ANALYSIS REPORT

DATASET - **Housing prices**

```
      price  lotsize  bedrooms  bathrms  ...  gashw airco garagepl prefarea
0   42000.0     5850         3        1  ...     no    no        1       no
1   38500.0     4000         2        1  ...     no    no        0       no
2   49500.0     3060         3        1  ...     no    no        0       no
3   60500.0     6650         3        1  ...     no    no        0       no
4   61000.0     6360         2        1  ...     no    no        0       no

[5 rows x 12 columns]
```

Given all the other 11 features , our task was to find out the prices.We have used three implementations for this.

- Normal Equation :

  $$h(\theta) = \theta_0 x_0 + \theta_1 x_1 + \ldots \theta_n x_n$$

  $$\theta = \left(X^T X\right)^{-1} . \left(X^T y\right)$$

  After implementing the equation on the data I got the following results:

  ```
  Co-efficient matrix after applying Normal Equation :
  [3.62636621e+00 1.39376008e+03 1.27236059e+04 6.60997094e+03
   5.16405029e+03 4.49862938e+03 5.17738242e+03 1.44745763e+04
   1.21882815e+04 4.40594192e+03 1.38408223e+04]

  Percentage Error from Normal Equation = 8.060706597718553
  ```

- Using gradient descent method
  Mainly these two equations are used , the value of theta is updated in each iteration as u can see in the equation below and J is the error function.

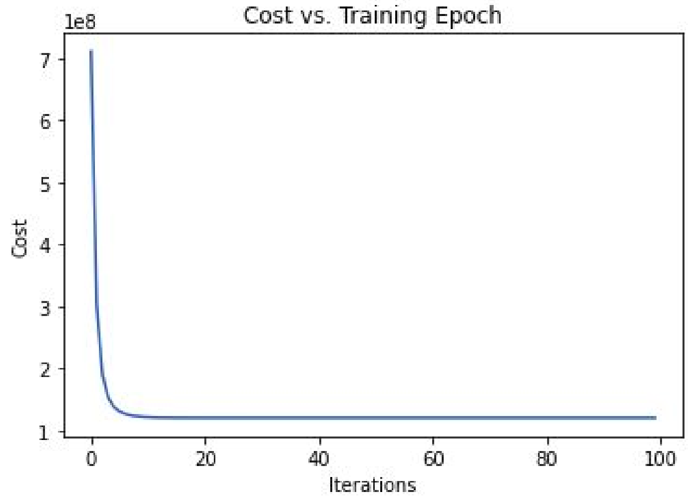
```
theta = theta - (alpha/len(trainX)) * np.sum(trainX * (trainX @
theta.T - trainY), axis=0)
```

```
J=np.power(((trainX.dot(theta.T))-trainY),2)
```

```
Percentage error =16.301718455861945

Optimal Theta = [[ 8013.53766164  1292.30903077  6478.175517    5773.5513855
   1947.89830493  1695.44254345  2519.87501107  3012.04466102
   5653.88417102  3730.88025634  5729.37866692 68902.13755875]]

[<matplotlib.lines.Line2D at 0x7f95f14ecda0>]
```



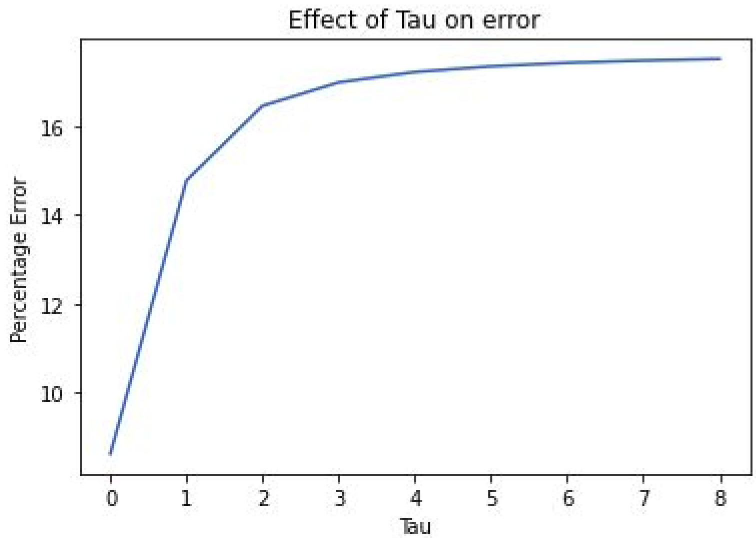
Therefore more the iterations , more the accurate value of theta and less the cost.

- Locally Weighted Regression

  This uses the following equation :
  The error function J is modified as it is multiplied by a weight factor where w is the weight associated with the training pt x.

  W = `np.exp(np.sum((X-X[testXInd])**2,axis=1)/(-2*tau**2))`

  Here value of tau is very crucial . It is called the bandwidth parameter . You can see in the graph.


Effect of Tau on error
Percentage Error
10
12
14
16
0
1
2
3
4
5
6
7
8
Tau