**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 ІП-11

Олександр Головня

**Перевірив:**

Професор кафедри ОТ ФІОТ Олексій Писарчук

**Київ 202**

**І. Мета:**

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

**ІІ. 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):

**ІІІ 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).

**ІІІ. Результати виконання лабораторної роботи.**

Блок схема алгоритму:

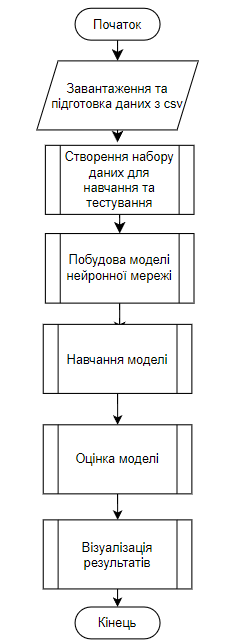


Рис.1 – Блок-схема алгоритму програми

* 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.

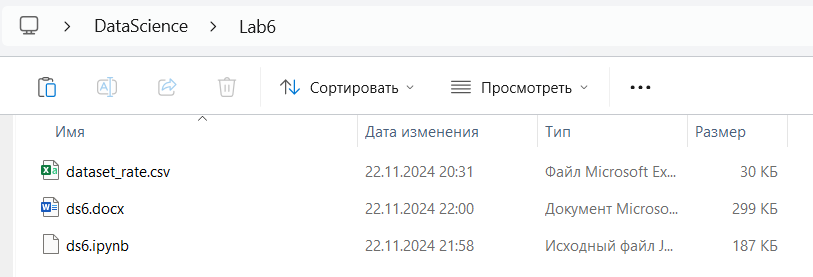


Рис.2 – Структура проєкту

ds6.ipynb – файл програмного коду лабораторної роботи;

ds6.docx – файл звіту лабораторної роботи

dataset\_rate.xlsx – dataset

* 1. **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

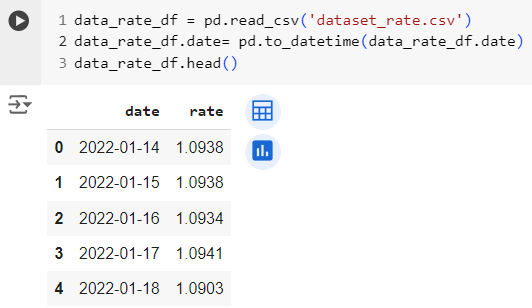


Рис 3.1 – Зчитування даних у датафрейм

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.

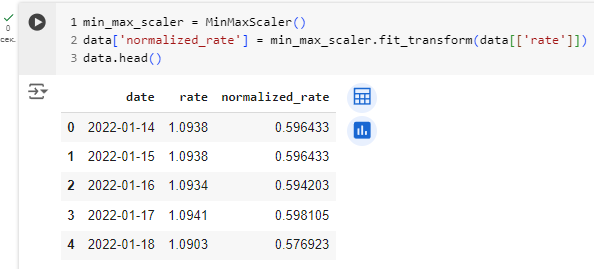


Рис. 3.2 – Візуалізація даних з нормалізацією

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:

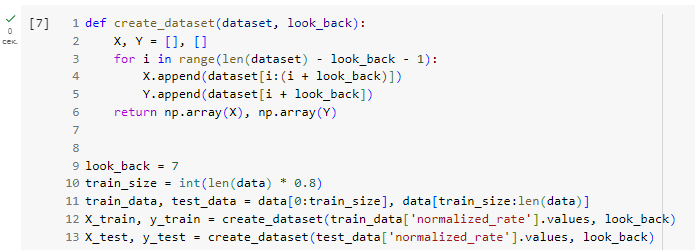


Рис. 3.3 – Створення наборів

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.

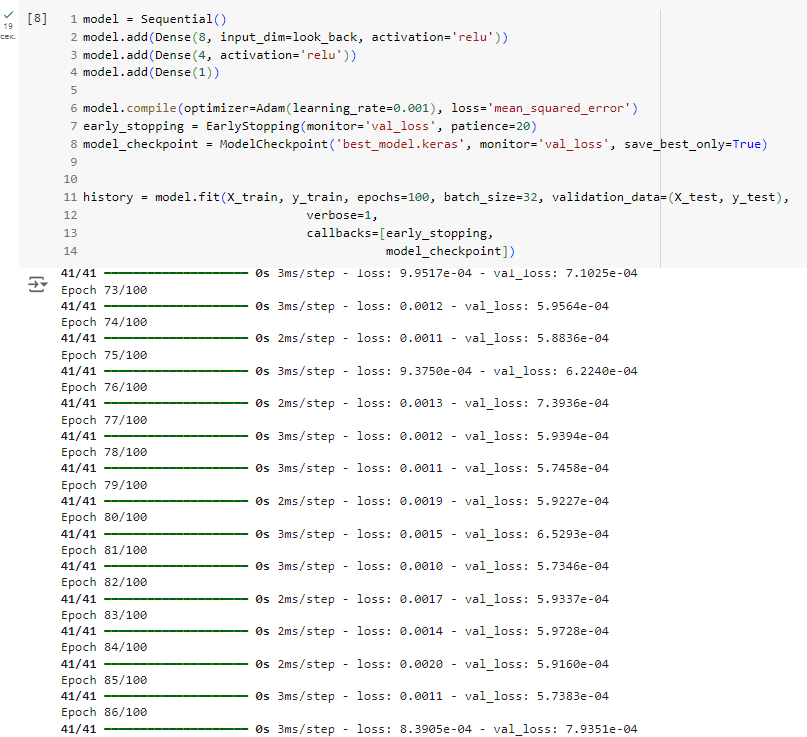


Рис. 3.4 – Навчання мережі

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



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:

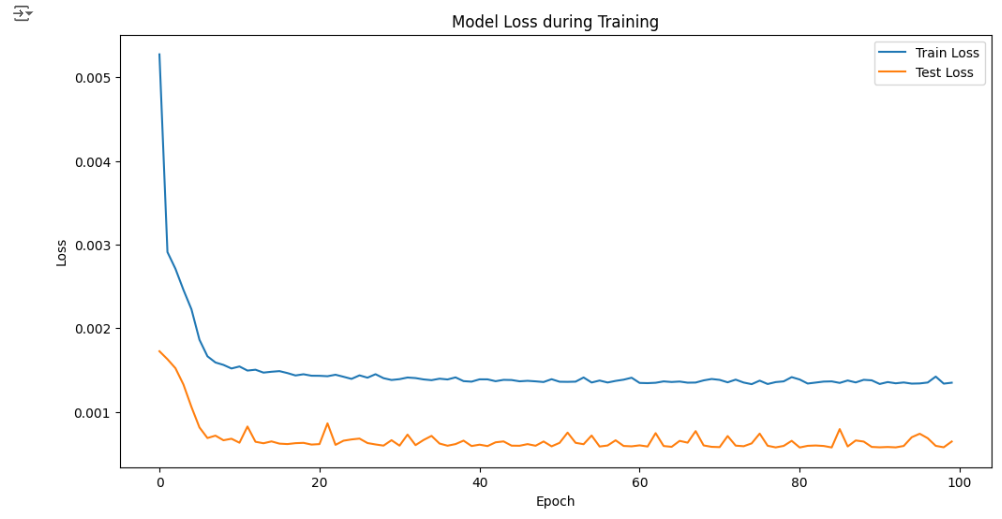


Рис. 3.6 – Графік втрат під час навчання

Графік порівняння реальних та прогнозованих значень:

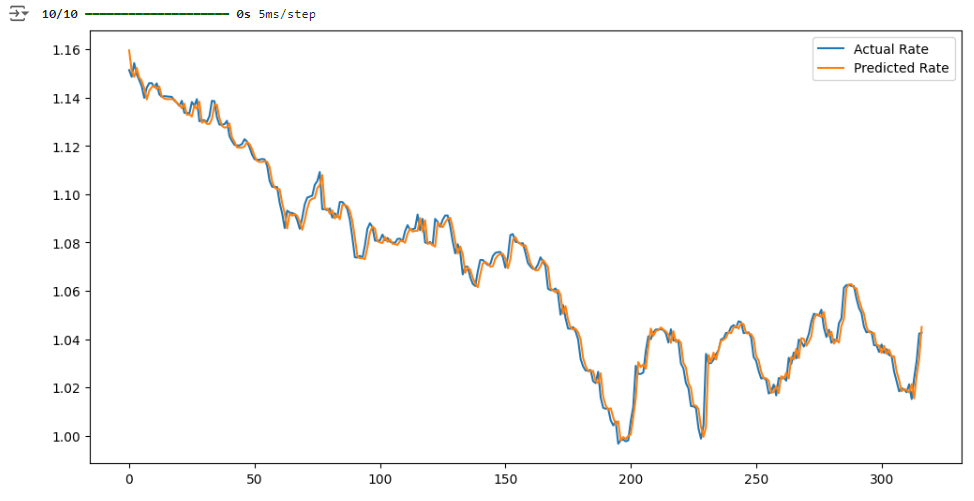


Рис. 3.6 – Графік порівняння реальних та прогнозованих значень

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

* 1. **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.

* 1. **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.