INSTITUTO TECNOLOGICO DE COSTA RICA

Proyecto Final - Optimización Evolutiva para una Aplicación Neuronal

MT8008 - Inteligencia Artificial

Ernesto Pocasangre Kreling

Andrew Eliecer Vargas Puffenberger

Prof. Juan Luis Crespo Marino, Dr. Ing.

Fecha de entrega: 21 de noviembre de 2023

Índice

1. Descripción del Problema

El problema consiste en un problema de optimización que pretende encontrar un arreglo de valores que cumplan la condición de calidad solicitada. Los valores corresponden a los hiperparámetros seleccionados para la elaboración y el entrenamiento de una red neuronal artificial o ANN. Esta red neuronal realiza una labor de clasificación de 4 posibles categorías, a partir de 24 entradas. Se pide desarrollar un algoritmo evolutivo que optimice estos hiperparámetros, tal que la red neuronal final realice la clasificación de la mejor manera.

2. Análisis e Interpretación del Problema

Como se mencionó previamente, el problema a tratar se compone de dos capas. La primera capa corresponde al problema de clasificación como tal, y la segunda al problema de optimización de parámetros de la red neuronal.

2.1 Problema de Clasificación

Un problema de clasificación se refiere al problema en el que se pretende asignar etiquetas o categorías a un grupo de entradas, basándose en patrones. Estos patrones o relaciones se establecen en un periodo de entrenamiento, para el cual se requiere de un conjunto de datos, o dataset, que contenga tanto la información de las entradas, como sus etiquetas. A esto se le conoce como un conjunto de datos etiquetado o labeled dataset. Puede ser clasificación binaria, donde se asigna una de dos clases posibles, o multiclase, con más de dos clases.

En la terminología de aprendizaje automático o machine learning, la clasificación se considera como el aprendizaje a partir de ejemplos. Se debe de disponer de un conjunto de entrenamiento compuesto por observaciones correctamente identificadas, y estas se utilizan para iterar sobre un modelo que describe y distingue entre las clases. A dicho modelo se le denomina clasificador. [1] Este tipo de entrenamiento o de aprendizaje se conoce como supervisado.

El caso específico del problema consiste en datos recopilados por un robot llamado SCITOS G5 [2], que se moviliza en un cuarto siguiendo la pared en un sentido de las manecillas del reloj. Este proceso se realizó 4 veces, utilizando 24 sensores ultrasónicos colocados de

manera circular en el robot. Los sensores tenían un ángulo de separación de 15 grados entre ellos. El conjunto de datos consiste en las mediciones de estos 24 sensores, y la categoría correspondiente a la acción por tomar según a su posición y orientación en el cuarto. Las 4 categorías corresponden a: moverse hacia adelante, girar ligeramente hacia la derecha, girar bruscamente hacia la derecha, y girar ligeramente hacia la izquierda. El objetivo para este problema de clasificación es predecir de manera correcta la acción determinada a partir de los valores medidos en los sensores ultrasónicos. Los datos de los sensores fueron muestreados a una taza de 9 muestras por segundo.

2.2 Problema de hiperparámetros Red Neuronal

Una red neuronal artificial (ANN) es un modelo computacional conexionista inspirado en la estructura y funcionamiento del cerebro humano. Está compuesta por neuronas artificiales o perceptrones individuales. El perceptrón de Rosenblatt es una "red de nervios" que consiste en elementos neuronales lógicamente simplificados. Se ha demostrado que esta unidad computacional es capaz de aprender a discriminar y reconocer patrones perceptuales. [3] En un modelo de Perceptrón Multicapa o MLP denso, se emplea el uso de múltiples capas de varios perceptrones, entre las cuales todas las unidades de una capa se conectan con todas las unidades de la siguiente capa. A continuación, en la figura 1 se puede observar una representación gráfica de un MLP de ejemplo, que cuenta con 2 capas ocultas, 10 neuronas en cada capa, y 2 neuronas de entrada.

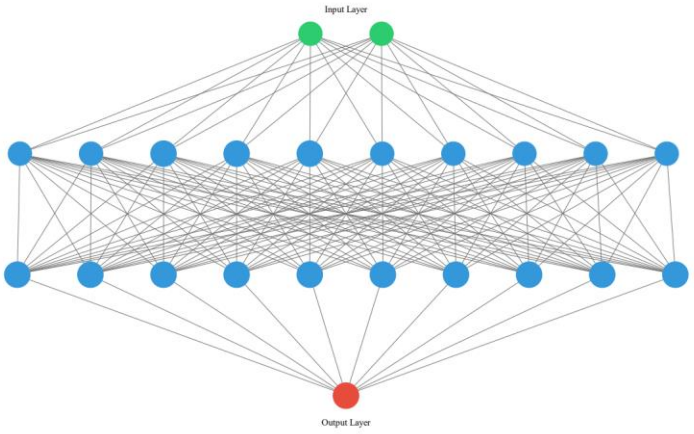

Figura 1. Ejemplo de Red Neuronal Artificial

La red debe de ser entrenada mediante un conjunto de datos o dataset. Luego de esto a la hora de hacer una inducción, la información fluye a través de la red desde la capa de entradas, en donde se introducen los datos, hasta la capa de salida, donde se produce la predicción o la clasificación. Cada conexión entre neuronas tiene un peso que se ajusta durante el entrenamiento del modelo. Estos pesos, normalmente son inicializados de una manera aleatoria, o por una técnica similar, y se van modificando según algún algoritmo de optimización. Los algoritmos de optimización para redes neuronales normalmente incluyen algún tipo de computación de gradiente del error en la salida del modelo, que se transmite neurona por neurona de adelante hacia atrás en lo que se conoce como Back Propagation.

Las ANN han demostrado ser muy útiles en tares complejas en las que se tiene alguna componente de reconocimiento de patrones. Entre las aplicaciones principales de este tipo de modelo se encuentran: visión por computadora, procesamiento de lenguaje natural, sistemas de recomendación, sistemas generativos, etc.

Los hiperparámetros en una red neuronal son valores que influyen en su estructura y comportamiento durante el proceso de su definición y de su entrenamiento. A diferencia de los pesos que se aprenden internamente, los hiperparámetros deben ajustarse de antemano solo una vez y no son modificados luego de esto. Entre los hiperparámetros más comunes se tienen la tasa de aprendizaje, que determina el tamaño de los ajustes realizados a los pesos en cada iteración, el número de capas, la cantidad de neuronas en cada capa, el tipo de función de error, entre otras. La elección adecuada de hiperparámetros es de suma importancia, ya que puede afectar significativamente el rendimiento y la capacidad de aprendizaje de la red neuronal. Experimentar con diferentes configuraciones de hiperparámetros es una parte importante del proceso de desarrollo de modelos de redes neuronales para lograr un rendimiento adecuado en la tarea deseada.

Las características solicitadas de hiperparámetros con respecto a la red neuronal a construir son las siguientes:

- Número de capas ocultas: de 1 a 6
- Número de neuronas por capa oculta: de 1 a 14
- Tasa de aprendizaje: de 0.00001 a 0.5
- Tipo de optimizador: al menos tres tipos diferentes

- Uso de término de momento: binario (si o no)

2.3 Análisis Exploratorio del Conjunto de Datos

Para entender de mejor manera el problema y el conjunto de datos, se decidió hacer un análisis exploratorio del dataset. El análisis exploratorio del conjunto de datos, también conocido como EDA, implica examinar y comprender las características fundamentales de los datos antes de aplicar modelos o algoritmos. Esto es de suma importancia ya que puede dar a conocer características importantes del dataset, como por ejemplo valores anómalos, o patrones interesantes, que pueden ser tomados en cuenta a la hora de plantear el modelo final.

Los datos fueron cargados a un dataframe de pandas, para tener mayor facilidad a la hora de realizar el análisis. Algunas características importantes que se averiguaron incluyen que el conjunto de datos no contiene datos faltantes, para ninguna columna. La cantidad de filas fue de 5456, en donde cada una corresponde a una medición tomada. Hay 25 columnas, de las cuales 24 corresponden a los valores tomados de los sensores ultrasónicos, y la última columna corresponde a la etiqueta o categoría asignada. Los valores de las mediciones van aproximadamente de 0 a 5, lo cual probablemente se debe a un voltaje medido. Las categorías fueron codificadas como b1, b2, b3, y b4.

Se realizo un análisis preliminar de la distribución de las mediciones, según las 4 categorías diferentes. Es decir, se analizó el balance del conjunto de datos. Para esto se contaron las instancias de cada tipo de categoría y se graficó este número como se puede observar en la figura 2.

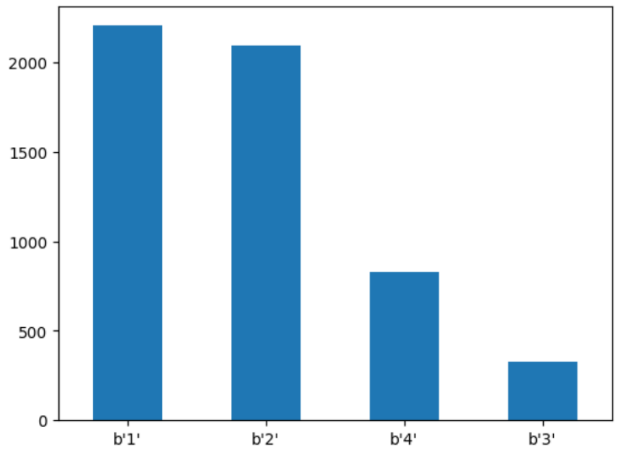

Figura 2. Distribución del Conjunto de Datos

Es importante destacar el desbalance presente en el dataset. Existen muchas más instancias para la acción 1 y la acción 2, que para la acción 3 o la acción 4. Como el conjunto de datos tampoco es significativamente grande, esto puede causar problemas a la hora de realizar el entrenamiento y validación del sistema, ya que los datos pueden no ser distribuidos de la mejor manera al hacer la partición del conjunto.

3. Descripción de solución

3.1 Entorno de Implementación

Para la implementación de la solución se utilizó el lenguaje de programación Python por medio de un notebook virtual en la nube de Google Colab. Para acceder a las funcionalidades relacionadas con modelos conexionistas de redes neuronales se empleó el uso de la librería Keras, la cual está construida encima de Tensorflow. Tensorflow es una plataforma que permite la construcción, el entrenamiento, y la validación de sistemas de aprendizaje automático. [4] Keras es una librería que funciona como API o interfaz de aplicación de alto nivel, facilitando el desarrollo de código que es fácil de entender y de modificar. [5] Estas librerías juntas permiten generar código rápido, eficiente, y sencillo.

La librería que se empleó para obtener las funcionalidades de computación evolutiva y de algoritmos genéticos concretamente fue DEAP. DEAP (Distributed Evolutionary Algorithms in Python) es una biblioteca de Python diseñada para implementar algoritmos evolutivos, incluidos algoritmos genéticos, programación genética y estrategias evolutivas. Fue diseñada para el prototipado rápido y prueba de ideas. Busca hacer explícitos los algoritmos y transparentes las estructuras de datos relacionados a este entorno. [6] Esta librería proporciona herramientas y estructuras de datos para la creación y manipulación eficiente de poblaciones, operadores evolutivos y evaluación de funciones de aptitud.

3.2 Red Neuronal Artificial

Como se explicó anteriormente la red neuronal artificial es un paradigma de tipo conexionista se ha venido utilizando cada vez más en los últimos años. Se compone de capas de neuronas que realizan computaciones simples cada una, pero al combinar su desempeño pueden llegar a aprender patrones nuevos y muy complicados. En la actualidad estos modelos han tenido un gran auge debido a la disponibilidad de grandes conjuntos de datos, y debido al gran poder computacional que se ha liberado.

El uso de una red neuronal para el problema concreto de clasificación es muy acertado debido a muchas razones. El hecho de que se tengan 24 entradas diferentes multiplica la complejidad del problema considerablemente. Esta complejidad puede ser tratada con mayor facilidad por un modelo conexionista de red neuronal que, con otros paradigmas, ya que básicamente consiste en operaciones de matrices. El hecho de que la solución puede generalizar su aprendizaje a datos nuevos es algo muy valioso. Esto debido a que alguna anomalía en el entorno se podría presentar, y el robot tomaría una decisión para ver qué acción es la más adecuada, su funcionalidad no pararía por anomalías de este tipo. Incluso el robot se podría colocar en un cuarto completamente nuevo, y su comportamiento para seguir la pared, girar en las esquinas y demás, debería ser extrapolado.

El hecho de que una red neuronal sea básicamente una seria de multiplicaciones de matrices, permite desplegar modelos computacionales de estas redes neuronales en sistemas físicos no tan complejos, como por ejemplo el chip del robot en cuestión. Esto permite la reacción del robot a tiempo real, y no solo de manera de análisis de datos. Otra ventaja importante es el aprovechamiento del conjunto de datos. Mientras más datos se tengan mejor (siempre y cuando su calidad sea decente). Esto quiere decir también que, si en algún momento se tienen más datos, se puede entrenar el sistema bajo este nuevo conjunto.

3.2.1 Hiperparámetros

Ya definido el paradigma a utilizar de red neuronal artificial, es importante establecer los Hiperparámetros del sistema de igual manera. Según los requerimientos dados ya se definieron algunos parámetros como se explicó previamente. Estos son los de número de capas ocultas (de 1 a 6), número de neuronas por capa oculta (de 1 a 14), tasa de aprendizaje

(de 0.00001 a 0.5), tipo de optimizador (al menos tres tipos diferentes), uso de término de momento (si o no).

Con respecto a los tipos de optimizador se decidió optar por Descenso de Gradiente Estocástico o SGD, optimizador Adam, y optimizador AdamW. El SGD es un optimizador que calcula el gradiente de la función de costo en cada iteración por medio de mini lotes (batches) aleatorios de datos en cada paso. Este enfoque estocástico introduce una variabilidad que puede ayudar a evitar mínimos locales y acelerar la convergencia.

El optimizador Adam (Adaptive Moment Estimation) es un algoritmo para la optimización de funciones objetivo estocásticas basado en gradientes de primer orden, se fundamenta en estimaciones adaptativas de momentos. El método es fácil de implementar, computacionalmente eficiente, con bajos requisitos de memoria, y adecuado para problemas con conjuntos de datos y/o parámetros grandes. [7] Adam ha demostrado ser eficaz en una variedad de tareas y se ha vuelto popular en la práctica debido a su rendimiento sólido y a su capacidad para adaptar diferentes parámetros. Adam es uno de los optimizadores más utilizados en la actualidad.

AdamW es una variante del optimizador Adam que incorpora una corrección de peso en la actualización de los parámetros. El decaimiento de peso (que hace referencia a la W) es una técnica de regularización que penaliza los pesos más grandes en el modelo, ayudando a prevenir el sobreajuste.

Con respecto a la función de costo empleada se utilizó Categorical Cross Entropy. es una función de pérdida comúnmente utilizada en problemas de clasificación multiclase. Es particularmente adecuada para situaciones en las que cada instancia pertenece a una única clase entre varias clases mutuamente excluyentes. Esta función de pérdida compara la distribución de probabilidad de las predicciones con la distribución de probabilidad real de las etiquetas de clase.

En el tema de las capas de entrada y salida, estas son definidas por la naturaleza del problema. Como se tienen 24 mediciones de sensor en cada instancia, el número de neuronas en la primera capa del modelo es de 24. Para la capa final, se pretende clasificar la entrada a una

de 4 categorías, según la acción a tomar por el robot. Esto quiere decir que la capa de salida se debe componer de 4 neuronas.

La función de activación empleada al final de la combinación lineal para las capas ocultas fue elegida como Sigmoide. Es una función de activación que toma cualquier número real como entrada y produce una salida en el rango de 0 a 1. Con respecto a la capa final de salida, se eligió la función de activación Softmax. Esta función normaliza las salidas, y las representa en forma de probabilidades. Se toma la salida con mayor probabilidad como la predicción del modelo.

3.2.2 Aspectos adicionales

Para poder mejorar la capacidad de generalización y de entrenamiento de la red neuronal se decidió emplear la Normalización por Lote o Batch Normalization como se conoce en inglés. Esta es una técnica utilizada en redes neuronales para mejorar la estabilidad y el rendimiento durante el entrenamiento. Fue propuesta para abordar problemas como el desvanecimiento o explosión del gradiente y la inestabilidad en la convergencia especialmente para modelos profundos. [8] Básicamente el procedimiento desarrollado por la normalización por lote es el de normalizar con la técnica Z la salida de una capa para cada lote de ejemplos. Es decir, se calcula el promedio y la desviación estándar para cada lote, y se restan y dividen respectivamente. El uso de esta técnica tiene muchas ventajas, entre ellas se encuentran principalmente: mayor estabilidad del entrenamiento, aceleración del entrenamiento, reducción de sensibilidad a la inicialización, y un efecto regularizador implícito.

A parte de esta normalización por lote se utilizó una técnica llamada EMA para mejorar los resultados del modelo igualmente. EMA se refiere a la Media Móvil Exponencial, y básicamente es una técnica utilizada para suavizar series temporales o secuencias de datos. En el contexto del entrenamiento de redes neuronales, la EMA se utiliza a menudo para suavizar las actualizaciones de los parámetros del modelo, como los pesos durante el entrenamiento. Esto puede ayudar a mejorar la estabilidad del proceso de optimización y a evitar oscilaciones abruptas en la actualización de los pesos.

3.3 Algoritmo Genético

Se decidió implementar un algoritmo genético para la solución del problema de optimización para la red neuronal, debido a que permite generar una gran variedad de combinaciones de valores y optimizarlos dependiendo de distintos criterios. Esto es justo el funcionamiento que se busca en un sistema automático de generación de hiperparámetros.

En computación evolutiva, un algoritmo genético es una técnica de optimización inspirada en la evolución biológica que utiliza operadores como selección, cruce y mutación para buscar soluciones eficientes a problemas complejos. [9] Los algoritmos evolutivos son muy útiles en situaciones en las cuales se tiene un problema sobre el cual las soluciones planteadas se pueden evaluar, pero orientar la solución o resolver el sistema de una forma analítica no es posible o es sumamente complejo. Estos paradigmas requieren conocer solo el valor de la función a optimizar en cada punto, y no sus derivadas, algo que permite a esta técnica abordar un numero de problemas de optimización no tratables mediante métodos de gradientes. [10] Es importante mencionar que se obtienen soluciones muy buenas, más sin embargo no son óptimas debido a la naturaleza del algoritmo como tal.

Para implementar un algoritmo genético se requieren de cuatro componentes fundamentales. El primer componente es un problema cuyas soluciones puedan ser expresadas como un vector. El problema fue extensamente explicado en la sección anterior. Luego de esto se encuentra el cromosoma, que es la codificación de la solución en un vector. Los componentes o valores del vector se conocen como los genes. También se debe de definir un espacio de alelos, el cual se refiere a los rangos posibles para los diferentes valores de los genes. Por último, se requiere de una función de calidad que permita evaluar de forma absoluta o relativa cada solución o individuo.

3.3.1 Definición de Cromosomas

Con respecto a la definición o codificación de los cromosomas se deben de considerar las variables necesarias e independientes que caracterizan la solución. Los requisitos de hiperparámetros para la red neuronal representan las variables independientes que se variarán en cada individuo, por lo tanto, se elige el siguiente orden para los genes del cromosoma:

1. Número de capas ocultas

2. Número de neuronas por capa oculta
3. Tasa de aprendizaje
4. Tipo de optimizador
5. Uso de término de momento

3.3.2 Definición Espacio de Alelos

El espacio de alelos representó un problema interesante en el proceso de definición, específicamente por el proceso de mutación. Los sistemas de mutación que se disponen de en DEAP priorizaban sistemas de mutación que se aplicaban a todos los genes por igual, pero estos no serían compatibles con genes de distintos tipos. Para solucionar esto, se decidió utilizar valores normalizados del 0 al 1 para el espacio de alelos, y mapear estos valores en el inicio de la función de calidad. Esto significa que el espacio de alelos va desde un valor muy cercano a 0, hasta un valor muy cercano a 1 para cada gen. Los extremos no se usan debido a generar problemas en las funciones de transformación de normalizado a real. La definición de los alelos individuales son los siguientes:

1. Numero de capas ocultas: va entre 1 y 6, se usa la función $\lfloor 6x + 1 \rfloor$
2. Números de neuronas por capa: va entre 1 y 14, se usa la función $\lfloor 14x + 1 \rfloor$
3. Tasa de aprendizaje: va desde 0.000005 a 0.5, se usa la función $\frac{10^{5x-5}}{2}$, esta función $\frac{10^{5x-5}}{2}$ transforma exponencialmente el gen normalizado, esto debido a que se necesitaba probar valores muy pequeños que, aunque se encontraban en el rango normalizado, casi nunca iban a aparecer. Además, el proceso de mutación es lineal, en el sentido que agrega valores mayoritariamente en el mismo orden que la desviación estándar, por lo que sería muy improbable terminar de explorar toda la zona de posibilidades.
4. Tipo de optimizador: va desde 1 a 3, se usa la función $\lfloor 3x + 1 \rfloor$. El 1 representa SGD, el 2 Adam, y el 3 AdamW.
5. Uso de termino de momento: si es menor que 0.5 se toma como 0, si es mayor, se toma como 1, se usa la función $\lfloor x + 0{,}5 \rfloor$.

3.3.3 Función de Calidad

Para la función de calidad se realizan varios pasos debido a la complejidad del sistema de cálculo de calidad de los hiper parámetros. Se decidió realizar la creación, entrenamiento, y validación de la red neuronal que definen los hiper parámetros del individuo a una cantidad baja de epochs. Esto permite obtener una "imagen previa" del funcionamiento de la red neuronal entrenada por más tiempo. Se decidió usar el parámetro f1 para representar la calidad de la red neuronal. El f1 Score es una métrica crucial en problemas de clasificación que combina la Precisión y el Recall en una única medida. La Precisión evalúa la proporción de instancias positivas correctamente clasificadas entre todas las clasificadas como positivas, mientras que Recall mide la proporción de instancias positivas correctamente clasificadas entre todas las instancias realmente positivas. Se utilizó el promedio macro, ya que este se puede usar cuando las clases están balanceadas, proceso que se logró en el preprocesado.

Las etapas de la función de calidad son las siguientes:

- Transformación de genes en hiper parámetros. Se toman los valores de los genes individuales y se mapean en los valores que se usaran para los hiperparámetros. Estos se pasan a la siguiente función para la construcción de la red neuronal.
- Construcción del modelo. Se usan los hiperparámetros para generar el modelo correspondiente para ser entrenado.
- Entrenamiento del modelo. Se usa el modelo creado en la función anterior y los datos preprocesados para entrenar el modelo. Este se entrena por 10 epochs para reducir el tiempo de ejecución de la función de calidad.
- Verificación del modelo. Se evalúa la red neuronal para obtener el valor de f1 score que se retorna como el valor de calidad del individuo. Se guardan las demás medidas de Precisión y Recall, pero el f1 score es la medida que se toma como referencia.

El valor que se retorna es un número entre 0 y 1, donde 1 representa una red neuronal con precisión y recall perfecto, y donde 0 representa una red neuronal que no predice de manera correcta ningún valor. Por lo que el sistema evolutivo representa un sistema de maximización del valor de calidad.

3.4 Preprocesado

Con respecto al procesado de datos, antes de ser introducidos a la red neuronal como tal, se emplearon algunas técnicas que permitieron el mejor funcionamiento del sistema. Entre estas técnicas, se encuentran el balanceado mediante la técnica SMOTE, el normalizado, y la división del conjunto de datos.

3.4.1 Balanceado SMOTE

SMOTE, que significa "Synthetic Minority Over-sampling Technique", es una técnica utilizada para abordar el desequilibrio de clases en un conjunto de datos generando sintéticamente muestras para la clase minoritaria o menos representada. Esto quiere decir que este es un método de over-sampling, ya que genera datos nuevos, en lugar de borrar datos existentes. SMOTE opera creando instancias sintéticas de la clase minoritaria mediante la interpolación entre instancias de esa clase. [11]

Al aplicar SMOTE, se busca obtener un conjunto de datos más equilibrado al generar muestras adicionales para la clase minoritaria, mejorando así la capacidad del modelo para aprender patrones en todas las clases de manera más equitativa. Esta técnica es particularmente útil en situaciones donde la clase minoritaria contiene información valiosa, pero está subrepresentada en comparación con la clase mayoritaria. También es útil que se hayan creado más instancias en el dataset, siendo un conjunto relativamente pequeño de datos. Cuando se realiza la división del conjunto para entrenamiento, test, y validación, es más probable que cada partición este bien representada. A continuación, se puede observar en la figura 3 el conjunto de datos luego de haberse realizado el balanceo de datos mediante SMOTE.

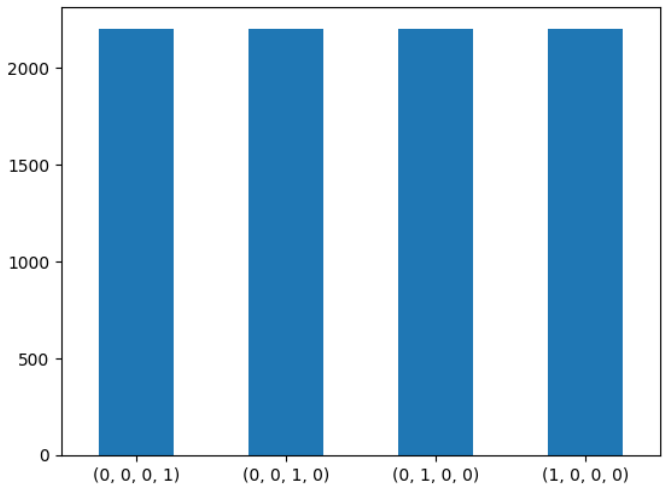

Figura 3. Conjunto de Datos Balanceado

3.4.2 Normalizado

La normalización en un conjunto de datos es el proceso de ajustar los valores de las variables para que se encuentren dentro de una escala específica o sigan una distribución particular. La normalización ayuda a mejorar la convergencia de los modelos conexionistas y evita que variables con escalas grandes dominen aquellas con escalas pequeñas.

El tipo de normalizado que se empleó para el conjunto de datos se denomina normalizado Z. En esta técnica, al conjunto de datos se le resta su promedio, y posteriormente se divide entre su desviación estándar.

3.4.3 Partición Aleatoria del Conjunto de Datos

El dataset se partió en tres subconjuntos para ser utilizados en las siguientes fases de la implementación: Entrenamiento, Test, y Validación. La proporción utilizada al dividir los datos fue de aproximadamente 70:20:10. El conjunto de datos para entrenamiento es el utilizado por el modelo neuronal para modificar sus pesos y llegar a una mejor predicción. Este conjunto de datos se utiliza la cantidad de epochs que se desee. El conjunto de datos de test es el empleado cuando se finaliza cada epoch en el entrenamiento, para ir midiendo el desempeño tanto del modelo como del entrenamiento. Y por último, el conjunto utilizado

para validación es el que se emplea solo una vez al final de todo el proceso, para dar una nota final el modelo sobre datos que no ha visto anteriormente. Cabe destacar que esta división se hizo una vez que se randomizaron las filas en el dataset, para evitar cualquier tipo de sesgo a la hora de haber introducido los datos en el conjunto.

4. Estudio de Hiperparámetros

Para el estudio de hiper parámetros del algoritmo genético se decidieron tomar 3 combinaciones diferentes. Para definir las 3 combinaciones de hiper parámetros que se usaron se decidió basarse en el concepto de exploración vs explotación. Un algoritmo que se basa más en la exploración intenta mantener y buscar soluciones variadas al problema, priorizando la variabilidad. Esto permite tener un pool genético diverso. En cambio, si un algoritmo prioriza la explotación, este se acerca rápidamente a una solución o un conjunto de soluciones similares y deja por lado la variabilidad de las soluciones.

Para las 3 combinaciones a realizar, se decidió priorizar la explotación en uno, la exploración en otro, e intentar un método balanceado para el restante. Para cada hiperparámetro que se podía variar en el algoritmo genético se eligieron 3 valores organizándolos de mayor exploración a mayor explotación, y luego ser distribuidos en los 3 casos de estudio. A continuación, aparecen los valores elegidos para cada hiper parámetro.

- Población inicial. Una población inicial alta tiene dos efectos principales, lo primero es aumentar la variabilidad del sistema, debido a tener más genes iniciales de los que se pueden escoger para los individuos del evolutivo. Lo otro es que se aumenta la cantidad de tiempo que se necesita para correr el algoritmo. Se decidió usar 30 individuos es el de explotación, 40 en el balanceado, y 50 en el de explotación. El valor de 30 como valor mínimo se decidió para garantizar que la mayoría de las opciones de valores para los hiperparámetros de la red neuronal se encontrarían decentemente representadas en el pool genético inicial, y que no necesitarían del proceso de mutación para aparecer.

- Sistema de selección: Se eligieron 3 procesos de selección, estos fueron el torneo de 4 individuos, el de 2 individuos, y la ruleta. Entre estos, el torneo de 2 representa la mayor variabilidad, ya que hay más probabilidad de que seleccione un individuo de baja calidad. La ruleta puede ser el método más explotativo, ya que los individuos que pasan lo hacen con una probabilidad dependiente de su calidad de una manera más directa. El torneo de 4 se podría considerar como la opción balanceada, ya que como son grupos de 4, hay mayor probabilidad de elegir el de calidad más alta que en el torneo de 2, pero no es tan explotativo como la ruleta.
- Sistema de cruce: Se eligieron 3 procesos de cruce, estos fueron el cruce de dos puntos variables, el cruce de un punto variable, y el cruce por combinación lineal. Entre estos, el cruce de dos puntos genera más variabilidad que el de un punto, debido a cortar y recombinar el código genético en más puntos. El cruce por combinación lineal se considera como el más explotativo, ya que los descendientes tienden a parecerse a la combinación de ambos padres, acercando los distintos genes a uno solo, en vez de mantener la variabilidad de los genes de los padres.
- Desviación estándar y probabilidad de mutación: La mutación seleccionada fue la de mutación gaussiana, debido a la naturaleza de los genes y a su codificación. En general la probabilidad de mutación aumenta la variabilidad, creando nuevos valores de genes que no se habían considerado en la población inicial, o que habían desaparecido del pool de genes. La desviación estándar al aumentar significa una mutación que afecta en mayor proporción al gen (para el tipo de mutación Gaussiana), aumentando la variabilidad. La probabilidad de mutación directamente aumenta la cantidad de mutaciones.
- Probabilidad de combinación: la probabilidad de combinación define la probabilidad que tienen los individuos seleccionados de reproducirse, aumentar esto significa mayor cantidad de recombinaciones genéticas debido a cruce, generando mayor variabilidad.

Un resumen de los parámetros elegidos para cada prueba de hiperparámetros se encuentra a continuación en la tabla 1.

Tabla 1. Resumen de Parámetros elegidos para 3 configuraciones de Hiperparámetros

	Escenario 1 (Exploración)	Escenario 2 (Balanceado	Escenario 3 (Explotación)
Población Inicial	50	40	30
Selección	Torneo 2	Torneo 4	Ruleta
Cruce	1 punto variable	2 puntos Variables	Combinación Lineal
Mutación stdv	0.2	0.1	0.01
Mutación Gen prob	0.5	0.3	0.15
Mutación prob	0.5	0.3	0.15
Combinación prob	0.3	0.2	0.15

Se ejecuto el programa 3 veces para las 3 configuraciones descritas. Los archivos denominados Logbooks contienen información sobre la mejor y peor calidad, así como la calidad promedio y desviación estándar de cada generación. Las tablas con toda la información aparecen en el anexo 1. Se realiza un análisis con las gráficas que estas tablas generan para las figuras 4, 5 y 6.

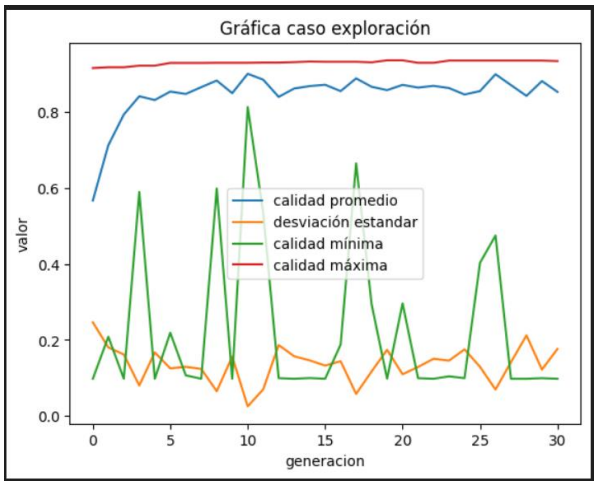

Figura 4. Gráfica caso exploración

En la figura 4, se aprecia la calidad promedio, mínima, y máxima, así como la desviación estándar del algoritmo genético por generación con los hiper parámetros del caso de exploración.

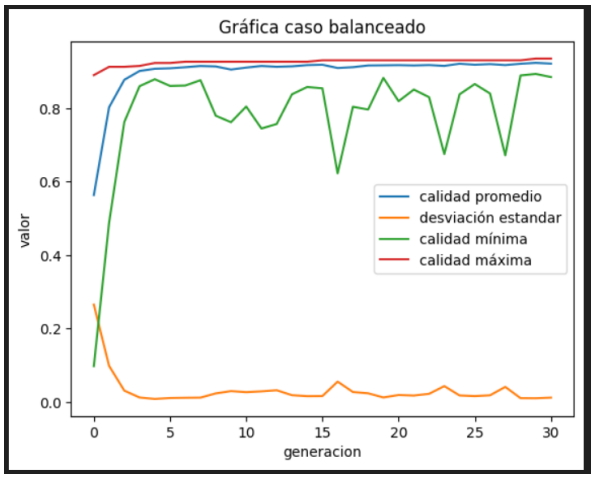

Figura 5. Gráfica caso balanceado

En la figura 5, se aprecia la calidad promedio, mínima, y máxima, así como la desviación estándar del algoritmo genético por generación con los hiper parámetros del caso balanceado.

