다음은 자녀의 학력, 연간 교육비, 연간 소득이 포함된 데이터이다. 자녀의 학력은 초등학교, 중학교, 고등학교로 분류 되어있으며, 가시성을 위해 각각 1, 2, 3 으로 표기하였다. 또한 화폐단위는 KRW 이고 현재 데이터의 개수는 50개 이다.

자녀 학력	연간 교육비	연간소득
1	1000000	32844348
1	2000000	37834645
2	1500000	40576828
2	2500000	45567125
1	1500000	34305348
3	3000000	52899822
3	2500000	47848803
2	1000000	37834645
1	500000	31122311
2	2000000	42818813
2	3000000	47848803
3	500000	36112652
1	2500000	39349331
2	2500000	45567125
1	2000000	34305348
1	1500000	34305348
3	1500000	42818813
2	2000000	42818813
1	2500000	39349331
2	500000	36112652
1	1000000	32844348

자녀 학력	연간 교육비	연간소득
3	1500000	42818813
3	2000000	47848803
1	3000000	42818813
1	500000	31122311
1	500000	31122311
2	2500000	45567125
2	1000000	37834645
1	2000000	34305348
1	2500000	39349331
3	500000	36112652
3	2000000	47848803
3	1500000	42818813
1	500000	31122311
2	1500000	40576828
2	3000000	47848803
1	1000000	32844348
1	1500000	34305348
2	2500000	45567125
1	2000000	34305348
1	500000	31122311

자녀 학력	연간 교육비	연간소득
2	1500000	40576828
2	2000000	42818813
2	1500000	40576828
1	2500000	39349331
3	3000000	52899822
3	2500000	47848803
1	1500000	34305348
2	1500000	40576828
1	500000	31122311
1	1000000	32844348
2	2000000	42818813
1	1500000	34305348
1	2500000	39349331
2	3000000	47848803
3	500000	36112652
3	3000000	52899822
1	500000	31122311
1	2000000	34305348
2	1000000	37834645

위 데이터를 활용하여 가족의 예상 연간 소득을 추정하는 것이 목표이다. 따라서 자녀의 학력과 연간 교육비를 입력값으로 받아 가족의 연간 소득을 출력값으로 분류한다.

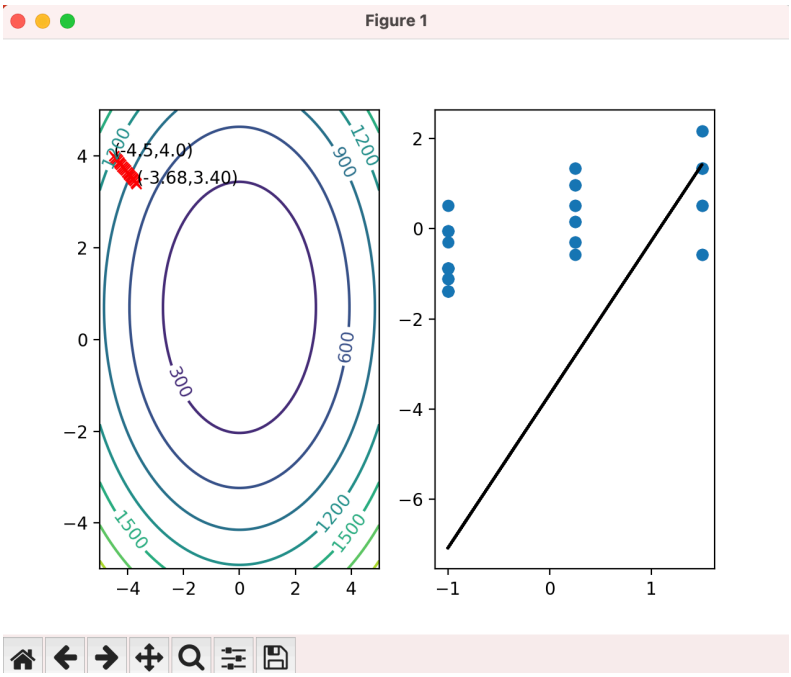
이를 위해 선택한 알고리즘은 Linear Square Method 이다.

