ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки

Короткострокове прогнозування середньорічної температури повітря за допомогою штучних нейронних мереж

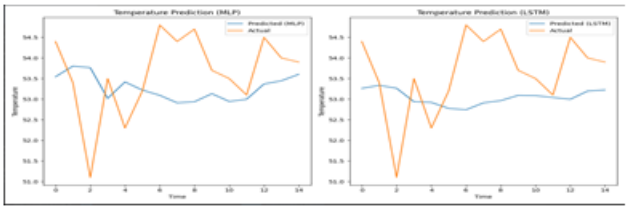
Виконав:
маг. гр. СПм-22-4 Кононенко О. М.

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

Актуальність проблеми

На сьогоднішній день, прогнозування температури є вкрай актуальним завданням. Від результатів прогнозування можуть залежати економічний стан, фінансовий стан, та інші.

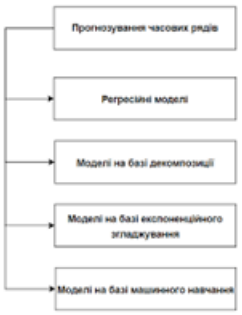
Інформація про середньорічну температуру повітря, може бути представлена у вигляді часових рядів, тому для вирішення завдання, буде доцільним використання методів аналізу та прогнозування часових рядів

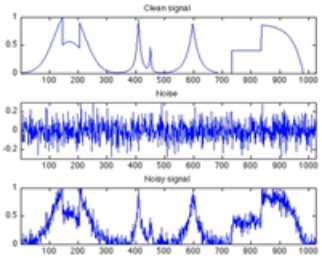


3

Поширені методи прогнозування

- штучні нейронні мережі (ШНМ);
- наївні методи прогнозування;
- штучні імунні системи (ШІС);
- авторегресійні моделі: ARIMA, ARMA.



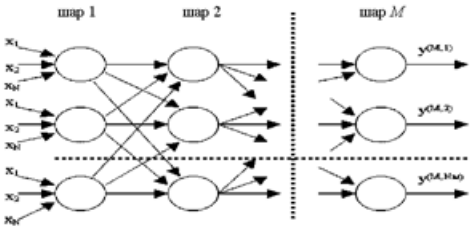


5

Багатошаровий персептрон (MLP)

Багатошаровий персептрон (MLP) — це тип штучної нейронної мережі, який належить до класу алгоритмів глибокого навчання. Він складається з трьох основних типів шарів:

- вхідний шар;
- приховані шари:
- вихідний шар.

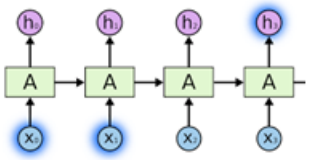


6

Довга короткострокова пам'ять (LSTM)

Архітектура LSTM є модифікацією RNN мережі, в якій структура осередків пам'яті ускладнена, щоб збільшити запам'ятовуючу здатність мережі та уповільнити забування вже вивченої інформації.

Сьогодні LSTM (Long short-term memory) є однією з найкращих архітектур для прогнозування часових рядів.



7

Використані технології

- мова програмування Python;
- фреймворк машинного навчання Tensorflow;
- бібліотеки візуалізації: Seaborn та Matplotlib;
- бібліотеки обробки та аналізу даних: Pandas та Numpy
- бібліотека Keras.

pandas

8

Програмна реалізація

- Sequential;
- Dense;
- LSTM;
- MLP.

```
model_mlp.compile(optimizer='adam',
loss='mse')
# Тренування моделі MLP
model_mlp.fit(X_train_mlp, y_train_mlp,
epochs=50, batch_size=32, validation_data=
(X_test_mlp, y_test_mlp), verbose=2)
```

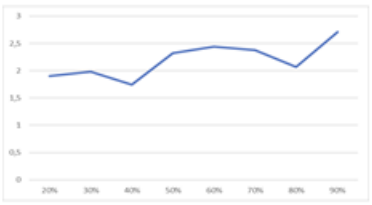
```
model_lstm = Sequential()
model_lstm.add(LSTM(50, activation='relu',
input_shape=(seq_length, 1)))
model_lstm.add(Dense(horizon))
model_lstm.compile(optimizer='adam',
loss='mse')
model_lstm.fit(X_train_lstm, y_train_lstm,
epochs=50, batch_size=32, validation_data=
(X_test_lstm, y_test_lstm), verbose=2)
predictions_lstm =
model_lstm.predict(X_test_lstm)
predictions_lstm =
scaler.inverse_transform(predictions_lstm)
```

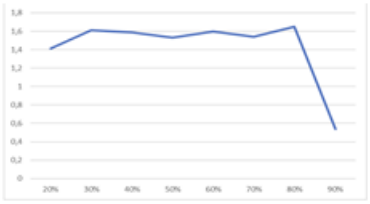
```
def create_sequences(data, seq_length,
horizon):
    sequences = []
    labels = []
    for i in range(len(data) - seq_length -
horizon + 1):
        seq = data[i:i + seq_length]
        label = data[i + seq_length:i +
seq_length + horizon]
        sequences.append(seq)
        labels.append(label)
    return np.array(sequences),
np.array(labels)
seq_length = 5  # Довжина послідовності
horizon = 5   # Горизонт прогнозу
X_train_lstm, y_train_lstm =
create_sequences(train_normalized,
seq_length, horizon)
X_test_lstm, y_test_lstm =
```

9

Залежність помилки прогнозу від розміру навчальної вибірки

Параметр	Помилка			
	MLP		LSTM	
train_size	MAE	MAPE	MAE	MAPE
20	0,74	1,41	1,00	1,9
30	0,84	1,61	1,04	1,98
40	0,84	1,59	0,91	1,74
50	081	1,53	1,24	2,32
60	0,85	1,6	1,31	2,44
70	0,83	1,54	1,29	2,38
80	0,88	1,65	1,11	2,07
90	0,29	0,54	1,47	2,71

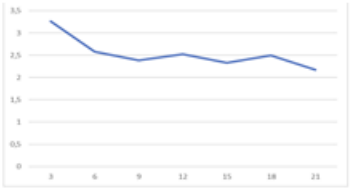


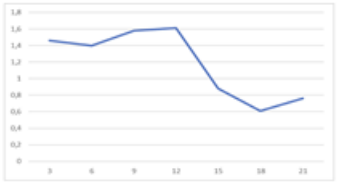


10

Залежність помилки від розміру вибірки даних для прогнозування

Параметр	Помилка			
	MLP		LSTM	
predict_row	MAE	MAPE	MAE	MAPE
3	0,78	1,46	1,77	3,26
6	0,75	1,4	1,39	2,58
9	0,84	1,58	1,29	2,39
12	0,86	1,61	1,36	2,52
15	0,48	0,88	1,26	2,33
18	0,33	0,61	1,36	2,5
21	0,41	0,76	1,18	2,17

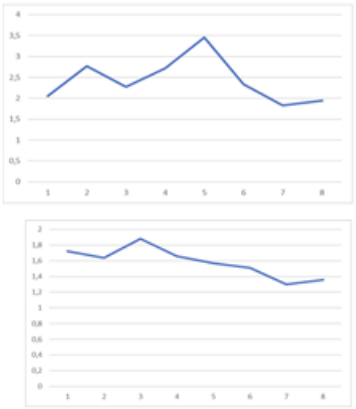




11

Залежність помилки прогнозу від горизонту прогнозування

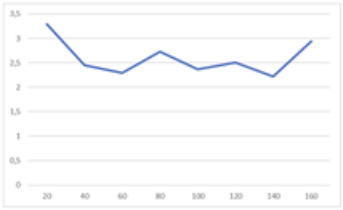
Параметр	Помилка			
	MLP		LSTM	
horizon	MAE	MAPE	MAE	MAPE
1	0,92	1,72	1,11	2,05
2	0,87	1,64	1,49	2,77
4	1,01	1,88	1,22	2,27
6	0,89	1,66	1,47	2,72
8	0,84	1,57	1,86	3,45
10	0,81	1,51	1,26	2,34
12	0,7	1,3	0,99	1,83
14	0,72	1,36	1,05	1,95

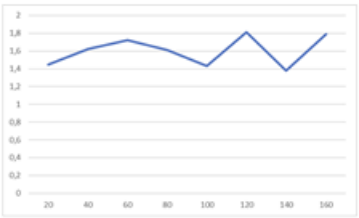


12

Вплив кількості нейронів у шарі на помилку прогнозу

Параметр	Помилка			
	MLP		LSTM	
units	MAE	MAPE	MAE	MAPE
20	0,77	1,45	1,77	3,29
40	0,872	1,62	1,32	2,45
60	0,92	1,72	1,24	2,29
80	0,86	1,61	1,47	2,73
100	0,77	1,43	1,28	2,37
120	0,97	1,81	1,36	2,51
140	0,74	1,38	1,19	2,22
160	0,96	1,79	1,59	2,94

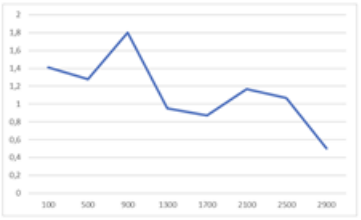


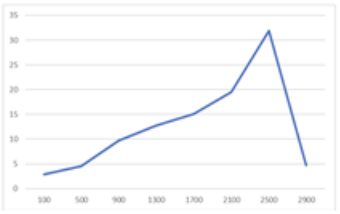


13

Вплив кількості епох навчання на помилку прогнозу

Параметр	Помилка			
	MLP		LSTM	
epochs	MAE	MAPE	MAE	MAPE
100	0,76	1,41	1,57	2,91
500	0,69	1,28	2,47	4,58
900	0,97	1,8	5,20	9,71
1300	0,51	0,95	6,86	12,78
1700	0,47	0,87	8,11	15,11
2100	0.63	1,17	10,44	19,49
2500	0,58	1,07	17,09	31,94
2900	0,27	0,5	2,51	4,69





14

Висновки

Проаналізовано моделі штучних нейронних мереж для прогнозування середньорічної температури повітря. Для роботи були обрані такі моделі, як багатошаровий персептрон та довга короткострокова пам'ять.

Модель багатошарового персептрону показала високу точність прогнозування температури.

Це свідчить про те, що моделі MLP можуть бути ефективними інструментами для прогнозування часових рядів температури за умови ретельної оптимізації їхніх параметрів.

Опублікована стаття на тему "Моделі глибокого навчання для прогнозування часових рядів" у журналі "Системи управління, навігації та зв'язку".

ДОДАТОК Б

Вихідний код розробленого програмного засобу

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error,
mean_absolute_percentage_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
def mean_absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
# Набір даних
data = {
    'Year': [1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907,
1908, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917,
             1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925,
1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934, 1935,
             1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943,
1944, 1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953,
             1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961,
1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971,
             1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979,
1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989,
             1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997,
1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007,
             2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015,
2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023],
    'Temperature': [53.9, 53.5, 52.1, 50.6, 51.8, 51.7, 51.2,
52.4, 50.1, 51.8, 54.5, 51.2, 50.9, 51.3, 53.2, 53.2, 51.1,
                    50.7, 52.3, 52.5, 51.3, 54.4, 51.2, 50.8,
51.5, 53.1, 53.9, 52.5, 50.7, 49.3, 49.4, 49.9, 48.2, 50.5,
                    55.3, 51.9, 52.2, 51.4, 52.0, 51.7, 53.8,
51.5, 49.7, 52.2, 49.4, 50.5, 52.0, 51.6, 50.6, 50.1, 51.6,
                    50.4, 51.1, 53.2, 52.8, 49.6, 51.5, 51.7,
53.6, 51.7, 51.9, 52.3, 50.6, 50.9, 48.3, 50.5, 52.4, 51.9,
                    50.3, 52.0, 51.6, 50.5, 52.3, 51.4, 52.9,
50.8, 51.6, 53.2, 53.3, 52.7, 52.8, 54.3, 51.0, 53.5, 51.0,
                    51.9, 52.7, 53.4, 53.3, 52.2, 53.4, 51.8,
53.6, 50.3, 54.7, 53.9, 54.3, 53.8, 52.6, 53.9, 54.0, 53.5,
                    54.8, 53.9, 54.7, 54.2, 55.0, 53.3, 53.1,
54.4, 53.4, 51.1, 53.5, 52.3, 53.2, 54.8, 54.4, 54.7, 53.7,
                    53.5, 53.1, 54.5, 54.0, 53.9]
}
# Створення DataFrame
df = pd.DataFrame(data)
# Перевірка даних у DataFrame
```

