gradient-descent-optimization-on-randomized-data

April 27, 2023

import libraries

[1]:
```
import pandas as pd
import numpy as np
from numpy import random
import sklearn
import matplotlib.pyplot as plt
import plotly_express as px
```

create the randomized dataset(using numpy) for linear regression

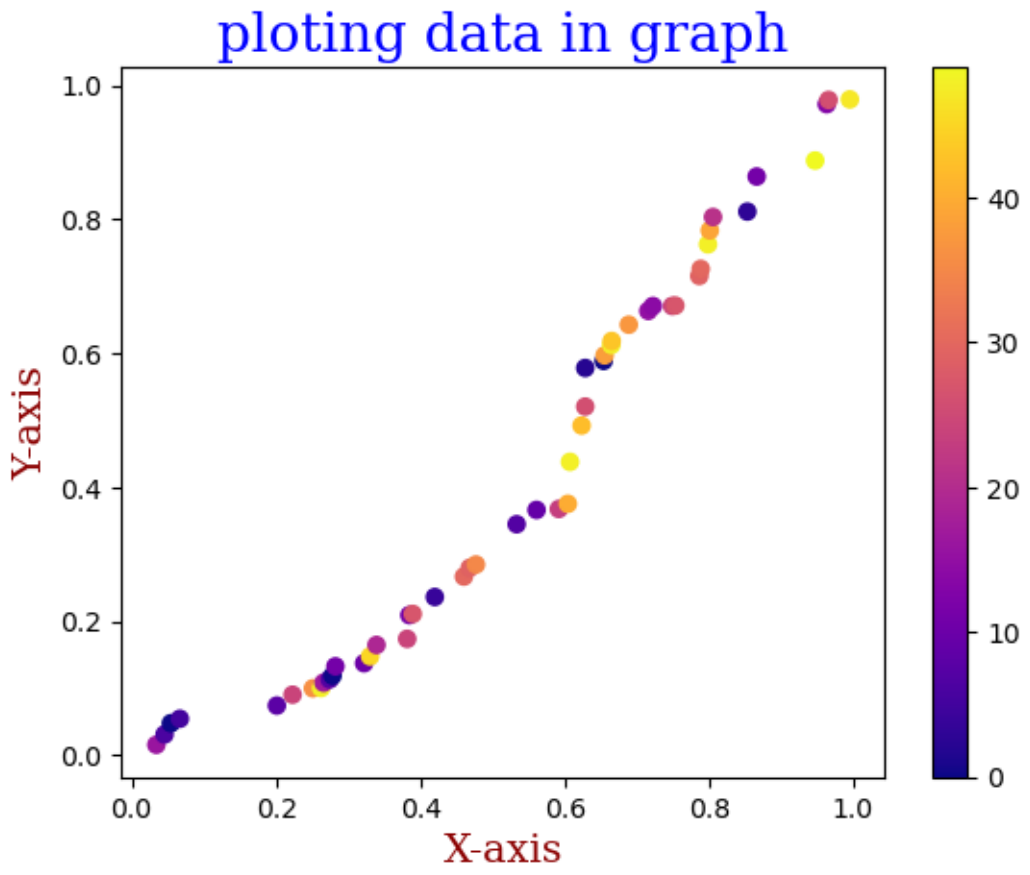
[2]:
```
X = np.random.uniform(0,1,50)
X.sort()
print(X)

Y = np.random.uniform(0,1,50)
Y.sort()
print(Y)
```

```
[0.03403026 0.04543708 0.05421123 0.06665765 0.2011638  0.22292488
 0.25116138 0.26246843 0.26596127 0.27442536 0.27840961 0.28236206
 0.32211058 0.33045117 0.33932555 0.38189007 0.38493798 0.38964082
 0.42019979 0.46051311 0.4686622  0.47725627 0.53370761 0.56127849
 0.59220733 0.60470652 0.60766842 0.62377355 0.62878675 0.62879456
 0.65431975 0.65568564 0.66496485 0.66546445 0.68916868 0.71624485
 0.72268279 0.74931969 0.7537448  0.78697707 0.78897457 0.7990028
 0.8015222  0.80595968 0.8536144  0.8665496  0.94758591 0.96364276
 0.96625042 0.99579617]
[0.01586683 0.03130362 0.04806441 0.05472983 0.07438532 0.0905773
 0.10008893 0.10053942 0.10852516 0.11361031 0.11904825 0.132954
 0.1375805  0.14783402 0.16483353 0.17392785 0.20939023 0.21113518
 0.23640799 0.26687116 0.27994388 0.28484256 0.3450342  0.36638775
 0.36782806 0.37549105 0.43812054 0.49227896 0.52061371 0.5782129
 0.58878485 0.59702052 0.61258419 0.61846635 0.64278211 0.66337841
 0.67071025 0.6707167  0.67136569 0.71549577 0.72610444 0.76273657
 0.78347045 0.80350319 0.81160844 0.86390812 0.88777627 0.9719288
 0.97816192 0.97900246]
```

