### Deep Learning approaches to build a predictive model

#import necessary libraries  
import pandas as pd  
import numpy as np  
import tensorflow as tf  
from tensorflow import keras  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from keras.callbacks import EarlyStopping

/Users/gangalingden/anaconda3/lib/python3.6/site-packages/h5py/\_\_init\_\_.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.  
 from .\_conv import register\_converters as \_register\_converters  
Using TensorFlow backend.

#load the data  
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data'  
col\_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',  
 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV' ]  
house\_df = pd.read\_csv(url, sep= '\s+', names=col\_names )  
house\_df.head()

CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX \  
0 0.00632 18.0 2.31 0 0.538 6.575 65.2 4.0900 1 296.0   
1 0.02731 0.0 7.07 0 0.469 6.421 78.9 4.9671 2 242.0   
2 0.02729 0.0 7.07 0 0.469 7.185 61.1 4.9671 2 242.0   
3 0.03237 0.0 2.18 0 0.458 6.998 45.8 6.0622 3 222.0   
4 0.06905 0.0 2.18 0 0.458 7.147 54.2 6.0622 3 222.0   
  
 PTRATIO B LSTAT MEDV   
0 15.3 396.90 4.98 24.0   
1 17.8 396.90 9.14 21.6   
2 17.8 392.83 4.03 34.7   
3 18.7 394.63 2.94 33.4   
4 18.7 396.90 5.33 36.2

'''====== Data Exploration and Preprocessing ======'''  
  
#shape of dataframe  
house\_df.shape

(506, 14)

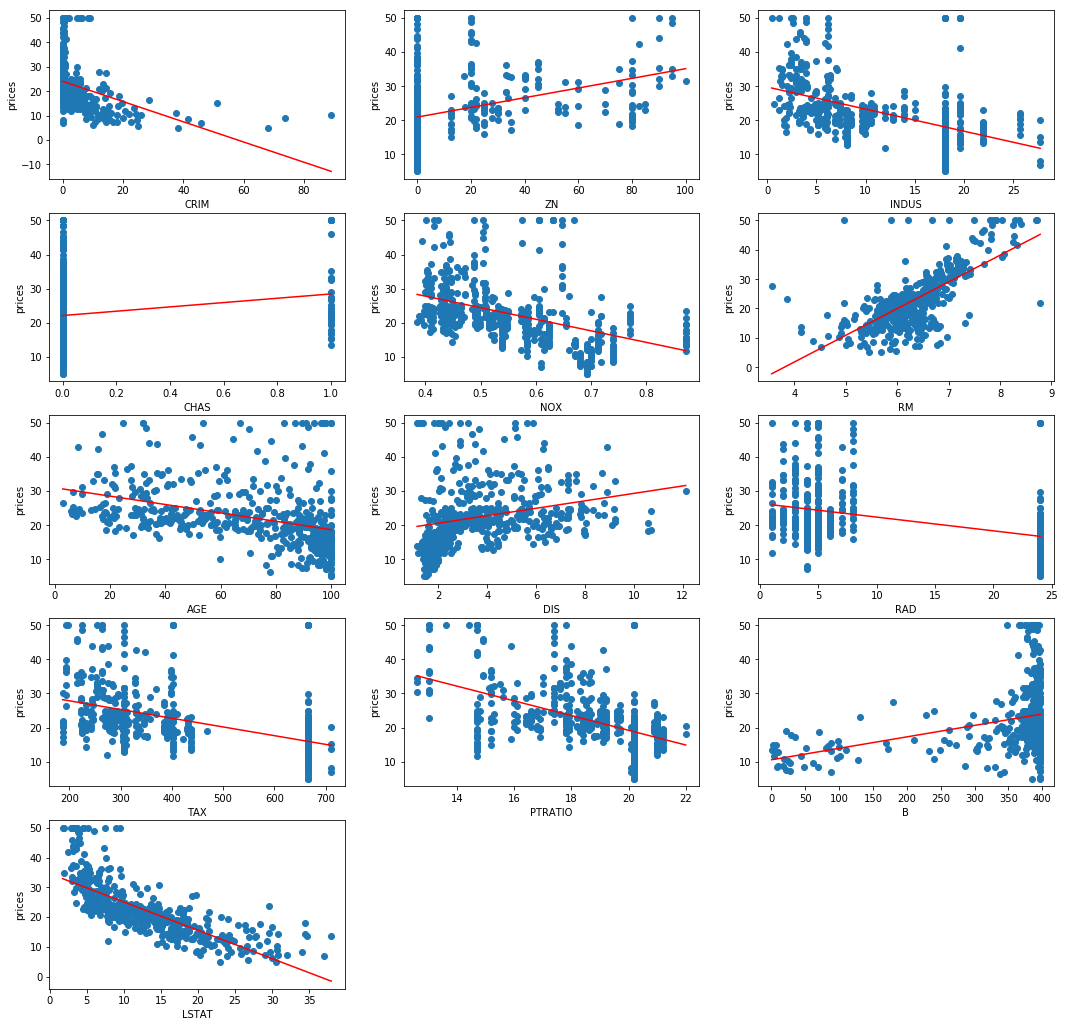
#check there are any NAN values  
house\_df.isnull().values.any()

False

'''=== show the statistics analysis of each attributes ==='''  
  
#descriptive statistics   
house\_df.describe()

CRIM ZN INDUS CHAS NOX RM \  
count 506.000000 506.000000 506.000000 506.000000 506.000000 506.000000   
mean 3.613524 11.363636 11.136779 0.069170 0.554695 6.284634   
std 8.601545 23.322453 6.860353 0.253994 0.115878 0.702617   
min 0.006320 0.000000 0.460000 0.000000 0.385000 3.561000   
25% 0.082045 0.000000 5.190000 0.000000 0.449000 5.885500   
50% 0.256510 0.000000 9.690000 0.000000 0.538000 6.208500   
75% 3.677082 12.500000 18.100000 0.000000 0.624000 6.623500   
max 88.976200 100.000000 27.740000 1.000000 0.871000 8.780000   
  
 AGE DIS RAD TAX PTRATIO B \  
count 506.000000 506.000000 506.000000 506.000000 506.000000 506.000000   
mean 68.574901 3.795043 9.549407 408.237154 18.455534 356.674032   
std 28.148861 2.105710 8.707259 168.537116 2.164946 91.294864   
min 2.900000 1.129600 1.000000 187.000000 12.600000 0.320000   
25% 45.025000 2.100175 4.000000 279.000000 17.400000 375.377500   
50% 77.500000 3.207450 5.000000 330.000000 19.050000 391.440000   
75% 94.075000 5.188425 24.000000 666.000000 20.200000 396.225000   
max 100.000000 12.126500 24.000000 711.000000 22.000000 396.900000   
  
 LSTAT MEDV   
count 506.000000 506.000000   
mean 12.653063 22.532806   
std 7.141062 9.197104   
min 1.730000 5.000000   
25% 6.950000 17.025000   
50% 11.360000 21.200000   
75% 16.955000 25.000000   
max 37.970000 50.000000

'''=== Show the linear relationship between features and price (MEDV). Thus, it provides that how the scattered   
 they are and which features has more impact in prediction of house price. ==='''  
  
# visiualize all variables with price(MEDV)  
from scipy import stats  
#creates figure  
plt.figure(figsize=(18, 18))  
  
for i, col in enumerate(house\_df.columns[0:13]): #iterates over all columns except for price column (last one)  
 plt.subplot(5, 3, i+1) # each row three figure  
 x = house\_df[col] #x-axis  
 y = house\_df['MEDV'] #y-axis  
 plt.plot(x, y, 'o')  
   
 # Create regression line  
 plt.plot(np.unique(x), np.poly1d(np.polyfit(x, y, 1)) (np.unique(x)), color='red')  
 plt.xlabel(col) # x-label  
 plt.ylabel('prices') # y-label



# separate the training and target variable  
feature = house\_df.iloc[:,0:13] # training variables  
target = house\_df.iloc[:,13] # target varible  
print(feature.head())  
print('\n',target.head())

CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX \  
0 0.00632 18.0 2.31 0 0.538 6.575 65.2 4.0900 1 296.0   
1 0.02731 0.0 7.07 0 0.469 6.421 78.9 4.9671 2 242.0   
2 0.02729 0.0 7.07 0 0.469 7.185 61.1 4.9671 2 242.0   
3 0.03237 0.0 2.18 0 0.458 6.998 45.8 6.0622 3 222.0   
4 0.06905 0.0 2.18 0 0.458 7.147 54.2 6.0622 3 222.0   
  
 PTRATIO B LSTAT   
0 15.3 396.90 4.98   
1 17.8 396.90 9.14   
2 17.8 392.83 4.03   
3 18.7 394.63 2.94   
4 18.7 396.90 5.33   
  
