Introducción:

Para esta entrega se entrenaron varios modelos y ensambles de clasificación con diferentes características entre sí y se realizaron diferentes predicciones en kaggle con cada una de ellas. Se trabajó mucho principalmente en los tiempos de entrenamiento de cada modelo en sí y también en la búsqueda de hyper parámetros a tal punto de que tuvimos que guardar los modelos entrenados dentro del dataset para no tener que volver a entrenarlos.

Checkpoint 3: Ensamble de Modelos:

A continuación, detallamos cada uno de los modelos utilizados para esta entrega:

KNN: Realizamos una búsqueda de hyper parámetros con cross validation y nos guardamos los mejores resultados ya que demoraba mucho en entrenar. Nos enfocamos en buscar todos los hyperparameters y no solamente en la cantidad de vecinos cercanos. Con este modelo, conseguimos métricas que no fueron significativas en comparación con los otros modelos.

SVM: Se realizaron 3 SVMs con los kernels radial, lineal y polinómico. Solo se modificó la variable C ya que era la única que mostraba una diferencia en este caso. Se notó que los kernels radiales y polinómicos son más precisos que el lineal. No tuvieron la mejor performance pero mostraron buenos resultados con el set de entrenamiento.
Se intentó utilizar GridSearchCV y RandomSearchCV pero los tiempos resultaron demasiados altos y terminaba arrojando una excepción ***time-out.*** Antes de implementar los SVMs se utilizó el SVM default para ver qué tipo que media mejor, si escalado minmax o normalizado.
Nota: Decidimos NO optar por hacer una reducción de dimensionalidad para SVM ya que obtuvimos buenos resultados con las features originales.

XGBoost: Este modelo es uno de los que mejores performo en tiempos, ya que en cuestión de segundos teníamos el modelo entrenado y la predicción lista con unos resultados excelentes de f1 score. Realizamos un random search con el método integrado en xgb y no vimos mejoras significativas en relación a los parámetros default.
También, optamos por realizar un random search con la librería de scikit-learn pero esto nos tomó más de 1 hora y media y decidimos descartar este método.

Random Forest: Este ensamble también fue uno de los mejores en cuanto a performance de tiempo y score, ya que en menos de 15 segundos se entrenaba y hacía las predicciones obteniendo 0.88 de f1 score. También hicimos una optimización de hiperparametros con RandomizedSearchCV pero lo mejor que obtuvimos fue un f1score de 0.864.

Ensambles híbridos: Se crearon dos tipos de ensambles híbridos, para ambos casos hicimos cross validation con la misma proporción de todos los modelos (80-20) con los mejores modelos entrenados anteriormente (xgboost default, RF default y KNN con params optimizados), cabe agregar que probamos agregando un SVC pero los tiempos de entrenamientos se disparaban de alrededor de 1-3 minutos a unos 40 min los de tipo voting y más de 3 horas (y nunca terminó) el de stacking. A su vez con el de tipo voting probamos con voting hard y soft y obtuvimos métricas muy parecidas, para el de stacking dejamos el estimador final por default (LinearRegressionModel)

Conclusión

Los modelos/ensambles que mejor se adaptaron a nuestros datos de entrenamiento fueron xgboost, random forest y ambos tipos de ensambles híbridos ya que nos dieron las mejores métricas y la mejor performance en tiempos (todos en cuestión de segundos). Por otro lado, cabe destacar que conseguimos buenos resultados con SVM pero el costo de tiempo fue mucho mayor en relación a los modelos mencionados anteriormente. Además, estos últimos permormaron bien con los datos de entrenamiento pero no con los de prueba (bajo de 0.84 a 0.62)