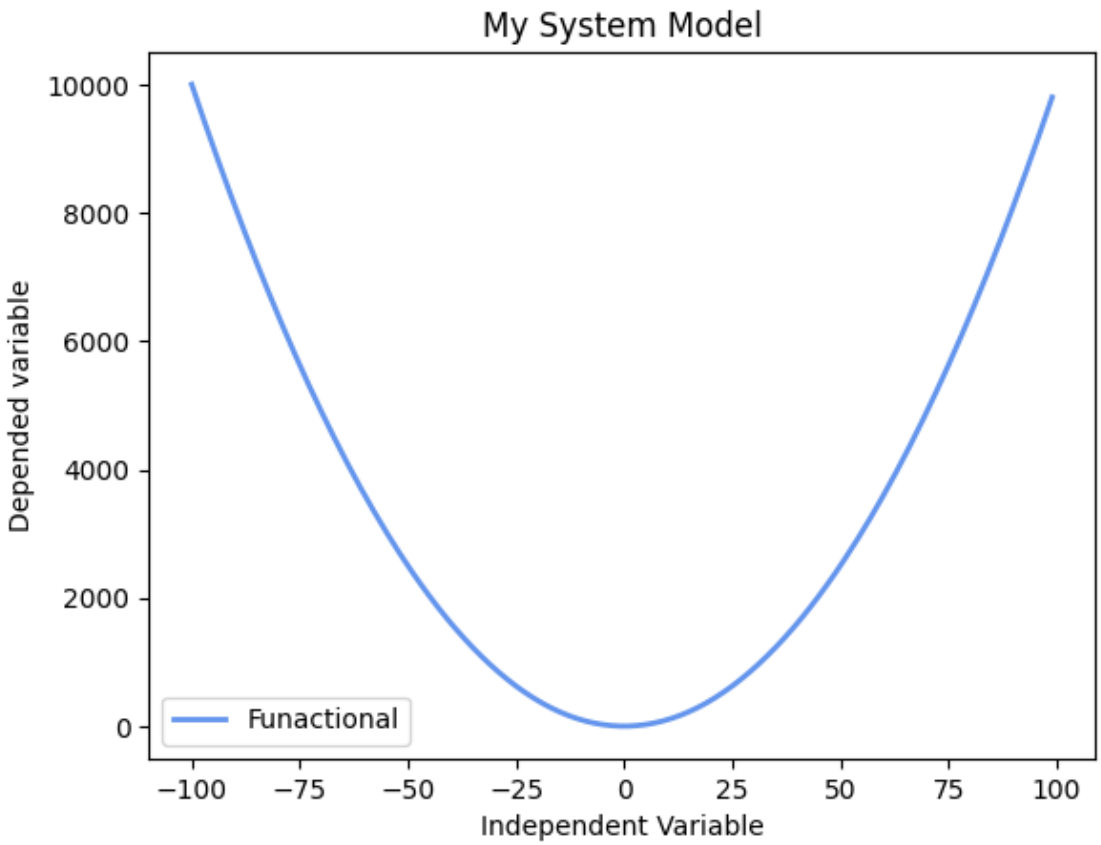
assignment

April 29, 2025

0.1 Program 1:

[1]:
```
import matplotlib.pyplot as plt
import numpy
from sympy import *

x = symbols("x")
exp = "x**2"
X = numpy.arange(-100, 100)
f = lambdify(x, exp, "numpy")
Y = f(X)

plt.figure()
# plt.scatter(X, Y, edgecolor="black", c="darkorange", label="Funactional")
plt.plot(X, Y, color="cornflowerblue", label="Funactional", linewidth=2)
plt.xlabel("Independent Variable")
plt.ylabel("Depended variable")
plt.title("My System Model")
plt.legend()
plt.show()
```


My System Model
10000
8000
6000
4000
2000
0
Depended variable
Funactional
−100
−75
−50
−25
0
25
50
75
100
Independent Variable

0.2 Program 2:

[2]:
```
import scipy.optimize as opti

C = [1, 1]
Aineq = [[1, 1], [2, 4]]
bineq = [10000, 20000]

Aeq = [[1, 1]]
beq = [5000]

x0_bounds = (0, 4000)
x1_bounds = (3000, 8000)

res = opti.linprog(
    C, A_eq=Aeq, b_eq=beq, A_ub=Aineq, b_ub=bineq, bounds=[x0_bounds, x1_bounds]
)

print("Objective Function Value=", res.fun)
print("Solution =", res.x)
```

```
Objective Function Value= 5000.0
Solution = [2000. 3000.]
```

0.3 Program 3:

[ ]:
```
from sklearn import linear_model

finmodel = linear_model.LinearRegression()

emi = [
    [1, 1],
    [2, 1],
    [3, 2],
    [4, 2],
    [5, 3],
    [6, 7],
    [7, 12],
    [8, 15],
    [9, 20],
    [10, 30],
]
income = [10, 15, 23, 25, 30, 39, 42, 50, 60, 75]

finmodel.fit(emi, income)
print(finmodel.predict([[4.5, 2], [12, 7]]))
```

```
Requirement already satisfied: scikit-learn in ./lib/python3.9/site-packages
(1.6.1)
Requirement already satisfied: numpy>=1.19.5 in ./lib/python3.9/site-packages
```

```
(from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in ./lib/python3.9/site-packages
(from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in ./lib/python3.9/site-packages
(from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in ./lib/python3.9/site-
packages (from scikit-learn) (3.6.0)
[26.43338101 60.74242804]
```

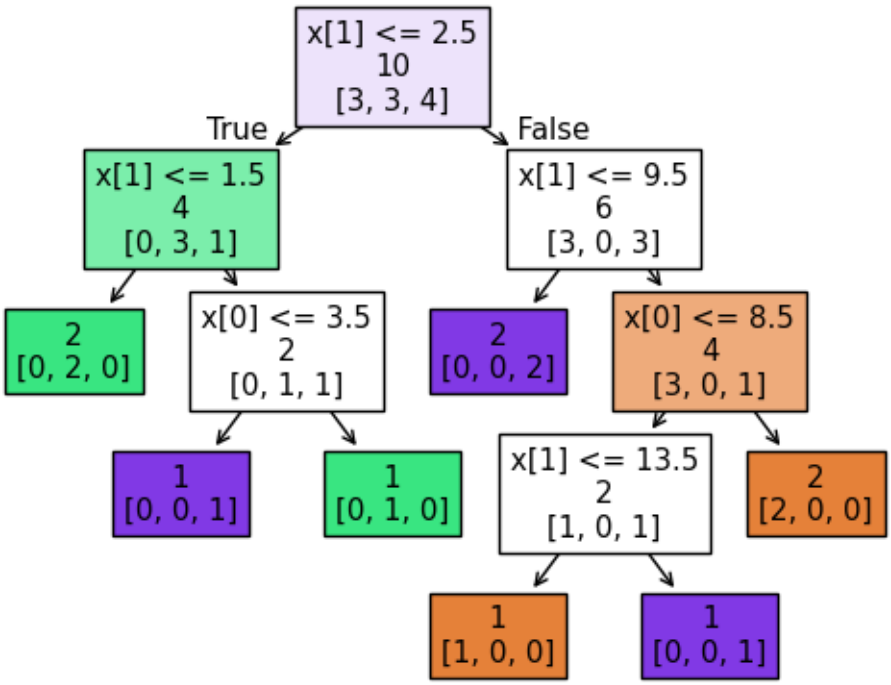
0.4 Program 4:

[4]:
```
from sklearn import tree

finmodel = tree.DecisionTreeClassifier()

emi = [
    [1, 1],
    [2, 1],
    [3, 2],
    [4, 2],
    [5, 3],
    [6, 7],
    [7, 12],
    [8, 15],
    [9, 20],
    [10, 30],
]
income = ["LESS", "LESS", "MED", "LESS", "MED", "MED", "HIGH", "MED", "HIGH",␣
 ↪"HIGH"]

finmodel.fit(emi, income)
print(finmodel.predict([[4.5, 2], [12, 7]]))
tree.plot_tree(
    finmodel,
    class_names=None,
    label="none",
    impurity=False,
    filled=True,
    node_ids=False,
)
```

```
['LESS' 'MED']
```

[4]:
```
[Text(0.4444444444444444, 0.9, 'x[1] <= 2.5\n10\n[3, 3, 4]'),
 Text(0.2222222222222222, 0.7, 'x[1] <= 1.5\n4\n[0, 3, 1]'),
 Text(0.3333333333333333, 0.8, 'True  '),
 Text(0.1111111111111111, 0.5, '2\n[0, 2, 0]'),
 Text(0.3333333333333333, 0.5, 'x[0] <= 3.5\n2\n[0, 1, 1]'),
```

```
Text(0.2222222222222222, 0.3, '1\n[0, 0, 1]'),
Text(0.4444444444444444, 0.3, '1\n[0, 1, 0]'),
Text(0.6666666666666666, 0.7, 'x[1] <= 9.5\n6\n[3, 0, 3]'),
Text(0.5555555555555556, 0.8, '  False'),
Text(0.5555555555555556, 0.5, '2\n[0, 0, 2]'),
Text(0.7777777777777778, 0.5, 'x[0] <= 8.5\n4\n[3, 0, 1]'),
Text(0.6666666666666666, 0.3, 'x[1] <= 13.5\n2\n[1, 0, 1]'),
Text(0.5555555555555556, 0.1, '1\n[1, 0, 0]'),
Text(0.7777777777777778, 0.1, '1\n[0, 0, 1]'),
Text(0.8888888888888888, 0.3, '2\n[2, 0, 0]')]
```



0.5 Program 5:

[5]:
```
class Universe:
    def EnergyCalculation(this):
        E = this.m * this.c * this.c
        return E


u = Universe()
u.m = 2
u.c = 3 * 10**8
EN = u.EnergyCalculation()
```

```
print(EN)
```

```
180000000000000000
```