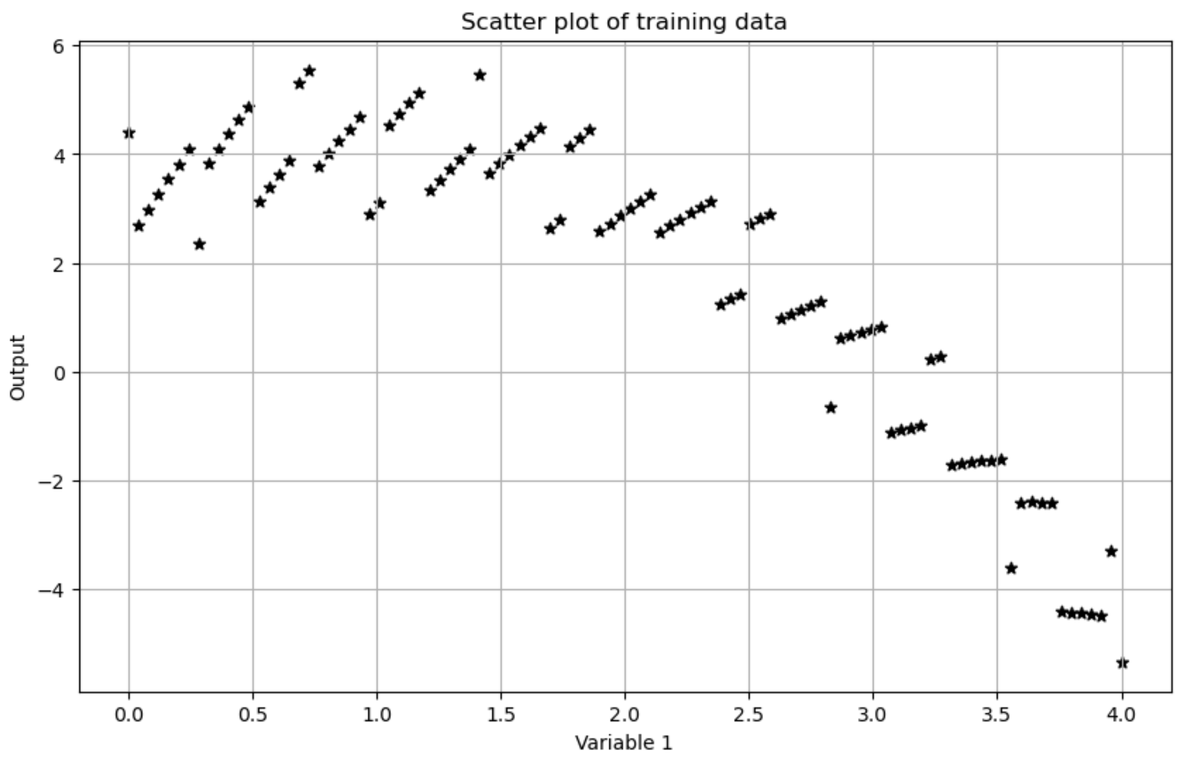
In [90]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [91]:
```
df = pd.read_csv('D3.csv')
X1 = df.values[:, 0]
X2 = df.values[:, 1]
X3 = df.values[:, 2]
y = df.values[:, 3]
len_y = len(y)
print('X1 = ', X1[: 5])
print('X2 = ', X2[: 5])
print('X3 = ', X3[: 5])
print('y = ', y[: 5])
print('length of y = ', len_y)
```

```
X1 =  [0.         0.04040404 0.08080808 0.12121212 0.16161616]
X2 =  [3.44        0.1349495  0.82989899 1.52484848 2.21979798]
X3 =  [0.44        0.88848485 1.3369697  1.78545454 2.23393939]
y =  [4.38754501 2.6796499  2.96848081 3.25406475 3.53637472]
length of y =  100
```

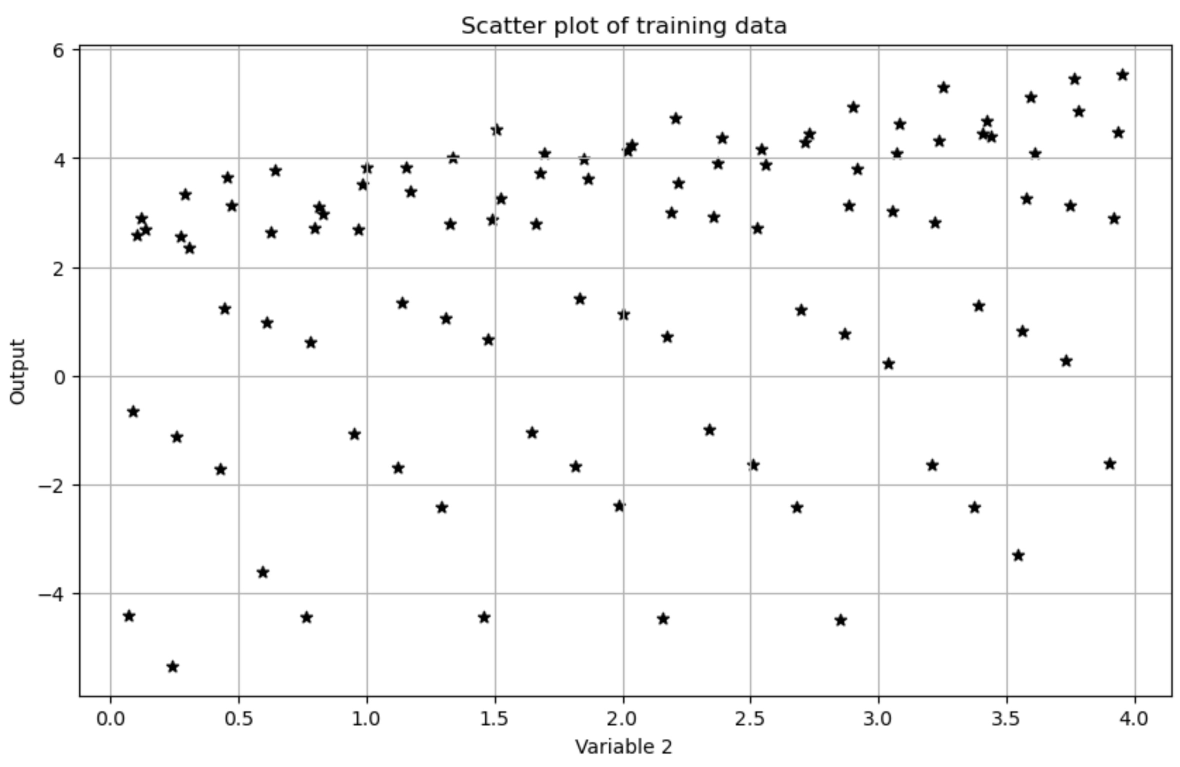
In [92]:
```
plt.scatter(X1,y, color='black',marker= '*')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('Variable 1')
plt.ylabel('Output')
plt.title('Scatter plot of training data')
```

Out[92]:
```
Text(0.5, 1.0, 'Scatter plot of training data')
```



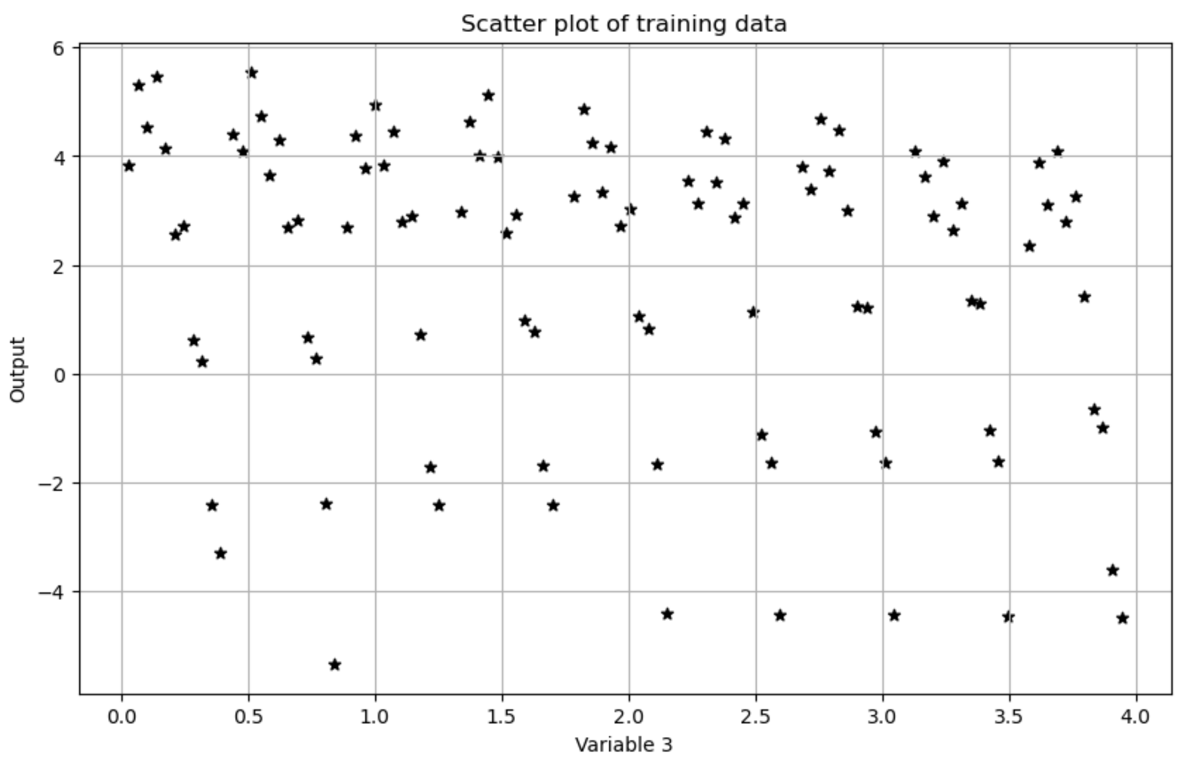
In [93]:
```
plt.scatter(X2,y, color='black',marker= '*')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('Variable 2')
plt.ylabel('Output')
plt.title('Scatter plot of training data')
```

Out[93]:
```
Text(0.5, 1.0, 'Scatter plot of training data')
```



In [94]:
```
plt.scatter(X3,y, color='black',marker= '*')
plt.grid()
plt.rcParams["figure.figsize"] = (10,6)
plt.xlabel('Variable 3')
plt.ylabel('Output')
plt.title('Scatter plot of training data')
```

Out[94]:
```
Text(0.5, 1.0, 'Scatter plot of training data')
```



In [95]:
```
X_0 = np.ones((m, 1))
X_0[:5]
```

```
Out[95]: array([[1.],
                [1.],
                [1.],
                [1.],
                [1.]])
```

```
In [96]: X_1 = X1.reshape(m, 1)
         X_2 = X2.reshape(m, 1)
         X_3 = X3.reshape(m, 1)
         print('X_1 = ', X_1[: 5])
         print('X_2 = ', X_2[: 5])
         print('X_3 = ', X_3[: 5])
```

```
X_1 =  [[0.        ]
 [0.04040404]
 [0.08080808]
 [0.12121212]
 [0.16161616]]
X_2 =  [[3.44      ]
 [0.1349495 ]
 [0.82989899]
 [1.52484848]
 [2.21979798]]
X_3 =  [[0.44      ]
 [0.88848485]
 [1.3369697 ]
 [1.78545454]
 [2.23393939]]
```

```
In [97]: X1 = np.hstack((X_0, X_1))
         X2 = np.hstack((X_0, X_2))
         X3 = np.hstack((X_0, X_3))
         print('X1 = ', X1[: 5])
         print('X2 = ', X2[: 5])
         print('X3 = ', X3[: 5])
```

```
X1 =  [[1.         0.        ]
 [1.         0.04040404]
 [1.         0.08080808]
 [1.         0.12121212]
 [1.         0.16161616]]
X2 =  [[1.         3.44      ]
 [1.         0.1349495 ]
 [1.         0.82989899]
 [1.         1.52484848]
 [1.         2.21979798]]
X3 =  [[1.         0.44      ]
 [1.         0.88848485]
 [1.         1.3369697 ]
 [1.         1.78545454]
 [1.         2.23393939]]
```

```
In [98]: theta = np.zeros(2)
         theta
```

```
Out[98]: array([0., 0.])
```

```
In [99]: def compute_cost(A, y, theta):
             predictions = A.dot(theta)
             errors = np.subtract(predictions, y)
             sqrErrors = np.square(errors)
             J = 1 / (2 * m) * np.sum(sqrErrors)
             return J
```

In [100…
```
def gradient_descent(A, y, theta, alpha, iterations):
    cost_history = np.zeros(iterations)
    for i in range(iterations):
        predictions = A.dot(theta)
        errors = np.subtract(predictions, y)
        sum_delta = (alpha / m) * A.transpose().dot(errors);
        theta = theta - sum_delta;
        cost_history[i] = compute_cost(A, y, theta)

    return theta, cost_history
```