plot the graph based on randomized dataset

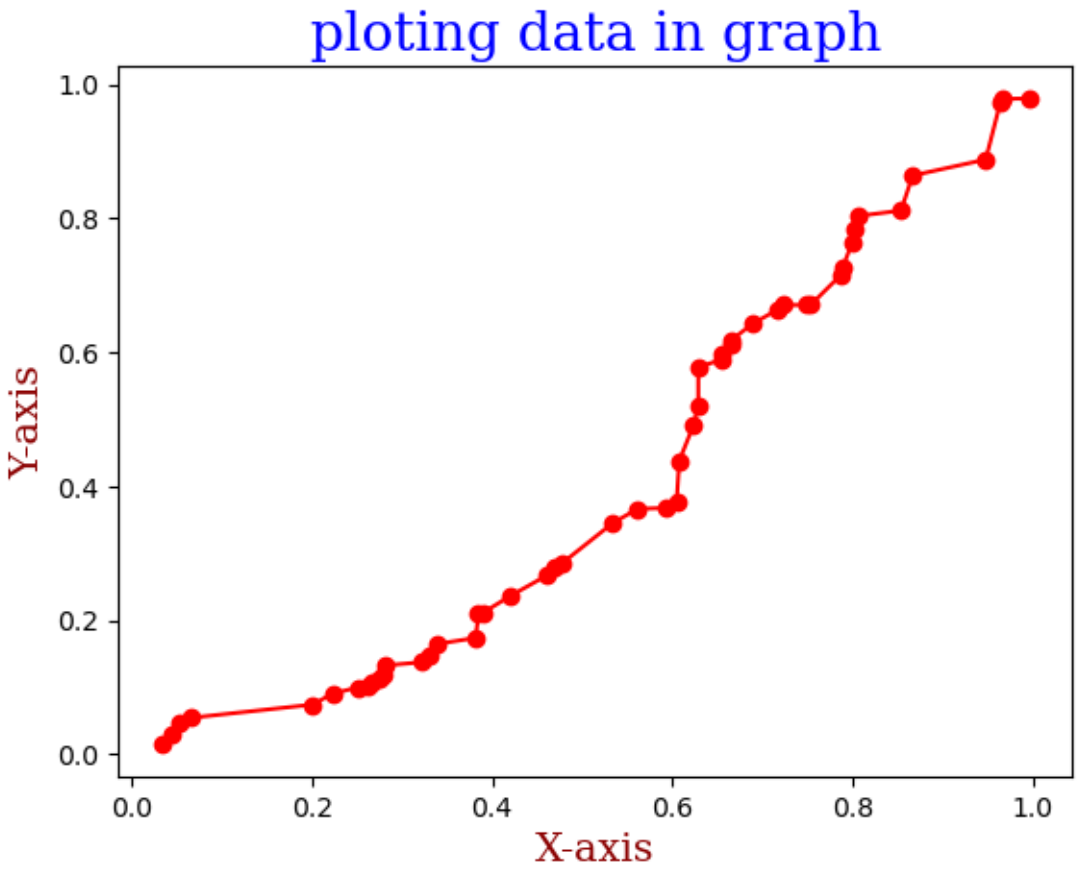
[3]:
```
m=1
c=0
y = m*X+c
```

[4]:
```
colors = np.random.randint(50,size=(50))
plt.scatter(X,Y,c=colors,cmap='plasma')
plt.colorbar()
font1 = {'family':'serif','color':'blue','size':'20'}
font2 = {'family':'serif','color':'darkred','size':'15'}
plt.title('ploting data in graph',fontdict=font1)
plt.xlabel('X-axis',fontdict=font2)
plt.ylabel('Y-axis',fontdict=font2)
plt.show()
```



