НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Факультет прикладної математики
Кафедра прикладної математики

«До захисту допущено»
Завідувач кафедри
_____ О. Р. Чертов
«___» ___ 2017 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.040301 «Прикладна математика»

на тему: Математична модель формування дози внутрішнього опромінення внаслідок аварії на ЧАЕС на основі нейронних мереж

Виконав: студент IV курсу, групи КМ-31

	Чернявський Андрій Сергійович	_____
Керівник	старший викладач Любашенко Н.Д.	_____
Консультант із нормоконтролю	старший викладач Мальчиков В. В.	_____
Рецензент	к.т.н., доцент Павловський В.І.	_____

Засвідчую, що в цій дипломній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент _____

Київ — 2017

Національний технічний університет України

«Київський політехнічний інститут»

Факультет прикладної математики

Кафедра прикладної математики

Рівень вищої освіти — перший (бакалаврський)

Напрям підготовки 6.040301 «Прикладна математика»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О. Р. Чертов

«___» _____ 2017 р.

ЗАВДАННЯ

на дипломну роботу студенту

Чернявському Андрію Сергійовичу

1. Тема роботи: «Математична модель формування дози внутрішнього опромінення внаслідок аварії на ЧАЕС на основі нейронних мереж»,
керівник роботи Любашенко Наталія Дмитрівна, ст. викладач,
затверджені наказом по університету від «19» травня 2017 р. № 1574-С.

2. Термін подання студентом роботи: «14» червня 2017 р.

3. Вихідні дані до роботи: вихідними даними є набір даних, наданий відділом радіоекології інституту агроекології та природокористування НААН України.

4. Зміст роботи: виконати аналіз існуючих методів розв’язання поставленої задачі, обрати структуру штучної нейронної мережі, провести попередню обробку даних для використання їх при побудові математичної моделі формування внутрішньої дози опромінення, здійснити програмну реалізацію спроектованої системи, провести тестування розробленої системи на контрольних прикладах, оформити документацію до дипломної роботи.

5. Перелік ілюстративного матеріалу: архітектурні графи нейронних мереж, знімки екранних форм, графіки функцій, графіки похибок моделі, порівняльна діаграма зі значеннями похибок існуючих методів.

6. Дата видачі завдання: «20» лютого 2017 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд літератури за тематикою та збір даних	12.11.2017	
2	Проведення порівняльного аналізу математичних методів аналізу та моделювання даних	14.12.2017	
3	Проведення порівняльного аналізу математичних методів аналізу та моделювання даних	24.12.2017	
4	Підготовка матеріалів першого розділу роботи	01.02.2017	
5	Розроблення математичного забезпечення для моделювання формування дози внутрішнього опромінення внаслідок аварії на ЧАЕС	01.03.2017	
6	Підготовка матеріалів другого розділу роботи	15.03.2017	
7	Підготовка матеріалів третього розділу роботи	05.04.2017	
8	Розроблення програмного забезпечення для моделювання формування дози внутрішнього опромінення	15.04.2017	
9	Підготовка матеріалів четвертого розділу роботи	03.05.2017	
10	Оформлення пояснювальної записки	01.06.2017	

Студент _____ Чернявський А.С.

Керівник роботи _____ Любашенко Н.Д.

АНОТАЦІЯ

Дипломну роботу виконано на 65 аркушах, вона містить 2 додатки та перелік посилань на використані джерела з 27 найменувань. У роботі наведено 26 рисунків та 2 таблиці.

Метою даної роботи є створення математичної моделі формування дози внутрішнього опромінення на основі нейронних мереж. У роботі розглянуто такі методи, як дерева прийняття рішень, метод опорних векторів, методи регресії, нейронні мережі, метод випадкових лісів. На основі сформульованих критеріїв для розв'язання поставленої задачі вибрано нейронні мережі.

Було розроблено систему, що реалізує обраний метод. Вихідними даними для системи є дані, надані відділом радіоекології інституту агроекології та природокористування НААН України. Виконано тестування розробленої системи.

Ключові слова: штучні нейронні мережі, навчання з учителем, доза внутрішнього опромінення, машинне навчання, попередня обробка даних, багатошаровий перцептрон, множинна нелінійна регресія.

ABSTRACT

The thesis is presented on 65 pages. It contains 2 appendixes and bibliography of 27 references. Twenty six figures and two tables are shown in the thesis.

The goal of the thesis is to develop a mathematical model of internal radiation dose of the population due to the Chernobyl accident based on neural networks.

In the thesis, existing solutions are analyzed, such as decision trees, support vector machines, regression methods, neural networks, random forests. On the basis of defined criteria for the solution of the problem the neural network was chosen.

A system that implements the chosen method was developed and tested. Input data for the system was provided by the Department of Radiology from Institute of Agroecology and Environmental Management of National Academy of Agricultural Sciences of Ukraine.

Keywords: artificial neural networks, supervised learning, internal radiation dose, machine learning, data preprocessing, multilayer perceptron, plural nonlinear regression.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЕОМ – електронна обчислювальна машина.

МЗП – метод зворотного поширення.

ПЗ – програмне забезпечення.

ЧАЕС – Чорнобильська атомна електростанція.

ШНМ – штучна нейронна мережа.

DNN (Deep neural network) – глибока нейронна мережа.

MAE (Mean absolute error) – середня абсолютна похибка.

MLP (Multilayer perceptron) – багатошаровий перцептрон.

MSE (Mean squared error) – середньоквадратична похибка.

RBF (Radial basis function) – функція радіального базису.

SBF (Sigmoidal basis function) – функція сигмоідального базису.

SVM (Support vector machine) – метод опорних векторів.

ВСТУП

У наш час інформаційні технології досягли значного розвитку й продовжують розвиватися. Разом із тим, науковці та дослідники продовжують покращувати свої знання про Всесвіт, вивчати залежності між оточуючими їх предметами, організмами та явищами, а отже й підвищується актуальність задач регресійного аналізу.

Однією з таких задач є задача створення математичної моделі формування дози внутрішнього опромінення населення внаслідок аварії на ЧАЕС. Україна зазнала значних збитків як під час Чорнобильської катастрофи, так і в період ліквідації наслідків. Викиди радіації викликали забруднення прилеглої території, а пізніше поширилися на все навколишнє середовище. Як відомо, на сьогоднішній день немає жодного місця на Землі, де б не залишився слід радіоактивної хмари. Вона облетіла весь світ.

З часів жахливої аварії минув тридцять один рік, проте відголоски її наслідків ми можемо спостерігати і в наші дні. Проживання людини на радіоактивно забруднених територіях неминуче зумовлює певний ступінь ризику отримати додаткове внутрішнє опромінення внаслідок надходження радіонуклідів до її організму з продуктами харчування місцевого виробництва, що призводить до змін стану здоров’я [1].

Емпіричні дані, отримані в ході опитувань та експериментів, є початковим етапом взаємодії між людиною та базою даних. Збір та зберігання даних стали значно дешевшими, що збільшило обсяги отримуваної інформації. На цьому етапі дані називаються «сирими» (raw data) [2], адже вони не зазнали ще ніякої обробки та аналізу, і така інформація є майже марною. Вона не дає жодних знань для маркетологів, економістів, лікарів, геофізиків та інших представників природничих наук.

На щастя, сьогодні існують легкодоступні інструменти, здатні уповільнити нескінченний потік інформації та перетворити її на знання. Стало простіше, ніж коли-небудь, отримати цінне розуміння поведінки явищ, що відбуваються навколо. Це дозволяє швидше і ефективніше реагувати на зміни. Наприклад, розуміння поведінки радіонуклідів на певній території з плином часу може підказати план дій на майбутнє, який допоможе максимально ефективно зберегти та покращити стан здоров'я місцевого населення.

На сьогоднішній день, в розумінні тенденцій та явищ фаворитом вважається штучний інтелект. Комп'ютеру надаються властивості до пізнання, що співставні із можливостями людського мозку, і він вдало ними користується. Прогрес дійшов до того, що спеціалісти інформаційних технологій отримують результати не завдяки програмуванню, а за допомогою навчання штучного інтелекту. Електронні обчислювальні машини вже успішно можуть робити прогнози, розпізнавати обличчя, емоції, образи, класифікувати інформацію та багато чого іншого.

У даній роботі проведено огляд існуючих математичних методів та програмних інструментів для аналізу та обробки даних, а також для вирішення задач класифікації та нелінійної множинної регресії.

1 ПОСТАНОВКА ЗАДАЧІ

Метою даної дипломної роботи є створення математичного та програмного забезпечення для створення математичної моделі формування дози внутрішнього опромінення населення внаслідок аварії на ЧАЕС.

Потрібно максимально проаналізувати отримані дані та врахувати наступні показники:

а) кількісні:

1) вік, років;
2) площа лісу в радіусі 3 км, %;
3) питома площа лісу на 1 люд. в радіусі 3 км, км^2/люд.;
4) рівень забруднення території Цезієм-137, сБк/м^2;
5) відстань до найближчого районного центра, км;
6) відстань до найближчого лісового масиву, км;
7) час від аварії на ЧАЕС до вимірювання внутрішньої дози, місяців;

б) якісні:

1) тип ґрунту;
2) професія;
3) зона забруднення.

При розробленні відповідного забезпечення потрібно розв’язати наступні завдання:

а) провести порівняльний аналіз існуючих методів аналізу даних;

б) вибрати й адаптувати існуючий метод для вирішення задачі створення математичної моделі формування дози внутрішнього опромінення населення внаслідок аварії на ЧАЕС;

в) розробити програмне забезпечення на базі вибраного математичного методу;

г) провести тестування програмної реалізації на контрольних прикладах.

Реалізована система має задовольняти такі вимоги:

а) моделювати залежність дози внутрішнього опромінення від множини факторів;

б) мати мінімальну похибку;

в) перетворювати вихідні дані у сумісний з нейронною мережею формат;

г) адаптуватися шляхом регулювання основних параметрів моделі;

д) зберігати ваги синапсів після навчання, для забезпечення можливості використання системи без попереднього навчання у майбутньому;

е) бути спроможною навчатися на нових вибірках даних.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ МОДЕЛЮВАННЯ ДАНИХ

2.1 Аналіз предметної області

Проживання людини на радіоактивного забруднених територіях неминуче зумовлює певний ступінь ризику отримати додаткове внутрішнє опромінення внаслідок надходження радіонуклідів до її організму з продуктами харчування місцевого виробництва, що призводить до змін стану здоров'я [1]. Тому дослідження вчених зосереджені на моніторингу забруднення сільгоспугідь поблизу населених пунктів, радіоактивності продуктів харчування, тощо. Без знання причини формування дози опромінення, її величини не можна правильно оцінити або прогнозувати дозу опромінення, тим більше планувати протирадіаційні заходи.

Нині проведення протирадіаційних заходів ґрунтується не на закономірностях формування дози опромінення людини, а на тому, що розподіл дози серед мешканців певного населеного пункту співпадає з розподілом забруднення в навколишньому природному середовищі та/або з розподілом питомої активності радіонуклідів у продуктах харчування. Тому всі дослідження зосереджені на моніторингу забруднення сільгоспугідь поблизу населених пунктів, радіоактивності продуктів харчування, тощо. Однак, як би коректно не були оцінені ці фактори, неможливо повно розкрити причини формування дози опромінення, без знання яких не можна правильно оцінити або прогнозувати дозу, тим більше планувати протирадіаційні заходи.

У дослідженні дозоутворення в групах однотипної соціальної поведінки сільських мешканців узято за основу концепцію А.М. Скрябіна [3, 4], суть якої полягає в тому, що людина або група людей, що має певні особистісні та соціально-економічні характеристики, взаємодіє з навколишнім природним середовищем проживання, сприяючи формуванню дози, яка є його властивістю. Цей підхід пояснює деякі закономірності дозоутворення, але пов'язаний з трудомістким збором

непрямих даних про окремих людей або ж їх груп зі схожою поведінкою і залишає відкритими деякі питання. Наприклад, неможливо пояснити наявність високих доз внутрішнього опромінення у деяких людей, що проживають у сім'ях, інші члени яких мають значно нижчу дозу опромінення.

Аналіз причини формування дози опромінення населення дає доволі чітке уявлення про механізми формування дози внутрішнього опромінення місцевих мешканців. У віддалений період після аварії на ЧАЕС спостерігається диференціація населених пунктів за характером формування дози внутрішнього опромінення. Це можна пояснити соціально-демографічними та еколого-економічними їх особливостями.

Очевидними є вплив середнього віку мешканців населеного пункту, рівня їхньої освіти, контакту з лісом, частки робітників певних категорій, соціально-економічне становище, тощо. Все це, в кінцевому рахунку, формує харчову поведінку. Варто зазначити, що дотепер в переважній більшості постраждалих від аварії на Чорнобильській АЕС регіонах спостерігався дещо інший характер розподілу доз внутрішнього опромінення, а саме логарифмічно нормальний.

Споживання радіоактивно забруднених продуктів харчування (грибів, ягід, дичини, молока) є основним дозоутворюючим чинником, що визначається низкою непрямих чинників, пов'язаних з соціально-економічними характеристиками населеного пункту [4]. Отже, можна припустити, що кожен населений пункт повинен характеризуватися «своєю» дозою. Оскільки населений пункт формує дозу, то і доза є властивістю конкретного населеного пункту.

2.2 Дерева ухвалення рішень

Дерева ухвалення рішень - один з методів автоматичного аналізу величезних масивів даних [5]. Зазвичай використовуються для рішення задач трьох класів:

1) класифікації;

2) опис даних;

3) регресія.

Дерево ухвалення рішень - це графічне зображення послідовності рішень і станів середовища з указівкою відповідних ймовірностей і виграшів для будь-яких комбінацій альтернатив і станів середовища (рис. 2.1).

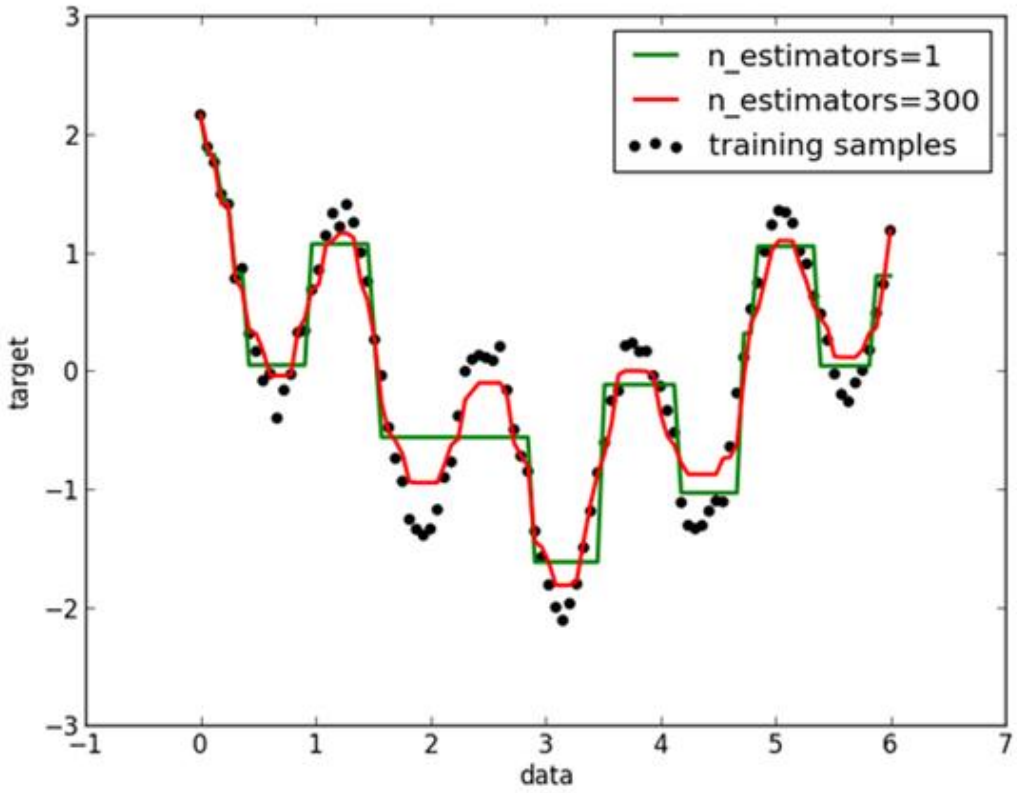


Рисунок 2.1 – Приклад дерева ухвалення рішень

Побудова дерева ухвалення рішень виконується "зверху вниз" - від задач більш складних, більш важливих - до завдань менш складних, що вимагають менше часу (коштів, сил, ресурсів) для їх здійснення. На схемі дерева верхнє положення займає кінцева мета розв'язання проблеми (кінцевий результат).

Чим складніше можна вирішити завдання, тим більше має бути число рівнів розгляду проблеми і тим більше число завдань, що вирішуються на кожному рівні.

Для кожного дерева будується матриця, вводяться коефіцієнти взаємної корисності рішень, одержувані опитуванням експертів. Вони показують вплив ступеня важливості одних рішень на інші.

Переваги методу:

- швидкість;
- легкий у використанні;
- невибагливий до типу вихідних даних;
- зрозуміло, які атрибути є найважливішими.

Недоліки методу:

- схильний до перенавчання;
- потрібно слідкувати за довжиною гілок і, за потреби, «обрізати» їх;
- незначні зміни даних можуть сильно змінити дерево.

2.3 Метод опорних векторів

Метод опорних векторів — це метод аналізу даних для класифікації та регресійного аналізу[5]. Однією з найбільш популярних методологій машинного навчання по прецедентах(засноване на виявленні загальних закономірностей по емпіричним даним) є побудова машини опорних векторів, відомої в англомовній літературі під назвою SVM (Support Vector Machine).

Опорно-вектора машина будує гіперплощину, або набір гіперплощин (рис. 2.2) у просторі високої або нескінченної вимірності, які можна використовувати для класифікації, регресії та інших задач. Інтуїтивно, добре розділення досягається гіперплощиною, яка має найбільшу відстань до найближчих точок тренувальних даних будь-якого з класів (так зване функційне розділення), оскільки в загальному випадку що більшим є розділення, то нижчою є похибка узагальнення класифікатора [6].

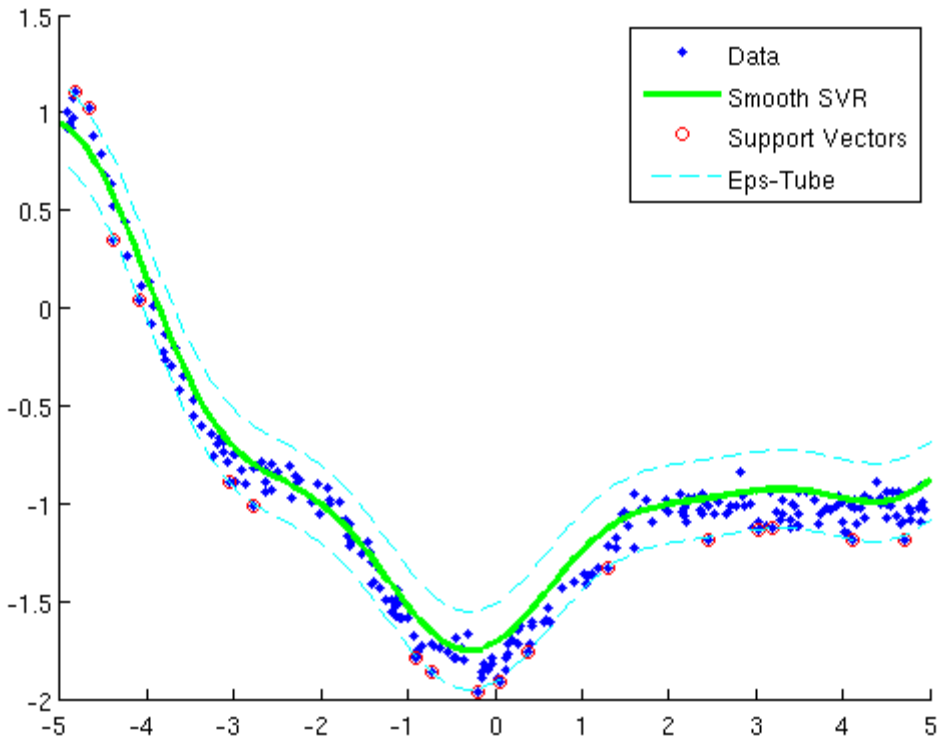


Рисунок 2.2 – Приклад використання SVM

Переваги методу:

- швидкий підбір факторів;
- ефективний у багатовимірних просторах;
- рішення точно буде глобальним мінімумом;
- не дуже вибагливий до типу даних, з якими працює;

- малі затрати пам’яті ЕОМ.

Недоліки методу:

- час навчання;
- одне вихідне значення;
- погано працює, коли є шум в даних

2.4 Штучні нейронні мережі

Штучні нейронні мережі (ШНМ) - це сукупність штучних нейронів, які виконують роль суматорів (рис. 2.

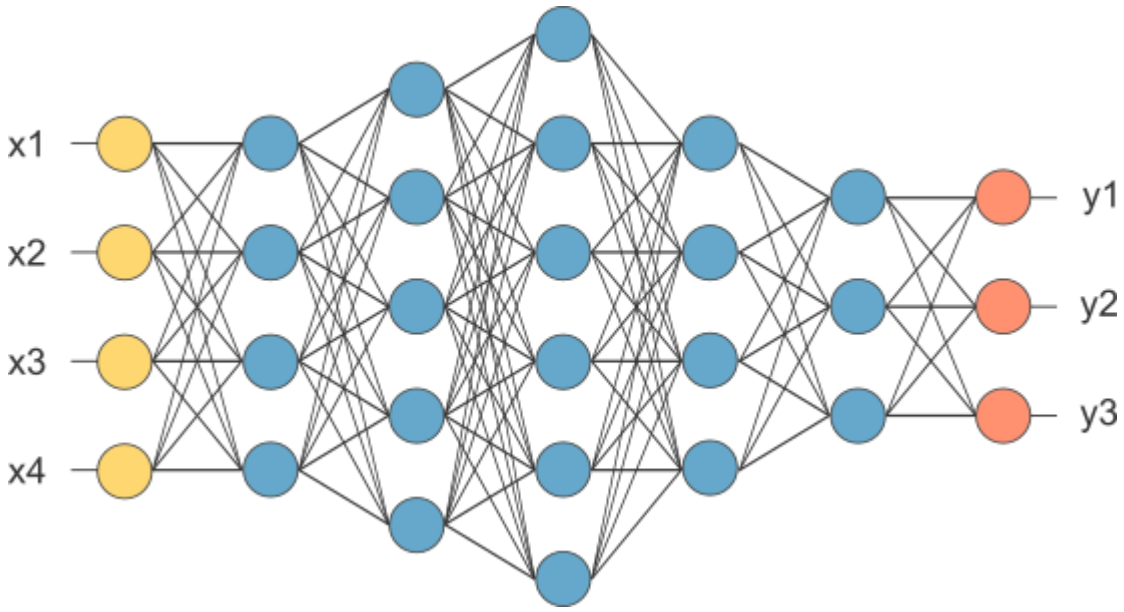


Рисунок 2.3 – Приклад штучної нейронної мережі

ШНМ потрібні для вирішення складних завдань, наприклад: прогнозування, розпізнавання образів, кластеризація, задачі керування. Так само вони застосовуються в області машинного навчання і штучного інтелекту.

Нейронні мережі не просто програмуються, вони навчаються, що і є однією з переваг. Мета навчання: знаходження коефіцієнтів зв’язків між нейронами(одиницями, з яких складається нейронна мережа) [7, 8]. У процесі навчання, нейронна мережа здатна виявляти складні залежності між вихідними даними, робити узагальнення, класифікувати інформацію. Тобто, після успішного навчання нейронна мережа може повернути результат, якого не було у навчальній вибірці, коли на вхід вона отримає нові дані, яких вона до цього часу не зустрічала.

Навчання нейронних мереж ділиться на три типи [9]:

I) навчання з учителем;

II) навчання без учителя;

III) навчання з підкріпленням.

Навчання з учителем передбачає відомі правильні відповіді, які нейронна мережа повинна отримати після обрахунків. Для такого типу навчання нам потрібні вихідні дані та результати.

Для навчання без вчителя достатньо мати лише вихідні дані. Мережа сама навчається виконувати поставлене завдання без втручання експериментатора. Такий тип, зазвичай, підходить, коли нейронна мережа повинна виявити внутрішні взаємозв’язки, залежності, закономірності, які існують між об’єктами.

Навчання з підкріпленням – такий тип навчання, коли нейронна мережа одержує винагороду в тих ситуаціях, коли отримує правильний результат і штрафи, коли отримує неправильний. Мережа отримує можливість знайти будь-який спосіб досягнення мети, доки він буде давати задовільний результат. Нейронна мережа починає «розуміти», чого від неї вимагають і намагається знайти найкращий шлях для досягнення цієї цілі.

Переваги методу:

- повертають гарний результат при роботі з даними, що мають велику кількість вихідних факторів;
- швидкість навчання;

- стійкість до шуму в даних;
- фіксований розмір;
- може мати багато вихідних значень.

Недоліки методу:

- велика кількість гіперпараметрів, які потрібно підбирати вручну;
- не можна зрозуміти алгоритм, за яким ШНМ отримала результат;
- можливість дійти до стану «перенавчання».

2.5 Метод випадкових лісів

Метод випадкових лісів - алгоритм машинного навчання, що полягає у використанні комітету вирішальних дерев [10].

Усі дерева комітету будуються незалежно один від одного за такою процедурою:

1. Генерується випадкова підвибірка з повторенням розміром n з навчальної вибірки. (Таким чином, деякі приклади потраплять в неї кілька разів, а приблизно $N/3$ прикладів не ввійдуть у неї взагалі)
2. Будується вирішальне дерево, яке класифікує приклади даної підвибірки, причому в ході створення чергового вузла дерева обирається ознака, на основі якої проводиться розбиття, не з усіх M ознак, а лише з m випадково вибраних. Вибір найкращого з цих m ознак може здійснюватися різними способами.
3. Дерево будується до повного вичерпання підвибірки і не піддається процедурі «відсікання» гілок.

Класифікація об'єктів проводиться шляхом голосування: кожне дерево комітету відносить об'єкт, який класифікується до одного з класів, і перемагає клас, за який проголосувало найбільше число дерев (рис. 2.4).

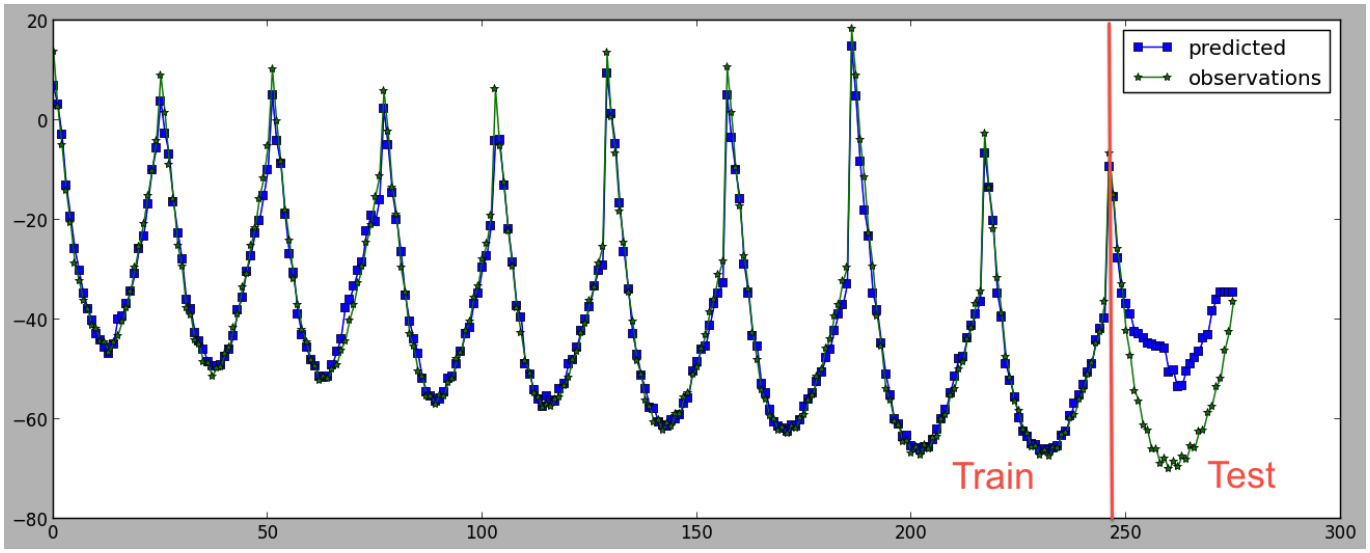


Рисунок 2.4 – Приклад застосування методу випадкових лісів для задачі регресії

Випадкові ліси, отримані в результаті застосування технік, описаних раніше, можуть бути природним чином використані для оцінки важливості змінних в задачах регресії та класифікації.

Для того, щоб оцінити важливість *j*-го фактора після тренування, значення *j*-го фактора перемішуються для всіх записів тренувального набору та помилка рахується знову. Важливість фактора оцінюється шляхом усереднення по всіх деревах різниці показників помилок до і після перемішування значень.

Переваги методу:

- легкі у використанні;
- можуть бути ефективно використані з великими базами даних;
- повертають гарний результат при роботі з даними, що мають велику кількість вихідних факторів.

Недоліки методу:

- результати не завжди легко зрозуміти, на відміну від дерев ухвалення рішень;
- швидкість обрахунку;
- погано взаємодіє з категоріальними змінними.

2.6 Методи регресійного аналізу

Методами регресії називають методи моделювання залежності між скаляром y та векторною змінною x. Взаємозв'язок між даними моделюється за допомогою лінійних та нелінійних функцій, а невідомі фактори моделі оцінюються за вихідними даними. За допомогою регресійного аналізу можна знайти, наскільки змінна залежить від інших, але не можна кластеризувати дані.

Переваги методу:

- повертають гарні результати при нормальному розподілі даних;
- прості формули для знаходження оптимальних значень факторів;
- широкий вибір базисних функцій;
- багато шляхів для регуляризації моделей;
- в будь-якому випадку знайдуться оптимальні коефіцієнти.

Недоліки методу:

- повертають погані результати при роботі з великою кількістю факторів;
- чутливі до даних, що значно відрізняються від інших (outliers);
- може бути важко інтерпретувати окремі коефіцієнти регресії.

2.6.1 Лінійна регресія

Лінійна регресійна модель (рис. 2.5) має наступний вигляд:

$$y(x, w) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_D x_D, \quad (2.1)$$

де w_i - коефіцієнти регресії;

x_i - фактори регресії;

D - кількість факторів регресії.

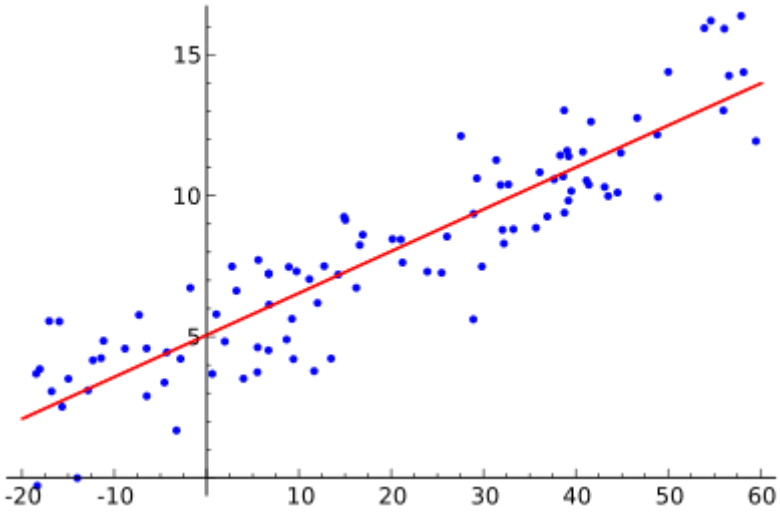


Рисунок 2.5 – Приклад лінійної регресії

Коефіцієнти регресії показують швидкість зміни залежного фактору від даного незалежного фактору, за тої умови, що інші фактори фіксовані.

Формулу (2.1) можна записати в наступному вигляді:

$$y(x,w) = \sum_{j=0}^{M-1} w_j x_j = w^T x, \tag{2.2}$$

де $w^T = (w_0, w_1, ..., w_{M-1})$ - вектор коефіцієнтів;

$x = (x_1, x_2, ..., x_{M-1})$ - вектор-стовпець факторів.

2.6.2 Поліноміальна регресія

Поліноміальна регресія від лінійної відрізняється лише представленням вектора-стовпця факторів:

$$x = (x^0, x^1, x^2, ..., x^j) \tag{2.3}$$

І тоді формула (2.1) набуває вигляду:

$$y(x, w) = w_0 x \ + w_1 x^1 + ... + w_D x^D \tag{2.4}$$

Завдяки цьому модель стає більш гнучкою. Графіки базисних функцій для поліноміальної регресії можна побачити на рис. 2.6:

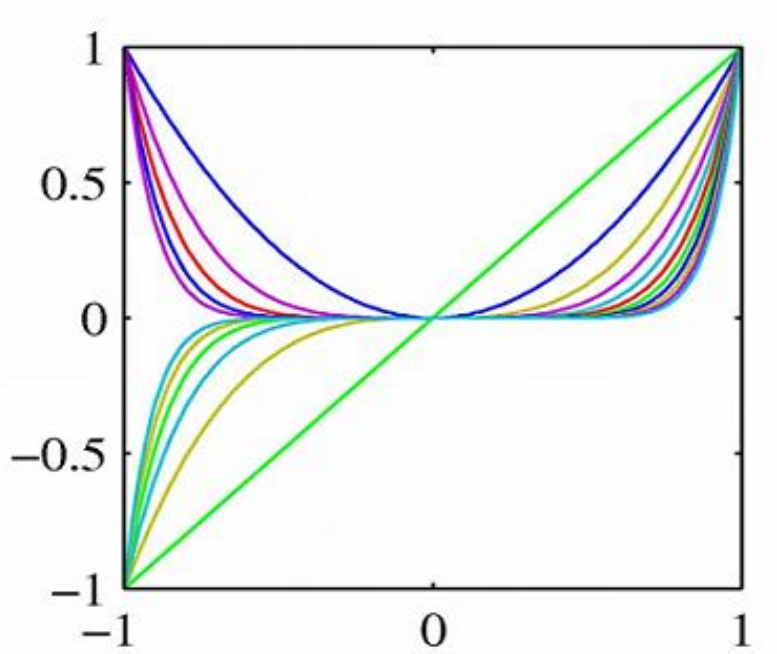


Рисунок 2.6 – Графіки функцій різних порядків, що є базисами поліноміальної регресії

2.6.3 Radial basis

Іншим прикладом є використання радіального базиса. У такому випадку маємо:

$$x = (e^{-\frac{(x-\mu_0)^2}{2s^2}}, e^{-\frac{(x-\mu_1)^2}{2s^2}}, ..., e^{-\frac{(x-\mu_D)^2}{2s^2}}), \quad (2.5)$$

де μ_j - середнє значення;

s - значення дисперсії.

