

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИНТЕЗУ
НЕЙРОМЕРЕЖЕВИХ ДІАГНОСТИЧНИХ МОДЕЛЕЙ ДЛЯ
ПРОГНОЗУВАННЯ ФІНАНСУВАННЯ СІРІТ У ЗАКЛАДАХ ВИЩОЇ
ОСВІТИ

RESEARCH AND SOFTWARE IMPLEMENTATION OF THE SYNTHESIS OF
NEURAL NETWORK DIAGNOSTIC MODELS TO FORECAST FINANCING
THE ORPHANS AT HIGHER EDUCATION INSTITUTIONS

Виконав: студент(ка) 2 курсу, групи КНТ-122м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ЦИБУЛЬСЬКИЙ К.Д.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТЕРЕЩЕНКО Е.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЦИБУЛЬСЬКОГО Костянтина Дмитровича

(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація синтезу
нейромережових діагностичних моделей для прогнозування фінансування сиріт
у закладах вищої освіти. Research and software implementation of the synthesis of
neural network diagnostic models to forecast financing the orphans at Higher
Education Institutions

керівник проєкту (роботи) к.т.н., доцент, СТЕПАНЕНКО Олександр
Олексійович

(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “14” листопада 2023 року
№ 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір та обґрунтування структури
системи, яка проєктується. 3. Основні рішення щодо реалізації компонентів
системи. 4. Керівництво програміста. 5. Керівництво оператора.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень,
кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “ 04 ” вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1-2 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2-3 тиждень	Розділ 2
4	Розробка архітектури програми.	4-5 тиждень	Розділ 3
5	Розробка програми.	5-7 тиждень	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	8 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї.	9-10 тиждень	Додатки
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент

_____ Костянтин ЦИБУЛЬСЬКИЙ
(підпис) (ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Олександр СТЕПАНЕНКО
(підпис) (ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
82 с., 4 табл., 19 рис., 4 дод., 23 джерел.

ДІТИ-СИРОТИ, ФІНАНСУВАННЯ, PYTHON, TENSORFLOW,
НЕЙРОМЕРЕЖА, ПРОГНОЗУВАННЯ, ЗАКЛАД ВИЩОЇ ОСВІТИ, KERAS

Об'єкт дослідження – фінансування сиріт у закладах вищої освіти.

Предмет дослідження – методи та алгоритми синтезу нейромережових діагностичних моделей для прогнозування фінансування сиріт у закладах вищої освіти.

Мета роботи – розробка нейромережових діагностичних моделей для прогнозування фінансування сиріт у закладах вищої освіти.

Матеріали, методи та технічні засоби: інтернет-браузер Google Chrome, сервіс Google Scholar, сервіс Google Colab, Adobe Reader, IDE Visual Studio Code, Python, TensorFlow.

Актуальність розробки обумовлена використанням передових технологій управління та аналізу даних, зокрема нейромережових моделей, що дозволяє покращити точність прогнозування та оптимізувати процеси управління фінансами. А також загостренням проблеми із збільшенням кількості сиріт у наслідок бойових дій та наданням дітям-сиротам та дітям, позбавленим батьківського піклування належних умов для здобуття вищої освіти.

Вищеназвані проблеми вирішені за допомогою створення інструментів для фінансування сиріт у закладах вищої освіти на основі нейромережових діагностичних моделей.

В результаті роботи було розроблено нейромережову діагностичну модель яка здатна робити прогноз на основі статистичних даних.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 82 pages, 19 figures, 4 tables, 4 appendixes, 23 sources.

ORPHANS, FUNDING, PYTHON, TENSORFLOW, NEUROMERAGE, FORECASTING, INSTITUTION OF HIGHER EDUCATION, KERAS

The object of the research is the financing of orphans in institutions of higher education.

The subject of the research is the methods and algorithms of the synthesis of neural network diagnostic models for forecasting the financing of orphans in institutions of higher education.

The purpose of the work is to develop neural network diagnostic models for forecasting the financing of orphans in higher education institutions.

Materials, methods and technical means: Internet browser Google Chrome, Google Scholar service, Google Colab service, Adobe Reader, Visual Studio Code IDE, Python, TensorFlow.

The relevance of the development is due to the modern possibilities of processing and predicting data to increase the efficiency and accuracy of obtaining results based on this data. As well as the aggravation of the problem with the increase in the number of orphans as a result of hostilities and providing orphans and children deprived of parental care with proper conditions for obtaining higher education.

The above-mentioned problems were solved by creating tools for financing orphans in higher education institutions based on neural network diagnostic models.

As a result of the work, a neural network diagnostic model was developed that is able to make a forecast based on statistical data.

ЗМІСТ

Реферат.....	4
Перелік скорочень та умовних познач 8	8
Вступ	9
1 Аналіз предметної області	11
1.1 Діти-сироти: визначення та проблематика	11
1.2 Штучні нейронні мережі та їх архітектури.....	13
1.2.1 Рекурентні нейронні мережі.....	17
1.2.2 Нейромережі з довгою короткочасною пам'яттю (LSTM)	18
1.2.3 Вентельний рекурентний вузол	17
1.3 Постановка завдання до роботи	21
1.4 Висновки за розділом 1	21
2 Вибір та обґрунтування структури проектованої системи.....	22
2.1 Вибір інструментарію розробки.....	22
2.1.1 Вибір середовища розробки	22
2.1.2 Вибір мови програмування.....	23
2.2 Бібліотеки Python для машинного навчання	26
2.3 Висновки за розділом 2.....	29
3 Основні рішення щодо реалізації компонентів системи	30
3.1 Загальний принцип роботи моделі LSTM.....	30
3.2 Функції активації	36
3.3 Бібліотека NumPy	37
3.4 Побудова графіків за допомогою Matplotlib.....	37
3.5 Збір та представлення даних	38
3.6 Висновки за розділом 3	39
4 Керівництво програміста	40
4.1 Призначення й умови використання	40
4.2 Характеристики програмного продукту	40
4.3 Звернення до програмного застосунку.....	41

	7
4.4 Вхідні і вихідні дані.....	41
4.5 Повідомлення.....	44
4.6 Висновки за розділом 4	44
5 Керівництво користувача.....	45
5.1 Призначення й умови виконання.....	45
5.2 Виконання програмного продукту.....	45
5.3 Повідомлення оператора	46
5.4 Висновки за розділом 5	46
Висновки.....	47
Перелік джерел посилання	48
Додаток А Технічне завдання.....	50
Додаток Б Опис програмного продукту.....	55
Додаток В Текст програми	62
Додаток Г Слайди презентації.....	75

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ДС – діти-сироти

LSTM – Long Short-Term Memory

ПК – персональний комп'ютер

ПЗ – програмний застосунок

IDE – Integrated Development Environment

RNN – Recurrent Neural Network

ШНМ – Штучна нейронна мережа.

ЗВО – Заклад Вищої Освіти;

НМ – Нейронна мережа;

ВСТУП

У сучасному світі питання соціальної відповідальності та вирішення проблем соціально вразливих груп стає все більш актуальним і належить до пріоритетних завдань суспільства. Однією з таких груп є діти-сироти, які, втративши сімейне забезпечення, часто змушені стикатися із складними життєвими ситуаціями та обмеженнями можливостей.

Високий рівень соціальної відповідальності вимагає розглядати не лише питання безпосередньої допомоги, але й надавати рівні можливості для отримання вищої освіти дітям-сиротам. Висвітлення цієї теми стає необхідним, оскільки вища освіта є ключовим фактором для соціальної інтеграції та розвитку особистості.

Україна, як і багато інших країн, розглядає питання освіти як один із пріоритетів. Дитячі будинки стають місцем проживання для багатьох дітей-сиріт, і їхній перехід у вищі навчальні заклади часто вимагає додаткової уваги та ресурсів.

Проблеми, з якими стикаються діти-сироти при вступі у вищі навчальні заклади, багатоаспектні. Це включає фінансові труднощі, соціальну адаптацію, вибір професійного шляху та забезпечення необхідною підтримкою під час навчання.

Магістерська дипломна робота присвячена дослідженню та програмній реалізації синтезу нейромережових діагностичних моделей для прогнозування фінансування дітей-сиріт у закладах вищої освіти. Цей проект має на меті розробку ефективних інструментів для аналізу та прогнозування фінансових потреб цільової групи.

Робота ґрунтується на ідеї того, що використання передових технологій управління та аналізу даних може значно полегшити процеси фінансового планування та сприяти більш ефективному використанню ресурсів для забезпечення доступу дітей-сиріт до вищої освіти.

Ця робота виконана для вдосконалення системи фінансування вищої освіти для дітей-сиріт, та в контексті розвитку соціально відповідальної освітньої системи, яка забезпечує рівні можливості для кожного студента, незалежно від його соціального походження.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Далі буде розглянуто основні визначення та особливості предметної області, проведено аналіз та порівняння існуючих нейромережових моделей, сформульовано переваги обраної моделі, сформована постановка задачі.

1.1 Діти-сироти: визначення та проблематика

Закон України «Про забезпечення організаційно-правових умов соціального захисту дітей-сиріт та дітей, позбавлених батьківського піклування» який визначає правові, організаційні, соціальні засади та гарантії державної підтримки дітей-сиріт та дітей, позбавлених батьківського піклування, а також осіб із їх числа, і є складовою частиною законодавства про охорону дитинства, дає наступні визначення.

Дитина-сирота - дитина, в якої померли чи загинули батьки.

Діти, позбавлені батьківського піклування, - діти, які залишилися без піклування батьків у зв'язку з позбавленням їх батьківських прав, відібранням у батьків без позбавлення батьківських прав, визнанням батьків безвісно відсутніми або недієздатними, оголошенням їх померлими, відбуванням покарання в місцях позбавлення волі та перебуванням їх під вартою на час слідства, розшуком їх органами Національної поліції, пов'язаним з відсутністю відомостей про їх місцезнаходження, тривалою хворобою батьків, яка перешкоджає їм виконувати свої батьківські обов'язки, а також діти, розлучені із сім'єю, підкинуті діти, батьки яких невідомі, діти, від яких відмовилися батьки, діти, батьки яких не виконують своїх батьківських обов'язків з причин, які неможливо з'ясувати у зв'язку з перебуванням батьків на тимчасово окупованій території України, в районах проведення антитерористичної операції, здійснення заходів із забезпечення національної безпеки і оборони, відсічі і стримування збройної агресії Російської Федерації у Донецькій та Луганській областях, та безпритульні діти [1]-[2].

Так згідно чинного законодавства діти-сироти та діти, позбавленні батьківського піклування знаходяться на цілковитому державному забезпеченні. При чому у разі продовження навчання пільги зберігаються до 23 років або до закінчення навчальних закладів. Також держава гарантує для таких осіб вільний вибір напрямку освіти та професійної діяльності [2].

Загальні види пільг для дітей-сиріт у ВНЗ можна побачити на схемі, що зображена на рис. 1.1.



Рисунок 1.1 – Пільги для дітей-сиріт [3]

Діти-сироти часто стикаються із складнішими умовами життя та обмеженим доступом до різних ресурсів, включаючи фінансові. Це може призвести до обмеженого доступу до вищої освіти, внаслідок високих витрат на навчання.

Також багато дітей-сиріт не мають стабільних джерел доходу та фінансової підтримки, що робить навчання у вищому навчальному закладі великим фінансовим викликом. Вони можуть стикатися із проблемами в оплаті навчання, проживання та інших пов'язаних витрат.

Саме для запобігання дискримінації таких соціальних груп та забезпечення рівних умов у здобуванні освіти державою й прийнято низку заходів.

Але для їх успішного впровадження необхідно залучення додаткового

фінансування. Його розмір не є сталим і планується як частина річного бюджету для закладів вищої освіти.

При закладені такого роду фондів вплив мають наступні чинники:

- кількість абітурієнтів, які вступили в учбовому році;
- похибка кількості вступивших до ЗВО;
- відсоток інфляції;
- субвенція (харчування, проживання, одяг, проїзд тощо)
- заробітна платня викладачів ЗВО;
- середній відсоток дітей-сиріт, що вступили до ЗВО;
- похибка на відсоток дітей-сиріт, що закінчили навчання;
- стипендія для дітей-сиріт, що вступили та навчаються у ЗВО;
- вік абітурієнтів;
- напрям навчання.

Маючи вищезгаданий перелік параметрів та статистичні дані за великий проміжок часу можна перекласти тягар з прогнозування кількості коштів, що треба закласти у відповідний бюджет на діагностичні моделі штучних нейронних мереж, які останнім часом набирають популярність та розширюють коло свого застосування.

Існують різноманітні підходи до організації архітектур таких мереж. Нижче буде дано визначення штучним нейронним мережам та розглянуті основні архітектури побудови нейромереж, що підходять для рішення задач, які розглядаються у цій роботі.

1.2 Штучні нейронні мережі та їх архітектури

Штучні нейронні мережі (ШНМ) – це математичні моделі та їх програмне та\або апаратне виконання, що наслідують принцип побудови і функціонування нервових клітин живих організмів.

Надихаючись мозком живих істот, були побудовані системи, де

головним елементом виступає штучний нейрон, що імітує біологічні.

ШНМ – це система просто зв'язаних і інтерактивних процесорів (штучних нейронів). Ці процесори, як правило, досить прості, особливо в порівнянні з процесорами, які використовуються в персональних комп'ютерах. Кожен процесор у одноранговій мережі обробляє лише сигнали, які він регулярно отримує, і сигнали, які він регулярно надсилає іншим процесорам. І все ж ці локально прості процесори, об'єднані в досить велику мережу з керованою взаємодією, здатні разом виконувати досить складні завдання [4].

Математично нейрон - це нелінійна функція, яка залежить від значень вхідних сигналів, їх ваг і активатора.

Вага - це фактор, за допомогою якого може змінюватися інформація, що передається між структурними підрозділами.

Активатор вирішує, при яких вхідних значеннях нейрон повинен передавати сигнал.

Мережа складається з нейронів, розташованих шарами. На вхід надходять дані, а на виході – результат моделі. Між ними є приховані шари. Вони спеціалізуються на обробці інформації, але розробники ШІ не завжди розуміють, що відбувається за лаштунками [5].

У випадку коли прихований шар один, то мережа називається неглибокою, якщо їх більше, то глибокою. Приклади таких мереж можна побачити на рис. 1.2 та рис. 1.3.

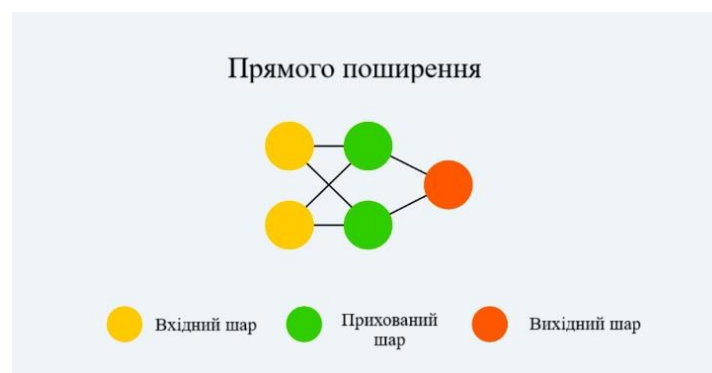


Рисунок 1.2 – Неглибока ШНМ

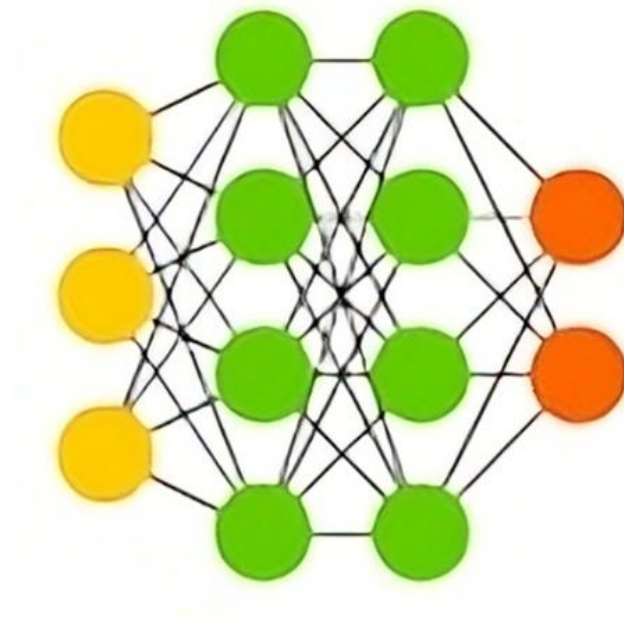


Рисунок 1.3 – Глибока ШНМ

Для того щоб визначитись з використовуваною моделлю виберемо декілька, на мою думку, найбільш підходящих, складемо загальну порівняльну таблицю та розглянемо кожну з них.

Таблиця 1.1 – Порівняння моделей нейромереж

Характеристика	Рекурентні нейронні мережі (RNN)	Довготривалі і краткострокові пам'яті (LSTM)	GRU мережі	Глибокі нейронні мережі (DNN)
Основна ідея	Враховує попередні вхідні дані за допомогою зв'язків в часі.	Розширення RNN, яке забезпечує зберігання та використання довготривалих залежностей.	Використовує вентиля для управління потоком інформації, ефективно для довгих послідовностей.	Мережі з багатьма шарами нейронів для моделювання складних залежностей.

Продовження таблиці 1.1

Характеристика	Рекурентні нейронні мережі (RNN)	Довготривалі краткострокові пам'яті (LSTM)	GRU мережі	Глибокі нейронні мережі (DNN)
Застосування	Прогнозування часових рядів, рекурентні зв'язки в даних.	Прогнозування складних динамічних процесів, довготривалі зв'язки в даних.	Прогнозування з використанням вентилів, ефективно для довгих послідовностей.	Прогнозування параметрів з великою кількістю взаємозв'язків.
Можливі проблеми	Проблема зникнення/згасання градієнту при тривалих послідовностях.	Можливий ефект зникнення/згасання градієнту; схильність до перенавчання.	Може бути менше схильним до перенавчання, але все ще можливі проблеми.	Може вимагати великої кількості даних для тренування; ризик перенавчання.
Ефективність	Ефективні для послідовних даних, таких як часові ряди.	Добре пристосовані для моделювання довготривалих залежностей.	Добре справляється з довгими послідовностями, швидше за LSTM.	Можуть бути дуже ефективними для великих обсягів даних та складних завдань.
Застосування в галузі фінансів	Прогнозування фінансових часових рядів.	Прогнозування фінансових тенденцій та складних ринкових умов.	Прогнозування фінансових параметрів з використанням вентилів.	Моделювання фінансових процесів та оптимізація портфеля.

Глибокі нейронні мережі (DNN) були мною відкинуті через їх надлишковість для поставленої задачі. Інші розглянемо детальніше.

1.2.1 Рекурентні нейронні мережі

Рекурентні нейронні мережі (RNN) - це мережі, схожі на FFNN, але з деякою особливістю: нейрони отримують інформацію не тільки від попереднього шару, але й від самих себе з попередньої ітерації. Це означає, що порядок введення даних і навчання мережі стає важливим. Головною проблемою з RNN є проблема зникнення (або вибухового) градієнта, коли інформація швидко втрачається з часом. Звичайно, це впливає лише на ваги, а не на стан нейрона, але інформація накопичується в нейроні. Як правило, цей тип мережі використовується для автоматичного заповнення інформації [6]. Схему такої моделі можна побачити на рис. 1.4

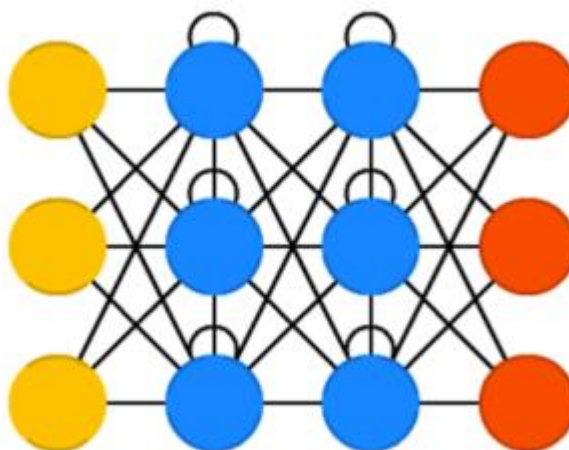


Рисунок 1.4 – Схема рекурентної нейронної мережі

Отже базовий RNN — це мережа нейроноподібних вузлів, організованих у послідовні шари. Кожен вузол на даному рівні має спрямовані (односторонні) з'єднання з усіма іншими вузлами на наступному рівні. Кожен вузол (нейрон) має фактичну активацію, яка змінюється з часом. Кожне з'єднання (синапс) має змінну реальну вагу. Вузли можуть бути вхідними вузлами (отримують дані ззовні мережі), вихідними вузлами (виробляють результати) або прихованими вузлами (змінюють дані на шляху від входу до виводу).

Для контрольованого навчання в контексті дискретного часу послідовність дійсних вхідних векторів подається на вхідний вузол по одному вектору за раз. На певному кроці часу кожен блок без введення обчислює свою поточну активацію (результат) як нелінійну функцію зваженої суми активацій усіх підключених до нього блоків. Спеціальні вихідні пристрої можуть бути оснащені цільовою активацією, визначеною супервайзером через певні проміжки часу. Наприклад, якщо вхідна послідовність є мовним сигналом, що відповідає вимовленим цифрам, кінцевим цільовим виходом у кінці послідовності може бути мітка, яка класифікує цифри.

У навчальному середовищі з підкріпленням жоден вчитель не дає цільових підказок. Замість цього, щоб оцінити продуктивність RNN, іноді використовується фітнес-функція або функція винагороди, яка впливає на її вхідний потік через вихідний блок актора, підключений до середовища. Це можна використовувати в іграх, де прогрес вимірюється набраними очками [7].

Кожній послідовності присвоюється помилка як сума обчислених мережею відхилень усіх цільових сигналів від відповідної активації. Для навчального набору з кількома послідовностями загальна помилка є сумою помилок усіх окремих послідовностей.

1.2.2 Нейромережі з довгою короткочасною пам'яттю (LSTM)

Мережі довгої короткочасної пам'яті (LSTM) намагаються вирішити вищевказану проблему втрати інформації за допомогою фільтрів і явних блоків зберігання. Кожен нейрон має блок зберігання і три фільтри: вхідний, вихідний і забувальний. Призначення цих фільтрів – захист інформації. Вхідні фільтри визначають, скільки інформації з попереднього рівня зберігається в комірці. Вихідні фільтри визначають, скільки інформації отримує наступний рівень. Що ж, фільтри забуття також виконують корисну функцію, якими б вони не здавалися дивними: наприклад, коли мережа вивчає книгу та переходить до нового розділу, вона може вирішити забути деякі символи зі старих розділів [8].

На рис. 1.5 зображено схему нейромережі з довгою короткочасною пам'яттю, де синіми комірками позначено нейрони пам'яті.

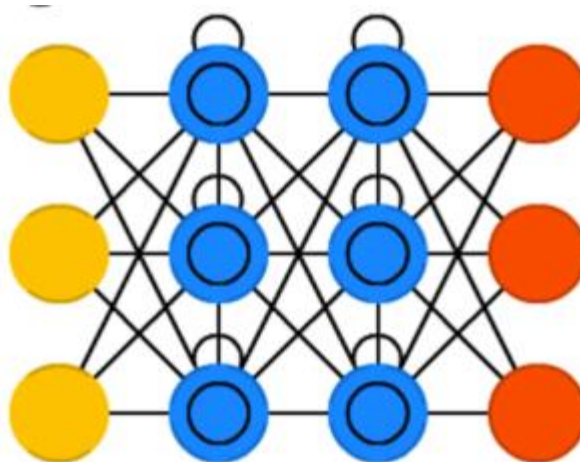


Рисунок 1.5 – Схема нейромережі з довгою короткочасною пам'яттю

1.2.3 Вентельний рекурентний вузол (GRU)

Gated Recurrent Unit (GRU) – це архітектура рекурентної нейронної мережі (RNN), представлена Kyunghyun Cho та його колегами в 2014 році. Основна мета GRU — вирішити проблему зникаючого градієнта, поширену проблему під час навчання традиційних RNN. Ця проблема виникає, коли нахил функції втрат стає занадто малим відносно параметрів моделі, що ускладнює для моделі вивчення довгострокової залежності від даних. Показано, що GRU та його механізм активації є ефективними у виявленні довгострокових залежностей і зменшенні проблеми зникнення градієнта.

Головним нововведенням GRU є його механізм керування, який складається з двох воріт: шлюзу оновлення та шлюзу скидання. Ці ворота контролюють потік інформації в мережі та дозволяють моделі вирішувати, яку інформацію зберігати або відкидати в кожен момент часу. Шлюз оновлення контролює, наскільки зберігається попередній прихований стан, тоді як вентиль скидання контролює, наскільки попередній прихований стан впливає на

поточний вхід. Цей механізм тестування дозволяє GRU адаптивно вивчати відповідний контекст для кожної вхідної послідовності, що робить їх особливо придатними для завдань, пов'язаних із безперервними даними, наприклад: обробка природної мови, розпізнавання мовлення та прогнозування часових рядів. Схему цієї моделі можна побачити на рис. 1.6.

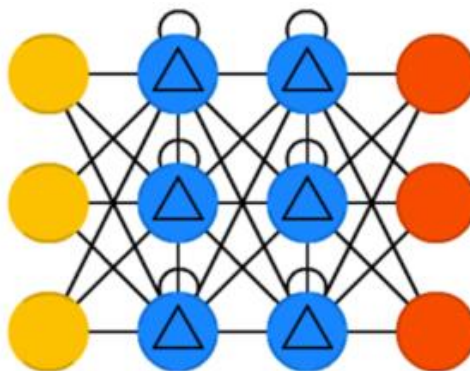


Рисунок 1.6 – Схема ШНМ з вентельними рекурентними вузлами

Важливим застосуванням GRU є прогнозування часових рядів, де вони використовуються для моделювання складних часових моделей у даних. GRU використовувався для прогнозування цін на акції, аналізу кліматичних даних, прогнозування споживання енергії тощо. Їх здатність фіксувати довгострокову залежність і обробляти шумові дані робить їх ідеальними для вирішення таких проблем, часто перевершуючи традиційні моделі часових рядів [9].

Хоча GRU мають вражаючу продуктивність у різноманітних програмах, вони не позбавлені обмежень. Однією з найбільших проблем у використанні GRU є обчислювальна складність, пов'язана з навчанням великих моделей, особливо проблеми, пов'язані з великовимірними даними або довгими послідовностями. Однак останні досягнення в апаратному забезпеченні та методах оптимізації допомагають пом'якшити цю проблему та дозволяють розробити більш ефективні та масштабовані моделі на основі GRU.

1.3 Постановка завдання до роботи

Проблема дітей-сиріт в Україні досить гостра, особливо зважаючи на бойові дії, що йдуть на території нашої держави.

Тому система прогнозування на основі нейромережових діагностичних моделей може допомогти оптимізувати бюджети та зменшити кількість браку фінансування у окремих випадках. Було проведено вибір даних, необхідних для такого прогнозування. Ці ознаки можна побачити у таблиці 1.2.

Таблиця 1.2 – Вхідні показники ШНМ

Параметр	Позначення	Одиниці виміру
К-ть абітурієнтів	X_1	Одиниць
Похибка вступу	X_2	%
Інфляція	X_3	%
Субвенція	X_4	грн
Зарплатня службовцям	X_5	грн
Частка ДС-вступників	X_6	%
Похибка ДС-випускн.	X_7	%
Стипендія ДС	X_8	грн
Вік студентів	X_9	роки
Напрямок навчання	X_{10}	

1.4 Висновки за розділом 1

У даному розділі було досліджено предметну область, досліджено переваги та недоліки різних моделей ШНМ. Визначено вхідні дані для навчання нейромережових моделей.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ ПРОЄКТОВАНОЇ СИСТЕМИ

2.1 Вибір інструментарію розробки

Перед початком розробки визначимось з середовищем розробки та мовою програмування.

Як середовище розробки для цього проєкту мною було обрано онлайн застосунок від Google під назвою Colaboratory. Він був обраний через низку переваг які ми розглянемо нижче.

Також в цьому розділі буде проведено порівняння мов програмування в розрізі їх використання для реалізації нейромережових моделей.

2.1.1 Вибір середовища розробки

Побудова та навчання нейромережових моделей досить складний і кропіткий процес який вимагає великої кількості засобів, як програмних так й апаратних. До того ж робота з даними теж вимагає специфічних інструментів. Доступ до всього цього надає компанія Google за допомогою свого онлайн лабораторії Colab.

Google Colab (або Google Collaboratory) – це безкоштовна хмарна платформа для написання, запуску та спільного використання коду Python. Він надає доступ до обчислювальних ресурсів Google, зокрема графічних процесорів (GPU) і тензорних процесорів (TPU), для полегшення робочих навантажень машинного навчання та аналітики даних.

Google Colab базується на проєкті Jupyter Notebook і має схожу структуру. Код у Colab організований у комірки, які можуть містити текстові описи, програмний код або результати його виконання. Це дозволяє створювати інтерактивні документи, які містять код, візуалізації та текст, щоб їх можна було легко використовувати для навчання, дослідження та співпраці.

Однією з переваг Google Colab є можливість легко імпортувати та експортувати файли за допомогою Google Drive, а також легко ділитися та редагувати файли з іншими користувачами в реальному часі.

Google Colab не потребує складного налаштування чи встановлення, оскільки він запускається безпосередньо у вашому браузері. Він також містить попередньо встановлені бібліотеки Python, які не потребують налаштування для використання [10].

Наразі цей сервіс підтримує лише мову Python, з цього ясно яку мову було обрано для розробки. Далі докладніше розглянемо причини такого вибору.

2.1.2 Вибір мови програмування

Вибір мови програмування є досить важливим при проектуванні будь-яких програмних застосунків. Сьогодні все більше і більше мов програмування адаптуються для розробки штучних нейронних мереж тому вибір дійсно дуже широкий. Для порівняння було обрано декілька загальноприйнятих в цій сфері мов, таких як Python, R та C++, а також додамо JavaScript як нову мову для даної галузі.

Почнемо з Python. Це досить популярна мова програмування, що здобула поширення через швидкий розвиток, потужність та можливість застосування для виконання широкого спектру завдань.

Також така популярність зумовлена досить простим синтаксисом мови, легкістю читання, широкими можливостями для візуалізації даних та сумісністю з багатьма іншими мовами програмування, що значно розширює сферу її застосування. Через те, що ця мова є досить популярною можна відзначити велику кількість користувачів цієї мови, а також докладну документації по ній та її бібліотекам.

Недоліками же мови Python є те, що це інтерпритована мова, а не компільована, що спричиняє деяке зменшення швидкодії. Ще одним

недоліком модна відмітити непристосованість цієї мови для багатопотокового виконання [11].

Якщо підсумовувати переваги та недоліки цієї мови, то можна відзначити наступні.

Переваги:

- зручність та простота;
- легкість вивчення;
- легка візуалізація даних;
- проста читаємість;
- сумісність з іншими мовами;
- велика к-ть спеціалізованих бібліотек;
- велика спільнота користувачів;
- якісна документація.

Недоліки:

- знижена швидкодія;
- залежність від бібліотек у деяких задачах.

Досить високою популярністю серед аналітиків даних та розробників нейронних мереж користується мова програмування R.

Це мова програмування та програмне середовище обчислень з відкритим вихідним кодом для статистичної обробки даних. Як і Python це інтерпритована мова тому має схожі проблеми зі швидкодією. Досить великим плюсом є його розширюваність завдяки величезній кількості пакетів, їх близько 15 тисяч.

До мінусів можна віднести, як зазначалось раніше, швидкодію та недостатньо розвинені бібліотеки для машинного навчання [12].

Мова програмування C++ досить стара, добре відома і дуже універсальна.

На відміну від попередньо зазначених вона є компільованою мовою, що досить добре позначається на її швидкодії. Завдяки продуктивності, гнучкості, ефективності та можливості керування пам'яттю ця мова ідеально підходить

для рішень на основі штучного інтелекту, де необхідна гарна оптимізація чи підвищена швидкість роботи.

Плюси:

- швидкодія: C++ є компільованою мовою, що дозволяє отримувати високу швидкодію виконання;
- контроль над пам'яттю: Можливість контролювати використання пам'яті може бути корисною для оптимізації продуктивності.

Мінуси:

- складність: C++ є більш складною мовою, що може затруднити розробку та відладку коду;
- менша кількість вбудованих бібліотек для нейромереж: Порівняно з Python, бібліотеки для нейромереж можуть бути менш розгалуженими в C++[13].

Останній на черзі JavaScript. Хоч це й дуже стара мова програмування, але за час свого існування вона кардинально еволюціонувала. З мови для якоїсь базової анімації на Веб-сайтах вона перетворилась на серйозний рушій для величезної кількості застосувань.

Для машинного навчання JavaScript пропонує цікавий підхід буквально навчання у браузері. Це дозволяє простіше створювати Веб-застосунки в яких використовується ШІ.

Але так як шлях цієї мови у світі нейронних мереж почався відносно нещодавно, кількість бібліотек для їх реалізації є досить обмеженою [14]-[15].

Насамкінець було складено порівняльну таблицю, у якій зібрані переваги та недоліки розглянутих мов програмування.

Таблиця 2.1 – Порівняння мов програмування

Характеристика	Python	R	C++	JavaScript
Швидкодія виконання	Середня	Середня	Висока	Середня
Спільнота та Екосистема	Велика	Велика	Значна	Велика
Легкість вивчення	Висока	Висока	Середня	Висока

Продовження таблиці 2.1

Характеристика	Python	R	C++	JavaScript
Можливості статистики	Обмежені	Високі	Середні	Обмежені
Бібліотеки для нейромереж	TensorFlow, PyTorch, Keras	TensorFlow, Keras	TensorFlow, Caffe	TensorFlow.js, Brain.js

2.2 Бібліотеки Python для машинного навчання

Для мови програмування Python існує чимало різноманітних бібліотек, зокрема для використання при машинному навчанні. Розглянемо декілька з них та проведемо їх порівняння.

Для розгляду були обрані такі популярні бібліотеки як TensorFlow, Keras та PyTorch.

PyTorch – це фреймворк глибокого навчання з відкритим кодом, створений як гнучкий і модульний для досліджень, зі стабільністю та підтримкою, необхідною для розгортання виробництва. PyTorch надає пакет Python для таких високорівневих функцій, як тензорні обчислення (наприклад, NumPy) із сильним прискоренням GPU та TorchScript для легкого переходу між режимом захоплення та режимом графа. З останнім випуском PyTorch фреймворк забезпечує виконання на основі графів, розподілене навчання, мобільне розгортання та квантування.

У той час як статичні графіки чудово підходять для розгортання виробництва, дослідницький процес, пов'язаний із розробкою наступного чудового алгоритму, справді динамічний. PyTorch використовує техніку під назвою автоматична диференціація у зворотному режимі, яка дозволяє розробникам довільно змінювати поведінку мережі з нульовою затримкою або накладними витратами, прискорюючи ітерації дослідження [16].

TensorFlow – це бібліотека з відкритим вихідним кодом, розроблена Google переважно для програм глибокого навчання. Він також підтримує традиційне машинне навчання. TensorFlow спочатку був розроблений для

великих чисельних обчислень без глибокого навчання. Однак він також виявився дуже корисним для розробки глибокого навчання, тому Google відкрила його з відкритим кодом.

TensorFlow приймає дані у формі багатовимірних масивів вищих розмірів, які називаються тензорами. Багатовимірні масиви дуже зручні в роботі з великими обсягами даних.

TensorFlow працює на основі графів потоку даних, які мають вузли та ребра. Оскільки механізм виконання має форму графіків, набагато простіше виконувати код TensorFlow розподіленим способом у кластері комп'ютерів, використовуючи графічні процесори.

Однією з найкращих якостей TensorFlow є те, що він полегшує розробку коду. Легко доступні API позбавляють користувачів від переписування частини коду, що в іншому випадку зайняло б багато часу. TensorFlow прискорює процес навчання моделі. Крім того, ймовірність помилок у програмі також зменшується, як правило, на 55–85 відсотків.

Іншим важливим аспектом є те, що TensorFlow дуже масштабований. Ви можете написати свій код, а потім змусити його працювати на ЦП, ГП або в кластері цих систем з метою навчання.

Як правило, навчання моделі – це те, на що йде значна частина обчислень. Крім того, процес навчання повторюється кілька разів для вирішення будь-яких проблем, які можуть виникнути. Цей процес призводить до споживання більшої потужності, а отже, потрібні розподілені обчислення. Якщо вам потрібно обробити великі обсяги даних, TensorFlow полегшує це, запускаючи код розподіленим способом.

GPU, або графічні процесори, стали дуже популярними. Nvidia є одним із лідерів у цій галузі. Він добре виконує математичні обчислення, наприклад множення матриць, і відіграє важливу роль у глибокому навчанні. TensorFlow також має інтеграцію з C++ і Python API, що робить розробку набагато швидшою [17].

Keras — це високорівневий API глибокого навчання, розроблений Google для впровадження нейронних мереж. Він написаний на Python і використовується для полегшення впровадження нейронних мереж. Він також підтримує численні серверні обчислення нейронної мережі.

Keras відносно легко освоїти та працювати з ним, тому що він забезпечує інтерфейс Python з високим рівнем абстракції, маючи можливість використовувати кілька серверних інтерфейсів для обчислень. Це робить Keras повільнішим, ніж інші фреймворки глибокого навчання, але надзвичайно зручним для початківців.

Keras дозволяє перемикатися між різними серверами. Keras підтримує такі фреймворки:

- Tensorflow;
- Теано;
- PlaidML;
- MXNet;
- CNTK (Microsoft Cognitive Toolkit).

З цих п'яти фреймворків TensorFlow прийняв Keras як офіційний API високого рівня. Keras вбудовано в TensorFlow і може використовуватися для швидкого глибокого навчання, оскільки він надає вбудовані модулі для всіх обчислень нейронної мережі. У той же час, обчислення з використанням тензорів, обчислювальних графіків, сеансів тощо можна створювати на замовлення за допомогою Tensorflow Core API, який дає вам повну гнучкість і контроль над вашою програмою та дозволяє реалізувати ваші ідеї за відносно короткий час [18].

Виходячи з вище сказаного була створена порівняльна таблиця 2.2, де наведено основні характеристики досліджуваних бібліотек.

Таблиця 2.2 – Порівняння бібліотек машинного навчання

Характеристика	TensorFlow	PyTorch	Keras
Простота використання	Середня	Висока	Висока
Спільнота	Велика	Зростає	Велика
Гнучкість та Контроль	Висока	Висока	Середня
Обчислювальні графи	Статичні (з TensorFlow 2.0 – динамічні)	Динамічні	Статичні (динамічні з TensorFlow 2.0)
Вбудовані функції та моделі	Велика кількість	Зростає	Залежить від бекенда
Призначення	Стандарт для досліджень та продакшну	Підходить для експериментів та налагодження	Зручний для швидких експериментів та простих задач

2.3 Висновки за розділом 2

В даному розділі були розглянуті основні мови програмування, що підходять для розробки нейромережових моделей, проведено їх порівняння та обрано мову для використання.

Також обрано середовище розробки та розглянуті основні бібліотеки для машинного навчання та обробки даних.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

Метою роботи є дослідження та синтез нейромережових діагностичних моделей для прогнозування фінансування сирот у вищих навчальних закладах.

Огляд основних підходящих під нашу задачу архітектур було проведено у розділі 1.2, а їх порівняння приведено у таблиці 1.1.

Модель GRU було відкинуто через її надлишковість для виконуваної задачі. А чиста модель RNN може демонструвати ефект згасаючого градієнту, через це її використання було визнано недоцільним..

Тому як основну модель для побудови нейромережі було прийнято рішення використати LSTM. Далі детально розберемо особливості її роботи.

Також розглянемо популярні функції активації та використані бібліотеки мови Python.

3.1 Загальний принцип роботи моделі LSTM

Прості рекурентні нейронні мережі хоч і додають до штучних нейронів пам'ять, але вона виходить короткочасною. Бо на кожному етапі навчання нова інформація доміщується до наявної і через декілька повторів повністю перезаписується.

LSTM модель створені як раз для того, щоб уникнути ефекту довготривалої залежності. Вона може запам'ятовувати значення як на коротких, так і на довгих часових проміжках.

Це було досягнуто завдяки тому, що LSTM не має всередині своїх рекурентних компонентах функції активації. Так при тренуванні моделі запобігається ефект згасання градієнту при використанні метода зворотнього поширення помилки.

Як і прості рекурентні мережі LSTM має будову схожу на лонцюжок, але на відміну від них замість одного рівня мережі існує чотири, які особливим

чином взаємодіють між собою. Схему модуля мережі LSTM показано на рис. 3.1.

Основні компоненти такого модуля є стан комірки та різноманітні фільтри.

Стан комірки власне і можна назвати пам'ятю мережі, яка передається крізь весь модуль. Через це інформація навіть з попередніх ітерацій може бути доступною в наступних, що прибирає ефект короточасної пам'яті. На рис. 3.2 його можна побачити у вигляді вектора у верху схеми.

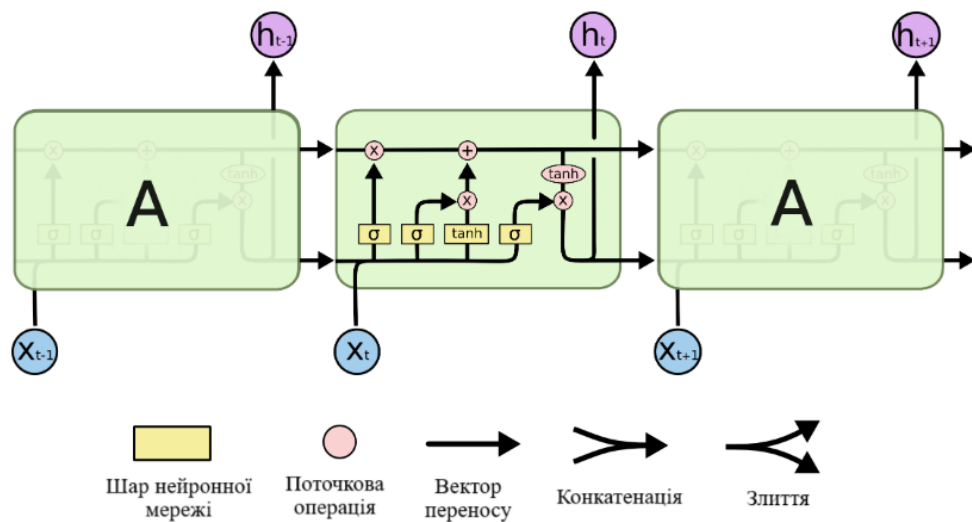


Рисунок 3.1 – Схема модуля LSTM

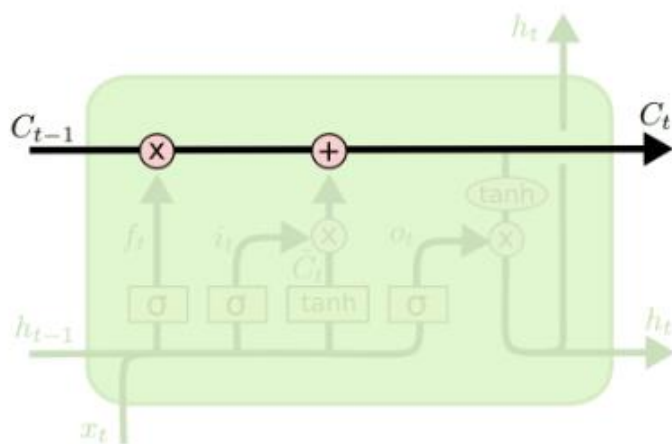


Рисунок 3.2 – Стан комірки LSTM

Під час навчання стан комірки змінюється структурами, що додають або видаляють інформацію. Їх називають фільтрами. Вони керуючись деякими умовами, контролюють потік інформації на вхід та на вихід модуля. Фільтри складаються з шару сигмоподібної нейронної мережі та операції поточкового множення.

Сигмоподібний шар виводить числа від нуля до одиниці, описуючи, скільки кожного компонента має пройти. Значення нуль означає «нічого не пропускати», тоді як значення один означає «пропускати все!»

LSTM має три таких фільтри для захисту та контролю стану комірки.

Далі покроково мережа робить нищезначені дії.

Спочатку за допомогою шару фільтра забування визначається та інформація, яку треба лишити, а яку забути. Це відбувається наступним чином значення минулого виходу h_{t-1} та цього входу x_t проходять через сигмоїдальний шар. Отримані значення знаходяться у проміжку від 0 до 1. Ті які наближені до 0 видаляються, ті, що ближче до 1 – зберігаються. Це проілюстроване на рис. 3.3 та формулою (3.1).

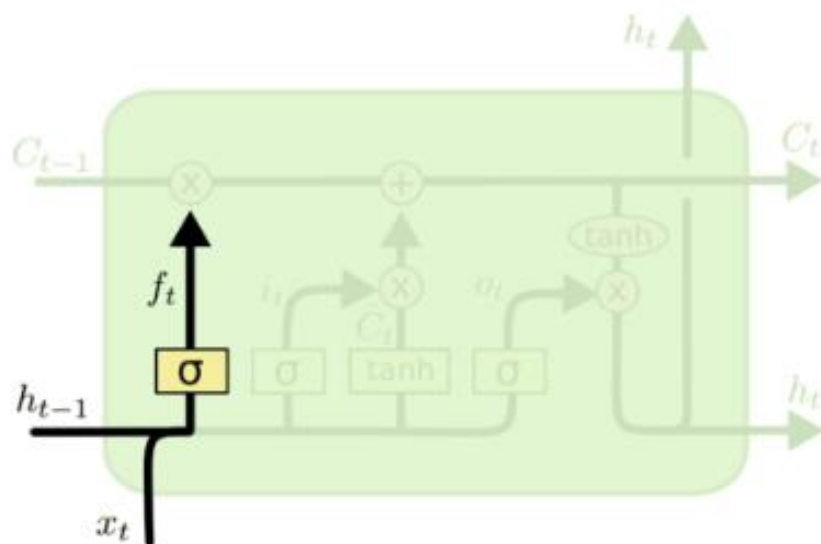


Рисунок 3.3 Робота фільтра забування

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \quad (3.1)$$

де h_{t-1} — значення минулого виходу;

x_t — значення поточного входу;

W_f, b_f — матриця і вектор параметрів;

σ — сигмоїдна функція активація.

Наступним кроком необхідно вирішити, що саме потрібно зберегти у стані чарунки. Це проходить у двох незалежних шарах, куди також потрапляють об'єднані значення минулого виходу h_{t-1} та поточного входу x_t .

У шарі, де функція активації $\tanh$ вектори x_t і h_{t-1} множаться на вагову матрицю W_c , отриману в процесі навчання. До результату додається зміщення, а векторна сума передається від одного елемента до іншого за допомогою функції активації гіперболічного тангенса $\tanh$. Ось так вибудовується вектор нових значень, які треба записати у «вектор пам'яті». Назвемо це «вектором нових значень» (формула 3.2).

Інший нейронний шар множить вхідні дані за допомогою вивченої вагової матриці W_i - вхідна ваги. Додається зсув b_i , і векторна сума передається елемент за елементом сигмоїдній функції. Ось як створюється «оціночний» вектор i_t : він множиться на «вектор нових значень». Якщо вектор оцінки містить нулі, кожен елемент нових значень, що множиться на них не потрапляє в контекст (формула 3.3).

Так LSTM виражає те, що важливо додати до контексту. Але нічого в нього ще нічого не записує. Весть процес зображено на рис. 3.4.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \quad (3.2)$$

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c), \quad (3.3)$$

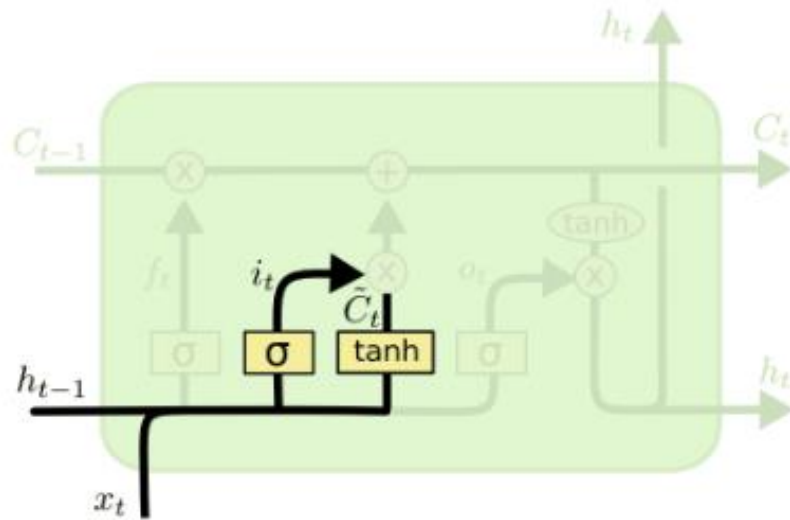


Рисунок 3.4 – Процес запам'ятовування у LSTM

Далі, як зображено на рис. 3.5, необхідно оновити стан комірки $\tilde{C}_{t-1}$, перезаписавши його новим вектором. Для цього старий необхідно помножити на оціночний вектор f_t для забування непотрібного. Наступним кроком додаємо добуток i_t та $\tilde{C}_t$. Це масштабовані значення кандидатів на оновлення стану (формула 3.4).

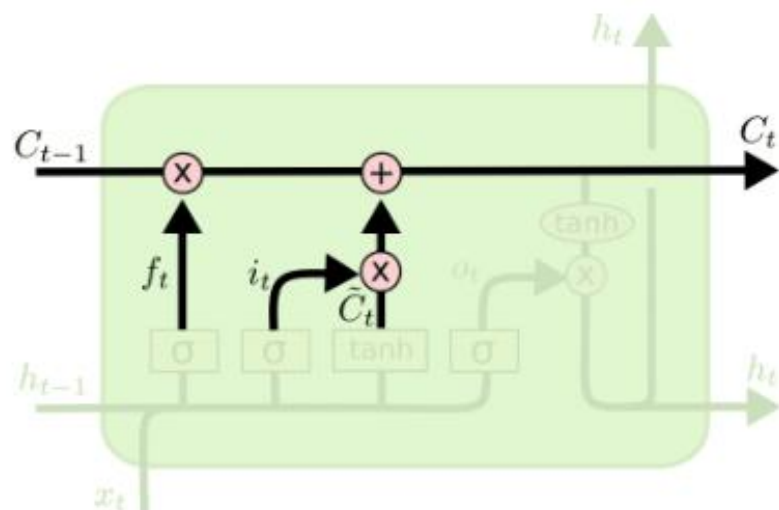


Рисунок 3.5 – Процес збереження у комірку LSTM

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t, \quad (3.4)$$

На останньому кроці визначається, яка інформація буде отримана на виході. Результат базуватиметься на стані комірки, і до неї будуть застосовані певні фільтри. Спочатку значення попереднього виведення h_{t-1} і поточний вхід x_t передається на сигмоподібний рівень, який вирішує, яка інформація про стан комірки буде виведена (формула 3.5). Значення стану комірки передаються на рівень $\tanh$ для отримання вихідних значень від -1 до 1 і множаться на вихідні значення сигмоподібного шару, дозволяючи виводити лише необхідну інформацію (формула 3.6)[19]. Цей процес зображено на рис. 3.6.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (3.5)$$

$$h_t = o_t * \tanh(C_t) \quad (3.6)$$

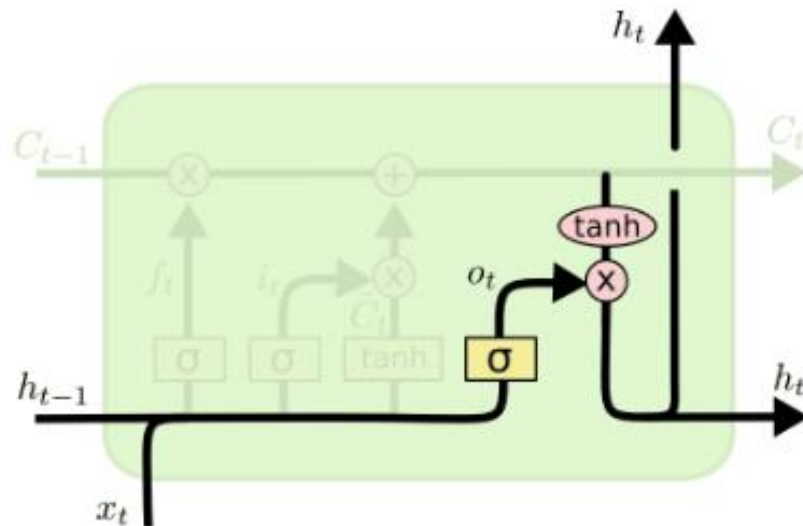


Рисунок 3.6 – Процес виведення даних

3.2 Функції активації

При машинному навчанні та штучних нейронних мережах дуже важливу роль у обробці даних відіграють шари активації.

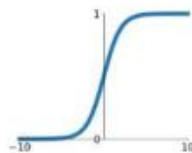
Шари активації це одні з основних шарів у нейронних мережах, вони є функцією, що дає нелінійність до виходу минулого шару. Це дозволяє нейромережам більш точно передбачати результати.

Таким чином шари активації приймають результати роботи минулого шару і перетворюють їх у вихідне значення, що передається наступному шару. Використовуючи функції активації вони визначають яким чином будуть перетворені дані.

Для задач машинного навчання шари активації є досить важливими і залежать від конкретного завдання, що вирішується. На рис. 3.7 зображено популярні графіки функцій активації з їх формулами.

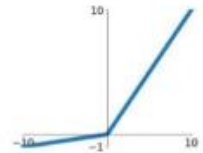
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



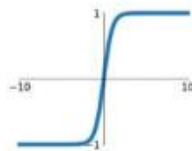
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

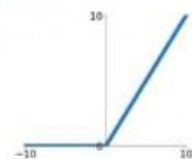


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

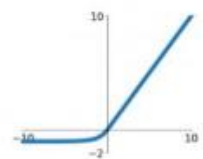


Рисунок 3.7 – Популярні функції активації

Загалом вибір шару активації залежить від архітектури мережі та даних на яких вона навчається. Тому шляхом експериментів необхідно вибрати ту, що підходить краще для кожної конкретної задачі[20].

3.3 Бібліотека NumPy

NumPy це модуль для мови Python з відкритим вихідним кодом, що забезпечує загальні математичні числові операції у вигляді вже підготовлених швидких функцій. Вони об'єднані у високорівневі пакети. Цей модуль забезпечує функціонал близький до можливостей MatLab. NumPy (Numeric Python) надає базові методи для маніпуляції з масивами та матрицями.

SciPy (Scientific Python) ще більше збільшує функціонал цієї бібліотеки, додаючи величезну кількість алгоритмів, таких як мінімізація, перетворення Фур'є, регресія та ін [21].

До переваг бібліотеки NumPy належать:

- швидкодія;
- зручність використання;
- інтеграція з іншими бібліотеками.

Все це робить дану бібліотеку дуже корисним інструментом для обробки та маніпуляції даними.

3.4 Побудова графіків за допомогою Matplotlib

Matplotlib це бібліотека Python для візуалізації даних. Вона дає можливість будувати двовимірні та тривимірні графіки.

Python Matplotlib є схожим на засіб візуалізації у MatLab. Matplotlib інтерфейс об'єктно-орієнтований, тому користувач має змогу взаємодіяти з кожним об'єктом безпосередньо. За допомогою коду можна довільно змінювати будь-який елемент графіка.

Matplotlib має високорівневий інтерфейс pyplot. У ньому все досить автаматизовано, тому опанувати його досить просто. Приклад графіка побудованого за допомогою цієї бібліотеки зображено на рис. 3.8 [22].

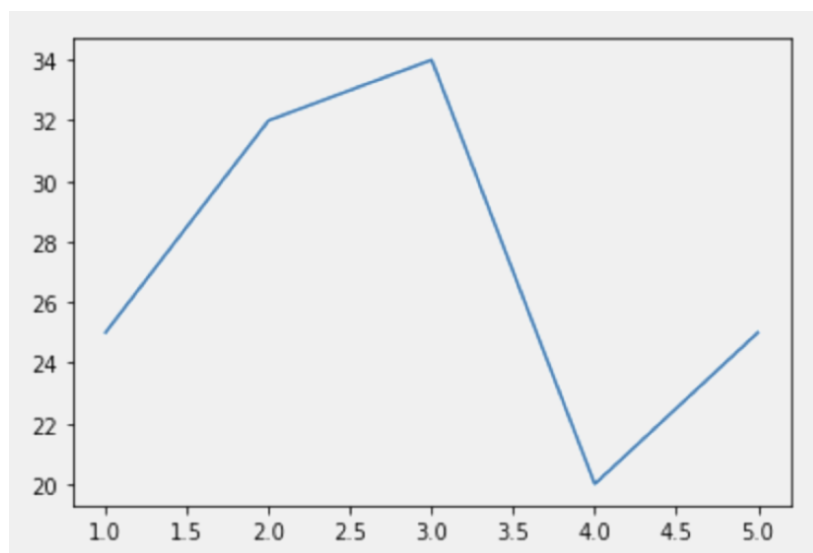


Рисунок 3.8 – Графік Matplotlib

3.5 Збір та представлення даних

Дані для навчання штучної нейронної мережі були зібрані з відкритих джерел.

Дані були перетворені у формат csv через зручність читання даних як людиною, так і машиною, а також через легкість трансформації даних у об'єкти для введення до програми.

Comma-separated values (CSV) – це формат текстового файлу, який використовує значення, розділені комами. Файли CSV зберігають табличні дані (числа та текст) як звичайний текст, де кожен рядок у файлі зазвичай представляє запис даних. Кожен запис у файлі CSV має однакову кількість полів, розділених комами. Роздільники полів можуть з'являтися на самому полі, а поля можна брати в лапки.

Формат файлу CSV є форматом файлу з роздільниками. Загальні роздільники включають коми, табуляції, пробіли та крапки з комою. Файли з роздільниками зазвичай мають розширення ".csv", навіть якщо роздільник полів не є комою. Більшість програм або бібліотек, які використовують або

створюють файли CSV, мають можливість установлювати різні порогові значення.

CSV не відповідає стандарту RFC 4180, який вимагає, щоб програми введення даних підтримували декілька форматів CSV. Незважаючи на цей недолік, CSV широко використовується в програмах обробки даних і широко підтримується різними програмами, включаючи звичайні програми для роботи з електронними таблицями, такі як Microsoft Excel. Переваги CSV включають читабельність і простоту формату [23].

3.6 Висновок за розділом 3

В даному розділі були розглянуті принципи роботи мережі з архітектурою LSTM, описані основні бібліотеки мови Python, що були використані для реалізації цієї моделі та обрано формат csv як формат файлів для вхідних даних.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови використання

Проектована діагностична модель ШНМ призначена для прогнозування фінансування дітей-сирот у закладах вищої освіти.

Головними вимогами для навчання та використання моделі на локальному пристрої є:

- операційна система Microsoft Windows 7 і вище, або Linux з ядром системи версії Kernel 12 та вище;
- центральний процесор з тактовою частотою 3.0 ГГц чи вище та кількістю ядер не менше двох;
- об'єм оперативної пам'яті вище за 6 Гб;
- об'єм постійної пам'яті вище за 8 Гб;
- встановлене середовище виконання Python та підключені бібліотеки NumPy, TensorFlow та Keras.

При збільшенні наявних ресурсів системи пропорційно збільшується й швидкість навчання моделі.

Середовище Google Colab надає достатню кількість ресурсів для даного проекту та дозволяє працювати з ним онлайн і не вимагає встановлення громіздких залежностей.

4.2 Характеристики програмного продукту

Програмний продукт розроблений мовою Python, використанням фреймворку TensorFlow з інтерфейсом високого рівня Keras і являє собою штучну нейронну мережу за моделлю LSTM.

Для збереження даних було обрано файл формату CSV. Таке представлення даних дуже зручно перетворюється на об'єкти для можливості маніпулювання ними.

4.3 Звернення до програмного застосунку

Для використання програмної реалізації діагностичної моделі штучної нейронної мережі на локальному пристрої необхідно завантажити з репозиторію системи контролю версій Git, мати на комп'ютері, на якому буде виконуватись робота, встановлене середовище виконання мови Python і встановити всі необхідні залежності.

Використовуючи побудоавну модель онлайн в середовищі Google Colab, необхідно попередньо отримати доступ до так званого нотатку за відповідним посиланням.

Також це середовище дозволяє гнучко налаштовувати процес навчання, змінюючи параметри в процесі через значні ресурси виділені для роботи, що еквівалентні дуже потужним, а часом спеціалізованим комп'ютерам.

4.4 Вхідні та вихідні дані

Як вхідні дані для цього програмного рішення є набір статистичних даних за певний період часу. Всі вхідні статистичні дані були отримані з відкритих довідників. Вони представлені у формі табличних даних у форматі CSV. Цей формат передбачає текстове представлення даних, де роздільником полів виступає знак крапки з комою («;»), а рядків перехід на новий рядок.

Цей файл містить наступні поля:

- к-ть абітурієнтів;
- похибка вступу;
- інфляція;
- субвенція;
- зарплатня службовцям;
- частка дс-вступників;
- похибка дс-випускн.;
- стипендія дс;

- вік студентів;
- напрям навчання.

Їх можна побачити на рис. 4.1

applicants	entrance_error	inflation	subvention	salary_staff
2978.75	6.74	6.53	3121.36	13641.20
2692.03	3.63	4.34	2302.82	10972.86
2837.51	5.41	5.70	2913.02	11189.56
1816.18	1.55	3.41	3198.25	13407.65
2141.10	5.47	6.63	2946.22	12346.58
...	...	...	...	...
2176.53	2.90	4.84	1371.84	10911.69
1651.61	2.51	5.46	2451.09	8878.53
1004.16	1.42	6.65	2624.88	7070.88
2547.91	4.49	3.27	4422.19	8446.71
1524.90	5.28	4.97	2127.54	9251.12

orphans_ratio	graduate_error	orphans_stipend	students_age
2.22	2.22	2469.96	18.46
3.09	4.86	3939.09	18.37
3.82	2.59	2903.38	18.94
3.73	2.33	3772.42	18.52
3.23	4.43	3836.89	18.08
...	...	...	...
1.59	3.36	3403.04	19.35
4.75	5.95	3010.26	18.85
1.28	4.75	2234.36	18.45
4.16	3.74	2704.33	19.34
3.82	5.08	3269.06	18.49

Рисунок 4.1 – Вхідні параметри моделі

У якості основних вихідних даних є передбачене значення фінансування.

Також цікавим прикладом вихідних даних є відомості про використанні на виконання ресурси системи, що надає середовище . Це можна побачити на рисунку 4.2.

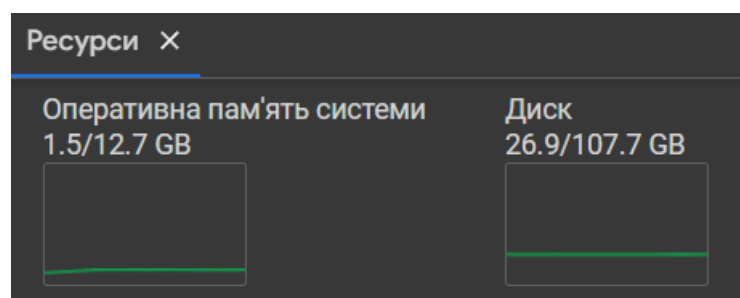


Рисунок 4.2 – Використання ресурсів системи

Втрати при навчанні (синім) та перевірці (червоним) моделі проілюстровані на рис. 4.3. Як видно перенавчання не відбулося і збіжність досить висока.

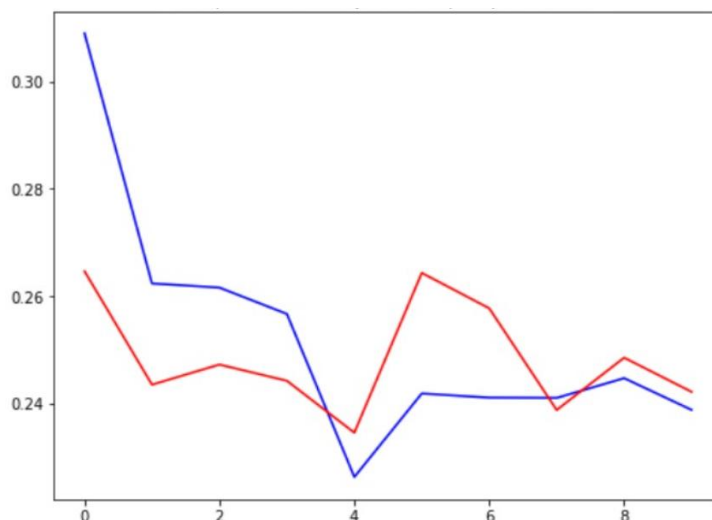


Рисунок 4.3 – Втрати моделі

Як приклад роботи моделі можна навести рис. 4.4, на якому синім зображено реальні, усереднені дані фінансування дітей-сиріт з урахуванням всіх необхідних видатків, а прогнозовані зображені жовтим. Як можемо побачити результати наближені одні до одних.

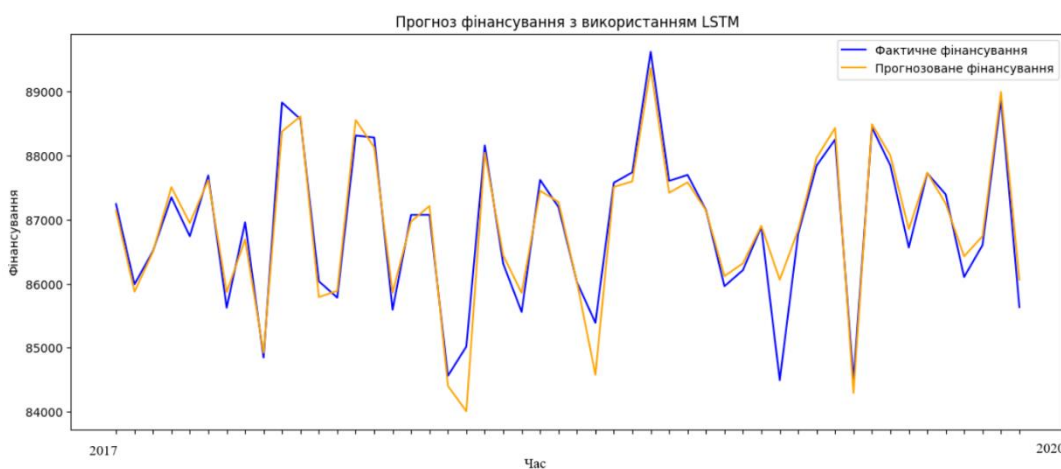


Рисунок 4.4 – Реальні й прогнозовані дані

4.5 Повідомлення

Google Colab як середовище розробки надає зокрема можливості відлагодження виконувануваного коду. У разі виникнення помилки при виконанні роботи середовище надсилає повідомлення про помилку, в якому точно вказується тип та місце появи помилки. Приклад такого повідомлення зображено на рис. 4.5.

```
ValueError                                Traceback (most recent call last)
<ipython-input-16-fbdc260df13d> in <cell line: 91>()
    89 plt.figure(figsize=(15, 6))
    90 plt.plot(y_test.index, y_test.values, label='Фактичне фінансування', color='blue')
--> 91 plt.plot(y_test.index, {prediction[0][0]}, label='Прогнозоване фінансування', color='orange')
    92 plt.title('Прогноз фінансування з використанням LSTM')
    93 plt.xlabel('Timestamp')
```

Рисунок 4.5 – Повідомлення про помилку

4.6 Висновки за розділом 4

У даному розділі було сформовано призначення програмного продукту та мінімальні системні вимоги для його роботи. Було описано порядок дій для отримання можливості доступу до застосунку, його будова та рішення розробки. Були описані вхідні та вихідні дані. Наведено приклади повідомлень, що використовує середовище для взаємодії з програмістом, зокрема повідомлення про помилки

5 КЕРІВНИЦТВО КОРИСТУВАЧА

5.1 Призначення та умови виконання

Спроектований програмний продукт являє собою онлайн-сервісний застосунок призначений для інформування та полегшення взаємодії співвласників багатоквартирного будинку.

Для використання програмний застосунок вимагає наявності інтернет-браузера та стабільного з'єднання з мережею Інтернет.

5.2 Використання застосунку

Це програмне забезпечення написане мовою програмування Python. Розробка та тестування виконувалися онлайн у застосунку Google Colab. Для локального використання на комп'ютері він має бути із наступними характеристиками:

- операційна система Microsoft Windows 7 і вище, або Linux з ядром системи версії Kernel 12 та вище;
- центральний процесор з тактовою частотою 3.0 ГГц чи вище та кількістю ядер не менше двох;
- об'єм оперативної пам'яті вище за 6 Гб;
- об'єм постійної пам'яті вище за 8 Гб;

Отриманий програмний продукт можна успішно використовувати на операційних системах Windows 7/10/11. З урахуванням використання різних компонентів у розробленому програмному пакеті, він повністю сумісний з операційною системою та відповідає всім вимогам програм. Проте, варто взяти до уваги, що в разі відсутності певних компонентів, які необхідні для операційної системи Windows, може виникнути неправильна робота продукту або помилки на початкових етапах його використання. Основні залежності включають:

- мова програмування Python;
- бібліотека неймереж TensorFlow;
- бібліотека машинного навчання scikit-learn.

5.3 Повідомлення оператору

При роботі застосунка користувач має можливість отримати повідомлення, які інформують про помилковість деяких дій чи виникнені виключних подій.

Приклад можливих повідомлень наведено у розділі «Керівництво програміста» на рис. 4.2.

5.4 Висновки за розділом 5

У даному розділі були розглянуті вимоги до апаратного забезпечення для використання застосунку та представлено загальні концепції роботи з розроблюваним застосунком.

ВИСНОВКИ

У рамках даної магістерської дипломної роботи було проведено комплексне дослідження та розроблено програмну реалізацію синтезу нейромережових діагностичних моделей для прогнозування фінансування сиріт у закладах вищої освіти

Під час дослідження виявлено, що застосування архітектур нейромереж, таких як LSTM та використання різних бібліотек, таких як TensorFlow та Keras, дозволяє отримати ефективні фінансування.

Розроблена програмна реалізація дозволяє використовувати отримані моделі на практиці, що робить їх придатними для впровадження в реальне середовище управління фінансами закладів вищої освіти. Основні переваги використання нейромережових моделей полягають у їхній здатності адаптуватися до складних та змінюючихся вхідних даних, що робить їх ефективним інструментом для прогнозування фінансових витрат та ресурсів для потреб сиріт в системі вищої освіти.

Отже, в рамках цієї роботи було показано, що застосування нейромережових діагностичних моделей є обіцяною стратегією для вирішення завдань прогнозування фінансування сиріт у закладах вищої освіти. Результати дослідження створюють основу для подальших досліджень та розвитку цієї області, сприяючи вдосконаленню управління фінансовими ресурсами та підтримці освітніх програм для цільової аудиторії..

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Про забезпечення організаційно-правових умов соціального захисту дітей-сиріт та дітей, позбавлених батьківського піклування [Електронний ресурс] : Закон України від 13.01.2005 р. № 2342-IV : станом на 3 серп. 2023 р. – Режим доступу: <https://zakon.rada.gov.ua/laws/show/2342-15#Text> (дата звернення: 14.10.2023).
2. Про охорону дитинства [Електронний ресурс] : Закон України від 26.04.2001 р. № 2402-III : станом на 5 жовт. 2023 р. – Режим доступу: <https://zakon.rada.gov.ua/laws/show/2402-14#Text> (дата звернення: 15.10.2023)
3. Студенти із числа дітей-сирот: визначаємо пільги та гарантії [Електрон. ресурс]. – Режим доступу: <https://buhgalter.com.ua/articles/soctsialnyy-zahist/studenti-iz-chisla-ditey-sirit-viznachayemo-pilgi-ta-garantiyi/>
4. Lawrence, J. Introduction to Neural Networks: Design, Theory, and Applications. / J. Lawrence. – Los Angeles: California Scientific Software, 2011. – 348 p.
5. Gurney, K. An Introduction to Neural Networks / K. Gurney. – Boca Raton: CRC Press, 1997. – 246 p.
6. Salem, Fathi M. Recurrent Neural Networks: From Simple to Gated Architectures / F.M. Salem. – Berlin: Springer, 2022. – P. 42-57
7. Mandic, D. Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability. / D. Mandic, J. Chambers. – New York: Wiley, 2021. – 304 p.
8. Hochreiter, S. Long Short-Term Memory. / S. Hochreiter, Jürgen Schmidhuber // Neural Computation. – 1997. – Vol. 9, №8. – P. 1735-1780.
9. Chung, J. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. / J. Chung, C. Gulcehre, KyungHyun Cho, Y. Bengio // Deep Learning and Representation Learning Workshop. – Vol. 1, №2. – 2014. – P. 1-9.
10. Colaboratory [Electronic resource]. – Access mode: <https://research.google.com/colaboratory/intl/uk/faq.html>

11. Welcome to Python.org [Electronic resource]. – Access mode: <https://www.python.org>
12. What is R [Electronic resource]. – Access mode: <https://www.r-project.org/about.html>
13. Страуструп, Б. Мова програмування C ++. Короткий курс. / Б. Страуструп. – К.: Діалектика, 2021. – 320 с.
14. JavaScript | MDN [Electronic resource]. – Access mode: <https://developer.mozilla.org/ru/docs/Web/JavaScript>
15. Машинне навчання в JavaScript [Електрон. ресурс]. – Режим доступу: https://w3schoolsua.github.io/ai/ai_javascript.html#gsc.tab=0
16. PyTorch [Electronic resource]. – Access mode: <https://ai.meta.com/tools/pytorch/>
17. Introduction to TensorFlow [Electronic resource]. – Access mode: <https://developers.google.com/machine-learning/crash-course/first-steps-with-tensorflow/toolkit>
18. Keras 3 is a multi-framework deep learning API [Electronic resource]. – Access mode: <https://keras.io/about/>
19. Christopher, Olah. Understanding LSTM Networks [Electronic resource]. – Access mode: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
20. Understanding Activation Functions in Neural Networks [Electronic resource]. – Access mode: <https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0>
21. NumPy documentation [Electronic resource]. – Access mode: <https://numpy.org/doc/stable/>
22. Matplotlib documentation [Electronic resource]. – Access mode: <https://matplotlib.org/stable/index.html>
23. Comma Separated Values [Electronic resource]. – Access mode: <https://docs.fileformat.com/spreadsheet/csv/>

ДОДАТОК А
Технічне завдання

Вступ

Ця робота виконана для вдосконалення системи фінансування вищої освіти для дітей-сиріт, та в контексті розвитку соціально відповідальної освітньої системи, яка забезпечує рівні можливості для кожного студента, незалежно від його соціального походження.

В результаті роботи було розроблено нейромережеву діагностичну модель яка здатна робити прогноз на основі статистичних даних.

A.1 Мета розробки і її призначення

Об'єкт дослідження – фінансування сиріт у закладах вищої освіти.

Предмет дослідження – методи та алгоритми синтезу нейромережевих діагностичних моделей для прогнозування фінансування сиріт у закладах вищої освіти.

Мета роботи – розробка нейромережевих діагностичних моделей для прогнозування фінансування сиріт у закладах вищої освіти.

Матеріали, методи та технічні засоби: інтернет-браузер Google Chrome, сервіс Google Scholar, сервіс Google Colab, Adobe Reader, IDE Visual Studio Code, Python, TensorFlow.

Актуальність розробки обумовлена використанням передових технологій управління та аналізу даних, зокрема нейромережевих моделей, що дозволяє покращити точність прогнозування та оптимізувати процеси управління фінансами. А також загостренням проблеми із збільшенням кількості сиріт у наслідок бойових дій та наданням дітям-сиротам та дітям, позбавленим батьківського піклування належних умов для здобуття вищої освіти.

Вищеназвані проблеми вирішені за допомогою створення інструментів для прогнозування фінансування сиріт у закладах вищої освіти на основі нейромережевих діагностичних моделей.

A.2 Стислий зміст основних теоретичних посилок до розробки на основі проведених пошуків

Діти-сироти часто стикаються із складнішими умовами життя та обмеженим доступом до різних ресурсів, включаючи фінансові. Це може призвести до обмеженого доступу до вищої освіти, внаслідок високих витрат на навчання.

Також багато дітей-сиріт не мають стабільних джерел доходу та фінансової підтримки, що робить навчання у вищому навчальному закладі великим фінансовим викликом. Вони можуть стикатися із проблемами в оплаті навчання, проживання та інших пов'язаних витрат.

І саме деякі з таких проблем можна вирішити використанням передових технологій управління та аналізу даних, зокрема нейромережевих моделей, що дозволяє покращити точність прогнозування та оптимізувати процеси управління фінансами задля забезпечення належних умов для здобуття вищої освіти дітьми, що позбавлені батьківського піклування.

A.3 Підстави для розробки

Проектування програмного продукту виконується на підставі завдання на дипломний проєкт на тему «Дослідження та програмна реалізація синтезу нейромережевих діагностичних моделей для прогнозування фінансування сиріт у закладах вищої освіти», затверджене наказом по НУ «Запорізька політехніка».

A.4 Основні вимоги до програми, що розробляється, початковим даним і результату

A.4.1 Вимоги до надійності

Застосунок має стабільно працювати у нормальних умовах та надійному з'єднанні з мережею Інтернет.

A.4.2 Умови експлуатації

Програмний застосунок не вимагає персоналу для обслуговування. Інтерфейс програмного продукту інтуїтивно зрозумілий та не вимагає спеціального навчання.

A.4.3 Вимоги до складу та параметрів технічних засобів

Розробка та тестування виконувалися онлайн у застосунку Google Colab. Для локального використання на комп'ютері він має бути із наступними характеристиками:

- операційна система Microsoft Windows 7 і вище, або Linux з ядром системи версії Kernel 12 та вище;
- центральний процесор з тактовою частотою 3.0 ГГц чи вище та кількістю ядер не менше двох;
- об'єм оперативної пам'яті вище за 6 Гб;
- об'єм постійної пам'яті вище за 8 Гб;

Отриманий програмний продукт можна успішно використовувати на операційних системах Windows 7/10/11. З урахуванням використання різних компонентів у розробленому програмному пакеті, він повністю сумісний з операційною системою та відповідає всім вимогам програм. Проте, варто взяти до уваги, що в разі відсутності певних компонентів, які необхідні для операційної системи Windows, може виникнути неправильна робота продукту або помилки на початкових етапах його використання. Основні залежності включають:

- мова програмування Python;
- бібліотека нейромереж TensorFlow;
- бібліотека машинного навчання scikit-learn.

A.4.4 Вимоги для транспортування та зберігання

Розроблений програмний продукт не вимагає особливих умов для транспортування та зберігання.

A.5 Вимоги до програмної документації

Програмна документація повинна містити в собі:

- технічне завдання;
- керівництво програміста;
- керівництво оператора;
- опис програми.

A.6 Існуючі обмеження для інсталяцію чи доступу до програмного застосунку

Спроектований онлайн-сервісний застосунок не вимагає встановлення. Безперебійне використання гарантується за умови відповідності системи вимогам до технічних засобів.

ДОДАТОК Б
Опис програмного продукту

Б.1 Загальні відомості

В результаті роботи було розроблено нейромережеву діагностичну модель яка здатна робити прогноз на основі статистичних даних.

Розробка та тестування виконувалися онлайн у застосунку Google Colab. Для локального використання на комп'ютері він має бути із наступними характеристиками:

- операційна система Microsoft Windows 7 і вище, або Linux з ядром системи версії Kernel 12 та вище;
- центральний процесор з тактовою частотою 3.0 ГГц чи вище та кількістю ядер не менше двох;
- об'єм оперативної пам'яті вище за 6 Гб;
- об'єм постійної пам'яті вище за 8 Гб;

Отриманий програмний продукт можна успішно використовувати на операційних системах Windows 7/10/11. З урахуванням використання різних компонентів у розробленому програмному пакеті, він повністю сумісний з операційною системою та відповідає всім вимогам програм. Проте, варто взяти до уваги, що в разі відсутності певних компонентів, які необхідні для операційної системи Windows, може виникнути неправильна робота продукту або помилки на початкових етапах його використання. Основні залежності включають:

- мова програмування Python;
- бібліотека нейромереж TensorFlow;
- бібліотека машинного навчання scikit-learn.

Б.2 Функціональне призначення

Дана нейромережева діагностична модель призначено для прогнозування фінансування дітей-сиріт у вищих навчальних закладах на основі статистичних даних.

Б.3 Опис структури

Проектований застосунок являє собою модель штучної нейронної мережі LSTM схематичне зображення роботи якої можна побачити на рис. Б.1.

Мережі довгої короткочасної пам'яті (LSTM) намагаються вирішити проблему втрати інформації за допомогою фільтрів і явних блоків зберігання.

Кожен нейрон має блок зберігання і три фільтри: вхідний, вихідний і забувальний. Призначення цих фільтрів – захист інформації. Вхідні фільтри визначають, скільки інформації з попереднього рівня зберігається в комірці. Вихідні фільтри визначають, скільки інформації отримує наступний рівень. Що ж, фільтри забуття також виконують корисну функцію, якими б вони не здавалися дивними: наприклад, коли мережа вивчає книгу та переходить до нового розділу, вона може вирішити забути деякі символи зі старих розділів[8].

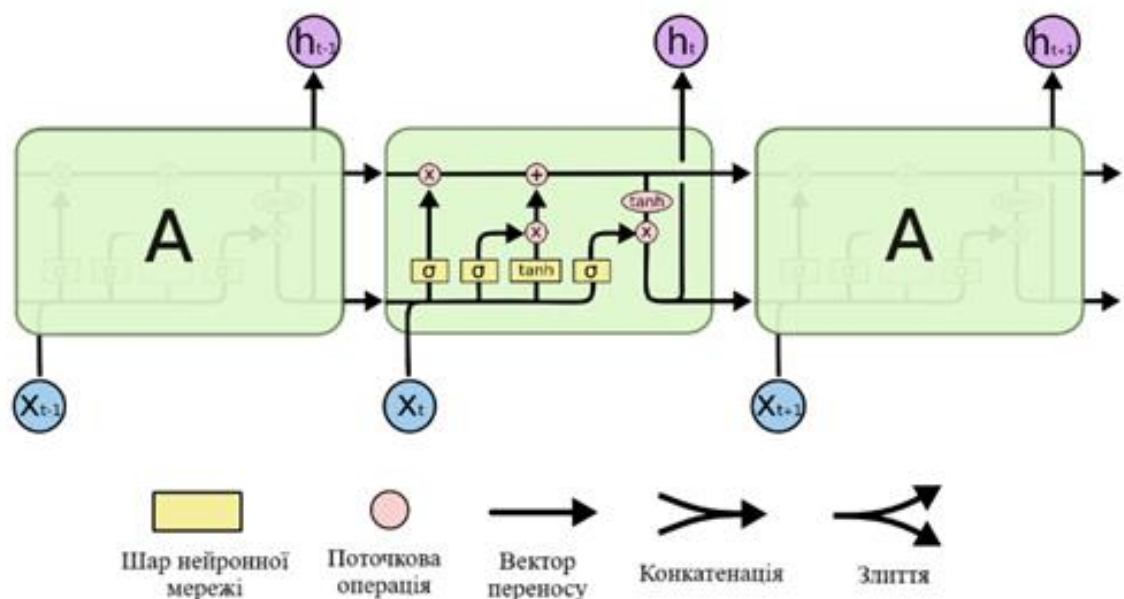


Рисунок Б.1 – Схема роботи LSTM

Б.4 Вибір інструментів та засобів розробки

Для реалізації проєкту було обрано мову програмування Python, як більш підходящу для виконуваних задач. Її перевагами є підтримуваність, швидкість роботи, велика кількість додаткових бібліотек та легкість вивчення.

Також було застосовано бібліотеки TensorFlow та Keras.

TensorFlow — це бібліотека з відкритим вихідним кодом, розроблена Google переважно для програм глибокого навчання. Він також підтримує традиційне машинне навчання. TensorFlow спочатку був розроблений для великих чисельних обчислень без глибокого навчання. Однак він також виявився дуже корисним для розробки глибокого навчання, тому Google відкрила його з відкритим кодом.

Keras — це високорівневий API глибокого навчання, розроблений Google для впровадження нейронних мереж. Він написаний на Python і використовується для полегшення впровадження нейронних мереж. Він також підтримує численні серверні обчислення нейронної мережі.

Keras відносно легко освоїти та працювати з ним, тому що він забезпечує інтерфейс Python з високим рівнем абстракції, маючи можливість використовувати кілька серверних інтерфейсів для обчислень. Це робить Keras повільнішим, ніж інші фреймворки глибокого навчання, але надзвичайно зручним для початківців.

Дані для використання при навчанні нейромережі були зібрані з відкритих джерел та були перетворені у формат csv через зручність читання даних як людиною, так і машиною, а також через легкість трансформації даних у об'єкти для введення до програми.

Comma-separated values (CSV) — це формат текстового файлу, який використовує значення, розділені комами. Файли CSV зберігають табличні дані (числа та текст) як звичайний текст, де кожен рядок у файлі зазвичай представляє запис даних. Кожен запис у файлі CSV має однакову кількість

полів, розділених комами. Роздільники полів можуть з'являтися на самому полі, а поля можна брати в лапки.

Б.5 Звернення до програмного продукту

Для використання програмної реалізації діагностичної моделі штучної нейронної мережі на локальному пристрої необхідно завантажити з репозиторію системи контролю версій Git, мати на комп'ютері, на якому буде виконуватись робота, встановлене середовище виконання мови Python і встановити всі необхідні залежності.

Використовуючи побудоавну модель онлайн в середовищі Google Colab, необхідно попередньо отримати доступ до так званого нотатку за відповідним посиланням.

Також це середовище дозволяє гнучко налаштовувати процес навчання, змінюючи параметри в процесі через значні ресурси виділені для роботи, що еквівалентні дуже потужним, а часом спеціалізованим комп'ютерам.

Б.6 Вхідні та вихідні дані

Як вхідні дані для цього програмного рішення є набір статистичних даних за певний період часу. Всі вхідні статистичні дані були отримані з відкритих довідників. Вони представлені у формі табличних даних у форматі CSV. Цей формат передбачає текстове представлення даних, де роздільником полів виступає знак крапки з комою («;»), а рядків перехід на новий рядок.

Цей файл містить наступні поля:

- к-ть абітурієнтів;
- похибка вступу;
- інфляція;
- субвенція;
- зарплатня службовцям;

- частка дс-вступників;
- похибка дс-випускн.;
- стипендія дс;
- вік студентів;
- напрям навчання.

Їх можна побачити на рис. Б.2

applicants	entrance_error	inflation	subvention	salary staff
2978.75	6.74	6.53	3121.36	13641.20
2692.03	3.63	4.34	2302.82	10972.86
2837.51	5.41	5.70	2913.02	11189.56
1816.18	1.55	3.41	3198.25	13407.65
2141.10	5.47	6.63	2946.22	12346.58
...	...	...	...	...
2176.53	2.90	4.84	1371.84	10911.69
1651.61	2.51	5.46	2451.09	8878.53
1004.16	1.42	6.65	2624.88	7070.88
2547.91	4.49	3.27	4422.19	8446.71
1524.90	5.28	4.97	2127.54	9251.12
orphans_ratio	graduate_error	orphans_stipend	students age	
2.22	2.22	2469.96	18.46	
3.09	4.86	3939.09	18.37	
3.82	2.59	2903.38	18.94	
3.73	2.33	3772.42	18.52	
3.23	4.43	3836.89	18.08	
...	...	...	...	
1.59	3.36	3403.04	19.35	
4.75	5.95	3010.26	18.85	
1.28	4.75	2234.36	18.45	
4.16	3.74	2704.33	19.34	
3.82	5.08	3269.06	18.49	

Рисунок Б.2 – Вхідні параметри моделі

У якості основних вихідних даних є передбачене значення фінансування.

Також цікавим прикладом вихідних даних є відомості про використані на виконання ресурси системи, що надає середовище . Це можна побачити на рис. Б.3.

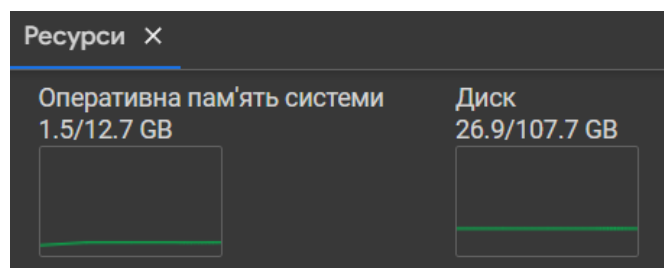


Рисунок Б.3 – Використання ресурсів системи

Втрати при навчанні (синім) та перевірці (червоним) моделі проілюстровані на рис. Б.4. Як видно перенавчання не відбулося і збіжність досить висока.

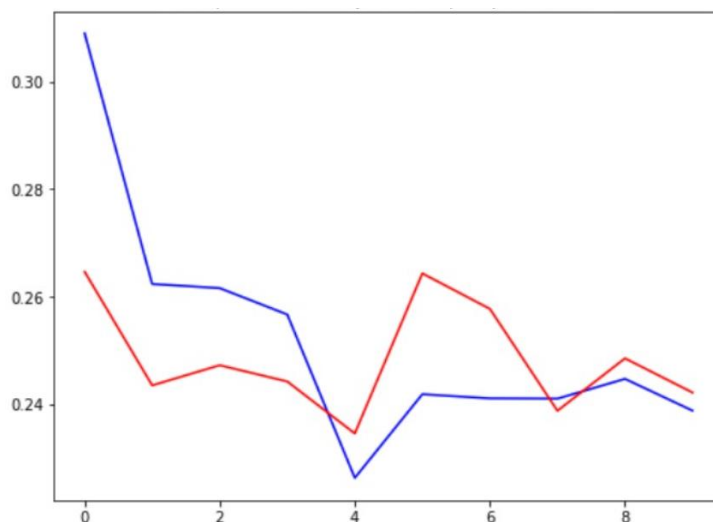


Рисунок Б.4 – Втрати моделі

Як приклад роботи моделі можна навести рис. Б.5, на якому синім зображено реальні, усереднені дані фінансування дітей-сиріт з урахуванням всіх необхідних видатків, а прогнозовані зображені жовтим. Як можемо побачити результати наближені одні до одних.

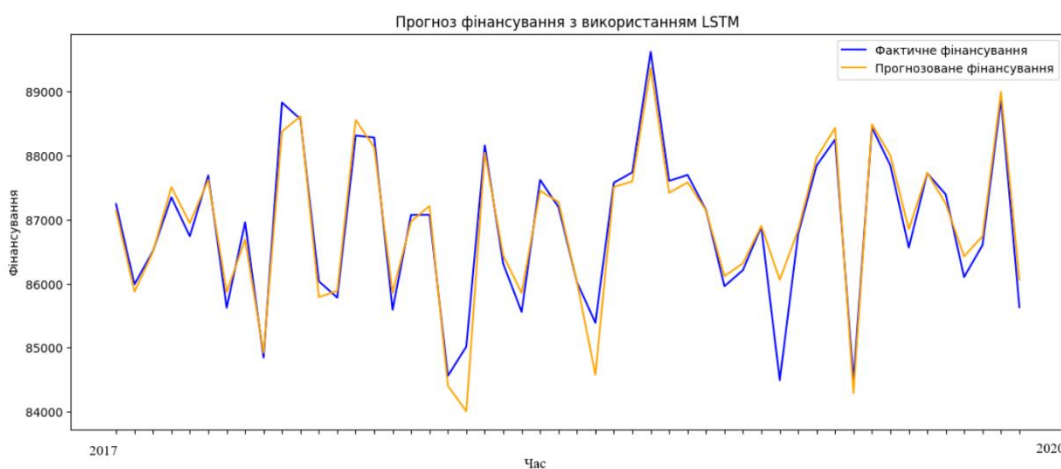


Рисунок Б.5 – Реальні й прогнозовані дані

ДОДАТОК В
Текст програми

В.1 Текст файла Prediction.py

```
import tensorflow as tf

import tkinter as tk

from tkinter import ttk

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

import numpy as np

import os

import pandas as pd

import matplotlib as mpl

import matplotlib.pyplot as plt


mpl.rcParams['figure.figsize'] = (8, 6)

mpl.rcParams['axes.grid'] = False


csv_path, _ = os.path.splitext(zip_path)


funding_df = pd.read_csv(csv_path)

funding_df.head()
```

```
selected_features = ['Фінансування','К-ть абітурієнтів', 'Похибка вступу',
'Інфляція', 'Субвенція','Зарплата службовцям','Частка ДС-вступників','Похибка
ДС-випускн.','Стипендія ДС','Вік студентів']
```

```
data_types_selected_features = {
    'Фінансування': np.dtype('float64'),
    'К-ть абітурієнтів': np.dtype('float64'),
    'Похибка вступу': np.dtype('float64'),
    'Інфляція': np.dtype('float64'),
    'Субвенція': np.dtype('float64'),
    'Зарплата службовців': np.dtype('float64'),
    'Частка ДС-вступників': np.dtype('float64'),
    'Похибка ДС-випускн.': np.dtype('float64'),
    'Стипендія ДС': np.dtype('float64'),
    'Вік студентів': np.dtype('float64')
}
```

```
funding_features = funding_df[selected_features]
```

```
funding_features = funding_features.astype(data_types_selected_features)
```

```

funding_features.index = funding_df['Date Time']

funding_features.head()


funding_dataset = funding_features.values

data_mean = funding_dataset[:TRAIN_SPLIT].mean(axis=0)

data_std = funding_dataset[:TRAIN_SPLIT].std(axis=0)

funding_dataset = (funding_dataset - data_mean) / data_std


def mul_data(data_set, target, st_i, end_i, h_s,
             t_s, step, s_step=False):

    data = []

    labels = []

    st_i = st_i + h_s

    if end_index is None:

        end_index = len(data) - t_s


    for i in range(start_index, end_index):

        indices = range(i - h_s, i, step)

        data.append(data_set[indices])

```

```

    if single_step:

        labels.append(target[i + t_s])

    else:

        labels.append(target[i:i + t_s])

return np.array(data), np.array(labels)

history = 840

future_target = 64

STEP = 3

x_train, y_train = mul_data(funding_dataset, funding_dataset[:, 1], 0,

                             len(funding_dataset)*0.7, history,

                             future_target, STEP,

                             s_step=True)

x_val, y_val = mul_data(funding_dataset, funding_dataset[:, 1],

                         len(funding_dataset)*0.7, None, history,

                         future_target, STEP,

                         s_step=True)

```

```
def time_gen(length):  
    return list(range(-length, 0))  
  
def make_plot(pdata, delta, title):  
  
    time_steps = time_gen(pdata[0].shape[0])  
  
    if delta:  
        future = delta  
    else:  
        future = 0  
  
    plt.title(title)  
  
    for i, x in enumerate(pdata):  
        if i:  
            plt.plot(future, pdata[i], marker[i], markersize=10,  
                    label=labels[i])  
        else:  
            plt.plot(time_steps, pdata[i].flatten(), marker[i], label=labels[i])
```

```
plt.legend()

plt.xlim([time_steps[0], (future + 5) * 2])

plt.xlabel('Data')

return plt
```

```
BATCH = 512
```

```
BUFFER = 20000
```

```
EVALUATION = 120
```

```
EPOCHS = 15
```

```
train_data = tf.data.Dataset.from_tensor_slices((x_train, y_train))
```

```
train_data = train_data.cache().shuffle(BUFFER).batch(BATCH).repeat()
```

```
val_data = tf.data.Dataset.from_tensor_slices((x_val, y_val))
```

```
val_data = val_data.batch(BATCH).repeat()
```

```
step_model = tf.keras.models.Sequential()
```

```
step_model.add(tf.keras.layers.LSTM(64,input_shape=x_train.shape[-2:]))
```

```
step_model.add(tf.keras.layers.Dense(1))
```

```
step_model.compile(optimizer=tf.keras.optimizers.RMSprop(), loss='mae')
```

```
for x, y in val_data.take(1):

    print(step_model.predict(x).shape)

step_history = step_model.fit(train_data, epochs=EPOCHS,

                               steps_per_epoch=EVALUATION,

                               validation_data=val_data,

                               validation_steps=50)

def pl_gen_history(history, title):

    loss = history.history['loss']

    val_loss = history.history['val_loss']

    epochs = range(len(loss))

    plt.figure()

    plt.plot(epochs, loss, 'b', label='Втрати при навчанні')

    plt.plot(epochs, val_loss, 'r', label='Втрати при оцінюванні')

    plt.title(title)
```

```
plt.legend()
```

```
plt.show()
```

```
pl_gen_history(step_history,  
               'Втрати при навчанні та оцінювані моделі')
```

```
for x, y in val_data.take(1):  
    plot = show_plot([x[0][:, 1].numpy(), y[0].numpy(),  
                     single_step_model.predict(x)[0]], 12,  
                     'Предбачення')  
    plot.show()
```

```
class FinancialWindowGenerator():  
    def __init__(self, input_width, label_width, shift,  
                 train_data=train_data, val_data=val_data, test_data=test_data,  
                 label_columns=None):
```

```
    self.train_data = train_data
```

```
    self.val_data = val_data
```

```
self.test_data = test_data

self.label_columns = label_columns

if label_columns is not None:

    self.label_columns_indices = {name: i for i, name in
                                  enumerate(label_columns)}

self.column_indices = {name: i for i, name in
                       enumerate(train_data.columns)}

self.input_width = input_width

self.label_width = label_width

self.shift = shift

self.total_window_size = input_width + shift

self.input_slice = slice(0, input_width)

self.input_indices = np.arange(self.total_window_size)[self.input_slice]

self.label_start = self.total_window_size - self.label_width

self.labels_slice = slice(self.label_start, None)

self.label_indices = np.arange(self.total_window_size)[self.labels_slice]
```

```

def __repr__(self):

    return '\n'.join([

        f'Total window size: {self.total_window_size}',

        f'Input indices: {self.input_indices}',

        f'Label indices: {self.label_indices}',

        f'Label column name(s): {self.label_columns}'])


def split_financial_window(self, features):

    input_data = features[:, self.input_slice, :]

    label_data = features[:, self.labels_slice, :]

    if self.label_columns is not None:

        label_data = tf.stack(

            [label_data[:, :, self.column_indices[name]] for name in self.label_columns],

            axis=-1)

    input_data.set_shape([None, self.input_width, None])

    label_data.set_shape([None, self.label_width, None])

    return input_data, label_data

```

```
FinancialWindowGenerator.split_financial_window = split_financial_window
```

В.2 Текст класу інтерфейсу

```
class PredictionApp:
```

```
    def __init__(self, root):
```

```
        self.root = root
```

```
        self.root.title("Прогнозування фінансування з використанням LSTM")
```

```
        self.figure, self.ax = plt.subplots(figsize=(10, 5))
```

```
        self.canvas = FigureCanvasTkAgg(self.figure, master=root)
```

```
        self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
```

```
        self.update_button = ttk.Button(root, text="Оновити графік",
command=self.update_plot)
```

```
        self.update_button.pack()
```

```
self.label_actual = ttk.Label(root, text="Фактичне фінансування:")  
  
self.label_actual.pack()  
  
self.label_actual_value = ttk.Label(root, text=str(y_test.values))  
  
self.label_actual_value.pack()  
  
  
self.label_predicted = ttk.Label(root, text="Прогнозоване фінансування:")  
  
self.label_predicted.pack()  
  
self.label_predicted_value = ttk.Label(root, text=str(predictions.flatten()))  
  
self.label_predicted_value.pack()  
  
  
def update_plot(self):  
  
    self.ax.clear()  
  
    self.ax.plot(y_test.index, y_test.values, label='Фактичне фінансування',  
color='blue')  
  
    self.ax.plot(y_test.index, predictions.flatten(), label='Прогнозоване  
фінансування', color='orange')  
  
    self.ax.legend()  
  
    self.canvas.draw()
```

```
root = tk.Tk()

app = PredictionApp(root)

root.mainloop()
```

ДОДАТОК Г
Слайди презентації

Національний університет «Запорізька Політехніка»

Факультет комп'ютерних наук і технологій

Кафедра програмних засобів

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИНТЕЗУ НЕЙРОМЕРЕЖЕВИХ ДІАГНОСТИЧНИХ МОДЕЛЕЙ ДЛЯ ПРОГНОЗУВАННЯ ФІНАНСУВАННЯ СИРІТ У ЗАКЛАДАХ ВИЩОЇ ОСВІТИ

Виконав
Студент гр. КНТ-122м

К.Д. Цибульський

Керівник
Доцент

О.О. Степаненко

Рисунок Г.1 – Слайд №1

Мета, об'єкт, предмет дослідження

Об'єкт дослідження – фінансування сиріт у закладах вищої освіти.

Предмет дослідження – методи та алгоритми синтезу нейромережевих діагностичних моделей для прогнозування фінансування сиріт у закладах вищої освіти.

Мета роботи – розробка нейромережевих діагностичних моделей для прогнозування фінансування сиріт у закладах вищої освіти.

Актуальність розробки обумовлена використанням передових технологій управління та аналізу даних, зокрема нейромережевих моделей, що дозволяє покращити точність прогнозування та оптимізувати процеси управління фінансами.. А також загостренням проблеми із збільшенням кількості сиріт у наслідок бойових дій та наданням дітям-сиротам та дітям, позбавленим батьківського піклування належних умов для здобуття вищої освіти.

Рисунок Г.2 – Слайд №2

Постановка завдання до роботи

Були визначені наступні параметри для побудови діагностичної моделі:

Параметр	Позначення	Одиниці виміру
К-ть абітурієнтів	X_1	Одиниць
Похибка вступу	X_2	%
Інфляція	X_3	%
Субвенція	X_4	грн
Зарплатня службовцям	X_5	грн
Частка ДС-вступників	X_6	%
Похибка ДС-випускн.	X_7	%
Стипендія ДС	X_8	грн
Вік студентів	X_9	роки
Напрямок навчання	X_{10}	

Рисунок Г.3 – Слайд №3

Огляд існуючих мов програмування

Характеристика	Python	R	C++	JavaScript
Швидкодія виконання	Середня	Середня	Висока	Середня
Спільнота та Екосистема	Велика	Велика	Значна	Велика
Легкість вивчення	Висока	Висока	Середня	Висока
Можливості статистики	Обмежені	Високі	Середні	Обмежені
Бібліотеки для нейромереж	TensorFlow, PyTorch, Keras	TensorFlow, Keras	TensorFlow, Caffe	TensorFlow.js, Brain.js

Рисунок Г.4 – Слайд №4

Огляд існуючих моделей ШНМ

Характеристика	Рекурентні нейронні мережі (RNN)	Довготривалі краткострокові пам'яті (LSTM)	GRU мережі
Основна ідея	Враховує попередні вхідні дані за допомогою зв'язків в часі.	Розширення RNN, яке забезпечує зберігання та використання довготривалих залежностей.	Використовує вентилі для управління потоком інформації, ефективне для довгих послідовностей.
Застосування	Прогнозування часових рядів, рекурентні зв'язки в даних.	Прогнозування складних динамічних процесів, довготривалі зв'язки в даних.	Прогнозування з використанням вентилів, ефективне для довгих послідовностей.
Можливі проблеми	Проблема зникнення/згасання градієнту при тривалих послідовностях.	Можливий ефект зникнення/згасання градієнту; схильність до перенавчання.	Може бути менше схильним до перенавчання, але все ще можливі проблеми.
Ефективність	Ефективні для послідовних даних, таких як часові ряди.	Добре пристосовані для моделювання довготривалих залежностей.	Добре справляється з довгими послідовностями, швидше за LSTM.
Застосування в галузі фінансів	Прогнозування фінансових часових рядів.	Прогнозування фінансових тенденцій та складних ринкових умов.	Прогнозування фінансових параметрів з використанням вентилів.

Рисунок Г.5– Слайд №5

Огляд бібліотек машинного навчання

Характеристика	TensorFlow	PyTorch	Keras
Простота використання	Середня	Висока	Висока
Спільнота	Велика	Зростає	Велика
Гнучкість та Контроль	Висока	Висока	Середня
Обчислювальні граfi	Статичні (з TensorFlow 2.0 – динамічні)	Динамічні	Статичні (динамічні з TensorFlow 2.0)
Вбудовані функції та моделі	Велика кількість	Зростає	Залежить від бекенда
Призначення	Стандарт для досліджень та продакшну	Підходить для експериментів та налагодження	Зручний для швидких експериментів та простих задач

Рисунок Г.6 – Слайд №6

Схема роботи LSTM моделі

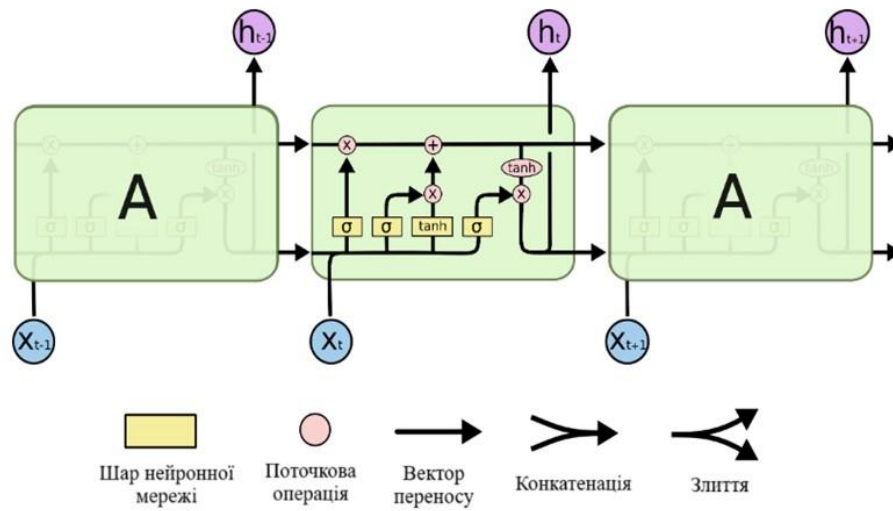


Рисунок Г.7 – Слайд №7

Приклад вхідних

applicants	entrance_error	inflation	subvention	salary_staff
2978.75	6.74	6.53	3121.36	13641.20
2692.03	3.63	4.34	2302.82	10972.86
2837.51	5.41	5.70	2913.02	11189.56
1816.18	1.55	3.41	3198.25	13407.65
2141.10	5.47	6.63	2946.22	12346.58
...	...	...	...	...
2176.53	2.90	4.84	1371.84	10911.69
1651.61	2.51	5.46	2451.09	8878.53
1004.16	1.42	6.65	2624.88	7070.88
2547.91	4.49	3.27	4422.19	8446.71
1524.90	5.28	4.97	2127.54	9251.12
orphans_ratio	graduate_error	orphans_stipend	students age	
2.22	2.22	2469.96	18.46	
3.09	4.86	3939.09	18.37	
3.82	2.59	2903.38	18.94	
3.73	2.33	3772.42	18.52	
3.23	4.43	3836.89	18.08	
...	...	...	...	
1.59	3.36	3403.04	19.35	
4.75	5.95	3010.26	18.85	
1.28	4.75	2234.36	18.45	
4.16	3.74	2704.33	19.34	
3.82	5.08	3269.06	18.49	

Приклад повідомлення про помилку

```

ValueError                                Traceback (most recent call last)
<ipython-input-16-fbdc269df13d> in <cell line: 91>()
    89 plt.figure(figsize=(15, 6))
    90 plt.plot(y_test.index, y_test.values, label='Фактичне фінансування', color='blue')
--> 91 plt.plot(y_test.index, {prediction[0][0]}, label='Прогнозоване фінансування', color='orange')
    92 plt.title('Прогноз фінансування з використанням LSTM')
    93 plt.xlabel('Timestamp')
  
```

Рисунок Г.8 – Слайд №8

Графік втрат моделі при навчанні та перевірці

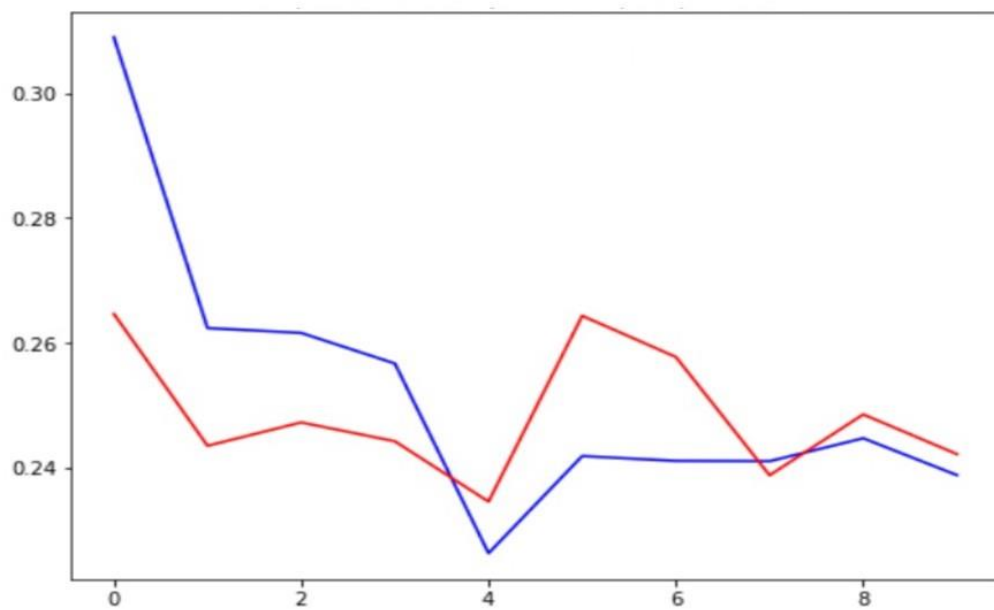


Рисунок Г.9 – Слайд №9

Графік прогнозованих і реальних даних

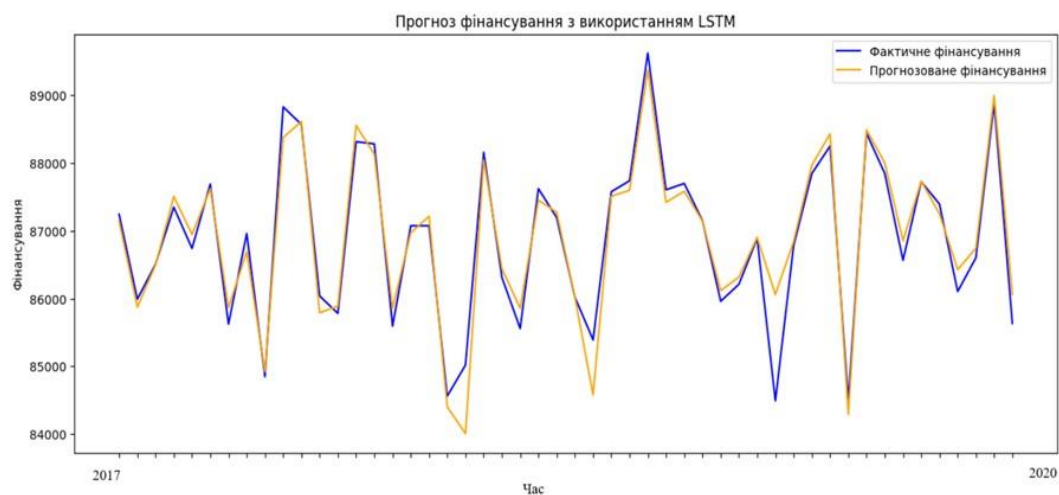


Рисунок Г.10 – Слайд №10

Висновки

Для виконання дипломної було зроблено аналіз предметної області та досліджено існуючі проблеми та шляхи їх вирішення.

Також в рамках аналізу предметної області було проведено дослідження підходящих моделей штучних нейронних мереж для рішення поставленої задачі. Було сформовано перелік переваг та недоліків для врахування їх при подальшому проектуванні.

За мету було поставлено полегшити процеси фінансового планування та сприяти більш ефективному використанню ресурсів для забезпечення доступу дітей-сиріт до вищої освіти.

В результаті роботи було розроблено нейромережеву діагностичну модель яка здатна робити прогноз на основі статистичних даних.



Рисунок Г.11 – Слайд №11