[5]:
```
colors = np.random.randint(50,size=(50))
plt.plot(X,Y,'r',marker='o')
font1 = {'family':'serif','color':'blue','size':'20'}
font2 = {'family':'serif','color':'darkred','size':'15'}
plt.title('ploting data in graph',fontdict=font1)
```

```
plt.xlabel('X-axis',fontdict=font2)
plt.ylabel('Y-axis',fontdict=font2)
plt.show()
```



[6]:
```
fig = px.line(x=X,y=Y,title='plot data in graph')
fig.update_layout(
    font_family="Courier New",
    font_color="blue",
    title_font_family="Times New Roman",
    title_font_color="red",
    legend_title_font_color="green"
)
fig.update_xaxes(title_font_family="Arial")
fig.show()
```

making of gradient descent model to optimize linear regression

[7]:
```
#gradient descent model
```

```
def gradient(iter):
  m = 0
  c = 0

  L = 0.01

  n = float(len(X))

# iter = 10

  for i in range(iter):
    y_pred = m*X+c

    der_m = (-2/n)*np.sum(X*(Y-y_pred))
  # print(der_m)

    der_c = (-2/n)*np.sum(Y-y_pred)
  # print(der_c)

  #update the value of m and c

    m = m - L * der_m

    c = c - L * der_c

  print(m,c)
  return m,c
```

making a function for plotting a graph of optimized model by various iterations

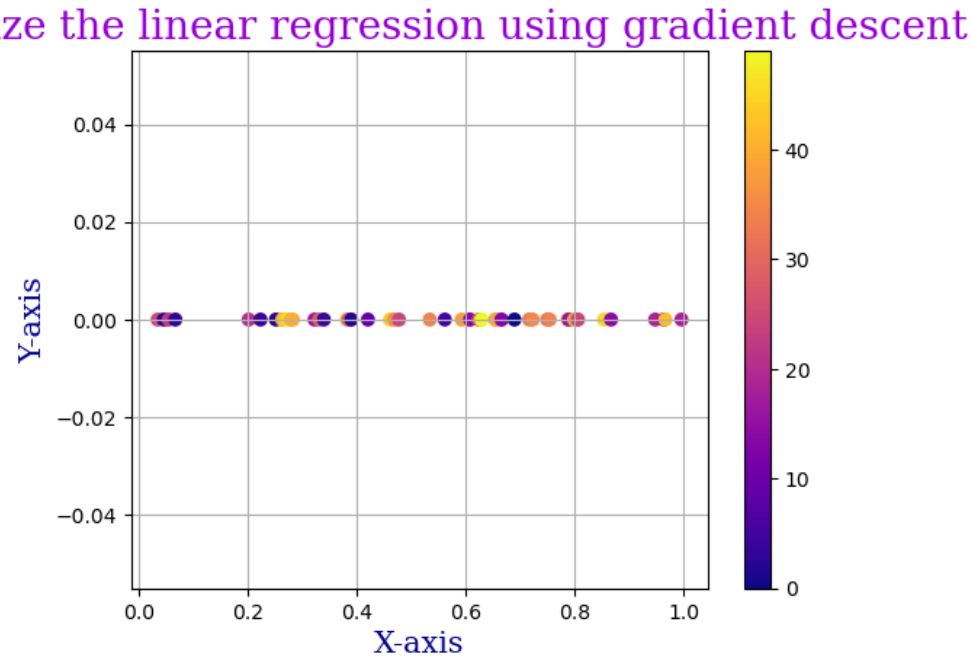
[8]:
```
def graph():
   iter=[0,10,20,50,100,1000]
   for i in range(len(iter)):
     #iterative for n iter

     m,c = gradient(iter[i])

     y_pred = m*X + c

     plt.figure()
     colors = np.random.randint(50,size=(50))

     plt.scatter(X,y_pred,c=colors,cmap='plasma')

     plt.colorbar()

     font1 = {'family':'serif','color':'darkviolet','size':'20'}
```

```
    font2 = {'family':'serif','color':'darkblue','size':'15'}

    plt.title('optimize the linear regression using gradient descent',fontdict␣
↪= font1)
    plt.xlabel('X-axis',fontdict = font2)
    plt.ylabel('Y-axis',fontdict = font2)

    plt.grid()

    plt.show()
```

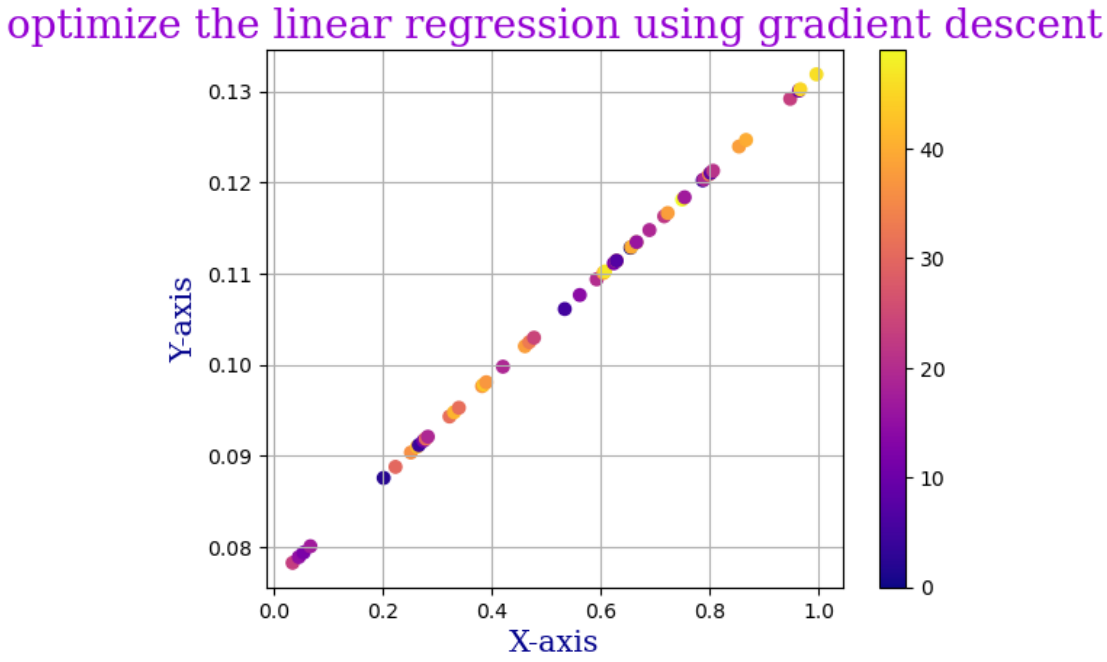
Scatter graph plotting having iteration range[0,10,20,50,100,1000]

