.UBAfiuba
FACULTAD DE INGENIERÍA

75.06 /95.58 Organización de Datos
Dr. Ing. Rodríguez - 2°C 2023

# Informe Final - Grupo 16 GPWin

## Introducción

El dataset analizado posee 31 columnas y 61913 observaciones. Existen solo 4 columnas con valores nulos. La variable "company" se eliminó debido a su bajo aporte de información. Se utilizó one hot encoding para las variables categóricas y se empleó el análisis univariado con box-plot para eliminar los outliers en la mayoría de las variables. En dos variables específicas, se aplicó el método de imputación MICE para manejar los valores nulos.
Se comenzó con el entrenamiento del modelo de predicción más simple, un árbol de decisión , con hiperparametros seleccionados manualmente. Se continuó con un modelo donde se optimizaron los parámetros de árbol mediante "parametersGrid" lo cual implicó una mejora considerable en la eficiencia.
Se realizó también un prueba con un dataset que poseía diferente manejo de features como ser la inclusión de los outliers y la eliminación de los nulos imputados mediante mice, obteniendo resultados superadores.
A posteriori, se trabajó con varios modelos KNN / Random Forest / SVM / XGBoost obteniendo el mejor resultado con este último.
Luego se utilizaron ensambles de tipo Voting y Stacking logrando el mejor modelo con un ensamble de tipo Stacking. Particularmente, utilizando un meta-modelo de regresión logística y aplicando en el stacking solo nuestros dos mejores modelos (RandomForest y XGboost) logramos nuestro mejor predictor en Kaggle.
Finalmente se procedió al desarrollo de redes neuronales, primero simples y luego múltiples no mejorando las métricas conseguidas con modelos de ensamble.

## Cuadro de Resultados

Presentamos un cuadro de resultados donde exponemos las métricas F1, Precision y Recall que obtuvimos de nuestros modelos en test y la métrica F1 obtenida de las predicciones subidas a Kaggle.

| Modelo | CHPN | F1-Test | Precision Test | Recall Test | Kaggle |
|---|---|---|---|---|---|
| **Árbol de decisión** | 2 | 0.8382 | 0.8120 | 0.8662 | 0,84482 |
| **Voting** | 3 | 0.8694 | 0.8546 | 0.8847 | 0.85803 |
| **Stacking** | 3 | 0.8688 | 0.8576 | 0.8804 | 0.86055 |
| **Perceptrón multicapa** | 4 | 0.8278 | 0.8204 | 0.8354 | 0.82457 |

Descripción de modelos:

- **Árbol de decisión:** árbol con profundidad máxima de 45, cuyos hiperparámetros fueron optimizados con random search durante 1000 iteraciones recorriendo una grilla de parámetros de criterion, max_depth, ccp_alpha y otros más. Se realizó k-fold cross-validation. El valor del parámetro de regularización (poda) es de ~0.0002.
- **Voting**: Realizamos un ensamble con los mejores dos modelos, es decir, el mejor RF y el mejor XGBoost, con un sistema de voting que realiza las predicciones tomando el argmax de la suma de las probabilidades de los modelos individuales.
- **Stacking** : al igual ensamble anterior se utilizaron los mejores dos modelos, es decir, el mejor RF y el mejor XGBoost, con un modelo de regresión logística como meta-modelo y el passthrough desactivado, este ejercicio terminó en una disminución del ~0.0003% del f1 score en test pero una mejora en Kaggle, convirtiéndose así en nuestro modelo más performante.
- **Perceptrón multicapa**: con capa de entrada tipo InputLayer donde pasa las variables de entrada a una capa oculta. La capa oculta es única fully-connected tanto con la capa input como output, manteniendo la misma cantidad de neuronas que variables y activación ReLU. Por último el modelo tiene una capa output con una función sigmoidea para asignación del resultado binario en función de la probabilidad.

## Conclusiones generales

Sin lugar a dudas no solo es útil, sino necesario realizar un análisis exploratorio de los datos, dado que utilizar los dataset directamente nos llevaría a peores estimaciones, incluso a errores groseros o bien a la complejización del problema utilizando variables que no aportan datos.

Si bien las tareas de preprocesamiento son indispensables, dado que es necesario adecuar los datos, como bien puede ser dar tratamiento a los outlier o realizar una normalización de algún feature para poder generar ciertos modelos, estas tareas normalmente tienden a mejorar la performance de los modelos. Sin embargo, no implica que con el primer análisis y preprocesamiento se consigan los mejores resultados sino que es un proceso iterativo lo que lleva a tener que realizar varios preprocesamientos diferentes, entrenar nuestros modelos y en función de los testeos ver realmente qué dataset generó una mejora significativa.

Como se puede apreciar en los checkpoints de nuestro TP, muchas veces los valores de test que obtuvimos no necesariamente coincidieron con los score de validación al entrenar el modelo. Es decir, conseguimos métricas en validación mejores en algunos modelos, sin embargo al testearlo (sobre todo en kaggle), los resultados eran peores.

Los mejores valores de test, los conseguimos con el modelo de Voting dando un F1 score aproximado de 0.8694.
El modelo más performante en Kaggle ha sido el modelo de stacking, descrito en el cuadro del punto anterior. Si bien el mismo en test tuvo una performance menor que voting (0.8688) en Kaggle obtuvo un score F1 de 0.86055.
Algunos modelos, como random forest, han tenido tiempos de entrenamiento relativamente cortos y un desempeño muy bueno.
XGBoost tambien consiguio un buen desempeño considerando un tiempo de entrenamiento comparable con RandomForest. Aunque le hicimos una optimización de hiperparámetros más exhaustiva que incrementó el tiempo total de training.
Sin embargo, tanto con voting como con stacking se percibe una mejora significativa con un muy bajo tiempo de entrenamiento, siempre que los modelos de base se encuentren ya entrenados y con un score de predicción aceptable. Aunque también notamos una buena performance utilizando ensambles con modelos no optimizados de forma exhaustiva y por ende habiendo consumido menor tiempo.
Basándonos en la información de la que disponemos, consideramos que es posible utilizar en nuestro modelo ya entrenado en forma productiva si bien el score que conseguimos no es extremadamente elevado.
Para mejorar la performance del modelo, podríamos probar nuevos ensambles (es decir, ensambles con nuevos modelos, generando diversas combinaciones). También se podrían evaluar otras arquitecturas de redes neuronales, ya que existe una gran variedad de ellas. No proponemos fijar el objetivo en la optimización de parámetros de los mismos modelos dado que entendemos que hemos realizado una optimización exhaustiva. A su vez, también podríamos re-analizar el dataset, realizar un nuevo proceso de feature engineering y probar con otras variantes de post-procesado del dataset.

## Tareas Realizadas

| Integrante | Promedio Semanal (hs) |
|---|---|
| DIEM, Walter Gabriel | 10 |
| MAIOLO, Alejandro | 8 |
| RUIZ, Karen Belén | 6 |