A college professor believes that if the grade for internal examination is high in a class, the grade for external examination will also be high. A random sample of 7 students in that class was selected, and the data is given below: Input 0.1 0.2 0.3 0.4 0.5 0.6 0.7 Target 1.2 1.4 1.55 1.75 2.01 2.2 2.35 Write a python program for linear regression using a single neuron (with proper activation function) on the above dataset, and find the coefficients w1,and b. Predict the external marks if internal marks are 0.15. Draw the scatter plot between Internal Exam and External Exam .Draw a straight line with red line using above w1, w2 and b.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Input data
internal_marks = np.array([0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7])
external_marks = np.array([1.2, 1.4, 1.55, 1.75, 2.01, 2.2, 2.35])
```

In [2]:
```
class LinearRegression:
    def __init__(self):
        self.w1 = None
        self.b = None
    def fit(self, X, y):
        X_mean = np.mean(X)
        y_mean = np.mean(y)
        nums=np.sum((X-X_mean)*(y-y_mean))
        den=np.sum((X-X_mean)**2)
        self.w1=nums/den
        self.b=y_mean-self.w1*X_mean

    def predict_model(self,X):
      return self.w1*X+self.b
```

In [3]:
```
model=LinearRegression()
```

In [4]:
```
model.fit(internal_marks,external_marks)
```

In [5]:
```
w1=model.w1
b=model.b
```

In [6]:
```
print("W1:",w1)
print("b:",b)
```

```
W1: 1.9678571428571432
b: 0.9928571428571425
```

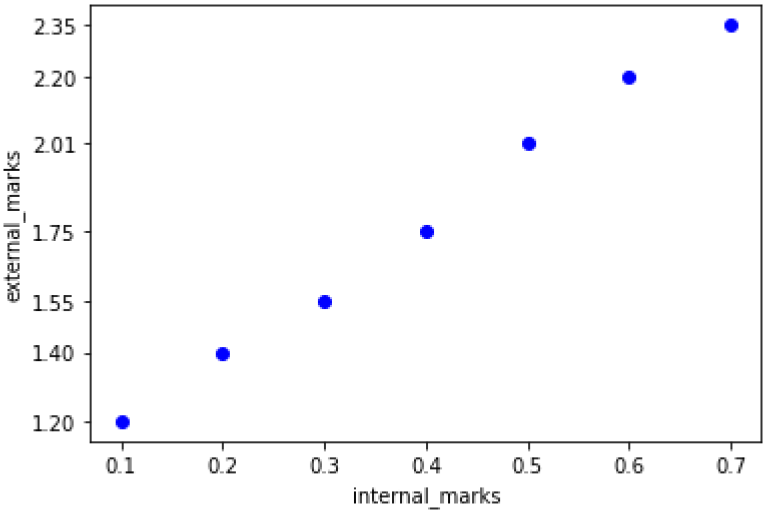
In [7]:
```
int_marks=0.15
external_marks_pred=model.predict_model(int_marks)
```

In [8]:
```
print("External marks for internal_marks 0.15 is:",external_marks_pred)
```