```
print(df.head())
# Функція для створення послідовностей та міток
def create_sequences(data, seq_length, horizon):
    sequences = []
    labels = []
    for i in range(len(data) - seq_length - horizon + 1):
        seq = data[i:i + seq_length]
        label = data[i + seq_length:i + seq_length + horizon]
        sequences.append(seq)
        labels.append(label)
    return np.array(sequences), np.array(labels)
# Визначення параметрів
seq_length = 10  # Довжина послідовності
horizon = 6      # Горизонт прогнозу
# Розділення даних на тренувальний та тестовий набори
train_size = int(len(df) * 0.8)
train, test = df.iloc[:train_size], df.iloc[train_size:]
# Нормалізація даних
scaler = MinMaxScaler()
train_normalized = scaler.fit_transform(train[['Temperature']])
test_normalized = scaler.transform(test[['Temperature']])
# Створення датасету для LSTM моделі
X_train_lstm, y_train_lstm = create_sequences(train_normalized,
seq_length, horizon)
X_test_lstm, y_test_lstm = create_sequences(test_normalized,
seq_length, horizon)
# Перетворення в формат, придатний для LSTM
X_train_lstm = X_train_lstm.reshape((X_train_lstm.shape[0],
seq_length, 1))
X_test_lstm = X_test_lstm.reshape((X_test_lstm.shape[0],
seq_length, 1))
# Створення та компіляція моделі LSTM
model_lstm = Sequential()
model_lstm.add(LSTM(50, activation='relu',
input_shape=(seq_length, 1)))
model_lstm.add(Dropout(0.2))
model_lstm.add(Dense(horizon))  # Змінено кількість вихідних
нейронів для горизонту
model_lstm.compile(optimizer='adam', loss='mean_squared_error')
# Навчання моделі LSTM
model_lstm.fit(X_train_lstm, y_train_lstm, epochs=200,
batch_size=32, validation_split=0.2, verbose=2)
# Прогнозування на тестовому наборі для LSTM
predictions_lstm = model_lstm.predict(X_test_lstm)
Кононенко, [22.06.2024 12:16]
# Інвертування нормалізації
predictions_lstm =
scaler.inverse_transform(predictions_lstm.reshape(-1, horizon))
y_test_lstm = scaler.inverse_transform(y_test_lstm.reshape(-1,
horizon))
# Розрахунок MAE і MAPE для LSTM
mae_lstm = mean_absolute_error(y_test_lstm, predictions_lstm)
mape_lstm = mean_absolute_percentage_error(y_test_lstm,
```

```
predictions_lstm)
print(f"LSTM MAE: {mae_lstm}")
print(f"LSTM MAPE: {mape_lstm:.2f}%")
# Відображення результатів для LSTM
plt.figure(figsize=(12, 6))
plt.plot(range(len(y_test_lstm)), y_test_lstm[:, 0],
label='Actual')  # Використовуємо перший стовпець для
відображення
plt.plot(range(len(predictions_lstm)), predictions_lstm[:, 0],
label='Predicted')  # Те саме для прогнозу
plt.title('LSTM Model: Actual vs Predicted')
plt.xlabel('Time')
plt.ylabel('Temperature')
plt.legend()
plt.show()
# Створення датасету для MLP моделі
X_train_mlp = X_train_lstm.reshape((X_train_lstm.shape[0],
seq_length))
X_test_mlp = X_test_lstm.reshape((X_test_lstm.shape[0],
seq_length))
# Створення та компіляція моделі MLP
model_mlp = Sequential()
model_mlp.add(Dense(50, activation='relu',
input_shape=(seq_length,)))
model_mlp.add(Dropout(0.2))
model_mlp.add(Dense(horizon))
model_mlp.compile(optimizer='adam', loss='mean_squared_error')
# Навчання моделі MLP
model_mlp.fit(X_train_mlp, y_train_lstm, epochs=200,
batch_size=32, validation_split=0.2, verbose=2)
# Прогнозування на тестовому наборі для MLP
predictions_mlp = model_mlp.predict(X_test_mlp)
predictions_mlp = scaler.inverse_transform(predictions_mlp)
y_test_mlp = scaler.inverse_transform(y_test_lstm[:,
:horizon].reshape(-1, horizon))
# Розрахунок MAE і MAPE для MLP
mae_mlp = mean_absolute_error(y_test_mlp, predictions_mlp)
mape_mlp = mean_absolute_percentage_error(y_test_mlp,
predictions_mlp)
print(f"MLP MAE: {mae_mlp}")
print(f"MLP MAPE: {mape_mlp:.2f}%")
# Відображення результатів для MLP
plt.figure(figsize=(12, 6))
plt.plot(range(len(y_test_mlp)), y_test_mlp[:, 0],
label='Actual')
plt.plot(range(len(predictions_mlp)), predictions_mlp[:, 0],
label='Predicted')
plt.title('MLP Model: Actual vs Predicted')
plt.xlabel('Time')
plt.ylabel('Temperature')
plt.legend()
plt.show()
```