Resultados

```
Epoch 1/10
1500/1500 - 3s - loss: 0.5277 - accuracy: 0.7716 - 3s/epoch - 2ms/step
Epoch 2/10
1500/1500 - 2s - loss: 0.5159 - accuracy: 0.7786 - 2s/epoch - 1ms/step
Epoch 3/10
1500/1500 - 2s - loss: 0.5156 - accuracy: 0.7788 - 2s/epoch - 1ms/step
Epoch 4/10
1500/1500 - 2s - loss: 0.5160 - accuracy: 0.7789 - 2s/epoch - 2ms/step
Epoch 5/10
1500/1500 - 2s - loss: 0.5158 - accuracy: 0.7788 - 2s/epoch - 2ms/step
Epoch 6/10
1500/1500 - 2s - loss: 0.5161 - accuracy: 0.7785 - 2s/epoch - 1ms/step
Epoch 7/10
1500/1500 - 2s - loss: 0.5158 - accuracy: 0.7788 - 2s/epoch - 2ms/step
Epoch 8/10
1500/1500 - 2s - loss: 0.5160 - accuracy: 0.7788 - 2s/epoch - 1ms/step
Epoch 9/10
1500/1500 - 2s - loss: 0.5154 - accuracy: 0.7788 - 2s/epoch - 1ms/step
Epoch 10/10
1500/1500 - 2s - loss: 0.5157 - accuracy: 0.7787 - 2s/epoch - 1ms/step
Best: 0.778800 using {'batch_size': 20, 'epochs': 10}
0.778800 (0.000000) with: {'batch_size': 20, 'epochs': 10}
0.778800 (0.000000) with: {'batch_size': 20, 'epochs': 40}
0.221200 (0.000000) with: {'batch_size': 20, 'epochs': 60}
0.778800 (0.000000) with: {'batch_size': 40, 'epochs': 10}
...
0.778800 (0.000000) with: {'batch_size': 40, 'epochs': 60}
0.778800 (0.000000) with: {'batch_size': 60, 'epochs': 10}
0.762300 (0.023335) with: {'batch_size': 60, 'epochs': 40}
0.778800 (0.000000) with: {'batch_size': 60, 'epochs': 60}
```

Resultados Grid Search

```
Epoch 1/10
1500/1500 - 3s - loss: 0.5284 - accuracy: 0.7692 - 3s/epoch - 2ms/step
Epoch 2/10
1500/1500 - 3s - loss: 0.5164 - accuracy: 0.7788 - 3s/epoch - 2ms/step
Epoch 3/10
1500/1500 - 3s - loss: 0.5163 - accuracy: 0.7788 - 3s/epoch - 2ms/step
Epoch 4/10
1500/1500 - 3s - loss: 0.5165 - accuracy: 0.7788 - 3s/epoch - 2ms/step
Epoch 5/10
1500/1500 - 3s - loss: 0.5167 - accuracy: 0.7788 - 3s/epoch - 2ms/step
Epoch 6/10
1500/1500 - 3s - loss: 0.5165 - accuracy: 0.7788 - 3s/epoch - 2ms/step
Epoch 7/10
1500/1500 - 3s - loss: 0.5156 - accuracy: 0.7788 - 3s/epoch - 2ms/step
Epoch 8/10
1500/1500 - 3s - loss: 0.5163 - accuracy: 0.7788 - 3s/epoch - 2ms/step
Epoch 9/10
1500/1500 - 3s - loss: 0.5154 - accuracy: 0.7788 - 3s/epoch - 2ms/step
Epoch 10/10
1500/1500 - 3s - loss: 0.5155 - accuracy: 0.7788 - 3s/epoch - 2ms/step
Best: 0.778800 using {'epochs': 10, 'batch_size': 20}
0.778800 (0.000000) with: {'epochs': 10, 'batch_size': 20}
0.407067 (0.262855) with: {'epochs': 40, 'batch_size': 20}
0.592933 (0.262855) with: {'epochs': 60, 'batch_size': 20}
0.778800 (0.000000) with: {'epochs': 10, 'batch_size': 40}
...
0.592933 (0.262855) with: {'epochs': 60, 'batch_size': 40}
0.778800 (0.000000) with: {'epochs': 10, 'batch_size': 60}
0.778800 (0.000000) with: {'epochs': 40, 'batch_size': 60}
0.778800 (0.000000) with: {'epochs': 60, 'batch_size': 60}
```

Resultados Random Search

Universidad del Valle de Guatemala
Deep Learning y Sistemas Inteligentes
Sección 20

Laboratorio 4

Andrés de la Roca 20332
Jun Woo Lee 20358

Descripción de la red

La red neuronal se compone de las siguientes capas:

Capa de Entrada (Input Layer): La red toma un vector de entrada de 5 dimensiones, correspondiente a las características del conjunto de datos.

