TP3 - PERCEPTRÓN SIMPLE Y MULTICAPA

Grupo 3

ARTIFICIAL INTE (AI)

EJERCICIO 1

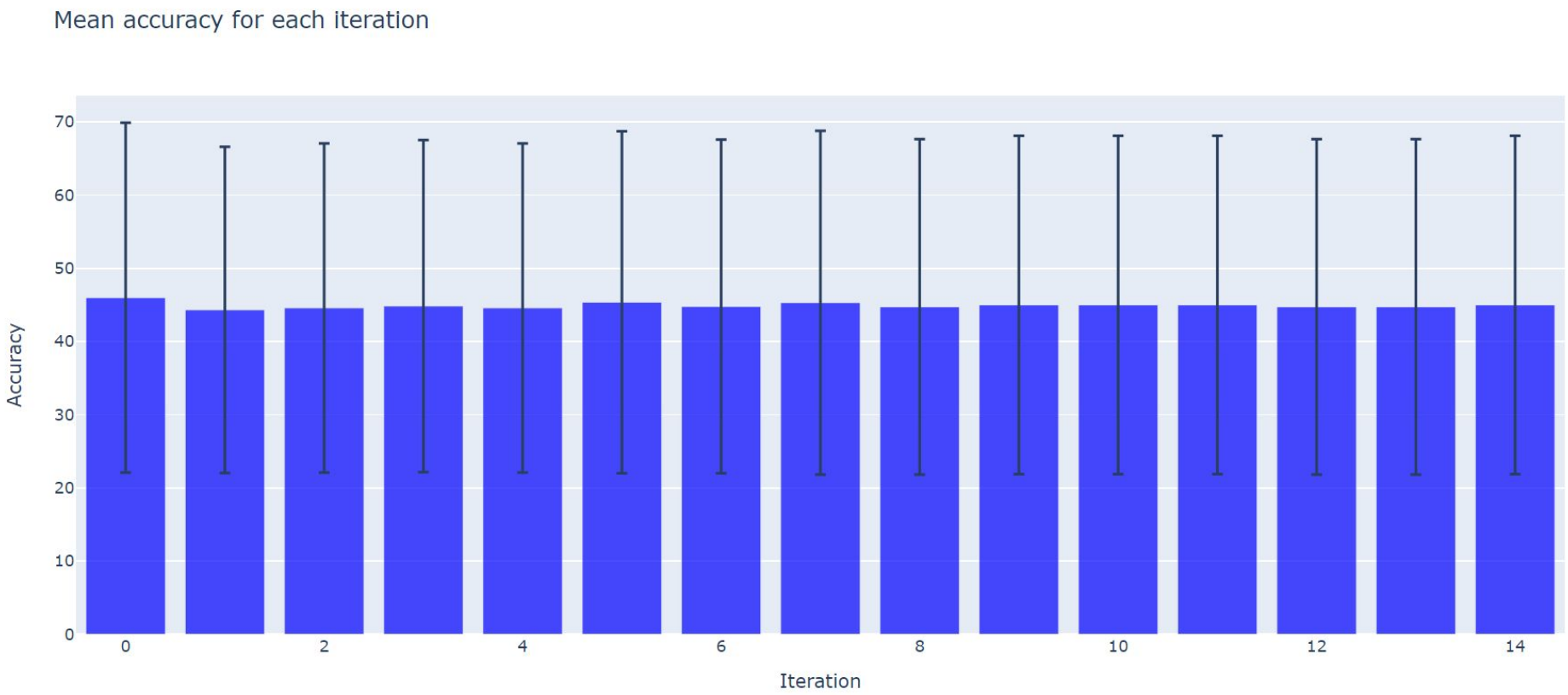
Perceptrón simple escalonado
AND y XOR

2 valores por input (coordenada)
1 de salida: que tan verdadero es

(AI)

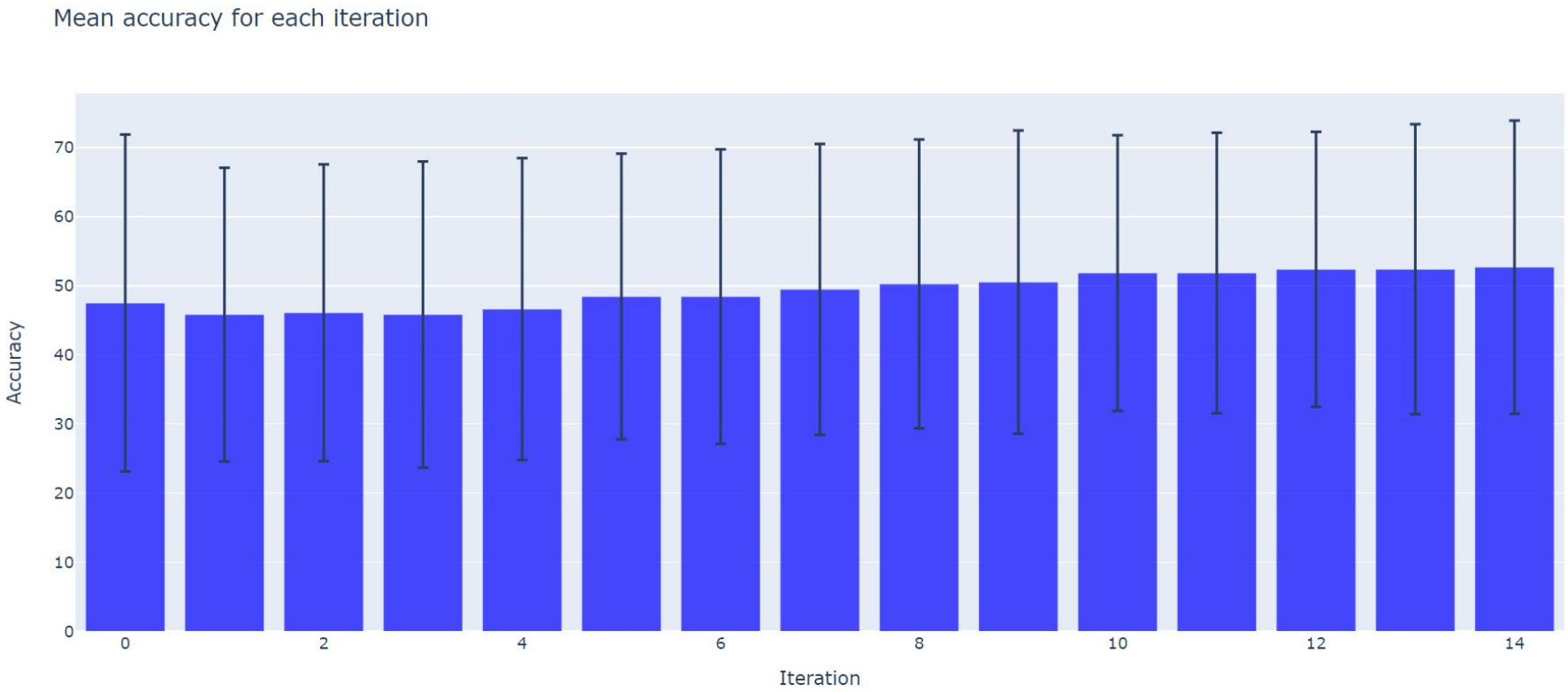
PROMEDIO DE 100 CORRIDAS: AND

LEARNING CONSTANT = 0.02, LIMIT = 15 ITERATIONS



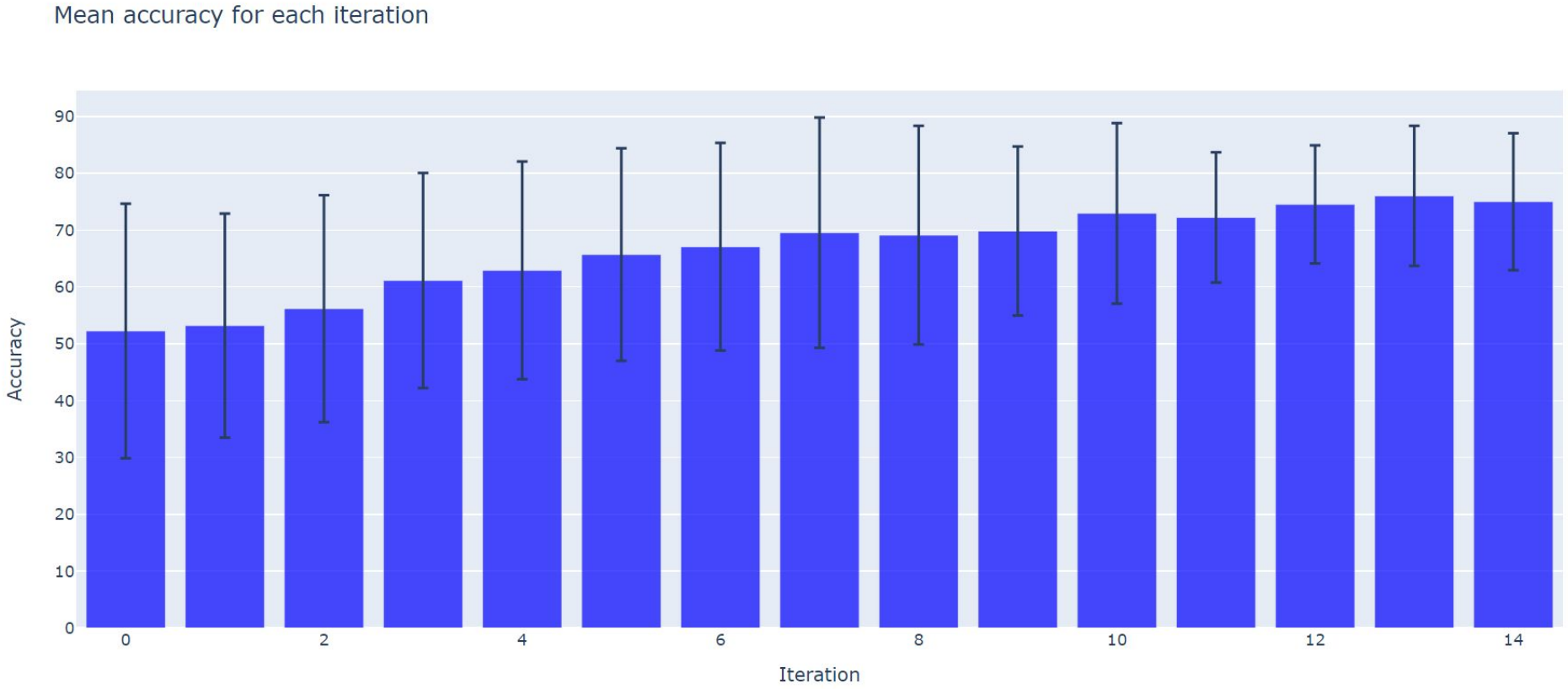
PROMEDIO DE 100 CORRIDAS: AND

LEARNING CONSTANT = 0.1, LIMIT = 15 ITERATIONS



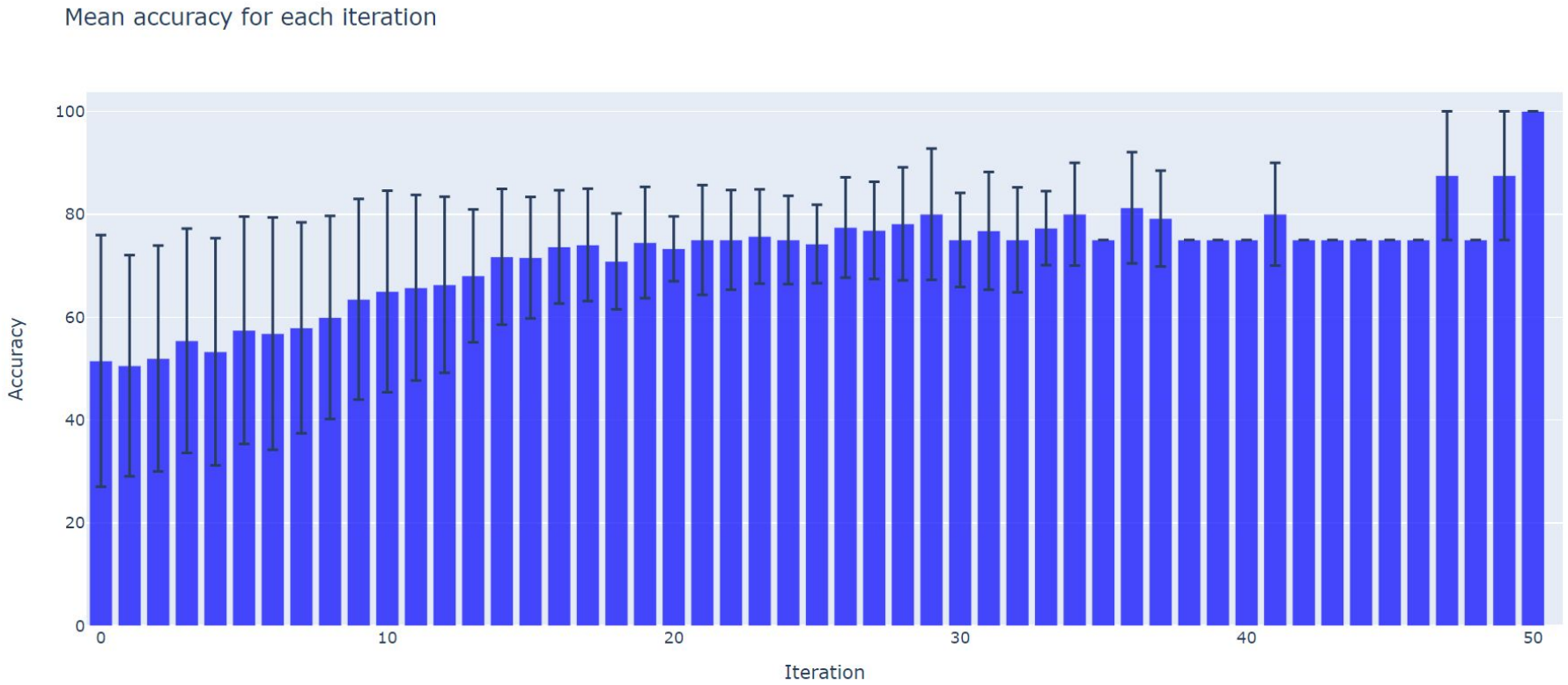
PROMEDIO DE 100 CORRIDAS: AND

LEARNING CONSTANT = 0.8, LIMIT = 15 ITERATIONS



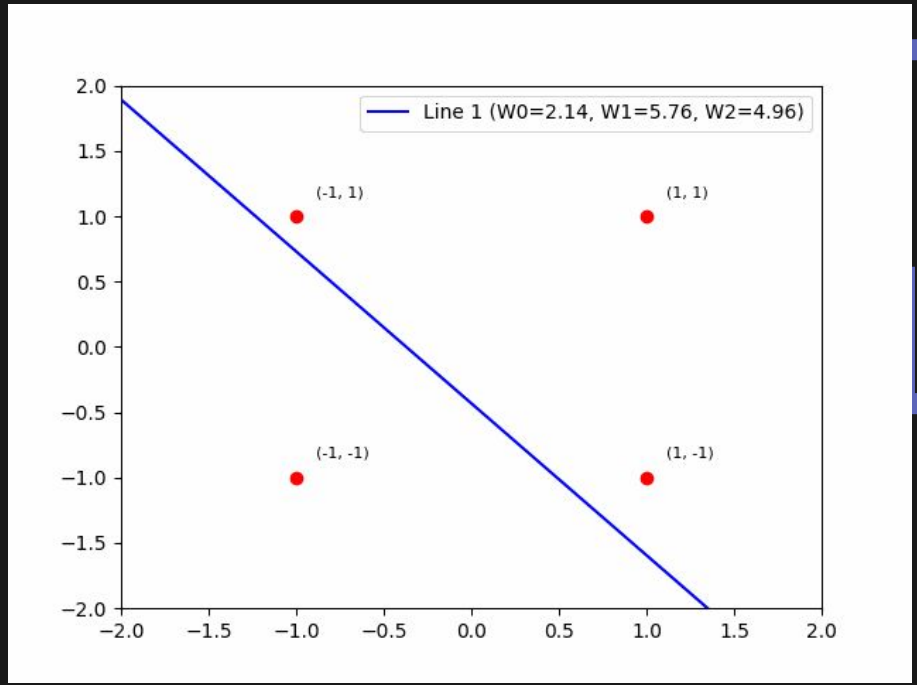
PROMEDIO DE 100 CORRIDAS: AND

LEARNING CONSTANT = 0.5, LIMIT = 100 ITERATIONS



AND DATA SET

LEARNING CONSTANT = 0.1



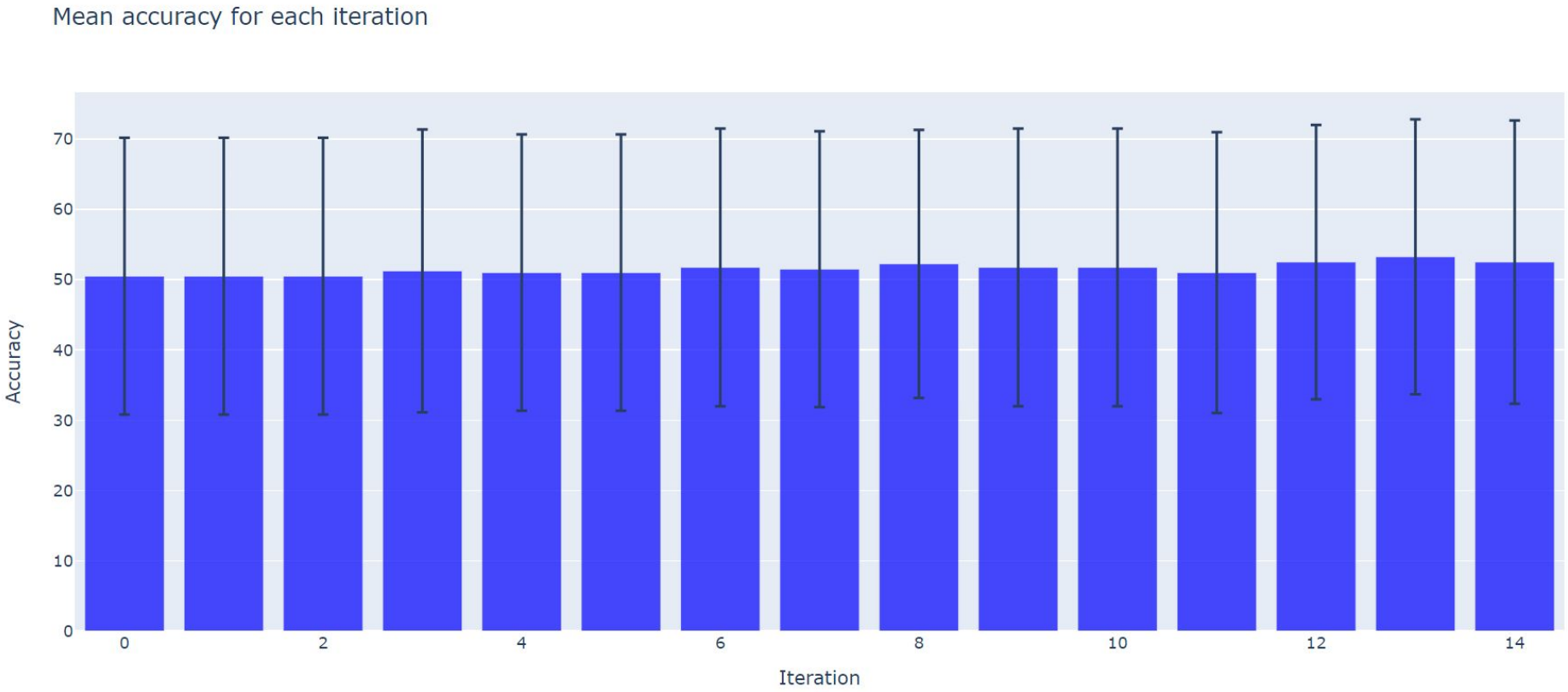
TIFICIAL

LIGENCE

(AI)

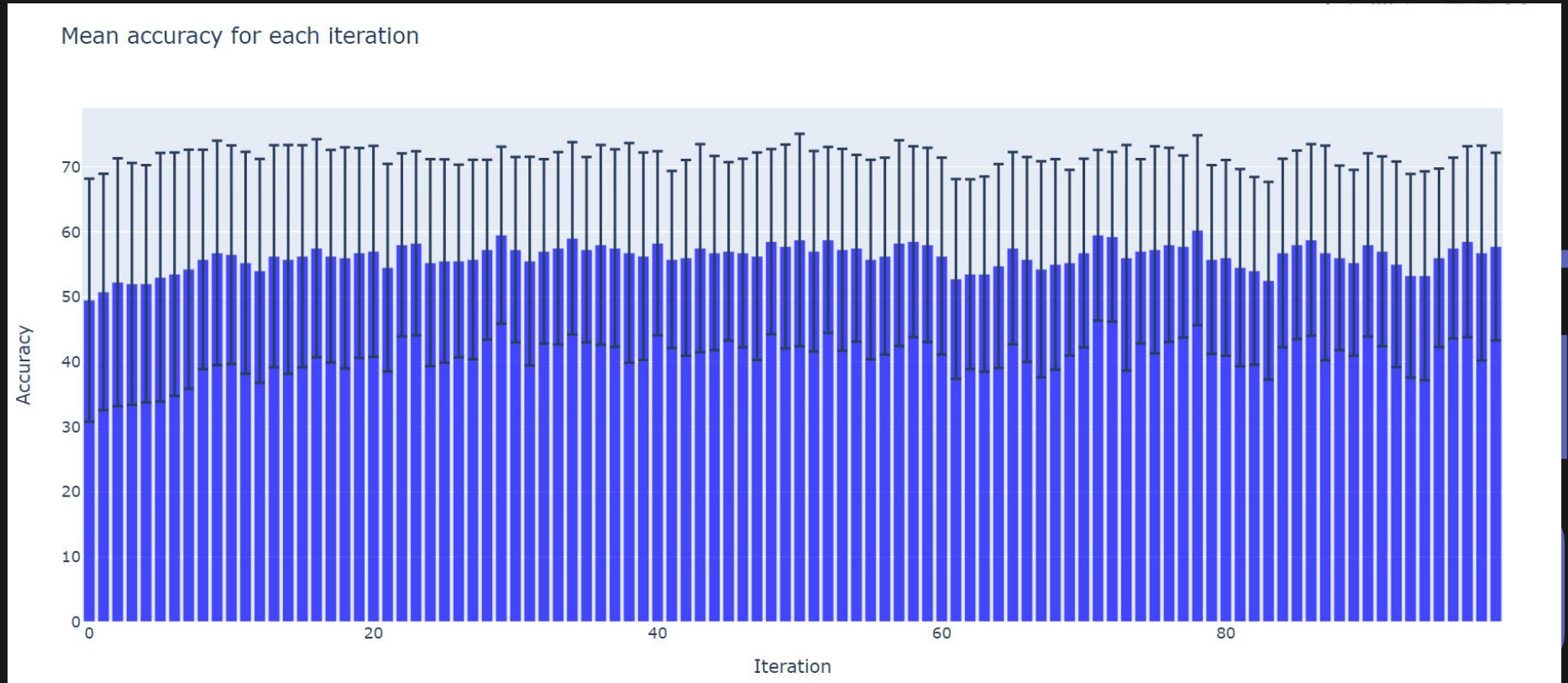
PROMEDIO DE 100 CORRIDAS: XOR

LEARNING CONSTANT = 0.1, LIMIT = 15 ITERATIONS



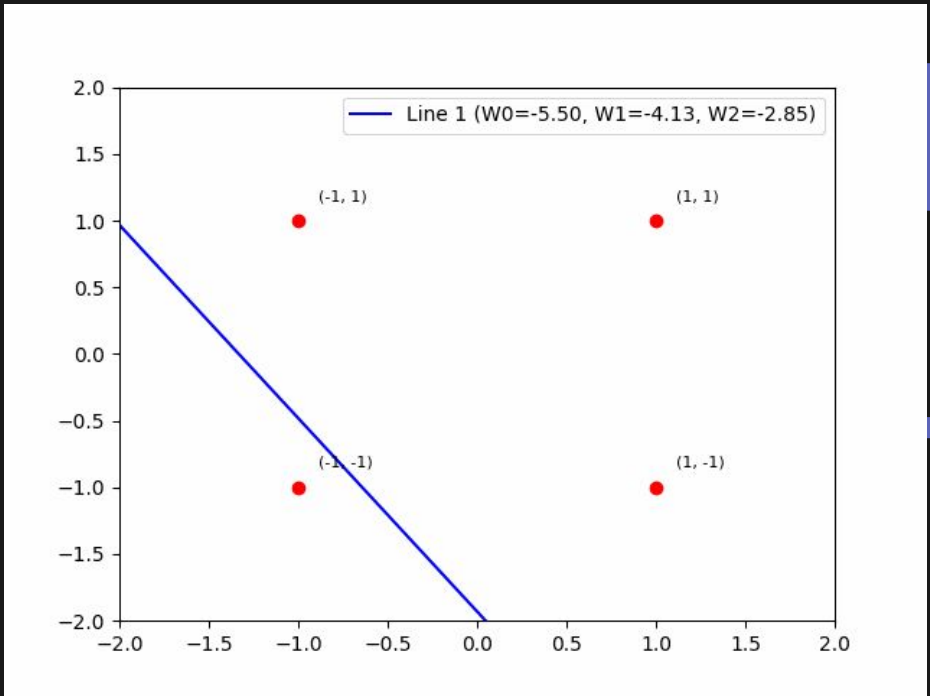
PROMEDIO DE 100 CORRIDAS: XOR

LEARNING CONSTANT = 0.5, LIMIT = 100 ITERATIONS



XOR DATA SET

LEARNING CONSTANT = 0.1 LIMIT = 40



TIFICIAL
LIGENCE
(AI)

CONCLUSIONES EJERCICIO 1

‘AND’ ES LINEALMENTE SEPARABLE

Existen infinitas rectas que separan en 2 al conjunto de datos

‘XOR’ NO ES LINEALMENTE SEPARABLE

Se necesitaría otro criterio de separación, tal vez usar 2 rectas o una línea curva

EL LEARNING CONSTANT AFECTA QUÉ TAN RÁPIDO CONVERGE

Un valor chico (0.02) hace que sea más lento

Un valor grande (0.5 ~ 0.8) hace que converga rápido

ARTIFICIAL INTE (AI)

EJERCICIO 2

Perceptrón Lineal y No LIneal
3 valores de entrada y 1 de salida

(AI)

CONFIGURACIÓN BASE

```
{
  "seed": 1,
  "training_data": "../training_data/ex2.csv",
  "learning_constant": 0.005,
  "limit": 300,
  "epsilon": 0,
  "method": {
    "type": "linear",
    "theta": "logistic",
    "beta": 0.3
  },
  "selection_method": {
    "type": "simple",
    "proportion": 0.8,
    "folds": 5
  }
}
```

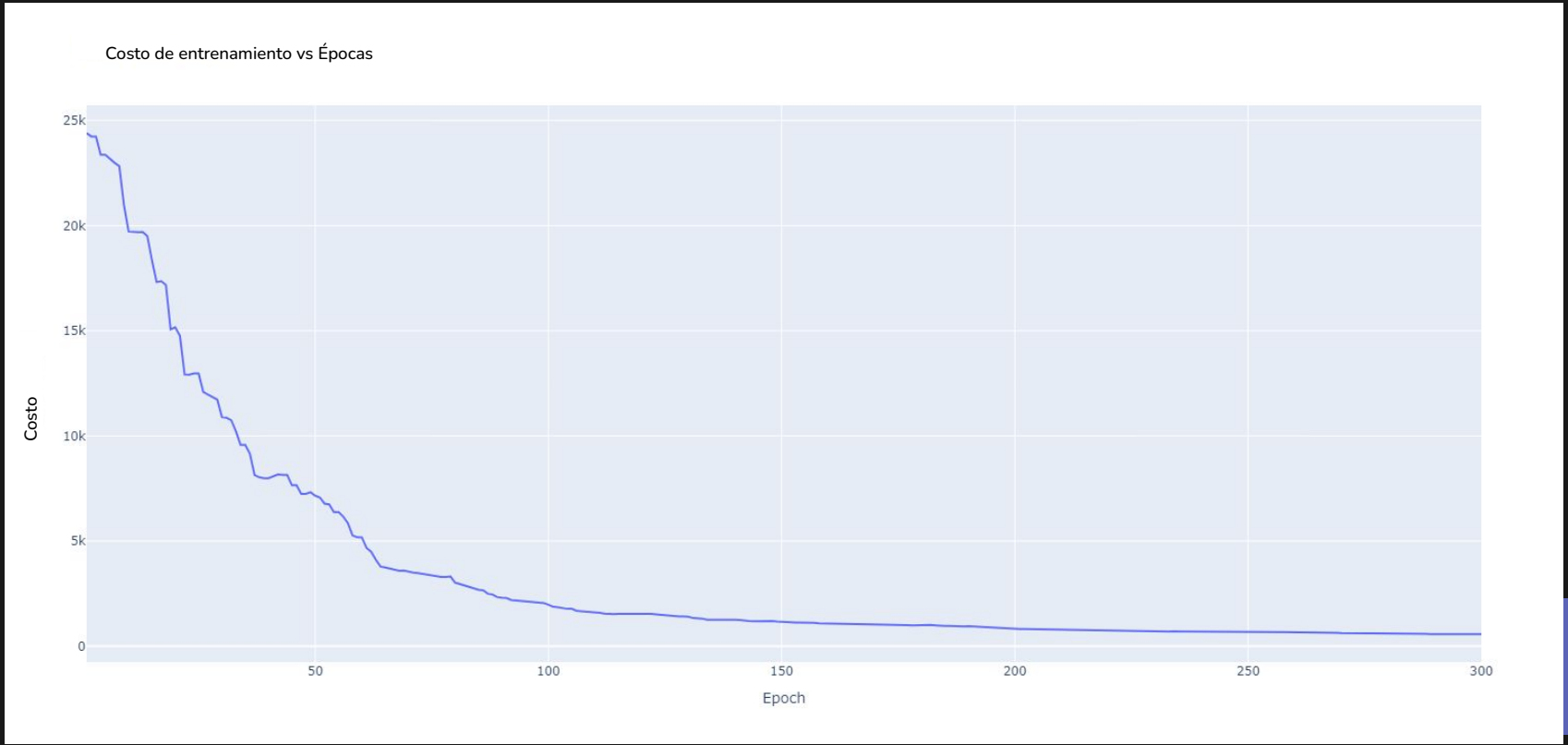
PERCEPTRÓN SIMPLE LINEAL

/(AI)/(AI)/

ARTI
CIAL
INTE
IGEN
(AI)

COSTO DE ENTRENAMIENTO VS ÉPOCAS: TP = 80%

Learning Constant: 0.005, Selection = Simple, Train percentage = 80%, Limit = 300



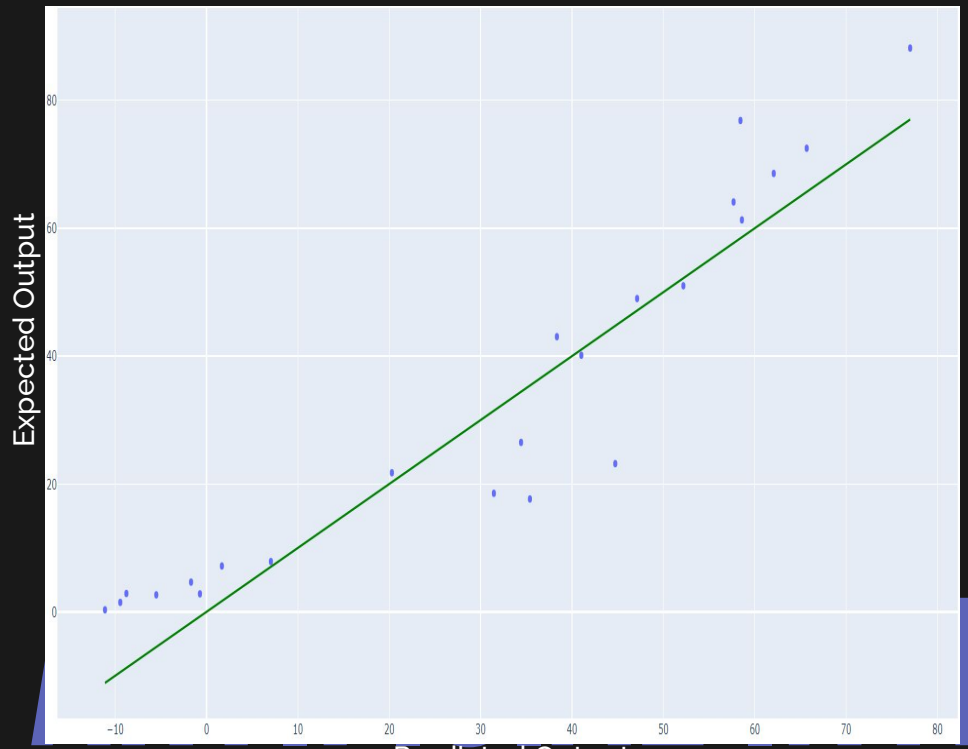
ZOOM COSTO DE ENTRENAMIENTO VS ÉPOCAS

Learning Constant: 0.005, Selection = Simple, Train percentage = 80%, Limit = 10000

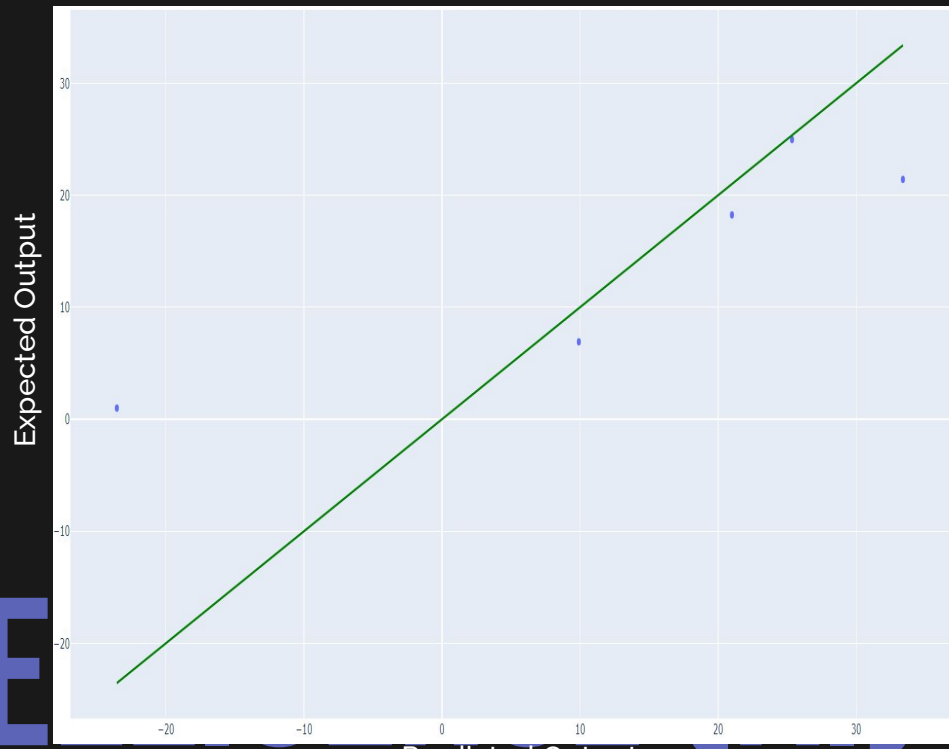
ESPERADO VS PREDICHO: TP = 80%

Learning Constant: 0.005, Selection = Simple, Train percentage = 80%, Limit = 300

Training Data

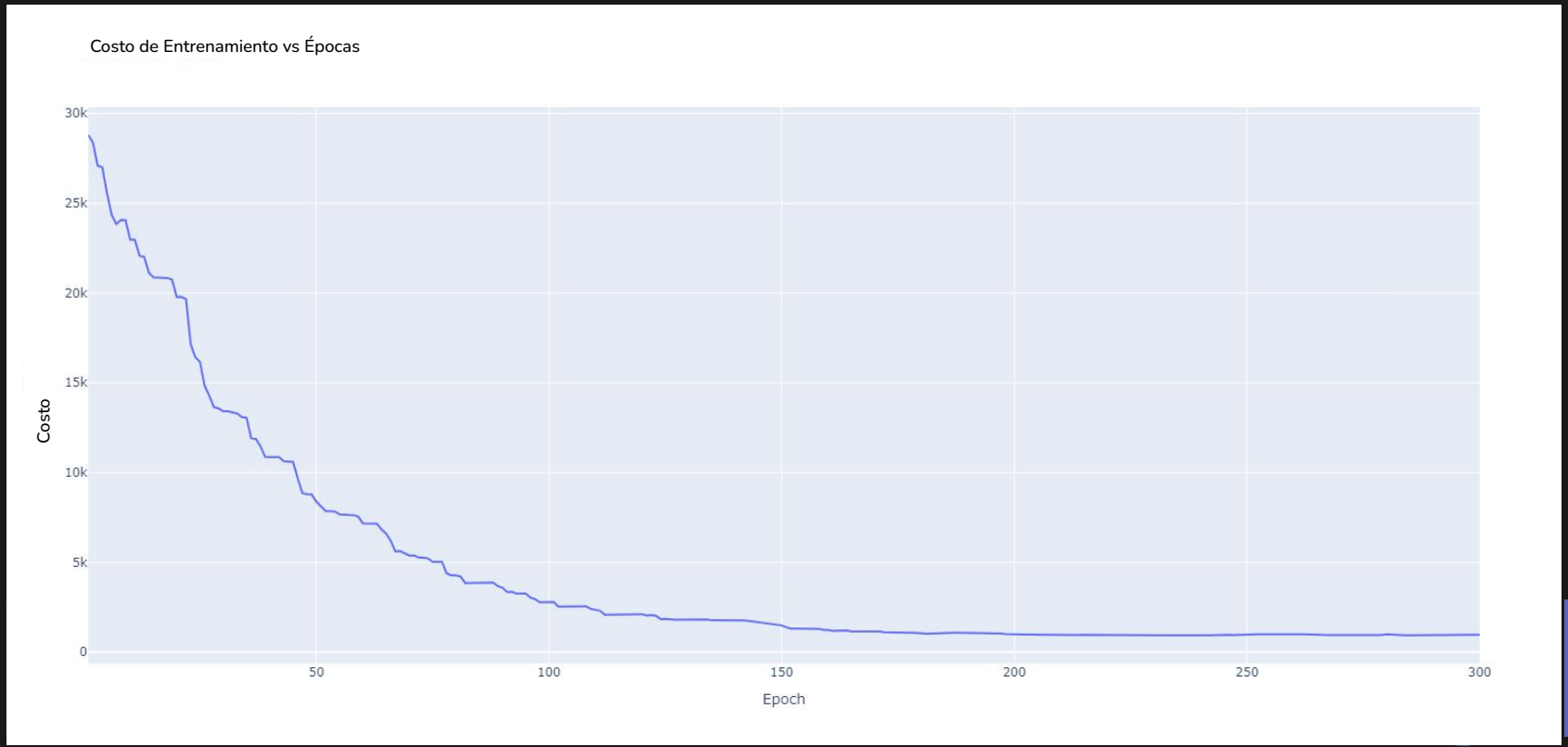


Test Data



COSTO DE ENTRENAMIENTO VS ÉPOCAS: TP = 40%

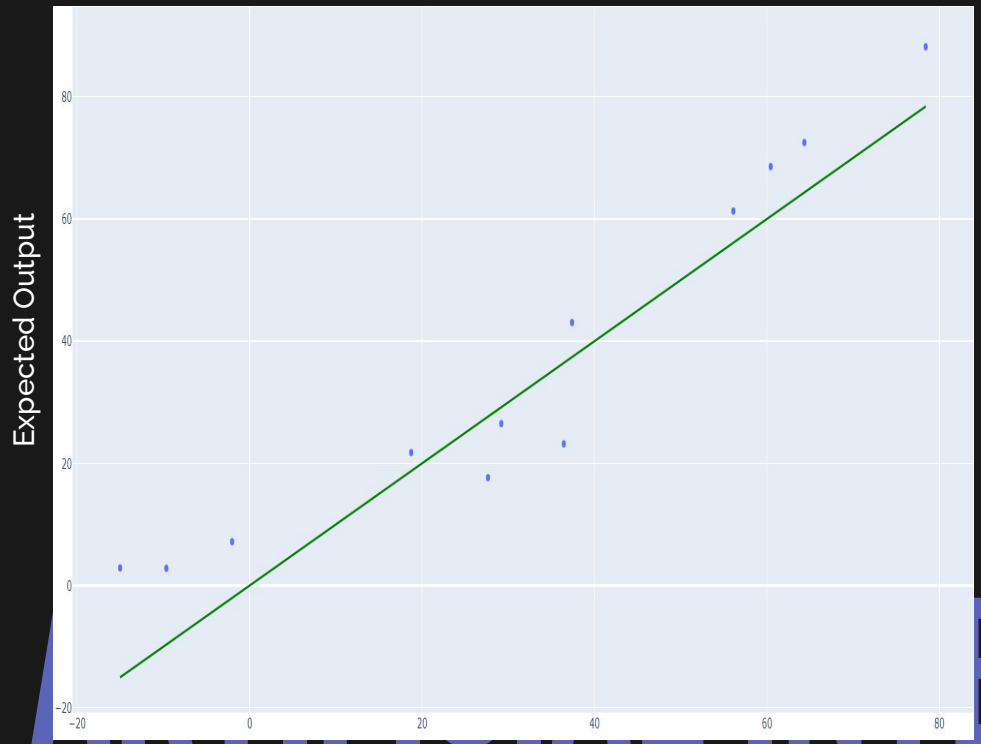
Learning Constant: 0.005, Selection = Simple, Train percentage = 40%, Limit = 300



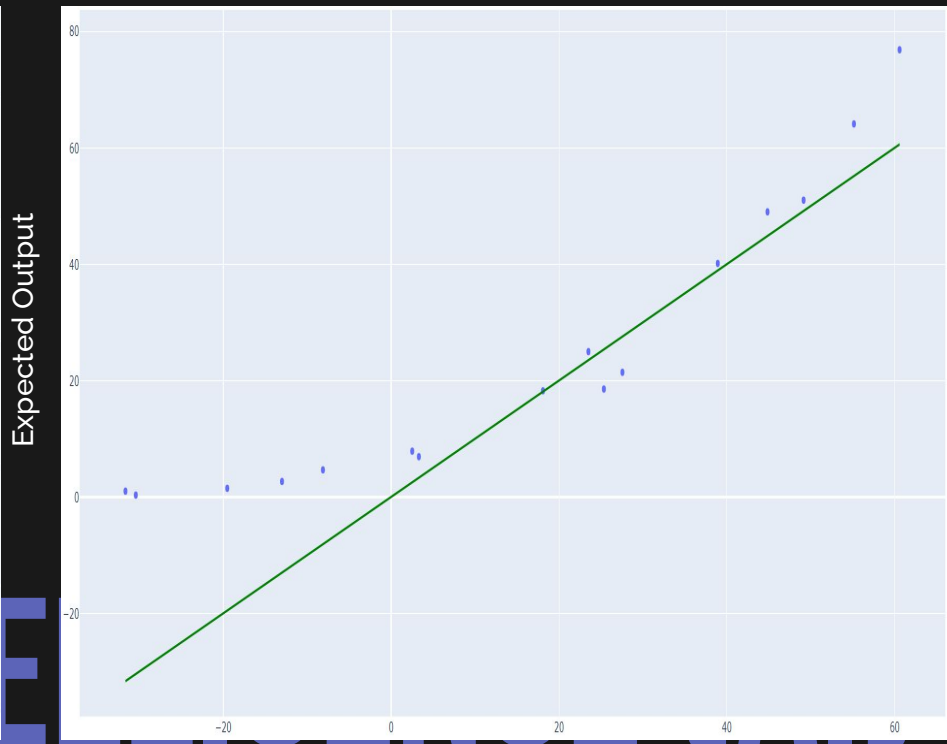
ESPERADO VS PREDICHO: TP = 40%

Learning Constant: 0.005, Selection = Simple, Train percentage = 40%, Limit = 300

Training Data

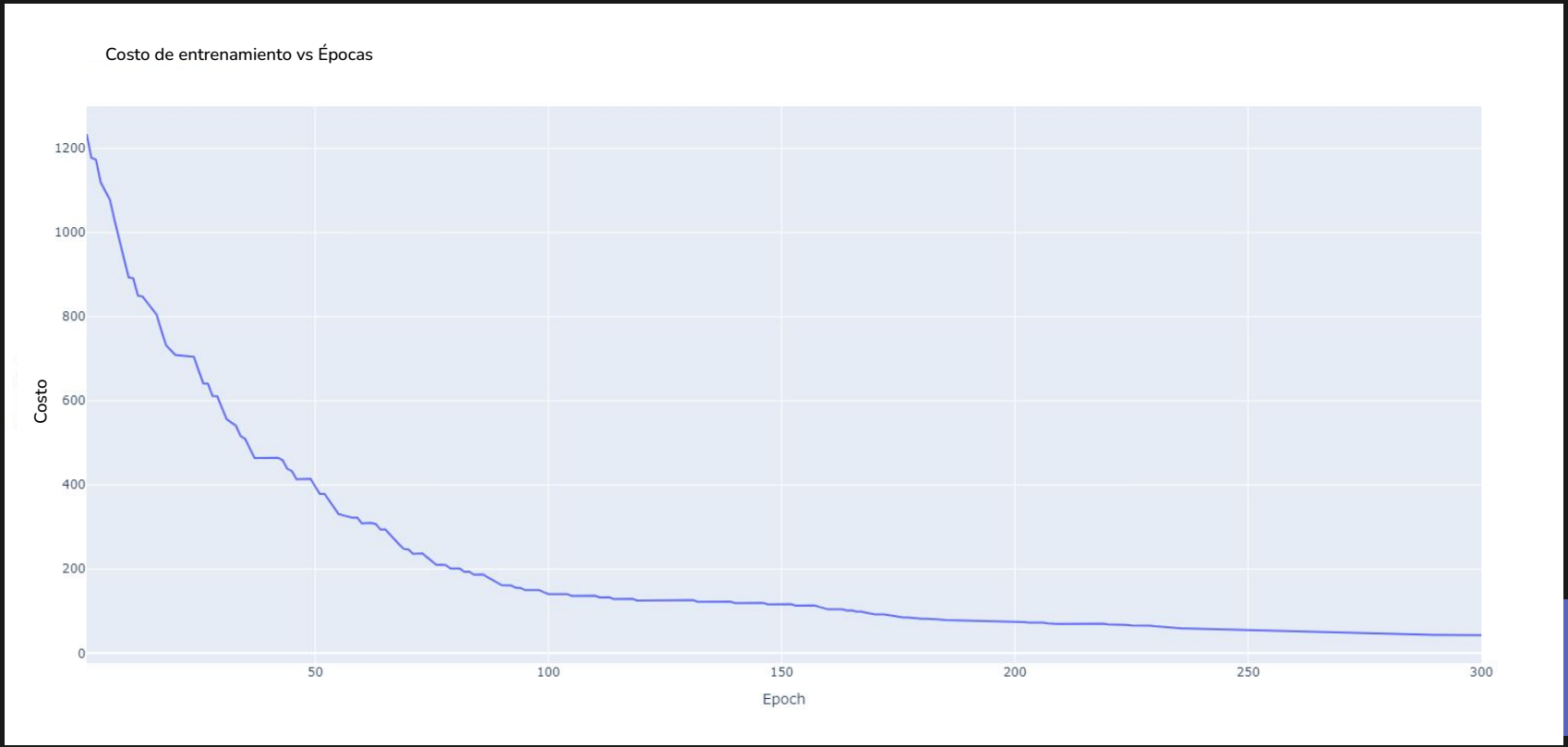


Test Data



COSTO DE ENTRENAMIENTO VS ÉPOCAS: TP = 10%

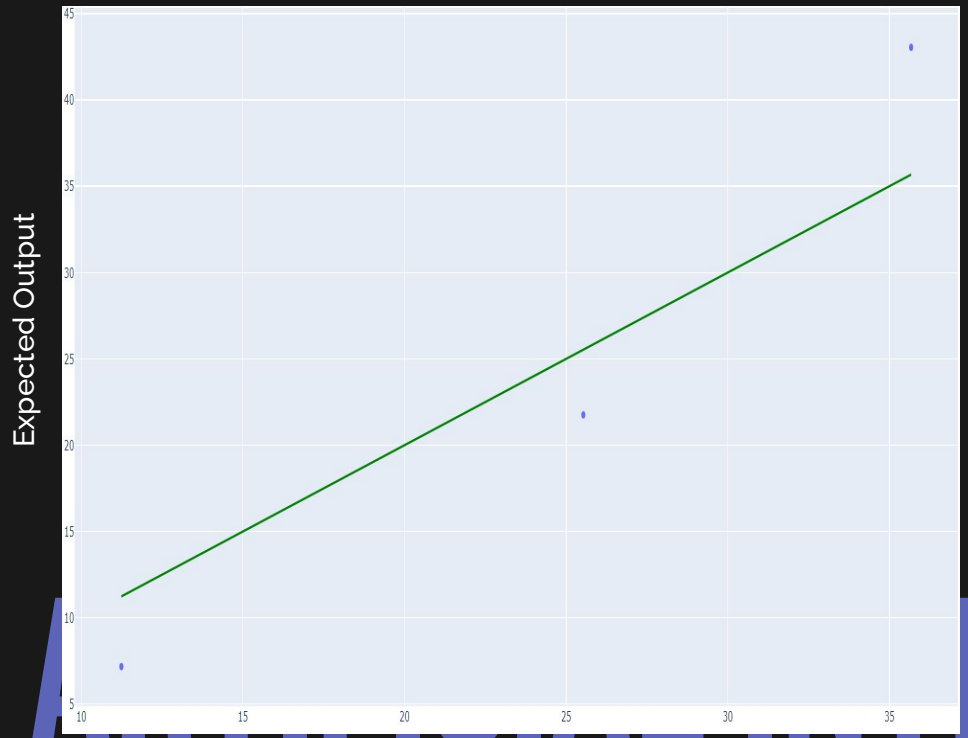
Learning Constant: 0.005, Selection = Simple, Train percentage = 10%, Limit = 300



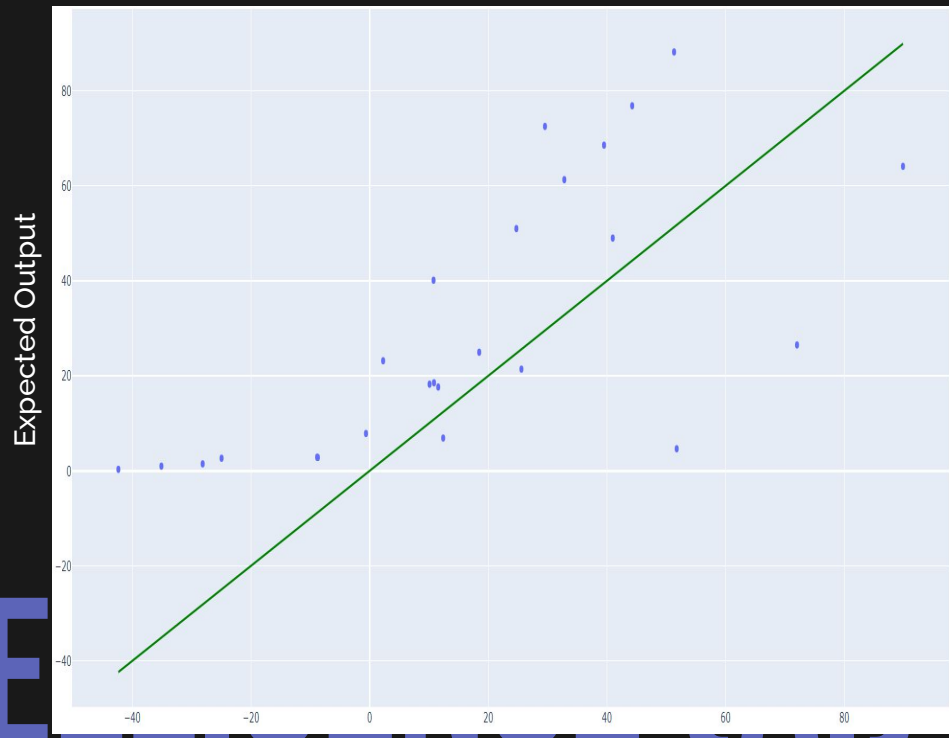
ESPERADO VS PREDICHO: TP = 10%

Learning Constant: 0.005, Selection = Simple, Train percentage = 10%, Limit = 300

Training Data



Test Data



COMPARACIÓN DE TRAIN PERCENTAGE

Con train % = 80% el MSE de testing es: **372.404**
MSE de training : 11795201.367

Con train % = 40% el MSE de testing es: **637.8**
MSE de training = 4161857.496

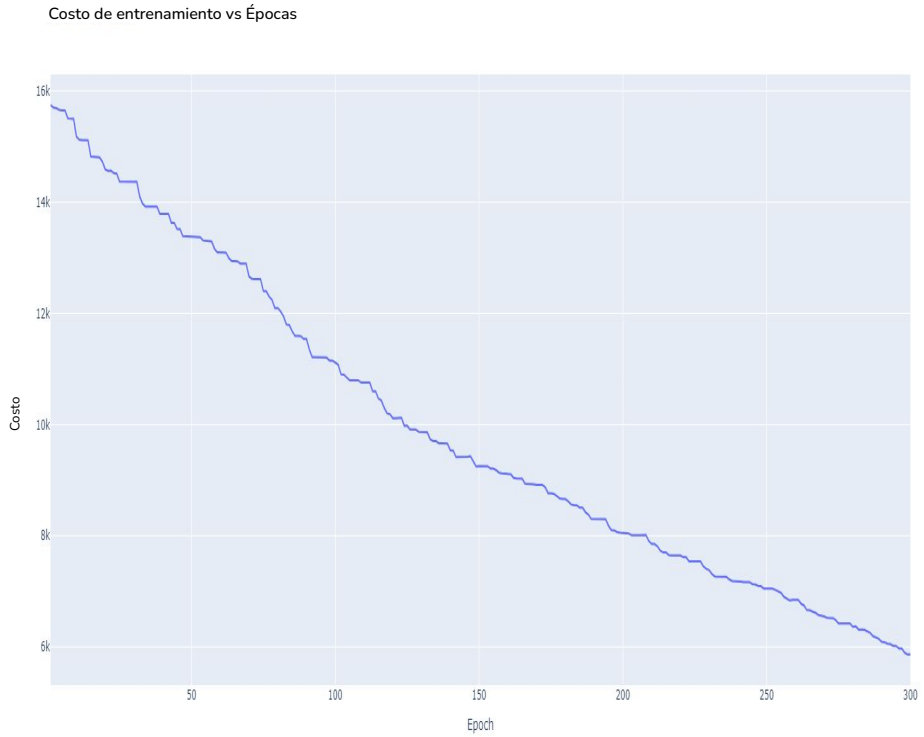
Con train % = 10% el MSE de testing es: **732.815**
MSE de training = 58282.093

Cuidado con underfitting y overfitting!

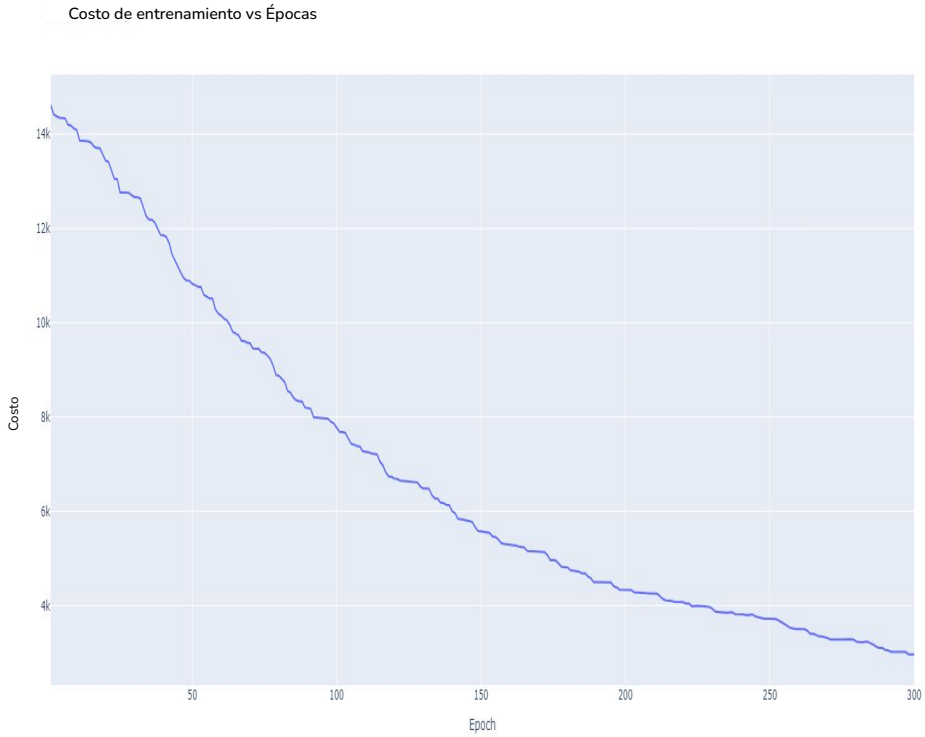
ARTIFICIAL INTELLIGENCE (AI)

VARIACIÓN DE LC: COSTO DE ENTRENAMIENTO VS ÉPOCAS

Learning Constant: 0.0005, Selection = Simple,
Train percentage = 80%, Limit = 300

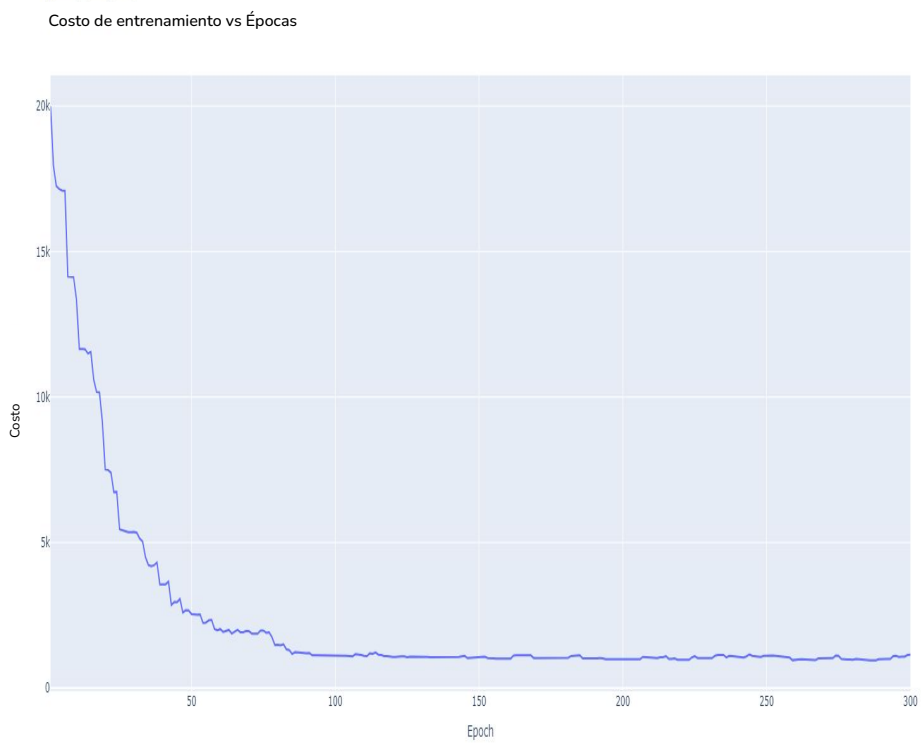


Learning Constant: 0.001, Selection = Simple,
Train percentage = 80%, Limit = 300

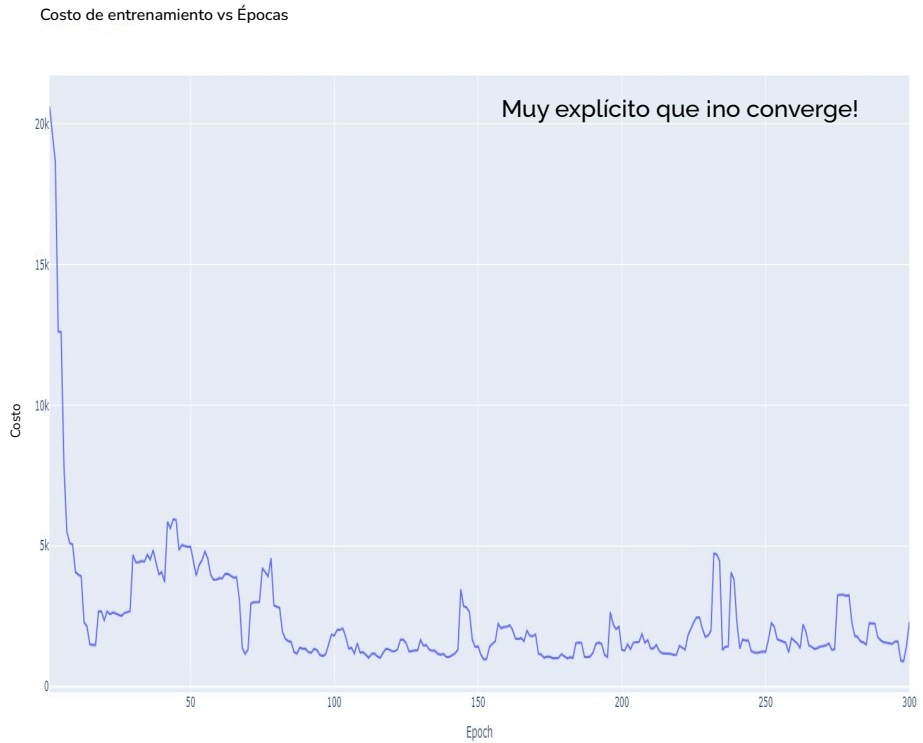


VARIACIÓN DE LC: COSTO DE ENTRENAMIENTO VS ÉPOCAS

Learning Constant: 0.01, Selection = Simple,
Train percentage = 80%, Limit = 300



Learning Constant: 0.03, Selection = Simple,
Train percentage = 80%, Limit = 300



VARIACIÓN DE LEARNING CONSTANT/RATE

Ideal: learning constant bajo pero no mínimo para evitar overfitting ya que el conjunto no es muy grande

Cuanto más pequeño es el aprendizaje, más constante es la convergencia (casi funciones lineales)

Cuanto más alto, genera una convergencia más rápida pero una solución más volátil

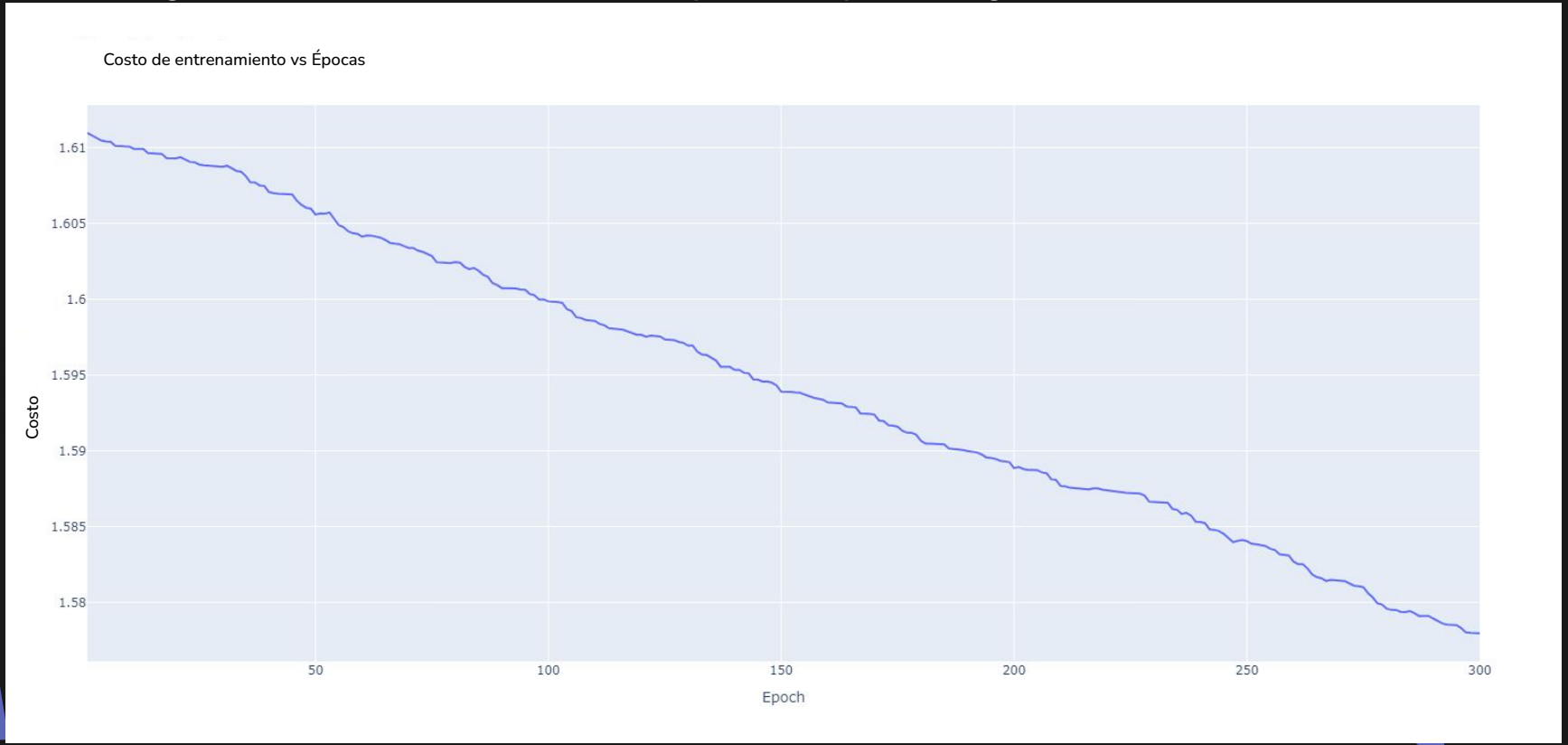
PERCEPTRÓN SIMPLE NO LINEAL

ARTI
CIAL
INTE
IGEN
(AI)

/(AI)/(AI)/

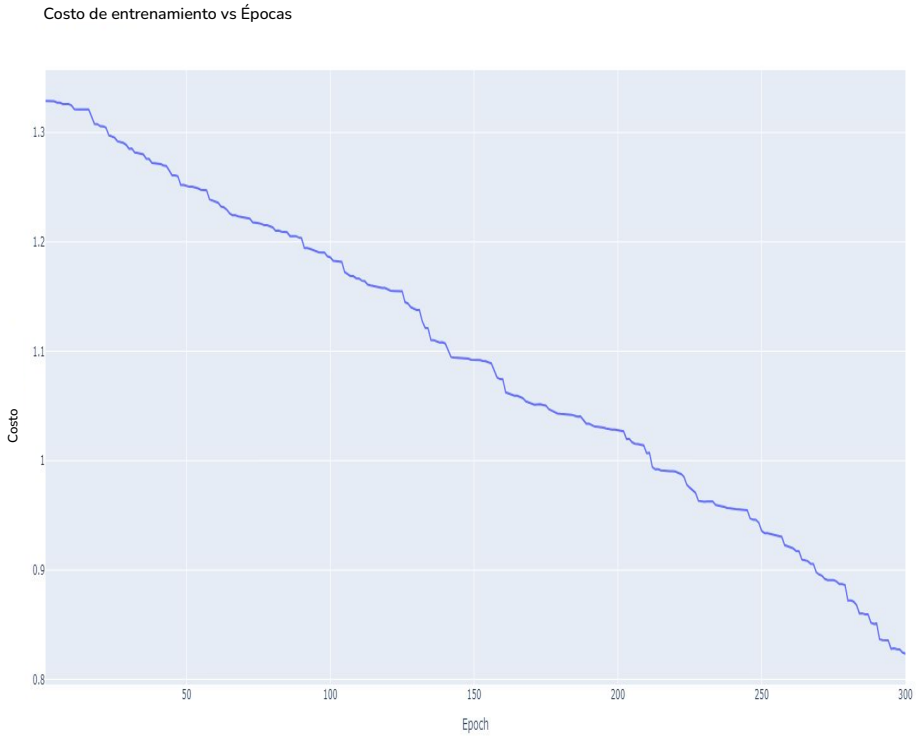
VARIACIÓN DE COSTO DE ENTRENAMIENTO CON BETA

Learning Constant: 0.005, Selection = Simple, Train percentage = 80%, Limit = 300, Beta = 0.1

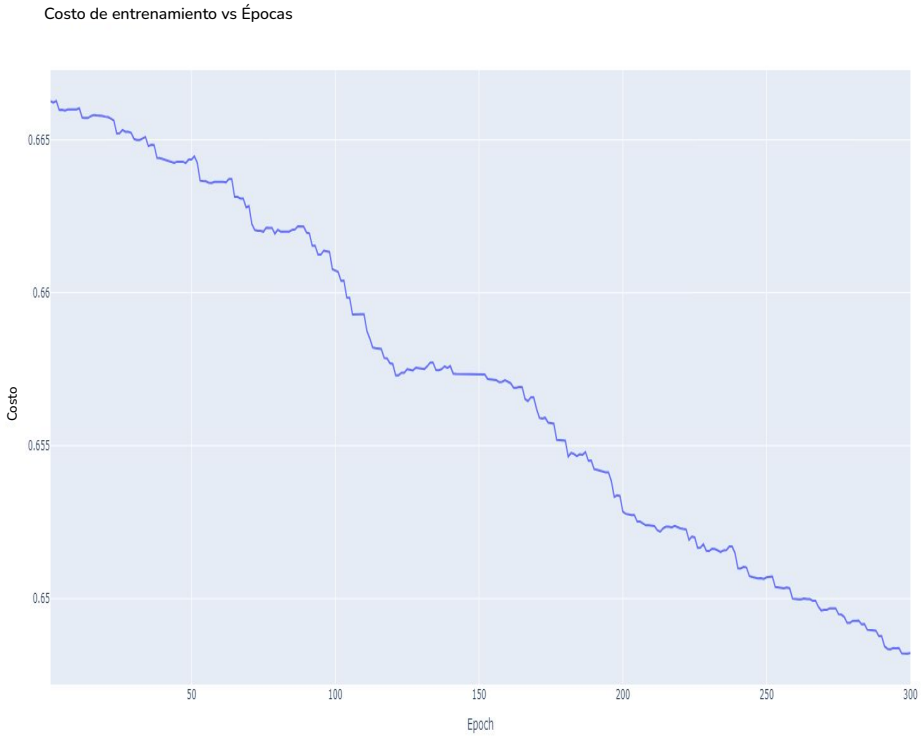


VARIACIÓN DE COSTO DE ENTRENAMIENTO CON BETA

Learning Constant: 0.005, Selection = Simple,
Train percentage = 80%, Limit = 300, Beta = 0.5

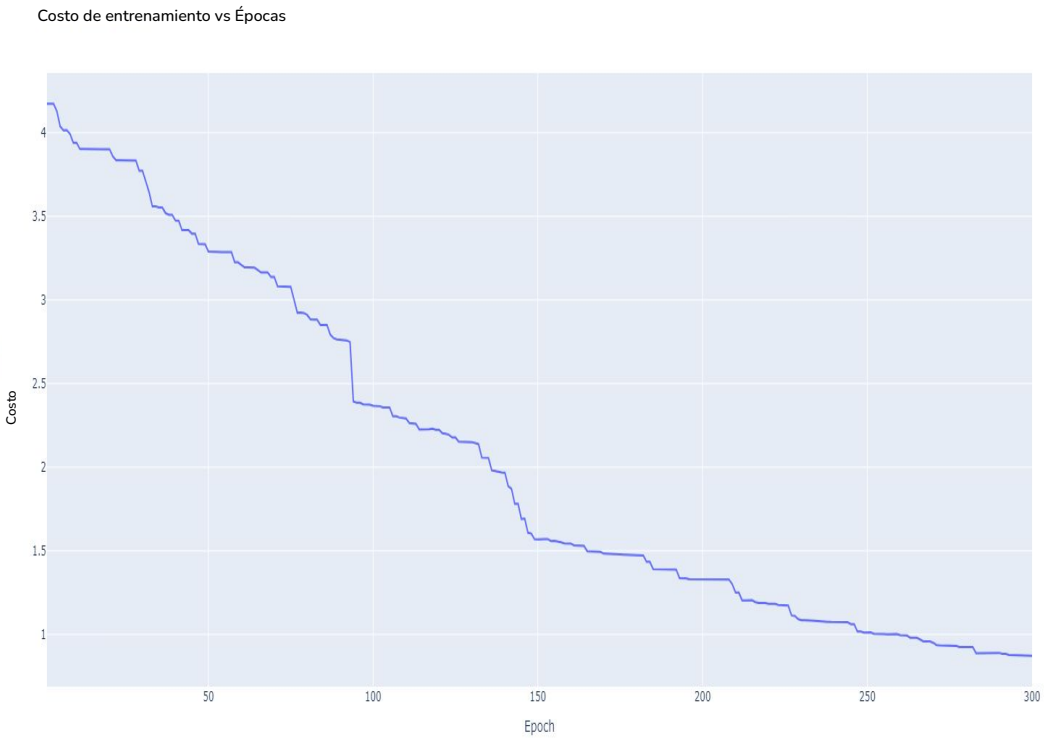


Learning Constant: 0.005, Selection = Simple,
Train percentage = 80%, Limit = 300, Beta = 1



VARIACIÓN DE COSTO DE ENTRENAMIENTO CON BETA

Learning Constant: 0.005, Selection = Simple,
Train percentage = 80%, Limit = 300, Beta = 3



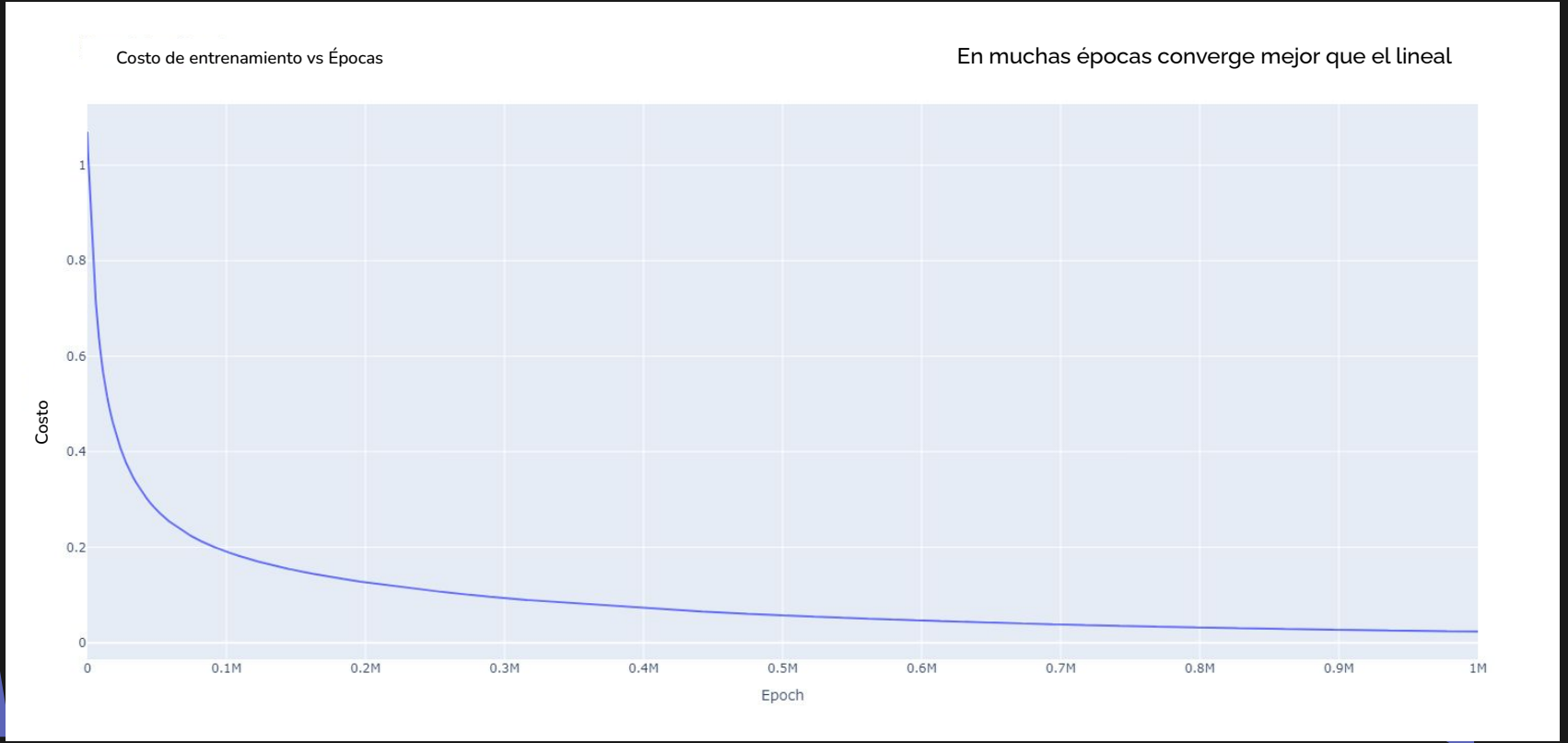
Beta más alto -> más escalonada

Beta más bajo -> más lineal

LIGENCE (AI)

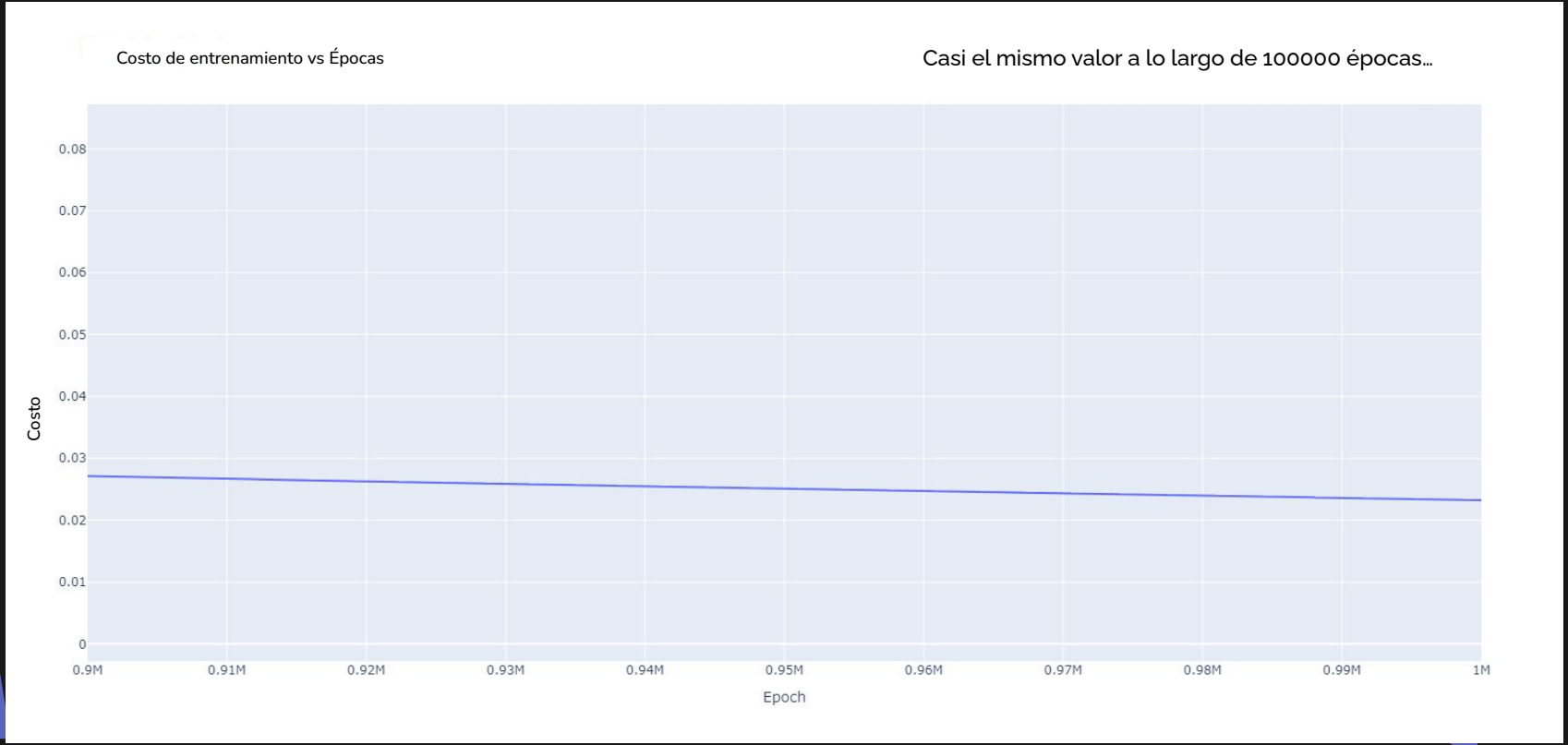
COSTO DE ENTRENAMIENTO VS ÉPOCAS

Learning Constant: 0.005, Selection = Simple, Train percentage = 80%, Limit = 1000000



ZOOM COSTO DE ENTRENAMIENTO VS ÉPOCAS

Learning Constant: 0.005, Selection = Simple, Train percentage = 80%, Limit = 1000000



COMPARACIÓN DE MSE CON DISTINTOS TP ENTRE LINEAL Y NO LINEAL

Training Lineal con TP = 80% : 11795201.367
Testing Lineal con TP = 80% : 372.403
Training No lineal con TP = 80% : 6.4x10^(-6)
Testing No lineal con TP = 80% : 3.67x10^(-3)

Training Lineal con TP = 40% : 4161857.496
Testing Lineal con TP = 40% : 637.8
Training No lineal con TP = 40% : 6.459x10^(-7)
Testing No lineal con TP = 40% : 0.0128

Training Lineal con Tp = 10% : 58282.093
Testing Lineal con TP = 10% : 732.815
Training No lineal con TP = 10%: 1.804413075x10^(-5)
Testing No lineal con TP = 10% : 0.113

Cuidado con underfitting y overfitting!

El MSE del NO lineal es OdM más bajo que el del lineal -> un hiperplano no aproxima bien al conjunto de datos

CONCLUSIONES EJERCICIO 2

El No lineal aproxima mucho mejor que el lineal el conjunto de datos dado ya que este logra converger a un error aceptable y el otro no. Los errores generados por el lineal son órdenes de magnitud superior en comparación a los generados por el no lineal.

CONCLUSIONES EJERCICIO 2

- CONJUNTOS REPRESENTATIVOS Y DIVERSOS
- EVITAR OVER Y UNDER FITTING
- ENTRENAR CON SUFICIENTES DATOS

IDEAL : TRAINING PERCENTAGE = 40%<=X<80%
LEARNING CONSTANT = 0.005
NO LINEAL LOGÍSTICA Y BETA = 1

ARTIFICIAL INTE (AI)

EJERCICIO 3 PERCEPTRÓN MULTICAPA

(AI)

EJERCICIO 3A

XOR con perceptrón multicapa

2 valores por input (coordenada)
2 de salida: que tan verdadero es y que tan falso es

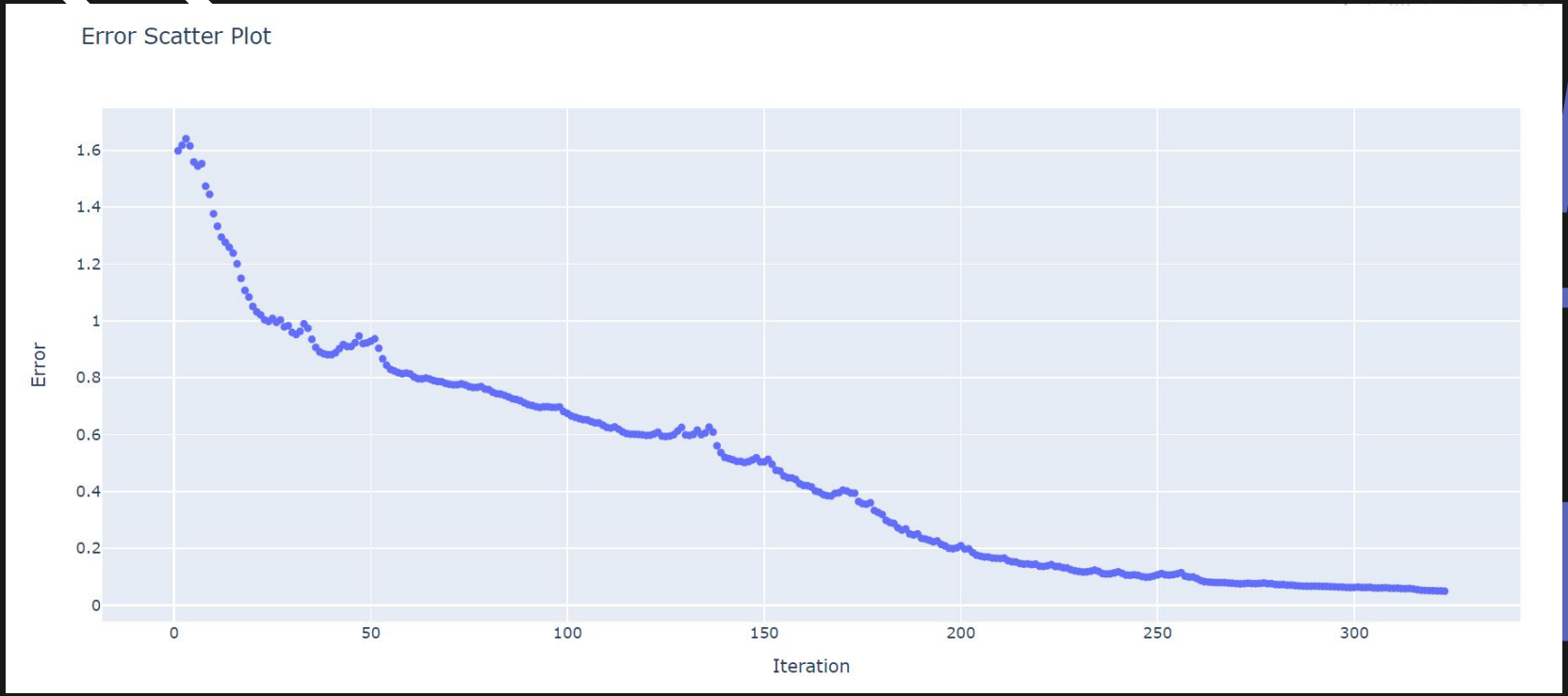
CONFIGURACIÓN BASE

NEURONAS DE ENTRADA	NEURONAS DE SALIDA	CANTIDAD DE CAPAS INTERMEDIAS	NEURONAS POR CAPA
2	2	1	3

CONFIGURACIÓN BASE

LEARNING CONSTANT	ACTIVATION FUNCTION	OPTIMIZACIÓN
0.2	LOGISTIC (BETA = 1)	MOMENTUM (ALPHA = 0.5)

$$E(O) = \frac{1}{2}\sum_{\mu}\sum_{i}(\zeta_i^{\mu} - O_i^{\mu})^2$$



ARTI
AL
TE
EN
(AI)

XOR VALORES DE RETORNO

epsilon = 0.015

INPUT	EXPECTED VALUE	OUTPUT
[-1, 1]	[1, 0]	[0.9853, 0.0144]
[1, -1]	[1, 0]	[0.9854, 0.0143]
[-1, -1]	[0, 1]	[0.0237, 0.9775]
[1, 1]	[0, 1]	[0.0055, 0.9922]

ERROR 0

ACCURACY = 1

PRECISIÓN = 1

RECALL = 1

F1-SCORE = 1

Logra resolver lo que el perceptrón simple no podía

CONCLUSIONES EJERCICIO 3A

PERCEPTRÓN MULTICAPA
PUEDE RESOLVER PROBLEMAS
QUE EL SIMPLE NO

PROBLEMAS DE
CLASIFICACIÓN NO
LINEALMENTE SEPARABLES

ENTRENAR PARA TENER ERROR
O PUEDE SER MUY COSTOSO,
TOMAR UN EPSILON

EJERCICIO 3B

Paridad de números en imagen 7x5

Entrada de 35 inputs (7x5)
Salida de 2: que tan par es y qué tan imar

CONJUNTO DE DATOS REDUCIDO

CON ENTRENAMIENTO COMPLETO

Puede perfectamente determinar si un número es par o no, si se testea, no hay ningún número que no haya visto

SEPARACIÓN ENTRE ENTRENAMIENTO Y TESTEO

CONFIGURACIÓN BASE

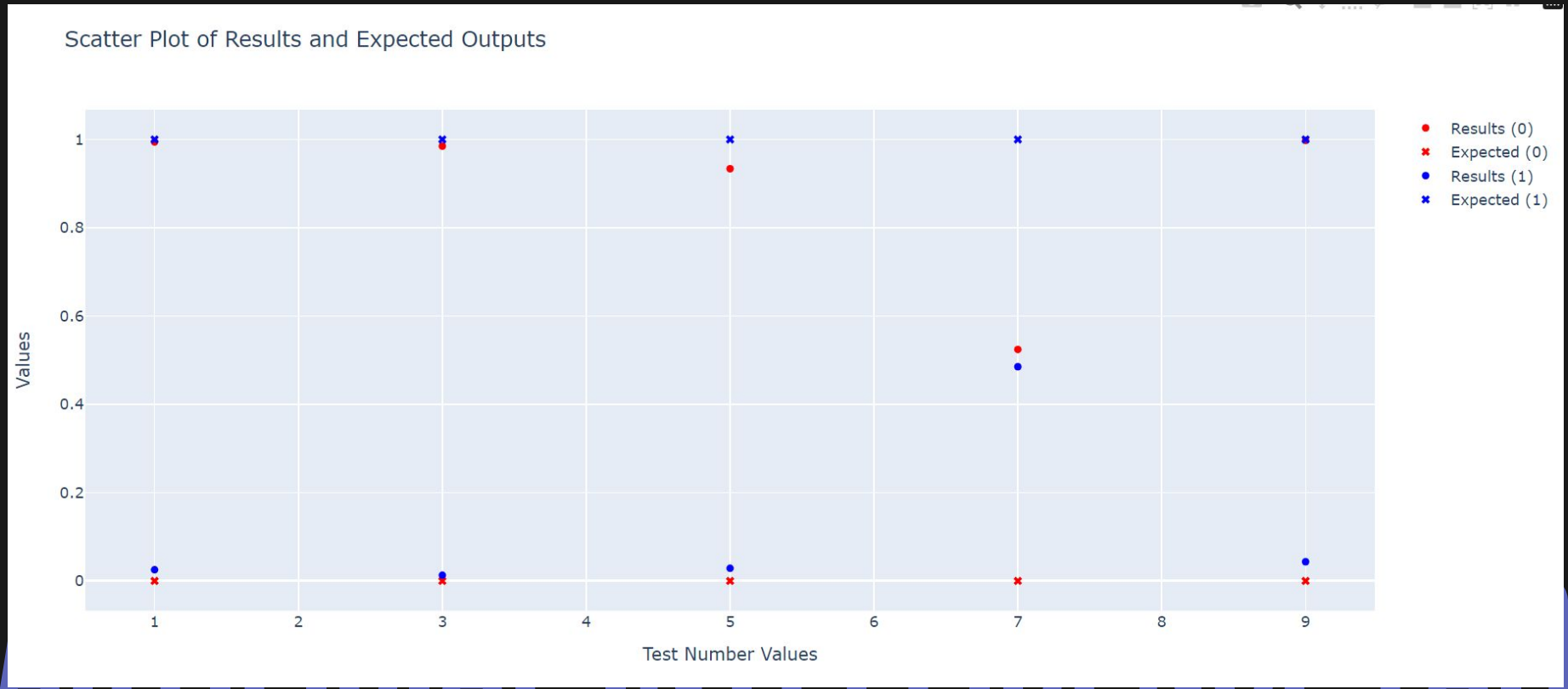
NEURONAS DE ENTRADA	NEURONAS DE SALIDA	CANTIDAD DE CAPAS INTERMEDIAS	NEURONAS POR CAPA
35	2	2	5

CONFIGURACIÓN BASE

LEARNING CONSTANT	ACTIVATION FUNCTION	OPTIMIZACIÓN
0.5	LOGISTIC (BETA = 1)	MOMENTUM (ALPHA = 0.5)

ENTRENANDO CON NÚMEROS PARES
TESTEANDO CON NÚMEROS IMPARES

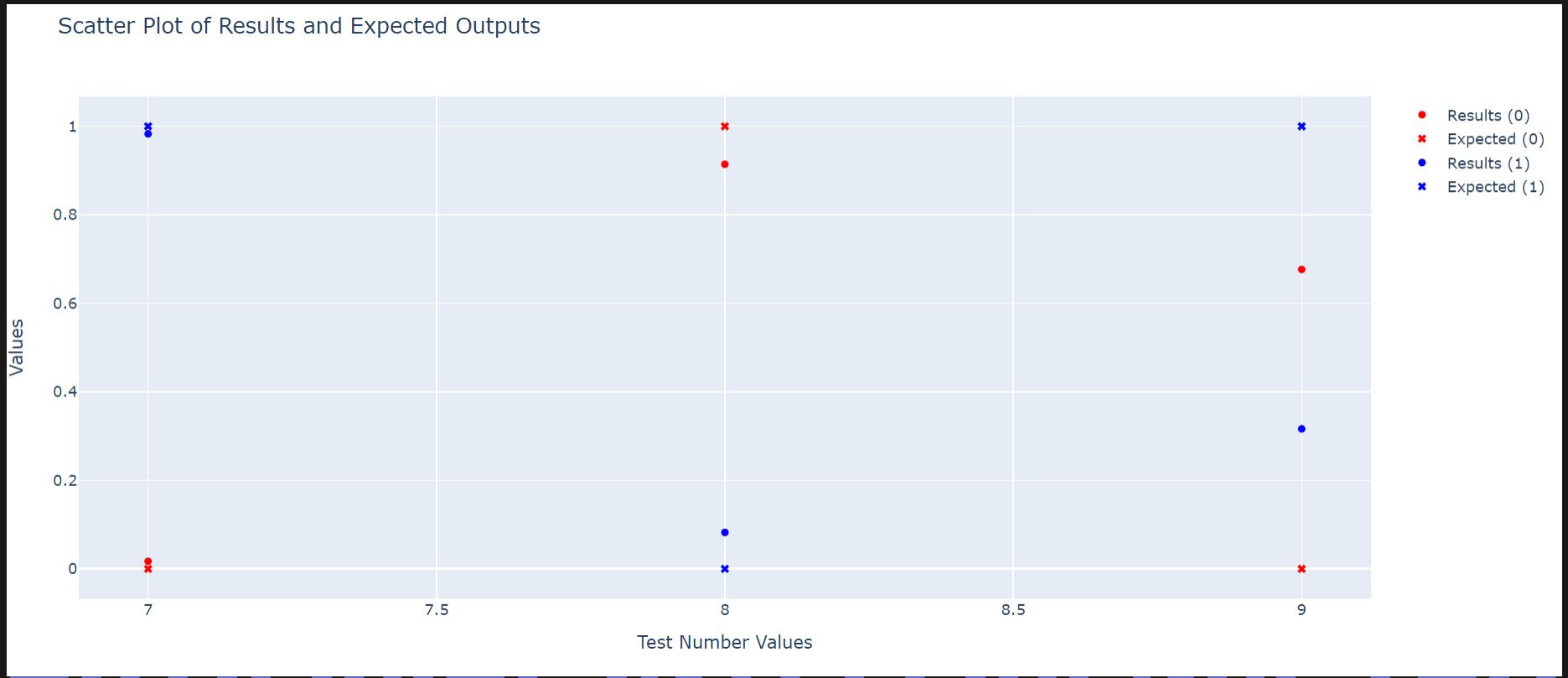
ERROR = 0.001
ACCURACY = 0, PRECISIÓN = 0, RECALL = 0



ARTIFICIAL INTELLIGENCE (AI)

ENTRENANDO CON NÚMEROS <= 6
TESTEANDO CON NÚMEROS > 6

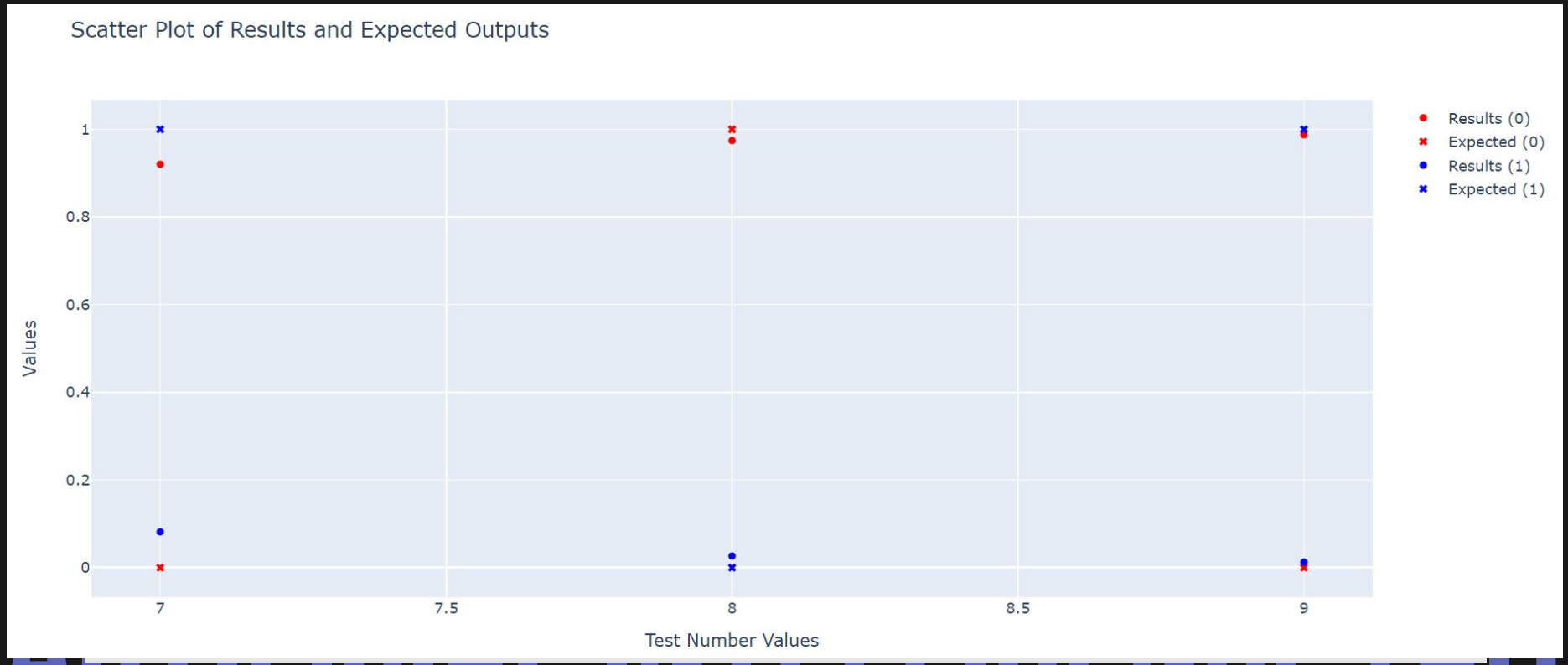
ERROR = 0.001
ACCURACY = ⅔, PRECISIÓN = ⅔,
RECALL = ⅔, F1_SCORE = ⅔



ARTIFICIAL INTELLIGENCE (AI)

ENTRENANDO CON NÚMEROS <= 6
TESTEANDO CON NÚMEROS > 6

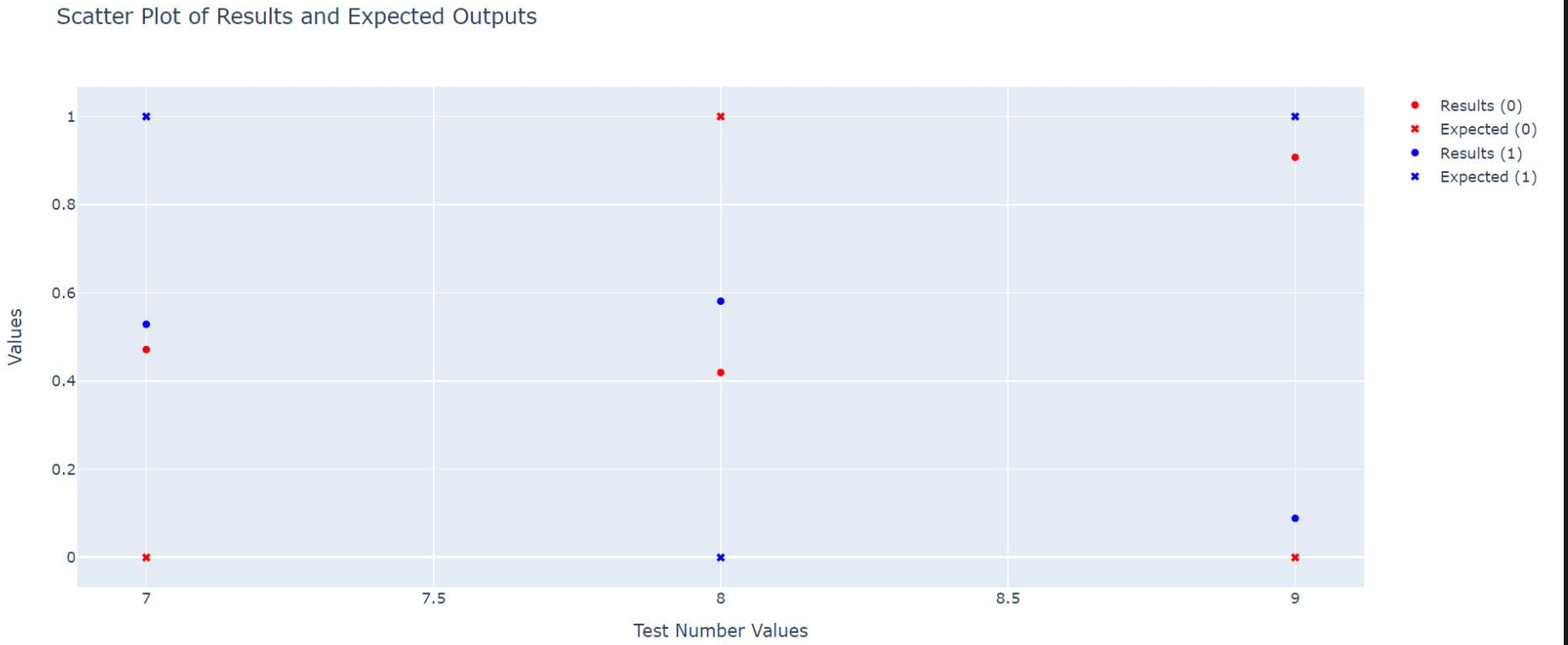
ERROR = 0.001
ACCURACY = ⅓, PRECISIÓN = ⅓,
RECALL = ⅓, F1_SCORE = ⅓



ARTIFICIAL INTELLIGENCE (AI)

ENTRENANDO CON NÚMEROS <= 6
TESTEANDO CON NÚMEROS > 6

ERROR = 0.001
ACCURACY = 0, PRECISIÓN = 0, RECALL = 0



ARTIFICIAL INTELLIGENCE (AI)

CONCLUSIONES EJERCICIO 3B

IMPORTANCIA DE SEPARAR LOS DATOS ENTRE TRAINING Y TESTING

IMPORTANCIA DE TENER VARIOS INPUTS (DISTINTOS) PARA UN MISMO VALOR

EL PERCEPTRÓN MULTICAPA NO PUEDE ESTIMAR ALGO QUE NUNCA VIO

EJERCICIO 3C

Reconocimiento de números

CONFIGURACIÓN BASE

NEURONAS DE ENTRADA	NEURONAS DE SALIDA	CANTIDAD DE CAPAS INTERMEDIAS	NEURONAS POR CAPA
35	10	1	10

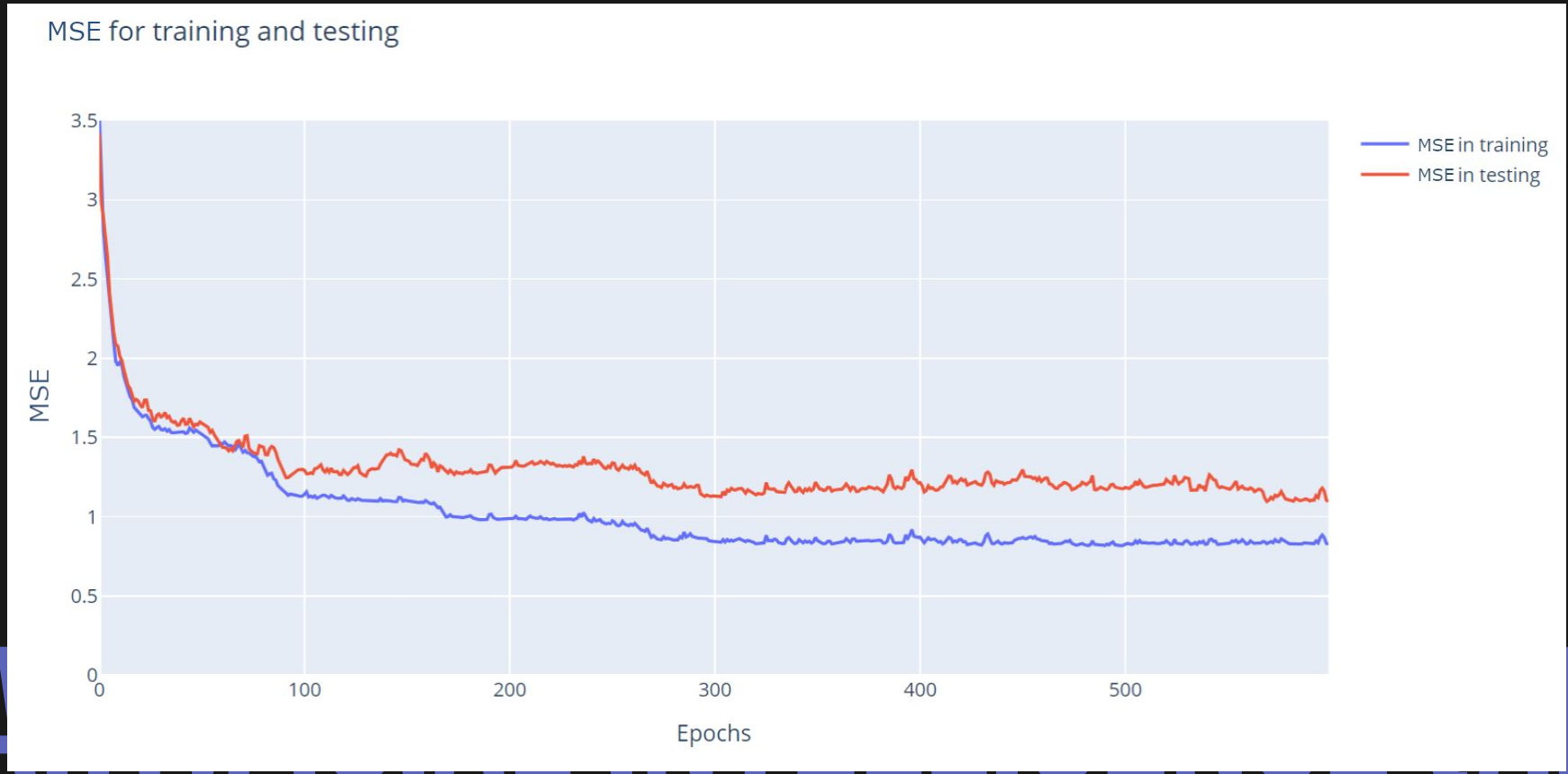
CONFIGURACIÓN BASE

LEARNING CONSTANT	ACTIVATION FUNCTION	OPTIMIZACIÓN
0.05	LOGISTIC (BETA = 1)	MOMENTUM (ALPHA = 0.2)

SEPARACIÓN DE ENTRENAMIENTO Y TESTEO

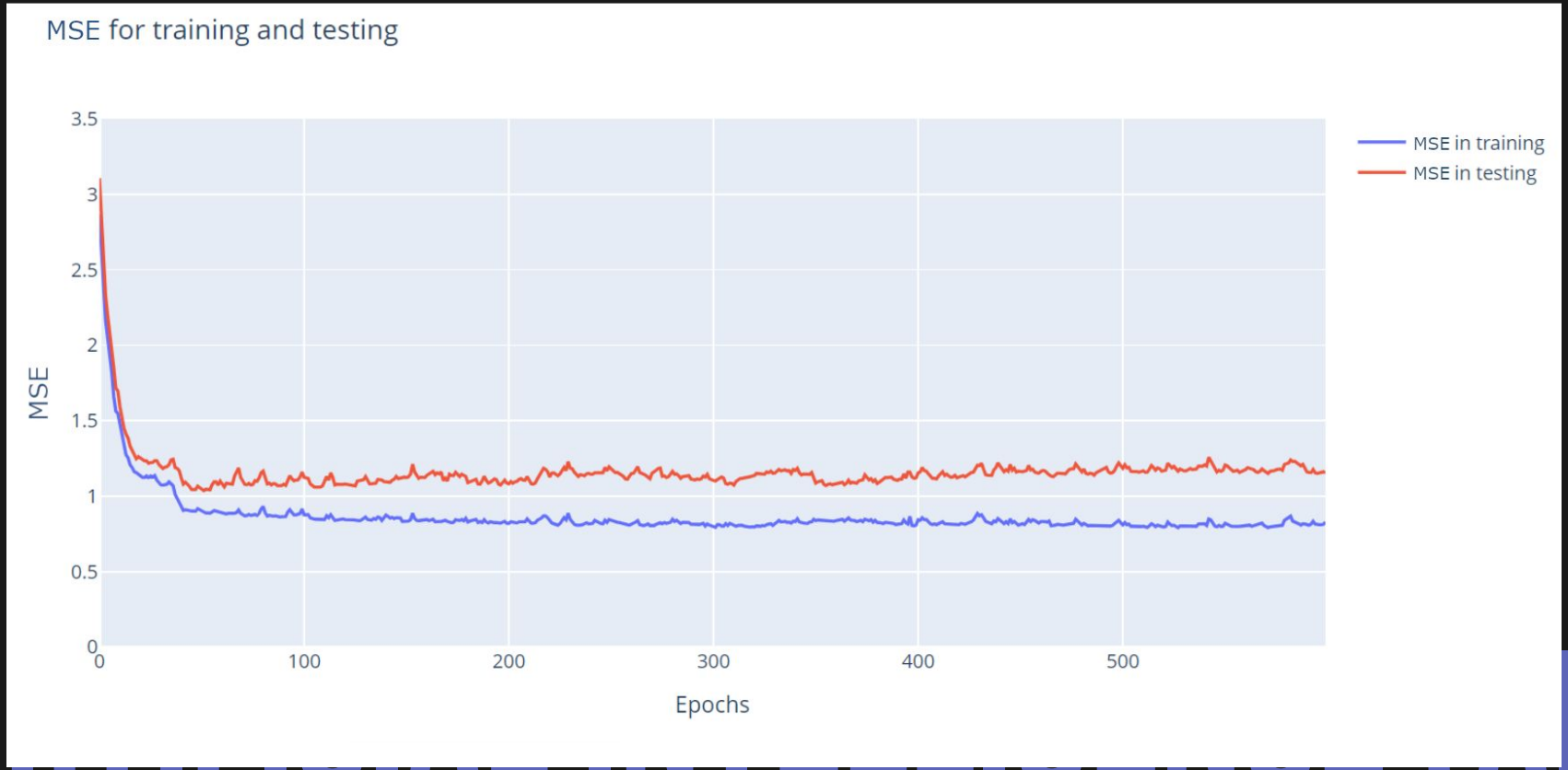
PARTICIÓN 90% 10%

MSE = ((ESPERADO - GENERADO)^2) / N
PROMEDIO DE 10 TESTEOS



PARTICIÓN 80% 20%

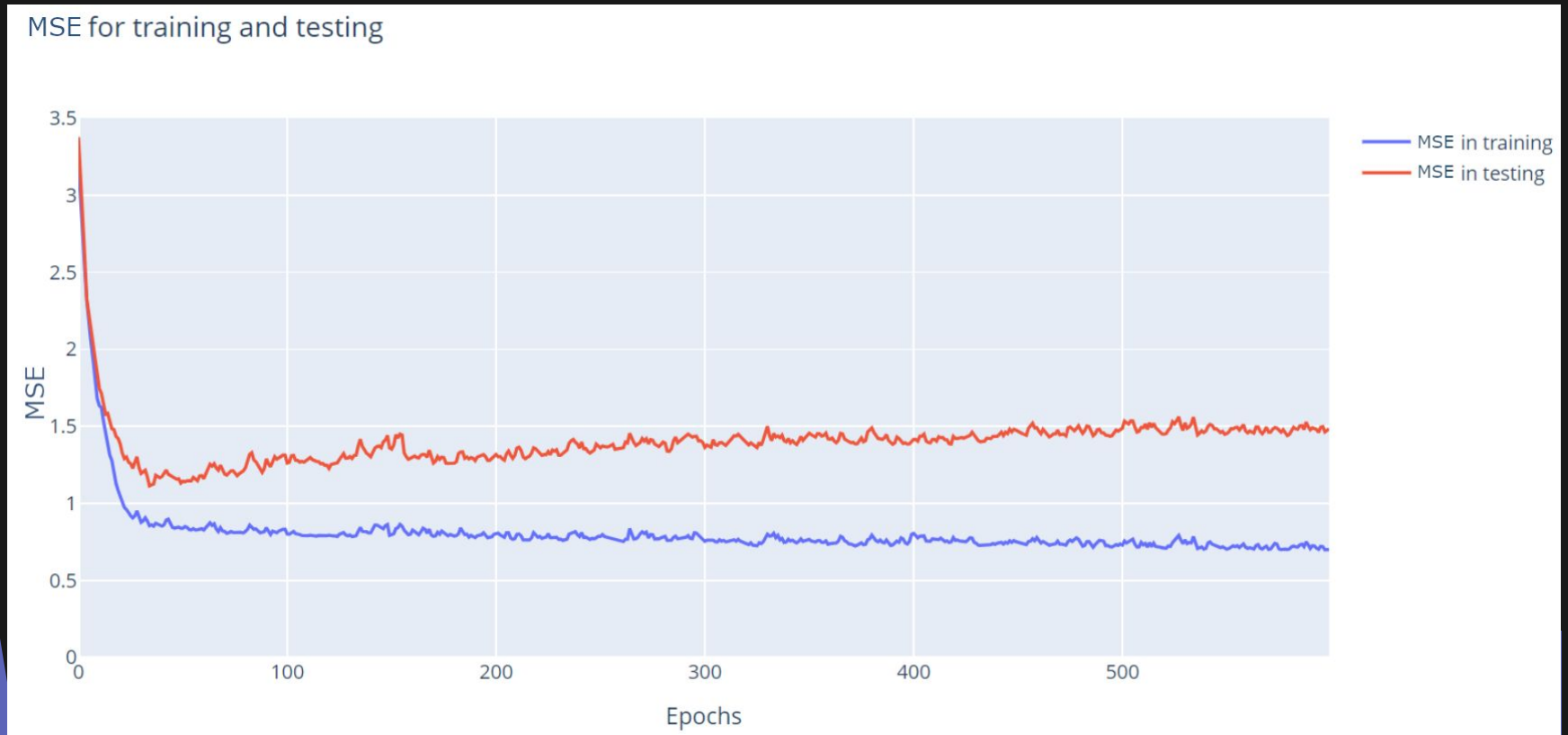
MSE = ((ESPERADO - GENERADO)^2) / N
PROMEDIO DE 10 TESTEOS



ARTIFICIAL INTELLIGENCE (AI)

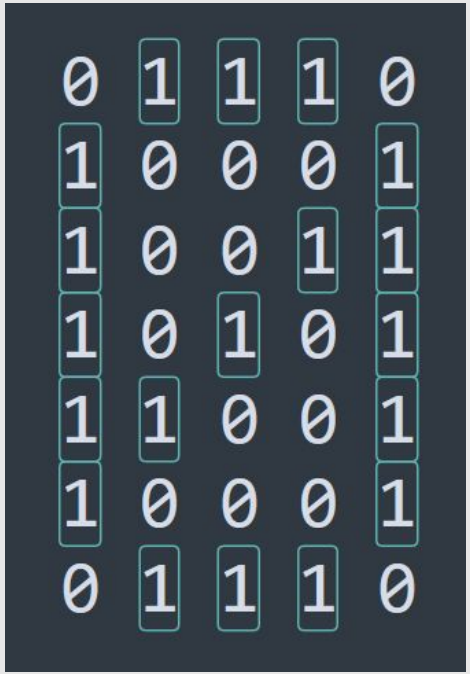
PARTICIÓN 70% 30%

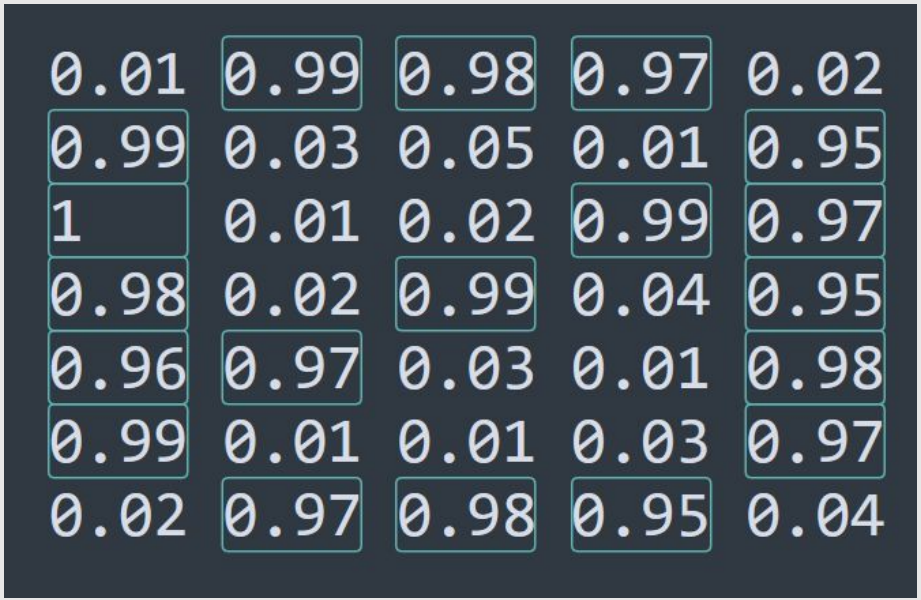
MSE = ((ESPERADO - GENERADO)^2) / N
PROMEDIO DE 10 TESTEOS



ARTIFICIAL INTELLIGENCE (AI)

RUIDO



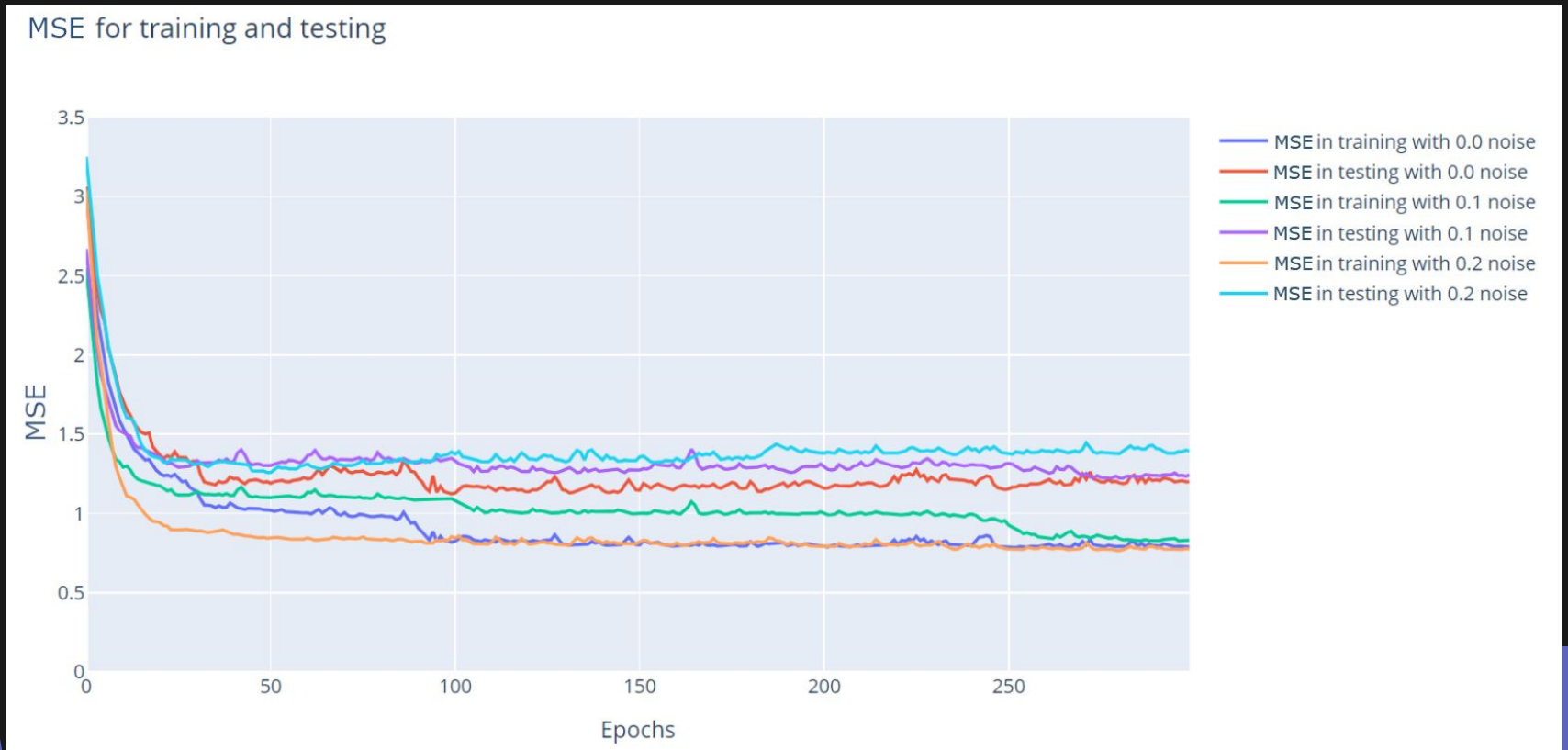


RUIDO MÁXIMO = 0.05

AR AI)
AR AI)
ARTIFICIAL INTELLIGENCE (AI)

RUIDO

PARTICIÓN 70% 30%
PROMEDIO DE 10 CORRIDAS



ARTIFICIAL INTELLIGENCE (AI)

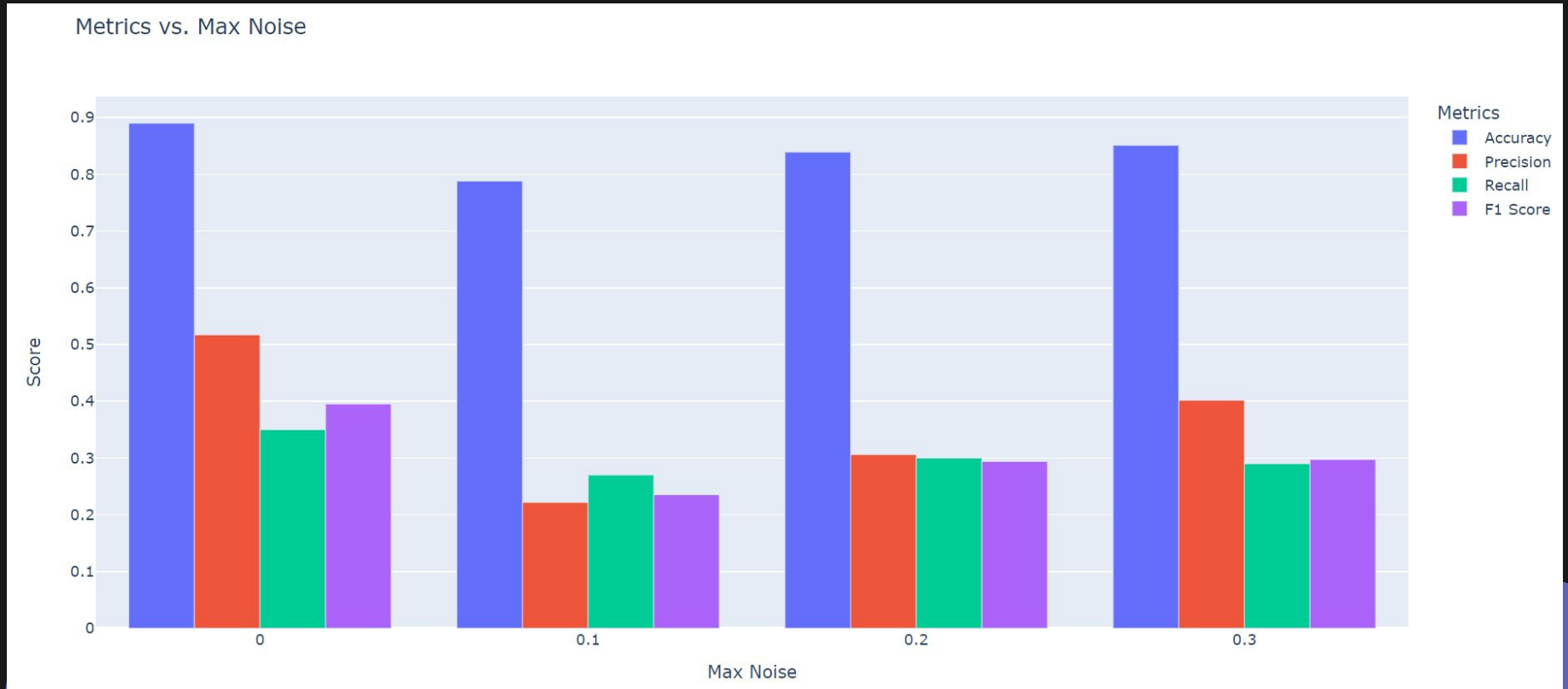
ARTIFICIAL INTE (AI)

ELIMINAMOS LA PARTICIÓN DE TEST Y TRAIN

(AI)

RUIDO

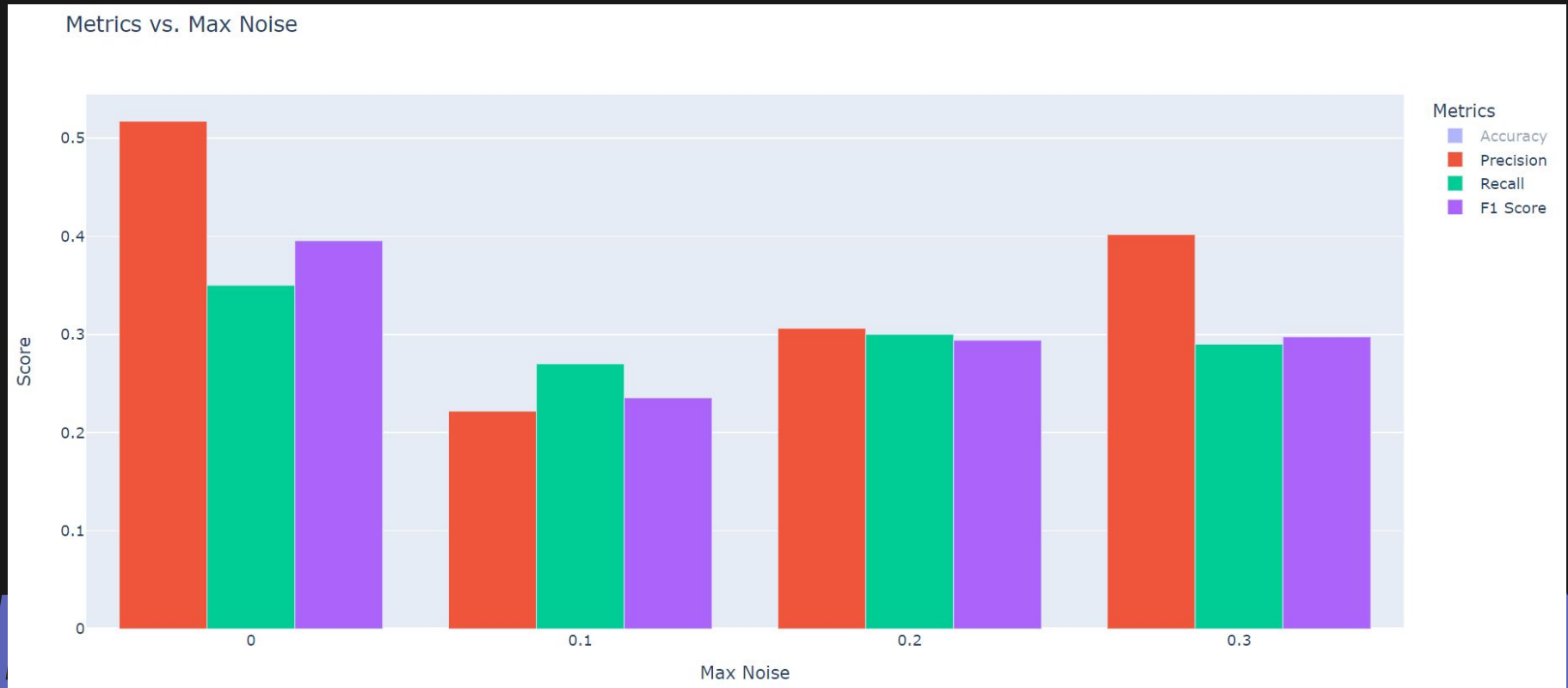
SIN PARTICIÓN
PROMEDIO DE 10 TESTEOS



ARTIFICIAL INTELLIGENCE (AI)

RUIDO

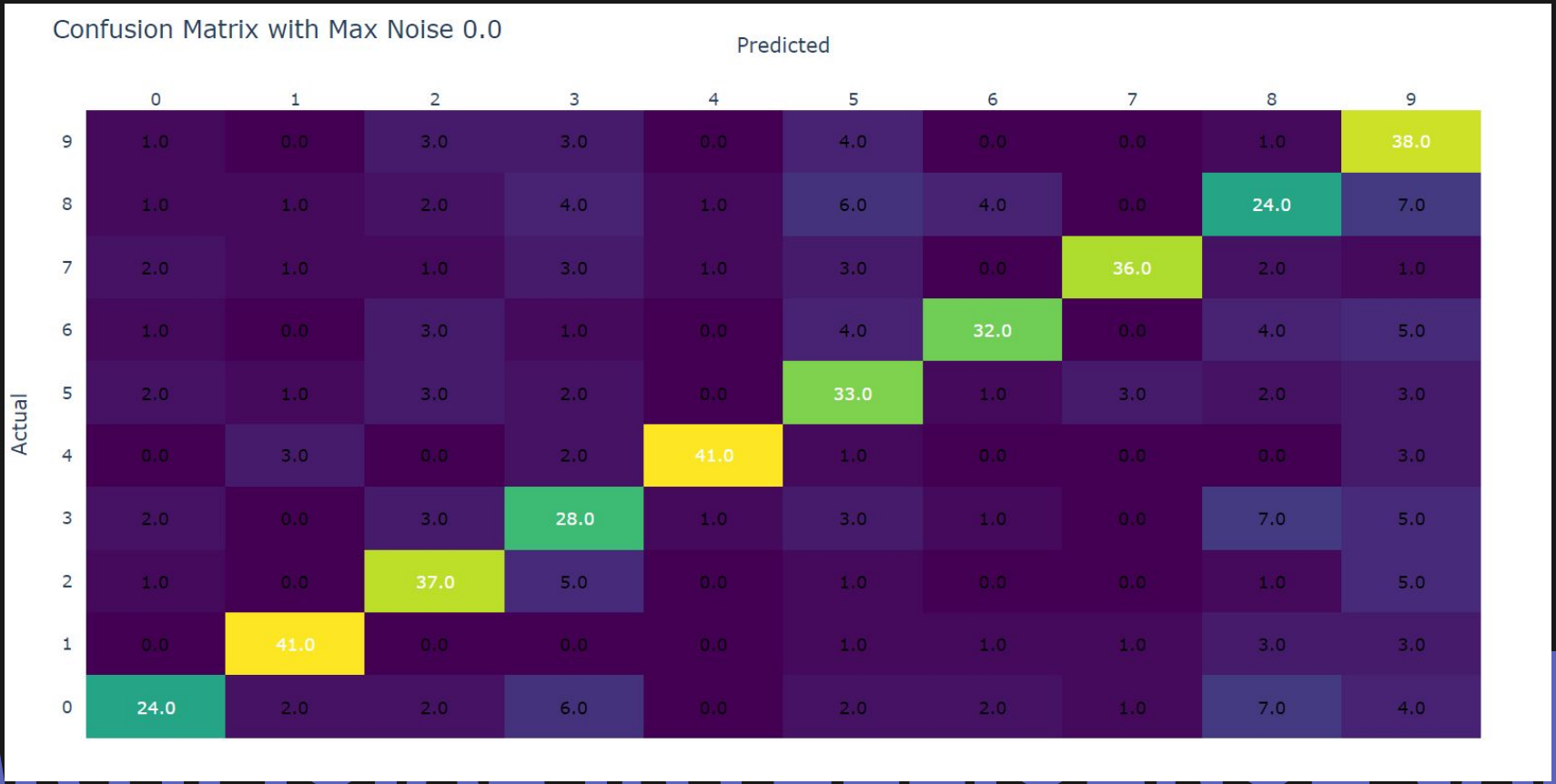
SIN PARTICIÓN
PROMEDIO DE 10 TESTEOS



ARTIFICIAL INTELLIGENCE (AI)

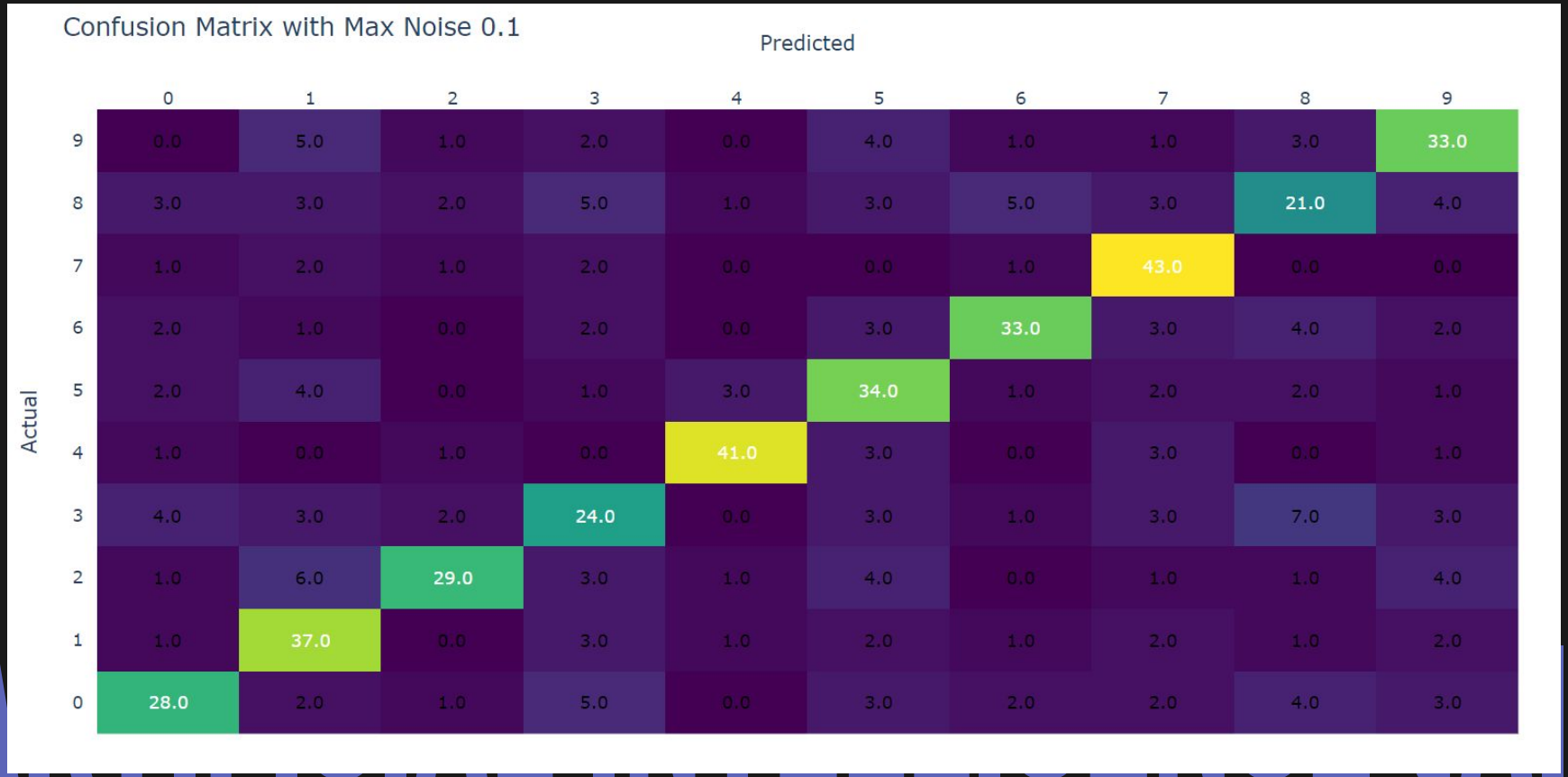
MATRIZ DE CONFUSIÓN

SIN PARTICIÓN
PROMEDIO DE 50 TESTEOS



MATRIZ DE CONFUSIÓN

SIN PARTICIÓN
PROMEDIO DE 50 TESTEOS

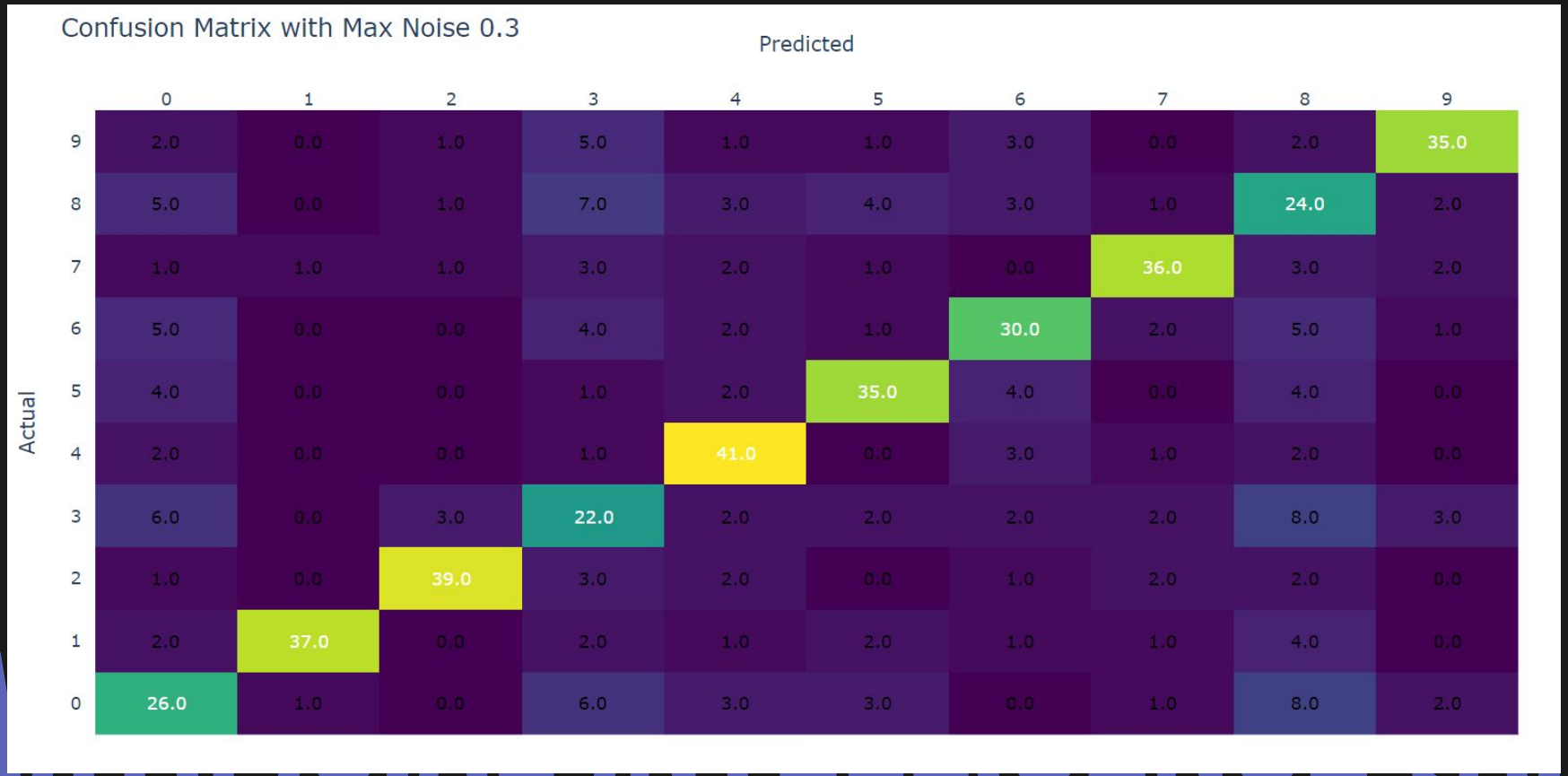


MATRIZ DE CONFUSIÓN

SIN PARTICIÓN
PROMEDIO DE 50 TESTEOS

MATRIZ DE CONFUSIÓN

SIN PARTICIÓN
PROMEDIO DE 50 TESTEOS



CONCLUSIONES EJERCICIO 3C

AL AUMENTAR EL RUIDO, LE CUESTA MÁS DISTINGUIR CIERTOS NÚMEROS

AL AUMENTAR LA PARTICIÓN DE TESTING, AUMENTA CONSIDERABLEMENTE EL ERROR DURANTE EL TESTEO

ARTIFICIAL INTE (AI)

COMPARACIÓN MÉTODOS DE OPTIMIZACIÓN

(AI)

GRADIENTE DESCENDENTE VS MOMENTUM

GRADIENTE DESCENDENTE

$$w_{k+1} = w_k + \Delta w_k$$

MOMENTUM

$$w_{k+1} = w_k + \Delta w_k + \alpha \Delta w_{k-1}$$

PASO ALPHA: 0.01

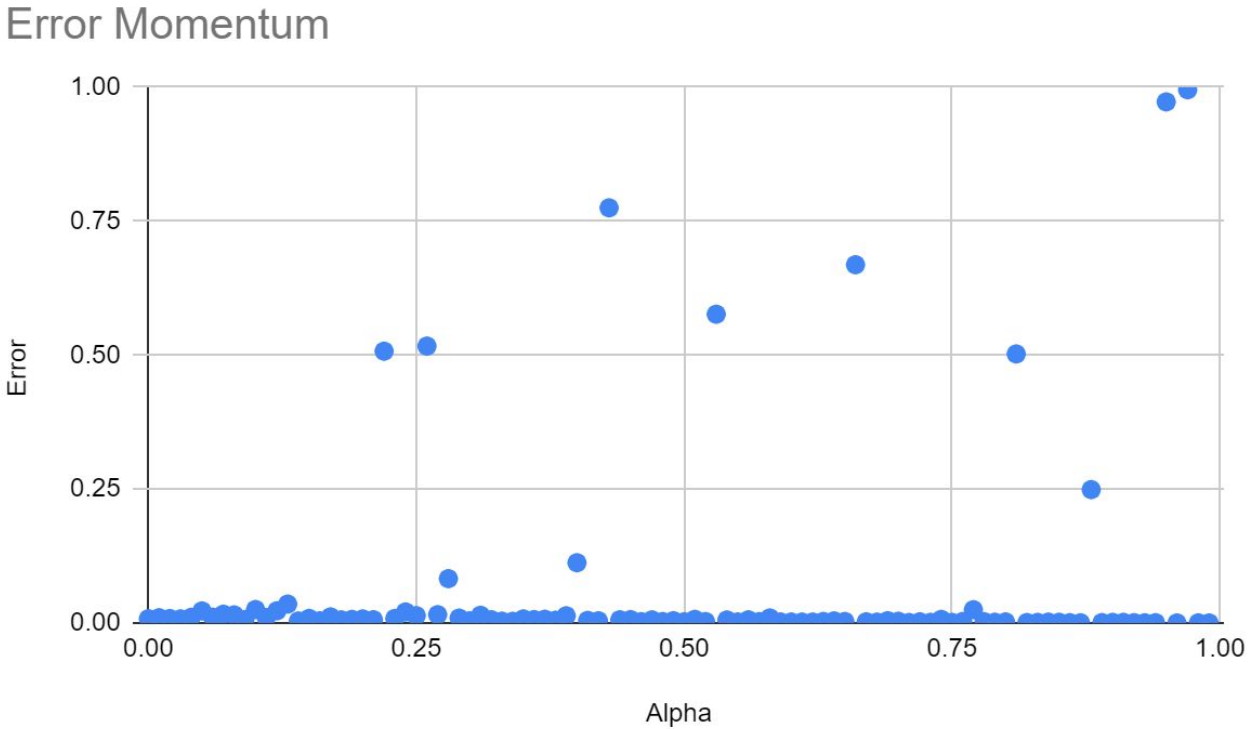
CONFIGURACIÓN USADA ÍTEM A

```
{
  "seed": -1,
  "learning_constant": 0.2,

  "limit": 2000,
  "epsilon": 0.000 ,
  "batch_size": 1,

  "hidden_layer_amount": 2,
  "neurons_per_layer": 5,

  "activation_function": {
    "type": "logistic_function",
    "beta": 1
  },
  "optimization_method": {
    "type": "momentum",
    "alpha": 0.5
  },
  "generate_error_graph": false,
  "print_final_values": true
}
```

COMPARACIÓN ÍTEM A

Hay valores que no llegan a converger...

Criterio para ver valores que convergen: usamos valores menores que 0.025

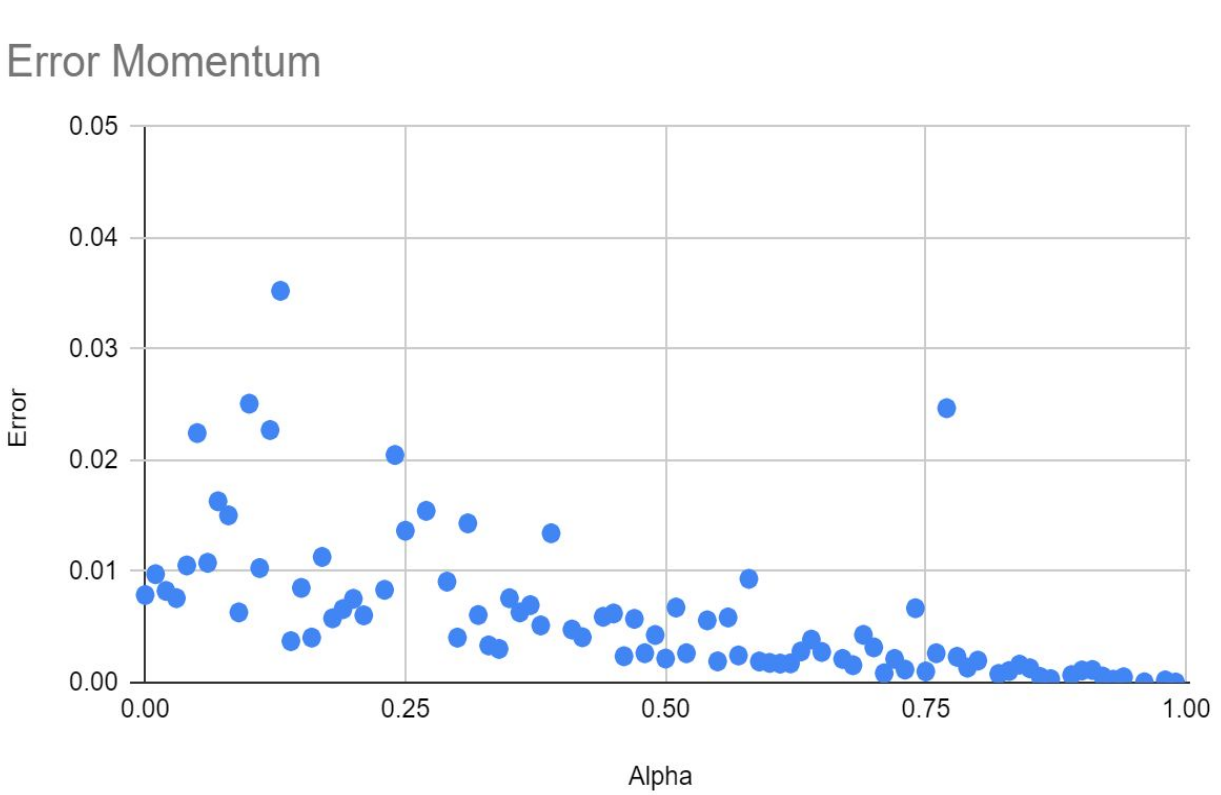


ZOOM ÍTEM A

Sin momentum
- Error: 0.00787778

Con momentum
(Análisis con paso 0.01)
Peor resultado
- Alpha: 0.97
- Error: 0.99519343

Mejor Resultado
- Alpha: 0.99
- Error: 0.00000424
- Promedio (< 0.025)
  - 0.005675506322
- Promedio:
  - 0.0651162021
- Desviación
  - 0.1947997609

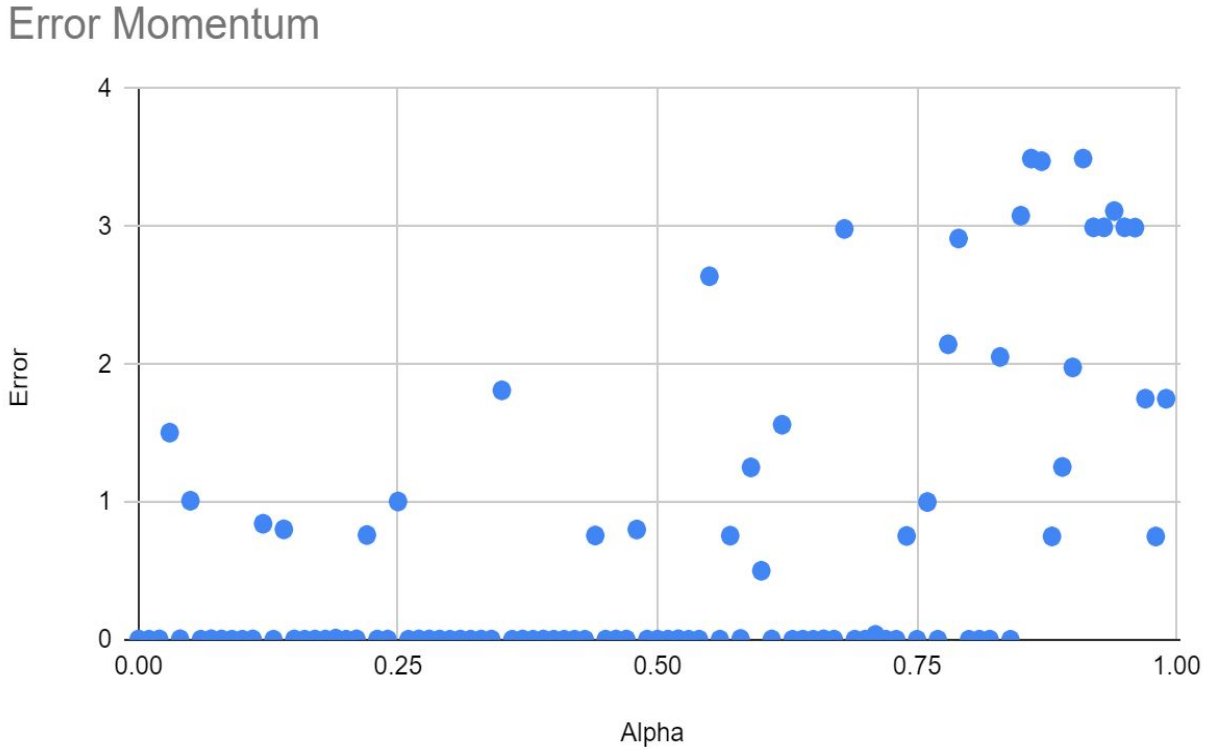


CONFIGURACIÓN USADA ÍTEM B

```
{
  "seed": -1,
  "training_data_input": "../../training_data/ej3-digitos.txt",
  "training_data_output": "../../training_data/ej3-B-respuestas.txt",

  "learning_constant": 0.5,
  "limit": 2000,
  "epsilon": 0.000,
  "batch_size": 1,

  "hidden_layer_amount": 1,
  "neurons_per_layer": 3,

  "activation_function": {
    "type": "logistic_function",
    "beta": 1
  },
  "optimization_method": {
    "type": "momentum",
    "alpha": 0.5
  },
  "print_results": true,
  "generate_graph": true
}
```

COMPARACIÓN ÍTEM B

Nuevamente tenemos outliers, veamos con zoom.

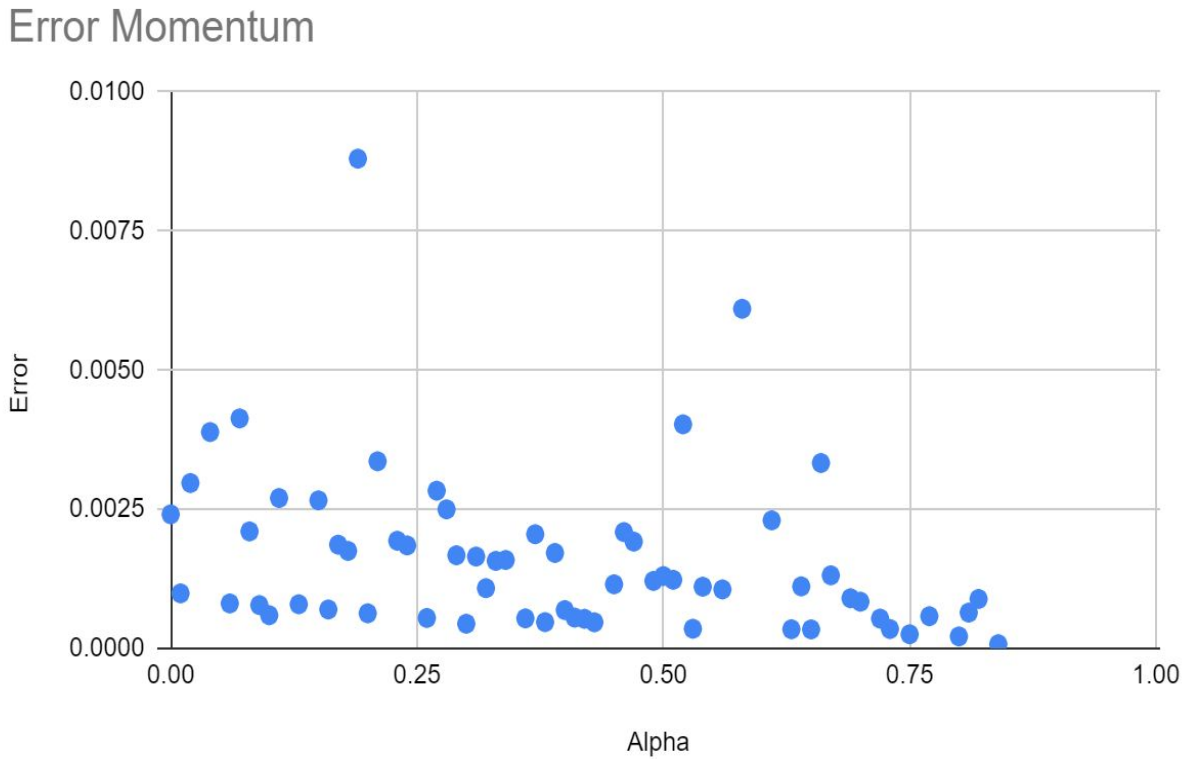


COMPARACIÓN ÍTEM B (ZOOM)

Sin momentum
- Error: 0.00240921

Con momentum
(Análisis con paso 0.01)
Peor resultado
- Alpha: 0.86
- Error: 3.493007

Mejor Resultado
- Alpha: 0.84
- Error: 0.0000758
- Promedio (< 0.025)
  - 0.0015971125
- Promedio:
  - 0.6484047638
- Desviación
  - 1.065674812

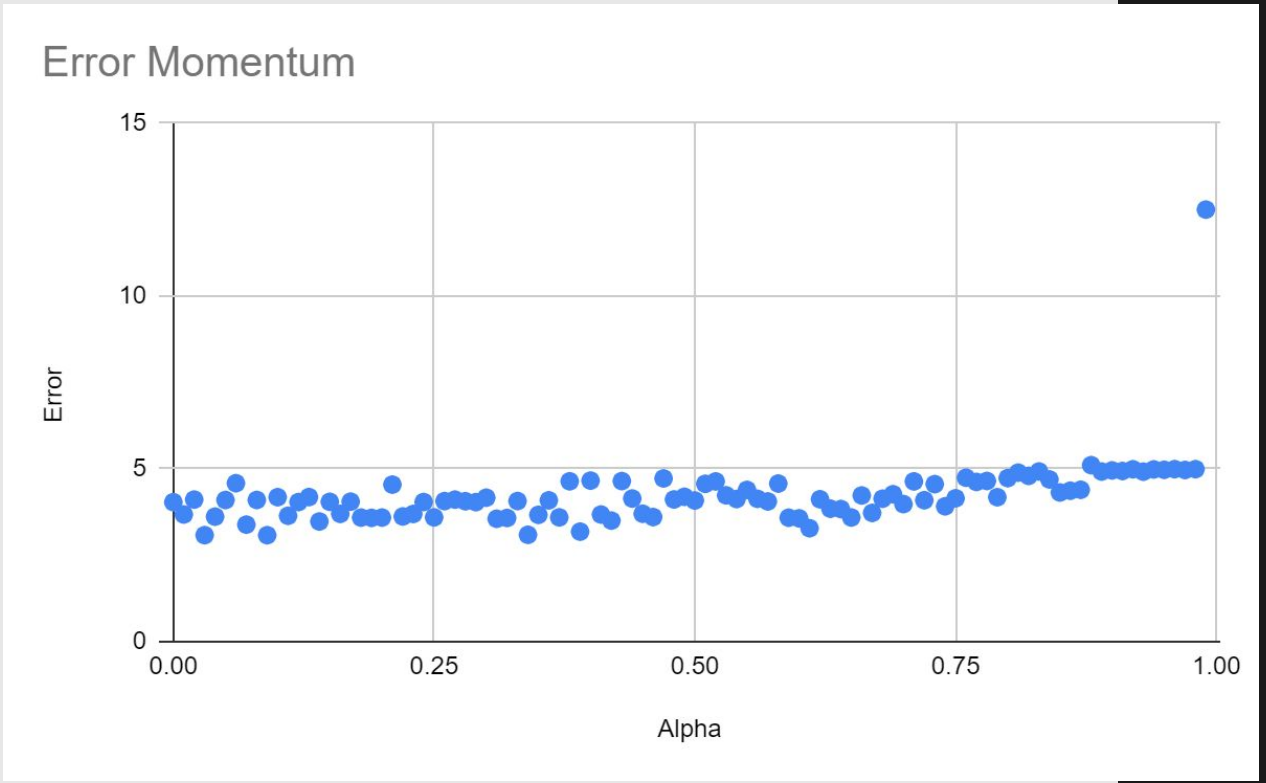


CONFIGURACIÓN USADA ÍTEM C

```
{
  "seed": -1,
  "training_data_input": "../../training_data/ej3-digitos.txt",
  "training_data_output": "../../training_data/ej3-C-respuestas.txt",

  "selection_method" : {
    "size": 5000,
    "method": "simple",
    "proportion": 0.9
  },
  "activation_function_beta": 1,

  "learning_constant": 0.05,
  "limit": 5000,
  "epsilon": 0.00,
  "batch_size": 1,

  "hidden_layer_amount": 1,
  "neurons_per_layer": 3,

  "optimization_method": {
    "type": "momentum",
    "alpha": 0.20
  }
}
```

COMPARACIÓN ÍTEM C

Nuevamente tenemos un outlier...



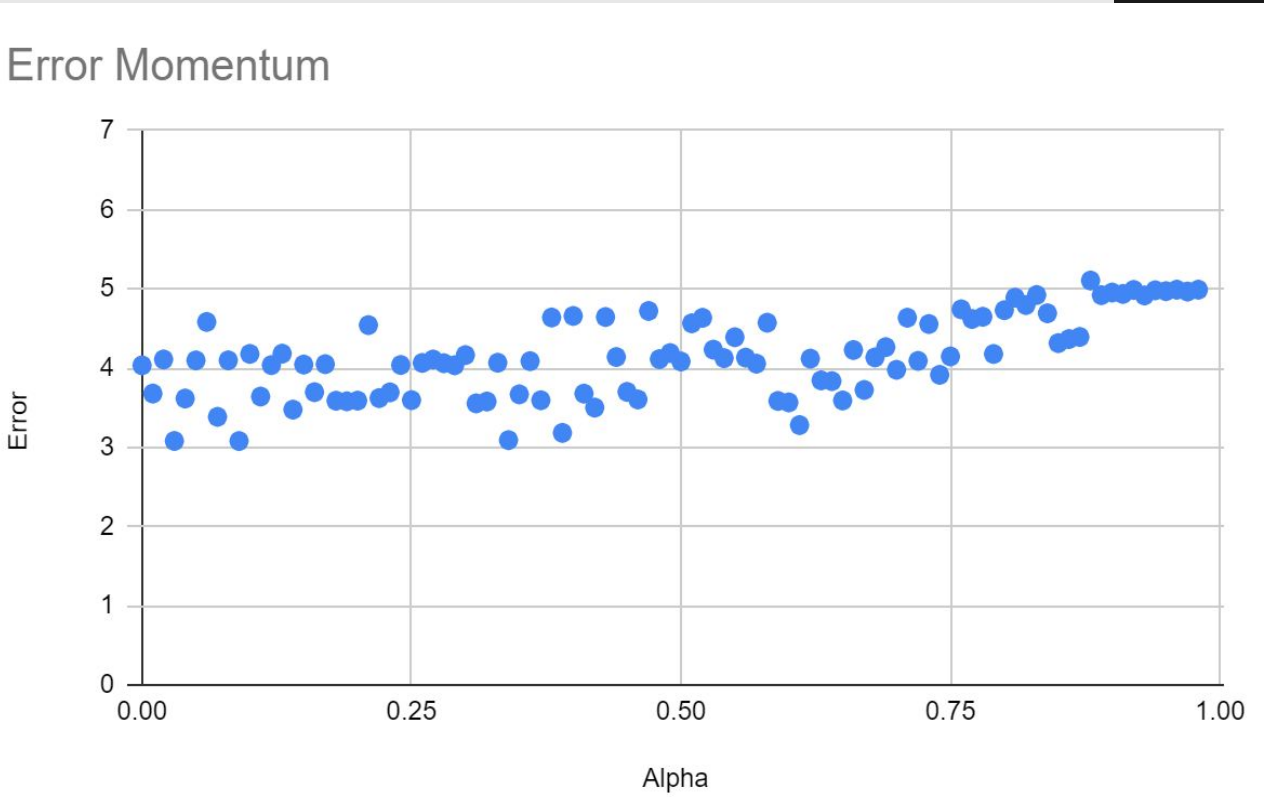
COMPARACIÓN ÍTEM C (ZOOM)

Sin momentum
- Error: 4.03485016

Con momentum (Análisis con paso 0.01)

Peor resultado
- Alpha: 0.99
- Error: 12.5

Mejor Resultado
- Alpha: 0.09
- Error: 3.07902272
- Promedio (< 7)
  - 4.146266379
- Promedio:
  - 4.229803716
- Desviación
  - 0.9796778293



CONCLUSIONES

- Si bien por motivos de tiempo de ejecución quedaron outliers. Usar métodos de optimización trae en líneas generales **MEJORES RESULTADOS** que no usarlos.
  - El promedio al utilizar alpha de 0 a 1 suele resultar mejor que no usarlo
  - En nuestras pruebas, nunca conseguimos un resultado óptimo con alpha = 0 que pueda superar otro valor de alpha probado
- Hay varios, pero no todos, alphas que ayudan a conseguir resultados superiores

- Hay que tener cuidado con cuánta importancia le damos a los pasos anteriores.

Outliers:

- Varianza vector gradiente => Puede llevar a movimientos erráticos
- Taza de aprendizaje alta para ciertos valores de alfa => Momentum puede dar un paso muy largo sobrepasando el mínimo. (ADAM y RMSProp prevén esto)