Objetivo

El objetivo del trabajo práctico fue poder realizar predicciones usando diferentes modelos y ensambles de modelos de clasificación para saber si la reserva de un hotel va a ser cancelada o no.

Desarrollo

Durante el desarrollo del trabajo práctico realizamos el entrenamiento de diferentes modelos enfocándonos principalmente en la métrica f1-score para poder optimizar al máximo las predicciones de los mismos.

Previo al entrenamiento, hicimos un análisis exploratorio donde aplicamos diferentes técnicas de feature engineering tales como imputación por Hot Deck y por la media en casos especiales, también analizamos las observaciones atípicas aplicando la estrategia de Mahalanobis (multivariado) y Rango IQR (univariado)

Comenzamos con el entrenamiento de un árbol de decisión realizando la búsqueda de hiper parámetros utilizando cross validation. Si bien el modelo entrenó en buenos tiempos, tuvimos que limitar la profundidad máxima del árbol y agregarle poda para evitar el sobreajuste.

Luego, realizamos entrenamientos de los siguientes modelos para probarlos individualmente y también poder utilizarlos dentro de ensambles:

- **KNN**: Optimizamos todos los hyper parámetros con cross validation y no nos enfocamos específicamente en la cantidad de vecinos. Si bien, el modelo entrenó en buenos tiempos, al correr cross validation tardó mucho en buscar los mejores parámetros.
- **SVM**: Para este modelo, normalizamos los valores de nuestros dataset con z-score y con Min Max para evaluarlos pero nos dieron muy similares.
    - Lineal
    - Polinómico
    - Radial

  Al entrenar los modelos, pudimos ver que los kernels polinomial y radial daban muy buenas métricas en comparación al lineal. Pero a pesar de dar buenas métricas con el dataset de entrenamiento, cuando se utilizó el dataset de test las métricas fueron muy bajas (10-20% más bajas). Un pendiente que nos quedó fue la reducción de dimensionalidad, para reducir el overfitting y los tiempos de ejecución. Se intentó implementar pero por los tiempos extensos de ejecución en las búsquedas de parámetros (***terminaban en time-out***) y del propio entrenamiento lo volvió imposible.
- **Random Forest**: Este ensamble fue uno de los mejores en cuanto a performance de tiempo y score. También hicimos una optimización de hiperparametros con RandomizedSearchCV pero lo mejor que obtuvimos fue un f1score de 0.864.
- **XGBoost**: Este modelo también fue de los que mejor performó en tiempos en relación de todos los demás evaluados y además nos dio unas buenas métricas de f1-score tanto en el dataset de entrenamiento como en el de test.
- **Ensamble híbrido:** Los ensambles fueron construidos con los modelos que nos dieron las mejores métricas de f1 score, estas fueron: xgboost default, Random forest default y KNN con params optimizados
    - **Voting**: Se realizó este ensamble con y sin votación ponderada y ambas métricas fueron muy buenas

- **Stacking**: Este sin duda fue el mejor ensamble de todos, y contando los modelos también fue el que terminó dando la mejor métrica en entrenamiento y en Kaggle también 🥇🏆

Finalmente, entrenamos un modelo de redes neuronales multicapa en donde fuimos optimizando ajustando la cantidad de neuronas, capas y funciones de activación de cada una donde fuimos obteniendo buenos resultados pero sin mucha diferencia entre ellos (f1 score entre 0.85+-0.01). Para este modelos, pudimos concluir que a medida que agregamos capas ocultas, mejoraban un poco las métricas hasta cierto punto ya que en determinado valor, no encontrábamos mejoras significativas pero los tiempos aumentaban exponencialmente.

Conclusión

Para el contexto de nuestro trabajo práctico, los modelos que tuvieron un mejor rendimiento en relación tiempo de entrenamiento - métricas fueron los de XGBoost, random forest y los ensambles híbridos (stacking el mejor en kaggle) y por su parte el de redes neuronales fue de los más estables, por más de intentar distintas optimizaciones de neuronas, capas y funciones activación no se lograba tanto incremento en el performance.

Cabe aclarar que estas buenas métricas se presentaron bien tanto en entrenamiento como en test, a diferencia de SVM que performaba bien en entrenamiento pero no en test.

¿Modelos o ensambles?: acá podemos ver el top 3 liderado por ensambles híbridos

Submission and Description	Private Score	Public Score	Selected
predict_Stacking_Default.csv Complete · Luis Escalante · 16d ago · predict de stacking con: RF KNN xgboost Todos default	0.88446	0.8814	☐
stacking_modelSubmit.csv Complete · Luis Escalante · 14d ago · Ensamble stacking con KNN optimizado	0.88382	0.88366	☐
votingClasifHard.csv Complete · Luis Escalante · 14d ago · Ensamble voting tipo hard con KNN optimizado	0.88355	0.87864	☐

Luego algo a destacar es la amplia diferencia entre los peores ensambles y los mejores modelos:
Ensambles con las métricas más bajas

votingClasifSoft.csv Complete · Mateo Liberini · 14d ago	0.87337	0.86959	☐
predict_xgboost.csv Complete · Cristian Ariel Pared · 20d ago	0.87246	0.87097	☐

Top 3 modelos con métricas más altas:

predict_SVM_Linear.csv Complete · Mateo Liberini · 16d ago	0.79246	0.78618	☐
predict_SVM_Linear_2.csv Complete · Mateo Liberini · 16d ago	0.79246	0.78618	☐
knn_hyperparametros_submit.csv Complete · Luis Escalante · 14d ago · KNN con hyper params optimizados	0.78804	0.7858	☐

Hay una diferencia de un 8% entre ellos, por lo cual es visible que *"Muchos estimadores mediocres promediados pueden ser muy buenos"* y entre ellos se encuentran las RNA entre un 0.85 y 0.84.