Checkpoint 3 - Grupo 26

Introducción

En primer lugar tomamos los dataset resultantes tanto del checkpoint anterior como del checkpoint número 1 para luego generar los conjuntos x e y de train y test, a partir del dataset de train con una proporción de 70/30 respectivamente, es decir, un 70% para entrenar y un 30% para testear.

Luego nos dividimos los distintos clasificadores, y para cada uno de ellos evaluamos su performance, precisión, métricas y matriz de confusión.

Por último elegimos quedarnos con el dataset resultante del clasificador con mejor resultado para realizar un intento de nuestra mejor *submission*. El elegido, por unanimidad, fue el Random Forest ya que obtuvo los mejores resultados de manera contundente tanto en f1-score, como en precision, recall, accuracy y kaggle.

Construcción del modelo

- Hiperparámetros optimizados para KNN:
  - n_neighbors: número de vecinos más cercanos para tomar una decisión de clasificación.
  - weights: determina cómo se ponderan los votos de los vecinos cercanos.
  - algorithm: especifica el algoritmo a utilizar para calcular los vecinos más cercanos.
  - metric: metrica de distancia.
- Hiperparámetros optimizados para SVN:
  - C: parametro de regularización, evita el overfitting.
  - loss: controla el tipo de función de pérdida.
  - penalty: especifica el tipo de regularización que se aplicará.
  - dual: booleano que controla la formulación del problema de optimización subyacente.
  - multiclass: especifica cómo se manejan los problemas de clasificación con múltiples clases.
- Hiperparametros optimizados para RF:
  - n_estimators: número de árboles de decisión en el bosque aleatorio.
  - max_depth: profundidad máxima de los árboles de decisión en el bosque.

    - min_samples_split: número mínimo de muestras requeridas para dividir un nodo interno del árbol.
    - min_samples_leaf: número mínimo de muestras requeridas en una hoja del árbol.
- Hiperparámetros optimizados para XGBoost:
    - n_estimators: números de árboles en el conjunto.
    - max_depth: profundidad máxima de cada árbol.
    - learning_rate: tasa de aprendizaje que controla cuánto contribuye cada árbol al modelo final.
    - subsample: proporción de muestras de entrenamiento que se utilizan para ajustar cada árbol.
    - colsample_bytree: proporción de características que se utilizan para ajustar cada árbol.
- Modelos usados para el ensamble de tipo voting:
    - KNN
    - RF
- Modelos usados para el ensamble tipo stacking:
    - KNN
    - RF (meta learner)

Cuadro de Resultados

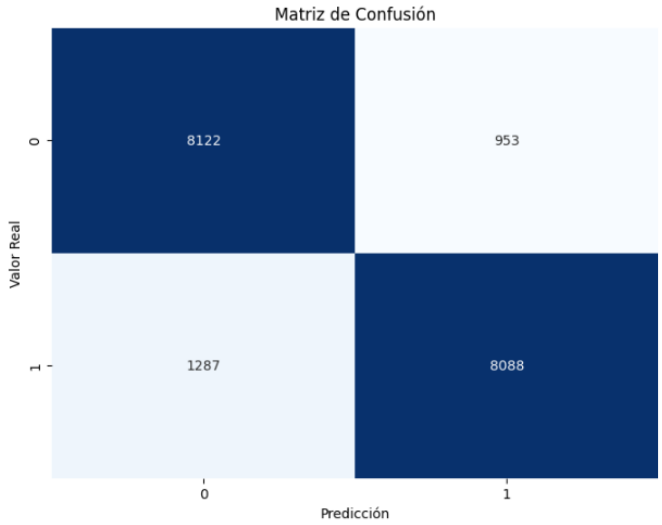
Modelo	F1-Test	Presicion Test	Recall Test	Accuracy	Kaggle
KNN	0.77001	0.75200	0.78890	0.76054	0.7529
SVM	0.79753	0.80881	0.78656	0.79707	0.79997
Random Forest	**0.87836**	**0.89459**	**0.86272**	**0.87859**	**0.87284**
XGBoost	0.87617	0.87458	0.87776	0.87392	0.86507
Voting	0.77061	0.75315	0.78890	0.76135	0.75417
Stacking	0.87325	0.87601	0.87050	0.87159	0.86834

Detalles breves de cada modelo:

- KNN: algoritmos de clasificación y regresión. Se basa en asignar una determinada etiqueta a un punto, en función de las etiquetas de los puntos cercanos.
- SVM: algoritmo de clasificación, que encuentra el mejor hiperplano que separe las clases.

- XGBoost: implementación a partir del algoritmo de **gradient boosting**. Se usa para clasificación y regresión.
- Voting: modelo de ensamble que combina predicciones de varios modelos, y toma una decisión en base a los votos.
- Stacking: parecido al ensamble Voting, solo que combina sus predicciones con otro modelo (meta-learner).
- **RF**: combina múltiples árboles de decisión. Cada árbol entrena un subconjunto de datos. Toma decisiones en base a la mayoría de los votos de los árboles.

Matriz de Confusion



Esta matriz de confusión, es una herramienta que nos permite ver el rendimiento de nuestro modelo de clasificación, que en este caso vendría a ser el **Random Forest**, ya que fue el mejor clasificador de los demás.

- Verdaderos positivos (TP): casos en donde el modelo predijo de forma correcta cuando una reserva es cancelada (8088).
- Verdaderos negativos (TN): casos en donde el modelo predijo de forma correcta cuando una reserva no es cancelada (8122).
- Falsos positivos (FP): casos en donde el modelo predijo incorrectamente si la reserva fue cancelada (953).
- Falsos negativos (FN): casos en donde el modelo predijo de forma incorrecta si la reserva no fue cancelada (1287).

Vemos que nuestro modelo elegido, a comparación del anterior (chp2), mejoró bastante en la predicción de los verdaderos negativos, mientras que en los TP se mantuvo. Es por esa razón que obtuvimos una mejor métrica de **f1_score.**

Tareas Realizadas

Integrante	Tarea
Garcia, Nicolas	Armado de reporte Entrenamiento KNN
Vallcorba, Agustin	Entrenamiento RF Entrenamiento XGBoost Entrenamiento ensambles
Carbajal Robles, Kevin Emir	Armado de reporte Entrenamiento SVM