ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки

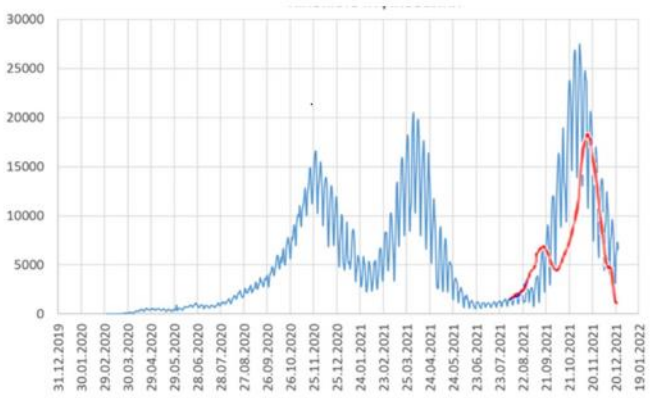
НЕЙРОМЕРЕЖЕВІ МЕТОДИ ПРОГНОЗУВАННЯ ПОШИРЕННЯ ЗАХВОРЮВАНЬ

Виконав:
ст. гр. КСМм-23-1 Гуренко Д. М.

Науковий керівник:
доц. каф. ЕОМ Іващенко Г. С.

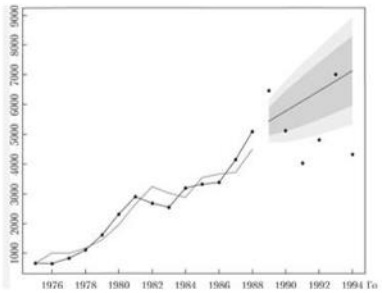
Актуальність проблеми

- Поширення пандемій захворювань.
- Затримки у прийнятті управлінських рішень щодо протиепідемічних заходів.
- Можливість використання сучасних нейромережевих архітектур (RNN, LSTM та CNN).



Поширені методи прогнозування

- Наївні методи.
- Авторегресійні (ARIMA, SARIMA, ARIMAX).
- Експоненціальне згладжування.
- Засоби ML.



3

Постановка задачі

У роботі розглядаються принципи використання та побудови нейромережевих методів для прогнозування захворювань.

У результаті аналізу актуальних наукових досліджень, для вирішення задачі були обрані такі моделі штучних нейронних мереж:

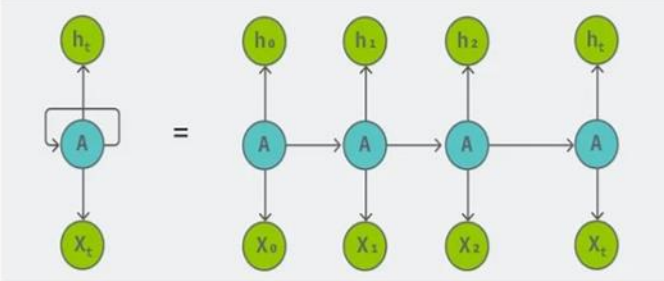
- рекурентна нейронна мережа (RNN);
- довга короткострокова пам'ять (LSTM);
- згорткова нейронна мережа (CNN).

Для навчання моделей використовуються набори даних з відкритих джерел, таких як ВООЗ, дослідницькі та аналітичні платформи (Kaggle).

4

RNN

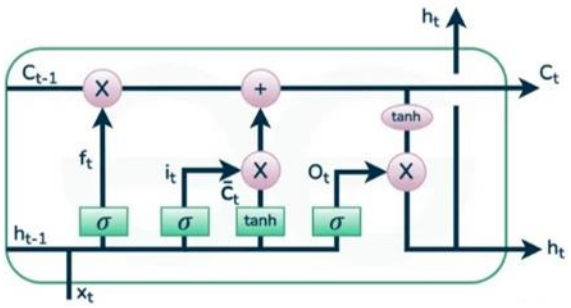
RNN (Recurrent Neural Networks) – це тип штучних нейронних мереж, які дозволяють використовувати попередні виходи як вхідні, маючи приховані стани.



5

LSTM

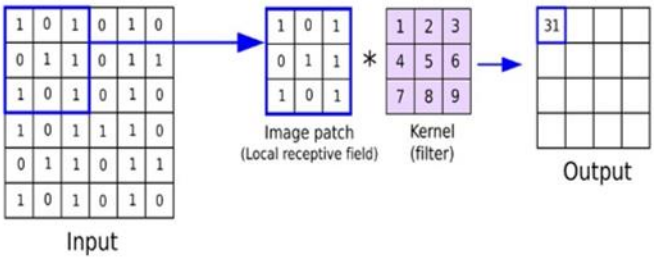
LSTM (Long Short -Term Memory) – це тип рекурентної нейронної мережі RNN, створений для обробки довгих послідовностей даних та збереження інформації за тривалий час. Здатність обробляти довготривалі залежності в послідовностях.



6

CNN

CNN (Convolutional Neural Network) – це особливий вид нейронних мереж, що використовують механізм згортки, який дозволяє вилучати важливі ознаки з даних.



7

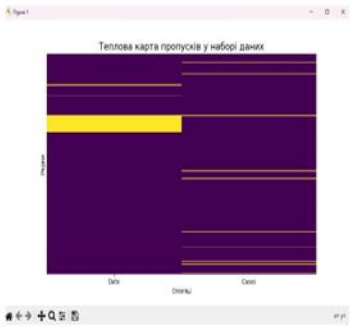
Технології програмної реалізації

- Python. PyCharm;
- Tenserflow;
- Keras.

8

Попередня обробка часових рядів

- Аналіз даних.
- Виявлення викидів.
- Згладжування.



9

Прогнозування за допомогоюRNN

№	units	optimizer	activation	MSE	Час навчання, с
1	2	adam	relu	0.6648	4.34
2	2	sgd	tanh	0.5346	4.10
3	4	adam	relu	0.4081	4.28
4	4	sgd	tanh	0.4232	4.15
5	8	adam	relu	0.3587	4.52
6	8	sgd	tanh	0.3701	4.15
7	16	adam	relu	0.3527	4.62
8	16	sgd	tanh	0.3561	4.33
9	32	adam	relu	0.3483	4.93
10	32	sgd	tanh	0.3448	4.71
11	64	adam	relu	0.3483	5.43
12	64	sgd	tanh	0.3512	4.99
13	128	adam	relu	0.3598	6.01
14	128	adam	tanh	0.3687	5.87
15	128	sgd	relu	0.3701	5.99

10

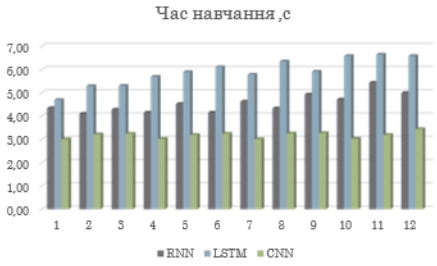
Прогнозування за допомогоюLSTM

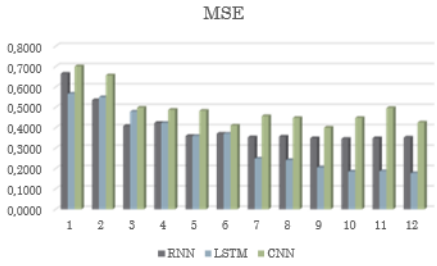
№	units	optimizer	activation	MSE	Час навчання, с
1	2	adam	relu	0.5667	4.69
2	2	sgd	tanh	0.5495	5.29
3	4	adam	relu	0.4784	5.30
4	4	sgd	tanh	0.4232	5.69
5	8	adam	relu	0.3587	5.89
6	8	sgd	tanh	0.3701	6.10
7	16	adam	relu	0.2485	5.78
8	16	sgd	tanh	0.2411	6.35
9	32	adam	relu	0.2047	5.91
10	32	sgd	tanh	0.1845	6.58
11	64	adam	relu	0.1864	6.64
12	64	sgd	tanh	0.1767	6.58
13	128	adam	relu	0.1825	7.21
14	128	adam	tanh	0.1902	8.55
15	128	sgd	relu	0.1809	8.23

Прогнозування за допомогоюCNN

№	kernel_size	filters	MSE	Час навчання, с
1	2	32	0.7012	3.01
2	2	64	0.6567	3.22
3	2	128	0.4977	3.24
4	3	32	0.4875	3.02
5	3	64	0.4826	3.19
6	3	128	0.4103	3.24
7	4	32	0.4567	3.01
8	4	64	0.4475	3.25
9	4	128	0.4002	3.27
10	5	32	0.4476	3.03
11	5	64	0.4965	3.19
12	5	128	0.4254	3.44

Порівняння результатів





Висновки

У роботі досліджено застосування нейромережевих методів для рішення задачі прогнозування поширення захворювань.

Проаналізовано ефективність нейромережевих моделей, таких як CNN, RNN та LSTM, в задачах прогнозування поширення захворювань, де вихідні дані описані у вигляді часових рядів.

Напрямки подальших досліджень:

- додаткова оптимізації параметрів використаних мереж;
- гібридизація різних прогнозних моделей для покращення результату
- використання засобів паралельних обчислень для забезпечення роботи у режимі реального часу.

Результати дослідження були наведені у рамках Дванадцятої міжнародної науково-технічної конференції "Проблеми інформатизації ".

ДОДАТОК Б

Вихідний код розроблених програмних засобів

Лістинг Б.1 – Лістинг RNN моделі

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, SimpleRNN
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.layers import Dropout
import matplotlib.pyplot as plt
import tkinter as tk
from tkinter import filedialog, messagebox
import time  # Імпорт модуля для вимірювання часу
# Глобальні змінні
model = None
scaler = None
sequence_length = 30  # 7 днів для прогнозування наступного дня
# Завантаження та підготовка даних для навчання
def load_training_data():
    global scaler, X_train, y_train, X_test, y_test
    file_path = filedialog.askopenfilename(title="Виберіть файл
для навчання", filetypes=(("Excel Files", "*.xlsx"),))
    if not file_path:
        return
    data = pd.read_excel(file_path)
    data['Date'] = pd.to_datetime(data['Date'])
    data.set_index('Date', inplace=True)
    # Масштабування даних
    scaler = MinMaxScaler(feature_range=(0, 1))
    data['Cases'] = scaler.fit_transform(data[['Cases']])
    # Підготовка даних для моделі
    X, y = [], []
    for i in range(len(data) - sequence_length):
        X.append(data['Cases'].values[i:i+sequence_length])
        y.append(data['Cases'].values[i+sequence_length])
    X, y = np.array(X), np.array(y)
    # Поділ на навчальну і тестову вибірки
    train_size = int(len(X) * 0.8)
    global X_train, y_train, X_test, y_test
    X_train, X_test = X[:train_size], X[train_size:]
    y_train, y_test = y[:train_size], y[train_size:]
    # Зміна форми для RNN
    X_train = X_train.reshape((X_train.shape[0],
X_train.shape[1], 1))
```

```
    X_test = X_test.reshape((X_test.shape[0], X_test.shape[1],
1))
    messagebox.showinfo("Дані", "Дані для навчання успішно
завантажено!")
# Створення моделі
def create_model():
    global model
    model = Sequential([
        SimpleRNN(32, activation='relu',
input_shape=(sequence_length, 1)),
        Dropout(0.2),  # Dropout із ймовірністю 20%
        Dense(1)
    ])
    model.compile(optimizer='sgd', loss='mean_squared_error')
# Навчання моделі
def train_model():
    if model is None:
        create_model()
    if X_train is None or y_train is None:
        load_training_data()
    start_time = time.time()  # Початок вимірювання часу
    history = model.fit(X_train, y_train, epochs=30,
batch_size=5, validation_data=(X_test, y_test))
    end_time = time.time()  # Кінець вимірювання часу
    training_time = end_time - start_time  # Обчислення
тривалості
    messagebox.showinfo("Навчання", f"Модель успішно навчена!
Час навчання: {training_time:.2f} секунд")
# Прогнозування на новому наборі даних
def make_prediction():
    if model is None:
        messagebox.showerror("Помилка", "Спочатку навчіть
модель!")
        return
    file_path = filedialog.askopenfilename(title="Виберіть файл
для прогнозування", filetypes=(("Excel Files", "*.xlsx"),))
    if not file_path:
        return
    data = pd.read_excel(file_path)
    data['Date'] = pd.to_datetime(data['Date'])
    data.set_index('Date', inplace=True)
    # Масштабування даних
    data['Cases'] = scaler.transform(data[['Cases']])
    n_steps = 30  # Кількість днів для порівняння
    comparison_data = data['Cases'].values[-(n_steps +
sequence_length):]
    X_comparison = np.array([
        comparison_data[i:i + sequence_length]
        for i in range(len(comparison_data) - sequence_length)
    ]).reshape(-1, sequence_length, 1)
    # Прогноз
    predictions = model.predict(X_comparison)
```

```
    # Обчислення MSE
    real_values =
scaler.inverse_transform(comparison_data[sequence_length:].resha
pe(-1, 1))
    predictions_rescaled = scaler.inverse_transform(predictions)
    mse = np.mean((comparison_data - predictions) ** 2)
    print(f"Mean Squared Error (MSE): {mse:.4f}")
    # Візуалізація даних і прогнозу
    plt.figure(figsize=(12, 6))
    plt.plot(data.index[-n_steps:], real_values, label='Реальні
дані', color='blue')
    plt.plot(data.index[-n_steps:], predictions_rescaled,
label='Прогноз', color='red', linestyle='dashed')
    plt.title(f'Порівняння реальних даних і прогнозу (MSE:
{mse:.4f})')
    plt.xlabel('Дата')
    plt.ylabel('Кількість випадків')
    plt.legend()
    plt.show()
# tkinter
root = tk.Tk()
root.title("Прогноз захворюваності")
btn_load_train_data = tk.Button(root, text="Завантажити дані для
навчання", command=load_training_data, width=30, height=2)
btn_load_train_data.pack(pady=10)
btn_train = tk.Button(root, text="Навчити", command=train_model,
width=30, height=2)
btn_train.pack(pady=10)
btn_predict = tk.Button(root, text="Зробити прогноз",
command=make_prediction, width=30, height=2)
btn_predict.pack(pady=10)
root.mainloop()
```

Лістинг Б.2 – Лістинг LSTM моделі

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
# Завантаження даних
file_path_1 = 'up_trend.xlsx'  # Перший файл з даними
file_path_2 = 'up_trendL.xlsx'         # Другий файл з іншими
даними
data_1 = pd.read_excel(file_path_1)
data_2 = pd.read_excel(file_path_2)
# Перевірка та перетворення дат
for df in [data_1, data_2]:
    df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d')
    df.sort_values('Date', inplace=True)
```

```
    df.reset_index(drop=True, inplace=True)
# Масштабування даних
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled =
scaler.fit_transform(data_1['Cases'].values.reshape(-1, 1))
# Формування вхідних даних для LSTM
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset) - look_back):
        X.append(dataset[i:(i + look_back), 0])
        Y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(Y)
look_back = 5  # Кількість днів для прогнозу
X, y = create_dataset(data_scaled, look_back)
# Перетворення вхідних даних для LSTM
X = np.reshape(X, (X.shape[0], X.shape[1], 1))
# Побудова моделі LSTM
# Побудова LSTM моделі
model = Sequential([
    LSTM(32, activation='tanh', input_shape=(look_back, 1)),
    Dense(1)])
model.compile(optimizer='adam', loss='mse')
# Навчання моделі
model.fit(X, y, batch_size=5, epochs=100)
# Прогноз на n днів вперед
n_days = 20  # Задане число днів
last_days = data_scaled[-look_back:]
forecast = []
current_input = last_days.reshape(1, look_back, 1)
for _ in range(n_days):
    pred = model.predict(current_input, verbose=0)
    forecast.append(pred[0, 0])
    current_input = np.append(current_input[:, 1:, :],
pred.reshape(1, 1, 1), axis=1)
forecast = scaler.inverse_transform(np.array(forecast).reshape(-
1, 1))
# Додавання прогнозу до дат
forecast_dates = pd.date_range(data_1['Date'].iloc[-1] +
pd.Timedelta(days=1), periods=n_days)
forecast_df = pd.DataFrame({'Date': forecast_dates, 'Forecast':
forecast.flatten()})
# Побудова графіка
plt.figure(figsize=(12, 6))
plt.plot(data_1['Date'], data_1['Cases'], label='Original Data')
plt.plot(data_2['Date'], data_2['Cases'], label='Other Data')
plt.plot(forecast_df['Date'], forecast_df['Forecast'],
label='Forecast', linestyle='--')
plt.title('Cases Forecasting')
plt.xlabel('Date')
plt.ylabel('Cases')
plt.legend()
plt.grid()
plt.show()
```

Лістинг Б.3 – Лістинг CNN моделі

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv1D, Flatten
from sklearn.preprocessing import MinMaxScaler
# Завантаження даних
file1 = "smoothed_data100d.xlsx"
file2 = "smoothed_data100dl.xlsx"
data1 = pd.read_excel(file1)
data2 = pd.read_excel(file2)
# Перетворення дат і значень
data1['Date'] = pd.to_datetime(data1['Date'])
data2['Date'] = pd.to_datetime(data2['Date'])
# Перетворення значень у рядки перед заміною
data1['Cases'] = data1['Cases'].astype(str).str.replace(',',
'.').astype(float)
data2['Cases'] = data2['Cases'].astype(str).str.replace(',',
'.').astype(float)
# Підготовка даних для CNN
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data1[['Cases']])
def create_sequences(data, sequence_length):
    X, y = [], []
    for i in range(len(data) - sequence_length):
        X.append(data[i:i+sequence_length])
        y.append(data[i+sequence_length])
    return np.array(X), np.array(y)
sequence_length = 30  # Довжина послідовності
X, y = create_sequences(data_scaled, sequence_length)
# Розбиття на тренувальний і тестовий набори
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
# Перетворення для CNN (додавання виміру для входу)
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1],
1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))
# Побудова CNN моделі
model = Sequential([
    Conv1D(64, kernel_size=3, activation='relu',
input_shape=(sequence_length, 1)),
    Flatten(),
    Dense(50, activation='relu'),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')
# Навчання моделі
model.fit(X_train, y_train, epochs=200, batch_size=5,
validation_data=(X_test, y_test))
```