[9]:
```
graph()
```

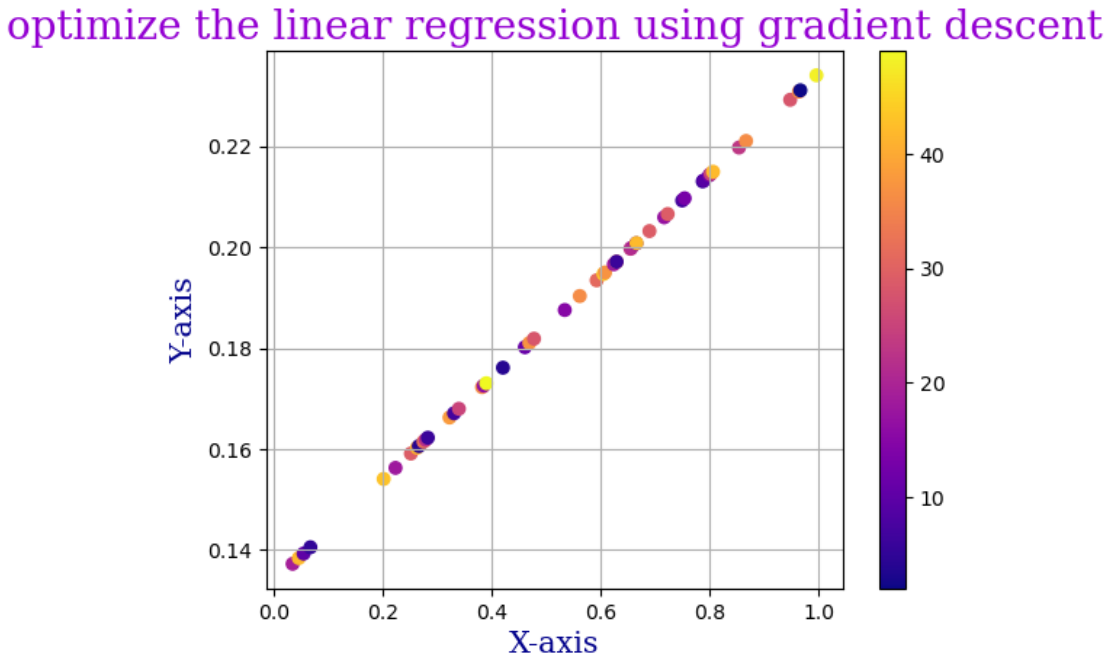
```
0 0
```



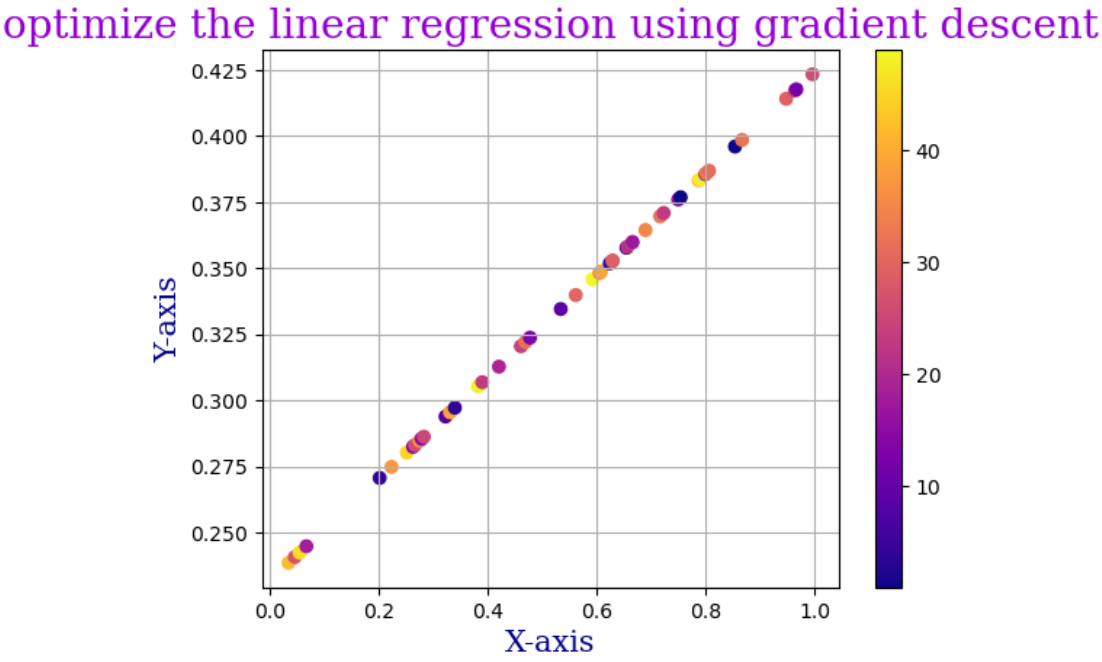

```
0.05570014650590924 0.07639301556979419
```



