

ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ВИШІЙ НАВЧАЛЬНИЙ ЗАКЛАД
«МІЖРЕГІОНАЛЬНА АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ»»

Інститут комп'ютерно-інформаційних технологій

Кафедра комп'ютерних інформаційних систем і технологій

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

**Тема «Розробка системи оцінки ефективності нейромереж у задачах
обробки та аналізу метеоданих»**

Виконав студент групи Т16–9–21–М1ПЗ(1.6д)

Носов М.Б.

Керівник: к.т.н., доцент

Рябий М.О.

Рецензент: к.т.н., доцент

Рудніченко М.Д.

Допущено до захисту

Завідувач кафедри

д.е.н., професор Кавун С.В.

(підпис)

Київ, 2023

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 79 с., 29 рис., 3 додатки, 32 джерела.

МЕТЕОРОЛОГІЯ, ІНФОРМАЦІЙНА СИСТЕМА, АВТОМАТИЗАЦІЯ, SQLITE, PYTHON, DJANGO, WEB-САЙТ.

Об'єктом дослідження є методи інтелектуального аналізу даних, які використовуються в метеорології, та дозволяють автоматизувати робочі процеси, виконувати впровадження цифрових рішень до робочих процесів, адаптувати інформаційну систему для впровадження в різні напрямки метеорологічної діяльності.

Метою роботи є дослідження особливостей вирішення задачі аналізу метеоданих за допомогою використання алгоритмів машинного навчання. Реалізація такої системи дозволить спростити роботу синоптиків та метеорологів, а також зможе поширити інформацію про метеостан територій та прискорити аналіз доступної метеоінформації.

Для розробки використовується мова програмування Python включно з додатковим фреймворком для веб-розробки Django 3, а також система керування базою даних SQLite 3.

Результатом роботи є розробка інформаційної системи для оцінки ефективності нейромереж у задачах обробки та аналізу метеоданих.

METEOROLOGY, INFORMATION SYSTEM, AUTOMATION, DOCUMENTATION, SQLITE, PYTHON, DJANGO, WEB SITE.

The object of research is the methods of intellectual data analysis used in meteorology, which allow automating work processes, implementing digital solutions to work processes, and adapting the information system for implementation in various areas of meteorological activity.

The purpose of the work is to study the peculiarities of solving the problem of meteorological data analysis using machine learning algorithms. The implementation of such a system will simplify the work of forecasters and meteorologists, and will also be able to spread information about the weather conditions of the territories and speed up the analysis of available weather information.

For development, the Python programming language is used, including the additional Django 3 web development framework, as well as the SQLite 3 database management system.

The result of the work is the development of an information system for evaluating the effectiveness of neural networks in the tasks of processing and analyzing meteorological data.

ЗМІСТ

ВСТУП.....	4
1 АНАЛІЗ ПРЕДМЕТНОЇ ТЕМАТИКИ ДОСЛІДЖЕННЯ	7
1.1 Аналіз особливостей метеорологічних задач.....	7
1.2 Аналіз специфіки машинного навчання	15
1.3 Аналіз можливостей сучасних засобів розробки програмного забезпечення з використанням алгоритмів машинного навчання.....	21
1.4 Висновки до першого розділу.....	30
2. ОБГРУНТУВАННЯ ВИБОРУ НАПРЯМКУ ДОСЛІДЖЕННЯ.....	31
2.1 Постановка завдання регресії.....	31
2.2 Порівняльний аналіз та вибір алгоритму регресії	34
2.3 Концепція попередньої обробки та аналізу даних.....	43
2.4 Висновок до другого розділу	46
3 РОЗРОБКА СИСТЕМИ ОЦІНКИ ЕФЕКТИВНОСТІ НЕЙРОМЕРЕЖ У ЗАДАЧАХ ОБРОБКИ ТА АНАЛІЗУ МЕТЕОДАНИХ.....	47
3.1 Розробка проекту інформаційної системи.....	47
3.2 Опис структури та бази даних інформаційної системи	58
3.3 Опис розробленого інтерфейсу до інформаційної системи.....	65
3.4 Дослідження результатів застосування алгоритмів машинного навчання при вирішенні задач обробки та аналізу метеоданих	72
3.5 Висновки до третього розділу.....	74
ВИСНОВКИ.....	76
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	77
ДОДАТОК А	80
ДОДАТОК Б	93
ДОДАТОК В	99
ДОДАТОК Г	106

ВСТУП

Інформаційні технології активно використовуються в метеорології і роботі синоптиків, так як дозволяють не тільки автоматизувати їх роботу, а також підвищити зручність її виконання і значно підвищити точність результату.

Метеорологія - це наука, яка вивчає процеси, явища, що відбуваються в атмосфері.

Сучасна метеорологія не змогла б розвинутися без впровадження в неї інформаційних технологій і особливо суперкомп'ютерів, пов'язано це з тим що робота синоптиків, зокрема побудова прогнозу погоди, вимагає великих як тимчасових так і людських ресурсів. Починаючи від простого побудови прогнозу погоди закінчуючи попередженням про насування погодних катастроф, які могли б привести до великих людських жертв і масштабних руйнувань.

Використання обчислювальної техніки і відповідних програм дозволяє вирішувати завдання метеорології з великою точністю, для значно більших територій, при цьому за більш короткий термін і з застосуванням менших людських ресурсів. У зв'язку з цим дана галузь вкрай сильно потребує розробки зручних і високопродуктивних програмних засобах.

Тема більш ніж актуальна так як розвинена метеорологія дозволить поліпшити важливі аспекти життя людей, наприклад: збільшити ефективність сільськогосподарської галузі, дозволить визначати безпечні для життя людей місця, прокладати більш оптимальні маршрути кораблям і літакам, мінімізувати наслідки природних катастроф.

Зв'язок роботи з програмами наукових досліджень кафедри ПІ. Для поліпшення загальних процесів автоматизації та створення єдиного інформаційного простору для сфери метеорології повинні використовуватися сучасні інформаційні технології, які дозволяють розробити сучасний програмний засіб, здатний виконати поставлене завдання за менший проміжок часу і забезпечити більш високу точність, ніж людина.

Щодо алгоритмів застосовуваних у цій сфері можна виділити проблеми складності алгоритму, довгого часу його виконання, а також гнучкості алгоритму. Складність алгоритму виявляється у тому що сформована математична модель складна і має оптимальну складність алгоритму виконання. Проблема гнучкості виникає внаслідок того, що складена математична модель пов'язана на певних константах прив'язаних до конкретної території або регіону.

Використання нейронних мереж може вирішити обидві ці проблеми, нейронна мережа простіше у використанні, має універсальний інтерфейс отримання вхідних даних і легко адаптується до змін завдяки своїй структурі. У результаті можна отримати універсальний алгоритм, який зможе обробляти дані з будь-яких регіонів.

Розширення погляду залежність набору вхідних даних. Поточні алгоритми, що використовуються в метеорології, використовують для своєї роботи набір вхідних параметрів, що складається з температури повітря, його вологості, напряму вітру та його швидкості. Ці параметри просто збирати, але аналіз погоди на їх підставі може мати серйозну похибку через "прогалини" в моделі. В даному випадку під "проблами" маються на увазі метеорологічні змінні, які не враховуються в даній моделі, але можуть значно впливати на результат. Нейронні мережі дозволяють легко оцінити значущість своїх вхідних параметрів на результат і могли б допомогти у вирішенні цього завдання у сфері метеорології.

Мета і задачі дослідження. Метою роботи є дослідження ефективності роботи різних варіантів нейронних мереж на задачах обробки та аналізу метеоданих. Для виконання поставленої задачі, необхідно виконати:

- аналіз особливостей метеорологічних задач;
- аналіз специфіки машинного навчання;
- аналіз можливостей сучасних засобів розробки програмного забезпечення з використанням алгоритмів машинного навчання;
- постановка завдання регресії;
- порівняльний аналіз та вибір алгоритму регресії;

- розробка концепції попередньої обробки та аналізу даних;
- розробка проекту інформаційної системи;
- опис структури та бази даних інформаційної системи;
- опис розробленого інтерфейсу до інформаційної системи;
- дослідження результатів застосування алгоритмів машинного навчання при вирішенні задач обробки та аналізу метеоданих.

Об'єкт дослідження. Об'єктом дослідження є різні варіації нейромереж та набори вхідних параметрів для них.

Методи дослідження. В якості наукової основи дослідження використовують: алгоритми машинного навчання, методи оцінки якості моделей прогнозування та класифікації даних.

Наукова новизна одержаних результатів. Наукова новизна отриманих в результаті виконання даної роботи полягає у виявленні найбільш гнучкої та точної моделі нейромережі, яка спроможна забезпечити високі показники метрик оцінки якості вирішення завдань прогнозування метеоданих.

Практичне значення одержаних результатів. Розроблене програмне забезпечення може бути застосовано для завдань аналізу даних, вивчення можливостей програмної імплементації моделей штучних нейронних мереж та у науково-дослідних експериментах по обробці метеоданих.

Публікації. Результати розробки та дослідження створеної інформаційної системи опубліковано у двох науково-практичних конференціях, за результатами яких сформовані збірники тез (додаток Б).

1 АНАЛІЗ ПРЕДМЕТНОЇ ТЕМАТИКИ ДОСЛІДЖЕННЯ

1.1 Аналіз особливостей метеорологічних задач

Метеорологія є комплексом наук все атмосферні явища, такі як: погода, клімат, хмари, склад атмосфери, її будова, тепловий режим і влагообмін, оптичні та акустичні, поведінка фронтів, вітру і електричного поля. Всі ці дослідження допомагають знайти і зрозуміти фізичні залежності які відбуваються в атмосфері, що в кінцевому рахунку наблизить людство до розуміння життєвих циклів планети.

Сучасних метеорологів можна розділити по виду діяльності на наступні категорії або наукові напрямки [1]:

- Фізична метеорологія – займаються розробкою радіолокаційних і космічних методів дослідження атмосферних явищ.
- Динамічна метеорологія – вивчення фізичних механізмів атмосферних процесів.
- Синоптична метеорологія – наука про закономірності зміни погоди.
- Кліматологія – наука вивчаюча клімат як сукупність погодних характеристик за багаторічний період і їх зміна властивих певної території.
- Аерологія – наука вивчає верхні шари атмосфери для декількох десятків кілометрів від поверхні землі.

Прикладні напрямки сучасної метеорології:

- Авіаційна метеорологія – наука вивчає вплив погоди на діяльність авіації.
- Агрометеорологія – наука яка вивчає вплив погоди на сільське господарство.
- Біометеорологія – наука вивчає вплив атмосферних процесів на людину та інші живі організми.

- Ядерна метеорологія – наука вивчає природну і штучну радіоактивність, поширення в атмосфері радіоактивних домішок, вплив ядерних вибухів.

- Радіометеорологія – наука вивчає поширення радіохвиль в атмосфері.

Якщо узагальнити те кожен розділ займається пошуком деяких погодних залежностей і вивчення впливу погодних явищ на деяку галузь. Для виконання цієї роботи необхідно взаємодіяти злагоджено і задіяти безліч фахівців по всьому світу, які кожні три години збирають інформацію про погоду зі всіх доступних джерел, далі проводять їх обробку і будують різні карти описують графічним методом поточний погодний стан атмосфери в різних регіонах. Далі за отриманими картками проводиться аналіз виявляє необхідну інформацію, наприклад це може бути аналіз погодних аномалій для розуміння того чи можуть вони перерости в природні катастрофи і якщо це так варто попередити населення цих регіонів.

Метеорологічні данні які найчастіше збирають для досліджень [3]:

- Місце та час коли були зняті метеодані, базова інформація яка потрібна для прив'язки метеоданих на карті.

- Висота хмар від рівня моря.

- Дальність видимості. Кількість метрів далі котрих не можна побачити наземні об'єкти.

- Вид хмар, який преобладає в даному секторі.

- Напрям вітру.

- Швидкість вітру.

- Температура повітря в час збору даних.

- Точка роси, це температура повітря при якій починає випадати роса.

- Атмосферний тиск на рівні метеостанції.

- Атмосферний тиск на рівні моря.

- Максимальна та мінімальна температура за добу.

- Погодне явище.

- Кількість опадів.

На базі цих метеоданих будуються прогнози погоди та більшості з усіх метеорологічних розрахунків.

Основою для прогнозу погоди є облік періодичних і неперіодичних змін метеорологічних величин і явищ погоди [4]. Періодичні зміни тієї чи іншої метеорологічної величини обумовлюються добовим і річним ходом цієї величини, неперіодичні – еволюцією і переміщенням синоптичних об'єктів: циклонів і антициклонів, повітряних мас і атмосферних фронтів. Ось чому прогнозом погоди завжди передує прогноз синоптичного положення. Найбільшу трудність і практичний інтерес представляє облік імені не періодичних змін.

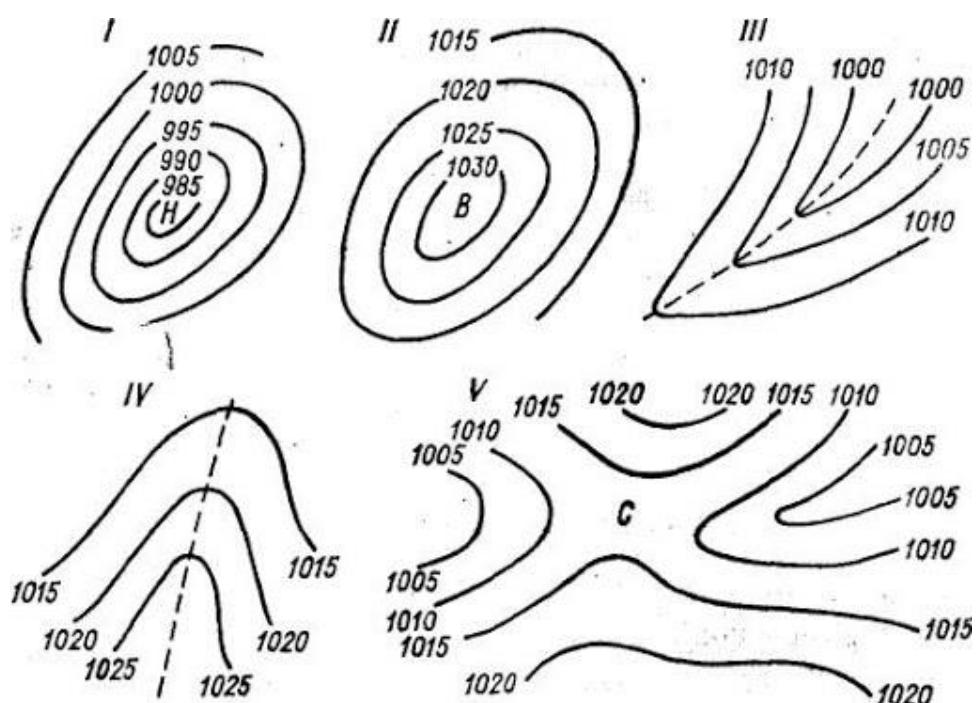


Рисунок 1.1 – Приклад відображення баричних систем на картах.

Синоптичний метод в даний час є основним при розробці довгострокового прогнозу погоди.

Суть методу в тому, що на підставі аналізу карт погоди за кілька послідовних термінів складають прогноз синоптичного положення, який заключає в прогнозі виникнення, переміщення і еволюції повітряних мас, атмосферних

фронтів, баричних систем, приклад відображення баричних систем на картах рис 1.1.

Карта, на яку наносять передбачуване положення синоптичних об'єктів, називається прогностичною, рис 1.2.

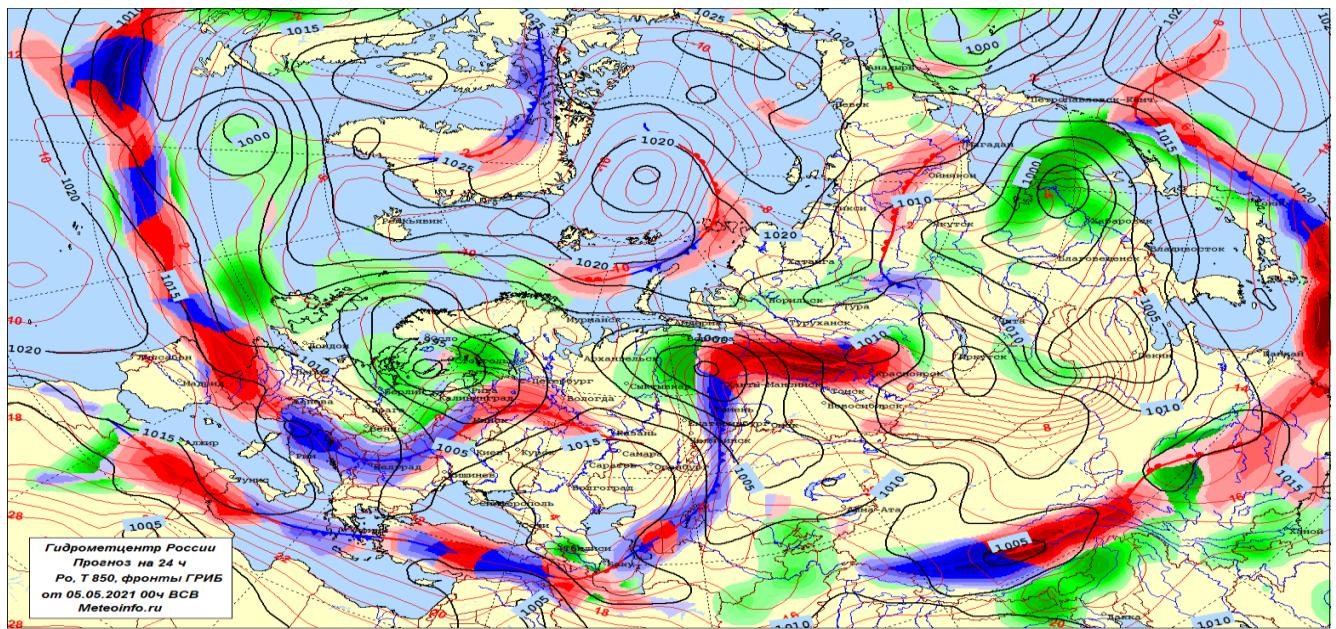


Рисунок 1.2 – Приклад прогностичної метеорологічної карти

Прогностичну карту складають на певний момент часу наступної доби, а іноді на двоє або більше доби вперед.

На підставі прогностичної карти складають прогноз погоди в тому чи іншому районі [4].

Причому прогноз погоди є логічним продовженням прогнозу синоптичного положення і виходить з основного принципу, що полягає в припущення, що з переміщенням і еволюцією синоптичних об'єктів переносяться з певними змінами і властиві їм умови погоди.

Тому за прогностичне значення метеорологічної величини в першому наближенні приймаються їх значення в районі, звідки очікується переміщення синоптичного об'єкту, в район, для якого складається прогноз погоди.

При такому великому обсязі роботи та даних необхідних до зберігання дана галузь не може обійтися без використання інформаційних технологій. Особливо новітніх програмних засобів, які би спростили роботу з розрахунками та потужних комп'ютерів, які би ці алгоритми оброблювали.

Інформаційні технології щільно увійшли і намертво закріпилися в житті сучасних метеорологів.

У своїй щоденній роботі їм необхідно використовувати масивні програмні комплекси які виконують завдання збору, зберігання і перегляду інформації з джерел метеоданих, програми для побудови погодних карт, для визначення і відстеження баричних систем і атмосферних фронтів, побудови прогностичних карт, виявлення погодних аномалій, аналізу погодних даних, побудови графіків залежностей, отримання статистики і графічного відображення метеоданих, а також прогнозування катастроф і інші.

Вплив інформаційних технологій на галузь досить велике, так як без них виконання більшості завдань сучасної метеорології було б неможливим. Якби метеоролози використовували старі або "класичні" методи, то всьому людству довелося б освоїти цю професію і спільно виконувати її завдання. Сам процес збору даних залучає більше кількість людей, а її аналіз зажадав би все доступне в добі час.

Для вирішення та прискорення цих процесів людство прийшло до використання суперкомп'ютерів [5].

Суперкомп'ютером називається комплекс електронних обчислювальних машин працюють як одне ціле, що дає всій системі велику обчислювальну потужність.

Однак навіть суперкомп'ютерів необхідно достатньо часу для розрахунків еволюції баричних систем і прогнозу погоди.

Наприклад, для розрахунку короткострокового прогнозу погоди для всієї території України, з точністю в п'ять квадратних кілометрів, середньостатистичному суперкомп'ютера знадобиться близько години.

На даний момент процес побудови прогнозу запускається не частіше ніж раз на три години, цього цілком достатньо для більшості територій.

Винятками ж є території з високою ймовірністю появи атмосферних катаклізмів. Як приклад таких областей можна привести

Японію, з їх штормами і цунамі, південні штати Америки, з місцевими повенями і торнадо і т.д.

З кожним днем необхідних метеорологічних розрахунків стає все більше, через те що все більше галузей виявилися залежні від інформації про стан атмосфери або про її вплив на свою галузь і отже отримали необхідність в аналізі метеоданих.

Завдяки цій тенденції сильно збільшилося фінансування метеорології що в кінцевому рахунку дозволило метеорологам почати безліч досліджень, які в майбутньому допоможуть краще розуміти процеси атмосфери і їх вплив на наше життя.

Основна частина фінансової підтримки необхідна для придбання обладнання та розробки програмного забезпечення для всіх галузей даної сфери [5].

При активному розвитку інформаційних технологій в сфері метеорології для людства можуть відкритися можливості:

- Ідеальна організація і оптимальний розподіл вирощуваних культур. При точному знанні того які метео-параметри були раніше і які будуть в кожній точці світу люди зможуть розподілити висаджуються культури ідеально по необхідним умовам. Дане нововведення дозволить збільшити обсяги вирощуваних культур, але також пристосуватися до змін клімату і вчасно реагувати на випадки коли земля стає менш придатною для вирощування конкретної культури.

- Оптимізація і полегшення маршрутів суден і літаків. Дане нововведення дозволяє запобігти небезпеці в дорозі і потенційно врятували б життя людей, мінімізували б витрати на ремонтні роботи транспорту, а також не допустити забруднення природи від можливих аварій.

- Можливість побудови різних архітектурних споруд. Для побудови різних споруд важливо знати які погодні умови переважають в районі будівлі щоб зрозуміти як сильно треба змінити конструкцію будівлі для її стійкості.
- Ремонтні роботи. Частково відноситься в попередньому пункті, для деяких ремонтних робіт важливі певні погодні умови з чим і можуть допомогти нові системні комплекси метеорологів, точно вибравши проміжок часу з ідеальною погодою для ремонтних робіт.

Не менш значущий вплив інформаційні технології привнесли в системи для вимірювання метеоданих.

Раніше використовувані аналогові прилади, хоч і відрізнялися дешевизною і примітивністю в роботі з ними, проте були недостатньо точними і досить громіздкими [5].



Рисунок 1.3 – Схема сучасної метеостанції

Сучасні прилади компактні, рис 1.3, мають можливість передавати результати вимірювань по лініях зв'язку і при цьому мають велику точність. Це дозволяє прискорити процес збору даних, а їх автономність дає можливість вільно масштабувати мережу метеостанцій, рис 1.4.

Подібні системи мають основний стовп на якому закріплені різні датчики, наприклад датчики напряму вітру, кількості опадів, атмосферного тиску та різні датчики для аналізу повітря.

Система автономного живлення представляє собою джерело енергії, систему стабілізації енергії та систему накопичування енергії.

Як джерело енергії може бути сонячні панелі чи вітряки. Система накопичування енергії це акумуляторна батарея яка буде живити станцію в час коли енергії системи живлення не буде достатньо енергії [5].

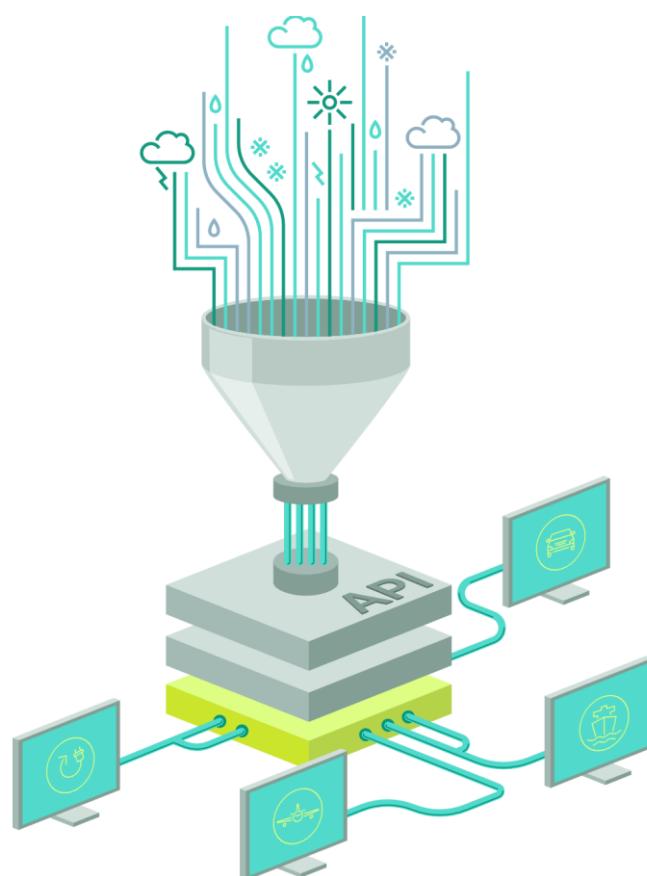


Рисунок 1.4. – Приклад схеми сучасної логіки роботи метеокомплексів

1.2 Аналіз специфіки машинного навчання

Проаналізувавши завдання метеорології можна дійти висновку, що в ній можна застосувати майже всі види та варіації нейромереж та алгоритмів машинного навчання.. Щодо алгоритмів застосовуваних у цій сфері можна виділити проблеми складності алгоритму, довгого часу його виконання, а також гнучкості алгоритму. Складність алгоритму виявляється у тому що сформована математична модель велика і має оптимальну складність алгоритму виконання. Проблема гнучкості виникає внаслідок того, що складена математична модель пов'язана на певних константах прив'язаних до конкретної території або регіону.

Використання нейронних мереж може вирішити обидві ці проблеми, нейронна мережа простіше у використанні, має універсальний інтерфейс отримання вхідних даних і легко адаптується до змін завдяки своїй структурі. У результаті можна отримати універсальний алгоритм, який зможе обробляти дані з будь-яких регіонів.

Розширення погляду на залежність набору вхідних даних. Поточні алгоритми, що використовуються в метеорології, використовують для своєї роботи набір вхідних параметрів, що складається з температури повітря, його вологості, напряму вітру та його швидкості. Ці параметри просто збирати, але аналіз погоди на їх підставі може мати серйозну похибку через "прогалини" в моделі. В даному випадку під "пробілами" маються на увазі метеорологічні змінні, які не враховуються в даній моделі, але можуть значно впливати на результат. Нейронні мережі дозволяють легко оцінити значущість своїх вхідних параметрів на результат і могли б допомогти у вирішенні цього завдання у сфері метеорології [5].

Починаючи від найскладнішого і найкомплекснішого штучного інтелекту, для обробки загальної картини поведінки метеорологічних об'єктів, до простих математичних моделей для аналізу даних конкретного датчика на вузькому участі землі. Для завдань прогнозування використовуються відповідні нейронні мережі,

навчені на сотнях тисяч записів метеоданих з різних точок світу, завдання визначення погодних явищ вирішуються алгоритмами класифікації.

Також значний вплив на результат надає процес підступу та аналізу аномалій.

Аномаліями називають будь-яке значне відхилення в даних, їх поява може будувати висновки про різного роду сторонньому вплив на метеоситуацію в місці їх збору.

У цьому контексті, в першу чергу варто уточнити значущість і види сил, які можуть мати вплив.

Безліч різних факторів може впливати на метеорологічну ситуацію, наприклад, це можуть бути різні природні катастрофи, наприклад, потопи і пожежі або техногенні катастрофи у вигляді тих же пожеж або витоків різних речовин. Подібні явища безпосередньо впливають на метеоситуацію в певній області, безпосередньо впливає на температуру, швидкість і напрям вітру, тиск, вологість. Залежно від масштабів це може вплинути і на загальну метеорологічну картину у світі [6]. Приклад метеостанції домашнього типу наведено на рис.1.5.



Рисунок 1.5 – Приклад метеостанції домашнього типу

Якщо природні катастрофи відрізнятися досить просто, то техногенні можна виділити тільки завдяки датчикам аналізу складу повітря і різні домішки в ньому. Подібні станції сильно відрізняються від професійних аналогів але можуть збирати дані про температуру, вологість повітря і атмосферний тиск, рис 1.5. Деякі моделі дозволяють аналізувати якість повітря, конкретно його склад і кількість домішок. Питання відстеження та аналізу стану повітря не менш важливий для метеорології. У повітрі можуть міститися різні домішки які можуть той чи інший ефект на погоду в регіоні [6]. Таким чином вивчаючи склад повітря можна в майбутньому передбачити згубні явища в регіоні по зміні складу його повітря і навпаки, припустити склад повітря через погодні явища в регіоні, рис 1.6.



Рисунок 1.6 – Карта забруднення повітря і ґрунту України

Факторами зміни можуть бути, як досить очевидні, наприклад заводи або аварії, так і менш очевидні і внаслідок чого більш небезпечні, наприклад розкриття покладів метану внаслідок землетрусів. Аналіз складу повітря більш актуальний для другого випадку, конкретно для систем запобігання катастроф.

Система вивчає аномалії в повітрі і може виявити виниклі проблеми, після чого попередити про це відповідні служби. Існують приклади місць які вимагають постійного моніторингу ситуації, наприклад озера в парку Йеллоустоун, які є потенційно небезпечною зоною і можуть завдати серйозної шкоди регіону [6]. Також подібні системи фахівці можуть відслідковувати так звані «мертві зони» в морях і океанах, це зони в яких сталася якась аномалія в слідстві якої живі організми в ній вимерли або покинули її, рис. 1.7. Найчастіше подібний ефект викликаний забрудненнями вод і подібні місця відслідковуються за складом повітря.

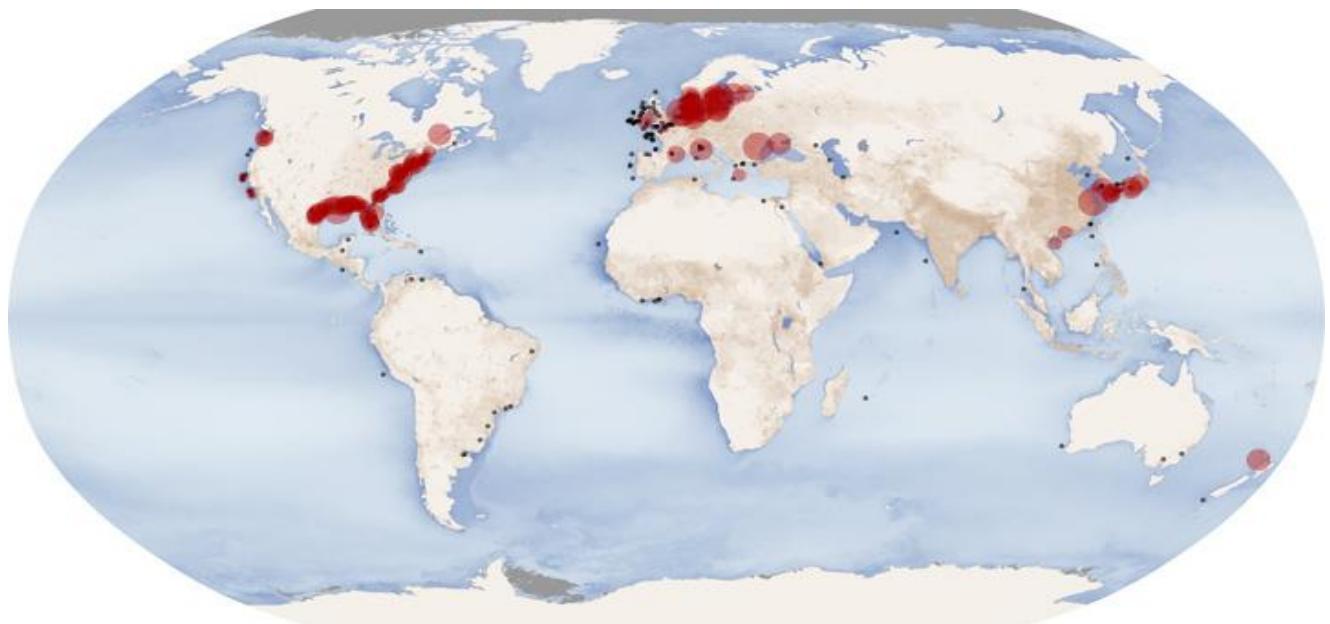


Рисунок 1.7 – Карта світового океану із зазначеними на ній мертвими зонами

Все це створює велику кількість параметрів та їх комбінацій, які треба враховувати при аналізі метеоданих, а про багатьох важливих метеорологічних змінних можуть навіть не підозрювати. У вирішенні цих завдань використання

нейронних мереж є вкрай ефективним. Подібні алгоритми можуть визначати різні види аномалій залежно від величезних відхилень у даних.

У деяких випадках навіть стабільні дані можуть бути ознакою аномалії. На планеті відбувається багато внутрішніх циклів, які мають свій вплив на метеоситуацію, наприклад такі як зміна пір року та цикли кліматичних періодів.

Ефективність розвитку сфери сильно залежить від стандартизації її елементів, наприклад розробки визначених стандартів для метеостанцій, що дозволить максимально спростити процес розробки ПЗ для них, вибрати найоптимальніші алгоритми роботи [7].

Створення специфічних програмних бібліотек і фреймворків дозволить полегшити написання програм, оскільки базові алгоритми будуть вже готові, це зменшить шанс помилки і збільшить швидкість розробки шляхом перевикористання коду. Стосовно циклів кліматичних періодів, рис. 1.8, варто описати докладніше.

Так як кругообіг циклів кліматичних періодів, як наприклад льодовиковий період, тривають велику кількість років, то варто також враховувати факти еволюції метеорологічних змінних. Під еволюцією у разі мається на увазі природне зміні очікуваних значень метеорологічних змінних протягом тривалого часу.

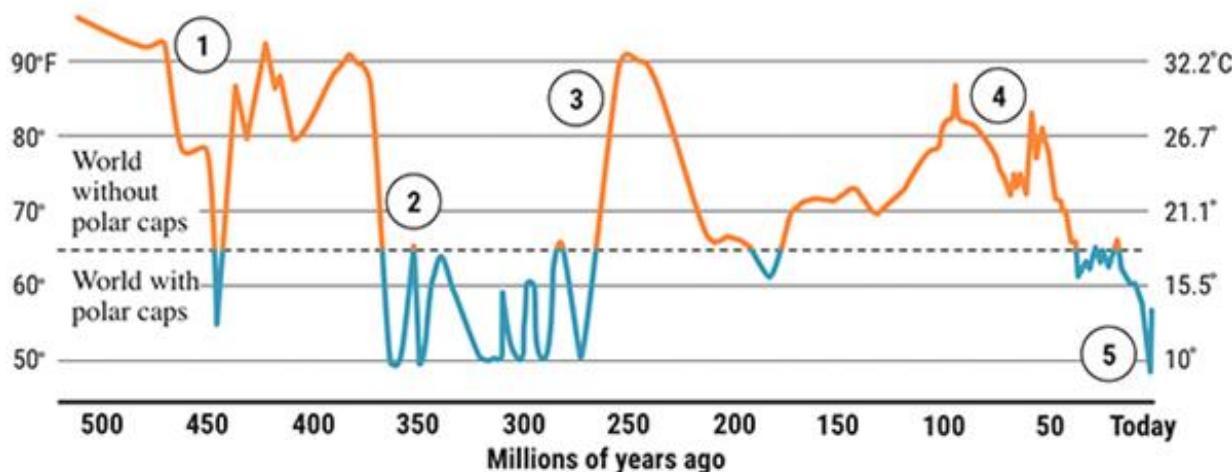


Рисунок 1.8 – Цикли леднікового періоду

Враховуючи це, можна зрозуміти, що доведеться регулярно адаптувати алгоритми обробки та аналізу під відповідні норми метеоданих, щоб уникнути випадкового визначення їх як аномальних.

Існують методики для визначення значущості вхідних параметрів на результат роботи мережі, для цього можна запитати коефіцієнти, що склалися у нейронів при навчанні і на їх підставі можна зробити висновок. Необхідно це для того, щоб дати оцінку тому, наскільки ефективно складено набір вхідних даних [7].

За допомогою цього можна або виключити параметри, які мінімально впливають на результат, що вкрай важливо, так як це дає можливість прискорити алгоритм і зменшити обсяг даних, що зберігаються.

Також завдяки цій можливості можна збільшити точність роботи свого алгоритму, для цього необхідно виділити всі теоретично можливі метеорологічні змінні, які можуть брати участь у процесі, який аналізуєте, після чого пробувати підставляти їх комбінації у вигляді вхідних параметрів.

Дана методика лізу даних за допомогою нейронних мереж, з подальшим аналізом ваг нейронів у них дозволяє виявити навіть найбільш неочевидні залежності.

Наприклад, у 2018 році нейромережа виявила залежність між парою людських генів і схильністю людини до раку.

Це відкриває можливість до нових досліджень і може в майбутньому зробити значний внесок у лікування раку.

Подібні взаємозв'язки можуть бути виявлені і у сфері метеорології, наприклад взаємозв'язок виду поверхні на те, як поводиться температура повітря.

Іноді дана методика може застосовуватися для аналізу впливу будь-яких сучасних розробок на загальну метеорологічну ситуацію. У серпні 2022 року був розроблений матеріал, що відображає 97.9% світла, що дозволяє знизити нагрівання покритих нею об'єктів. У кінцевому підсумку це світло розсіюється, отже впливає навколоїшні об'єкти, отже нагріває їх, в такий спосіб загальна картина змінюється і може привести до несподіваних наслідків [7].

1.3 Аналіз можливостей сучасних засобів розробки програмного забезпечення з використанням алгоритмів машинного навчання

Сучасні засоби розробки систем з використанням алгоритмів машинного навчання рясніють різними функціями на вирішення будь-яких заадч. До таких систем належать TensorFlow, PyTorch, sklearn.

TensorFlow – це бібліотека для машинного навчання, групи технологій, яка дозволяє навчати штучний інтелект вирішенню різних завдань. Бібліотека спочатку розроблена для Python і найчастіше використовується з ним.

Існують продаж TensorFlow для інших мов: C#, C++, Go, Java, Swift і так далі. Вони використовуються рідше за основну — головним чином для написання коду під специфічні платформи. Сама бібліотека написана мовою Python із використанням швидкого та продуктивного C++ для вирішення математичних завдань. Тому вона ефективно працює зі складними обчисленнями. Бібліотека розроблена Google як продовження внутрішньої бібліотеки компанії. TensorFlow безкоштовна, вона має відкритий вихідний код, який можна переглянути на GitHub, її активно підтримує спільнота ентузіастів. Назва читається як «тензор флоу» та утворена від двох понять: тензор та потік даних [7].

Для чого використовується TensorFlow Сама бібліотека включає безліч інструментів для різних напрямків ML, але найчастіше використовується для роботи з нейронними мережами. Це структури, натхненні пристроєм мереж нейронів у людській нервовій системі. Нейронні мережі складаються із програмних елементів-«нейронів» та зв'язків між ними, і такий пристрій дозволяє їм навчатися. TensorFlow працює зі звичайними та глибокими нейронними мережами різних типів: рекурентними, згортковими тощо. Також вона використовується для машинного та глибокого навчання.

Приклади використання технологій — розпізнавання природної мови, зображень та рукописних текстів, різноманітні завдання класифікації чи кластеризації, обробка великих даних, як-от метеоданих.

Можна виділити такі особливості TensorFlow:

- У TensorFlow моделі представлені за допомогою графів математичних абстракцій, які складаються з вершин і шляхів між ними. Граф можна порівняти із схемою доріг між різними точками. У програмуванні це зазвичай потрібно при вирішенні «маршрутних» завдань та при створенні нейронних мереж.
- TensorFlow працює з тензорами - багатовимірними структурами даних у векторному, тобто спрямованому просторі. Вони використовуються в лінійній алгебрі та фізиці. Звідси походить назва бібліотеки. За допомогою тензорів описуються шляхи графа, а вершини це математичні операції.
- Обчислення TensorFlow виражаються як потоки даних через граф. Це означає, що інформація «рухається» по графу, передається шляхами від вершини до вершини.
- Бібліотека може працювати на потужностях звичайного центрального процесора (CPU) або використовувати потужності графічного процесора (GPU). Режим перемикається у коді. Існує спеціальний тензорний процесор TPU, створений розробниками бібліотеки, - ним можна скористатися через хмарні сервіси Google.

Перевагами TensorFlow можна виділити високий рівень абстракції. Бібліотека написана так, що не потрібно думати про технічну реалізацію абстрактних понять. Можна зосередитись на описі логіки програми та на математиці, а спосіб реалізації обчислень – завдання TensorFlow, а не програміста. Це полегшує розробку та дозволяє сконцентруватися на важливих завданнях [7].

Інтерактивна розробка TensorFlow дозволяє працювати з компонентами моделі окремо і створювати її на ходу, при цьому окремо перевіряти кожен елемент. Це зручніше, ніж описувати граф як єдину монолітну структуру. Підхід робить розробку більш інтерактивною — структуру можна гнучко налаштовувати та змінювати.

Гнучкість TensorFlow можна використовувати для створення нейронних мереж, для глибокого навчання та інших напрямів ML. Його функції різноманітні

на вирішення широкого спектра завдань. Завдяки графам та тензорам у TensorFlow можна легко зобразити складну математичну структуру. Гнучкість стосується як функцій, а й технічної боку питання: бібліотека працює і з центральним, і з графічним процесором, її легко використовувати з іншими інструментами для машинного навчання. Наприклад, TensorFlow застосовують із API Keras.

Кросплатформенність TensorFlow, як і сам Python, працює в популярних операційних системах, локально або в хмарі. Вона має розширення для мобільних пристройів, IoT та браузерних додатків. Для мобільних пристройів та інтернету речей можна скористатися середовищем TensorFlow Lite. А якщо модель машинного навчання має працювати у браузері, підійде TensorFlow.js - версія для використання з JavaScript та Node.js.

Велика спільнота TensorFlow — популярна бібліотека, тому на багато питань легко можна знайти відповідь у спільноті. Воно розвиває технологію, створює нові продукти та доповнення, пов'язані з TensorFlow, пише документацію та туторіали. Все це полегшує старт і дозволяє отримати з бібліотеки максимум користі [8].

Власні стандарти TensorFlow – продукт Google. Компанія відома своїми стандартами для технологій. Перша версія бібліотеки була призначена для внутрішнього використання. Досі в TensorFlow зустрічається неочевидна поведінка, через яку код може бути складніше налагоджувати. Складнощі можна компенсувати, якщо уважно вивчати документацію та користуватися додатковими інструментами для налагодження.

Високе споживання пам'яті, якщо TensorFlow використовується з графічним процесором, вона забирає всю його пам'ять. Це знижує продуктивність. Наприклад, якщо моделей кілька і вони написані на різних фреймворках, TensorFlow забере відеопам'ять у інших – виникне помилка. Щоб цього не було, споживання пам'яті бібліотекою треба обмежувати вручну. Складність у вивчені машинного навчання загалом. З TensorFlow складності можуть виникнути через її специфічні стандарти — це не найприязніша до новачків бібліотека.

TensorFlow встановлюється за допомогою pip, пакетного менеджера Python. Версії для роботи з CPU та GPU завантажуються окремо. Також для коректної роботи потрібна Anaconda – спеціальний дистрибутив Python для машинного навчання [8].

Більш детально про внутрішній пристрій TensorFlow можна сказати що TensorFlow — це наскрізна платформа для машинного навчання. Він підтримує наступне:

- Числове обчислення на основі багатовимірного масиву (подібно до NumPy.).
- GPU і розподілена обробка.
- Автоматична диференціація.
- Побудова моделі, навчання та експорт.

Тензори, TensorFlow працює з багатовимірними масивами або тензорами, представленими як об'єкти `tf.Tensor`. Найважливішими атрибутами `tf.Tensor` є його форма і `dtype`: `Tensor.shape`: повідомляє розмір тензора вздовжожної з його осей. `Tensor.dtype`: повідомляє вам тип усіх елементів у тензорі.

TensorFlow реалізує стандартні математичні операції над тензорами, а також багато спеціалізованих операцій для машинного навчання.

Змінні, звичайні об'єкти `tf.Tensor` є незмінними. Щоб зберегти вагові коефіцієнти моделі (або інший змінний стан) у TensorFlow, використовуйте `tf.Variable` [8].

Градієнтний спуск і пов'язані з ним алгоритми є наріжним каменем сучасного машинного навчання. Щоб увімкнути це, TensorFlow реалізує автоматичне диференціювання (autodiff), яке використовує обчислення для обчислення градієнтів. Зазвичай ви використовуєте це для обчислення градієнта помилки або втрати моделі відносно її ваг.

Хоча можно використовувати TensorFlow в інтерактивному режимі, як і будь-яку іншу бібліотеку Python, TensorFlow також надає інструменти для:

- Оптимізація продуктивності: для прискорення навчання та висновків.

- Експорт: щоб ви могли зберегти свою модель після завершення навчання.

Для цього потрібно використовувати `tf.function`, щоб відокремити чистий код TensorFlow від Python [9]. Під час першого запуску функції `tf.function`, хоча вона виконується на Python, вона фіксує повний оптимізований графік, що представляє обчислення TensorFlow, виконані в межах функції. Під час наступних викликів TensorFlow виконує лише оптимізований графік, пропускаючи будь-які кроки, не пов'язані з TensorFlow. Графік може бути непридатним для повторного використання для вхідних даних із іншою підписом (форма і `dtype`), тому натомість створюється новий графік. Ці отримані графіки дають дві переваги:

- У багатьох випадках вони забезпечують значне прискорення виконання (хоча це не тривіальний приклад).
- Можна експортувати ці графіки за допомогою `tf.saved_model` для запуску в інших системах, як-от сервер або мобільний пристрій, встановлення Python не потрібне.

Модулі, шари та моделі. `tf.Module` – це клас для керування вашими об'єктами `tf.Variable` та об'єктами `tf.function`, які з ними працюють. Клас `tf.Module` необхідний для підтримки двох важливих функцій:

Можна зберігати та відновлювати значення своїх змінних за допомогою `tf.train.Checkpoint`. Це корисно під час навчання, оскільки можна швидко зберегти та відновити стан моделі.

Можна імпортувати та експортувати значення `tf.Variable` і графіки `tf.function` за допомогою `tf.saved_model`. Це дозволяє запускати вашу модель незалежно від програми Python, яка її створила.

Отримана `SavedModel` не залежить від коду, який її створив. Ви можете завантажити `SavedModel` з Python, інших мовних прив'язок або TensorFlow Serving.

Ви також можете конвертувати його для запуску з TensorFlow Lite або TensorFlow JS. Класи `tf.keras.layers.Layer` та `tf.keras.Model` побудовані на `tf.Module`

забезпечуючи додаткову функціональність і зручні методи для створення, навчання та збереження моделей. Деякі з них демонструються в наступному розділі.

Однак альтернативою TensorFlow можна представити PyTorch. PyTorch визначається як бібліотека машинного навчання з відкритим кодом для Python. Він використовується для програм, таких як обробка природної мови.

Спочатку він був розроблений дослідницькою групою зі штучного інтелекту Facebook та програмним забезпеченням Uber Pyro для ймовірнісного програмування, яке ґрунтуються на ньому [9].

Спочатку PyTorch був розроблений Х'ю Перкінсом як оболонка Python для LusJIT, заснована на платформі Torch. Є два варіанти PyTorch.

PyTorch перепроектує та впроваджує Torch у Python, спільно використовуючи ті ж основні бібліотеки С для внутрішнього коду.

Розробники PyTorch налаштували цей внутрішній код для ефективної роботи з Python.

Вони також зберегли апаратне прискорення на основі графічного процесора, а також функції розширення, які зробили Torch на базі Lua.

Простий інтерфейс – PyTorch пропонує простий використання API; отже, він вважається дуже простим у роботі та працює на Python. Виконання коду у цьому середовищі досить просто.

Використання Python – це бібліотека вважається Pythonic, яка плавно інтегрується зі стеком даних Python. Таким чином, він може використовувати всі сервіси та функціональні можливості, які пропонують середовищем Python.

Обчислювальні графи PyTorch надає відмінну платформу, яка пропонує динамічні обчислювальні графи. Таким чином користувач може змінити їх під час виконання.

Це дуже корисно, коли розробник не знає скільки пам'яті потрібно для створення моделі нейронної мережі [9].

У таблиці 1.1 наведено порівняння між TensorFlow та PyTorch

Таблиця 1.1 – Порівняння бібліотек PyTorch та TensorFlow

PyTorch	TensorFlow
PyTorch тісно пов'язаний із заснованим на lua фреймворком Torch, який активно використовується у Facebook.	TensorFlow розроблено Google Brain і активно використовується в Google.
PyTorch є відносно новим у порівнянні з іншими конкурентними технологіями.	TensorFlow не є новим і розглядається багатьма дослідниками та професіоналами як інструмент, необхідний для роботи.
PyTorch включає все в імперативній і динамічній манері.	TensorFlow включає статичні та динамічні графіки у вигляді комбінації.
Граф обчислень у PyTorch визначається під час виконання.	TensorFlow не включає опцію часу виконання.
PyTorch включає можливість розгортання для мобільних і вбудованих платформ.	TensorFlow краще працює для вбудованих фреймворків.

Переваги PyTorch:

- Код легко налагоджувати та розуміти.
- Він включає багато шарів, як Torch.
- Включає безліч функцій втрати.
- Це можна як розширення NumPy для графічних процесорів.
- Це дозволяє будувати мережі, структура яких залежить самих обчислень.

Чим швидше виходить обчислювати свою функцію і чим гнучкіші можливості для її визначення, тим краще. Тепер, коли кожен фреймворк вміє використовувати всю потужність відеокарт, перший критерій перестав відігравати значну роль [9].

Тензорні обчислення - основа PyTorch, каркас, навколо якого збільшується вся інша функціональність. На жаль, не можна сказати, що міць і виразність бібліотеки в даному аспекті збігаються з такою у NumPy. Щодо роботи з тензорами, PyTorch керується принципом максимальної простоти та прозорості, надаючи тонку обгортку над викликами BLAS. Цей фреймворк є низькорівневим у сенсі реалізації інтерфейсу його використання. Однак його більш ніж достатньо для вирішення майже будь-яких задач машинного навчання.

Ще однією цікавою бібліотекою для розв'язання задач машинного навчання є sklearn. Це бібліотека яка допомагає створювати прості моделі для машинного навчання, вони застосовуються для обчислення моделей, що легко збираються, для алгоритмів де потрібна швидкості і мала витрата ресурсів пам'яті.

Які завдання вирішує бібліотека, наприклад, препроцесинг. термін "препроцесинг" (preprocessing) означає попередню обробку даних. Вони приводяться до виду, необхідного для подачі на вхід будь-якого алгоритму. Наприклад, зображення підганяються під один розмір та колірну схему. З даних вилучаються ключові ознаки, якими буде вчитися модель, і також призводять до потрібного формату.

Зменшення розмірності. Часто у даних міститься надмірна інформація. Наприклад, деякі ознаки можна отримати з інших. Щоб зробити подальший аналіз ефективнішим, розмірність вибірки зменшується — так, щоб зберегти максимум корисних даних. І тому використовуються спеціальні методи, наприклад метод головних компонент.

Виявлення аномалій. Алгоритми «відсікають» з набору даних помилкові чи дивні записи, які лише додають зайві похибки. Це потрібно, щоб аналіз та навчання працювали точніше.

Вибір датасету. Якщо потрібно познайомитися з бібліотекою, можна скористатися одним із готових навчальних наборів. Вони теж є у бібліотеці.

Вибір моделі. Функції та алгоритми допомагають оцінити ефективність різних моделей для вирішення задачі. Їх можна порівнювати один з одним, проводити валідацію результатів, вибирати точніші. Це потрібно для покращення якості навчання [9]. Приклад регресії наведено на рис.1.9.

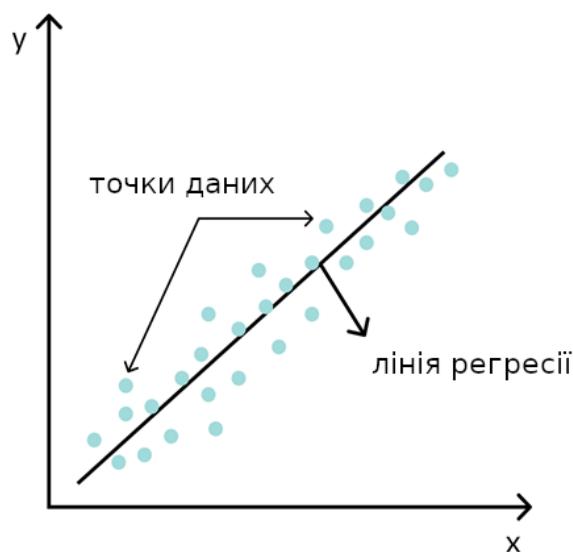


Рисунок 1.9 — Приклад регресії

Регресія. Це прогнозування показників за наявними даними, які можуть приймати нескінченну кількість різних значень. Ці показники би мало бути пов'язані з будь-яким об'єктом, тобто. бути його атрибутами. Наприклад, прогноз кількості користувачів на сайті у різні дні – це завдання регресії.

Класифікація - прогнозування показника з кінцевою кількістю значень. Простіше визначення — прогнозування категорії, до якої належить об'єкт. Розпізнавання жанру тексту чи об'єктів на зображенні – це завдання класифікації [10].

Кластеризація. Це розподіл даних з датасету за великими групами — кластерами, тобто їх угруповання. Схожі об'єкти об'єднуються у класи. Критерії, якими визначається схожість, залежить від моделі та умов завдання.

1.4 Висновки до першого розділу

Результатом проведеного аналізу предметної області, було визначено що сучасна метеорологія є добрим напрямком для впровадження сучасних інформаційних технологій. Це пов'язано з актуальним використанням засобів комунікації та інформаційного обміну між метеостанціями та метеоцентраторами [10]. Проведений аналіз інформаційних матеріалів показав, що сфера потребує нові алгоритми аналізу метеоданих, які будуть більш гнучкими та будуть аналізувати більше різноманітних метеорологічних змінних. Результати аналізу існуючих алгоритмів прогнозу метеоданих з ціллю найти серед них більш оптимальний та точний. Також потрібно провести дослідження з ціллю знайти найкращу комбінацію метеорологічних змінних для розрахунку прогнозу погоди. Розглянувши найпопулярніші засоби розробки, що використовуються в галузі машинного навчання, можна дійти висновку, що найбільш зручною для вирішення задач роботи є мова Python та фреймворк для створення нейромереж TensorFlow 2, адже саме в ній є всі необхідні функції. Отримані в рамках розділу результати будуть використані під час обґрунтування вибірку напрямку дослідження та формалізації завдання з розробки програмного забезпечення у подальших розділах роботи.

2. ОБГРУНТУВАННЯ ВИБОРУ НАПРЯМКУ ДОСЛІДЖЕННЯ

2.1 Постановка завдання регресії

З метою визначення усіх функціональних можливостей інформаційних систем для метеорологів необхідно проаналізувати існуючі рішення та сформувати головні вимоги до таких систем. Для об'єктивного порівняння, необхідно використовувати тільки web-рішення, які автоматизують робочі задачі метеорологів, або спрощують роботу метеорологічного центру.

Відсутність системи збору метеоданих це критичний недолік для майбутнього метеорології, який не дасть їй можливості розвиватися.

Основним завданням системи є аналіз метеорологічних даних з різних джерел і побудова короткострокового прогнозу погоди з прогнозуванням можливих надзвичайних погодних явищ. Гроза і сильна злива, град і шквальний вітер, сніговий буран або екстремальна температура часто формуються протягом короткого часу і часто мають відносно невеликі розміри, що не дозволяє створення прогнозу погоди звичайними інструментами.

Система короткострокового прогнозу погоди і попередження про надзвичайні погодні умови аналізує метеорологічні дані в режимі реального часу в рамках заданої місцевості обмеженою відносно невеликою територією і буде прогноз виникнення надзвичайних погодних умов в рамках декількох годин, що дозволяє надати більш точний прогноз [11].

Починається весь процес роботи системи зі збору даних з якими далі доведеться працювати, після чого вони обробляються та зберігаються у системі. Після цього в дію вступає модуль аналізу цих даних, він складається з алгоритму розрахунку регресії та алгоритму класифікації. На даний момент розглянемо алгоритм розрахунку регресії, адже саме це завдання вирішує система під час побудови короткострокового прогнозу погоди.

Розрахунок алгоритму регресії починається з побудови математичного рівняння, яке оцінює лінію простої (парної) лінійної регресії:

$$Y = a + bx, \quad (2.1)$$

де x називається незалежною змінною чи предиктором. Y – залежна змінна або змінна відгук. Це значення, що ми очікуємо для y (y середньому), якщо знаємо величину x , тобто. це «передбачене значення y » a – вільний член (перетин) лінії оцінки; це значення Y коли $x=0$, рис 2.1. b – кутовий коефіцієнт чи градієнт оціненої лінії; вона є величину, яку Y збільшується загалом, якщо ми збільшуємо x однією одиницею. a та b називають коефіцієнтами регресії оціненої лінії, хоча цей термін часто використовують тільки для b [11].

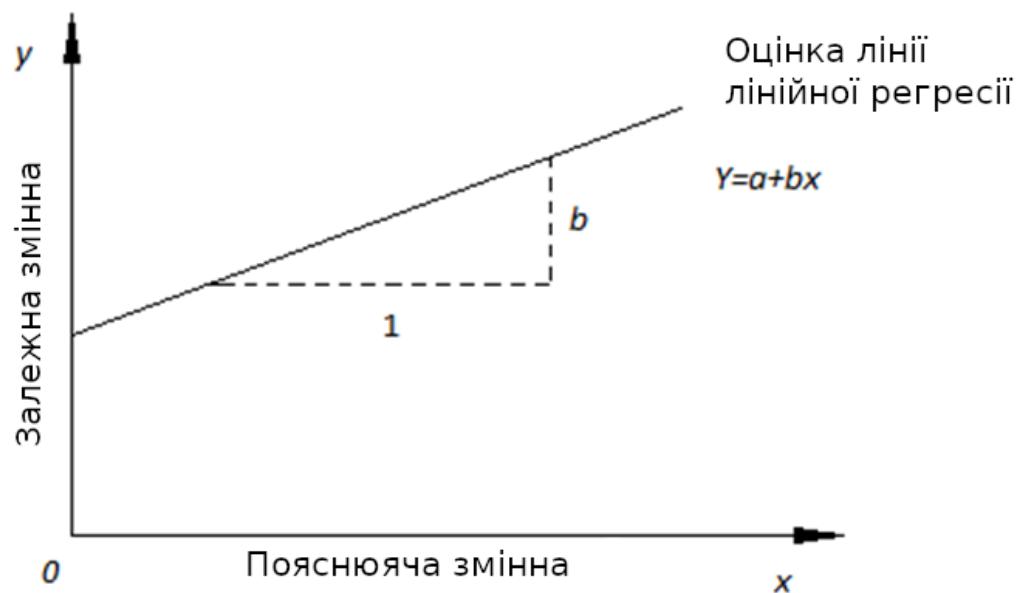


Рисунок 2.1 - Лінія лінійної регресії, що показує перетин a та кутовий коефіцієнт b (величину зростання Y при збільшенні x на одну одиницю)

Парну лінійну регресію можна розширити, включивши до неї більше однієї незалежної змінної; у цьому випадку вона відома як множинна регресія.

Аномальні значення (викиди) та точки впливу. "Впливове" спостереження, якщо воно опущене, змінює одну або більше оцінок параметрів моделі (тобто кутовий коефіцієнт або вільний член). Викид (спостереження, що суперечить більшості значень у наборі даних) може бути "впливовим" спостереженням і може добре виявлятися візуально, під час огляду двовимірної діаграми розсіювання або графіка залишків. І для викидів, і для "впливових" спостережень (крапок) використовують моделі як з їх включенням, так і без них звертають увагу на зміну оцінки (коефіцієнтів регресії). При проведенні аналізу не варто відкидати викиди або точки впливу автоматично, оскільки звичайне ігнорування може вплинути на отримані результати [11].

Гіпотеза лінійної регресії при побудові лінійної регресії перевіряється нульова гіпотеза про те, що генеральний кутовий коефіцієнт лінії регресії β дорівнює нулю. Якщо кутовий коефіцієнт лінії дорівнює нулю, між x і y немає лінійного співвідношення: зміна x не впливає на y . Для тестування нульової гіпотези про те, що дійсний кутовий коефіцієнт β дорівнює нулю використовується наступний алгоритм:

Обчислити статистику критерію, що дорівнює відношенню $blSE(b)$, яка підпорядковується t – розподілу з $(n - 2)$ ступенями свободи, де $SE(b)$ - стандартна помилка коефіцієнта b .

$$b = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2} \quad (2.2)$$

$$SE(b) = \frac{s_{rez}}{\sqrt{\sum(x-\bar{x})^2}} \quad (2.3)$$

Зазвичай, якщо досягнутий рівень значущості $P < 0.05$ нульова гіпотеза відхиляється. Можна розрахувати 95% довірчий інтервал для генерального кутового коефіцієнта β :

$$b \pm t_{0.05} SE(b) \quad (2.4)$$

де $t_{0.05}$ - відсоткова точка t - розподілу зі ступенями свободи ($n - 2$) що дає можливість двостороннього критерію 0.05. Це той інтервал, який містить генеральний кутовий коефіцієнт з ймовірністю 95%.

Для великих вибірок, де, $n \geq 100$ можно апроксимувати $t_{0.05}$ значенням 1,96 (тобто статистика критерію прагнутиме до нормального розподілу).

Оцінка якості лінійної регресії: коефіцієнт детермінації R^2 . Через лінійне співвідношення y і x очікується, що y змінюється, у міру того як змінюється x , і називаємо це варіацією, яка обумовлена чи пояснюється регресією. Залишкова варіація має бути якнайменше.

Якщо це так, то більшість варіації y пояснюватиметься регресією, а точки лежатимуть близько до лінії регресії, тобто. лінія добре відповідає даним [11].

Частка загальної дисперсії y , яка пояснюється регресією називають коефіцієнтом детермінації, зазвичай виражаютъ через відсоткове співвідношення та позначають R^2 (у парній лінійній регресії це величина r^2 , квадрат коефіцієнта кореляції), що дозволяє суб'єктивно оцінити якість рівняння регресії.

Різниця $(100 - R^2)$ є відсотком дисперсії який не можна пояснити регресією.

Немає формального тесту для оцінки R^2 потрібно покластися на суб'єктивне судження, щоб визначити якість припасування лінії регресії.

2.2 Порівняльний аналіз та вибір алгоритму регресії

У разі під алгоритмом регресії мається на увазі збірний образ всіх алгоритмів прогнозування будь-яких даних [11]. Далі розглядаються різні варіанти архітектури нейронних мереж, які застосовуються в алгоритмах прогнозування.

Першою розглянутою архітектурою буде багатошаровий перцептрон, рис 2.2.

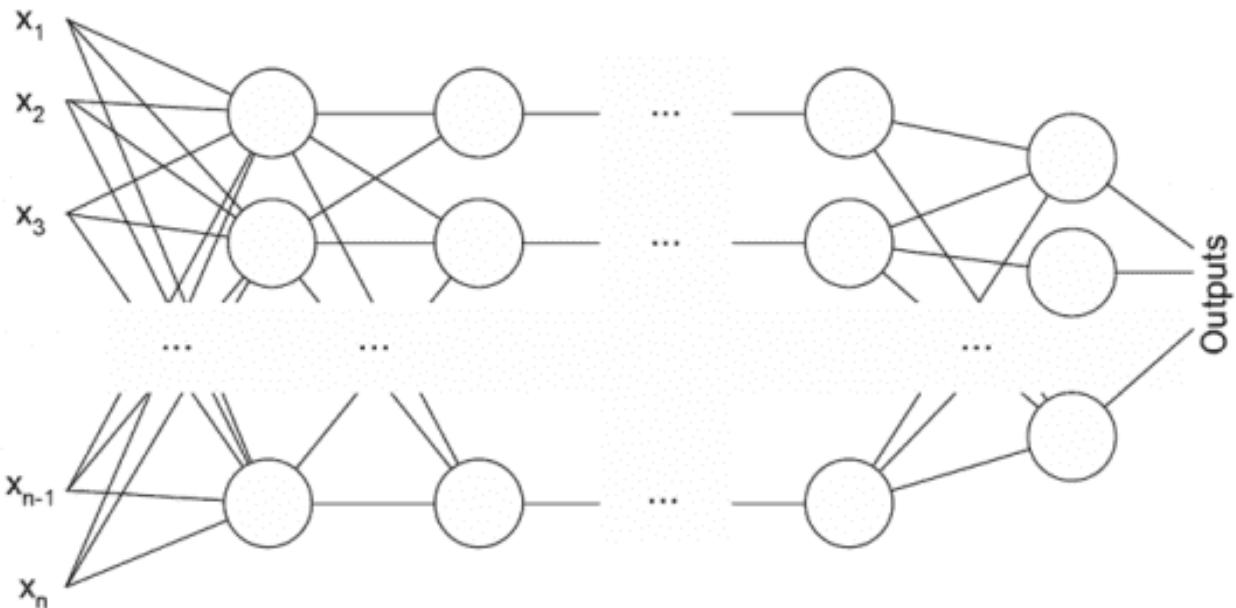


Рисунок 2.2 – Багатошаровий перцептрон

Багатошаровий перцептрон складається з трьох або більше шарів. Кожен вузол у шарі з'єднаний з кожним вузлом у наступному шарі, що робить мережу повністю пов'язаною. Приховані шари містять вузли мережі (одиниці), що не піддаються спостереженню. Кожна прихована одиниця є функцією виваженої суми вхідних даних.

Ця функція є функцією активації, при цьому значення ваги визначаються алгоритмом оцінки. Мережа містить другий прихований рівень, кожна прихована одиниця у якому є функцією від виваженої суми одиниць у першому прихованому рівні. В обох рівнях використовують однакову функцію активації. Кількість прихованих шарів, багатошаровий перцептрон може мати один або два приховані рівні [11].

Функція активації "зв'язує" шар зважених сум об'єктів із наступним шаром значень даних об'єктів.

Гіперболічний тангенс. Це така функція: $y(c) = \tanh(c) = \frac{(ec - e^{-c})}{(ec + e^{-c})}$. Вона визначена для дійсних змінних та переводить їх у діапазон $(-1, 1)$. При використанні автоматичного вибору архітектури це функція для всіх нейронів у прихованих шарах.

Сігмоїд. Це така функція: $y(c) = \frac{1}{(1+e^{-c})}$. Вона визначена для дійсних змінних та переводить їх у діапазон $(0, 1)$.

Кількість нейронів. Кількість одиниць у кожному прихованому шарі може бути задано явно або автоматично визначено алгоритмом оцінки. Вихідний шар містить цільові (залежні) змінні. функція активації. Функція активації "зв'язує" шар зважених сум об'єктів із наступним шаром значень даних об'єктів.

Тотожність. Це функція $\gamma(c) = c$. Вона визначена всім аргументам і повертає їх незміненими. При використанні автоматичного вибору архітектури це функція активації для нейронів у прихованих шарах, якщо не існує кількісних залежних змінних.

Софтмакс. Це така функція: $y(ck) = \frac{\exp(ck)}{\sum j \exp(cj)}$. Вона приймає вектор дійсних аргументів і перетворює їх на вектор, компоненти якого укладені в діапазоні $(0, 1)$, а їх сума дорівнює 1. Функція softmax доступна тільки в тому випадку, якщо всі залежні категоріальні змінні. При використанні автоматичного вибору архітектури це функція активації для нейронів у вихідному шарі, якщо всі залежні категоріальні змінні.

Така архітектура частіше знаходить застосування у завданнях розпізнавання мови та машинному перекладі [12].

Рекурсивні нейронні мережі - вид нейронних мереж, що працюють з даними змінної довжини. Моделі рекурсивних мереж використовують ієрархічні структури зразків під час навчання. Наприклад, зображення, що складаються зі сцен, що поєднують підсцени, що включають багато об'єктів. Виявлення структури сцени та її деконструкція-нетривіальне завдання. При цьому необхідно ідентифікувати окремі об'єкти, так і всю структуру сцени.

У рекурсивних мережах нейрони з одинаковими вагами активуються рекурсивно відповідно до структури мережі. У процесі роботи рекурсивної мережі виробляється модель для передбачення як структур змінної розмірності, так і скалярних структур через активацію структури відповідно до топології. Мережі RvNNs успішно застосовуються при навчанні послідовних структур та дерев у завданнях обробки природної мови, при цьому фрази та речення моделюються через векторне уявлення слів. RvNNs спочатку з'явилися для розподіленого уявлення структур, використовуючи предикати математичної логіки. Розробки рекурсивних мереж та перші моделі почалися в середині 1990-х.

У найпростішій архітектурі вузли мережі сходяться до батьків через матрицю ваг прихованого шару, що використовується багаторазово через всю мережу, і нелінійну функцію активації типу гіперболічного тангенсу. Архітектура простої рекурсивної мережі (W - навчена матриця ваг) наведена на рис.2.3 [15].

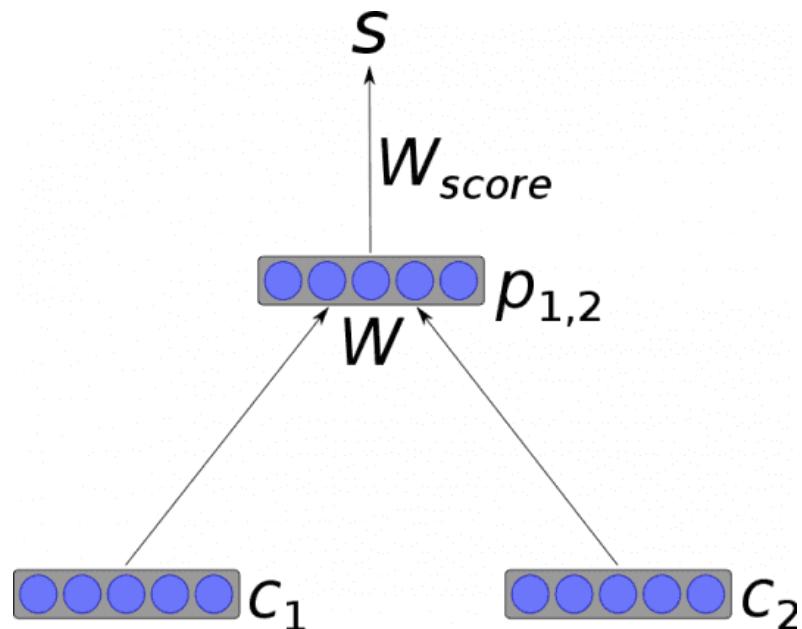


Рисунок 2.3 – Архітектура простої рекурсивної мережі

Ця архітектура з деяким удосконаленням використовується для послідовного дешифрування натуральних сцен зображення або для структурування речень природної мови.

Рекурсивна каскадна кореляція RecCC - це підхід до конструювання рекурсивних мереж, що оперують із трьома доменами, перші додатки такого роду з'явилися в хімії, а розширення утворює спрямований ациклічний граф. У 2004 році було запропоновано систему навчання рекурсивної мережі без вчителя. Тензорні рекурсивні мережі використовують одну функцію тензора для всіх вузлів дерева.

Рекурентна нейронна мережа, на відміну прямої нейронної мережі, є варіантом рекурсивної IHC, у якій зв'язки між нейронами — спрямовані цикли. Останнє означає, що вихідна інформація залежить не тільки від поточного входу, але також станів нейрона на попередньому кроці. Така пам'ять дозволяє користувачам вирішувати завдання NLP: розпізнавання рукописного тексту чи мови. У статті Natural Language Generation, Paraphrasing and Summarization of User Reviews with Recurrent Neural Networks автори показують модель рекурентної мережі, яка генерує нові пропозиції та короткий зміст текстового документа. Siwei Lai, Liheng Xu, Kang Liu, та Jun Zhao у своїй роботі Recurrent Convolutional Neural Networks for Text Classification створили рекурентну згорткову нейромережу для класифікації тексту без рукотворних ознак. Модель порівнюється з існуючими методами класифікації тексту - Bag of Words, Bigrams + LR, SVM, LDA, Tree Kernels, рекурсивними та згортковими мережами. Описана модель перевершує за якістю традиційні методи для всіх використовуваних датасетів [15].

Існує багато різновидів, рішень та конструктивних елементів рекурентних нейронних мереж. Проблема рекурентної мережі полягає в тому, що якщо враховувати кожен крок часу, то стає необхідним для кожного кроку часу створювати свій шар нейронів, що викликає серйозні обчислювальні складності. Крім того, багатошарові реалізації виявляються обчислювально нестійкими, тому що в них зазвичай зникають або зашкалюють ваги.

Якщо обмежити розрахунок фіксованим тимчасовим вікном, то отримані моделі не відображатимуть довгострокових трендів. Різні підходи намагаються

вдосконалити модель історичної пам'яті та механізм запам'ятовування та забування.

Цілком рекурентна мережа, ця базова архітектура розроблена у 1980-х. Мережа будується з вузлів, кожен із яких з'єднаний з усіма іншими вузлами. У кожного нейрона поріг активації змінюється з часом і є речовим числом. Кожна сполучка має змінну речовинну вагу.

Вузли поділяються на вхідні, вихідні та приховані для навчання з учителем з дискретним часом, кожен (дискретний) крок часу на вхідні вузли подаються дані, а інші вузли завершують свою активацію, і вихідні сигнали готуються для передачі нейроном наступного рівня.

Якщо мережа відповідає за розпізнавання мови, в результаті на вихідні вузли надходять вже мітки (розпізнані слова) [15].

У навчанні з підкріпленим (reinforcement learning) немає вчителя, що забезпечує цільові сигнали для мережі, натомість іноді використовується функція пристосованості (придатності) або функція оцінки (reward function), за якою проводиться оцінка якості роботи мережі, при цьому значення на виході впливає на поведінку мережі на вході, зокрема, якщо мережа реалізує гру, на виході вимірюється кількість пунктів виграшу чи оцінки позиції.

Кожен ланцюжок обчислює помилку як сумарну девіацію за вихідними сигналами мережі.

Якщо є набір зразків навчання, помилка обчислюється з урахуванням помилок кожного зразка.

Мережа довгої короткострокової пам'яті (Long Short-Term Memory, LSTM) - різновид архітектури рекурентної нейромережі, створена для більш точного моделювання часових послідовностей та їх довгострокових залежностей, ніж традиційна рекурентна мережа.

Структура LSTM мережі наведена на рис.2.4. LSTM-мережа не використовує функцію активації в рекурентних компонентах, збережені значення не модифікуються, а градієнт не прагне зникнути під час тренування.

Часто LSTM застосовується у блоках по кілька елементів. Ці блоки складаються з 3 або 4 затворів (наприклад, вхідного, вихідного та гейту забування), які контролюють побудову інформаційного потоку по логістичній функції.

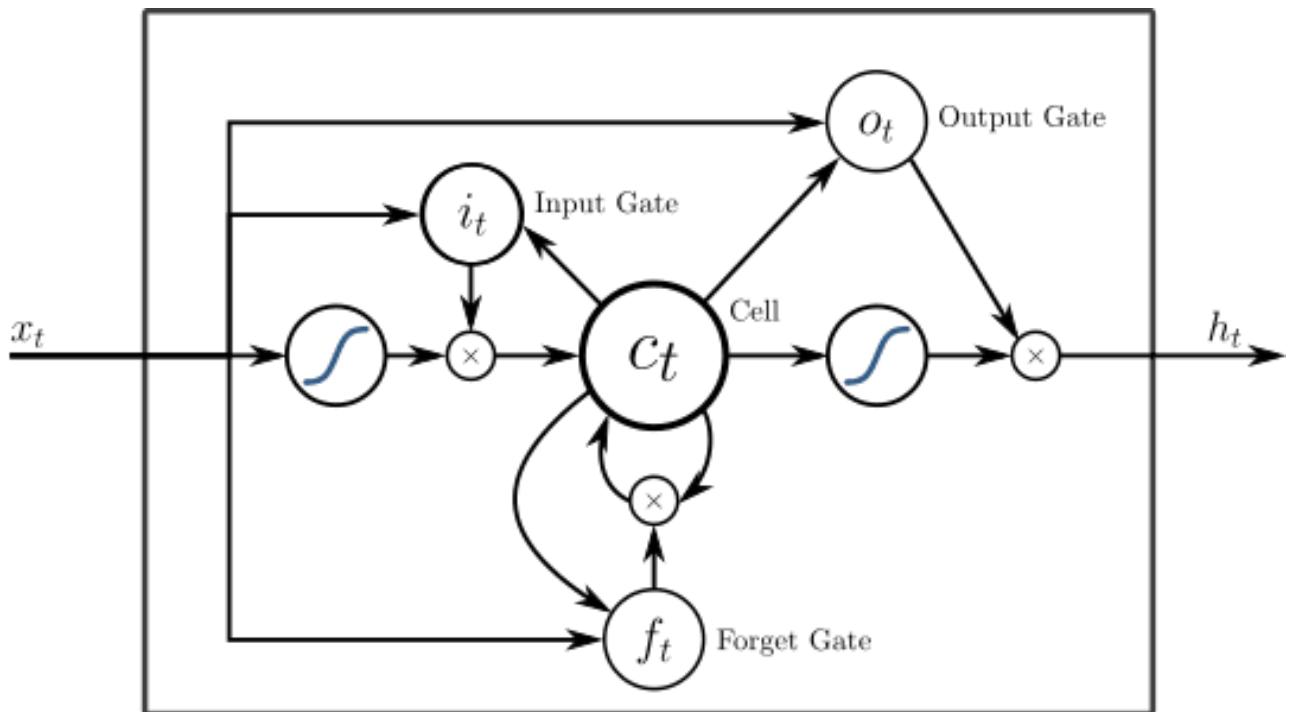


Рисунок 2.4 – Структура LSTM мережі

У Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling автори показують архітектуру глибокої LSTM рекурентної мережі, яка досягає хороших результатів для великомасштабного акустичного моделювання [15].

У роботі Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network представлена модель автоматичної морфологічної розмітки. Модель показує точність 97.4 % у розмітці. Apple, Amazon, Google, Microsoft та інші компанії впровадили у продукти LSTM-мережі як фундаментальний елемент. Модель послідовності, якщо розглядати на високому

рівні, модель seq2seq має кодер, декодер і проміжний крок як основні компоненти (рис.2.5).

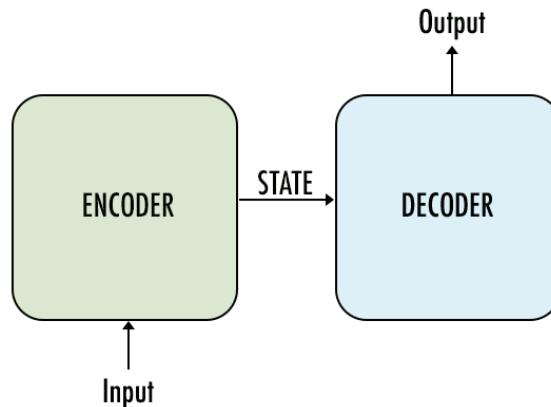


Рисунок 2.5 – Схема архітектури автоенкодера

Використовуючи вбудовування, спочатку треба скласти «словниковий» список, який містить усі слова, які потрібні, щоб модель могла використовувати або читати [16]. Вхідні дані моделі повинні бути тензорами, що містять ідентифікатори слів у послідовності. Однак існує чотири символи, які повинні бути в словнику. Логіка роботи автоенкодера наведена на рис.2.6.

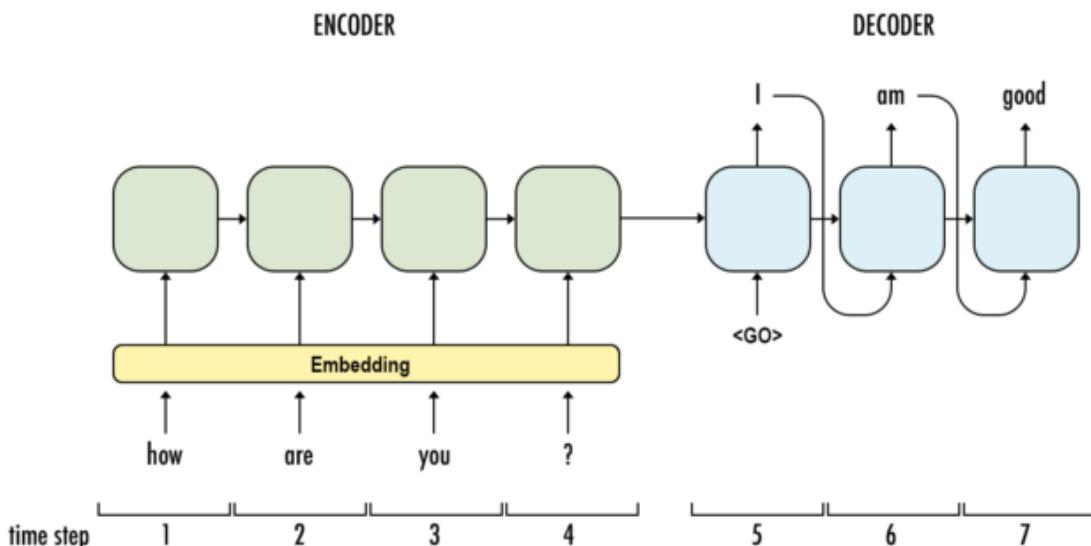


Рисунок 2.6 – Логіка роботи автоенкодера

Словники Seq2seq зазвичай резервують перші чотири місця для цих елементів:

- <PAD>: під час навчання потрібно буде передавати приклади в мережу пакетами. Усі вхідні дані в цих пакетах мають бути однакової ширини, щоб мережа могла виконати обчислення. Однак приклади не мають однакової довжини. Ось чому потрібно буде доповнити коротші вхідні дані, щоб привести їх до однакової ширини пакета
- <EOS>: це ще одна необхідність пакетування, але більше з боку декодера. Це дозволяє повідомляти декодеру, де закінчується речення, а також дозволяє декодеру вказувати те саме у своїх виводах.
- <Nk>: Якщо тренувати модель на реальних даних, виявиться, що можна значно підвищити ефективність ресурсів моделі, ігноруючи слова, які не часто виявляються у словникові запаси, щоб гарантувати розгляд.
- <GO>: Це вхідні дані для первого тимчасового кроку декодера, щоб повідомити декодеру, коли почати генерувати вихідні дані.

Примітка, для представлення цих функцій можна використовувати інші теги. Наприклад <s> i </s> замість <GO> i <EOS>. Тому потрібно переконатися, що все, що використовується, узгоджено через попередню обробку та навчання моделі/висновок.

Підготовка вхідних даних для навчального графіка трохи складніша з двох причин:

1. Ці моделі працюють набагато краще, якщо передати декодеру цільову послідовність, незалежно від того, які її часові кроки фактично виводяться під час тренування. Отже, на відміну від графіка, не потрібно передавати вихідні дані декодера самому собі на наступному часовому етапі.

2. Дозування, одна з оригінальних статей про послідовність дій, Sutskever et al. 2020, повідомляє про кращу продуктивність моделі, якщо вхідні дані змінені. Таким чином, також можна змінити порядок слів у послідовності введення.

Під час попередньої обробки створюється свій словниковий запас з унікальних слів, створити копію розмов із заміною слів на їхні ідентифікатори, можна додати ідентифікатори слів <GO> і <EOS> до цільового набору даних зараз або зробити це під час навчання [16].

2.3 Концепція попередньої обробки та аналізу даних

У практиці використання нейронних мереж використовують різні підходи до нормалізації даних. Але всі вони спрямовані на утримання даних навчальної вибірки та вихідних даних прихованих шарів нейронної мережі в заданому діапазоні та з певними статистичними характеристиками вибірки, такими як дисперсія та медіана. У нейронах мережі застосовуються лінійні перетворення, які у процесі навчання зміщують вибірку у бік антиградієнта.

Розглянемо повнозв'язковий перцептрон з двома прихованими шарами. При прямому проході кожен шар генерує деяку сукупність даних, які є навчальною вибіркою для наступного шару. Результат роботи вихідного шару порівнюється з еталонними даними і зворотному проході поширюється градієнт помилки від вихідного шару через приховані шари до вихідних даних. Отримавши кожному нейроні свій градієнт помилки і оновлюючи вагові коефіцієнти, підлаштовуючи нейронну мережу під навчальні вибірки останнього прямого проходу. Виникає конфлікт: підлаштовуючи другий прихований шар (H_2) під вибірку даних на виході первого прихованого шару (H_1), в той час як змінивши параметри первого прихованого шару змінили масив даних [16]. Підлаштовуючи другий прихований шар під вже неіснуючу вибірку даних. Analogічна ситуація з вихідним шаром, який підлаштовується під вже змінений вихід другого прихованого шару. А якщо ще врахувати спотворення між першим і другим прихованими шарами, масштаби помилки збільшуються. І чим глибша нейронна

мережа, тим сильніший прояв цього ефекту. Це було названо внутрішнім підступним зрушеннем.

У класичних нейронних мережах зазначена проблема частково вирішувалася зменшенням коефіцієнта навчання. Невеликі зміни вагових коефіцієнтів не дуже змінюють розподіл вибірки на виході нейронного шару. Але такий підхід не вирішує масштабування проблеми зі зростанням кількості шарів нейронної мережі та знижує швидкість навчання. Ще одна проблема невеликого коефіцієнта навчання - застрягання в локальних мінімумах.

У лютому 2020 року Sergey Ioffe та Christian Szegedy запропонували метод пакетної нормалізації даних (Batch Normalization) для вирішення проблеми внутрішнього зсуву коваріації. Суть методу полягала у нормалізації кожного окремого нейрона на певному часовому інтервалі зі зміщенням медіани вибірки до нуля та приведенням дисперсії вибірки k_1 [16].

Алгоритм проведення нормалізації наступний. Спочатку за вибіркою даних вважається середнє значення.

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i \quad (2.5)$$

де m – розмір вибірки (batch).

Потім вважаємо дисперсію вихідної вибірки.

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad (2.6)$$

І нормалізуємо дані вибірки, привівши вибірку до нульового середнього та одиничної дисперсії.

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (2.7)$$

В знаменнику до дисперсії вибірки додається константа ϵ , невелике позитивне число з метою виключити поділ на нуль.

Така нормалізація може спроворити вплив вихідних даних. Тому автори методу додали ще один крок — масштабування та усунення. Було введено 2 змінні γ і β , які навчаються разом із нейронною мережею методом зворотного градієнтного спуску.

$$y_i = y\hat{x}_i + \beta \equiv BN_{y,\beta}(x_i) \quad (2.8)$$

Застосування цього методу дозволяє кожному етапі навчання отримувати вибірку даних з однаковим розподілом, що практично робить навчання нейронної мережі більш стабільним і дозволяє збільшити коефіцієнт навчання. Загалом це дозволить підвищити якість навчання за менших витрат часу на навчання нейронної мережі [16].

Але в той же час зростають витрати на зберігання додаткових коефіцієнтів. А також для розрахунку ковзної середньої та дисперсії потрібно зберігання в пам'яті історичних даних кожного нейрона на весь розмір пакету. Нагадаємо формулу експоненційної ковзної середньої.

$$\mu_i = \frac{(m-1)}{m} \mu_{i-1} + \frac{1}{m} x_i \quad (2.9)$$

де m - розмір вибірки (batch), i – ітерація.

Для зближення графіків ковзної дисперсії та експоненційної ковзної дисперсії потрібно трохи більше ітерацій (310-320), але в цілому картина схожа. Що стосується дисперсією застосування експоненціальної дає як економію пам'яті, а й значно знижує кількість обчислень, т.к. для ковзної дисперсії перераховується відхилення від середньої для всього batch-a.

Експерименти, проведені авторами методу, показують, що застосування методу Batch Normalization виступає у ролі регуляризатора [19]. Це дозволяє відмовитися від інших методів регуляризації, зокрема від розглянутого раніше Dropout. Більше того, є пізніші роботи, в яких показано, що спільне використання Dropout та Batch Normalization негативно позначається на результатах навчання нейронної мережі. У сучасних архітектурах нейронних мереж запропонований алгоритм нормалізації можна зустріти у різних варіаціях. Автори пропонують використовувати Batch Normalization безпосередньо перед нелінійністю (формулою активації). Як одну із варіацій даного алгоритму можна розглядати метод Layer Normalization, представлений у липні 2021 року. Головна думка цього розділу в тому, що процес нормалізації вкрай важливий, щоб навчання нейронної мережі проходило ефективно і різні похибки даних не впливали на її роботу.

2.4 Висновок до другого розділу

У другому розділі проведено постановку завдання прогнозування. Зроблено порівняльний аналіз та вибір алгоритму регресії. Розглянуто процес попередньої обробки і аналізу даних. Виконавши аналіз сучасних інформаційних систем, які використовуються в якості головних електронних інструментів в метеорології, було сформовано концепцію розробки програмного засобу.

Таке рішення дозволить розробити сучасну та функціональну систему, а також покращити показники ефективності алгоритмів обробки та аналізу. Так як даних буде багато то програмний комплекс повинен бути оптимізований для роботи з великими даними, в даному випадку кращим рішенням буде використовувати систему управління базами даних SQLite [19].

Отримані в рамках виконання даного розділу результати будуть застосовані під час практичної реалізації програмного забезпечення.

З РОЗРОБКА СИСТЕМИ ОЦІНКИ ЕФЕКТИВНОСТІ НЕЙРОМЕРЕЖ У ЗАДАЧАХ ОБРОБКИ ТА АНАЛІЗУ МЕТЕОДАНИХ

3.1 Розробка проекту інформаційної системи

Початковим етапом при розробці програмного забезпечення є проектування. Проектування дозволяє визначити головні функціональні можливості програмного забезпечення, його архітектуру, складові, а також компоненти з якими взаємодіє програмне забезпечення.

В більшості випадків проект складається з розробки головних діаграм, які в графічному вигляді дозволяють визначити усі складові.

В якості засобу проектування використовується мова UML, яка є спеціалізованою мовою, яку використовують розробники для графічного представлення програмного забезпечення. В якості середовищ проектування може бути використано різні редактори, які дозволяють взаємодіяти зі спеціалізованими об'єктами проектування [20].

Одним із засобів проектування, є створення діаграми варіантів використання, завдяки якій, визначаються усі вимоги до кінцевої системи.

Подібні діаграми дуже зручно створювати за допомогою інструментів моделювання схем, як наприклад Draw.io.

Draw.io – інструмент для створення діаграм, блок-схем, інтелект-карт, бізнес-макетів, відносин сутностей, програмних блоків та іншого. Сервіс розповсюджується на безкоштовній основі з відкритим кодом. Draw.io має багатий набір функцій для візуалізації більшості завдань користувача.

При вході на сервіс користувач одразу потрапляє до робочого інтерфейсу. Користувач не має можливості для авторизації або реєстрації, є лише опція вибору місця для експорту проекту. Текст майданчика Стартпак. Процес створення проекту виглядає так: користувач перетягує з лівої панелі фігури або елементи на робочу поверхню, потім змінює їх – змінює колір, розмір, шрифт тексту, властивості фігури (прозорість, форма тощо). Draw.io дозволяє

відстежувати та відновлювати зміни готових проектів, імпортувати та експортувати до PDF, PNG, XML, VSDX, HTML, а також автоматично публікувати та ділитися роботами.

Інструмент працює з Google Диск, Google Workspace та Dropbox, глибоко інтегрований та зручний для роботи з продуктами Confluence та Jira від Atlassian. Користувачі також можуть працювати з діаграмами в автономному режимі та зберігати їх локально, використовуючи настільний додаток для персональних комп'ютерів.

Інструмент дозволяє створювати: графіки, діаграми, таблиці, презентації, блок-схеми, плани приміщенъ, вирви продажів, ментальні карти, карти сайтів.

Особливості Draw.io:

- більше 500 шаблонів елементів та фігур;
- спрощений інтерфейс, в якому за короткий проміжок часу можна створити готовий проект;
- підтримка гарячих клавіш, задіяних у більшості графічних редакторів;
- експорт у формати: JPG, PNG, SVG, VDSX;
- можливість спільної роботи;
- наявність різноманітних фонових тем;
- мультивідповідний інтерфейс.

Діаграма варіантів використання складається з об'єктів, які мають вигляд овалу, та запису функціональної ролі. Зовнішні об'єкти, якими можуть бути як користувачі так і сервера, зображаються у вигляді чоловічка.

Одним з важливіших елементів на діаграмі варіантів використання є використання стрілок, кожна з яких має своє спеціалізоване призначення. В більшості випадків, саме стрілки, дозволяють розробникам орієнтуватися в програмній системі, а також створювати систему зв'язків елементів між собою[20].

Діаграма варіантів використання для інформаційних системи для збору, обробки та аналізу метеоданих, зображене на рис. 3.1.

Програмне рішення має web-архітектуру, отже дозволяє користувачам отримувати усі функціональні можливості без інсталяції.

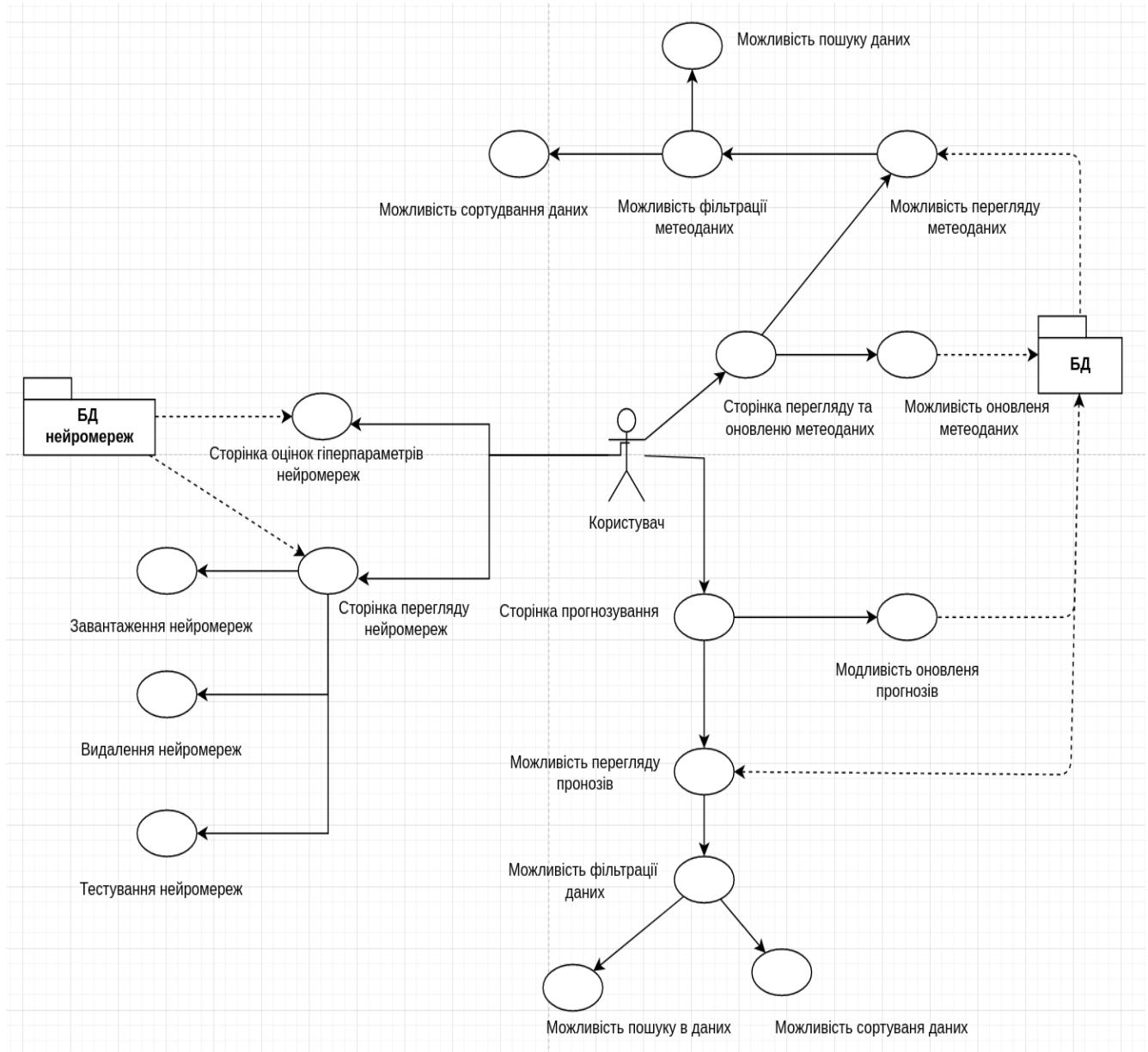


Рисунок 3.1 – Діаграма варіантів використання до інформаційної системи

Після виконання всіх етапів проектування, здійснюється перехід до стадії розробки програмного забезпечення.

Добре обґрунтований проект, дозволяє виявити усі недоліки системи, та виправити їх, ще до початка написання коду [20].

Також, проектування є допоміжним засобом, для подальшого тестування, за рахунок того, що воно поєднує у собі усі особистості роботи системи, та допомагає проаналізувати схожість розробленої системи з проектом[21].

Усі елементи програмного забезпечення є взаємопов'язаними з роботою з метеорологічними даними. Це дозволяє отримати повний огляд метеорологічної ситуації.

Кожен із варіантів використання має спеціалізоване призначення, і має бути описаний індивідуально.

Перегляд завантажених нейромереж. Для перегляду завантажених нейромереж користувачу потрібно в шапці сайту вибрati пункт «Нейромережі». Після переходу на дану сторінку користувач бачить таблицю з усіма завантаженими нейромережами для обробки та аналізу метеоданих. На цій сторінці користувач може завантажити архів з файлами нейромережі, для цього треба вказати яку функцію виконує нейромережа, обробка метеоданих чи їх аналіз, та додати опис її шарів. Після завантаження нейромережі вона підлягає тестуванню в системі, після чого результати оцінки ефективності системи будуть відображені на сторінці.

Перегляд зібраних метеоданих. Для перегляду зібраних метеоданих користувачу потрібно в шапці сайту вибрati пункт «метеодані». Після переходу на дану сторінку користувач бачить таблицю з усіма зібраними метеоданими, доступна функція фільтрації, пошуку та сортування.

Перегляд розрахованих прогнозів. Для перегляду розрахованих прогнозів користувачу потрібно в шапці сайту вибрati пункт «перегляд прогнозів». Після переходу на цю сторінку користувач бачить таблицю з усіма прогнозами метеоданих та погодних явищ, доступна функція фільтрації, пошуку та сортування.

Перегляд даних в таблицях. Користувач може переглядати дані виведені в таблиці на сторінках сервісу. Перший рядок таблиці це рядок з іменами стовпчиків, при натисканні на ім'я виконується сортування таблиці в алфавітному

порядку за цим ім'ям. При повторному натисканні буде відображене в зворотному алфавітному порядку.

Пошук. Є розширенням інформаційної системи та дозволяє відшукувати данні в таблицях за будь-яким ключем. Ключем може бути будь-який текст, будь-то цифри, символи чи букви.

Оновлення метеоданих. На сторінці перегляду метеоданих користувач може натиснути кнопку оновлення метеоданих, це придасть сигнал на сервер та він почне викачувати нові метеодані з ресурсів. Після оновлення система запускає алгоритм пошуку аномалій в цих даних, далі на вилучених та «чистих» даних система робить прогнозування метеоданих на наступні 3 доби і вже по цим метеоданих система прогнозує погодні явища. Усі данні які оброблюються на даному етапі записуються в базу даних.

Оновлення прогнозу. На сторінці перегляду прогнозів користувач має можливість оновити зроблений системою прогноз, ця функція потрібна для тестування алгоритму прогнозування, та використовується коли нових даних не має або сервер оновив нейромережі які працювали з даними.

Можливість проведення тестування ефективності навчання нейромереж на метеоданих. Процес тестування починається з того що екземпляр нейромережі завантажується в систему, завантажуються дані для тестування, після цього нейромережа оцінюється декількома метриками, детальний розгляд яких може допомогти у вдосконалені нейромережі.

Тестування нейромережі проходить і в тому що система оцінює ефективність виконання задач аналізу та обробки метеоданих, завдяки використанню нейромереж система може оцінити важливість вхідних гіперпараметрів. Це дає можливість дізнатися які гіперпараметри найбільше впливатимуть на результат роботи нейромережі, та які з них в цілому важливі для задачі. Такий підхід використовується в двох випадках, коли треба зробити аналіз нейромережі для її подальшої оптимізації, та коли треба оцінити взаємозв'язок гіперпараметрів, в цьому випадку можна більш детально проаналізувати як повинна працювати модель вирішуваного завдання. Другий випадок все частіше

призводить до дивовижних відкриттів які надають все більше інформації про те, як працює цей світ.

Після визначення усіх функціональних можливостей програмного забезпечення, є необхідність виконати побудову головних класів [21]. Діаграма класів дозволяє визначити розробникам усі складові елементи системи, та їх особливості роботи. В більшості випадків діаграма класів будується до початка розробки, а потім модифікується, в залежності від зміни умов, або функціоналу програмного забезпечення.

Для ефективного створення діаграм класів можна використовувати інструмент для створення UML діаграм PlanttextUML.

PlantText — онлайн-інструмент, який швидко створює зображення з тексту. В першу чергу він використовується для створення діаграм UML (Unified Modeling Language). На відміну від більшості інструментів, PlantText може будувати діаграми без використання миші, оскільки він покладається на текстову мову під назвою PlantUML.

За допомогою PlantText архітектори програмного забезпечення можуть створювати діаграми UML простою мовою, не відволікаючись на естетичні деталі чи використання миші. Просто введіть PlantUML у редакторі та оновіть екран, щоб створити професійну чітку діаграму.

Приклад коду завдяки якого сервісом PlantUML можна згенерувати діаграму класів.

```
@startuml
class MainMenu{
    ForecastModel fnet
    SummaryModel snet
    Tester tester
    test(net_id)
    update_forecast()
    update_meteodata()
    get_top_labels(model)
}
class Tester{
    NNOBJ obj
    void test(net_id)
```

```

void hyperparameters_importance_test(net_id)
void update_data(test_result)
}
class Parser{
List<string> parse(date limit)
void saveMeteodata()
}
class NNBase{
NNBase NNBase()
void ~NNBase()
void load_net()
void load_data_set()
void save_net()
NNOBJ build_rnn_net()
NNOBJ build_perc_net()
}
class ForecastModel{
List<Dict> predict(List<Dict> data)
List<Dict> encode(List<Dict> data)
List<Dict> decode(List<Dict> data)
string test(net_id)
}
class SummaryModel{
List<Dict> predict(List<Dict> data)
List<Dict> encode(List<Dict> data)
List<Dict> decode(List<Dict> data)
string test(net_id)
}
class DB{}
class Views{}

note right of DB
data base api from django framework
end note

note right of Views
views api from django framework
end note

MainMenu --> Tester
MainMenu --> Parser
MainMenu --> ForecastModel
MainMenu --> SummaryModel
ForecastModel .down.> NNBase
SummaryModel .down.> NNBase

DB .up.> Parser
DB .up.> NNBase
DB .up.> Tester
Views --> MainMenu
@enduml

```

Інформаційну систему можна розділити на два головних компоненти, кожен з них був відображенний в діаграмі класів. Компонент інтерфейсу та контролеру та компонент роботи з алгоритмами обробки та нейромережами [22]. Діаграма класів компонентів, зображене на рис. 3.2.

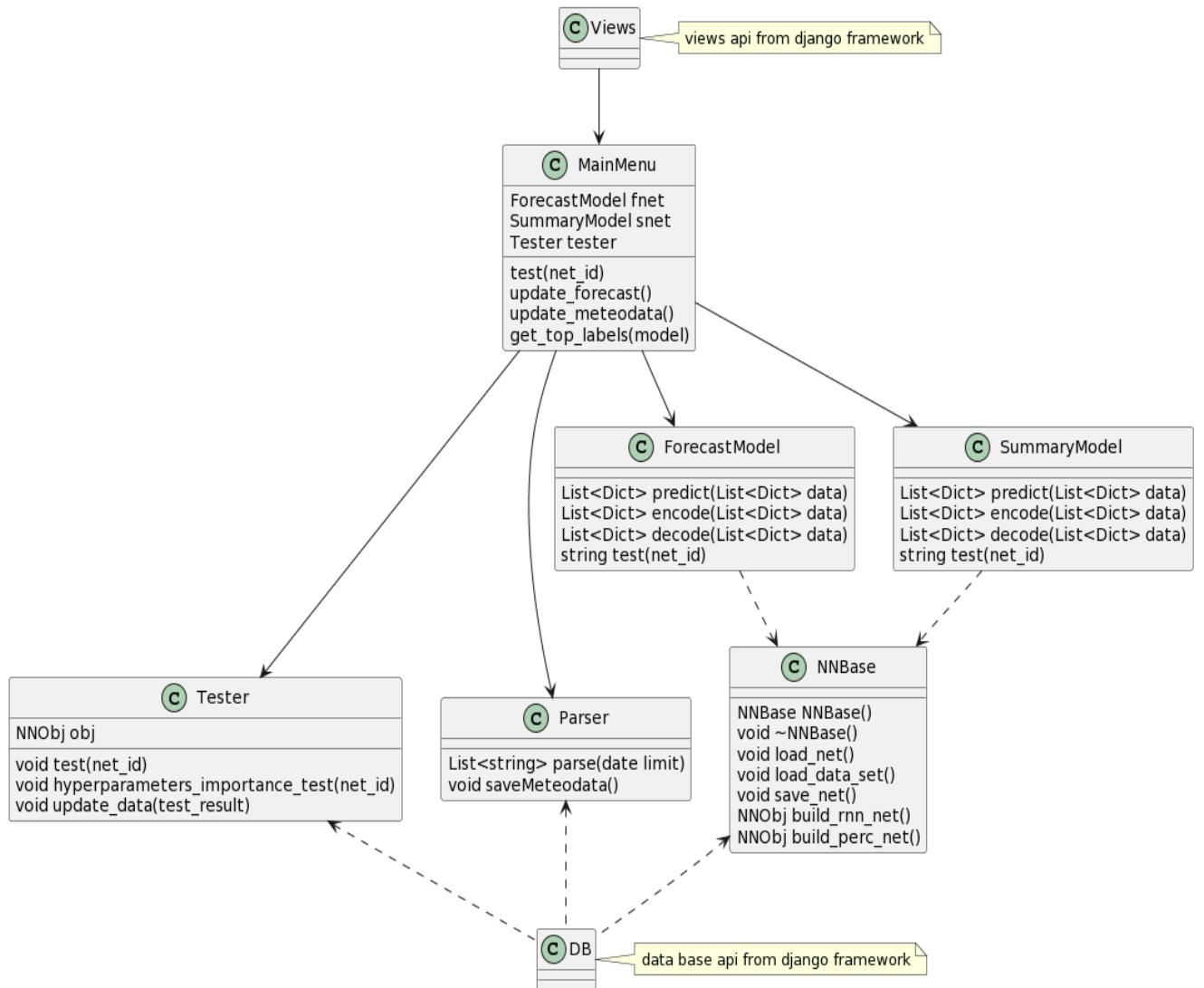


Рисунок 3.2 – Діаграма класів системи

На діаграмі зображену функціональну залежність компонентів. У зв'язку з тим, що це web-система то простір класів розроблюється з точки зору окремих компонентів.

В залежності від типу проекту діаграма класів може відрізнятися одна від одної, але в більшості випадків вона має єдину логічну структуру.

Також на діаграмі визначається залежність компонентів один від одного та вказується система зв'язку між ними.

Реляційні зв'язки є досить суттєвими та необхідні для функціонального аналізу проекту системи.

На цій діаграмі можна розглянути усі задіяні класи. В першу чергу варто описати класи, що надаються фреймворком безпосередньо, вони використовуються в двох класах оболонках, це View і DB. Клас View, це клас, що поєднує в собі всі класи сторінок, на які переходить користувач. У них описана логіка того, як сторінка повинна бути відображенна і як користувач може з нею взаємодіяти. Наприклад, які сигнали від нього сторінка може приймати і як обробляти дані з форм. Клас DB надає зручний інтерфейс для отримання, оновлення та внесення даних до бази даних, має функції сортування, фільтрації та іншого, цей функціонал повністю реалізований на стороні фреймворку. Однак функціонал для збереження та запиту, що завантажується користувачем файлу нейронної мережі, розробляється окремо.

Компонент контролера складається з одного класу MainMenu, який пов'язує в собі логіку передачі сигналів від користувача до компонентів системи, в ньому ініціалізуються об'єкт класу для тестування нейромереж та об'єкти класів самих нейромереж, а також клас збирача метеоданих.

Клас Parser реалізує функціонал збору даних з зовнішніх джерел, він використовує [https](https://) запити та бібліотеку BS4 для парсингу.

Компонент алгоритмів розрахунку та нейромереж складається з класів NNBase, SummaryModel та ForecastModel. NNBase це базовий клас для всієї моделей нейромереж, у ньому реалізовані методи для збереження та завантаження нейромереж, їх складання та завантаження даних для них. Клас SummaryModel реалізує функціонал оцінки ймовірності тих чи інших кліматичних явищ при вхідних метеоданих. Також у ньому реалізовані методи нормалізації даних, метод тестування ефективності нейромережі під час використання різних оціночних

метрик і методи прогнозування. Клас ForecastModel вирішує задачу прогнозування значень метеорологічних змінних на наступний проміжок часу.

Модуль Tester використовує відповідний клас для проведення операцій з оцінки нейромереж. У класі реалізовані методи для стандартної оцінки роботи нейромережі за різними метриками, а також методи для більш глибокої оцінки внутрішніх ваг нейронів у кожному шарі нейромережі. За підсумками способу оцінки терезів проводиться аналіз важливості гіперпараметрів і точніше зрозуміти залежність з-поміж них.

Таким чином можна обчислити ті залежності про які раніше не було підозр, а також виключити із системи ті метеорологічні змінні, які мінімально впливають на результат і включити до неї інші, більш значущі.

Для більш покращеного розуміння алгоритмічного складу розробленого програмного забезпечення, використовується побудова алгоритмічних блок-схем які дозволяють зрозуміти користувачам та розробникам, як саме працює програмне забезпечення [22].

Блок-схема представлена в даному розділі – це схема обробки отриманих даних і побудова прогнозу на слухні три доби, рис. 3.3, кожна замість прогнозу містить прогнозовану інформацію про погоду і прогнозоване за цими даними погодне явище, різниця між записами о третій годині часу.

Кроки по блок-схемі наступні, вичитування даних з обох списків, далі підготовка нейромереж під ці дані і складання прогнозу метеоданих, слідом інша нейромережа за прогнозом робить прогноз погодного явища, в процесі системи аналізує точність роботи всіх нейромереж і порівнює дані їх точності [22].

Ці результати потрібні для основної мети даної роботи, аналіз впливу систем пошуку аномалій в метеорологію шляхом порівняння точності алгоритмів прогнозування навчених на даних з аномаліями і без. Алгоритм представлений з додатковими оптимізаціями, внаслідок чого алгоритм простий для розуміння і при тому швидкий та зручний в використані [23].

Інші алгоритми використанні в системі більш прості та не потребують додаткового зображення у вигляді блочної схеми.

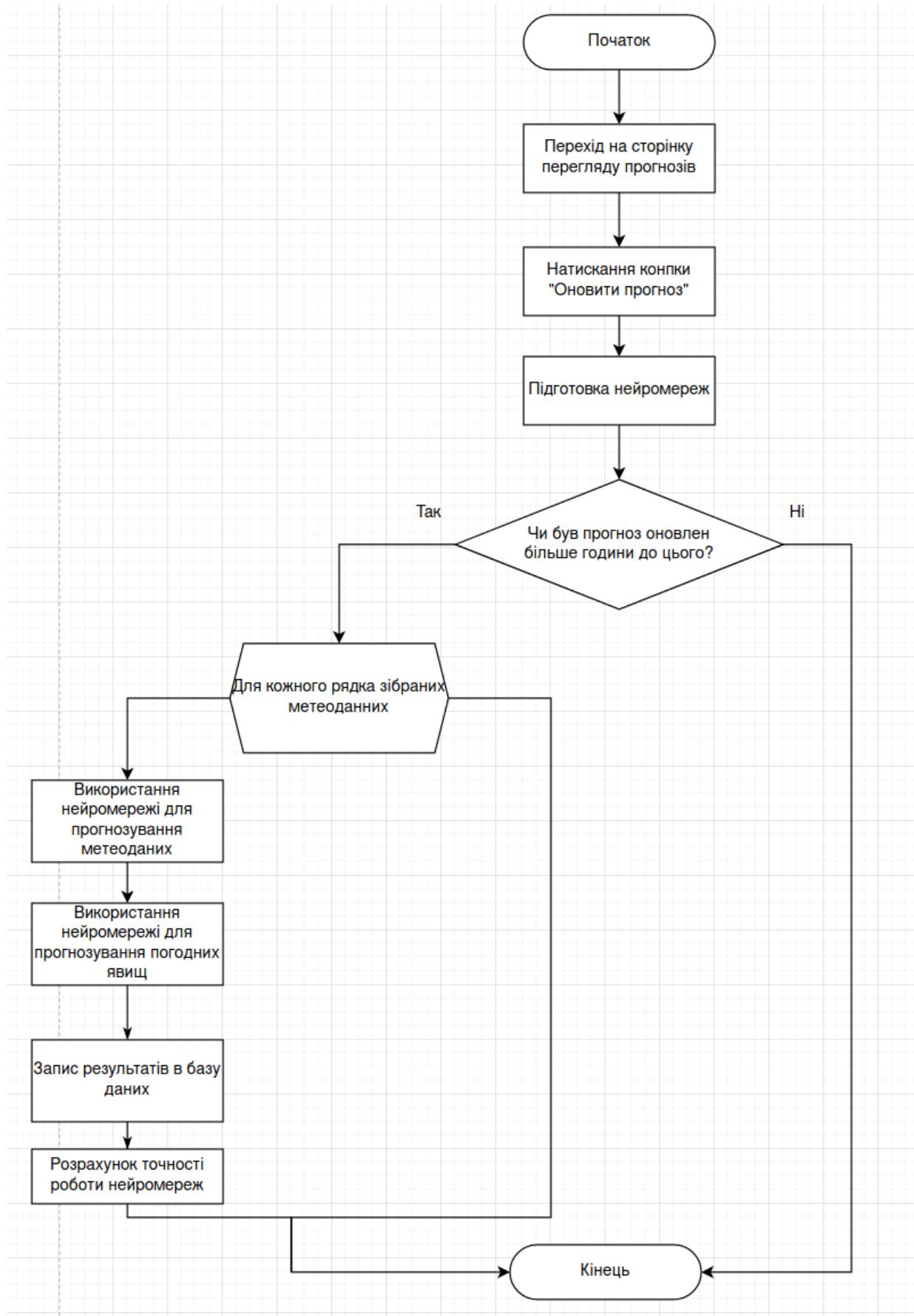


Рисунок 3.3 – Схема обробки отриманих даних і побудова прогнозу

Це алгоритми пошуку, фільтрації, сортування, вичитки даних, вичитки даних з зовнішніх джерел, роботи з базою даних та розміщення даних та збору сторінок виводу інформації.

На базі отриманих результатів стає можливою подальша розробка сховища для зберігання даних.

3.2 Опис структури та бази даних інформаційної системи

Для опису структури самого проекту в першу чергу варто уточнити, що цей веб-додаток створений на основі фреймворку Django [23].

Специфіка проектів побудованих на ньому в тому, що в перший модуль який описується це модуль з сторінками користувача, моделями таблиць даних, формами.

Далі модуль в якому описуються шаблони HTML які і будуть використовуватися для складання сторінок користувача.

Останній і ключовий модуль це модуль реалізації функціонала користувача. Наприклад функціонал збору даних, їх аналізу та обробки, складання прогнозу та тестування нейронних мереж.

Під час розробки інформаційної системи було визначено перелік таблиць та їх структуру, які є необхідними для доброго функціонування усієї системи.

Ці таблиці складають структуру організації даних в базі з якою і працює система.

Кожна таблиця складається з стовпців, кожен з яких має своє ім'я та тип даних, це частини даних які записуються в базу. Кожен рядок таблиці це окремих запис з даних.

Однією з перших таблиць, яка була розроблена є таблиця метеорологічних даних, рис. 3.4.

	Id [PK] bigint	datetime timestamp with time zone	place bigint	placeName character varying (80)	wind_way bigint	wind_speed double precision	air_pressure double precision	water_pressure double precision	weather text	temperature double precision
1	4069709	2017-11-01 02:00:00+02	33287	Рава Русская	2	0	760	760	{Нет}	3.8321031331816076
2	4069710	2017-11-01 05:00:00+02	33287	Рава Русская	2	0	760	760	{Нет}	3.8321031331816076
3	4069711	2017-11-01 08:00:00+02	33287	Рава Русская	2	0	760	760	{Нет}	3.8321031331816076
4	4069712	2017-11-01 11:00:00+02	33287	Рава Русская	2	0	760	760	{Морось (н...	3.8321031331816076
5	4069713	2017-11-01 14:00:00+02	33287	Рава Русская	2	0	760	760	{Количество...	3.8321031331816076
6	4069714	2017-11-01 17:00:00+02	33287	Рава Русская	2	0	760	760	{Нет}	3.8321031331816076
7	4069715	2017-11-01 20:00:00+02	33287	Рава Русская	2	0	760	760	{Нет}	3.8321031331816076
8	4069716	2017-11-01 23:00:00+02	33287	Рава Русская	2	0	760	760	{Нет}	3.8321031331816076
9	4069717	2017-11-02 02:00:00+02	33287	Рава Русская	2	0	760	760	{Ливневой ...	3.8321031331816076
10	4069718	2017-11-02 05:00:00+02	33287	Рава Русская	2	0	760	760	{Ливневой ...	3.8321031331816076
11	4069719	2017-11-02 08:00:00+02	33287	Рава Русская	2	0	760	760	{Количество...	3.8321031331816076
12	4069720	2017-11-02 11:00:00+02	33287	Рава Русская	1	0	760	760	{Нет}	3.8321031331816076
13	4069721	2017-11-02 14:00:00+02	33287	Рава Русская	2	0	760	760	{Нет}	3.8321031331816076
14	4069722	2017-11-02 17:00:00+02	33287	Рава Русская	1	0	760	760	{Ливневой ...	3.8321031331816076
15	4069723	2017-11-02 20:00:00+02	33287	Рава Русская	2	0	760	760	{Ливневой ...	3.8321031331816076
16	4069724	2017-11-02 23:00:00+02	33287	Рава Русская	2	0	760	760	{Ливневой ...	3.8321031331816076
17	4069725	2017-11-03 02:00:00+02	33287	Рава Русская	2	0	760	760	{Ливневой ...	3.8321031331816076

Рисунок 3.4 – Вікно таблиці метеоданих в базі даних

Дана таблиця створена засобами фреймворка Django і виконується наступним кодом.

```
class Meteodata(models.Model):
    datetime = models.DateTimeField("Date_and_time")
    place = models.BigIntegerField("Place_id")
    place_name = models.CharField("Place_name", max_length=80)
    temperature = models.FloatField("Air temperature")
    wind_way = models.BigIntegerField("Wind way")
    wind_speed = models.FloatField("Wind speed")
    air_pressure = models.FloatField("Pressure")
    water_pressure = models.FloatField("Sea_level_pressure")
    weather = models.TextField("Weather")
    class Meta:
        verbose_name = "Метеоданные"
        verbose_name_plural = "Метеоданные"
```

Детально по кожному полю таблиці метеоданих:

- Datetime це підлозі в якому зберігається дата і час коли було зроблено запис.

- Place це поле в якому зберігається назва регіону в якому був зроблений запис.
- Place_name це поле в якому зберігається номер регіону в якому був зроблений запис.
- temperature поле температури яка була зафікована в момент запису.
- wind_way поле зберігає напрямок вітру.
- wind_speed поле зберігає значення швидкості вітру.
- air_pressure поле зберігає значення атмосферного тиску.
- water_pressure поле зберігає значення атмосферного тиску на рівні моря.
- weather поле зберігає погодне явище відбувалося в момент запису.

Таблиця даних про тестування нейронних мереж описана наступним кодом, вона містить поля для зберігання результату оцінки гіперпараметрів нейромережі та супутні дані для її ідентифікації.

```
class Test(models.Model):
    neuralnet_id = models.BigIntegerField("NeuralNet_id")
    datetime = models.DateTimeField("Date_and_time")
    conclusion = models.TextField("Conclusion")

    def __str__(self):
        return "( {}) members: {} ".format(self.pk,
                                             self.test_records)

    class Meta:
        verbose_name = "Test"
        verbose_name_plural = "Tests"
```

Детально по кожному полю таблиці тестування:

- neuralnet_id поле зберігає значення ідентифікаційного номеру протестованої нейромережі;
- datetime поле зберігає значення дати та часу коли проходило відповідне тестування;
- conclusion поле зберігає значення вироку системи щодо важливості гіперпараметрів нейромережі.

Таблиця даних про збережені нейромережі описана наступним кодом. У ній описані всі поля, необхідні для збереження файлу нейромережі та її ідентифікації..

```
class NeuralNet(models.Model):

    name = models.CharField("Name", max_length=80)
    file_data=models.FileField('NetFile', upload_to='models/')
    target=models.BigIntegerField("Target")
    metric= models.BigIntegerField("Metric")
    description = models.TextField("Description")
    conclusion = models.TextField("Conclusion")

    def __str__(self):
        return "({})--{}".format(self.pk, self.name)

    def delete(self, *args, **kwargs):
        self.file_data.delete()
        super().delete(*args, **kwargs)

class Meta:
    verbose_name = "NeuralNet"
    verbose_name_plural = "NeuralNet"
```

Детально по кожному полю таблиці нейромереж:

- name поле зберігає назvu нейромережі;
- file_data поле зберігає архівний файл нейромережі;
- target поле зберігає яку задачу виконує нейромережа в системі;
- metric поле зберігає яка метрика використовувалась для тренування нейромережі;
- description поле зберігає опис всіх шарів нейромережі;
- conclusion поле зберігає оцінку при тестуванні нейромережі за допомогою декількох метрик.

Таблиця даних прогнозованих метеоданих містить усі необхідні поля для збереження метеоданих оброблених та проаналізованих системою.

```
class ForecastMeteodata(models.Model):
    datetime = models.DateTimeField("Date_and_time")
    place = models.BigIntegerField("Place_id")
    place_name = models.CharField("Place_name", max_length=80)
    temperature = models.FloatField("Air temperature")
```

```

wind_way = models.BigIntegerField("Wind way")
wind_speed = models.FloatField("Wind speed")
air_pressure = models.FloatField("Pressure")
water_pressure = models.FloatField("Sea_level_pressure")
weather = models.TextField("Weather")

def __str__(self):
    return self.pk

class Meta:
    verbose_name = "Forecast"
    verbose_name_plural = "Forecast"

```

Детально по кожному полю таблиці прогнозованих метеоданих:

- Datetime це підлозі в якому зберігається дата і час коли було зроблено запис.
- Place це поле в якому зберігається назва регіону в якому був зроблений запис.
- Place_name це поле в якому зберігається номер регіону в якому був зроблений запис.
- temperature поле температури яка була зафікована в момент запису.
- wind_way поле зберігає напрямок вітру.
- wind_speed поле зберігає значення швидкості вітру.
- air_pressure поле зберігає значення атмосферного тиску.
- water_pressure поле зберігає значення атмосферного тиску на рівні моря.
- weather поле зберігає погодне явище відбувалося в момент запису.

Також варто звернути увагу що були отримані раніше таблиці є додатковими, крім них Django створює ради основних таблиць, це таблиці зберігають в собі дані про підключених до системи з'єднань, їх унікальні коди, паролі адміністратора, таблиці логування дій в системі, таблиця логування помилок в системі, таблиця підключених модулів і т.д.

База даних вкрай чутливий елемент системи і потребує до себе великої уваги [24]. Так як система працює з великими даними повинна бути можливість легко взаємодіяти з нею, здійснювати запис в базу даних з високою швидкістю.

Для цього розроблені таблиці даних повинні бути схожі, а дані в них однорідні. Дане завдання оптимізації процесів вирішується на рівні програмного інтерфейсу надається фреймворком Django 3, а також додатковими бібліотеками, які максимально спрощують створення системи.

Як і було позначено раніше Django генерує таблиці в базі даних, для цього використовується python код який описує структуру яку повинна мати таблиця. Для цього використовується наступні типи даних:

- BinaryField. Створює BLOB-поле для зберігання двійкових даних (наприклад, зображень, аудіо чи інших мультимедійних об'єктів)
- BooleanField. Створює логічне поле для зберігання значень True / False (або 0/1)
- DateField. Створює поле дати для зберігання дат.
- FloatField. Створює стовпець для зберігання чисел з плаваючою комою.
- TextField. Створює текстове поле для зберігання тексту.
- BigIntegerField. Створює велике ціле число, щоб вмістити числа від – 9223372036854775808 до 9223372036854775807. Цей діапазон може змінюватися залежно від марки БД
- EmailField. Примушує текст – це дійсний електронний лист із внутрішнім Django EmailValidator, щоб визначити, що є, а що не є дійсним. Працює так само, як за замовчуванням CharField має максимальну довжину 254 символи, а також примушує рядок – це дійсний електронний лист.

Для аналізу ієархії таблиць бази даних іноді використовуються діаграм зв'язку таблиць, це необов'язкові метод аналізу який потрібен тільки для наочності та полегшеного розуміння структури бази даних [24]. Детально описуючи його можна виділити наступне.

Всередині діаграми бази даних у кожній зв'язку є три окремих елемента: кінцеві точки, стиль лінії і зв'язані таблиці.

Кінцеві точки. Кінцеві точки лінії показують вид зв'язку: "один до одного" або "один до багатьох".

Якщо на одній кінцевій точці зв'язку знаходиться ключ, а на іншій – цифра вісім, то це зв'язок «один до багатьох». Якщо в зв'язку по одному ключу на кожній кінцевій точці, то це зв'язок «один до одного».

Стиль лінії. Різновид лінії (не її кінцеві точки) показує, перевіряє чи СКБД посилальну цілісність для зв'язку при додаванні нових даних в таблицю, пов'язану з допомогою зовнішнього ключа.

Якщо зв'язок намальована у вигляді суцільної лінії, це означає, що СКБД перевіряє посилальну цілісність для зв'язку при додаванні або зміні рядків у таблиці, пов'язаної з допомогою зовнішнього ключа.

Якщо пунктирна лінія, це означає, що СКБД не перевіряє посилальну цілісність для зв'язку при додаванні або зміні рядків у таблиці, пов'язаної з допомогою зовнішнього ключа.

Зв'язані таблиці. Лінія зв'язку показує, що дві таблиці, зв'язані з допомогою зовнішнього ключа.

Для зв'язку «один до багатьох» таблиця, зв'язана з допомогою ключа, – це таблиця близько цифри на лінії.

Якщо обидві кінцеві точки лінії приєднані до однієї таблиці, це означає зворотну зв'язок.

Для будь-якої бази даних можна створити будь-яку необхідну кількість діаграм; кожна з таблиць бази даних може використовуватися у будь-якій кількості діаграм [24].

Таким чином, для візуалізації різних частин бази даних або для акцентування різних аспектів її конструювання можна створювати різні діаграми. Наприклад, можна створити велику діаграму, в якій будуть відображатися всі таблиці і стовпчики, а також меншу діаграму, в якій будуть відображатися всі таблиці, але не стовпців.

Всередині діаграми бази даних кожна таблиця має три окремих елементів: заголовка вікна, список вибору рядків і набір стовпців властивостей. Рядок заголовка в рядку заголовка відображається ім'я таблиці.

Якщо таблицю було змінено, але ще не збережено, то після імені таблиці з'являється зірочка (*), що показує наявність незбережених змін. Область виділення рядка.

Щоб вибрати стовпець бази даних в таблиці, клацніть область виділення рядка. Якщо стовпець є первинним ключем таблиці, то в цьому списку відображається символ ключа. Стовпці властивостей. Набір стовпців властивостей видно на поданні таблиці. Таблицю можна переглянути в будь-якому з п'яти різних уявлень, що дозволяють підібрати підходящий розмір і розміщення елементів діаграми.

3.3 Опис розробленого інтерфейсу до інформаційної системи

Інформаційну систему розроблено в кращих традиціях web-архітектури, за рахунок чого сайт складається з адміністраторської частини та клієнтської [25].

Адміністратор системи може додавати будь-яку інформацію на сайт, змінювати поточну, адмініструвати облікові записи та інше.

Панель адміністратора виконана у вигляді переліку, що спрощує навігацію, та дозволяє поступово виконувати різні дії з контентом.

Головне вікно панелі адміністратора зображене на рис. 3.5.

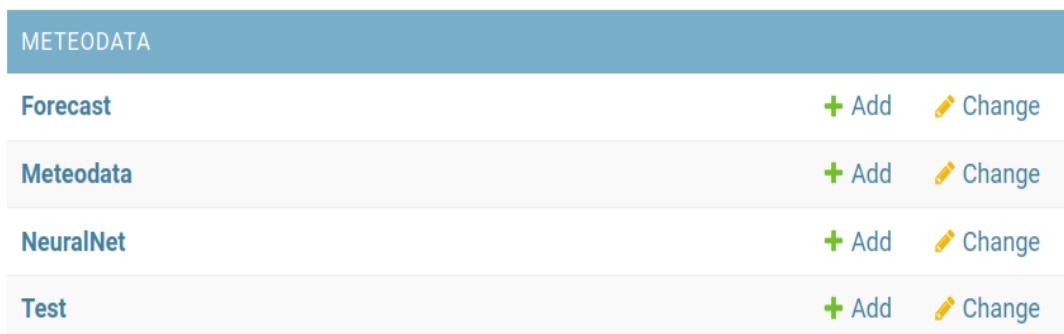


Рисунок 3.5 – Вікно панелі адміністратора

Головними складовими функціями адміністратора є додання нового контенту.

Для цього розроблено спеціалізовані форми, які дозволяють спрощувати цей процес та роботи його зрозумілим для кожного, рис. 3.6.

Це виконано з розрахунку на те, що в більшості метеорологічних центрів, адмініструванням контенту займаються прості метеорологи, які не мають спеціалізованого досвіду роботи з адмініструванням сайтів [26].

Отже це було одним з головних критеріїв під час розробки панелі адміністратора.

Вона повинна бути як можливо спрощеною та відповідати усім вимогам до відповідного програмного засобу.

Форма додання нового запису метеоданих включає в себе:

- Id це поле целочисленном типу зберігає унікальний номер запису;
- Datetime це підлозі в якому зберігається дата і времм коли було зроблено запис;
- Place це поле в якому зберігається назва регіону в якому був зроблений запис;
- placeNumber це поле в якому зберігається номер регіону в якому був зроблений запис;
- temperature поле температури яка була зафіксована в момент запису;
- wind_way поле зберігає напрямок вітру;
- wind_speed поле зберігає значення швидкості вітру;
- air_pressure поле зберігає значення атмосферного тиску;
- water_pressure поле зберігає значення атмосферного тиску на рівні моря.

Основні джерела містять інформацію подібному форматі, різні інтернет ресурси закордонних спеціалізованих метеоцентри, що базуються в провідних університетах і великих компаніях [26]. Це оптимальний набір даних для аналізу системою, який дає можливість легко збирати дані. Також різні архіви метеоданих містять всі вибрані для аналізу даної системою дані, що є важливим критерієм

вибору, бо дає доступ до майже нескінченним ресурсів зборів Європейського метеоцентри.

The screenshot shows a web-based form titled "Add Meteodata". The form is designed for inputting meteorological data. It includes fields for date and time (with "Today" and "Now" buttons), place identification, place name, air temperature, wind direction, wind speed, pressure, sea level pressure, and weather conditions. There are also buttons for saving the data or adding another entry.

Рисунок 3.6 – Вікно додання нового запису метеоданих

Так само в системі є форми для додавання нових нейронних мереж та інших даних, рис 3.7 і рис 3.8.

Якщо необхідно відредактувати вже існуючий запис, то його форма буде максимально відповідати нормам змін. Усі додані до системи записи виводяться користувачам у вигляді переліків, що є нормою для перегляду великої кількості контенту схожого типу. Вікно форми редагування інформації зображено на рис. 3.9.

Add NeuralNet

Name:	<input type="text"/>
NetFile:	<input type="button" value="Browse..."/> No file selected.
Target:	<input type="button"/>
Metric:	<input type="button"/>
Description:	<input type="text"/>
Conclusion:	<input type="text"/>
<input type="button" value="Save and add another"/> <input type="button" value="Save and continue editing"/> <input type="button" value="SAVE"/>	

Рисунок 3.7 – Вікно додання нового запису нейромереж

Add Test

NeuralNet_id:	<input type="text"/>
Date_and_time:	Date: <input type="text"/> Today <input type="button"/> Time: <input type="text"/> Now <input type="button"/> <small>Note: You are 1 hour ahead of server time.</small>
Conclusion:	<input type="text"/>
<input type="button" value="Save and add another"/> <input type="button" value="Save and continue editing"/> <input type="button" value="SAVE"/>	

Рисунок 3.8 – Вікно додання нового запису о тестуванні

The screenshot shows the Django administration interface. The top navigation bar says "Django administration". Below it, the breadcrumb navigation shows "Home > Meteodata > Meteodata". On the left, there's a sidebar with "AUTHENTICATION AND AUTHORIZATION" and "METEODATA" sections. Under "METEODATA", "Meteodata" is highlighted in yellow, while other items like "Forecast", "NeuralNet", and "Test" are white. Each item has a "+ Add" button. The main content area is titled "Select Meteodata to change". It includes a "Action:" dropdown and a "Go" button, followed by a message "0 of 100 selected". A list of 20 meteodata entries is shown, each with a checkbox and a link: METEODATA, 2881937, 2881936, 2881935, 2881934, 2881933, 2881932, 2881931, 2881930, 2881929, 2881928, 2881927, 2881926, 2881925, and 2881924.

Рисунок 3.9 – Список існуючих таблиць та список метеоданих в панелі адміністратора

Усі інші форми редагування чи додання інформації розроблені аналогічним образом. Отже їх опис не є обов'язковим. Після розгляду функціональної складової панелі адміністратора, необхідно зробити опис користувацької частини.

Інтерфейс сайту виконаний в стилі подібному месенджер Telegram, який є еталонним з боку простоти дизайну і його приємності [28].

Стиль спокійний і «однозначний», його особливості в тому що користувачу зрозуміло як користуватися елементами інтерфейсу і яких дій від них чекати.

Базова сторінка є сторінкою перегляду метеоданих, вона складається з декількох загальних елементів.

Перший елемент необхідний для роботи з системою це верхня панель переходу між сторінками.

На цій панелі розташовані три кнопки для переходу між сторінками перегляду метеоданих, перегляду прогнозів і перегляду аномалій шляхом натискання кнопок «перегляд метеоданих», «прогнозування метеоданих» і «Аналіз метеоданих» відповідно.

Кнопка «Оновити дані» запускає основний алгоритм обробки даних, це основна кнопка по роботі з системою, рис. 3.10.



Рисунок 3.10 – Приклад зовнішнього вигляду верхній панелі управління

Другим важливим елементом на кожній сторінці є основний блок, в ньому може розташовуватися найрізноманітніша інформація, в даному випадку в ньому розташовані таблиці з даними для вільного та відповідного їх перегляду користувачем [28].

Використовувані таблиці мають панель управління, кожна робоча таблиця своя і саму таблицю в якій відображені дані.

Дані в таблиці розбиті по стовпцях з іменами параметрів виведені в рядках нижче, при натисканні на назив параметра проводиться сортування, а при подвійному натисканні зворотна сортування по параметру тексту, утримуючи кнопки заголовка.

Інформація в таблиці виводиться в своїх окремих осередках, в центрі кожної з яких розміщений текст зі значенням параметру стовпця в цьому рядку записи, рис.3.11.