 0 24.0  
1 21.6  
2 34.7  
3 33.4  
4 36.2  
Name: MEDV, dtype: float64

'''=== Noramlization the features. Since it is seen that features have different ranges, it is best practice to  
normalize/standarize the feature before using them in the model ==='''  
  
#feature normalization  
normalized\_feature = keras.utils.normalize(feature.values)  
print(normalized\_feature)

[[1.26388341e-05 3.59966795e-02 4.61957387e-03 ... 3.05971776e-02  
 7.93726783e-01 9.95908132e-03]  
 [5.78529889e-05 0.00000000e+00 1.49769546e-02 ... 3.77071843e-02  
 8.40785474e-01 1.93620036e-02]  
 [5.85729947e-05 0.00000000e+00 1.51744622e-02 ... 3.82044450e-02  
 8.43137761e-01 8.64965806e-03]  
 ...  
 [1.23765824e-04 0.00000000e+00 2.43009593e-02 ... 4.27762066e-02  
 8.08470305e-01 1.14884669e-02]  
 [2.24644719e-04 0.00000000e+00 2.44548909e-02 ... 4.30471676e-02  
 8.06519433e-01 1.32831260e-02]  
 [9.69214289e-05 0.00000000e+00 2.43887924e-02 ... 4.29308164e-02  
 8.11392431e-01 1.61092778e-02]]

'''==== Multi-Layer Perception architecture is used for prediction of house price ===='''  
  
# shuffle and split data into train (~80%) and test (~20%)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(normalized\_feature, target.values,   
 test\_size=0.2, random\_state=42)   
print('training data shape: ',X\_train.shape)  
print('testing data shape: ',X\_test.shape)

training data shape: (404, 13)  
testing data shape: (102, 13)

'''===== Build MLP Network ====='''  
  
#get number of columns in training data  
n\_cols = X\_train.shape[1]  
  
# builds model  
model = keras.Sequential()  
  
model.add(keras.layers.Dense(150, activation=tf.nn.relu,   
 input\_shape=(n\_cols,)))  
model.add(keras.layers.Dense(150, activation=tf.nn.relu))  
model.add(keras.layers.Dense(150, activation=tf.nn.relu))  
model.add(keras.layers.Dense(150, activation=tf.nn.relu))  
model.add(keras.layers.Dense(150, activation=tf.nn.relu))  
model.add(keras.layers.Dense(1))  
  
#compile model  
model.compile(loss='mse', optimizer='adam', metrics=['mae']) # use metric as mean absolute error  
  
#inspect the model  
model.summary()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Layer (type) Output Shape Param #   
=================================================================  
dense\_316 (Dense) (None, 150) 2100   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_317 (Dense) (None, 150) 22650   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_318 (Dense) (None, 150) 22650   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_319 (Dense) (None, 150) 22650   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_320 (Dense) (None, 150) 22650   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_321 (Dense) (None, 1) 151   
=================================================================  
Total params: 92,851  
Trainable params: 92,851  
Non-trainable params: 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#train model and perform validation test  
  
early\_stop = EarlyStopping(monitor='val\_loss', patience=15) # stops training when it doesn't show improvemnet.  
  
history = model.fit(X\_train, y\_train, epochs=300,   
 validation\_split=0.2, verbose=1, callbacks=[early\_stop])

