ML Submission 3

Multivariate Linear Regression

J042 AVANI NARVEKAR

imports

In [1]:

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import r2_score
import time
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
```

In [2]:

```
data_location = 'Downloads/ex1data2.txt'
data = pd.read_csv(data_location, header = None)
data.columns = 'Area','Bedrooms','Price'
data.head()
```

Out[2]:

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

CHECKING NULL VALUES

In [3]:

```
data.isnull().sum()
```

Out[3]:

```
Area        0
Bedrooms    0
Price       0
dtype: int64
```

In [4]:

```
data.describe()
```

Out[4]:

	Area	Bedrooms	Price
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

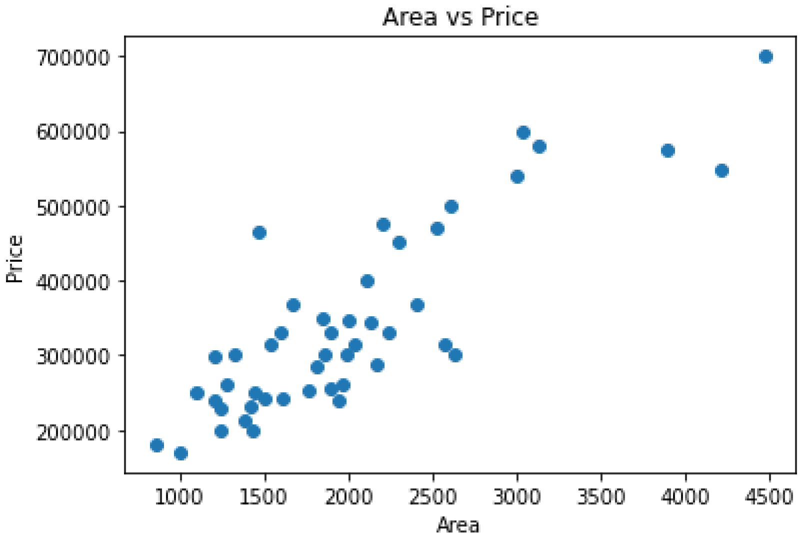
AREA VS PRICE PLOT

In [5]:

```
plt.scatter(data['Area'], data['Price'])
#plt.xticks(np.arange(5,30,step=5))
#plt.yticks(np.arange(-5,30,step=5))
plt.xlabel('Area')
plt.ylabel('Price')
plt.title('Area vs Price')
```

Out[5]:

```
Text(0.5, 1.0, 'Area vs Price')
```

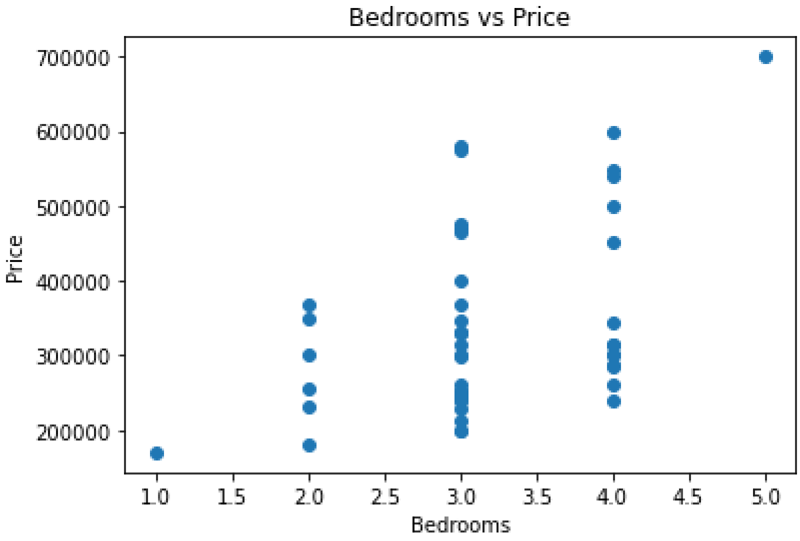


In [6]:

```
plt.scatter(data['Bedrooms'], data['Price'])
#plt.xticks(np.arange(5,30,step=5))
#plt.yticks(np.arange(-5,30,step=5))
plt.xlabel('Bedrooms')
plt.ylabel('Price')
plt.title('Bedrooms vs Price')
```

Out[6]:

```
Text(0.5, 1.0, 'Bedrooms vs Price')
```



SCALING

In [7]:

```
col = ['Area','Price']
scaler = MinMaxScaler()
data[col] = pd.DataFrame(scaler.fit_transform(data[col]),columns = data[col].columns)
```

In [8]:

```
data.head()
```

Out[8]:

	Area	Bedrooms	Price
0	0.345284	3	0.433962
1	0.206288	3	0.301887
2	0.426917	3	0.375660
3	0.155543	2	0.117170
4	0.592388	4	0.698113

SLICING THE DATASET

In [9]:

```
y= np.array(data['Price'][:-1])
X = np.array(data.drop('Price',axis = 1)[:-1])
X.shape, y.shape
```

Out[9]:

```
((46, 2), (46,))
```

RESHAPING

In [10]:

```
y = y.reshape(y.shape[0],1)
X = np.c_[np.ones(X.shape[0]),X]
X.shape, y.shape
```

Out[10]:

```
((46, 3), (46, 1))
```

In [11]:

```
theta = np.zeros((3,1))
theta.shape
```

Out[11]:

```
(3, 1)
```

computeCost, gradientDescent, predict functions

In [ ]:

```
def computeCost(X,y,theta):
    """
    Take in a numpy array X,y,theta and get cost function using theta as parameter in a linear regression model
    """
    m = len(y)
    predictions = X.dot(theta)
    square_err = (predictions - y)**2

    return 1/(m)*np.sum(square_err)
```

In [13]:

```
def gradientDescent(X,y,theta,alpha,num_iters):
    """
    Take numpy array for X,y,theta and update theta for every iteration of gradient steps
    
    return theta and the list of cost of theta during each iteration
    """
    
    m = len(y)
    J_history = []
    for i in range(num_iters):
        predictions = X.dot(theta)
        error = np.dot(X.transpose(), (predictions-y))
        descent = alpha * 1/m * error
        theta-=descent
        J_history.append(computeCost(X,y,theta))
    
    return theta, J_history
```

In [16]:

```
def predict(X,theta):
    predictions = np.dot(X,theta)
    return predictions
```

In [17]:

```
alpha = 0.01
num_iter = 50000
start = time.time()
theta , j_history = gradientDescent(X,y,theta,alpha,num_iter)
gd_time = time.time()-start
```

In [18]:

```
prediction = predict(X,theta)
```

RESULTING EQUATION

In [20]:

```
print(f"h(x) ={str(round(theta[0,0],2))}+{str(round(theta[1,0],2))}x1{str(round(theta[2,0],2))}x2")
```

```
h(x) =0.07+0.95x1-0.02x2
```

In [21]:

```
print(f'Accuracy: {round(r2_score(y,prediction)*100,3)}%')
```

```
Accuracy: 72.914%
```

PREDICTION

In [22]:

```
predict2 = predict(X[-1],theta)*1000000
```

In [24]:

```
print(f'predicted price:{predict2}')
```

```
predicted price:[268335.49803222]
```

In [25]:

```
data.tail(1)
```

Out[25]:

	Area	Bedrooms	Price
46	0.096801	3	0.131321

COST PLOT

In [26]:

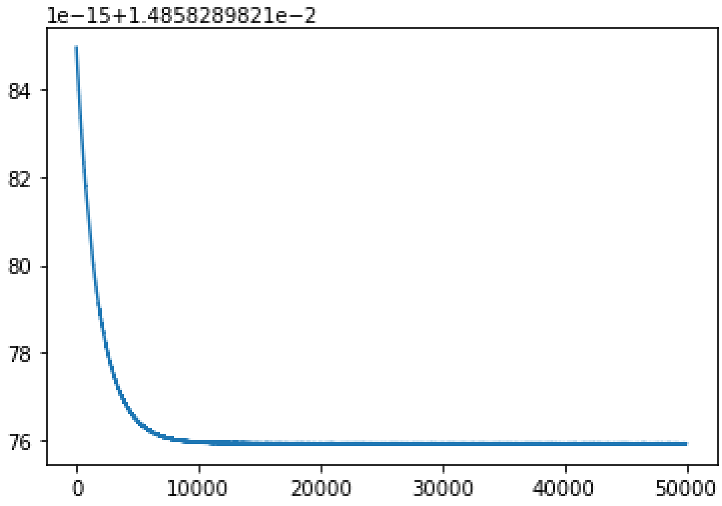
```
cost_arr = np.asarray(j_history)
cost_arr = cost_arr.reshape((cost_arr.shape[0],1))
cost_arr.shape
```

Out[26]:

```
(50000, 1)
```

In [27]:

```
plt.plot(cost_arr)
plt.show()
```



In [ ]:

```

```