ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки

МЕТОДИ КОРОТКОСТРОКОВОГО ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ НА ОСНОВІ ОБЧИСЛЮВАЛЬНОГО ІНТЕЛЕКТУ

Виконав:
маг. гр. СПм-21-2 Понамарьов В. О.

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

2

Актуальність проблеми

Аналіз даних, представлених у вигляді часових рядів, є одним із ключових інструментів у сучасному світі, оскільки дані відіграють все більш важливу роль у прийнятті рішень та плануванні процесів. Серед задач аналізу часових рядів важливим є прогнозування майбутніх значень часового ряду.

Real	6.87	8.88	7.2	7.4	
Forecast MLP	6.6360693	9.247807	7.814073	8.011697	**MLP**
MAE MLP	0.2339307	0.367807	0.614073	0.611697	**0.46**
MAPE MLP	3.405104803	4.141970721	8.528791667	8.266175676	**6**
Forecast LSTM	6.7327895	8.716443	7.2427287	7.3705115	**LSTM**
MAE LSTM	0.1372105	0.163557	0.0427287	0.0294885	**0.09**
MAPE LSTM	1.99724163	1.841858108	0.5934541667	0.3984932432	1
Forecast CNN	7.2089944	9.553276	6.346908	7.7096953	**CNN**
MAE CNN	0.3389944	0.673276	0.853092	0.3096953	**0.54**
MAPE CNN	4.934416303	7.581936937	11.8485	4.185071622	7

3

Поширені методи прогнозування

- штучні нейронні мережі (ШНМ);
- генетичні алгоритми (ГА);
- штучні імунні системи (ШІС).

4

Постановка задачі

Метою роботи є дослідження методів короткострокового прогнозування часових рядів на основі засобів обчислювального інтелекту. Об'єктом дослідження є процес прогнозування часових рядів. Предметом дослідження є використання засобів обчислювального інтелекту для короткострокового прогнозування.

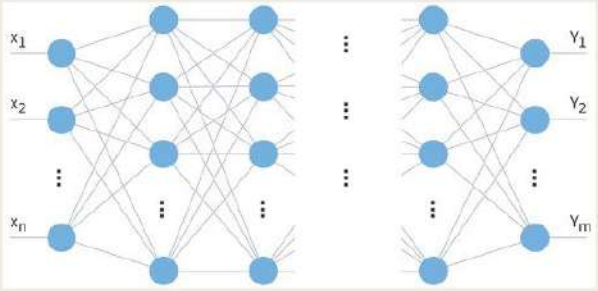
На основі аналізу існуючих досліджень для вирішення задачі були обрані такі моделі штучних нейронних мереж:

- багатошаровий персептрон (MLP);
- довга короткострокова пам'ять (LSTM);
- згорткова нейронна мережа (CNN).

5

Багатошаровий персептрон (MLP)

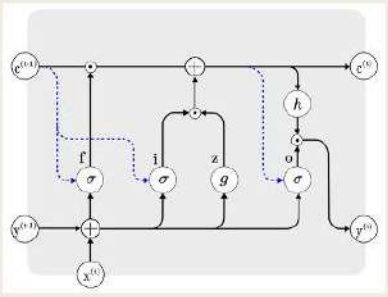
Багатошаровий персептрон (Multi-layer Perceptron, MLP) – це тип нейронної мережі, який складається з декількох шарів нейронів, включаючи вхідний, прихований та вихідний шари, які з'єднані між собою за допомогою зв'язків з певною вагою.



6

Довга короткострокова пам'ять (LSTM)

Довга короткострокова пам'ять (Long Short-Term Memory, LSTM) – це вид рекурентної нейронної мережі, здатний обробляти та зберігати послідовності даних з довгими залежностями між ними. LSTM складається з вхідного шару, комірного шару забування, вхідного комірного шару, шару стану пам'яті та вихідного комірного шару.



7

Згорткова нейронна мережа CNN

Згорткова нейронна мережа (Convolutional Neural Network, CNN) – це глибока нейронна мережа, яка може бути використана для роботи з часовими рядами. CNN складається з згорткових шарів, шару пулінгу та повнозв'язного шару.

8

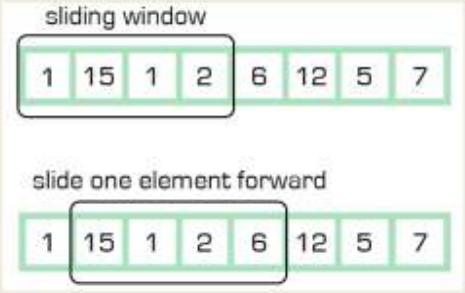
Використані технології

- мова програмування Python;
- фреймворк машинного навчання Tensorflow;
- бібліотека Keras.

9

Етапи роботи

1. Отримання вихідних даних (PM10 по Харкову від SaveDnipro).
2. Підготовка даних до навчання.
3. Налаштування параметрів моделей.
4. Прогнозування.
5. Аналіз результатів прогнозування.



SaveEcoBot

10

Програмна реалізація

- Sequential;
- Dense;
- LSTM;
- Conv1D;
- MaxPooling1D;
- Flatten.

```
def model_mlp(row_sequence, row_sequence_result, row_sequence_in):
    model = Sequential()
    model.add(Dense(64, activation='relu', input_dim=steps))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    model.fit(row_sequence, row_sequence_result, epochs=2000, verbose=0)
    return model.predict(row_sequence_in, verbose=0)
```

```
def model_lstm(row_sequence, row_sequence_result, row_sequence_in):
    model = Sequential()
    model.add(LSTM(64, activation='relu', input_shape=(steps, features)))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    model.fit(row_sequence, row_sequence_result, epochs=2000, verbose=0)
    return model.predict(row_sequence_in, verbose=0)
```

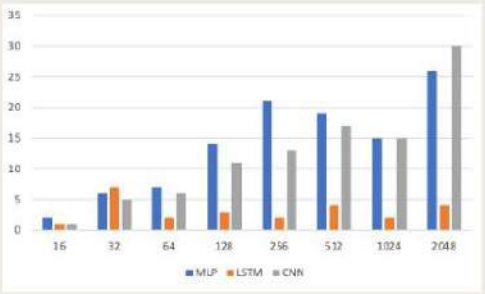
```
def model_cnn(row_sequence, row_sequence_result, row_sequence_in):
    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=2, activation='relu', input_shape=(steps, features)))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    model.fit(row_sequence, row_sequence_result, epochs=2000, verbose=0)
    return model.predict(row_sequence_in, verbose=0)
```

```
MLP: [[7.5132003]]
LSTM: [[7.3916063]]
CNN: [[7.59331]]
```

11

Залежність помилки прогнозу від розміру навчальної вибірки

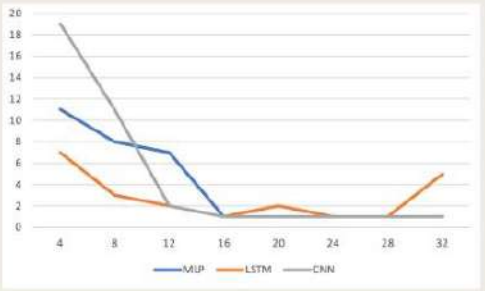
Параметр	Помилка					
	MLP		LSTM		CNN	
row	MAE	MAPE	MAE	MAPE	MAE	MAPE
16	0,07	2	0,03	1	0,05	1
32	0,28	6	0,34	7	0,26	5
64	0,69	7	0,17	2	0,57	6
128	0,3	14	0,1	3	0,38	11
256	1,6	21	0,14	2	1	13
512	1,37	19	0,31	4	1,24	17
1024	1,32	15	0,12	2	1,32	15
2048	1,32	26	0,24	4	1,4	30



12

Залежність помилки від розміру вибірки даних для прогнозування

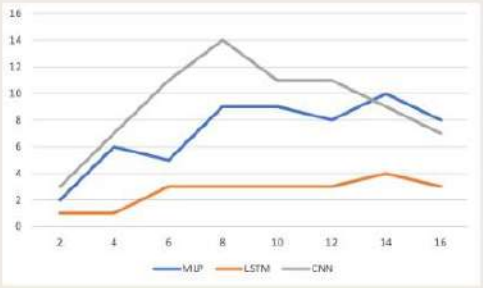
Параметр	Помилка					
	MLP		LSTM		CNN	
predict_row	MAE	MAPE	MAE	MAPE	MAE	MAPE
4	1,12	11	0,6	7	1,58	19
8	0,86	8	0,24	3	1,01	11
12	0,67	7	0,19	2	0,15	2
16	0,13	1	0,11	1	0,1	1
20	0,08	1	0,16	2	0,04	1
24	0,05	1	0,04	1	0,04	1
28	0,11	1	0,11	1	0,05	1
32	0,38	1	0,46	5	0,04	1



13

Залежність помилки прогнозу від горизонту прогнозування

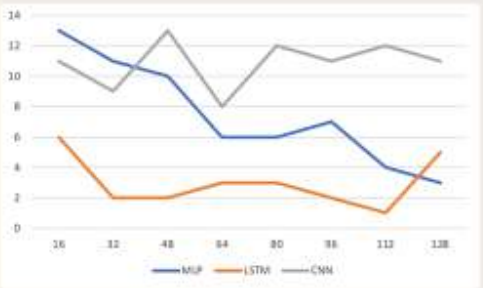
Параметр	Помилка					
	MLP		LSTM		CNN	
horizont	MAE	MAPE	MAE	MAPE	MAE	MAPE
2	0,13	2	0,08	1	0,19	3
4	0,46	6	0,09	1	0,54	7
6	0,34	5	0,23	3	0,79	11
8	0,86	9	0,23	3	1,17	14
10	0,87	9	0,21	3	0,83	11
12	0,73	8	0,22	3	0,99	11
14	0,78	10	0,28	4	0,64	9
16	0,71	8	0,23	3	0,57	7



14

Вплив кількості нейронів у шарі на помилку прогнозу

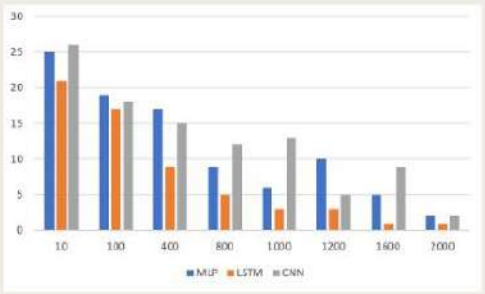
Параметр	Помилка					
	MLP		LSTM		CNN	
units	MAE	MAPE	MAE	MAPE	MAE	MAPE
16	1,18	13	0,6	6	0,88	11
32	1,12	11	0,17	2	0,81	9
48	1	10	0,14	2	1,05	13
64	0,66	6	0,24	3	0,7	8
80	0,59	6	0,2	3	0,96	12
96	0,7	7	0,14	2	0,94	11
112	0,41	4	0,12	1	0,98	12
128	0,35	3	0,36	5	0,96	11



15

Вплив кількості епох навчання на помилку прогнозу

Параметр	Помилка					
	MLP		LSTM		CNN	
epochs	MAE	MAPE	MAE	MAPE	MAE	MAPE
10	2,16	25	2,02	21	2,45	26
100	1,57	19	1,5	17	1,62	18
400	1,45	17	0,83	9	1,35	15
800	0,87	9	0,4	5	1,02	12
1000	0,59	6	0,25	3	1,07	13
1200	1,01	10	0,22	3	0,39	5
1600	0,4	5	0,08	1	0,58	9
2000	0,15	2	0,08	1	0,13	2



16

Висновки

Проаналізовано засоби обчислювального інтелекту для прогнозування часових рядів індексу забруднення повітря. Для роботи були обрані такі моделі, як багатошаровий персептрон, довга короткострокова пам'ять та згорткова нейронна мережа.

Опублікована стаття на тему «Короткострокове прогнозування нестаціонарних часових рядів з використанням моделей MLP та LSTM» у журналі «Системи управління навігації та зв'язку».

Результати роботи представлені на міжнародних науково-технічних конференціях «Проблеми інформатизації» та «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління».

ДОДАТОК Б

Результати тестування

Таблиця 1 – Залежність помилки прогнозу від розміру навчальної вибірки

№	Параметр	Помилка					
		MLP		LSTM		CNN	
	row	MAE	MAPE	MAE	MAPE	MAE	MAPE
1	16	0,07	2	0,03	1	0,05	1
2	32	0,28	6	0,34	7	0,26	5
3	64	0,69	7	0,17	2	0,57	6
4	128	0,3	14	0,1	3	0,38	11
5	256	1,6	21	0,14	2	1	13
6	512	1,37	19	0,31	4	1,24	17
7	1024	1,32	15	0,12	2	1,32	15
8	2048	1,32	26	0,24	4	1,4	30

Таблиця 2 – Залежність помилки від розміру вибірки для прогнозування

№	Параметр	Помилка					
		MLP		LSTM		CNN	
	predict_row	MAE	MAPE	MAE	MAPE	MAE	MAPE
1	4	1,12	11	0,6	7	1,58	19
2	8	0,86	8	0,24	3	1,01	11
3	12	0,67	7	0,19	2	0,15	2
4	16	0,13	1	0,11	1	0,1	1
5	20	0,08	1	0,16	2	0,04	1
6	24	0,05	1	0,04	1	0,04	1
7	28	0,11	1	0,11	1	0,05	1
8	32	0,38	1	0,46	5	0,04	1

Таблиця 3 – Залежність помилки прогнозу від горизонту прогнозування

№	Параметр	Помилка					
		MLP		LSTM		CNN	
	horizont	MAE	MAPE	MAE	MAPE	MAE	MAPE
1	2	0,13	2	0,08	1	0,19	3
2	4	0,46	6	0,09	1	0,54	7
3	6	0,34	5	0,23	3	0,79	11
4	8	0,86	9	0,23	3	1,17	14
5	10	0,87	9	0,21	3	0,83	11
6	12	0,73	8	0,22	3	0,99	11
7	14	0,78	10	0,28	4	0,64	9
8	16	0,71	8	0,23	3	0,57	7

Таблиця 4 – Вплив кількості нейронів у шарі на помилку прогнозу

№	Параметр	MLP		LSTM		CNN	
	units	MAE	MAPE	MAE	MAPE	MAE	MAPE
1	16	1,18	13	0,6	6	0,88	11
2	32	1,12	11	0,17	2	0,81	9
3	48	1	10	0,14	2	1,05	13
4	64	0,66	6	0,24	3	0,7	8
5	80	0,59	6	0,2	3	0,96	12
6	96	0,7	7	0,14	2	0,94	11
7	112	0,41	4	0,12	1	0,98	12
8	128	0,35	3	0,36	5	0,96	11

Таблиця 5 – Загальні результати

MLP		LSTM		CNN	
SMAE	SMAPE	SMAE	SMAPE	SMAE	SMAPE
0,73	10	0,31	6	0,77	12

Таблиця 6 – Вплив кількості епох навчання на помилку прогнозу

№	Параметр	Помилка					
		MLP		LSTM		CNN	
	epochs	MAE	MAPE	MAE	MAPE	MAE	MAPE
1	10	2,16	25	2,02	21	2,45	26
2	100	1,57	19	1,5	17	1,62	18
3	400	1,45	17	0,83	9	1,35	15
4	800	0,87	9	0,4	5	1,02	12
5	1000	0,59	6	0,25	3	1,07	13
6	1200	1,01	10	0,22	3	0,39	5
7	1600	0,4	5	0,08	1	0,58	9
8	2000	0,15	2	0,08	1	0,13	2

ДОДАТОК В

Вихідний код розробленого програмного засобу

Лістинг 1 – Фрагмент лістингу розробленого програмного засобу

```
from numpy import array
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Conv1D
from keras.layers import MaxPooling1D
from keras.layers import Flatten
def split(sequence, steps):
    row_sequence, row_sequence_result = list(), list()
    for i in range(len(sequence)):
        row_sequence_len = i + steps
        if row_sequence_len > len(sequence) - 1:
            break
        row_sequence_list, row_sequence_result_list =
sequence[i:row_sequence_len], sequence[row_sequence_len]
        row_sequence.append(row_sequence_list)
        row_sequence_result.append(row_sequence_result_list)
    return array(row_sequence), array(row_sequence_result)
def model_mlp(row_sequence, row_sequence_result,
row_sequence_in):
    model = Sequential()
    model.add(Dense(64, activation='relu',
input_dim=steps))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    model.fit(row_sequence, row_sequence_result,
epochs=2000, verbose=0)
    return model.predict(row_sequence_in, verbose=0)
def mlp():
    row_sequence, row_sequence_result = split(row, steps)
    row_sequence_in = array(predict)
    row_sequence_in = row_sequence_in.reshape((samples,
steps))
    result = model_mlp(row_sequence, row_sequence_result,
row_sequence_in)
    print('MLP:', result)
def model_lstm(row_sequence, row_sequence_result,
row_sequence_in):
    model = Sequential()
    model.add(LSTM(64, activation='relu',
input_shape=(steps, features)))
```

```
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    model.fit(row_sequence, row_sequence_result,
epochs=2000, verbose=0)
    return model.predict(row_sequence_in, verbose=0)
def lstm():
    row_sequence, row_sequence_result = split(row, steps)
    row_sequence =
row_sequence.reshape((row_sequence.shape[0],
row_sequence.shape[1], features))
    row_sequence_in = array(predict)
    row_sequence_in = row_sequence_in.reshape((samples,
steps, features))
    result = model_lstm(row_sequence, row_sequence_result,
row_sequence_in)
    print('LSTM:', result)
def model_cnn(row_sequence, row_sequence_result,
row_sequence_in):
    model = Sequential()
    model.add(Conv1D(filters=64, kernel_size=2,
activation='relu', input_shape=(steps, features)))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    model.fit(row_sequence, row_sequence_result,
epochs=2000, verbose=0)
    return model.predict(row_sequence_in, verbose=0)
def cnn():
    row_sequence, row_sequence_result = split(row, steps)
    row_sequence =
row_sequence.reshape((row_sequence.shape[0],
row_sequence.shape[1], features))
    row_sequence_in = array(predict)
    row_sequence_in = row_sequence_in.reshape((samples,
steps, features))
    result = model_cnn(row_sequence, row_sequence_result,
row_sequence_in)
    print('CNN:', result)
def models():
    mlp()
    lstm()
    cnn()
if __name__ == '__main__':
    models()
```