CODES OF HOMEWORK 1

from numpy import \*

from matplotlib.pyplot import \*

import pandas as pd

from scipy import linalg

data = pd.read\_csv("C:/Users/pc/Desktop/kurs/HW1\_DATA.csv")

x=data.x

y=data.y

plot(x,y,'.')

show()

xsum=0

ysum=0

yx=y.dot(x)

for i in range (0,1000):

xsum=xsum+x[i]

ysum=ysum+y[i]

def mean(num):

return num/1000

den=mean(x.dot(x))-mean(xsum)\*\*2

a= mean(yx)-mean(xsum)\*mean(ysum)

a=a/den

b=mean(ysum)\*mean(x.dot(x))-mean(yx)\*mean(xsum)

b=b/den

yhat=a\*x+b #predicted y with calculated a and b values.

plot(x,y,'.')

plot(x,yhat,'.')

show()

sren=0

stot=0

for i in range (0,999):

sren=sren+(y[i]-yhat[i])\*\*2

stot=stot+(y[i]-mean(ysum))\*\*2

rsquare=1-sren/stot

stot=0

for i in range (0,999):

stot=stot+(y[i]-mean(ysum))\*\*2

def ypoly(p,x):

yhat=0

l=len(p)-1

for i in range (0,len(p)):

yhat=yhat+p[i]\*x\*\*(l-i)

return yhat

def rsquare(yhat,y,stot):

sren=0

for i in range (0,999):

sren=sren+(y[i]-yhat[i])\*\*2

return 1-sren/stot

ones=np.ones(1000)

xhat= np.c\_[x,ones]

#finding model paramaters(w) for polynomials which have different degree.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

w = np.linalg.solve(np.transpose(xhat).dot(xhat),np.transpose(xhat).dot(y) )

print(w)

xhat= np.c\_[x\*\*2,xhat]

w = np.linalg.solve(np.transpose(xhat).dot(xhat),np.transpose(xhat).dot(y) )

print(w)

xhat= np.c\_[x\*\*3,xhat]

w = np.linalg.solve(np.transpose(xhat).dot(xhat),np.transpose(xhat).dot(y) )

print(w)

xhat= np.c\_[x\*\*4,xhat]

w = np.linalg.solve(np.transpose(xhat).dot(xhat),np.transpose(xhat).dot(y) )

print(w)

xhat= np.c\_[x\*\*5,xhat]

w = np.linalg.solve(np.transpose(xhat).dot(xhat),np.transpose(xhat).dot(y) )

print(w)

xhat= np.c\_[x\*\*6,xhat]

w = np.linalg.solve(np.transpose(xhat).dot(xhat),np.transpose(xhat).dot(y) )

print(w)

xhat= np.c\_[x\*\*7,xhat]

w = np.linalg.solve(np.transpose(xhat).dot(xhat),np.transpose(xhat).dot(y) )

print(w)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

y2=ypoly(w,x)

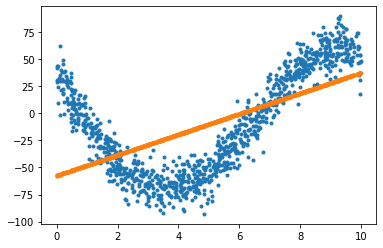
a2=rsquare(y2,y,stot)

plot(x,y,'.',color='b')

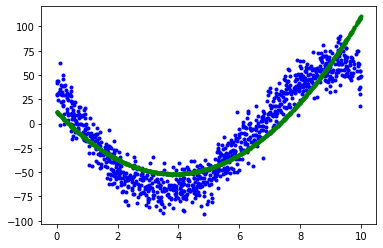
plot(x,y2,'.',color='g')

INFERENCES

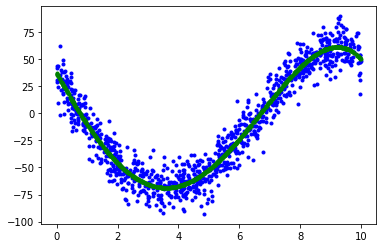
When implementing linear regression instructions, prediction of y. (as shown below) So, error is quite excessive.



When implementing polynomial regression instructions error will be decrease.And by increasing complexity to 4, predicted y will be approach the real data.But from 4 to 7 change won’t be too much.



If degree is 2.



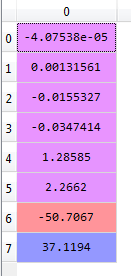
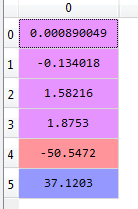
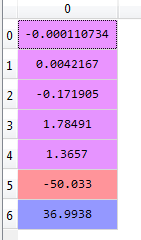
If degree is 7.

|  |  |
| --- | --- |
| Degree | Rsquare |
| 1 | 0.35 |
| 2 | 0.81 |
| 3 | 0.92 |
| 4 | 0.93 |
| 5 | 0.93 |
| *6* | 0.93 |
| *7* | 0.93 |

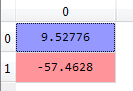
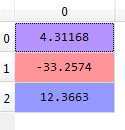
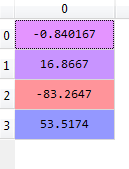
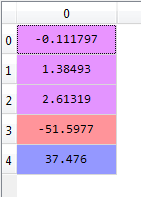
\*\* If Rsquare is 1 this point out that model is best model for us.

We see that as complexity increases, model approaches best model.

After each estimation, arrays of weights . (as shown below)

Degree is 7. Degree is 6. Degree is 5.



Degree is 4. Degree is 3. Degree is 2. Degree is 1.