```
# Прогноз на n днів вперед
n_days = 20
last_sequence = data_scaled[-sequence_length:]
predictions = []
for _ in range(n_days):
    pred = model.predict(last_sequence[np.newaxis, :, :])[0, 0]
    predictions.append(pred)
    last_sequence = np.append(last_sequence[1:],
pred).reshape(sequence_length, 1)
predictions =
scaler.inverse_transform(np.array(predictions).reshape(-1, 1))
# Візуалізація результатів
plt.figure(figsize=(12, 6))
plt.plot(data1['Date'], data1['Cases'], label='Original Data',
color='blue')
plt.plot(data2['Date'], data2['Cases'], label='Additional Data',
color='orange')
future_dates = [data1['Date'].iloc[-1] + pd.Timedelta(days=i)
for i in range(1, n_days+1)]
plt.plot(future_dates, predictions, label='Predicted Data',
color='green', linestyle='dashed')
plt.xlabel('Date')
plt.ylabel('Cases')
plt.legend()
plt.title('Disease Forecasting with CNN')
plt.grid()
plt.show()
```

Лістинг Б.4 – Лістинг Z-оцінки

```
import pandas as pd
import numpy as np
# Завантаження даних
data = pd.read_excel('italyData100d.xlsx')
# Конвертація дати у формат datetime
data['Date'] = pd.to_datetime(data['Date'])
# Обчислення середнього значення та стандартного відхилення для
стовпця 'cases'
mean = data['Cases'].mean()  # Середнє значення
std = data['Cases'].std()    # Стандартне відхилення
# Розрахунок Z-оцінки для кожного значення
data['Z-score'] = (data['Cases'] - mean) / std
# Визначення викидів (значення, де |Z-оцінка| > 3)
outliers = data[abs(data['Z-score']) > 3]
# Видалення викидів із набору даних
cleaned_data = data[abs(data['Z-score']) <= 3]
# Збереження очищених даних у новий файл (опціонально)
cleaned_data.to_excel('z_score.xlsx', index=False)
# Виведення результатів
print("Кількість викидів:", len(outliers))
print("Очищені дані збережено у 'cleaned_data_z_score.xlsx'")
```

Лістинг Б.5 – Лістинг Moving Average

```
import pandas as pd
import matplotlib.pyplot as plt
# Завантаження даних
data = pd.read_excel('italyData.xlsx')
data['Date'] = pd.to_datetime(data['Date'])
data.set_index('Date', inplace=True)
# Усунення пропущених значень
data['Cases'].fillna(method='ffill', inplace=True)
# Рухоме середнє (вікно = 7 днів)
data['Moving_Average'] = data['Cases'].rolling(window=7).mean()
# Підготовка даних для збереження (видалення пустих рядків)
output_data =
data[['Moving_Average']].rename(columns={'Moving_Average':
'Cases'})
output_data.reset_index(inplace=True)
output_data = output_data.dropna(subset=['Date', 'Cases'])  #
Видаляємо рядки, де немає Date або Cases
# Збереження даних у новий файл
output_file = 'smoothed_data.xlsx'
output_data.to_excel(output_file, sheet_name='Smoothed Data',
index=False)
print(f"Згладжені дані успішно збережено у файл {output_file}")
# Візуалізація
plt.figure(figsize=(10, 6))
plt.plot(data['Cases'], label='Оригінальні дані', alpha=0.6)
plt.plot(data['Moving_Average'], label='Рухоме середнє (7
днів)', color='red')
plt.title('Згладжування даних методом рухомого середнього')
plt.xlabel('Дата')
plt.ylabel('Кількість випадків')
plt.legend()
plt.show()
```

Лістинг Б.6 – Лістинг теплової карти

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
data = pd.read_excel('cleaned_data.xlsx') # Завантаження даних
missing_data = data.isnull()# Перевірка наявності пропусків
plt.figure(figsize=(10, 6)) # Створення теплової карти
sns.heatmap(missing_data, cbar=False, cmap="viridis",
yticklabels=False)
plt.title("Теплова карта пропусків у наборі даних", fontsize=16)
plt.xlabel("Стовпці")
plt.ylabel("Рядки")
plt.show()
```