І формула (2.1) буде записана в новому вигляді:

$$y(x, w) = w_0 e^{-\frac{(x-\mu_0)^2}{2s^2}} + w_1 e^{-\frac{(x-\mu_1)^2}{2s^2}} + w_2 e^{-\frac{(x-\mu_2)^2}{2s^2}} + ... + w_D e^{-\frac{(x-\mu_D)^2}{2s^2}} \quad (2.6)$$

Графік базисної функції для радіальної регресії можна побачити на рис. 2.7:

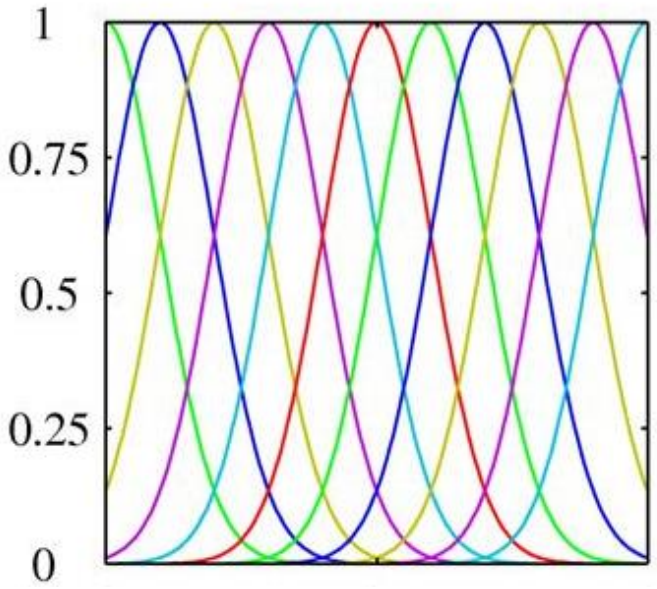


Рисунок 2.7 – Графік функції для регресії з радіальним базисом

2.6.4 Sigmoidal basis

При використанні регресії з сигмоїдальним базисом ми маємо:

$$x = \left(\frac{1}{1+e^{\frac{\mu_0 - x}{s}}} + \frac{1}{1+e^{\frac{\mu_1 - x}{s}}} + \frac{1}{1+e^{\frac{\mu_2 - x}{s}}} + ... + \frac{1}{1+e^{\frac{\mu_D - x}{s}}} \right) \tag{2.7}$$

Тепер підставимо вираз (2.7) у (2.1). Отримаємо наступне рівняння:

$$y(x,w) = w_0 \frac{1}{1+e^{\frac{\mu_0 - x}{s}}} + w_1 \frac{1}{1+e^{\frac{\mu_1 - x}{s}}} + ... + w_D \frac{1}{1+e^{\frac{\mu_D - x}{s}}} \tag{2.8}$$

Графік базисної функції для регресії з сигмоїдальним базисом представлено на рис. 2.8:

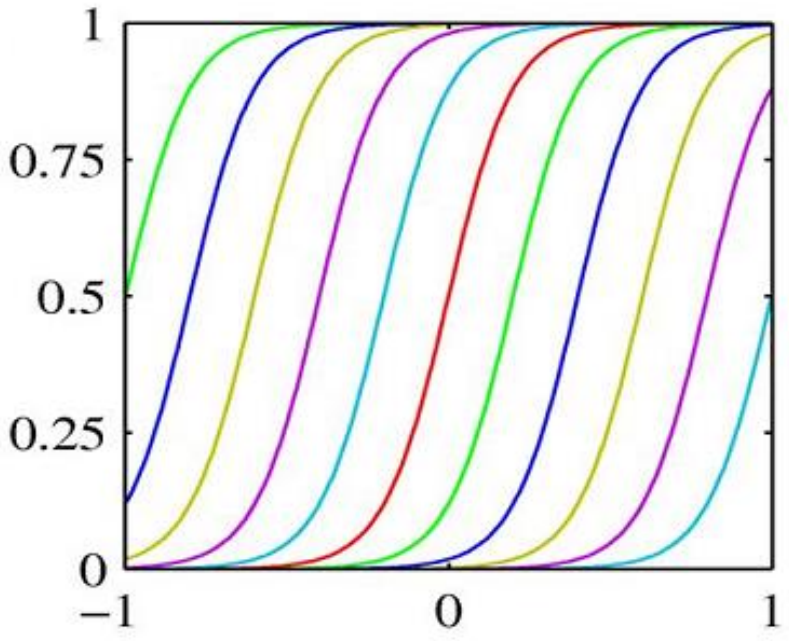


Рисунок 2.8 – Графік функції для регресії з сигмоїдальним базисом

2.7 Порівняння математичних методів розв’язання задачі

Порівняння існуючих методів було проведено на вибірці даних, що була надана відділом радіології інституту агроекології та природокористування НААН України, і містить 2992 рядки з даними. Порівняння було проведено за допомогою безкоштовного ПЗ для машинного навчання та аналізу даних під назвою WEKA [11]. Результати порівняння наведені у таблиці 2.1 та на рисунку 2.9:

Таблиця 2.1 – Порівняння математичних методів

Назва методу	Mean squared error	Mean absolute error	Median absolute error
Дерева ухвалення рішень	0.006026	0.041364	0.002299
Опорні вектори	0.021987	0.091002	0.099897
ШНМ	0.004135	0.039671	0.022752
Випадкові ліси	0.006165	0.051759	0.027849
Лінійна регресія	0.004302	0.045543	0.028935
Поліноміальна регресія 2 порядку	0.019224	0.090228	0.056271
Поліноміальна регресія 3 порядку	0.0159199	0.071886	0.034277
Регресія з радіальним базисом	0.003113	0.044548	0.023069
Регресія з сигмоїдальним базисом	0.006033	0.044633	0.033857

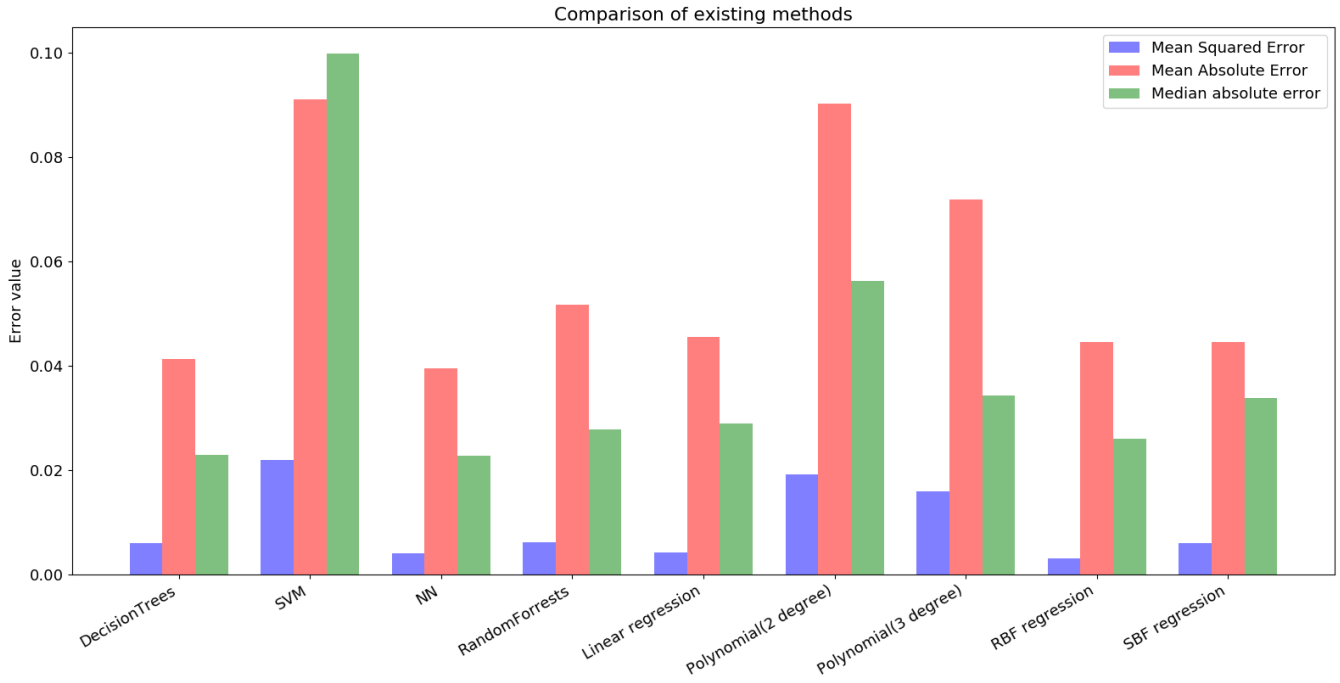


Рисунок 2.9 – Діаграма порівняння похибок, отриманих після використання існуючих методів

З рисунку 2.9 можна побачити, що штучні нейронні мережі повертають менші значення середньої квадратичної похибки, середньої абсолютної похибки та медіани абсолютної похибки. У тих методах, де значення MSE та MAE близькі до відповідних значень похибок ШНМ, різниця між медіаною абсолютної похибки та середньою абсолютною похибкою є меншою, ніж та сама різниця в ШНМ. Це значить, що штучна нейронна мережа прогнозує більше даних з меншою похибкою.

Отже, враховуючи критерії, сформульовані для поставленої задачі, доцільним є використання підходів, що базуються на побудові штучних нейронних мереж.

2.8 Огляд існуючих програмних рішень

2.8.1 WEKA

WEKA - бібліотека алгоритмів машинного навчання для вирішення завдань інтелектуального аналізу даних (інтелектуальний аналіз даних). Система дозволяє безпосередньо застосовувати алгоритми до вибірок даних, а також викликати алгоритми з програм на мові Java.

WEKA розшифровується як Waikato Environment for Knowledge Analysis - середовище для аналізу знань, розроблений в університеті Вайкато (Нова Зеландія).

Цілі проекту - створити сучасне середовище для розробки методів машинного навчання і застосування їх до реальних даних, зробити методи машинного навчання доступними для повсюдного застосування. Передбачається, що за допомогою даного середовища фахівець в прикладній області зможе використовувати методи машинного навчання для отримання корисних знань безпосередньо з даних, навіть дуже великого обсягу.

Користувачами WEKA є дослідники в області машинного навчання і прикладних наук. Вона також широко використовується в навчальних цілях.

WEKA містить засоби для попередньої обробки даних, класифікації, регресії, кластеризації, відбіру ознак, пошуку асоціативних правил і візуалізації. WEKA добре підходить для розробки нових підходів в машинному навчанні.

WEKA має користувальницький інтерфейс Explorer, але та ж функціональність доступна через компонентний інтерфейс Knowledge Flow і з командного рядка. Є окремий додаток Experimenter для порівняння можливостей алгоритмів машинного навчання на заданому наборі завдань.

Explorer має кілька панелей:

1) Панель попередньої обробки (preprocess panel) - дозволяє імпортувати дані з бази, CSV файлу і т. д., і застосовувати до них алгоритми фільтрації, наприклад,

переводити кількісні ознаки в дискретні, видаляти об'єкти і ознаки по заданому критерію.

2) Панель класифікації (Classify panel) - дозволяє застосовувати алгоритми класифікації та регресії (в WEKA вони не розрізняються і називаються classifiers) до вибірки даних, оцінювати передбачувану здатність алгоритмів, візуалізувати помилкові передбачення, ROC-криві, і сам алгоритм, якщо це можливо (зокрема, дерева ухвалення рішень).

3) Панель пошуку асоціативних правил (Associate panel) - переймається тим виявлення всіх значущих взаємозв'язків між ознаками, що надаються у вихідних даних.

4) Панель кластеризації (Cluster panel) - Дає доступ до алгоритму k-середніх, EM-алгоритму для суміші гауссіанів і іншим.

5) Панель відбору ознак (Select attributes panel) - дає доступ до методів відбору ознак.

6) Панель візуалізації (Visualize panel) - будує матрицю графіків розкиду (scatter plot matrix), дозволяє вибирати і збільшувати графіки, і т. д.

2.8.2 IBM SPSS Statistics

Продукт IBM SPSS Statistics (раніше відомий просто як «SPSS») надає широкі можливості для аналізу даних. Інтуїтивно зрозумілий інтерфейс програмного забезпечення включає в себе всі функції управління даними, статистичні процедури і засоби створення звітів для проведення аналізу будь-якого ступеня складності.

IBM SPSS Statistics і продукти IBM SPSS Amos, Sample Power, VizDesigner, Data Collection, Collaboration and Deployment Services утворюють модульний, повністю інтегрований програмний комплекс [12, 13]. Інтеграція продуктів IBM SPSS в єдину

лінійку дозволяє впевнено працювати, не стикаючись з проблемами переходу від одного програмного продукту до іншого.

Великий вибір процедур в базовому модулі IBM SPSS Statistics дає широкі можливості аналізу даних різних типів. Вбудовані додаткові модулі розширюють аналітичні можливості настільки, наскільки це необхідно.

Фахівці в галузі аналізу даних цінують логічність, продуманість і взаємопов'язаність компонентів програмного забезпечення IBM SPSS. Початківців особливо вразить інтуїтивно зрозумілий інтерфейс, повнота довідкової системи російською мовою і якість технічної підтримки.

Редакції IBM SPSS Statistics:

1) IBM SPSS Statistics Base (базовий модуль) - ключовий елемент пакета SPSS Statistics, що забезпечує доступ до даних, управління даними, підготовку даних до аналізу, аналіз даних і створення звітів. Додаткові модулі вбудовуються в SPSS Statistics Base і дозволяють розширювати аналітичні можливості програмного забезпечення.

2) IBM SPSS Statistics Standard - дозволяє здійснювати базові аналітичні операції для вирішення широкого спектра господарських і дослідницьких проблем. Включає модулі IBM SPSS Statistics Base, IBM SPSS Custom Tables, IBM SPSS Regression і IBM SPSS Advandced Statistics.

3) IBM SPSS Statistics Professional - відкриває додаткові можливості, пов'язані із забезпеченням якості даних, а також автоматизації функцій статистики та прогнозування. До складу IBM SPSS Statistics Professional, крім модулів Standard, входять ще й IBM SPSS Data Preparation, IBM SPSS Categories, IBM SPSS Decision Trees, IBM SPSS Forecasting і IBM SPSS Missing Values.

4) IBM SPSS Statistics Premium - найповніший набір аналітичних можливостей: система моделювання на основі структурних рівнянь (SEM), докладна оцінка і перевірка вибіркових даних, процедури прямого маркетингу.

Цей комплект забезпечує організацію найсучаснішими методами аналізу і обробки даних. Крім модулів Professional, в комплект включені IBM SPSS Bootstrapping, IBM SPSS Complex Samples, IBM SPSS Conjoint, IBM SPSS Direct Marketing, IBM SPSS Exact Tests, IBM SPSS Neural Networks, IBM SPSS Sample power і IBM SPSS Visualization Designer.

2.8.3 Порівняння програмних рішень для розв’язання задачі

Результати аналізу розглянутих програмних рішень наведено в таблиці 2.2.

Таблиця 2.2 – Порівняння комерційних програмних рішень

Назва	Наявність графічного інтерфейсу	Вартість	Використані модулі, функції	MAE	MSE	Точність
IBM SPSS Statistics	+	99$ / місяць	IBM SPSS NeuralNetwork	0.023	0.0025	90%
WEKA	+	Free	MLP	0.041	0.0042	86%

2.9 Висновки до розділу

На основі порівняльного аналізу (таблиця 2.1, рисунок 2.9) можна зробити висновок, що підхід із застосуванням штучних нейронних мереж в середньому краще задовольняє критерії точності, що може бути зумовлено ліпшим математичним описанням поставленої задачі.

Ефективність обраного методу можна поліпшити за рахунок зміни структури нейронної мережі та кращої попередньої обробки вхідних даних.

Як можна побачити з таблиці 2.2, кожне з існуючих рішень має ряд власних переваг та недоліків, але жодне з них не вирішує поставлену задачу напряму, а є лише набором модулів та функцій, що можуть сприяти рішенню задачі.

Розроблене програмне забезпечення націлене саме на створення математичної моделі формування дози внутрішнього опромінення населення і зберігає баланс у співвідношенні вартості та точності розрахунків.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура штучних нейронних мереж

У даній роботі мова буде йти про багатошарові нейронні мережі (multilayer perceptron, MLP) зі зворотними зв’язками.

У загальному випадку, нейронна мережа складається з вхідного шару нейронів, який отримує інформацію, вихідного шару, що повертає результат, та прихованих шарів, що розташовані посередині (рис. 3.1).

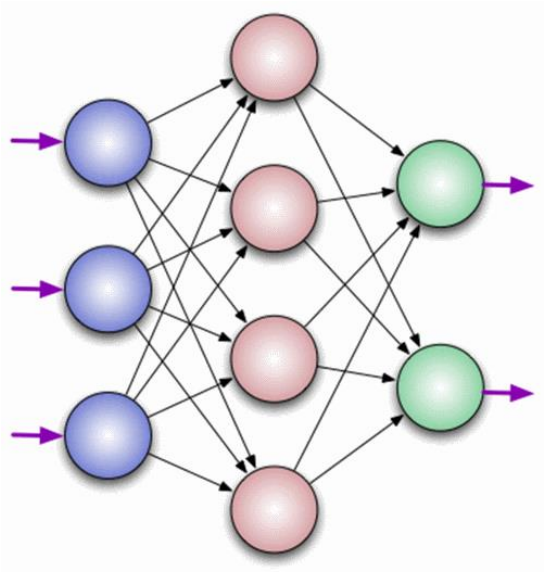


Рисунок 3.1 – Штучна нейронна мережа

Вхідні сигнали x_m, зважені ваговими коефіцієнтами ω_m, додаються, результат передається до функції активації, про яку детально написано в розділі 3.3. Функція активації і повертає кінцевий результат(рис. 3.2).

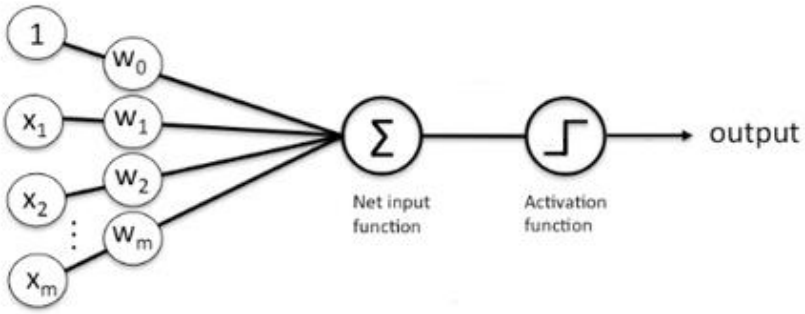


Рисунок 3.2 – Представлення нейрону

Кожен з шарів може містити довільну кількість нейронів, що залежить від поставленої задачі. Зазвичай, кількість нейронів у вхідному шарі дорівнює кількості одиниць даних (наприклад, факторів або характеристик), які подаються на вхід [14].

Прихованих шарів може бути декілька, а може і зовсім не бути. Їх мета – перетворення даних, які вони отримують, у той вигляд, в якому їх зможе використати наступний шар нейронів. Кількість прихованих шарів та нейронів у них залежать як від обмежень по часу, так і обчислювальної здатності ЕОМ. Замало нейронів може призвести до того, що мережа не матиме змоги навчатись(underfitting), забагато – стане причиною збільшення часу навчання і, можливо, призведе до перенавчання (ovefitting) мережі [15, 16]. Перенавчання – це такий стан, коли нейронна мережа «запам’ятовує» навчальну вибірку і показує задовільні результати лише на ній, а на інших вибірках помилка є занадто великою. Коли мова заходить за велику кількість прихованих шарів нейронів, то має місце термін «глибинне навчання» (deep learning), а нейронні мережі, що використовуються при цьому – глибинні нейронні мережі (deep neural networks, DNN).

У вихідному шарі кількість нейронів дорівнює кількості значень, які потрібно отримати в ході обчислень. Саме вихідний шар нейронів повертає той результат, який цікавить користувача ШНМ.

Зв’язок між двома нейронами називається синапсом. Нейрон кожного шару, окрім вихідного, зв’язується з кожним нейроном наступного шару, як показано на рисунку

3.3. Синапс має лише один параметр – вагу. Завдяки синапсу вхідна інформація змінюється, проходячи через нього.

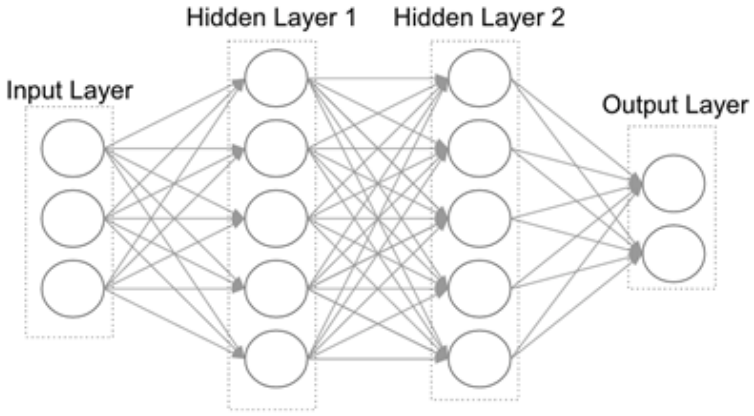


Рисунок 3.3 – ШНМ з двома прихованими шарами нейронів

На рисунку 3.3 представлено нейронну мережу з трьома нейронами у вхідному шарі, двома прихованими шарами, кожен з яких містить п’ять нейронів, а також вихідний шар, до якого входять два нейрони.

3.2 Формування вектору вихідних даних

Прогнозування величин та знаходження залежностей між факторами відбувається значно ефективніше, коли ШНМ працює із числовими даними. Варто помітити, що нейрони оперують числами в діапазоні [-1; 1]. З цього можна зробити висновок, що дані, які не попадають у цей проміжок потрібно нормалізувати.

Існує багато підходів до нормалізації даних, серед яких ефективними є Z-score normalization, Label Encoding, One-Hot Encoding [17].

3.2.1 Z-score normalization

Цей метод нормалізації використовується для числових даних. За формулою (3.1) дані приводяться до потрібного діапазону значень.

$$Z = \frac{x - \mu}{\sigma}, \tag{3.1}$$

де z - нормалізоване значення;

x - значення до нормалізації;

μ - середнє значення;

σ - відхилення від середнього значення.

3.2.2 Label Encoding

Даний метод слід використовувати при кодуванні категоріальних змінних, таких як стать, статус, освіта тощо. Label encoding дозволяє нормалізувати дані у діапазоні $[0; n-1]$, де n - кількість категоріальних класів. Отже, змінну «стать», що має два значення, ми можемо записати як 0 та 1, а якщо наша характеристика «професія» має чотири різних значень у таблиці з даними, то вона буде закодована значеннями 0, 1, 2, 3.

3.2.3 One-Hot Encoding

Цей метод нормалізації перетворює кожне значення категоріальної характеристики з n значеннями на бінарне значення, де буде $n-1$ нулів та одна одиниця. Наприклад, ми маємо характеристику «колір», яка приймає 3 значення: білий, червоний, зелений. Значення будуть закодовані як [1,0,0], [0,1,0], [0,0,1] відповідно.

3.3 Пряме поширення сигналів у ШНМ

Наша мережа штучних нейронів буде навчатися з учителем. Це значить, що наші навчальні сети міститимуть результати, на які ШНМ буде опиратись в ході навчання. Вихідна інформація, при навчанні, перемножується на вагу відповідного синапса, через який проходить, а потім додається у нейроні прихованого шару. Після цього, отримане число підставляється в якості аргументу в функцію активації. Функція активації, у свою чергу, повертає результат, який порівнюється з результатом, що був у навчальному сеті. Під час ініціалізації нейронної мережі ваги на кожному з синапсів виставляються випадковим чином.

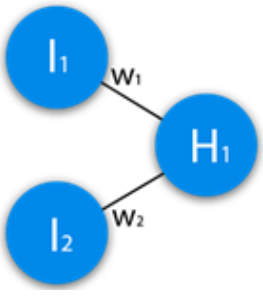


Рисунок 3.4 – Синапси між нейронами вхідного шару I_k та прихованого H_k

$$H_{1input} = (l_1 * \omega_1) + (l_2 * \omega_2), \quad (3.2)$$

де I_k - вхідні нейрони;

H – нейрон прихованого шару;

ω_k - ваги синапсів.

Функція активації – функція, що обраховує вихідний сигнал штучного нейрону, а також вона потрібна для нормалізації даних. Якщо ми в результаті отримуємо велике число, то, пройшовши через функцію активації, воно буде знаходитись у потрібному діапазоні. Функцій активації досить багато, тому розглянемо лише деякі з них: сигмоїда (логістична) (рис. 3.5) і гіперболічний тангенс (рис. 3.6). Головна їх відмінність - це діапазон значень [18].

Сигмоїда

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3.3)$$

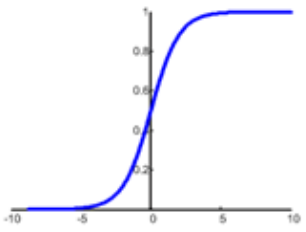


Рисунок 3.5 – Графік сигмоїди

Ця функція має діапазон значень [0;1]. Якщо у поставленій задачі присутні від’ємні числа, то потрібно обирати функцію, що охоплює й їх.

Гіперболічний тангенс

$$f(x)=\frac{e^{2x}-1}{e^{2x}+1} \tag{3.4}$$

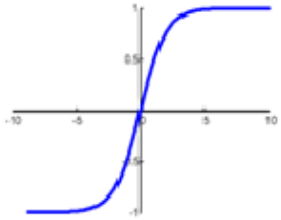


Рисунок 3.6 – Графік гіперболічного тангенса

Дану функцію раціонально використовувати, коли значення можуть бути як від'ємними, так і додатними, бо діапазон функції [-1;1].

Як вже було сказано, нейронна мережа навчається за допомогою тренувальних сетів. Навчання на тренувальних сетах відбувається ітеративно. Коли всі тренувальні сети пройдені, то закінчується одна навчальна епоха. Максимальна кількість епох задається вручну.

Зазвичай, вводять коефіцієнт відсіву – коефіцієнт, що не дозволяє ШНМ запам'ятовувати навчальні сети. Значення коефіцієнта знаходяться в інтервалі від 0 до 1. Це вірогідність, з якою значення буде використане при певній навчальній епосі. Таким чином, навчальні сети не будуть повторюватись, бо будуть щоразу містити в собі різні послідовності підмножин даних [19, 20]. Це призведе до того, шо нейронна мережа змушена буде шукати саме закономірності, а не запам'ятовувати одні й ті самі послідовності. Ефективність нейронних мереж, що використовують коефіцієнт відсіву значно більша [21].

Ще однією стандартною технікою для покращення результуючої точності моделі є перехресна перевірка. Для використання цієї техніки, потрібно розбити дані

для навчання на n частин. Перехресна перевірка у такому випадку буде відбуватись у n ітерацій. Спочатку, перша частина буде залишена для тестування отриманих результатів, а $[2;n]$ - для навчання. На другій ітерації, друга частина буде використана для тестування, а 1 та $[3;n]$ - для навчання. Таким чином, на кожній ітерації одна частина залишається для перевірки зпрогнозованих значень, а $n-1$ - для навчання. У результаті, на кожній ітерації буде підрахована точність моделі, а обрана буде та модель, що має найкращий показник точності. Ця модель і буде використана для перевірки ШНМ на тестовому сеті даних.

3.4 Обрахунок похибки

Похибка – процентна величина, що показує розбіжність між отриманим та бажаним результатами. Похибка формується кожну епоху і має зменшуватися з кожним разом. Існує багато методів обрахування похибки. Найпоширеніші з них такі: Mean Squared Error (MSE), Mean Absolute Error (MAE), Root MSE і Arctan [22, 23]. Їх вибір не має таких обмежень, як вибір функції активації.

1) MSE

$$E_n = \frac{(i_1 - a_1)^2 + (i_2 - a_2)^2 + ... + (i_n - a_n)^2}{n}, \tag{3.5}$$

де i_n - вектор шуканих результатів;

a_n - вектор отриманих від перцептрона результатів;

E_n - вектор похибок;

n - розмірність векторів.

2) MAE

$$E_n = \frac{(i_1 - a_1) + (i_2 - a_2) + ... + (i_n - a_n)}{n} \tag{3.6}$$

3) Root MSE

$$E_n = \sqrt{\frac{(i_1 - a_1)^2 + (i_2 - a_2)^2 + ... + (i_n - a_n)^2}{n}} \tag{3.7}$$

4) Arctan

$$E_n = \frac{\arctan^2(i_1 - a_1) + \arctan^2(i_2 - a_2) + ... + \arctan^2(i_n - a_n)}{n} \tag{3.8}$$

Значення похибки у Arctan майже завжди буде найбільшим, бо він працює за принципом: чим більша різниця, тим більша похибка. У Root MSE – найменшим. MSE та MAE зберігають баланс, тому і користується найбільшою популярністю серед цих методів.

3.5 Алгоритм навчання нейронної мережі

Коли завершується епоха, тоді настає час навчання мережі. Час, коли її вагові коефіцієнти корегуються. На цьому етапі застосовують наступні правила [24]:

а) Правила Хеба

1) Якщо нейронна мережа повертає правильний результат, то вагові коефіцієнти не змінюються.
2) Якщо нейронна мережа помилилась і повернула неправильну відповідь, то вагові коефіцієнти зменшуються.
3) Якщо нейронна мережа помилилась і відкинула правильну відповідь, то вагові коефіцієнти збільшуються.

б) Дельта-правило – метод навчання перцептрона за принципом градієнтного спуска. Це правило з'явилось на основі правил Хеба. Його подальший розвиток призвів до створення метода зворотного поширення (МЗП) похибки.

Нехай вектор $X = x_1, x_2, ..., x_m$ - вектор вхідних сигналів,а вектор $D = d_1, d_2, ..., d_n$ - вектор сигналів, які має отримати перцептрон після обробки вектора вхідних сигналів. Вхідні сигнали, поступивши на вхід перцептрона, були зважені та додані, пройшли через функції активації і в результаті був отриманий вектор $Y = y_1, y_2, ..., y_n$. Його розмірність співпадає з вектором вихідних сигналів. Тоді, вектор похибок E записується як різність між очікуваним та реальним значенням вихідного сигналу:

$$E = D - Y \tag{3.9}$$

А формула для корегування j-ї ваги i-го нейрона буде виглядати наступним чином [23]:

$$w_j(t+1) = w_j(t) + e_i x_j, \tag{3.10}$$

де $j = [1; m]$ - номер сигналу;

$i = [1; n]$ - номер нейрона;

t - номер ітерації навчання.

Таким чином вага з кожною ітерацією зменшується пропорційно до похибки. Часто вводять коефіцієнт швидкості навчання, на який множиться похибка. Після цього формула (3.10) набуває наступного вигляду:

$$w_j(t+1) = w_j(t) + \eta e_i x_j, \tag{3.11}$$

де η - коефіцієнт швидкості навчання.

Коефіцієнт навчання також є гіперпараметром, що регулюється вручну. Із занадто великим коефіцієнтом навчання мережа може пропустити точку глобального мінімуму і ніколи в неї не потрапить. Саме тому прийнято ставити маленький коефіцієнт швидкості навчання, що забезпечить попадання в точку глобального мінімуму, хоч і збільшить час навчання [25].

На етапі навчання ваги синапсів зазнають постійних змін і це є основою досягнення максимально точних результатів. Одним з найбільш поширених методів навчання є метод зворотного поширення похибки (backpropagation). Ідея цього методу полягає в поширенні сигналів похибки від вихідного шару до вхідного шару нейронів, тобто в зворотному напрямку, що дозволяє обраховувати та послідовно змінювати ваги на кожному з синапсів, що поєднують нейрони скритих шарів. Цей метод є модифікацією методу градієнтного спуску [26, 27].

Отже, метод зворотного поширення похибки є дуже ефективним при розрахунках похідної похибки $\frac{\partial E_n}{\partial \omega_{ji}}$. Спочатку потрібно надати ШНМ тренувальні дані x_n, як було описано в розділі 3.3, та зберігати усі значення a_j, що передаються до функцій активації в якості аргументу. Ми можемо написати так:

$$\frac{\partial E_n}{\partial \omega_{j_i}} = \frac{\partial}{\partial \omega_{j_i}} E_n(a_{j_1}, a_{j_2}, ..., a_{j_m}), \tag{3.12}$$

де $\{j_i\}$ - індекси усіх нейронів, що знаходяться в j-му шарі.

Перетворимо отриману формулу (3.12):

$$\frac{\partial E_n}{\partial \omega_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial \omega_{ji}} + \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial \omega_{ji}}, \tag{3.13}$$

де $\sum_k$ - сума, що обраховується по всім нейронам відповідного шару.

Так як a_k не залежать від ω_{j_i}, то сума буде рівна 0. Тоді маємо:

$$\frac{\partial E_n}{\partial \omega_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial \omega_{ji}} \tag{3.14}$$

Введемо похибку $\delta_j \equiv \frac{\partial E_n}{\partial a_j}$. Тоді формула (3.14) набуває наступного вигляду:

$$\frac{\partial E_n}{\partial \omega_{ji}} = \delta_j \frac{\partial a_j}{\partial \omega_{ji}} \tag{3.15}$$

Розглянемо іншу частину формули (3.14):

$$\frac{\partial a_j}{\partial \omega_{ji}} = \frac{\partial}{\partial \omega_{ji}}\sum_k \omega_{jk} x_k = x_i \tag{3.16}$$

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \frac{\partial a_k}{\partial a_j}, \tag{3.17}$$

де $\sum_k$ - сума всіх нейронів k-го шару, що йдуть після j-го шару.

$$\delta_j = h'(a_j)\sum_k \omega_{kj}\delta_k\,, \tag{3.18}$$

де h - функція активації.

Аналізуючи формулу (3.18), стає зрозумілим, що якщо одна з $h'(a_j)$ є близькою до нуля, тоді весь добуток буде малим. Як наслідок, рух до оптимуму буде дуже повільним. Ця проблема має назву «затухаючий градієнт».

Нашою метою є збільшення похідних, щоб збільшився шаг у напрямку, який задає градієнт. Це легко вирішується правильним підбором функції активації. Розглянемо сигмоїду (3.3). Вона має наступну похідну:

$$\sigma' = \sigma(1-\sigma) = \sigma - \sigma^2, \tag{3.19}$$

де σ - функція сигмоїда.

Якщо дані не нормалізовані, то великі значення даних (такі як рік народження) призведуть до того, що сигмоїда буде мати значення, близькі до 1. Тоді $(1-\sigma)$ буде близьким до 0, а отже шаг буде маленьким і будемо довго йти до глобального мінімуму. Якщо ж дані нормалізовані, тоді значення x буде близьким до 0, а отже значення σ буде близьке до 0.5 (рис. 3.5). Підставляючи це значення в формулу (3.19) отримаємо $\sigma' \approx 0.25$, що також є близьким до 0 і сповільнить шлях до глобального мінімуму.

Розглянемо тепер гіперболічний тангенс (3.4). Його похідна має наступний вигляд

$$\tanh'(x) = 1 - \tanh^2(x) \tag{3.20}$$

В області $\tanh(x) = 0$ значення похідної буде дорівнювати 1. Так як наші дані після нормалізації будуть знаходитись в області 0, то цей варіант є ідеальним, адже дозволить рухатись до області глобального мінімуму швидше.

Можна зробити висновок, що обидві функції потребують нормалізації, але сигмоїда буде давати менше значення градієнта через особливості, пов'язані з самою функцією.

3.6 Обрана архітектура нейронної мережі

Установлення параметрів системи відбувається завдяки навчанню з учителем. Вхідний шар має містити 10 нейронів, які будуть приймати на вхід значення факторів по кожному рядку даних (рис. 3.7). Прихований шар буде містити 12 нейронів. Таке значення було підібране в ході регулювання параметрів нейронної мережі. Наступний прихований шар має коефіцієнт відсіву, рівний 0.5 для утворення неповторних підмножин множини даних. Вихідний шар містить 1 нейрон, бо результатом є лише значення внутрішньої дози опромінення. Навчання буде відбуватися протягом 16 епох, а дані будуть передаватися пакетами, що містять 8 рядків даних. Такі параметри теж були обрані в ході регулювання структури моделі.

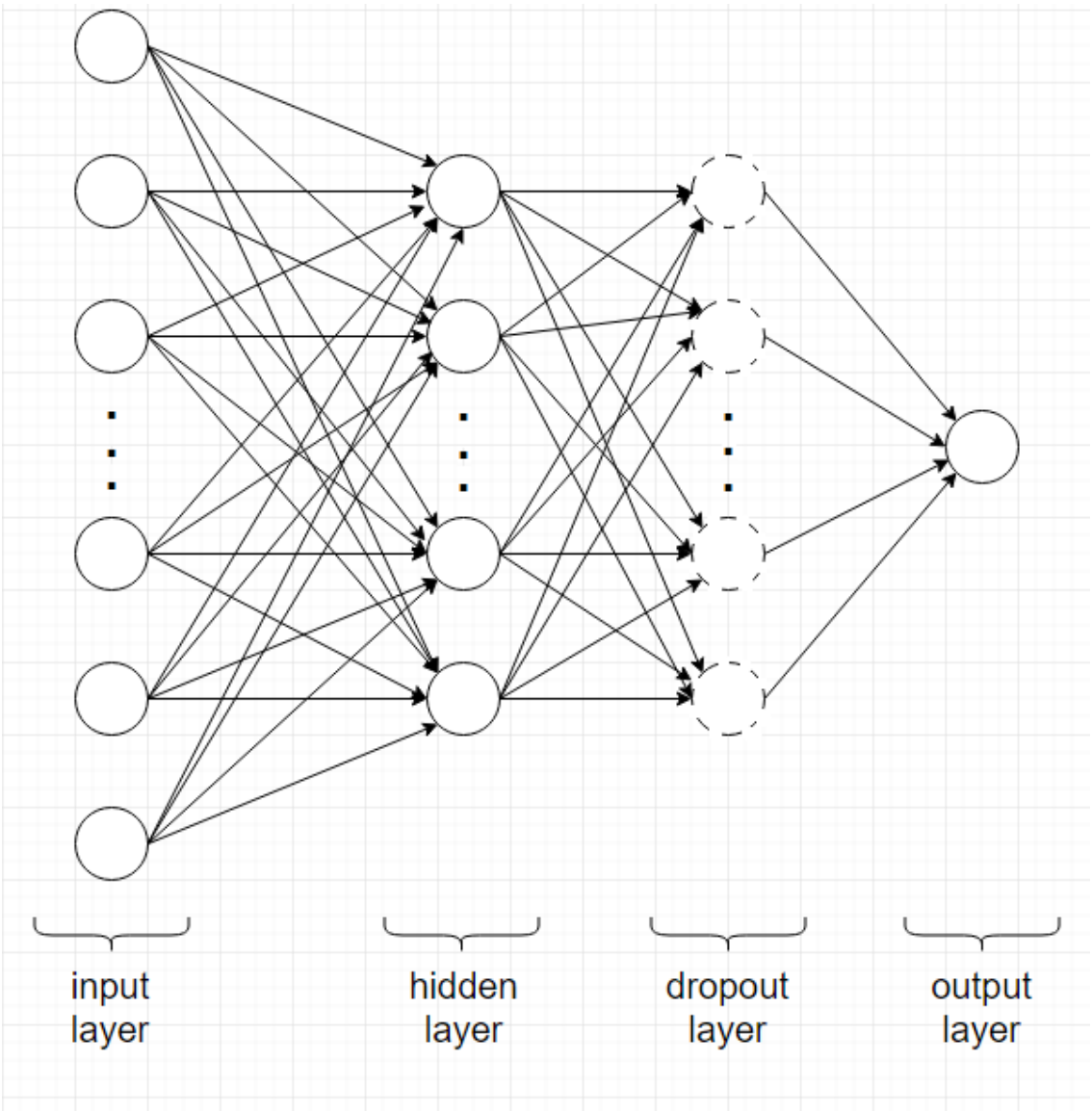


Рисунок 3.7 – Обрана архітектура нейронної мережі

Вихідні дані ШНМ зазнають обробки: значення категоріальних факторів кодуються за допомогою OneHotEncoding в числові значення, що знаходяться в потрібному діапазоні; значення числових факторів нормалізуються за допомогою Z-score.

У ході перехресної перевірки дані було розбито на 10 частин. У якості функції активації обрано функцію гіперболічного тангенса. За критерій зупинки навчання взято погіршення ефективності прогнозування на тестовому наборі даних. Обрахована модель містить мінімізовану похибку, що забезпечується методом

зворотного поширення похибки. Сама похибка обраховується за формулами MSE, MAE.

3.7 Висновки до розділу

У цьому розділі розроблено математичне забезпечення формування дози внутрішнього опромінення населення внаслідок аварії на ЧАЕС. У реалізації алгоритму було зосереджено увагу на виконанні усіх поставлених у меті роботи завдань.

У результаті використання даного математичного забезпечення користувач отримує модель формування дози на основі відомих статистичних даних, зібраних інститутом агроекології та природокористування НААН України.

Розроблене математичне забезпечення, пристосоване до предметної області, надасть можливість працівникам відділу радіоекології зменшити затрати часу та фінансів, потрібні для вимірювання дози внутрішнього опромінення населення.

4 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Опис програми

Для створення математичної моделі формування дози внутрішнього опромінення населення внаслідок аварії на ЧАЕС за поточними даними та прогнозування результатів за даними, що будуть зібрані в майбутньому, було розроблено відповідне програмне забезпечення. Воно дозволяє користувачу обирати: провести навчання нейронної мережі та зберегти отримані ваги, або спрогнозувати значення дози внутрішнього опромінення з використанням вже існуючого файлу з вагами для ШНМ, а також виводити на екран відповідні графіки.

Програма розроблена з використанням мови програмування Python та оболонки PyQt. Python з його величезною кількістю бібліотек для обробки та аналізу даних, на мою думку, є одним з найкращих інструментів для роботи з файлами різних форматів та створення штучної нейронної мережі, а оболонка PyQt зв’язує його з Qt – інструментарієм для створення графічного інтерфейсу користувача.

Програма має інтуїтивно зрозумілий та простий інтерфейс (рис. 4.1). Він дозволяє користувачу виставляти необхідні параметри нейронної мережі та обирати подальший хід роботи програми: навчання або прогнозування. Після чого користувач має можливість переглянути побудовані графіки для отриманої похибки після кожної епохи та візуально порівняти спрогнозовані значення з тими значеннями, які мали бути отримані.

Також користувач має можливість вносити свої корективи до зберігаємих даних, а саме редагувати їх та видаляти.

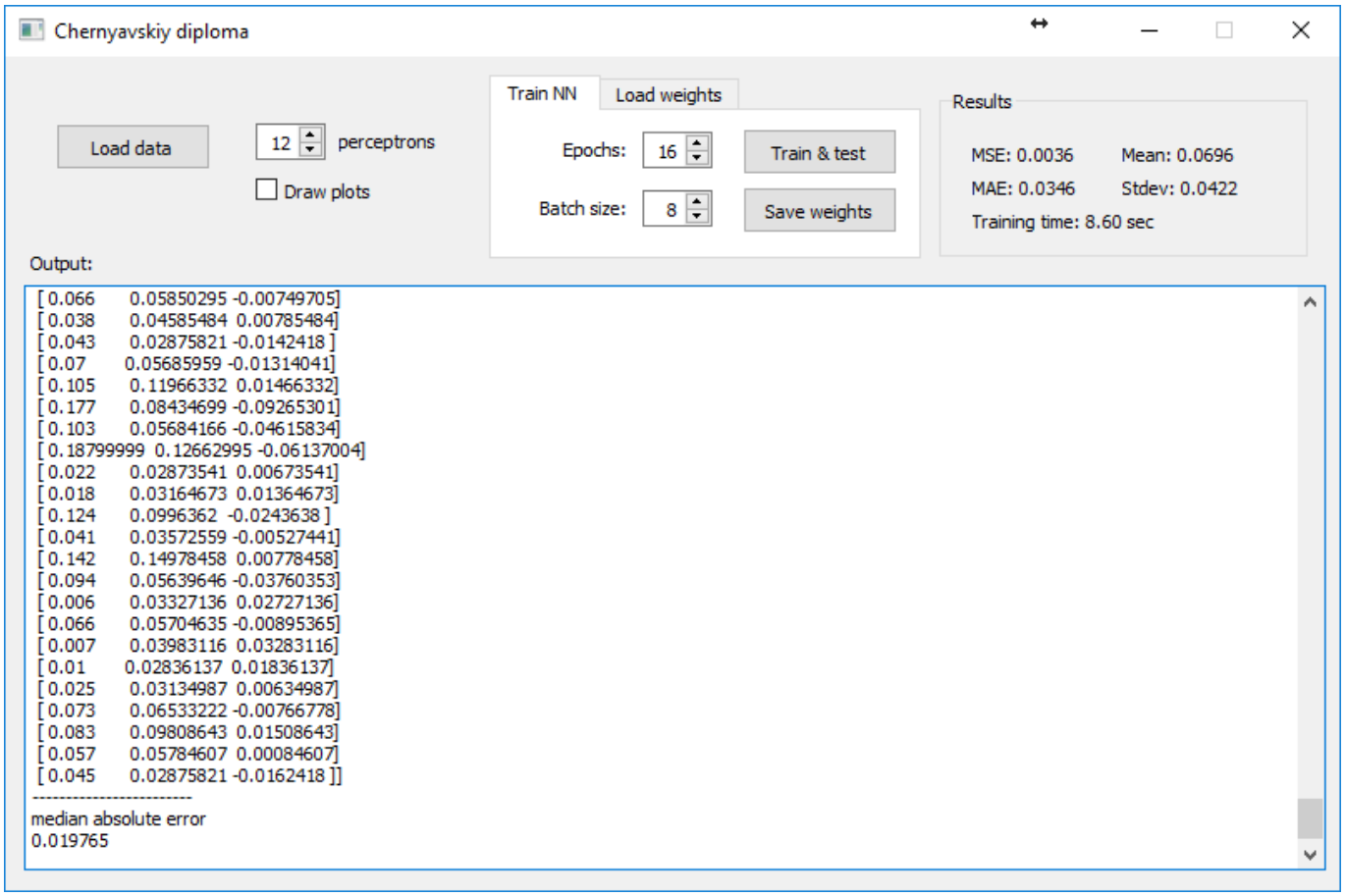


Рисунок 4.1 - Інтерфейс програми

4.2 Формат даних

4.2.1 Формат вихідних даних для навчання ШНМ

Вихідними даними для ініціалізації системи є всі рядки, починаючи з другого, файлу з даними, який збережено в форматі CSV. У першому рядку знаходиться шапка з назвами факторів (DOS_CS_137, BIRTH_Y, PROFESSION, YEAR_MEASSUR, Month_MEASUR, Population, Forest area in 3km, Specific forest area in 3km, Education leve, Cs-137 soil contamination, Contamination zone, Soil type 1, Soil type 2, Soil type 3,

Distance to the nearest district center, Distance to forest), а в інших рядках містяться значення факторів. Файл для навчання складається з 2992 рядків

Приклад даних для навчання ШНМ:

0.182, 1999, 6, 2005, 11, 109, 53, 0.137481, 1, 83, 4, 1, 4, 35, 40, 0.52

Після завантаження файлу в програму, числові дані нормалізуються, категоріальні дані кодуються в числові, рік та місяць вимірювання дози переводяться в кількість місяців від аварії на ЧАЕС до часу, коли доза була виміряна. Нові стовпці додаються до таблиці, а старі – видаляються.

Після завантаження та обробки даних користувач власноруч обирає структуру штучної нейронної мережі (кількість нейронів у прихованому шарі, кількість навчальних епох, кількість даних, які будуть пакетно передаватись в кожній епосі).

Важливість параметру “Batch size” важко переоцінити. Цей параметр відповідає за те, скільки даних буде міститись у одному пакеті. При значенні цього параметру рівному 1, ми маємо ситуацію, коли значення одного рядка рухаються по ШНМ до кінця, а потім знов до початку, підраховуючи градієнт. Коли значення даного параметру дорівнює, наприклад, 5, то кожен з п’яти рядків рухається так само вперед та назад, у кожному випадку обраховується градієнт, але в результаті значення градієнта буде дорівнювати середньому значенню з цих п’яти. Таким чином напрямок руху не буде стрибати в різних напрямках, а буде більш гладким. Ідеальним прикладом можна вважати ситуацію, коли досліджуване значення в якомусь рядку сильно відрязняється від інших. У такому випадку ШНМ зрозуміє, що їй потрібно значно змінювати значення вагів, а це не буде правильним рішенням, адже рядок з істотно відмінними значеннями може з’являтись в даних не систематично, а досить рідко.

Далі дані розбиваються на навчальні та тестувальні у співвідношенні 9:1. Це значить, що нейронна мережа буде навчатися на 90% наданих їй даних, а на інших 10% можна перевірити, наскільки точні результати прогнозує натренована ШНМ.

Завдяки такому розбиттю даних створюється ситуація, коли точність мережі тестується на даних, які вона ще не бачила, а отже результати є більш справедливими.

4.2.2 Формат вихідних даних для передбачення

Вихідними даними для прогнозування є файл формату CSV, що містить дані, але, на відміну від формату даних, описаного в 4.2.1, перший стовпець заповнений нулями, адже в цьому випадку не відома доза внутрішнього опромінення, а лише відомі значення факторів, які впливають на формування дози. Користувач має можливість вносити нові дані до файлу, але не може додавати нові типи категоріальних даних (наприклад, додати новий тип професії до стовпця PROFESSION).

Крім цього, вхідними даними для прогнозування також є файл з вагами формату H5. Файл з вагами може бути отриманий в результаті навчання. Варто уваги, що структура нейронної мережі, що була використана для створення файлу з вагами і структура нейронної мережі, яка використовує цей файл мають бути однаковими, адже ми не можемо використовувати ваги для синапсів між 5 нейронами прихованого шару в ШНМ, що містить у прихованому шарі більшу або меншу кількість нейронів.

4.2.3 Формат результуючих даних

У результаті роботи програми ми отримуємо файл з вагами формату H5, як було описано в розділі 4.2.2, а також графіки з середньоквадратичною похибкою та

середньою абсолютною похибкою (після навчання мережі) або графіки зі спрогнозованими значеннями для дози внутрішнього опромінення (після прогнозування). Графіки можуть бути збережені у форматах PNG, JPG, PDF, SVG тощо.

4.3 Тестування розробленого ПЗ

4.3.1 Результати тестування структури нейронної мережі

Структура нейронної мережі була обрана за допомогою методу перебору. Оптимальними виявились такі параметри:

- Кількість нейронів у прихованому шарі: 12;
- Кількість навчальних епох: 16;
- Кількість даних у кожному пакеті: 8.

Швидкість навчання було встановлено у значення 0.001. Параметр відсіву даних: 0.5. З такими параметрами нейронна мережа повернула результати, що зображені на рис. 4.1:

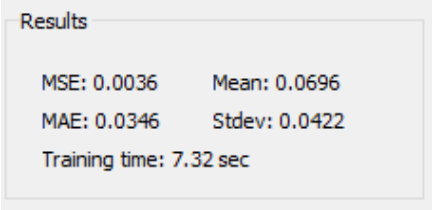


Рисунок 4.1 – Результати навчання ШНМ

З рисунку 4.1 можна побачити, що значення середньоквадратичної похибки дорівнює 0.0036, середньої абсолютної похибки – 0.0346, середнє значення – 0.0696,

а середньоквадратичне відхилення (корень із дисперсії) – 0.0422. Ці значення відповідають графікам, наведеним на рисунках 4.2 - 4.3:

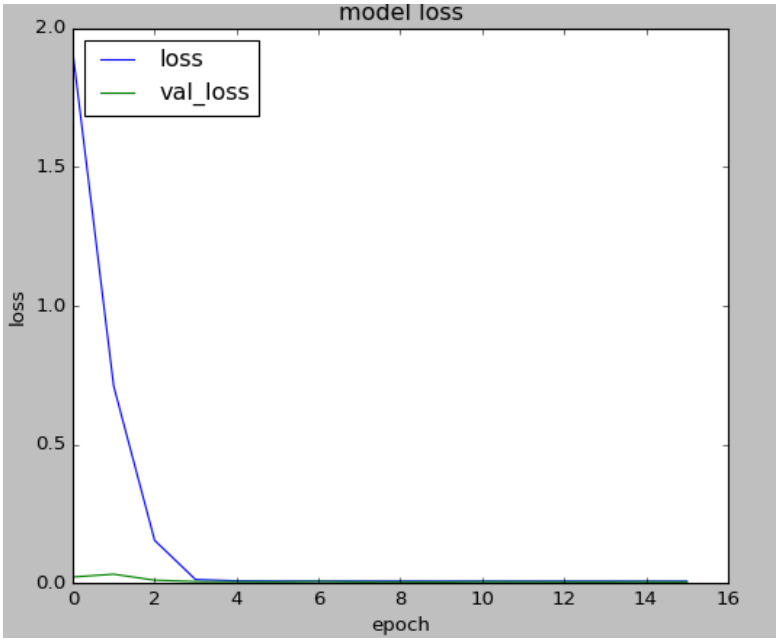


Рисунок 4.2 – Графік середньоквадратичної похибки для кожної епохи

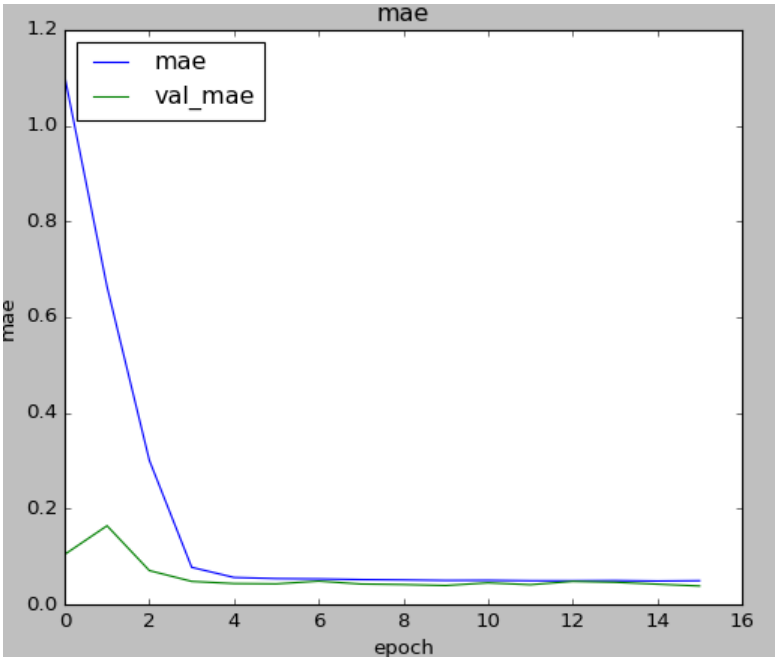


Рисунок 4.3 – Графік середньої абсолютної похибки для кожної епохи

Зміна параметрів (кількості нейронів у прихованому шарі або кількості навчальних епох) призводить до збільшення похибок, як показано на рисунках 4.4-4.6:

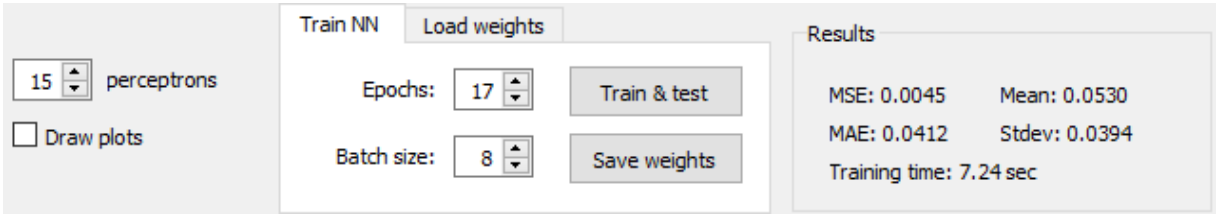


Рисунок 4.4 – Результати роботи ШНМ з 15 нейронами у прихованому шарі та 17 навчальними епохами

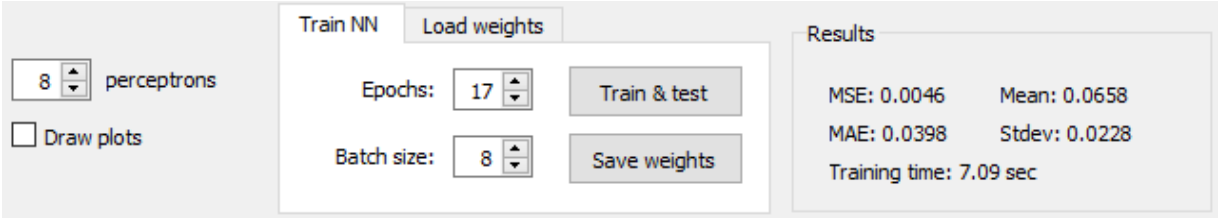


Рисунок 4.5 – Результати роботи ШНМ з 8 нейронами у прихованому шарі та 17 навчальними епохами

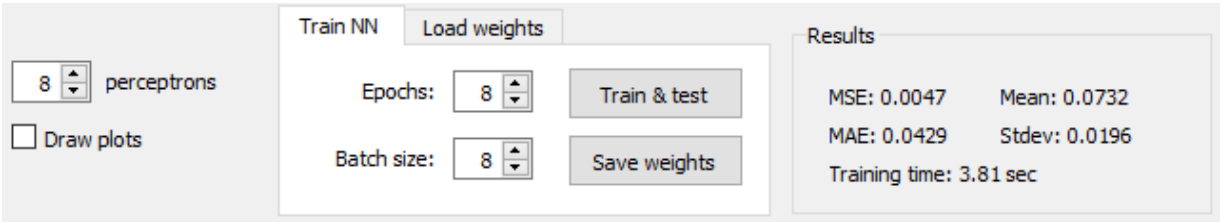


Рисунок 4.6 – Результати роботи ШНМ з 8 нейронами у прихованому шарі та 8 навчальними епохами

4.3.2 Результати тестування прогнозування дози внутрішнього опромінення

Тестування розробленого програмного забезпечення було проведено на основі двох файлів, що містять набори зі 100 (рис. 4.7) та 200 (рис. 4.8) даних, які не використовувалися при навчанні нейронної мережі.

