МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

WEB-Аналітика

ЛАБОРАТОРНА РОБОТА №1

Виконав

студент групи ФІ-42мн

Бєш Радомир Андрійович

Київ 2025

Завдання:

На основі будь-якого access.log сформувати датасет, що надав би інформацію про користувачів веб-ресурсу, а потім виконати наступні кроки:

a. Визначити кількість користувачів за днями

b. Ранжувати користувачів за User-Agent

c. Ранжувати користувачів за операційними системами

d. Ранжувати користувачів за країною запиту

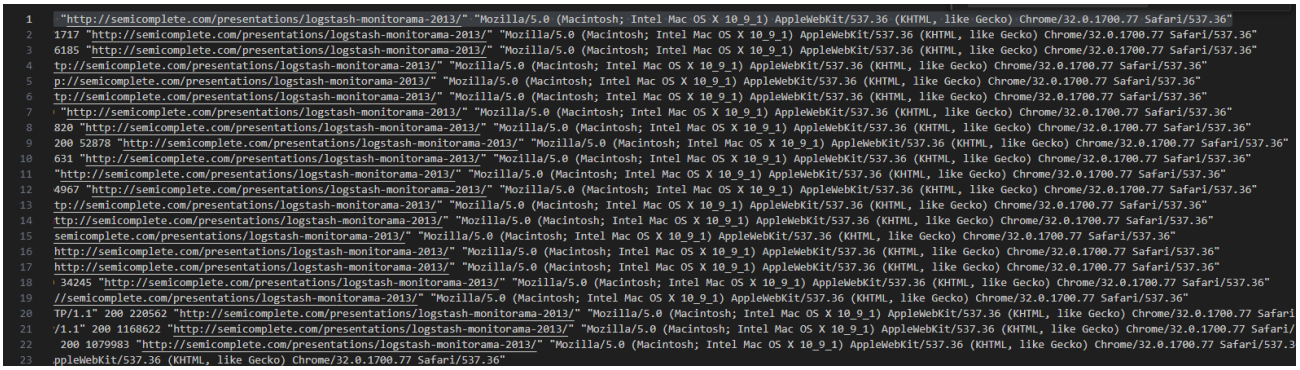
e. Виокремити пошукових ботів

f. Детектувати аномалії (якщо такі є)

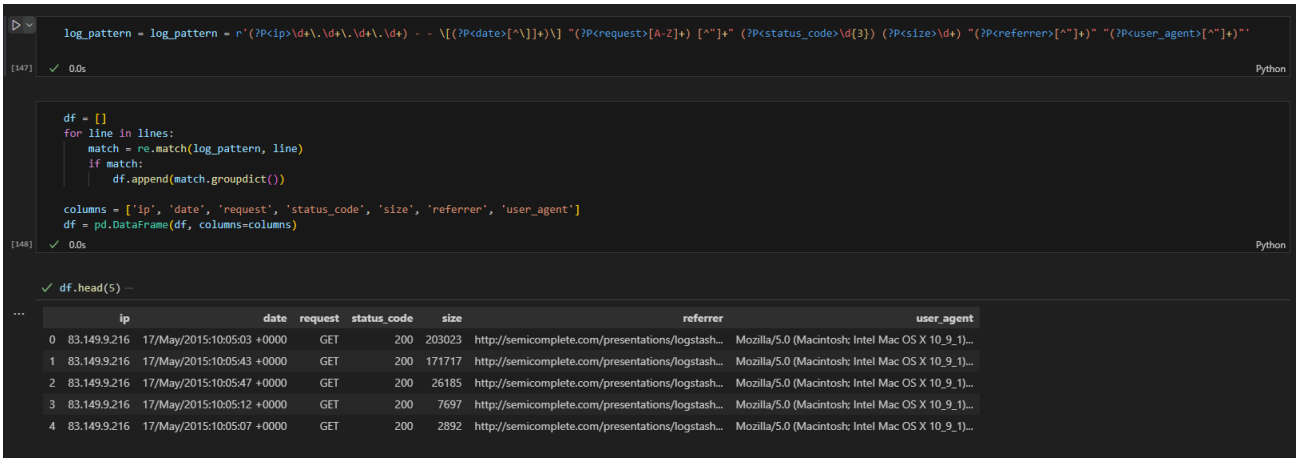
Хід роботи

Було обрано датасет apache_logs.txt, знайшов на Github але нажаль я загубив лінк, де я його знайшов.

Датасет виглядає так:

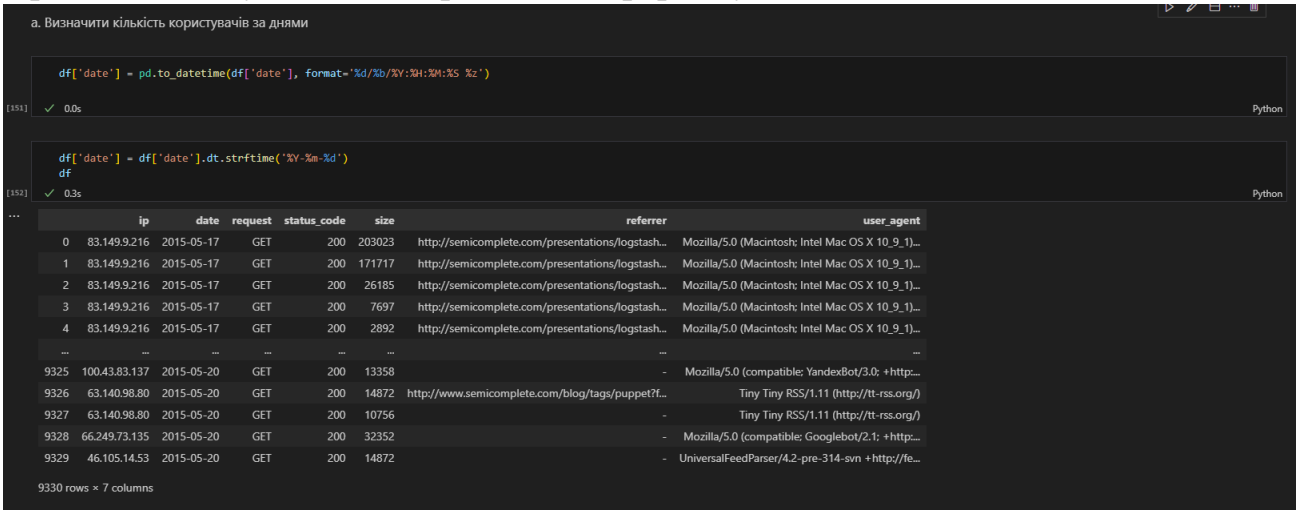


Тому приведемо його в “нормальний вид”.



a. Визначити кількість користувачів за днями

Приведемо дату до більш прийнятого формату



a. Визначити кількість користувачів за днями

```
df['date'] = pd.to_datetime(df['date'], format='%d/%b/%Y:%H:%M:%S %z')
```

[151] ✓ 0.0s Python

```
df['date'] = df['date'].dt.strftime('%Y-%m-%d')
df
```

[152] ✓ 0.3s Python

	ip	date	request	status_code	size	referrer	user_agent
0	83.149.9.216	2015-05-17	GET	200	203023	http://semicomplete.com/presentations/logstash...	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1)...
1	83.149.9.216	2015-05-17	GET	200	171717	http://semicomplete.com/presentations/logstash...	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1)...
2	83.149.9.216	2015-05-17	GET	200	26185	http://semicomplete.com/presentations/logstash...	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1)...
3	83.149.9.216	2015-05-17	GET	200	7697	http://semicomplete.com/presentations/logstash...	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1)...
4	83.149.9.216	2015-05-17	GET	200	2892	http://semicomplete.com/presentations/logstash...	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1)...
...	...	...	...	...	...	...	...
9325	100.43.83.137	2015-05-20	GET	200	13358	-	Mozilla/5.0 (compatible; YandexBot/3.0; +http:...
9326	63.140.98.80	2015-05-20	GET	200	14872	http://www.semicomplete.com/blog/tags/puppet?f...	Tiny Tiny RSS/1.11 (http://tt-rss.org/)
9327	63.140.98.80	2015-05-20	GET	200	10756	-	Tiny Tiny RSS/1.11 (http://tt-rss.org/)
9328	66.249.73.135	2015-05-20	GET	200	32352	-	Mozilla/5.0 (compatible; Googlebot/2.1; +http:...
9329	46.105.14.53	2015-05-20	GET	200	14872	-	UniversalFeedParser/4.2-pre-314-svn +http://fe...

9330 rows × 7 columns

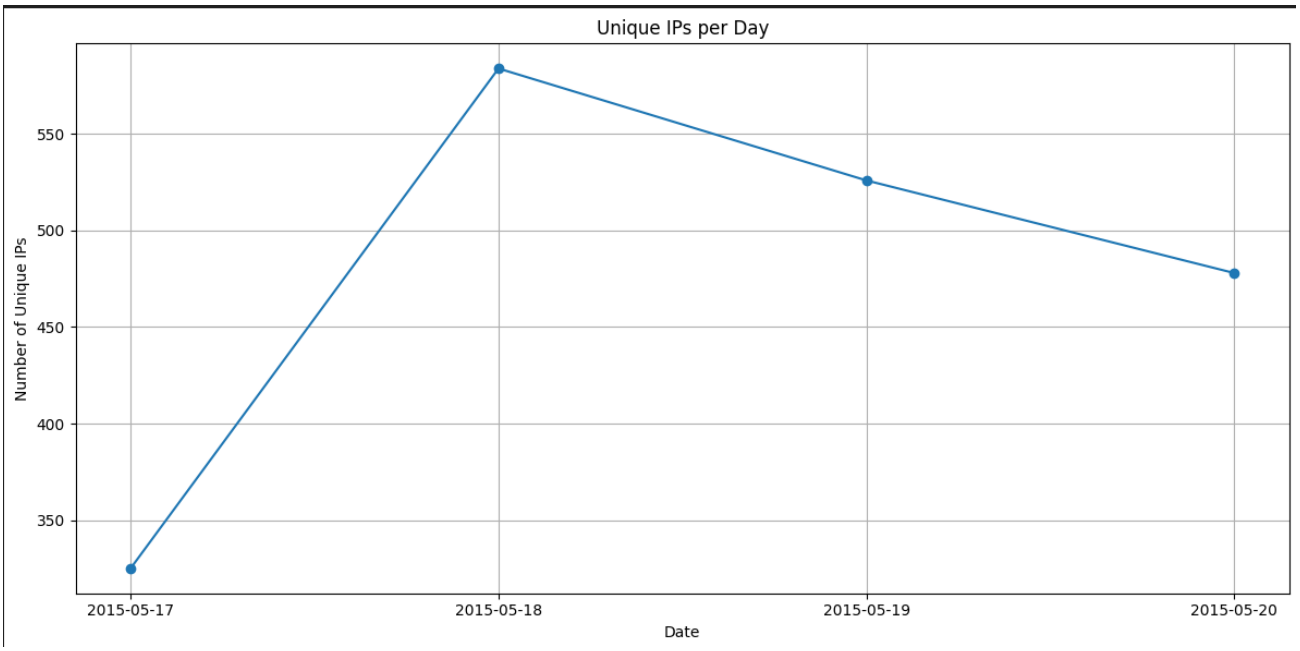
Отже кількість унікальних користувачів становить:

```
df_unique_ip = df.groupby('date')['ip'].nunique().reset_index()
df_unique_ip
```

[165] ✓ 0.0s

	date	ip
0	2015-05-17	325
1	2015-05-18	584
2	2015-05-19	526
3	2015-05-20	478

Візуалізуємо:



b. Ранжувати користувачів за User-Agent

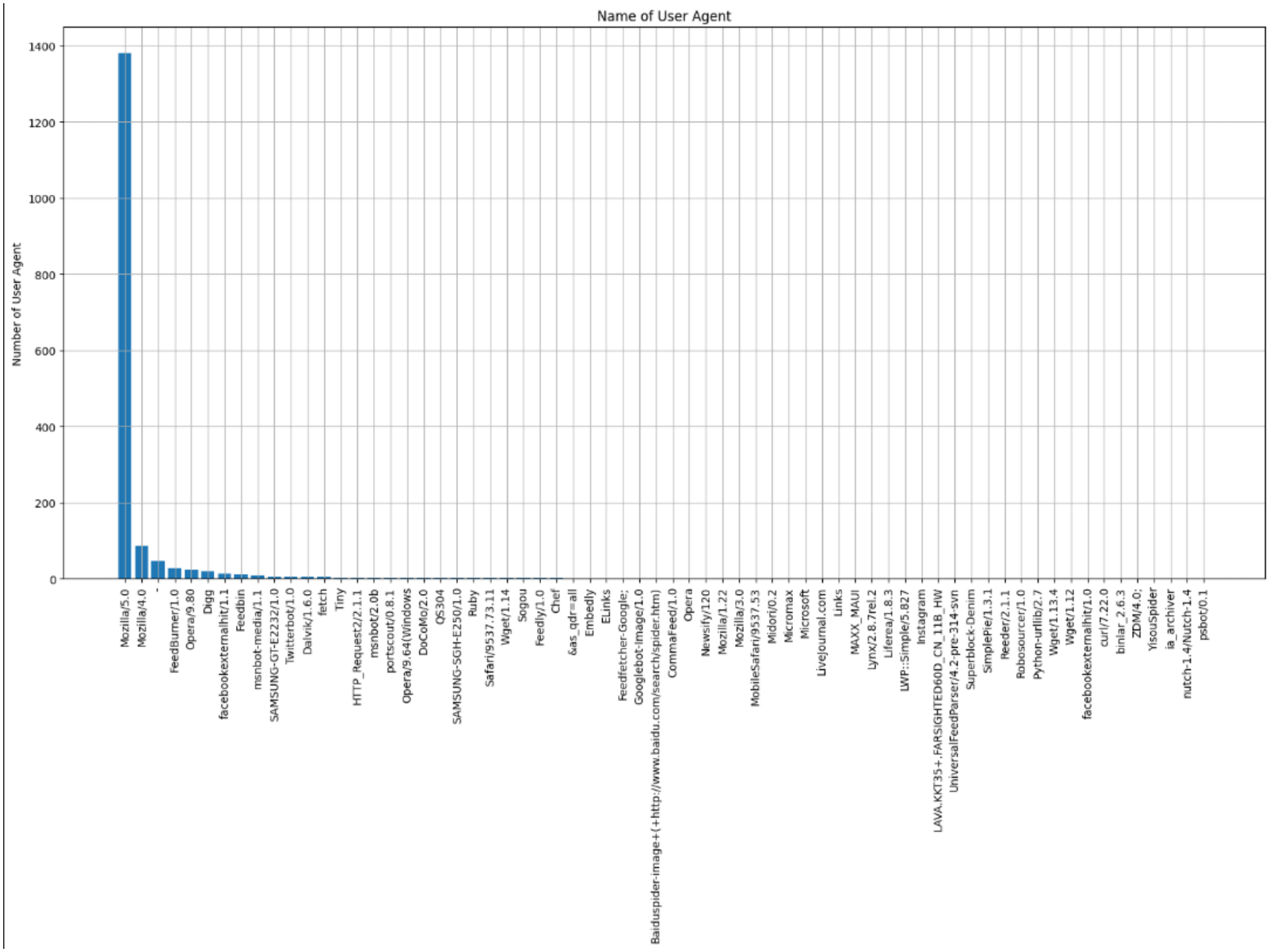
Просто відокремлемо першу частину User-Agent

```
[168]  df['ua_short'] = df['user_agent'].apply(lambda x: x.split(' ')[0])
       ✓ 0.0s

[169]  df_user_agent = df.groupby('ua_short')['ip'].nunique().reset_index().sort_values(by='ip', ascending=False)
       df_user_agent
       ✓ 0.0s
```

	ua_short	ip
31	Mozilla/5.0	1380
30	Mozilla/4.0	86
1	-	47
10	FeedBurner/1.0	28
35	Opera/9.80	24
...	...	...
54	ZDM/4.0;	1
53	YisouSpider	1
60	ia_archiver	1
63	nutch-1.4/Nutch-1.4	1
65	psbot/0.1	1

66 rows × 2 columns

Візуалізація:



c. Ранжувати користувачів за операційними системами

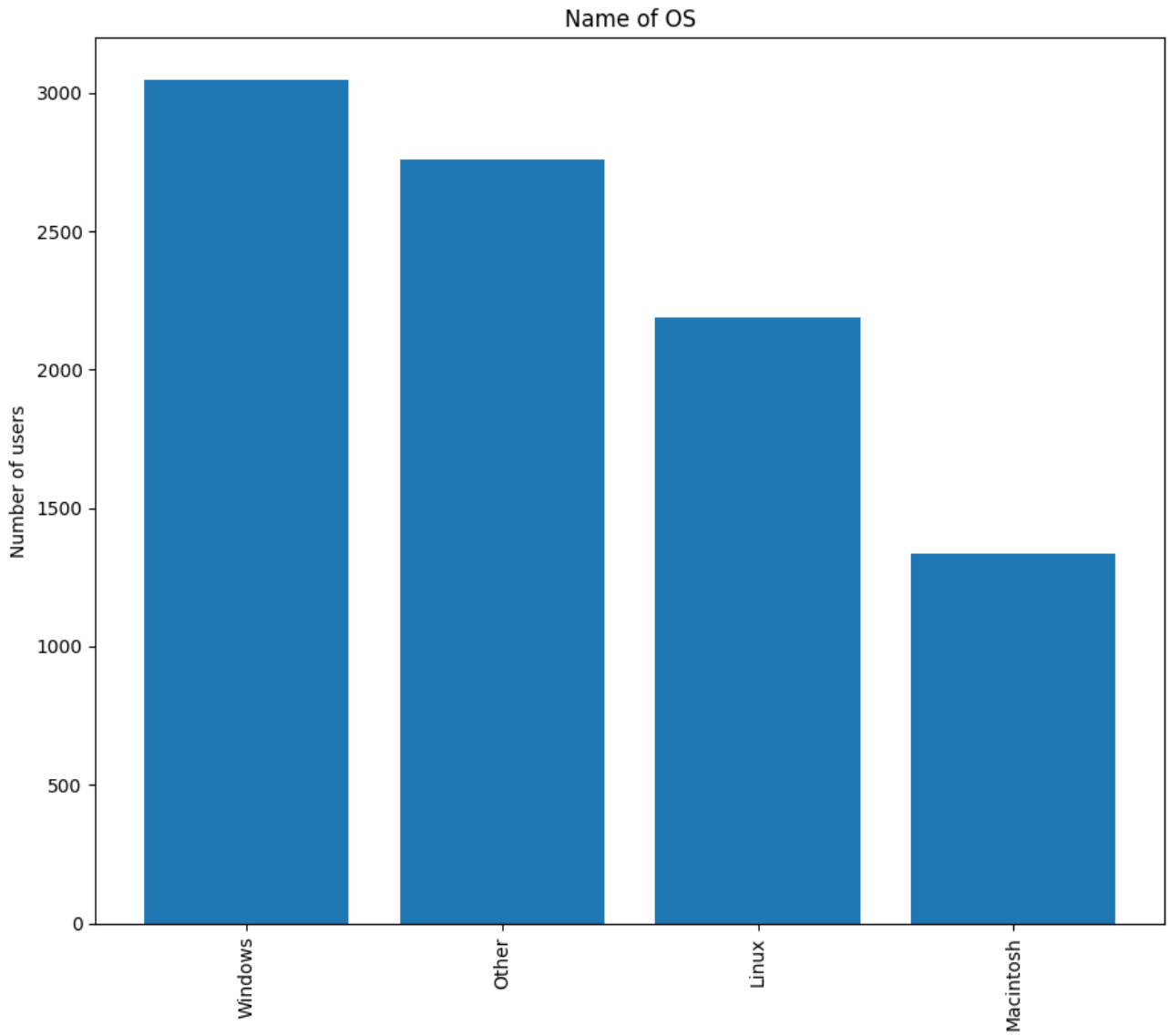
Оберемо найпопулярніші системи:

```
def operation_system(ua_list):
    os = []
    for ua in df['user_agent']:
        if 'Windows' in ua:
            os.append('Windows')
        elif 'Linux' in ua:
            os.append('Linux')
        elif 'Macintosh' in ua:
            os.append('Macintosh')
        else:
            os.append('Other')
    return os

ua_list = df['user_agent'].tolist()
os_unique = operation_system(ua_list)
os_df = pd.Series (os_unique).value_counts().reset_index()
os_df
```

