

Checkpoint 3 - Grupo 33

Introducción

En todos los modelos decidimos no reemplazar los NaNs de *Company* y *Agent* por su media, que fue lo que hicimos anteriormente. En vez, los reemplazamos por un -1 para indicar que eran su propia categoría. Con esta misma idea, para ciertos modelos rápidos probamos agregar estas dos columnas en el One Hot Encoding para poder tratarlas verdaderamente como variables cualitativas, aunque nos terminamos arrepintiendo porque empeoraron los porcentajes.

En ciertos modelos decidimos reescribir la fecha en una cierta forma numérica para que el modelo la pueda interpretar (utilizando la funcion *toordinal*).

Para SVM que fue un modelo que tardó mucho tiempo en construirse intentamos hacer un poco de Feature Engineering y PCA (con pipeline) para dimensionar. Cuando realizamos el F.E. nos dimos cuenta que todas las columnas de *reserved_room_type* estaban muy fuerte correlacionadas a su correspondiente *assigned_room_type*. Lo mismo pasaba con *market_segment_Undefined* y *distribution_channel_Undefined*, las cuales ni siquiera aparecen en el dataset de testeo. Pero, cuando realizamos PCA, terminó dando un valor peor a si si directamente utilizabamos un menor porcentaje para training, por lo cual no lo terminamos utilizando.

Construcción del modelo (Hiperparametros)

KNN: {'weights': 'distance', 'n_neighbors': 24, 'metric': 'manhattan', 'algorithm': 'kd_tree'}

SVN: Corrimos tres formas diferentes; sin parámetros, con un kernel lineal y otro con RandomSearchCV para encontrar sus mejores hiperparametros para una kernel poly. Estos hiperparametros son: {'gamma': 0.00046415888336127773, 'degree': 2, 'coef0': -0.300000000000000016, 'C': 61}

Decision Tree: {'min_samples_split' : 4, 'min_samples_leaf' : 3, 'max_depth': 15, 'criterion' : 'gini', 'ccp_alpha' : 0.000111111111111112}

XGBOOST: {'subsample': 0.7, 'objective': 'binary:logistic', 'n_estimators': 638, 'max_depth': 25, 'learning_rate': 0.1, 'lambda': 0, 'gamma': 1, 'colsample_bytree': 1.0, 'alpha': 0.1}

Random Forest: {'n_estimators': 52, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_depth': 47, 'criterion': 'entropy'}

Voting: Utilizamos los modelos Random Forest, XGBoost y Árbol de Decisión.

Stacking: Utilizamos los modelos Random Forest, XGBoost y Árbol de Decisión como Modelos Base y otro XGBoost como Meta Modelo.

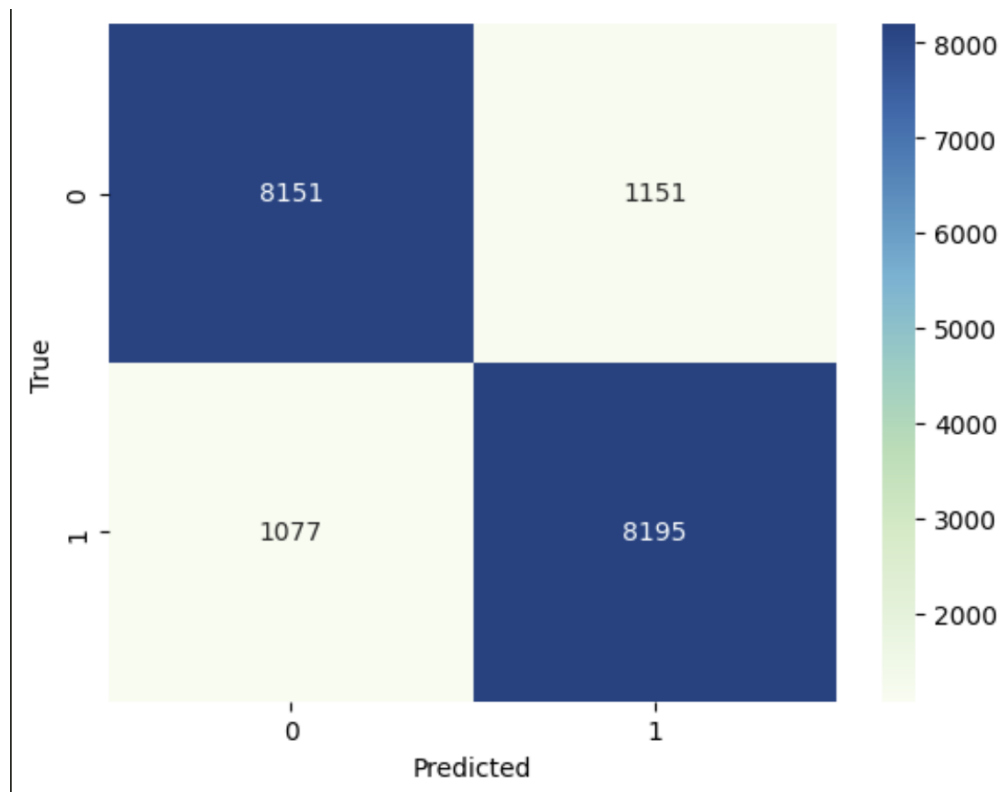
Cuadro de Resultados

Modelo	F1-Test	Presicion Test	Recall Test	Accuracy	Kaggle
KNN	0.766666 6666666 666	0.75146137787 05637	0.7825	0.764078 819855712 2	0.7676
SVM	0.807112 65314973 43	0.8130483853 351295	0.80126296 1501202	0.8084419 08043501 7	0.80854
Random Forest	0.872400 2868648 978	0.8857398902 206789	0.8594565 217391305	0.8754710 88618499	0.87285
XGBoost	0.88444 75569681 624	0.8835432138 62878	0.88535375 32355479	0.8845159 90093679 3	0.87989
Voting	0.879393 74395356 34	0.8764731090 636383	0.8823339 085418465	0.8791859 58867233 7	0.87437
Stacking	0.880330 8626060 802	0.8768457093 943933	0.8838438 308886971	0.880047 37805534 62	0.88065

Nuestro mejor modelo fue Stacking, lo cual tiene sentido lógico ya que este utiliza tres modelos no híbridos como modelos base y luego vuelve a usar XGBoost para actuar como modelo de “votación”. Este modelo tiene la gran ventaja de que se va entrenando a partir de lo que aprenden modelos anteriores, y a diferencia de Voting este hace otro tipo de lógica para poder decidir con qué resultado quedarse, en vez de quedarse con el promedio.

KNN terminó siendo nuestro peor modelo, lo cual era de esperarse. Este modelo se basa en predecir resultados a partir de observaciones “cercanas”. Este modelo es muy sensible a outliers y a conjuntos de datos no balanceados. Por lo cual, aunque le hicimos un tratamiento a los outliers, igualmente estos tuvieron su peso en el resultado final, lo cual se puede ver en su puntaje bajo de Accuracy.

Matriz de Confusion



Se puede apreciar cómo obtuvimos pocos falsos positivos y falsos negativos

Tareas Realizadas

Integrante	Tarea
Lucas Raimondi	XGBoost, Stacking, Voting, RF, KNN
Manuel Davila	XGBoost, SVM, KNN, RF, Informe
Dolores Levi	XGBoost, SVM, KNN, RF, Informe