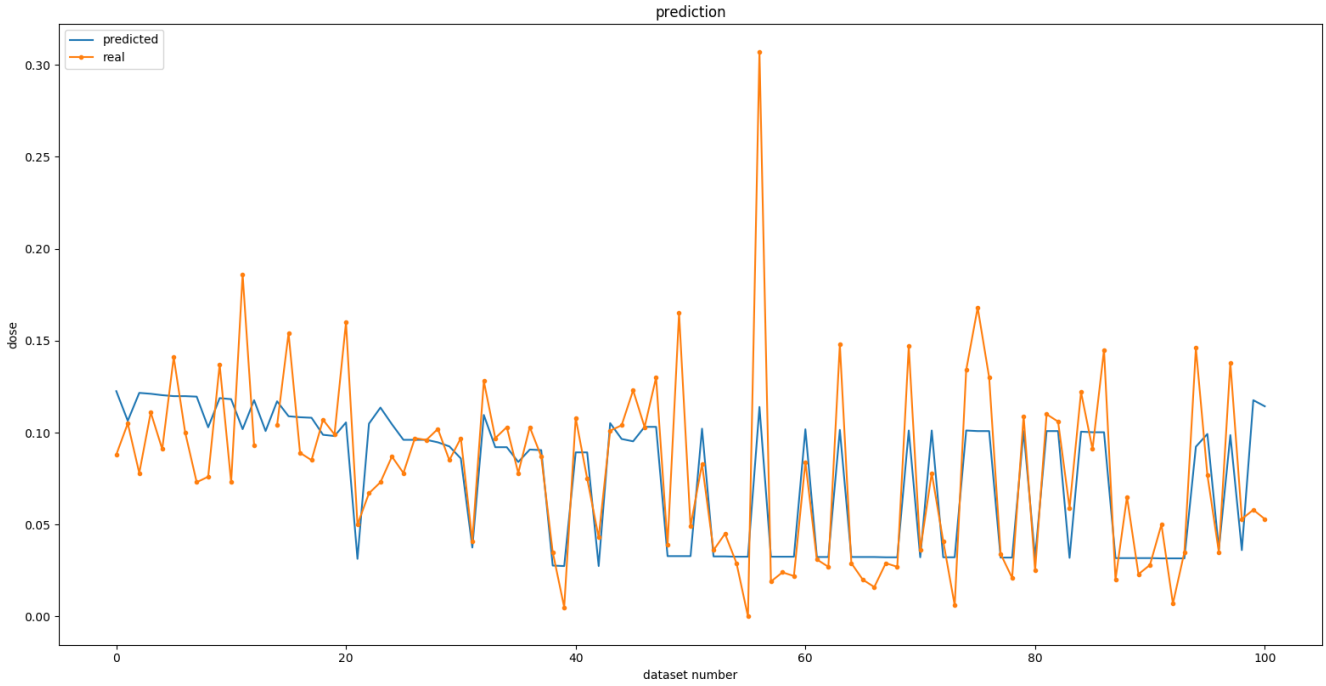


Рисунок 4.7 – Графік спрогнозованих доз внутрішнього опромінення у порівнянні зі справжніми значеннями для набору зі 100 даних

На рисунку 4.7 представлений набір зі 100 даних, які попередньо відсортовані за віком таким чином, що на осі абсцис ближче до 0 знаходяться люди, які мають більший вік. Можна помітити, що у наборі присутнє значення, яке помітно відрізняється від інших (outlier). У такій ситуації нейронна мережа намагається згладжувати значення.

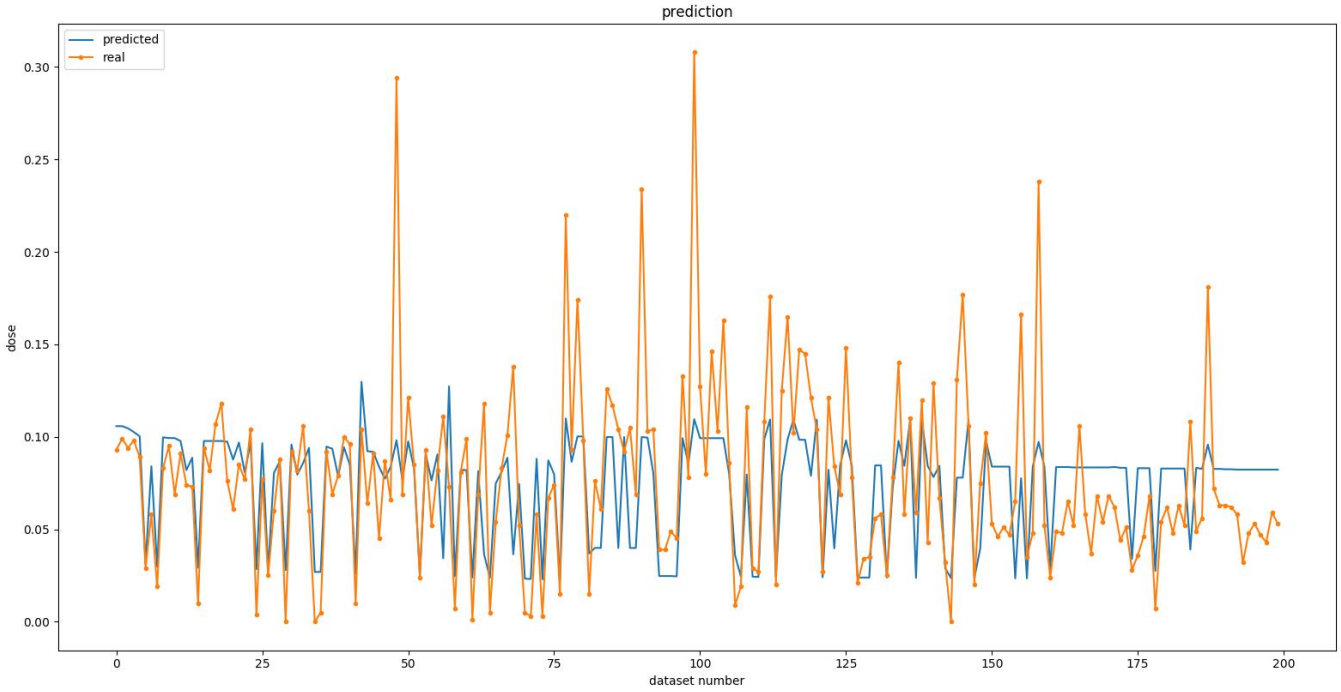


Рисунок 4.8 - Графік спрогнозованих доз внутрішнього опромінення у порівнянні зі справжніми значеннями для набору з 200 даних

З рисунку 4.8 бачимо, що ШНМ краще навчилась прогнозувати дозу внутрішнього опромінення для старшого населення.

Розглянемо розподіл абсолютної похибки поміж даними (рис. 4.9). З рисунка 4.9 видно, що медіана абсолютної похибки (0.0197633) знаходиться лівіше, ніж значення середньої абсолютної похибки (0.0345). Це значить, що переважна більшість зпрогнозованих даних мають похибку, яка менша за середню, і лише невеликий відсоток даних мають значення похибки, яке перевищує середнє.

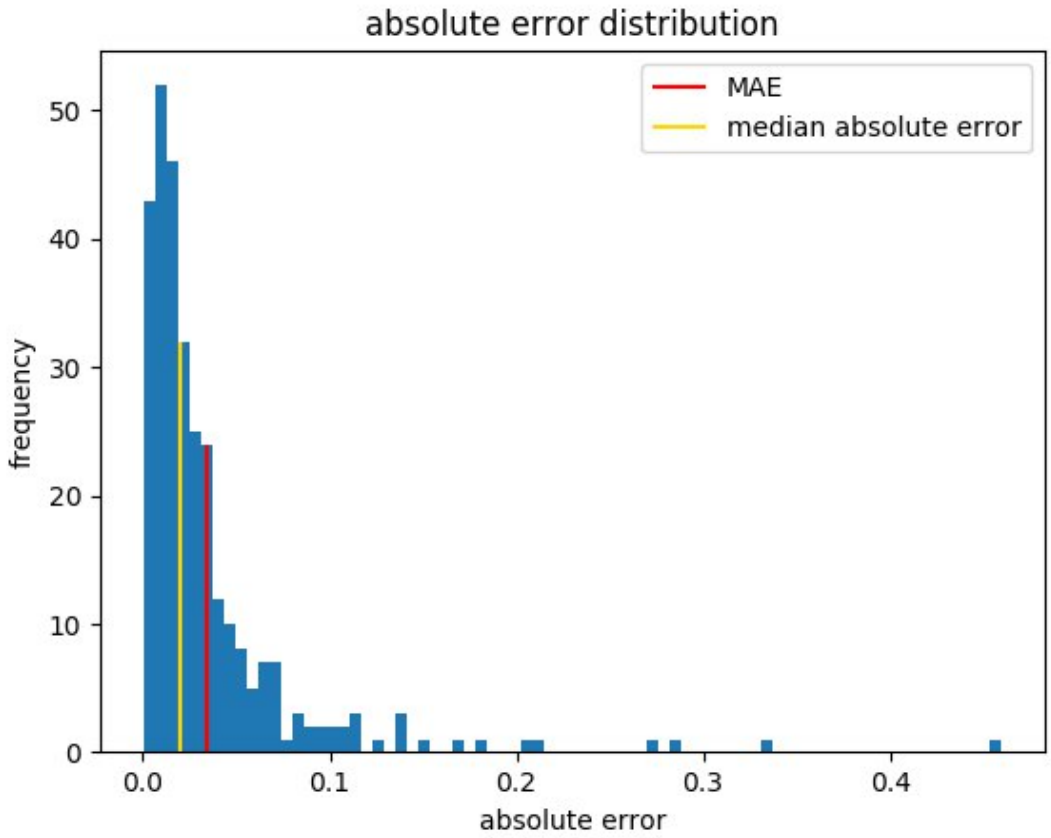


Рисунок 4.9 – Графік розподілу абсолютної похибки

4.4 Висновки до розділу

У цьому розділі розроблено програмне забезпечення для створення математичної моделі формування дози внутрішнього опромінення населення внаслідок аварії на ЧАЕС на основі нейронних мереж.

Програмний продукт написано на мові програмування Python з використанням оболонки PyQt для створення графічного інтерфейсу користувача.

Реалізовану програмно систему навчено на вибірці з 2992 даних, яку надано відділом радіоекології інституту агроекології та природокористування НААН України.

У ході тестування було обчислено показники середньоквадратичної похибки (MSE = 0.0036), середньої абсолютної похибки (MAE = 0.0346), медіану абсолютної похибки (0.0197514). Точність прогнозу на тестових наборах даних склала 89%.

Було показано, що майже 2/3 даних мають значення абсолютної похибки, яке менше за середнє, що є гарним показником, адже такій моделі можна довіряти, а результати експериментів, проведених на такій моделі, будуть близькими до істинних.

Розроблене програмне забезпечення є унікальним для розглянутої предметної області. Висока точність прогнозу зумовлена тим, що система була адаптована для розв’язання задач саме такої специфіки.

ВИСНОВКИ

У даній роботі було розроблено математичну модель формування внутрішньої дози опромінення населення внаслідок аварії на ЧАЕС. Застосування математичної моделі здатне зменшити затрати коштів та часу науковців відділу радіології інституту агроекології та природокористування НААН України.

Було проведено аналіз існуючих математичних методів, таких як: дерева ухвалення рішень, метод опорних векторів, штучні нейронні мережі, метод випадкових лісів, методи регресійного аналізу. Також були розглянуті програмні рішення, що реалізують дані методи, проведено їх порівняльний аналіз (табл. 2.2). У результаті проведеного порівняльного аналізу за наперед визначеними критеріями для вирішення поставленої задачі обрано нейронні мережі.

З метою покращення ефективності було розглянуто й адаптовано методології попередньої обробки даних, а саме кодування категоріальних даних та нормалізація числових даних. У ході регулювання параметрів ШНМ було обрано оптимальну кількість епох для навчання, кількість даних у кожному пакеті та власне архітектуру нейронної мережі. Нейронна мережа має достатньо просту структуру (10 нейронів у вхідному шарі, два шари по 10 нейронів у прихованому шарі, а також 1 нейрон у вихідному шарі), що забезпечує ефективне навчання та робить неможливим вхід нейронної мережі в стан перенавчання.

Спроектоване математичне забезпечення програмно було реалізовано на мові програмування Python з використання оболонки PyQt для створення графічного інтерфейсу.

Розроблену систему навчено на вибірці з 2687 даних із використанням перехресної перевірки і протестовано на вибірці з 299 даних. Система перетворює вихідні дані в формат, що є сумісним для розробленої математичної моделі. Мінімізацію похибки виконано за допомогою методу зворотного поширення

похибки. Розроблене програмне забезпечення надає користувачеві можливість регулювати параметри нейронної мережі, забезпечує графічне представлення значень похибки на кожній навчальній епосі, а також розподіл абсолютної похибки поміж тестовими даними. Передбачено можливість збереження вагів синапсів для використання в майбутньому без етапу навчання, а також можливість навчання на нових наборах даних, що відповідають відповідному формату.

У ході тестування виявлено показники ефективності: на тестовому наборі з 299 даних ефективність навчання склала 89%, а показники середньої абсолютної похибки та медіани абсолютної похибки склали 0.0346 та 0.0197514 відповідно, що говорить про те, що більшість тестових даних містить абсолютну похибку меншу за її середнє значення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Є. І. Степанова, Г. М. Чоботько, І. Є. Колпаков, О. М. Литвинець, Ю. М. Пісковий. Деякі аспекти дозиметричної характеристики та особливості внутрішньоклітинного метаболізму дітей-мешканців радіоактивно забруднених територій. Агроекологічний журнал. — 2013. — №1. — С. 22-27.

2. Raw data (source data or atomic data) [Електронний ресурс] — Режим доступу: http://searchdatamanagement.techtarget.com/definition/raw-data

3. Скрябин А.М. Чернобыль сегодня: социальные аспекты радиационной защиты: Сб. тез. III съезда по радиационным исследованиям, Москва, 14—17 окт. 1997 г. – С. 300.

4. Скрябин А.М. Дозиметрический контроль: техника и методы: М-лы междунар. симпоз. “Актуальные проблемы дозиметрии”. — Мн., 1997. — С. 159–160.

5. Ананій В. Левітін. Алгоритми: введення в розробку й аналіз = Introduction to The Design and Analysis of Algorithms. — 2006 — 576 с.

6. M. Jordan, J. Kleinberg, B. Scholkopf. Information Science and Statistics, 2006.

7. Терехов В.А., Ефимов Д.В., Тюкин И.Ю. Нейронные системы управления. — М.: Высшая школа, 2002. — 184 с.

8. Горбань А.Н. Обучение нейронных сетей. — М.: СССР-США СП «Параграф», 1990. — 160 с.

9. Хайкин С. Нейронные сети: полный курс : пер. с англ. / С. Хайкин. — [2-е изд.]. — М. : Издательский дом «Вильямс». — 2006. — 1104 с.

10. T. Hastie, R. Tibshirani, J. Friedman. The elements of Statistical Learning: Data Mining, Inference, and Prediction. — 2nd ed. — Springer-Verlag, 2009. — 746 p.

11. Ian H. Witten, Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes, and Sally Jo Cunningham Weka: Practical Machine Learning Tools and Techniques with Java Implementations // Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop on

Emerging Knowledge Engineering and Connectionist-Based Information Systems. — 1999. — С. 192-196.

12. А. Ю. Алексеева, О. Г. Ечевская, Г. Д. Ковалева, П. С. Ростовцев. Анализ социологических данных с применением пакета SPSS. Сборник практических заданий. — Новосибирск: Редакционно-издательский центр НГУ, 2003.

13. Бююль Ахим, Цёфель Петр. SPSS: Искусство обработки информации. Анализ статистических данных и восстановление скрытых закономерностей: Пер. с нем. / Ахим Бююль, Петр Цёфель — Спб.: «ДиаСофтЮП», 2005—608 стр.

14. S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight-sharing. Neural Computation, 4(4), 1992.

15. Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961

16. S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight-sharing. Neural Computation, 4(4), 1992.

17. Werbos P. J., Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, Cambridge, MA, 1974.

18. Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function Mathematics of Control, Signals, and Systems, 2(4), p. 303–314.

19. P. Baldi and P. Sadowski. The Dropout Learning Algorithm. Artificial Intelligence, 2014.

20. S. Wang and C. D. Manning. Fast dropout training. In Proceedings of the 30th International Conference on Machine Learning, pages 118–126. ACM, 2013.

21. S. Wager, S. Wang, and P. Liang. Dropout training as adaptive regularization. In Advances in Neural Information Processing Systems 26, pages 351–359, 2013.

22. L. Bottou. Stochastic learning. In O. Bousquet and U. von Luxburg, editors, Advanced Lectures on Machine Learning, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004

23. Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations. MIT Press, 1986.

24. Y. LeCun, I. Kanter, and S.A.Solla: "Second-order properties of error surfaces: learning time and generalization", Advances in Neural Information Processing Systems, vol. 3, pp. 918-924, 1991.

25. Lakhmi C. Jain; N.M. Martin Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications. — CRC Press, CRC Press LLC, 1998

26. Y. LeCun, L. Bottou, G. Orr and K. Muller: "Efficient BackProp", in Orr, G. and Muller K. (Eds), Neural Networks: Tricks of the trade, Springer, 1998

27. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4):541–551, 1989.

Додаток А

Лістинги програм

Лістинг файлу nn.py — файл зі структурою нейронної мережі та функціями для навчання та тестування

```
from keras.layers.core import Dense, Dropout
from keras.models import Sequential
from sklearn import preprocessing
import matplotlib.pyplot as plt
import numpy as np
import pandas
from sklearn.metrics import explained_variance_score, r2_score, median_absolute_error


def load_and_prepare(file_name):
    dataframe = pandas.read_csv(file_name, engine='python', skipfooter=1)      #reading csv file
    dataset = dataframe.values.astype('float32')    #set data type

    np.random.seed(42)  # fix random seed for reproducibility

    # enforce input randomness
    np.random.shuffle(dataset)

    # remove outliers
    outlier_indices = np.where(dataset[:, 0] > 1.0)[0]
    dataset = np.delete(dataset, outlier_indices, axis=0)
    print('Removed {:d} outliers'.format(len(outlier_indices)))

    dataset, cat_dataset = _prepare_columns(dataset)
    return _split_dataset(dataset, cat_dataset)


def _prepare_columns(dataset):
    # Year_MEASUR, Month_MEASUR to number of months
    new_col = np.array((12 * dataset[:, 3] + dataset[:, 4]) - (12 * 1986 + 4))
    new_col.shape = (len(new_col), 1)

    # PROFESSION, Contamination_zone encoding
    new_prof_columns = _hot_encode(dataset[:, [2]], [1, 2, 3, 4, 5, 6, 7])
    new_zone_columns = _hot_encode(dataset[:, [10]], [2, 3, 4, 24])

    # Soil_type_1/2/3 encoding
    new_soil_columns = _hot_encode(dataset[:, [11, 12, 13]], [1, 2, 3, 4, 29, 31, 32, 35])

    # combine all categorical features
    cat_dataset = new_col  # not categorical, it just doesn't need scaling
    cat_dataset = np.append(cat_dataset, new_prof_columns, axis=1)
    cat_dataset = np.append(cat_dataset, new_zone_columns, axis=1)
    cat_dataset = np.append(cat_dataset, new_soil_columns, axis=1)

    # clean table: profession, year, month, population, edu lvl, zone, soil types
    dataset = np.delete(dataset, [2, 3, 4, 5, 8, 10, 11, 12, 13], axis=1)
    return dataset, cat_dataset


# encoding categorical features
def _hot_encode(features, values):
    values = np.array(values)
    cols = np.zeros((len(features), len(values)))
    for i in range(features.shape[0]):
        for j in range(features.shape[1]):
            feature_index = np.where(values == features[i, j])[0]
            if len(feature_index) == 0:
```

```
                raise ValueError('Unknown value: {:d}'.format(int(features[i, j])))
            cols[i, feature_index[0]] = 1
    return cols


def _split_dataset(dataset, cat_dataset):
    # split into training and test sets
    train_size = int(len(dataset) * 0.9)
    train, test = dataset[0:train_size, :], dataset[train_size:len(dataset), :]
    cat_train, cat_test = cat_dataset[0:train_size, :], cat_dataset[train_size:len(dataset), :]
    print('{:d} train samples, {:d} test samples'.format(len(train), len(test)))

    # extract target values
    train_x = np.delete(train, 0, axis=1)
    train_y = train[:, 0]
    test_x = np.delete(test, 0, axis=1)
    test_y = test[:, 0]

    # scale all numerical features (z-score)
    preprocessing.scale(train_x, copy=False)
    preprocessing.scale(test_x, copy=False)

    # add encoded categorical features
    train_x = np.append(train_x, cat_train, axis=1)
    test_x = np.append(test_x, cat_test, axis=1)

    return train_x, train_y, test_x, test_y


# creating neural network model
def _create_model(input_shape, perceptrons):
    model = Sequential()
    model.add(Dense(perceptrons, activation='tanh', input_shape=input_shape)) #input and hidden layer, tanh activation function
    model.add(Dense(perceptrons))   #activityRegularization
    model.add(Dropout(0.5))   #dropout = 0.5
    model.add(Dense(1))     #output layer
    model.compile(loss='mean_squared_error', optimizer='rmsprop', metrics=['mae'])
    return model


# training model
def fit_model(train_x, train_y, perceptrons, epochs, batch_size):
    np.random.seed(42)  # fix random seed for reproducibility

    model = _create_model((train_x.shape[1],), perceptrons)
    history = model.fit(train_x, train_y, epochs=epochs, batch_size=batch_size, verbose=2, validation_split=0.1)
    # list all data in history
    # print(self.history.history.keys())
    return model, history


# testing model
def test_model(model, test_x, test_y):
    # Compare predicted values to actual values
    prediction = model.predict(test_x)
    test_y.shape = (len(test_y), 1)  # for np.hstack()
    np.set_printoptions(suppress=True)  # no scientific notation
    np.set_printoptions(threshold=np.nan)   # print styling
    print('(Test_Y), (Prediction), (Prediction - Test_Y)')
    print(np.hstack((test_y, prediction, prediction-test_y)))       # printing test_y, prediction and their difference
    print("-------------------------")
    mear = median_absolute_error(test_y, prediction)  # calculating median absolute error
    print('median absolute error')

    print(mear)

    # Estimate model performance
    mse = np.mean(np.square(prediction - test_y))
    mae = np.mean(np.abs(prediction - test_y))


 # histogram with absolute error distribution
    hist = plt.hist(np.abs(prediction - test_y), bins=75)
```

```
    plt.vlines(x=mae, color='red', ymin=0, ymax=hist[0][np.max(np.where(hist[1] <= mae))], label='MAE')
    plt.vlines(x=mear, color='gold', ymin=0, ymax=hist[0][np.max(np.where(hist[1] <= mear))], label='median absolute error')
    plt.title('absolute error distribution')
    plt.ylabel('frequency')
    plt.xlabel('absolute error')
    plt.legend()
    plt.show()


    # print('Test scores: {:.4f} MSE, {:.4f} MAE'.format(mse, mae))

    mean = np.mean(prediction)
    std = np.std(prediction)
    return mse, mae, mean, std   # returning testing results


 # predicting dose
def predict(test_x, weights_file, perceptrons):
    model = _create_model((test_x.shape[1],), perceptrons)
    model.load_weights(weights_file)

    prediction = model.predict(test_x)
    return prediction

 # function for plot drawing
def draw_plots(history):
    plt.rcParams["figure.figsize"] = (16, 6)  # default: (8, 6)

    # summarize history for loss/mse
    plt.subplot(1, 2, 1)
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['loss', 'val_loss'], loc='upper left')

    # summarize history for mae
    plt.subplot(1, 2, 2)
    plt.plot(history.history['mean_absolute_error'])
    plt.plot(history.history['val_mean_absolute_error'])
    plt.title('mae')
    plt.ylabel('mae')
    plt.xlabel('epoch')
    plt.legend(['mae', 'val_mae'], loc='upper left')

    plt.show()
```

Лістинг файлу main_ui.py — файл, що створює інтерфейс користувача

```
from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Form(object):
    def setupUi(self, Form):    #setting up user interface
        Form.setObjectName("Form")
        Form.resize(800, 500)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(Form.sizePolicy().hasHeightForWidth())
        Form.setSizePolicy(sizePolicy)
        Form.setMinimumSize(QtCore.QSize(800, 500))
        Form.setMaximumSize(QtCore.QSize(800, 500))
        self.tabWidget = QtWidgets.QTabWidget(Form)
        self.tabWidget.setGeometry(QtCore.QRect(290, 11, 261, 111))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.tabWidget.sizePolicy().hasHeightForWidth())
        self.tabWidget.setSizePolicy(sizePolicy)
```

```
self.tabWidget.setTabShape(QtWidgets.QTabWidget.Rounded)
self.tabWidget.setElideMode(QtCore.Qt.ElideNone)
self.tabWidget.setDocumentMode(False)
self.tabWidget.setObjectName("tabWidget")
self.tab_1 = QtWidgets.QWidget()
self.tab_1.setObjectName("tab_1")
self.saveWeightsBtn = QtWidgets.QPushButton(self.tab_1)
self.saveWeightsBtn.setGeometry(QtCore.QRect(150, 45, 93, 28))
self.saveWeightsBtn.setObjectName("saveWeightsBtn")
self.trainBtn = QtWidgets.QPushButton(self.tab_1)
self.trainBtn.setGeometry(QtCore.QRect(150, 10, 93, 28))
self.trainBtn.setObjectName("trainBtn")
self.batchSizeSpBox = QtWidgets.QSpinBox(self.tab_1)
self.batchSizeSpBox.setGeometry(QtCore.QRect(90, 47, 42, 22))
self.batchSizeSpBox.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.batchSizeSpBox.setMinimum(1)
self.batchSizeSpBox.setProperty("value", 8)
self.batchSizeSpBox.setObjectName("batchSizeSpBox")
self.batchSizeLbl = QtWidgets.QLabel(self.tab_1)
self.batchSizeLbl.setGeometry(QtCore.QRect(10, 47, 71, 20))
self.batchSizeLbl.setLayoutDirection(QtCore.Qt.LeftToRight)
self.batchSizeLbl.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.batchSizeLbl.setObjectName("batchSizeLbl")
self.epochsSpBox = QtWidgets.QSpinBox(self.tab_1)
self.epochsSpBox.setGeometry(QtCore.QRect(90, 12, 42, 22))
self.epochsSpBox.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.epochsSpBox.setMinimum(1)
self.epochsSpBox.setProperty("value", 16)
self.epochsSpBox.setObjectName("epochsSpBox")
self.epochsLbl = QtWidgets.QLabel(self.tab_1)
self.epochsLbl.setGeometry(QtCore.QRect(10, 12, 71, 20))
self.epochsLbl.setLayoutDirection(QtCore.Qt.LeftToRight)
self.epochsLbl.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.epochsLbl.setObjectName("epochsLbl")
self.tabWidget.addTab(self.tab_1, "")
self.tab = QtWidgets.QWidget()
self.tab.setObjectName("tab")
self.loadWeightsBtn = QtWidgets.QPushButton(self.tab)
self.loadWeightsBtn.setGeometry(QtCore.QRect(150, 10, 93, 28))
self.loadWeightsBtn.setObjectName("loadWeightsBtn")
self.predictBtn = QtWidgets.QPushButton(self.tab)
self.predictBtn.setGeometry(QtCore.QRect(150, 45, 93, 28))
self.predictBtn.setObjectName("predictBtn")
self.tabWidget.addTab(self.tab, "")
self.outputLbl = QtWidgets.QLabel(Form)
self.outputLbl.setGeometry(QtCore.QRect(15, 115, 43, 16))
self.outputLbl.setObjectName("outputLbl")
self.plainTextEdit = QtWidgets.QPlainTextEdit(Form)
self.plainTextEdit.setGeometry(QtCore.QRect(11, 137, 781, 351))
self.plainTextEdit.setFrameShadow(QtWidgets.QFrame.Sunken)
self.plainTextEdit.setLineWrapMode(QtWidgets.QPlainTextEdit.NoWrap)
self.plainTextEdit.setReadOnly(True)
self.plainTextEdit.setPlainText("")
self.plainTextEdit.setObjectName("plainTextEdit")
self.loadDataBtn = QtWidgets.QPushButton(Form)
self.loadDataBtn.setGeometry(QtCore.QRect(30, 40, 93, 28))
self.loadDataBtn.setObjectName("loadDataBtn")
self.perceptronsSpBox = QtWidgets.QSpinBox(Form)
self.perceptronsSpBox.setGeometry(QtCore.QRect(150, 40, 42, 22))
self.perceptronsSpBox.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
self.perceptronsSpBox.setSuffix("")
self.perceptronsSpBox.setMinimum(1)
self.perceptronsSpBox.setMaximum(256)
self.perceptronsSpBox.setProperty("value", 12)
self.perceptronsSpBox.setObjectName("perceptronsSpBox")
self.perceptronsLbl = QtWidgets.QLabel(Form)
self.perceptronsLbl.setGeometry(QtCore.QRect(200, 42, 71, 16))
self.perceptronsLbl.setLayoutDirection(QtCore.Qt.LeftToRight)
self.perceptronsLbl.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
self.perceptronsLbl.setObjectName("perceptronsLbl")
self.resultsGroupBox = QtWidgets.QGroupBox(Form)
self.resultsGroupBox.setGeometry(QtCore.QRect(560, 20, 221, 101))
self.resultsGroupBox.setObjectName("resultsGroupBox")
```

```
        self.mseLbl = QtWidgets.QLabel(self.resultsGroupBox)
        self.mseLbl.setGeometry(QtCore.QRect(20, 30, 81, 16))
        self.mseLbl.setObjectName("mseLbl")
        self.maeLbl = QtWidgets.QLabel(self.resultsGroupBox)
        self.maeLbl.setGeometry(QtCore.QRect(20, 50, 81, 16))
        self.maeLbl.setObjectName("maeLbl")
        self.meanLbl = QtWidgets.QLabel(self.resultsGroupBox)
        self.meanLbl.setGeometry(QtCore.QRect(110, 30, 101, 16))
        self.meanLbl.setObjectName("meanLbl")
        self.stdLbl = QtWidgets.QLabel(self.resultsGroupBox)
        self.stdLbl.setGeometry(QtCore.QRect(110, 50, 101, 16))
        self.stdLbl.setObjectName("stdLbl")
        self.timeLbl = QtWidgets.QLabel(self.resultsGroupBox)
        self.timeLbl.setGeometry(QtCore.QRect(20, 70, 151, 16))
        self.timeLbl.setObjectName("timeLbl")
        self.drawPlotsCbx = QtWidgets.QCheckBox(Form)
        self.drawPlotsCbx.setGeometry(QtCore.QRect(150, 70, 91, 20))
        self.drawPlotsCbx.setChecked(False)
        self.drawPlotsCbx.setObjectName("drawPlotsCbx")

        self.retranslateUi(Form)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):              # setting up text labels
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Chernyavskiy diploma"))
        self.saveWeightsBtn.setText(_translate("Form", "Save weights"))
        self.trainBtn.setText(_translate("Form", "Train && test"))
        self.batchSizeLbl.setText(_translate("Form", "Batch size:"))
        self.epochsLbl.setText(_translate("Form", "Epochs:"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_1), _translate("Form", "Train NN"))
        self.loadWeightsBtn.setText(_translate("Form", "Load weights"))
        self.predictBtn.setText(_translate("Form", "Predict"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), _translate("Form", "Load weights"))
        self.outputLbl.setText(_translate("Form", "Output:"))
        self.loadDataBtn.setText(_translate("Form", "Load data"))
        self.perceptronsLbl.setText(_translate("Form", "perceptrons"))
        self.resultsGroupBox.setTitle(_translate("Form", "Results"))
        self.mseLbl.setText(_translate("Form", "MSE: 0.0000"))
        self.maeLbl.setText(_translate("Form", "MAE: 0.0000"))
        self.meanLbl.setText(_translate("Form", "Mean: 0.0000"))
        self.stdLbl.setText(_translate("Form", "Stdev: 0.0000"))
        self.timeLbl.setText(_translate("Form", "Training time: 0.00 sec"))
        self.drawPlotsCbx.setText(_translate("Form", "Draw plots"))
```

Лістинг файлу app.py — файл, що поєднує інтерфейс користувача з функціоналом

```
from main_ui import Ui_Form
from PyQt5 import QtCore, QtGui, QtWidgets
import nn
import numpy as np
import sys
import time
import warnings
import matplotlib.pyplot as plt


class MyApp(Ui_Form):
    def __init__(self, form):
        Ui_Form.__init__(self)
        self.setupUi(form)

        # Redirect output to plainTextEdit
        stream = EmittingStream(text_written=self.__output_written)
        sys.stdout = stream
        sys.stderr = stream

        self.data_file = self.weights_file = ''
        self.train_x = self.train_y = self.test_x = self.test_y = []
```

```
        self.model = self.history = None

        # Callbacks
        self.loadDataBtn.clicked.connect(self.__load_data_clicked)
        self.trainBtn.clicked.connect(self.__train_clicked)
        self.saveWeightsBtn.clicked.connect(self.__save_weights_clicked)
        self.loadWeightsBtn.clicked.connect(self.__load_weights_clicked)
        self.predictBtn.clicked.connect(self.__predict_clicked)
        self.tabWidget.currentChanged.connect(self.__tab_changed)

    def __output_written(self, text):
        self.plainTextEdit.moveCursor(QtGui.QTextCursor.End)
        self.plainTextEdit.insertPlainText(text)


# function for data loading
    def __load_data_clicked(self):
        self.data_file = QtWidgets.QFileDialog.getOpenFileName(caption='Load data', filter='Dataset (*.csv)')[0]
        if not self.data_file:
            return
        print('Loaded dataset: {:s}'.format(self.data_file))
        self.train_x, self.train_y, self.test_x, self.test_y = nn.load_and_prepare(self.data_file)

# function for training NN
    def __train_clicked(self):
        if len(self.train_x) == 0:
            print('No data loaded')
            return
        start_time = time.perf_counter()
        self.model, self.history = nn.fit_model(self.train_x, self.train_y,
                                    self.perceptronsSpBox.value(),
                                    self.epochsSpBox.value(),
                                    self.batchSizeSpBox.value())
        mse, mae, mean, std = nn.test_model(self.model, self.test_x, self.test_y)
        self.mseLbl.setText('MSE: {:.4f}'.format(mse))
        self.maeLbl.setText('MAE: {:.4f}'.format(mae))
        self.meanLbl.setText('Mean: {:.4f}'.format(mean))
        self.stdLbl.setText('Stdev: {:.4f}'.format(std))
        self.timeLbl.setText('Training time: {:.2f} sec'.format(time.perf_counter() - start_time))
        if self.drawPlotsCbx.isChecked():
            nn.draw_plots(self.history)

 #function for weights saving
    def __save_weights_clicked(self):
        if not self.model:
            print('No model trained')
            return
        default_file_name = 'weights_{:d}.h5'.format(self.perceptronsSpBox.value())
        self.weights_file = QtWidgets.QFileDialog.getSaveFileName(caption='Save model', filter='Weights (*.h5)',
                                                directory=default_file_name)[0]
        if not self.weights_file:
            return
        self.model.save_weights(self.weights_file)
        print('Saved weights: {:s}'.format(self.weights_file))

# func for weights loading
    def __load_weights_clicked(self):
        self.weights_file = QtWidgets.QFileDialog.getOpenFileName(caption='Load model', filter='Weights (*.h5)')[0]
        if not self.weights_file:
            return
        print('Loaded weights: {:s}'.format(self.weights_file))

# func for predicting
    def __predict_clicked(self):
        if len(self.test_x) == 0:
            print('No data loaded')
            return
        if not self.weights_file:
            print('No weights loaded')
            return
        test_x = np.append(self.train_x, self.test_x, axis=0)
        prediction = nn.predict(test_x, self.weights_file, self.perceptronsSpBox.value())
```

```
        test_y = np.append(self.train_y, self.test_y, axis=0)

        #sort
        sort_col = test_x[:,0]              #test_y to sort by cs_dose
        prediction = [x for (y, x) in sorted(zip(sort_col, prediction), key=lambda pair: pair[0])]
        test_y = [x for (y, x) in sorted(zip(sort_col, test_y), key=lambda pair: pair[0])]

        plt.plot(prediction)
        plt.plot(test_y, marker=".")

        plt.title('prediction')
        plt.ylabel('dose')
        plt.xlabel('dataset number')
        plt.legend(['predicted', 'real'], loc='upper left')
        # plt.legend(['predicted'], loc='upper left')
        plt.show()
        print(prediction)

# func for changing tab with training/testing and loading weights/predicting
    def __tab_changed(self):
        is_tab_1 = (self.tabWidget.currentIndex() == 0)
        self.drawPlotsCbx.setVisible(is_tab_1)
        self.resultsGroupBox.setVisible(is_tab_1)


class EmittingStream(QtCore.QObject):
    text_written = QtCore.pyqtSignal(str)

    def write(self, text):
        self.text_written.emit(str(text))
```

Додаток Б

Ілюстративний матеріал

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Факультет прикладної математики
Кафедра прикладної математики

Дипломна робота

на здобуття ступеня бакалавра

на тему: «Математична модель формування дози внутрішнього опромінення внаслідок аварії на ЧАЕС на основі нейронних мереж»

Виконав: студент групи КМ-31 Чернявський А.С.

Керівник: старший викладач Любашенко Н.Д.

Київ - 2017

Рисунок Б.1 – Слайд 1

Вступ

- Вживання продуктів харчування місцевого виробництва на радіоактивно забруднених територіях неминуче призводить до змін стану здоров'я.

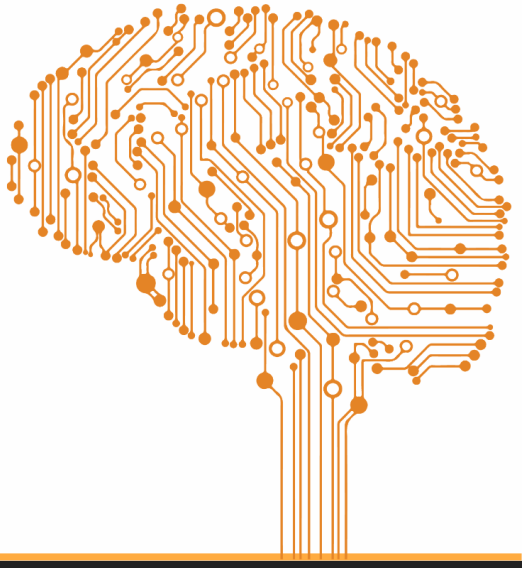
2

Рисунок Б.2 – Слайд 2

Вступ. Машинний інтелект

Напрямки та технології, що розвиваються завдяки методам машинного навчання:

- Робототехніка;
- Складні задачі з рисунками та текстом;
- Прогнозування та кластеризація;
- Автопілот для засобів пересування;
- Розпізнавання образів;
- Пошук нових лікарських засобів.



3

Рисунок Б.3 – Слайд 3

Вступ. Необхідність знаходження моделі

- Складність планування протирадіаційних дій без знання причини формування дози внутрішнього опромінення;
- Зменшення затрат часу та коштів на збір даних про населення;
- Потреба інституту моделювати гіпотетичні ситуації.

4

Рисунок Б.4 – Слайд 4

Постановка задачі. Етапи виконання

- Провести порівняльний аналіз існуючих методів аналізу даних;
- Вибрати та адаптувати обраний метод для вирішення задачі створення математичної моделі формування дози внутрішнього опромінення;
- Розробити програмне забезпечення на базі обраного математичного методу;
- Провести тестування програмної реалізації на контрольних прикладах.

5

Рисунок Б.5 – Слайд 5

Постановка задачі. Критерії до системи

- Моделювання залежності дози внутрішнього опромінення від множини факторів;
- Розроблювана система повинна мати мінімальну похибку;
- Перетворення вхідних даних у сумісний з нейронною мережею формат;
- Адаптація шляхом регулювання основних параметрів моделі;
- Зберігання вагів синапсів після навчання, для забезпечення можливості використання системи без попереднього навчання;
- Спроможність навчатися на нових вибірках даних.

6

Рисунок Б.6 – Слайд 6

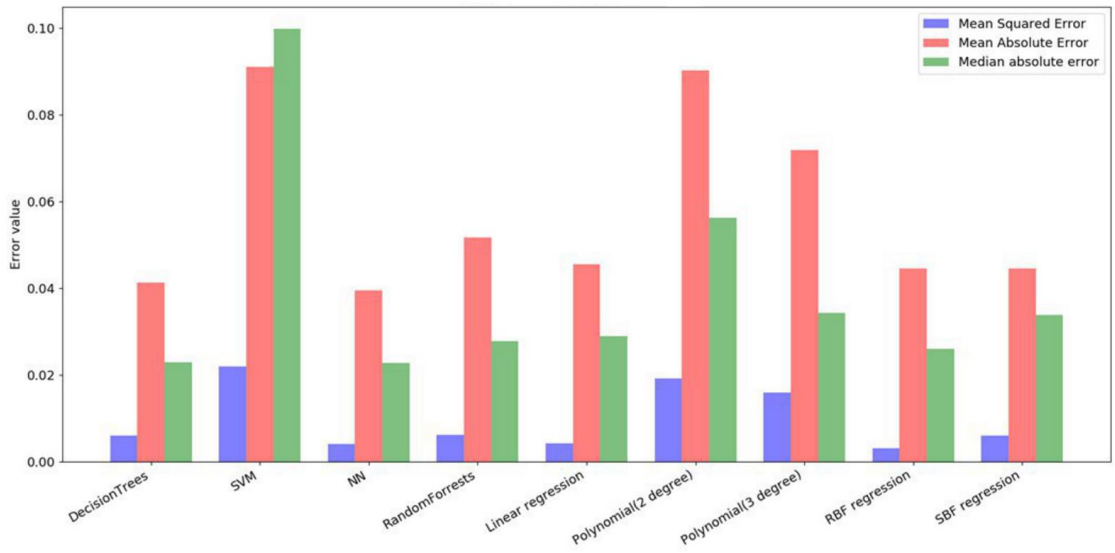
Огляд існуючих математичних методів

- Дерева ухвалення рішень;
- Метод опорних векторів;
- Штучні нейронні мережі;
- Метод випадкових лісів;
- Методи регресійного аналізу:
  - Лінійна регресія;
  - Поліноміальна регресія;
  - Регресія з радіальним базисом;
  - Регресія з сигмоїдальним базисом.

7

Рисунок Б.7 – Слайд 7

Порівняння існуючих методів



8

Рисунок Б.8 – Слайд 8

Існуючі програмні рішення

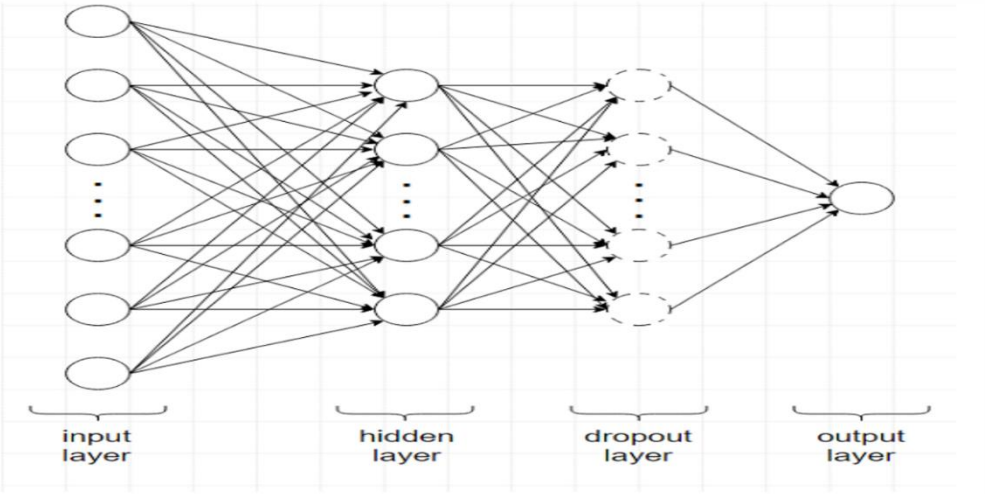
- Waikato Environment for Knowledge Analysis (Weka)

- IBM SPSS Statistics

9

Рисунок Б.9 – Слайд 9

Математичне забезпечення



10

Рисунок Б.10 – Слайд 10

Математичне забезпечення. Значення параметрів

- Розмір вхідного шару: 10 нейронів;
- Розмір прихованого шару: 12 нейронів;
- Розмір вихідного шару: 1 нейрон;
- Коефіцієнт відсіву: 0.5;
- Кількість навчальних епох: 16;
- Кількість даних у пакеті: 8;
- Перехресна перевірка: розбиття на 10 частин;
- Функція активації: гіперболічний тангенс;
- Формули для обрахунку похибок: MAE, MSE.

11

Рисунок Б.11 – Слайд 11

Програмне забезпечення. Вхідні дані

- Файл з даними: текстовий, формат CSV:

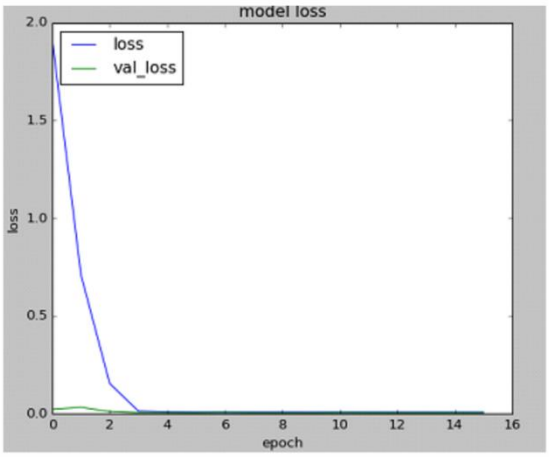
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	DOS_CS	BIRTH_Y	PROFESSI	Year_ME/	Month_M	Populatio	Forest are	Specific fc	Educatior	Cs-137 sc	Contamin	Soil type	Soil type	Soil type	Distance t	Distance to forest. k	
2	0.182	1999	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
3	0.264	1999	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
4	0.147	1990	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
5	0.133	1989	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
6	0.149	1998	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
7	0.146	1996	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
8	0.114	1994	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
9	0.134	1995	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
10	0.113	1994	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
11	0.307	1993	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
12	0.145	1992	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
13	0.19	1993	6	2005	11	109	53	0.13748	1	83	4	1	4	35	40	0.52	
14	0.226	1969	3	2005	11	109	53	0.13748	2	83	4	1	4	35	40	0.52	
15	0.25	1967	4	2005	11	109	53	0.13748	3	83	4	1	4	35	40	0.52	
16	0.153	1962	3	2005	11	109	53	0.13748	2	83	4	1	4	35	40	0.52	
17	0.134	1972	3	2005	11	109	53	0.13748	3	83	4	1	4	35	40	0.52	
18	0	2006	6	2013	10	526	7	0.00376	1	235	2	4	4	4	23.73	5.2	
19	0	2007	6	2013	10	526	7	0.00376	1	235	2	4	4	4	23.73	5.2	
20	0	2007	6	2013	10	526	7	0.00376	1	235	2	4	4	4	23.73	5.2	
21	0	2006	6	2013	10	526	7	0.00376	1	235	2	4	4	4	23.73	5.2	

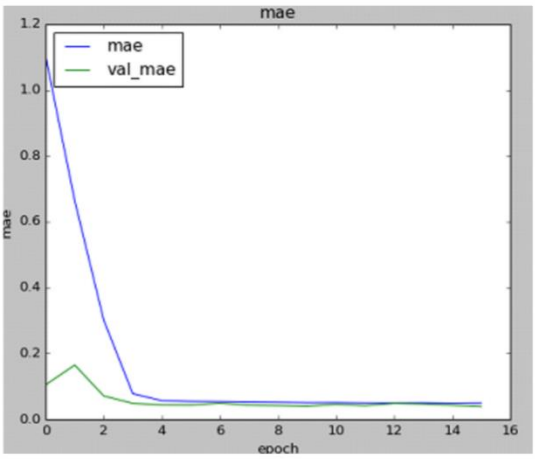
12

Рисунок Б.12 – Слайд 12

Програмне забезпечення. Вихідні дані

- Графіки похибок обрахованих за формулами MSE та MAE:



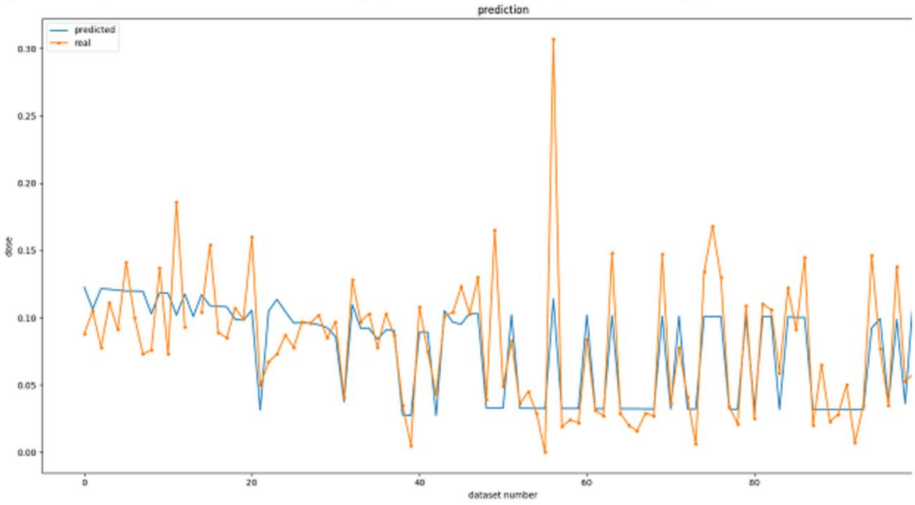


13

Рисунок Б.13 – Слайд 13

Випробовування програмного забезпечення

- Графік спрогнозованих для набору зі 100 даних:

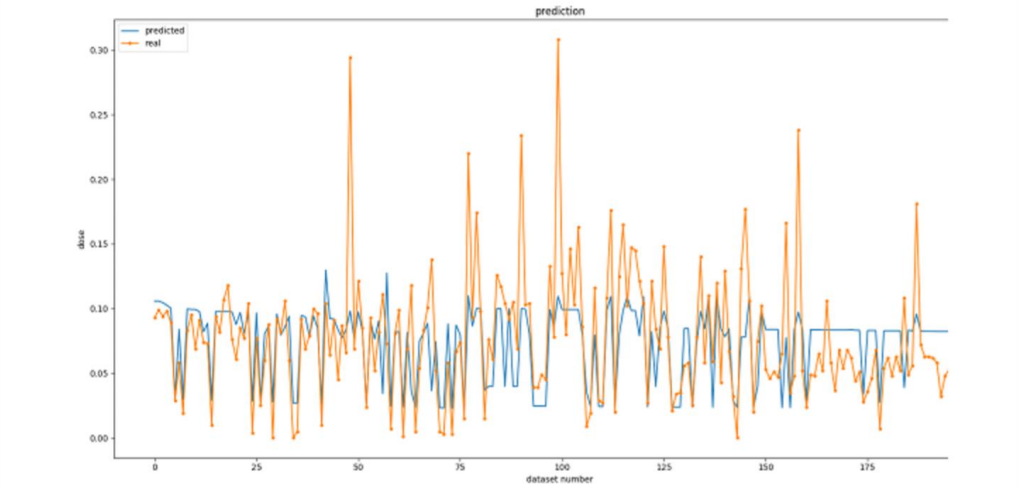


14

Рисунок Б.14 – Слайд 14

Випробовування програмного забезпечення

- Графік спрогнозованих для набору з 200 даних:

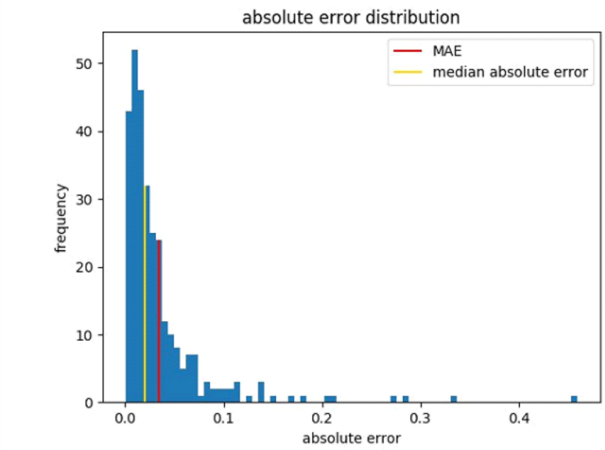


15

Рисунок Б.15 – Слайд 15

Випробовування програмного забезпечення

- Графік розподілу абсолютної похибки:



16

Рисунок Б.16 – Слайд 16

Висновки

- Було розглянуто існуючі математичні методи та програмні рішення, проведено їх порівняльний аналіз.
- Відповідно до обраних критеріїв було реалізовано програмне забезпечення для створення математичної моделі формування дози внутрішнього опромінення.
- Було проведено випробовування розробленого програмного засобу на наборах даних, порівняно їх з вже існуючими реалізаціями методів.

17

Рисунок Б.17 – Слайд 17

Дякую за увагу!

Рисунок Б.18 – Слайд 18