ML ASSIGNMENT 3

KABIR CHATURVEDI , J011

IMPORTING:

In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

UNIVARIATE REGRESSION:

In [2]:
```python
data = pd.read_csv('ex1data1.txt',header=None)
```

In [3]:
```python
data.head()
```

Out[3]:

	0	1
0	6.1101	17.5920
1	5.5277	9.1302
2	8.5186	13.6620
3	7.0032	11.8540
4	5.8598	6.8233

In [4]:
```python
# NOW WE NAME THE COLUMNS
data.columns = ['Population','Profit']
```

In [5]:
```python
data.head()
```

Out[5]:

	Population	Profit
0	6.1101	17.5920
1	5.5277	9.1302
2	8.5186	13.6620
3	7.0032	11.8540
4	5.8598	6.8233

In [6]:
```python
data.isna().sum()
```

Out[6]:
```
Population    0
Profit        0
dtype: int64
```

In [7]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97 entries, 0 to 96
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Population  97 non-null     float64
 1   Profit      97 non-null     float64
dtypes: float64(2)
memory usage: 1.6 KB
```

In [8]:
```python
data.describe()
```

Out[8]:

	Population	Profit
count	97.000000	97.000000
mean	8.159800	5.839135
std	3.869884	5.510262
min	5.026900	-2.680700
25%	5.707700	1.986900
50%	6.589400	4.562300
75%	8.578100	7.046700
max	22.203000	24.147000

In [9]:
```python
X = data.iloc[:,:-1].values
y = data.iloc[:,-1].values.reshape(-1,1)
```

In [10]:
```python
X.shape, y.shape
```

Out[10]:
```
((97, 1), (97, 1))
```

In [11]:
```python
m = y.size
ones = np.ones((m,1)) # ADDING A MATRIX FULL OF ONES
X1 = np.c_[ones,X]
```

In [12]:
```python
X1.shape, y.shape
```

Out[12]:
```
((97, 2), (97, 1))
```

In [13]:
```python
n = X1.shape[1]
theta1 = np.zeros((n,1)) # ADDING A MATRIX FULL OF ZEROES
theta1.shape
```

Out[13]:
```
(2, 1)
```

In [14]:
```python
def calCost(error):
    m = error.shape[0]
    squared_error = np.dot(error.T,error)
    return squared_error[0][0]/m
```

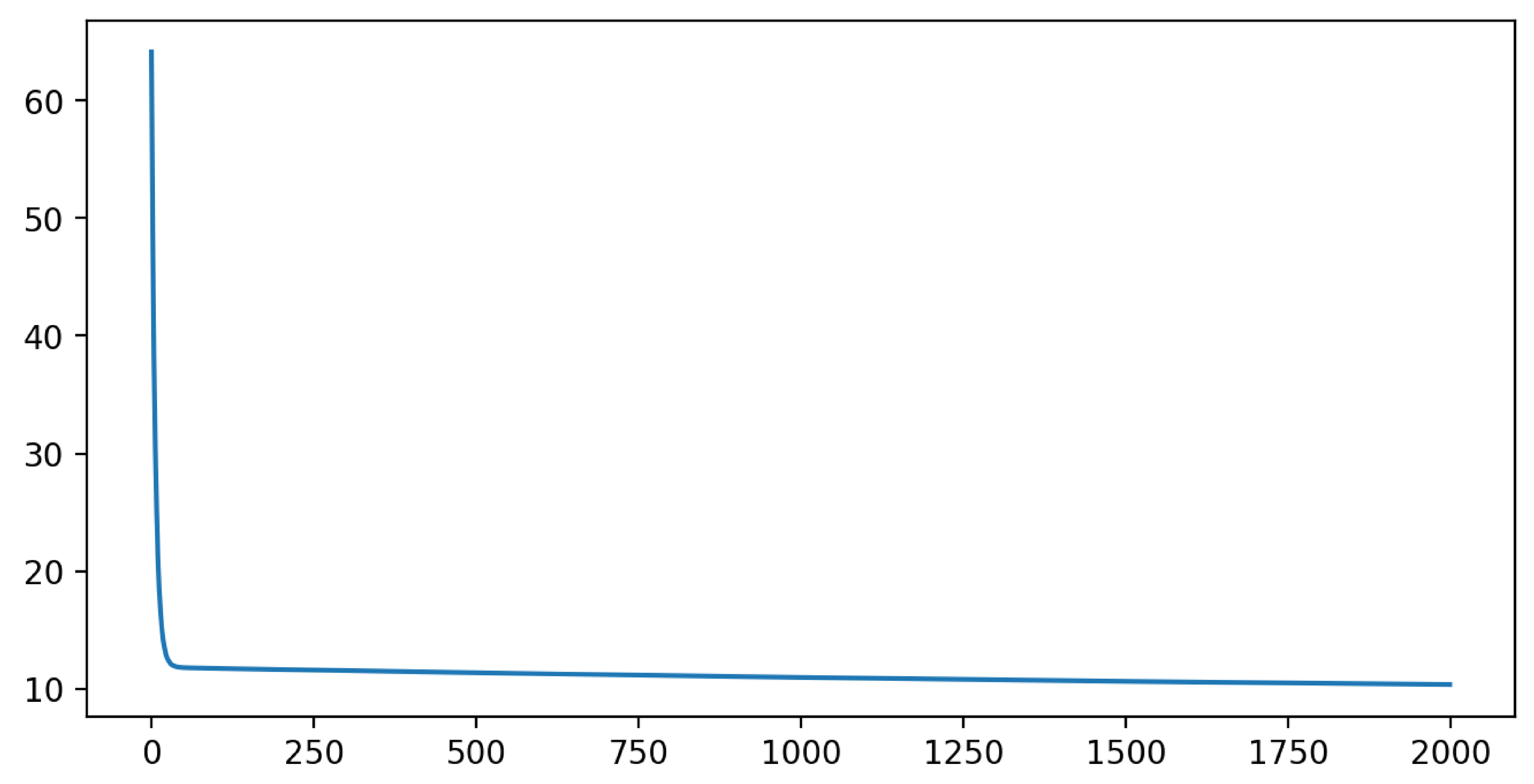
In [15]:
```python
def GradientDescent(X,y,theta,alpha,epochs):
    cost_history = []
    theta_history = [theta]
    m = y.size
    for i in range(epochs):
        h_theta = np.dot(X,theta)
        error = h_theta - y

        cost_history.append(calCost(error))
        diff = np.dot(X.T, error)/m
        theta = theta - (alpha*diff)
        theta_history.append(theta)
    return cost_history, theta_history
```

In [16]:
```python
cost_history, theta_history = GradientDescent(X=X1,y=y,theta=theta1,alpha=0.001,epochs=2000)
```

- **Plotting the cost history:**

In [17]:
```python
plt.figure(figsize=(8,4),dpi=200)
plt.plot(cost_history);
```



In [18]:
```python
theta_history[-1] # the latest value of theta
```

Out[18]:
```
array([[-1.12369018],
       [ 0.91454707]])
```

In [19]:
```python
from sklearn.linear_model import LinearRegression
```

In [20]:
```python
model = LinearRegression()
```

In [21]:
```python
model.fit(X1,y)
```

Out[21]:
```
LinearRegression()
```

In [22]:
```python
model.intercept_ , model.coef_
```

Out[22]:
```
(array([-3.89578088]), array([[0.        , 1.19303364]]))
```

TIME FOR MULTIVARIATE REGRESSION:

In [23]:
```python
new_data = pd.read_csv('ex1data2.txt',header=None)
```

In [24]:
```python
new_data.head()
```

Out[24]:

	0	1	2
0	2104	3	399900
1	1600	3	329900
2	2400	3	369000
3	1416	2	232000
4	3000	4	539900

In [25]:
```python
new_data.isna().sum()
```

Out[25]:
```
0    0
1    0
2    0
dtype: int64
```

In [26]:
```python
new_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47 entries, 0 to 46
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       47 non-null     int64
 1   1       47 non-null     int64
 2   2       47 non-null     int64
dtypes: int64(3)
memory usage: 1.2 KB
```

In [27]:
```python
new_data.describe()
```

Out[27]:

	0	1	2
count	47.000000	47.000000	47.000000
mean	2000.680851	3.170213	340412.659574
std	794.702354	0.760982	125039.899586
min	852.000000	1.000000	169900.000000
25%	1432.000000	3.000000	249900.000000
50%	1888.000000	3.000000	299900.000000
75%	2269.000000	4.000000	384450.000000
max	4478.000000	5.000000	699900.000000

In [28]:
```python
new_data.columns = ['Size of the house(Sq. Feet)', 'No. of Bedrooms', 'Price']
```

In [29]:
```python
new_data.head()
```

Out[29]:

	Size of the house(Sq. Feet)	No. of Bedrooms	Price
0	2104	3	399900
1	1600	3	329900
2	2400	3	369000
3	1416	2	232000
4	3000	4	539900

Creating a function to normalize the data:

In [30]:
```python
def normalize(dataframe):
    df=dataframe.copy()
    for col in df.columns:
        df[col] = (df[col]-df[col].mean())/df[col].std()
    return df
```

In [31]:
```python
new_data.columns
```

Out[31]:
```
Index(['Size of the house(Sq. Feet)', 'No. of Bedrooms', 'Price'], dtype='object')
```

In [32]:
```python
new_data = normalize(new_data)
```

In [33]:
```python
X2 = new_data.iloc[:,:-1].values
y2 = new_data.iloc[:,-1].values.reshape(-1,1)
```

In [34]:
```python
X2.shape, y2.shape
```

Out[34]:
```
((47, 2), (47, 1))
```

In [35]:
```python
m = y2.size
ones = np.ones((m,1)) # ADDING A MATRIX FULL OF ONES
X2 = np.c_[ones,X2]
```

In [36]:
```python
X2.shape, y2.shape
```

Out[36]:
```
((47, 3), (47, 1))
```

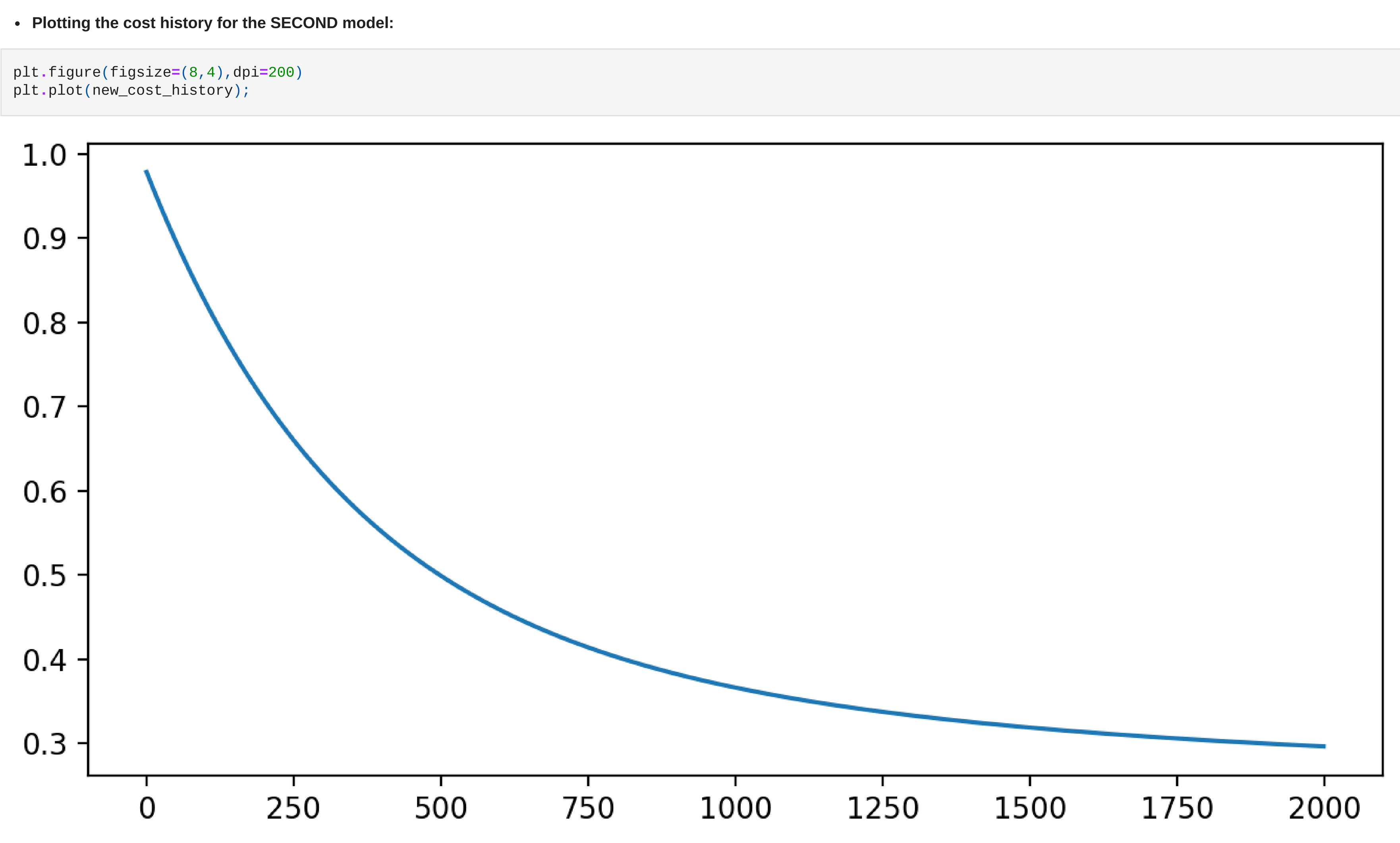
In [37]:
```python
n = X2.shape[1]
theta2 = np.zeros((n,1)) # ADDING A MATRIX FULL OF ZEROES
theta2.shape
```

Out[37]:
```
(3, 1)
```

In [38]:
```python
new_cost_history, new_theta_history = GradientDescent(X2,y2,theta2,alpha=0.001,epochs=2000)
```

- **Plotting the cost history for the SECOND model:**

In [40]:
```python
plt.figure(figsize=(8,4),dpi=200)
plt.plot(new_cost_history);
```



In [41]:
```python
new_theta_history[-1] #latest theta
```

Out[41]:
```
array([[-1.19667869e-16],
       [ 6.67041350e-01],
       [ 1.25393882e-01]])
```

In [42]:
```python
new_model = LinearRegression()
```

In [43]:
```python
new_model.fit(X2,y2)
```

Out[43]:
```
LinearRegression()
```

In [44]:
```python
new_model.intercept_ , new_model.coef_
```

Out[44]:
```
(array([-1.15685754e-16]), array([[ 0.        ,  0.88476599, -0.05317882]]))
```