```
import numpy as np
import math
```

Gradient Descent

```
def gradient_descent(X, y, learing_rate=0.01, iterations=1000):
    m, a, b=len(y), 0, 0
    costs=[]
    for _ in range(iterations):
        y_pred=a+b*X
        error=y_pred-y
        a=a-learing_rate*sum(error)/m
        b=b-learing_rate*error.dot(X)
        cost=(1/(2*m))*sum(error**2)
        costs.append(cost)
    return a, b, costs
```

Root Mean Squared Error

```
def root_mean_sqaured_error(y_pred, y):
    error=y-y_pred
    error=error**2
    s=sum(error)
    s/=len(y)
    return math.sqrt(s)
```

Mean Absolute Error

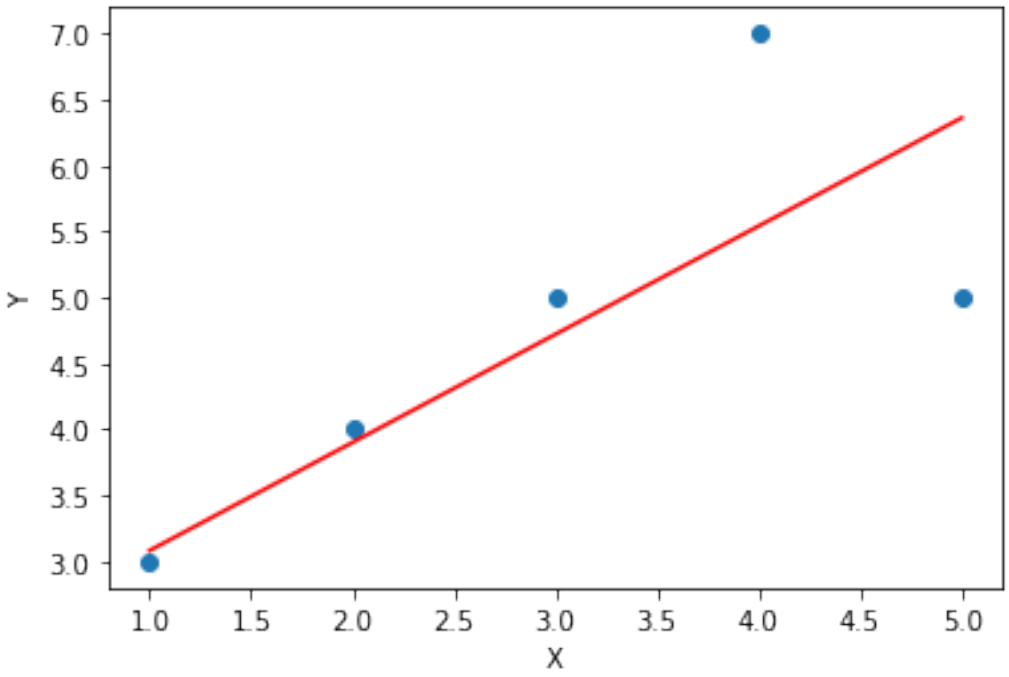
```
def mean_absoulute_error(y_pred, y):
    error=abs(y-y_pred)
    return sum(error)/len(y)
```

R2 Score

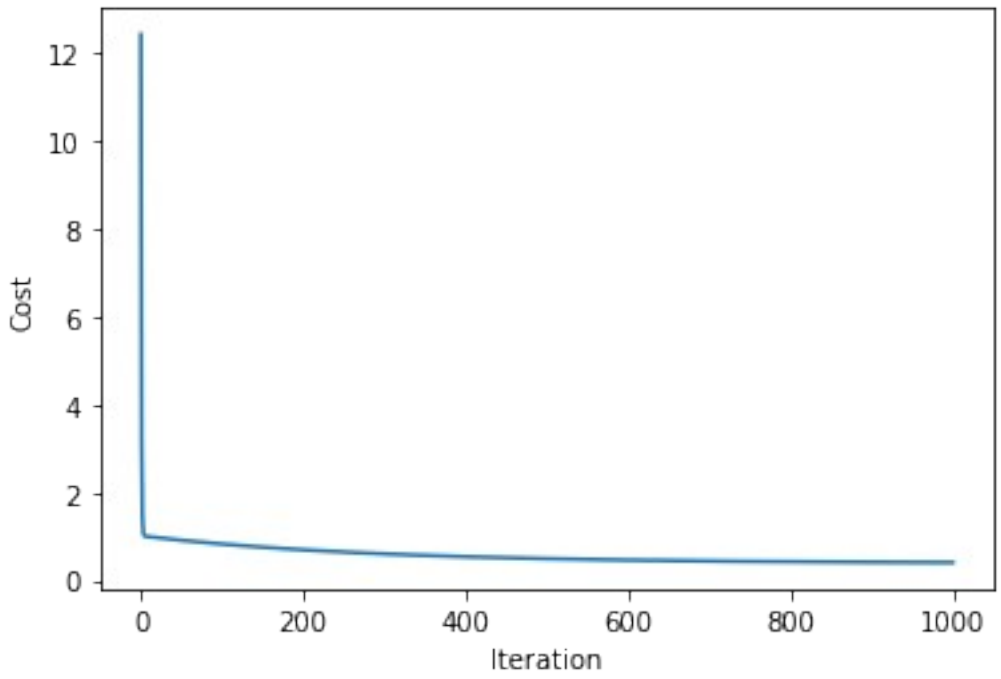
```
def r2_score(y_pred, y):
    avg=y.mean()
    SSres=sum((y-y_pred)**2)
    SStot=sum((y-avg)**2)
    return 1-SSres/SStot

X=np.arange(1, 6)
Y=np.array([3, 4, 5, 7, 5])

a, b, costs=gradient_descent(X, Y)
y_pred=a+b*X

print(a, b, root_mean_sqaured_error(y_pred, Y),
mean_absoulute_error(y_pred, Y), r2_score(y_pred, Y))
```

```
2.263493307840752 0.8194363043824198 0.9025768411894179
0.6556395558023976 0.537133548720757
```

```
import matplotlib.pyplot as plt

plt.scatter(X, Y)
plt.plot(X, y_pred, color="red")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```




```
plt.plot(costs)
plt.xlabel("Iteration")
plt.ylabel("Cost")
plt.show()
```



Ridge

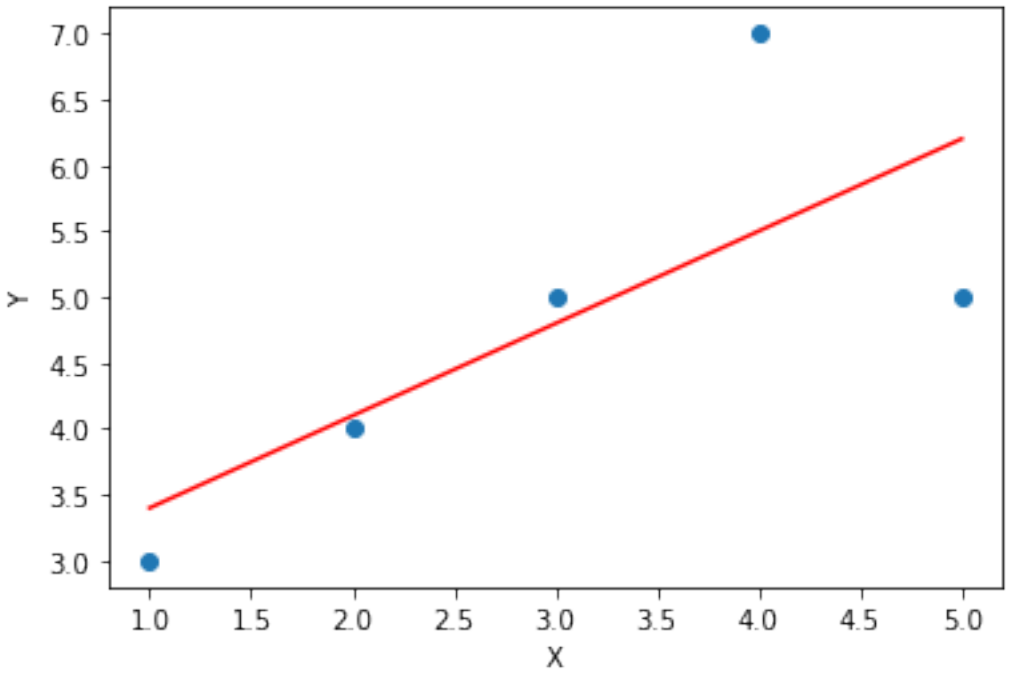
```
def Regularization(X, Y):
    from sklearn.linear_model import Ridge

    alphas = np.array([2**i for i in range(-18, 51, 2)])

    best_mse, best_alpha = float("inf"), None

    from sklearn.metrics import mean_squared_error

    for alpha in alphas:
        regressor = Ridge(alpha=alpha)
        regressor.fit(X, Y)
        y_pred = regressor.predict(X)
        mse = mean_squared_error(Y, y_pred)

        if mse < best_mse:
            best_mse, best_alpha = mse, alpha
    return np.array([best_mse, best_alpha])

X=X.reshape(-1, 1)
Y=Y.reshape(-1, 1)
mse, alpha=Regularization(X, Y)

print(mse, alpha)
```

```
0.7800000000001427 3.814697265625e-06
```

```
from sklearn.linear_model import Ridge

regressor=Ridge(alpha=alpha)
regressor.fit(X, Y)
```

```
Ridge(alpha=3.814697265625e-06)
```

```
import matplotlib.pyplot as plt

plt.scatter(X, Y)
plt.plot(X, regressor.predict(X), color="red")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```



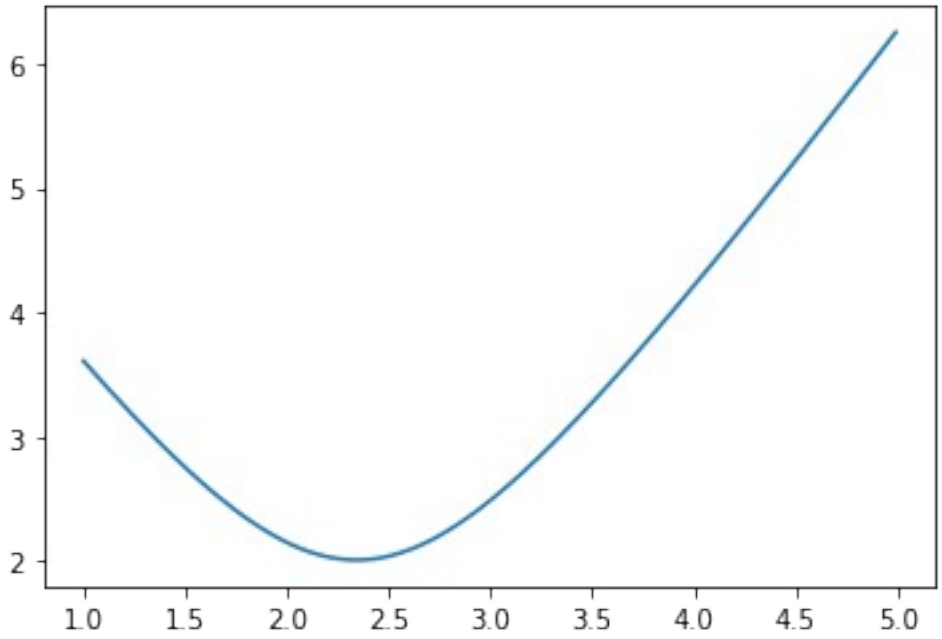

```
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(Y, regressor.predict(X))
print(mse)
```

```
0.7800000000001427
```

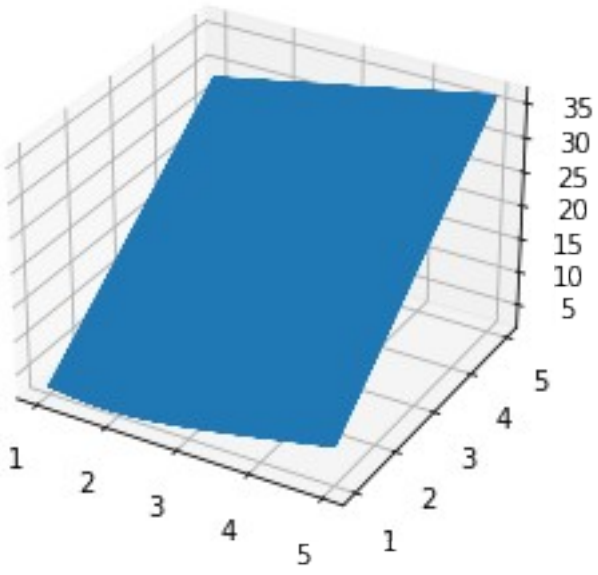
```
costs=[]
A=np.arange(1, 5, 0.01)
for a in A:
    y_pred=a+X*b
    error=math.sqrt(sum((Y-y_pred)**2))
    costs.append(error)

costs=np.array(costs)
```

```
plt.plot(A, costs)
```

```
[<matplotlib.lines.Line2D at 0x24e897cb340>]
```




```
A = np.arange(1, 5, 0.01)
B = np.arange(1, 5, 0.01)
x, y=[], []
costs = []
for a in A:
    for b in B:
        y_pred = a + X * b
        x.append(a)
        y.append(b)
        error = math.sqrt(sum((Y - y_pred) ** 2))
        costs.append(error)

x=np.array(x)
y=np.array(y)
costs=np.array(costs)

ax=plt.axes(projection="3d")
ax.plot3D(x, y, costs)
plt.show()
```


35
30
25
20
15
10
5
1
2
3
4
5
1
2
3
4
5