**9.** **Implement the non-parametric Locally Weighted Regression algorithm in order**

**to fit data points. Select appropriate data set for your experiment and draw graphs.**

THEORY: Linear regression is a supervised learning algorithm used for computing linear relationships between input (X) and output (Y).

Locally weighted linear regression is a non-parametric algorithm, that is, the model does not learn a fixed set of parameters as is done in ordinary linear regression. Rather parameters \theta are computed individually for each query point x. While computing \theta, a higher “preference” is given to the points in the training set lying in the vicinity of x than the points lying far away from x.

* This algorithm is used for making predictions when there exists a non-linear relationship between the features.
* Locally weighted linear regression is a supervised learning algorithm.
* It a non-parametric algorithm.
* doneThere exists No training phase. All the work is done during the testing phase/while making predictions.

The modified cost function is: J(\theta) = $\sum\_{i=1}^{m} w^{(i)}(\theta^Tx^{(i)} - y^{(i)})^2

where, w^{(i)} is a non-negative “weight” associated with training point x^{(i)}.

For x^{(i)}s lying closer to the query point x, the value of w^{(i)} is large, while for x^{(i)}s lying far away from x the value of w^{(i)} is small.

A typical choice of w^{(i)} is: w^{(i)} = exp(\frac{-(x^{(i)} - x)^2}{2\tau^2})

where, \tau is called the bandwidth parameter and controls the rate at which w^{(i)} falls with distance from x

Clearly, if |x^{(i)} - x| is small w^{(i)} is close to 1 and if |x^{(i)} - x| is large w^{(i)} is close to 0.

Thus, the training-set-points lying closer to the query point x contribute more to the cost J(\theta) than the points lying far away from x.

**PROCEDURE/PROGRAM:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

**def** local\_regression(x0,X,y,tau):

    x0=np.r\_[1,x0]

    X=np.c\_[np.ones(len(X)),X]

    xw=X.T\*radial\_kernel(x0,X,tau)

    beta=np.linalg.pinv(xw @ X) @ xw @ y

    return x0 @ beta

print(np.r\_[np.array([1,2,3]),0,0,0,np.array([4,5,6])])

print(np.c\_[np.array([1,2,3]),np.array([4,5,6])])

[1 2 3 0 0 0 4 5 6]

[[1 4]

[2 5]

[3 6]]

**def** radial\_kernel(x0,X,tau):

    return np.exp(np.sum((X-x0)\*\*2,axis=1)/(-2\*tau\*tau))

data=pd.read\_csv("tips.csv")

print(data)

total\_bill tip sex smoker day time size

0 16.99 1.01 Female No Sun Dinner 2

1 10.34 1.66 Male No Sun Dinner 3

2 21.01 3.50 Male No Sun Dinner 3

3 23.68 3.31 Male No Sun Dinner 2

4 24.59 3.61 Female No Sun Dinner 4

5 25.29 4.71 Male No Sun Dinner 4

6 8.77 2.00 Male No Sun Dinner 2

7 26.88 3.12 Male No Sun Dinner 4

8 15.04 1.96 Male No Sun Dinner 2

9 14.78 3.23 Male No Sun Dinner 2

10 10.27 1.71 Male No Sun Dinner 2

11 35.26 5.00 Female No Sun Dinner 4

12 15.42 1.57 Male No Sun Dinner 2

13 18.43 3.00 Male No Sun Dinner 4

14 14.83 3.02 Female No Sun Dinner 2

15 21.58 3.92 Male No Sun Dinner 2

16 10.33 1.67 Female No Sun Dinner 3

17 16.29 3.71 Male No Sun Dinner 3

18 16.97 3.50 Female No Sun Dinner 3

19 20.65 3.35 Male No Sat Dinner 3

20 17.92 4.08 Male No Sat Dinner 2

21 20.29 2.75 Female No Sat Dinner 2

22 15.77 2.23 Female No Sat Dinner 2

23 39.42 7.58 Male No Sat Dinner 4

24 19.82 3.18 Male No Sat Dinner 2

25 17.81 2.34 Male No Sat Dinner 4

26 13.37 2.00 Male No Sat Dinner 2

27 12.69 2.00 Male No Sat Dinner 2

28 21.70 4.30 Male No Sat Dinner 2

29 19.65 3.00 Female No Sat Dinner 2

.. ... ... ... ... ... ... ...

214 28.17 6.50 Female Yes Sat Dinner 3

215 12.90 1.10 Female Yes Sat Dinner 2

216 28.15 3.00 Male Yes Sat Dinner 5

217 11.59 1.50 Male Yes Sat Dinner 2

218 7.74 1.44 Male Yes Sat Dinner 2

219 30.14 3.09 Female Yes Sat Dinner 4

220 12.16 2.20 Male Yes Fri Lunch 2

221 13.42 3.48 Female Yes Fri Lunch 2

222 8.58 1.92 Male Yes Fri Lunch 1

223 15.98 3.00 Female No Fri Lunch 3

224 13.42 1.58 Male Yes Fri Lunch 2

225 16.27 2.50 Female Yes Fri Lunch 2

226 10.09 2.00 Female Yes Fri Lunch 2

227 20.45 3.00 Male No Sat Dinner 4

228 13.28 2.72 Male No Sat Dinner 2

229 22.12 2.88 Female Yes Sat Dinner 2

230 24.01 2.00 Male Yes Sat Dinner 4

231 15.69 3.00 Male Yes Sat Dinner 3

232 11.61 3.39 Male No Sat Dinner 2

233 10.77 1.47 Male No Sat Dinner 2

234 15.53 3.00 Male Yes Sat Dinner 2

235 10.07 1.25 Male No Sat Dinner 2

236 12.60 1.00 Male Yes Sat Dinner 2

237 32.83 1.17 Male Yes Sat Dinner 2

238 35.83 4.67 Female No Sat Dinner 3

239 29.03 5.92 Male No Sat Dinner 3

240 27.18 2.00 Female Yes Sat Dinner 2

241 22.67 2.00 Male Yes Sat Dinner 2

242 17.82 1.75 Male No Sat Dinner 2

243 18.78 3.00 Female No Thur Dinner 2

[244 rows x 7 columns]

bill=data.total\_bill.values

print(bill)

[16.99 10.34 21.01 23.68 24.59 25.29 8.77 26.88 15.04 14.78 10.27 35.26

15.42 18.43 14.83 21.58 10.33 16.29 16.97 20.65 17.92 20.29 15.77 39.42

19.82 17.81 13.37 12.69 21.7 19.65 9.55 18.35 15.06 20.69 17.78 24.06

16.31 16.93 18.69 31.27 16.04 17.46 13.94 9.68 30.4 18.29 22.23 32.4

28.55 18.04 12.54 10.29 34.81 9.94 25.56 19.49 38.01 26.41 11.24 48.27

20.29 13.81 11.02 18.29 17.59 20.08 16.45 3.07 20.23 15.01 12.02 17.07

26.86 25.28 14.73 10.51 17.92 27.2 22.76 17.29 19.44 16.66 10.07 32.68

15.98 34.83 13.03 18.28 24.71 21.16 28.97 22.49 5.75 16.32 22.75 40.17

27.28 12.03 21.01 12.46 11.35 15.38 44.3 22.42 20.92 15.36 20.49 25.21

18.24 14.31 14. 7.25 38.07 23.95 25.71 17.31 29.93 10.65 12.43 24.08

11.69 13.42 14.26 15.95 12.48 29.8 8.52 14.52 11.38 22.82 19.08 20.27

11.17 12.26 18.26 8.51 10.33 14.15 16. 13.16 17.47 34.3 41.19 27.05

16.43 8.35 18.64 11.87 9.78 7.51 14.07 13.13 17.26 24.55 19.77 29.85

48.17 25. 13.39 16.49 21.5 12.66 16.21 13.81 17.51 24.52 20.76 31.71

10.59 10.63 50.81 15.81 7.25 31.85 16.82 32.9 17.89 14.48 9.6 34.63

34.65 23.33 45.35 23.17 40.55 20.69 20.9 30.46 18.15 23.1 15.69 19.81

28.44 15.48 16.58 7.56 10.34 43.11 13. 13.51 18.71 12.74 13. 16.4

20.53 16.47 26.59 38.73 24.27 12.76 30.06 25.89 48.33 13.27 28.17 12.9

28.15 11.59 7.74 30.14 12.16 13.42 8.58 15.98 13.42 16.27 10.09 20.45

13.28 22.12 24.01 15.69 11.61 10.77 15.53 10.07 12.6 32.83 35.83 29.03

27.18 22.67 17.82 18.78]

tip=data.tip.values

print(tip)

[ 1.01 1.66 3.5 3.31 3.61 4.71 2. 3.12 1.96 3.23 1.71 5.

