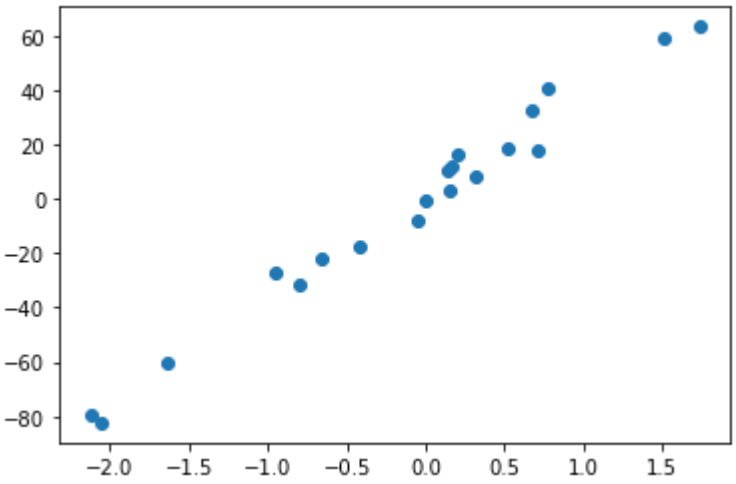
In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn.datasets import make_regression
```

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

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

Out[4]:
```
<matplotlib.collections.PathCollection at 0x1d2fb3d31c0>
```



In [5]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

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

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

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

In [8]:
```
mm = lin_reg.coef_
mm
```

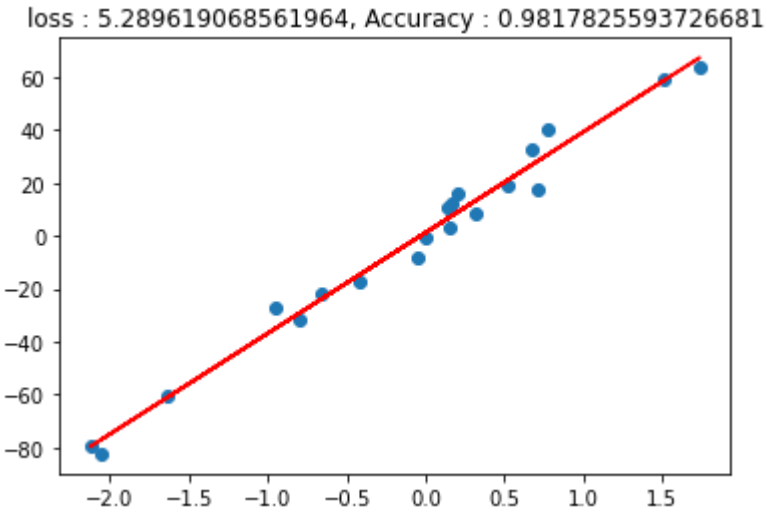
Out[8]:
```
array([38.07864358])
```

In [9]:
```
bb =lin_reg.intercept_
bb
```

Out[9]:
```
1.2371577950740122
```

In [10]:
```
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)))}, Accura
```

Out[10]:
```
Text(0.5, 1.0, 'loss : 5.289619068561964, Accuracy : 0.9817825593726681')
```



In [12]:
```
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.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 [13]:
```
gd = GDRegressor(0.001,500)
```

In [14]:
```
gd.fit(x,y)
```

```
38.07864348876774 1.2371576843786072
```

In [ ]:
```

```