In [1]:

```
# Applying linear regression model to fake housing data to predict the prices
```

In [2]:

```
# Usual imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:

```
df = pd.read_csv('USA_Housing.csv')
```

In [4]:

```
# Let's checkout the info() method which provides information about shape, name of colu
mns, data type, non-null values
df.info() # clearly the 'Prices' is the target column
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
Avg. Area Income                5000 non-null float64
Avg. Area House Age             5000 non-null float64
Avg. Area Number of Rooms       5000 non-null float64
Avg. Area Number of Bedrooms    5000 non-null float64
Area Population                 5000 non-null float64
Price                           5000 non-null float64
Address                         5000 non-null object
dtypes: float64(6), object(1)
memory usage: 273.5+ KB
```

In [5]:

```
df.describe()
```

Out[5]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.00000
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.23207
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.53117
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.59386
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.97577
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.23266
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.47121
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.46906

In [6]:

```
df.columns
```

Out[6]:

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Roo
ms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Addres
s'],
      dtype='object')
```

In [7]:

```
# Generally speaking column_names without a underscore is not my thing and I like to co
nvert it to one with underscores
df.columns = df.columns.str.replace(' ','_')
df.columns
```

Out[7]:

```
Index(['Avg._Area_Income', 'Avg._Area_House_Age', 'Avg._Area_Number_of_Roo
ms',
       'Avg._Area_Number_of_Bedrooms', 'Area_Population', 'Price', 'Addres
s'],
      dtype='object')
```

In [8]:

```
df.head()
```

Out[8]:

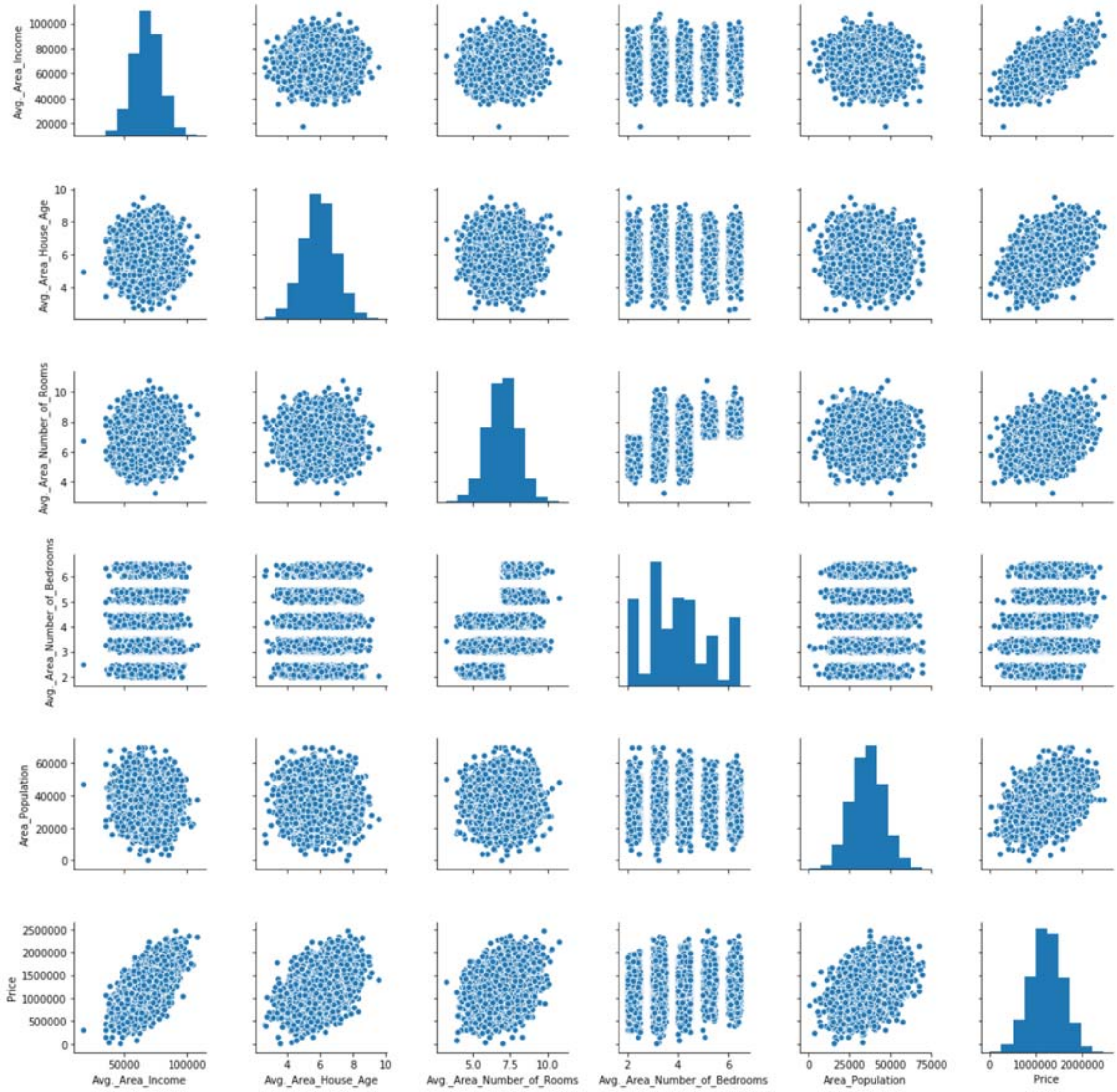
	Avg._Area_Income	Avg._Area_House_Age	Avg._Area_Number_of_Rooms	Avg._A
0	79545.458574	5.682861	7.009188	4.09
1	79248.642455	6.002900	6.730821	3.09
2	61287.067179	5.865890	8.512727	5.13
3	63345.240046	7.188236	5.586729	3.26
4	59982.197226	5.040555	7.839388	4.23

In [9]:

```
sns.pairplot(df)
```

Out[9]:

```
<seaborn.axisgrid.PairGrid at 0x1fe86b2b278>
```

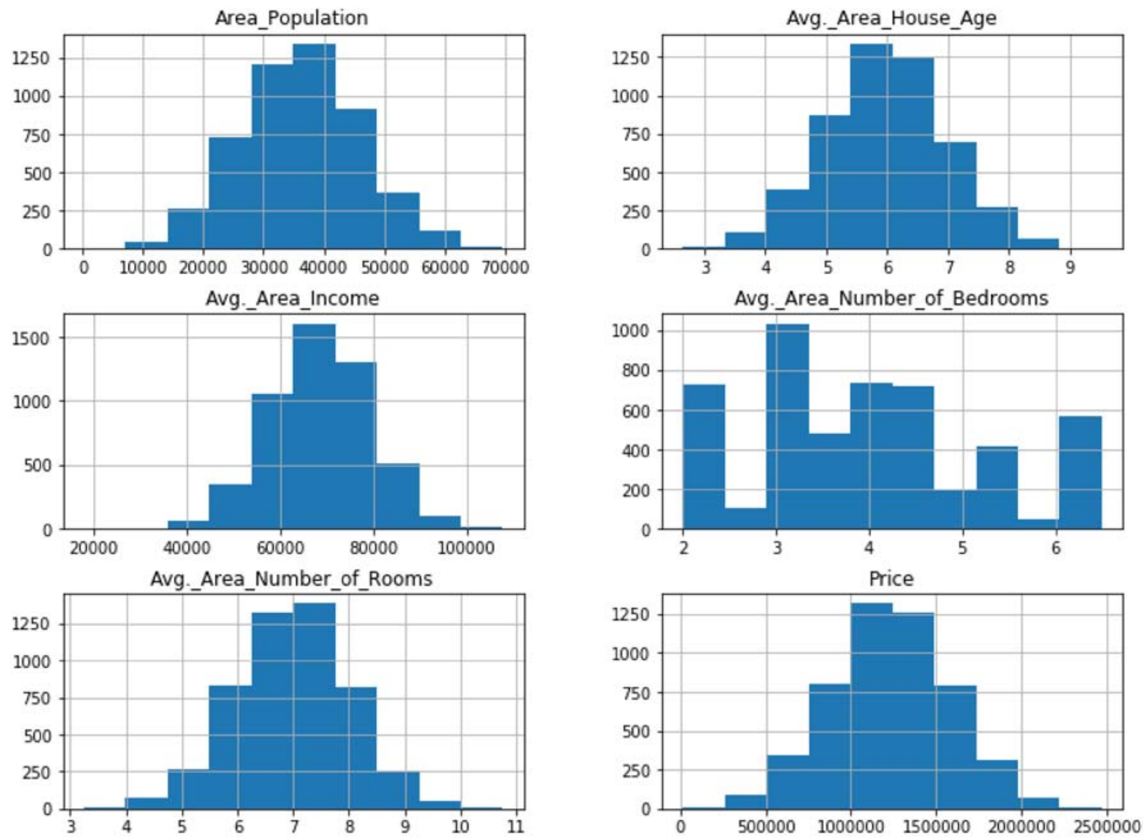

Avg._Area_Income
Avg._Area_House_Age
Avg._Area_Number_of_Rooms
Avg._Area_Number_of_Bedrooms
Area_Population
Price

In [10]:

```
df.hist(figsize=(12,9))
```

Out[10]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001FE8902635
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001FE8900BA2
0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001FE8907C32
0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001FE8952889
8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001FE89558E4
8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001FE8959443
8>]],
      dtype=object)
```

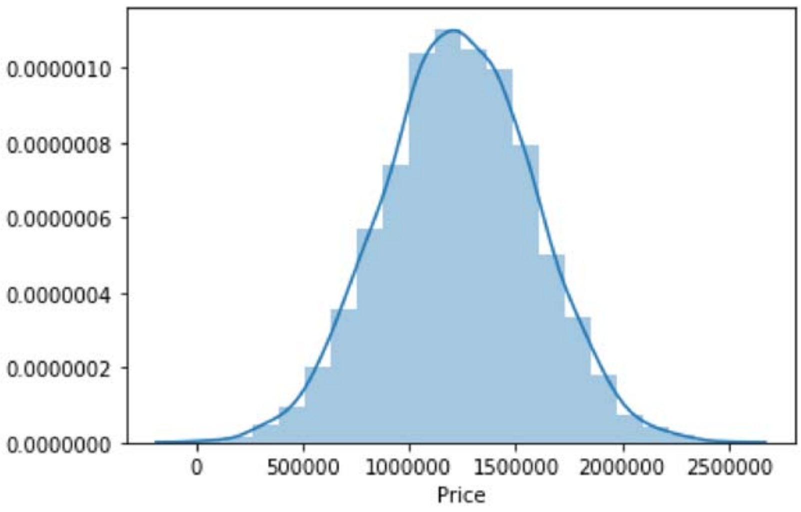


In [11]:

```
sns.distplot(df['Price'],bins=20)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe899990f0>
```



In [12]:

```
df.corr()
```

Out[12]:

	Avg._Area_Income	Avg._Area_House_Age	Avg._A
Avg._Area_Income	1.000000	-0.002007	-0.011
Avg._Area_House_Age	-0.002007	1.000000	-0.009
Avg._Area_Number_of_Rooms	-0.011032	-0.009428	1.0000
Avg._Area_Number_of_Bedrooms	0.019788	0.006149	0.4626
Area_Population	-0.016234	-0.018743	0.0020
Price	0.639734	0.452543	0.3356

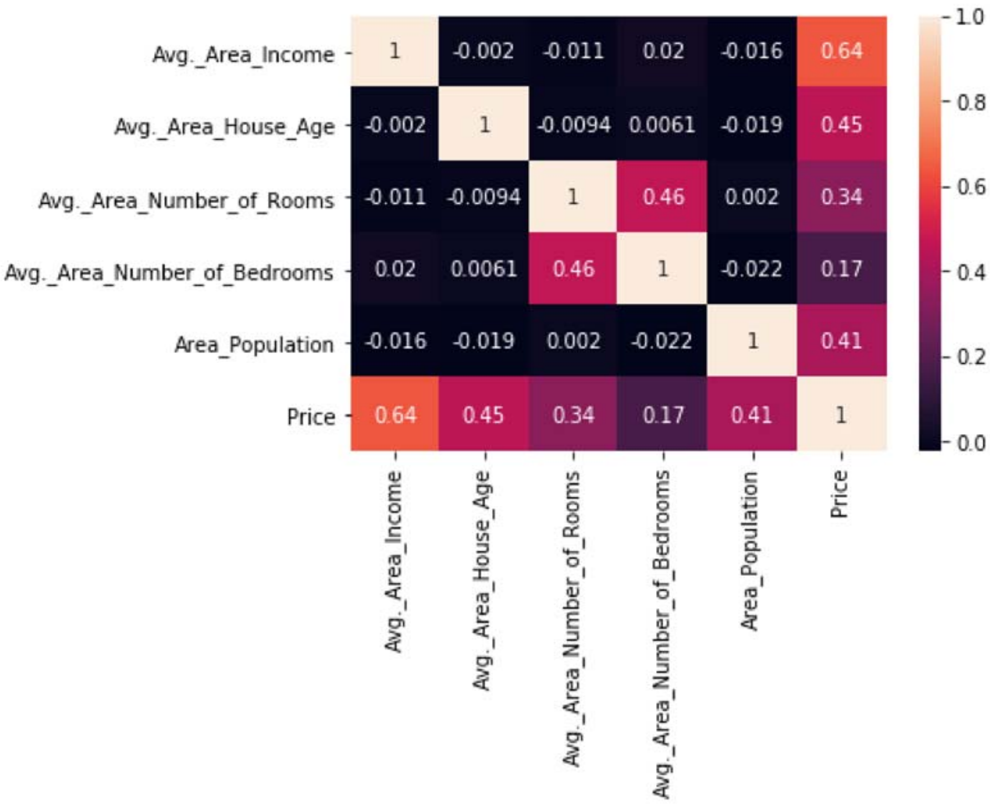
In [13]:

```
# Above table is hard to read and comprehend
# The below chart solves this problem

sns.heatmap(df.corr(), annot=True)
```

Out[13]:

<matplotlib.axes._subplots.AxesSubplot at 0x1fe896ccc88>



In [14]:

```
# Copy the numerical column names from df.columns excluding the 'Price' column to make
 new (class)data frame
# This data frame is the set of all the veriables that affect the price of the perticul
ar house
# and Y as the target or the 'Price' dataframe
X = df[['Avg._Area_Income', 'Avg._Area_House_Age', 'Avg._Area_Number_of_Rooms',
       'Avg._Area_Number_of_Bedrooms', 'Area_Population']]
y = df['Price']
```

In [15]:

```
# Let's check X and y (head())
print(X.head())
print('--------------')
print('Data Type of X is: ',type(X))
print('**************')
print(y.head())
print('--------------')
print('Data Type of y is: ',type(y))
```

```
   Avg._Area_Income  Avg._Area_House_Age  Avg._Area_Number_of_Rooms  \
0      79545.458574             5.682861                   7.009188   
1      79248.642455             6.002900                   6.730821   
2      61287.067179             5.865890                   8.512727   
3      63345.240046             7.188236                   5.586729   
4      59982.197226             5.040555                   7.839388   

   Avg._Area_Number_of_Bedrooms  Area_Population  
0                          4.09     23086.800503  
1                          3.09     40173.072174  
2                          5.13     36882.159400  
3                          3.26     34310.242831  
4                          4.23     26354.109472  
--------------
Data Type of X is:  <class 'pandas.core.frame.DataFrame'>
**************
0    1.059034e+06
1    1.505891e+06
2    1.058988e+06
3    1.260617e+06
4    6.309435e+05
Name: Price, dtype: float64
--------------
Data Type of y is:  <class 'pandas.core.series.Series'>
```

In [16]:

```
from sklearn.model_selection import train_test_split
```

In [17]:

```
# unpacking train_test_split tuple
# This will split the dataset in two groups called training set and test set
# Because we have y which is our target, the model will work with training set and then
we can confirm the accuracy on the test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1
01)
# Random state is chosen to 101 to get exactly same result what I got last time as last
time also I had chosen 101
```

In [18]:

```
# import the LinearRegression model
from sklearn.linear_model import LinearRegression
# Make an instance of this model or instantiate the model
lm = LinearRegression()
```

In [19]:

```
# Let's fit the training set to the model
lm.fit(X_train, y_train)
```

Out[19]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=F
alse)
```

In [22]:

```
# We need two important values from this model
# 1. intercept which you can get it from the attribute lm.intercept_
# 2. coefficient which you can get it from the attribute lm.coef_
print('Intercept is: ',lm.intercept_)
print('Set of coefficients is: ',lm.coef_)
```

```
Intercept is:  -2640159.796851911
Set of coefficients is:  [2.15282755e+01 1.64883282e+05 1.22368678e+05 2.2
3380186e+03
 1.51504200e+01]
```

In [23]:

```
# Intercept shown above do not talk to us directly
# So we use a better way of representing them
# Let's make a data frame out of it
# the columns names corresponding to the co-efficients can be index
# The coefficients can be the body of the DataFrame
# and column name can be coeff
Coeff_df = pd.DataFrame(lm.coef_, index=X.columns, columns=['Co-efficients'])
Coeff_df
```

Out[23]:

	Co-efficients
Avg._Area_Income	21.528276
Avg._Area_House_Age	164883.282027
Avg._Area_Number_of_Rooms	122368.678027
Avg._Area_Number_of_Bedrooms	2233.801864
Area_Population	15.150420

In [24]:

```
# We still haven't passed the test set to this model
# And we still haven't compared its generation of y values and our already available y
 values
pred_values = lm.predict(X_test)
print(type(pred_values))
print('***************')
a = pred_values/y_test
print(a)
```

```
<class 'numpy.ndarray'>
**************
1718    1.007408
2511    0.947930
345     1.026779
2521    0.916032
54      1.052625
2866    0.884542
2371    0.928265
2952    1.212477
45      0.963788
4653    0.933742
891     1.008175
3011    1.016302
335     0.987783
3050    0.887521
3850    0.964823
834     1.094442
3188    0.849544
4675    0.848376
2564    1.136980
1866    0.865799
1492    1.263964
3720    1.218885
618     1.044775
3489    0.998544
2145    0.963020
3200    0.987091
4752    0.985603
602     1.011727
4665    1.016204
79      1.035449
          ...
4668    0.923755
3762    1.342201
236     1.026918
4897    0.948148
1283    1.157236
2443    1.282226
3600    1.032961
2138    1.067112
254     1.174238
3987    0.918753
527     0.977258
1362    0.988514
4577    0.940858
2642    0.984865
4297    0.967352
1114    0.971898
1041    0.984857
3666    1.055580
4571    0.926021
3698    0.955747
412     0.986969
3385    0.995617
3057    0.896668
3863    1.048679
4165    1.000425
1776    1.017135
4269    0.959212
1661    2.456270
```

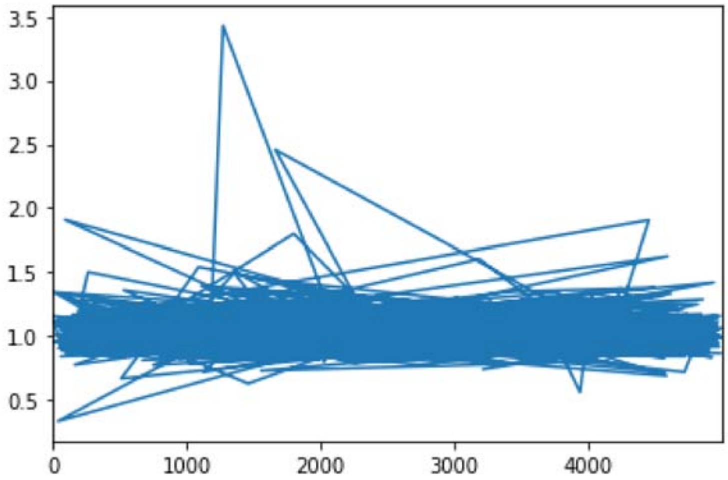
```
2410    1.015919
2302    1.004457
Name: Price, Length: 2000, dtype: float64
```

In [25]:

```
a.plot()
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe8b37c7f0>
```

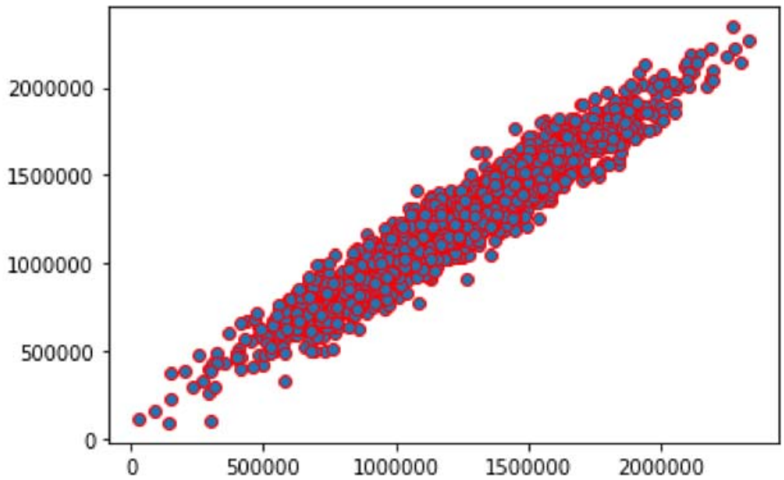


In [26]:

```
plt.scatter(y_test,pred_values,edgecolors='red')
```

Out[26]:

```
<matplotlib.collections.PathCollection at 0x1fe8b73de48>
```



In [27]:

```
f = (lambda x: x>320000)
above_320T = y_test.apply(f)
print(above_320T)
```

```
1718     True
2511     True
345      True
2521     True
54       True
2866     True
2371     True
2952     True
45       True
4653     True
891      True
3011     True
335      True
3050     True
3850     True
834      True
3188     True
4675     True
2564     True
1866     True
1492     True
3720     True
618      True
3489     True
2145     True
3200     True
4752     True
602      True
4665     True
79       True
        ...
4668     True
3762     True
236      True
4897     True
1283     True
2443     True
3600     True
2138     True
254      True
3987     True
527      True
1362     True
4577     True
2642     True
4297     True
1114     True
1041     True
3666     True
4571     True
3698     True
412      True
3385     True
3057     True
3863     True
4165     True
1776     True
4269     True
1661    False
2410     True
```

```
2302     True
Name: Price, Length: 2000, dtype: bool
```

In [28]:

```
sns.scatterplot(x=y_test,y=a,hue=above_320T)
# To me this is a great plot
# This plot explains that the accuracy of the model is better for predicting houses wit
h price above 320,000
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fe8b40c390>
```