Checkpoint 4 - Grupo 32

Introducción

Decidimos mantener el dataset que venimos utilizando, con las mismas columnas removidas y feature engineering aplicado.
Para encontrar la mejor combinación de hipers para la red neuronal, usamos K-Fold Cross Validation. Intentamos optimizar la conexión entre neuronas y si bien se puede ver en la notebook el código funcional, no pudimos resolver un error relacionado a la dimensionalidad según lo que entendimos que nos impidió usar conexiones como LSTM o Conv2D.
Para buscar la cantidad de epochs y número batch_size iteramos manualmente y vimos aquel método que maximice f1-score .
Decidimos utilizar la función sigmoidea como neurona final por la simplicidad que brinda y la naturaleza binaria de nuestro problema de clasificación..

Construcción del modelo

Detallar como mínimo los siguientes puntos para el “mejor” modelo obtenido:

- Capa entrada
  - 120
  - Relu
  - Densa
- Capas ocultas
  - 1 capa
  - 50
  - Relu
  - Densa
- Capa de salida
  - 1 (forzado por la naturaleza del problema)
  - Sigmoidea

Optimizamos los hiper parámetros relacionados a la arquitectura, decidimos dejar los del optimizador de Back Propagation fijos ya que no nos resultó necesario optimizar el tiempo, usamos **Nadam** como optimizador.

Optimizamos la cantidad de neuronas para la entrada y capas ocultas de manera independiente, así también las funciones de activación para ambas. Sólo optimizamos el learning rate porque es una mejora significativa en rendimiento y puede causar problemas si está mal seteada (valores muy cercanos a 1). También buscamos el número óptimo de capas ocultas para nuestra red.

Introdujimos una función que simula un "embudo" de neuronas, también pusimos la posibilidad de utilizar esta función como un hiper parámetro extra.

Decidimos no utilizar técnicas de regularización ya que pudimos observar que obtuvimos scores muy similares en train y test al igual que en Kaggle hablando de f1-score por lo que una técnica de regularización que es usada para overfitting opinamos (y probamos manualmente) que no mejora resultados.

Utilizamos 100 ciclos de entrenamiento con 3 folds para cada, asi teniendo 100 combinaciones de hipers probadas y 300 iteraciones de entrenamiento.

Cuadro de Resultados

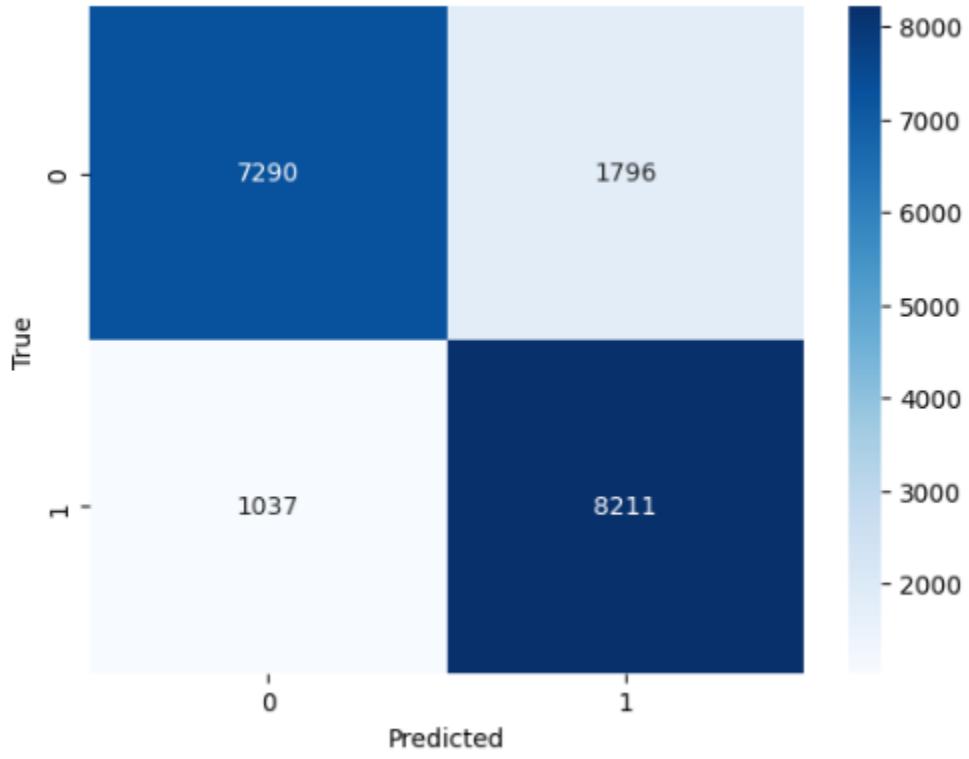
Realizar un cuadro de resultados comparando los modelos que entrenaron (entre ellos debe figurar cuál es el que seleccionaron como mejor predictor). Confeccionar el siguiente cuadro con esta información:

Medidas de rendimiento en el conjunto de TEST:

Modelo	F1-Test	Presicion Test	Recall Test	Kaggle
modelo_1	0.800	0.75	0.853	0.784
modelo_2	0.852	0.81	0.842	0.844
modelo_3	0.853	0.82	0.887	0.845

Modelo_1: No optimizamos hiperparámetros
Modelo_2: Cross Validation - Número de neuronas igual para todas las capas, función de activación sigmoide para todas las neuronas. Optimizamos a mano.
Modelo_3 (mejor resultado): Optimización de hiperparámetros tal como mencionamos arriba, con la cantidad mencionada de folds e iteraciones. Decidimos la cantidad de epochs y batch_size probando a mano este mejor modelo, utilizamos una función propia para definir el umbral de decisión final según la probabilidad resultante.

Matriz de Confusion



Observamos que el modelo predice en su mayoría positivos, lo que justifica el alto Recall por tener menor cantidad de FN

Tareas Realizadas

Integrante	Tarea
Daniel Agustin Marianetti	Armado de modelos Optimizacion de hiperparámetros Creación de funciones custom para análisis
Ezequiel Lazarte	Confección de la notebook Optimizacion de hiperparámetros Armado del reporte
Franco Ezequiel Rodriguez	Armado de modelos Optimizacion de hiperparámetros Armado del reporte