Sect2. Linear Regression

In [3]:

```
from tqdm import tqdm_notebook
import tensorflow as tf

x_train = [1,2,3]
y_train = [1,2,3]

W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = x_train*W + b

#cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - y_train))

#Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate = 0.01)
train = optimizer.minimize(cost)

sess = tf.Session()

sess.run(tf.global_variables_initializer())
```

In [8]:

```
# Fit the line
for step in tqdm_notebook(range(2001)):
    sess.run(train)
    if step % 200 == 0 or step < 5:
        # print(step, cost_val, W_val, b_val)
        print("step={step}, ₩t cost={cost_val},  ₩t W={W_val},  ₩t b={b_val}".format(
            step=step,
            cost_val=sess.run(cost),
            W_val=sess.run(W),
            b_val=sess.run(b)
        ));
```

```
step=0,          cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=1,          cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=2,          cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=3,          cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=4,          cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=200,        cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=400,        cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=600,        cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=800,        cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=1000,       cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=1200,       cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=1400,       cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=1600,       cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=1800,       cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
step=2000,       cost=3.069544618483633e-12,     W=[0.99999785],         b=[4.3506
05e-06]
```

Ex2. Plaeholder 사용

In [21]:

```
import tensorflow as tf

W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

X = tf.placeholder(tf.float32, shape=[None])
Y = tf.placeholder(tf.float32, shape=[None])

# Our hypothesis XW+b
hypothesis = X * W + b

# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))

# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

# Launch the graph in a session.
sess = tf.Session()

sess.run(tf.global_variables_initializer())
```

In [22]:

```
# Fit the line
steps     = []
cost_vals = []
W_vals    = []
b_vals    = []
for step in tqdm_notebook(range(2001)):
    cost_val, W_val, b_val, _ = sess.run([cost, W, b, train],
                                         feed_dict={X: [1, 2, 3],
                                                    Y: [1, 2, 3]})

    steps.append(step)
    cost_vals.append(cost_val)
    W_vals.append(W_val)
    b_vals.append(b_val)
    # W_vals.append(float(W_val))
    # b_vals.append(float(b_val))

    if step % 200 == 0 or step < 5:
        # print(step, cost_val, W_val, b_val)
        print("step={step}, ₩t cost={cost_val}, ₩t W={W_val}, ₩t b={b_val}".format(
            step=step, cost_val=cost_val, W_val=W_val, b_val=b_val
            ));
```

```
step=0,         cost=8.060706138610 84,         W=[-0.08663595],        b=[-0.190
21866]
step=1,         cost=6.3732733726501465,        W=[0.02239215],         b=[-0.142
94884]
step=2,         cost=5.039438724517822,         W=[0.11935351],         b=[-0.100
98554]
step=3,         cost=3.9851067066192627,        W=[0.2055866],          b=[-0.063
73997]
step=4,         cost=3.151705503463745,         W=[0.28228146],         b=[-0.030
68863]
step=200,       cost=0.003200454404577613,      W=[0.93445253],         b=[0.1490
0489]
step=400,       cost=0.0012220909120514989,     W=[0.95949554],         b=[0.0920
7611]
step=600,       cost=0.00046665058471262455,    W=[0.9749708],          b=[0.0568
9729]
step=800,       cost=0.00017818868218455464,    W=[0.9845334],          b=[0.0351
5899]
step=1000,      cost=6.804210715927184e-05,     W=[0.99044263],         b=[0.0217
2621]
step=1200,      cost=2.5981024009524845e-05,    W=[0.99409413],         b=[0.0134
2542]
step=1400,      cost=9.92135210253764e-06,      W=[0.9963505],          b=[0.0082
9613]
step=1600,      cost=3.788671165239066e-06,     W=[0.99774474],         b=[0.0051
2664]
step=1800,      cost=1.4468799918176956e-06,    W=[0.99860626],         b=[0.0031
6811]
step=2000,      cost=5.525607775780372e-07,     W=[0.99913865],         b=[0.0019
5792]
```

In [23]:

```
W_val, W_val[0], type(W_val[0])
```

Out[23]:

```
(array([0.99913865], dtype=float32), 0.99913865, numpy.float32)
```

In [24]:

```
type(b_val), str(b_val), float(b_val)
```

Out[24]:

```
(numpy.ndarray, '[0.00195792]', 0.0019579243380576372)
```

Ex3. Plaeholder 사용

In [25]:

```
for step in range(2001) :

    cost_val, W_val, b_val, _ = sess.run([cost, W, b, train],
                                         feed_dict={ X: [1,2,3,4,5],
                                                     Y : [2.1,3.1,4.1,5.1,6.1]})
    if step % 100 == 0 or step < 10 :
        print("step={step}, ₩t cost={cost_val}, ₩t W={W_val},  ₩t b={b_val}".format(
            step=step, cost_val=cost_val, W_val=W_val, b_val=b_val
            ));
```

```
step=0,          cost=1.211379051208496,         W=[1.0652107],          b=[0.0239
7044]
step=1,          cost=0.7836043238639832,        W=[1.1154261],          b=[0.0415
7839]
step=2,          cost=0.5337945818901062,        W=[1.1535376],          b=[0.0558
2126]
step=3,          cost=0.38769659399986267,       W=[1.1824101],          b=[0.0674
9257]
step=4,          cost=0.3020406663417816,        W=[1.2042303],          b=[0.0771
9812]
step=5,          cost=0.25161099433898926,       W=[1.2206677],          b=[0.0854
0034]
step=6,          cost=0.22171273827552795,       W=[1.2329968],          b=[0.0924
5227]
step=7,          cost=0.2037825584411621,        W=[1.2421904],          b=[0.0986
2341]
step=8,          cost=0.19283032417297363,       W=[1.2489911],          b=[0.1041
1952]
step=9,          cost=0.18594779074192047,       W=[1.2539659],          b=[0.1090
9766]
step=100,        cost=0.09611531347036362,       W=[1.2005966],          b=[0.3757
827]
step=200,        cost=0.04882347211241722,       W=[1.1429689],          b=[0.5838
37]
step=300,        cost=0.02480069361627102,       W=[1.1018964],          b=[0.7321
2117]
step=400,        cost=0.012597923167049885,      W=[1.0726234],          b=[0.8378
063]
step=500,        cost=0.006399328354746103,      W=[1.0517602],          b=[0.9131
2975]
step=600,        cost=0.0032506585121154785,     W=[1.0368904],          b=[0.9668
1386]
step=700,        cost=0.0016512200236320496,     W=[1.0262924],          b=[1.0050
758]
step=800,        cost=0.0008387707057408988,     W=[1.0187391],          b=[1.0323
457]
step=900,        cost=0.0004260689893271774,     W=[1.0133557],          b=[1.0517
814]
step=1000,       cost=0.00021643241052515805,    W=[1.0095189],          b=[1.0656
335]
step=1100,       cost=0.00010994078911608085,    W=[1.0067843],          b=[1.0755
064]
step=1200,       cost=5.5845575843704864e-05,    W=[1.0048352],          b=[1.0825
43]
step=1300,       cost=2.8369115170789883e-05,    W=[1.0034462],          b=[1.0875
579]
step=1400,       cost=1.440978485334199e-05,     W=[1.0024562],          b=[1.0911
324]
step=1500,       cost=7.320407803490525e-06,     W=[1.0017506],          b=[1.0936
795]
step=1600,       cost=3.719134838320315e-06,     W=[1.0012479],          b=[1.0954
949]
step=1700,       cost=1.8895603943747 12e-06,    W=[1.0008895],          b=[1.0967
888]
step=1800,       cost=9.60047145781573e-07,      W=[1.0006341],          b=[1.0977
11]
step=1900,       cost=4.878638151240011e-07,     W=[1.000452],           b=[1.0983
683]
step=2000,       cost=2.47953067855633e-07,      W=[1.0003222],          b=[1.0988
368]
```

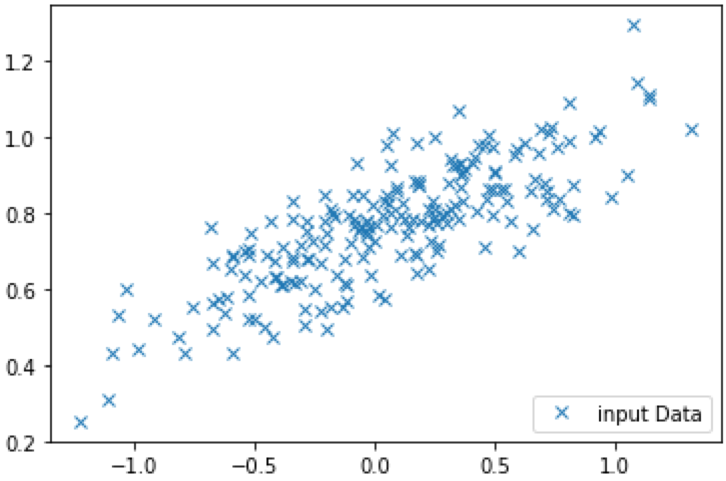
Ex4. Linear Regression

In [26]:

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

number_of_points = 200
x_point = []
y_point = []

w = 0.25
b = 0.75

for i in range(number_of_points) :
    x = np.random.normal(0.0, 0.5)
    y = w*x + b + np.random.normal(0.0, 0.1)
    x_point.append([x])
    y_point.append([y])
```

In [27]:

```
plt.plot(x_point, y_point, 'x', label = 'input Data')
plt.legend(loc=4)
plt.show()
```



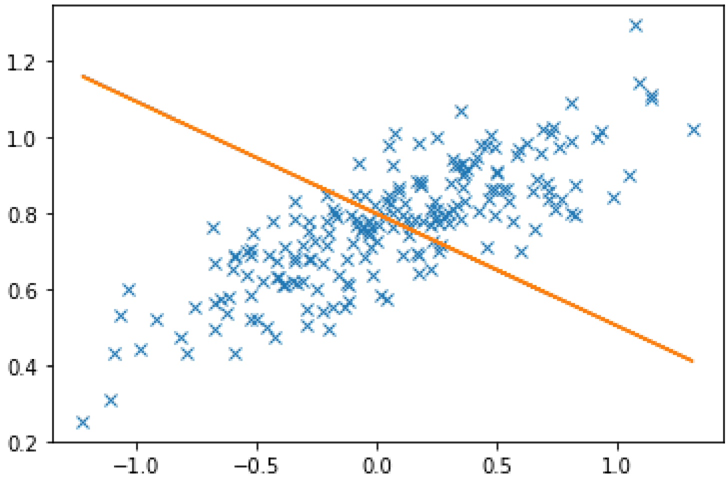
In [29]:

```
W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
B = tf.Variable(tf.zeros([1]))
y = W * x_point + B

cost_function = tf.reduce_mean(tf.square(y - y_point))

optimizer = tf.train.GradientDescentOptimizer(learning_rate = 0.5)

train = optimizer.minimize(cost_function)

model = tf.global_variables_initializer()
```

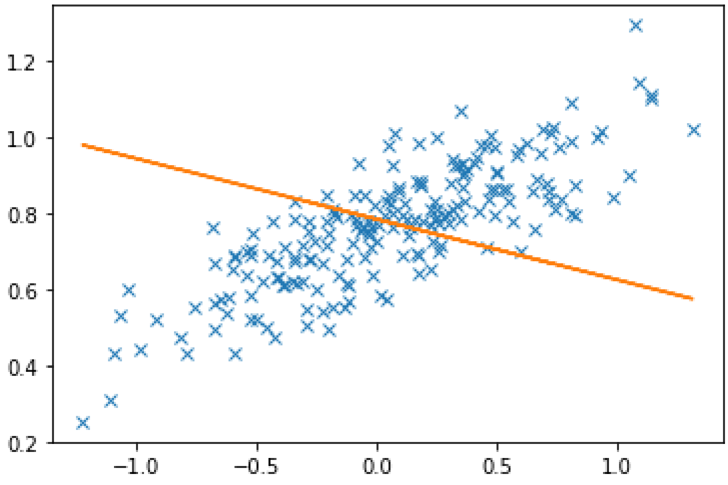
In [31]:

```
with tf.Session() as sess :
    sess.run(model)
    for step in range(0, 2001) :
        sess.run(train)
        if (step % 200) == 0 or step < 5 :
            print("₩n y = {w} x + {b} ".format(w=sess.run(W), b=sess.run(B)))
            plt.plot(x_point, y_point, 'x', label = 'step={}'.format(step))
            plt.plot(x_point, sess.run(W)*x_point+sess.run(B)) #라인
            plt.show()
```

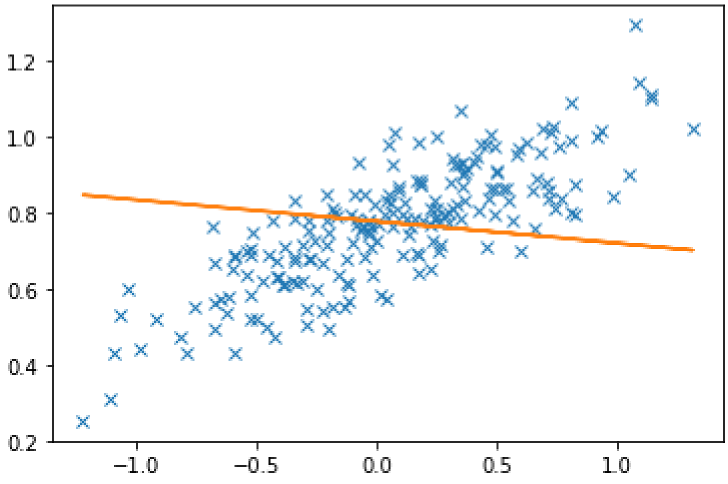
```
y = [-0.29585424] x + [0.7978536]
```



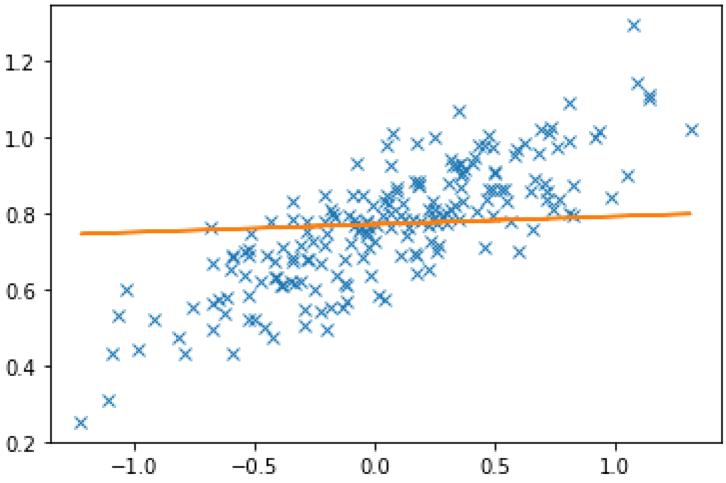
```
y = [-0.15988411] x + [0.7845333]
```



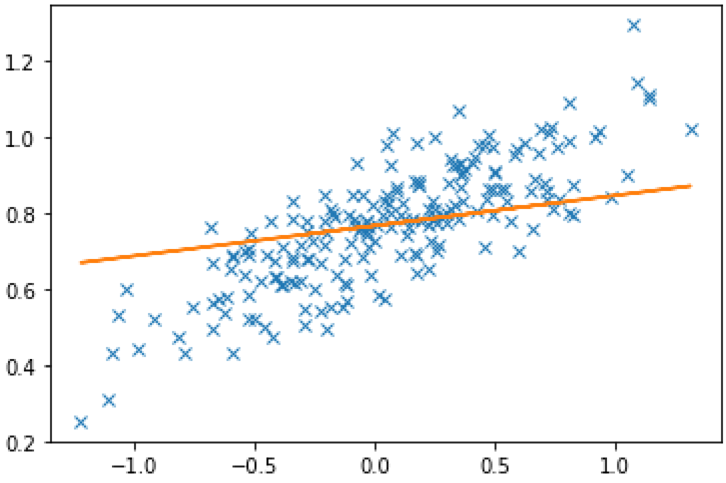
```
y = [-0.05690182] x + [0.77697957]
```



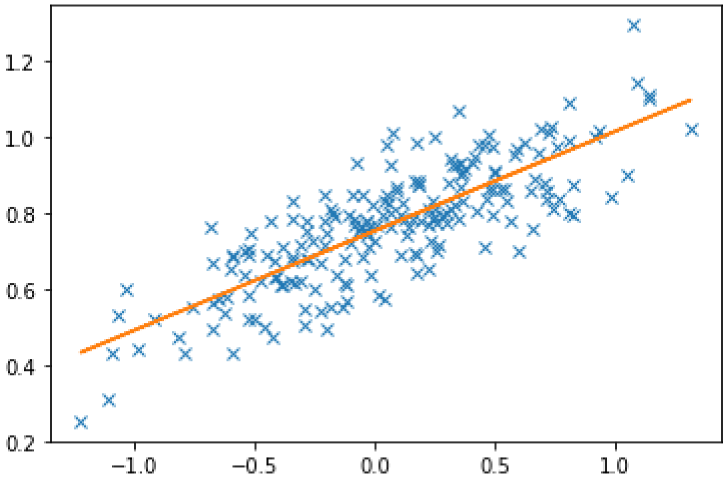
```
y = [0.02095502] x + [0.7712584]
```



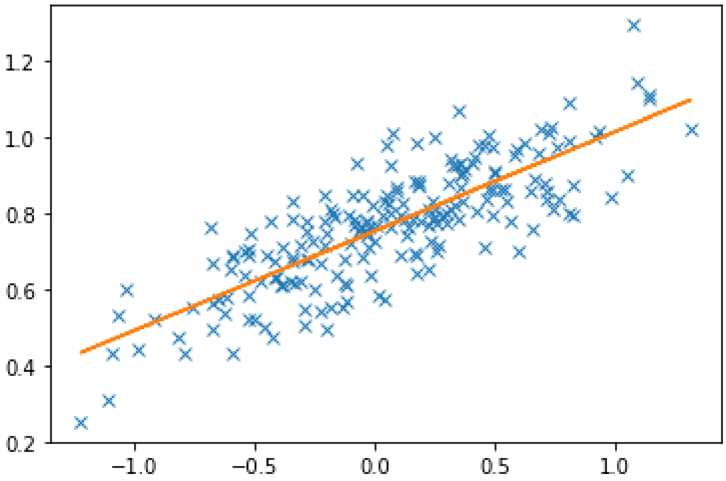
```
y = [0.07981704] x + [0.76693314]
```



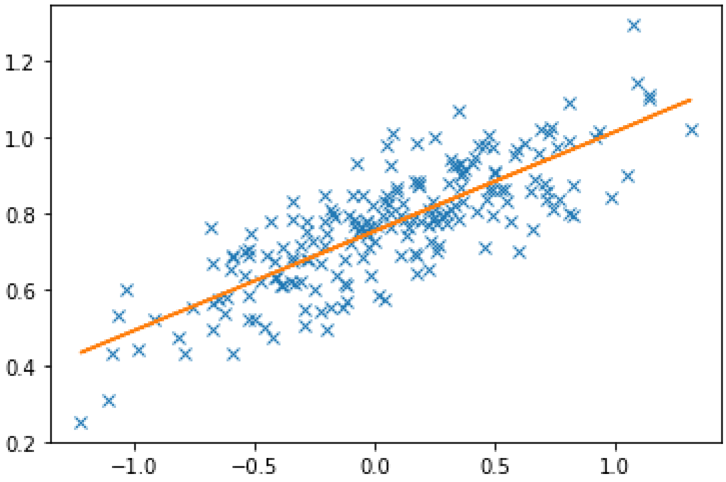
```
y = [0.2622214] x + [0.7535297]
```



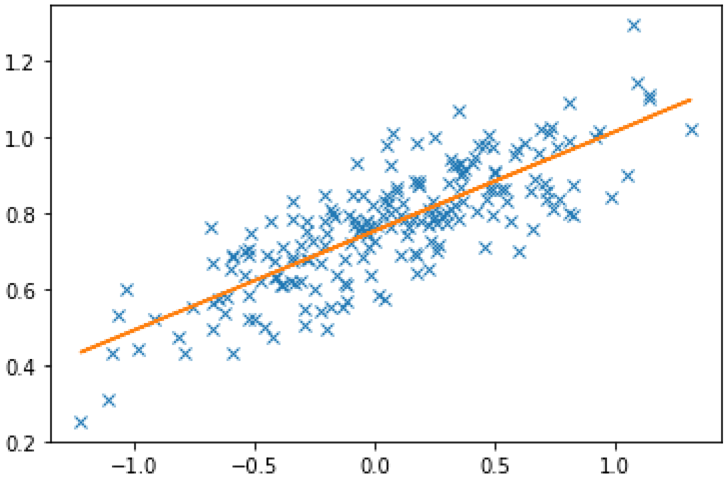
```
y = [0.2622214] x + [0.7535297]
```



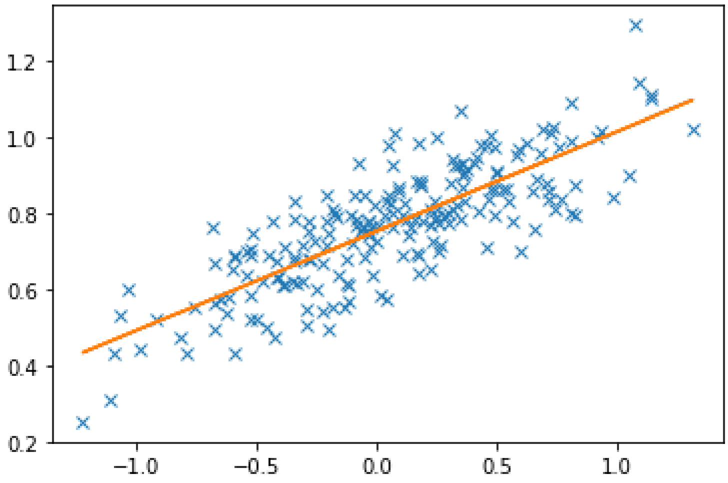
```
y = [0.2622214] x + [0.7535297]
```



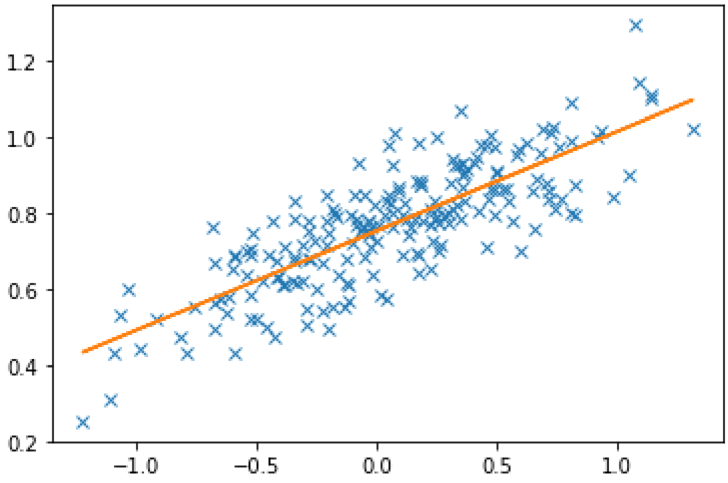
```
y = [0.2622214] x + [0.7535297]
```



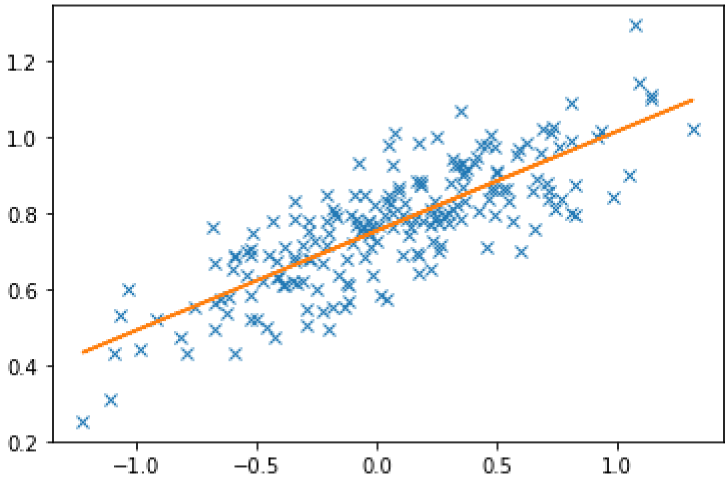
```
y = [0.2622214] x + [0.7535297]
```



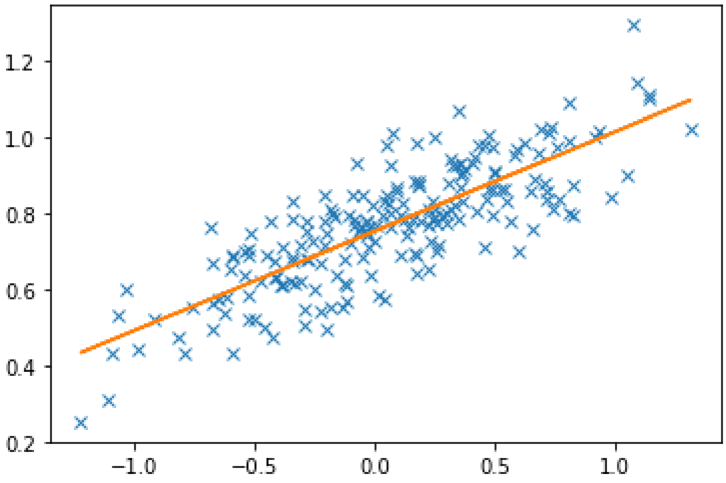
```
y = [0.2622214] x + [0.7535297]
```



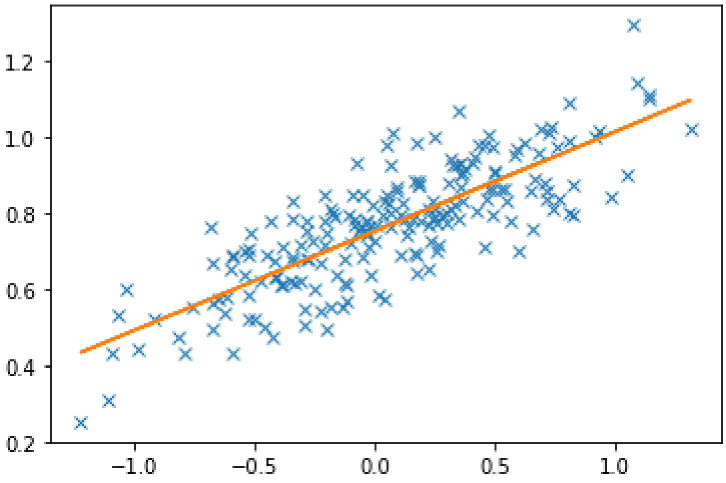
```
y = [0.2622214] x + [0.7535297]
```



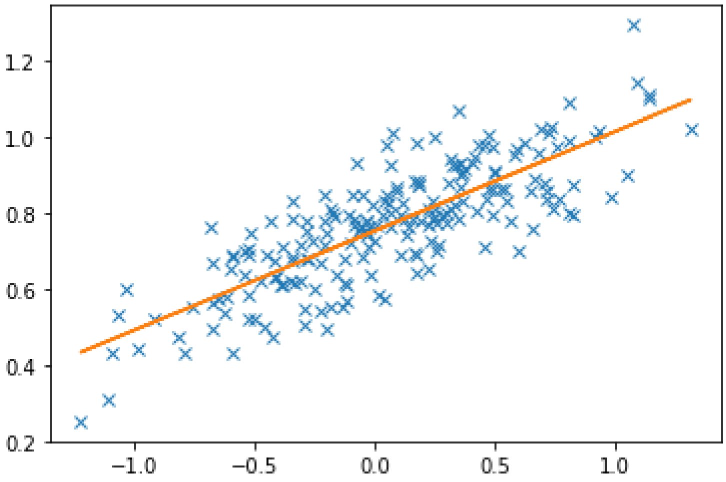
```
y = [0.2622214] x + [0.7535297]
```



```
y = [0.2622214] x + [0.7535297]
```



```
y = [0.2622214] x + [0.7535297]
```



In [ ]: