MANISA CELAL BAYAR UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF CIVIL

ENGINEERING

INTERNSHIP PROGRAM

MID-TERM REPORT

SUPERVISOR

Res. Asst. Dr. Aybike ÖZYÜKSEL ÇİFTÇİOĞLU

PREPARED BY

150303055 Burak Mustafa KARAKAYA

Table of Contents

Figures

1. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an optimization algorithm developed by Dr. Kennedy and Dr. Eberhart in 1995, inspired by the observation that the movements exhibited by some animals in a herd, while fulfilling their basic needs such as finding food, affect other members of the herd and make it easier for the herd to achieve its goal.

In PSO (Particle Swarm Optimization), each individual searching for a solution is called a particle, while the population of particles is referred to as a swarm. To determine how close an individual is to the solution, a fitness function is used. The fitness function evaluates whether the individual is sufficient to solve a genetic algorithm problem, and serves as the objective function.

During a particle's search for a solution, the individual's best known position throughout the search is called pbest (personal best), whereas the position of the particle in the swarm that comes closest to the solution during the entire search is referred to as gbest (global best).

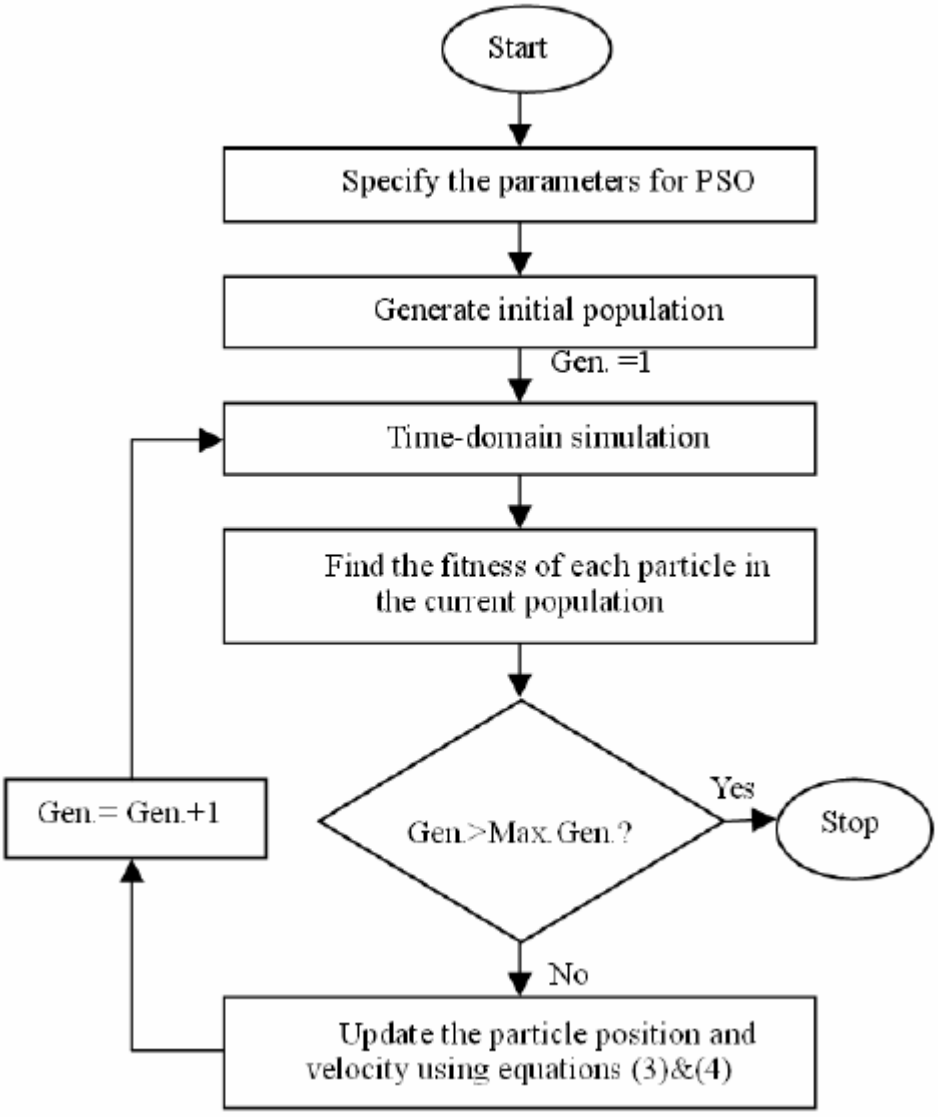

Figure 1 Flowchart of Particle Swarm Optimization

2. Grey Wolf Optimization (GWO)

GWO algorithm is an intuitive optimization method. Intuitive optimization is a method of approaching a problem's solution towards the best without emphasizing whether its accuracy can be proven. The GWO algorithm mimics the leadership and hunting mechanism of grey wolves in nature. There are 4 types in the hierarchy of grey wolves: Alpha, Beta, Delta, and Omega. The leader in the hierarchy is the Alpha wolf, responsible for the pack's decision-making in hunting, sleeping, waking up, etc. The Beta wolf is known as the assistant to the Alpha wolf in decision-making and other pack activities. The Beta wolf relays the orders of the Alpha wolf to the other wolves and provides feedback. The Beta wolf takes over in case of the Alpha wolf's aging or absence. The Omega wolf, the lowest-level wolf in the hierarchy, always waits for other wolves to satisfy their hunger and takes its turn. The Omega wolf always chooses the dominant wolf. The Delta wolf, third in the hierarchy, prioritizes hunting behavior in the wolf pack hierarchy. Wolves engage in hunting behavior first, then follow, approach, and observe their prey, and finally pursue, harass, and surround it. It is a 5-stage algorithm.

2.1. Social Hierarchy: AlfaAlpha is used as the best solution. Then, Beta and Delta are named as the second and third solutions, respectively. Omega is considered as a candidate solution as well.

2.2. Prey Encircling: As mentioned above, gray wolves encircle the prey during hunting. Gray wolves will move in a hypercube shape around the best solution obtained so far.

2.3. Hunting: Gray wolves have the ability to recognize their territory and surround it. The hunting is usually directed by Alpha, while Beta and Delta may also join the hunt. Let's assume that Beta is better informed about the potential location of the prey than Alpha (the best candidate solution), and Delta has even better knowledge. Therefore, the best first three results obtained so far are recorded, and the positions of other search agents (including Omegas) are updated according to the location of the best search agents.

2.4. Exploitation:: Gray wolves end the hunt by attacking the prey when it stops.

2.5. Prey Search (Exploration): Gray wolves search according to the location of other wolves. They spread out to search and gather together to attack the prey when found.

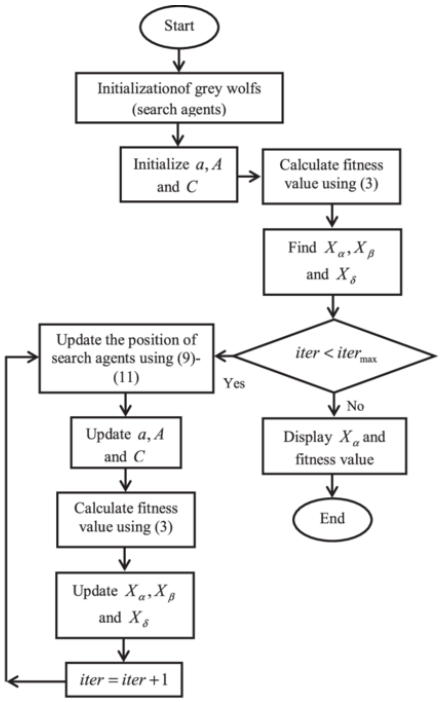

Figure 2 Grey Wolf Optimization Flowchart.

3. Comparison of Particle Swarm Optimization and Grey Wolf Optimization in Matlab

We will conduct the comparison using the functions shown below on Matlab.

Function	Dim	Range	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j\right)^2$	30	$[-100, 100]$	0
$f_4(x) = \max_i \{\lvert x_i \rvert, 1 \leqslant i \leqslant n\}$	30	$[-100, 100]$	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0
$f_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	30	$[-100, 100]$	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	$[-1.28, 1.28]$	0

Figure 3 Functions

Function	Dim	Range	f_{min}
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]$	-418.9829×5
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	30	$[-32, 32]$	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]$	0
$F_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	30	$[-50, 50]$	0
$F_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0
$F_{14}(x) = -\sum_{i=1}^{n} \sin(x_i) \cdot \left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m}, \; m = 10$	30	$[0, \pi]$	-4.687
$F_{15}(x) = \left[e^{-\sum_{i=1}^{n} (x_i/\beta)^{2m}} - 2e^{-\sum_{i=1}^{n} x_i^2}\right] \cdot \prod_{i=1}^{n} \cos^2 x_i, \; m = 5$	30	$[-20, 20]$	-1
$F_{16}(x) = \{[\sum_{i=1}^{n} \sin^2(x_i)] - \exp(-\sum_{i=1}^{n} x_i^2)\} \cdot \exp[-\sum_{i=1}^{n} \sin^2 \sqrt{\lvert x_i \rvert}]$	30	$[-10, 10]$	-1

Figure 4 Functions

Function	Dim	Range	f_{min}
$F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	$[-65,65]$	1
$F_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	$[-5,5]$	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5,5]$	-1.0316
$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	$[-5,5]$	0.398
$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	$[-2,2]$	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	$[1,3]$	-3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	$[0,1]$	-3.32
$F_{21}(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0,10]$	-10.1532
$F_{22}(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0,10]$	-10.4028
$F_{23}(x) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0,10]$	-10.5363

Figure 5 Functions

- F1

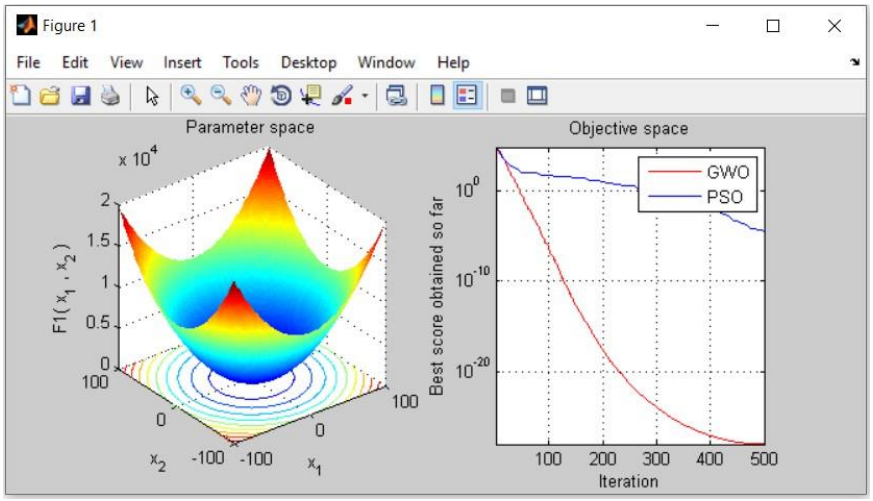

Figure 6 F1 Function Results

The best optimal value of the objective funciton found by GWO is : 8.3787e-29

- F2

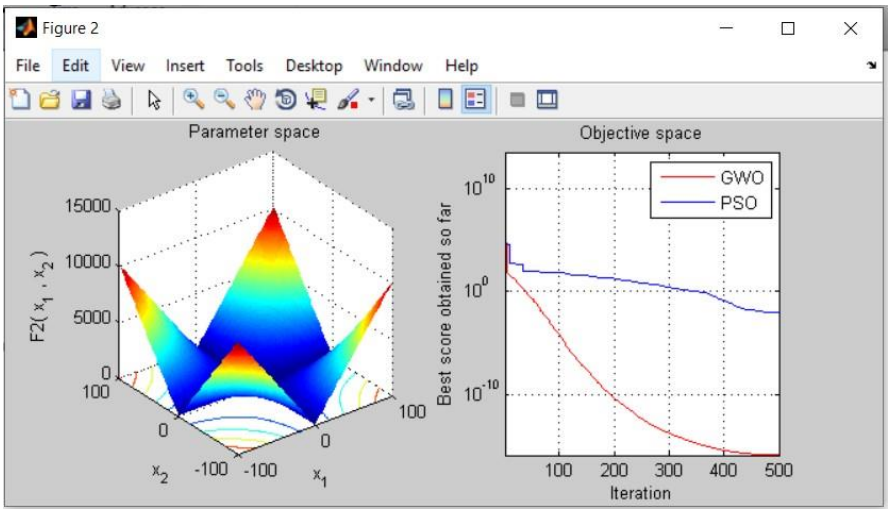

Figure 7 F2 Function Results

The best optimal value of the objective funciton found by GWO is : 1.1194e-16

- F3

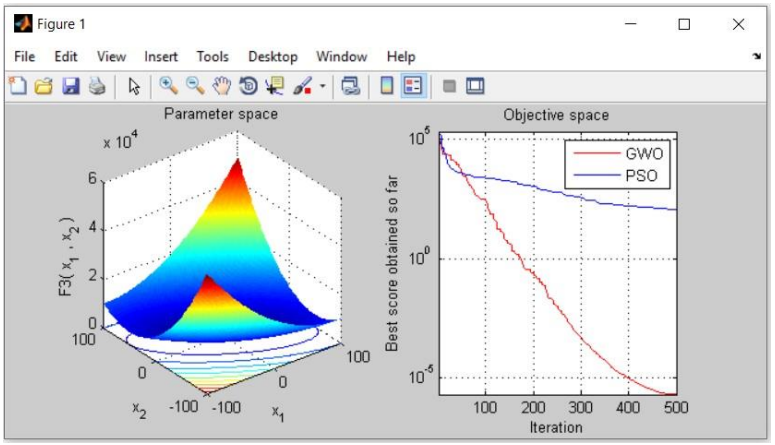

Figure 8 F3 Function Results

The best optimal value of the objective funciton found by GWO is : 1.9786e-06

- F4

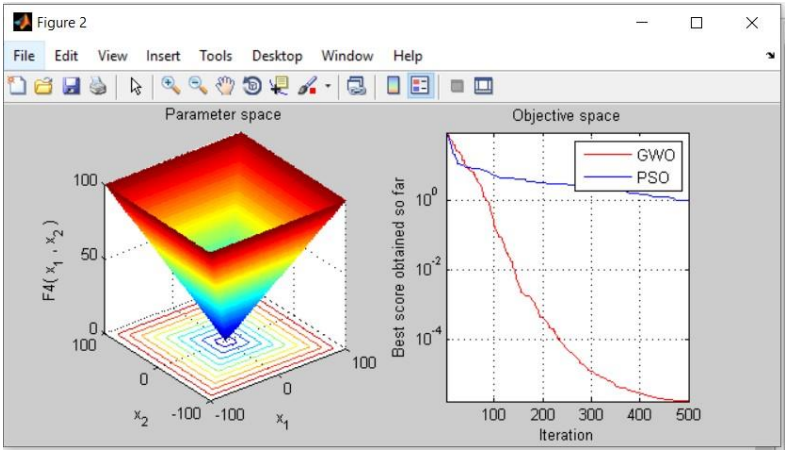

Figure 9 F4 Function Results

The best optimal value of the objective funciton found by GWO is : 1.736e-06

- F5

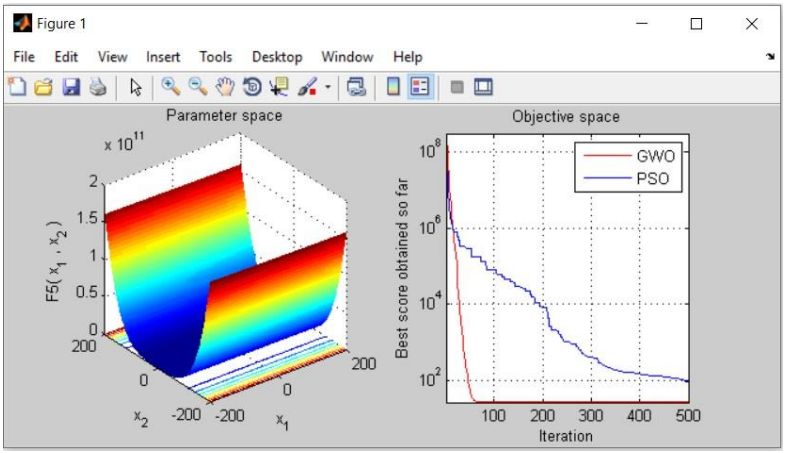

Figure 10 F5 Function Results

The best optimal value of the objective funciton found by GWO is : 27.147

- F6

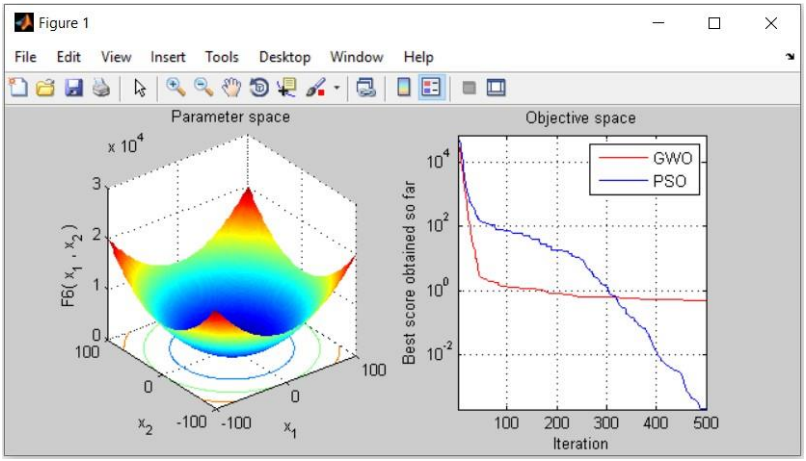

Figure 11 F6 Function Results

The best optimal value of the objective funciton found by PSO is : 0,000200250289120245

- F7

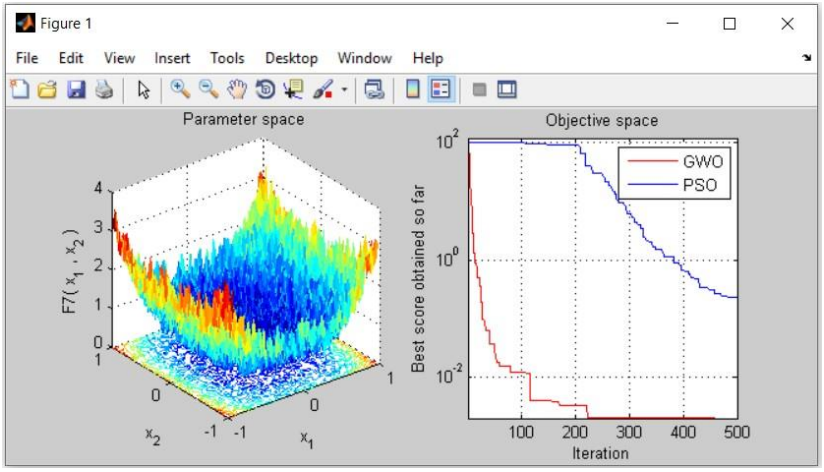

Figure 12 F7 Function Results

The best optimal value of the objective funciton found by GWO is : 0.0020325

- F8

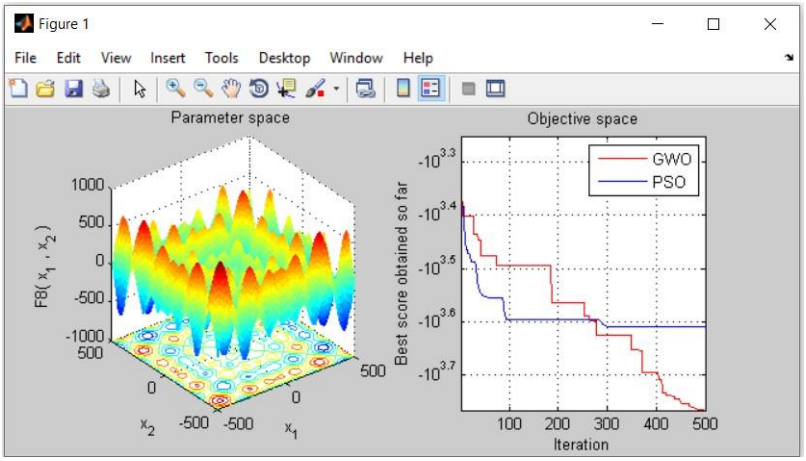

Figure 13 F8 Function Results

The best optimal value of the objective funciton found by GWO is : -5853.2079

- F9

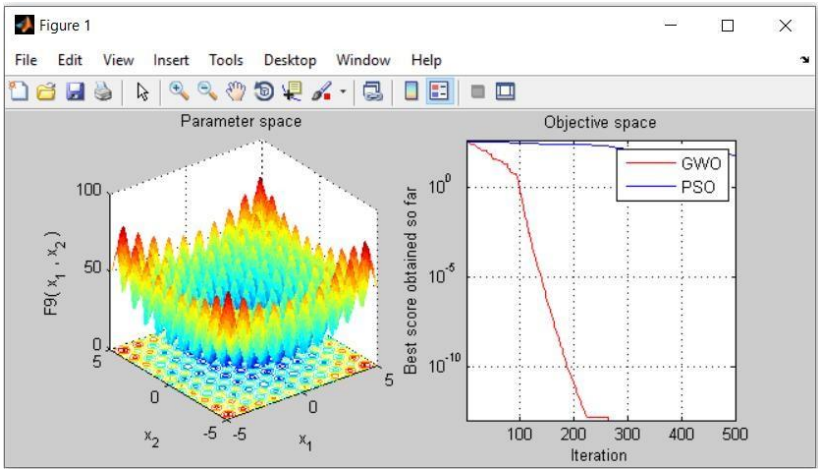

Figure 14 F9 Function Results

The best optimal value of the objective funciton found by GWO is : 1.1369e-13

- F10

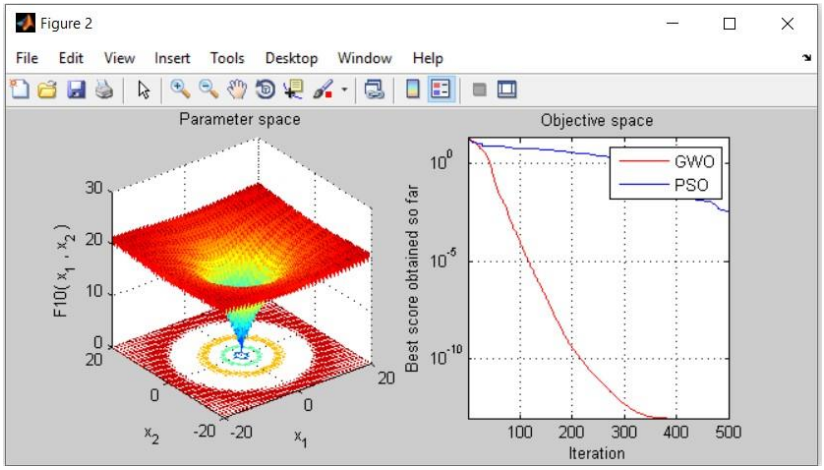

Figure 15 F10 Function Results

The best optimal value of the objective funciton found by GWO is : 8.6153e-14

- F11

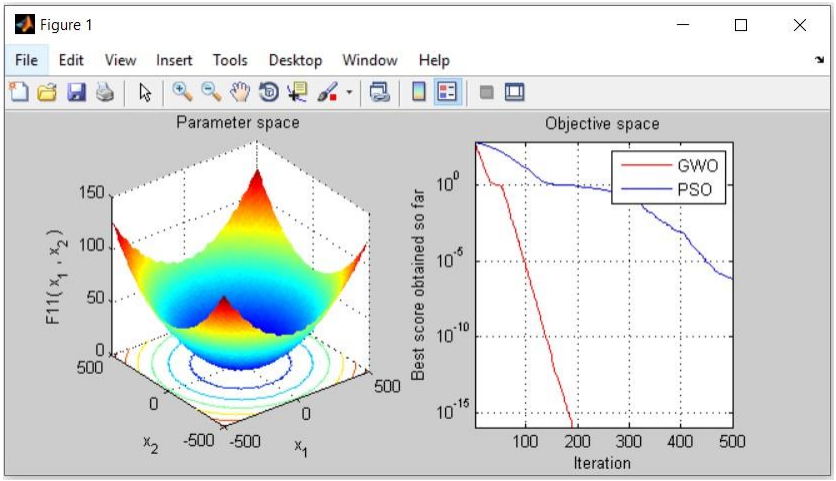

Figure 16 F11 Function Results

The best optimal value of the objective funciton found by GWO is: 0

- F12

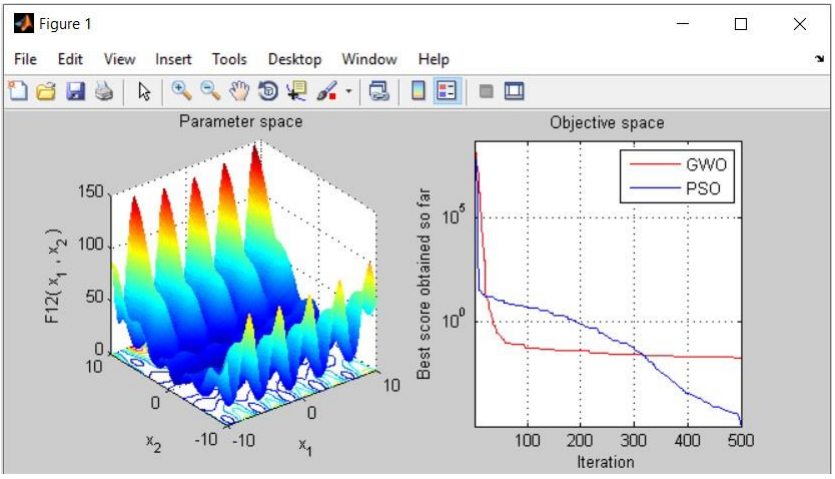

Figure 17 F12 Function Results

The best optimal value of the objective funciton found by PSO is: 1,02050649030209e-05