Capa Densa (Dense Layer 1): Esta capa consta de 64 neuronas y utiliza la función de activación ReLU (Rectified Linear Unit). Su propósito es aprender representaciones no lineales de las características de entrada.

Capa de Normalización por Lotes (Batch Normalization 1): Se aplica normalización por lotes después de la primera capa densa para mejorar la estabilidad y acelerar la convergencia del entrenamiento.

Capa Densa (Dense Layer 2): Esta capa consta de 32 neuronas y utiliza la función de activación ReLU. Continúa extrayendo características más complejas de los datos.

Capa de Normalización por Lotes (Batch Normalization 2): Se aplica normalización por lotes después de la segunda capa densa para mantener una distribución de activaciones estable.

Capa Densa (Dense Layer 3): Consta de 16 neuronas y utiliza la función de activación ReLU. Su objetivo es aprender representaciones más abstractas y de menor dimensionalidad.

Capa de Normalización por Lotes (Batch Normalization 3): Se aplica normalización por lotes después de la tercera capa densa.

Capa de Salida (Output Layer): Esta capa consta de una sola neurona con una función de activación sigmoide. La función sigmoide produce una probabilidad de clasificación binaria.

La red neuronal se entrena utilizando los hiperparametros siguientes:
La función de pérdida de entropía cruzada binaria, que es adecuada para tareas de clasificación binaria. El optimizador utilizado es Adam, que es un algoritmo de optimización eficiente. Además, se registra la métrica de precisión durante el entrenamiento para evaluar el rendimiento del modelo.

Métodos por el cual se llego a la red

El desarrollo de redes neuronales se basa en un enfoque metódico y basado en datos. El primer paso importante es el preprocesamiento de datos. Antes de cualquier esfuerzo de modelado, los datos se importaron desde un archivo de Excel llamado default_cc_clients.xlsx. Una vez introducidos en el entorno laboral, pasan por un proceso de limpieza y estructuración. Se eliminaron las columnas que no contribuían al análisis en cuestión y se identificaron claramente las variables de entrada y salida. Este paso es esencial

porque para que una red neuronal funcione de manera óptima, requiere datos claros y bien estructurados.

Después de asegurarse de que los datos estuvieran en el formato correcto, se determinó la arquitectura de la red. Se eligió una estructura que consta de tres capas densas, cada una seguida de una capa de normalización masiva. El objetivo de estas capas es aprender representaciones de alto nivel de los datos, mientras que las capas de normalización por lotes garantizan una convergencia rápida y estable del modelo durante el entrenamiento. Finalmente, la red se completó con una capa de salida equipada con una función de activación sigmoidea, lo que la hace ideal para tareas de clasificación binaria. Sin embargo, la creación de modelos no se limita a definir su arquitectura. La optimización de hiper parámetros es uno de los pasos más importantes en el desarrollo de redes neuronales. Para el modelo mencionado, se utilizaron técnicas de búsqueda en cuadrícula y búsqueda aleatoria para encontrar la mejor combinación de número de épocas y tamaño de lote. Estas técnicas de optimización permiten explorar de forma exhaustiva o estocástica un espacio de hiper parámetros predefinido para encontrar la combinación que proporcione el mejor rendimiento.

<u>Mejoras a la red</u>

- Exploración de Arquitecturas: Experimentar con diferentes números de capas, neuronas por capa, o tipos de capas puede revelar estructuras más efectivas para el problema específico.
- Optimizadores y Tasas de Aprendizaje: Ajustar la tasa de aprendizaje, ya sea manualmente o a través de una tasa de aprendizaje variable, puede mejorar la convergencia del modelo.
- Expansión de la Búsqueda de Hiper Parámetros: La inclusión de otros hiper parámetros, como tasas de aprendizaje, tasas de dropout, o diferentes funciones de activación, podría mejorar el rendimiento.