```
External marks for internal_marks 0.15 is: 1.288035714285714
```

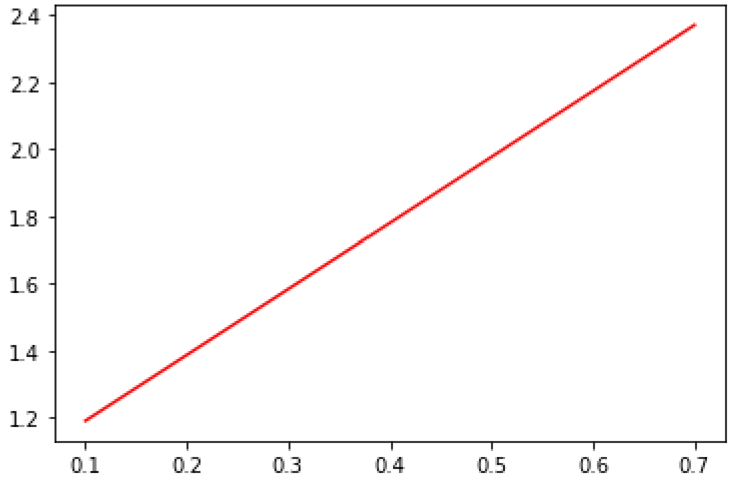
In [9]:
```
p1=plt.scatter(internal_marks,external_marks,color='blue',label='Data point
plt.xticks(internal_marks)
plt.yticks(external_marks)
plt.xlabel('internal_marks')
plt.ylabel('external_marks')
```

Out[9]:
```
Text(0, 0.5, 'external_marks')
```



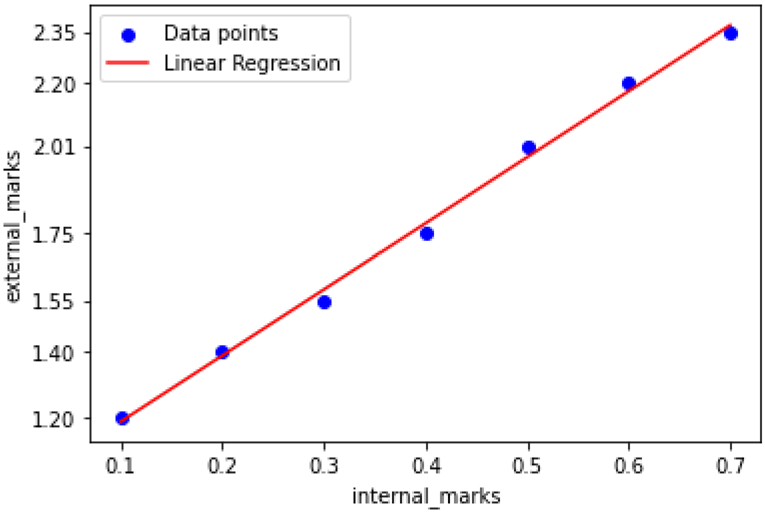
In [10]:
```
ext_marks_pred=model.predict_model(internal_marks)
```

In [11]:
```
p2=plt.plot(internal_marks,ext_marks_pred,color='red',label='Linear Regress
```



In [12]:
```
p1=plt.scatter(internal_marks,external_marks,color='blue',label='Data point
plt.xticks(internal_marks)
plt.yticks(external_marks)
plt.xlabel('internal_marks')
plt.ylabel('external_marks')
p2=plt.plot(internal_marks,ext_marks_pred,color='red',label='Linear Regress
plt.legend()
```

Out[12]: <matplotlib.legend.Legend at 0x1edc9b1d990>



Generate 51 points for = 1 1 + exp(−3x),where x ∈ [−2, 3].Use this dataset to train sigmoid neuron using gradient descent learning algorithm. Draw two curves with different colours, for target and output(y) of the trained neuron.

In [13]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [14]:
```
x=np.linspace(-2,3,51)
y=1/(1+np.exp(-3*x))
```

In [15]:
```
x
```

Out[15]:
```
array([-2. , -1.9, -1.8, -1.7, -1.6, -1.5, -1.4, -1.3, -1.2, -1.1, -1. ,
       -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1,  0. ,  0.1,
        0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ,  1.1,  1.2,
        1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2. ,  2.1,  2.2,  2.3,
        2.4,  2.5,  2.6,  2.7,  2.8,  2.9,  3. ])
```

In [16]:
```
y
```

Out[16]:
```
array([0.00247262, 0.00333481, 0.00449627, 0.0060598 , 0.00816257,
       0.01098694, 0.01477403, 0.01984031, 0.02659699, 0.03557119,
       0.04742587, 0.06297336, 0.0831727 , 0.10909682, 0.14185106,
       0.18242552, 0.23147522, 0.2890505 , 0.35434369, 0.42555748,
       0.5       , 0.57444252, 0.64565631, 0.7109495 , 0.76852478,
       0.81757448, 0.85814894, 0.89090318, 0.9168273 , 0.93702664,
       0.95257413, 0.96442881, 0.97340301, 0.98015969, 0.98522597,
       0.98901306, 0.99183743, 0.9939402 , 0.99550373, 0.99666519,
       0.99752738, 0.99816706, 0.99864148, 0.99899323, 0.99925397,
       0.99944722, 0.99959043, 0.99969655, 0.99977518, 0.99983344,
       0.99987661])
```

In [17]:
```
class SigmoidNeuron:
  def __init__(self):
    self.w=np.random.rand(1)
    self.b=np.random.rand(1)

