In [83]:

```python
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

Multivariate Linear Regression

In [84]:

```python
data=pd.read_csv(r"C:\Users\ishan\Desktop\Year3\SEM_5\ML\dataset_A_3.csv")
data.head()
```

Out[84]:

	col1	col2	col3
0	2104	3	399900
1	1600	3	329900
2	2400	3	369000
3	1416	2	232000
4	3000	4	539900

In [85]:

```python
data.describe()
```

Out[85]:

	col1	col2	col3
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 [86]:

```python
plt.scatter(data['col1'], data['col3'])
plt.xticks(np.arange(5,30,step=5))
plt.yticks(np.arange(-5,30,step=5))
plt.xlabel('col1 (in 10,000s)')
plt.ylabel('col2 (in 10,000$)')
plt.title('col2 vs col1')
```

Out[86]:

```
Text(0.5, 1.0, 'col2 vs col1')
```

In [87]:

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

In [88]:

```
normallized_data = normalize(data)
normallized_data.head()
```

Out[88]:

	col1	col2	col3
0	0.130010	-0.223675	0.475747
1	-0.504190	-0.223675	-0.084074
2	0.502476	-0.223675	0.228626
3	-0.735723	-1.537767	-0.867025
4	1.257476	1.090417	1.595389

In [117]:

```
X = normallized_data.iloc[:,:-1].values
y = normallized_data.iloc[:,-1].values
```

In [118]:

```
m = y.size
n = data.shape[1]
```

In [119]:

```
y.shape
```

Out[119]:

```
(47,)
```

In [120]:

```
y = y.reshape(m,1)
y.shape
```

Out[120]:

```
(47, 1)
```

In [121]:

```
ones = np.ones((m,1))
X1 = np.concatenate((ones,X),axis=1)
X1[:5]
```

Out[121]:

```
array([[ 1.        ,  0.13000987, -0.22367519],
       [ 1.        , -0.50418984, -0.22367519],
       [ 1.        ,  0.50247636, -0.22367519],
       [ 1.        , -0.73572306, -1.53776691],
       [ 1.        ,  1.25747602,  1.09041654]])
```

In [122]:

```
alpha = 0.01
theta = np.random.rand(3,1)
epoch = 10000
```

In [123]:

```
def GD(X1,y,theta,epoch,alpha,decimal=5):
    past_cost = []
    past_theta = [theta]
    m = y.size
    n = X1.shape[1]
    for i in range(epoch):
        h_theta = np.dot(X1,theta)
        error = h_theta-y
        cost = np.dot(error.T, error)/(2*m)
        past_cost.append(cost[0][0])
        diff = np.dot(X1.T, error)/m
        theta = theta - (alpha*diff)
        past_theta.append(theta)
        if  np.equal(np.round(past_theta[i],decimals=decimals),np.round(past_theta[i+1],d
ecimals=decimals)).sum() == n:
            break
    return past_cost, past_theta, i+1
```

In [124]:

```
pastCost, pastTheta,stop_epoch = GD_j2(X1=X1, y=y, theta=theta, epoch=epoch,alpha=alpha)
```

In [127]:

```
print(f'the model performed {stop_epoch} epochs out of {epoch}')
```

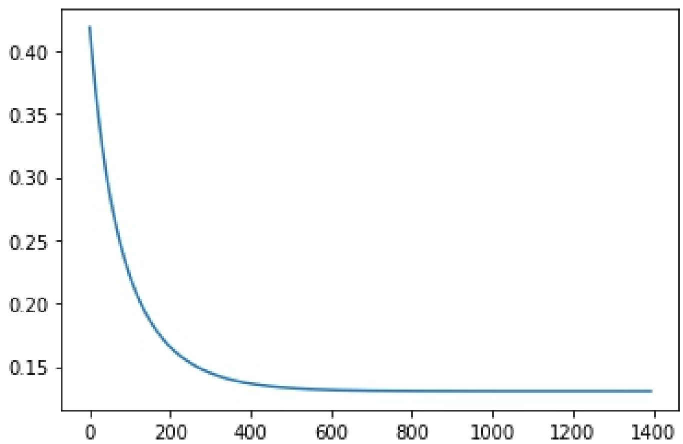
```
the model performed 1394 epochs out of 10000
```

In [128]:

```
plt.plot(pastCost)
```

Out[128]:

```
[<matplotlib.lines.Line2D at 0x22ca4afdf88>]
```



In [ ]:

In [ ]: