

**Міністерство освіти і науки України
Національний технічний університет України «КПІ» імені Ігоря Сікорського
Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ
з лабораторної роботи №6
з навчальної дисципліни «Технології Data Science»**

Тема:

РЕАЛІЗАЦІЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

Виконав:

Студент 4 курсу кафедри ФІОТ,
Навчальної групи ПП-11
Олександр Головня

Перевірив:

Професор кафедри ОТ ФІОТ
Олексій Писарчук

I. Мета:

Виявити дослідити та узагальнити особливості підготовки різних типів даних, синтезу, навчання та застосування штучних нейронних мереж (Artificial Neural Networks).

II. Завдання:

Розробити програмний скрипт мовою Python що реалізує обчислювальний алгоритм за технологіями штучних нейронних мереж (Artificial Neural Networks): підготовка даних; конструювання нейромережі; навчання штучної нейронної мережі; застосування нейромережі (класифікація / ідентифікація / прогнозування):

III рівень складності 9 балів за самостійним вибором напрямку:

1. Відповідно до технічних умов, табл.2 додатку, але в якості Data Set – обрати реальні дані у форматі числових / часових рядів, наприклад, як результат виконання лабораторних робіт із статистичного навчання (парсинг самостійно обраного сайту).

III. Результати виконання лабораторної роботи.

Блок схема алгоритму:

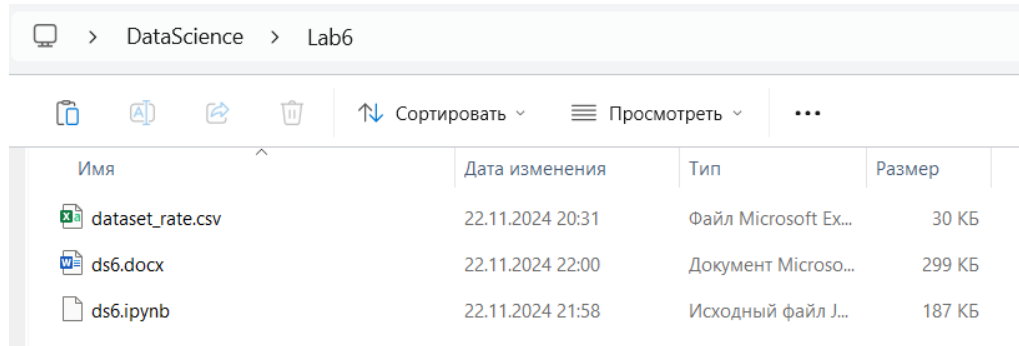


Рис.1 – Блок-схема алгоритму програми

3.1. Опис структури проекту програми.

Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей інтегрованого середовища сформовано проект.

Проект базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.



Имя	Дата изменения	Тип	Размер
dataset_rate.csv	22.11.2024 20:31	Файл Microsoft Ex...	30 КБ
ds6.docx	22.11.2024 22:00	Документ Microso...	299 КБ
ds6.ipynb	22.11.2024 21:58	Исходный файл J...	187 КБ

Рис.2 – Структура проекту

ds6.ipynb – файл програмного коду лабораторної роботи;

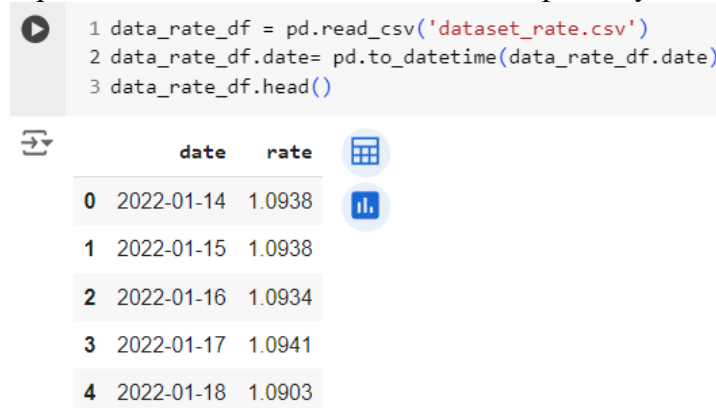
ds6.docx – файл звіту лабораторної роботи

dataset_rate.xlsx – dataset

3.2. Результати роботи програми відповідно до завдання.

Для виконання цього завдання було обрано реальний набір даних, який містить числові значення, що відповідають за значення курсу валют чи інших показників у форматі часового ряду.

