# Informe Final - Grupo 12

## Introducción

A lo largo de estas semanas en relación al trabajo práctico, estuvimos explorando algunas técnicas de análisis de datos, como el tratamiento de los valores faltantes y outliers (datos anómalos o atípicos) ó transformaciones sobre los datos como las normalizaciones para su posterior uso,  y también exploramos el uso de algunos modelos predictivos, donde algunos utilizados fueron: árboles de decisión, modelos de ensamble híbridos (Voting,Stacking) y no híbridos (RandomForest, XGBoost), redes neuronales, donde este último nos enfocamos en crear distintas arquitecturas.

Uno de los objetivos era mejorar la performance de estos modelos predictivos, y para eso nos enfocamos en la búsqueda de los mejores valores de los hiperparámetros de cada modelo tal que maximicen la métrica del f1-score, y así encontrar el mejor modelo, donde buscamos entrenarlos y evaluarlos con nuestro dataset de test.

## Cuadro de Resultados

| Modelo | CHPN | F1-Test | Precision Test | Recall Test | Accuracy | Kaggle |
|---|---|---|---|---|---|---|
| modelo_1 | 2 | 0.84/0.83 | 0.82/0.86 | 0.87/ 0.81 | 0.84 | 0.77 |
| modelo_2 | 3 | 0.86 | 0.84 | 0.87 | 0.86 | 0.84374 |
| modelo_3 | 4 | 0.78 | 0.74 | 0.83 | 0.77 | 0.76431 |

**Nota: indicar brevemente en qué consiste cada modelo de la tabla y detallar el caso del mejor modelo.**

El modelo_1 consiste en un árbol de decisión cuyos hiperparámetros fueron optimizados, con el objetivo de mejorar la performance en términos de la métrica F1-score.

El modelo_2 es un modelo de ensamble híbrido del tipo Stacking, el cual se construyó con los mejores modelos que habíamos encontrado en el checkpoint 3 de KNN, RandomForest y XGBoost. Este para nosotros fue el mejor modelo porque tuvo un gran rendimiento en datos conocidos (los datos de test), donde su f1-score había dado 0.86, y un muy buen rendimiento en otro set de datos no vistos, por ejemplo los de kaggle, teniendo un f1-score de 0.84.

El modelo_3 consistió en una arquitectura de red neuronal especializada en problemas de clasificación, donde se buscó optimizar sus hiperparámetros para mejorar la performance en sus predicciones.

## Conclusiones generales

- **¿Fué útil realizar un análisis exploratorio de los datos?**
  *Si, el análisis exploratorio nos ayudó a encontrar las correlaciones más fuertes para tener en cuenta a la hora de crear los modelos como también nos ayudó a encontrar datos faltantes y outliers.*
- **¿Las tareas de preprocesamiento ayudaron a mejorar la performance de los modelos?**
  *Si, la normalización de datos que utilizamos a partir de la construcción del modelo SVM nos ayudó a obtener mejor performance en general. Obtuvimos una mejora de 0.31 en la métrica f1 en la predicción del dataset de test con la normalización.*
- **¿Cuál de todos los modelos obtuvo el mejor desempeño en TEST?**
  *El modelo que mejor se desempeñó en test en términos de la métrica de f1-score fue el modelo de ensamble de tipo Stacking, que incluía en su armado los mejores 3 modelos que habíamos obtenido de KNN,RandomForest y XGBoost.*

- **¿Cuál de todos los modelos obtuvo el mejor desempeño en Kaggle?**
  *Tanto en test como en Kaggle el modelo de Stacking obtuvo los mejores resultados.*
- **¿Cuál fue el modelo más sencillo de entrenar y más rápido? ¿Es útil en relación al desempeño obtenido?**
  *RandomForest fue el modelo más sencillo y rápido de entrenar, este modelo nos dió métricas similares al mejor modelo de Stacking y requirió menor tiempo de búsqueda de hiperparámetros. Con XGBoost también obtuvimos métricas similares en poco tiempo.*

- **¿Cree que es posible usar su mejor modelo de forma productiva?**
  *Si bien consideramos que el mejor modelo obtuvo buenas métricas, habría que analizar si el costo de error es justificado a la hora de usarlo en producción. Hay que tener en cuenta su aplicación para poder saber si el modelo será suficiente, como también, tener en cuenta los posibles cambios en correlaciones que se pueden dar durante el tiempo.*
- **¿Cómo podría mejorar los resultados?**
  *En términos generales, para mejorar las predicciones se podría dar un rango más amplio de valores a probar para los hiperparámetros, dado que podríamos hacer una búsqueda más extensa de los mejores hiperparámetros tal que maximicen la performance en términos de la métrica f1-score, cómo también probar evaluar a los modelos mediante una mayor cantidad de folds para el cross validation y de combinaciones aleatorias a probar (utilizando en este caso RandomizedSearchCV). Lo que sucede es que cuanto más extensa sea la búsqueda, mayor tiempo demandará la misma, por lo tanto se podría dejar ejecutar más tiempo el código para luego obtener posibles mejores rendimientos, pero analizando los riesgos del overfitting.*

*Además, se podría haber aplicado alguna técnica de reducción de la dimensionalidad ya que puede acelerar el proceso de búsqueda, ya que se pueden probar diferentes combinaciones de hiperparámetros en un espacio de menor dimensión, y esto ahorraría tiempo y recursos computacionales. Además, se podría eliminar las características no útiles o irrelevantes en términos de aporte de información, mejorando así la calidad del modelo y reduciendo el riesgo de overfitting.*

## Tareas Realizadas

Teniendo en cuenta que el trabajo práctico tuvo una duración de 9 (nueve) semanas, le pedimos a cada integrante que indique cuántas horas (en promedio) considera que dedicó semanalmente al TP

| Integrante | Promedio Semanal (hs) |
|---|---|
| Gaston Sabaj | 16-20hs |
| Juan Yago Pimenta | 6hs |