Tarea 4 - Programación Genética

October 18, 2025

1 Tarea 4: Programación Genética

1.1 4.1 Tabla de verdad con F1 = {and, or, not}

Encontrar las funciones que hacen cierta la tabla de verdad anexa. Con el propósito de observar el impacto de una buena selección en el conjunto de funciones, se usarán dos F1 = {and, or, not}, F2 = {and, or, not, xor} y el conjunto terminal T = {A, B, C, [0, 1]}. Donde A, B, C, solo los valores de la tabla adjunta, y 0, 1 son constantes opcionales.

[1]:
```
import deap
import math
import operator
import random
import functools
import numpy as np
import pandas as pd
from deap import gp, base, creator, tools, algorithms
import plotly.graph_objs as go

# Plotly render svgs
import plotly.io as pio
pio.renderers.default = "svg"
```

[2]:
```
# Cargar tabla de verdad desde un archivo CSV
tabla = pd.read_csv('data/Paridad.csv')
```

[3]:
```
tabla
```

[3]:

	A	B	C	S
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

[4]:
```python
POP_SIZE = 200
N_GEN = 40
CXPB = 0.8   # probabilidad de cruzamiento
MUTPB = 0.2  # probabilidad de mutación
RANDOM_SEED = 42
random.seed(RANDOM_SEED)
```

[5]:
```python
# Primitivas con tres variables A,B,C
pset = gp.PrimitiveSet("MAIN", 3)
pset.renameArguments(ARG0='A')
pset.renameArguments(ARG1='B')
pset.renameArguments(ARG2='C')

# Funciones
pset.addPrimitive(operator.and_, 2)
pset.addPrimitive(operator.or_, 2)
pset.addPrimitive(operator.not_, 1)

# Terminales constantes
pset.addTerminal(0)
pset.addTerminal(1)

# Fitness
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", gp.PrimitiveTree, fitness=creator.FitnessMin)

# Toolbox
toolbox = base.Toolbox()
toolbox.register("expr", gp.genFull, pset=pset, min_=1, max_=3)
toolbox.register("individual", tools.initIterate, creator.Individual, toolbox.
 ↪expr)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("compile", gp.compile, pset=pset)

def evalParity(individual):
    """
    Evaluar el individuo en la tabla de verdad.
    El error es la suma de los errores cuadráticos.
    """
    func = toolbox.compile(expr=individual)
    errors = (
        (func(row.A, row.B, row.C) - row.S)**2
        for _, row in tabla.iterrows()
    )
    return math.fsum(errors),

toolbox.register("evaluate", evalParity) # Evaluación del individuo
```

```
toolbox.register("select", tools.selTournament, tournsize=3) # Selección por␣
 ↪torneo
toolbox.register("mate", gp.cxOnePoint) # Cruce de un punto
toolbox.register("expr_mut", gp.genFull, min_=0, max_=2) # Mutación
toolbox.register("mutate", gp.mutUniform, expr=toolbox.expr_mut, pset=pset)

# Limitar altura
toolbox.decorate("mate", gp.staticLimit(key=operator.attrgetter("height"),␣
 ↪max_value=6))
toolbox.decorate("mutate", gp.staticLimit(key=operator.attrgetter("height"),␣
 ↪max_value=6))
```

[6]:

```
# Ejecutar evolución con F1 (and, or, not)

# Estadísticas y registro del mejor
stats = tools.Statistics(lambda ind: ind.fitness.values[0])
stats.register("min", min)
stats.register("avg", np.mean)
stats.register("std", np.std)

hof = tools.HallOfFame(1) # Mejor individuo

pop = toolbox.population(n=POP_SIZE) # Población inicial

pop, log = algorithms.eaSimple(pop, toolbox, cxpb=CXPB, mutpb=MUTPB, ngen=N_GEN,
                               stats=stats, halloffame=hof, verbose=True)

best = hof[0]
print("Mejor individuo F1:", best)
print("Altura:", best.height)
print("Fitness (SSE):", best.fitness.values[0])

# Calcular exactitud F1
func_best = toolbox.compile(expr=best)
aciertos = 0
for _, row in tabla.iterrows():
    pred = func_best(row.A, row.B, row.C)
    aciertos += int(pred == row.S)
accuracy = aciertos / len(tabla)
print(f"Exactitud F1: {accuracy:.3f}")

# Mostrar log en un DataFrame
log_df = pd.DataFrame(log)
log_df.tail()
```

```
gen     nevals  min     avg     std
0       200     3       3.995   0.212073
1       172     3       3.965   0.231894
```

```
2       167     3       3.93    0.339264
3       162     3       3.87    0.364829
4       170     3       3.855   0.392396
5       181     2       3.725   0.509289
6       172     2       3.555   0.571817
7       167     2       3.395   0.590741
8       169     2       3.3     0.632456
9       164     2       3.105   0.658768
10      176     2       3.035   0.695539
11      164     2       2.86    0.728286
12      176     2       2.77    0.732871
13      156     2       2.685   0.758798
14      163     2       2.58    0.695414
15      164     2       2.6     0.734847
16      158     2       2.64    0.741889
17      178     2       2.57    0.724638
18      165     2       2.435   0.660133
19      179     2       2.475   0.706665
20      166     2       2.525   0.699553
21      155     2       2.465   0.669907
22      163     2       2.535   0.72026
23      166     2       2.5     0.67082
24      175     2       2.545   0.691357
25      174     2       2.56    0.697424
26      171     2       2.605   0.747646
27      167     2       2.51    0.699929
28      164     2       2.415   0.642476
29      170     2       2.51    0.721041
30      153     2       2.435   0.675111
31      172     2       2.47    0.74773
32      171     2       2.425   0.659071
33      178     2       2.39    0.606548
34      175     2       2.48    0.713863
35      174     2       2.48    0.706824
36      164     2       2.46    0.662118
37      165     2       2.41    0.633956
38      182     2       2.405   0.633226
39      157     2       2.385   0.60562
40      162     2       2.425   0.627993
Mejor individuo F1: or_(and_(or_(and_(C, B), A), or_(C, B)), and_(or_(and_(A,
B), 0), B))
Altura: 4
Fitness (SSE): 2.0
Exactitud F1: 0.750
```

[6]:

	gen	nevals	min	avg	std
36	36	164	2.0	2.460	0.662118

37	37	165	2.0	2.410	0.633956
38	38	182	2.0	2.405	0.633226
39	39	157	2.0	2.385	0.605620
40	40	162	2.0	2.425	0.627993

1.2 4.2 Tabla de verdad con F2 = {and, or, not, xor}

[7]:
```
# Definir otro primitive set con XOR
pset2 = gp.PrimitiveSet("MAIN", 3)
pset2.renameArguments(ARG0='A')
pset2.renameArguments(ARG1='B')
pset2.renameArguments(ARG2='C')
pset2.addPrimitive(operator.and_, 2)
pset2.addPrimitive(operator.or_, 2)
pset2.addPrimitive(operator.not_, 1)
pset2.addPrimitive(operator.xor, 2)  # agregado
pset2.addTerminal(0)
pset2.addTerminal(1)

# Otro toolbox
creator.create("FitnessMin2", base.Fitness, weights=(-1.0,))
creator.create("Individual2", gp.PrimitiveTree, fitness=creator.FitnessMin2)

toolbox2 = base.Toolbox()
toolbox2.register("expr", gp.genFull, pset=pset2, min_=1, max_=3)
toolbox2.register("individual", tools.initIterate, creator.Individual2,␣
 ↪toolbox2.expr)
toolbox2.register("population", tools.initRepeat, list, toolbox2.individual)
toolbox2.register("compile", gp.compile, pset=pset2)

def evalParity2(individual):
    """
    Evaluar el individuo en la tabla de verdad usando toolbox2.
    """
    func = toolbox2.compile(expr=individual)
    errors = (
        (func(row.A, row.B, row.C) - row.S)**2
        for _, row in tabla.iterrows()
    )
    return math.fsum(errors),

toolbox2.register("evaluate", evalParity2) # Misma función de evaluación, pero␣
 ↪para evitar errores
toolbox2.register("select", tools.selTournament, tournsize=3)
toolbox2.register("mate", gp.cxOnePoint)
toolbox2.register("expr_mut", gp.genFull, min_=0, max_=2)
toolbox2.register("mutate", gp.mutUniform, expr=toolbox2.expr_mut, pset=pset2)
```

```
toolbox2.decorate("mate", gp.staticLimit(key=operator.attrgetter("height"),␣
 ↪max_value=5))
toolbox2.decorate("mutate", gp.staticLimit(key=operator.attrgetter("height"),␣
 ↪max_value=5))

stats2 = tools.Statistics(lambda ind: ind.fitness.values[0])
stats2.register("min", min)
stats2.register("avg", np.mean)
stats2.register("std", np.std)

hof2 = tools.HallOfFame(1)

pop2 = toolbox2.population(n=POP_SIZE)

pop2, log2 = algorithms.eaSimple(pop2, toolbox2, cxpb=CXPB, mutpb=MUTPB,␣
 ↪ngen=N_GEN,
                                 stats=stats2, halloffame=hof2, verbose=True)

best2 = hof2[0]
print("Mejor individuo F2:", best2)
print("Altura:", best2.height)
print("Fitness (SSE):", best2.fitness.values[0])

func_best2 = toolbox2.compile(expr=best2)
aciertos2 = sum(int(func_best2(r.A, r.B, r.C) == r.S) for _, r in tabla.
 ↪iterrows())
accuracy2 = aciertos2 / len(tabla)
print(f"Exactitud F2: {accuracy2:.3f}")

log2_df = pd.DataFrame(log2)
log2_df.tail()
```

```
gen     nevals  min     avg     std
0       200     2       4.035   0.551158
1       177     0       3.915   0.466664
2       171     0       3.89    0.712671
3       175     0       3.95    0.739932
4       181     0       3.835   0.926161
5       164     0       3.75    0.864581
6       169     0       3.755   0.935401
7       170     0       3.635   1.04965
8       174     0       3.485   1.0999
9       175     0       3.425   1.11552
10      173     0       3.24    1.29707
11      163     0       2.955   1.46047
12      172     0       2.89    1.59308
13      159     0       2.71    1.68104
```

```
14      160     0       2.435   1.72214
15      169     0       2.25    1.67854
16      162     0       2.325   1.75766
17      171     0       2.5     1.71756
18      164     0       2.555   1.70205
19      175     0       2.435   1.67803
20      159     0       2.15    1.73421
21      173     0       2.205   1.81741
22      170     0       2.355   1.83548
23      160     0       2       1.81108
24      169     0       2.175   1.82054
25      160     0       2.105   1.80941
26      159     0       1.93    1.848
27      162     0       1.98    1.85462
28      154     0       1.9     1.8412
29      168     0       1.955   1.82564
30      167     0       2.135   1.88063
31      162     0       1.95    1.84323
32      161     0       1.955   1.80637
33      168     0       2.07    1.85879
34      162     0       1.84    1.80677
35      163     0       1.92    1.7702
36      173     0       1.82    1.81041
37      168     0       2.15    1.84052
38      175     0       2.24    1.87947
39      176     0       1.825   1.83695
40      174     0       1.785   1.89968
Mejor individuo F2: not_(xor(xor(A, C), B))
Altura: 3
Fitness (SSE): 0.0
Exactitud F2: 1.000
```

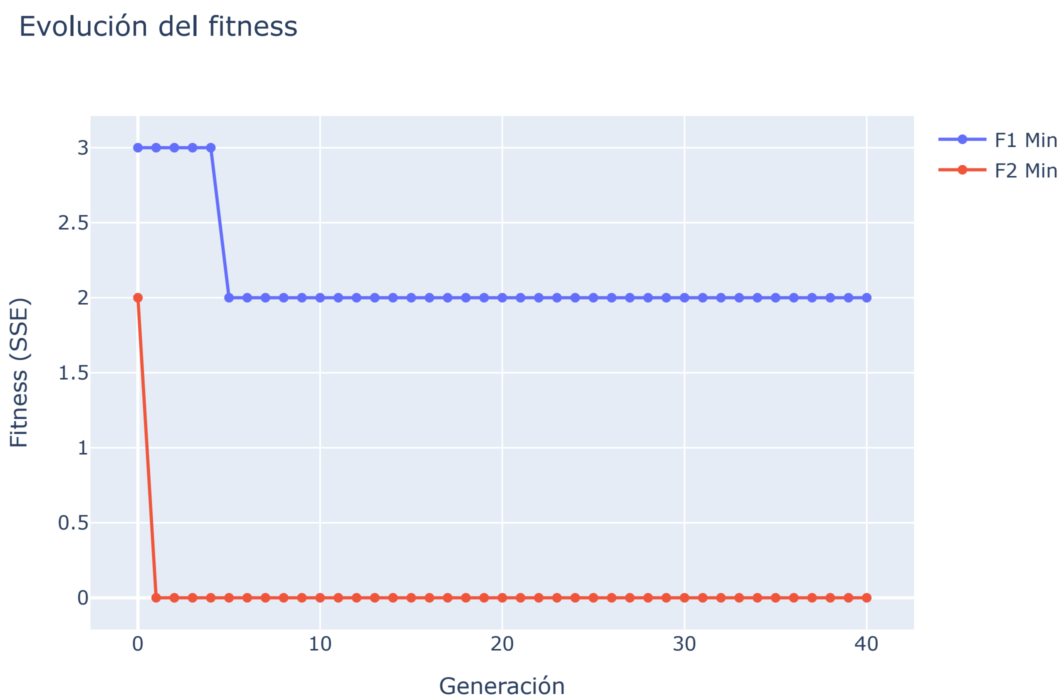
[7]:

	gen	nevals	min	avg	std
36	36	173	0.0	1.820	1.810414
37	37	168	0.0	2.150	1.840516
38	38	175	0.0	2.240	1.879468
39	39	176	0.0	1.825	1.836947
40	40	174	0.0	1.785	1.899678

[8]:

```python
# Plot fitness

fig = go.Figure()
fig.add_trace(go.Scatter(y=log_df['min'], mode='lines+markers', name='F1 Min'))
fig.add_trace(go.Scatter(y=log2_df['min'], mode='lines+markers', name='F2 Min'))
fig.update_layout(title='Evolución del fitness',
                  xaxis_title='Generación',
                  yaxis_title='Fitness (SSE)')
```

```
fig.show()
```



Comparando los entre el uso de F1 = {and, or, not} y F2 = {and, or, not, xor}, encontramos que el uso de F2 mejora significativamente la capacidad del modelo para ajustarse a la tabla de verdad dada.

Por lo que observamos que la elección del conjunto de funciones tiene un impacto crucial en la eficacia del modelo de programación genética.

1.3 Regresión simbólica con la tabla anexa

Adjunto al presente se incluye la tabla de valores. Para este caso el conjunto de funciones es F = {+, -, *, div, cos, sin, log, exp, abs, sqrt, x^y, …} sientanse en libertad de incluir o quitar las funciones según lo crean conveniente. A su vez, el conjunto terminal queda definido por T= {x, y, R, k's}. Donde x, y son los valores tomado de la tabla, R el conjunto de reales (recuerden que los pueden generar incluso con aleatorios), y k son constantes que ustedes consideren que pueden funcionar, (pi, e, …)

1.4 A

[9]:
```python
clase = pd.read_csv('data/Reg_Sybol_Class.csv')
clase.columns = ['x', 'f']
```

[10]:
```python
clase.head()
```

[10]:

	x	f
0	-10	4.85
1	-9	-3.20
2	-8	2.80
3	-7	1.30
4	-6	4.10

[11]:
```python
# Protected primitive functions

def protectedDiv(a, b):
    return a / b if abs(b) > 1e-9 else a

def protectedLog(a):
    a = float(a)
    return math.log(abs(a)) if abs(a) > 1e-9 else 0.0

def protectedSqrt(a):
    return math.sqrt(abs(a))

def protectedExp(a):
    a = max(min(a, 50), -50)
    return math.exp(a)

def protectedPow(a, b):
    try:
        if abs(a) > 1e6 or abs(b) > 8:
            return 1.0
        return float(pow(a, int(b)))
    except Exception:
        return 1.0
```

[12]:
```python
# Funciones y terminales
pset_sym = gp.PrimitiveSet("MAIN", 1)  # x
pset_sym.renameArguments(ARG0='x')

# Operadores aritméticos
pset_sym.addPrimitive(operator.add, 2)
pset_sym.addPrimitive(operator.sub, 2)
pset_sym.addPrimitive(operator.mul, 2)
pset_sym.addPrimitive(protectedDiv, 2)
```

```python
# Trig
pset_sym.addPrimitive(math.sin, 1)
pset_sym.addPrimitive(math.cos, 1)
pset_sym.addPrimitive(protectedLog, 1)
pset_sym.addPrimitive(protectedExp, 1)
pset_sym.addPrimitive(abs, 1)
pset_sym.addPrimitive(protectedSqrt, 1)
pset_sym.addPrimitive(protectedPow, 2)

# Constantes
pset_sym.addTerminal(math.pi, name="pi")
pset_sym.addTerminal(math.e, name="e")
pset_sym.addEphemeralConstant("rand", lambda: random.uniform(-10, 10))

#  Clases Fitness / Individuo
try:
    creator.FitnessMinSymb
except AttributeError:
    creator.create("FitnessMinSymb", base.Fitness, weights=(-1.0,))
try:
    creator.IndividualSymb
except AttributeError:
    creator.create("IndividualSymb", gp.PrimitiveTree, fitness=creator.
 ↪FitnessMinSymb)

#  Toolbox
toolbox_sym = base.Toolbox()
toolbox_sym.register("expr", gp.genHalfAndHalf, pset=pset_sym, min_=1, max_=3)
toolbox_sym.register("individual", tools.initIterate, creator.IndividualSymb,␣
 ↪toolbox_sym.expr)
toolbox_sym.register("population", tools.initRepeat, list, toolbox_sym.
 ↪individual)
toolbox_sym.register("compile", gp.compile, pset=pset_sym)
```

```
/Users/roicort/GitHub/PCIC/Genetics/GeneticProgramming/.venv/lib/python3.13/site
-packages/deap/gp.py:257: RuntimeWarning:

Ephemeral rand function cannot be pickled because its generating function is a
lambda function. Use functools.partial instead.
```

[13]:
```python
# Función de evaluación para regresión simbólica (ajuste a f(x))
def evalSymbolic(individual):
    func = toolbox_sym.compile(expr=individual)
    try:
        errors = ((func(row['x']) - row['f'])**2 for _, row in clase.iterrows())
        return (math.fsum(errors),)
```

```
    except (ValueError, OverflowError, ZeroDivisionError):
        return (1e10,)

toolbox_sym.register("evaluate", evalSymbolic)
toolbox_sym.register("select", tools.selTournament, tournsize=3)
toolbox_sym.register("mate", gp.cxOnePoint)
toolbox_sym.register("expr_mut", gp.genHalfAndHalf, min_=0, max_=2)
toolbox_sym.register("mutate", gp.mutUniform, expr=toolbox_sym.expr_mut,␣
 ↪pset=pset_sym)

# Limitar altura de los árboles
toolbox_sym.decorate("mate", gp.staticLimit(key=operator.attrgetter("height"),␣
 ↪max_value=10))
toolbox_sym.decorate("mutate", gp.staticLimit(key=operator.
 ↪attrgetter("height"), max_value=10))
```

