In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
from sklearn.datasets import make_regression
```

In [3]:

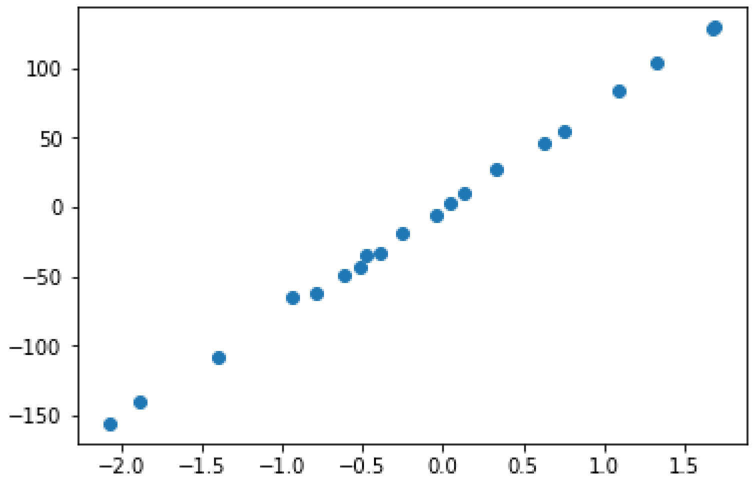
```
x,y = make_regression(n_samples=20,n_features=1,noise=3)
```

In [4]:

```
plt.scatter(x,y)
```

Out[4]:

```
<matplotlib.collections.PathCollection at 0xdf30690>
```



In [5]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error ,r2_score
```

In [6]:

```
lin_reg = LinearRegression()
lin_reg.fit(x,y)
```

Out[6]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Fals
e)
```

In [7]:

```
mm = lin_reg.coef_
mm
```

Out[7]:

```
array([76.07133985])
```

In [8]:

```
bb = lin_reg.intercept_
bb
```

Out[8]:

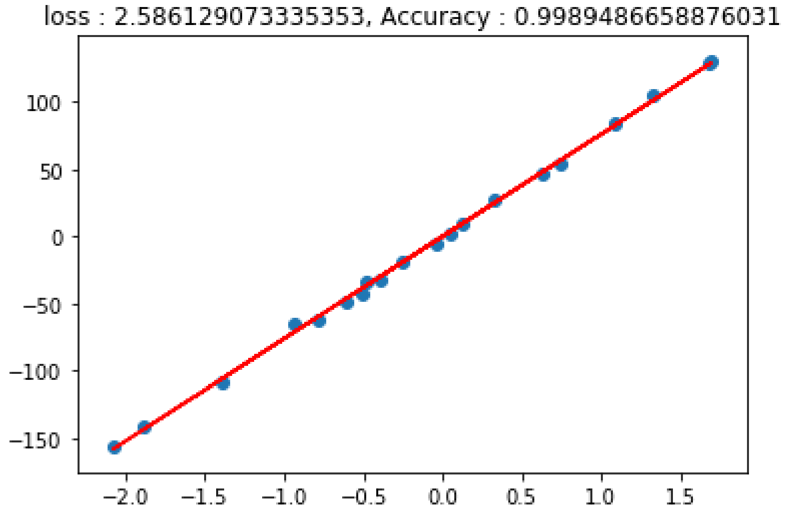
```
0.00019246078332457728
```

In [9]:

```
plt.plot(x,lin_reg.predict(x),'r-')
plt.scatter(x,y)
plt.title(f'loss : {np.sqrt(mean_squared_error(y,lin_reg.predict(x)))}, Accuracy : {r2_sc
```

Out[9]:

```
Text(0.5,1,'loss : 2.586129073335353, Accuracy : 0.9989486658876031')
```



In [10]:

```
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):
        # calcualte the b using GD
        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.ravel())

            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 [11]:

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

In [12]:

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

```
76.07133978210632 0.00019234406784976l7
```