다음 내용은 위 데이터값을 통해 Linear Square Method 를 구하여 등고선과 일차직선을 나타내는 Python 코드 이다.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from sklearn.preprocessing import StandardScaler

```

```python

def gradient_descent(alpha, x, y, numIterations):

    m = x.shape[0]  # number of samples
    x = np.c_[np.ones(m), x]  # reformat the input matrix x
    theta = np.array([-4.5, 4])  # initialize theta values
    lst_theta = [theta]  # list to hold all theta values
    x_transpose = x.transpose()
    for iter in range(numIterations):
        hypothesis = np.dot(x, theta)  # hypothesis function h(x) = theta_0 + theta_1*x
        loss = y - hypothesis  # difference between the actual and predicted values
        J = np.sum(loss ** 2) / (2 * m)  # cost function J(theta_0, theta_1)
        print("iter %s | J: %.3f" % (iter, J))  # print the cost after each iteration
        gradient = np.dot(x_transpose, loss) / m  # calculate the gradient
        theta = theta + alpha * gradient  # update theta values
        lst_theta.append(theta)  # append the updated theta values to the list

    return np.array(lst_theta)


def func(x, p1, p2):
    return p1 + x * p2  # linear function to fit the data


# normalize the data with mean and std

xx = np.loadtxt('/Users/iiki.kr/practice/datamining/Problems/data.txt',)  # load the data from the txt file
scaler = StandardScaler(copy=True, with_mean=True, with_std=True)  # create a scaler object
scaler.fit(xx)  # fit the scaler to the data
xx = scaler.transform(xx)  # scale the data

# least square method2
xdata = xx[:, 0]  # select the first column as x data
ydata = xx[:, 2]  # select the third column as y data

popt, pcov = curve_fit(func, xdata, ydata, p0=(50, 50))  # fit the data using curve_fit method
theta_0, theta_1 = popt  # extract the parameters
residuals = ydata - func(xdata, theta_0, theta_1)  # calculate the residuals
error = .5 * sum(residuals ** 2)  # calculate the error
print('theta=({0}, {1}), error = {2}'.format(theta_0, theta_1, error))  # print the parameters and error

theta_0 = np.linspace(-3, 3, num=100)
theta_1 = np.linspace(0, 5, num=100)
cost = np.zeros((theta_1.size, theta_0.size))  # initialize the cost matrix
for (x, p0) in enumerate(theta_0):
    for (y, p1) in enumerate(theta_1):
        cost[y, x] = .5 * sum((ydata - func(xdata, p0, p1)) ** 2)  # calculate the cost for all values of theta

theta = gradient_descent(0.01, xdata, ydata, 2000)  # find the parameters using gradient descent method
print(theta.shape)
print('Learned-parameters:', theta)


# plot the contour
plt.subplot(121)
cs = plt.contour(theta_0, theta_1, cost)
plt.plot(theta[:, 0], theta[:, 1], 'r-', marker='x')
plt.text(theta[0, 0], theta[0, 1], '(-4.5,4.0)')
str = "({0:.2f},{1:.2f})".format(theta[-1, 0], theta[-1, 1])
plt.text(theta[-1, 0], theta[-1, 1], str)
plt.clabel(cs)


# plot an estimate
### least square method 3
plt.subplot(122)
y_predict = theta[-1, 0] + theta[-1, 1] * xdata
plt.plot(xdata, ydata, 'o')
plt.plot(xdata, y_predict, 'k-')
plt.show()

```

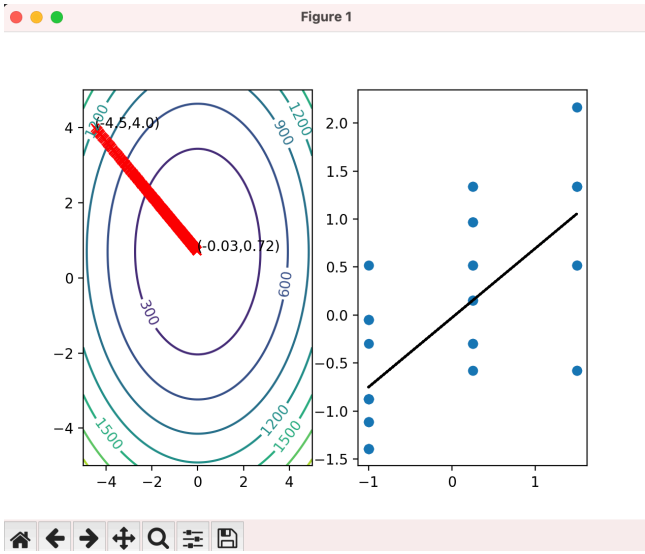
〈 코드 실행결과 〉



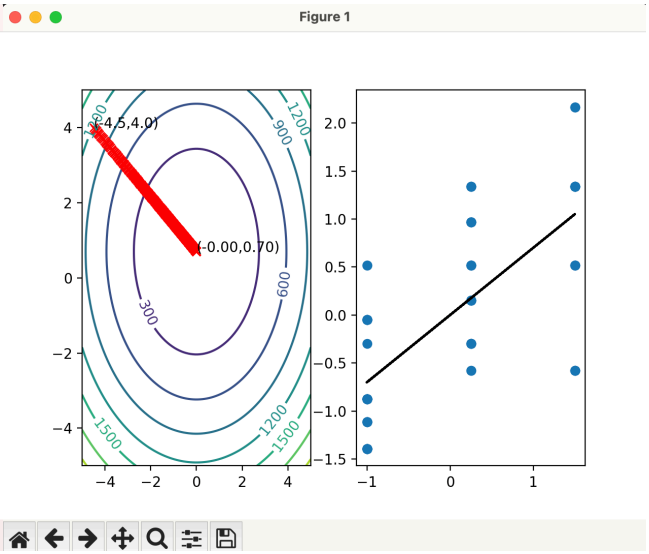
Linear Square Method 알고리즘을 이용한 등고선 그래프와 일차직선 그래프의 결과가 현재로써는 오차값이 높아 부정확하다고 판단하여, 정확한 결과값을 얻기 위해 다음과 같은 조치를 취했다.
Hyperparameter 값 중 numIterations 의 값을 20에서 500과 2000을 걸쳐서 조정한 후 결과를 다시 확인했다.

```
theta = gradient_descent(**0.01**, xdata, ydata, 500) #변경 전

#numIterations 값을 500 에서 2000 으로 조정

theta = gradient_descent(0.01, xdata, ydata, 2000) #변경 후
```

〈 numIterations = 500 〉

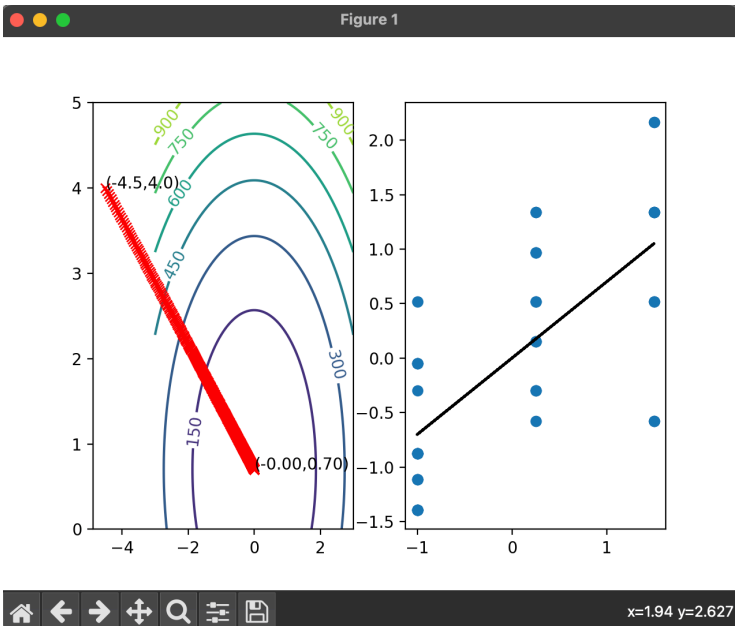


〈 numIterations = 2000 〉



등고선 그래프에서 각 축의 범위가 크면 등고선의 밀도가 낮아져 구체적 패턴을 파악하기 어렵다. 따라서 등고선 그래프의 해상도를 높여 구체적 결과를 얻을 수 있다.
기존에 설정된 x축의 범위를 (-5, 5)에서 (-3, 3)으로, y축의 범위를 (-5, 5)에서 (0, 5)으로 변경하였다.

