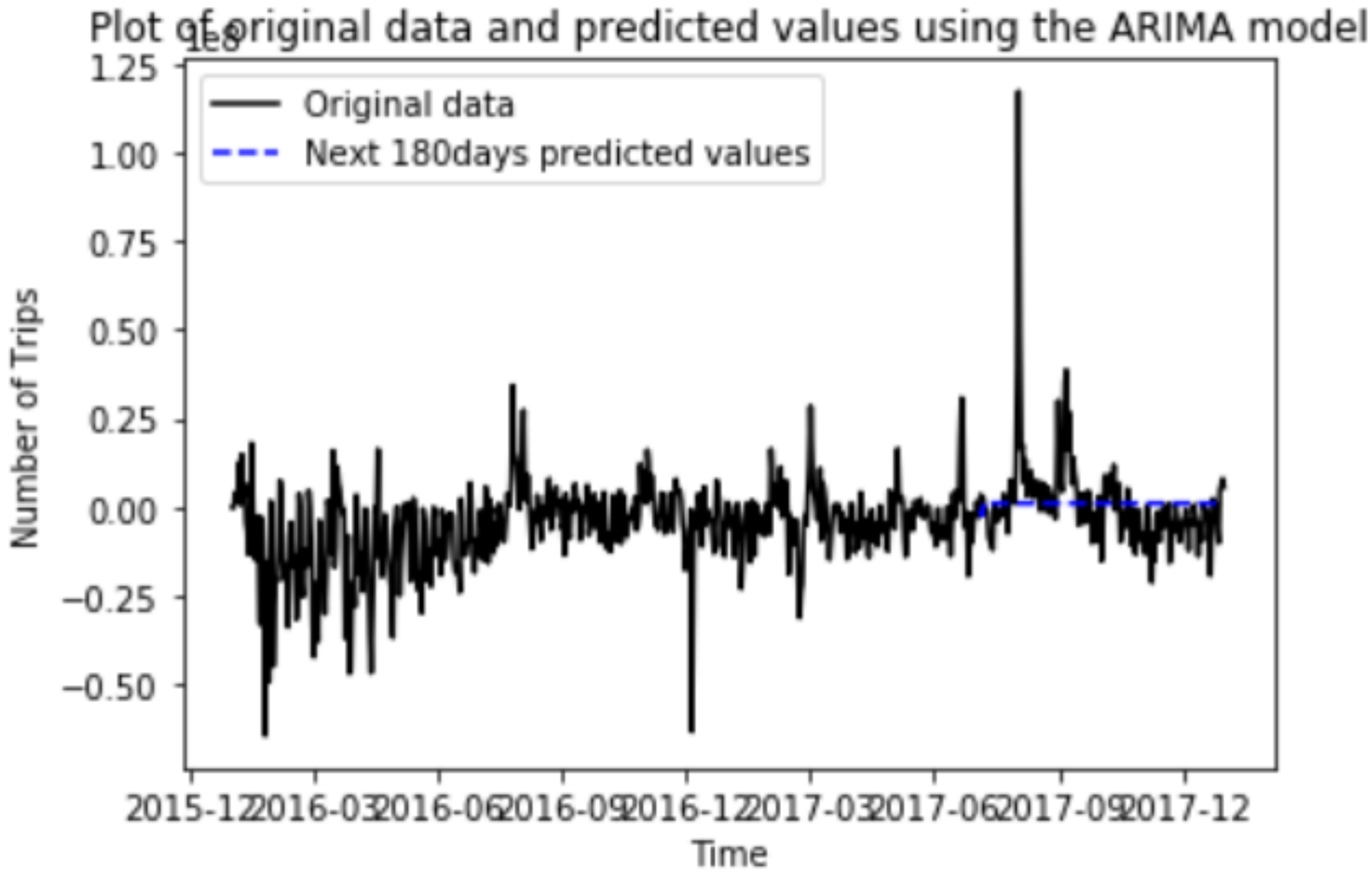
~~Time Series~~ predict

ARIMA


Plot of original data and predicted values using the ARIMA model
1e8
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
Number of Trips
Original data
Next 180days predicted values
2015-12
2016-03
2016-06
2016-09
2016-12
2017-03
2017-06
2017-09
2017-12
Time

RNN + LSTM

```
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
# invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
```

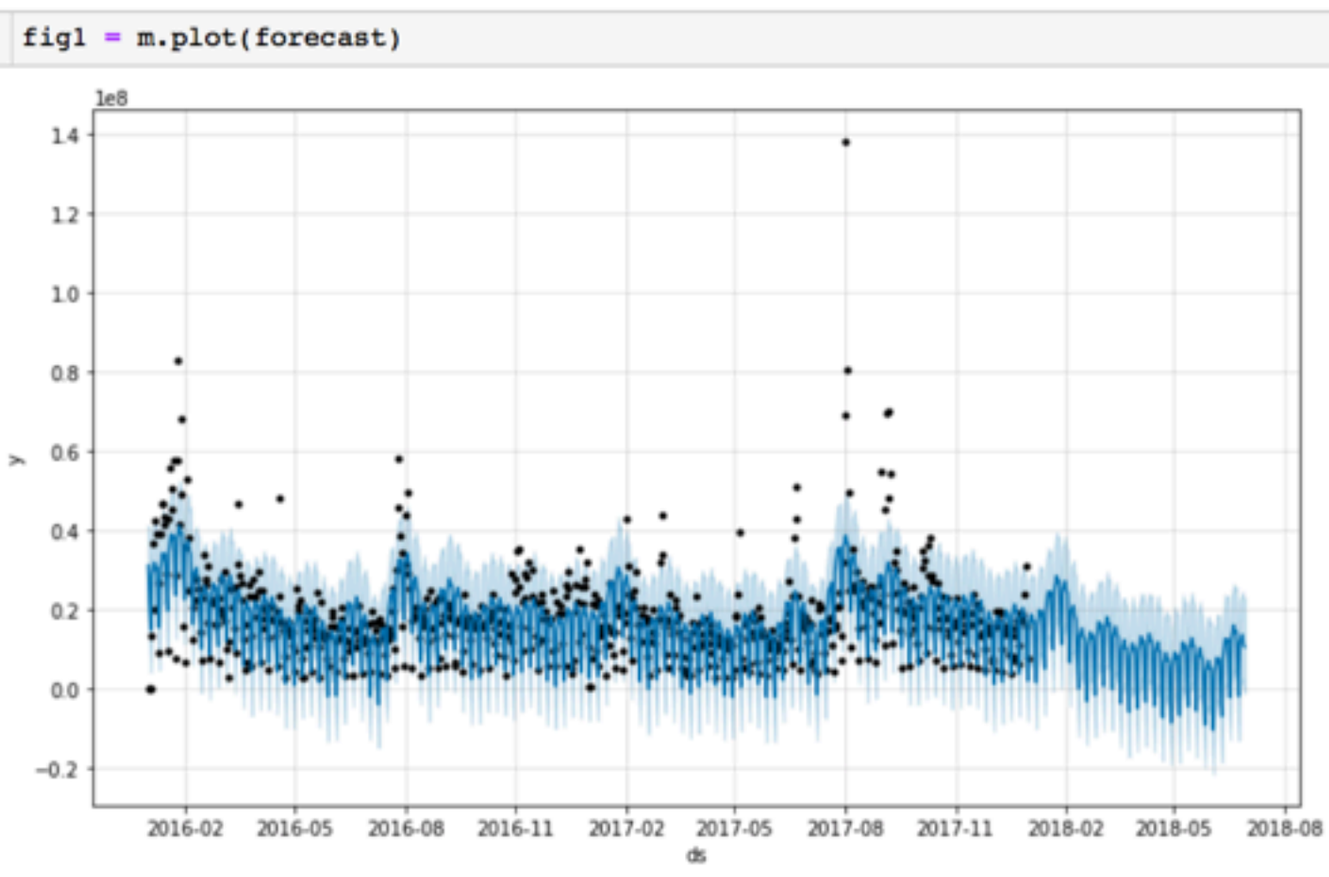
```
Train Score: 22294505.18 RMSE
Test Score: 22340019.78 RMSE
```

Prophet

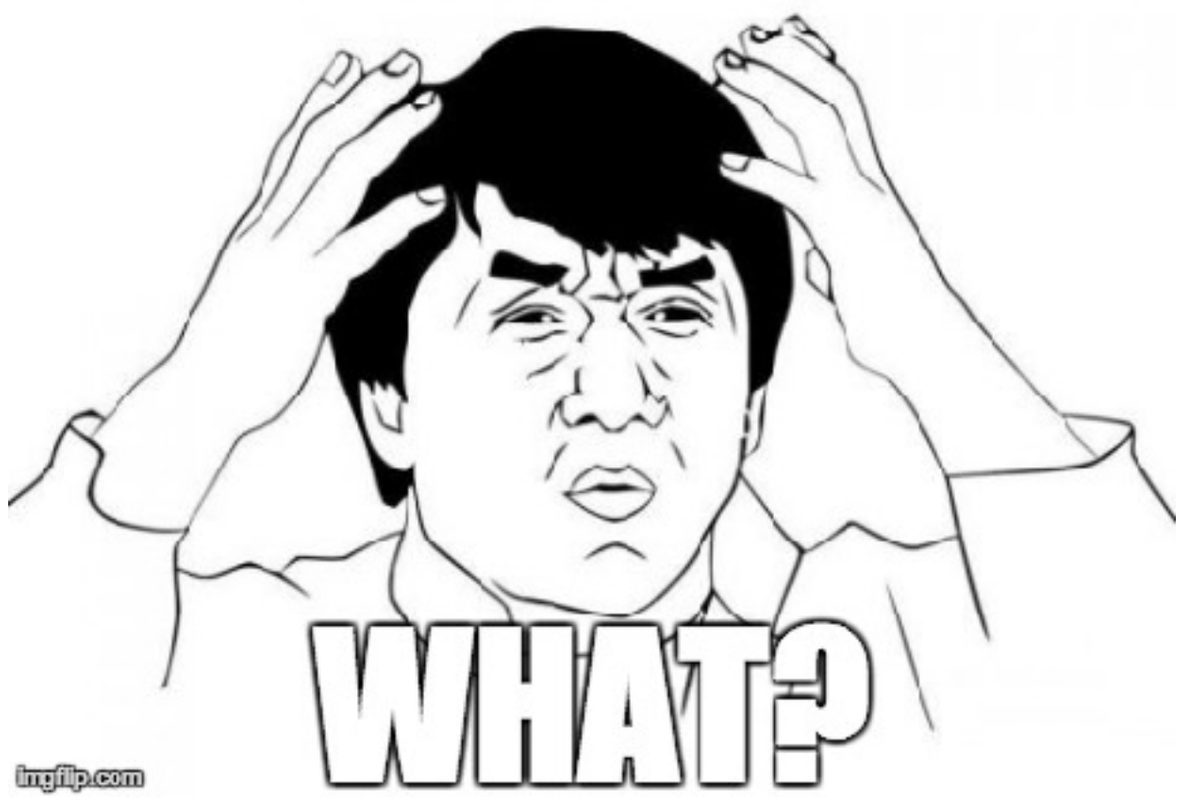
	ds	yhat	yhat_lower	yhat_upper
906	2018-06-25	1.168500e+07	8.331385e+05	2.277959e+07
907	2018-06-26	1.245475e+07	1.180051e+06	2.299161e+07
908	2018-06-27	1.354919e+07	2.462227e+06	2.444291e+07
909	2018-06-28	1.103810e+07	3.062443e+05	2.295632e+07
910	2018-06-29	1.050833e+07	-1.260338e+06	2.281156e+07

```
Train Score: 18038868.30 RMSE
Test Score: 15419039.01 RMSE
```

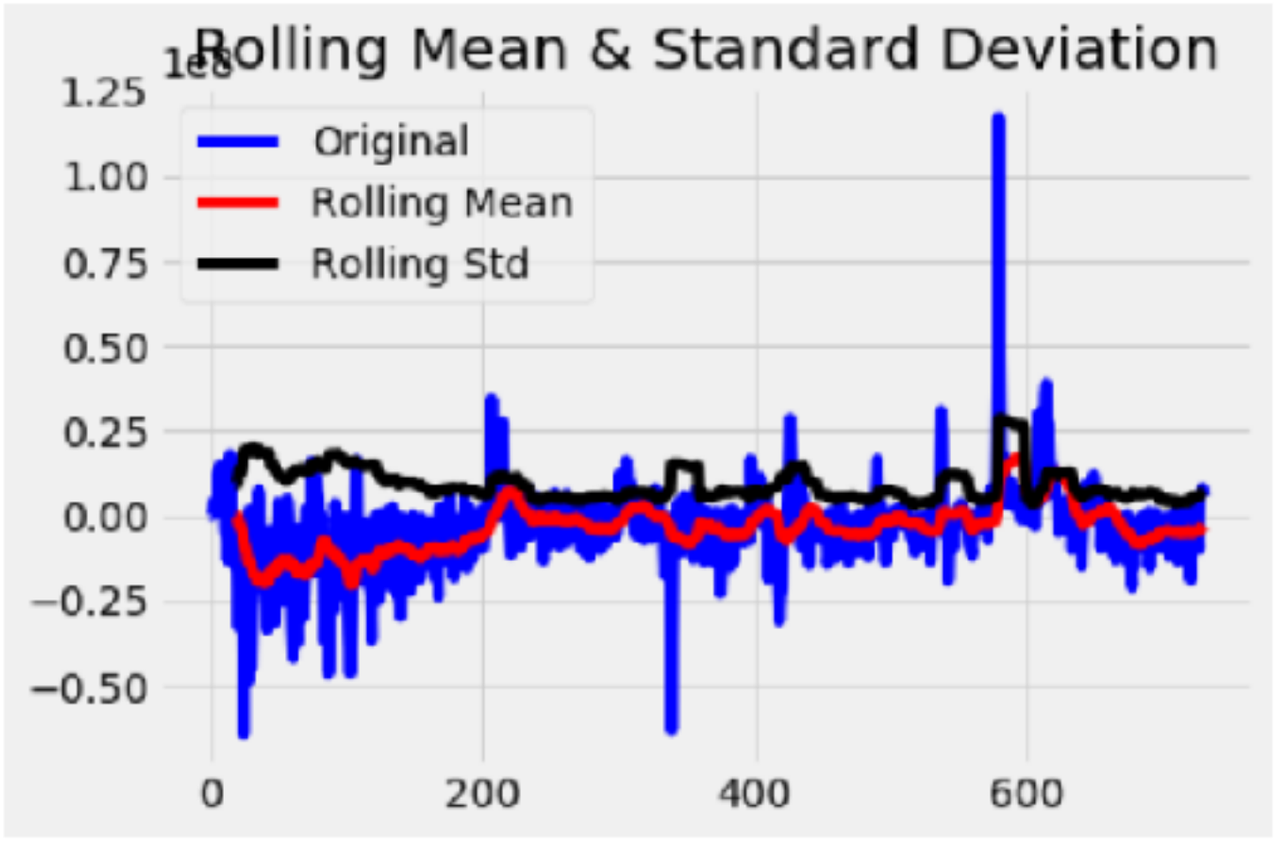
```
fig1 = m.plot(forecast)
```



Hmm, let’s drop date time!



Time series is stationary




```
Results of Dickey-Fuller Test:
Test Statistic                  -4.046451
p-value                          0.001186
#Lags Used                      13.000000
Number of Observations Used    717.000000
Critical Value (1%)             -3.439503
Critical Value (5%)             -2.865579
Critical Value (10%)            -2.568921
dtype: float64
```

- A stationary time series is one where values are not a function of time (null hypothesis - time series is non-stationary)

Real predictors

Predictor #1, sensitivity to exchange rate:

- Clients use deposit products to make money
- For example, when KZT devaluates they switch to FOREIGN CURRENCY deposits and vice versa

Exchange rate

DATE	USD	EUR	DELTA_NET_USD	DELTA_NET_EUR	DELTA_NET_FOR	DELTA_NET_KZT
1/8/2016	345.01	370.23	0.0000	0.0000	-94%	470%
1/9/2016	351.88	382.32	-0.0199	-0.0327	1991%	40%
1/10/2016	351.88	382.32	0.0000	0.0000	-82%	-64%
1/11/2016	351.88	382.32	0.0000	0.0000	154%	-235%
1/12/2016	356.88	388.46	-0.0142	-0.0161	-165%	-103%
1/13/2016	365.07	396.17	-0.0229	-0.0198	52%	-3870%
1/14/2016	364.89	394.67	0.0005	0.0038	100%	231%
1/22/2016	383.91	418.23	-0.0249	-0.0218	1211%	-127%

Classification

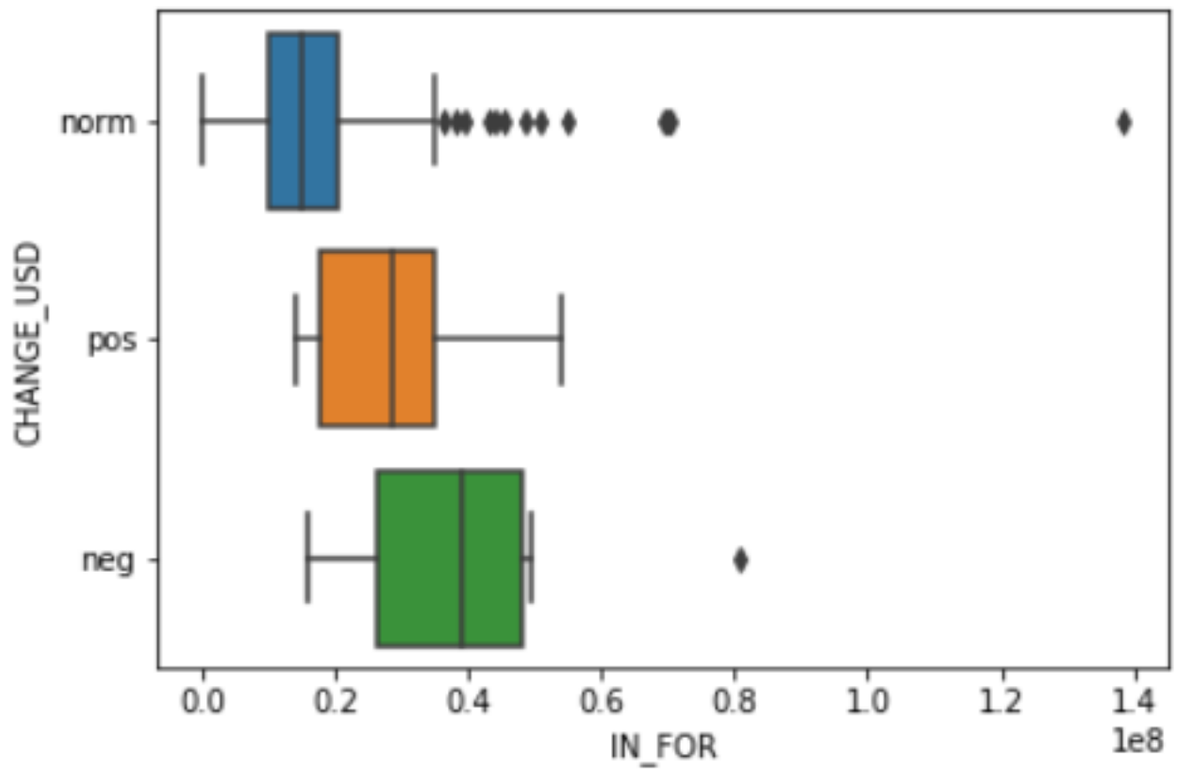
If daily percentage change:

- x < -1%, then negative
- -1 <= x <= 1, then normal
- x > 1, positive

Classification

```
In [25]: sns.boxplot(x='IN_FOR',y='CHANGE_USD', data=df_2017, orient='h')
```

```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1167bc1d0>
```



Lag + 2 days

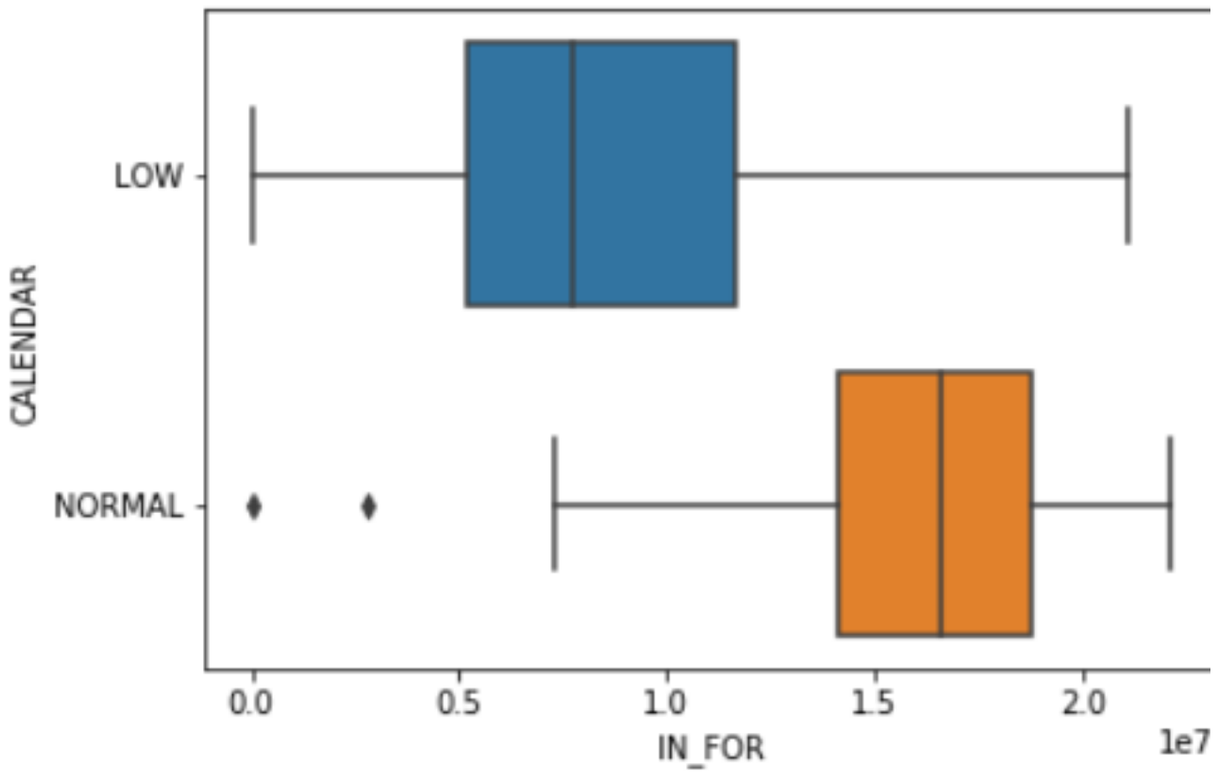
```
def change_currency_eur(df):
    change = False
    cnt = 0

    for i, val in enumerate(df['CHANGE_EUR']):
        if val == 'neg' or val == 'pos':
            unknown = val
            change = True
            cnt = 0
            #print(i, val)
        if val == 'norm' and change == True and df['CALENDAR'].iloc[i] == 'NORMAL' and cnt !=2:
            cnt+=1
            df['CHANGE_EUR'].iloc[i] = unknown
        if cnt == 2:
            change = False

    return df['CHANGE_EUR']
```

Real predictors

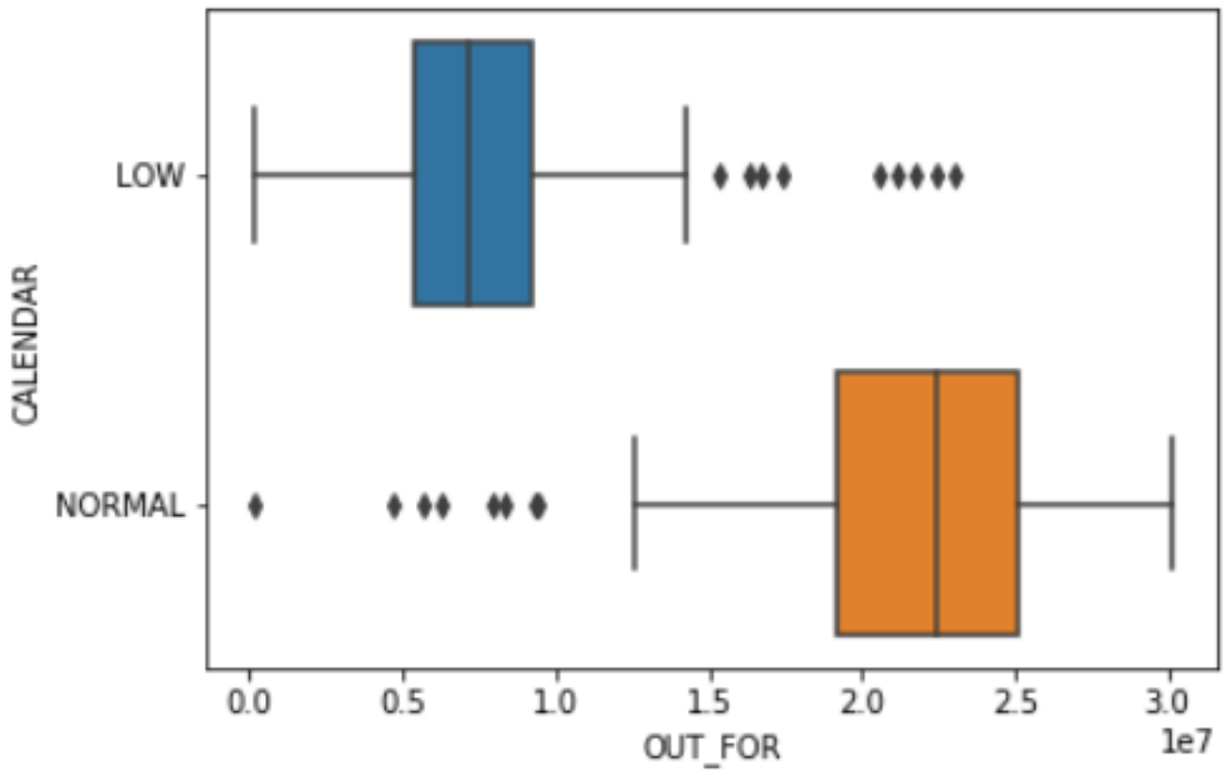
Predictor #2, low activity:

- Clients do not make as many changes during weekend and holidays

```
]: sns.boxplot(x='IN_FOR',y='CALENDAR', data=df_xls[df
]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1eb5f82
```




```
sns.boxplot(x='OUT_FOR',y='CALENDAR', data=df_xls[df
<matplotlib.axes._subplots.AxesSubplot at 0x1a1ed3f7f0
```



Linear model

Real targets, 4 models

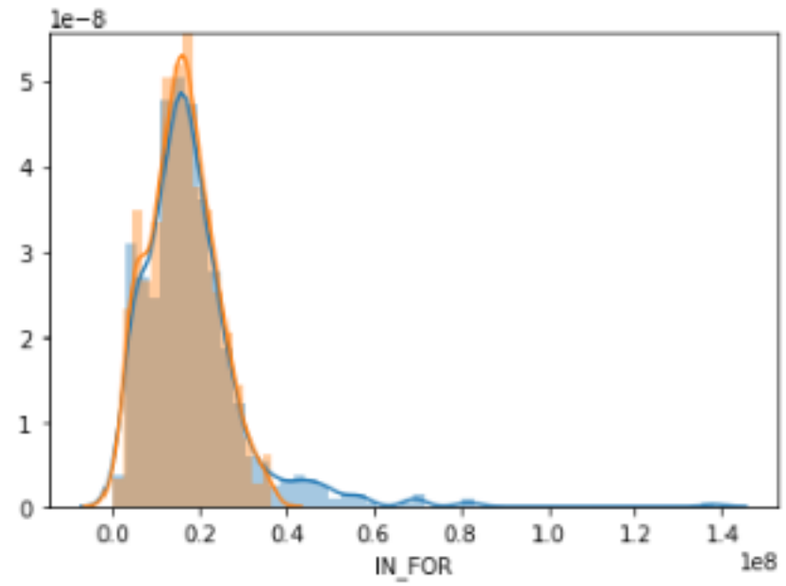
	IN_FOR	OUT_FOR	IN_KZT	OUT_KZT
0	0	122953	1899647	1584705
1	2942	115059	2706416	2315761
2	263	147905	3548855	2971268
3	13533594	9363025	23528654	11716898
4	36769881	32769452	37289979	22565273

Features

	0	1	2	3	6
IN_FOR	0	2942	263	13533594	20162470
CALENDAR_LOW	1	1	0	0	1
CALENDAR_NORMAL	0	0	1	1	0
USD_EUR_CHANGE_neg	0	0	0	0	0
USD_EUR_CHANGE_norm	1	1	1	1	1
USD_EUR_CHANGE_pos	0	0	0	0	0

Outliers

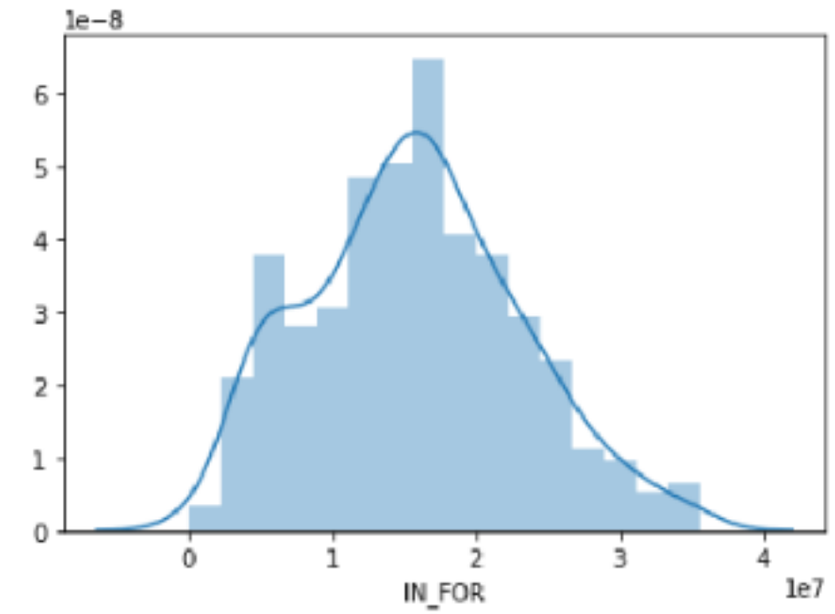
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 637 entries, 0 to 679
Data columns (total 4 columns):
IN_FOR                637 non-null int64
USD_EUR_CHANGE_neg    637 non-null uint8
USD_EUR_CHANGE_norm   637 non-null uint8
USD_EUR_CHANGE_pos    637 non-null uint8
dtypes: int64(1), uint8(3)
memory usage: 11.8 KB
9373426 14286848.0
5452144 14286848.0
16094583 14286848.0
16414107 14286848.0
18947640 14286848.0
```



==>

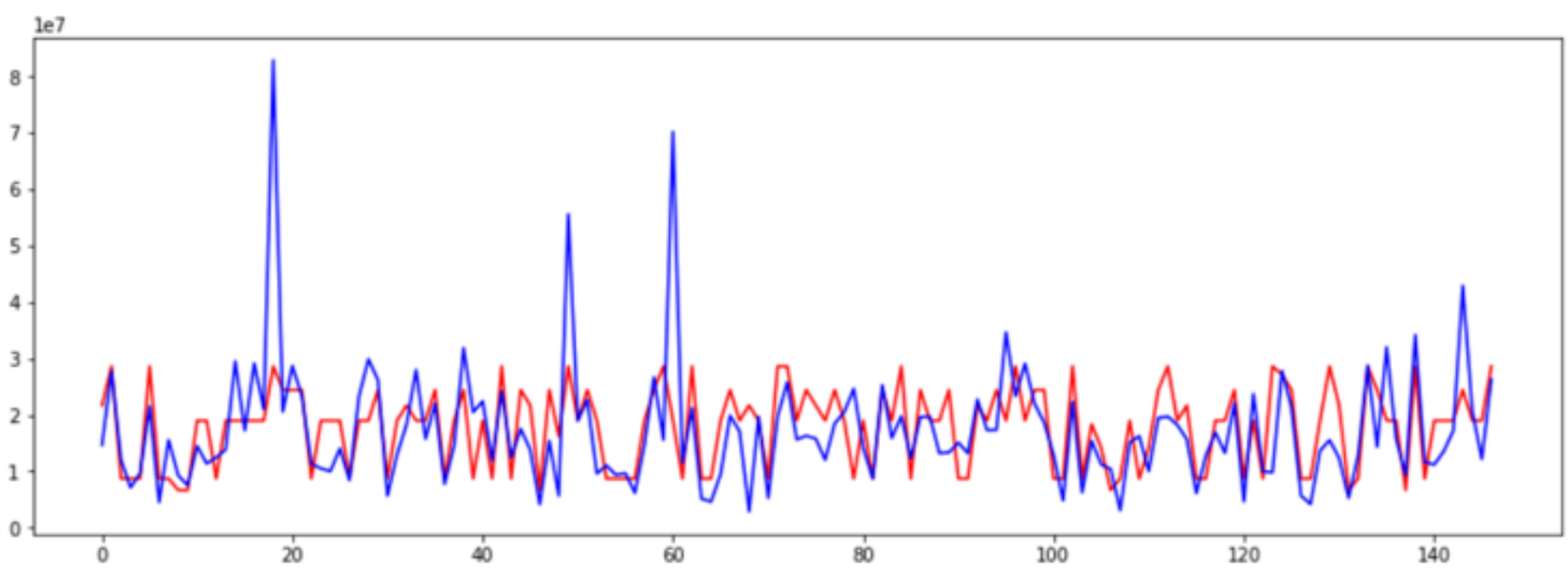
```
xl = df1['IN_FOR'].std()*3

df1[df1['IN_FOR'] > xl].count()

df1 = df1[df1['IN_FOR'] < xl]

sns.distplot(df1['IN_FOR'])
```

```
/Library/Frameworks/Python.framework/Versions
ning: The 'normed' kwarg is deprecated, and
  warnings.warn("The 'normed' kwarg is depre
<matplotlib.axes._subplots.AxesSubplot at 0x
```



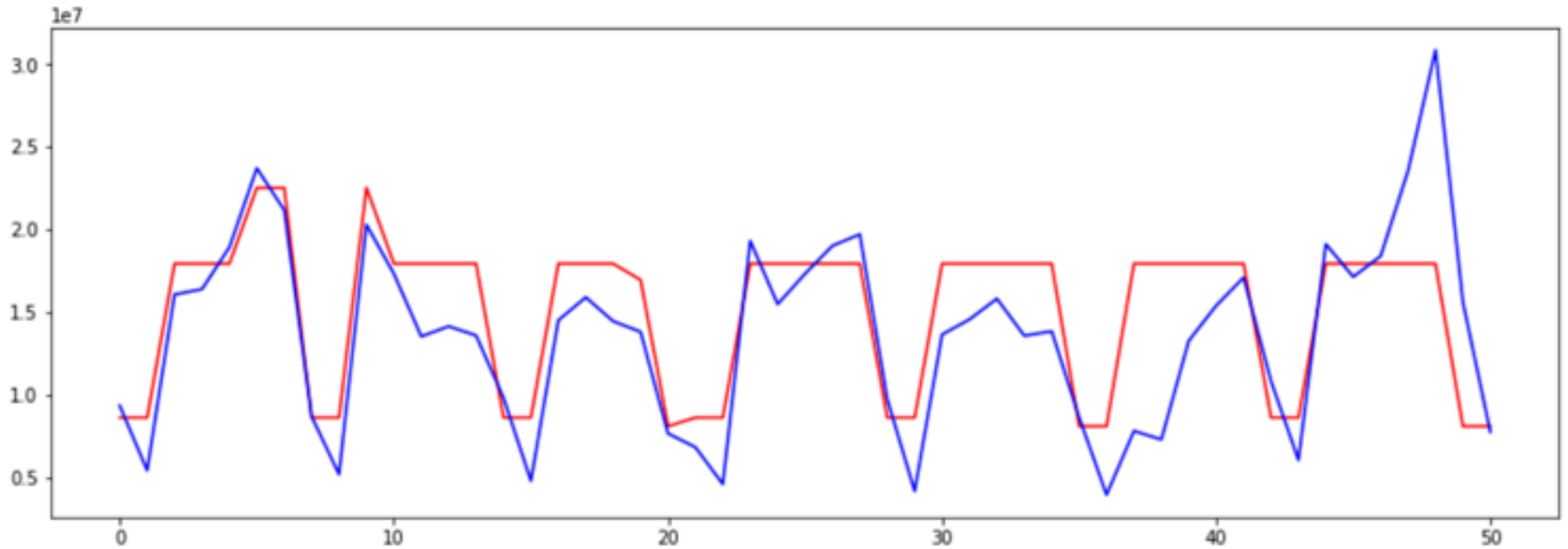
Without removing outliers

```
plt.figure(figsize=(15, 5))
plt.plot(prediction, color = 'red')
plt.plot(y_test, color = 'blue')
plt.show()
```



With outliers removed model.predict

```
plt.figure(figsize=(15, 5))
plt.plot(prediction, color = 'red')
plt.plot(y_test, color = 'blue')
plt.show()
```



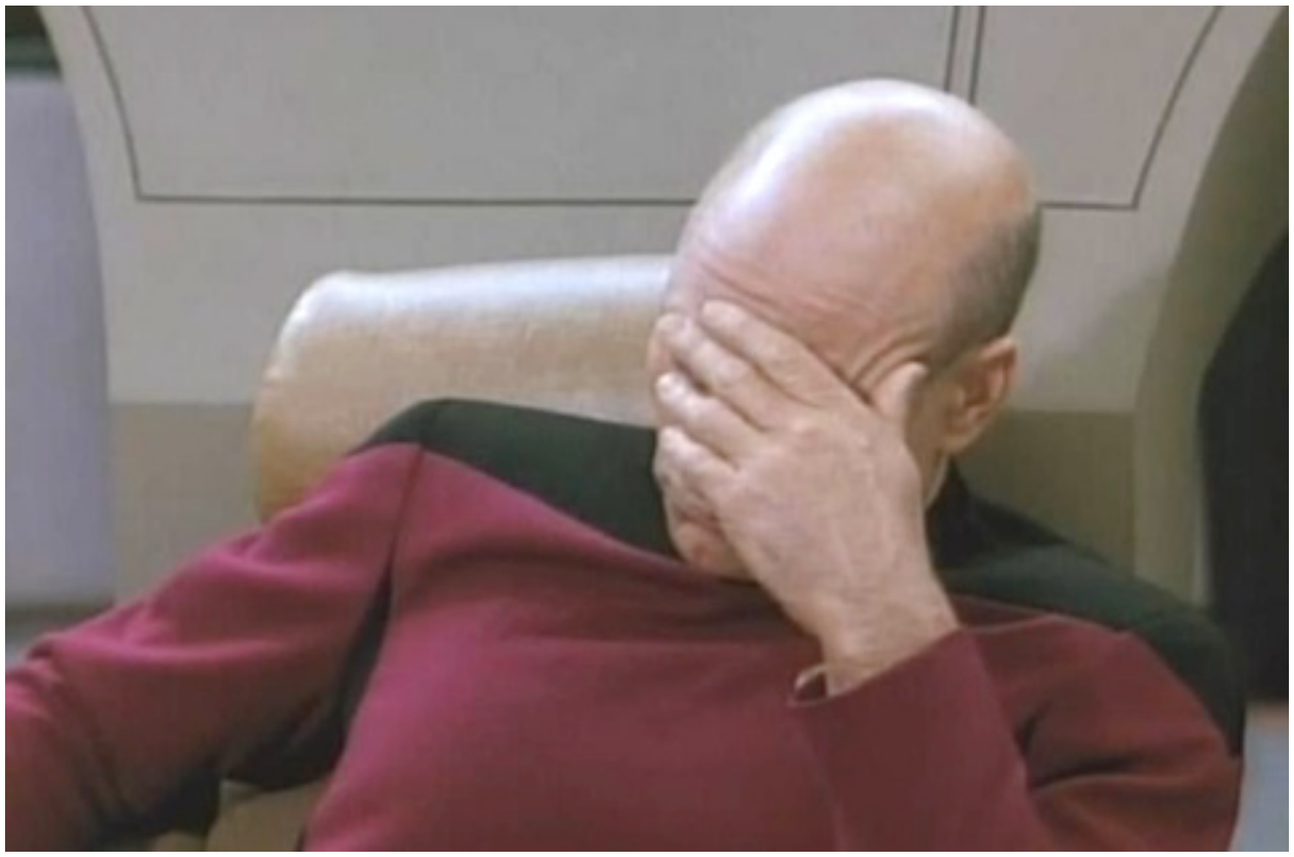

```
print("RMSE: {}".format(np.sqrt(mean_squared_error(y_test, prediction))))
```

3954580.452

Spikes

DATE	IN_FOR	OUT_FOR	NET_FOR	CALENDAR	CHANGE_USD	CHANGE_EUR
8/2/2017	138,256,883	21,073,823	117,183,060	NORMAL	norm	neg
1/25/2016	82,908,779	99,732,352	(16,823,573)	NORMAL	norm	norm
8/3/2017	80,764,407	32,761,530	48,002,877	NORMAL	neg	neg
9/6/2017	70,238,676	31,565,869	38,672,807	NORMAL	norm	norm
9/5/2017	69,806,112	34,627,545	35,178,567	NORMAL	norm	norm
8/1/2017	69,271,648	29,751,643	39,520,006	SALARY	norm	norm
1/29/2016	68,425,027	117,637,795	(49,212,768)	NORMAL	pos	norm
7/26/2016	58,348,233	24,073,709	34,274,524	NORMAL	norm	norm

2018 forecast



Exchange rate 2018

In [349]:
```
df_test1.groupby(by='CHANGE_USD').count()
```

Out[349]:

CHANGE_USD	DATE	CALENDAR_LOW	CALENDAR_NORMAL	CHANGE_EUR
neg	2	2	2	2
norm	179	179	179	179

In [350]:
```
df_test1.groupby(by='CHANGE_EUR').count()
```

Out[350]:

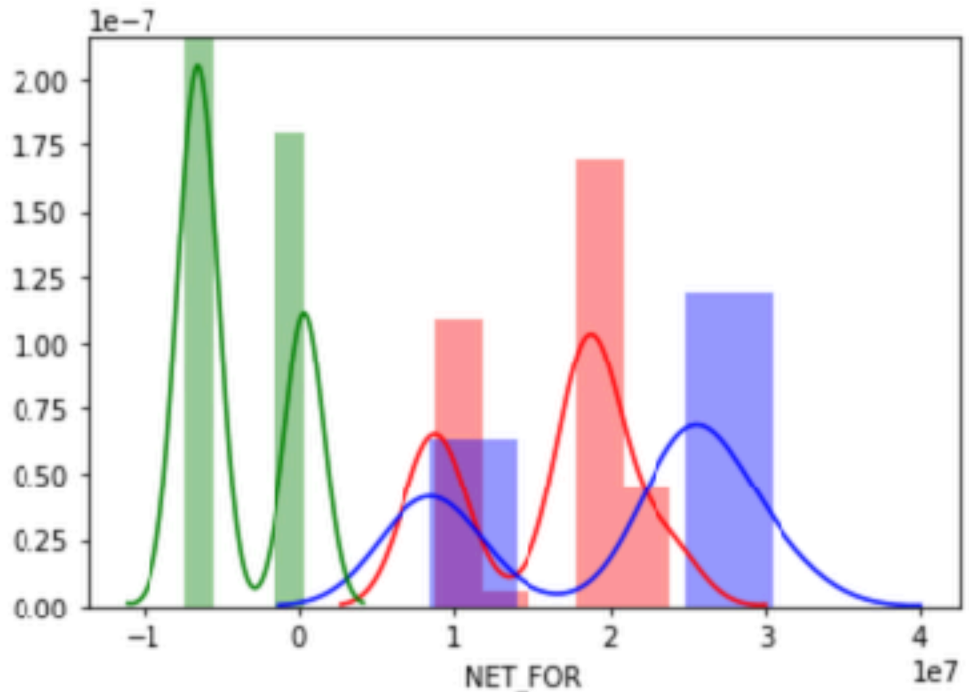
CHANGE_EUR	DATE	CALENDAR_LOW	CALENDAR_NORMAL	CHANGE_USD
neg	7	7	7	7
norm	170	170	170	170
pos	4	4	4	4

Predict model graph

```
sns.distplot(df_pred['in_for'],color='red')
sns.distplot(df_pred['out_for'],color='blue')
sns.distplot(df_out['NET_FOR'],color='green')
```

```
/Users/Daniyar/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462: UserWarning: T
he 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/Users/Daniyar/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462: UserWarning: T
he 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
/Users/Daniyar/anaconda3/lib/python3.6/site-packages/matplotlib/axes/_axes.py:6462: UserWarning: T
he 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d7ec080>
```



Take-aways!

Result is nothing, but lessons matter:

1. This was not a time series problem, perform Dickey-Fuller test first to check stationarity.

2. This was a essentially a behavioral modeling problem:

- a) which needs a sound understanding what governs the behavior (we were able to capture a small portion of it)

- b) clients can be not reasonable, they do not have access to all information and at the same time, and are not able to efficiently process it. Hence, extremely hard to predict

Thank you!