Martin Laurito, Lucas Franciulli, Dante Alejandro Finci

Informe Checkpoint 2

En el segundo checkpoint del trabajo práctico construimos un árbol de decisión :

- ## Árbol de Decisión

Empezamos el TP utilizando el archivo csv que modificamos en el checkpoint uno gracias a nuestro análisis de cada columna. Eliminamos las filas duplicadas, excluyendo la ‘id’ (ya que posee cantidades unitarias), cosa que no habíamos hecho anteriormente. Este archivo lo llamamos: “`hotels_modificado_train.csv`”. Luego empezamos haciendo One Hot Encoding a las variables: "hotel", "arrival_date_month", "meal", "country", "market_segment", "distribution_channel", "reserved_room_type", "assigned_room_type", "deposit_type", "agent" y "customer_type". Esto le permite analizar columnas cualitativas a nuestro árbol de decisión.
Luego de tener el Dataframe preparado lo separamos en dos, uno es simplemente la columna 'is_canceled' y el otro son el resto de columnas. Con esto separamos ambos en una proporción 75-25 donde utilizaremos 75 de los datos de ambos Dataframes para poder entrenar nuestro árbol y el resto 25 para poder validar nuestras predicciones.

Una vez hecho esto simplemente entrenamos un árbol con los siguientes hiperparametros: *criterion="entropy", max_depth = 15 y min_samples_leaf=10*

Al hacer esto y ver que tan bueno es nuestro árbol, aplicamos una poda donde disminuimos la cantidad de nodos que utiliza el árbol para hacer un predict. Esto nos ayudará a disminuir la complejidad computacional a la hora de aplicar nuestro árbol.

- ## Búsqueda de Hiperparametros

Para la búsqueda de mejores hiperparametros primero definimos un rango para cada variable, esto se puede ver en: *params_grid.* Además de esto utilizamos Cross Validation para mejorar nuestros predicts. Esto sería dividir en X partes nuestro dataset, específicamente la parte de train que dividimos antes, e ir validando nuestros resultados con una de estas partes e ir cambiándola. A estas partes se las conoce como Folds y nosotros decidimos utilizar 5 de estos. Una vez teniendo todo esto en cuenta, utilizamos RandomizedSearchCV para que agarre hasta 10 árboles diferentes con hiperparametros distintos pero que estén dentro del rango especificado. Una vez entrenó varios árboles se utilizó el Cross Validation para mejorar sus predicts, este se queda con el que tuvo un mejor F1_Score. En nuestro caso esto sucedió con los Hiperparametros:

{'min_samples_leaf': 10, 'max_depth': 15, 'criterion': 'entropy', 'ccp_alpha': 0.01}