[14]:
```
# Ejecutar evolución

POP_SIZE = 200
N_GEN = 40
CXPB = 0.8
MUTPB = 0.2
random.seed(42)

# Estadísticas multi: fitness y tamaño
hof = tools.HallOfFame(1)
stats_fit = tools.Statistics(lambda ind: ind.fitness.values)
stats_size = tools.Statistics(len)
mstats = tools.MultiStatistics(fitness=stats_fit, size=stats_size)
mstats.register("avg", np.mean)
mstats.register("std", np.std)
mstats.register("min", min)
mstats.register("max", max)

pop = toolbox_sym.population(n=POP_SIZE)

pop, log = algorithms.eaSimple(pop, toolbox_sym, cxpb=CXPB, mutpb=MUTPB,␣
 ↪ngen=N_GEN,
                               stats=mstats, halloffame=hof, verbose=True)

best = hof[0]
print("Mejor individuo:", best)
print("Altura:", best.height)
print("Error cuadrático (SSE):", best.fitness.values[0])

log_df = pd.DataFrame(log)
log_df.tail()
```

```
                                                                      fitness
size
                ----------------------------------------------------------------
-----------------------------------
-----------------------------------------------
gen     nevals  avg             gen     max                             min
nevals  std             avg     gen     max     min     nevals  std
0       200     2.15049e+42     0       (2.419305433663835e+44,)
(469.76118749420846,)   200     2.15653e+43     4.42    0       14      2
200     2.59299
1       174     2.2187e+07      1       (4426337606.207691,)
(469.76118749420846,)   174     3.12203e+08     3.96    1       14      1
174     2.2646
2       163     2.16675e+67     2       (4.333504521772235e+69,)
(469.76118749420846,)   163     3.05658e+68     3.825   2       12      1
163     2.32258
3       166     5.0609e+33      3       (1.0121805429499375e+36,)
(457.3328187901389,)    166     7.13928e+34     3.505   3       12      1
166     2.28691
4       172     9.712e+24       4       (1.942400830668118e+27,)
(469.7541406840884,)    172     1.37005e+26     2.83    4       10      1
172     1.99527
5       176     6.44278e+08     5       (128855239976.56868,)
(469.7541406840884,)    176     9.08863e+09     2.37    5       10      1
176     1.97816
6       160     53706.2         6       (10611320.572601173,)
(469.7541406840884,)    160     748410          2.345   6       11      1
160     2.29259
7       172     1.28038e+12     7       (256075577304656.7,)
(436.96374740111776,)   172     1.8062e+13      2.77    7       13      1
172     2.80662
8       164     1.29618e+20     8       (2.5923527643295616e+22,)
(392.3913446304858,)    164     1.82848e+21     3.73    8       15      1
164     3.33723
9       167     2.07397e+20     9       (2.5923527643295906e+22,)
(392.3913446304858,)    167     2.12769e+21     4.51    9       14      1
167     3.70674
10      170     2.11164e+20     10      (3.6292938699116357e+22,)
(392.3913446304858,)    170     2.59185e+21     4.955   10      15      1
170     3.76603
11      163     1.32554e+08     11      (25646942017.559666,)
(388.3917173816135,)    163     1.80904e+09     5.535   11      17      1
163     4.04213
12      173     2.59244e+20     12      (3.6292938699116357e+22,)
(388.3917173816135,)    173     2.78004e+21     5.535   12      18      1
173     4.32074
13      161     2.8025e+15      13      (5.6050066398316525e+17,)
(388.3917173816135,)    161     3.95342e+16     5.355   13      20      1
```

```
161     4.25899
14      166     3.13157e+20     14      (2.5923527643296325e+22,)
(372.52546562052373,)   166     2.42437e+21     7.44    14      24      1
166     5.42645
15      169     6.80028e+20     15      (3.110826499541018e+22,)
(372.52546562052373,)   169     3.97155e+21     7.9     15      23      1
169     5.72538
16      166     1.11471e+21     16      (1.0887881608963576e+23,)
(372.52546562052373,)   166     1.08369e+22     8.245   16      21      1
166     5.82795
17      174     7.18303e+19     17      (1.4366057957452168e+22,)
(343.91650902831276,)   174     1.01329e+21     9.155   17      26      1
174     6.09762
18      187     1.29619e+21     18      (1.0887881608963576e+23,)
(372.52546562052373,)   187     8.97649e+21     10.165  18      26      1
187     5.99148
19      164     8.97222e+25     19      (1.7944403569865414e+28,)
(372.52546562052373,)   164     1.26568e+27     11.035  19      27      1
164     6.40029
20      169     8.08822e+20     20      (1.0887881608963576e+23,)
(372.52546562052373,)   169     7.87077e+21     11.84   20      37      1
169     7.00103
21      169     1.43958e+21     21      (1.0887881608963576e+23,)
(333.4160785839512,)    169     1.12939e+22     12.225  21      35      1
169     6.86326
22      168     8.42815e+41     22      (1.6856297040422289e+44,)
(288.5191357493558,)    168     1.18894e+43     12.66   22      35      1
168     7.3215
23      173     8.25888e+20     23      (1.0887881608963576e+23,)
(341.6589472939765,)    173     7.84555e+21     13.195  23      35      1
173     8.24724
24      168     1.09279e+21     24      (5.703176080555078e+22,)
(346.4745686895749,)    168     6.29952e+21     13.075  24      36      1
168     7.25392
25      175     5.18482e+19     25      (1.0369635879309074e+22,)
(346.4745686895749,)    175     7.31409e+20     13.115  25      34      1
175     7.02864
26      183     6.37455e+20     26      (3.629412779132408e+22,)
(339.03516597423845,)   183     4.32498e+21     13.5    26      32      1
183     6.99285
27      171     5.61274e+20     27      (3.784283899686066e+22,)
(339.03516597423845,)   171     3.72795e+21     14.06   27      32      1
171     7.25923
28      172     8.45666e+20     28      (1.0887881608963576e+23,)
(339.03516597423845,)   172     8.24711e+21     14.295  28      31      1
172     6.71029
```

```
/var/folders/4w/72wtbysj7kb189x1nn694p9h0000gn/T/ipykernel_30289/1937063526.py:2
```

```
1: RuntimeWarning:

divide by zero encountered in scalar power

/Users/roicort/GitHub/PCIC/Genetics/GeneticProgramming/.venv/lib/python3.13/site
-packages/numpy/_core/_methods.py:190: RuntimeWarning:

invalid value encountered in subtract


29      169     4.66623e+20     29      (5.703176080712048e+22,)
(339.03516597423845,)   169     4.31267e+21     14.465  29      37      1
169     7.09075
30      165     1.81465e+20     30      (2.073885481050632e+22,)
(311.4330539799348,)    165     1.82407e+21     15.08   30      32      1
165     6.05505
31      182     3.37015e+20     31      (4.666234975588674e+22,)
(313.9954633510754,)    182     3.481e+21       14.39   31      28      1
182     6.36065
32      164     inf             32      (inf,)
(313.9954633510754,)    164     nan             15.695  32      34      1
164     6.67248
33      167     5.44394e+20     33      (1.0887881608963576e+23,)
(307.0280742129721,)    167     7.67962e+21     15.625  33      35      1
167     7.35285
34      178     7.79286e+20     34      (1.0887881608963576e+23,)
(307.34712836032475,)   178     7.89273e+21     16.195  34      37      1
178     7.18101
35      154     inf             35      (inf,)
(307.34712836032475,)   154     nan             16.29   35      37      1
154     7.13133
36      174     1.41078e+12     36      (282155223277796.5,)
(272.9196644907471,)    174     1.99014e+13     17.525  36      37      1
174     6.81905
37      183     1.57315e+20     37      (2.109358588714731e+22,)
(272.9196644907471,)    183     1.65456e+21     17.205  37      37      2
183     7.48084
38      167     1.29618e+20     38      (2.592354076563038e+22,)
(272.9196644907471,)    167     1.82848e+21     17.74   38      37      2
167     8.04751
39      172     5.18471e+19     39      (1.0369411253118563e+22,)
(272.9196644907471,)    172     7.31393e+20     17.575  39      47      2
172     8.32553
40      178     1.03694e+20     40      (1.5554116582060163e+22,)
(239.7794277775339,)    178     1.15469e+21     16.775  40      45      2
178     7.85267
Mejor individuo: protectedSqrt(protectedExp(sub(e, sub(sub(abs(x),
sub(abs(protectedSqrt(sub(x, pi))), protectedSqrt(sub(cos(protectedDiv(pi, e)),
```

```
sin(sin(pi)))))), protectedSqrt(sub(x,
protectedSqrt(protectedSqrt(7.525595035783958)))))))
Altura: 10
Error cuadrático (SSE): 239.7794277775339
```

[14]:

	gen	nevals
36	36	174
37	37	183
38	38	167
39	39	172
40	40	178

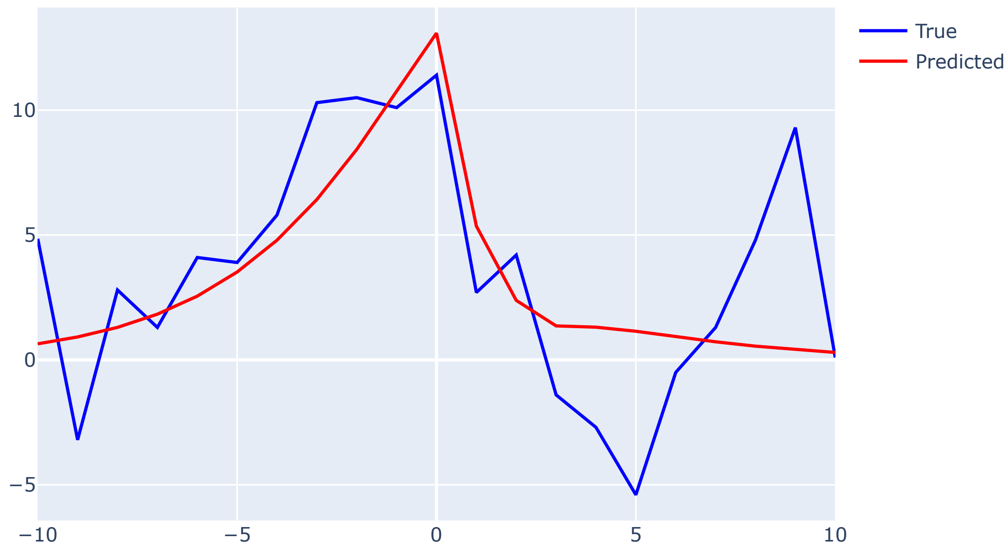
[15]:

```
log_df.columns
```

[15]: Index(['gen', 'nevals'], dtype='object')

[16]:

```
# Plot

x_vals = clase['x']
y_true = clase['f']
func_best = toolbox_sym.compile(expr=best)
y_pred = [func_best(x) for x in x_vals]

fig = go.Figure()
fig.add_trace(go.Scatter(x=x_vals, y=y_true, mode='lines', name='True',␣
↪line=dict(color='blue')))
fig.add_trace(go.Scatter(x=x_vals, y=y_pred, mode='lines', name='Predicted',␣
↪line=dict(color='red')))
fig.show()
```



1.5 B

[17]:
```
symreg = pd.read_csv('data/symbolic_regression.csv')
symreg.columns = ['x', 'y', 'f']
symreg.head()
```

[17]:

	x	y	f
0	-100	-100	19.059929
1	-100	-90	86.073703
2	-100	-80	104.763090
3	-100	-70	78.242133
4	-100	-60	22.679452

[18]:
```
# Conjunto de funciones y terminales

pset_sym2 = gp.PrimitiveSet("MAIN", 2)  # x, y
pset_sym2.renameArguments(ARG0='x')
pset_sym2.renameArguments(ARG1='y')

# Operadores aritméticos
pset_sym2.addPrimitive(operator.add, 2)
pset_sym2.addPrimitive(operator.sub, 2)
```

```python
pset_sym2.addPrimitive(operator.mul, 2)
pset_sym2.addPrimitive(protectedDiv, 2)
pset_sym2.addPrimitive(protectedPow, 2)
# Trig
pset_sym2.addPrimitive(math.sin, 1)
pset_sym2.addPrimitive(math.cos, 1)
pset_sym2.addPrimitive(protectedLog, 1)
pset_sym2.addPrimitive(math.exp, 1)
pset_sym2.addPrimitive(math.exp2, 1)
#pset_sym2.addPrimitive(abs, 1)
pset_sym2.addPrimitive(protectedSqrt, 1)
# Constantes
pset_sym2.addTerminal(math.pi, name="pi")
pset_sym2.addTerminal(math.e, name="e")
pset_sym2.addEphemeralConstant("rand", functools.partial(random.uniform, -100,␣
↪100))

# Clases Fitness
try:
    creator.FitnessMinSymb2
except AttributeError:
    creator.create("FitnessMinSymb2", base.Fitness, weights=(-1.0,))
try:
    creator.IndividualSymb2
except AttributeError:
    creator.create("IndividualSymb2", gp.PrimitiveTree, fitness=creator.
↪FitnessMinSymb2)

# Toolbox
toolbox_sym2 = base.Toolbox()
toolbox_sym2.register("expr", gp.genHalfAndHalf, pset=pset_sym2, min_=1, max_=3)
toolbox_sym2.register("individual", tools.initIterate, creator.IndividualSymb2,␣
↪toolbox_sym2.expr)
toolbox_sym2.register("population", tools.initRepeat, list, toolbox_sym2.
↪individual)
toolbox_sym2.register("compile", gp.compile, pset=pset_sym2)

# Función de evaluación para regresión simbólica (ajuste a f(x,y))
def evalSymbolic2(individual):
    func = toolbox_sym2.compile(expr=individual)
    try:
        errors = ((func(row['x'], row['y']) - row['f'])**2 for _, row in symreg.
↪iterrows())
        return (math.fsum(errors),)
    except (ValueError, OverflowError, ZeroDivisionError):
        # Penalización alta si ocurre error matemático
        return (1e10,)
```

```
toolbox_sym2.register("evaluate", evalSymbolic2)
toolbox_sym2.register("select", tools.selTournament, tournsize=3)
toolbox_sym2.register("mate", gp.cxOnePoint)
toolbox_sym2.register("expr_mut", gp.genHalfAndHalf, min_=0, max_=2)
toolbox_sym2.register("mutate", gp.mutUniform, expr=toolbox_sym2.expr_mut,␣
↪pset=pset_sym2)
toolbox_sym2.decorate("mate", gp.staticLimit(key=operator.attrgetter("height"),␣
↪max_value=3))
toolbox_sym2.decorate("mutate", gp.staticLimit(key=operator.
↪attrgetter("height"), max_value=3))
```

[19]:
```
# Ejecutar evolución (multivariable) con estadísticas de fitness y tamaño
POP_SIZE = 500
N_GEN = 100
CXPB = 0.8
MUTPB = 0.2
random.seed(42)

hof = tools.HallOfFame(3)

stats_fit2 = tools.Statistics(lambda ind: ind.fitness.values)
stats_size2 = tools.Statistics(len)
mstats2 = tools.MultiStatistics(fitness=stats_fit2, size=stats_size2)
mstats2.register("avg", np.mean)
mstats2.register("std", np.std)
mstats2.register("min", min)
mstats2.register("max", max)

pop2 = toolbox_sym2.population(n=POP_SIZE)

pop2, log2 = algorithms.eaSimple(pop2, toolbox_sym2, cxpb=CXPB, mutpb=MUTPB,␣
↪ngen=N_GEN,
                                 stats=mstats2, halloffame=hof, verbose=True)

best2 = hof[0]
print("Mejor individuo multivariable:", best2)
print("Altura:", best2.height)
print("Longitud (nodos):", len(best2))
print("Error cuadrático (SSE):", best2.fitness.values[0])

log2_df = pd.DataFrame(log2)
log2_df.tail()
```

```
/var/folders/4w/72wtbysj7kb189x1nn694p9h0000gn/T/ipykernel_30289/2475699097.py:4
7: RuntimeWarning:
```

```
overflow encountered in scalar power

/Users/roicort/GitHub/PCIC/Genetics/GeneticProgramming/.venv/lib/python3.13/site
-packages/numpy/_core/_methods.py:190: RuntimeWarning:

invalid value encountered in subtract


                                          fitness
size
                ----------------------------------------------------------
-----------------------------------------------
gen     nevals  avg     gen     max     min                     nevals  std
avg     gen     max     min     nevals  std
0       500     inf     0       (inf,)  (2043019.3055838281,)   500     nan
4.31    0       14      2       500     2.50557
1       439     1.3668e+198     1       (6.833999036102657e+200,)
(2053044.3320912262,)   439     inf     3.746   1       12      1       439
2.06627

/Users/roicort/GitHub/PCIC/Genetics/GeneticProgramming/.venv/lib/python3.13/site
-packages/numpy/_core/_methods.py:193: RuntimeWarning:

overflow encountered in multiply

/var/folders/4w/72wtbysj7kb189x1nn694p9h0000gn/T/ipykernel_30289/1937063526.py:2
1: RuntimeWarning:

divide by zero encountered in scalar power


2       427     inf             2       (inf,)
(2053044.3320912262,)   427     nan     3.522   2       12      1       427
1.98331
3       419     inf             3       (inf,)
(2051330.2241006324,)   419     nan     3.28    3       10      1       419
1.75659
4       416     inf             4       (inf,)
(2046568.8905938168,)   416     nan     3.212   4       10      1       416
1.65259
5       422     1.21396e+86     5       (1.5174544944341143e+88,)
(2045075.67355705,)     422     1.35181e+87     3.218   5       9       1
422     1.54223
6       415     2.19302e+172    6       (1.09650863485047e+175,)
(2045075.67355705,)     415     inf             3.284   6       8       1
415     1.61967
7       421     3.03491e+85     7       (1.5174544944341143e+88,)
(2045075.67355705,)     421     6.77947e+86     3.258   7       10      1
421     1.67315
```

```
8       402     5.16451e+117    8       (2.582254803344729e+120,)
(2042534.743525733,)    402     1.15366e+119    3.542   8       7       1
402     1.61871
9       436     4.09533e+169    9       (2.0476670804066035e+172,)
(2040248.436318405,)    436     inf             3.832   9       8       1
436     1.62227
10      427     inf             10      (inf,)
(2029169.6511734857,)   427     nan             4.102   10      9       1
427     1.63695
11      429     7.07702e+64     11      (3.5384946568299716e+67,)
(2029169.6511734857,)   429     1.58088e+66     4.208   11      11      1
429     1.66155
12      431     inf             12      (inf,)
(2029169.6511734857,)   431     nan             4.24    12      8       1
431     1.59198
13      422     1.16682e+188    13      (2.9170499939859413e+190,)
(2029169.6511734857,)   422     inf             4.21    13      8       1
422     1.56138
14      418     inf             14      (inf,)
(2029169.6511734857,)   418     nan             4.418   14      8       1
418     1.4803
15      423     inf             15      (inf,)
(2029169.6511734857,)   423     nan             4.37    15      8       1
423     1.50103
16      413     inf             16      (inf,)
(2029169.6511734857,)   413     nan             4.452   16      8       1
413     1.54263
17      419     inf             17      (inf,)
(2029169.6511734857,)   419     nan             4.564   17      8       1
419     1.60309
18      423     inf             18      (inf,)
(2029169.6511734857,)   423     nan             4.596   18      8       1
423     1.64462
19      428     inf             19      (inf,)
(2029169.6511734857,)   428     nan             4.476   19      8       1
428     1.67971
20      413     1.56535e+93     20      (7.826752194650517e+95,)
(2029169.6511734857,)   413     3.49673e+94     4.42    20      7       1
413     1.72962
21      433     5.16451e+117    21      (2.582254803344729e+120,)
(2029169.6511734857,)   433     1.15366e+119    4.32    21      8       1
433     1.70106
22      432     inf             22      (inf,)
(2029169.6511734857,)   432     nan             4.228   22      8       1
432     1.60749
23      415     1.34983e+59     23      (3.3745731111875846e+61,)
(2029169.6511734857,)   415     2.12999e+60     4.006   23      8       1
415     1.39927
```

```
24      430     3.22016e+57     24      (1.610081198606933e+60,)
(2029169.6511734857,)   430     7.1933e+58      3.768   24      8       1
430     1.22889
25      444     9.50078e+116    25      (2.375194235089355e+119,)
(2029169.6511734857,)   444     1.4992e+118     3.626   25      8       1
444     1.05741
26      417     2.07844e+30     26      (2.664668475787619e+31,)
(2029169.6511734857,)   417     7.14588e+30     3.538   26      7       1
417     0.978037
27      441     3.03491e+93     27      (1.517454493358496e+96,)
(2029169.6511734857,)   441     6.77947e+94     3.472   27      8       1
441     1.00061
28      404     2.89651e+99     28      (1.4482564184609178e+102,)
(2029169.6511734857,)   404     6.47032e+100    3.458   28      8       1
404     0.992087
29      422     2.72714e+166    29      (1.3635675960644387e+169,)
(2029169.6511734857,)   422     inf             3.416   29      8       1
422     0.995462
30      423     1.56535e+93     30      (7.826752194650517e+95,)
(2029169.6511734857,)   423     3.49673e+94     3.466   30      7       1
423     0.863043
31      412     1.18584e+86     31      (5.929198901016902e+88,)
(2029169.6511734857,)   412     2.64897e+87     3.498   31      7       1
412     0.930589
32      426     7.66877e+41     32      (3.8343862035421485e+44,)
(2029169.6511734857,)   426     1.71307e+43     3.472   32      8       1
426     0.92586
33      435     9.10473e+85     33      (1.5174544944341143e+88,)
(2029169.6511734857,)   435     1.17188e+87     3.538   33      8       1
435     0.953182
34      429     2.25812e+42     34      (5.6453022941415275e+44,)
(2029169.6511734857,)   429     3.56325e+43     3.444   34      8       1
429     0.985324
35      430     3.03491e+85     35      (1.5174544944341143e+88,)
(2029169.6511734857,)   430     6.77947e+86     3.48    35      8       1
430     1.0245
36      416     6.74915e+58     36      (3.3745731111875846e+61,)
(2029169.6511734857,)   416     1.50765e+60     3.444   36      5       1
416     0.831182
37      412     6.09705e+202    37      (3.048525907707829e+205,)
(2029169.6511734857,)   412     inf             3.5     37      8       1
412     0.949737
38      413     1.56535e+93     38      (7.826752194650517e+95,)
(2029169.6511734857,)   413     3.49673e+94     3.486   38      7       1
413     0.932633
39      416     2.25812e+42     39      (5.6453022941415275e+44,)
(2029169.6511734857,)   416     3.56325e+43     3.518   39      7       1
416     0.89982
```

```
40      415     4.75039e+116    40      (2.375194235089355e+119,)
(2029169.6511734857,)   415     1.06116e+118    3.552   40      8       1
415     0.855158
41      408     3.03491e+85     41      (1.5174544944341143e+88,)
(2029169.6511734857,)   408     6.77947e+86     3.476   41      8       1
408     0.890744
42      428     3.03491e+85     42      (1.5174544944341143e+88,)
(2029169.6511734857,)   428     6.77947e+86     3.42    42      7       1
428     0.958958
43      421     1.6282e+88      43      (8.125842194080243e+90,)
(2029169.6511734857,)   421     3.63034e+89     3.448   43      8       1
421     0.920487
44      402     4.19866e+86     44      (2.099328734024008e+89,)
(2029169.6511734857,)   402     9.37909e+87     3.57    44      8       1
402     0.895042
45      422     3.1307e+93      45      (7.826752194650517e+95,)
(2029169.6511734857,)   422     4.94016e+94     3.542   45      7       1
422     0.832007
46      406     2.95499e+30     46      (1.7180556341932396e+32,)
(2029169.6511734857,)   406     1.09493e+31     3.544   46      8       1
406     0.836698
47      411     1.56535e+93     47      (7.826752194650517e+95,)
(2029169.6511734857,)   411     3.49673e+94     3.588   47      8       1
411     0.868479
48      416     5.3729e+234     48      (2.6864512639924023e+237,)
(2029169.6511734857,)   416     inf             3.5     48      8       1
416     0.861394
49      413     3.03494e+85     49      (1.5174544944341143e+88,)
(2029169.6511734857,)   413     6.77947e+86     3.608   49      8       1
413     0.81874
50      436     2.61138e+30     50      (2.664668475787619e+31,)
(2029169.6511734857,)   436     7.92245e+30     3.49    50      7       1
436     0.915369
51      422     1.56535e+93     51      (7.826752194650517e+95,)
(2029169.6511734857,)   422     3.49673e+94     3.56    51      7       1
422     0.915642
52      417     3.03491e+85     52      (1.5174544944341143e+88,)
(2029169.6511734857,)   417     6.77947e+86     3.49    52      8       1
417     0.939095
53      407     3.03491e+85     53      (1.5174544944341143e+88,)
(2029169.6511734857,)   407     6.77947e+86     3.468   53      7       1
407     0.930041
54      415     6.74915e+58     54      (3.3745731111875846e+61,)
(2029169.6511734857,)   415     1.50765e+60     3.506   54      7       1
415     0.89998
55      425     2.23832e+30     55      (2.664668475787619e+31,)
(2029169.6511734857,)   425     7.39147e+30     3.492   55      6       1
425     0.825794
```

```
56      427     2.18507e+30     56      (2.664668475787619e+31,)
(2029169.6511734857,)   427     7.3109e+30      3.56    56      8       1
427     0.93723
57      416     3.03491e+85     57      (1.5174544944341143e+88,)
(2029169.6511734857,)   416     6.77947e+86     3.54    57      8       1
416     0.872009
58      411     3.03491e+85     58      (1.5174544944341143e+88,)
(2029169.6511734857,)   411     6.77947e+86     3.572   58      7       1
411     0.837148
59      419     7.39318e+58     59      (3.3745731111875846e+61,)
(2029169.6511734857,)   419     1.51078e+60     3.492   59      8       1
419     0.90881
60      408     2.82935e+80     60      (1.4146727922853661e+83,)
(2029169.6511734857,)   408     6.32028e+81     3.544   60      7       1
408     0.92955
61      423     8.64895e+30     61      (2.736841198851839e+33,)
(2029169.6511734857,)   423     1.22648e+32     3.498   61      7       1
423     0.87977
62      418     1.04429e+171    62      (5.221469711288646e+173,)
(2029169.6511734857,)   418     inf             3.542   62      8       1
418     0.885571
63      412     3.03491e+85     63      (1.5174544944341143e+88,)
(2029169.6511734857,)   412     6.77947e+86     3.532   63      7       1
412     0.846744
64      437     2.20391e+30     64      (2.664668475787619e+31,)
(2029169.6511734857,)   437     7.31369e+30     3.5     64      8       1
437     0.866025
65      427     2.25911e+42     65      (5.6453022941415275e+44,)
(2029169.6511734857,)   427     3.56325e+43     3.496   65      7       1
427     0.904425
66      434     1.56535e+93     66      (7.826752194650517e+95,)
(2029169.6511734857,)   434     3.49673e+94     3.464   66      7       1
434     0.88809
67      403     6.06982e+85     67      (1.5174544944341143e+88,)
(2029169.6511734857,)   403     9.57801e+86     3.424   67      8       1
403     0.992081
68      423     6.06982e+85     68      (1.5174544944341143e+88,)
(2029169.6511734857,)   423     9.57801e+86     3.448   68      8       1
423     0.90515
69      421     6.06982e+85     69      (1.5174544944341143e+88,)
(2029169.6511734857,)   421     9.57801e+86     3.542   69      8       1
421     0.894559
70      418     2.82935e+80     70      (1.4146727922853661e+83,)
(2029169.6511734857,)   418     6.32028e+81     3.512   70      8       1
418     0.904354
71      409     1.81197e+30     71      (2.664668475787619e+31,)
(2029169.6511734857,)   409     6.70819e+30     3.428   71      7       1
409     0.910393
```

```
72      422     6.74915e+58     72      (3.3745731111875846e+61,)
(2029169.6511734857,)   422     1.50765e+60     3.51    72      7       1
422     0.943345
73      407     1.34983e+59     73      (3.3745731111875846e+61,)
(2029169.6511734857,)   407     2.12999e+60     3.386   73      7       1
407     0.919241
74      415     1.56535e+93     74      (7.826752194650517e+95,)
(2029169.6511734857,)   415     3.49673e+94     3.514   74      7       1
415     0.890957
75      427     3.03491e+85     75      (1.5174544944341143e+88,)
(2029169.6511734857,)   427     6.77947e+86     3.516   75      8       1
427     0.875068
76      428     inf             76      (inf,)
(2029169.6511734857,)   428     nan             3.516   76      7       1
428     0.851906
77      425     6.74915e+58     77      (3.3745731111875846e+61,)
(2029169.6511734857,)   425     1.50765e+60     3.578   77      7       1
425     0.843751
78      408     1.12906e+42     78      (5.6453022941415275e+44,)
(2029169.6511734857,)   408     2.52213e+43     3.508   78      7       1
408     0.932704
79      428     2.13174e+30     79      (2.664668475787619e+31,)
(2029169.6511734857,)   428     7.22906e+30     3.498   79      6       1
428     0.849703
80      413     3.03491e+85     80      (1.5174544944341143e+88,)
(2029169.6511734857,)   413     6.77947e+86     3.514   80      7       1
413     0.86128
81      418     1.56535e+93     81      (7.826752194650517e+95,)
(2029169.6511734857,)   418     3.49673e+94     3.476   81      6       1
418     0.851718
82      407     6.74915e+58     82      (3.3745731111875846e+61,)
(2029169.6511734857,)   407     1.50765e+60     3.42    82      6       1
407     0.911921
83      423     5.91413e+43     83      (2.957063106455085e+46,)
(2029169.6511734857,)   423     1.32112e+45     3.512   83      8       1
423     0.865942
84      420     1.34983e+59     84      (3.3745731111875846e+61,)
(2029169.6511734857,)   420     2.12999e+60     3.486   84      8       1
420     0.851941
85      431     2.82935e+80     85      (1.4146727922853661e+83,)
(2029169.6511734857,)   431     6.32028e+81     3.482   85      8       1
431     0.886384
86      424     3.03491e+85     86      (1.5174544944341143e+88,)
(2029169.6511734857,)   424     6.77947e+86     3.484   86      7       1
424     0.865878
87      414     1.12906e+42     87      (5.6453022941415275e+44,)
(2029169.6511734857,)   414     2.52213e+43     3.582   87      8       1
414     0.929126
```

```
88      421     5.3729e+234     88      (2.6864512639924023e+237,)
(2029169.6511734857,)   421     inf             3.554   88      8       1
421     0.954507
89      423     6.74915e+58     89      (3.3745731111875846e+61,)
(2029169.6511734857,)   423     1.50765e+60     3.464   89      8       1
423     0.949054
90      433     6.06982e+85     90      (1.5174544944341143e+88,)
(2029169.6511734857,)   433     9.57801e+86     3.49    90      7       1
433     0.9219
91      436     6.74915e+58     91      (3.3745731111875846e+61,)
(2029169.6511734857,)   436     1.50765e+60     3.456   91      7       1
436     0.871816
92      408     1.56535e+93     92      (7.826752194650517e+95,)
(2029169.6511734857,)   408     3.49673e+94     3.414   92      6       1
408     0.954256
93      420     3.03491e+85     93      (1.5174544944341143e+88,)
(2029169.6511734857,)   420     6.77947e+86     3.44    93      8       1
420     0.9992
94      417     3.04812e+85     94      (1.5174544944341143e+88,)
(2029169.6511734857,)   417     6.77948e+86     3.44    94      7       1
417     0.863944
95      428     2.25812e+42     95      (5.6453022941415275e+44,)
(2029169.6511734857,)   428     3.56325e+43     3.532   95      8       1
428     0.892735
96      435     1.56535e+93     96      (7.826752194650517e+95,)
(2029169.6511734857,)   435     3.49673e+94     3.53    96      7       1
435     0.844452
97      406     1.38203e+59     97      (3.3745731111875846e+61,)
(2029169.6511734857,)   406     2.131e+60       3.53    97      7       1
406     0.90835
98      424     8.02618e+85     98      (4.0130887578227084e+88,)
(2029169.6511734857,)   424     1.79291e+87     3.482   98      7       1
424     0.828056
99      423     3.03491e+85     99      (1.5174544944341143e+88,)
(2029169.6511734857,)   423     6.77947e+86     3.47    99      7       1
423     0.897274
100     420     6.06982e+85     100     (1.5174544944341143e+88,)
(2029169.6511734857,)   420     9.57801e+86     3.428   100     7       1
420     0.949113
Mejor individuo multivariable: protectedSqrt(exp2(protectedSqrt(y)))
Altura: 3
Longitud (nodos): 4
Error cuadrático (SSE): 2029169.6511734857
```

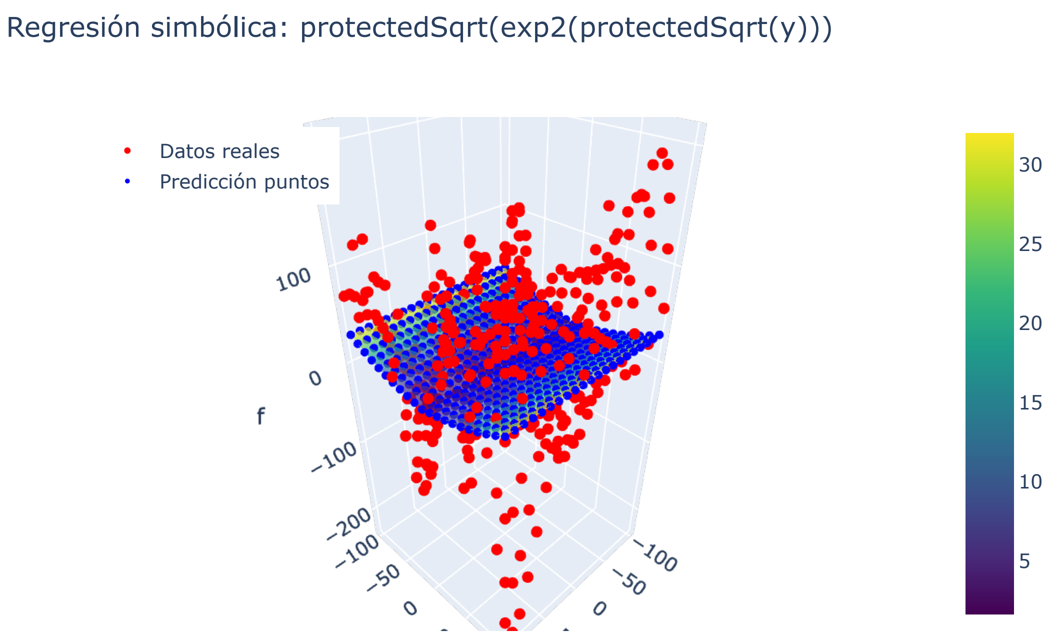
[19]:

	gen	nevals
96	96	435
97	97	406

```
98     98       424
99     99       423
100   100       420
```

[20]:
```python
# Plot 3D

func_best2 = toolbox_sym2.compile(expr=best2)

# Puntos reales
x_data = symreg['x'].values
y_data = symreg['y'].values
z_true = symreg['f'].values
z_pred_pts = [func_best2(xd, yd) for xd, yd in zip(x_data, y_data)]

# Crear una malla para superficie
nx, ny = 50, 50
x_lin = np.linspace(x_data.min(), x_data.max(), nx)
y_lin = np.linspace(y_data.min(), y_data.max(), ny)
Xg, Yg = np.meshgrid(x_lin, y_lin)
Zg = np.zeros_like(Xg, dtype=float)
for i in range(Xg.shape[0]):
    for j in range(Xg.shape[1]):
        try:
            Zg[i, j] = func_best2(Xg[i, j], Yg[i, j])
        except Exception:
            Zg[i, j] = np.nan

fig3d = go.Figure()

# Superficie del modelo
fig3d.add_trace(go.Surface(x=Xg, y=Yg, z=Zg, colorscale='Viridis', opacity=0.7,␣
 ↪name='Modelo'))

# Puntos reales de la tabla
fig3d.add_trace(go.Scatter3d(x=x_data, y=y_data, z=z_true,
                             mode='markers', name='Datos reales',
                             marker=dict(size=4, color='red')))

# Puntos predichos
fig3d.add_trace(go.Scatter3d(x=x_data, y=y_data, z=z_pred_pts,
                             mode='markers', name='Predicción puntos',
                             marker=dict(size=3, color='blue', symbol='circle')))

fig3d.update_layout(title=f'Regresión simbólica: {str(best2)}',
                    scene=dict(xaxis_title='x', yaxis_title='y',␣
 ↪zaxis_title='f'),
                    legend=dict(x=0.02, y=0.98))
```

```python
fig3d.show()

# Squared Sum Error (SSE) en los puntos originales
sse_pts = float(sum((zp - zt)**2 for zp, zt in zip(z_pred_pts, z_true)))
print('SSE en puntos originales:', sse_pts)
```



SSE en puntos originales: 2029169.6511734866