Checkpoint 4 : Redes Neuronales

Para la realización de este último checkpoint tuvimos que encarar la construcción de diversas arquitecturas de redes neuronales, a su vez, distintos modelos generados a partir de la experimentación con los hiperparámetros ajustables.

En primera medida se buscó avanzar de menor a mayor en cuanto a las complejidades de nuestras redes y las relaciones internas que se dan entre cada capa.

Se inició con una red que consta de dos capas de una neurona cada una, en la capa de salida se determinó usar función de activación sigmoidea, ya que es de las más efectivas a la hora de tratar problemas de clasificación binaria. En el marco de esta arquitectura inicial se generaron modelos con Stochastic gradient descent y Adam como funciones de optimización según la métrica F1-score (que debió ser implementada de manera propia ya que keras no cuenta con una), luego, como función de pérdida se eligió binary crossentropy ya que es de las más utilizadas en este tipo de clasificación. Para la optimización de los hiperparametros se utilizó GridSearch buscando la cantidad de epochs, tamaño de lote y learning rate que dejen mejor perfilado a nuestras redes. A la vez se utilizó Early Stopping para encontrar casos en los que el aumento de la cantidad de épocas ya no aplicaban una mejora considerable a nuestra red y solo la acomplejaba temporalmente.

Para la segunda arquitectura se buscó replicar la primera pero únicamente cambiando la cantidad de neuronas en cada capa y agregando una función relu en la de entrada, esto con el fin de estudiar cuánta importancia toman estas en nuestro problema de estudio.

En la tercera arquitectura se optó por el análisis de las variables categóricas haciendo uso de la función de pérdida entropía cruzada categórica. La capa de entrada con dos neuronas, función de activación relu e inicializador de kernel uniforme. Para la capa de salida se optó por brindarla de tantas neuronas como variables hay en análisis junto una función de activación softmax que es comúnmente en problemas de clasificación de este tipo.

Para la cuarta arquitectura agregamos una capa oculta con función de activación relu, al igual que en la capa de entrada, en donde la capa de entrada contaba con 64 neuronas, la capa oculta 32 y para la capa de salida utilizamos una función de activación sigmoidea. De función de optimización elegimos Adam ya que es mejor temporalmente que SGD y función de pérdida de entropía cruzada binaria.

Para la quinta arquitectura se eligió hiperparametrizar nuestra cuarta arquitectura que es la que mejor rindió. Para ello se utilizó GridSearch con distintos tamaños de lote y cantidad de epocas.

Conclusiones finales

Luego de la realización del trabajo, en base al mejor entendimiento del problema y el entrenamiento y evaluación de modelos predictivos, pudimos encontrar varias conclusiones.

Comenzando por el análisis exploratorio de los datos consideramos la importancia de entender la correlación entre las variables para una posterior reducción de dimensiones, dado que esto disminuye considerablemente la complejidad de varios métodos. Así mismo, comprender la distribución de las variables también la consideramos importante, ya que a partir de eso es que se puede entender mejor la presencia de valores faltantes o atípicos que alteran la composición del dataset. En lo que respecta al preprocesamiento de los datos, pudimos observar que la normalización o estandarización de las variables numéricas mejoraba de manera significativa el rendimiento de los modelos. En ésta sección nos hubiese gustado encontrar mejores relaciones para la creación de variables, como en el caso de '*missasigned_room*', la cual nos pareció útil posteriormente.

Adentrado en lo que respecta a la confección, entrenamiento y evaluación de modelos, pudimos concluir que la mejor performance fue de un Random Forest hiperparametrizado a partir de Grid Search Cross Validation.

La razones por las cuales consideramos que ocurrió están vinculadas al tipo de modelo y la forma de validación y búsqueda de hiperparámetros. Por un lado, Random Forest es un ensamble de árboles de decisión, por lo que su performance de por sí ya es mucho mejor a cualquiera de estos por separado, pero la duda surge en por qué es mejor que otro tipo de ensambles. Ahí es donde entra la naturaleza del ensamble, es decir, nos encontramos con un ensamble homogéneo entrenado por bagging, lo cual significa que el entrenamiento de los diferentes modelos ensamblados se da en paralelo y por ende significa menor complejidad computacional. Dicha ventaja en complejidad es la que logró que este ensamble pudiera ser hiperparametrizado a partir de Grid Search, una búsqueda mucho más exhaustiva y precisa que su contraparte Randomized Search, pero a la vez demandante en complejidad. Creemos que un ensamble XGBoost con hiperparametros por Grid Search hubiese tenido un mejor rendimiento, de hecho lo intentamos hacer mediante una reducción de dimensionalidad y aún así tuvo un tiempo exagerado de entrenamiento en comparación, por lo que no pudo ser evaluado.

Finalmente, para el caso de las redes neuronales, estudiamos diferentes tipos de arquitecturas, relaciones entre capas y neuronas, funciones de activación, optimización y pérdida, e hiperparametros. Se pudo observar como la manipulación de los hiperparametros como la cantidad de lotes o épocas impactan fuertemente en el rendimiento temporal de entrenamiento así como también en el resultado de las métricas, sin embargo un abuso en la cantidad de épocas llevó a una pobre generalización. También notamos como la correcta elección de las funciones previamente comentadas juegan un rol fundamental para el desarrollo de nuestra red, esta elección dependerá únicamente del tipo de problema que enfrentemos.