Лабораторна 1

«WEB-аналітика»

ФБ-31мп Синчук Владислав

Завдання: На основі будь-якого access.log сформувати датасет, що надав би інформацію про користувачів веб-ресурсу

Виконати наступні кроки:

Функції для парсингу та перетворення файлу логів в датафрейм:

```
def parse_line(line: str) -> dict | None:
    regex = r'(?P<ip>\d+\.\d+\.\d+\.\d+) - - \[(?P<timestamp>.*?)\]
"(?P<request>.*?)" (?P<status>\d+) (?P<bytes_sent>\d+)
"(?P<referrer>.*?)" "(?P<user_agent>.*?)"'
    match = re.match(regex, line)
    return match.groupdict() if match else None


def parse_file(filepath: str) -> pd.DataFrame:

    with open(filepath, 'r') as file:
        df_rows = []
        for line in file:
            row = parse_line(line)
            # print(row)
            if not(row is None):
                df_rows.append(row)

    df = pd.DataFrame(df_rows)
    # етап обробки датафрейму
    df['timestamp'] = pd.to_datetime(df['timestamp'],
    format='%d/%b/%Y:%H:%M:%S %z', errors='coerce')
    df['status'] = pd.to_numeric(df['status'], errors='coerce')
    df['bytes_sent'] = pd.to_numeric(df['bytes_sent'],
errors='coerce')
    df['date'] = df['timestamp'].dt.date
    df['endpoint'] = df['request'].apply(lambda x: x.split()[1])
    return copy.deepcopy(df)
```

Виконаємо функції

```
# ======= task a
    filepath = 'lab-1/access.log'
```

```
df = parse_file(filepath)
df.to_csv('lab-1/access_log.csv')
```

Прочитаємо та переглянемо збережений датафрейм:

```
[16] 1 filename = '/content/drive/MyDrive/web-analytics-synchuk/access_log.csv'
     2 df = pd.read_csv(filename)

     1 df.head()
```

	Unnamed: 0	ip	timestamp	request	status	bytes_sent	referrer	user_agent	date	endpoint
0	0	1.202.218.8	2012-06-20 19:05:12+02:00	GET /robots.txt HTTP/1.0	404	492	-	\	2012-06-20	/robots.txt
1	1	208.115.113.91	2012-06-20 19:20:16+02:00	GET /logs/?C=M;O=D HTTP/1.1	200	1278	-	Mozilla/5.0 (compatible; Ezooms/1.0; ezooms.bo...	2012-06-20	/logs/?C=M;O=D
2	2	123.125.71.20	2012-06-20 19:30:40+02:00	GET / HTTP/1.1	200	912	-	Mozilla/5.0 (compatible; Baiduspider/2.0; +htt...	2012-06-20	/
3	3	220.181.108.101	2012-06-20 19:31:01+02:00	GET / HTTP/1.1	200	912	-	Mozilla/5.0 (compatible; Baiduspider/2.0; +htt...	2012-06-20	/
4	4	123.125.68.79	2012-06-20 19:53:24+02:00	GET / HTTP/1.1	200	625	-	Mozilla/5.0 (compatible; Baiduspider/2.0; +htt...	2012-06-20	/

a) Визначити кількість користувачів за днями

```
[19] 1 # кількість користувачів за днями
     2 df.groupby('date')['ip'].nunique()
```

```
date
2012-06-20    21
2012-06-21    69
2012-06-22    68
2012-06-23    83
2012-06-24    78
2012-06-25    73
2012-06-26    90
2012-06-27    73
2012-06-28    88
2012-06-29    93
2012-06-30    98
2012-07-01    82
2012-07-02    30
Name: ip, dtype: int64
```

b) Ранжувати користувачів за User-Agent

```
1 # ранжувати користувачів за user-agent
2 df.groupby('user_agent')['ip'].nunique().sort_values(ascending=False)
```

```
user_agent
Mozilla/5.0 (compatible; Baiduspider/2.0; +http://www.baidu.com/search/spider.html)                    193
Mozilla/5.0 (compatible; MJ12bot/v1.4.3; http://www.majestic12.co.uk/bot.php?+)                         32
Mozilla/5.0 (compatible; AhrefsBot/3.1; +http://ahrefs.com/robot/)                                      22
Mozilla/5.0 (Windows NT 5.1; rv:6.0.2) Gecko/20100101 Firefox/6.0.2                                     15
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)                                13
                                                                                                      ...
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.1 (KHTML, like Gecko) Chrome/14.0.814.0 Safari/535.1    1
Mozilla/5.0 (Windows NT 6.1) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.56 Safari/536.5     1
Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.1 Safari/535.19    1
Mozilla/5.0 (Windows NT 6.0; rv:12.0) Gecko/20100101 Firefox/12.0                                       1
siclab (cboc-test@lab.ntt.co.jp)                                                                        1
Name: ip, Length: 101, dtype: int64
```

c) Ранжувати користувачів за операційними системами

```
# ранжувати користувачів за операційними системами
# ex. : Mozilla/5.0 (Windows NT 5.1; rv:6.0.2)
Gecko/20100101 Firefox/6.0.2
# =>
Windows                                          15
def extract_parentheses_content(user_agent_str:
str) -> str | None:
    pattern = r'\((.*?)\)'
    match_ = re.search(pattern, user_agent_str)
    return match_.group(1) if match_ else None


def identify_os(content: str) -> str:
    # Define patterns for known operating systems
    os_patterns = {
        'Windows': r'Windows',
        'Mac OS': r'Macintosh|Mac OS|Mac_PowerPC',
        'Linux': r'Linux',
        'Android': r'Android',
        'iOS': r'iPhone|iPad|iPod',
        'Chrome OS': r'CrOS'
    }

    for os, pattern in os_patterns.items():
        if re.search(pattern, content,
re.IGNORECASE):
            return os
    return 'Unknown'

def extract_os_from_user_agent(user_agent_str: str)
-> str:
    content =
str(extract_parentheses_content(user_agent_str))
    return identify_os(content)
```

```
df['os'] =
df['user_agent'].apply(extract_os_from_user_agent)
df.groupby('os')['ip'].nunique()
```

```
os
Linux       10
Mac OS       2
Unknown    329
Windows    119
Name: ip, dtype: int64
```

d) Ранжувати користувачів за країною запиту

```
# Ранжувати користувачів за країною запиту
import geoip2.database
# reader =
geoip2.database.Reader('/content/drive/MyDrive/web-
analytics-synchuk/GeoLite2-City.mmdb')
reader =
geoip2.database.Reader('/content/drive/MyDrive/web-
analytics-synchuk/GeoLite2-Country.mmdb')

# def get_country_from_ip(ip: str) -> str:
#     response =
requests.get(f"https://geolocation-
db.com/json/{ip}&position=true").json()
#     # print(response)
#     return response['country_name']

def get_country_from_ip(ip: str, reader) -> str:
    try:
        response = reader.country(ip)
        return response.country.name
    except Exception:
        return 'Unknown'


df['country'] = df['ip'].apply(lambda ip:
get_country_from_ip(ip, reader))
```

```
df.groupby('country')['ip'].nunique().sort_values(ascending=False)
```

```
country
China             193
United States     102
Japan              24
Russia             22
France             14
Germany            14
Ukraine            12
The Netherlands     6
United Kingdom      5
Brazil              5
Czechia             5
Indonesia           5
India               4
Israel              4
Norway              4
Romania             2
Sweden              2
Ecuador             2
Canada              2
Thailand            2
Malaysia            2
Argentina           1
Türkiye             1
Sri Lanka           1
```

е) Виокремити пошукових ботів

```
# Виокремити пошукових ботів

known_bots = ['Googlebot', 'Bingbot', 'Yahoo! Slurp', 'DuckDuckBot', 'Baiduspider']
# Identify bots in the User-Agent column and create a new 'Bot' column
df['bot'] = df['user_agent'].apply(
    lambda user_agent: next((bot for bot in known_bots if bot in user_agent), None)
)
df.groupby('bot')['ip'].nunique().sort_values(ascending=False)
```

Loading...

```
bot
Baiduspider    206
Googlebot       13
Name: ip, dtype: int64
```

Детектувати аномалії (якщо такі є)

```
from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

X = df[['hour', 'bytes_sent']]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
kmeans = KMeans(n_clusters=4)
df['cluster'] = kmeans.fit_predict(X_scaled)

plt.figure(figsize=(10, 6))
```

```
for cluster_label in df['cluster'].unique():
    cluster_data = df[df['cluster'] ==
cluster_label]

    plt.scatter(cluster_data['hour'],
cluster_data['bytes_sent'],
    label=f'Cluster {cluster_label}')

plt.xlabel('Hour of the Day')
plt.ylabel('Bytes Sent')
plt.title('Clusters of Data')
plt.legend()
plt.grid(True)
plt.show()
```