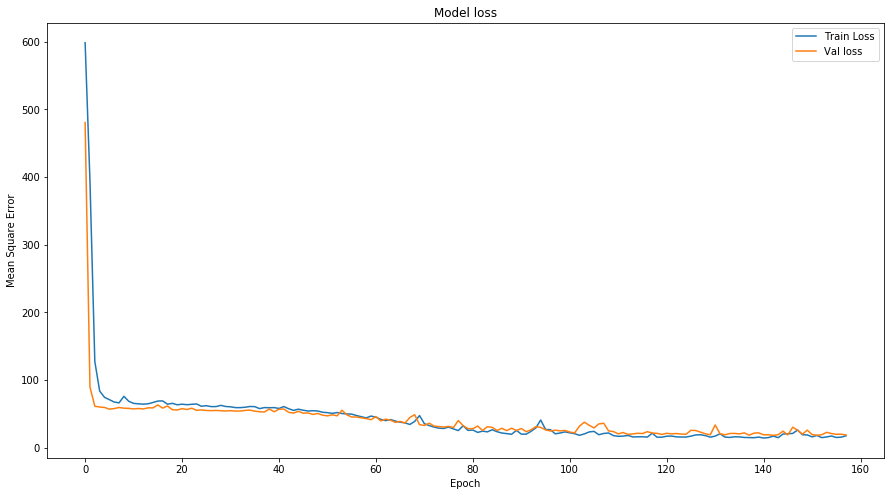
Train on 323 samples, validate on 81 samples  
Epoch 1/300  
323/323 [==============================] - 8s 23ms/step - loss: 598.3431 - mean\_absolute\_error: 22.5095 - val\_loss: 480.4472 - val\_mean\_absolute\_error: 20.2859  
Epoch 2/300  
323/323 [==============================] - 0s 464us/step - loss: 393.8183 - mean\_absolute\_error: 17.2987 - val\_loss: 89.6488 - val\_mean\_absolute\_error: 6.4877  
Epoch 3/300  
323/323 [==============================] - 0s 473us/step - loss: 127.4023 - mean\_absolute\_error: 9.3184 - val\_loss: 61.3814 - val\_mean\_absolute\_error: 5.6134  
Epoch 4/300  
323/323 [==============================] - 0s 519us/step - loss: 83.7834 - mean\_absolute\_error: 6.2351 - val\_loss: 60.1824 - val\_mean\_absolute\_error: 5.0162  
Epoch 5/300  
323/323 [==============================] - 0s 459us/step - loss: 74.5199 - mean\_absolute\_error: 6.2339 - val\_loss: 59.5460 - val\_mean\_absolute\_error: 5.8249  
Epoch 6/300  
323/323 [==============================] - 0s 465us/step - loss: 71.0907 - mean\_absolute\_error: 6.0731 - val\_loss: 57.0163 - val\_mean\_absolute\_error: 5.3314  
Epoch 7/300  
323/323 [==============================] - 0s 492us/step - loss: 67.5342 - mean\_absolute\_error: 6.1185 - val\_loss: 57.8725 - val\_mean\_absolute\_error: 5.5005  
Epoch 8/300  
323/323 [==============================] - 0s 404us/step - loss: 66.3194 - mean\_absolute\_error: 5.8955 - val\_loss: 59.5197 - val\_mean\_absolute\_error: 5.6858  
Epoch 9/300  
323/323 [==============================] - 0s 470us/step - loss: 75.9696 - mean\_absolute\_error: 6.7980 - val\_loss: 58.4470 - val\_mean\_absolute\_error: 4.9144  
Epoch 10/300  
323/323 [==============================] - 0s 516us/step - loss: 68.6456 - mean\_absolute\_error: 5.6470 - val\_loss: 58.1660 - val\_mean\_absolute\_error: 5.5189  
Epoch 11/300  
323/323 [==============================] - 0s 402us/step - loss: 65.5770 - mean\_absolute\_error: 5.8818 - val\_loss: 57.3787 - val\_mean\_absolute\_error: 5.3638  
Epoch 12/300  
323/323 [==============================] - 0s 477us/step - loss: 64.8818 - mean\_absolute\_error: 5.9557 - val\_loss: 57.9682 - val\_mean\_absolute\_error: 5.4483  
Epoch 13/300  
323/323 [==============================] - 0s 510us/step - loss: 64.4099 - mean\_absolute\_error: 6.0238 - val\_loss: 57.3117 - val\_mean\_absolute\_error: 5.1348  
Epoch 14/300  
323/323 [==============================] - 0s 537us/step - loss: 64.9248 - mean\_absolute\_error: 5.6089 - val\_loss: 58.9645 - val\_mean\_absolute\_error: 5.5976  
Epoch 15/300  
323/323 [==============================] - 0s 535us/step - loss: 66.8694 - mean\_absolute\_error: 6.3495 - val\_loss: 58.8827 - val\_mean\_absolute\_error: 4.9724  
Epoch 16/300  
323/323 [==============================] - 0s 541us/step - loss: 68.9863 - mean\_absolute\_error: 5.6652 - val\_loss: 63.1727 - val\_mean\_absolute\_error: 5.9705  
Epoch 17/300  
323/323 [==============================] - 0s 509us/step - loss: 69.1606 - mean\_absolute\_error: 6.5297 - val\_loss: 58.5465 - val\_mean\_absolute\_error: 4.8865  
Epoch 18/300  
323/323 [==============================] - 0s 528us/step - loss: 64.3571 - mean\_absolute\_error: 5.5473 - val\_loss: 61.6786 - val\_mean\_absolute\_error: 5.8994  
Epoch 19/300  
323/323 [==============================] - 0s 501us/step - loss: 65.6024 - mean\_absolute\_error: 6.1408 - val\_loss: 56.1786 - val\_mean\_absolute\_error: 4.9637  
Epoch 20/300  
323/323 [==============================] - 0s 530us/step - loss: 63.4776 - mean\_absolute\_error: 5.8097 - val\_loss: 55.8043 - val\_mean\_absolute\_error: 5.1812  
Epoch 21/300  
323/323 [==============================] - 0s 505us/step - loss: 64.3481 - mean\_absolute\_error: 5.5093 - val\_loss: 57.7443 - val\_mean\_absolute\_error: 5.5499  
Epoch 22/300  
323/323 [==============================] - 0s 549us/step - loss: 63.6045 - mean\_absolute\_error: 6.0530 - val\_loss: 56.5141 - val\_mean\_absolute\_error: 4.8404  
Epoch 23/300  
323/323 [==============================] - 0s 529us/step - loss: 64.1138 - mean\_absolute\_error: 5.5916 - val\_loss: 58.4078 - val\_mean\_absolute\_error: 5.6409  
Epoch 24/300  
323/323 [==============================] - 0s 534us/step - loss: 64.7211 - mean\_absolute\_error: 5.8299 - val\_loss: 55.2688 - val\_mean\_absolute\_error: 5.2198  
Epoch 25/300  
323/323 [==============================] - 0s 526us/step - loss: 61.2827 - mean\_absolute\_error: 5.6277 - val\_loss: 55.8777 - val\_mean\_absolute\_error: 5.3358  
Epoch 26/300  
323/323 [==============================] - 0s 412us/step - loss: 62.1122 - mean\_absolute\_error: 5.7419 - val\_loss: 55.0769 - val\_mean\_absolute\_error: 5.1009  
Epoch 27/300  
323/323 [==============================] - 0s 409us/step - loss: 60.7822 - mean\_absolute\_error: 5.6032 - val\_loss: 54.8274 - val\_mean\_absolute\_error: 5.0294  
Epoch 28/300  
323/323 [==============================] - 0s 467us/step - loss: 60.8289 - mean\_absolute\_error: 5.5587 - val\_loss: 55.0846 - val\_mean\_absolute\_error: 5.2305  
Epoch 29/300  
323/323 [==============================] - 0s 533us/step - loss: 62.6092 - mean\_absolute\_error: 5.9047 - val\_loss: 54.7566 - val\_mean\_absolute\_error: 4.9559  
Epoch 30/300  
323/323 [==============================] - 0s 594us/step - loss: 60.9398 - mean\_absolute\_error: 5.6884 - val\_loss: 54.2935 - val\_mean\_absolute\_error: 4.9849  
Epoch 31/300  
323/323 [==============================] - 0s 513us/step - loss: 60.4961 - mean\_absolute\_error: 5.4714 - val\_loss: 54.7965 - val\_mean\_absolute\_error: 5.1895  
Epoch 32/300  
323/323 [==============================] - 0s 459us/step - loss: 59.2971 - mean\_absolute\_error: 5.6351 - val\_loss: 54.1272 - val\_mean\_absolute\_error: 5.0727  
Epoch 33/300  
323/323 [==============================] - 0s 482us/step - loss: 59.2567 - mean\_absolute\_error: 5.6483 - val\_loss: 54.2306 - val\_mean\_absolute\_error: 5.1076  
Epoch 34/300  
323/323 [==============================] - 0s 523us/step - loss: 59.8351 - mean\_absolute\_error: 5.5488 - val\_loss: 55.1470 - val\_mean\_absolute\_error: 5.2940  
Epoch 35/300  
323/323 [==============================] - 0s 512us/step - loss: 60.9429 - mean\_absolute\_error: 5.4472 - val\_loss: 55.6660 - val\_mean\_absolute\_error: 5.3621  
Epoch 36/300  
323/323 [==============================] - 0s 510us/step - loss: 60.7257 - mean\_absolute\_error: 5.4231 - val\_loss: 54.0484 - val\_mean\_absolute\_error: 5.2447  
Epoch 37/300  
323/323 [==============================] - 0s 481us/step - loss: 57.7990 - mean\_absolute\_error: 5.4212 - val\_loss: 53.2115 - val\_mean\_absolute\_error: 5.2183  
Epoch 38/300  
323/323 [==============================] - 0s 496us/step - loss: 59.3488 - mean\_absolute\_error: 5.8150 - val\_loss: 52.8177 - val\_mean\_absolute\_error: 4.5392  
Epoch 39/300  
323/323 [==============================] - 0s 806us/step - loss: 59.0694 - mean\_absolute\_error: 5.2812 - val\_loss: 57.1264 - val\_mean\_absolute\_error: 5.6719  
Epoch 40/300  
323/323 [==============================] - 0s 608us/step - loss: 59.3011 - mean\_absolute\_error: 5.6111 - val\_loss: 53.2539 - val\_mean\_absolute\_error: 5.2423  
Epoch 41/300  
323/323 [==============================] - 0s 582us/step - loss: 58.0436 - mean\_absolute\_error: 5.8162 - val\_loss: 56.8930 - val\_mean\_absolute\_error: 4.6892  
Epoch 42/300  
323/323 [==============================] - 0s 597us/step - loss: 60.8370 - mean\_absolute\_error: 5.2625 - val\_loss: 57.2164 - val\_mean\_absolute\_error: 5.5725  
Epoch 43/300  
323/323 [==============================] - 0s 709us/step - loss: 57.6354 - mean\_absolute\_error: 5.2800 - val\_loss: 52.3213 - val\_mean\_absolute\_error: 5.0031  
Epoch 44/300  
323/323 [==============================] - 0s 632us/step - loss: 55.1902 - mean\_absolute\_error: 5.4302 - val\_loss: 51.1816 - val\_mean\_absolute\_error: 4.8584  
Epoch 45/300  
323/323 [==============================] - 0s 518us/step - loss: 56.9757 - mean\_absolute\_error: 5.1223 - val\_loss: 53.8380 - val\_mean\_absolute\_error: 5.3596  
Epoch 46/300  
323/323 [==============================] - 0s 554us/step - loss: 55.4245 - mean\_absolute\_error: 5.2700 - val\_loss: 51.0624 - val\_mean\_absolute\_error: 5.0928  
Epoch 47/300  
323/323 [==============================] - 0s 624us/step - loss: 54.2591 - mean\_absolute\_error: 5.3264 - val\_loss: 51.4221 - val\_mean\_absolute\_error: 4.6201  
Epoch 48/300  
323/323 [==============================] - 0s 532us/step - loss: 54.6548 - mean\_absolute\_error: 5.1859 - val\_loss: 49.2222 - val\_mean\_absolute\_error: 4.7596

Epoch 49/300  
323/323 [==============================] - 0s 651us/step - loss: 54.3352 - mean\_absolute\_error: 5.0011 - val\_loss: 50.6653 - val\_mean\_absolute\_error: 5.1419  
Epoch 50/300  
323/323 [==============================] - 0s 502us/step - loss: 52.4426 - mean\_absolute\_error: 5.1520 - val\_loss: 48.1862 - val\_mean\_absolute\_error: 4.7738  
Epoch 51/300  
323/323 [==============================] - 0s 627us/step - loss: 51.8490 - mean\_absolute\_error: 5.1046 - val\_loss: 47.0889 - val\_mean\_absolute\_error: 4.6211  
Epoch 52/300  
323/323 [==============================] - 0s 638us/step - loss: 50.7996 - mean\_absolute\_error: 5.1295 - val\_loss: 48.6481 - val\_mean\_absolute\_error: 4.4411  
Epoch 53/300  
323/323 [==============================] - 0s 620us/step - loss: 52.0058 - mean\_absolute\_error: 4.8466 - val\_loss: 47.1465 - val\_mean\_absolute\_error: 4.7506  
Epoch 54/300  
323/323 [==============================] - 0s 659us/step - loss: 50.5755 - mean\_absolute\_error: 4.7621 - val\_loss: 55.4162 - val\_mean\_absolute\_error: 5.6921  
Epoch 55/300  
323/323 [==============================] - 0s 572us/step - loss: 49.9149 - mean\_absolute\_error: 5.1510 - val\_loss: 48.6064 - val\_mean\_absolute\_error: 4.3701  
Epoch 56/300  
323/323 [==============================] - 0s 484us/step - loss: 49.6636 - mean\_absolute\_error: 4.9829 - val\_loss: 45.4197 - val\_mean\_absolute\_error: 4.7533  
Epoch 57/300  
323/323 [==============================] - 0s 501us/step - loss: 47.5945 - mean\_absolute\_error: 4.6898 - val\_loss: 45.3740 - val\_mean\_absolute\_error: 4.9520  
Epoch 58/300  
323/323 [==============================] - 0s 430us/step - loss: 45.8820 - mean\_absolute\_error: 4.6753 - val\_loss: 44.1442 - val\_mean\_absolute\_error: 4.6299  
Epoch 59/300  
323/323 [==============================] - 0s 512us/step - loss: 44.0978 - mean\_absolute\_error: 4.6367 - val\_loss: 43.0202 - val\_mean\_absolute\_error: 4.2227  
Epoch 60/300  
323/323 [==============================] - 0s 414us/step - loss: 46.8128 - mean\_absolute\_error: 4.7540 - val\_loss: 41.4316 - val\_mean\_absolute\_error: 4.3759  
Epoch 61/300  
323/323 [==============================] - 0s 641us/step - loss: 45.1157 - mean\_absolute\_error: 4.5686 - val\_loss: 46.1175 - val\_mean\_absolute\_error: 5.0768  
Epoch 62/300  
323/323 [==============================] - 0s 648us/step - loss: 41.9801 - mean\_absolute\_error: 4.5836 - val\_loss: 39.6476 - val\_mean\_absolute\_error: 4.2429  
Epoch 63/300  
323/323 [==============================] - 0s 417us/step - loss: 40.0859 - mean\_absolute\_error: 4.6008 - val\_loss: 42.2541 - val\_mean\_absolute\_error: 3.9363  
Epoch 64/300  
323/323 [==============================] - 0s 436us/step - loss: 41.6618 - mean\_absolute\_error: 4.5889 - val\_loss: 40.7761 - val\_mean\_absolute\_error: 3.9340  
Epoch 65/300  
323/323 [==============================] - 0s 629us/step - loss: 39.4207 - mean\_absolute\_error: 4.4004 - val\_loss: 37.4899 - val\_mean\_absolute\_error: 3.7149  
Epoch 66/300  
323/323 [==============================] - 0s 656us/step - loss: 37.8069 - mean\_absolute\_error: 4.3327 - val\_loss: 38.8912 - val\_mean\_absolute\_error: 4.4403  
Epoch 67/300  
323/323 [==============================] - 0s 580us/step - loss: 36.6832 - mean\_absolute\_error: 4.2534 - val\_loss: 36.5916 - val\_mean\_absolute\_error: 4.0493  
Epoch 68/300  
323/323 [==============================] - 0s 680us/step - loss: 34.4309 - mean\_absolute\_error: 4.1865 - val\_loss: 44.7403 - val\_mean\_absolute\_error: 5.1168  
Epoch 69/300  
323/323 [==============================] - 0s 602us/step - loss: 38.8930 - mean\_absolute\_error: 4.6338 - val\_loss: 48.7869 - val\_mean\_absolute\_error: 5.4303  
Epoch 70/300  
323/323 [==============================] - 0s 523us/step - loss: 47.5561 - mean\_absolute\_error: 5.2456 - val\_loss: 34.0967 - val\_mean\_absolute\_error: 3.9820  
Epoch 71/300  
323/323 [==============================] - 0s 450us/step - loss: 35.5478 - mean\_absolute\_error: 4.2697 - val\_loss: 32.9527 - val\_mean\_absolute\_error: 3.8362  
Epoch 72/300  
323/323 [==============================] - 0s 487us/step - loss: 32.7296 - mean\_absolute\_error: 4.1877 - val\_loss: 36.2849 - val\_mean\_absolute\_error: 4.2123  
Epoch 73/300  
323/323 [==============================] - 0s 510us/step - loss: 30.5179 - mean\_absolute\_error: 4.0652 - val\_loss: 32.0222 - val\_mean\_absolute\_error: 3.6397  
Epoch 74/300  
323/323 [==============================] - 0s 474us/step - loss: 28.9435 - mean\_absolute\_error: 3.9581 - val\_loss: 31.3334 - val\_mean\_absolute\_error: 3.6387  
Epoch 75/300  
323/323 [==============================] - 0s 475us/step - loss: 28.4876 - mean\_absolute\_error: 3.8458 - val\_loss: 30.6562 - val\_mean\_absolute\_error: 3.5259  
Epoch 76/300  
323/323 [==============================] - 0s 520us/step - loss: 30.5200 - mean\_absolute\_error: 4.0395 - val\_loss: 31.5359 - val\_mean\_absolute\_error: 3.7692  
Epoch 77/300  
323/323 [==============================] - 0s 537us/step - loss: 27.8002 - mean\_absolute\_error: 3.8257 - val\_loss: 29.9082 - val\_mean\_absolute\_error: 3.4365  
Epoch 78/300  
323/323 [==============================] - 0s 637us/step - loss: 25.4170 - mean\_absolute\_error: 3.6860 - val\_loss: 40.2089 - val\_mean\_absolute\_error: 4.8069  
Epoch 79/300  
323/323 [==============================] - 0s 486us/step - loss: 32.4650 - mean\_absolute\_error: 4.4093 - val\_loss: 32.5140 - val\_mean\_absolute\_error: 3.9136  
Epoch 80/300  
323/323 [==============================] - 0s 487us/step - loss: 25.6404 - mean\_absolute\_error: 3.7916 - val\_loss: 28.2536 - val\_mean\_absolute\_error: 3.4131  
Epoch 81/300  
323/323 [==============================] - 0s 519us/step - loss: 26.1870 - mean\_absolute\_error: 3.8025 - val\_loss: 28.0335 - val\_mean\_absolute\_error: 3.3865  
Epoch 82/300  
323/323 [==============================] - 0s 620us/step - loss: 22.6742 - mean\_absolute\_error: 3.4775 - val\_loss: 32.0755 - val\_mean\_absolute\_error: 3.9444  
Epoch 83/300  
323/323 [==============================] - 0s 485us/step - loss: 24.5284 - mean\_absolute\_error: 3.8054 - val\_loss: 25.5294 - val\_mean\_absolute\_error: 3.2163  
Epoch 84/300  
323/323 [==============================] - 0s 473us/step - loss: 23.4799 - mean\_absolute\_error: 3.6172 - val\_loss: 31.0337 - val\_mean\_absolute\_error: 3.5960  
Epoch 85/300  
323/323 [==============================] - 0s 485us/step - loss: 26.6011 - mean\_absolute\_error: 3.9441 - val\_loss: 30.1655 - val\_mean\_absolute\_error: 3.4621  
Epoch 86/300  
323/323 [==============================] - 0s 615us/step - loss: 23.6951 - mean\_absolute\_error: 3.6427 - val\_loss: 25.5803 - val\_mean\_absolute\_error: 3.2219  
Epoch 87/300  
323/323 [==============================] - 0s 602us/step - loss: 21.7389 - mean\_absolute\_error: 3.5032 - val\_loss: 28.7504 - val\_mean\_absolute\_error: 3.5646  
Epoch 88/300  
323/323 [==============================] - 0s 669us/step - loss: 20.8073 - mean\_absolute\_error: 3.4227 - val\_loss: 25.3333 - val\_mean\_absolute\_error: 3.1902  
Epoch 89/300  
323/323 [==============================] - 0s 663us/step - loss: 19.9579 - mean\_absolute\_error: 3.4411 - val\_loss: 28.8949 - val\_mean\_absolute\_error: 3.3788  
Epoch 90/300  
323/323 [==============================] - 0s 507us/step - loss: 25.6551 - mean\_absolute\_error: 3.8374 - val\_loss: 25.4060 - val\_mean\_absolute\_error: 3.2131  
Epoch 91/300  
323/323 [==============================] - 0s 504us/step - loss: 20.1363 - mean\_absolute\_error: 3.3741 - val\_loss: 28.4312 - val\_mean\_absolute\_error: 3.4919  
Epoch 92/300  
323/323 [==============================] - 0s 479us/step - loss: 20.0585 - mean\_absolute\_error: 3.4301 - val\_loss: 23.7159 - val\_mean\_absolute\_error: 3.1539  
Epoch 93/300  
323/323 [==============================] - 0s 478us/step - loss: 24.1867 - mean\_absolute\_error: 3.6917 - val\_loss: 26.5504 - val\_mean\_absolute\_error: 3.2181  
Epoch 94/300  
323/323 [==============================] - 0s 485us/step - loss: 29.3716 - mean\_absolute\_error: 4.1012 - val\_loss: 31.1112 - val\_mean\_absolute\_error: 3.5147  
Epoch 95/300  
323/323 [==============================] - 0s 517us/step - loss: 40.9959 - mean\_absolute\_error: 4.7141 - val\_loss: 30.0867 - val\_mean\_absolute\_error: 3.6885  
Epoch 96/300  
323/323 [==============================] - 0s 623us/step - loss: 27.0034 - mean\_absolute\_error: 3.8716 - val\_loss: 26.4328 - val\_mean\_absolute\_error: 3.3112  
Epoch 97/300  
323/323 [==============================] - 0s 513us/step - loss: 26.6752 - mean\_absolute\_error: 3.8090 - val\_loss: 24.7247 - val\_mean\_absolute\_error: 3.1129  
Epoch 98/300  
323/323 [==============================] - 0s 548us/step - loss: 20.5814 - mean\_absolute\_error: 3.4011 - val\_loss: 25.9436 - val\_mean\_absolute\_error: 3.2262  
Epoch 99/300  
323/323 [==============================] - 0s 523us/step - loss: 21.7839 - mean\_absolute\_error: 3.4498 - val\_loss: 24.9994 - val\_mean\_absolute\_error: 3.2548  
Epoch 100/300  
323/323 [==============================] - 0s 478us/step - loss: 23.4307 - mean\_absolute\_error: 3.6952 - val\_loss: 25.5364 - val\_mean\_absolute\_error: 3.1541  
Epoch 101/300  
323/323 [==============================] - 0s 489us/step - loss: 22.1701 - mean\_absolute\_error: 3.5619 - val\_loss: 23.2621 - val\_mean\_absolute\_error: 3.0950  
Epoch 102/300  
323/323 [==============================] - 0s 441us/step - loss: 20.7503 - mean\_absolute\_error: 3.3744 - val\_loss: 21.8587 - val\_mean\_absolute\_error: 2.9654  
Epoch 103/300  
323/323 [==============================] - 0s 430us/step - loss: 18.4008 - mean\_absolute\_error: 3.1975 - val\_loss: 31.9535 - val\_mean\_absolute\_error: 4.1444  
Epoch 104/300  
323/323 [==============================] - 0s 433us/step - loss: 20.3999 - mean\_absolute\_error: 3.2675 - val\_loss: 37.7497 - val\_mean\_absolute\_error: 4.7521  
Epoch 105/300  
323/323 [==============================] - 0s 409us/step - loss: 23.3596 - mean\_absolute\_error: 3.5633 - val\_loss: 33.2998 - val\_mean\_absolute\_error: 4.4326  
Epoch 106/300  
323/323 [==============================] - 0s 478us/step - loss: 24.1741 - mean\_absolute\_error: 3.5960 - val\_loss: 29.2784 - val\_mean\_absolute\_error: 3.9552  
Epoch 107/300  
323/323 [==============================] - 0s 551us/step - loss: 19.2655 - mean\_absolute\_error: 3.2783 - val\_loss: 35.1975 - val\_mean\_absolute\_error: 4.4366  
Epoch 108/300  
323/323 [==============================] - 0s 549us/step - loss: 21.1208 - mean\_absolute\_error: 3.4219 - val\_loss: 36.2219 - val\_mean\_absolute\_error: 4.5074  
Epoch 109/300  
323/323 [==============================] - 0s 666us/step - loss: 21.8153 - mean\_absolute\_error: 3.5012 - val\_loss: 24.9038 - val\_mean\_absolute\_error: 3.3882  
Epoch 110/300  
323/323 [==============================] - 0s 551us/step - loss: 17.9336 - mean\_absolute\_error: 3.1375 - val\_loss: 23.9076 - val\_mean\_absolute\_error: 3.2501  
Epoch 111/300  
323/323 [==============================] - 0s 478us/step - loss: 16.8927 - mean\_absolute\_error: 3.0895 - val\_loss: 20.5409 - val\_mean\_absolute\_error: 2.9445  
Epoch 112/300  
323/323 [==============================] - 0s 470us/step - loss: 17.1977 - mean\_absolute\_error: 3.1158 - val\_loss: 22.4566 - val\_mean\_absolute\_error: 3.0455  
Epoch 113/300  
323/323 [==============================] - 0s 463us/step - loss: 18.1860 - mean\_absolute\_error: 3.2077 - val\_loss: 19.7414 - val\_mean\_absolute\_error: 2.8788  
Epoch 114/300  
323/323 [==============================] - 0s 431us/step - loss: 16.0083 - mean\_absolute\_error: 3.0266 - val\_loss: 20.4625 - val\_mean\_absolute\_error: 2.9498  
Epoch 115/300  
323/323 [==============================] - 0s 515us/step - loss: 16.1414 - mean\_absolute\_error: 3.0126 - val\_loss: 21.3845 - val\_mean\_absolute\_error: 3.0964  
Epoch 116/300  
323/323 [==============================] - 0s 481us/step - loss: 16.2666 - mean\_absolute\_error: 3.0105 - val\_loss: 20.9788 - val\_mean\_absolute\_error: 3.1022  
Epoch 117/300  
323/323 [==============================] - 0s 468us/step - loss: 15.7143 - mean\_absolute\_error: 2.9975 - val\_loss: 23.7193 - val\_mean\_absolute\_error: 3.1168  
Epoch 118/300  
323/323 [==============================] - 0s 602us/step - loss: 21.4500 - mean\_absolute\_error: 3.4063 - val\_loss: 21.7557 - val\_mean\_absolute\_error: 2.9644  
Epoch 119/300  
323/323 [==============================] - 0s 654us/step - loss: 15.6686 - mean\_absolute\_error: 2.9927 - val\_loss: 21.2763 - val\_mean\_absolute\_error: 3.0163  
Epoch 120/300  
323/323 [==============================] - 0s 450us/step - loss: 15.5738 - mean\_absolute\_error: 2.9837 - val\_loss: 19.6506 - val\_mean\_absolute\_error: 2.8714  
Epoch 121/300  
323/323 [==============================] - 0s 541us/step - loss: 17.0687 - mean\_absolute\_error: 3.1262 - val\_loss: 21.3515 - val\_mean\_absolute\_error: 2.9850  
Epoch 122/300  
323/323 [==============================] - 0s 608us/step - loss: 17.2720 - mean\_absolute\_error: 3.1602 - val\_loss: 20.5530 - val\_mean\_absolute\_error: 2.9779  
Epoch 123/300  
323/323 [==============================] - 0s 489us/step - loss: 16.0687 - mean\_absolute\_error: 2.9957 - val\_loss: 20.9882 - val\_mean\_absolute\_error: 2.9952  
Epoch 124/300  
323/323 [==============================] - 0s 474us/step - loss: 15.8363 - mean\_absolute\_error: 3.0057 - val\_loss: 20.2334 - val\_mean\_absolute\_error: 2.8743  
Epoch 125/300  
323/323 [==============================] - 0s 520us/step - loss: 15.8632 - mean\_absolute\_error: 2.9366 - val\_loss: 19.9147 - val\_mean\_absolute\_error: 2.8539  
Epoch 126/300  
323/323 [==============================] - 0s 595us/step - loss: 17.1518 - mean\_absolute\_error: 3.0475 - val\_loss: 25.7943 - val\_mean\_absolute\_error: 3.3107  
Epoch 127/300  
323/323 [==============================] - 0s 486us/step - loss: 18.9627 - mean\_absolute\_error: 3.2641 - val\_loss: 25.4559 - val\_mean\_absolute\_error: 3.2724  
Epoch 128/300  
323/323 [==============================] - 0s 548us/step - loss: 19.1722 - mean\_absolute\_error: 3.3064 - val\_loss: 22.9752 - val\_mean\_absolute\_error: 3.0183  
Epoch 129/300  
323/323 [==============================] - 0s 473us/step - loss: 17.9011 - mean\_absolute\_error: 3.2168 - val\_loss: 20.5686 - val\_mean\_absolute\_error: 2.9258  
Epoch 130/300  
323/323 [==============================] - 0s 468us/step - loss: 15.6785 - mean\_absolute\_error: 2.9469 - val\_loss: 19.2439 - val\_mean\_absolute\_error: 2.8194  
Epoch 131/300  
323/323 [==============================] - 0s 497us/step - loss: 17.1847 - mean\_absolute\_error: 3.1627 - val\_loss: 33.5791 - val\_mean\_absolute\_error: 3.8147  
Epoch 132/300  
323/323 [==============================] - 0s 461us/step - loss: 20.6983 - mean\_absolute\_error: 3.4025 - val\_loss: 20.7409 - val\_mean\_absolute\_error: 2.8950  
Epoch 133/300  
323/323 [==============================] - 0s 446us/step - loss: 15.8012 - mean\_absolute\_error: 3.0218 - val\_loss: 19.0712 - val\_mean\_absolute\_error: 2.8448  
Epoch 134/300  
323/323 [==============================] - 0s 451us/step - loss: 15.2290 - mean\_absolute\_error: 2.9714 - val\_loss: 21.0433 - val\_mean\_absolute\_error: 2.9466  
Epoch 135/300  
323/323 [==============================] - 0s 412us/step - loss: 16.2159 - mean\_absolute\_error: 2.9996 - val\_loss: 21.1860 - val\_mean\_absolute\_error: 3.0643  
Epoch 136/300  
323/323 [==============================] - 0s 431us/step - loss: 16.0152 - mean\_absolute\_error: 3.0304 - val\_loss: 20.4564 - val\_mean\_absolute\_error: 2.9678  
Epoch 137/300  
323/323 [==============================] - 0s 459us/step - loss: 15.2107 - mean\_absolute\_error: 2.9155 - val\_loss: 21.9898 - val\_mean\_absolute\_error: 3.1315  
Epoch 138/300  
323/323 [==============================] - 0s 456us/step - loss: 14.9844 - mean\_absolute\_error: 2.8852 - val\_loss: 18.5978 - val\_mean\_absolute\_error: 2.8226  
Epoch 139/300  
323/323 [==============================] - 0s 462us/step - loss: 14.8513 - mean\_absolute\_error: 2.8873 - val\_loss: 21.5651 - val\_mean\_absolute\_error: 3.0575  
Epoch 140/300  
323/323 [==============================] - 0s 433us/step - loss: 15.7047 - mean\_absolute\_error: 2.9737 - val\_loss: 21.9738 - val\_mean\_absolute\_error: 3.0669  
Epoch 141/300  
323/323 [==============================] - 0s 414us/step - loss: 14.3573 - mean\_absolute\_error: 2.8321 - val\_loss: 19.0558 - val\_mean\_absolute\_error: 2.8030  
Epoch 142/300  
323/323 [==============================] - 0s 619us/step - loss: 15.0365 - mean\_absolute\_error: 2.9397 - val\_loss: 19.2368 - val\_mean\_absolute\_error: 2.8009  
Epoch 143/300  
323/323 [==============================] - 0s 576us/step - loss: 17.2414 - mean\_absolute\_error: 3.0212 - val\_loss: 18.2791 - val\_mean\_absolute\_error: 2.7824  
Epoch 144/300  
323/323 [==============================] - 0s 482us/step - loss: 14.8968 - mean\_absolute\_error: 2.9055 - val\_loss: 19.3692 - val\_mean\_absolute\_error: 2.8660  
Epoch 145/300

323/323 [==============================] - 0s 622us/step - loss: 20.2387 - mean\_absolute\_error: 3.2359 - val\_loss: 24.4703 - val\_mean\_absolute\_error: 3.3075  
Epoch 146/300  
323/323 [==============================] - 0s 590us/step - loss: 20.6854 - mean\_absolute\_error: 3.4236 - val\_loss: 18.9732 - val\_mean\_absolute\_error: 2.8340  
Epoch 147/300  
323/323 [==============================] - 0s 447us/step - loss: 21.1540 - mean\_absolute\_error: 3.3399 - val\_loss: 30.3674 - val\_mean\_absolute\_error: 3.7411  
Epoch 148/300  
323/323 [==============================] - 0s 585us/step - loss: 26.1566 - mean\_absolute\_error: 3.7330 - val\_loss: 26.0125 - val\_mean\_absolute\_error: 3.2010  
Epoch 149/300  
323/323 [==============================] - 0s 579us/step - loss: 19.1924 - mean\_absolute\_error: 3.1922 - val\_loss: 20.0197 - val\_mean\_absolute\_error: 3.0454  
Epoch 150/300  
323/323 [==============================] - 0s 680us/step - loss: 18.9605 - mean\_absolute\_error: 3.1551 - val\_loss: 26.0110 - val\_mean\_absolute\_error: 3.6248  
Epoch 151/300  
323/323 [==============================] - 0s 521us/step - loss: 16.1073 - mean\_absolute\_error: 2.9584 - val\_loss: 19.5288 - val\_mean\_absolute\_error: 2.9820  
Epoch 152/300  
323/323 [==============================] - 0s 690us/step - loss: 18.1684 - mean\_absolute\_error: 3.1132 - val\_loss: 18.4515 - val\_mean\_absolute\_error: 2.7519  
Epoch 153/300  
323/323 [==============================] - 0s 442us/step - loss: 14.9584 - mean\_absolute\_error: 2.9894 - val\_loss: 19.2568 - val\_mean\_absolute\_error: 2.8592  
Epoch 154/300  
323/323 [==============================] - 0s 547us/step - loss: 15.9624 - mean\_absolute\_error: 2.9429 - val\_loss: 22.7639 - val\_mean\_absolute\_error: 3.0654  
Epoch 155/300  
323/323 [==============================] - 0s 601us/step - loss: 17.3198 - mean\_absolute\_error: 3.0439 - val\_loss: 20.9926 - val\_mean\_absolute\_error: 2.9255  
Epoch 156/300  
323/323 [==============================] - 0s 576us/step - loss: 15.1819 - mean\_absolute\_error: 2.9624 - val\_loss: 19.9080 - val\_mean\_absolute\_error: 2.8610  
Epoch 157/300  
323/323 [==============================] - 0s 646us/step - loss: 15.6287 - mean\_absolute\_error: 2.8931 - val\_loss: 20.0563 - val\_mean\_absolute\_error: 2.9420  
Epoch 158/300  
323/323 [==============================] - 0s 552us/step - loss: 17.7185 - mean\_absolute\_error: 3.1272 - val\_loss: 18.8596 - val\_mean\_absolute\_error: 2.8000

# show the graph of model loss in trainig and validation   
  
plt.figure(figsize=(15,8))  
plt.xlabel('Epoch')  
plt.ylabel('Mean Square Error')  
plt.plot(history.epoch, history.history['loss'],  
 label='Train Loss')  
plt.plot(history.epoch, history.history['val\_loss'],  
 label = 'Val loss')  
plt.title('Model loss')  
plt.legend()

<matplotlib.legend.Legend at 0x1a62916e80>



#check the model performace in test dataset  
score = model.evaluate(X\_test, y\_test, verbose=1)  
print('loss value: ', score[0])  
print('Mean absolute error: ', score[1])

102/102 [==============================] - 0s 190us/step  
loss value: 16.24074539483762  
Mean absolute error: 2.63373737709195

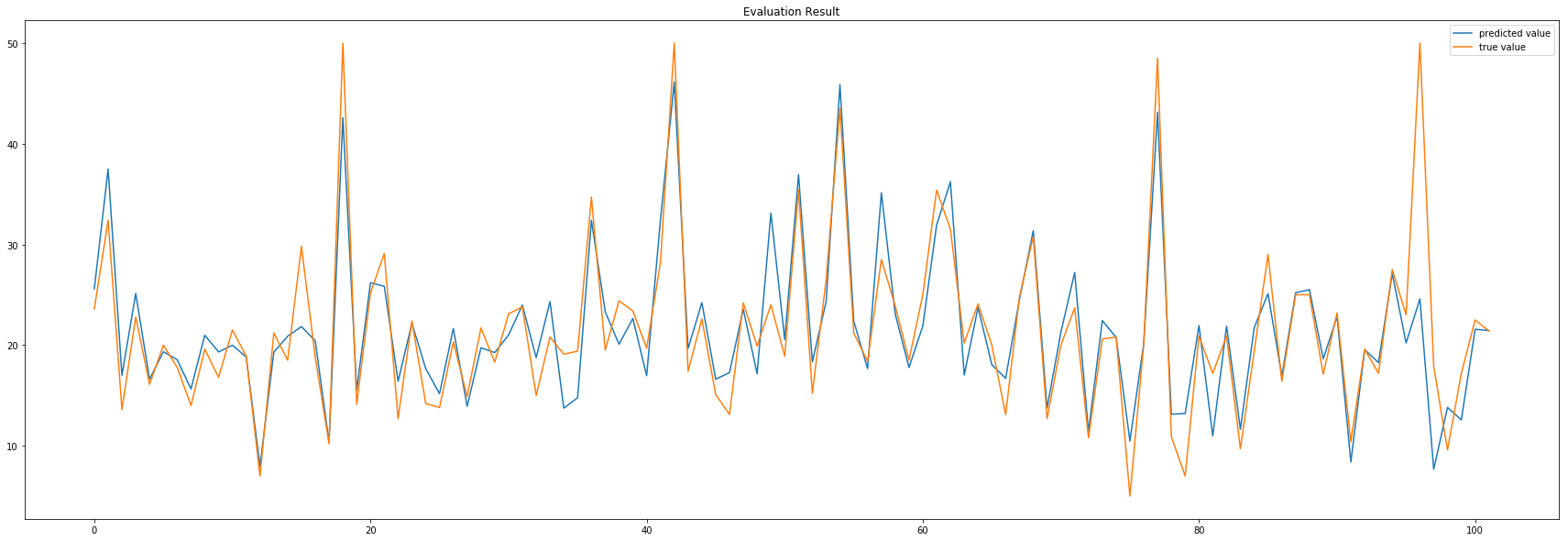
'''=== predict the house price ==='''  
  
# predict house price using the test data   
test\_predictions = model.predict(X\_test).flatten()  
print(test\_predictions)

[25.590422 37.48625 17.000551 25.128817 16.607439 19.340845  
 18.559471 15.650881 20.977253 19.326307 19.98703 18.825693  
 7.963045 19.30431 20.878283 21.842907 20.425995 10.342217  
 42.59475 15.513068 26.198084 25.837694 16.401852 22.17246  
 17.672476 15.174794 21.648037 13.920175 19.735518 19.255856  
 21.002058 23.976297 18.756481 24.323591 13.738233 14.774037  
 32.396496 23.303461 20.088331 22.647577 16.961424 32.42199  
 46.141693 19.655474 24.222631 16.615934 17.266104 23.568766  
 17.146776 33.11348 20.551208 36.927723 18.336277 24.31582  
 45.866516 22.391039 17.662512 35.135452 23.100254 17.779528  
 21.89772 31.920958 36.220825 17.020712 23.7465 18.046824  
 16.703196 24.537752 31.364645 13.754661 21.278679 27.217371  
 11.388164 22.438303 20.772512 10.475509 20.167238 43.128677  
 13.133572 13.203116 21.949951 10.975419 21.89226 11.641559  
 21.72124 25.111378 16.917246 25.202017 25.50582 18.634295  
 22.83381 8.391959 19.552364 18.26517 27.113409 20.20947  
 24.590448 7.6676183 13.81203 12.573187 21.571423 21.43662 ]

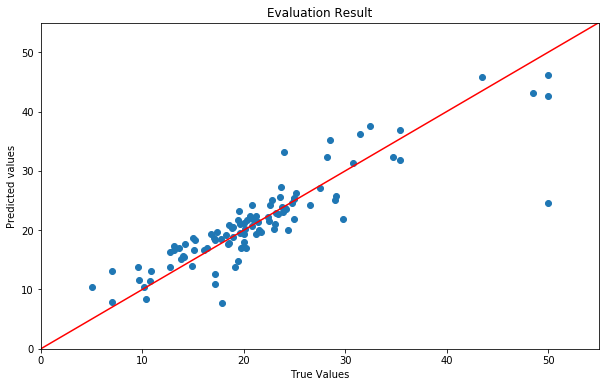
# show the true value and predicted value in dataframe  
true\_predicted = pd.DataFrame(list(zip(y\_test, test\_predictions)),   
 columns=['True Value','Predicted Value'])  
true\_predicted.head(10)

True Value Predicted Value  
0 23.6 25.590422  
1 32.4 37.486252  
2 13.6 17.000551  
3 22.8 25.128817  
4 16.1 16.607439  
5 20.0 19.340845  
6 17.8 18.559471  
7 14.0 15.650881  
8 19.6 20.977253  
9 16.8 19.326307

#visiulize the true value with predicted value (using line graph)  
x = test\_predictions  
y = y\_test  
plt.figure(figsize=(30,10))  
plt.plot(x, label='predicted value')  
plt.plot(y, label='true value')  
plt.title('Evaluation Result')  
plt.legend()  
plt.show()



'''=== Visualize the model evaluation skill ==='''  
  
  
# visualize the prediction uisng diagonal line  
y = test\_predictions #y-axis  
x = y\_test #x-axis  
fig, ax = plt.subplots(figsize=(10,6)) # create figure  
ax.scatter(x,y) #scatter plots for x,y  
ax.set(xlim=(0,55), ylim=(0, 55)) #set limit  
ax.plot(ax.get\_xlim(), ax.get\_ylim(), color ='red') # draw 45 degree diagonal in figure  
plt.xlabel('True Values')  
plt.ylabel('Predicted values')  
plt.title('Evaluation Result')  
plt.show()



======= Remove outliers from the data set ========

Tried to see the model perfomance after removing the outliers, the perfomance doesn't show much improvemnt. This might be less data size after remoing the outliers.

#calcualtes quartiles and interquaterlies  
Q1 = house\_df.quantile(0.25)  
Q3 = house\_df.quantile(0.75)  
IQR = Q3 - Q1

house\_data = house\_df.iloc[:,0:14]  
house\_data.shape

(506, 14)

#data after outliers remvoing  
clean\_data = house\_df[~((house\_data < (Q1 - 1.5 \* IQR)) |(house\_data > (Q3 + 1.5 \* IQR))).any(axis=1)]  
clean\_data.shape

(268, 14)

feature = clean\_data.iloc[:,0:13] # training variables  
target = clean\_data.iloc[:,13] # target varible  
print(feature.head())  
print('\n',target.head())

CRIM ZN INDUS CHAS NOX RM AGE DIS RAD TAX \  
0 0.00632 18.0 2.31 0 0.538 6.575 65.2 4.0900 1 296.0   
1 0.02731 0.0 7.07 0 0.469 6.421 78.9 4.9671 2 242.0   
2 0.02729 0.0 7.07 0 0.469 7.185 61.1 4.9671 2 242.0   
3 0.03237 0.0 2.18 0 0.458 6.998 45.8 6.0622 3 222.0   
4 0.06905 0.0 2.18 0 0.458 7.147 54.2 6.0622 3 222.0   
  
 PTRATIO B LSTAT   
0 15.3 396.90 4.98   
1 17.8 396.90 9.14   
2 17.8 392.83 4.03   
3 18.7 394.63 2.94   
4 18.7 396.90 5.33   
  
 0 24.0  
1 21.6  
2 34.7  
3 33.4  
4 36.2  
Name: MEDV, dtype: float64

#feature normalization  
normalized\_feature = keras.utils.normalize(feature.values)  
print(normalized\_feature)

[[1.26388341e-05 3.59966795e-02 4.61957387e-03 ... 3.05971776e-02  
 7.93726783e-01 9.95908132e-03]  
 [5.78529889e-05 0.00000000e+00 1.49769546e-02 ... 3.77071843e-02  
 8.40785474e-01 1.93620036e-02]  
 [5.85729947e-05 0.00000000e+00 1.51744622e-02 ... 3.82044450e-02  
 8.43137761e-01 8.64965806e-03]  
 ...  
 [1.23765824e-04 0.00000000e+00 2.43009593e-02 ... 4.27762066e-02  
 8.08470305e-01 1.14884669e-02]  
 [2.24644719e-04 0.00000000e+00 2.44548909e-02 ... 4.30471676e-02  
 8.06519433e-01 1.32831260e-02]  
 [9.69214289e-05 0.00000000e+00 2.43887924e-02 ... 4.29308164e-02  
 8.11392431e-01 1.61092778e-02]]

# shuffle and split data into train (~80%) and test (~20%)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(normalized\_feature, target.values, test\_size=0.2, random\_state=42)   
print('training data shape: ',X\_train.shape)  
print('testing data shape: ',X\_test.shape)

training data shape: (214, 13)  
testing data shape: (54, 13)

#get number of columns in training data  
n\_cols = X\_train.shape[1]  
  
# builds model  
model = keras.Sequential()  
  
model.add(keras.layers.Dense(150, activation=tf.nn.relu,   
 input\_shape=(n\_cols,)))  
model.add(keras.layers.Dense(150, activation=tf.nn.relu))  
model.add(keras.layers.Dense(150, activation=tf.nn.relu))  
model.add(keras.layers.Dense(150, activation=tf.nn.relu))  
model.add(keras.layers.Dense(150, activation=tf.nn.relu))  
model.add(keras.layers.Dense(1))  
  
#compile model  
model.compile(loss='mse', optimizer='adam', metrics=['mae']) # use metric as mean absolute error  
  
#inspect the model  
model.summary()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Layer (type) Output Shape Param #   
=================================================================  
dense\_355 (Dense) (None, 150) 2100   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_356 (Dense) (None, 150) 22650   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_357 (Dense) (None, 150) 22650   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_358 (Dense) (None, 150) 22650   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_359 (Dense) (None, 150) 22650   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense\_360 (Dense) (None, 1) 151   
=================================================================  
Total params: 92,851  
Trainable params: 92,851  
Non-trainable params: 0  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#train model  
early\_stop = EarlyStopping(monitor='val\_loss', patience=20) # stops training when it doesn't show improvemnet.  
  
history = model.fit(X\_train, y\_train, epochs=300,   
 validation\_split=0.2, verbose=1, callbacks=[early\_stop])

Train on 171 samples, validate on 43 samples  
Epoch 1/300  
171/171 [==============================] - 10s 57ms/step - loss: 485.8513 - mean\_absolute\_error: 21.4565 - val\_loss: 434.8489 - val\_mean\_absolute\_error: 20.4979  
Epoch 2/300  
171/171 [==============================] - 0s 635us/step - loss: 457.4068 - mean\_absolute\_error: 20.7782 - val\_loss: 379.3888 - val\_mean\_absolute\_error: 19.0971  
Epoch 3/300  
171/171 [==============================] - 0s 596us/step - loss: 369.2564 - mean\_absolute\_error: 18.5282 - val\_loss: 229.3178 - val\_mean\_absolute\_error: 14.6483  
Epoch 4/300  
171/171 [==============================] - 0s 634us/step - loss: 168.7987 - mean\_absolute\_error: 11.6760 - val\_loss: 17.2939 - val\_mean\_absolute\_error: 3.1044  
Epoch 5/300  
171/171 [==============================] - 0s 636us/step - loss: 61.6155 - mean\_absolute\_error: 6.6113 - val\_loss: 71.8958 - val\_mean\_absolute\_error: 7.7419  
Epoch 6/300  
171/171 [==============================] - 0s 610us/step - loss: 47.8748 - mean\_absolute\_error: 5.4899 - val\_loss: 18.5151 - val\_mean\_absolute\_error: 3.2289  
Epoch 7/300  
171/171 [==============================] - 0s 560us/step - loss: 36.6663 - mean\_absolute\_error: 4.6790 - val\_loss: 21.0218 - val\_mean\_absolute\_error: 3.4761  
Epoch 8/300  
171/171 [==============================] - 0s 706us/step - loss: 29.4570 - mean\_absolute\_error: 4.1391 - val\_loss: 16.4801 - val\_mean\_absolute\_error: 3.3389  
Epoch 9/300  
171/171 [==============================] - 0s 791us/step - loss: 25.8531 - mean\_absolute\_error: 3.9217 - val\_loss: 19.4626 - val\_mean\_absolute\_error: 3.6593  
Epoch 10/300  
171/171 [==============================] - 0s 797us/step - loss: 24.1366 - mean\_absolute\_error: 3.6705 - val\_loss: 14.5543 - val\_mean\_absolute\_error: 2.9448  
Epoch 11/300  
171/171 [==============================] - 0s 682us/step - loss: 23.6316 - mean\_absolute\_error: 3.6090 - val\_loss: 14.5897 - val\_mean\_absolute\_error: 2.9397  
Epoch 12/300  
171/171 [==============================] - 0s 673us/step - loss: 22.2146 - mean\_absolute\_error: 3.5071 - val\_loss: 15.5073 - val\_mean\_absolute\_error: 3.1438  
Epoch 13/300  
171/171 [==============================] - 0s 568us/step - loss: 22.2247 - mean\_absolute\_error: 3.5623 - val\_loss: 16.0700 - val\_mean\_absolute\_error: 3.2230  
Epoch 14/300  
171/171 [==============================] - 0s 663us/step - loss: 21.3771 - mean\_absolute\_error: 3.4522 - val\_loss: 14.9895 - val\_mean\_absolute\_error: 2.9480  
Epoch 15/300  
171/171 [==============================] - 0s 830us/step - loss: 21.5314 - mean\_absolute\_error: 3.4580 - val\_loss: 15.6799 - val\_mean\_absolute\_error: 3.1458  
Epoch 16/300  
171/171 [==============================] - 0s 655us/step - loss: 20.4172 - mean\_absolute\_error: 3.4347 - val\_loss: 15.5421 - val\_mean\_absolute\_error: 3.1033  
Epoch 17/300  
171/171 [==============================] - 0s 732us/step - loss: 20.0186 - mean\_absolute\_error: 3.3983 - val\_loss: 15.7394 - val\_mean\_absolute\_error: 3.1172  
Epoch 18/300  
171/171 [==============================] - 0s 694us/step - loss: 19.6416 - mean\_absolute\_error: 3.3765 - val\_loss: 16.0938 - val\_mean\_absolute\_error: 3.1637  
Epoch 19/300  
171/171 [==============================] - 0s 671us/step - loss: 19.7924 - mean\_absolute\_error: 3.4336 - val\_loss: 16.4807 - val\_mean\_absolute\_error: 3.2113  
Epoch 20/300  
171/171 [==============================] - 0s 711us/step - loss: 19.0623 - mean\_absolute\_error: 3.3359 - val\_loss: 16.0409 - val\_mean\_absolute\_error: 3.0794  
Epoch 21/300  
171/171 [==============================] - 0s 605us/step - loss: 18.9553 - mean\_absolute\_error: 3.3054 - val\_loss: 16.8043 - val\_mean\_absolute\_error: 3.2327  
Epoch 22/300  
171/171 [==============================] - 0s 591us/step - loss: 18.7965 - mean\_absolute\_error: 3.3551 - val\_loss: 17.2816 - val\_mean\_absolute\_error: 3.2939  
Epoch 23/300  
171/171 [==============================] - 0s 566us/step - loss: 18.9399 - mean\_absolute\_error: 3.3351 - val\_loss: 16.3736 - val\_mean\_absolute\_error: 3.0679  
Epoch 24/300  
171/171 [==============================] - 0s 603us/step - loss: 18.3353 - mean\_absolute\_error: 3.2753 - val\_loss: 17.7674 - val\_mean\_absolute\_error: 3.3537  
Epoch 25/300  
171/171 [==============================] - 0s 1ms/step - loss: 18.2173 - mean\_absolute\_error: 3.3008 - val\_loss: 16.6106 - val\_mean\_absolute\_error: 3.1064  
Epoch 26/300  
171/171 [==============================] - 0s 1ms/step - loss: 18.3380 - mean\_absolute\_error: 3.2533 - val\_loss: 16.6205 - val\_mean\_absolute\_error: 3.1195  
Epoch 27/300  
171/171 [==============================] - 0s 568us/step - loss: 18.6424 - mean\_absolute\_error: 3.3697 - val\_loss: 17.3186 - val\_mean\_absolute\_error: 3.2750  
Epoch 28/300  
171/171 [==============================] - 0s 843us/step - loss: 18.7328 - mean\_absolute\_error: 3.3429 - val\_loss: 16.5648 - val\_mean\_absolute\_error: 3.0187  
Epoch 29/300  
171/171 [==============================] - 0s 930us/step - loss: 17.7666 - mean\_absolute\_error: 3.2192 - val\_loss: 17.8601 - val\_mean\_absolute\_error: 3.3623  
Epoch 30/300  
171/171 [==============================] - 0s 571us/step - loss: 17.7363 - mean\_absolute\_error: 3.2998 - val\_loss: 16.5255 - val\_mean\_absolute\_error: 3.0624

#check the model performace in test dataset  
score = model.evaluate(X\_test, y\_test, verbose=1)  
print('loss value: ', score[0])  
print('Mean absolute error: ', score[1])

54/54 [==============================] - 0s 235us/step  
loss value: 15.922834820217556  
Mean absolute error: 2.903298015947695

# predict house price using the test data   
test\_predictions = model.predict(X\_test).flatten()  
print(test\_predictions)

[18.709263 24.179756 21.231794 18.435421 18.776245 23.072487 17.167145  
 22.763042 22.791042 23.87087 18.341373 22.413925 23.46059 21.999714  
 21.186483 23.117607 22.900814 23.213331 20.809013 18.835995 19.954365  
 21.541697 19.941307 22.95221 23.379967 23.682293 20.848503 24.778103  
 23.419899 19.07947 24.877907 23.532867 22.197716 17.287327 21.742983  
 20.573864 18.542067 21.40238 24.69552 18.346947 20.827618 19.10169  
 21.3207 17.233566 22.04654 20.005966 22.29559 22.386326 20.634472  
 22.941322 17.13425 17.214645 23.408298 17.418644]

# show the true value and predicted value in dataframe  
true\_predicted = pd.DataFrame(list(zip(y\_test, test\_predictions)), columns=['True Value','Predicted Value'])  
true\_predicted.head(10)

True Value Predicted Value  
0 23.0 18.709263  
1 18.7 24.179756  
2 14.5 21.231794  
3 19.1 18.435421  
4 18.1 18.776245  
5 23.3 23.072487  
6 17.7 17.167145  
7 23.2 22.763042  
8 18.9 22.791042  
9 28.0 23.870871

# visualize the prediction uisng diagonal line  
y = test\_predictions #y-axis  
x = y\_test #x-axis  
fig, ax = plt.subplots(figsize=(10,6)) # create figure  
ax.scatter(x,y) #scatter plots for x,y  
ax.set(xlim=(0,55), ylim=(0, 55)) #set limit  
ax.plot(ax.get\_xlim(), ax.get\_ylim(), color ='red') # draw 45 degree diagonal in figure  
plt.xlabel('True Values')  
plt.ylabel('Predicted values')  
plt.title('Evaluation Result')  
plt.show()

