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***

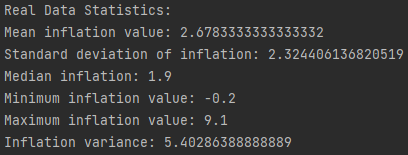
import json  
import openpyxl  
import csv  
from bs4 import BeautifulSoup  
  
  
def create\_json(data: list) -> None:  
 with open('outputs/output.json', 'w', encoding='utf-8') as json\_file:  
 json.dump(data, json\_file, ensure\_ascii=False, indent=4)  
  
  
def create\_xlsx(data: list) -> None:  
 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()

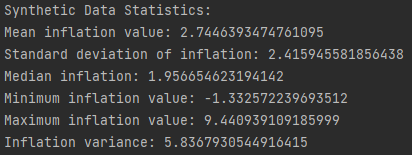
3-6. Для динаміки тренду та статистичних даних я створив файл ***analytics.py***. Я створив дві функції перша будує графік по реальним даним та розраховує статистичні характеристики для даних парсингу, друга робить математичну модель, розраховує статистичні характеристики та лінію тренду.

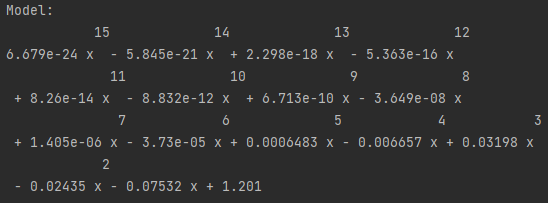
Файл ***analitics.py***

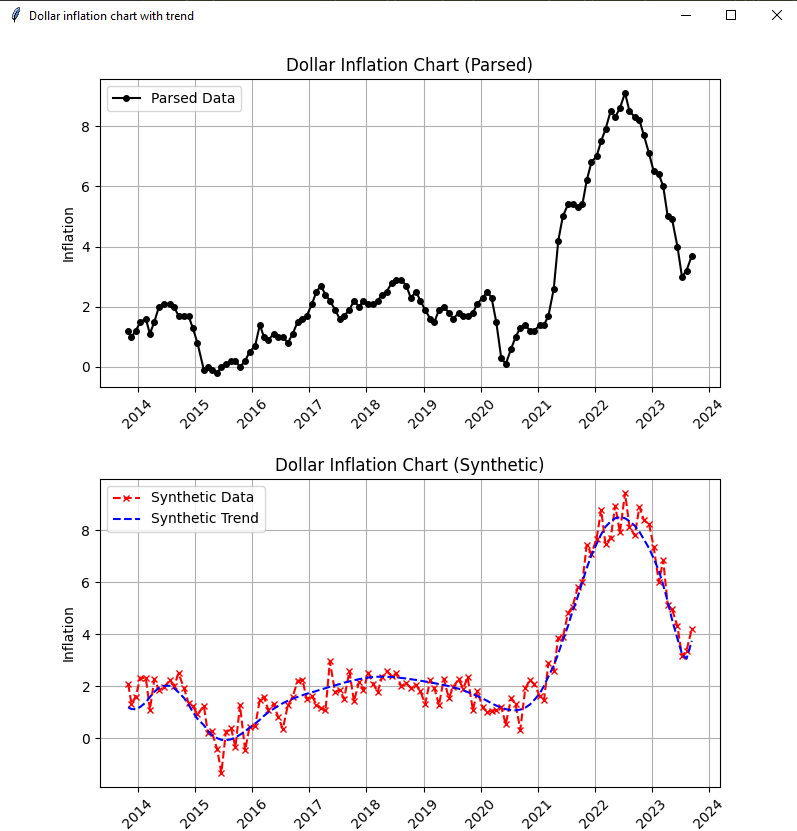
import numpy as np  
from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  
from scrapper import scrap\_file  
import matplotlib.pyplot as plt  
from datetime import datetime  
import tkinter as tk  
  
  
def analyze\_real\_data(dates: list, real\_inflation\_values: list) -> None:  
 real\_mean\_inflation = 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:**

****



****



**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.