171] ✓ 0.0s

	index	count
0	Windows	3047
1	Other	2760
2	Linux	2189
3	Macintosh	1334

Візуалізація:



d. Ранжувати користувачів за країною запиту

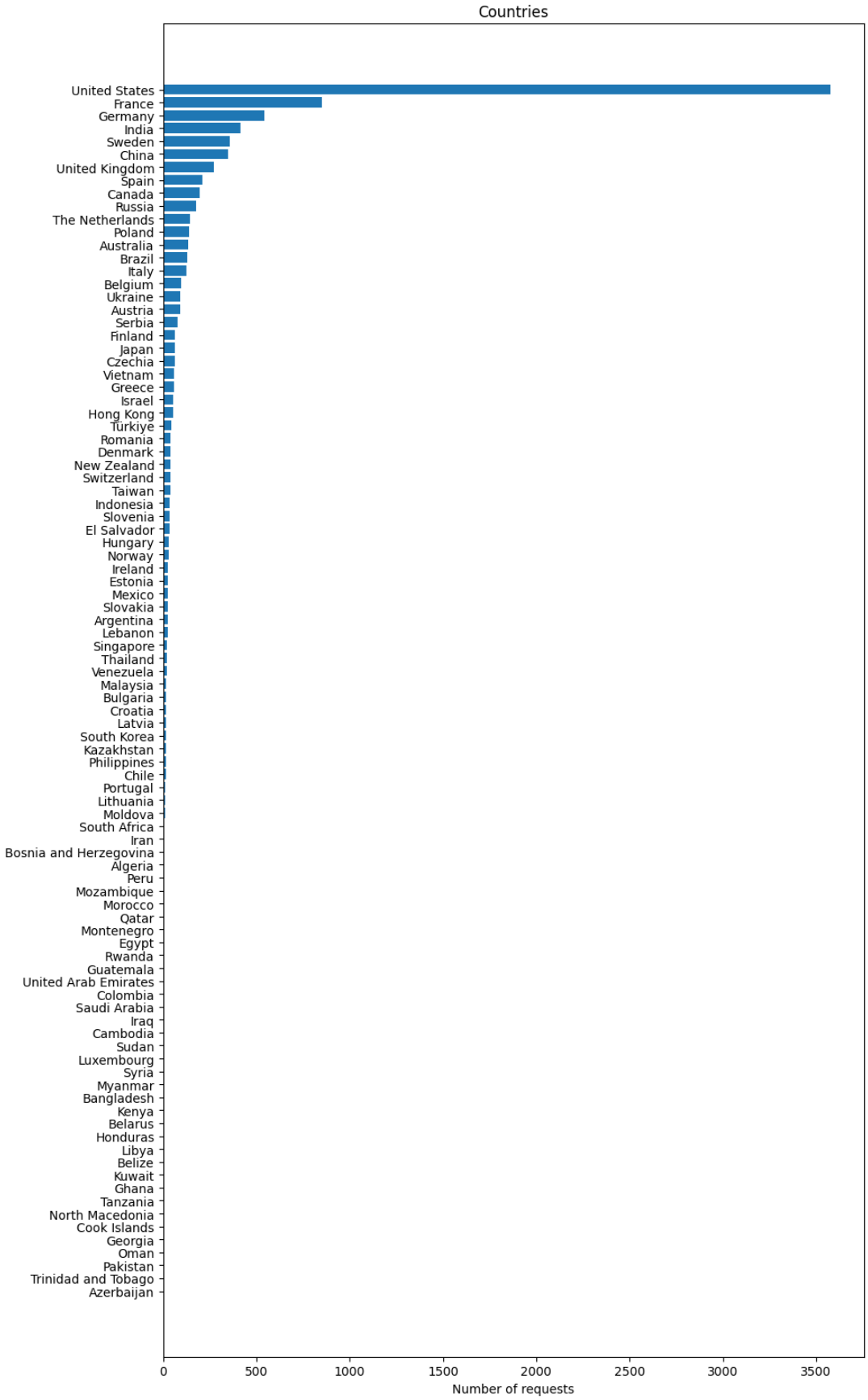
Для цього встановимо модуль geoip2 та завантажемо файл GeoLite2-Country.mmdb.

```
import geoip2.database

reader = geoip2.database.Reader('GeoLite2-Country.mmdb')

def user_country(ip):
    try:
        response = reader.country(ip)
        return response.country.name
    except Exception as e:
        print(f"Error for IP {ip}: {e}")
        return 'Unknown'

df['Country'] = df['ip'].apply(lambda x: user_country(x))
country_count = df['Country'].value_counts()
country_count
```

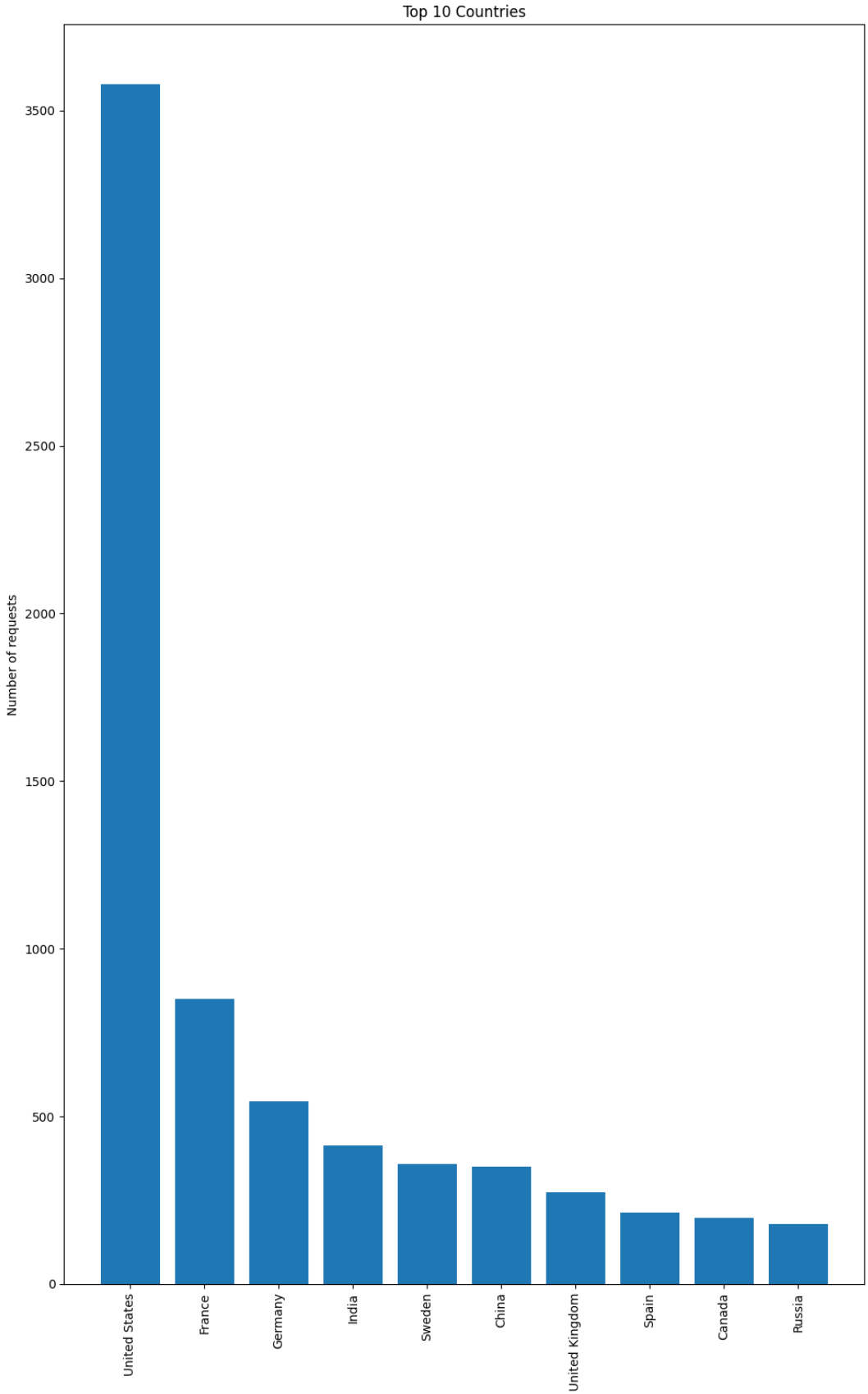
[174] ✓ 0.3s

```
Country
United States          3578
France                  851
Germany                 545
India                   413
Sweden                  358
                       ...
Georgia                   1
Oman                      1
Pakistan                  1
Trinidad and Tobago       1
Azerbaijan                1
Name: count, Length: 94, dtype: int64
```

Візуалізація:



Топ 10 країн:


Top 10 Countries
Number of requests
0
500
1000
1500
2000
2500
3000
3500
United States
France
Germany
India
Sweden
China
United Kingdom
Spain
Canada
Russia

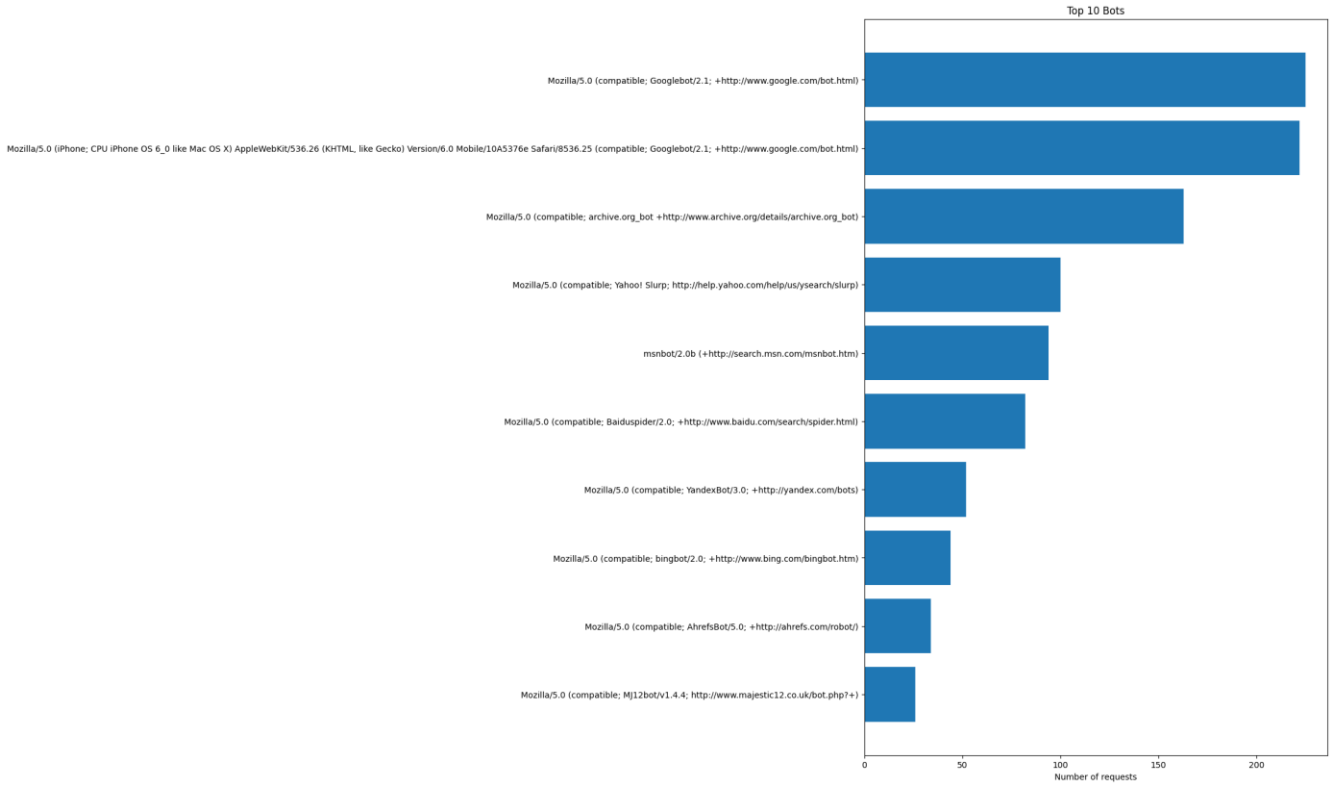
e. Виокремити пошукових ботів

Будемо шукати найпоширеніших ботів.

```
bot_keywords = ['bot', 'spider', 'crawl', 'Googlebot', 'Bingbot', 'Yandex', 'Slurp', 'DuckDuckBot', 'Baiduspider']

bots = df[df['user_agent'].str.contains('|'.join(bot_keywords), case=False, na=False)]
bot_counts = bots['user_agent'].value_counts().head(10)
```

177] ✓ 0.1s

Візуалізація:



f. Детектувати аномалії (якщо такі є)

Аномалії будемо шукати за статус кодами і розмірами запитів.

Для цього виведемо, які статус коди маємо:

```
df['status_code'].unique()
```
182] ✓ 0.0s

```
array(['200', '404', '301', '206', '403', '416', '500'], dtype=object)
```

Далі за допомогою методу кластеризації K-Means спробуємо кластеризувати статус-коди з використанням K-Means і виявлення аномалій на основі відстані до центроїд.

Спочатку стандартизуємо ознаку.

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```
✓ 16.5s

```
scaler = StandardScaler()
status_scaled = scaler.fit_transform(df[['status_code']])
```
✓ 0.0s

```
kmeans = KMeans(n_clusters=4, random_state=42)
df['cluster'] = kmeans.fit_predict(status_scaled)


distances = np.linalg.norm(status_scaled - kmeans.cluster_centers_[df['cluster']], axis=1)
df['distance_to_centroid'] = distances


threshold = np.percentile(distances, 95)
df['is_anomaly'] = df['distance_to_centroid'] > threshold


df[df['is_anomaly']].head(10)
```

На виході маємо такі дані:

	ip	date	request	status_code	size	referrer	user_agent	ua_short	Country	cluster	distance_to_centroid	is_anomaly
62	66.249.73.185	2015-05-17	GET	404	294	-	Mozilla/5.0 (compatible; Googlebot/2.1; +http:...	Mozilla/5.0	United States	1	0.003203	True
173	208.91.156.11	2015-05-17	GET	404	324	-	Chef Client/10.18.2 (ruby-1.9.3-p327; ohai-6.1...	Chef	United States	1	0.003203	True
305	111.199.235.239	2015-05-17	GET	404	364	http://semicomplete.com/presentations/logstash...	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5)...	Mozilla/5.0	China	1	0.003203	True
323	111.199.235.239	2015-05-17	GET	404	364	http://semicomplete.com/presentations/logstash...	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5)...	Mozilla/5.0	China	1	0.003203	True
345	208.91.156.11	2015-05-17	GET	404	324	-	Chef Client/10.18.2 (ruby-1.9.3-p327; ohai-6.1...	Chef	United States	1	0.003203	True
366	144.76.194.187	2015-05-17	GET	404	292	-	-	-	Germany	1	0.003203	True
367	144.76.194.187	2015-05-17	GET	404	303	-	-	-	Germany	1	0.003203	True
413	173.252.73.114	2015-05-17	GET	206	97173	-	facebookexternalhit/1.1 (+http://www.facebook....	facebookexternalhit/1.1	United States	0	0.181629	True
579	89.2.87.1	2015-05-17	GET	206	65536	http://www.google.fr/url?sa=t&rct=j&q=&esrc=s&...	Mozilla/5.0 (Windows NT 5.1; rv:26.0) Gecko/20...	Mozilla/5.0	France	0	0.181629	True
580	89.2.87.1	2015-05-17	GET	206	55278	http://www.google.fr/url?sa=t&rct=j&q=&esrc=s&...	Mozilla/5.0 (Windows NT 5.1; rv:26.0) Gecko/20...	Mozilla/5.0	France	0	0.181629	True

Візуалізація:

Більшість точок на графіку мають статус-код 200 і позначені синім кольором - це означає, що запити були успішними і вважаються нормальними. Червоні точки вказують на аномальні значення, які відрізняються від загальної маси. Серед них зустрічаються такі коди, як 404 (не знайдено), 403 (заборонено), 301 (перенаправлення), 206 (частковий вміст), 416 (некоректний діапазон) і навіть 500 (внутрішня помилка сервера). Наявність цих кодів серед аномалій вказує на

потенційні проблеми - наприклад, помилки в роботі серверу, неправильні або підозрілі запити, можливу активність ботів чи сканерів. Також деякі аномалії мають великі розміри відповіді, що може бути нехарактерним для відповідних кодів і свідчити про спроби отримати заборонену або неіснуючу інформацію.

Висновки: в ході виконання лабораторної роботи було проаналізовано лог-файл веб-сервера, сформовано датасет із ключовою інформацією про користувачів, здійснено аналіз за User-Agent, операційними системами, країнами запитів та виявлено пошукових ботів. Особливу увагу було приділено виявленню аномалій за допомогою кластеризації методом K-Means. Аналіз показав, що більшість запитів були успішними (код 200), однак також були виявлені аномальні статус-коди, серед яких 404, 403, 301, 206, 416, 500, що може свідчити про наявність технічних збоїв, помилкових запитів або потенційно підозрілу активність. Отримані результати підтверджують доцільність використання методів аналізу логів для покращення безпеки та стабільності веб-ресурсів.