- F13

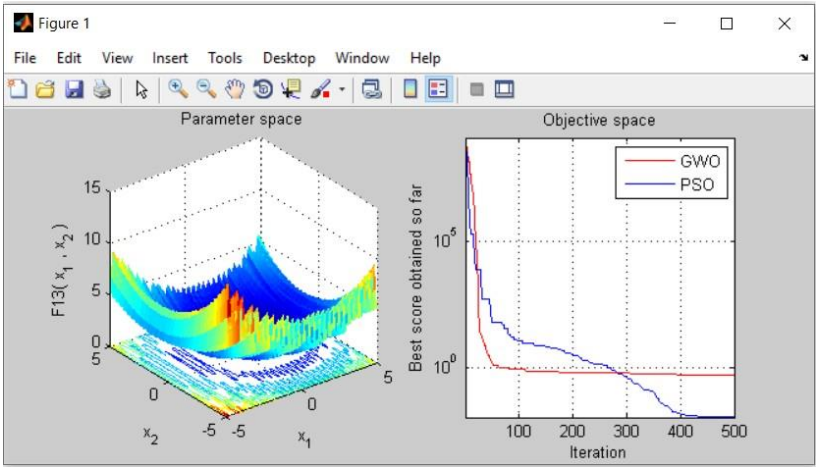

Figure 18 F13 Function Results

The best optimal value of the objective funciton found by PSO is : 0,0110330208902530

- F14

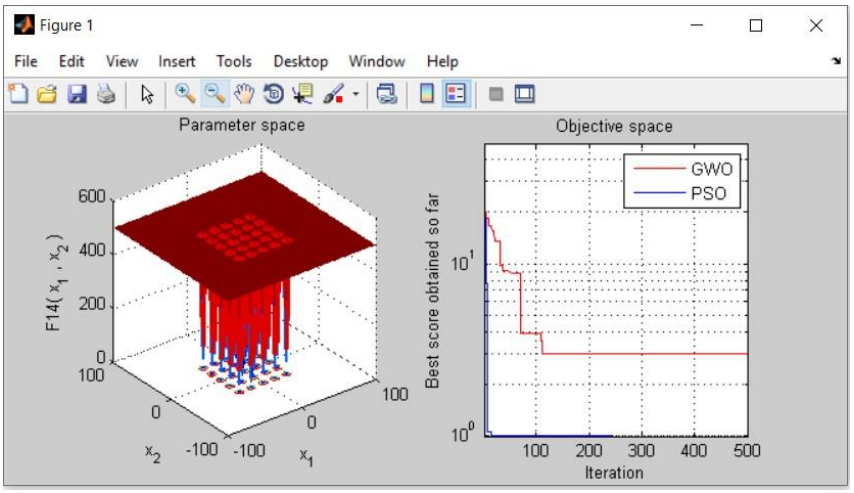

Figure 19 F14 Function Results

The best optimal value of the objective funciton found by PSO is : 0,998003837794450

- F15

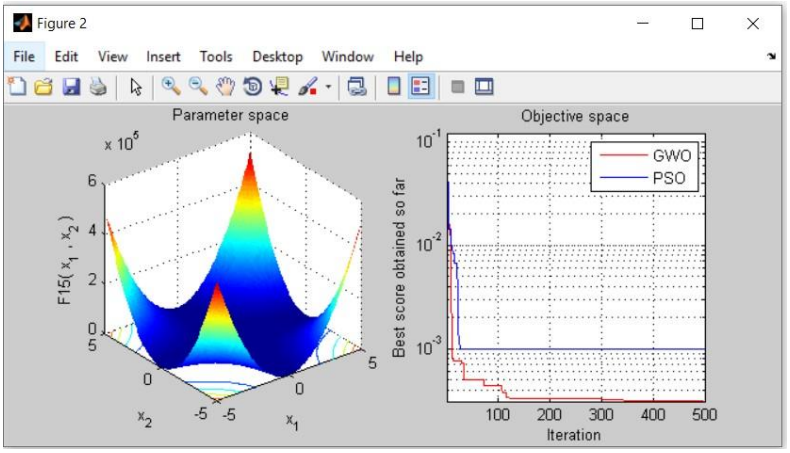

Figure 20 F15 Function Results

The best optimal value of the objective funciton found by GWO is : 0.00030881

- F16

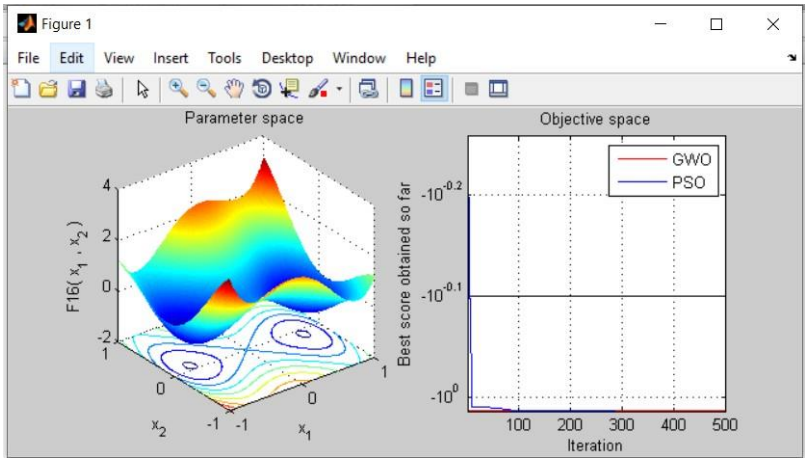

Figure 21 F16 Function Results

The best optimal value of the objective funciton found by GWO is : -1.0316

- F17

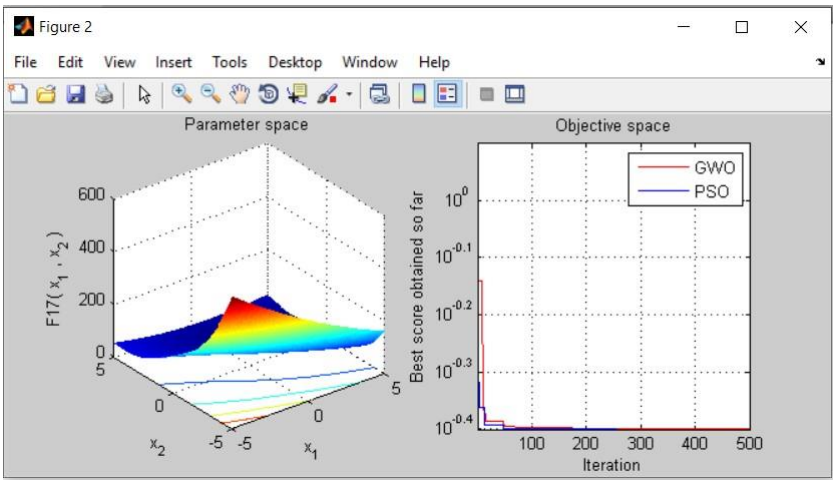

Figure 22 F17 Function Results

The best optimal value of the objective funciton found by GWO is : 0.39789

- F18

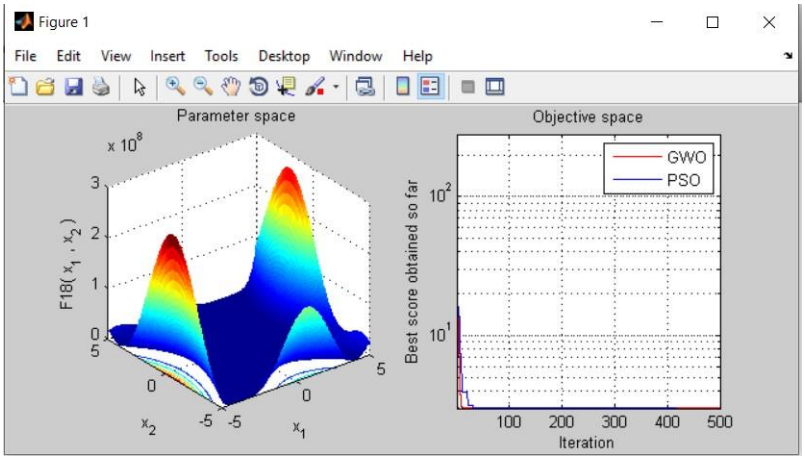

Figure 23 F18 Function Results

The best optimal value of the objective funciton found by GWO is : 3

- F19

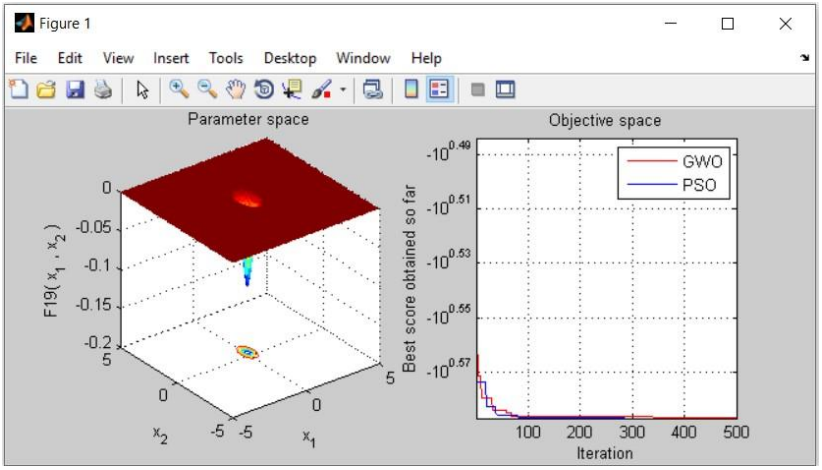

Figure 24 F19 Function Results

The best optimal value of the objective funciton found by GWO is : -3.8606

- F20

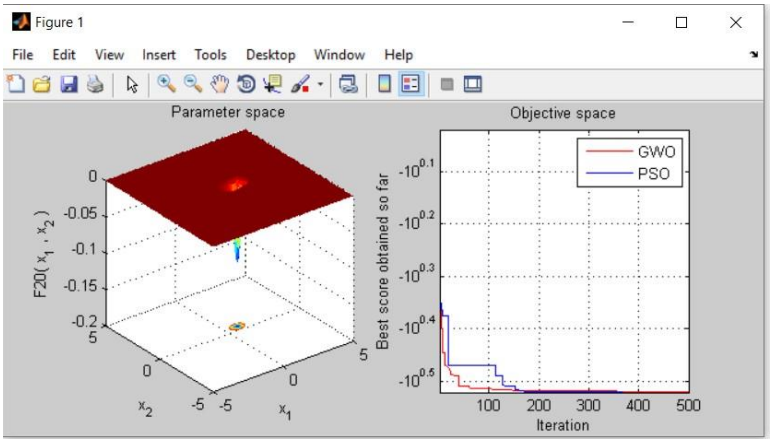

Figure 25 F20 Function Results

The best optimal value of the objective funciton found by GWO is : -3.322

- F21

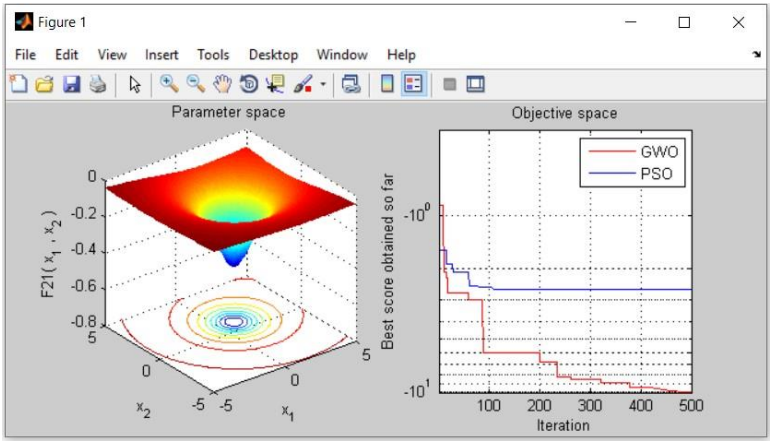

Figure 26 F21 Function Results

The best optimal value of the objective funciton found by GWO is : -10.1511

- F22

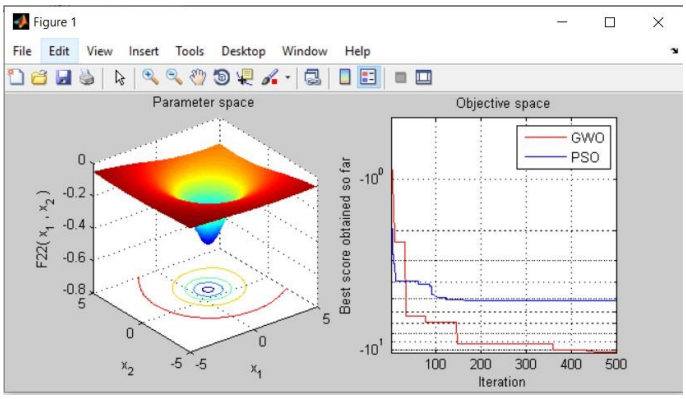

Figure 27 F22 Function Results

The best optimal value of the objective funciton found by GWO is : -10.4006

- F23

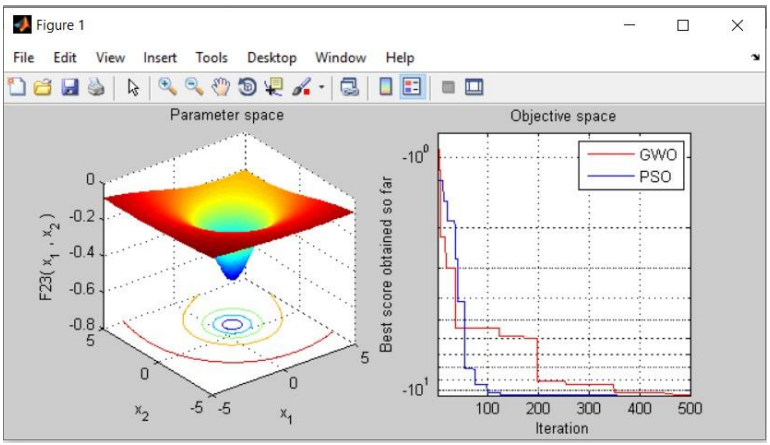

Figure 28 F23 Function Results

The best optimal value of the objective funciton found by PSO is : - 10,5364098166920

4. Machine Learning

Machine learning is a subfield of computer science that was developed primarily from the study of numerical learning and pattern recognition in artificial intelligence in 1959. It is a system that investigates the construction and operation of algorithms that can learn structurally and make predictions based on data. Such algorithms work by constructing a model to make data-driven predictions and decisions instead of strictly following static program instructions.

Machine learning algorithms can be classified into several categories based on the intended outcome:

- Supervised learning - produces a function that maps inputs to desired outputs.
- Unsupervised learning - models a set of inputs.
- Reinforcement learning - a learning method based on the perception of the world. Each action produces an effect on the environment, which provides feedback in the form of rewards that guide the learning algorithm.
- Semi-supervised learning - considers labeled and unlabeled examples together to create appropriate functions or classifiers.
- Learning to learn - leverages previous experience.

Currently, machine learning is used in a wide range of applications. Perhaps one of the best-known examples of machine learning is the recommendation engine that powers the feeds on Facebook, Instagram, and Twitter.

Facebook uses machine learning to personalize each user's homepage. For example, if a user stops to read posts from a certain group, the recommendation engine will start showing more content from that group.

In addition to recommendation engines, other areas where machine learning is used include customer relationship management, business intelligence, human resource information systems, autonomous vehicles, and virtual assistants.

One of the main advantages is that it helps businesses better understand their customers. Machine learning algorithms can learn relationships by collecting customer data and associating it with behaviors over time, which can help teams tailor their product development and marketing efforts to customer demand.

Some internet companies use machine learning as the primary driving force in their business models. For example, Uber uses algorithms to match drivers with riders, while Google uses machine learning to display accurate ads in search results.

However, there are also some disadvantages to machine learning. First and foremost, it is costly. Machine learning projects are usually led by expert data scientists, which increases the cost of the team, and the projects may require expensive software infrastructure.

There is also a bias problem in machine learning. Algorithms trained on data sets that exclude or contain errors for certain segments of society can lead to models that are at best unsuccessful and at worst discriminatory. When an organization relies on biased models for their core business processes, they may face legal and reputational issues.

Below is an example of a Python code applied through Colaboratory.

10 girişten 1-10 aralığındakiler Filtre

Country	Age	Salary	Purchased
France	44	72000	No
Spain	27	48000	Yes
Germany	30	54000	No
Spain	38	61000	No
Germany	40		Yes
France	35	58000	Yes
Spain		52000	No
France	48	79000	Yes
Germany	50	83000	No
France	37	67000	Yes

Her sayfada 10 satır göster

Figure 29 Data Set

Importing the libraries

```
[5]  import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
```

Figure 30 Checking out from the library

Importing the dataset

```
[6]  dataset = pd.read_csv('Data.csv')
     X = dataset.iloc[:, :-1].values
     y = dataset.iloc[:, -1].values
```

```
[7]  print(X)
```

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 nan]
 ['France' 35.0 58000.0]
 ['Spain' nan 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
```

```
[8]  print(y)
```

```
['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']
```

Figure 31 Obtaining the Dataset

Taking care of missing data

```
[9]  from sklearn.impute import SimpleImputer
     imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
     imputer.fit(X[:, 1:3])
     X[:, 1:3] = imputer.transform(X[:, 1:3])
```

```
[10] print(X)
```

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 63777.77777777778]
 ['France' 35.0 58000.0]
 ['Spain' 38.77777777777778 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
```

Figure 32 32 Reorganizing Missing Data

Encoding categorical data

Encoding the Independent Variable

```
[11] from sklearn.compose import ColumnTransformer
     from sklearn.preprocessing import OneHotEncoder
     ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='pa
     X = np.array(ct.fit_transform(X))
```

```
[12] print(X)
```

```
[[1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [0.0 1.0 0.0 30.0 54000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 35.0 58000.0]
 [0.0 0.0 1.0 38.77777777777778 52000.0]
 [1.0 0.0 0.0 48.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 37.0 67000.0]]
```

Figure 33 Encoding independent variables

Encoding the Dependent Variable

```
[13] from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     y = le.fit_transform(y)
```

```
[ ] print(y)

    [0 1 0 0 1 1 0 1 0 1]
```

Figure 34 Encoding dependent variables

Splitting the dataset into the Training set and Test set

```
[14] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_stat
```

```
[15] print(X_train)

     [[0.0 0.0 1.0 38.77777777777778 52000.0]
      [0.0 1.0 0.0 40.0 63777.77777777778]
      [1.0 0.0 0.0 44.0 72000.0]
      [0.0 0.0 1.0 38.0 61000.0]
      [0.0 0.0 1.0 27.0 48000.0]
      [1.0 0.0 0.0 48.0 79000.0]
      [0.0 1.0 0.0 50.0 83000.0]
      [1.0 0.0 0.0 35.0 58000.0]]
```

```
[16] print(X_test)

     [[0.0 1.0 0.0 30.0 54000.0]
      [1.0 0.0 0.0 37.0 67000.0]]
```

```
[17] print(y_train)

     [0 1 0 0 1 1 0 1]
```

```
[18] print(y_test)

     [0 1]
```

Figure 35 Splitting the Data

Feature Scaling

```
[19] from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train[:, 3:] = sc.fit_transform(X_train[:, 3:])
     X_test[:, 3:] = sc.transform(X_test[:, 3:])
```

```
[20] print(X_train)

     [[0.0 0.0 1.0 -0.19159184384578545 -1.0781259408412425]
      [0.0 1.0 0.0 -0.014117293757057777 -0.07013167641635372]
      [1.0 0.0 0.0 0.566708506533324 0.633562432710455]
      [0.0 0.0 1.0 -0.30453019390224867 -0.30786617274297867]
      [0.0 0.0 1.0 -1.9018011447007988 -1.420463615551582]
      [1.0 0.0 0.0 1.1475343068237058 1.232653363453549]
      [0.0 1.0 0.0 1.4379472069688968 1.5749910381638885]
      [1.0 0.0 0.0 -0.7401495441200351 -0.5646194287757332]]
```

```
[21] print(X_test)

     [[0.0 1.0 0.0 -1.4661817944830124 -0.9069571034860727]
      [1.0 0.0 0.0 -0.44973664397484414 0.2056403393225306]]
```

Figure 36 Predictions

5. Regression Analysis

Regression analysis is a method used to measure the relationship between two or more quantitative variables. If only one variable is used in the analysis, it is called simple regression, and if multiple variables are used, it is referred to as multiple regression analysis. Regression analysis can provide information about the presence of a relationship between variables and the strength of that relationship, if it exists.

For example, an agricultural engineer may want to know the relationship between wheat yield and fertilizer amount, an engineer may be interested in the relationship between pressure and temperature, an economist may want to know the relationship between income level and consumption expenditures, or an educator may be interested in the relationship between the number of days absent and the level of academic achievement for students. Regression not only shows the functional form of the linear relationship between two (or more) variables as a linear equation with one variable as the dependent and the other as the independent variable, but it also allows for estimation of one variable when the value of the other is known. Typically, it is necessary for all variables to be measured quantitatively.

In regression, one variable must be dependent while the others are independent variables. The logic here is that the variable on the left side of the equation is influenced by the variables on the right side. The variables on the right side, however, are not affected by the other variables. Here, not being affected means that when we put these variables into a linear equation, they have an effect.

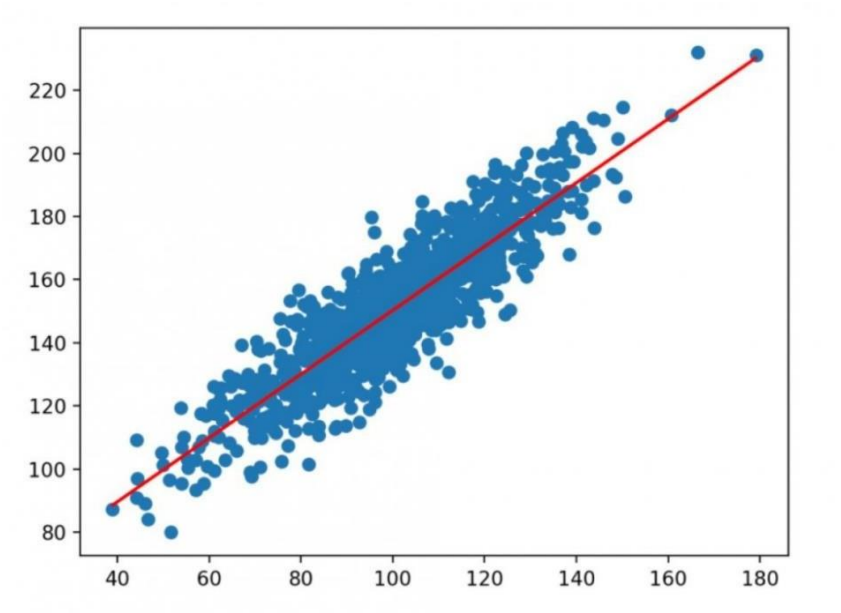

Figure 37 Sample Analysis

5.1. Simple Linear Regression

Simple linear regression is used to estimate the relationship between two quantitative variables.

You can use it to learn::

- How strong the relationship is between two variables (e.g. the relationship between rainfall and soil erosion).
- The value of the dependent variable at a specific value of the independent variable (e.g. the amount of soil erosion at a certain level of rainfall).

Simple linear regression is a parametric test, meaning it makes assumptions about the data. These assumptions are:

- **Homoscedasticity:** the size of the error in our prediction is constant across the values of the independent variable.
- **Independence of observations:** the observations in the dataset were collected using valid sampling methods and there are no hidden relationships among the observations.
- **Normallik:** the data follows a normal

  distrubition

Linear regression makes an additional assumption:

- **Linearity:** the relationship between the independent and dependent variable is linear, meaning the best fitting line through the data points is a straight line (as opposed to a curve or some kind of grouping factor).

$$y = \beta_0 + \beta_1 X + \varepsilon$$

Figure 38 Formula of Simple Linear Regression

- y is the predicted value of the dependent variable (y) for any value of the independent variable (x).
- B0 is the intercept, the predicted value of y when x equals 0.
- x is the independent variable (the one affecting y).
- e is the error of the prediction, or how much variation there is in our regression coefficient estimate.

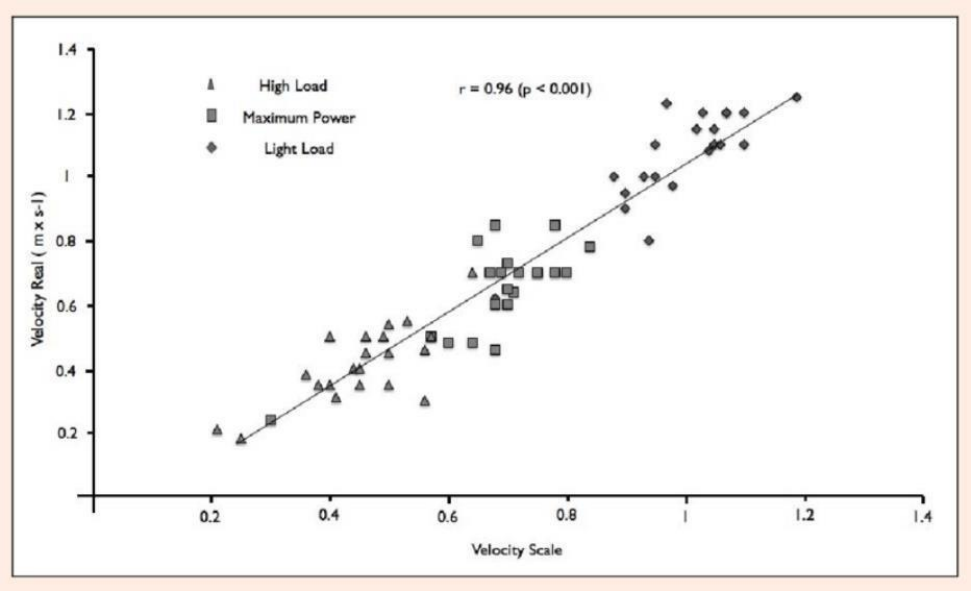

Figure 39 Example Graph of Simple Linear Regression

Below are step-by-step visualizations of the study conducted on a dataset containing salaries based on years of work

YearsExperience	Salary
1.1	39343.00
1.3	46205.00
1.5	37731.00
2.0	43525.00
2.2	39891.00
2.9	56642.00
3.0	60150.00
3.2	54445.00
3.2	64445.00
3.7	57189.00

Her sayfada 10 satır göster 1 2 3

Figure 40 Years of Experience and Salaries

Importing the libraries

```
[2]  import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
```

