In [335]:
```
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score
from sklearn import metrics
from sklearn.model_selection import train_test_split
from pandas import Series

import pandas as pd
import numpy as np
```

In [336]:
```
data = pd.read_csv('/Users/juanrquilesjr/Documents/Machine_Learning-Spring_2019/assignment-6/nba.csv')
```

In [337]:
```
data.head(5)
```

Out[337]:

	age	g	gs	mp	fg	fga	fg.	x3p	x3pa	x3p.	...	ft.	orb	drb	trb	ast	stl	blk	tov	pf	pts
0	23	63	0	847	66	141	0.468	4	15	0.266667	...	0.660	72	144	216	28	23	26	30	122	1
1	20	81	20	1197	93	185	0.503	0	0	0.000000	...	0.581	142	190	332	43	40	57	71	203	1
2	27	53	12	961	143	275	0.520	0	0	0.000000	...	0.639	102	204	306	38	24	36	39	108	1
3	28	73	73	2552	464	1011	0.459	128	300	0.426667	...	0.815	32	230	262	248	35	3	146	136	3
4	25	56	30	951	136	249	0.546	0	1	0.000000	...	0.836	94	183	277	40	23	46	63	187	1

5 rows × 26 columns

In [338]:
```
data.columns
```

Out[338]:
```
Index(['age', 'g', 'gs', 'mp', 'fg', 'fga', 'fg.', 'x3p', 'x3pa', 'x3p.',
       'x2p', 'x2pa', 'x2p.', 'efg.', 'ft', 'fta', 'ft.', 'orb', 'drb', 'trb',
       'ast', 'stl', 'blk', 'tov', 'pf', 'pts'],
      dtype='object')
```

In [339]:
```
#feature variables
X = data.drop('pts' ,axis =1).values

#target variables
y = data['pts'].values
```

**** L1 Regularization ****

In [350]:
```
# data divide into train and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=5)
```

In [351]:
```
#instantiate the LR model
lr = LogisticRegression(multi_class='multinomial', penalty = 'l1', solver='saga', tol=0.1)

# fit the model with data
lr.fit(X_train,y_train)
```

Out[351]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='multinomial',
          n_jobs=1, penalty='l1', random_state=None, solver='saga',
          tol=0.1, verbose=0, warm_start=False)
```

In [352]:
```
y_pred = lr.predict(X_test)
```

In [353]:
```
accuracy = metrics.accuracy_score(y_test,y_pred)

micro = f1_score(y_test, y_pred, average = 'micro')
macro = f1_score(y_test, y_pred, average = 'macro')
weighted = f1_score(y_test, y_pred, average = 'weighted')

#f_1 score/accuracy with l1 regularization
print('l1 accuracy: %.2f' % accuracy)
print('l1 Micro f1_score: %.2f' % micro)
print('l1 Micro f1_score: %.2f' % macro)
print('l1 Micro f1_score: %.2f' % weighted)
```

```
l1 accuracy: 0.81
l1 Micro f1_score: 0.81
l1 Micro f1_score: 0.45
l1 Micro f1_score: 0.80

/Users/juanrquilesjr/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: UndefinedMetricWarning: F-score is ill-defined
and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [354]:
```
lr.coef_
```

Out[354]:
```
array([[ 5.89972602e-03,  5.02549318e-03, -1.60039136e-03,
         3.71475227e-03, -3.29426162e-03, -4.58889455e-03,
         8.78728239e-05, -8.61213090e-04, -1.59313959e-03,
         3.88798976e-05, -2.43101789e-03, -2.99373159e-03,
         9.34367276e-05,  9.62770675e-05, -2.15718567e-03,
        -2.33539817e-03,  1.28258697e-04,  8.55167851e-04,
        -1.01348852e-04,  7.54995488e-04, -1.15384321e-03,
         4.07726515e-05,  3.93268264e-04,  2.18328828e-04,
         3.56244957e-03],
       [-1.62157761e-03, -9.85808088e-04,  1.42453229e-04,
         2.86859550e-03, -6.93181636e-04, -2.09886661e-03,
        -2.26219512e-05, -6.02993211e-05, -1.77574557e-05,
        -9.90828939e-06, -6.31006895e-04, -2.08032073e-03,
        -2.39680588e-05, -2.47715212e-05, -1.42907103e-03,
        -1.73182916e-03, -3.20167611e-05,  5.44023997e-04,
        -4.06319232e-05,  5.04888794e-04, -1.93070116e-03,
         1.43455892e-04, -6.52331435e-05, -8.00355908e-04,
         4.98733001e-04],
       [-1.86604900e-03, -1.91917858e-03,  7.62196570e-04,
        -4.49928074e-05,  3.83946030e-04,  9.95433257e-04,
        -2.68968307e-05,  6.73092640e-05,  1.26630303e-04,
        -1.04030588e-05,  3.14644427e-04,  8.66805244e-04,
        -2.87690642e-05, -2.97062778e-05,  6.25399798e-05,
         3.17556468e-04, -4.22899879e-05, -2.15487437e-06,
         7.26158225e-05,  6.88416894e-05,  4.77985389e-04,
        -2.20382360e-04,  1.33318696e-04,  4.05776215e-05,
        -1.77497830e-03],
       [-1.34003396e-03, -1.24229996e-03,  5.00848363e-04,
        -3.38919689e-03,  1.98473566e-03,  3.36339327e-03,
        -1.83665155e-05,  4.76865052e-04,  7.41923832e-04,
        -7.56351015e-06,  1.50588987e-03,  2.61949174e-03,
        -1.97373343e-05, -2.03039282e-05,  1.73641607e-03,
         1.79218246e-03, -2.75731256e-05, -9.09108761e-04,
        -1.38875639e-04, -1.05002204e-03,  2.01132886e-03,
         4.09006910e-05, -2.92844160e-04,  3.02581280e-04,
        -1.21980584e-03],
       [-1.06594858e-03, -8.72124424e-04,  1.89359643e-04,
        -3.14728784e-03,  1.61699784e-03,  2.32723158e-03,
        -1.38848111e-05,  3.75772441e-04,  7.39598788e-04,
        -5.00855410e-06,  1.23962047e-03,  1.58605091e-03,
        -1.48557314e-05, -1.53926250e-05,  1.78561677e-03,
         1.95582350e-03, -2.02913289e-05, -4.85887207e-04,
         2.08524590e-04, -2.77071396e-04,  5.93639997e-04,
        -6.55188890e-06, -1.66647139e-04,  2.34727221e-04,
        -1.06435863e-03]])
```

In [363]:

```
features = data.drop('pts' ,axis =1).columns
for i in lr.coef_:
    print(Series(i,features).sort_values())
```

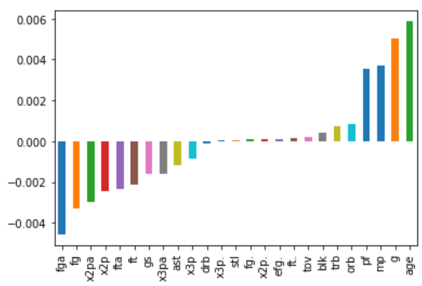
```
fga    -0.004589
fg     -0.003294
x2pa   -0.002994
x2p    -0.002431
fta    -0.002335
ft     -0.002157
gs     -0.001600
x3pa   -0.001593
ast    -0.001154
x3p    -0.000861
drb    -0.000101
x3p.    0.000039
stl     0.000041
fg.     0.000088
x2p.    0.000093
efg.    0.000096
ft.     0.000128
tov     0.000218
blk     0.000393
trb     0.000755
orb     0.000855
pf      0.003562
mp      0.003715
g       0.005025
age     0.005900
dtype: float64
fga    -0.002099
x2pa   -0.002080
ast    -0.001931
fta    -0.001732
age    -0.001622
ft     -0.001429
g      -0.000986
tov    -0.000800
fg     -0.000693
x2p    -0.000631
blk    -0.000065
x3p    -0.000060
drb    -0.000041
ft.    -0.000032
efg.   -0.000025
x2p.   -0.000024
fg.    -0.000023
x3pa   -0.000018
x3p.   -0.000010
gs      0.000142
stl     0.000143
pf      0.000499
trb     0.000505
orb     0.000544
mp      0.002869
dtype: float64
g      -0.001919
age    -0.001866
pf     -0.001775
stl    -0.000220
mp     -0.000045
ft.    -0.000042
efg.   -0.000030
x2p.   -0.000029
fg.    -0.000027
x3p.   -0.000010
orb    -0.000002
tov     0.000041
ft      0.000063
x3p     0.000067
trb     0.000069
drb     0.000073
x3pa    0.000127
blk     0.000133
x2p     0.000315
fta     0.000318
fg      0.000384
ast     0.000478
gs      0.000762
x2pa    0.000867
fga     0.000995
dtype: float64
mp     -0.003389
age    -0.001340
g      -0.001242
pf     -0.001220
trb    -0.001050
orb    -0.000909
blk    -0.000293
drb    -0.000139
ft.    -0.000028
efg.   -0.000020
x2p.   -0.000020
fg.    -0.000018
x3p.   -0.000008
stl     0.000041
tov     0.000303
x3p     0.000477
gs      0.000501
x3pa    0.000742
x2p     0.001506
ft      0.001736
fta     0.001792
fg      0.001985
ast     0.002011
x2pa    0.002619
fga     0.003363
dtype: float64
mp     -0.003147
age    -0.001066
```

```
pf      -0.001064
g       -0.000872
orb     -0.000486
trb     -0.000277
blk     -0.000167
ft.     -0.000020
efg.    -0.000015
x2p.    -0.000015
fg.     -0.000014
stl     -0.000007
x3p.    -0.000005
gs       0.000189
drb      0.000209
tov      0.000235
x3p      0.000376
ast      0.000594
x3pa     0.000740
x2p      0.001240
x2pa     0.001586
fg       0.001617
ft       0.001786
fta      0.001956
fga      0.002327
dtype: float64
```

**** Coefficient Visualizations(least and most important) ****

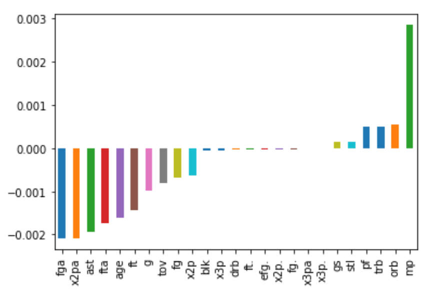
In [356]:
```
coef = Series(lr.coef_[0],features).sort_values()
coef.plot(kind = 'bar')

# Top 3 features:  mp, g, age
```

Out[356]: <matplotlib.axes._subplots.AxesSubplot at 0x1a24524630>



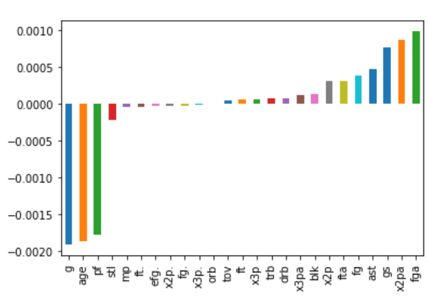
In [357]:
```
coef = Series(lr.coef_[1],features).sort_values()
coef.plot(kind = 'bar')

#Top 3 features:  trb, orb, mp
```

Out[357]: <matplotlib.axes._subplots.AxesSubplot at 0x117f62e80>



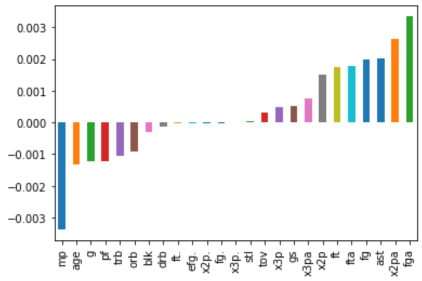
In [358]:
```
coef = Series(lr.coef_[2],features).sort_values()
coef.plot(kind = 'bar')
```

Out[358]: <matplotlib.axes._subplots.AxesSubplot at 0x117d78e80>



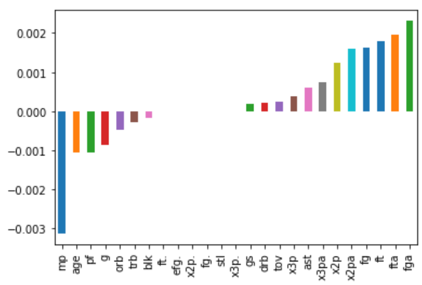
In [359]:
```
coef = Series(lr.coef_[3],features).sort_values()
coef.plot(kind = 'bar')

#top 3 features:  ast, x2pa, fga
```

Out[359]: <matplotlib.axes._subplots.AxesSubplot at 0x117d97d68>



In [360]:
```
coef = Series(lr.coef_[4],features).sort_values()
coef.plot(kind = 'bar')

#top 3 features: ft, fta, fga
```

Out[360]: <matplotlib.axes._subplots.AxesSubplot at 0x1180479e8>



**** L2 Regularization ****

In [364]:
```
# data divide into train and testing sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state=5)
```

In [365]:
```
#instantiate the LR model
lr = LogisticRegression(multi_class='multinomial', penalty = 'l2', solver='saga', tol=0.1)

# fit the model with data
lr.fit(X_train,y_train)
```

Out[365]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='multinomial',
          n_jobs=1, penalty='l2', random_state=None, solver='saga',
          tol=0.1, verbose=0, warm_start=False)
```

In [366]:
```
y_pred = lr.predict(X_test)

accuracy = metrics.accuracy_score(y_test,y_pred)

micro = f1_score(y_test, y_pred, average = 'micro')
macro = f1_score(y_test, y_pred, average = 'macro')
weighted = f1_score(y_test, y_pred, average = 'weighted')

#f_1 score/accuracy with l2 regularization
print('l2 accuracy: %.2f' % accuracy)
print('l2 Micro f1_score: %.2f' % micro)
print('l2 Micro f1_score: %.2f' % macro)
print('l2 Micro f1_score: %.2f' % weighted)
```

```
l2 accuracy: 0.77
l2 Micro f1_score: 0.77
l2 Micro f1_score: 0.42
l2 Micro f1_score: 0.76

/Users/juanrquilesjr/anaconda3/lib/python3.6/site-packages/sklearn/metrics/classification.py:1135: UndefinedMetricWarning: F-score is ill-defined
and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [367]:
```
lr.coef_
```

Out[367]:
```
array([[ 2.96888374e-03,  2.64774640e-03, -8.62457354e-04,
         2.73977248e-03, -2.04163540e-03, -3.25124393e-03,
         4.50394978e-05, -4.91606230e-04, -9.24648392e-04,
         2.11642790e-05, -1.55002917e-03, -2.32659554e-03,
         4.81767598e-05,  4.93970246e-05, -1.43276168e-03,
        -1.59107716e-03,  6.62735319e-05,  5.12569886e-04,
         1.29869750e-04,  6.42439636e-04, -7.98654421e-04,
         8.71803367e-05,  2.64448298e-04, -3.30603814e-05,
         2.00067068e-03],
       [-7.51805426e-04, -4.01962990e-04,  8.62620375e-05,
         1.98185599e-03, -5.21767364e-04, -1.25169810e-03,
        -1.13006997e-05,  7.99367335e-05,  2.48081710e-04,
        -5.54871170e-06, -6.01704097e-04, -1.49977981e-03,
        -1.20017235e-05, -1.22344907e-05, -9.73117449e-04,
        -1.18094317e-03, -1.55303999e-05,  4.30788486e-04,
         8.20501201e-05,  5.12838606e-04, -1.21368227e-03,
        -1.58825330e-05, -3.27618370e-05, -4.56375811e-04,
         3.70021381e-04],
       [-9.47024490e-04, -1.03259040e-03,  4.26940675e-04,
        -3.73991721e-04,  4.89043220e-04,  1.14423416e-03,
        -1.46657383e-05,  3.46682150e-05,  6.09270384e-05,
        -6.59887393e-06,  4.54375005e-04,  1.08330712e-03,
        -1.57212449e-05, -1.61858940e-05,  3.31589937e-04,
         5.08525200e-04, -2.27024038e-05, -1.32556389e-04,
        -1.20950531e-04, -2.53506921e-04,  4.73918379e-04,
        -9.50569055e-05,  5.45229958e-05,  1.08432318e-04,
        -1.02204961e-03],
       [-7.24027525e-04, -7.35245702e-04,  2.60605443e-04,
        -2.32697150e-03,  1.30219521e-03,  2.25461754e-03,
        -1.09870180e-05,  2.35578898e-04,  3.70065308e-04,
        -5.21452149e-06,  1.06661631e-03,  1.88455224e-03,
        -1.17991362e-05, -1.20686980e-05,  1.19380784e-03,
         1.29762518e-03, -1.63438841e-05, -5.66224867e-04,
        -1.27500279e-04, -6.93725146e-04,  1.26135043e-03,
         4.61758653e-05, -1.91843472e-04,  2.72826343e-04,
        -7.67181662e-04],
       [-5.46026302e-04, -4.77947301e-04,  8.86491990e-05,
        -2.02066525e-03,  7.72164335e-04,  1.10409032e-03,
        -8.08604175e-06,  1.41422384e-04,  2.45574336e-04,
        -3.80217194e-06,  6.30741951e-04,  8.58515985e-04,
        -8.65465528e-06, -8.90794193e-06,  8.80481349e-04,
         9.65869955e-04, -1.16968441e-05, -2.44577116e-04,
         3.65309406e-05, -2.08046176e-04,  2.77067884e-04,
        -2.24167636e-05, -9.43659850e-05,  1.08177532e-04,
        -5.81460795e-04]])
```

**** Coefficient Visualizations (least and most important) ****

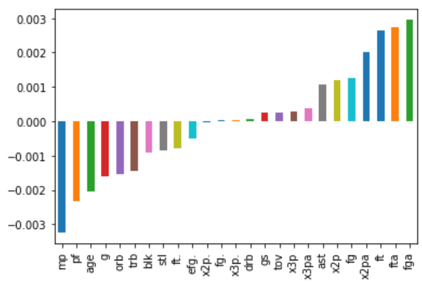
In [371]:

```
features = data.drop('pts' ,axis =1).columns
for i in lr.coef_:
    print(Series(i,features).sort_values())
```

```
fga    -0.003251
x2pa   -0.002327
fg     -0.002042
fta    -0.001591
x2p    -0.001550
ft     -0.001433
x3pa   -0.000925
gs     -0.000862
ast    -0.000799
x3p    -0.000492
tov    -0.000033
x3p.    0.000021
fg.     0.000045
x2p.    0.000048
efg.    0.000049
ft.     0.000066
stl     0.000087
drb     0.000130
blk     0.000264
orb     0.000513
trb     0.000642
pf      0.002001
g       0.002648
mp      0.002740
age     0.002969
dtype: float64
x2pa   -0.001500
fga    -0.001252
ast    -0.001214
fta    -0.001181
ft     -0.000973
age    -0.000752
x2p    -0.000602
fg     -0.000522
tov    -0.000456
g      -0.000402
blk    -0.000033
stl    -0.000016
ft.    -0.000016
efg.   -0.000012
x2p.   -0.000012
fg.    -0.000011
x3p.   -0.000006
x3p     0.000080
drb     0.000082
gs      0.000086
x3pa    0.000248
pf      0.000370
orb     0.000431
trb     0.000513
mp      0.001982
dtype: float64
g      -0.001033
pf     -0.001022
age    -0.000947
mp     -0.000374
trb    -0.000254
orb    -0.000133
drb    -0.000121
stl    -0.000095
ft.    -0.000023
efg.   -0.000016
x2p.   -0.000016
fg.    -0.000015
x3p.   -0.000007
x3p     0.000035
blk     0.000055
x3pa    0.000061
tov     0.000108
ft      0.000332
gs      0.000427
x2p     0.000454
ast     0.000474
fg      0.000489
fta     0.000509
x2pa    0.001083
fga     0.001144
dtype: float64
mp     -0.002327
pf     -0.000767
g      -0.000735
age    -0.000724
trb    -0.000694
orb    -0.000566
blk    -0.000192
drb    -0.000128
ft.    -0.000016
efg.   -0.000012
x2p.   -0.000012
fg.    -0.000011
x3p.   -0.000005
stl     0.000046
x3p     0.000236
gs      0.000261
tov     0.000273
x3pa    0.000370
x2p     0.001067
ft      0.001194
ast     0.001261
fta     0.001298
fg      0.001302
x2pa    0.001885
fga     0.002255
dtype: float64
mp     -0.002021
pf     -0.000581
```

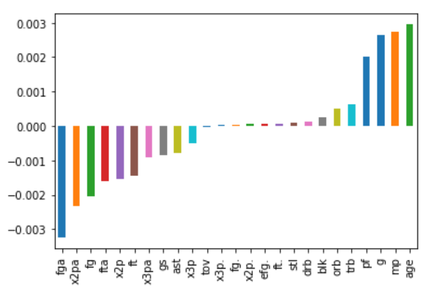
```
age    -0.000546
g      -0.000478
orb    -0.000245
trb    -0.000208
blk    -0.000094
stl    -0.000022
ft.    -0.000012
efg.   -0.000009
x2p.   -0.000009
fg.    -0.000008
x3p.   -0.000004
drb     0.000037
gs      0.000089
tov     0.000108
x3p     0.000141
x3pa    0.000246
ast     0.000277
x2p     0.000631
fg      0.000772
x2pa    0.000859
ft      0.000880
fta     0.000966
fga     0.001104
dtype: float64
```

In [376]:
```python
features = data.drop('pts' ,axis =1).columns
for i in lr.coef_:
    Series(i,features).sort_values().plot(kind = 'bar')
```



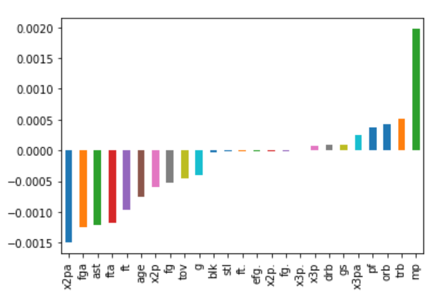
In [369]:
```python
coef = Series(lr.coef_[0],features).sort_values()
coef.plot(kind = 'bar')

#Top 3 features:  g, mp, age
```

Out[369]: <matplotlib.axes._subplots.AxesSubplot at 0x11811b780>



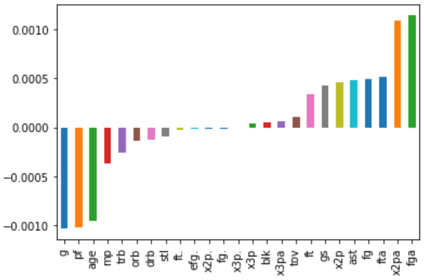
In [370]:
```python
coef = Series(lr.coef_[1],features).sort_values()
coef.plot(kind = 'bar')

#Top 3 features:  orb, trb, mp
```

Out[370]: <matplotlib.axes._subplots.AxesSubplot at 0x117dd4550>



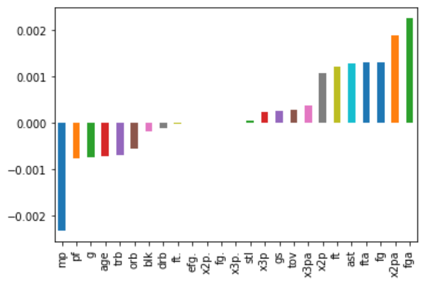
In [372]:
```
coef = Series(lr.coef_[2],features).sort_values()
coef.plot(kind = 'bar')

#Top 3 features:  fta, x2pa, fga
```

Out[372]: <matplotlib.axes._subplots.AxesSubplot at 0x1a243a17f0>



In [373]:
```
coef = Series(lr.coef_[3],features).sort_values()
coef.plot(kind = 'bar')

#Top 3 features:  fg, x2pa, fga
```

Out[373]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2448e828>



In [374]:
```
coef = Series(lr.coef_[4],features).sort_values()
coef.plot(kind = 'bar')

#Top 3 features:  ft, fta, fga
```

Out[374]: <matplotlib.axes._subplots.AxesSubplot at 0x1a24565438>