ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Харківський національний університет радіоелектроніки

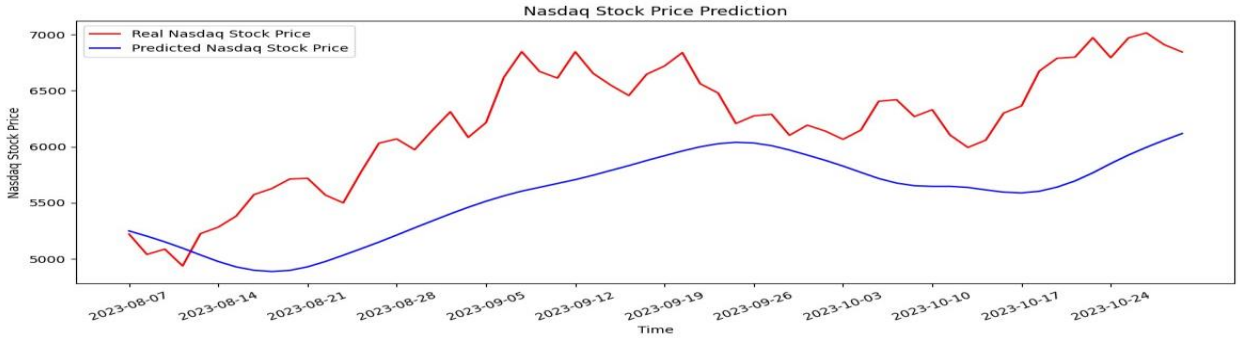
МЕТОДИ ПРОГНОЗУВАННЯ НЕСТАЦІОНАРНИХ ЧАСОВИХ РЯДІВ НА ОСНОВІ РЕКУРЕНТНИХ НЕЙРОННИХ МЕРЕЖ

Виконав:
маг. гр. СПм-22-3 Близнюк О. В.

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

Актуальність задачі

Актуальність теми:
•Прогнозування часових рядів є критично важливим для багатьох галузей, таких як фінанси, медицина, енергетика, логістика та інші.
•Сучасні умови вимагають точних і швидких прогнозів для прийняття обґрунтованих рішень.
Практичне значення:
•Ефективне прогнозування може запобігти втратам, оптимізувати ресурси та підвищити ефективність управління.



3

Розглянуте дослідження «Forecasting energy consumption using LSTM neural networks»

Ціль дослідження:
Прогнозування енергоспоживання з використанням LSTM.
Аналіз часових рядів:
Фактори**:** погода, час доби, день тижня, святкові дні.
Дані за останні 5 років: щогодинне споживання, температура, вологість, календарні дані.
Розробка та навчання моделі:
Модель LSTM навчалась на 80% даних, тестування на 20%.
Результат:
Точність прогнозування: MAE = 2,5, MSE = 9,6.
Оптимізація управління для енергетичних компаній.

4

Розглянуте дослідження «Weather prediction using recurrent neural networks»

Ціль:
Прогнозування погоди з використанням RNN.
Аналіз часових рядів:
Дані останніх 10 років з щогодинною деталізацією.
Розробка моделі:
Навчання RNN на 70% даних, тестування на 30%.
Висновки:
RNN показала точність: MAE - 1.2, MSE - 2.5, краще ніж традиційні методи.

Постановка задачі

5

- **Ціль:**
  - Розробка ефективних методів прогнозування нестаціонарних часових рядів за допомогою RNN.
- **Планується:**
  - Проведення експериментів для оцінки впливу гіперпараметрів на результати прогнозування.

Використовувані підходи

6

Рекурентні нейронні мережі (RNN)

- **Основи:**
  - Обробка послідовних даних.
  - Збереження контексту попередніх станів.
- **Типи:**
  - **Прості RNN**
  - **LSTM:** Довгострокова пам'ять.
  - **GRU:** Спрощена LSTM.

Застосування:
•Часові ряди, текст, мова.
Переваги:
•Врахування попередніх даних.
•Гнучкість.
Виклики:
•Проблеми з навчанням.
•Високі обчислювальні вимоги.

7

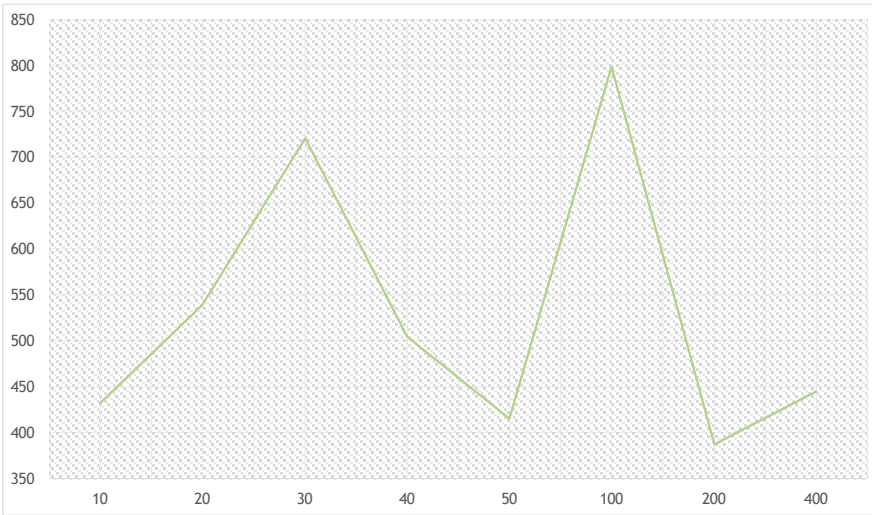
Використовувані технології

- **Використані технології**
- **PyCharm:** середовище розробки для мови Python.
- **Pandas:** обробка та аналіз даних.
- **NumPy:** числові обчислення.
- **Matplotlib:** візуалізація даних.
- **Scikit-learn:** попередня обробка даних.
- **TensorFlow/Keras:** створення та навчання моделей RNN.
- **CSV:** збереження результатів.

8

Експеримент: вплив кількості епох

- Фіксовані параметри: Розмір пакета = 32, горизонт прогнозу = 60, розмір вікна = 60.

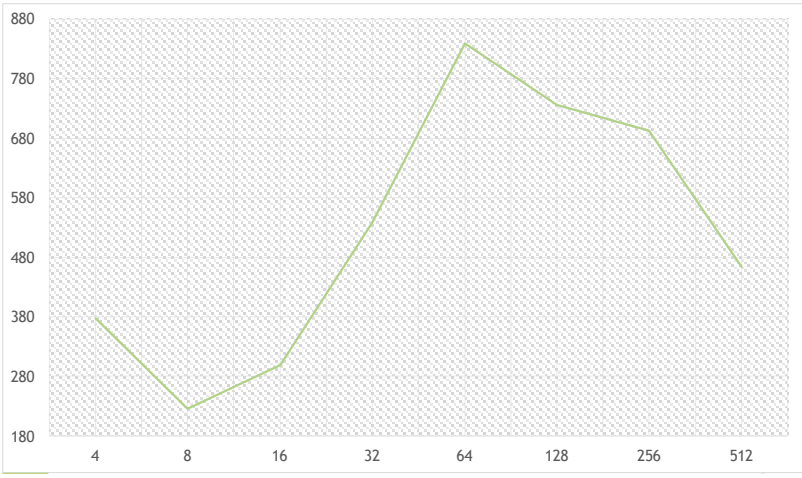


№	epochs	MAE
1	10	432,538105
2	20	538,8836417
3	30	720,51431
4	40	504,2608033
5	50	414,938895
6	100	798,7181417
7	200	387,615905
8	400	445,1998941

9

Експеримент: вплив розміру пакета

- Фіксовані параметри: Епохи = 50, горизонт прогнозу = 60, розмір вікна = 60.

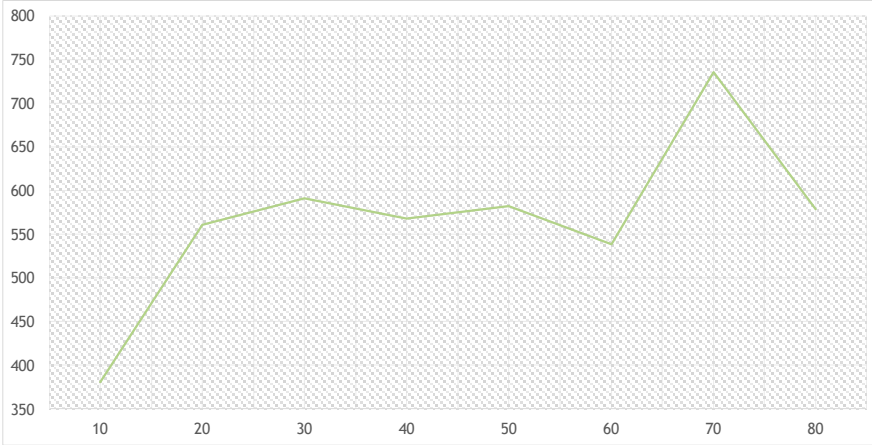


№	batch_size	MAE
1	4	376,8841933
2	8	226,0692767
3	16	298,6666667
4	32	538,8836417
5	64	838,9370217
6	128	735,8446367
7	256	692,5444767
8	512	464,2444767

10

Експеримент: вплив горизонту прогнозу

- Фіксовані параметри: Епохи = 50, розмір пакета = 32, розмір вікна = 60.



№	horizont	MAE
1	10	380,64595
2	20	561,054165
3	30	591,5938533
4	40	567,704755
5	50	582,081448
6	60	538,8836417
7	70	735,6843657
8	80	578,2885225

11

Експеримент: вплив розміру вікна

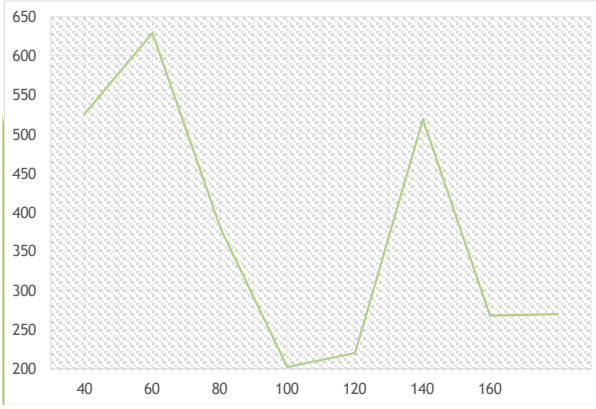
- Фіксовані параметри: Епохи = 50, розмір пакета = 32, горизонт прогнозу = 60.

№	Window_size	MAE
1	20	501,6021617
2	40	718,8064467
3	60	795,3677633
4	80	856,4461783
5	100	651,6773917
6	120	760,8953783
7	140	766,7327533
8	160	623,05766

12

Експеримент: вплив кількості нейронів у шарах

- Фіксовані параметри: Епохи = 50, розмір пакета = 32, горизонт прогнозу = 60, розмір вікна = 60, кількість шарів = 3.



№	Нейронів у шарі (3 шари)	MAE	MAPE
1	40	524,9315837	8,272021379
2	60	628,4324137	9,873688085
3	80	379,9315999	5,959300303
4	100	201,9811442	3,232335981
5	120	219,8934163	3,534161258
6	140	518,7171387	8,154393758
7	160	268,1054362	4,239372226
8	180	269,90271	4,246480152

Висновки

•**Ефективність RNN:** RNN є ефективним методом для прогнозування нестаціонарних часових рядів, показуючи високу точність і робастність.

•**Обмеження RNN:** Виявлено складність навчання та високі обчислювальні витрати, що є основними обмеженнями у застосуванні RNN.

•**Необхідність досліджень:** Потребується подальше дослідження для оптимізації та підвищення ефективності RNN в прогнозуванні часових рядів.

•**Майбутній розвиток:** З розвитком технологій та оптимізації методів можна очікувати подальшого зростання використання RNN для прогнозування часових рядів.

•**Публікація:** В процесі роботи опублікована стаття на тему «Моделі глибокого навчання для прогнозування часових рядів» у журналі «Системи управління навігації та зв'язку», №1(75).

ДОДАТОК Б

Фрагмент вихідного коду розробленого програмного засобу

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
data = pd.read_csv('nasdaq.csv', date_format = True)
data.head(10)
data_training = data[data['Date']<'2023-01-01'].copy()
data_test = data[data['Date']>='2023-01-01'].copy()
data_training = data_training.drop(['Date', 'Adj Close'], axis =
1)
scaler = MinMaxScaler()
data_training = scaler.fit_transform(data_training)
window_size = 60
data_training[0:5]
X_train = []
y_train = []
for i in range(window_size, data_training.shape[0]):
    X_train.append(data_training[i-window_size:i])
    y_train.append(data_training[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
model = Sequential()
model.add(LSTM(units = 60, activation = 'relu', return_sequences
= True, input_shape = (X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(LSTM(units = 60, activation = 'relu', return_sequences
= True))
model.add(Dropout(0.2))
model.add(LSTM(units = 80, activation = 'relu', return_sequences
= True))
model.add(Dropout(0.2))
model.add(LSTM(units = 120, activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 1))
model.compile(optimizer='adam', loss = 'mean_squared_error')
model.fit(X_train, y_train, epochs=50, batch_size=32)
data_test.head()
data_training = data[data['Date']<'2023-01-01'].copy()
past_days = data_training.tail(window_size)
df = pd.concat([past_days, data_test], ignore_index=True)
df = df.drop(['Date', 'Adj Close'], axis = 1)
df.head()
inputs = scaler.transform(df)
inputs
max_forecast_horizon = 60
```

```
X_test = []
y_test = []
for i in range(window_size, min(inputs.shape[0], window_size +
max_forecast_horizon)):
    X_test.append(inputs[i-window_size:i])
    y_test.append(inputs[i, 0])
X_test, y_test = np.array(X_test), np.array(y_test)
X_test.shape, y_test.shape
y_pred = model.predict(X_test)
scaler.scale_
scale = 1/scaler.scale_[0]
scale
y_pred = y_pred*scale
y_test = y_test*scale
results_df = pd.DataFrame({'Date': data_test['Date'].values[-
y_test.shape[0]:],
                           'Actual': y_test,
                           'Predicted': y_pred.flatten()})
merged_df = pd.merge(data_test, results_df, on='Date',
how='inner')
mae = np.mean(np.abs(merged_df['Actual'] -
merged_df['Predicted']))
mape = np.mean(np.abs((merged_df['Actual'] -
merged_df['Predicted']) / merged_df['Actual'])) * 100
summary_df = pd.DataFrame({'Date': [None],
                           'Actual': [None],
                           'Predicted': [None],
                           'MAE': [mae],
                           'MAPE': [mape]})
summary_df = summary_df.dropna(axis=1, how='all')
final_df = pd.concat([summary_df, merged_df], ignore_index=True)
final_df.to_csv('таблица.csv', index=False, columns=['Date',
'Actual', 'Predicted', 'MAE', 'MAPE'])
plt.figure(figsize=(14,5))
plt.plot(merged_df['Date'], merged_df['Actual'], color='red',
label='Real Nasdaq Stock Price')
plt.plot(merged_df['Date'], merged_df['Predicted'],
color='blue', label='Predicted Nasdaq Stock Price')
plt.title('Nasdaq Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Nasdaq Stock Price')
plt.legend()
plt.xticks(plt.xticks()[0][::5], rotation=25)
plt.savefig('график.png', bbox_inches='tight')
```