1.57 3. 3.02 3.92 1.67 3.71 3.5 3.35 4.08 2.75 2.23 7.58

3.18 2.34 2. 2. 4.3 3. 1.45 2.5 3. 2.45 3.27 3.6

2. 3.07 2.31 5. 2.24 2.54 3.06 1.32 5.6 3. 5. 6.

2.05 3. 2.5 2.6 5.2 1.56 4.34 3.51 3. 1.5 1.76 6.73

3.21 2. 1.98 3.76 2.64 3.15 2.47 1. 2.01 2.09 1.97 3.

3.14 5. 2.2 1.25 3.08 4. 3. 2.71 3. 3.4 1.83 5.

2.03 5.17 2. 4. 5.85 3. 3. 3.5 1. 4.3 3.25 4.73

4. 1.5 3. 1.5 2.5 3. 2.5 3.48 4.08 1.64 4.06 4.29

3.76 4. 3. 1. 4. 2.55 4. 3.5 5.07 1.5 1.8 2.92

2.31 1.68 2.5 2. 2.52 4.2 1.48 2. 2. 2.18 1.5 2.83

1.5 2. 3.25 1.25 2. 2. 2. 2.75 3.5 6.7 5. 5.

2.3 1.5 1.36 1.63 1.73 2. 2.5 2. 2.74 2. 2. 5.14

5. 3.75 2.61 2. 3.5 2.5 2. 2. 3. 3.48 2.24 4.5

1.61 2. 10. 3.16 5.15 3.18 4. 3.11 2. 2. 4. 3.55

3.68 5.65 3.5 6.5 3. 5. 3.5 2. 3.5 4. 1.5 4.19

2.56 2.02 4. 1.44 2. 5. 2. 2. 4. 2.01 2. 2.5

4. 3.23 3.41 3. 2.03 2.23 2. 5.16 9. 2.5 6.5 1.1

3. 1.5 1.44 3.09 2.2 3.48 1.92 3. 1.58 2.5 2. 3.

2.72 2.88 2. 3. 3.39 1.47 3. 1.25 1. 1.17 4.67 5.92

2. 2. 1.75 3. ]

tau=10

ypred=np.array([local\_regression(x0,bill,tip,tau) for x0 in bill])

print("YPRED",ypred)

**YPRED [2.7268086 1.9373207 3.16357671 3.42997873 3.51633213 3.58128485**

**1.74575924 3.72438753 2.50176793 2.47124009 1.92878673 4.43083574**

**2.54618056 2.88785842 2.4771194 3.2220979 1.93610165 2.64687501**

**2.72453903 3.12615891 2.83136711 3.08839096 2.58686061 4.82568169**

**3.03856248 2.81910225 2.30391956 2.22231784 3.2343044 3.02039613**

**1.8409488 2.87903818 2.50411159 3.1303338 2.81575242 3.46630673**

**2.64917273 2.71999731 2.91641655 4.09613433 2.61808699 2.77989264**

**2.37189436 1.85681398 4.02415732 2.87241286 3.28774109 4.18945095**

**3.86895131 2.84471472 2.20425145 1.93122515 4.39190117 1.88853866**

**3.60600781 3.00322967 4.68304419 3.68269381 2.0468769 6.10051242**

**3.08839096 2.35642868 2.02013067 2.87241286 2.79448878 3.06619949**

**2.66523434 1.06133091 3.08206252 2.49825118 2.14145993 2.73587829**

**3.72262312 3.58036571 2.46535676 1.9580398 2.83136711 3.7525037**

**3.34040171 2.76074818 2.99785156 2.68925208 1.90439686 4.21263647**

**2.61115978 4.39362136 2.26318279 2.8713078 3.52755625 3.17906333**

**3.9045567 3.31367154 1.37936878 2.6503213 3.33941529 4.90638073**

**3.75949977 2.14266965 3.16357671 2.19460712 2.06023987 2.54151737**

**5.42942362 3.30670864 3.15425528 2.53918471 3.10941617 3.57392472**

**2.86688515 2.41578298 2.37902458 1.56067946 4.68889724 3.45583063**

**3.61966545 2.76300381 3.98507723 1.97509508 2.19098893 3.468208**

**2.10149563 2.30989875 2.40986358 2.60769359 2.19701877 3.97423349**

**1.71526491 2.44060371 2.06388307 3.34631445 2.95894005 3.08628255**

**2.03836958 2.17047031 2.86909695 1.71404542 1.93610165 2.39682809**

**2.61346962 2.27877438 2.78101677 4.34832366 5.02235154 3.73935081**

**2.66294224 1.69453712 2.91093747 2.12330458 1.86901685 1.59226714**

**2.38733683 2.27517802 2.75736308 3.51258242 3.03322728 3.97840611**

**6.08109332 3.55452765 2.3063116 2.6698161 3.2139382 2.21870635**

**2.63767619 2.35642868 2.78551106 3.5097674 3.13762944 4.13245304**

**1.96778656 1.97265906 6.64501673 2.59149542 1.56067946 4.14400888**

**2.70749001 4.2308908 2.82802494 2.43588109 1.84705095 4.37645923**

**4.37817153 3.39617323 5.5884145 3.38060805 4.94870736 3.1303338**

**3.15218085 4.02913406 2.85692022 3.37377624 2.57758204 3.03749596**

**3.85958186 2.55317002 2.68011314 1.59834591 1.9373207 5.26315703**

**2.25958205 2.32065357 2.91860646 2.22833498 2.25958205 2.65950258**

**3.11360832 2.66752563 3.69871905 4.75450408 3.48621694 2.23074113**

**3.99590512 3.63598288 6.11223285 2.29195215 3.83650081 2.24757246**

**3.83478624 2.08936953 1.62023977 4.00256103 2.15838857 2.30989875**

**1.72258234 2.61115978 2.30989875 2.6445765 1.90683627 3.10521971**

**2.29314942 3.27671427 3.46154888 2.57758204 2.0917953 1.98970783**

**2.55898966 1.90439686 2.21148067 4.22507855 4.4809218 3.90962252**

**3.75075265 3.331514 2.8202184 2.92626333]**

SortIndex=bill.argsort(0)

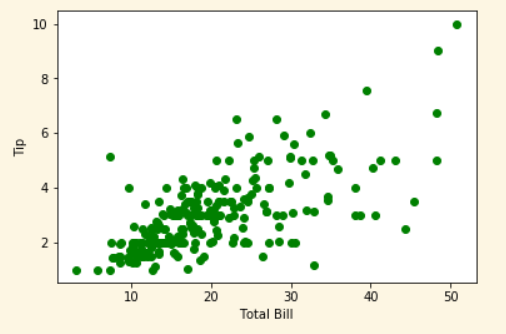
xsort=bill[SortIndex]

plt.scatter(bill,tip,color='green')

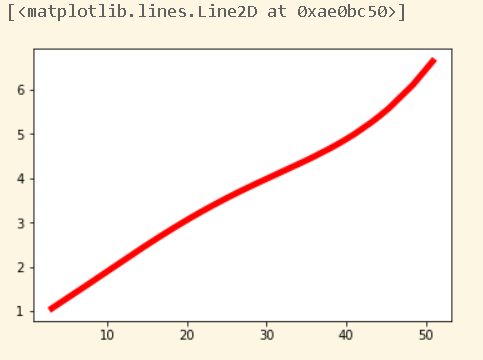
plt.xlabel("Total Bill")

plt.ylabel("Tip")

plt.show()



plt.plot(xsort,ypred[SortIndex] ,color ='red',linewidth=5)



plt.scatter(bill,tip,color='green')

plt.plot(xsort,ypred[SortIndex],color='red',linewidth=5)

plt.xlabel("Total Bill")

plt.ylabel("Tip")

plt.show()

