos observar una mayor o menor tendencia a hacia uno u otro parametro, tanto en los falsos positivos y/o negativos, asi como en los verdaderos positivos y/o negativos. A pesar de ello podemos observar

5. Tareas realizadas

Integrante	Tarea
Tomas Gabriel Ayala	Armado de Reporte, Busqueda de hiperparametros
Sebastian Olaran	Busqueda de hiperparametros y Armado de bases del modelo
Juan Ignacio Giacobbe	Busqueda de hiperparametros, Definicion de las variables predictivas