Realizamos distintos modelos para redes neuronales. Inicialmente, tratamos los datos necesarios, luego creamos una red neuronal bastante simple, con una sola capa de inputs y una de salida, la que únicamente tiene 2 neuronas. La capa de inputs tiene 68 entradas, ya que esa es la cantidad de variables con las que queda nuestro set de datos.
Luego, creamos una función que calcula el F1 score, ya que “keras” no la tiene implementada ya que es la métrica que más nos interesa a nosotros. Para esta función, la información fue obtenida del siguiente paper: paper f1 score en keras.

Después, realizamos una función llamada create_model, la que recibe distintos parámetros para la creación inicial del modelo y luego la optimización de los hiperparametros del mismo. Esta utiliza la función de activación ReLU. Para optimizar los modelos mediante cross validation se utilizan los siguientes hiperparametros:

- 'optimizer': Especifica el algoritmo de optimización que se utilizará para entrenar el modelo. Usamos 'Adam’ y 'Nadam'.
- 'activation': Este parámetro define la función de activación que se utilizará en las capas ocultas del modelo. Las opciones que seleccionamos son 'relu' y 'sigmoid'.
- 'neurons', 'neurons_2', 'neurons_3': Estos parámetros especifican el número de neuronas en cada capa oculta del modelo. Controlan la capacidad y complejidad del modelo.

Decidimos no optimizar con random search epochs y batchs, ya que esto nos llevaría demasiado tiempo.

A través de estos, creamos 3 modelos de redes neuronales, el primero utilizando regularización L2, activación softmax para la última capa, para las capas ocultas usa ReLU. La cantidad de neuronas decrece a medida que se va avanzando a las capas ocultas. Los resultados obtenidos fueron con un F1 cercano tanto en test como en train. Por lo que podemos concluir que previene el overfitting.

El segundo modelo, no tiene regularización,como optimizador utiliza ‘Adam’, como función de activación usa sigmoid. Los resultados obtenidos fueron un F1 score muy alto en train, 94 y en kaggle dió 50. Podemos ver que ocurrió un gran sobre ajuste.

El tercer modelo, primero obtenemos los mejores hiperparametros, luego para realizar el entrenamiento por epochs, lo realizamos con Early Stop, al que le asignamos como patience el valor 12. Los resultados obtenidos fueron balanceados, con falsos positivos y falsos negativos muy similares, aproximadamente 850. Y con un F1 cercano tanto en test como en train. Por lo que podemos concluir que previene el overfitting gracias a Early stop, ya que permite que el modelo detenga el entrenamiento cuando ya no mejora en 12 epochs consecutivas.