МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

<<ЛЬВІВСЬКА ПОЛІТЕХНІКА>>

Інститут ІКНІ

Кафедра систем штучного інтелекту



ЗВІТ

Лабораторна робота **№2**

З курсу “ Інтелектуальний аналіз даних”

Виконав:

**Бурак Марко**

гр. **КН-310**

Прийняв(ла):

Якимишин Х.М.

Львів – 2020

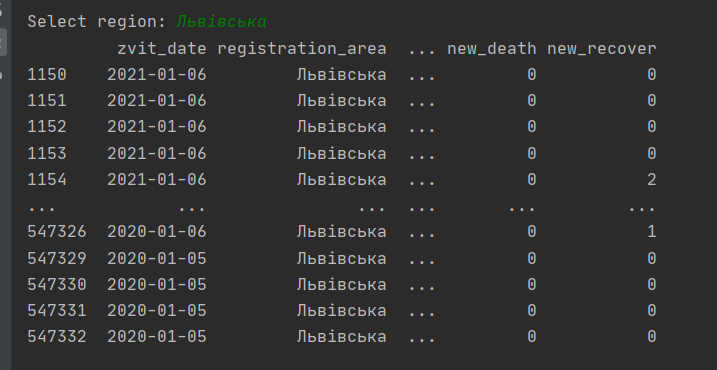
**Лабораторна №2**

**Попередня обробка даних**

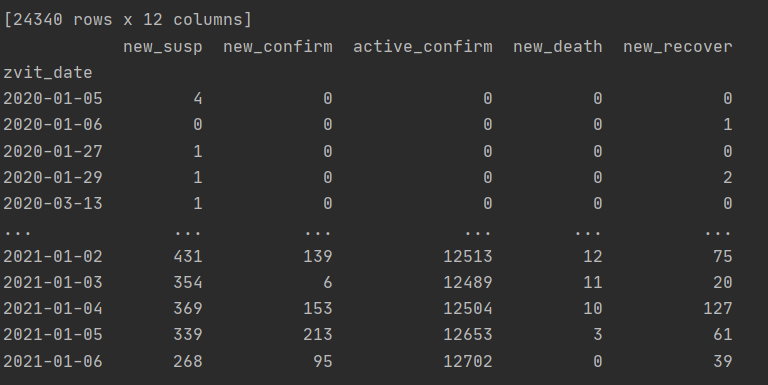
1. Написати скрипт – парсер для завантаження даних про динаміку поширення COVID-19 із <https://nszu.gov.ua/covid/dashboard>
2. Вибрати дані по одній з областей України
3. Згрупувати дані по ознаці “однакова дата” за операцією SUM
4. За допомогою розробленої функції візуалізації даних побудувати динаміку активних, підозрілих, підтверджених, летальних, госпіталізованих хворих
5. Провести порівняльний аналіз захворівших по різним областям.
6. Вивести статистичні дані па Україні на географічну карту
7. Результати аналізу імпортувати в Excel
8. Написати скрипт для автоматичної побудови графіку в Excel. (додаткові бали)
9. Оформити модуль на репозиторії GitHub з відповідним описом установки та використання

**Завдання №2**

Вибрані дані по області:

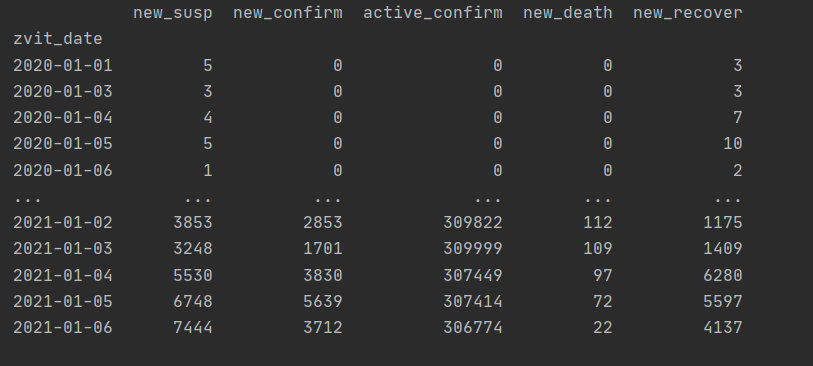


Вибрані дані по всій країні:



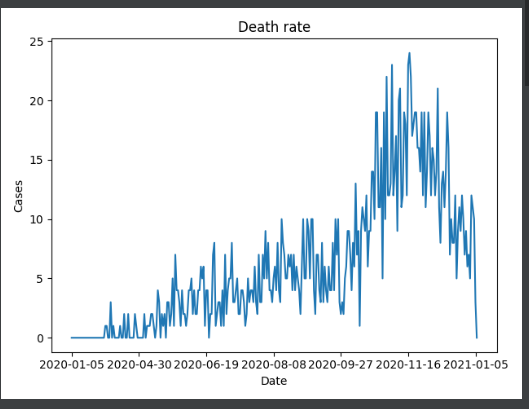
**Завдання №3**

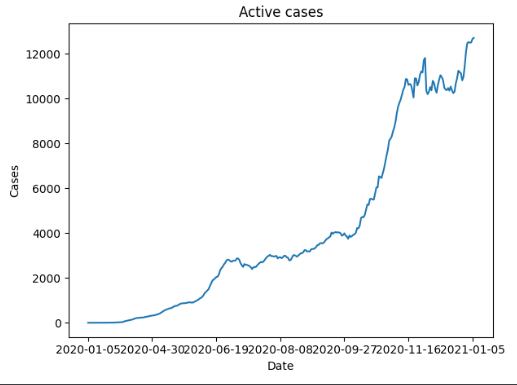
Дані згруповані за датою:

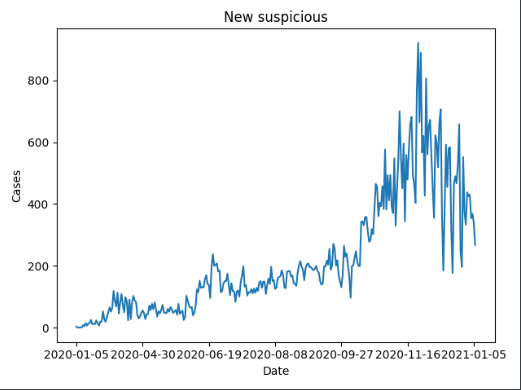


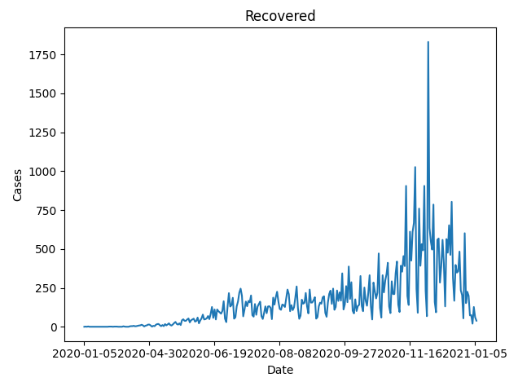
**Завдання №4**

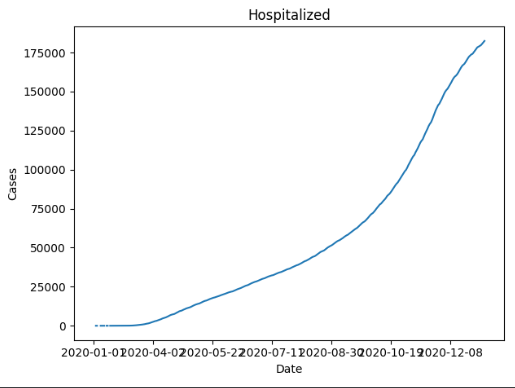
Динаміка активних, підозрілих, підтверджених, летальних, госпіталізованих хворих у Львівській області:





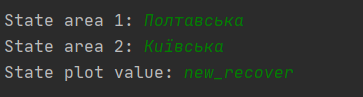


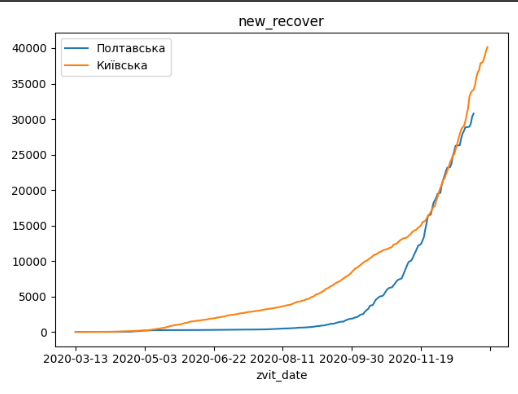




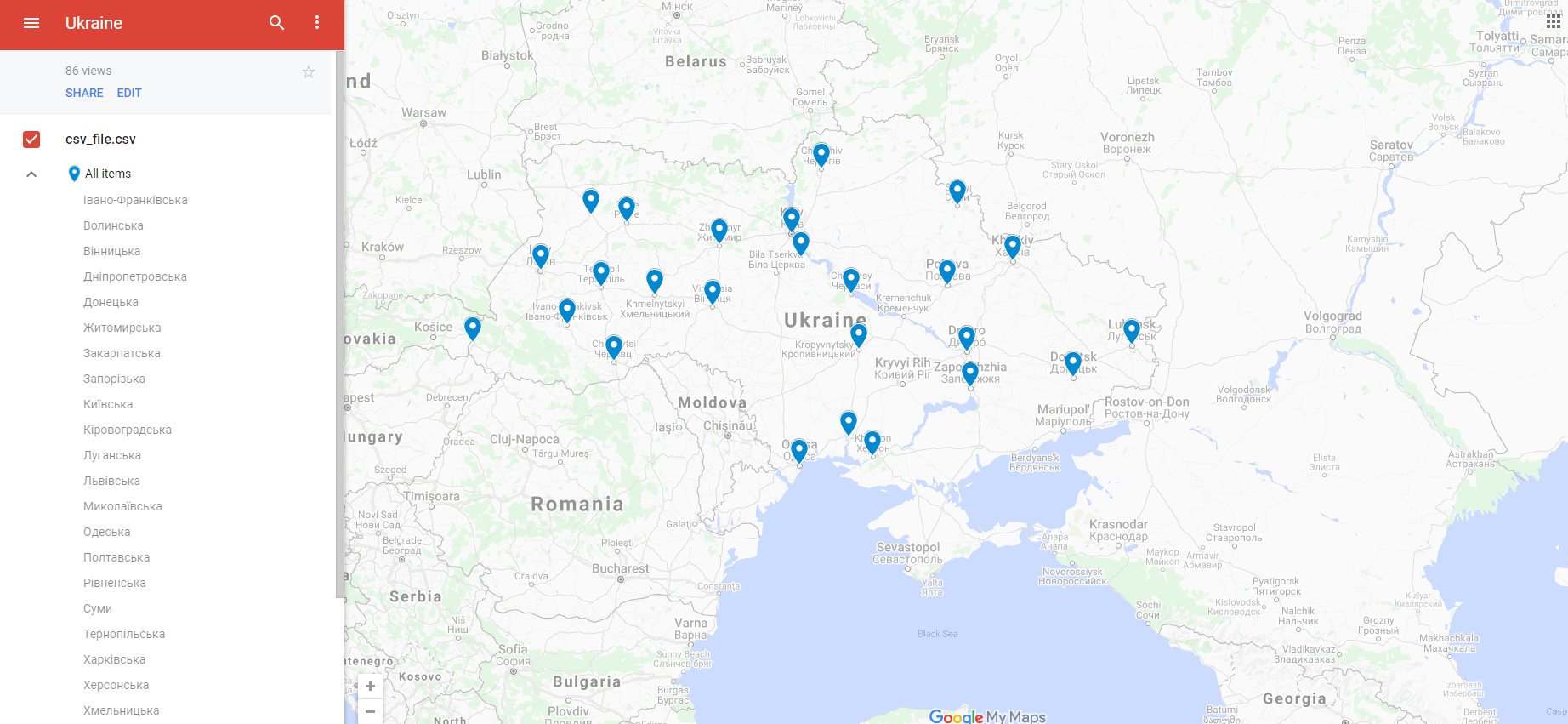
**Завдання №5**

Порівняння двох областей, для будь-якого поля



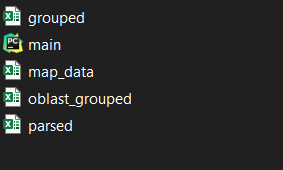


**Завдання №6**

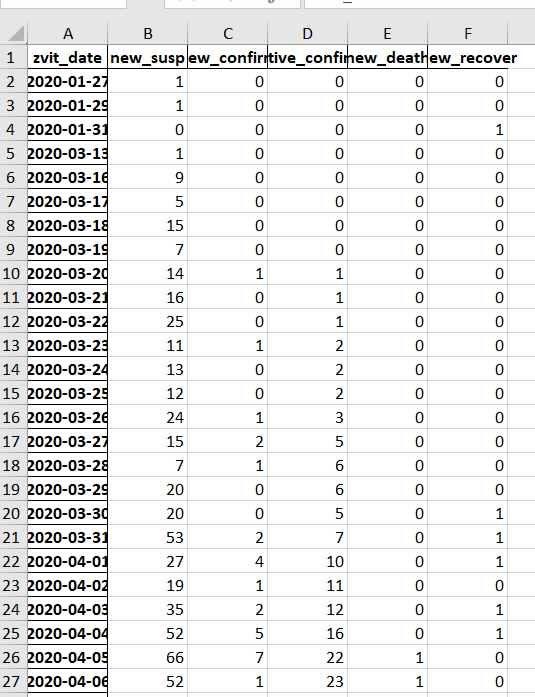


**Завдання №7**

Збереження всіх датафреймів, використаних для аналізу у ексель таблиці



Таблиця по Львівській області.



**Код приграми:**

import pandas as pd  
import matplotlib.pyplot as plt  
import webbrowser  
import matplotlib.patches as mpatches  
  
pd.set\_option('display.max\_columns', 100)  
pd.set\_option('display.max\_rows', 1000)  
  
url = 'https://raw.githubusercontent.com/VasiaPiven/covid19\_ua/master/covid19\_by\_area\_type\_hosp\_dynamics.csv'  
df = pd.read\_csv(url, error\_bad\_lines=False)  
  
  
# 1)  
def state(df, name):  
 # select one state  
 area = df.loc[df['registration\_area'] == name]  
 return area  
  
def state\_group(df):  
 state = df.groupby('zvit\_date').sum()  
 return state  
temp = pd.dataframe  
def plot\_analysis(state1, state2, column):  
 raw1 = state(df, state1)  
 raw2 = state(df, state2)  
 start1 = state\_group(raw1)  
 start2 = state\_group(raw2)  
  
 state1\_df = pd.concat(  
 [start1[['new\_susp', 'new\_confirm', 'new\_death', 'new\_recover']].cumsum(),  
 start1['active\_confirm'], raw1.loc[raw1['is\_required\_hospitalization'] == 'Так'].  
 groupby(['zvit\_date']).sum().cumsum().rename(columns={'new\_susp': 'hospitalized'})[  
 'hospitalized']],  
 axis=1)  
 state1\_df['test'] = state1  
  
 print(state1\_df)  
 state2\_df = pd.concat(  
 [start2[['new\_susp', 'new\_confirm', 'new\_death', 'new\_recover']].cumsum(),  
 start2['active\_confirm'], raw2.loc[raw2['is\_required\_hospitalization'] == 'Так'].  
 groupby(['zvit\_date']).sum().cumsum().rename(columns={'new\_susp': 'hospitalized'})[  
 'hospitalized']],  
 axis=1)  
  
 ax = plt.gca()  
  
 plot\_line(state1\_df, column, column)  
 plot\_line(state2\_df, column, column)  
  
 ax.legend([state1,state2])  
 # ax.grid('on', which='minor', axis='y')  
 # ax.grid('off', which='major', axis='y')  
 # plt.xticks(rotation=45)  
 # plt.xlabel('day/month/year')  
 # plt.ylabel('max Humidity %')  
 show()  
  
# 3)  
def plot\_variaty(df\_grouped, df\_raw):  
  
 variaty = pd.concat(  
 [df\_grouped[['new\_susp', 'new\_confirm', 'new\_death', 'new\_recover']].cumsum(),  
 df\_grouped['active\_confirm'], df\_raw.loc[df\_raw['is\_required\_hospitalization'] == 'Так'].  
 groupby(['zvit\_date']).sum().cumsum().rename(columns={'new\_susp': 'hospitalized'})[  
 'hospitalized']],  
 axis=1)  
  
 plot\_line(variaty, 'new\_confirm', 'Confirmed cases line')  
 show()  
 plot\_line(variaty, 'new\_susp', 'Suspected cases line')  
 show()  
 plot\_line(variaty, 'new\_recover', 'Recovered cases line')  
 show()  
 plot\_line(variaty, 'new\_death', 'Lethal cases line')  
 show()  
 plot\_line(variaty, 'hospitalized', 'Hospitalized cases line')  
 show()  
 plot\_line(variaty, 'active\_confirm', 'Active cases line')  
 show()  
  
def plot\_line(df, column, name):  
 df[column].plot(kind='line', title=name)  
def show():  
 plt.show()  
  
  
print('1)')  
  
print('Вінницька\t',  
 'Волинська\t',  
 'Дніпропетровська\t',  
 'Донецька\t',  
 'Житомирська\t',  
 'Закарпатська\t',  
 'Запорізька\t',  
 'Івано-Франківська\t',  
 'Київська\t',  
 'Кіровоградська\t',  
 'Луганська\t',  
 'Львівська\t',  
 'Миколаївська\t',  
 'м. Київ\t',  
 'Одеська\t',  
 'Полтавська\t',  
 'Рівненська\t',  
 'Сумська\t',  
 'Тернопільська\t',  
 'Харківська\t',  
 'Херсонська\t',  
 'Хмельницька\t',  
 'Черкаська\t',  
 'Чернівецька\t',  
 'Чернігівська\t',  
 )  
  
# df for state  
name\_state = input()  
df\_state = state(df, name\_state)  
print(df\_state)  
  
# 2)  
# df for every day  
print('2)')  
fake = input()  
df\_date = df.groupby('zvit\_date').agg(  
 {'new\_susp': 'sum', 'active\_confirm': 'sum', 'new\_death': 'sum', 'new\_recover': 'sum', 'new\_confirm': 'sum'})  
print(df\_date)  
  
# 3)  
print('3)')  
  
while True:  
 choice = input('plot for all or for state\nall\nstate\n')  
 if choice == 'all':  
 plot\_variaty(df\_date, df)  
 elif choice == 'state':  
 stateGrouped = state\_group(df\_state)  
 plot\_variaty(stateGrouped, df\_state)  
 else:  
 break  
  
print('4)')  
fake = input()  
webbrowser.open(  
 "https://www.google.com/maps/d/u/0/viewer?mid=1OejaE3qulsxbCS4N56G9mToVRv018nPg&ll=49.05356893509722%2C30"  
 ".797848999999978&z=6",  
 new=1)  
  
print('5)')  
while True:  
  
 print('Вінницька\t',  
 'Волинська\t',  
 'Дніпропетровська\t',  
 'Донецька\t',  
 'Житомирська\t',  
 'Закарпатська\t',  
 'Запорізька\t',  
 'Івано-Франківська\t',  
 'Київська\t',  
 'Кіровоградська\t',  
 'Луганська\t',  
 'Львівська\t',  
 'Миколаївська\t',  
 'м. Київ\t',  
 'Одеська\t',  
 'Полтавська\t',  
 'Рівненська\t',  
 'Сумська\t',  
 'Тернопільська\t',  
 'Харківська\t',  
 'Херсонська\t',  
 'Хмельницька\t',  
 'Черкаська\t',  
 'Чернівецька\t',  
 'Чернігівська\t',  
 )  
 print('choose 2 states')  
 state1 = input('state1 = ')  
 state2 = input('state2 = ')  
 print('new\_confirm\t',  
 'new\_susp\t',  
 'new\_recover\t',  
 'new\_death\t',  
 'hospitalized\t',  
 'active\_confirm\t')  
  
 option = input('type option = ')  
  
 plot\_analysis(state1, state2,option)  
 exit = input()  
 if exit == "exit":  
 break  
  
df.groupby('registration\_area')['new\_confirm'].sum().sort\_values(ascending=False).plot(kind='bar')  
plt.show()  
  
df\_date.to\_excel('date\_excel.xlsx')  
df\_state.to\_excel('state\_excel.xlsx')  
df.to\_excel('start.xlsx')  
states = df.groupby('registration\_area').agg(  
 {'new\_susp': 'sum', 'active\_confirm': 'sum', 'new\_death': 'sum', 'new\_recover': 'sum', 'new\_confirm': 'sum'})  
states.to\_csv('states.csv')  
states.to\_excel('states.xlsx')

**Висновок:** виконуючи лабораторну роботу, провів аналіз даних по розповсюдженню коронавірусу в Україні. Візуалізував їх на мапі, реалізував можливість будувати графіки, та порівнювати різні області за різними статистичними даними.