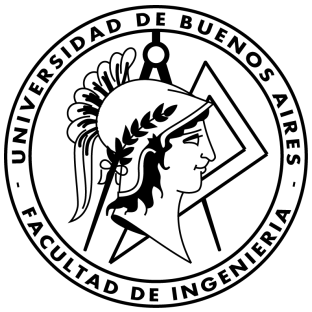


Trabajo Práctico I — Informe Final

[75.06/95.58] Organizacion de Datos
Primer cuatrimestre de 2023

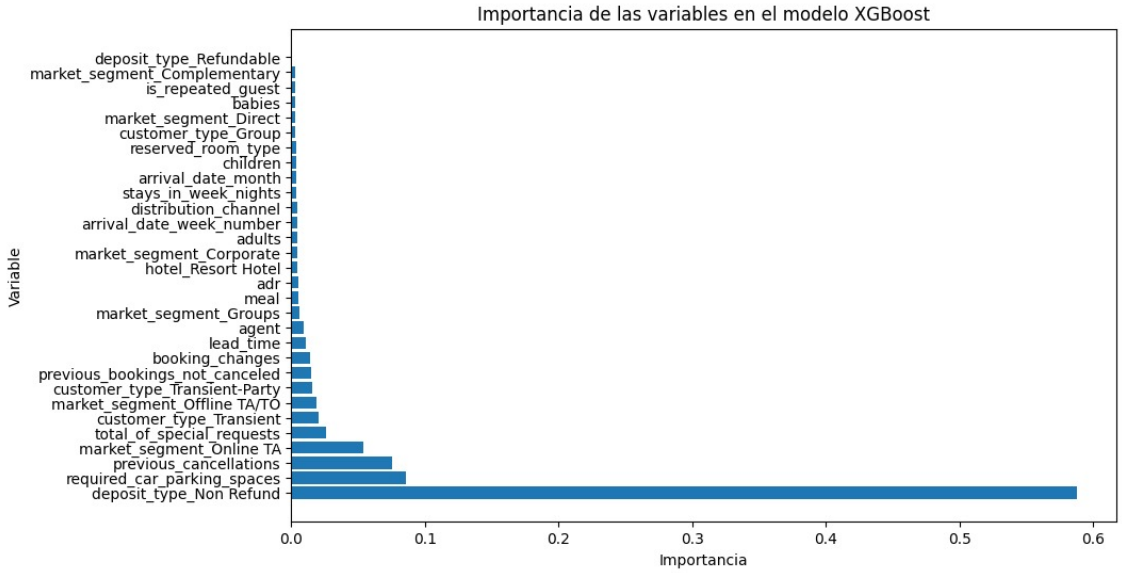
Alumno	Número de padrón	Email
Masri, Noah	108814	noahmasri19@gmail.com
Ayala, Camila	107440	cayala@fi.uba.ar
Loscalzo, Melina	106571	mloscalzo@fi.uba.ar

En este trabajo, fueron pedidas una gran variedad de tareas, comenzando por el preprocesamiento de los datos, para luego crear y entrenar numerosos modelos para la predicción de nuestro target.

Durante el checkpoint 1 se tomaron ciertas decisiones, como la eliminación de algunos valores atípicos que se alejaban más de 3 desvíos, eliminación de columnas, sustitución por la media, entre otros. Consideramos que la mayor parte de estas decisiones llevaron a tener un set bastante balanceado, que obtuvo buenas performances al ser el set sobre el que entrenamos.

Creemos que hubiese sido bueno para ciertas columnas, en vez de eliminarlas, trabajarlas y quedarnos con ciertas partes, como por ejemplo, la columna del país. Sería interesante realizar un análisis más profundo sobre los países más típicos, y quizás realizar one hot encoding manteniendo solo 1 o algunos de ellos. Decidimos no hacerlo ya que había una cantidad desproporcionada de reservas de Portugal, siendo estas más de la mitad, y estas en si estaban balanceadas, por lo que a simple vista no pensamos que aportaría información. Consideramos también, separar el mes de arribo por temporadas, pero en los gráficos de importancia vimos que esta columna no tenía mucho peso, por lo que descartamos esta posibilidad.

Adicionalmente notamos que en todos nuestros modelos la columna de reservation type: non-refund tuvo una importancia alta. Por ejemplo en el modelo xgb con todos sus hiperparametros optimizados obtuvimos la siguiente distribución de importancias.



En general, y sobre todo respecto de esta misma variable, encontramos ciertos patrones que fueron muy en contra de nuestra intuición. En este caso encontramos que la gente que realizaba reservas sin reembolso, casi el %100 de las veces cancelaba, mientras que quienes reservaban sin deposito acudian en mas del %50 de los casos.

Aprendimos sobre la existencia de muchos modelos de predicción, diversas técnicas de optimización de hiperparametros y a analizar las métricas obtenidas; en base a la matriz de confusión ya nos es fácil notar cuando un modelo sobreajusta. El árbol fue el primer modelo predictivo usado. Encontramos que este tipo de modelo no captaba la complejidad del problema. Buscamos diversos sets de hiperparametros pero no pudimos superar métricas de 0.75.

En el siguiente checkpoint usamos ensambles de modelos, los cuales resultaron ser más apropiados para nuestro dataset. Nuestro mayor problema con estos fue que en momentos eran demasiado complejos y terminaban aprendiendo demasiado los datos de entrenamiento. Encontramos que el modelo XGBoost cuenta con hiperparametros específicos para regularlo, por lo que buscamos optimizarlos y conseguimos un modelo que predecía cómo deseábamos.

Finalmente realizamos predicciones con redes neuronales. Comenzamos usando un modelo super simple (1 capa de entrada y 1 de salida y todos los parámetros default) con el que obtuvimos un buen rendimiento. Luego realizamos cambios en la arquitectura y los hiperparametros, construyendo en total 6 modelos de redes neuronales. Variamos la cantidad de capas, de neuronas, de epocas, el tamaño de los baches y los metodos de regulacion. Ninguno produjo metricas mucho mas altas por lo que elegimos el modelo que mostro menos sobreajuste y corrio mas rapidamente.

Nos hubiese gustado saber cómo optimizar los tiempos de entrenamiento de los modelos de alguna manera. Esto nos fue un gran obstáculo, ya que a veces la ejecución de KFolds tomaba más de 20 minutos, por lo que una vez que terminaba de correr no habia vuelta atras. Jamás logramos hacer que funcionen las importaciones de los modelos con Pickle.

A continuacion, dejamos una tabla a modo de resumen de la performance de los distintos modelos, y los hiperparametros obtenidos.

Modelo	F1-score	Accuracy	Hiperparametros
Arbol	0.76	0.75	'max_depth': 4 'max_features': 0.7 'min_samples_leaf': 2 min _samples _split: 5
KNN	0.79	0.78	'weights': distance, 'neighbors': 22, 'metric': manhattan, 'algorithm': brute
SVM	0.79	0.80	'kernel': radial, pca(comp: 15), std_scaler
Random Forest	0.82	0.80	'n_estimators': 119, 'min_samples_split': 2, 'min_samples_leaf': 3, 'max_depth': 9, criterion: gini
XGBoost	0.82	0.83	tree_method: approx, subsample: 0.75, 'n_estimators': 145, 'min_child_weight': 0.5375, 'max_depth': 6, 'learning_rate': 0.2, 'lambda': 1.55, 'gamma': 0.1, 'eta': 0.16, 'colsample_bytree': 0.7, 'alpha': 0.575
Hibrido Voting	0.82	0.82	knn, svm, xgb, random forest con hiperparametros por defecto
Hibrido Stacking	0.81	0.82	base: knn, svm, xgb, random forest meta: árbol
Red Neuronal	0.82	0.82	'learning_rate':0.01 'epochs': 85 'batch_size': 128

Nuestro modelo de preferencia resulto ser XGBoost, ya que mostro una performance similar en train y test, es decir, no sobreajustaba, mostro una performance similar a la hora de predecir en Kaggle, no tardo tanto como los ensambles hibridos en entrenar y ademas es el que obtuvo las mejores metricas en general.