En este checkpoint aplicamos los ensambles pedidos (KNN, RF, SVM, XGBoost, Stacking y Voting) al dataframe modificado de la entrega anterior. Esto con el fin de mantener un número reducido de columnas (66) y reducir el tiempo que toma en ejecutar cada modelo.

Empezando por K-Nearest Neighbors (KNN), la optimización se llevó a cabo variando los hiperparámetros *n_neighbors*, *weights* y *p*. El parámetro *algorithm* fue sacado de la optimización ya que suponía un aumento drástico en el tiempo de ejecución del modelo. Al finalizar el ajuste del modelo y la matriz de confusión vemos que el modelo obtuvo un rendimiento general bueno, ya que el puntaje F1 y la precisión toman valores similares. Cabe destacar que la precisión es ligeramente más baja que el recall, esto indica que el modelo puede estar enfocado en clasificar instancias como positivas, pero en general, ambos resultados son buenos.

Continuando con el Support Vector Machine, comenzamos transformando los datos del dataset para que pertenezcan al intervalo [0,1], ya que queremos evitar que el rango de valores que toman las variables sea muy grande. Luego, analizamos por separado cómo se comportan los diferentes tipos de kernel de los SVM. Al hacer esto notamos que el puntaje F1 de los SVM que utilizan los kernels lineal, polinomial y radial aumentan en este orden, siendo el puntaje más alto el del radial, que es de aproximadamente 0.8535, y el más bajo el lineal con 0.8181. Sin embargo, cuando intentamos optimizar los hiperparámetros de nuestro SVM se obtiene un modelo defectuoso, con una matriz de confusión desbalanceada que muestra una alta precisión, pero un recall muy pobre. Suponemos que se puede tratar de una mala optimización de hiperparámetros debido a varios factores, especialmente que el número de iteraciones elegido es muy bajo para evitar que el tiempo de ejecución llegue a números exorbitantes. Esto podría resolverse aplicando técnicas de reducción de la dimensionalidad. Además, existen hiperparámetros cuyos valores pueden ser contraproducentes para el rendimiento de nuestro modelo, por ejemplo, un parámetro C alto puede estar sobreentrenando nuestro modelo, o un kernel incorrecto que haga que se separen incorrectamente las clases. Esto último es interesante destacarlo, ya que al optimizar hiperparámetros se obtiene que el kernel polinomial es el más adecuado, mientras que en el análisis previo se observa mejores resultados usando el kernel radial.

En el caso de Random Forest observamos que incluso con una sola iteración, optimizar el modelo nos dió como resultado un puntaje F1 de 0.8727, que fue nuestro segundo mejor puntaje para esta entrega. Luego de este primer resultado decidimos optimizarlo utilizando tres iteraciones y asignando rangos más precisos a los hiperparámetros, lo que nos dio un resultado incluso mejor. Sin embargo, la optimización de nuestro Random Forest resultó tediosa debido al gran tiempo de ejecución que conlleva utilizar una cantidad de iteraciones superior a tres.

En cuanto a XGBoost, observamos que el rendimiento del clasificador *XGBClassifier* es aceptable incluso sin ser optimizado. Se nos ha explicado que este ensamble es el más certero que se encuentra por debajo de las redes neuronales, y esto se refleja adecuadamente en nuestros resultados. Luego, se obtuvieron mejores resultados optimizando con *CrossValidation*, y la matríz de confusión muestra un modelo con una precisión, recall y F1 muy similares, superando el umbral de 0.88 de puntaje F1 (lo cual es nuestro mejor puntaje hasta ahora).

Para los ensambles híbridos utilizamos tres modelos base: *Random Forest*, *XGBoost* y *KNN* para Stacking, Regresión lineal, *Random Forest* y Árboles de Decisión para Voting . Decidimos utilizar estos modelos ya que son los que menos tiempo necesitan para correr en comparación a otros con los que trabajamos (como SVM). Al finalizar el ajuste de los modelos observamos que su rendimiento es mejor que el de los modelos que lo componen,

y vemos en sus métricas que sus puntajes F1, recall y precisión son bastante similares, siendo Voting el ensamble híbrido con mejor puntaje (0.847).