  def sigmoid(self,x):
    return 1/(1+np.exp(-x))

  def forward_propogation(self,x):
    return self.sigmoid(self.w*x+self.b)

  def back_propogation(self,x,y,lr):
    y_pred=self.forward_propogation(x)
    w_diff=(y_pred-y)*(y_pred)*(1-y_pred)*x
    b_diff=(y_pred-y)*(y_pred)*(1-y_pred)
    self.w=self.w-lr*w_diff
    self.b=self.b-lr*b_diff
```

In [18]:
```
model=SigmoidNeuron()
```

In [19]:
```
epochs=100
learning_rate=0.01
```

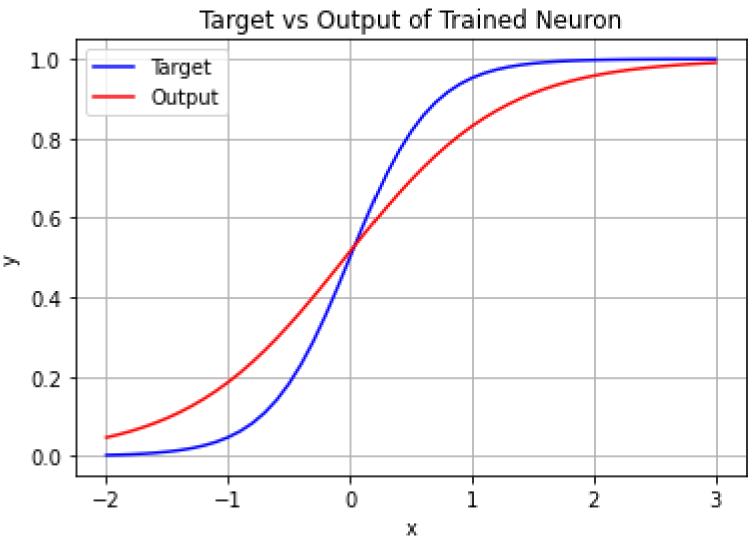
In [20]:
```
for epoch in range(epochs):
  for i in range(len(x)):
    model.back_propogation(x[i],y[i],learning_rate)
```

In [21]:
```
y_out=model.forward_propogation(x)
```

In [22]:
```
y_out
```

Out[22]:
```
array([0.04696534, 0.05433288, 0.06278004, 0.07243988, 0.08345372,
       0.09596886, 0.11013531, 0.12610125, 0.14400714, 0.16397853,
       0.18611739, 0.21049268, 0.23713021, 0.26600294, 0.29702242,
       0.33003272, 0.36480785, 0.40105373, 0.43841524, 0.47648837,
       0.51483673, 0.55301119, 0.59057066, 0.62710222, 0.66223845,
       0.69567065, 0.72715705, 0.75652594, 0.78367402, 0.80856101,
       0.8312016 , 0.85165584, 0.87001912, 0.88641242, 0.90097351,
       0.91384939, 0.9251901 , 0.93514389, 0.94385366, 0.95145448,
       0.95807204, 0.96382183, 0.96880878, 0.97312748, 0.9768625 ,
       0.98008901, 0.98287347, 0.98527439, 0.98734307, 0.98912434,
       0.9906573 ])
```

In [23]:
```
plt.plot(x, y, label='Target', color='blue')
plt.plot(x, y_out, label='Output', color='red')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Target vs Output of Trained Neuron')
plt.legend()
plt.grid(True)
plt.show()
```



In [ ]: