Informe Final - Grupo 33

Introducción

Antes de empezar con modelos hicimos un análisis extenso de las variables en el checkpoint 1:
Notamos que habían varios NaNs los cuales imputamos por la media, excepto para los NaN's de *Agent* y *Company* que fueron imputados por un el valor "-1" para señalar que la ausencia de este dato significaba que no se usó este método al momento de hacer la reserva.
Luego de realizar un análisis univariado con BoxPlots y demás técnicas, decidimos imputar sus valores tal y como lo hicimos con los NaNs. Para el análisis multivariado, no encontramos outliers que a nosotros nos parezca que deberíamos cambiarlos, por lo cual los dejamos.
También hicimos un análisis de correlación entre las columnas y nuestra columna target para darnos una idea de que atributos serían los que tuvieran más peso al momento de trabajar con modelos predictores.
Para finalizar, utilizamos One Hot Encoding para crear columnas dummies a partir de las columnas categóricas para que nuestros modelos pudiesen procesar los datos.

Siguiendo con los modelos, realizamos diferentes técnicas para cada uno ya que cada modelo presentaba sus propias complicaciones:
Para el modelo sensibles a outliers como árboles de decisión y/o Knn que son más sensibles a outliers hicimos un análisis más profundo de estos casos atípicos en el dataset de train, lo que nos llevó a probar muchas veces para ver cuanto mejoraron estos cambios en el dataset de entrenamiento.

Para la gran mayoría de modelos, utilizamos Random Search o Grid Search para optimizar los hiperparametros. Solamente fue con SVM que no pudimos ya que este modelo presentaba una complejidad que lo hacía muy lento. Para resolver eso, realizamos un poco de Feature Engineering y reducimos un poco la dimensionalidad en principio con PCA aunque seguía tardando demasiado el entrenamiento de los datos, luego la redujimos a mano al quitar las columnas con alta correlación entre sí.

Para modelos de ensambles tratamos de mejorar las métricas de cada modelo de forma individual y luego usar esos hiperparametros en el ensamble y ver si lograba

mejorar, el resultado fue bueno porque en stacking logramos superar a XG Boost usando como metamodelo este mismo.

En las predicciones finales con redes neuronales nos costó mucho lograr entrenar el modelo con Grid search para la búsqueda de hiperparametros, pero una vez conseguido esto, logramos aumentar en hasta un punto porcentual nuestra mejor predicción, tomamos bastante en cuenta los hiperparametros y las funciones que se necesitan para poder hacer un predicción binaria.

Cuadro de Resultados

Modelo	CHPN	F1-Test	Presicion Test	Recall Test	Accuracy	Kaggle
Stacking	3	0.880330862606080 2	0.8768457093943933	0.8838438308886971	0.8800473780553462	0.88065
XGBoost	3	0.8844475569681624	0.883543213862878	0.8853537532355479	0.8845159900936793	0.87989
Arbol	2	0.8511434067108358	0.8433926302414231	0.8590379637618637	0.8500053838699257	0.85087
Red Neuronal	4	0.8481184991559112	0.8518761623454765	0.8443938408154413	0.8498438677721546	0.8501

El stacking que fue nuestro mejor modelo predictor usó random forest, XG Boost y árbol de decisión, cada uno de estos fue pasado con sus mejores métricas de predicción, logrando superar por poco a XG Boost, que ocupa el segundo puesto, con pocos cambios en el dataset ya que puede interpretar valores faltantes y no es sensible a outliers.

Conclusiones generales

Con todos los análisis realizados en el primer checkpoint pudimos aprender más acerca del dominio del problema, y darnos cuenta que había algunos datos extraños que quizá hubiera sido bueno preguntarle a un experto, o a los propios trabajadores de estos hoteles, como el promedio de la tarifa en cero. Gracias al tratado de ciertos valores atípicos que influyen demasiado en la performance de los modelos, logramos que mejore la métrica “f1-score” en hasta un punto porcentual, pero no demasiado, ayudó mucho más la optimización de hiperparametros en subir otro poco la métrica. la métrica subió cuando empezamos a usar modelos más avanzados como random forest o XG boost, donde subió unos 3 puntos solamente optimizando sus hiperparametros, pero sin tocar el dataset, ya que de hacerle cambios solo conseguimos que bajen las métricas, XG Boost fue por lejos el mejor de los modelos, aunque en kaggle conseguimos que con el ensamble de tipo stacking suban unos decimales por arriba de la mejor predicción hecha con XG Boost, poniéndolo como el mejor de nuestros modelos predictores, pero es muy poca la diferencia, no logramos que el ensamble de modelos haga una gran diferencia. Por otro lado las redes neuronales artificiales fueron las más difíciles de programar en cuanto a que había demasiados parámetros para entrenarlas, para crearlas, y había que probar cantidades de capas y de neuronas para ver si las métricas cambiaban, la optimización de hiperparametros tardaba mucho tiempo, y por si fuera poco las métricas con suerte alcanzaron al arbol de desicion. Algo parecido pasó con svm donde tardaba muchísimo entrenarlo para que luego las métricas sean muy bajas. El más fácil de entrenar en cuanto a la optimización de los hiperparametros y el más rápido fue el árbol de decisión, y consideramos que si es útil en relación a su desempeño, porque con los modelos más complejos conseguimos una diferencia de solo 3 puntos porcentuales en comparación al árbol, así que es una muy buena opción.
Por otro lado considerando la complejidad de los datos que había en este dataset, consideramos que nuestra métrica obtenida por nuestro mejor modelo, 0,88, es un valor muy elevado y suficiente para utilizarlo de forma productiva e incluso si hubiéramos podido conseguir el modelo de ensambles en cascada quizá nos acercabamos a métricas cercanas al 0,90, en cualquier

caso pensamos que esta métrica es demasiado buena para la enorme variabilidad de los datos.

Tareas Realizadas

Integrante	Promedio Semanal (hs)
Lucas Nahuel Raimondi	35
Manuel Davila	20
Dolores Levi	20