Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №1

з дисципліни «Вступ до Data Science»

Підготовка та аналіз даних для статистичного навчання

Виконав:
Селютін Є.О.
Група ІО-15
Залікова книжка №1519

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О. О.

Київ – 2023

Мета: виявити, дослідити та узагальнити особливості застосування методів статистичного навчання для задач визначення статистичних характеристик вхідного потоку даних з використанням спеціалізованих пакетів мови програмування Python.

Завдання (III рівень):

1. Провести парсинг самостійно обраного сайту. Вміст даних, що підлягають парсингу – обрати самостійно.
2. Результати парсингу зберегти у файлі. Тип файлу обрати самостійно.
3. Оцінити динаміку тренду реальних даних.
4. Здійснити визначення статистичних характеристик результатів парсингу.
5. Синтезувати та веріфікувати модель даних, аналогічних за трендом і статистичними характеристиками реальним даним, які є результатом парсингу.
6. Провести аналіз отриманих результатів.

Виконання лабораторної роботи:

1. Для парсингу даних я обрав сайт https://ru.investing.com/economic-calendar/cpi-733, котрий має в собі табличку показників інфляції долара за певний проміжок часу. Для того, щоб спарсити дані мені довелося використати бібліотеку Selenium, адже там довелося натискати кнопку «показати ще» та закривати модальне вікно. Я написав скрипт, який під'єднується до сторінки, періодично натискає на кнопку «показати ще», а якщо вилазить модальне вікно, то він закриває його та натискає на кнопку певну кількість раз.

Файл ***parse_url.py***

```
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.webdriver import WebDriver as ChromeDriver
import time
```

```
def close_modal(driver: ChromeDriver) -> None:
    close_button = driver.find_element(By.XPATH, """// *[ @ id =
"PromoteSignUpPopUp"] / div[2] / i""")
    close_button.click()


def extract_table_html(driver: ChromeDriver) -> str:
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    table_element = soup.find('table', {'id': 'eventHistoryTable733'})
    table_html = str(table_element)
    return table_html


def click_show_more_button_multiple_times(url: str, output_filename: str,
num_clicks: int) -> None:
    driver = webdriver.Chrome()
    driver.get(url)
    for _ in range(num_clicks):
        try:
            show_more_button = driver.find_element(By.ID, "showMoreHistory733")
            show_more_button.click()

            time.sleep(5)
        except:
            close_modal(driver)
            time.sleep(5)

    table_html = extract_table_html(driver)

    with open(output_filename, 'w', encoding='utf-8') as html_file:
        html_file.write(table_html)
    driver.quit()


url = "https://ru.investing.com/economic-calendar/cpi-733"
output_filename = "output.html"
num_clicks = 20


click_show_more_button_multiple_times(url=url, output_filename=output_filename,
num_clicks=num_clicks)
```

2. Для вилучення та зберігання даних в певний файл, я написав скрапер, який вилучає данні в три формати CSV, JSON, XSLX. Він проходиться по збереженому файлі ***output.html***, дістає дані та зберігає в файлах.

Файл ***scrapper.py***

```
import json
import openpyxl
import csv
from bs4 import BeautifulSoup


def create_json(data: list) -> None:
```

```
    with open('outputs/output.json', 'w', encoding='utf-8') as json_file:
        json.dump(data, json_file, ensure_ascii=False, indent=4)


def create_xlsx(data: list) -> None:
    workbook = openpyxl.Workbook()
    sheet = workbook.active
    sheet.title = 'Inflation Data'

    sheet['A1'] = 'date'
    sheet['B1'] = 'actual_inflation'

    for row_index, entry in enumerate(data, start=2):
        sheet.cell(row=row_index, column=1, value=entry['date'])
        sheet.cell(row=row_index, column=2, value=entry['actual_inflation'])

    workbook.save('outputs/output.xlsx')


def create_csv(data: list) -> None:
    with open('outputs/output.csv', 'w', newline='', encoding='utf-8') as
csv_file:
        csv_writer = csv.writer(csv_file)
        csv_writer.writerow(['date', 'actual_inflation'])

        for entry in data:
            csv_writer.writerow([entry['date'], entry['actual_inflation']])


def scrap_file(create_file=True):
    with open('output.html', 'r', encoding='utf-8') as html_file:
        html_content = html_file.read()

    soup = BeautifulSoup(html_content, 'html.parser')
    table = soup.find('table', {'id': 'eventHistoryTable733'})
    data = []

    for row in table.find_all('tr', {'event_attr_id': '733'}):
        cells = row.find_all('td')
        if len(cells) == 6:
            date_str = cells[0].text.strip()
            date_str = date_str[0:10]
            actual_inflation = cells[2].text.strip()
            data.append({'date': date_str, 'actual_inflation':
actual_inflation})
    if create_file:
        create_json(data=data)
        create_csv(data=data)
        create_xlsx(data=data)
    else:
        return data


scrap_file()
```

3-6. Для динаміки тренду та статистичних даних я створив файл ***analytics.py***. Я створив дві функції перша будує графік по реальним даним та

розраховує статистичні характеристики для даних парсингу, друга робить математичну модель, розраховує статистичні характеристики та лінію тренду.

Файл ***analitics.py***

```
import numpy as np
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from scrapper import scrap_file
import matplotlib.pyplot as plt
from datetime import datetime
import tkinter as tk


def analyze_real_data(dates: list, real_inflation_values: list) -> None:
    real_mean_inflation = np.mean(real_inflation_values)
    real_std_deviation_inflation = np.std(real_inflation_values)
    real_median_inflation = np.median(real_inflation_values)
    real_min_inflation = np.min(real_inflation_values)
    real_max_inflation = np.max(real_inflation_values)
    real_variance_inflation = np.var(real_inflation_values)

    print("Real Data Statistics:")
    print("Mean inflation value:", real_mean_inflation)
    print("Standard deviation of inflation:", real_std_deviation_inflation)
    print("Median inflation:", real_median_inflation)
    print("Minimum inflation value:", real_min_inflation)
    print("Maximum inflation value:", real_max_inflation)
    print("Inflation variance:", real_variance_inflation)

    fig, ax = plt.subplots(figsize=(8, 4))
    ax.plot(dates, real_inflation_values, marker='o', linestyle='-',
color='black', markersize=4)
    ax.set_xlabel("Date")
    ax.set_ylabel("Inflation")
    plt.xticks(rotation=45)
    ax.grid(True)


def generate_and_analyze_synthetic_data(dates: list, real_inflation_values:
list) -> tuple:
    degree = 15
    noise_std = 0.5

    np.random.seed(0)
    coefficients = np.polyfit(np.arange(len(dates)), real_inflation_values,
degree)
    synthetic_trend_values = np.polyval(coefficients, np.arange(len(dates)))
    print("\nModel:")
    print(np.poly1d(coefficients))
    synthetic_inflation_values = synthetic_trend_values + np.random.normal(0,
noise_std, len(dates))

    # Use this if you want to create anomaly
    # anomaly_index = 20
    # anomaly_value = 10.0
    # synthetic_inflation_values[anomaly_index] = anomaly_value
```

```
    synthetic_mean_inflation = np.mean(synthetic_inflation_values)
    synthetic_std_deviation_inflation = np.std(synthetic_inflation_values)
    synthetic_median_inflation = np.median(synthetic_inflation_values)
    synthetic_min_inflation = np.min(synthetic_inflation_values)
    synthetic_max_inflation = np.max(synthetic_inflation_values)
    synthetic_variance_inflation = np.var(synthetic_inflation_values)

    print("\nSynthetic Data Statistics:")
    print("Mean inflation value:", synthetic_mean_inflation)
    print("Standard deviation of inflation:", synthetic_std_deviation_inflation)
    print("Median inflation:", synthetic_median_inflation)
    print("Minimum inflation value:", synthetic_min_inflation)
    print("Maximum inflation value:", synthetic_max_inflation)
    print("Inflation variance:", synthetic_variance_inflation)

    return synthetic_trend_values, synthetic_inflation_values


if __name__ == '__main__':
    data = scrap_file(create_file=False)
    data = data[::-1]
    parsed_dates = [datetime.strptime(d['date'], '%d.%m.%Y').date() for d in
data]
    parsed_inflation_values = [float(d['actual_inflation'].replace('%',
'').replace(',', '.')) for d in data]

    root = tk.Tk()
    root.title("Dollar inflation chart with trend")

    frame1 = tk.Frame(root)
    frame1.pack()

    analyze_real_data(dates=parsed_dates,
real_inflation_values=parsed_inflation_values)

    fig1, ax1 = plt.subplots(figsize=(8, 4))
    ax1.plot(parsed_dates, parsed_inflation_values, marker='o', linestyle='-',
color='black', markersize=4,
             label='Parsed Data')
    ax1.set_xlabel("Date")
    ax1.set_ylabel("Inflation")
    plt.xticks(rotation=45)
    ax1.grid(True)
    plt.legend()
    plt.title("Dollar Inflation Chart (Parsed)")

    canvas1 = FigureCanvasTkAgg(fig1, master=frame1)
    canvas_widget1 = canvas1.get_tk_widget()
    canvas_widget1.pack()

    frame2 = tk.Frame(root)
    frame2.pack()

    synthetic_trend_values, synthetic_inflation_values =
generate_and_analyze_synthetic_data(
        dates=parsed_dates,
        real_inflation_values=parsed_inflation_values)
```

```
    fig2, ax2 = plt.subplots(figsize=(8, 4))
    ax2.plot(parsed_dates, synthetic_inflation_values, marker='x', linestyle='--
', color='red', markersize=4,
             label='Synthetic Data')
    ax2.plot(parsed_dates, synthetic_trend_values, linestyle='--', color='blue',
label='Synthetic Trend')
    ax2.set_xlabel("Date")
    ax2.set_ylabel("Inflation")
    plt.xticks(rotation=45)
    ax2.grid(True)
    plt.legend()
    plt.title("Dollar Inflation Chart (Synthetic)")

    canvas2 = FigureCanvasTkAgg(fig2, master=frame2)
    canvas_widget2 = canvas2.get_tk_widget()
    canvas_widget2.pack()

    root.mainloop()
```

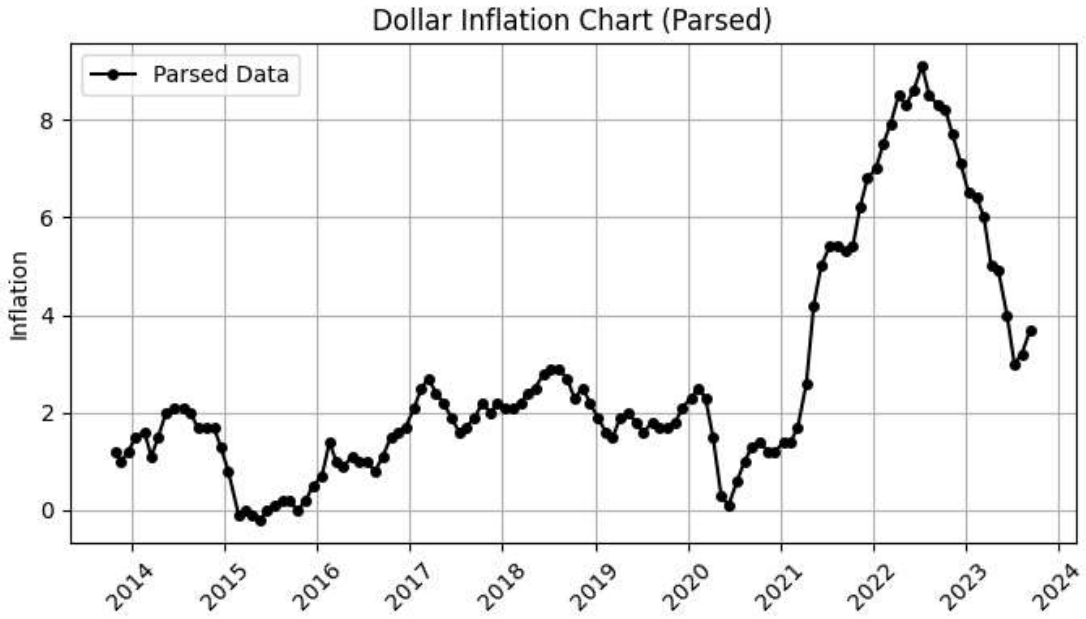
Результати виконання лабораторної роботи:

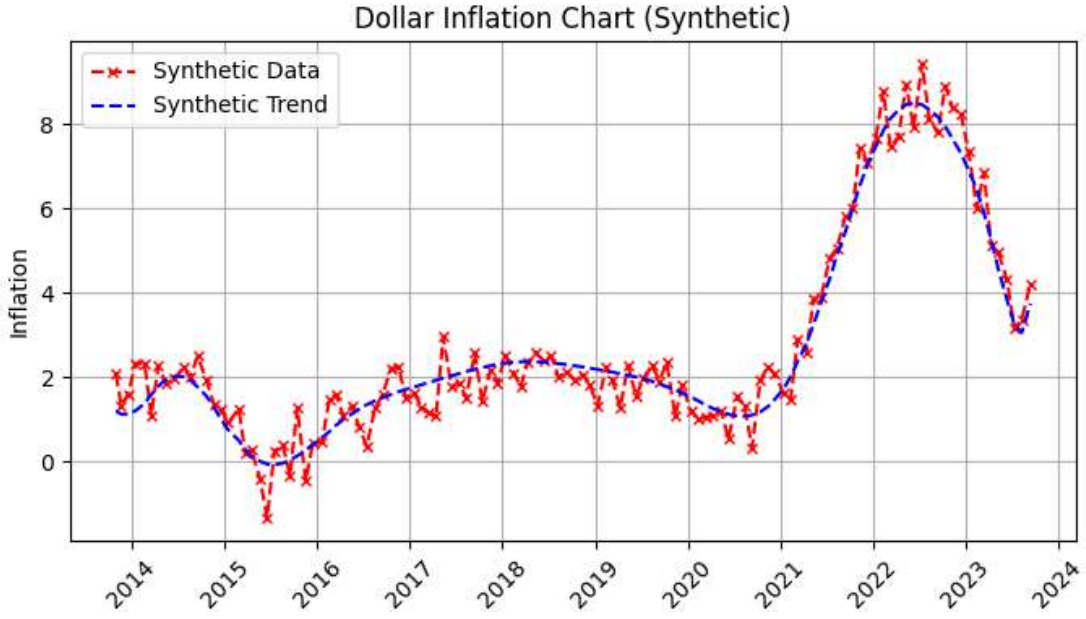
```
Real Data Statistics:
Mean inflation value: 2.6783333333333332
Standard deviation of inflation: 2.324406136820519
Median inflation: 1.9
Minimum inflation value: -0.2
Maximum inflation value: 9.1
Inflation variance: 5.402863888888889
```

```
Synthetic Data Statistics:
Mean inflation value: 2.7446393474761095
Standard deviation of inflation: 2.4159455818564 38
Median inflation: 1.956654623194142
Minimum inflation value: -1.3325722396935 12
Maximum inflation value: 9.440939109185999
Inflation variance: 5.8367930544916415
```

```
Model:
           15              14              13              12
6.679e-24 x  - 5.845e-21 x  + 2.298e-18 x  - 5.363e-16 x
             11              10              9              8
 + 8.26e-14 x  - 8.832e-12 x  + 6.713e-10 x - 3.649e-08 x
              7              6              5              4              3
 + 1.405e-06 x - 3.73e-05 x + 0.0006483 x - 0.006657 x + 0.03198 x
             2
 - 0.02435 x - 0.07532 x + 1.201
```

Dollar inflation chart with trend





Аналіз результатів лабораторної роботи:

За графіком даних парсингу інфляції долара видно, що з 2021 по 2023 рік інфляція сильно виросла, це обумовлено ковідом, війною в Україні та певними зовнішніми чинниками. Також треба зазначити, що в 2015 році інфляція була від’ємна і було укріплення долара.

Математична модель складена коректно, оскільки графіки співпадають. За цими даними можна прогнозувати подальшу інфляцію долара.

Висновки: у ході виконання лабораторної роботи проведено парсинг сайту з даними про інфляцію долара. Розроблено скрипт для вилучення даних зпарсеного файлу та збереження їх у різні форматі. Розроблено скрипт для аналітики цих даних, побудови графіків, створенню математичної моделі, розрахунку статистичних даних та побудови лінії тренду.