Gradient Descent

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
%matplotlib inline
x=np.linspace(0, 20, 50)
y=-2*x+1
```

```
def GD_single_LR(x,y,epochs,learning_rate):
  m=len(y)
  theta0=0
  theta1=0
  cost_history=[]
  theta0_history=[]
  theta1_history=[]
  for i in range(epochs):
    h = theta0 + theta1*x
    cost = sum([error**2 for error in (h-y)]) / 2*m
    cost_history.append(cost)

    theta0 = theta0 - (learning_rate * np.sum(h-y) / m)
    theta1 = theta1 - (learning_rate * (np.sum((h-y)*x) / m))

    theta0_history.append(theta0)
    theta1_history.append(theta1)
  return h,theta0_history,theta1_history,cost_history

h,theta0_history,theta1_history,cost_history=GD_single_LR(x,y,1000,0.004)
```

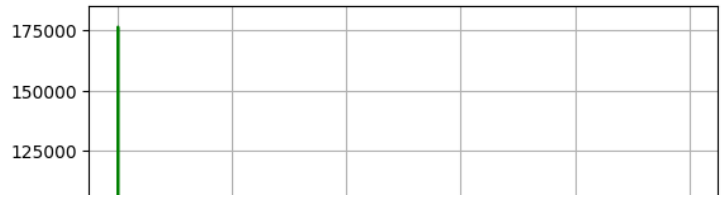
```
r2_score(y, h)
```

```
     0.9996876022909399
```

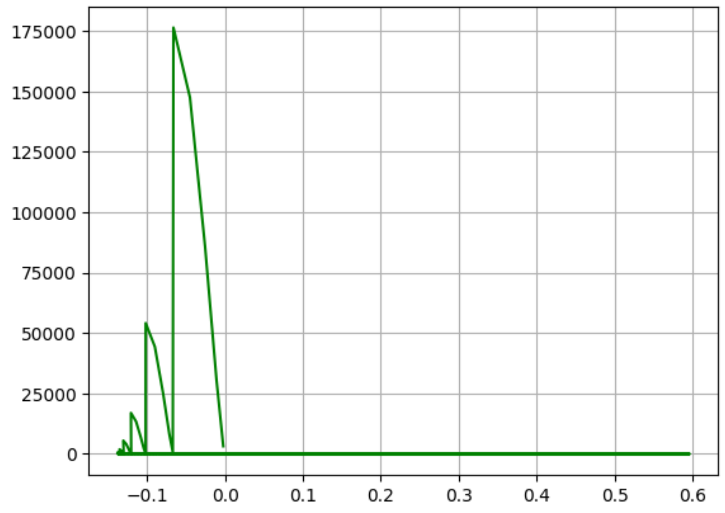
Mini Batch Gradient Descent

```
def GD_single_Mini_Patches_LR(x,y,epochs,learning_rate,batches):
  m=len(y)
  theta0=0
  theta1=0
  cost_history=[]
  theta0_history=[]
  theta1_history=[]
  for e in range(epochs):
    for j in range(0,m,batches):
      x_batch=x[j:j+batches]
      y_batch=y[j:j+batches]
      h = theta0 + theta1*x_batch
      cost = sum([error**2 for error in (h-y_batch)]) / 2*m
      cost_history.append(cost)

      theta0 = theta0 - (learning_rate * np.sum(h-y_batch) / m)
      theta1 = theta1 - (learning_rate * (np.sum((h-y_batch)*x_batch) / m))

      theta0_history.append(theta0)
      theta1_history.append(theta1)
  return theta0_history,theta1_history,cost_history

theta0_history,theta1_history,cost_history=GD_single_Mini_Patches_LR(x,y,1000,0.004,10)
h= theta0_history[-1] + theta1_history[-1]*x
```

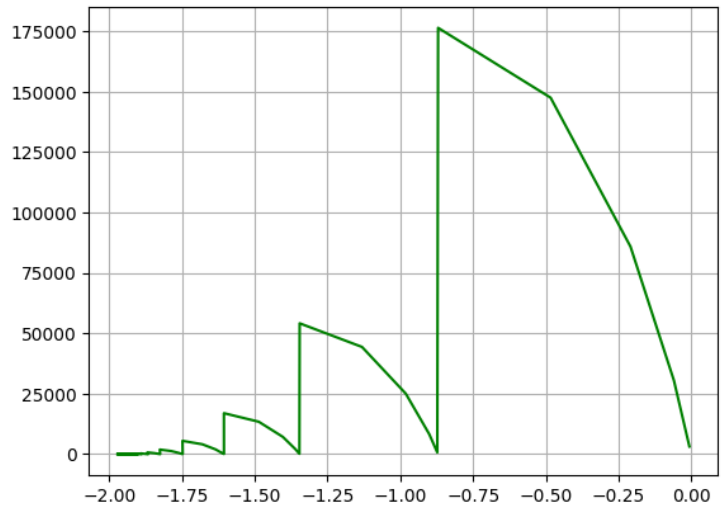
```
fig,ax=plt.subplots()
ax.plot(cost_history,'g-')
plt.grid()
plt.show()
```



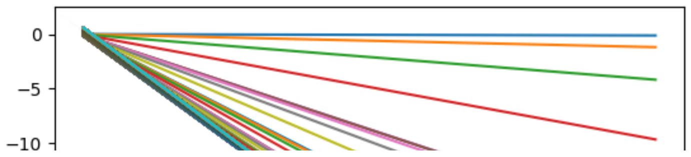
```
fig,ax=plt.subplots()
ax.plot(theta0_history,cost_history,'g-')
plt.grid()
plt.show()
```



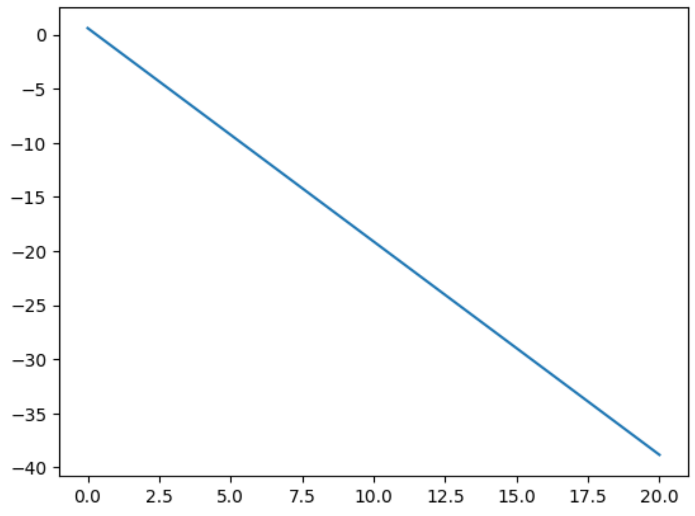
```
fig,ax=plt.subplots()
ax.plot(theta1_history,cost_history,'g-')
plt.grid()
plt.show()
```



```
fig, ax = plt.subplots()
for t0,t1 in zip(theta0_history,theta1_history):
    plt.plot(x,x*t1+t0)
plt.show()
```



```
fig, ax = plt.subplots()
plt.plot(x,x*theta1_history[-1]+theta0_history[-1])
plt.show()
```



```
r2_score(y, h)
```

```
    0.9996931809799962
```

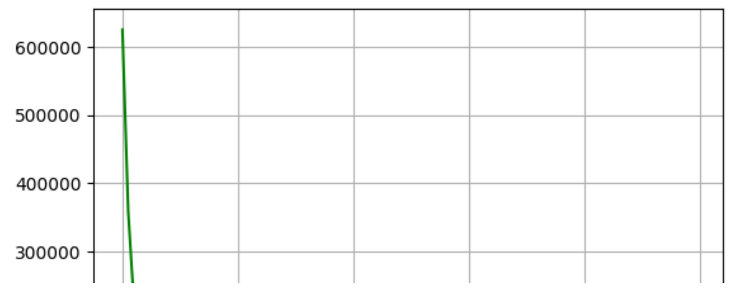
Stocastic gradinent Descent

```
def Stochastic_GD(x,y,learning_rate,epochs):
  theta0=0
  theta1=0
  m=float(len(y))
  cost_history=[]
  hypothesis=[]
  theta_0_history=[]
  theta_1_history=[]
  for i in range(epochs):
    h = theta0 + theta1*x
    cost = sum([data**2 for data in (h-y)]) / 2*m
    cost_history.append(cost)
    hypothesis.append(h)

    theta_0_history.append(theta0)
    theta_1_history.append(theta1)

    theta0 = theta0 - (learning_rate * (h-y) / m)
    theta1 = theta1 - (learning_rate * (((h-y)*x) / m))

  return theta0,theta1,cost_history,hypothesis,theta_0_history,theta_1_history

theta0, theta1, cost_history, hypothesis, theta_0_history, theta_1_history = Stochastic_GD(x,y,0.05,100)
```

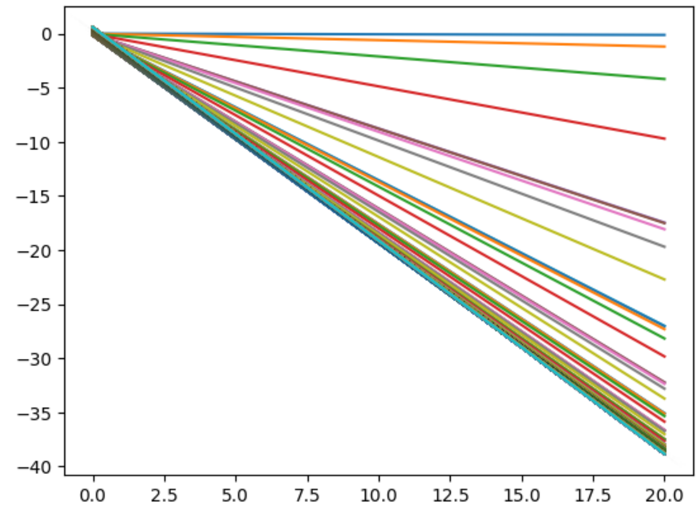
```
r2_score(y, hypothesis[-1])
```

```
    0.9964938986390629
```

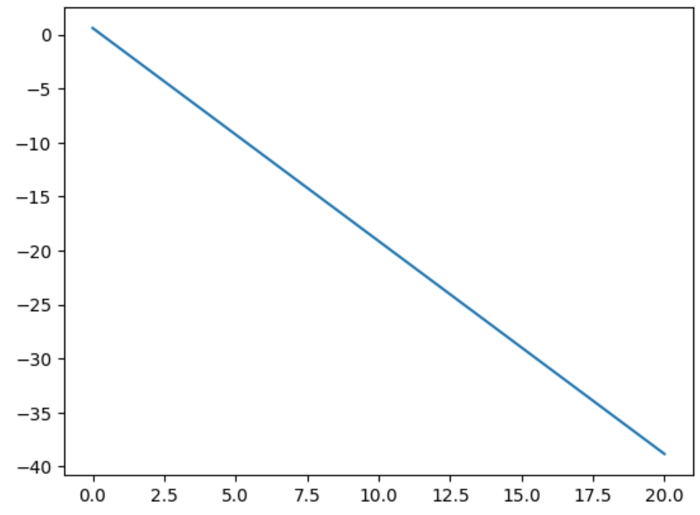
```
fig,ax=plt.subplots()
ax.plot(cost_history,'g-')
plt.grid()
plt.show()
```



```
fig, ax = plt.subplots()
for t0,t1 in zip(theta0_history,theta1_history):
    plt.plot(x,x*t1+t0)
plt.show()
```



```
fig, ax = plt.subplots()
plt.plot(x,x*theta1_history[-1]+theta0_history[-1])
plt.show()
```


0
−5
−10
−15
−20
−25
−30
−35
−40
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0