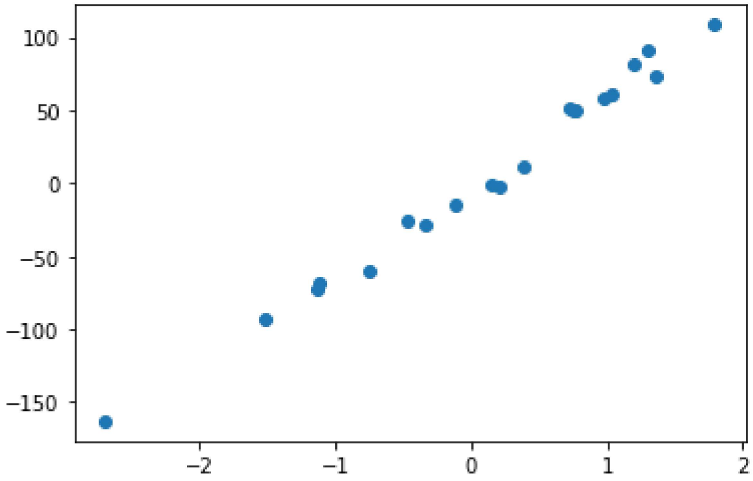
```
In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.datasets import make_regression
```

```
In [3]: x,y = make_regression(n_samples=20,n_features=1,noise=6)
```

```
In [4]: plt.scatter(x,y)
```

```
Out[4]: <matplotlib.collections.PathCollection at 0x20631249340>
```



```
In [5]: from sklearn.linear_model import LinearRegression
```

```
In [6]: lr = LinearRegression()
```

```
In [7]: lr.fit(x,y)
```

```
Out[7]: LinearRegression()
```

```
In [8]: m = lr.intercept_
        m
```

```
Out[8]: -2.297843690251465
```

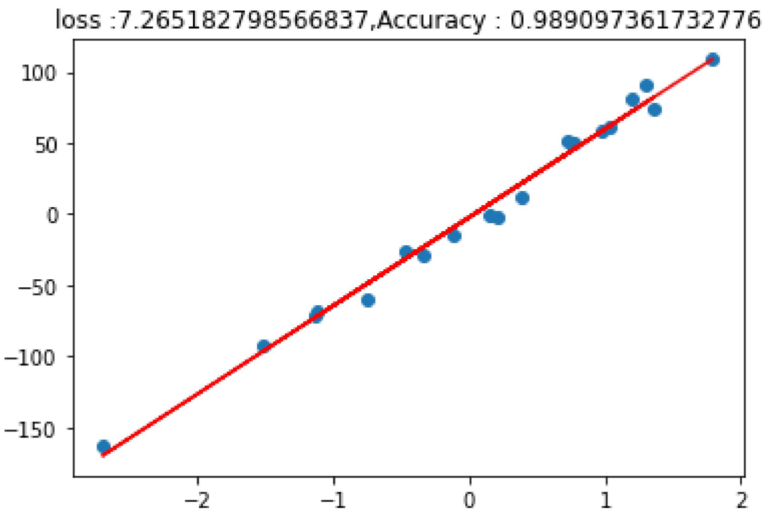
```
In [9]: b = lr.coef_
        b
```

```
Out[9]: array([61.99288171])
```

```
In [10]: from sklearn.metrics import mean_squared_error ,r2_score
```

In [16]:
```
plt.plot(x,lr.predict(x),'r-')
plt.scatter(x,y)
plt.title(f'loss :{np.sqrt(mean_squared_error(y,lr.predict(x)))},Accuracy
```

Out[16]:
```
Text(0.5, 1.0, 'loss :7.265182798566837,Accuracy : 0.989097361732776')
```



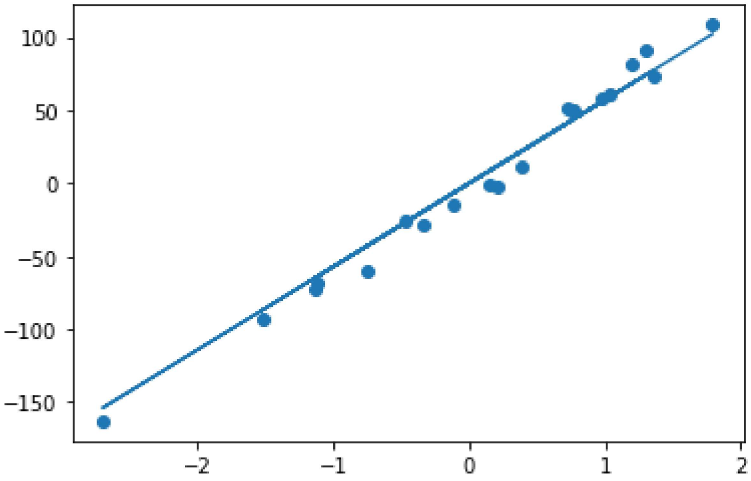
In [44]:

```
m = 0
b = 3
lr = 0.001
hh = []
slope = []
intercept = []
for i in range(50):
    loss_slope_b = -2 * np.sum(y - m*x.ravel() - b)
    loss_slope_m = -2  * np.sum((y - m*x.ravel() - b)*x.ravel())

    b = b - (lr * loss_slope_b)
    m = m - (lr * loss_slope_m)
    y_hat = np.sqrt(mean_squared_error(y,(m*x)+b))
    ht = hh.append(y_hat)
    ss = slope.append(m)
    ii = intercept.append(b)
print(f'Slope {m}, Y-intercept {b}, loss {y_hat}')
#print(hh)

plt.plot(x,slope[i]* x+ intercept[i])
plt.scatter(x,y)
plt.show()
```

```
Slope 57.00824751315233, Y-intercept 0.0399188760274288, loss 9.3110203337943
92
```



In [18]:

```
x.ravel()
```

Out[18]:

```
array([ 1.7935696 ,  0.20820925,  1.29508123,  0.15686144, -1.11538748,
        0.75037235, -0.75774354,  0.72784695,  0.97504291, -0.1224515 ,
       -2.70280222,  1.36313185,  1.19503582, -1.14239029,  0.384766  ,
        0.77610859,  1.03836406, -0.47411426, -0.340667  , -1.52440764])
```

In [30]:

```
class GDRegressor:
    def __init__(self,learning_rate,epochs):
        self.m=0
        self.b=0
        self.lr=learning_rate
        self.epochs = epochs

    def fit(self,X,y):
        for i in range(self.epochs):
            loss_slope_b = -2 * np.sum(y - self.m*X.ravel() - self.b)
            loss_slope_m = -2 * np.sum((y - self.m*X.ravel() - self.b)*X.r

            self.b = self.b - (self.lr * loss_slope_b)
            self.m = self.m - (self.lr * loss_slope_m)
        print(self.m,self.b)

    def predict(self,X):
        return self.m * X + self.b
```

In [31]:

```
gd =GDRegressor(0.001,500)
```

In [32]:

```
gd.fit(x,y)
```

```
61.99288167075977 -2.2978436007529375
```

In [ ]:

```

```