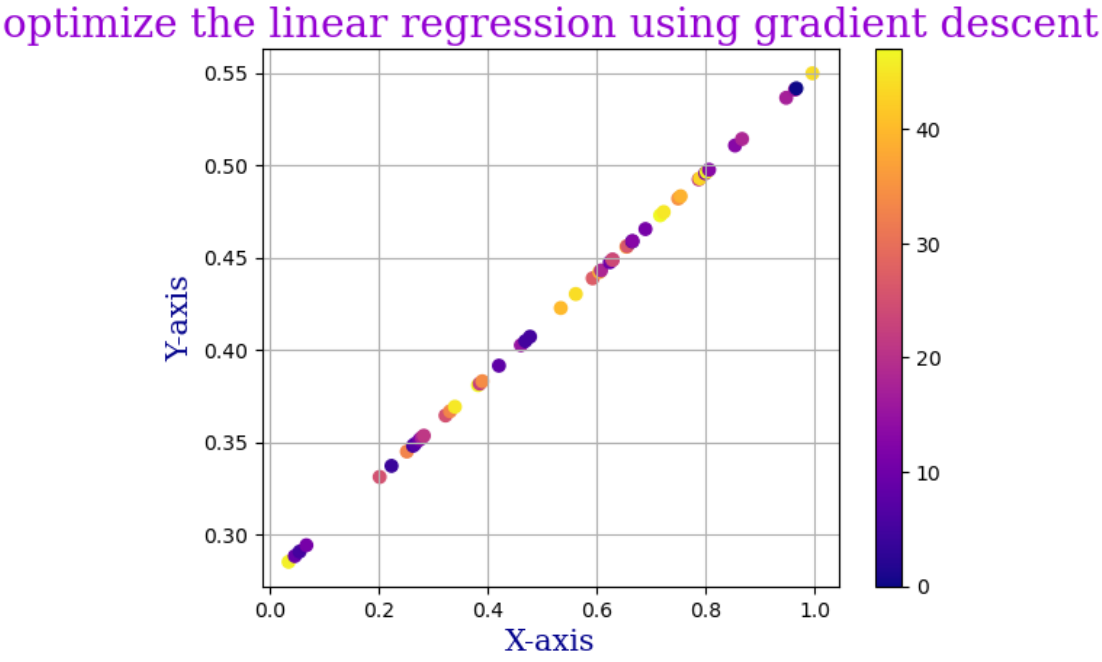
0.10055539090019369 0.13386183645922112



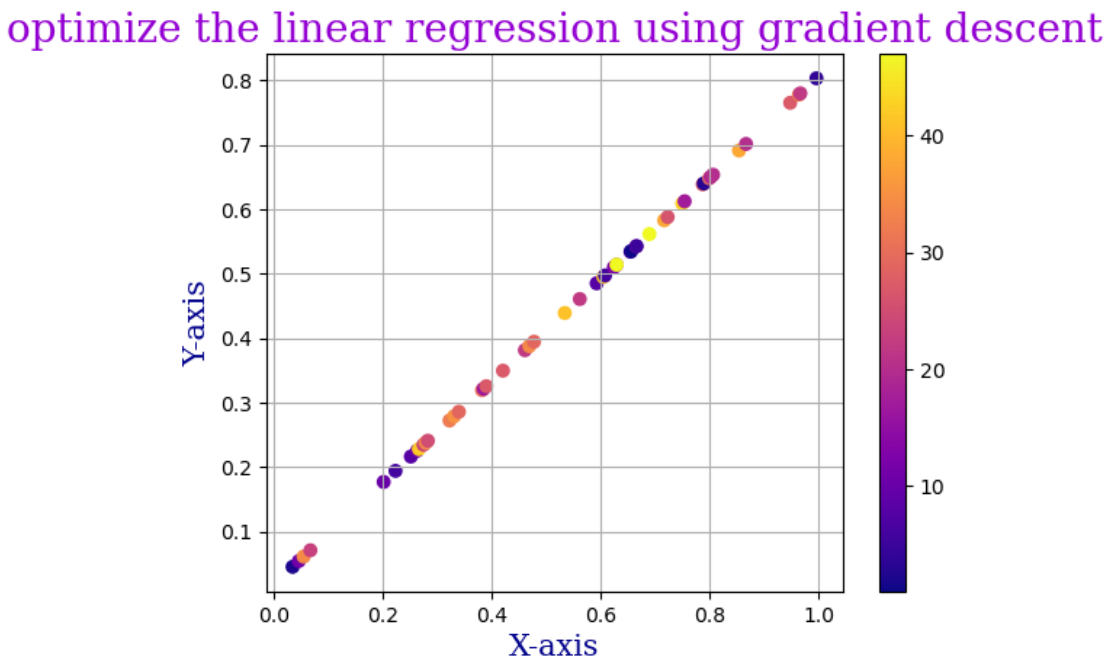
0.19225690467876388 0.23191272307172459



0.27503956038456556 0.2759510880154193



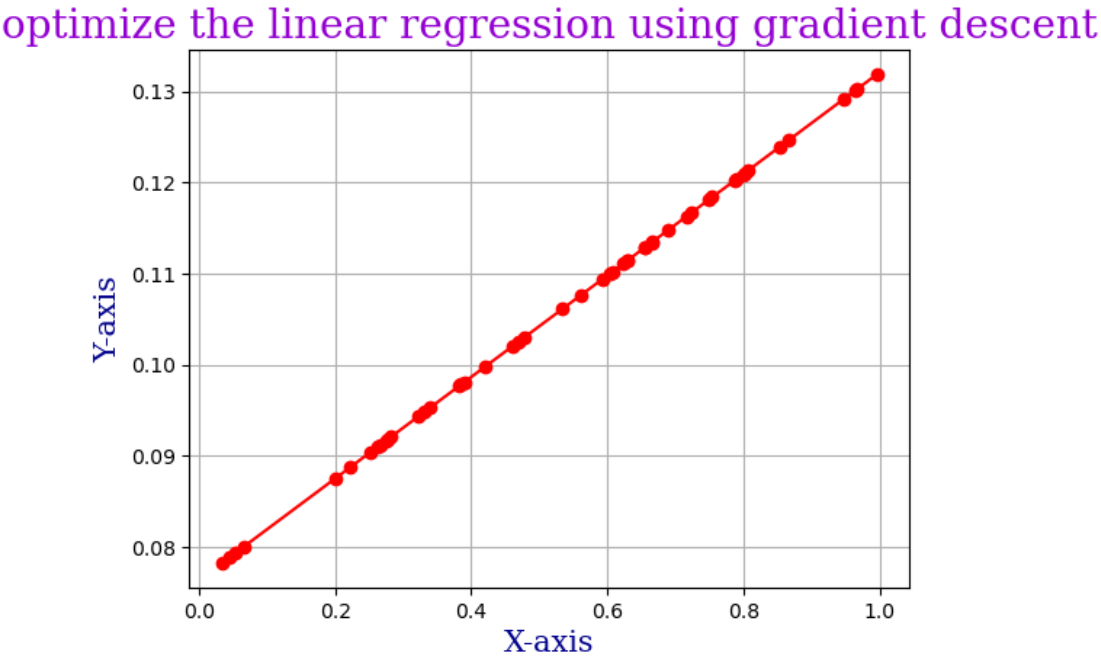
0.7880667284978206 0.018518766832494125



line graph plot of optimized model

[10]:
```
m,c = gradient(10)

y_pred = m*X+c

plt.plot(X,y_pred,'r',marker='o')

font1 = {'family':'serif','color':'darkviolet','size':'20'}
font2 = {'family':'serif','color':'darkblue','size':'15'}

plt.title('optimize the linear regression using gradient descent',fontdict =␣
 ↪font1)
plt.xlabel('X-axis',fontdict = font2)
plt.ylabel('Y-axis',fontdict = font2)

plt.grid()

plt.show()
```

```
0.05570014650590924 0.07639301556979419
```



plot a 2d graph using plotly-express

[11]:
```
def graph():
   iter=[0,10,20,50,100,1000]
   for i in range(len(iter)):
     #iterative for n iter

     m,c = gradient(iter[i])

     y_pred = m*X + c

     fig = px.line(x=X,y=y_pred)

     fig.show()
```

[12]:
```
graph()
```

```
0 0
0.05570014650590924 0.07639301556979419
0.10055539090019369 0.13386183645922112
0.19225690467876388 0.23191272307172459
0.27503956038456556 0.2759510880154193
0.7880667284978206 0.018518766832494125
```