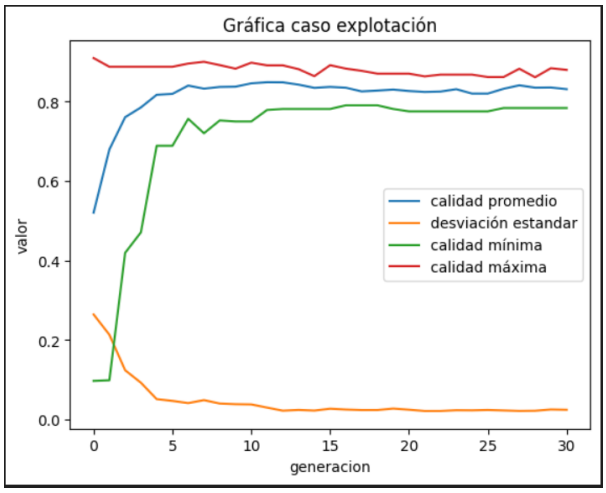

Figura 6. Gráfica caso explotación

En la figura 6, se observa la calidad promedio, mínima, y máxima, así como la desviación estándar del algoritmo genético por generación con los hiper parámetros del caso de explotación.

Al realizar un análisis de las distintas combinaciones de hiper parámetros se llegaron a las siguientes conclusiones:

- La grafica de exploración contiene muchas de las características que se esperarían de este tipo de algoritmo genético, específicamente el alto nivel de desviación estándar, y el alto nivel de variabilidad del valor mínimo de calidad en comparación con las demás gráficas. Esto se debe principalmente al alto nivel de mutación que permite buscar y encontrar más soluciones al problema, pero genera muchos individuos con muy baja calidad.
- En el caso del sistema balanceado se aprecian menores cambios en el valor de calidad mínima, aunque si se aprecian algunos puntos donde mutaciones y selección crean individuos de menor calidad.
- Por último, se tiene la gráfica de explotación. En el diagrama se aprecian algunos valores de calidad máxima que desaparecen en futuras generaciones, esto se debe a el método de ruleta utilizado, ya que, aunque es poco probable, el mejor individuo puede ser eliminado. Lo que más resalta es la consistencia relativa que se mantiene en la calidad mínima, esto se debe al bajo nivel de mutación que no permite generar individuos drásticamente diferentes que tienden a ser de menor calidad.

Al analizar los “hall of fame” de cada combinación, podemos obtener los mejores individuos de cada algoritmo. Cabe destacar que estos hall off ame se encuentran en el anexo 2. Con estos hiperparámetros se crea una red neuronal y se entrena por 100 epochs, calculando al final la calidad f1, para determinar cuál de los 3 individuos usar para la validación final.

- Exploración: 3 capas ocultas, 14 neuronas por capa oculta, tasa de aprendizaje de 0.007927168559160137, optimizador AdamW, sin momento, con calidad: 0.935630160910378

- Balanceado: 2 capas ocultas, 14 neuronas por capa oculta, tasa de aprendizaje de 0.00540727768076974, optimizador AdamW, sin momento con calidad: 0.962463234410229
- Explotación: 2 capas ocultas, 9 neuronas por capa oculta, tasa de aprendizaje de 0.00540727768076974, optimizador Adam, sin momento con calidad: 0.9534649449773085

De esto se elige el mejor individuo, correspondiente al del sistema balanceado. Algunos otros resultados que resaltan de la lista de individuos en el "hall of fame", es la cantidad de capas mayor a 2, un numero alto de neuronas por capa oculta, optimizador adam o adamW en su mayoría, y una tasa de aprendizaje en el orden de los milis. También resalta que no se utilizó momento en la mayoría de las soluciones.

5. Análisis de Resultados y Validación de la Solución

Como se mencionó previamente se toma el mejor individuo obtenido por la sección anterior, que corresponde al mejor individuo para la configuración balanceada en el algoritmo genético. Las características que posee son las siguientes: 2 Capas ocultas, 14 neuronas en cada capa, tasa de aprendizaje de 0.0054 aproximadamente, tipo de optimizador de AdamW, y sin Momento.

Para realizar una validación final y otorgar una calificación a la solución se procede a realizar el mismo proceso de creación de ANN, entrenamiento y evaluación, pero esta vez se ejecuta la evaluación al final del entrenamiento con la partición del conjunto de datos que no se ha utilizado anteriormente. Esto posibilita la calificación no sesgada del modelo, ya que son datos que no ha visto. A continuación, se pueden apreciar en las figuras 7 y 8 las gráficas de error y de accuracy para los datos de entrenamiento y los de prueba (no son los de validación todavía ya que estos solo se utilizan una vez al final).

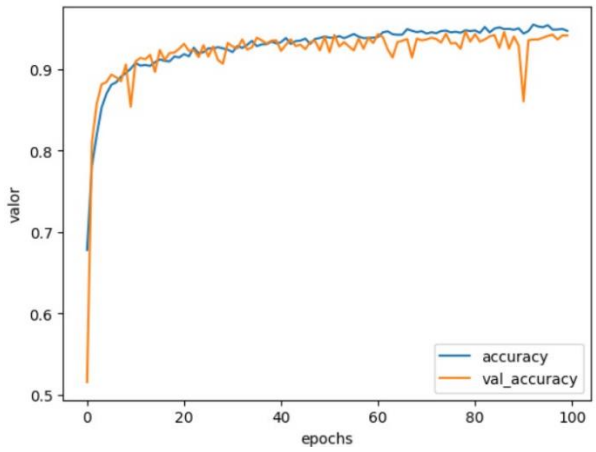

Figura 7. Grafica de Accuracy vs Epochs para Modelo Final

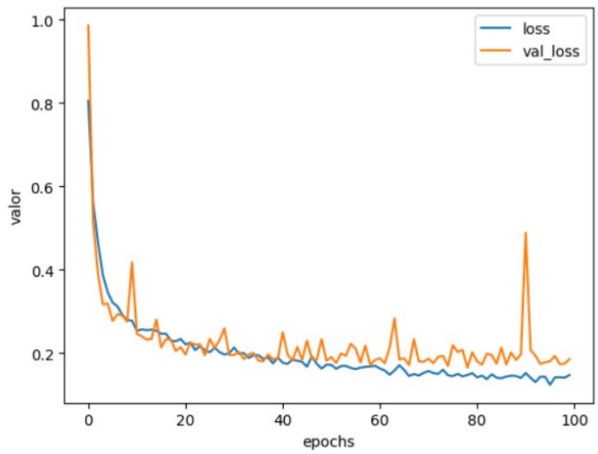

Figura 8. Grafica de Error vs Epochs para Modelo Final

Como se puede observar, el comportamiento de las gráficas es el esperado. El hecho de que el error tienda a estar siempre en bajada y el accuracy tienda a estar siempre en subida quiere decir que el modelo estaba entrenando correctamente. El hecho de que las curvas aproximan una asíntota horizontal quiere decir que están alcanzando su error basal, y que no están subajustadas (underfitting). También el hecho de que las curvas siguen una tendencia parecida, es decir que el error de prueba no sube bruscamente, o el accuracy de prueba baje bruscamente, quiere decir que no hay sobre entrenamiento en el modelo. En la figura 9 se muestra una representación gráfica del modelo neuronal final ya entrenado.

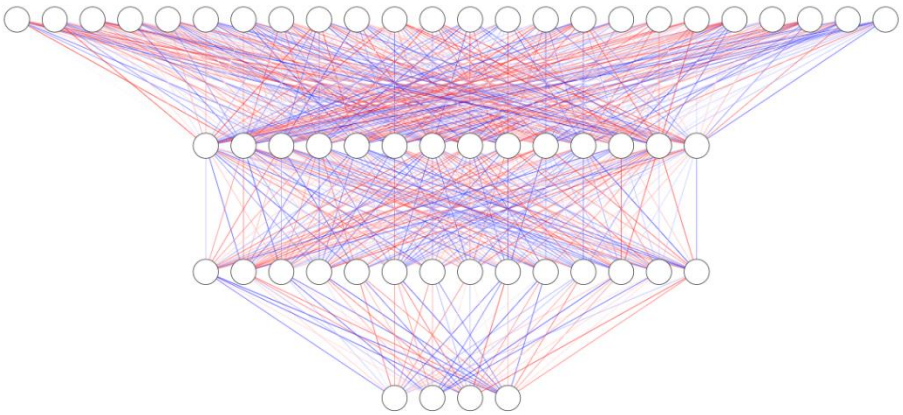

Figura 9. Visualización de Red Neuronal Final Entrenada

Como se mencionó, se realizó la validación final del modelo con el conjunto de datos para la validación solo una vez después del entrenamiento. El resultado de la matriz de confusión obtenida se muestra a continuación en la figura 10. Además, en la tabla 2 se pueden ver otras métricas adicionales en forma de porcentajes.

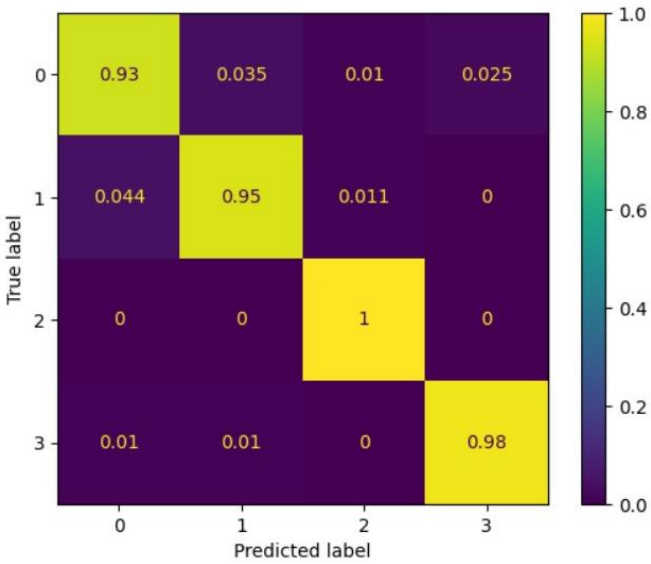

Figura 10. Matriz de Confusión del Modelo para datos de Validación

Tabla 2. Métricas de evaluación para Modelo Final

Accuracy	Precisión	Recall	F1 Score
96.47 %	96.39 %	96.35 %	96.36 %

Los resultados obtenidos son bastante buenos, tanto en la matriz de confusión, como en la tabla de métricas. El hecho de ver una diagonal marcada, con valores muy cercanos a 1 quiere decir que el modelo estaba prediciendo correctamente cada tipo de categoría. Se puede observar que la categoría 2 se predijo perfectamente en su totalidad. La categoría que se predijo menos correctamente fue la categoría 0.

Los datos obtenidos en la tabla 2 son de igual manera muy positivos. El accuracy indica qué tan bien el modelo clasifica correctamente todas las clases. Sin embargo, puede ser engañosa en conjuntos de datos desequilibrados. La precisión se centra en la precisión de las predicciones positivas. Es útil cuando los falsos positivos son costosos o cuando la clase positiva es crítica. El Recall se centra en la capacidad del modelo para identificar todas las instancias positivas. Es útil cuando los falsos negativos son costosos o críticos. Todos estos valores fueron muy altos, de arriba del 96 %. Además, el F1 score, que es un promedio armónico entre Precisión y Recall, dio un valor de 96.36 %.

Se decidió realizar un análisis de sensibilidad para las entradas a la red neuronal. Para esto se vario la entrada de cada valor desde -1 hasta 1, dejando todos los demás parámetros iguales, es decir, en ceteris paribus. Esto se pudo realizar debido a que los datos al ser normalizados mediante la técnica Z, tienen un promedio de 0, de manera que no es algo incorrecto asumir todas las entradas como 0. Las tablas de los Dataframes de entrada, y los resultados, se pueden encontrar en el anexo3. Se procedió a hacer 4 graficas para cada salida del modelo del Dataframe de resultados, las gráficas se muestran a continuación en las figuras 11, 12, 13, y 14.

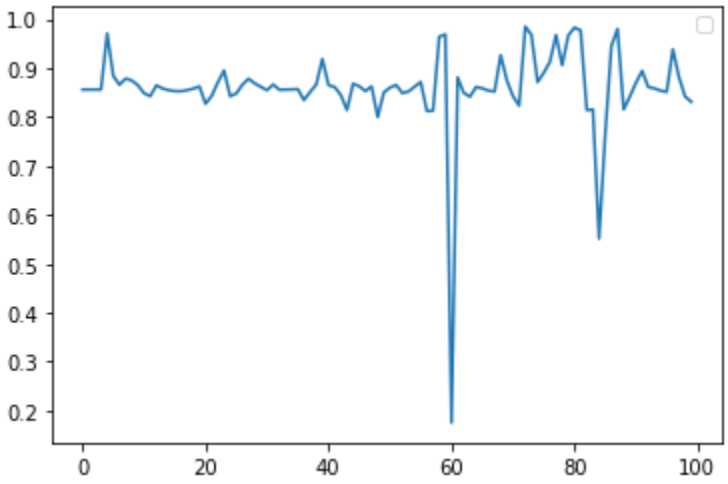

Figura 11. Resultados de Sensibilidad para salida b1

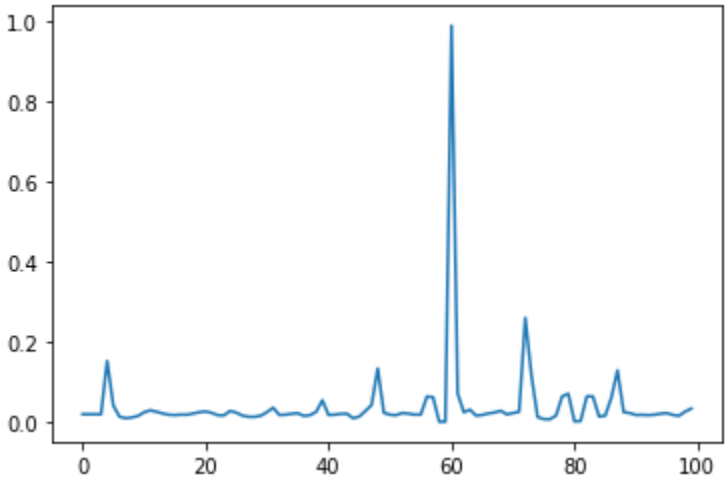

Figura 12. Resultados de Sensibilidad para salida b2

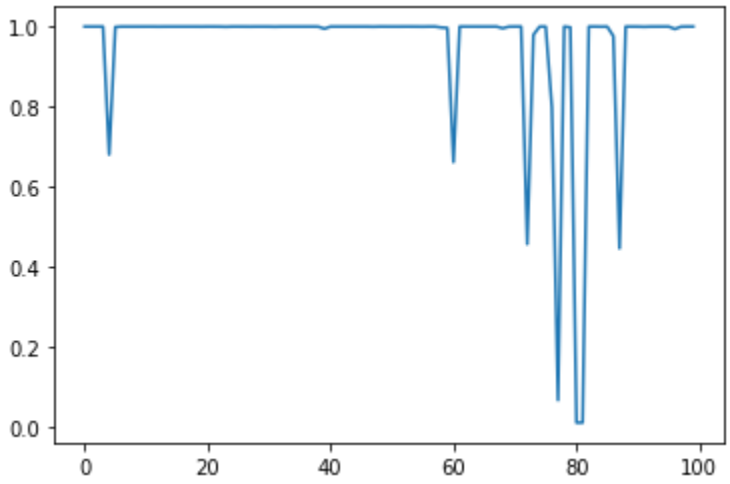

Figura 13. Resultados de Sensibilidad para salida b3

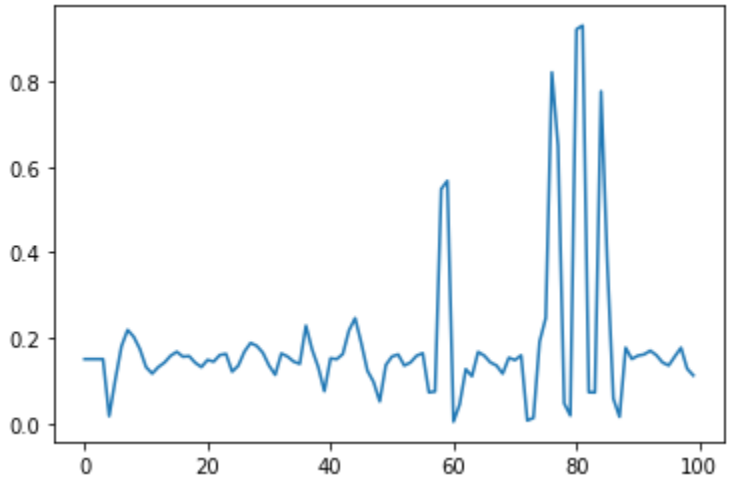

Figura 14. Resultados de Sensibilidad para salida b4

La escala de las figuras básicamente denota en el 0 la primera entrada, y en el 100 la entrada 24, de manera que se puede ver como su variación afecta la salida. Los hallazgos más importantes que se tuvieron del análisis de sensibilidad fueron los siguientes:

- Las entradas del rango 3 a 11 no tienen mucho impacto sobre las salidas del modelo.
- Las entradas del rango 13 a 20 tienen un impacto considerable sobre las salidas del modelo.
- Las demás entradas tienen un impacto intermedio sobre las salidas del modelo.
- Las salidas b1 y b3 son mucho más probables a ser activadas según entradas cercanas a 0 para el modelo.

