Формирование общего датасета наблюдений пампов и дампов

In [1]:
```
import numpy as np
import pandas as pd
from tqdm import tqdm
import json
import os
import glob
import time
import datetime
from datetime import datetime
import matplotlib.pyplot as plt
```

In [2]:
```
# Импортируем сформированный список всех тикеров по которым у нас есть данные:
with open("tickers_list_total.txt", "r") as fp:
    tickers_list_total = json.load(fp)
all_tickers = tickers_list_total[0]
len(all_tickers)
```

Out[4]: 8248

Рассмотрим распределение принтов, что может подсказать какое оптимальное кол-во принтов нам взять

In [3]:
```
all_liquid_obs = []
for i in tqdm(range(len(all_tickers))):
    ticker_name = all_tickers[i]
    parent_dir_one_min = 'C:/Users/Kuanysh/Downloads/pump_and_dump/agg_tickers_1m'
    file_one_min = os.path.join(parent_dir_one_min, ticker_name + '.csv')
    df_one_min = pd.read_csv(file_one_min)
    df_one_min['date'] = pd.to_datetime(df_one_min['time']).dt.date
    df_one_min['time'] = pd.to_datetime(df_one_min['time']).dt.time
    df_one_min['date'] = pd.to_datetime(df_one_min['date'])
    counts = df_one_min.groupby(['date']).count()[['close']].reset_index()
    counts = counts[(counts['date'] >= '2019-09-03') & (counts['date'] <= '2021-08-04')]
    counts_date = counts.date.astype(str)
    all_counts = counts.values.tolist()
    all_liquid_obs.append([ticker_name, all_counts])
```

In [ ]:
```
#with open('all_liquid_obs.txt', 'w') as fp:
#    json.dump(all_liquid_obs, fp)
```

In [5]:
```
with open("all_liquid_obs.txt", "r") as fp:
    all_liquid_obs = json.load(fp)
```

In [19]:
```
# рассмотрим распределение наблюдений по кол-ву принтов:
all_sessions = []
for ticker in all_liquid_obs:
    for day in ticker[1]:
        all_sessions.append(day[1])
n, bins, patches = plt.hist(all_sessions, bins=1000, log=True)
plt.show()
```

In [20]:
```
# действительно 390 принтов это кол-во минут в основную сессию:
opt_prints = bins[np.where(n == n[200:].max())[0]]
opt_prints
```

Out[20]: 391.71999999999997

In [9]:
```
# найдем даты в которых были какие-либо сессии
all_liquid_sessions = []
for ticker in all_liquid_obs:
    liquid_day = []
    for day in ticker[1]:
        if day[1] >= 1:
            liquid_day.append(day[0])
    all_liquid_sessions.append([ticker[0], liquid_day])
```

Рассмотрим распределение доходностей, чтобы понять от каких минимальных доходностей подразумевать пампы или дампы

In [ ]:
```
def get_all_pnd(all_liquid_sessions):
    all_pnd = []
    for i in tqdm(range(len(all_liquid_sessions))):
        ticker_name = all_liquid_sessions[i][0]
        days = all_liquid_sessions[i][1]
        parent_dir_min = 'C:/Users/Kuanysh/Downloads/pump_and_dump/agg_tickers_15m'
        file_min = os.path.join(parent_dir_min, ticker_name + '.csv')
        df_min = pd.read_csv(file_min)
        df_min['date'] = pd.to_datetime(df_min['time']).dt.date
        #df_min['time'] = pd.to_datetime(df_min['time']).dt.time
        df_min['date'] = pd.to_datetime(df_min['date'])

        per_ticker_list = []
        for day in days:
            mask = df_min[df_min['date'] == day]
            min_ser = mask.to_numpy()
            if min_ser.size > 0:
                # найдем макс и мин по всем обіс ценам
                max_price = np.max(min_ser[:,1:5])
                min_price = np.min(min_ser[:,1:5])

                # дневной диапазон цен и доходность day_ret, именно по нему отличаются пампы и дампы от просто
                day_range = max_price - min_price
                day_ret = max_price/min_price - 1

                # найдем max_coverage, покрытие дневного роста цен,
                # чем ближе к 100%, тем сильнее вероятность что это цена или несколько больших свеч, то есть д
                if day_range > 0:
                    all_coverage = []
                    hl = min_ser[:,2:4].astype(np.float32)
                    for i in range(hl.shape[0]):
                        perf_min = hl[i][0] - hl[i][1]
                        coverage = perf_min/day_range
                        all_coverage.append(coverage)
                    max_coverage = max(all_coverage)
                    per_ticker_list.append([day, day_ret, max_coverage])
        all_pnd.append([ticker_name, per_ticker_list])
    return all_pnd
```

In [ ]:
```
all_pnd = get_all_pnd(all_liquid_sessions)
```

In [ ]:
```
#with open('all_pnd3.txt', 'w') as fp:
#    json.dump(all_pnd, fp)
```

In [11]:
```
with open("all_pnd3.txt", "r") as fp:
    all_pnd = json.load(fp)
```

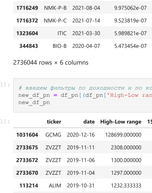
In [12]:
```
# переведем в формат: тикер, дата, дневной рост, покрытие диапазона роста
all_pnd_sessions = []
for ticker in all_pnd:
    for day in ticker[1]:
        all_pnd_sessions.append([ticker[0], day[0], day[1], day[2]])
```

In [13]:
```
all_pnd_sessions[:2]
```

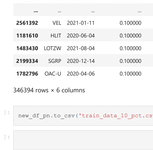
Out[13]:
```
[['A', '2019-09-03', 0.022437788748092935, 0.9069877071515293],
 ['A', '2019-09-04', 0.018010291595197403, 0.5238063751705088]]
```

In [14]:
```
def plot_loglist(x, bins):
    hist, bins = np.histogram(x, bins=bins)
    logbins = np.logspace(np.log10(bins[0]),np.log10(bins[-1]),len(bins))
    plt.hist(x, bins=logbins, log=True)
    plt.xscale('log')
```

In [28]:
```
# рассмотрим график диапазона цен:
all_pnd_sessions_df = pd.DataFrame(all_pnd_sessions, columns =['ticker','date','High-Low range', '15m candle co
plot_loglist(all_pnd_sessions_df['High-Low range'], 1000)
```



In [29]:
```
# рассмотрим график покрытия дневного диапазона:
plt.hist(all_pnd_sessions_df['15m candle coverage'], bins=1000, log=True);
```



In [30]:
```
# найдем эту самую популярную доходность:
hist, bins = np.histogram(all_pnd_sessions_df['High-Low range'], bins=1000)
logbins = np.logspace(np.log10(bins[0]),np.log10(bins[-1]),len(bins))
hist, new_bins = np.histogram(all_pnd_sessions_df['High-Low range'], bins=logbins)
new_bins[np.where(hist == hist.max())[0]]
```

Out[30]: array([0.04135757])

то есть можем начинать от дневного диапазона в 4%

Рассмотрим также и общее позитивное или негативное движение внутридневных цен, ведь предполагается, что большая часть движений пампов и дампов происходят в основном выше дневной цены открытия.

In [ ]:
```
def get_all_spikes(all_spikes):
    for i in tqdm(range(len(all_spikes))):
        ticker_name = all_spikes[i][0]
        days = [x[0] for x in all_spikes[i][1]]
        parent_dir_min = 'C:/Users/Kuanysh/Downloads/pump_and_dump/agg_tickers_15m'
        file_min = os.path.join(parent_dir_min, ticker_name + '.csv')
        df_min = pd.read_csv(file_min)
        df_min['date'] = pd.to_datetime(df_min['time']).dt.date
        df_min['time'] = pd.to_datetime(df_min['time']).dt.time
        df_min['date'] = pd.to_datetime(df_min['date'])
        df_min = df_min.sort_values(by='times', ascending=True)

        for j in range(len(days)):
            mask = df_min[df_min['date'] == days[j]]
            min_ser = mask.to_numpy()
            first_open = min_ser[0][1]
            close_sums = 0
            for k in range(len(min_ser)):
                close = min_ser[k][4]
                if close > first_open:
                    close_sums+=1
            pos_neg_coverage = close_sums/len(min_ser)
            all_spikes[i][1][j].extend([pos_neg_coverage, len(min_ser)])
    return all_spikes
```

In [ ]:
```
with open("all_pnd3.txt", "r") as fp:
    all_pnd = json.load(fp)
```

In [ ]:
```
all_spikes = get_all_spikes(all_pnd)
```

In [ ]:
```
#with open('all_spikes_1m.txt', 'w') as fp:
#    json.dump(all_spikes, fp)
```

In [15]:
```
with open("all_spikes_1m.txt", "r") as fp:
    all_spikes = json.load(fp)
```

In [16]:
```
# добавим еще описательных признаков: положительное или негативное движение и кол-во принтов
all_spikes_sessions = []
for ticker in all_spikes:
    for day in ticker[1]:
        all_spikes_sessions.append([ticker[0], day[0], day[1], day[2], day[3], day[4]])
```

Сформируем весь датасет:

In [18]:
```
df_pn = pd.DataFrame(all_spikes_sessions, columns =['ticker','date','High-Low range', '15m candle coverage', '
df_pn = df_pn.sort_values(by=['High-Low range', '15m candle coverage','pos_neg_coverage','number of prints'], a
df_pn
```

Out[18]:

	ticker	date	High-Low range	15m candle coverage	pos_neg_coverage	number of prints
1031604	GCMG	2020-12-16	1.2869904×05	0.943279	0.037037	27
2687364	XBIOW	2019-11-08	8.1380004×03	0.458098	1.000000	2
2415051	THWWW	2020-03-25	3.9990000×03	0.742686	0.000000	2
2733675	ZVZZT	2019-11-11	2.3080000×03	0.562392	0.075000	40
2733672	ZVZZT	2019-11-06	1.3000000×03	0.614615	0.000000	41
...	...	...	...	...	...	...
1716216	NMK-P-B	2021-01-29	1.0000000×06	0.991821	1.000000	1
1716249	NMK-P-B	2021-08-04	9.9750562×07	0.991821	0.000000	1
1716372	NMK-P-C	2021-07-14	9.5238119×07	0.991821	0.000000	1
1323604	ITIC	2021-03-30	5.9898216×07	0.915527	1.000000	1
344843	BIO-B	2020-04-07	5.4734546×07	1.068115	0.000000	1

2730444 rows × 6 columns

In [19]:
```
# введем фильтры по доходности и по кол-ву принтов (10 * 15мин = 150 принтов) баров:
new_df_pn = df_pn[(df_pn['High-Low range']>0.1) & (df_pn['number of prints']>10)].sort_values(by=['High-Low ra
new_df_pn
```

Out[19]:

	ticker	date	High-Low range	15m candle coverage	pos_neg_coverage	number of prints
1031604	GCMG	2020-12-16	128699.000000	0.943279	0.037037	27
2733675	ZVZZT	2019-11-11	2308.000000	0.562392	0.075000	40
2733672	ZVZZT	2019-11-06	1300.000000	0.614615	0.000000	41
2733670	ZVZZT	2019-11-04	1297.000000	0.155744	0.386364	44
113214	ALIM	2019-10-31	1232.333333	0.970246	0.000000	31
...	...	...	...	...	...	...
2561392	VEL	2021-01-11	0.100000	0.142857	0.117647	17
1181610	HLIT	2020-06-04	0.100000	0.140000	0.928571	28
1483430	LOTZW	2021-08-04	0.100000	0.130833	1.000000	14
2199334	SGRP	2020-12-14	0.100000	0.103000	0.000000	22
1782796	OAC-U	2020-04-06	0.100000	0.090000	0.000000	11

346394 rows × 6 columns

In [ ]:
```
new_df_pn.to_csv('train_data_10_pot.csv')
```

In [ ]:
```

```