```
theta_0 = np.linspace(-5, 5, num=100)
theta_1 = np.linspace(-5, 5, num=100) #변경 전

theta_0 = np.linspace(-3, 3, num=100)
theta_1 = np.linspace(0, 5, num=100) #변경 후
```



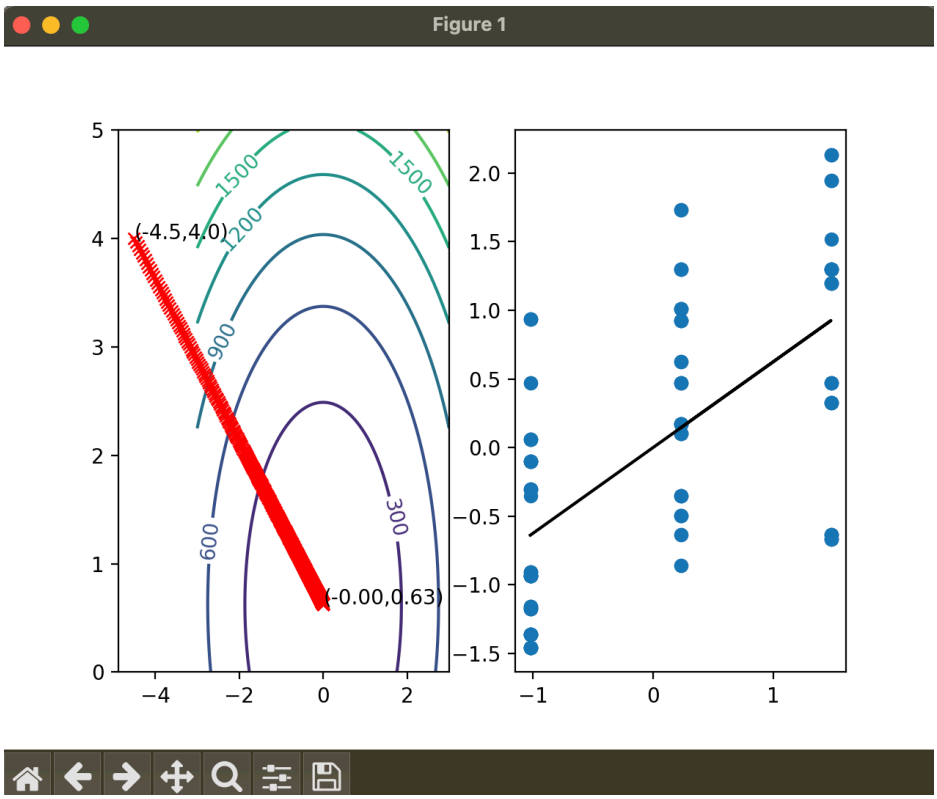
가족의 예상 연간소득의 정확성을 보다 개선하기 위해서 데이터의 양을 50개 추가하였다.

따라서 Linear Sqaure Method 는 총 100개의 데이터에 기반하여 작동할 것 이다.

자녀 학력	연간 교육비	연간소득
1	1500000	34450185
1	2000000	38111480
2	2500000	46082239
2	2000000	40994760
1	500000	31688885
1	2500000	40310846
3	2000000	47210402
1	1000000	32927535
2	2000000	40994760
3	1500000	41925820

자녀 학력	연간 교육비	연간소득
3	500000	35916440
2	1000000	36948000
1	3000000	45640000
2	3000000	50476000
1	500000	31688885
2	2500000	46082239
1	2000000	38111480
1	2500000	39349331
3	500000	36112652
3	2000000	47848803

자녀 학력	연간 교육비	연간소득
1	500000	31688885
2	2000000	40994760
2	500000	34759520
1	3000000	45640000
1	500000	31688885
2	3000000	50476000
2	1500000	43758449
3	2500000	49177183
2	2000000	40994760
1	1500000	34450185



〈최종 결과〉

이러한 개선 과정은 더욱 정교해질 수 있으며, 다양한 변수들을 고려할 수 있다. 예를 들어, 가족 구성원의 직업, 거주 지역, 학력 수준 등을 고려하여 보다 정확한 예측 결과를 도출할 수 있다. 또한, 이러한 예측 결과를 활용하여 가계의 재무 상황을 분석하고, 더욱 효율적인 가계 경제 관리 방법을 모색할 수 있다. 이를 통해 가족의 예상 연간소득뿐만 아니라, 가계의 재무 상황 전반에 대한 이해도를 높일 수 있다. 결과적으로, 자녀의 학력이 고학력일수록, 연간 교육비 지출이 높을수록, 가족의 예상 연간소득은 높아지는 경향이 있다는 것을 알 수 있다.