- Las salidas b2 y b4 son mucho menos probables de ser activadas según entradas cercanas a 0 para el modelo.

6. Conclusiones

- Se logro la correcta clasificación de la acción a tomar por un robot navegador con una calidad de 96.36 % para la medida de f1 score.
- Se obtuvo una configuración de hiperparámetros para la ANN que fue adecuada para la creación y entrenamiento de la red, permitiendo obtener los resultados esperados.
- Se determino la mejor configuración del algoritmo genético, en donde un enfoque balanceado entre explotación y exploración fue el más adecuado.
- Se concluyo que fue exitosa la implementación del método de SMOTE para agregar datos sintéticos al conjunto proporcionado, y así obtener una distribución uniforme de las clases.
- El uso de técnicas adicionales en la red neuronal como el batch normalization o el EMA resulto ser de utilidad para evitar problemas en el entrenamiento de la red.
- Se concluyó que, para el caso específico del problema, el optimizador AdamW es más adecuado que el Adam normal, validando la utilización del decaimiento de pesos.
- Para el análisis de Sensibilidad se concluyó que las entradas del rango 3 a 11 no tienen mucho impacto sobre las salidas del modelo. Las entradas del rango 13 a 20 tienen un impacto considerable sobre las salidas del modelo. Las demás entradas tienen un impacto intermedio sobre las salidas del modelo. Las salidas b1 y b3 son mucho más probables a ser activadas según entradas cercanas a 0 para el modelo. Las salidas b2 y b4 son mucho menos probables de ser activadas según entradas cercanas a 0 para el modelo.
- Se comprobó que la navegación de un robot es un problema de clasificación con un grado de complejidad relativamente alto, ya que las soluciones estaban orientadas a configuraciones con múltiples capas ocultas, y múltiples neuronas. Los clasificadores neuronales lineales, como la red MLP, son capaces de aprender la tarea y comandar al robot con éxito sin colisiones.

7. Recomendaciones y comentarios

- La grafica de exploración contiene muchas de las características que se esperarían de este tipo de algoritmo genético, específicamente el alto nivel de desviación estándar en comparación.
- Métodos de regularización prácticamente no fueron considerados, aparte de Batch Normalization y el tipo de optimizador, debido a los buenos resultados obtenidos por la ANN. Se recomienda probar métodos como el dropout, o la regularización L2 para observar si se tiene alguna mejora significativa.
- Con respecto al algoritmo evolutivo, debido a limitaciones en el poder computacional de eligieron valores máximos para la población inicial de 50, y para el numero de generaciones de 30. Si se quisiera ejecutar el programa y se dispusiera de mayor recurso computacional o mayor tiempo, se recomienda aumentar tanto la población como el número de generaciones para así obtener mejores resultados.
- Se recomienda el uso de PCA para simplificar el modelo, ya que muchas entradas son redundantes.

8. Bibliografía

[1] "Supervised Classification Problems–Taxonomy of Dimensions and Notation for Problems Identification," in IEEE Access, vol. 9, pp. 151386-151400, 2021, doi: 10.1109/ACCESS.2021.312562

[2] Ananda L. Freire, Guilherme A. Barreto, Marcus Veloso and Antonio T. Varela (2009), "Short-Term Memory Mechanisms in Neural Network Learning of Robot Navigation Tasks: A Case Study'. Proceedings of the 6th Latin American Robotics Symposium (LARS'2009), pp 1-6

[3] F. Rosenblatt, "Perceptron Simulation Experiments," in Proceedings of the IRE, vol. 48, no. 3, pp. 301-309, March 1960, doi: 10.1109/JRPROC.1960.287598. [En linea]. Disponible: https://arxiv.org/abs/1412.6980

[4] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," arXiv preprint arXiv:1603.04467, 2016. [En linea]. Disponible: https://arxiv.org/abs/1603.04467

[5] [1] F. Chollet, "Keras: Deep learning library for Theano and TensorFlow," GitHub. [Online]. Available: https://github.com/keras-team/keras.

[6] DEAP Project, “DEAP documentation — DEAP 1.4.1 documentation” [En línea] Disponible en https://deap.readthedocs.io/en/master/

[7] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, May 2015, pp. 1-13. [Online]. Available: https://arxiv.org/abs/1412.6980

[8] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in Proceedings of the 32nd International Conference on Machine Learning (ICML), Lille, France, July 2015, pp. 448-456. [Online]. Available: https://arxiv.org/abs/1502.03167

[9] Davis, L., “Handbook of Genetic Algorithms”, Van Nostrand Reinhold, 1991.

[10] P.E. Valencia, “Optimización mediante Algoritmos Geneticos”, Researchgate, Agosto, pp 83-92, 1997.

[11] [1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321-357, 2002 doi: https://doi.org/10.1613/jair.953

[12] L.D. Whitley. "Foundations of Genetic Algorithms", Vol. 2, Morgan Kaufmann, San Mateo, CA, 1993.

9. Anexos

Anexo 1. Logbooks obtenidos en la ejecución del Algoritmo Genético

Tabla 1-1. Logbook configuración explorativa

gen	nevals	avg	std	min	max
0	50	0.56657062	0.24608679	0.09756628	0.91528322
1	34	0.71213513	0.18007565	0.20809119	0.91740559
2	25	0.79301148	0.16144954	0.09756628	0.91740559
3	26	0.84098696	0.07977369	0.58931699	0.92156596
4	30	0.83116003	0.1670642	0.09756628	0.92156596
5	32	0.85335559	0.12482068	0.21864173	0.92873589
6	39	0.84709021	0.12912947	0.10660697	0.92873589
7	31	0.86503981	0.12336442	0.09756628	0.92873589
8	27	0.88235467	0.06477898	0.59829533	0.92910995
9	31	0.84915214	0.15591671	0.09756628	0.92910995
10	32	0.90020333	0.02495913	0.8127579	0.92910995
11	33	0.88469518	0.06965918	0.53489612	0.92973641
12	37	0.83941564	0.18632452	0.0991342	0.92973641
13	26	0.86147505	0.15674507	0.09756628	0.93088373
14	34	0.86776578	0.14593541	0.09948097	0.93249078
15	32	0.87129394	0.13232413	0.09756628	0.93184774
16	28	0.85453878	0.14378073	0.18758027	0.93184774
17	33	0.88804732	0.05768701	0.66435044	0.93184774
18	37	0.86599489	0.11683089	0.29382995	0.93036965
19	34	0.85716729	0.17376467	0.09756628	0.93563016
20	39	0.8709966	0.10937267	0.29601123	0.93563016
21	37	0.86385566	0.12863543	0.0991342	0.9291417
22	33	0.86843153	0.15022265	0.09756628	0.9291417

23	38	0.86259345	0.14532234	0.10394342	0.93498346
24	33	0.84563972	0.17495994	0.0991342	0.93498346
25	33	0.85462177	0.12894134	0.40272899	0.93498346
26	30	0.89902146	0.06926481	0.47451275	0.93498346
27	36	0.87037245	0.14130573	0.09756628	0.93498346
28	33	0.84203934	0.21179193	0.09756628	0.93498346
29	35	0.88101506	0.12168533	0.0991342	0.93487232
30	38	0.85256244	0.17604445	0.09756628	0.93364566

Tabla 1-2. Logbook configuración balanceada

gen	nevals	avg	std	min	max
0	40	0.56363964	0.26530527	0.09756628	0.89036655
1	22	0.80264133	0.09842795	0.4878437	0.91293709
2	16	0.87764255	0.03086259	0.76267454	0.91293709
3	22	0.90164099	0.01224096	0.86018707	0.91538674
4	20	0.90760699	0.00822437	0.87919967	0.92338703
5	20	0.90894362	0.01081383	0.86090282	0.92338703
6	16	0.91185209	0.01146405	0.86196329	0.92691856
7	17	0.91519298	0.01188097	0.87648065	0.92691856
8	16	0.91392539	0.02333762	0.77982693	0.92691856
9	28	0.9055262	0.02957131	0.7620607	0.92691856
10	13	0.91087398	0.02676179	0.8047837	0.92691856
11	18	0.91524936	0.02888271	0.74485744	0.92691856
12	14	0.91300736	0.03193778	0.75723233	0.92691856
13	23	0.9142631	0.01848074	0.83805016	0.92691856
14	13	0.91758397	0.01584702	0.85803463	0.92691856
15	14	0.91862893	0.01612388	0.85440027	0.93071569
16	14	0.90974375	0.05546234	0.62268437	0.93071569
17	20	0.91195157	0.02716144	0.80411269	0.93071569
18	16	0.91665218	0.02363286	0.79665069	0.93071569
19	23	0.91702993	0.01229861	0.882901	0.93071569
20	20	0.9175284	0.01895878	0.81931474	0.93071569
21	21	0.91667191	0.01751403	0.85098604	0.93071569
22	19	0.91772445	0.02218335	0.83029654	0.93071569
23	15	0.91549611	0.04324843	0.67515925	0.93071569
24	15	0.92134495	0.01774437	0.83802182	0.93071569
25	23	0.91855374	0.01602529	0.86610494	0.93071569
26	14	0.92028636	0.018138	0.84028184	0.93071569
27	15	0.91766565	0.04109901	0.67174603	0.93071569
28	22	0.92127062	0.01044682	0.88943809	0.93071569

29	16	0.92376288	0.01017947	0.89363774	0.93533729
30	18	0.92167116	0.01187444	0.88504596	0.93533729

Tabla 1-3. Logbook configuración explotativa

gen	nevals	avg	std	min	max
0	30	0.52087693	0.26474381	0.09756628	0.9096288
1	7	0.6797097	0.21371973	0.0991342	0.88772805
2	8	0.76088573	0.12440015	0.4196534	0.88772805
3	4	0.78546268	0.09299958	0.47127313	0.88772805
4	7	0.81723721	0.05173095	0.68892233	0.88772805
5	6	0.81960588	0.0472465	0.68892233	0.88772805
6	11	0.84038342	0.0417303	0.75689154	0.89578793
7	13	0.83266885	0.04918972	0.72051285	0.90028751
8	10	0.83701777	0.04060772	0.75273728	0.8918909
9	8	0.83768577	0.03898091	0.7501298	0.88294029
10	11	0.84605235	0.03840872	0.7501298	0.89802836
11	16	0.84882638	0.03043888	0.7790492	0.89127736
12	11	0.84860052	0.02262352	0.78154622	0.89127736
13	8	0.84276666	0.02452004	0.78154622	0.88173133
14	8	0.83473886	0.02290903	0.78154622	0.86406923
15	12	0.83711376	0.02760233	0.78154622	0.89169786
16	13	0.83505163	0.02551337	0.7907816	0.88312193
17	13	0.82573448	0.02422137	0.7907816	0.8775384
18	7	0.82782446	0.0242672	0.7907816	0.8704346
19	10	0.83037724	0.02794654	0.78158239	0.8704346
20	10	0.82672108	0.02511099	0.77557672	0.8704346
21	7	0.82441542	0.02179218	0.77557672	0.86353652
22	6	0.8253621	0.02184154	0.77557672	0.86798791
23	12	0.83133812	0.02374891	0.77557672	0.86798791
24	12	0.82030395	0.02347933	0.77557672	0.86798791
25	10	0.82028054	0.02440811	0.77557672	0.86187164
26	13	0.83232808	0.02318573	0.78397624	0.86187164
27	7	0.84109544	0.02196446	0.78397624	0.88279962
28	10	0.83519197	0.02238763	0.78397624	0.86137592
29	7	0.83560897	0.02566767	0.78397624	0.88411918
30	9	0.83141049	0.0249179	0.78397624	0.87988514

Anexo 2. Halls of Fame obtenidos en cada Configuración para AG

Tabla 2-1. Hall of Fame caso explorativo

[3, 14, 0.007927168559160137, 3, 0]

[3, 13, 0.008225458557130645, 3, 0]

[5, 14, 0.005484747829515145, 3, 1]

[3, 14, 0.0059909937705741444, 3, 0]

[3, 14, 0.00676063037140078, 3, 0]

[3, 13, 0.009087470938275236, 3, 1]

[2, 14, 0.005041149832960848, 3, 0]

[4, 14, 0.00676063037140078, 2, 0]

[3, 13, 0.008225458557130645, 3, 0]

[3, 14, 0.011369561755270558, 2, 0]

Tabla 2-2. Hall of fame caso balanceado

[2, 14, 0.00540727768076974, 3, 0]

[2, 12, 0.00540727768076974, 3, 0]

[2, 13, 0.00540727768076974, 3, 0]

[2, 12, 0.00540727768076974, 3, 0]

[2, 12, 0.005798735939586047, 3, 0]

[2, 10, 0.00540727768076974, 2, 0]

[2, 10, 0.008386650988826853, 2, 0]

[2, 10, 0.0058415397992345516, 2, 0]

[2, 10, 0.00540727768076974, 2, 0]

[2, 10, 0.004694775208289683, 3, 1]

Tabla 2-3. Hall of fame caso explotación

[2, 9, 0.00540727768076974, 2, 0]

[3, 8, 0.0017451014787518, 3, 1]

[4, 5, 0.004030286638914626, 3, 1]

[3, 8, 0.0020654875478694633, 3, 1]

[2, 9, 0.002112788749705132, 2, 1]

[4, 6, 0.003463483174147512, 3, 1]

[5, 7, 0.0033813968097127236, 3, 1]

[3, 6, 0.0037730019753176645, 2, 1]

[6, 7, 0.0034474292049374434, 3, 1]

[5, 5, 0.004462953246075919, 3, 1]

Anexo 3. análisis de sensibilidad

Tabla 3-1. Dataframe ceteris paribus para diferentes entradas

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.5
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Tabla 3-2. Resultados de la predicción del dataframe ceteris paribus

B1	B2	B3	B4
0.85648	0.018828	0.999778	0.150139
0.85648	0.018828	0.999778	0.150139
0.85648	0.018828	0.999778	0.150139
0.85648	0.018828	0.999778	0.150139
0.971214	0.151596	0.680147	0.016643
0.884136	0.040407	0.998838	0.102154
0.866308	0.01293	0.999774	0.180141
0.878857	0.008755	0.999732	0.218491
0.874989	0.010775	0.999722	0.202272
0.865239	0.014476	0.999755	0.173624
0.848966	0.02363	0.999795	0.131305
0.842917	0.0286	0.999806	0.116272
0.864911	0.025048	0.999633	0.131576
0.858775	0.02091	0.999745	0.14222
0.855139	0.017582	0.999788	0.158097
0.853293	0.016756	0.999787	0.16726
0.853043	0.017906	0.999783	0.155804
0.855217	0.017647	0.999786	0.157163
0.858352	0.020787	0.999754	0.141613
0.862847	0.023857	0.999689	0.131564
0.827329	0.025778	0.99981	0.148201
0.843512	0.022204	0.999804	0.144279
0.871189	0.016394	0.999695	0.159726
0.895392	0.016073	0.999292	0.162274
0.842801	0.027212	0.999805	0.120676
0.848397	0.022798	0.999798	0.134824

0.866508	0.015359	0.999741	0.167769
0.878526	0.012497	0.999666	0.187963
0.869035	0.012365	0.999771	0.181147
0.861985	0.015242	0.999782	0.164973
0.855117	0.024162	0.999727	0.13509
0.866837	0.035087	0.99942	0.11336
0.856132	0.016805	0.999783	0.163245
0.85635	0.017769	0.999782	0.156311
0.856724	0.020016	0.999771	0.144458
0.857329	0.021373	0.999758	0.138996
0.834885	0.014627	0.999768	0.228505
0.851382	0.016524	0.999788	0.17276
0.866581	0.024895	0.999614	0.131509
0.919067	0.053641	0.99436	0.074987
0.865282	0.017031	0.999754	0.152033
0.861689	0.017848	0.999769	0.149964
0.84535	0.019731	0.99978	0.162147
0.814468	0.020074	0.999762	0.217362
0.86844	0.008829	0.999734	0.246075
0.863189	0.013041	0.999769	0.187992
0.853245	0.027064	0.999738	0.122946
0.863113	0.041761	0.999474	0.096718
0.800172	0.133172	0.99985	0.051028
0.850766	0.022415	0.999796	0.136462
0.860469	0.017589	0.999761	0.156242
0.865937	0.016492	0.999733	0.161
0.849272	0.022042	0.99981	0.134984
0.852473	0.020306	0.999799	0.14225
0.862203	0.01787	0.999732	0.158035
0.871914	0.017936	0.999601	0.163887
0.812334	0.062881	0.999837	0.072332
0.813203	0.061344	0.999831	0.074247
0.964326	0.000321	0.997494	0.548629
0.969291	0.000265	0.996357	0.567565
0.175294	0.988164	0.661331	0.003603
0.881086	0.070762	0.999839	0.044237
0.849528	0.023991	0.999795	0.12664
0.841923	0.030057	0.999808	0.110067
0.861775	0.015306	0.999775	0.166942
0.859001	0.016989	0.999778	0.15831
0.854279	0.020839	0.999777	0.142405
0.852474	0.02304	0.999772	0.13506
0.926837	0.027532	0.995463	0.116044