Як і в попередній роботі, для завантаження даних використовується функція `read_csv`



```
1 data_rate_df = pd.read_csv('dataset_rate.csv')
2 data_rate_df.date= pd.to_datetime(data_rate_df.date)
3 data_rate_df.head()
```

	date	rate
0	2022-01-14	1.0938
1	2022-01-15	1.0938
2	2022-01-16	1.0934
3	2022-01-17	1.0941
4	2022-01-18	1.0903

Рис 3.1 – Зчитування даних у датафрейм

Після завантаження даних додано перетворення стовпця з датами у формат `datetime`, що дозволяє зручніше працювати з часом. Потім здійснюється нормалізація стовпця "rate" за допомогою методу `MinMaxScaler` з бібліотеки `sklearn.preprocessing`, щоб привести значення до діапазону від 0 до 1.

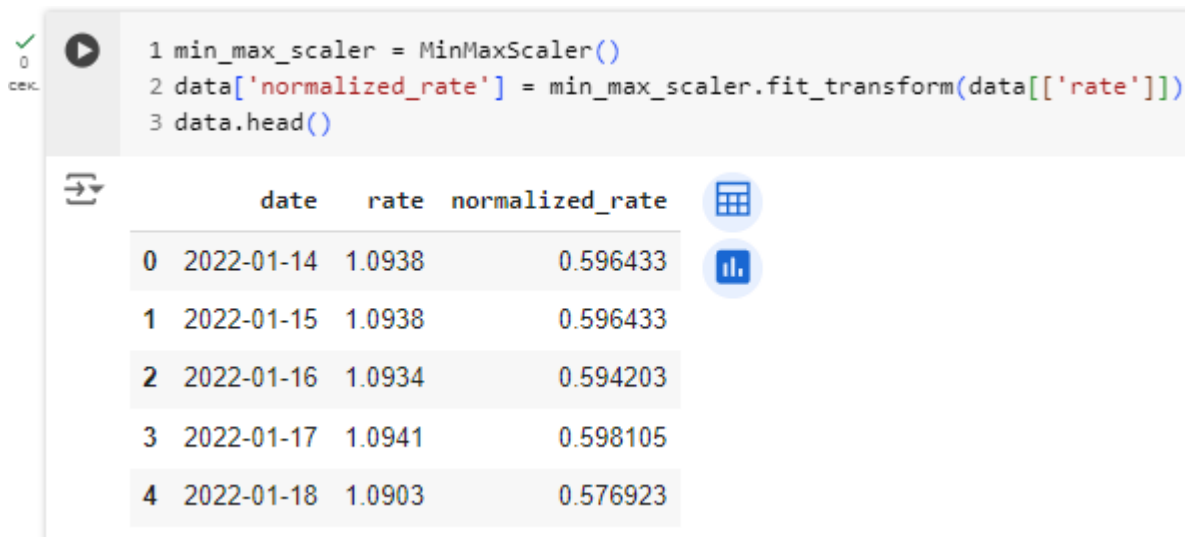


Рис. 3.2 – Візуалізація даних з нормалізацією

Наступним кроком є Створення наборів для навчання та тестування: Для побудови моделі необхідно поділити дані на тренувальний та тестовий набори. У цьому випадку використовувалося співвідношення 80/20, де 80% даних йде на навчання, а 20% — на тестування

Функція `create_dataset` використовується для створення вхідних та вихідних даних для нейромережі з використанням параметра `look_back`. Це визначає, скільки попередніх значень будуть використовуватися для прогнозування поточного значення:

```
[7] 1 def create_dataset(dataset, look_back):
2     X, Y = [], []
3     for i in range(len(dataset) - look_back - 1):
4         X.append(dataset[i:(i + look_back)])
5         Y.append(dataset[i + look_back])
6     return np.array(X), np.array(Y)
7
8
9 look_back = 7
10 train_size = int(len(data) * 0.8)
11 train_data, test_data = data[0:train_size], data[train_size:len(data)]
12 X_train, y_train = create_dataset(train_data['normalized_rate'].values, look_back)
13 X_test, y_test = create_dataset(test_data['normalized_rate'].values, look_back)
```

Рис. 3.3 – Створення наборів

Далі йде створення та налаштування нейронної мережі: Нейронна мережа створена за допомогою бібліотеки Keras. Вона містить:

Вхідний шар, що приймає `look_back` значень;

Приховані шари з функцією активації ReLU;

Вихідний шар для прогнозування одного значення (наприклад, ціни чи курсу).

Модель компілюється за допомогою оптимізатора Adam та функції втрат `mean_squared_error`, яка є стандартом для задач регресії

Для навчання моделі використовуються дані для тренування (`X_train` та `y_train`) з функцією `fit`. Для запобігання перенавчання були додані два колбеки:

- `EarlyStopping`: Припиняє навчання, якщо валідаційна втрата не зменшується після заданої кількості епох.

- `ModelCheckpoint`: Зберігає найкращу модель, яку можна завантажити в подальшому.

```

✓ 19 [8] 1 model = Sequential()
      2 model.add(Dense(8, input_dim=look_back, activation='relu'))
      3 model.add(Dense(4, activation='relu'))
      4 model.add(Dense(1))
      5
      6 model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error')
      7 early_stopping = EarlyStopping(monitor='val_loss', patience=20)
      8 model_checkpoint = ModelCheckpoint('best_model.keras', monitor='val_loss', save_best_only=True)
      9
     10
     11 history = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test),
     12                   verbose=1,
     13                   callbacks=[early_stopping,
     14                             model_checkpoint])
     14
  41/41 ————— 0s 3ms/step - loss: 9.9517e-04 - val_loss: 7.1025e-04
Epoch 73/100
 41/41 ————— 0s 3ms/step - loss: 0.0012 - val_loss: 5.9564e-04
Epoch 74/100
 41/41 ————— 0s 2ms/step - loss: 0.0011 - val_loss: 5.8836e-04
Epoch 75/100
 41/41 ————— 0s 3ms/step - loss: 9.3750e-04 - val_loss: 6.2240e-04
Epoch 76/100
 41/41 ————— 0s 2ms/step - loss: 0.0013 - val_loss: 7.3936e-04
Epoch 77/100
 41/41 ————— 0s 3ms/step - loss: 0.0012 - val_loss: 5.9394e-04
Epoch 78/100
 41/41 ————— 0s 3ms/step - loss: 0.0011 - val_loss: 5.7458e-04
Epoch 79/100
 41/41 ————— 0s 2ms/step - loss: 0.0019 - val_loss: 5.9227e-04
Epoch 80/100
 41/41 ————— 0s 3ms/step - loss: 0.0015 - val_loss: 6.5293e-04
Epoch 81/100
 41/41 ————— 0s 3ms/step - loss: 0.0010 - val_loss: 5.7346e-04
Epoch 82/100
 41/41 ————— 0s 2ms/step - loss: 0.0017 - val_loss: 5.9337e-04
Epoch 83/100
 41/41 ————— 0s 2ms/step - loss: 0.0014 - val_loss: 5.9728e-04
Epoch 84/100
 41/41 ————— 0s 2ms/step - loss: 0.0020 - val_loss: 5.9160e-04
Epoch 85/100
 41/41 ————— 0s 3ms/step - loss: 0.0011 - val_loss: 5.7383e-04
Epoch 86/100
 41/41 ————— 0s 3ms/step - loss: 8.3905e-04 - val_loss: 7.9351e-04

```

Рис. 3.4 – Навчання мережі

Оцінка моделі: Після завершення навчання модель зберігається, і можна оцінити її на тестових даних:

```

⇒ 10/10 ————— 0s 2ms/step - loss: 4.7329e-04
Validation loss: 0.00057

```

Рис. 3.5 – Оцінка моделі

Для візуалізації процесу навчання та прогнозів побудовані два графіки:

Графік втрат під час навчання, що показує, як змінювалася втрата як на тренувальних, так і на тестових даних:

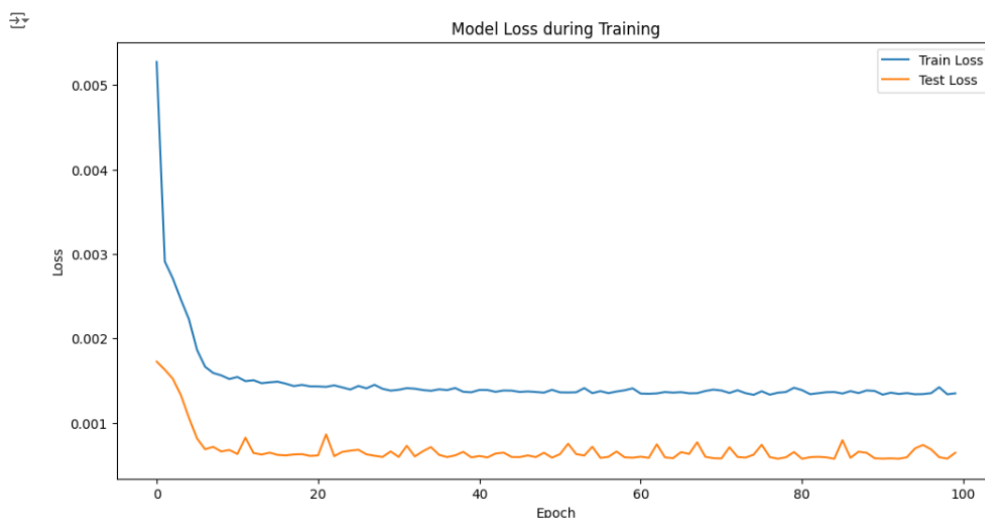


Рис. 3.6 – Графік втрат під час навчання

Графік порівняння реальних та прогнозованих значень:

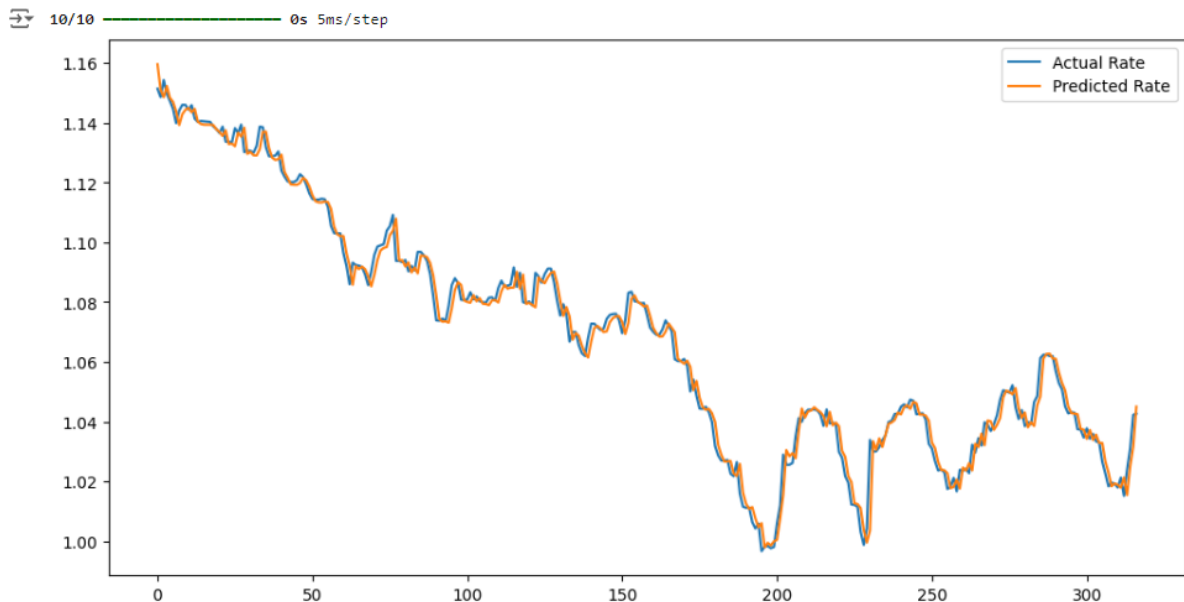


Рис. 3.6 – Графік порівняння реальних та прогнозованих значень

Цей підхід може бути адаптований для прогнозування різноманітних числових або часових рядів, таких як ціни на акції, курси валют або прогнози погоди.

3.3. Програмний код.

Програмний код послідовно реалізує алгоритм рис.1 та спрямовано на отримання результатів, поданих вище.

При цьому використано можливості Python бібліотек: `pip`; `pandas`; `numpy`; `sklearn`; `matplotlib`.

Контексні коментарі пояснюють сутність окремих скриптів наведеного коду програми.

3.4. Аналіз результатів відлагодження та верифікації результатів роботи програми.

Результати відлагодження та тестування довели працездатність розробленого коду. Це підтверджується результатами розрахунків, які не суперечать теоретичним положенням.

Верифікація функціоналу програмного коду, порівняння отриманих результатів з технічними умовами завдання на лабораторну роботу доводять, що усі завдання виконані у повному обсязі.

IV. Висновки.

Отже, в ході даної лабораторної я у результаті виконання програми побудував модель нейронної мережі для прогнозування числових даних часового ряду. Модель була навчена на тренувальних даних, а потім оцінена на тестовому наборі з використанням метрики втрат.

Основні кроки:

1. Підготовка та нормалізація даних.
2. Створення тренувальних та тестових наборів.
3. Конструювання нейронної мережі.
4. Навчання моделі з колбеками для запобігання перенавчанню.
5. Оцінка моделі та візуалізація результатів.