Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Лабораторна робота №2
з дисципліни
«Web - аналітика»

Виконав:
студент групи ФБ-31мп
Щур Павло
Перевірив:
Ткач В. М.

Київ-2024

Посилання на GitHub: https://github.com/ShchurPavlo/web-analytics-2024/tree/main/lab2

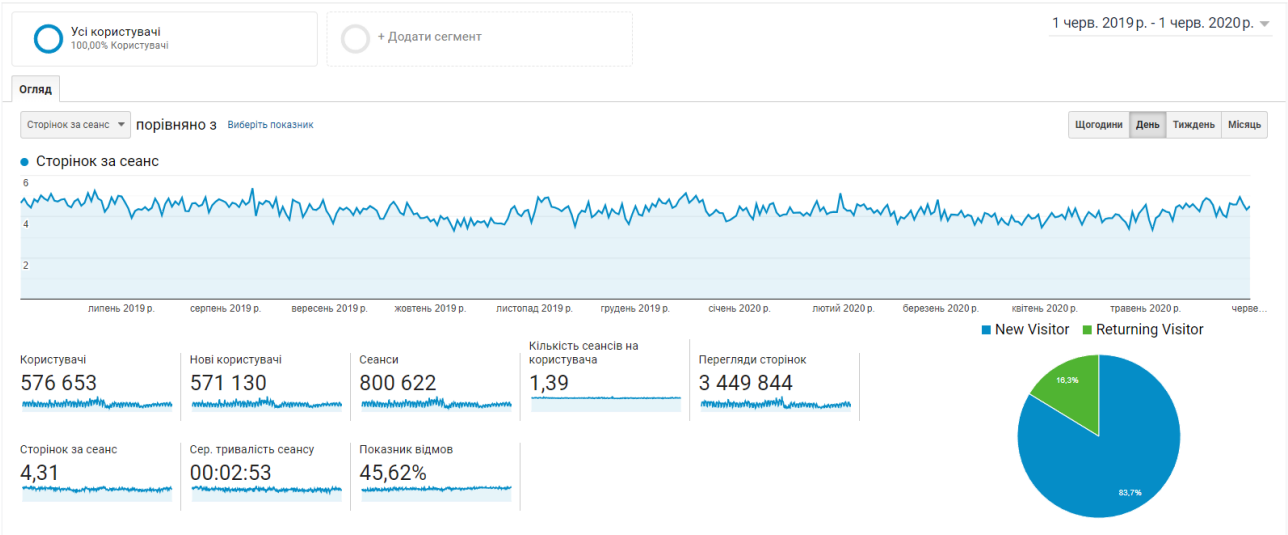
Завдання

1. На основі даних з Google Analytics Demo account вибрати ТРИ різні часові ряди і на їх основі шляхом застосування методів визначення аномалій визначити аномалії в поведінці системи.

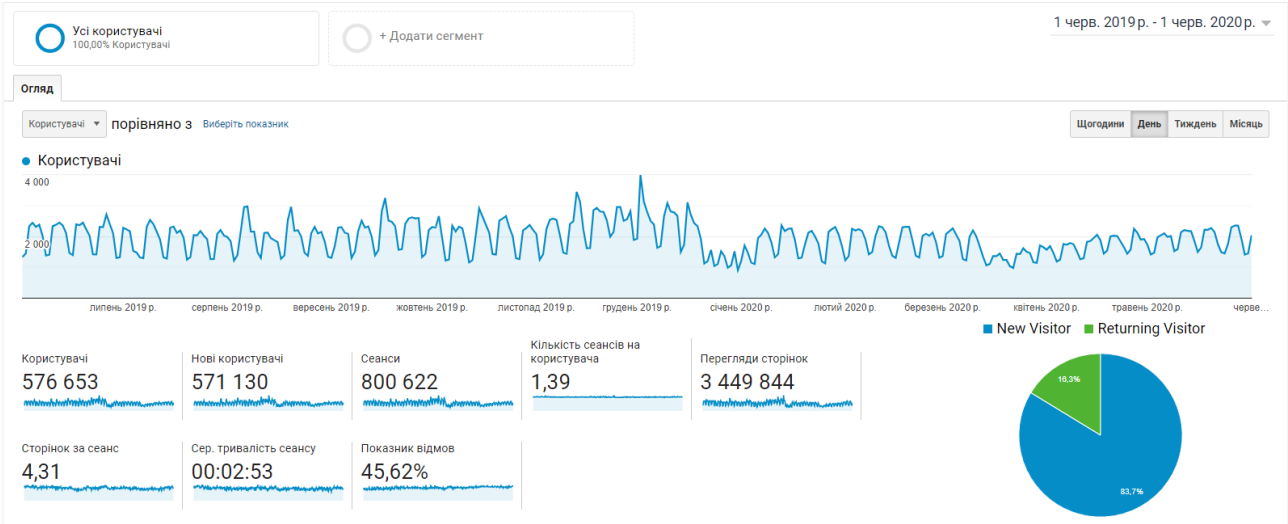
Виконання:

1) На Google Analytics на інтервалі 1 червня 2019 – 1 червня 2020 оберемо 3 показники:

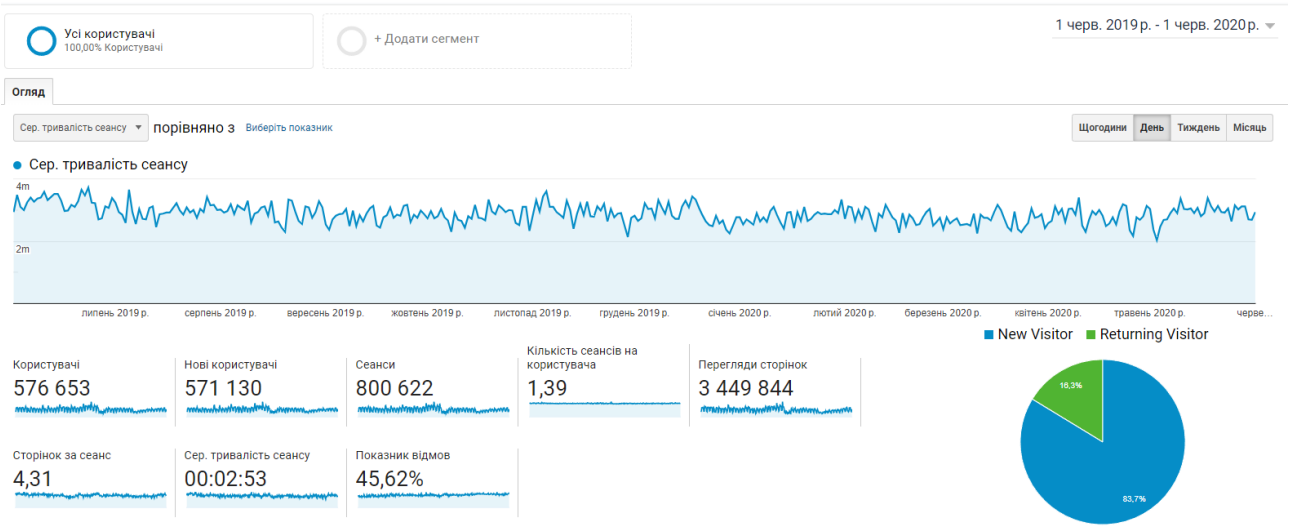
Кількість сторінок відвіданих за сеанс:



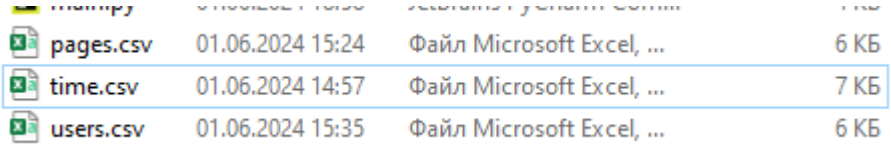
Кількість користувачів:



Середня тривалість сеансу:



Завантажимо ці датасети в форматі csv:



2) З файлів датасету отримаємо дані та переведемо їх в датафрейм pandas для простішої обробки:

Для датафрейму кількості користувачів переведемо кількість в формат int та відформатуємо дату:

```
users_df=pandas.read_csv( filepath_or_buffer:'users.csv',delimiter=',', names=['Date', 'Users'])
users_df['Users'] = users_df['Users'].astype(int)
users_df['Date'] = pandas.to_datetime(users_df['Date'], format="%d.%m.%y")
#print(users_df.head)
```

В результаті отримаємо:

	Date	Users
0	2019-06-01	1334
1	2019-06-02	1448
2	2019-06-03	2325
3	2019-06-04	2441
4	2019-06-05	2308
..	...	...
362	2020-05-28	2352
363	2020-05-29	1920
364	2020-05-30	1411
365	2020-05-31	1450
366	2020-06-01	2036

Для датасету з тривалістю сеансу проведемо схожі маніпуляції та переведемо числовий показник в секунди:

```
time_df = pandas.read_csv( filepath_or_buffer: "time.csv",delimiter=',', names=['Date', 'Duration'])
time_df['Date'] = pandas.to_datetime(time_df['Date'], format="%d.%m.%y")
time_df['Duration'] = pandas.to_datetime(time_df['Duration'], format='%H:%M:%S')
time_df['Duration'] =time_df['Duration'].dt.hour * 3600 + time_df['Duration'].dt.minute * 60 + time_df['Duration'].dt.second
print(time_df)
```

	Date	Duration
0	2019-06-01	176
1	2019-06-02	209
2	2019-06-03	186
3	2019-06-04	180
4	2019-06-05	195
..	...	...
363	2020-05-29	187
364	2020-05-30	162
365	2020-05-31	161
366	2020-06-01	176
367	NaT	173

Датасет з кількістю сторінок опрацюємо ідентично, за виключенням того що числове значення переведемо у float формат:

```
pages_df = pandas.read_csv( filepath_or_buffer: "pages.csv",delimiter=',', names=['Date', 'Pages'])
pages_df['Date'] = pandas.to_datetime(pages_df['Date'], format="%d.%m.%y")
pages_df['Pages'] = pages_df['Pages'].astype(float)
print(pages_df)
```

[368 rows x 2 columns]

	Date	Pages
0	2019-06-01	4.67
1	2019-06-02	4.88
2	2019-06-03	4.60
3	2019-06-04	4.44
4	2019-06-05	4.84
..	...	...
363	2020-05-29	4.95
364	2020-05-30	4.61
365	2020-05-31	4.34
366	2020-06-01	4.51

2) Для визначення аномальних показників будемо використовувати наступні методи:

- Z-score методика

Вона ґрунтується на вимірюванні відстані від кожного значення до середнього значення в одиницях стандартного відхилення. Реалізуємо окрему функцію для обрахунку:

```
49      def Calculate_z_score(df):
50          mean_val = df.iloc[:, 1].mean()
51          std_dev = df.iloc[:, 1].std()
52          df['Z_score'] = (df.iloc[:, 1] - mean_val) / std_dev
53          return df
```

Для стовпця числових даних обчислюється середнє значення (mean) і стандартне відхилення (std). Значення, для яких абсолютне значення Z-score перевищує встановлений поріг, вважаються аномальними.

- IQR (Interquartile Range) – метод

Вона використовує міжквартильний діапазон для визначення нижньої та верхньої меж аномалій на основі розподілу даних. Спочатку обчислюються перший (Q1) та третій (Q3) квартилі даних, вираховується міжквартильний діапазон та на основі цього діапазону визначаються граничні значення. Всі маніпуляції здійснюються в окремій функції:

```
55      def Calculate_IQR(df):
56          a = 1.2
57          q1 = df.iloc[:, 1].quantile(0.25)
58          q3 = df.iloc[:, 1].quantile(0.75)
59          iqr=q3-q1
60          lower_bound = q1 - (a * iqr)
61          upper_bound = q3 + (a * iqr)
62          print(lower_bound)
63          print(upper_bound)
64          df_anomalies = df[(df.iloc[:, 1] < lower_bound)| (df.iloc[:, 1] > upper_bound)]
65          return df_anomalies
```

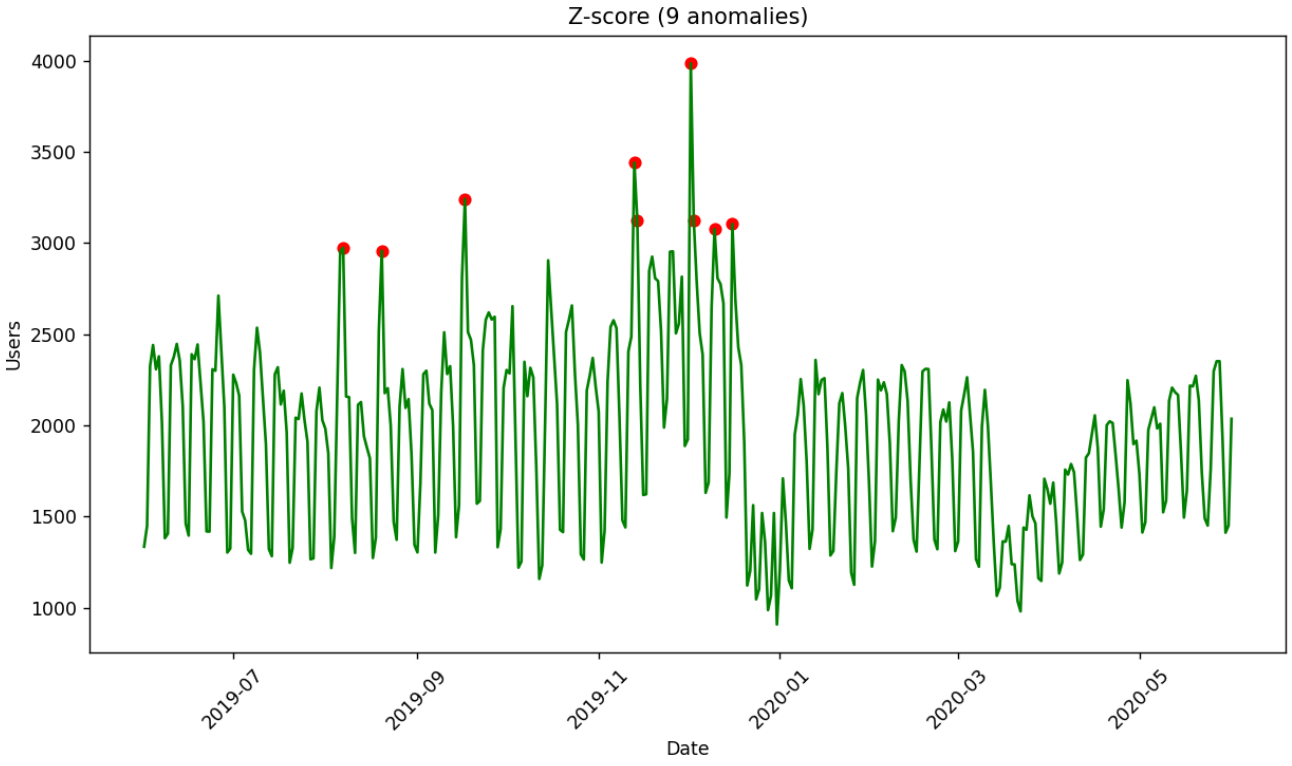
- Local Outlier Factor (LOF)

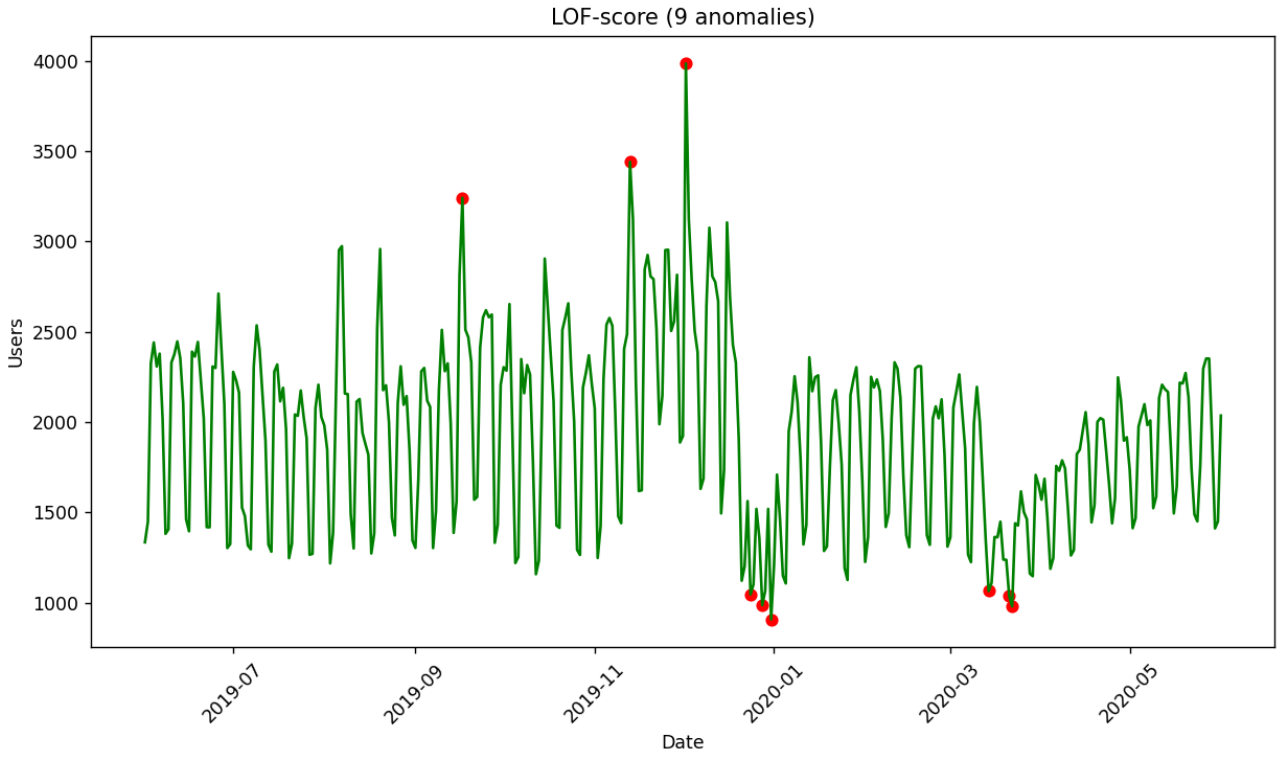
Використовує поняття щільності точок, щоб визначити, чи є точка аномальною відносно своїх сусідів. LOF порівнює щільність даної точки з щільністю її сусідів; якщо точка має значно нижчу щільність у порівнянні з сусідами, вона вважається аномалією.

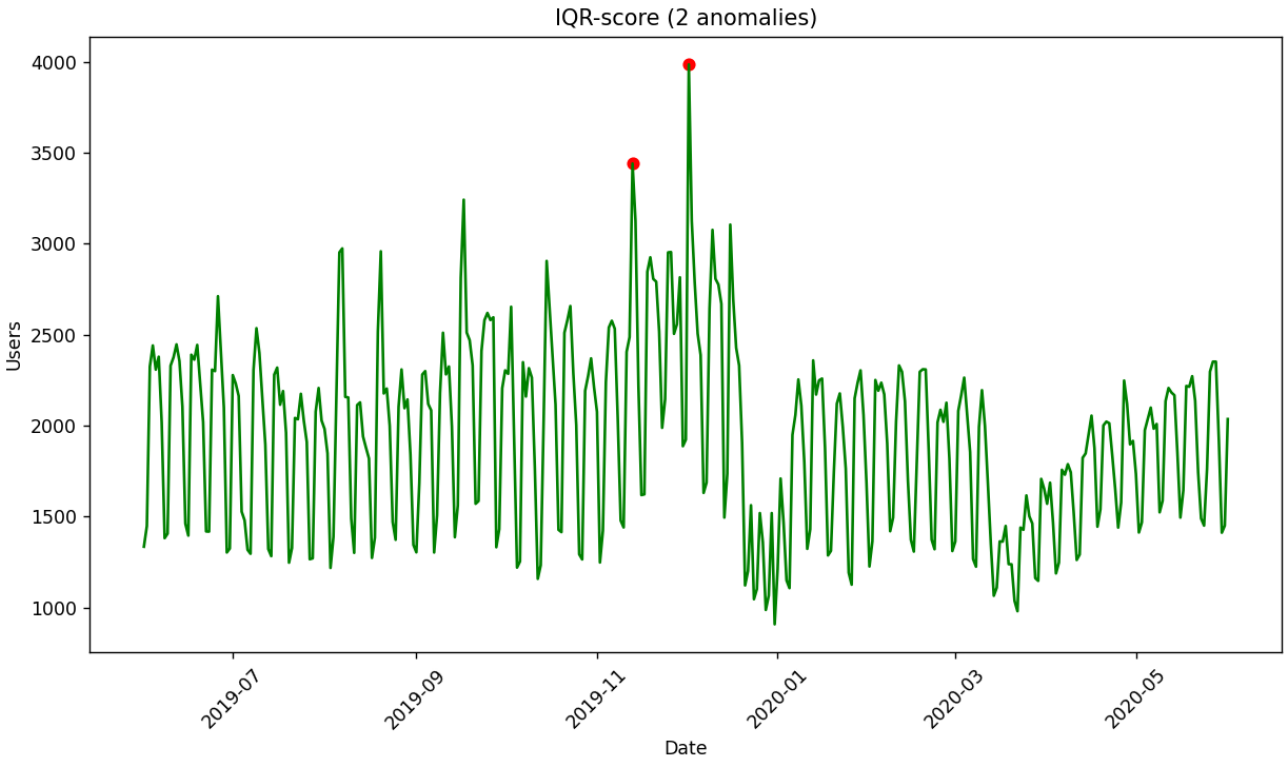
```
42  def Calculate_LOF(df):
43      X = df.iloc[:, 1].values.reshape(-1, 1)
44      lof = LocalOutlierFactor(n_neighbors = 19)
45      anomaly_scores = lof.fit_predict(X)
46      df['LOF_score'] = anomaly_scores
47      return df
```

3) В результаті для кожного з методів отримаємо:

Для датафрейму з кількістю користувачів:

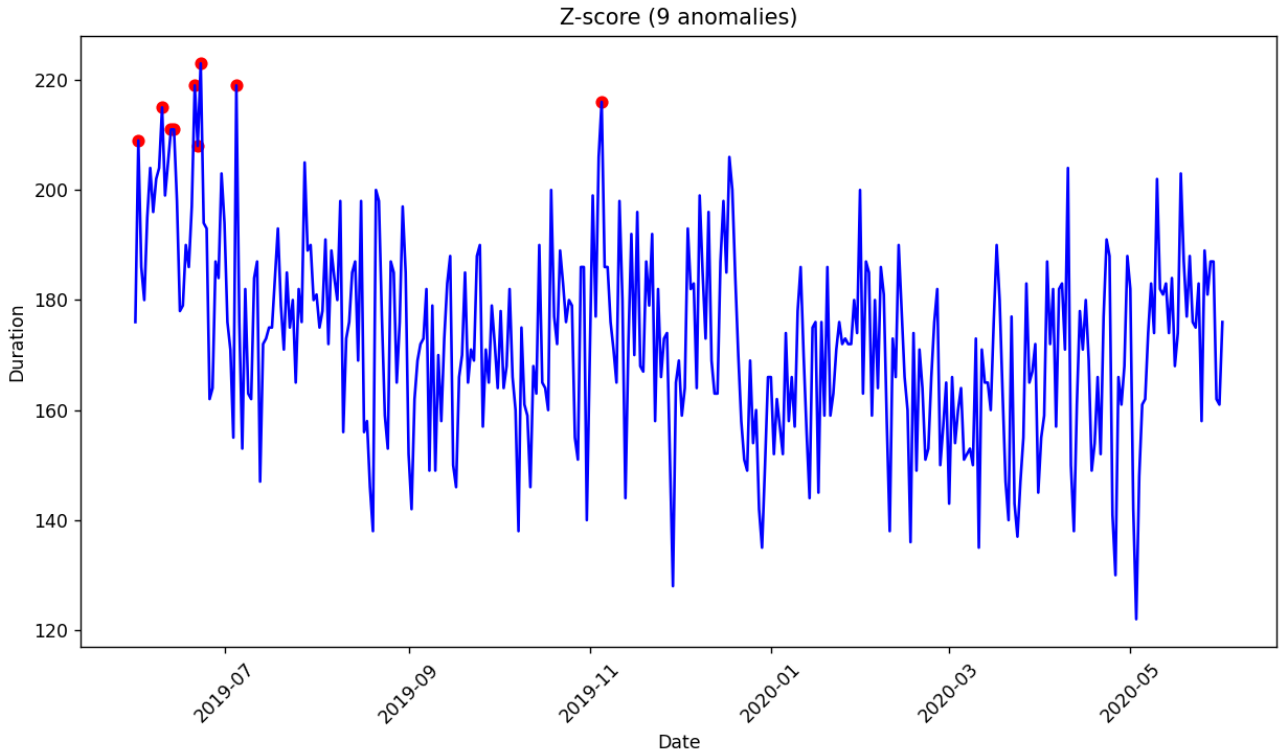


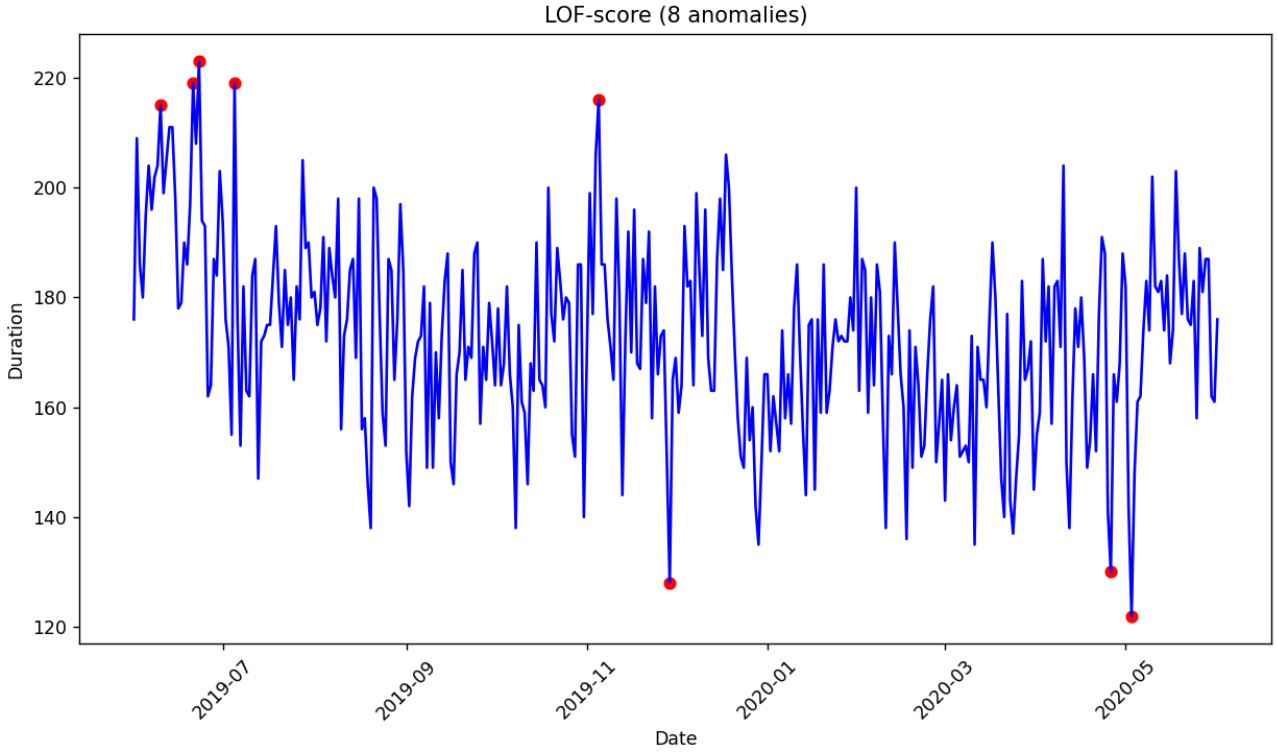


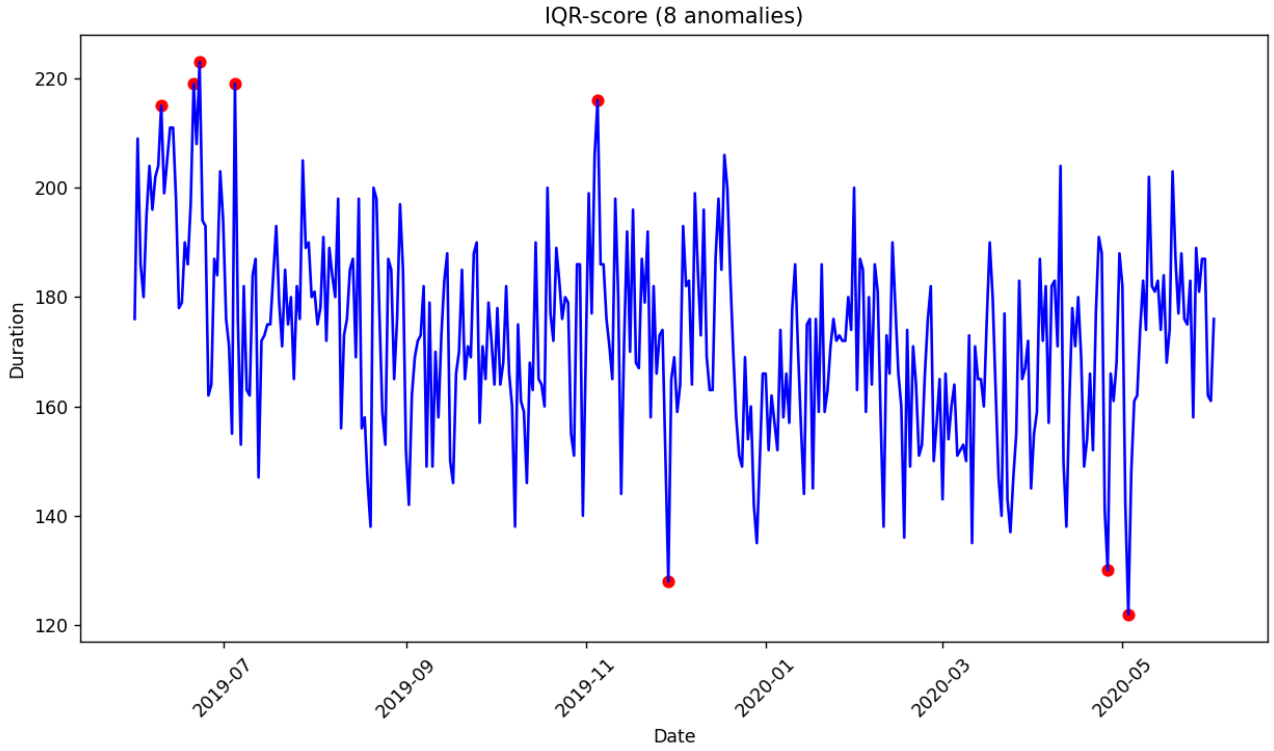


Згідно отриманих графіків найкраще себе показав LOF-метод: від ідентифікував як аномально високі так і аномально низькі показники.

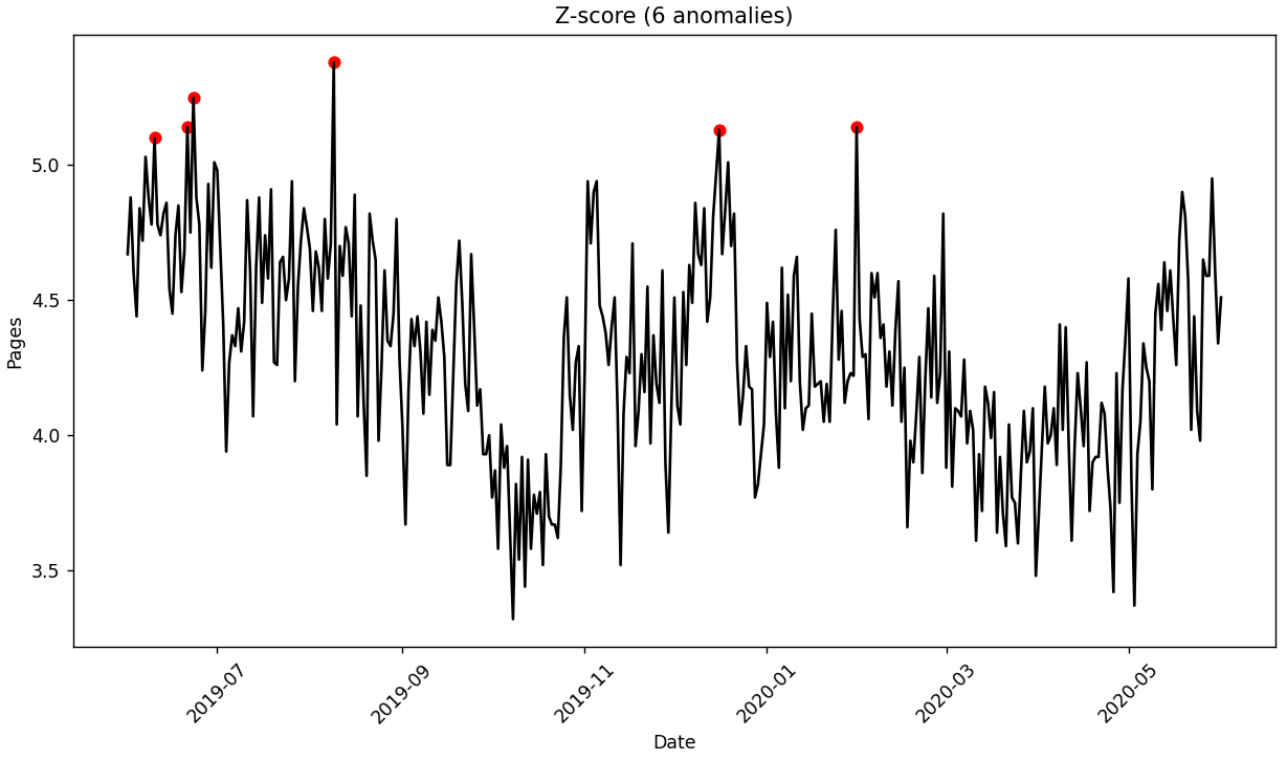
Для датасету з тривалістю сесій результат роботи виглядає наступним чином:

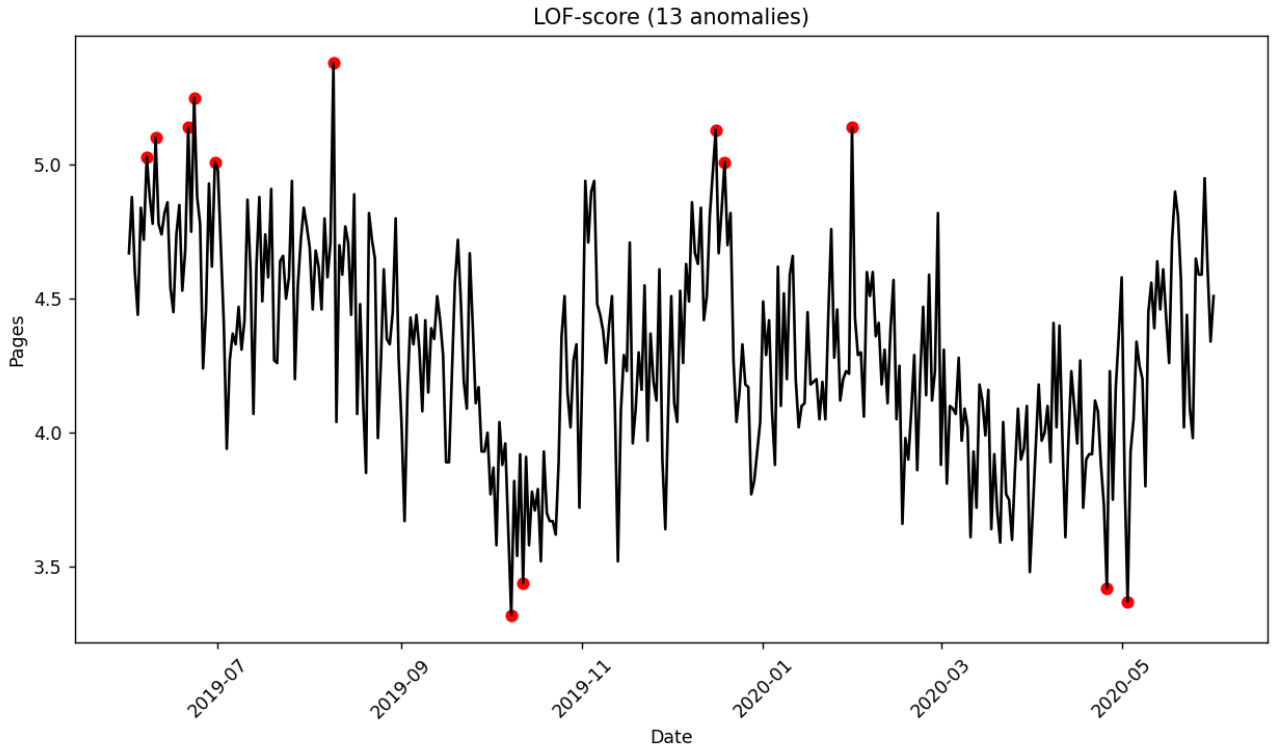


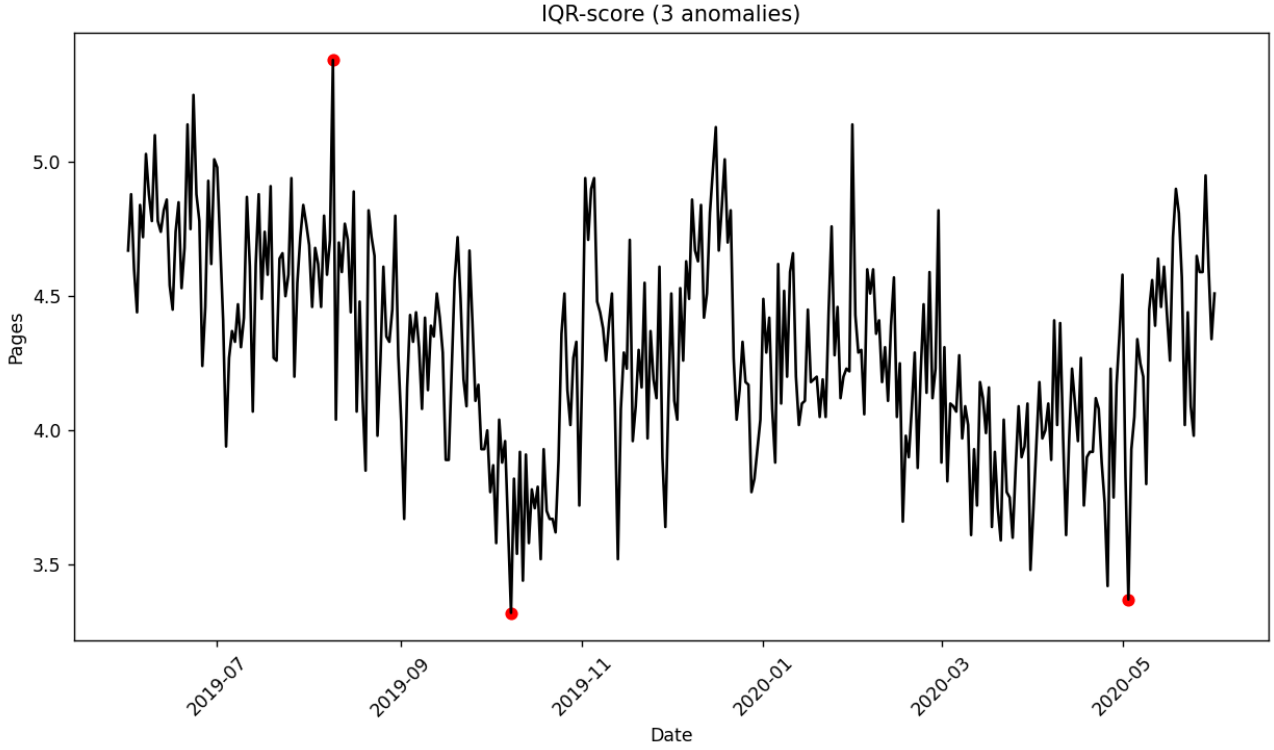




При аналізі датасету з кількістю







Згідно отриманих графіків можна зробити висновок що найбільш обширну картину дає LOF методі визначення аномалій. IQR метод дозволяє точніше ідентифікувати аномалії та максимізує границю входження в аномальну зону.