from selenium import webdriver

from selenium.webdriver.common.by import By

import csv

import time

# Запуск нового экземпляра браузера Chrome

browser = webdriver.Chrome()

# Переход на первую страницу веб-сайта

browser.get("https://www.spaceweatherlive.com/ru/arhiv/2011/01/01/xray.html")

# Инициализация пустого списка для хранения цитат

ratyng\_year = []

count = 1

while True:

# Поиск всех цитат на странице с помощью xpath

    quote\_elements = browser.find\_elements(By.XPATH,'//div[contains(@class,"col-md-8 pb-5")]')

# Извлечение текста каждой цитаты

    for quote\_element in quote\_elements:

        date = quote\_element.find\_element(By.XPATH,'.//h3').text

        power = quote\_element.find\_element(By.XPATH,'.//div[8]/div/table/tbody/tr/td[4]').text

        ratyng\_year.append({"date": date, "power": power})

# Проверка наличия следующей кнопки

    next\_button = browser.find\_elements(By.XPATH,'//div[contains(@class,"btn-group")]/div/a[1]')

# Нажатие следующей кнопки

    next\_button[0].click()

# Ожидание загрузки страницы

    time.sleep(1)

    count += 1

    if count > 31:

        break

# Запись данных в файл CSV

with open("ratyng\_1.csv", "w", encoding='UTF-8', newline="") as file:

    writer = csv.DictWriter(file, fieldnames=["date", "power"])

    writer.writeheader()

    writer.writerows(ratyng\_year)

# Закрытие браузера

browser.close()

import pandas as pd

import numpy as np

from scipy import stats

from scipy.stats import spearmanr

from sklearn.preprocessing import LabelEncoder

import seaborn as sns

import matplotlib.pyplot as plt

from itertools import product

import datetime

Подготовка датасета по солнечной активности

Считываем и объединяем датасеты за 11 месяцев 2011 года

files = ['ratyng\_1.csv', 'ratyng\_2.csv', 'ratyng\_3.csv', 'ratyng\_4.csv', 'ratyng\_5.csv', 'ratyng\_6.csv', 'ratyng\_7.csv', 'ratyng\_8.csv', 'ratyng\_9.csv', 'ratyng\_10.csv', 'ratyng\_11.csv']

dfs = [pd.read\_csv(file) for file in files]

final\_df = pd.concat(dfs)

final\_df

final\_df.info()

Добавляем id

from uuid import uuid4

final\_df['uuid'] = final\_df.index.to\_series().map(lambda x: uuid4())

final\_df.head()

new\_df = final\_df['date'].str.split(expand=True)

new\_df.head()

new\_df.columns=['1', '2', '3', '4', 'dey', 'month', 'year']

new\_df.head()

final\_df1 = pd.concat([final\_df,new\_df],axis=1)

final\_df1.head()

Удаление лишних столбцов

final\_df1.drop(final\_df1.columns[[0, 2, 3, 4, 5, 6]], axis=1, inplace=True)

final\_df1.head()

Меняем название месяца на цифровое представление

d = {'января': 1, 'февраля': 2, 'марта': 3, 'апреля': 4, 'мая': 5, 'июня': 6, 'июля': 7, 'августа': 8, 'сентября': 9, 'октября': 10, 'ноября': 11,}

final\_df1.month = final\_df1.month.map(d)

final\_df1.head()

final\_df1.info()

final\_df1[['dey', 'month', 'year']] = final\_df1[['dey', 'month', 'year']].astype (int)

final\_df1.head()

Подготовка данных для пребразование в формат datetime"

import io

final\_df1["month"] = final\_df1.month.map("{:02}".format)

final\_df1["dey"] = final\_df1.dey.map("{:02}".format)

final\_df1.head()

final\_df1[['dey', 'month', 'year']] = final\_df1[['dey', 'month', 'year']].astype (str)

final\_df1.head()

final\_df1.info()

final\_df1['date'] = final\_df1['year'].map(str) + final\_df1['month'].map(str) +  final\_df1['dey'].map(str)

final\_df1.head()

Конвертация в формат Datetime

final\_df1['date'] = pd.to\_datetime(final\_df1['date'], format='%Y%m%d')

final\_df1.head()

Подготовка датасета по дорожно-транспортные происшествия в Барселоне

df = pd.read\_csv('accidents\_opendata.csv')

df.head()

df.info()

Удаление лишних строк. Оставляем данные за 2011 год.

df1 = df[df['year'] > 2010]

df2 = df1[df1['year'] < 2012]

Удаление лишних столбцов

df2.drop(df2.columns[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26]], axis=1, inplace=True)

df2.info()

df2.head()

df2[['year', 'month', 'day']] = df2[['year', 'month', 'day']].astype (int)

df2.info()

Подготовка данных для пребразование в формат datetime

Добавление нуля перед днём и месяцем прописаных 1 цифрой

df2["month"] = df2.month.map("{:02}".format)

df2["day"] = df2.day.map("{:02}".format)

df2.head()

df2['date'] = df2['year'].map(str) + df2['month'].map(str) +  df2['day'].map(str)

df2.head()

Конвертация в формат Datetime

df2['date'] = pd.to\_datetime(df2['date'], format='%Y%m%d')

df2.head()

Создаем столбец с количеством эпизодов

df2['count']= 1

df2.head()

вычислить совокупную сумму по датам

df2['count\_sum'] = df2.groupby(['date'])['count']. cumsum ()

mask\_ = df2['count\_sum'] == df2.groupby('date')['count\_sum'].transform('max')

sum\_df2 = df2[mask\_]

sum\_df2.head()

Удаляем лишние столбцы

sum\_df2.drop(sum\_df2.columns[[0, 1, 2, 4]], axis=1, inplace=True)

sum\_df2.head()

sum\_df2.info()

Объединяем датасеты

merged\_df = pd.merge\_ordered(sum\_df2, final\_df1)

merged\_df

Удаляем строки со значением NaN, оставляя тем самым данные за 11 месяцев 2011 года

merged\_df = merged\_df.dropna()

merged\_df

Приведение данных по мощности вспышек в формат float

def power\_res(power):

  if power[0] == "X":

    return float(power[1::]) \* 10

  elif power[0] == "M":

    return float(power[1::])

  elif power[0] == "C":

    return float(power[1::]) / 10

  elif power[0] == "B":

    return float(power[1::]) / 100

merged\_df["power\_res"] = merged\_df["power"].apply(power\_res)

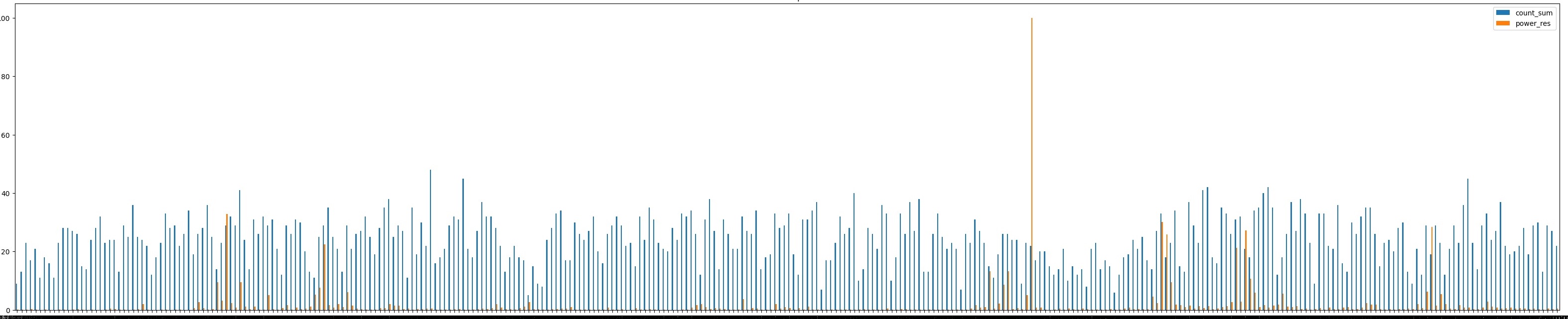
merged\_df.head()

Сохранение объединённого датасета в формае csv

merged\_df.to\_csv('merged\_df.csv')

Отрисовка полученых данных на графике

merged\_df.plot(x='date', y=['count\_sum', 'power\_res'], kind='bar', title="V comp", figsize=(40,8), ylabel='V', rot=0)



merged\_df['count\_sum'].describe().round(2)

merged\_df['power\_res'].describe().round(2)

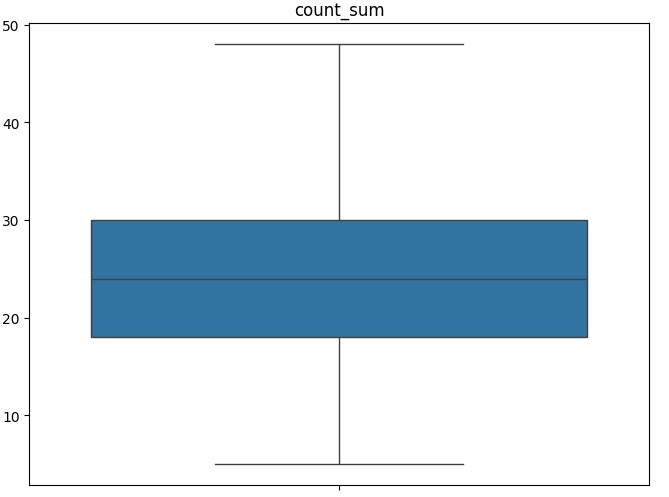
plt.figure(figsize=(8, 6))

sns.boxplot(data=merged\_df, y='count\_sum')

plt.title('count\_sum')

plt.ylabel('')

plt.show()

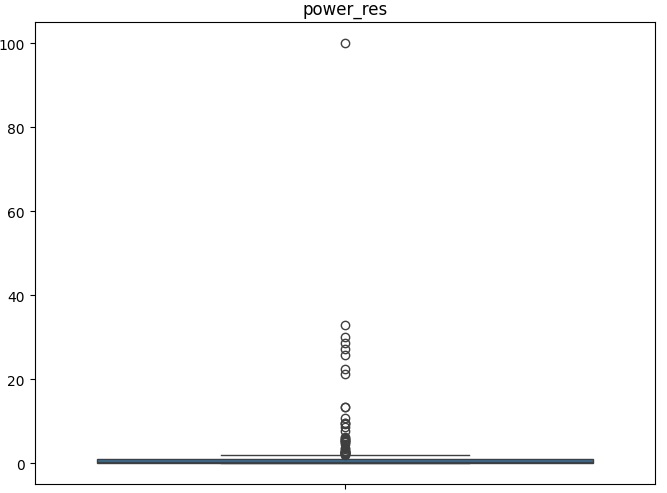


plt.figure(figsize=(8, 6))

sns.boxplot(data=merged\_df, y='power\_res')

plt.title('power\_res')

plt.show()



Обнаружение и обработка выбросов

z\_scores = np.abs(stats.zscore(merged\_df.select\_dtypes(include=np.number)))

merged\_df = merged\_df[(z\_scores < 3).all(axis=1)]

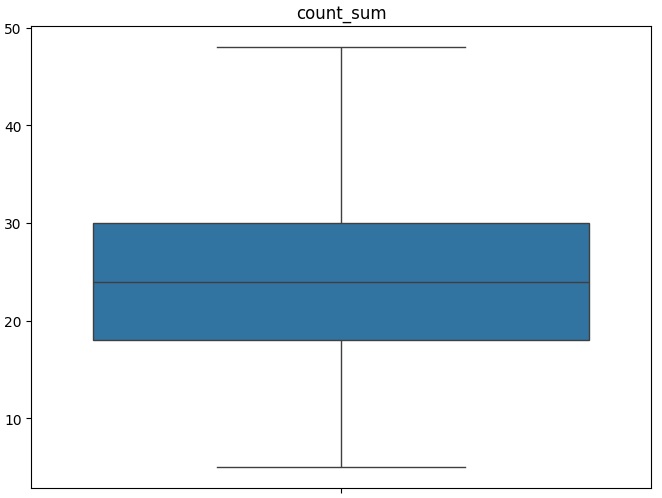
plt.figure(figsize=(8, 6))

sns.boxplot(data=merged\_df, y='count\_sum')

plt.title('count\_sum')

plt.ylabel('')

plt.show()

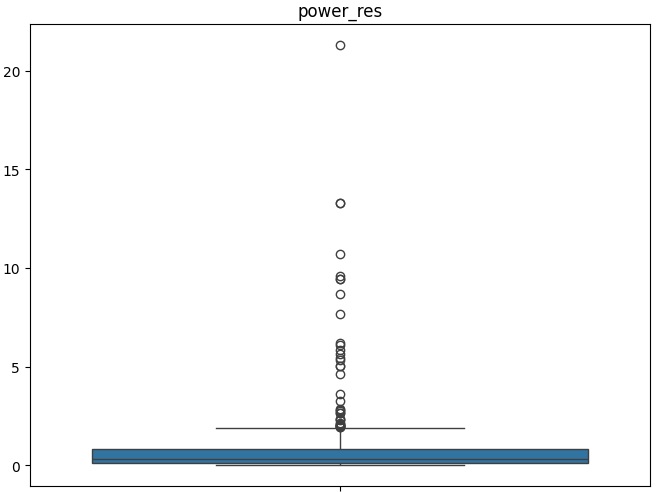


plt.figure(figsize=(8, 6))

sns.boxplot(data=merged\_df, y='power\_res')

plt.title('power\_res')

plt.show()



merged\_df.plot(x='date', y=['count\_sum', 'power\_res'], kind='bar', title="V comp", figsize=(40,8), ylabel='V', rot=0)

