In [1]:
```
import pandas as pd
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
from keras.utils import np_utils
from sklearn.preprocessing import StandardScaler, MaxAbsScaler,Robu
stScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,r2_score
from sklearn.ensemble import AdaBoostRegressor
import matplotlib.pyplot as plt
import math
```

```
Using TensorFlow backend.
```

In [2]:
```
def create_dataset_X(arr, size = 7):
    X=[]
    for i in range(len(arr)-size):
        def_arr = arr.copy()
#         for j in range(size):
#             for n in def_arr[i+j,:-1]:
#                 n = n*def_arr[i+j,-1]

        a=12
        for j in range(size-1):
            def_arr[i+j,:a] = def_arr[i+j,:a]-def_arr[i+size-1,:a]
#행, 열
#         a=-2
#         for j in range(size-1):
#             def_arr[i+j,a:] = def_arr[i+j,a:]-def_arr[i+size-1,a:]
#행, 열
        X.append(def_arr[i:i+size,1:])

    return np.array(X)
```

In [3]:
```
def create_dataset_Y(arr, size = 7):
    Y=[]
    for i in range(len(arr)-size):
#         Y.append(arr.T[-2:,i+size]-arr.T[-2:,i+size-1])
        Y.append(arr.T[-2:,i+size])
    return np.array(Y)
```

In [4]:
```
def Month_Day(df):
    df_m = df[df['일시']<'0000-00-00']
    for i in range(2016, 2019):
        for j in ['01','02','03','04','05','06','07','08','09','10'
,'11','12']:


            df_s = df[df['일시']<=str(i)+'-'+j+'-31']

            df_s['봄'] = (((int(j)+10)%12) <3)
            df_s['여름'] = (((int(j)+7)%12) <3)
            df_s['가을']  = (((int(j)+4)%12) <3)
            df_s['겨울'] = (((int(j)+1)%12) < 3)

            df_m = pd.merge(df_m, df_s, how = 'outer')

            df = df[df['일시']>str(i)+'-'+j+'-31']
    return df_m
```

In [5]:
```
def maxSeries(num, series):
    series_values = series.values
    rseries = -series.values

    series_values = series_values>0
    rseries = rseries>0

    '''for i in range(len(series_values)):
        if(series_values[i]>num):
            rseries[i] = num
        else:
            series_values[i] = num
'''
    return series_values, rseries
```

In [6]:
```
def float2int(arr_float):
    for i in arr_float:
        for j in range(len(i)):
            i[j] = int(round(i[j]))
    return arr_float
```

In [7]:
```
class CustomHistory(keras.callbacks.Callback):
    def init(self):
        self.train_loss = []
        self.val_loss = []

    def on_epoch_end(self, batch, logs={}):
        self.train_loss.append(logs.get('loss'))
        self.val_loss.append(logs.get('val_loss'))
```

In [9]:
```
df_i = []
for i in range(1992,2019):
    df_i.append(pd.read_csv('./서울날씨/서울%s.csv' %str(i)))
```

In [10]:
```
features = [
             '일시',
             '강수 계속시간(hr)',
             '9-9강수(mm)',
             '평균 증기압(hPa)',
             '평균 중하층운량(1/10)',
             '최대 풍속 풍향(16방위)',
             '최대 풍속(m/s)',
             '평균 상대습도(%)',
             '최저기온(°C)',
             '최고기온(°C)',
            ]
```

In [11]:
```
df = df_i[0][features]
for i in range(len(df_i)):
    df_i[i] = df_i[i][features]
    df = pd.merge(df, df_i[i],how = 'outer')
```

In [8]:
```
df = pd.read_csv('./서울 날씨.csv')

df = Month_Day(df)
```

```
/Users/sangok-lee/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
/Users/sangok-lee/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  # Remove the CWD from sys.path while we load stuff.
/Users/sangok-lee/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  # This is added back by InteractiveShellApp.init_path()
/Users/sangok-lee/anaconda3/lib/python3.7/site-packages/ipykernel_
launcher.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pan
das-docs/stable/indexing.html#indexing-view-versus-copy
  if sys.path[0] == '':
```

In [89]:
```
df['최고기온(°C)'] = df['최고기온(°C)'].fillna(method = 'ffill')
df['최저기온(°C)'] = df['최저기온(°C)'].fillna(method = 'ffill')

df['최대 풍속 풍향W'], df['최대 풍속 풍향E'] = maxSeries(0, np.cos(df['최대 풍속 풍향(16방위)']*math.pi/180))
df['최대 풍속 풍향N'], df['최대 풍속 풍향S'] = maxSeries(0, np.sin(df['최대 풍속 풍향(16방위)']*math.pi/180))
df['최대 풍속 풍향W'] = np.square(df['최대 풍속 풍향W'])* df['최대 풍속(m/s)']
df['최대 풍속 풍향E'] = np.square(df['최대 풍속 풍향E'])* df['최대 풍속(m/s)']
df['최대 풍속 풍향N'] = np.square(df['최대 풍속 풍향N'])* df['최대 풍속(m/s)']
df['최대 풍속 풍향S'] = np.square(df['최대 풍속 풍향S'])* df['최대 풍속(m/s)']


df['평균 증기압(hPa)'] =df['평균 증기압(hPa)'].fillna(method = 'ffill')
df['평균 상대습도(%)'] =df['평균 상대습도(%)'].fillna(method = 'ffill')
```

```
/Users/sangok-lee/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: RuntimeWarning: invalid value encountered in greater
  """
/Users/sangok-lee/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:6: RuntimeWarning: invalid value encountered in greater
```

In [90]:
```
df.columns
```

Out[90]:
```
Index(['지점', '일시', '최저기온(°C)', '최고기온(°C)', '강수 계속시간(hr)', '일강수량(mm)',
       '최대 순간 풍속(m/s)', '최대 순간 풍속 풍향(16방위)', '최대 풍속(m/s)', '최대 풍속 풍향(16방위)',
       '평균 풍속(m/s)', '최다풍향(16방위)', '평균 이슬점온도(°C)', '최소 상대습도(%)', '평균 상대습도(%)',
       '평균 증기압(hPa)', '최고 해면기압(hPa)', '최저 해면기압(hPa)', '합계 일조 시간(hr)',
       '합계 일사(MJ/m2)', '일 최심신적설(cm)', '일 최심적설(cm)', '평균 전운량(1/10)',
       '평균 중하층운량(1/10)', '평균 지면온도(°C)', '합계 대형증발량(mm)', '합계 소형증발량(mm)',
       '9-9강수(mm)', '안개 계속시간(hr)', '봄', '여름', '가을', '겨울', '최대 풍속 풍향W',
       '최대 풍속 풍향E', '최대 풍속 풍향N', '최대 풍속 풍향S'],
      dtype='object')
```

In [91]:
```
features = [

             '일강수량(mm)' ,
             '강수 계속시간(hr)',
             '평균 증기압(hPa)',
             '일 최심신적설(cm)',
             '최대 풍속 풍향E',
             '최대 풍속 풍향W',
             '최대 풍속 풍향N',
             '최대 풍속 풍향S',
             '평균 상대습도(%)',
             '최저기온(°C)',
             '최고기온(°C)',
            ]
```

In [114]:
```
df_f = df[features]

df_f = df_f.fillna(0)

sc = StandardScaler()
msc = MaxAbsScaler()

df_values = df_f.values.copy()
X_sc = msc.fit_transform(df_values)
X_ds = create_dataset_X(X_sc, 3)      #몇일 이전으로 할것인지

df_values = df_f.values.copy()
Y_ds = create_dataset_Y(df_values,3)

Y_ds = msc.fit_transform(Y_ds)
```

In [105]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X_ds, Y_ds, test_s
ize = 0.2, random_state=3)
#X_train,X_val,Y_train,Y_val = train_test_split(X_train,Y_train,tes
t_size=0.2, random_state=1)
```

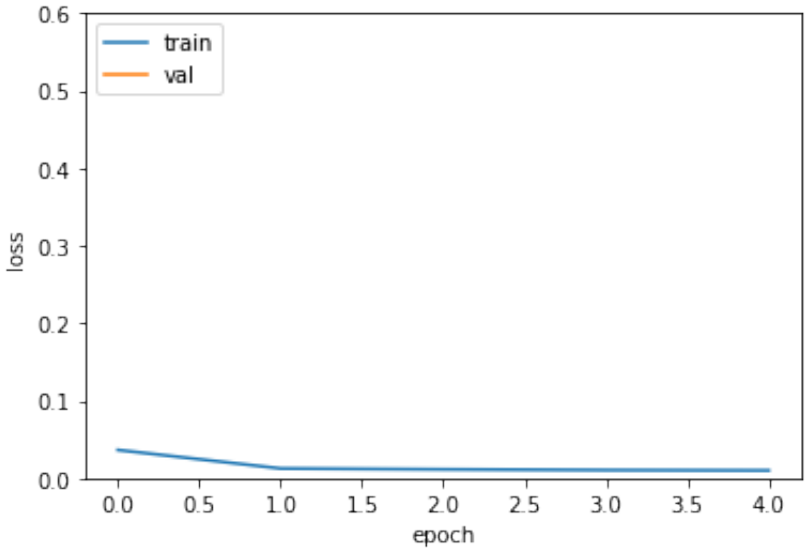
In [106]:
```
vec_size =2
model = Sequential()
model.add(LSTM(32, batch_input_shape=(1, 3,11), stateful=True, acti
vation='tanh'))
model.add(Dropout(0.4))
model.add(Dense(20))
model.add(Dense(7, activation='linear'))

model.add(Dense(vec_size))
```

In [107]:
```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=
[keras.metrics.mean_squared_logarithmic_error])
```

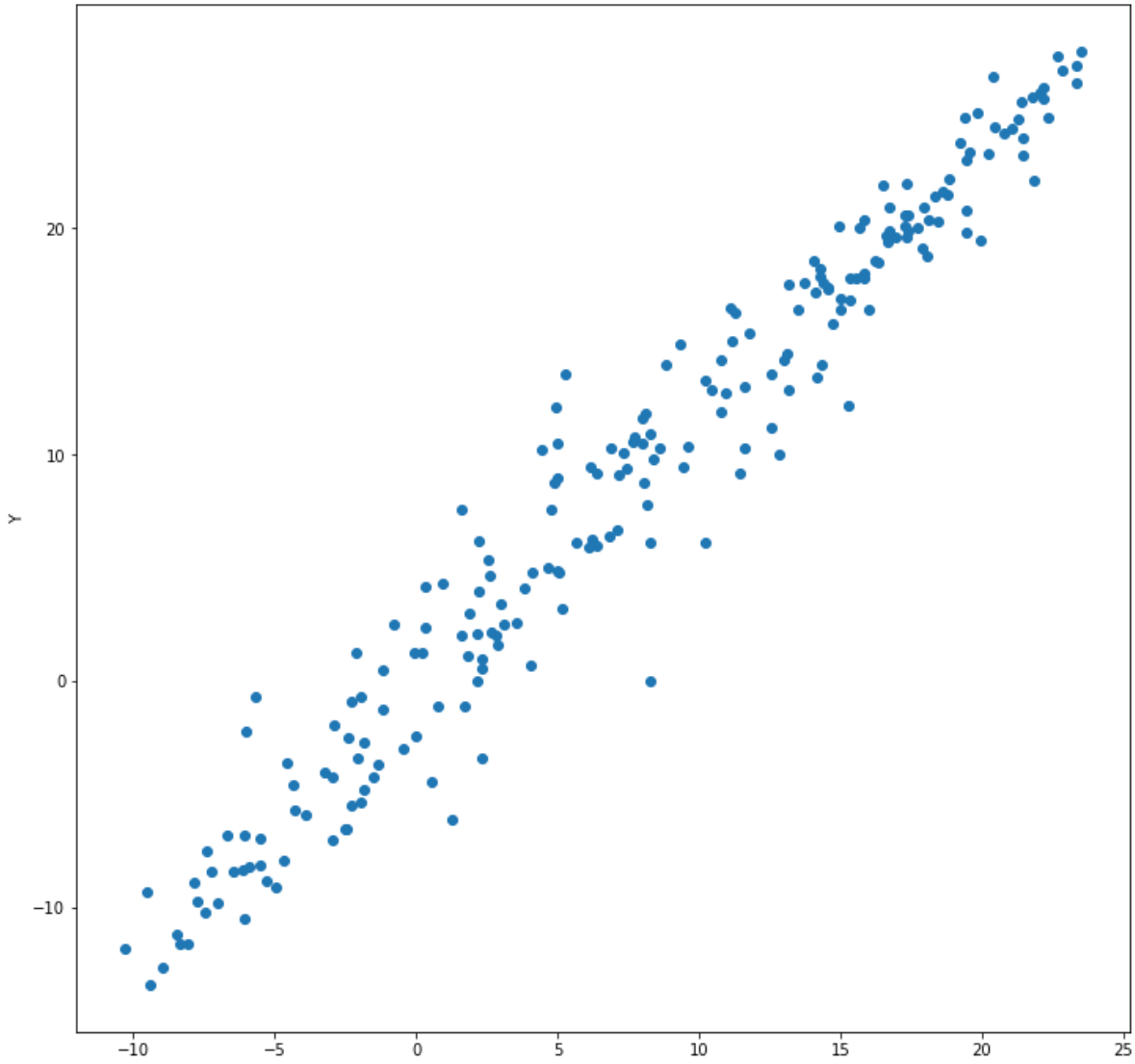
In [108]:
```
# 4. 모델 학습시키기
custom_hist = CustomHistory()
custom_hist.init()
for i in range(5):
    print('epochs =', i+1)

    model.fit(X_train, Y_train, epochs=1, batch_size=1, shuffle=Tru
e, callbacks=[custom_hist])#, validation_data=(X_val, Y_val))
    model.reset_states()
```

```
epochs = 1
Epoch 1/1
874/874 [==============================] - 7s 8ms/step - loss: 0.0
365 - mean_squared_logarithmic_error: 0.0144
epochs = 2
Epoch 1/1
874/874 [==============================] - 5s 5ms/step - loss: 0.0
127 - mean_squared_logarithmic_error: 0.0047
epochs = 3
Epoch 1/1
874/874 [==============================] - 4s 5ms/step - loss: 0.0
115 - mean_squared_logarithmic_error: 0.0044
epochs = 4
Epoch 1/1
874/874 [==============================] - 4s 4ms/step - loss: 0.0
106 - mean_squared_logarithmic_error: 0.0040
epochs = 5
Epoch 1/1
874/874 [==============================] - 4s 4ms/step - loss: 0.0
101 - mean_squared_logarithmic_error: 0.0038
```

In [109]:
```
# 5. 학습과정 살펴보기
plt.plot(custom_hist.train_loss)
plt.plot(custom_hist.val_loss)
plt.ylim(0.0, 0.60)
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

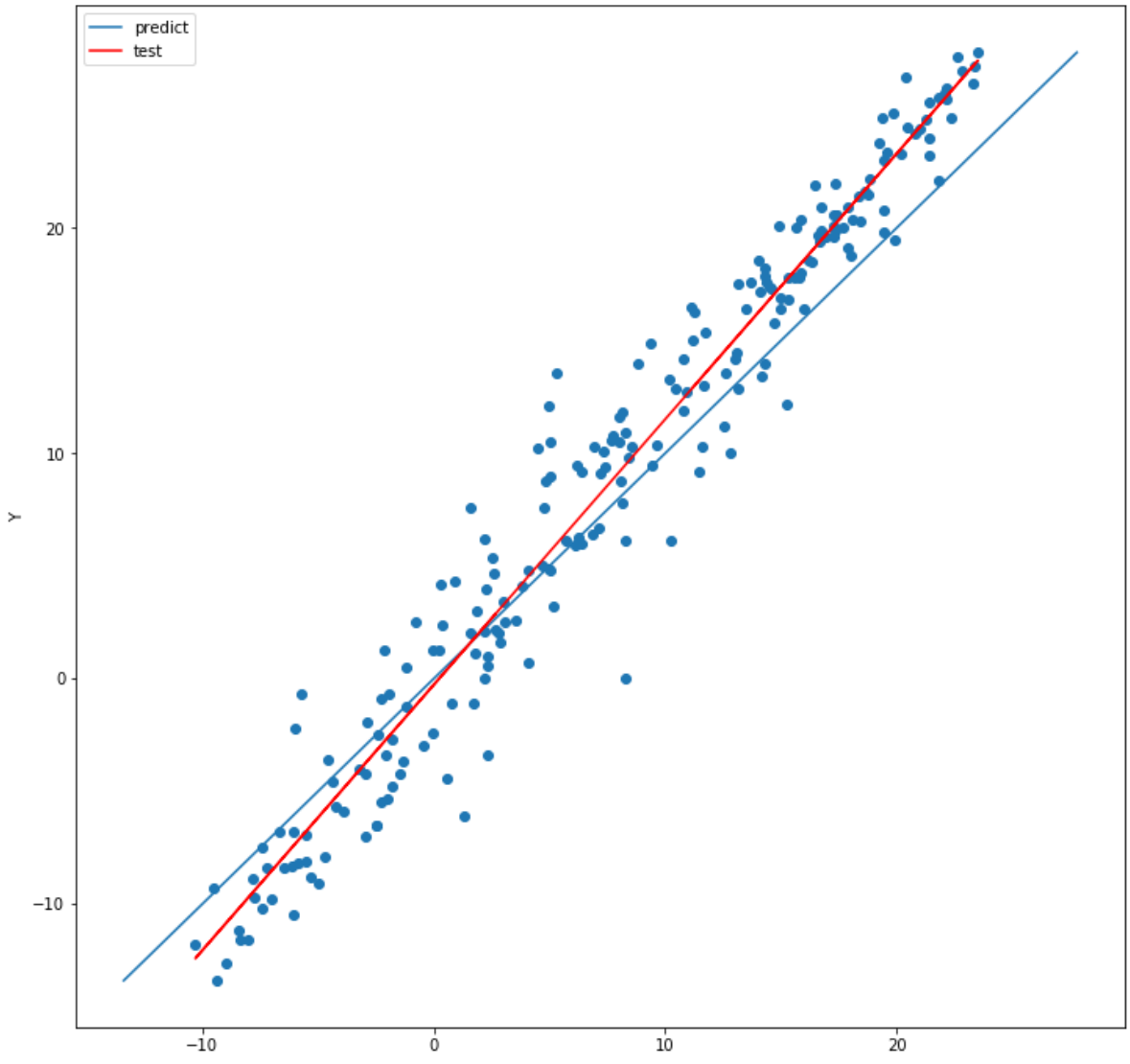


In [118]:
```
 tem = 0# 0최저온도, 1최고온도

pre = msc.inverse_transform(model.predict(X_test,batch_size=1))
Y = msc.inverse_transform(Y_test)
Y_PLOT = [min(Y[:,tem]),max(Y[:,tem])]
Y_PLOT_M = [min(Y[:,tem])-2,max(Y[:,tem])-2]
pre_PLOT =[min(pre[:,tem]),max(pre[:,tem])]
PLOT_Z = [0.5,0.5]
# 5. 학습결과 확인하기)
plt.figure(figsize=(12,12)).add_subplot(111)
plt.scatter(pre[:,tem],Y[:,tem])
plt.ylabel('Y')
plt.show()
des = pd.Series(pre[:,tem]-Y[:,tem])
des.describe()
```

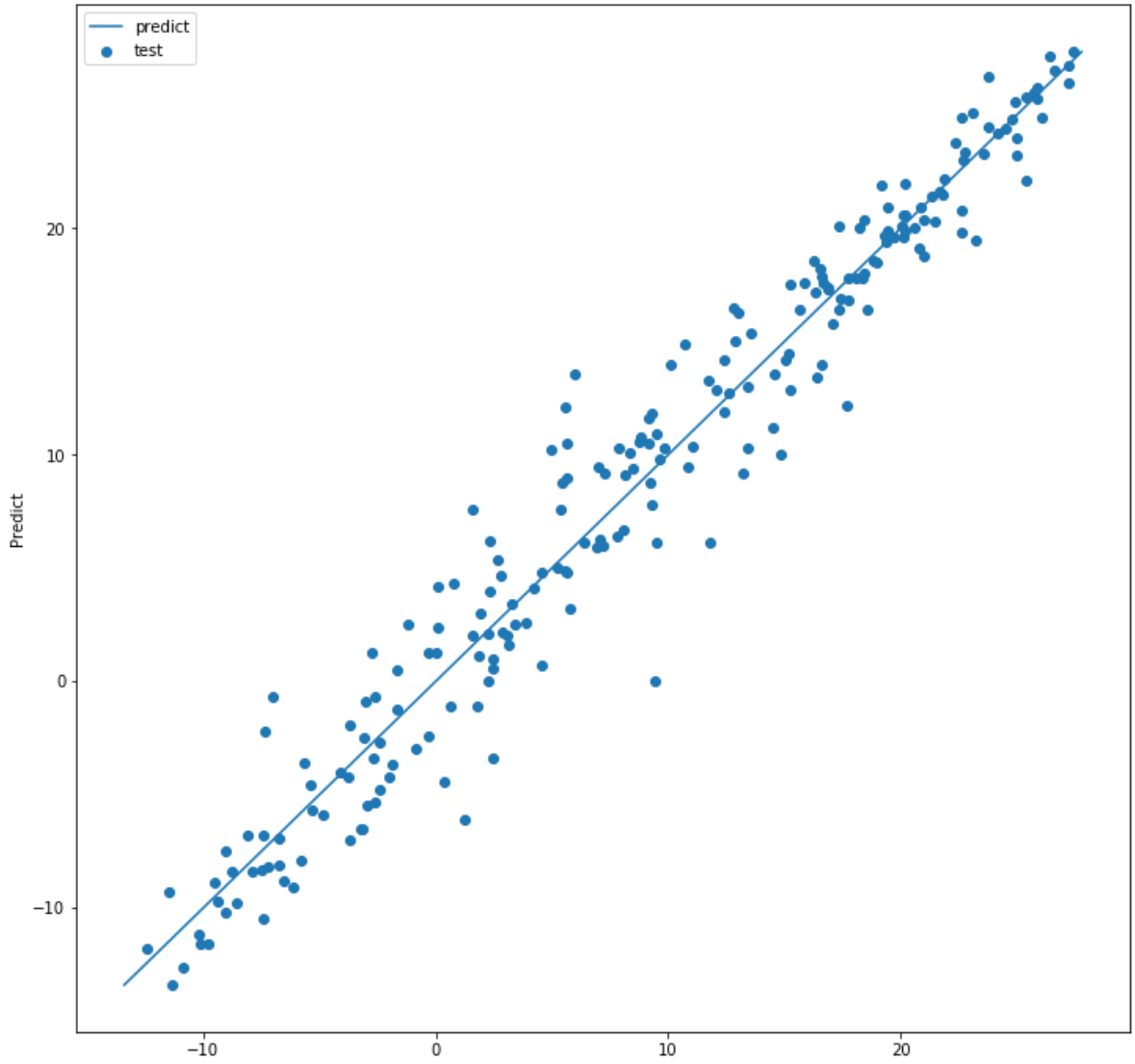


Out[118]:
```
count    219.000000
mean      -1.119942
std        2.868307
min       -8.302598
25%       -3.306548
50%       -1.415368
75%        0.848869
max        8.249768
dtype: float64
```

In [119]:
```
from sklearn.linear_model import LinearRegression
slr = LinearRegression()

slr.fit(pre[:,tem:tem+1], Y[:,tem:tem+1])  #X,Y
# 5. 학습결과 확인하기)
plt.figure(figsize=(12,12)).add_subplot(111)
plt.scatter(pre[:,tem],Y[:,tem])
plt.plot(Y_PLOT, Y_PLOT)
plt.plot(pre[:,tem:tem+1], slr.predict(pre[:,tem:tem+1]), color='red')
plt.ylabel('Y')
plt.legend(['predict', 'test'], loc='upper left')
plt.show()
```



In [120]:
```
pre = pre*slr.coef_[0]
for i in range(len(pre)):
    pre[i,tem] = pre[i,tem] + slr.intercept_
# 5. 학습결과 확인하기)
plt.figure(figsize=(12,12)).add_subplot(111)
plt.scatter(pre[:,tem],Y[:,tem])
plt.plot(Y_PLOT, Y_PLOT)

plt.ylabel('Y')
plt.ylabel('Predict')
plt.legend(['predict', 'test'], loc='upper left')
plt.show()

des = pd.Series(pre[:,tem]-Y[:,tem])
des.describe()
des = pd.Series(pre[:,tem]-Y[:,tem])
print(des.describe(),r2_score(pre[:,tem],Y[:,tem]))
```




```
count    2.190000e+02
mean     1.104999e-07
std      2.349263e+00
min     -7.629902e+00
25%     -1.499664e+00
50%      5.541878e-02
75%      1.246616e+00
max      9.449098e+00
dtype: float64 0.953302058824791
```

In [ ]:

In [ ]: