

# بسم الله الرحمن الرحيم

پروژه صفر درس مبانی علوم داده  
دکتر بهرک و دکتر یعقوب زاده

محمد امین توانایی - ۸۱۰۱۰۱۳۹۶

سید علی تهامی - ۸۱۰۱۰۱۳۹۷

مهدی وجهی - ۸۱۰۱۰۱۵۵۸

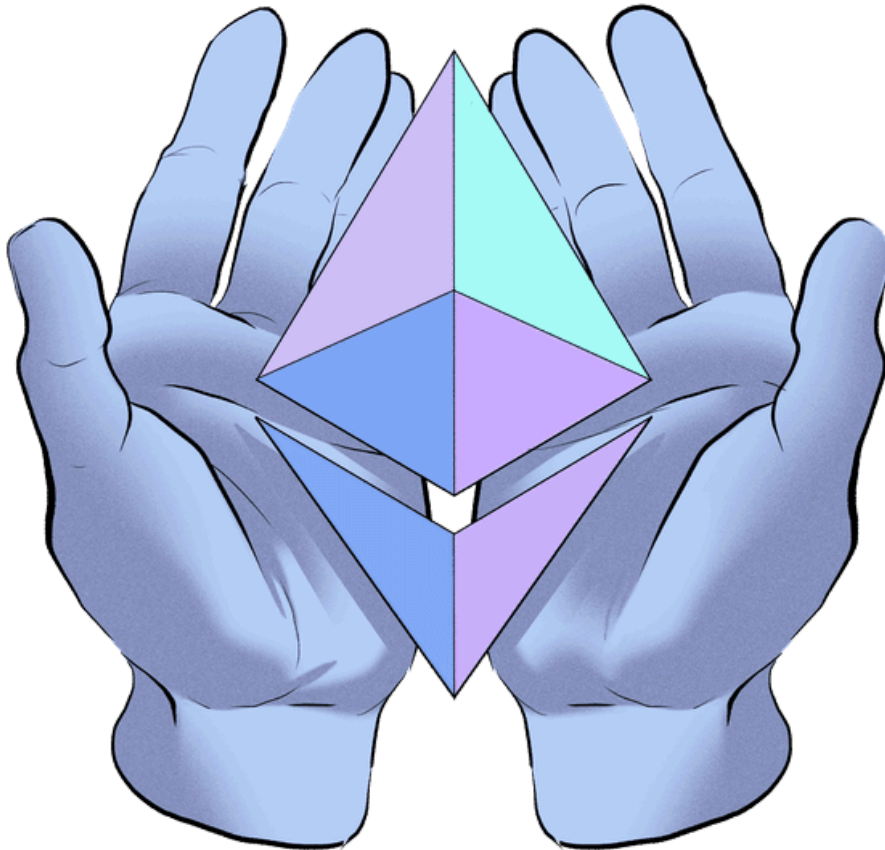
## فهرست

4.....	مقدمه
5.....	آماده سازی محیط
6.....	جمع آوری اطلاعات
6.....	راه اندازی
9.....	حل مشکل دوباره خوانده شدن یک تراکنش
9.....	طراحی استخراج کننده
10.....	تابع scrape_data
10.....	تابع open_page
10.....	تابع find_first_block
11.....	تابع read_blocks
11.....	تابع write_in_json_file
11.....	تابع parse_transaction
12.....	تابع collect_block_data
12.....	تابع convert_row_to_dict
13.....	تابع get_page_count
14.....	آنالیز داده ها
14.....	بارگذاری داده ها
14.....	پاکسازی داده ها
15.....	آنالیز آماری
16.....	بررسی بیشتر تبادل ۵۰۰ اتری
17.....	بصری سازی داده ها
17.....	داده های خام
18.....	لگاریتم داده ها
20.....	لگاریتم داده ها با حذف صفر ها
22.....	نمونه گیری
22.....	پراکندگی داده
22.....	نمونه گیری تصادفی
24.....	نمونه گیری طبقه ای
24.....	طبقه بندی بر اساس مقدار
26.....	طبقه بندی بر اساس method
27.....	طبقه بندی بر اساس block
29.....	پاسخ به سوالات

- سوال ۱: مشکلات استخراج کننده و محدودیت های سایت ..... 29
- سوال ۲: غیر قابل اعتماد شدن تحلیل ..... 30
- سوال ۳: مصور سازی داده ها ..... 30
- سوال ۴: مقایسه دو روش نمونه گیری ..... 31
- مراجع ..... 32

## مقدمه

با توجه به گسترش روزافزون استفاده از رمز ارز ها از جمله اتریوم تحلیل و زیر نظر گرفتن تبادلات آن می تواند از جنبه های مختلفی کاربرد داشته باشد. در این پروژه با استفاده از ابزار جمع آوری اطلاعات ابتدا اطلاعات را از سایت [etherscan.io](https://etherscan.io) جمع آوری می کنیم و سپس با کمک باقی ابزار ها داده را تمیز می کنیم و آن را تحلیل می کنیم.



## آماده سازی محیط

در این قسمت باید کتابخانه ها و ابزار مورد نیاز را نصب کنیم. برای برنامه یک فایل `requirements.txt` می نویسیم که کتابخانه ها در آن نوشته شده:

```
bs4
selenium
pandas
numpy
matplotlib
seaborn
plotly
scipy
```

سپس می توانیم با دستور زیر کتابخانه ها را نصب کنیم:

```
pip install -r requirements.txt
```

حال باید یک web driver نصب کنیم که از پیوند مربوطه از گیت هاب نسخه مناسب مرورگر خودمان را دانلود می کنیم.

# جمع آوری اطلاعات

## راه اندازی

ابتدا ابزار خود را با کد زیر امتحان می کنیم که درست نصب شده باشند.

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from bs4 import BeautifulSoup

driver = webdriver.Firefox()
driver.get("https://etherscan.io/txs")
```

که در نتیجه ی آن مرورگر باز می شود:

The screenshot shows the Etherscan website's Transactions page. At the top, there's a navigation bar with links to Home, Blockchain, Tokens, NFTs, Resources, Developers, and a Sign In button. Below the navigation bar, the 'Transactions' section is active. It features a summary of transaction statistics for the last 24 hours: 1,182,830 transactions (2.26% increase), 165,667 pending transactions (average), a network transaction fee of 524.08 ETH (1.57%), and an average transaction fee of 17.60 USD (9.56%). Below this, a table lists individual transactions. The table has columns for Txn Hash, Method, Block, Age, From, To, Value, and Txn Fee. The first transaction is a transfer of 0.79976 ETH from 'Orbiter Finance: Bridge' to '0x29632326...F5BB225d5'. The page also includes a cookie consent banner at the bottom.

حال ساختار صفحه را بررسی می کنیم:

The screenshot displays the raw HTML code of the Etherscan Transactions page. The code is shown in a dark-themed editor. It starts with the body tag, which has a class 'align-middle text-nowrap'. The code includes various HTML elements like divs, tables, and spans, which correspond to the visual elements seen in the previous screenshot. The table structure is clearly visible, with columns for transaction details. The code also includes some JavaScript snippets and CSS classes for styling and functionality.

همانطور که در تصویر می بینیم داده های مورد نظر ردیف های یک جدول هستند ( تگ های tr یک تگ tbody و هر خانه آن (تگ td) یک داده هستند. با کتابخانه bs4 صفحه را تجزیه می کنیم و قسمت مربوطه را پیدا می کنیم.

```
the_soup = BeautifulSoup(driver.page_source, 'html.parser')
t = the_soup.find("tbody", attrs={"class": "align-middle text-nowrap"})
```

حال باید تگ های tr را از آن استخراج کنیم و نتیجه را تا به اینجا بررسی کنیم:

```
row = [i for i in t.children]
str(row[1].text)
```

```
'\n\n\n\n\n\n\n0x0fba2da1cd9b331ccda63a48b0cd1ee4de7d0424f5c4cf26cf807f20c  
5a5f909\n\n\n\nTransfer\n\n19361335\n\n2024-03-04 10:55:2319 secs  
ago1709549723\n\n\n\nOrbiter Finance: Bridge\n\n\n\n\n\n\n\n\n0x29632326...F5BB225d5\n \n\n\n0.79976  
ETH\n0.0013637464.94033407\n'
```

حال هر ردیف را به td می شکنیم:

```
for i in range(1, 10):
    print(i, row[i].findAll("td")[2].text)
```

- 1 Transfer
- 2 Claim
- 3 Transfer
- 4 Transfer
- 5 Transfer
- 6 Transfer
- 7 Transfer
- 8 Transfer
- 9 Transfer

حال باید محتوای هر کدام از خانه ها را بررسی کنیم:

```
for i in range(len(row[1].findAll("td"))):
    print(i, repr(str(row[1].findAll("td")[i].text)))
```

```
0 '\n\n'
1
2 '\n\n0x0fba2da1cd9b331ccda63a48b0cd1ee4de7d0424f5c4cf26cf807f20c5a5f909\n\n'
3 'Transfer'
4 '19361335'
```

```

4 '2024-03-04 10:55:23'
5 '19 secs ago'
6 '1709549723'
7 '\n\nOrbiter Finance: Bridge\n \n'
8 '\n\n'
9 '\n\n0x29632326...F5BB225d5\n \n'
10 '0.79976 ETH'
11 '0.00136374'
12 '64.94033407'

```

مقادیر با همان ترتیب شماره ها داری مقدار های زیر هستند:

شماره	محتوا
۰	-
۱	خطا ها و Txn Hash
۲	(بعضی نام ها به طور کامل نیست) Method
۳	Block
۴	زمان به صورت ساعت - تاریخ
۵	زمان گذشته تا به حال
۶	زمان به ثانیه
۷	From
۸	اگر تبادل از یک فرد به خودش باشد مقدار SELF می گیرد
۹	To
۱۰	Value
۱۱	Txn Fee
۱۲	GasPrice

مقدار های هر کدام از این موارد در یکی از زیر مجموعه های تگ td است مثلا برای Method در یک text یک span ذخیره شده پس با کد زیر خوانده می شود:

```
td.find("span").text
```



## حل مشکل دوباره خوانده شدن یک تراکنش

با بررسی سایت متوجه می شویم که بلوک های جدید به سرعت در حال تولید هستند و طول هر یک ممکن است به بیش از ۸ صفحه نیز برسد بنابراین اگر ما بخواهیم داده را صفحه به صفحه بخوانیم با این مشکل مواجه می شویم که وقتی داده های چند صفحه را خواندیم با تولید یک بلوک جدید تراکنش ها چندین صفحه به عقب بروند و داده ها مجددا خوانده شوند یا حتی به دلیل بالا بودن سرعت تولید بلوک ممکن است اصلا به آخرین بلوک نرسیم. حل این مشکل پیچیده نیست، با بررسی سایت متوجه می شویم می شود تراکنش های فقط یک بلوک را در صفحه ای با پیوندی به شکل زیر مشاهده کرد:

`https://etherscan.io/txs?block={block}&p={page_number}`

Txn Hash	Method	Block	Age	From	To	Value	Txn Fee
0x0fba2da1cd9...	Transfer	19361335	55 mins ago	Orbiter Finance: Bridge	0x29632326...F5BB225d5	0.79976 ETH	0.00136374
0x2e835ea9db...	Claim	19361335	55 mins ago	0xCb7D1f23...E26bfa51b	0xC780e9c4...57E4b0026	0 ETH	0.00469953
0xd5727a35b6...	Transfer	19361335	55 mins ago	0x5A1faEcA...E827e5bA7	Coinbase 10	0.08355136 ETH	0.00136731
0x32993b924cf...	Transfer	19361335	55 mins ago	0xAEce00BA...86550EF35	Coinbase 10	0.0761341 ETH	0.00136731
0x6d03feb017b...	Transfer	19361335	55 mins ago	0xaa9A93a2...D2A912F76	Coinbase 10	0.077249554 ETH	0.00136731
0x12a2a6672bf...	Transfer	19361335	55 mins ago	0xCD4214fd...F54212589	Coinbase 10	0.031315193 ETH	0.00136731
0x13847b7cc0...	Transfer	19361335	55 mins ago	0x833D2C2D...3d776cC60	Coinbase 13	0.053386766 ETH	0.00136374
0xf90cf8e52c8...	Transfer	19361335	55 mins ago	0x93d6E014...2Ce3AD381	Coinbase 13	0.025189072 ETH	0.00136374

راه حل بالا ایراد گفته شده را برطرف می کند.

## طراحی استخراج کننده

حال که با ساختار سایت آشنا شدیم باید کدی بنویسیم که اطلاعات ۱۰ بلوک آخر را استخراج کند. برای این کار لازم است برنامه روند زیر را داشته باشد:

1. باز کردن سایت
2. خواندن عدد آخرین بلوک
3. خواندن بلوک ها
  - a. ساخت پیوند مرتبط با هر بلوک
  - b. ورود به صفحه اول بلوک و خواندن تعداد صفحات برای ساخت پیوند مناسب

c. خواندن و ذخیره کردن هر صفحه

- i. تجزیه جدول تبدلات به ردیف ها
  - ii. تبدیل داده html به قالب مناسب
  - iii. ذخیره در اطلاعات آن بلوک
  - iv. رفتن به صفحه بعدی
4. ذخیره داده ها در فایل با قالب مناسب

## تابع scrape\_data

این تابع، تابع اصلی برنامه است که به صورت زیر است:

```
def scrape_data():
    driver = open_page()
    first_block = find_first_block(driver)
    data = read_blocks(driver, first_block)
    write_in_json_file(data)
```

همانطور که می بینید گام های زیر برای استخراج طی می شود:

1. باز کردن صفحه ([open\\_page](#))
2. پیدا کردن اولین بلوک ([find\\_first\\_block](#))
3. خواندن بلوک ها ([read\\_blocks](#))
4. ثبت اطلاعات در فایل ([write\\_in\\_json\\_file](#))

## تابع open\_page

این تابع مسئولیت باز کردن مرورگر و سپس سایت را دارد و در آخر هم driver سلنیوم را برای کار های بعدی بر می گرداند.

```
def open_page():
    driver = webdriver.Firefox()
    driver.get("https://etherscan.io/txs")
    return driver
```

## تابع find\_first\_block

این تابع تراکنش های صفحه اول را می خواند ([parse\\_transaction](#)) و شماره بلوک آخرین تراکنش را بر می گرداند.

```
def find_first_block(driver) -> int:
    the_soup = BeautifulSoup(driver.page_source, 'html.parser')
    return int(parse_transaction(the_soup)[1]["Block"])
```

## تابع read\_blocks

این تابع برای ۱۰ بلاک آخر، تابع [collect\\_block\\_data](#) را صدا می زند و تراکنش های آنها را ذخیره می کنید. نکته قابل توجه این است که برای این که سایت ما را مسدود نکند و فکر نکند ربات هستیم یک ثانیه بین آنها تاخیر می گذاریم.

```
def read_blocks(driver, first_block) -> list[dict]:
    output = list()
    for i in range(BLOCK_COUNT):
        output.extend(collect_block_data(driver, first_block - i))
        print(f"sample block {first_block - i}: {output[-1]}\n")
        time.sleep(1)
    return output
```

## تابع write\_in\_json\_file

این تابع نکته خاصی ندارد و صرفاً خروجی را در یک فایل json ذخیره می کند.

```
def write_in_json_file(data_list):
    json_data = json.dumps(data_list, indent=4)

    file_path =
f"data_block_{data_list[0]['Block']}_{data_list[-1]['Block']}.json"
    with open(file_path, "w") as file:
        file.write(json_data)

    print(f"Data has been saved to {file_path}")
```

## تابع parse\_transaction

این تابع تراکنش های یک صفحه را استخراج می کند. همانطور که قبلاً توضیح دادیم این تابع جدول تراکنش ها را به ردیف ها تجزیه می کند و به تابع [convert\\_row\\_to\\_dict](#) می دهد. همچنین در صورت فعال کردن log نمونه ای از تراکنش ها را نمایش می دهد.

```
def parse_transaction(the_soup, log=False):
    table = the_soup.find("tbody", attrs={"class": "align-middle
text-nowrap"})
    rows = [i for i in table.children]
    if log:
        print(f"-----\nsample row:
{str(rows[1])[1:300]}...\n-----\n")
    row_data = [convert_row_to_dict(rows[i]) for i in range(1,
len(rows) - 1)]
    if log:
        print(f"-----\nsample data: {row_data[1]}\n-----\n")
```

```
return row_data
```

## تابع collect\_block\_data

هدف این تابع خواندن تراکنش های یک بلوک است.

این تابع روند زیر را طی می کند:

1. باز کردن صفحه مربوط به بلوک
2. خواندن تعداد صفحات تراکنش یک بلوک ([get\\_page\\_count](#))
3. پیمایش صفحات و خواندن آن صفحات ([parse\\_transaction](#))

```
def collect_block_data(driver, block):
    block_data = list()
    driver.get(f"https://etherscan.io/txs?block={block}")
    the_soup = BeautifulSoup(driver.page_source, 'html.parser')
    page_count = get_page_count(the_soup)
    for page_number in range(1, page_count + 1):
        driver.get(f"https://etherscan.io/txs?block={block}&p={page_number}")
        the_soup = BeautifulSoup(driver.page_source, 'html.parser')
        block_data.extend(parse_transaction(the_soup))
    return block_data
```

## تابع convert\_row\_to\_dict

این تابع ردیف های جدول را می گیرد و به فرمت دیکشنری با محتوای زیر تبدیل می کند:

1. هش معامله
2. روش معامله
3. شماره بلاک
4. زمان به تاریخ-ساعت
5. مبدا تراکنش
6. متغیر بولی آیا مبدا و مقصد تراکنش یکسانند
7. مقصد تراکنش
8. مقدار
9. مقدار مالیات
10. GasPrice
11. ارور ها

موارد بالا را با بررسی html و زیر مجموعه های td به دست می آید مثلا می بینیم که مقدار block در تگ span ذخیره شده است. باقی موارد هم به حالت مشابه به دست می آید.

```
def convert_row_to_dict(row):
```

```

cells = row.findAll("td")
return {"Txn Hash": cells[1].find("a").text,
        "Method": cells[2].find("span").text,
        "Block": cells[3].find("a").text,
        "Time": cells[4].find("span").text,
        "From": cells[7].find("a")["data-bs-title"] if
cells[7].find("a").has_attr('data-bs-title')\
        else cells[7].find("span")["data-bs-title"],
        "Self": cells[8].text == "SELF",
        "To": cells[9].find("a")["data-bs-title"] if
cells[9].find("a").has_attr('data-bs-title')\
        else cells[9].find("span")["data-bs-title"],
        "Value": cells[10].text,
        "Txn Fee": cells[11].text,
        "GasPrice": cells[12].text,
        "Error": cells[1].findAll("span")[0]["data-bs-title"]\
        if len(cells[1].findAll("span")) > 1 else ""}

```

```

{'Txn Hash':
'0x5830f362d58d12323ebad2285016faa45bd25276ab4b76c3bbe80a6f60205a1f',
'Method': 'Transfer',
'Block': '19347624',
'Time': '2024-03-02 12:56:59',
'From': 'Public Tag:
beaverbuild<br/>(0x95222290dd7278aa3ddd389cc1e1d165cc4baf5)',
'Self': False,
'To': '0xd6e4aa932147a3fe5311da1b67d9e73da06f9cef',
'Value': '0.04264153 ETH',
'Txn Fee': '0.00133034',
'GasPrice': '47.24401969',
'Error': ''}

```

تابع `get_page_count`

تگ مربوط به را پیدا می کنیم و مقدار آن را بر می گردانیم.

```

def get_page_count(the_soup):
    return int(the_soup.find(class_="page-link
text-nowrap").text.split()[3])

```

## آنالیز داده ها

در این قسمت باید داده هایی که در قسمت قبل گرفتیم را آنالیز کنیم.

## بارگذاری داده ها

برای بارگذاری داده ها کار خاصی لازم به انجام نیست و فقط کافیسیت کد زیر را اجرا کنیم:

```
df = pd.read_json("./data_block_19347625_19347616.json")
df
```

		Txn Hash	Method	Block	Time	From	Self	To	Value	Txn Fee	GasPrice	Error
0	0x2c57b0a5d26e4a777e47c62ad2684d924215f38dea6f...	Transfer	19347625	2024-03-02 12:57:11	0x88c6c46ebf353a52bdbab708c23d0c81daa8134a	False	Public Tag: stakefish: Fee Recipient (0xoff...	0.03527702 ETH	0.000993	47.233196		
1	0x38808202db7419cfff90deb1fcc01867d8398ee2ae...	Transfer	19347625	2024-03-02 12:57:11	0x4648451b5f87ff8f07d622bd40574bb97e25980	False	Public Tag: Tether: USDT Stablecoin (0xdac...	0 ETH	0.002178	47.233197		
2	0xcc23f903978f8c4bb06df0ebf86350e6cf26ac822...	Transfer	19347625	2024-03-02 12:57:11	0x3672b6ed599c9376092c5a174eb0a0d672859b3d	False	Public Tag: Pepe: PEPE Token (0x6982508145...	0 ETH	0.002631	47.234196		
3	0x6fa7e98544eea9d95b49384de9393a048fe71b2b4b91...	0x415565b0	19347625	2024-03-02 12:57:11	0x3e03428bf447e5adc53089a3e2f36d5b0fd08f4	False	Public Tag: 0x: Exchange Proxy (0xdef1c0de...	0 ETH	0.011291	47.234196		
4	0x2a1861829d94d6d42d6a630535be542290c87087b09...	Transfer	19347625	2024-03-02 12:57:11	0xe49e1f7151ac6f46c23ae2d90f2d0fc1bade5ce	False	0x652441195c870f53360088671c845b3325ad4e6	1.021732845 ETH	0.000992	47.234196		
...	...	...	...	...	...	...	...	...	...	...	...	...
1531	0x4510c5441a8a3470020ce2cf60859ebaea0952317f3...	0xa020e25d	19347616	2024-03-02 12:55:23	0xe75ed6f453c602bd696ce27af11565edc9b46bd	False	0x00000000009e50a7ddb7a7b0e2ee6604fd120e49	0.000000003 ETH	0.004622	47.622935		
1532	0x47c821d9a4c016dcd59e69180e35b08db940cee6b2...	0x28a00911	19347616	2024-03-02 12:55:23	0x111206594f2fb5927f719b4417ab1da3d41b14b1	False	Public Tag: MEV Bot: 0x000...1d3 (0x000000...	23 wei	0.028969	164.606820		
1533	0xf9536f708155998e5d521a4c3cd7822d6f252e77badf...	Execute	19347616	2024-03-02 12:55:23	0x86c4e7bac249275376bc4e500757c1d4196ca77a	False	Public Tag: Uniswap: Universal Router (0x3...	0 ETH	0.011128	47.623935		
1534	0x96b0faddcc84afec9130d65934723d3d48acfd4dd67...	Swap	19347616	2024-03-02 12:55:23	0x962bd36ff2b9bad3a93cf352cf64018e291ab0bd	False	Public Tag: Metamask: Swap Router (0x881d4...	0 ETH	0.011774	47.715144		
1535	0xec91672e00ee58324ffa9e3cedfb272a65822809516...	0x28a00911	19347616	2024-03-02 12:55:23	0x1111e3ef0bae32e14a55e0e7cd9b8905177c2bf	False	Public Tag: MEV Bot: 0x000...1d3 (0x000000...	72 wei	0.009708	47.622935		
1536 rows × 11 columns												

1536 rows x 11 columns

## پاکسازی داده ها

با توجه به مواردی که توضیح دادیم در داده ها، داده تکراری نداریم فقط مشکلی که وجود دارد این است که ارزش معامله گاهی به wei است که با یک جست و جوی ساده متوجه می شویم که ۱۰ به توان -۱۸ اتر ارزش دارد پس با کد زیر داده ها را تبدیل می کنیم:

```
df['Value'] = df['Value'].apply(
    lambda val: float(val.split()[0]) *
    (1 if val.split()[1] == "ETH" else (10 ** -18 if val.split()[1] ==
    "wei" else Exception("??")))
)
```

همچنین ستون های زمان را از قالب رشته به قالب زمان می بریم:

```
df["Time"] = pd.to_datetime(df["Time"])
```

## آنالیز آماری

در این قسمت متغیر های ارزش معامله و مالیات معامله را مورد بررسی قرار می دهیم. جدول زیر برخی از متغیر های آماری مربوط به داده ها را نشان می دهد.

متغیر آماری	با در نظر گرفتن تراکنش های صفر		بدون در نظر گرفتن تراکنش های صفر	
	Value	Txn Fee	Value	Txn Fee
میانگین	0.670632	0.005073	1.459054	0.003097
میانه	0.000000	0.002366	0.057348	0.001066
انحراف معیار	13.044758	0.009911	19.211157	0.004013
IQR	0.045676	0.004909	0.282837	0.004271

without zero				with zero			
	Value	Txn Fee	GasPrice		Value	Txn Fee	GasPrice
count	706.000000	706.000000	706.000000	count	1536.000000	1536.000000	1536.000000
mean	1.459054	0.003097	51.516851	mean	0.670633	0.005073	50.953806
std	19.224778	0.004016	18.340651	std	13.049007	0.009915	15.700495
min	0.000000	0.000980	46.678019	min	0.000000	0.000980	46.678019
25%	0.013132	0.001014	47.434373	25%	0.000000	0.001082	47.301360
50%	0.057349	0.001066	49.033394	50%	0.000000	0.002366	48.678019
75%	0.295970	0.005286	50.724862	75%	0.045677	0.005992	50.301624
max	500.000000	0.052443	361.503483	max	500.000000	0.121668	361.503483
median	0.057349	0.001066	49.033394	median	0.000000	0.002366	48.678019
IQR	0.282838	0.004271	3.290489	IQR	0.045677	0.004909	3.000264

نکات قابل توجه در داده ها:

- بیش از ۵۰ درصد داده ها صفر هستند.
- تفاوت میانه و میانگین بسیار زیاد است معادل ۱.۵ اتر یعنی چیزی حدود \$۴۵۰۰!!! دلیل این موضوع این است که پارامتر هایی مانند میانگین به داده های پرت حساس هستند ولی میانه نیست همین موضوع برای IQR و انحراف معیار صادق است به طوری که از IQR می توان نتیجه گرفت اختلاف تبدلات ۰.۲۸ اتر است اما انحراف معیار ۱۹.۲۲ اتر این مقدار را محاسبه می کند.
- برخلاف دو داده دیگر gas price میانگین و میانه نزدیک به یکدیگر دارد.
- یک تبادل ۵۰۰ اتری معادل \$۱.۸۰۰.۰۰۰ داریم.

## بررسی بیشتر تبادل ۵۰۰ اتری

تبادل به شرح زیر است:

	Txn Hash	Method	Block	Time	From	Self	To	Value	Txn Fee	GasPrice
677	0x0c8141f52e2009804a25e7a23ad68993af0a07179c0a...	Transfer	19347621	2024-03-02 12:56:23	0x4196c40de33062ce03070f058922baa99b28157b	False	Public Tag: Lido: wstETH Token- (0x7f39c581...	500.0	0.006144	50.747998

فرد فرستنده تبادلات زیر را داشته:

	Txn Hash	Method	Block	Time	From	Self	To	Value	Txn Fee	GasPrice
677	0x0c8141f52e2009804a25e7a23ad68993af0a07179c0a...	Transfer	19347621	2024-03-02 12:56:23	0x4196c40de33062ce03070f058922baa99b28157b	False	Public Tag: Lido: wstETH Token- (0x7f39c581...	500.0	0.006144	50.747998
1300	0x845a2fdd441097b91044aea49f8261bdf5a48ddadea6...	Withdraw ETH	19347617	2024-03-02 12:55:35	0x4196c40de33062ce03070f058922baa99b28157b	False	0x893411580e590d62ddba8a703d61cc4a8c7b2b9	0.0	0.018911	47.079028

فرد دریافت کننده را با اسمش جستجو کردیم و به تراکنش های زیر رسیدیم:

	Txn Hash	Method	Block	Time	From	Self	To	Value	Txn Fee	GasPrice
445	0xdfaf3e1ae4e2ef195cdce7f483b20f63735a9f07b...	Approve	19347623	2024-03-02 12:56:47	0x4dc921de85561bca35d33ca15c19f2e1e5043590	False	Public Tag: Lido: stETH Token- (0xae7ab9652...	0.000000	0.002168	50.069108
470	0xbe991696be31477650f01ca7197bd0a471a655b25478...	Transfer	19347623	2024-03-02 12:56:47	Public Tag: Binance 16- (0xdfd5293d8e347dfe...	False	Public Tag: Lido: LDO Token- (0x5a98fcb51...	0.000000	0.006703	51.279108
496	0xed1f0965a9b24cc3e30375062f9fddc1abb7b2e19388...	Transfer	19347622	2024-03-02 12:56:05	Public Tag: Titan Builder- (0x4838b106fce96...	False	Public Tag: Lido: Execution Layer Rewards Vaul...	0.056132	0.001101	49.809176
677	0x0c8141f52e2009804a25e7a23ad68993af0a07179c0a...	Transfer	19347621	2024-03-02 12:56:23	0x4196c40de33062ce03070f058922baa99b28157b	False	Public Tag: Lido: wstETH Token- (0x7f39c581...	500.000000	0.006144	50.747998
705	0xb3dbe655b904df10da6334bf96a1641ba73a28925e9...	Approve	19347621	2024-03-02 12:56:23	0xaeedd5c90df91d8a9f1e9ec96aee93e64c8f045a	False	Public Tag: Lido: stETH Token- (0xae7ab9652...	0.000000	0.003328	50.742275
903	0x23589249d4e5beeac45f0ca096221abeb8e0d7d32c8b...	Approve	19347620	2024-03-02 12:56:11	0xe81032708386337d65678303678caf2597f315e9	False	Public Tag: Lido: stETH Token- (0xae7ab9652...	0.000000	0.003102	47.302860
956	0x05434d3467ade2997c5ce99715b64ef3e4cc9169a9db...	Transfer	19347619	2024-03-02 12:55:39	Public Tag: Faith Builder- (0x5124fcc2b3f99...	False	Public Tag: Lido: Execution Layer Rewards Vaul...	0.014163	0.001068	48.291462

نام دریافت کننده را در اینترنت جستجو می کنیم و می بینیم که بزرگ ترین pool شبکه اتریوم هست و دارایی ای معادل \$۳۵.۰۰۰.۰۰۰.۰۰۰ دارد.

[Networks](#)
[Node operators](#)
[Governance](#)
[Analytics](#)
[Developers](#)
[Community](#)
[About](#)

## Liquid Ethereum Staking

Stake any amount of ETH to receive daily rewards and increase your balance through DeFi.

Staked amount

**9,845,044 ETH**

\$54,996,304,021

Lido APR

**3.4%**

[More info](#)

Rewards paid

**495,357 ETH**

\$1,760,853,097

[Stake Ethereum](#)

[Lido on Ethereum Scorecard](#)

### Why Lido?

With Lido Ethereum staking is made simple and accessible to anyone

**Biggest Ethereum pool**

Lido is the leading Ethereum staking pool letting you

**A global community**

Chat with the team, others in the community, and have

**Growing Protocol Ecosystem**

The growing Lido ecosystem lets you to put your



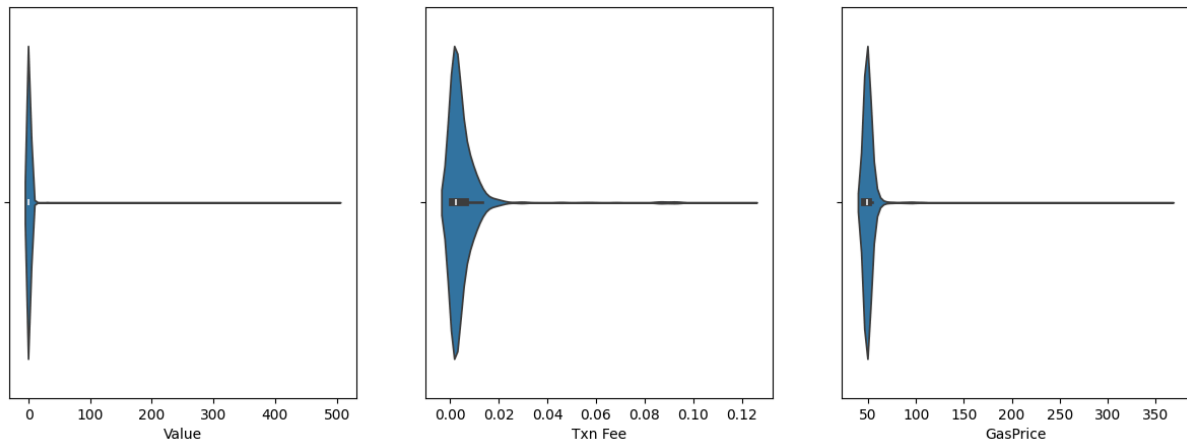
## بصری سازی داده ها

ما بصری سازی داده را در ۳ حالت انجام می دهیم:

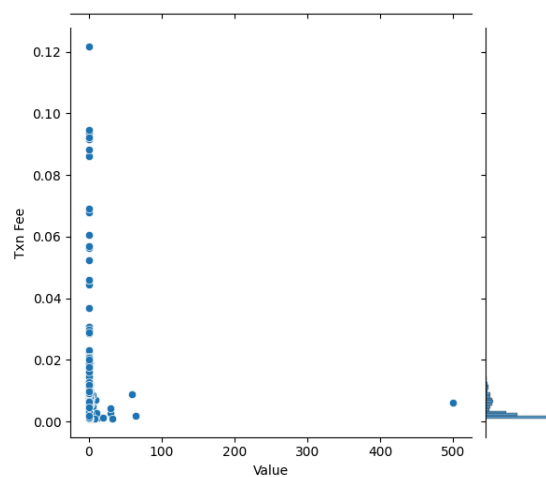
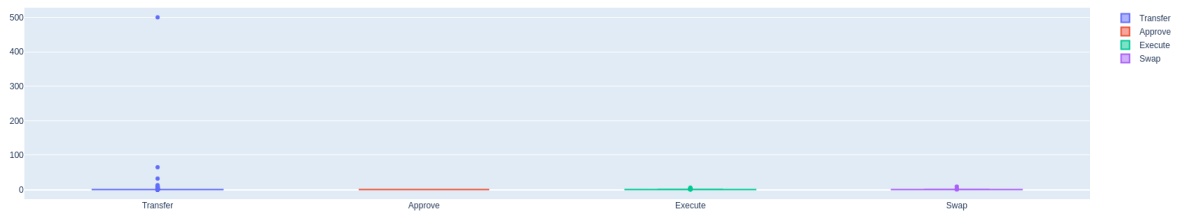
- داده های خام
- لگاریتم داده ها
- لگاریتم داده ها با حذف صفر ها

### داده های خام

در این حالت نمودار های معنا داری نمی گیریم اما نمودار ها به شرح زیر هستند:

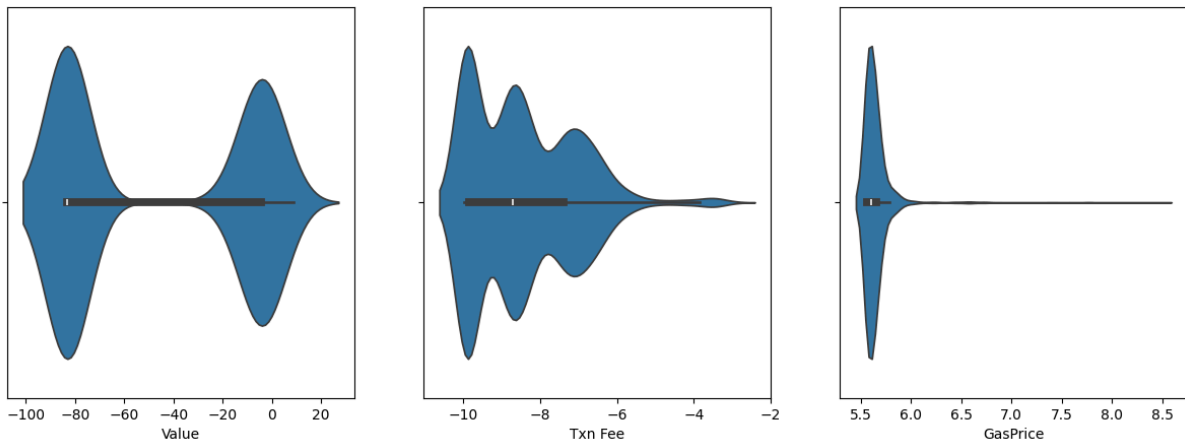


Value box plot by method



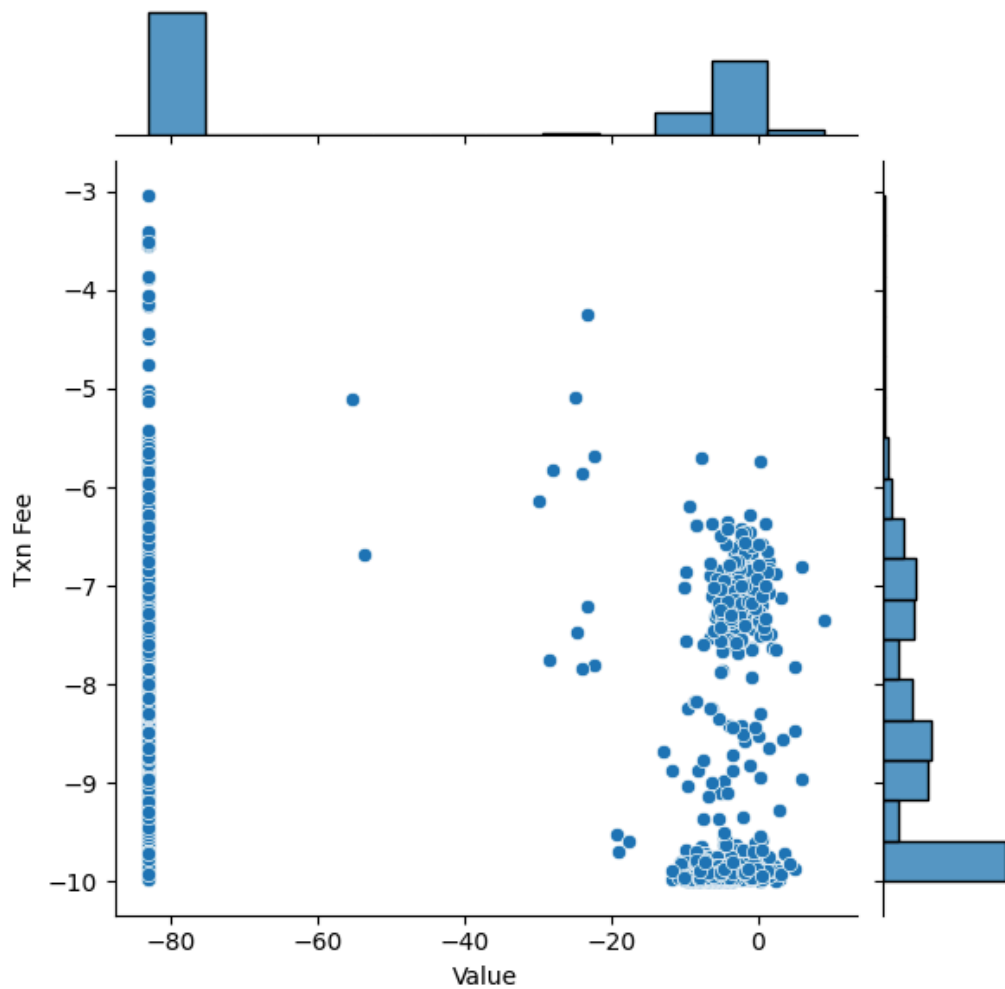
## لگاریتم داده ها

لگاریتم داده ها را می گیریم و برای ۰ ها مقدار اپسیلون (۱۰ به توان منفی ۲۵) را قرار می دهیم.

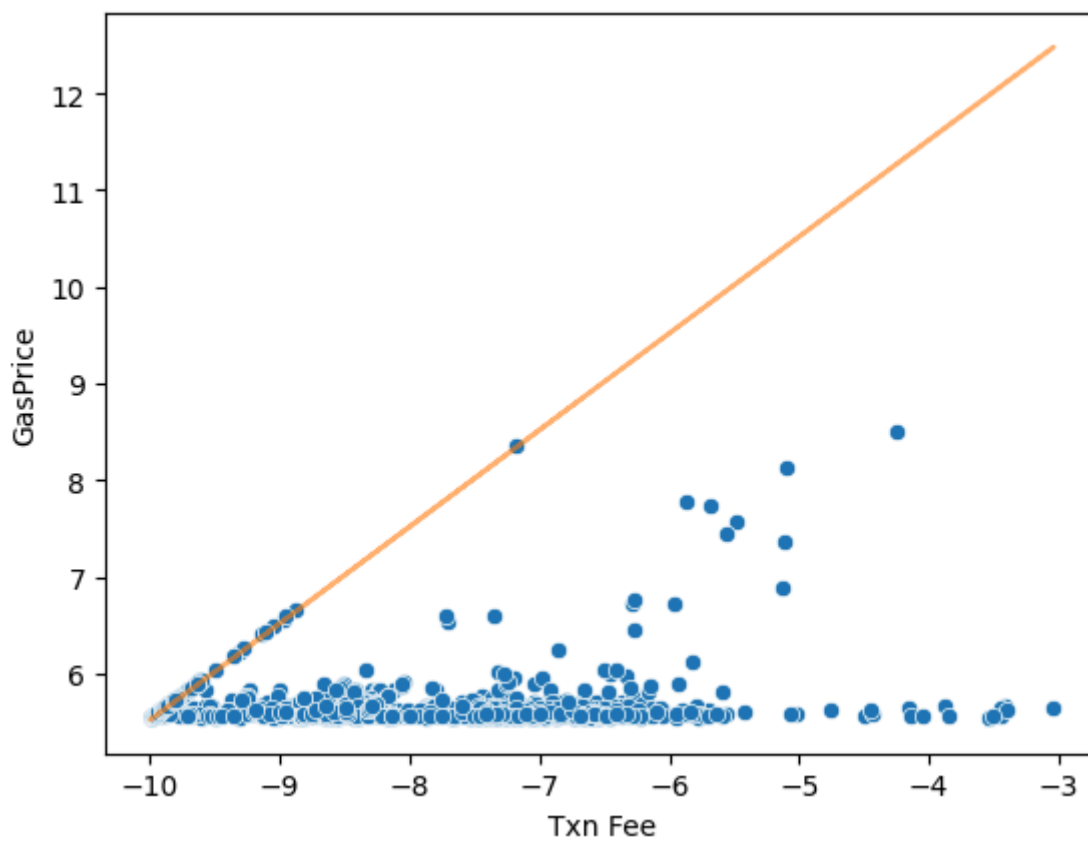
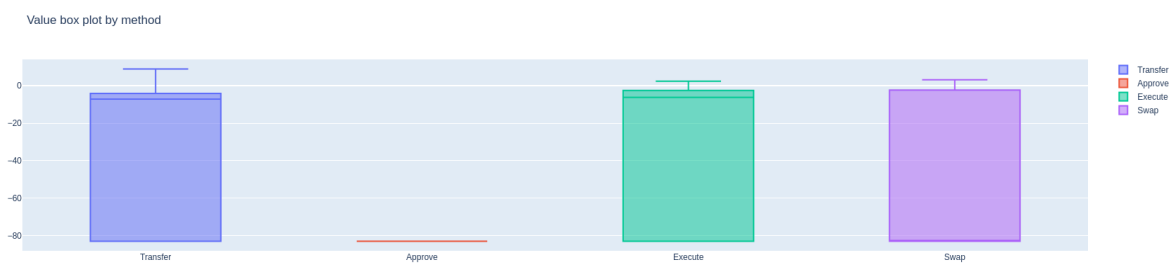


از نمودار ویالون موارد زیر را متوجه می شویم:

- نمودار value مدالیتی bimodal است. (به دلیل داده های صفر)
- نمودار txn fee مدالیتی multimodal است.
- نمودار gas price مدالیتی unimodal است.



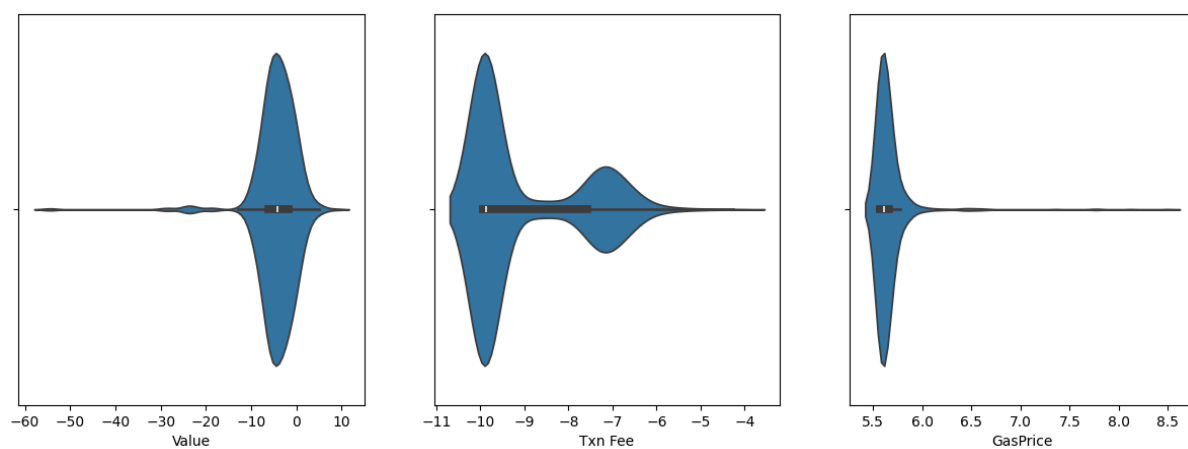
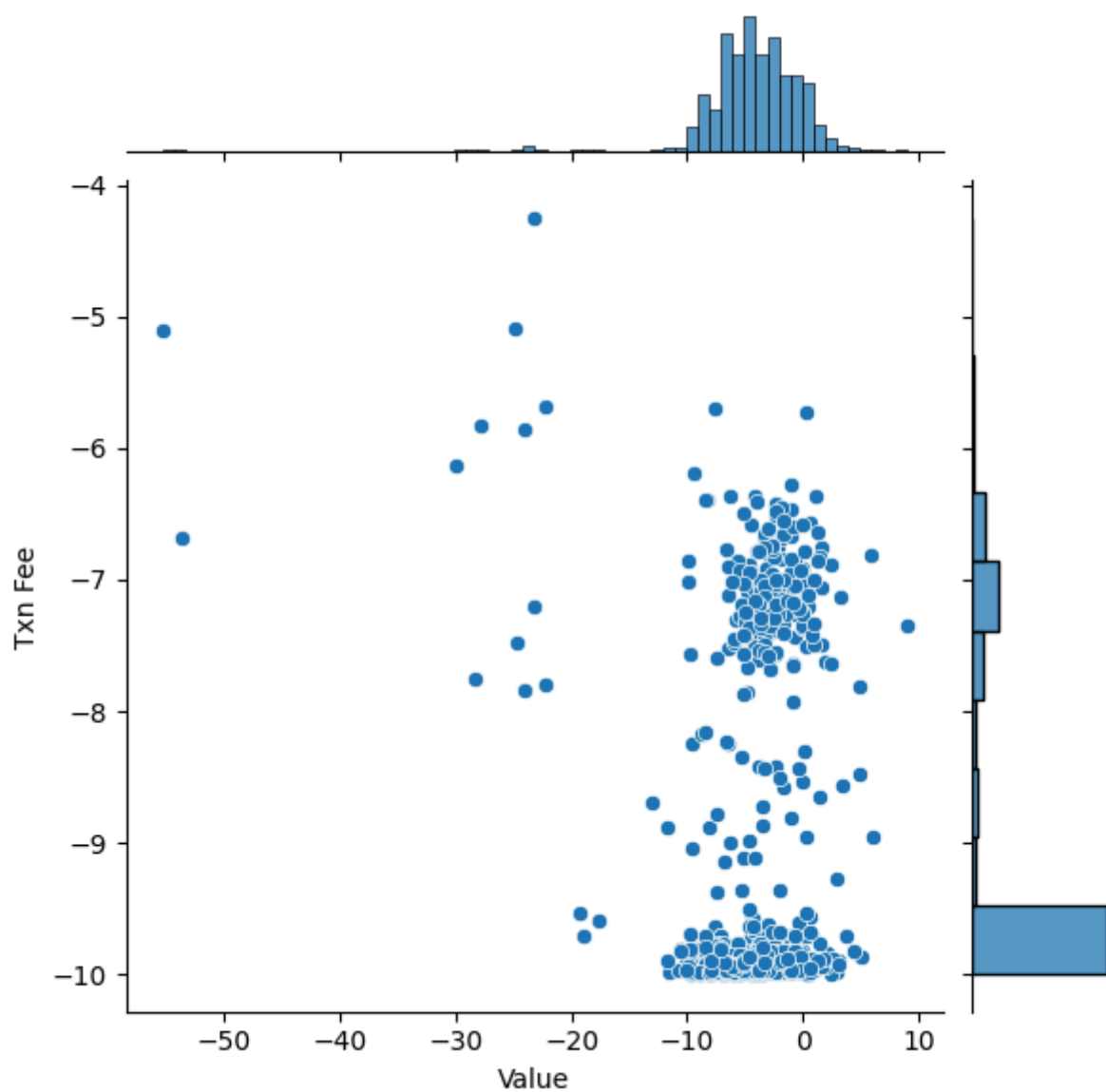
در سمت چپ نمودار بالا داده های صفر تجمیع شده اند.



در نمودار بالا به نظر می آید که رابطه زیر برقرار است.

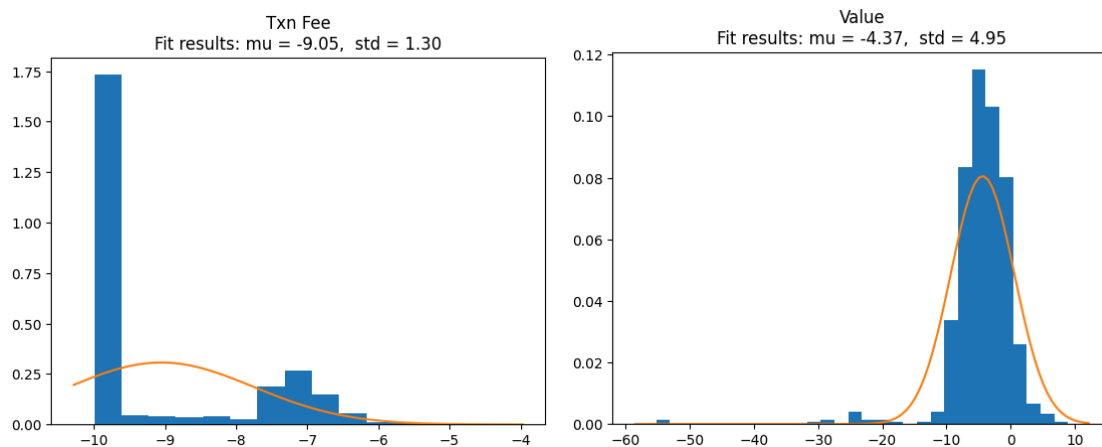
$$GasPrice \leq TxnFee + C$$

## لگاریتم داده ها با حذف صفر ها





روی این داده ها توزیع نرمال فیت می کنیم:



همانطور که می بینید value تقریباً از توزیع نرمال پیروی می کند ولی txn fee اینطور نیست.

## نمونه گیری

### پراکندگی داده

چون پراکندگی داده در این دیتاست زیاد بود تصمیم گرفتیم که برای نمایش داده ها از log آنها استفاده کنیم و بنابر اینکه تعداد زیادی داده صفر بود مینیمم داده را گرفتیم و از صفر ها را بعلاوه عددی از چند مرتبه کمتر از عدد مینیمم کردم.

```
for i in range(len(df)):
    if df.at[i, 'Value']==0:
        df.at[i, 'Value']+=1e-25
```

برای راحتی در کار و از دست نرفتن داده به همان دیتا فریم قبلی ستون هایی اضافه کردم که صرفا لگاریتم ستون value بر مبنای ۱۰ و لگاریتم ستون Txn Fee را بر پایه دو گرفتیم. (چون value پراکندگی بیشتری نسبت به Txn Fee داشت)

```
df.insert(10, "log10_value", [math.log10(df.at[i, 'Value']*10**18) for i in range(len(df))])
df.insert(11, "log2_fee", [math.log2(df.at[i, 'Txn Fee']) for i in range(len(df))])
```

**توجه :** برای درک بهتر چون هر eth ده توان هیجده wei است برای همین تبدیل واحد به اتریوم انجام دادم.

**توجه :** از قسمت ذیل به آخر تصمیم بر این بوده که تعداد نمونه یک بیستم تعداد جامعه باشد.

### نمونه گیری تصادفی

تصمیم بر این گرفتیم که تعداد نمونه ۵ درصد تعداد دیتاست باشد. اطلاعات این نمونه گیری به شرح زیر بود. دقت شود ۱e-۲۵ همان صفر است

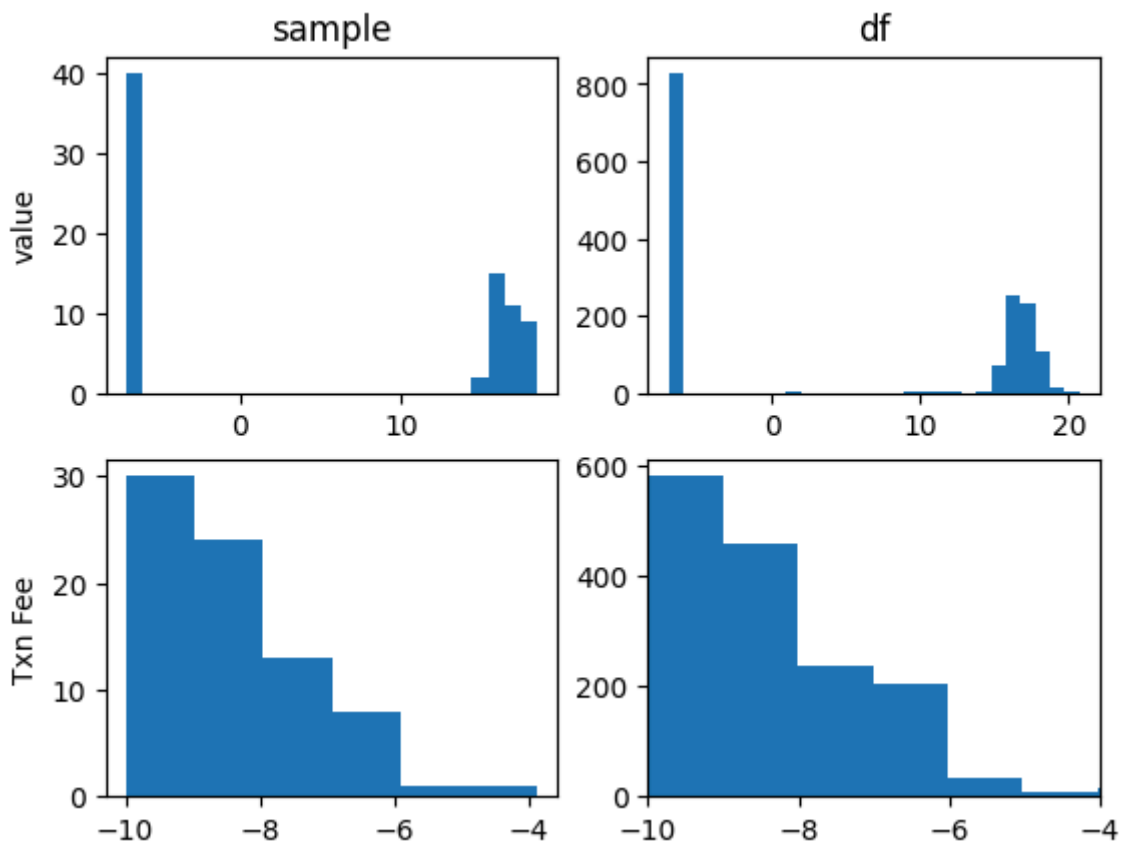
	Value	Txn Fee
count	1.536000e+03	1536.000000
mean	6.706330e-01	0.005073
std	1.304901e+01	0.009915
min	1.000000e-25	0.000980
25%	1.000000e-25	0.001082
50%	1.000000e-25	0.002366
75%	4.567656e-02	0.005992
max	5.000000e+02	0.121668
population		

	Value	Txn Fee
count	7.700000e+01	77.000000
mean	1.421553e-01	0.004687
std	4.482033e-01	0.008280
min	1.000000e-25	0.000988
25%	1.000000e-25	0.001056
50%	1.000000e-25	0.002359
75%	2.494750e-02	0.005471
max	2.500000e+00	0.067792
sample		

در اینجا مشاهده میکنیم که اختلاف بین انحراف معیار نمونه و جامعه زیاد است. **توجیه** : در جامعه چند داده پرت بسیار بزرگ وجود دارد که بر روی انحراف معیار تاثیر می گذارد همانطور که انحراف معیار یک داده حساس به outlier است. برای همین انحراف معیار جامعه افزایش می یابد اما در نمونه گیری احتمال انتخاب outlier بسیار کم است پس در اکثر اوقات انتخاب نمی شود. پس انحراف معیار کاهش پیدا میکند.

برای اینکه تحلیل درستی از نمونه داشته باشیم نیاز به چارک ها نیز داریم زیرا آنها به outlier حساس نیستند.

همانطور که مشاهده میشود در نمونه و جامعه چارک ها نزدیک به هم هستند. دومین چیزی که باعث شده انحراف معیار نمونه بسیار کمتر از انحراف معیار جامعه باشد اینست که انحراف معیار نمونه تخمینگری اریب برای انحراف معیار جامعه است و اگر بخواهیم واریانس نمونه تخمینگری ناریب برای جامعه باشد باید در مخرج کسر بجای  $n$  از  $n-1$  استفاده کرد.



این هم نمودار های مربوط به جامعه و نمونه که مربوط به ستون های Txn Fee و value است نمایش داده شده است. مشاهده میشود که نمودار ها شبیه به هم است. (در کشیدن این نمودار ها چون از داده ها لگاریتم گرفته شده و نمایش داده شده اند و اختلاف بازه عددی کم است در انتخاب bins از فرمول  $\max - \min$  استفاده کردم).

**توجه** : دقت شود واحد لگاریتم گرفته شده wei است و داده های منفی در نمودار value نشان دهنده صفر است.

## نمونه گیری طبقه ای

در این قسمت من سه نوع مختلف نمونه گیری انجام دادم و برای هر کدام دلایل آنها را نیز ذکر کرده ام. لازم به ذکر است که نمونه گیری طبقه ای استفاده شده در همه نمونه گیری های طبقه ای proportionate است یعنی نمونه گیری از هر طبقه نسبت به وزن جمعیت آن است.

### طبقه بندی بر اساس مقدار

چون مقدار ها از صفر تا ۱۰ بتوان بیست wei قابل تغییرند و نزدیک به صفر دیتا بسیار بیشتری وجود دارد من برای اینکه دیتا های بزرگ هم شانس انتخاب داشته باشند دیتا ست را بر اساس لگاریتم بر پایه ۱۰ بصورت بازه های به طول یک مرتب کردم.

```
"""I want to sample that stratified by value thus i classify value with
line bellow"""
df.insert(0,"Value_Category",[round(math.log10(df.at[i,'Value'])*10**18))
for i in range(len(df)) ],True)
proportionate_stratified_sample_based_on_value =
df.groupby('Value_Category', group_keys=False).apply(lambda x:
x.sample(frac =SAMPLE_PERCENT))
proportionate_stratified_sample_based_on_value[["Value","Txn
Fee"]].describe()
```

	Value	Txn Fee
count	1.536000e+03	1536.000000
mean	6.706330e-01	0.005073
std	1.304901e+01	0.009915
min	1.000000e-25	0.000980
25%	1.000000e-25	0.001082
50%	1.000000e-25	0.002366
75%	4.567656e-02	0.005992
max	5.000000e+02	0.121668

population

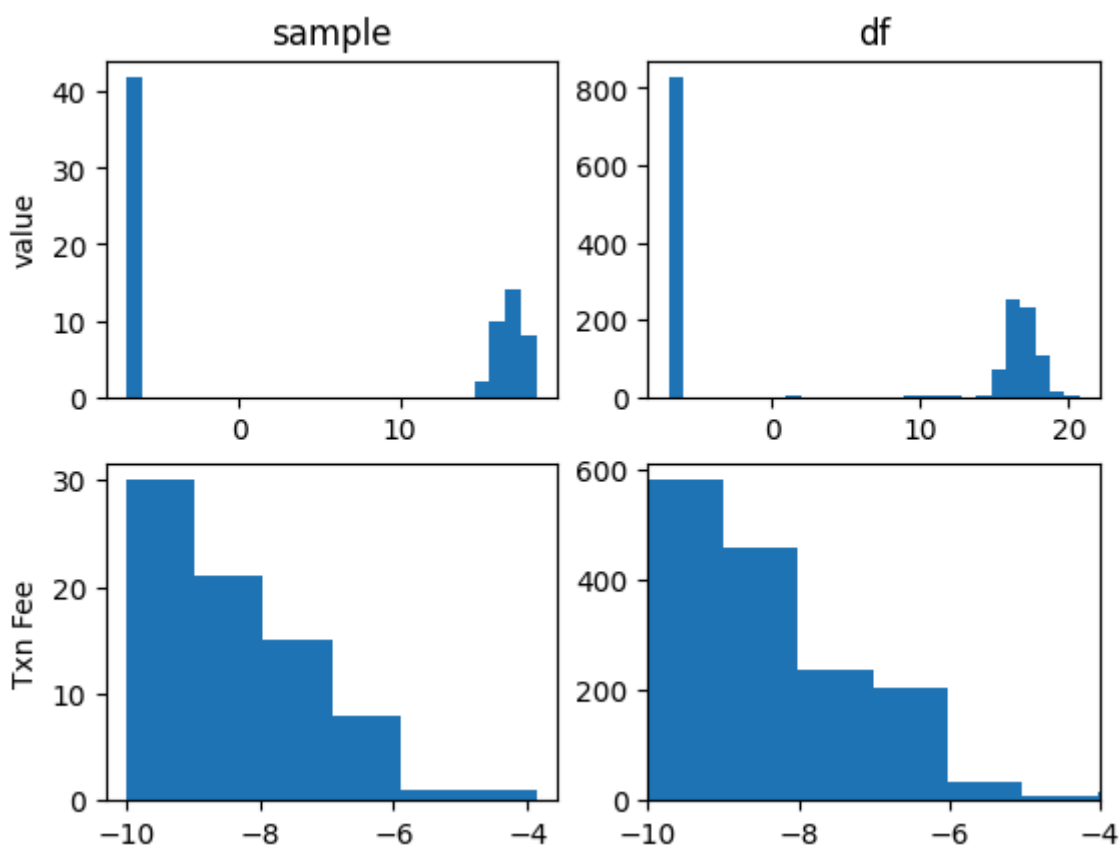
	Value	Txn Fee
count	7.600000e+01	76.000000
mean	1.531477e-01	0.004526
std	4.780717e-01	0.008254
min	1.000000e-25	0.000980
25%	1.000000e-25	0.001056
50%	1.000000e-25	0.002491
75%	5.413221e-02	0.005553
max	3.361160e+00	0.069124

sample



	Value	Txn Fee		Value	Txn Fee
count	7.600000e+01	76.000000	count	7.700000e+01	77.000000
mean	1.531477e-01	0.004526	mean	1.421553e-01	0.004687
std	4.780717e-01	0.008254	std	4.482033e-01	0.008280
min	1.000000e-25	0.000980	min	1.000000e-25	0.000988
25%	1.000000e-25	0.001056	25%	1.000000e-25	0.001056
50%	1.000000e-25	0.002491	50%	1.000000e-25	0.002359
75%	5.413221e-02	0.005553	75%	2.494750e-02	0.005471
max	3.361160e+00	0.069124	max	2.500000e+00	0.067792
Stratified by value			random		

همانطور که مشاهده می شود نمونه گیری طبقه ای که گرفته شده چنان فرقی با تصادفی ندارد و این نشان دهنده ی خوبی طبقه بندی ماست.



همانطور که دیده می شود نمونه گیری بسیار به واقعیت نزدیک است.

## طبقه بندی بر اساس method

وقتی متد های مختلف را مشاهده کردم دیدم در هر متد مقدار پول های متفاوتی جابجا شده است برای همین آنها را طبقه بندی کرده و از هر کدام بر حسب وزنشان نمونه انتخاب کردم.

```
proportionate_stratified_sample_based_on_method = df.groupby('Method',
group_keys=False).apply(lambda x: x.sample(frac=SAMPLE_PERCENT))
proportionate_stratified_sample_based_on_method[["Value", "Txn
Fee"]].describe()
```

	Value	Txn Fee
count	1.536000e+03	1536.000000
mean	6.706330e-01	0.005073
std	1.304901e+01	0.009915
min	1.000000e-25	0.000980
25%	1.000000e-25	0.001082
50%	1.000000e-25	0.002366
75%	4.567656e-02	0.005992
max	5.000000e+02	0.121668

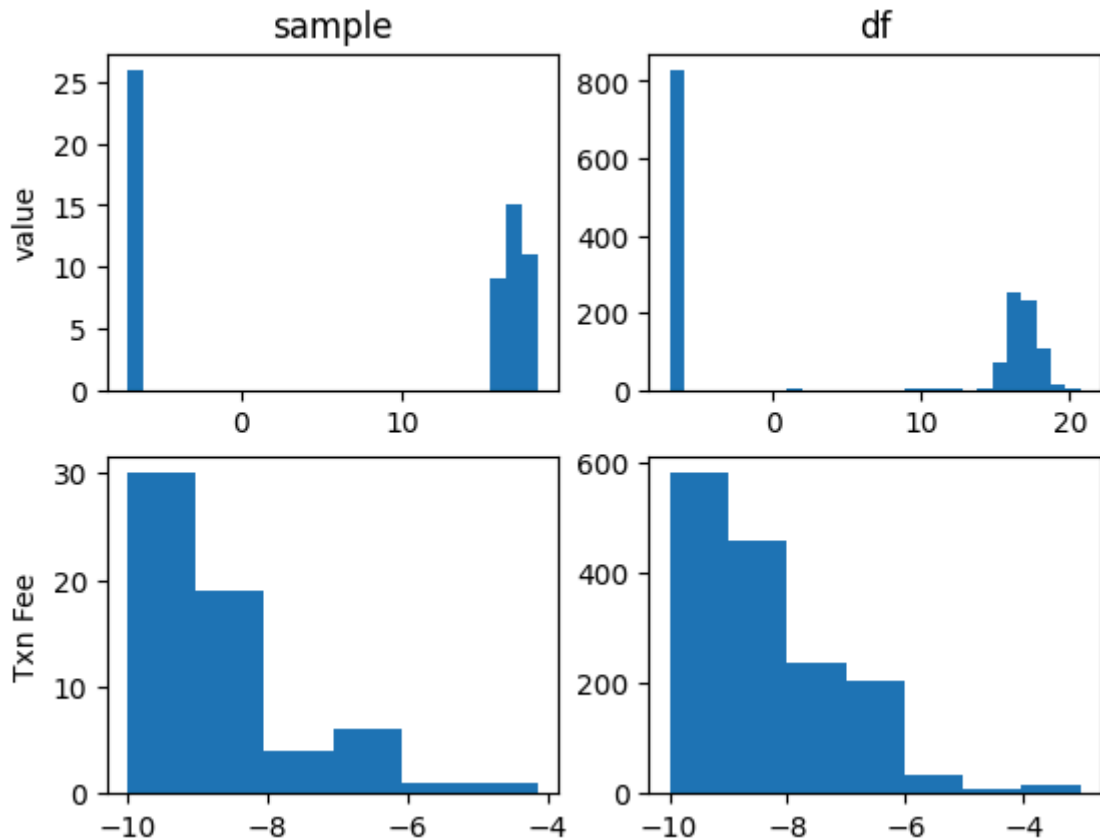
population

	Value	Txn Fee
count	6.100000e+01	61.000000
mean	8.339925e+00	0.004456
std	6.400080e+01	0.012148
min	1.000000e-25	0.000988
25%	1.000000e-25	0.001067
50%	1.000000e-25	0.002204
75%	4.650000e-02	0.003112
max	5.000000e+02	0.094585

sample

مشاهده میکنید که در این نوع طبقه بندی نه تنها انحراف معیار پایینتر از مقدار جامعه نبود بلکه بالاتر از آن هم هست.

دلیل آن اینست که نمونه گیری ما اریب است زیرا چون نمونه گیری طبقه ای ما proportionate است پس باید در هر طبقه ۵ درصد تعداد آن طبقه انتخاب شود در صورتی که بسیاری از متد ها یکتا هستند و ۵ درصد آنها است پس از بسیاری از طبقه ها هیچ انتخابی انجام نمیشود.



این اختلاف در نمودار ها نیز مشاهده میشود.

### طبقه بندی بر اساس block

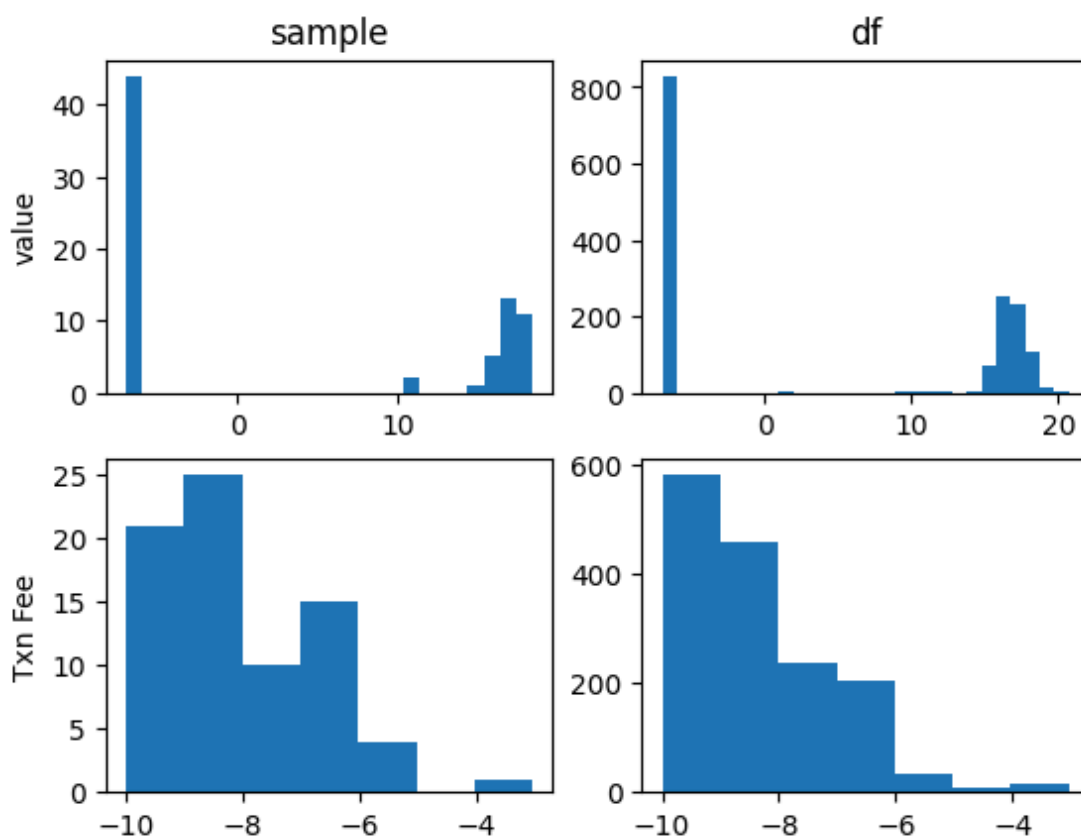
دلیل انتخاب من این بود که با اینکه نمیدانستم block ها چیست اما همینکه خود سایت اتریوم دسته بندی خود را بر اساس block های مختلف می گذارد یعنی block بندی معنا دار است و باعث تفاوت بین دیتا های هر block است. برای همین باید از هر block نمونه گیری انجام بگیرد.

```
proportionate_stratified_sample_based_on_blocks =
df.groupby('Block',group_keys=False).apply(lambda
x:x.sample(frac=SAMPLE_PERCENT))
proportionate_stratified_sample_based_on_blocks[["Value","Txn
Fee"]].describe()
```

	Value	Txn Fee
count	1.536000e+03	1536.000000
mean	6.706330e-01	0.005073
std	1.304901e+01	0.009915
min	1.000000e-25	0.000980
25%	1.000000e-25	0.001082
50%	1.000000e-25	0.002366
75%	4.567656e-02	0.005992
max	5.000000e+02	0.121668
population		

	Value	Txn Fee
count	7.600000e+01	76.000000
mean	1.531445e-01	0.004888
std	6.483671e-01	0.005023
min	1.000000e-25	0.000988
25%	1.000000e-25	0.001181
50%	1.000000e-25	0.002977
75%	6.655000e-02	0.007599
max	5.490000e+00	0.028969
sample		

باز هم تفاوت بین جامعه و نمونه کم است و صرفاً به دلیلی که قبلاً توضیح داده شد بین انحراف معیار نمونه و جامعه صرفاً تفاوت وجود دارد. پس این نشان‌دهنده از نا اریب بودن نمونه‌گیری طبقه‌ای ماست.



در نمودارها نیز دیده می‌شود تفاوت چندانی بین نمونه و جامعه وجود ندارد.

## پاسخ به سوالات

### سوال ۱: مشکلات استخراج کننده و محدودیت های سایت

دو مشکل سرعت پایین و بلاک شدن توسط سایت از اصلی ترین مشکلات و محدودیت های وب اسکرپینگ هستند.

سرعت پایین به این علت رخ می دهد که باید برنامه صفحات را مثل یک کاربر بگردد و نمی تواند به یکباره هزاران داده را بخواند و باید منتظر بارگذاری داده های بیهوده ای مثل فایل های CSS یا تصاویر بماند و صفحه به صفحه سایت را پیمایش کند. همچنین تبدیل صفحات HTML به ساختار داده مد نظر نیز می تواند زمان بر باشد. البته که در صورتی که سایت static باشد و محتوای آن متغیر نباشد (مثلا گوگل) می توان از کتابخانه هایی مثل request استفاده کرد که همانند نامش به جای بارگذاری صفحات وب با web driver که سلیوم از آن استفاده می کند این کتابخانه از request ها استفاده می کند و محتوا خام HTML را دریافت می کند و آن ها را در web driver پردازش نمی کند.

سایت های مختلف از برای استخراج داده ها محدودیت هایی می گذارند که علل مختلفی دارد:

- ایجاد بار اضافی بر روی سرور ها
- درآمدزایی از طریق فروش اطلاعات و API
- محدودیت روی استفاده و تحلیل اطلاعات

برای استخراج داده ها لازم است هزار درخواست به سرور های مقصد ارسال شود که این موضوع باعث ایجاد بار زیادی روی سرور ایجاد می کند. سایت های مختلف با فروش اطلاعات خود یا فروش اطلاعات درآمد زایی می کنند در این پروژه نیز خود سایت گزینه های مختلفی برای فروش API داشت:

ETHERSCAN API

### Build Precise & Reliable Apps with Etherscan APIs

Data from the leading Ethereum Block Explorer catered to your needs.

[API Pricing Plans](#)
[API Documentation](#)

```

1 // Get ERC-20 Token TotalSupply by Contract
2 // Address
3 {
4   "tokenName": "Token Name",
5   "symbol": "Token Symbol",
6   "divisor": "18",
7   "tokenType": "ERC20",
8   "descr":
9 }
10
11 https://api.etherscan.io/api
12 ?module=stats
13 &action=tokensupply
14 &contractaddress=0x5799b64a57749b8f932f1a3
15 395792e12e7055
16 &apikey=YourApiKeyToken

```

Ready to get started?

Select the plan that best suits you or contact us for a custom option.

[Monthly](#)
[Quarterly \(10% Off\)](#)
[Yearly \(15% Off\)](#)

FREE	STANDARD	ADVANCED	PROFESSIONAL	PRO PLUS
\$0	\$199/mo	\$299/mo	\$399/mo	\$899/mo
*Attribution required				
<ul style="list-style-type: none"> <li>✓ 5 call/second limit</li> <li>✓ Up to 100,000 API calls/day</li> <li>✓ All existing community endpoints</li> <li>✓ Community support</li> </ul>	<ul style="list-style-type: none"> <li>✓ 10 calls/second limit</li> <li>✓ Up to 200,000 API calls/day</li> <li>✓ All existing community endpoints</li> <li>✓ Access to <a href="#">API Pro endpoints</a></li> <li>✓ Escalated support</li> </ul>	<ul style="list-style-type: none"> <li>✓ 20 calls/second limit</li> <li>✓ Up to 500,000 API calls/day</li> <li>✓ All existing community endpoints</li> <li>✓ Access to <a href="#">API Pro endpoints</a></li> <li>✓ Escalated support</li> </ul>	<ul style="list-style-type: none"> <li>✓ 30 calls/second limit</li> <li>✓ Up to 1,000,000 API calls/day</li> <li>✓ All existing community endpoints</li> <li>✓ Access to <a href="#">API Pro endpoints</a></li> <li>✓ Escalated support</li> </ul>	<ul style="list-style-type: none"> <li>✓ 30 calls/second limit</li> <li>✓ Up to 1,500,000 API calls/day</li> <li>✓ All existing community endpoints</li> <li>✓ Access to <a href="#">API Pro endpoints</a></li> <li>✓ Access to <a href="#">Address Metadata endpoint</a></li> <li>✓ Escalated support</li> </ul>
*Limited to ONE app license.				
<a href="#">Get Started Now</a>	<a href="#">Get Started Now</a>	<a href="#">Get Started Now</a>	<a href="#">Get Started Now</a>	<a href="#">Get Started Now</a>

و از این طریق کسب درآمد می کرد. بعضی از سایت ها هم وجود دارند که نمی خواهند داده هایشان تحلیل شود مثلا در یک سایتی می شود استعلام گیری کرد اگر فردی برنامه ای روی این وبسایت بنویسد می تواند با بررسی تمام کد ها تمامی مدارک سایت را استخراج کند که این موضوع مطلوب سایت و آن شرکت نیست. در این پروژه نیز در هنگام استخراج در دفعه اول بعد از ۵ بلاک سایت ما را مسدود کرد ولی با گذاشتن یک تاخیر یک ثانیه بین صفحات این مشکل برطرف می شود البته که کندی اینترنت نیز اینجا مفید واقع شده. اما فکر نمی کنم این محدودیت ها در تحلیل ما مشکلی ایجاد کند زیرا همیشه راهی برای دور زدن آن وجود دارد.

## سوال ۲: غیر قابل اعتماد شدن تحلیل

ممکن است که داده های استخراج شده بازنمایی مناسبی از کل جامعه نداشته باشد مثلا درباره این پروژه موارد زیر می تواند مشکل ساز باشد:

- داده های در ساعت خاصی از روز هستند و ممکن است در تمام روز تراکنش های به این شکل نباشند. برای حل این مشکل می توان بلوک ها را در ساعت های مختلفی از روز دریافت کرد.
- شبکه اتریوم سال هاست که وجود دارد و برای تحلیل آن لازم است اطلاعات از گذشته نیز وجود داشته باشد و با داشتن داده های یک روز یا یک ماه اخیر نمی توان کل شبکه را به درستی تحلیل کرد.
- استفاده از تخمین گر های اریب مثلا اگر ما در پروژه برای تحلیل خود از واریانس با فرمول  $\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$  استفاده کنیم تحلیل ما اریب می شود چون درجه آزادی متغیر ها در این فرمول  $n - 1$  است.
- نمونه گیری اریب با دسته بندی نامناسب که در بخش آخر همانطور که مشاهده کردیم دسته بندی کردن داده ها با method روش مناسبی نیست زیرا مثلا بعضی method ها می تواند اصلا برای انتقال آتی پول نباشند. دسته بندی نامناسب در بخش جمع آوری داده ها نیز هست که چند مورد اول ذکر شده به همین موضوع اشاره داشت.
- استفاده از تخمین گر های حساس به داده های پرت می تواند ما را از درک درست داده ها دور کند و باید از تخمین گر هایی که از داده های پرت حساس نیستند نیز استفاده کرد

## سوال ۳: مصور سازی داده ها

ممکن است تعداد داده ها زیاد باشد و بدون مصور سازی داده ها نتوان به تحلیل دقیقی رسید و برای رسیدن به تحلیل بدون مصور سازی لازم است زمان بسیاری صرف دیدن تک تک داده ها شود و در آخر هم به دلیل خطای انسانی به تحلیل درستی نرسیم همانطور که در این پروژه هم در اولین رویارویی با داده ها پیشبینی نمیشد بیش از نیمی از داده ها دارای value صفر باشند و پس از اینکه روی نمودار پیاده شدند به این مهم دست یافتیم. همچنین استفاده از سنج ها (میانگین، میانه و...) به تنهایی برای درک داده ها کافی نیست و در کنار آن باید از نمودار ها نیز استفاده کرد تا درک بهتری پیدا کرد همچنین نمودار ها به ما کمک

می کند که ارتباط سنجه های خود را بهتر درک کنیم مثلاً نمودار جعبه ای دید خوبی نسبت به چارک های داده و همچنین داده های پرت آن می دهد یا کشیدن توزیع داده ها درک خوبی از چولگی، میانگین و واریانس آن می دهد.

## سوال ۴: مقایسه دو روش نمونه گیری

چون [نمونه گیری طبقه ای](#) من proportionate بود یعنی بر اساس وزن هر طبقه از آن طبقه انتخاب میشد فرق زیادی با [نمونه گیری تصادفی](#) نداشت اما در صورتی بد میشد که جمعیت بسیاری طبقات کم بود و برای مثال انتخاب کردن ۰.۰۵ درصدی از یک طبقه با تعداد ۱ یا ۲ معنا نداشت و اگر طبقات به اینگونه بسیار زیاد بود بسیاری از داده ها اصلاً در نمونه گیری شرکت داده نمی شدند. همانطور که در [نمونه گیری طبقه ای بر اساس method](#) این اتفاق افتاد.

## مراجع

1. [ethereum.org](https://ethereum.org)
2. [Beautiful Soup Documentation](https://beautifulsoup.readthedocs.io/)
3. [Selenium Documentation](https://selenium-python.readthedocs.io/)
4. [Pandas Documentation](https://pandas.pydata.org/pandas-docs/stable/)
5. [Seaborn Documentation](https://seaborn.pydata.org/)
6. [Numpy Documentation](https://numpy.org/doc/stable/)
7. [Matplot Documentation](https://matplotlib.org/)
8. [Scipy Documentation](https://docs.scipy.org/doc/scipy/)
9. [Plotly Documentation](https://plotly.com/python/)
10. [Geeks for geeks](https://www.geeksforgeeks.org/)
11. [Stack overflow](https://stackoverflow.com/)