PREDICCIÓN DE CANCELACIÓN DE RESERVA
ENTREGA FINAL
CARLOS EDUARDO CASTAÑO GARZÓN
JUAN ANTONIO ARANGO MORENO
JAGLER DAVID VELASQUEZ VELASQUEZ
INTRODUCCIÓN A LA IA
UDEA
2023

INTRODUCCION

En esta competencia se busca predecir el estado de la reserva de un hotel utilizando datos históricos sobre reservas. Debemos crear un modelo de aprendizaje automático a partir de los datos de entrenamiento para predecir el estado de la reserva (habilitada o no) en los datos de prueba proporcionados en el archivo "test.csv".

Nuestro dataset contiene tres archivos CSV: "train.csv", "test.csv" y "simple_submission.csv" y es tomado de https://www.kaggle.com/competitions/playground-series-s3e7/data. Recortamos algunas filas al archivo "train.csv", ha quedado con 28069.

Los datos de entrenamiento "train.csv" contienen información sobre reservas históricas de hoteles y proporciona una columna llamada "booking_status" que indica si la reserva está habilitada o se canceló. Eliminamos manualmente 1500 datos de tres columnas de "train.csv": arrival_year, arrival_month y arrival_date. Esto para cumplir con el requisito del 5% faltante de las 3 columnas.

En "train.csv" podemos ver 19 columnas, de las cuales 6 son categóricas según nuestro criterio, estas son: type_of_meal_plan, required_car_parking_space, room_type_reserved, market_segment_type, repeated_guest, booking_status. Las columnas no categóricas son:

- id
- no_of_adults
- no_of_children
- no_of_weekend_nights
- no_of_week_nights
- lead_time
- arrival_year
- arrival_month
- arrival_date
- no_of_previous_cancellations
- no_of_previous_bookings_not_canceled
- avg_price_per_room
- no_of_special_requests

Las columnas de “test.csv” tienen el mismo nombre que las de “train.csv” pero sin la columna “booking_status” para un total de 18 columnas. “test.csv” tiene entonces 5 columnas categóricas.

En resumen, el objetivo de la competencia es construir un modelo de aprendizaje automático para predecir si se cancelará una reserva de un hotel. El modelo será evaluado utilizando el área bajo la curva ROC entre la probabilidad predicha y el objetivo observado.

La meta de la aplicación del modelo debe ser lograr una reducción en la tasa de cancelación de reservas de al menos el 40% en relación con la tasa de cancelación antes de la implementación del modelo. Sin embargo, el acierto en las predicciones no garantiza que el trabajo logístico posterior logre tal cifra, por lo que esperamos al menos un 80% en la tasa de acierto del modelo. Así, el trabajo logístico para impedir las cancelaciones dependerá de la organización que aplique el modelo.

EXPLORACIÓN Y PROCESADO

Observando el archivo “train.csv” notamos que hay una columna que tiene por nombre un número. Además de contener un dato faltante justo en la última fila.

```
[7]  ## KEEPOUTPUT
     k = d_train.isna().sum()
     k[k!=0]

     arrival_year     1500
     arrival_month    1500
     arrival_date     1500
     67.5                1
     dtype: int64
```

Imagen 1. Datos faltantes

```
[6]  d_train = pd.read_csv("train.csv")
     d_train.head()
```

ed_guest	no_of_previous_cancellations	no_of_previous_bookings_not_canceled	67.5	no_of_special_requests	booking_status
1	11	0	72.25	0	0
0	0	0	52.00	0	0
0	0	0	56.00	0	0
0	0	0	100.00	0	0
0	0	0	212.06	0	1

Imagen 2. Columna extraña

```
[28] d_train = pd.read_csv("train.csv")
     d_train.tail()
```

ed_guest	no_of_previous_cancellations	no_of_previous_bookings_not_canceled	67.5	no_of_special_requests	booking_status
0	0	0	135.99	0	0
0	0	0	177.30	3	0
0	0	0	49.50	0	0
0	0	0	55.00	0	0
0	0	0	NaN	0	0

Imagen 3. Columna extraña 2

Asumimos entonces que se cometió un error en el llenado de esa columna en específico y decidimos corregirlo: Desplazamos todos los datos de la columna una posición hacia abajo y recuperamos el nombre de la columna copiándolo del archivo “test.csv”.

no_of_previous_bookings_not_canceled	avg_price_per_room	no_of_special_requests
0	67.50	0
0	72.25	0
0	52.00	0
0	56.00	0
0	100.00	0
...	...	...
0	105.30	0
0	135.99	3
0	177.30	0
0	49.50	0
0	55.00	0

Imagen 4. Columna corregida

```
k = d_train.isna().sum()
k[k!=0]
```

```
arrival_year     1500
arrival_month    1500
arrival_date     1500
dtype: int64
```

Imagen 5. Columna corregida 2

Luego, para rellenar los datos faltantes de “arrival_year”, “arrival-month” y “arrival_date” tomamos el documento CSV y convertirlo a XLS por columnas, posteriormente ejecutamos una tabla dinámica con las 3 columnas en cuestión y unificamos cada fecha de ambos años, es decir, sumamos las reservas de enero 1 de 2017 con las reservas de enero 1 de 2018, las de agosto 25 de 2017 con agosto 25 de 2018, etc. Lo anterior lo hicimos buscando conservar y sumar aquellas épocas de mayores reservas. Cabe aclarar que el Dataset fue “Synthetically-Generated” por lo que cada mes consta de 31 días. A continuación, hicimos 2 copias de la hoja

completa original: Una donde organizábamos los valores por cada columna de la forma Fill 1 (ver Imagen 6) y otra donde los organizábamos de la forma Fill 2 (ver Imagen 7).

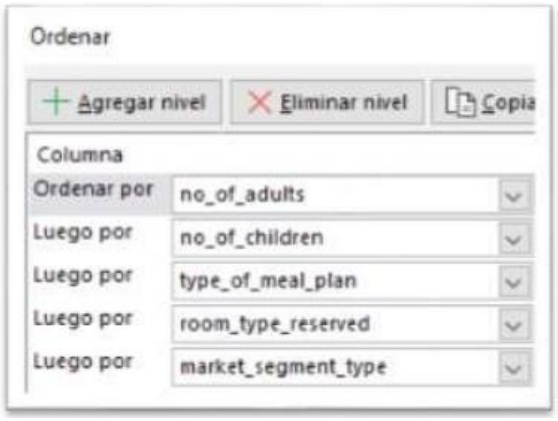

Imagen 6. Forma Fill1

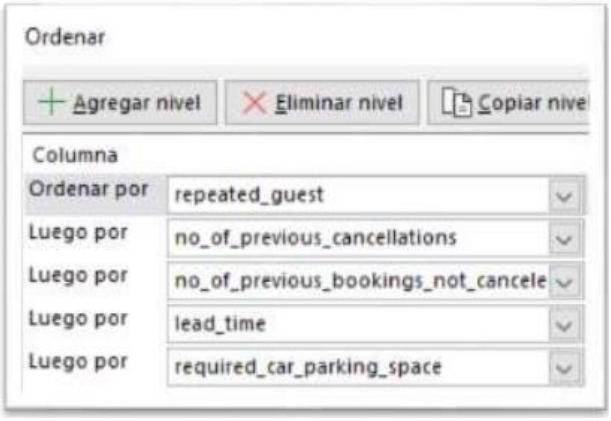

Imagen 7. Forma Fill 2

Para llenar los datos simplemente seleccionamos las columnas “arrival_year”, “arrival_month” y “arrival_date” (aquellas con datos faltantes) y por medio de la función de Excel Buscar y Seleccionar – Ir a Especial – Celdas en blanco llenamos la información con los datos de la celda inmediatamente superior (para cada una de las dos hojas, Fill 1 y Fill 2).

Finalmente llevando a cabo el mismo procedimiento realizado con la hoja original (tabla dinámica y suma de reservas de cada fecha), pudimos graficar las 3 tablas y comparar los datos originales con las dos formas de completar la información (Fill 1 y Fill 2). Esperábamos que la cantidad total aumentara (pues se crearían 1500 valores nuevos), pero tuvimos cuidado de seleccionar aquella que se ajustara más a la original.

A continuación, en la Imagen 8, verán los datos graficados del documento original (con datos faltantes) en color azul, luego se verán los datos Fill 1 en naranja y Fill 2 en gris. Será claro que las variaciones de la forma Fill 1 (naranja) son muy superiores y “radicales” comparadas con la forma de llenado Fill 2 que está en color gris.

Finalmente, decidimos trabajar con el documento ordenado y llenado de la forma Fill 2, lo que quiere decir que las columnas “repeated_guest”, “no_of_previous_cancellations”, “no_of_previous_bookings_not_canceled”, “lead_time” y “required_car_parking_space” están altamente relacionadas con las fechas de las reservas (que eran los datos que debíamos completar).

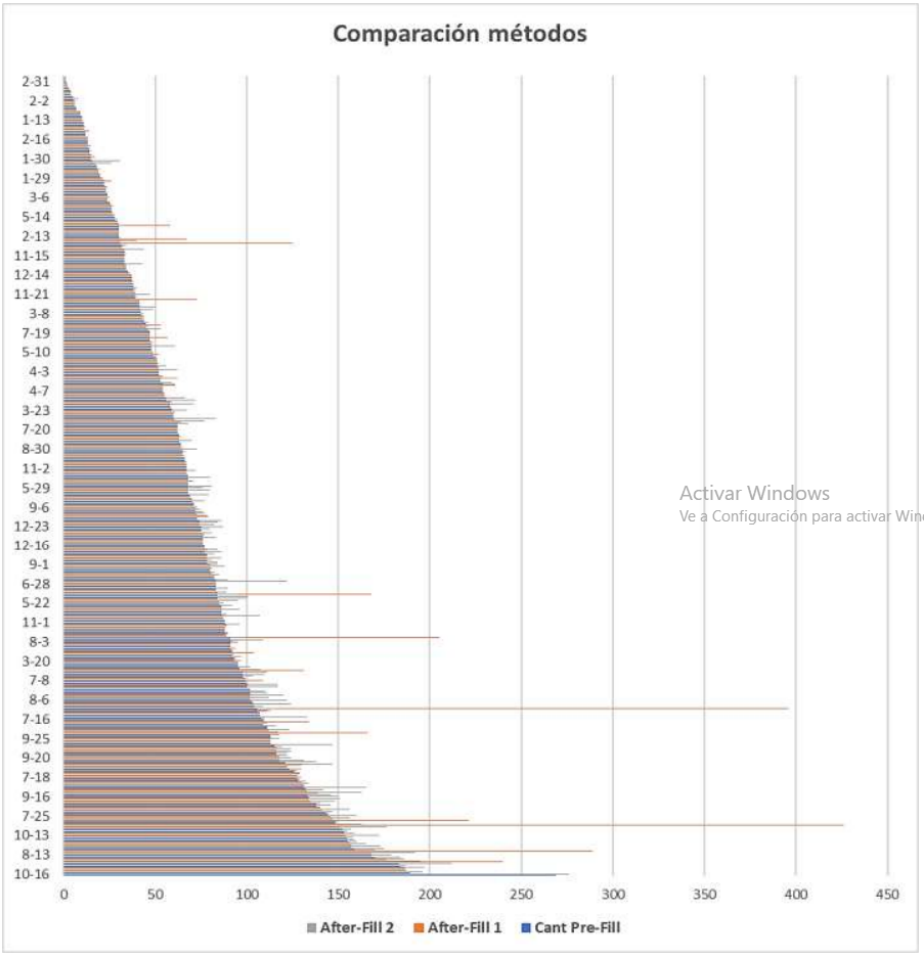

Imagen 8. Comparación Pre-Fill, After-Fill 1 y After-Fill 2

Las columnas “type_of_meal”, “room_type_reserved” y “market_segment_type” (de “train” y “test”) son categóricas, por lo que decidimos convertir sus valores en una representación numérica binaria.

```
## ESTOS DEBEMOS APLICARLE EL GET.DUMMIES()
print(d_train.type_of_meal_plan.unique())
### ESTE TAMBIEN
print(d_train.room_type_reserved.unique())
### tambien
print(d_train.market_segment_type.unique())
```

```
[1 0 2 3]
[0 2 3 1 4 5 6]
[1 0 2 3 4]
```

Imagen 9. Valores únicos de las columnas categóricas

MODELO CON LINEAR REGRESSION

Reducimos el tamaño de la población puesto que la ejecución fallaba debido al uso de RAM

```
[139] d_train = d_train.sample(n=15000, replace=False)
      X = d_train.iloc[:, :-1]
      y = d_train.booking_status
```

Imagen 10. Reducción de la población

```
from sklearn.model_selection import ShuffleSplit
from local.lib import calhousing as ch

cv = ShuffleSplit(n_splits=10, test_size=.3)
ch.plot_learning_curve(estimator, estimator.__class__.__name__, X, y,
                       cv=cv, scoring=calculate_accuracy_score, ylim=(0,0.7))
```

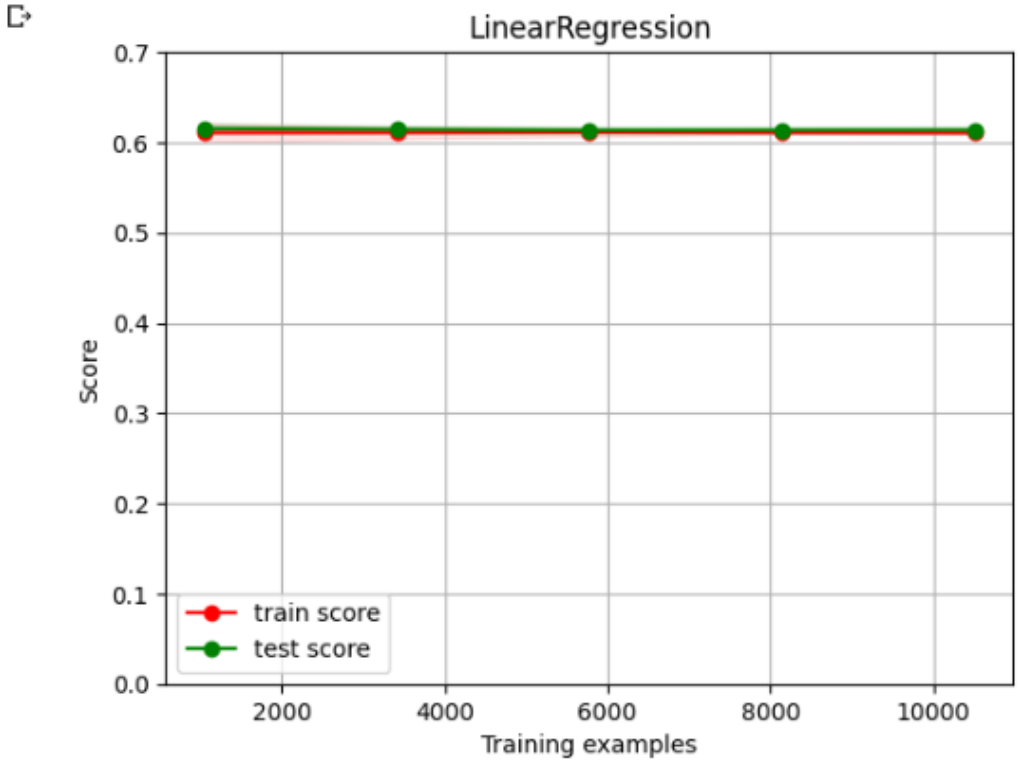

Imagen 11. Curva de aprendizaje LinearRegression

MODELO CON DECISION TREE REGRESSOR

```
from sklearn.tree import DecisionTreeRegressor
estimator = DecisionTreeRegressor(max_depth=8)
ch.plot_learning_curve(estimator, estimator.__class__.__name__, X, y, cv=cv,
                       scoring=calculate_accuracy_score, ylim=(0,0.7))
```

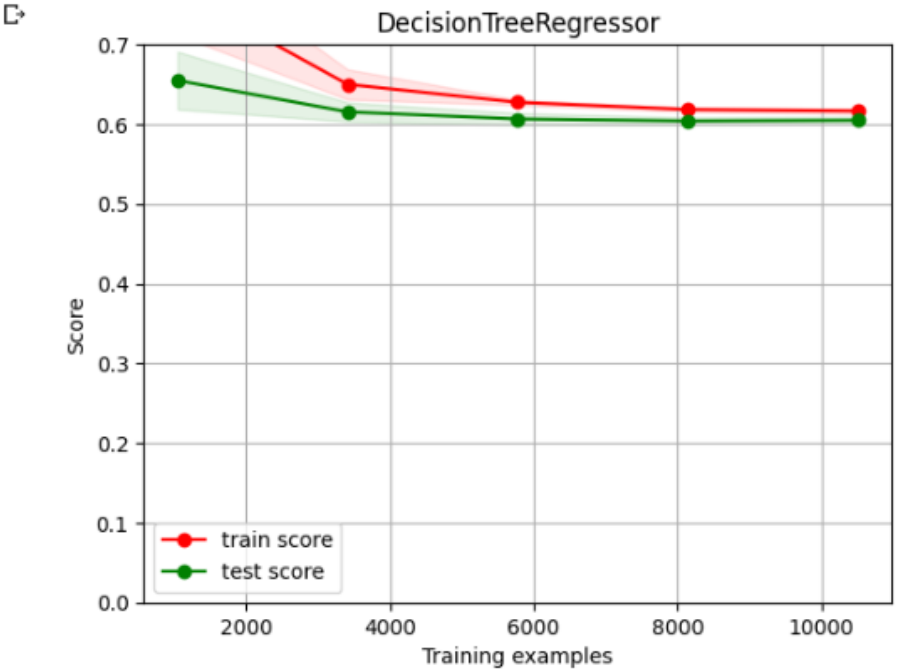

Imagen 11. Curva de aprendizaje DecisionTreeRegressor