In [101…
```
cost_1 = compute_cost(X1, y, theta)
cost_2 = compute_cost(X2, y, theta)
cost_3 = compute_cost(X3, y, theta)
print('The cost for given values of theta and Variable 1 =', cost_1)
print('The cost for given values of theta and Variable 2 =', cost_2)
print('The cost for given values of theta and Variable 3 =', cost_3)
```

```
The cost for given values of theta and Variable 1 = 5.524438459196242
The cost for given values of theta and Variable 2 = 5.524438459196242
The cost for given values of theta and Variable 3 = 5.524438459196242
```

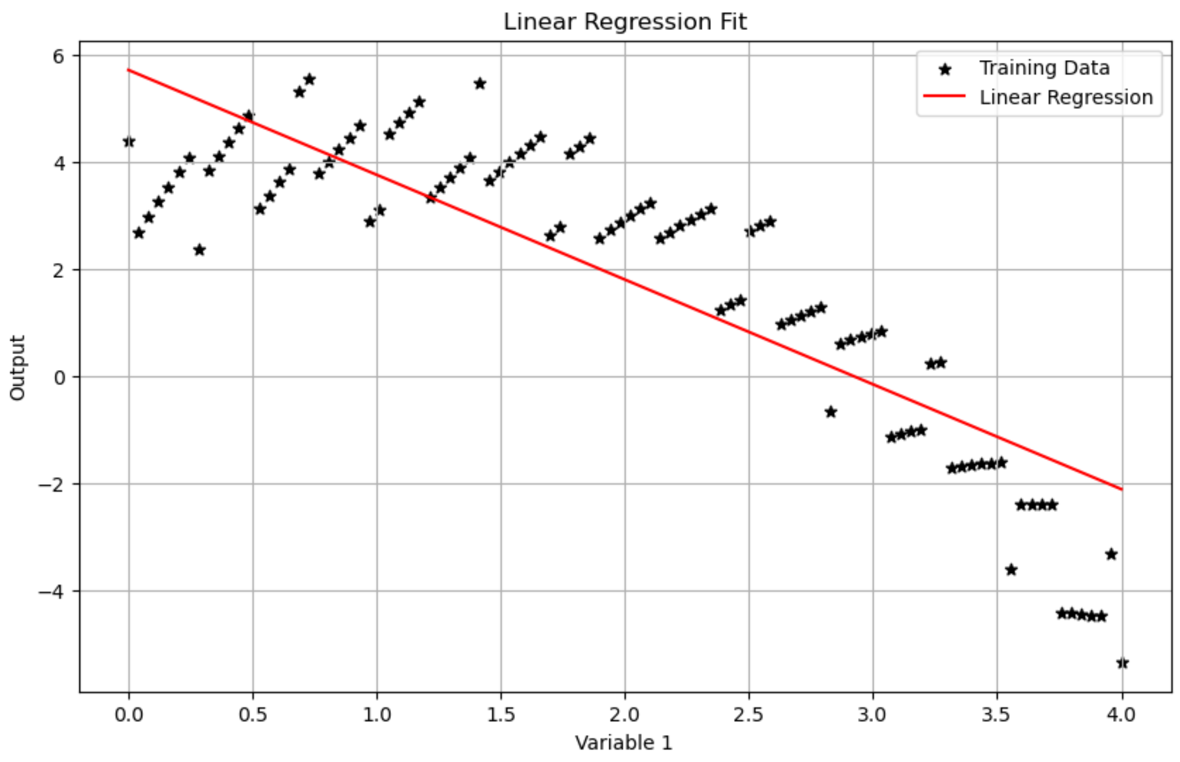
In [102…
```
theta = [0., 0.]
iterations = 1500;
alpha = 0.01
```

In [103…
```
theta, cost_history = gradient_descent(X1, y, theta, alpha, iterations)
print('Final value of theta (Variable 1)=', theta)
print('cost_history (Variable 1) =', cost_history)
```

```
Final value of theta (Variable 1)= [ 5.71850653 -1.9568206 ]
cost_history (Variable 1) = [5.48226715 5.44290965 5.40604087 ... 0.99063932 0.99061
433 0.99058944]
```

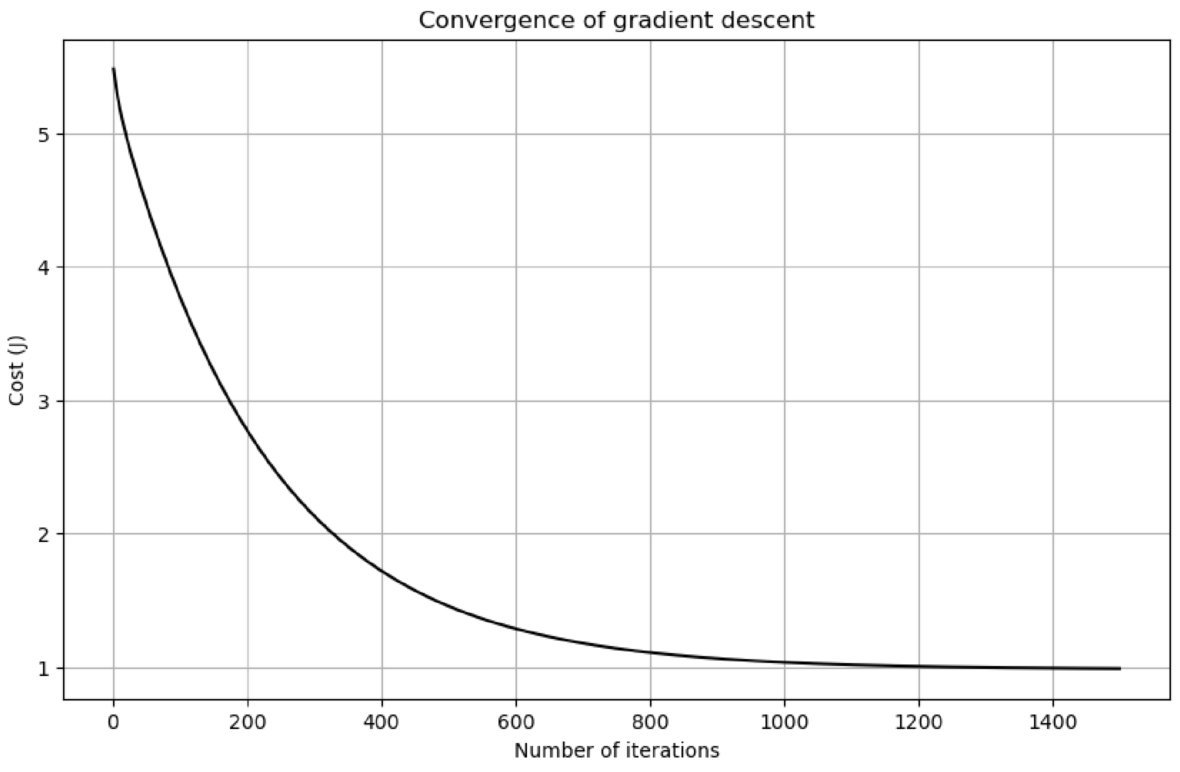
In [104…
```
plt.scatter(X1[:,1], y, color='black',marker= '*', label= 'Training Data')
plt.plot(X1[:,1],X1.dot(theta), color='red', label='Linear Regression')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Variable 1')
plt.ylabel('Output')
plt.title('Linear Regression Fit')
plt.legend()
```

Out[104]:
```
<matplotlib.legend.Legend at 0x215a2c74c40>
```



In [105…
```
plt.plot(range(1, iterations + 1),cost_history, color='black')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent')
```

Out[105]:
```
Text(0.5, 1.0, 'Convergence of gradient descent')
```



In [106…
```
theta, cost_history = gradient_descent(X2, y, theta, alpha, iterations)
print('Final value of theta (Variable 2)=', theta)
print('cost_history (Variable 2) =', cost_history)
```

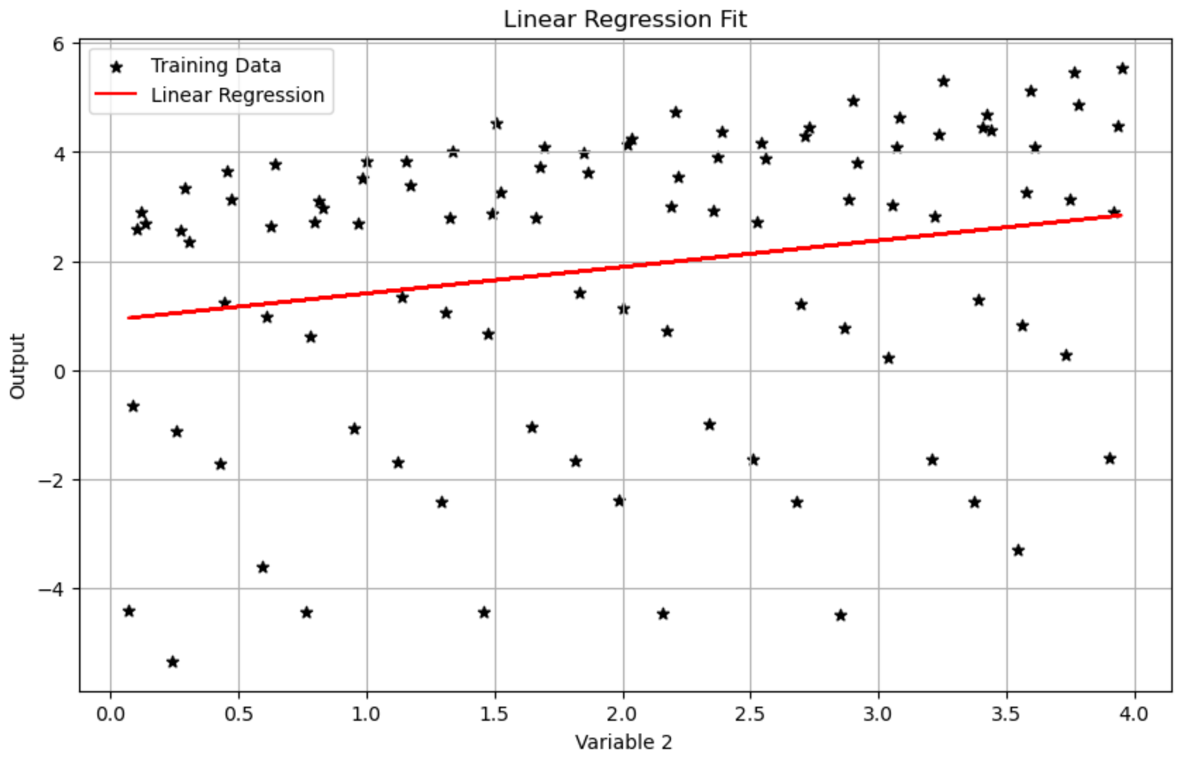
```
Final value of theta (Variable 2)= [0.92183754 0.48530143]
cost_history (Variable 2) = [7.78019751 7.67266456 7.57620204 ... 3.60380823 3.60378
856 3.60376899]
```

In [107…

```
plt.scatter(X2[:,1], y, color='black',marker= '*', label= 'Training Data')
plt.plot(X2[:,1],X2.dot(theta), color='red', label='Linear Regression')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Variable 2')
plt.ylabel('Output')
plt.title('Linear Regression Fit')
plt.legend()
```

Out[107]:

```
<matplotlib.legend.Legend at 0x215a311cc80>
```

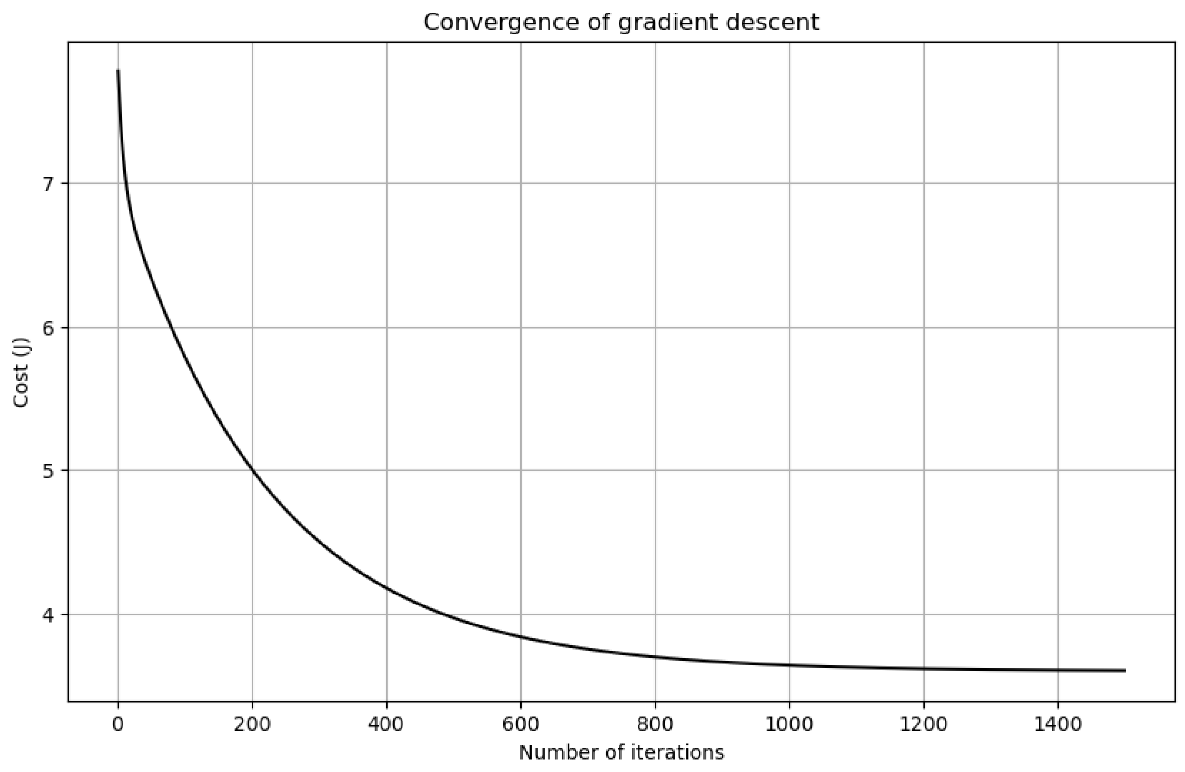


In [108…

```
plt.plot(range(1, iterations + 1),cost_history, color='black')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent')
```

Out[108]:

```
Text(0.5, 1.0, 'Convergence of gradient descent')
```

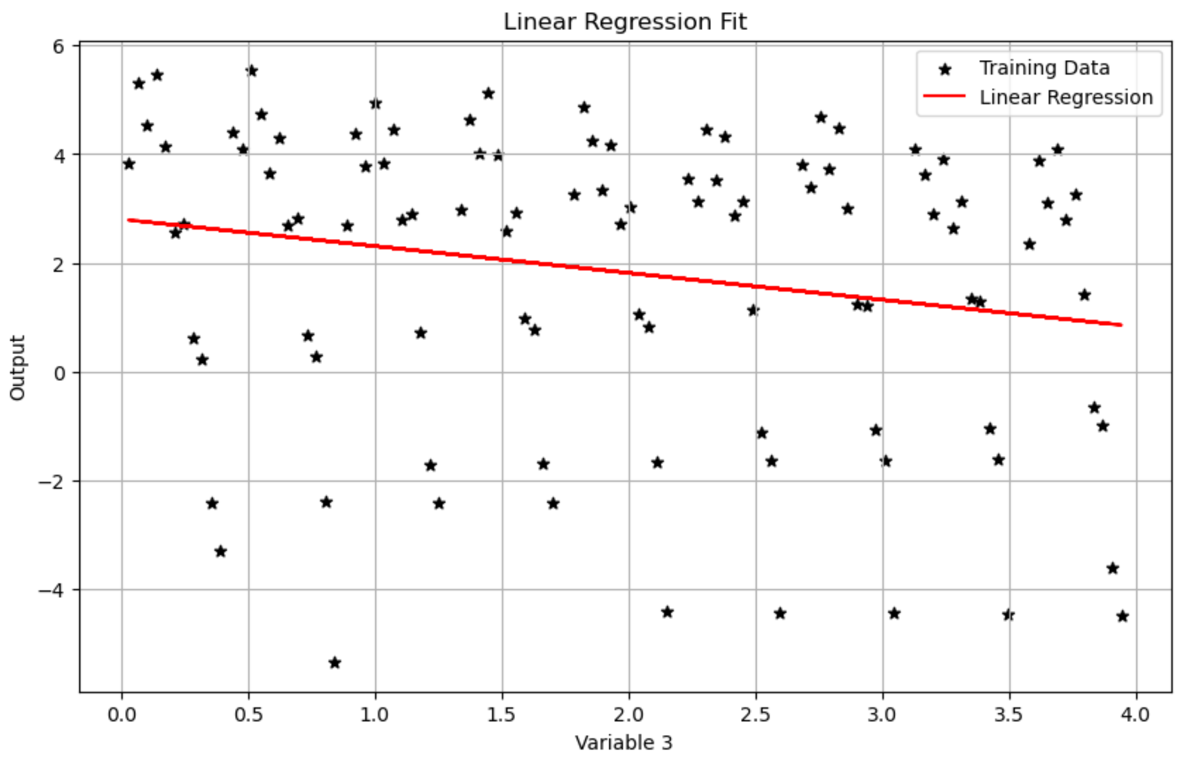


In [109…
```
theta, cost_history = gradient_descent(X3, y, theta, alpha, iterations)
print('Final value of theta (Variable 3)=', theta)
print('cost_history (Variable 3) =', cost_history)
```

```
Final value of theta (Variable 3)= [ 2.80205172 -0.49304729]
cost_history (Variable 3) = [4.28817888 4.27128079 4.25606956 ... 3.63008237 3.63007
954 3.63007671]
```

In [110…
```
plt.scatter(X3[:,1], y, color='black',marker= '*', label= 'Training Data')
plt.plot(X3[:,1],X3.dot(theta), color='red', label='Linear Regression')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Variable 3')
plt.ylabel('Output')
plt.title('Linear Regression Fit')
plt.legend()
```

Out[110]:
```
<matplotlib.legend.Legend at 0x215a37899a0>
```

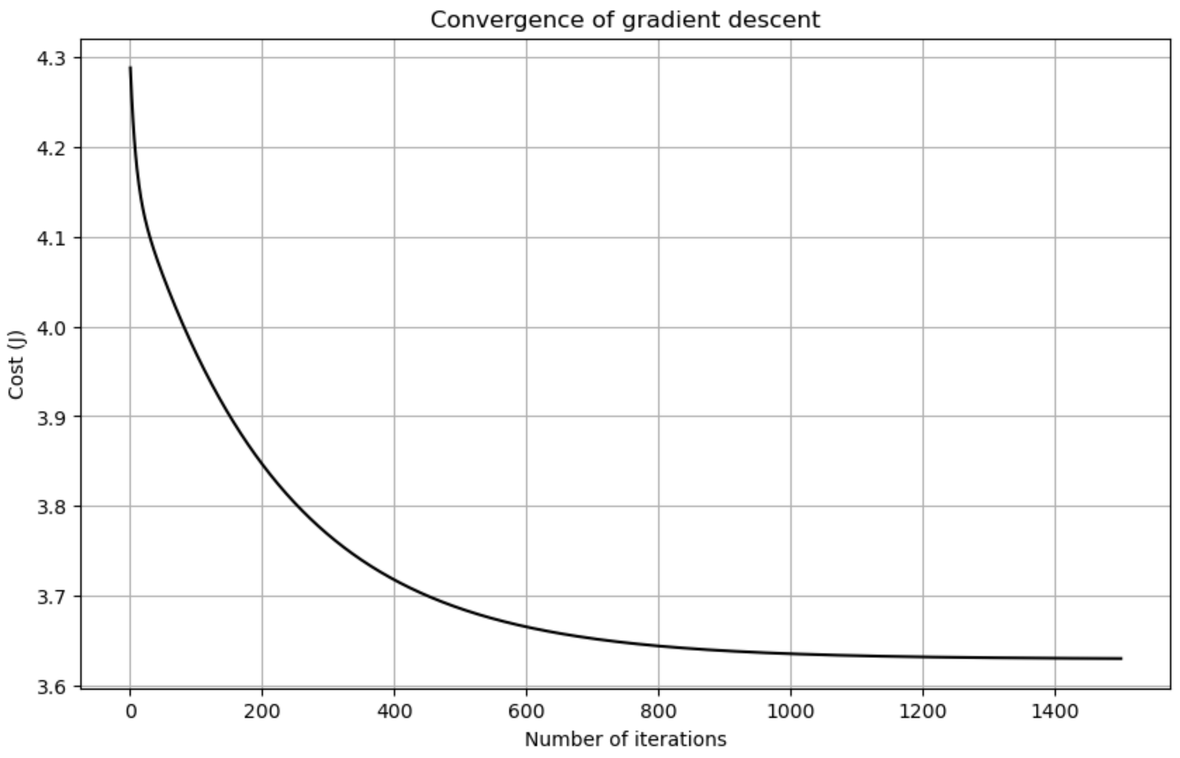


In [111…

```
plt.plot(range(1, iterations + 1),cost_history, color='black')
plt.rcParams["figure.figsize"] = (10,6)
plt.grid()
plt.xlabel('Number of iterations')
plt.ylabel('Cost (J)')
plt.title('Convergence of gradient descent')
```

Out[111]:

```
Text(0.5, 1.0, 'Convergence of gradient descent')
```


Convergence of gradient descent
Cost (J)
Number of iterations