Figure 41 Checking out from the library

Importing the dataset

```
[3]  dataset = pd.read_csv('Salary_Data.csv')
     X = dataset.iloc[:, :-1].values
     y = dataset.iloc[:, -1].values
```

Figure 42 Acquiring the Data

Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_stat
```

Figure 43 Splitting the Data

Training the Simple Linear Regression model on the Training set

```
[5]  from sklearn.linear_model import LinearRegression
     regressor = LinearRegression()
     regressor.fit(X_train, y_train)

     LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Figure 44 Training

Predicting the Test set results

```
[6]  y_pred = regressor.predict(X_test)
```

Figure 45 Test Predictions

▾ Visualising the Training set results

```
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Figure 46 Training Graph

▾ Visualising the Test set results

```
[8] plt.scatter(X_test, y_test, color = 'red')
    plt.plot(X_train, regressor.predict(X_train), color = 'blue')
    plt.title('Salary vs Experience (Test set)')
    plt.xlabel('Years of Experience')
    plt.ylabel('Salary')
    plt.show()
```

Figure 47 Test Graph

5.2. Multiple Linear Regression

Multiple linear regression is used to estimate the relationship between one dependent variable and two or more independent variables.

You can use multiple linear regression to learn:

- How strong the relationship is between one dependent variable and two or more independent variables (e.g. how precipitation, temperature, and amount of fertilizer added affect crop growth).
- The expected value of the dependent variable at specific values of the independent variables (e.g. the expected yield of a crop at specific levels of precipitation, temperature, and fertilizer).

Multiple linear regression makes the same assumptions as simple linear regression.

$$y = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n + \varepsilon$$

Figure 48 Multiple Linear Regression

- y = estimated value of the dependent variable

- B_0 = y-intercept (the value of y when all other parameters are set to 0)
- B_1X_1 = regression coefficient (B1) of the first independent variable (X1) (the effect of increasing the value of the independent variable on the estimated value of y)
- … = we do the same thing for many independent variables we test
- B_nX_n = regression coefficient of the last independent variable
- e = model error (the amount of variation in our y estimate)

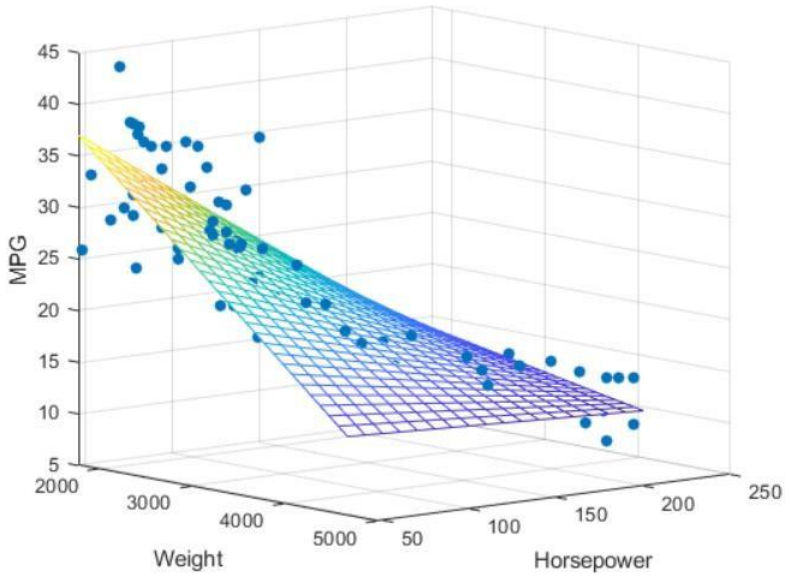

Figure 49 Multiple Linear Regression

Below, the multiple linear regression analysis conducted on the data of 50 different startups is presented with figures.

R&D Spend	Administration	Marketing Spend	State	Prof
76253.86	113867.3	298664.47	California	118474
78389.47	153773.43	299737.29	New York	111313
73994.56	122782.75	303319.26	Florida	110352
67532.53	105751.03	304768.73	Florida	108733
77044.01	99281.34	140574.81	New York	108552
64664.71	139553.16	137962.62	California	107404
75328.87	144135.98	134050.07	Florida	105733
72107.6	127864.55	353183.81	New York	105008
66051.52	182645.56	118148.2	Florida	103282
65605.48	153032.06	107138.38	New York	101004

Her sayfada 10 satır göster 1 2 3 4 5

Figure 50 Data

Importing the libraries

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
```

Figure 51 Checking out from the library

Importing the dataset

```
[ ]  dataset = pd.read_csv('50_Startups.csv')
     X = dataset.iloc[:, :-1].values
     y = dataset.iloc[:, -1].values
```

```
[ ]  print(X)
```

```
[[165349.2 136897.8 471784.1 'New York']
 [162597.7 151377.59 443898.53 'California']
 [153441.51 101145.55 407934.54 'Florida']
 [144372.41 118671.85 383199.62 'New York']
 [142107.34 91391.77 366168.42 'Florida']
 [131876.9 99814.71 362861.36 'New York']
 [134615.46 147198.87 127716.82 'California']
 [130298.13 145530.06 323876.68 'Florida']
 [120542.52 148718.95 311613.29 'New York']
 [123334.88 108679.17 304981.62 'California']
 [101913.08 110594.11 229160.95 'Florida']
 [100671.96 91790.61 249744.55 'California']
 [93863.75 127320.38 249839.44 'Florida']
 [91992.39 135495.07 252664.93 'California']
 [119943.24 156547.42 256512.92 'Florida']
 [114523.61 122616.84 261776.23 'New York']
 [78013.11 121597.55 264346.06 'California']
 [94657.16 145077.58 282574.31 'New York']
 [91749.16 114175.79 294919.57 'Florida']
```

Figure 52 Acquiring the Data

Encoding categorical data

```
[ ] from sklearn.compose import ColumnTransformer
    from sklearn.preprocessing import OneHotEncoder
    ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='pa
    X = np.array(ct.fit_transform(X))
```

```
[ ] print(X)
```

```
[[0.0 0.0 1.0 165349.2 136897.8 471784.1]
 [1.0 0.0 0.0 162597.7 151377.59 443898.53]
 [0.0 1.0 0.0 153441.51 101145.55 407934.54]
 [0.0 0.0 1.0 144372.41 118671.85 383199.62]
 [0.0 1.0 0.0 142107.34 91391.77 366168.42]
 [0.0 0.0 1.0 131876.9 99814.71 362861.36]
 [1.0 0.0 0.0 134615.46 147198.87 127716.82]
 [0.0 1.0 0.0 130298.13 145530.06 323876.68]
 [0.0 0.0 1.0 120542.52 148718.95 311613.29]
 [1.0 0.0 0.0 123334.88 108679.17 304981.62]
 [0.0 1.0 0.0 101913.08 110594.11 229160.95]
 [1.0 0.0 0.0 100671.96 91790.61 249744.55]
 [0.0 1.0 0.0 93863.75 127320.38 249839.44]
 [1.0 0.0 0.0 91992.39 135495.07 252664.93]
 [0.0 1.0 0.0 119943.24 156547.42 256512.92]
 [0.0 0.0 1.0 114523.61 122616.84 261776.23]
 [1.0 0.0 0.0 78013.11 121597.55 264346.06]
 [0.0 0.0 1.0 94657.16 145077.58 282574.31]
 [0.0 1.0 0.0 91749.16 114175.79 294919.57]
 [0.0 0.0 1.0 86419.7 153514.11 0.0]
 [1.0 0.0 0.0 76253.86 113867.3 298664.47]
```

Figure 53 Data Preprocessing

Splitting the dataset into the Training set and Test set

```
[ ] from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_stat
```

Figure 54 Splitting the Data

Training the Multiple Linear Regression model on the Training set

```
[ ] from sklearn.linear_model import LinearRegression
    regressor = LinearRegression()
    regressor.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Figure 55 Training

Predicting the Test set results

```
[ ]  y_pred = regressor.predict(X_test)
     np.set_printoptions(precision=2)
     print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1)
```

```
[[103015.2  103282.38]
 [132582.28 144259.4 ]
 [132447.74 146121.95]
 [ 71976.1   77798.83]
 [178537.48 191050.39]
 [116161.24 105008.31]
 [ 67851.69  81229.06]
 [ 98791.73  97483.56]
 [113969.44 110352.25]
 [167921.07 166187.94]]
```

Figure 56 Predictions

5.3. Polynomial Regression

Polynomial Regression is a regression analysis form where the relationship between independent and dependent variables is modeled in an n-degree polynomial. Polynomial Regression models are typically suitable for the method of least squares. The method of least squares minimizes the variance of the coefficients under the Gauss-Markov Theorem. Polynomial Regression is a special case of Linear Regression where we place the polynomial equation with a curvilinear relationship between the dependent and independent variables to the data.

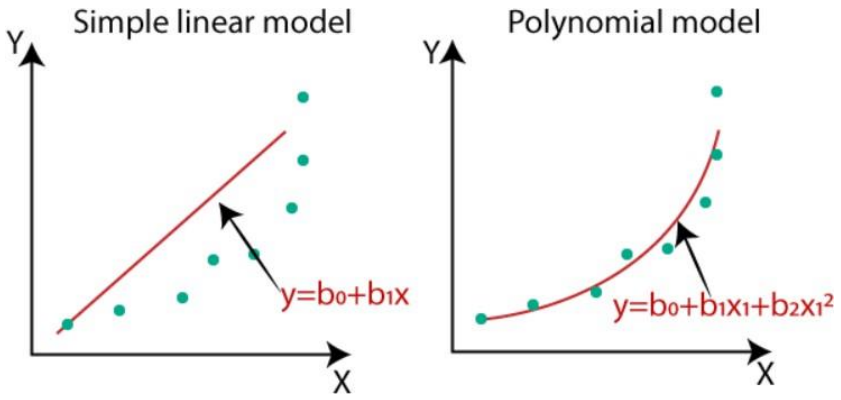

Figure 57 Difference between Linear and Polynomial Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

Figure 58 Polynomial Regression Formula

The following data includes salaries based on job positions. An application has been conducted on this data.

Position	Level	Salary
Business Analyst	1	45000
Junior Consultant	2	50000
Senior Consultant	3	60000
Manager	4	80000
Country Manager	5	110000
Region Manager	6	150000
Partner	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

Her sayfada 10 satır göster

Figure 59 Data

Importing the libraries

+ Kod + Metin

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
```

Figure 60 Step 1

Importing the dataset

```
[ ]  dataset = pd.read_csv('Position_Salaries.csv')
     X = dataset.iloc[:, 1:-1].values
     y = dataset.iloc[:, -1].values
```

Figure 61 Step 2

Training the Linear Regression model on the whole dataset

```
[3]  from sklearn.linear_model import LinearRegression
     lin_reg = LinearRegression()
     lin_reg.fit(X, y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Figure 62 Step 3

Training the Polynomial Regression model on the whole dataset

```
[4] from sklearn.preprocessing import PolynomialFeatures
    poly_reg = PolynomialFeatures(degree = 4)
    X_poly = poly_reg.fit_transform(X)
    lin_reg_2 = LinearRegression()
    lin_reg_2.fit(X_poly, y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Figure 63 Step 4

Visualising the Linear Regression results

```
[5] plt.scatter(X, y, color = 'red')
    plt.plot(X, lin_reg.predict(X), color = 'blue')
    plt.title('Truth or Bluff (Linear Regression)')
    plt.xlabel('Position Level')
    plt.ylabel('Salary')
    plt.show()
```

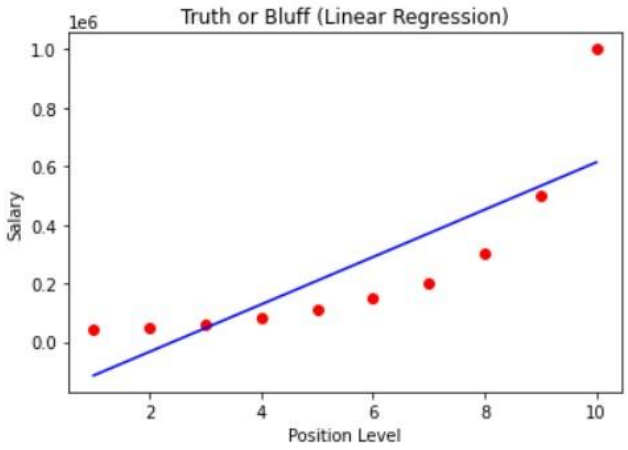

Figure 64 Step 5

Visualising the Polynomial Regression results

```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

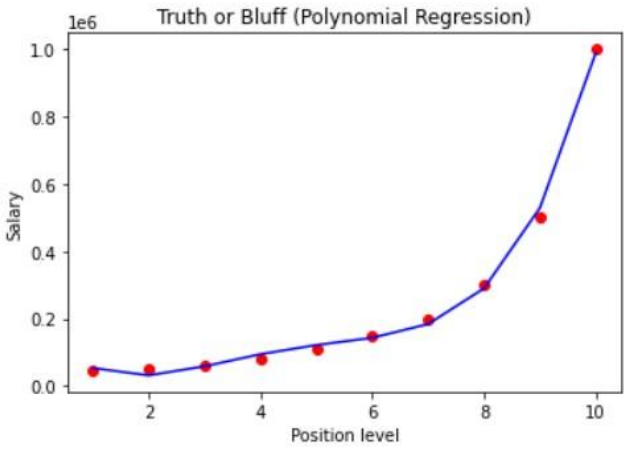

Figure 65 Step 6

Visualising the Polynomial Regression results (for higher resolution and smoother curve)

```
[7]  X_grid = np.arange(min(X), max(X), 0.1)
     X_grid = X_grid.reshape((len(X_grid), 1))
     plt.scatter(X, y, color = 'red')
     plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
     plt.title('Truth or Bluff (Polynomial Regression)')
     plt.xlabel('Position level')
     plt.ylabel('Salary')
     plt.show()
```

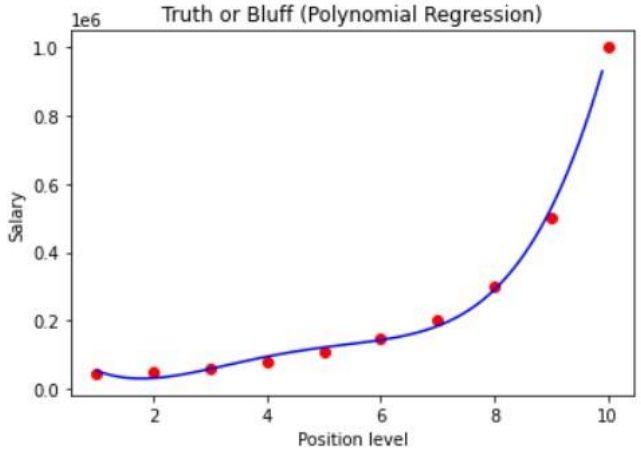

Figure 66 Step 7

Predicting a new result with Linear Regression

```
[8]  lin_reg.predict([[6.5]])
```

```
array([330378.78787879])
```

Predicting a new result with Polynomial Regression

```
lin_reg_2.predict(poly_reg.fit_transform([[6.5]]))
```

```
array([158862.45265155])
```

Figure 67 Step 8

6. Conclusion

In the MATLAB section, we observed that Grey Wolf Optimization performs better in a total of 18 functions, while Particle Swarm Optimization provides better results in the remaining 5 functions. Therefore, we can say that Grey Wolf Optimization (GWO) is a more applicable optimization method.

In the Python section, I learned how to process complex data problems and make them understandable. I also learned how to make predictions for the future and gained knowledge of the program's working infrastructure. I saw that these programs can be used in construction engineering for material and timing management and operations.

References

https://medium.com/deep-learning-turkiye/par%C3%A7ac%C4%B1k-s%C3%BCr%C3%BC-optimizasyonu-pso-1402d4fe924a

https://tr.wikipedia.org/wiki/Uygunluk_fonksiyonu

https://openaccess.iyte.edu.tr/xmlui/bitstream/handle/11147/7537/paper3.pdf?sequence=27&isAllowed=y

https://birbakmali.blogspot.com/2019/12/gri-kurt-algoritmas-hakknda-bilgi-slayt.html

https://www.endustri40.com/makine-ogrenimi-nedir/

https://tr.wikipedia.org/wiki/Makine_%C3%B6%C4%9Frenimi#%C3%96%C4%9Frenme_yakla%C5%9F%C4%B1mlar%C4%B1

https://digitalreport.com.tr/makine-ogrenimi-machine-learning-ya-da-ml-nedir-tipleri-denetimli-yari-denetimli-ve-denetimsiz-ml-nasil-calisir-pekistirmeli-ogrenme-nedir-kullanim-alanlari-ve-en-iyi-ornekleri-teknoloji-23977/

https://tr.wikipedia.org/wiki/Regresyon_analizi

https://www.scribbr.com/statistics/simple-linear-regression/#:~:text=Simple%20linear%20regression%20is%20a%20regression%20model%20that%20estimates%20the,Both%20variables%20should%20be%20quantitative.&text=Linear%20regression%20most%20often%20uses,the%20error%20of%20the%20model.

https://medium.com/@ekrem.hatipoglu/machine-learning-prediction-algorithms-multiple-linear-regression-part-3-c25a5a4205b6

https://www.scribbr.com/statistics/multiple-linear-regression/#:~:text=An%20introduction%20to%20multiple%20linear%20regression,-Published%20on%20February&text=Regression%20allows%20you%20to%20estimate,variables%20and%20one%20dependent%20variable.

https://medium.com/analytics-vidhya/understanding-polynomial-regression-5ac25b970e18