ID	Date_and_time	Place_id	Place_name	Air temperature	Wind way	Wind speed	Pressure	Sea_level_pressure	Weather
1	March 1, 2010, 2 a.m.	33562	Вінниця	-0.3	7	6.0	730.1	757.6	дъмка
2	March 1, 2010, 5 a.m.	33562	Вінниця	-0.1	7	6.0	728.6	756.0	замерзающая морось или дождь
3	March 1, 2010, 8 a.m.	33562	Вінниця	0.5	0	7.0	728.2	755.6	туман без изменений, небо не видно
4	March 1, 2010, 11 a.m.	33562	Вінниця	1.0	0	6.0	728.3	755.7	туман без изменений, небо не видно
5	March 1, 2010, 2 p.m.	33562	Вінниця	1.7	0	4.0	728.2	755.5	туман без изменений, небо не видно
6	March 1, 2010, 5 p.m.	33562	Вінниця	2.6	1	4.0	728.1	755.2	дъмка
7	March 1, 2010, 8 p.m.	33562	Вінниця	2.2	0	3.0	728.8	756.0	дъмка
8	March 1, 2010, 11 p.m.	33562	Вінниця	1.6	0	2.0	730.1	757.5	дъмка
9	March 2, 2010, 2 a.m.	33562	Вінниця	0.6	1	3.0	731.2	758.7	дъмка
10	March 2, 2010, 5 a.m.	33562	Вінниця	0.4	1	3.0	731.7	759.2	0
11	March 2, 2010, 8 a.m.	33562	Вінниця	-0.1	1	3.0	732.2	759.8	0
12	March 2, 2010, 11 a.m.	33562	Вінниця	4.0	2	6.0	733.4	760.6	0

Рисунок 3.11 – Таблиця для відображення даних

Панель управління таблицею складається з рядка відображення поточної сторінки і максимального номера станиці. Даний елемент являє собою акуратну рядок тексту.

Також на панелі управління розташована рядок пошуку і фільтрації даних, що виводяться і кнопка застосувати.

Частково прозора рядок введення необхідна для пошуку в даних і їх фільтрації, для цього достатньо ввести якийсь текст і натиснути кнопку застосувати.

Пошук виконується за всіма параметрами записи, рис. 3.12.

Рисунок 3.12 – Приклад зовнішнього вигляду панелі фільтрації таблиць

Описані вище елементи інтерфейсу повторюються на кожній сторінці системи і в детальному описі не потребують. Інтерфейс виконаний в мінімалістичному дизайні щоб мінімально відволікати користувача від роботи з системою. Система розроблялася з ідеєю максимально швидкої роботи з нею [28].

3.4 Дослідження результатів застосування алгоритмів машинного навчання при вирішенні задач обробки та аналізу метеоданих

З метою перевірки працездатності розробленої інформаційної системи, необхідно побудувати тестовий приклад, який дозволить проаналізувати ефективність роботи системи, а також забезпечить зворотній зв'язок щодо розробленого функціоналу.

Для базового прикладу функціональності програми досить перейти на сторінку перегляду метеоданих і натиснути на кнопку «Оновити дані». Після цього і будуть завантажені всі початкові дані необхідні для роботи алгоритмів і настройки всіх використовуваних нейромереж [29]. З боку користувача потрібно тільки очікування, весь подальший процес повністю автоматизований і не вимагає додаткових втручання людини. Першим етапом буде то що система завантажить інформацію про погоду, в разі якщо дані частково були завантажені раніше то система оновить їх вибравши тільки ті що з'явилися з дати останнього запису в системі.

Наступним етапом є підготовка нейромереж для прогнозування, після підготовки нейромережа будує прогноз метеоданих і погодних явищ. В процесі системи збирає статистику алгоритмів прогнозування для подальшого порівняння ефективності алгоритмів. В кінці цього процесу відбувається запис розрахованих прогнозів базу даних, прогноз пари виконує розрахунок в повний список прогнозів. Закінчення алгоритму в тому що користувачі переносяться на відповідну сторінку, на якій відображені всі створені дані [29].

На першому етапі тестування проводиться оцінка роботи нейромережі за допомогою кількох метрик. Даний метрики дають уявлення про те, наскільки добре нейромережа справляється зі своїми завданнями, наскільки ефективно пройшло її навчання і наскільки раціонально її використання, рис 3.13.

NeuralNet_id	Date_and_time	Conclusion
3	Jan. 7, 2023, 6:24 p.m.	Accuracy: 0.72 MeanSquaredError: 21.05 MeanAbsoluteError: 1.98 RootMeanSquaredError: 4.59

Рисунок 3.13 – Приклад запису тестів

Другим етапом алгоритму оцінки роботи нейромереж є процес аналізу ваг нейронів мережі, після нього вибудовуються всі гіперпараметри мережі в порядку їх важливості, інакше кажучи в порядку сили їх вплив на результат роботи мережі. Зберігаються найвпливовіший і не впливовий гіперпараметри. В таблиці на сторінці користувача, в одному з стовпчиків відмічаються знаками «I» - Important і «NI» - Not important, рис 3.14.

В наведеному прикладі, використовувалась нейромережа архітектури перцептрон в задачі виявлення метеорологічних явищ по метеорологічних даних, алгоритм аналізу виявив що найвпливовіший гіперпараметр це атмосферний тиск, самий невпливовий це температура повітря.

ID	Name	Target	Metric	Description	Conclusion
3	2022-12-112755-newPerc	2	2	layer1: 20, layer2: 1000, layer3: 100, layer4: 20. all dense	Important: air_pressure & Not important: temperature

Рисунок 3.14 – Приклад запису протестованої нейромережі

Це дослідження проводиться з метою того щоб з'ясувати, які нейронні мережі більше підходять для вирішення задач обробки та аналізу метеоданих, наскільки ефективно в цілому застосування нейромереж у подібних задачах [29].

У роботі брали участь та були досліджені три найпопулярніші архітектури нейромереж: перцептрон, рекурентна, автоенкодер. Кожна з них застосовувалася в задачах прогнозування метеоданих та визначення по них погодного явища. Всі вони були оцінені різними метриками, а також були оцінені на важливість гіперпараметрів.

Всі дані які були отримані в процесі записані у відповідні таблиці і можуть бути переглянуті користувачем пізніше. Головна перевага даної системи в тому що вона дозволяє легко перевірити алгоритми прогнозування. При постановці подібних задач користувачеві досить натиснути одну кнопку а для перевірки різних алгоритмів обробки і аналізу досить підмінити заздалегідь заготовлені файли нейромереж.

3.5 Висновки до третього розділу

Проаналізувавши різні методи розробки інформаційних систем, а також засоби реалізації, було спроектовано сучасне програмне рішення, яке може впроваджуватися в метеорологічні центри та використовуватися як самостійна система, або як додатковий модуль з розширенням функціональних можливостей

системи. При проектуванні інформаційної системи було визначено головні можливості системи, а також її функціональні особистості.

Головною складовою інформаційної системи є модуль обробки та аналізу даних. Додатковим програмним модулем є реалізація системи збору метеорологічних даних. Модуль опитує зовнішні джерела на предмет нових метеоданих та завантажує їх по конкретним запитам.

За результатами дослідження були обрані дві найбільш ефективні нейронні мережі за завданнями прогнозування та класифікації відповідно, це перцептрон для вирішення задачі визначення погодного явища та рекурентна мережа для прогнозування метеоданих. Вони були обрані відповідно до кращих оцінок метрик, що показали високу точність їхньої роботи [30].

Ключовими гіперпараметрами визначено атмосферний тиск та місце запису метеоданих, а найменшими виявилися температура повітря та рік запису метеоданих.

Розроблена інформаційна система може використовуватися в якості самостійного програмного засобу аналізу та спрощення роботи метеорологів, а також як додатковий програмний модуль, який може бути впроваджено до існуючої інформаційної системи метеорологічних центрів.

ВИСНОВКИ

В рамках виконання роботи були закріплені і розширені теоретичні знання, отримані при вивченні дисциплін, розвинені необхідні практичні вміння та навички відповідно до вимог рівня підготовки випускника та затвердженої тематики дослідження.

В результаті отриманих знань, і додатково вивчених матеріалів було проведене дослідження та розроблене програмне забезпечення що дозволяє оцінити ефективність нейромереж у задачах обробки та аналізу метеоданих.

Під час написання роботи, було виконано наступні завдання:

- проведено аналіз особливостей метеорологічних задач;
- здійснено аналіз специфіки машинного навчання;
- виконано аналіз можливостей сучасних засобів розробки програмного забезпечення з використанням алгоритмів машинного навчання;
- поставлено завдання регресії;
- здійснено порівняльний аналіз та вибір алгоритму регресії;
- розроблено концепцію попередньої обробки та аналізу даних;
- розроблено проект інформаційної системи;
- виконано опис структури та бази даних інформаційної системи;
- здійснено опис розробленого інтерфейсу до інформаційної системи;
- виконано дослідження результатів застосування алгоритмів машинного навчання при вирішенні задач обробки та аналізу метеоданих.

Розробку програмного комплексу виконано з використанням мови програмування Python та спеціалізованого фреймворку для роботи з web-системами Django. В якості системи керування базами даних використано SQLite.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 8302:2015 «Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання» – Чинний від 04.03.2016. – Київ: УкрНДНЦ, 2020. – 20 с.
2. Антоні С. Опануй самостійно програмування за 21 день / С. Антоні. – Москва: Вильямс, 2021. – 562 с.
3. Аронов І.З. Сучасні проблеми безпеки технічних систем і аналізу ризику / І.З. Аронов. – Москва: Інтуїт, 2021 – 451 с.
4. Астелс Д. Керівництво з екстремального програмування / Д. Астелс. – Москва: Вільямс, 2022. – 468 с.
5. Багриновский К.А. Нові інформаційні технології / К.А. Багриновский, Е.Ю. Хрустальов. – Москва: ЕКО, 2019. – 212 с.
6. Гмурман В.Є. Теорія ймовірностей і математична статистика / В.Є. Гмурман. – Москва: Вища. Шк, 2022. – 479с.
7. Бахтізін В.В. Технології розробки програмного забезпечення / В.В. Бахтізін, Л.А. Глухова. – Мінськ: БДУІР, 2020. – 267 с.
8. Бенін Д.М. Системи підтримки прийняття рішень / В.Л. Сніжко, Д.М. Бенін – Москва: Тріада, 2022. – 165 с.
9. Бхаграва А. Грокаєм алгоритми / А. Бхаграва – СПб : ПИТЕР, 2020. – 288с
10. Вірт Н. Алгоритми і структури даних. / Н. Вірт – Москва: Свет, 2019. – 651 с.
11. Владимиров В.А. Управління ризиком / В.А. Владимиров, Ю.Л. Воробйов, Г.Г. Малинецкий. – Москва: ЮНІТИ, 2020. – 352 с.
12. Будума Н. Основи глибокого навчання. Створення алгоритмів для штучного інтелекту наступного покоління / Будума Н. – Москва: ЮНІТИ, 2022 – 520с.

13. Ендрю Т. Грокаєм глибоке навчання / Т. Ендрю – СПб.: «БХВ-Петербург», 2021 – 712с
14. Прохоренок Н.А. Python 3 и PyQt 5. Розробка додатків / Н.А. Прохоренок, В.А. Дронов. – К.: БХВ-Петербург, 2020. – 832с.
15. Padmanabhan T.R. Programming with Python / T.R. Padmanabhan. – C.: Springer, 2020. – 349р.
16. Эрик М. Изучаем Python. Программирование игр, визуализация данных. Веб-додаток / М. Эрик. – К.: Питер, 2021. – 496с.
17. Себастіан Р. Python і машинне навчання / Р. Себастіан. – Х: Диалектика, 2021. – 752 с.
18. Бейлин Л. Вивчаемо MySQL / Л. Бейлин. – Х: Эксмо, 2022. – 1060 с.
19. Lane D. Web Database Application with PHP and MySQL / D. Lane. – New Jersey: O'Reilly, 2021. – 816 р.
20. Прохоренок Н.А. HTML, JavaScript, PHP и MySQL. Джентльменський набір Web-мастера / Н.А. Пархоменко. – О.: Екологія, 2019. – 768 с.
21. Роббінс Д. HTML5, CSS3 и JavaScript. Повний посібник / Д. Роббінс. – К.: Эксмо, 2019. – 528 с.
22. Шварц Б. MySQL. Оптимізація продуктивності / Б. Шварц. – Д.: Символ-Плю, 2019. – 483 с.
23. Гольцман В. MySQL 5.0 / В. Гольцман. – Х.: ПлюсТВ, 2019. – 764 с.
24. Яргер Р.Д. MySQL и mSQL. Бази даних для невеликих підприємств та інтернету/ Р.Д. Яргер. – С.: Символ-Плюс, 2021. – 929 с.
25. Уильман Л. MySQL. Посібник з вивчення мови / Л. Уильман. – К.: ДМК Прес, 2021. – 764 с.
26. Аткинсон Л. MySQL. Бібліотека професіонала / Л. Аткинсон. – Б.: Вильямс, 2021. – 1493 с.
27. Артеменко Ю. Н. MySQL. Довідник з мови / Ю. Н. Артеменко. – В.: Вильямс, 2021. – 843 с.

28. Никсон Р. «Learning PHP, MySQL, JavaScript, CSS & HTML5 A Step-by-Step Guide to Creating Dynamic Websites» / Р. Никсон. – O'Reilly Media, 2020. – 730 с.
29. Бастіан Ш. Великомасштабне машинне навчання / Ш. Бастіан. – NY.: Packt Publishing, 2022. – 579 р.
30. Жерон О. Прикладне машинне навчання з TensorFlow / О. Жерон – NY.: Apress, 2022. – 739 р.
31. Рудніченко М.Д. Проект інформаційної системи для збору, обробки та аналізу метеоданих / М.Д. Рудніченко, М.Б. Носов, О.В. Шведін // Інформатика, інформаційні системи та технології: тези доповідей дев'ятнадцятої всеукраїнської конференції студентів і молодих науковців. Одеса, 29 квітня 2022 р. – Одеса, 2022. – С. 45-47.
32. Носов М.Б. Проект інформаційної системи для збору, обробки та аналізу метеоданих / М.Д. Рудніченко, М.Б. Носов, Д.С. Шибаєв // Тези доповідей 57-ої конференції молодих дослідників НУОП “Сучасні інформаційні технології ”. – Одеса: НУОП, 2022. – С.115-118.

ДОДАТОК А

Лістинг коду інформаційної системи

```

def update_meteodata(request):
    MainMenu.data_update()
    return redirect('meteodata')

def update_forecast(request):
    MainMenu.magic()
    return redirect('forecast')

class MeteodataView(ListView):

    model = Meteodata
    template_name = 'meteodata/meteodataview.html'
    context_object_name = 'Meteodata'
    paginate_by = MainMenu.rows_count
    ordering = ['-datetime']
    allow_empty = True
    meteodata_top_labels = MainMenu.get_top_labels(Meteodata)

    def get_queryset(self):
        qs = super().get_queryset()
        qs = Meteodata.objects.all()
        return qs

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context.update({
            'top_labels': self.meteodata_top_labels,
            'current_menu_page': 0
        })
        context.update(MainMenu.get_context())
        return context

    def post(self, request):
        if request.POST['reason'] == 'sort':
            if MeteodataView.ordering ==
request.POST['sort_option']:
                MeteodataView.ordering = '-' +
str(request.POST['sort_option'])
            else:
                MeteodataView.ordering = request.POST['sort_option']
        return redirect('meteodata')

class ForecastView(ListView):

```

```

model = ForecastMeteodata
template_name = 'meteodata/forecastview.html'
context_object_name = 'ForecastMeteodata'
paginate_by = MainMenu.rows_count
ordering = ['-datetime']
allow_empty = True
forecast_top_labels = MainMenu.get_top_labels(ForecastMeteodata)

def get_queryset(self):
    qs = super().get_queryset()
    qs = ForecastMeteodata.objects.all()
    return qs

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context.update({
        'top_labels': self.forecast_top_labels,
        'current_menu_page': 1
    })
    context.update(MainMenu.get_context())
    return context

def post(self, request):
    if request.POST['reason'] == 'sort':
        if ForecastView.ordering == request.POST['sort_option']:
            ForecastView.ordering = '-' +
str(request.POST['sort_option'])
        else:
            ForecastView.ordering = request.POST['sort_option']
    return redirect('forecast')

class NeuralnetView(ListView):

    model = NeuralNet
    template_name = 'meteodata/neuralnetsview.html'
    context_object_name = 'NeuralNet'
    paginate_by = MainMenu.rows_count
    ordering = ['name']
    allow_empty = True
    neuralnets_top_labels = MainMenu.get_top_labels(NeuralNet,
['NetFile',''])

    def get_queryset(self):
        qs = super().get_queryset()
        qs = NeuralNet.objects.all()
        return qs

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        if 'Options' not in self.neuralnets_top_labels:

```

```

        self.neuralnets_top_labels.append('Options')
    context.update({
        'top_labels': self.neuralnets_top_labels,
        'current_menu_page': 2
    })
    context.update(MainMenu.get_context())
    return context

def post(self, request):
    if request.POST['reason'] == 'sort':
        if NeuralnetView.ordering ==
request.POST['sort_option']:
            NeuralnetView.ordering = '-' +
str(request.POST['sort_option'])
        else:
            NeuralnetView.ordering = request.POST['sort_option']
    elif request.POST['reason'] == 'addFile':
        print()
        data = {
            'name':
request.FILES['file_data'].name.split('.')[0],
            'target': request.POST['target'],
            'metric': request.POST['metric'],
            'description': request.POST['description'],
            'conclusion': '-'
        }
        form = NeuralNetForm(data, request.FILES)
        if form.is_valid():
            form.save()
        else:
            print(form.errors)
    elif request.POST['reason'] == 'test':
        MainMenu.make_test(int(request.POST['row_id']))
    elif request.POST['reason'] == 'remove':
        netfile =
NeuralNet.objects.get(pk=int(request.POST['row_id']))
        netfile.delete()
    return redirect('neuralnets')

class TestsView(ListView):

    model = Test
    template_name = 'meteodata/testsvview.html'
    context_object_name = 'Test'
    paginate_by = MainMenu.rows_count
    ordering = ['-datetime']
    allow_empty = True
    test_top_labels = MainMenu.get_top_labels(Test, ["ID", ''])

    def get_queryset(self):
        qs = super().get_queryset()

```

```

        qs = Test.objects.all()
        return qs

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context.update({
            'top_labels': self.test_top_labels,
            'current_menu_page': 3
        })
        context.update(MainMenu.get_context())
        return context

    def post(self, request):
        if request.POST['reason'] == 'sort':
            if TestsView.ordering == request.POST['sort_option']:
                TestsView.ordering = '-' +
str(request.POST['sort_option'])
            else:
                TestsView.ordering = request.POST['sort_option']
        return redirect('tests')

class Tester:

    def __init__(self):
        pass

    def makeFullTest(self, examiner_id):
        obj = NeuralNet.objects.get(pk=examiner_id)
        examinerLoader = 0
        if obj.target == 1:
            examinerLoader =
SimpleForecastModel(model_id=examiner_id)
        elif obj.target == 2:
            examinerLoader =
ForecastSummaryModel(model_id=examiner_id)
        conclusion = self.parametersImportantTest(examinerLoader)
        testResult = examinerLoader.test()
        print(testResult)
        self.saveConclusion(examiner_id, conclusion)
        self.saveTest(examiner_id, testResult)

    def parametersImportantTest(self, examinerLoader):
        conclusion = ''
        data =
examinerLoader._neuralNetObject.layers[0].get_weights()[0]
        result = {}
        labels = examinerLoader.getHyperparametersLabels()
        for i in range(len(data)):
            result.update({labels[i]:
abs(statistics.mean(data[i]))})
        important = labels[0]

```

```

not_important = labels[0]
for key, value in result.items():
    if value > result[important]:
        important = key
    if value < result[not_important]:
        not_important = key
conclusion = 'Important: {} & Not important: {}'.format(important, not_important)
print(result)
return conclusion

def saveConclusion(self, examiner_id, conclusion):
    obj = NeuralNet.objects.get(pk=examiner_id)
    obj.conclusion = conclusion
    obj.save()

def saveTest(self, examiner_id, testResult):
    obj = Test.objects.all().filter(neuralnet_id=examiner_id)
    if len(obj) != 0:
        obj = Test.objects.get(neuralnet_id=examiner_id)
        obj.datetime=datetime.today()
        obj.conclusion=testResult
        obj.save()
    else:
        data = {
            'neuralnet_id': examiner_id,
            'datetime': datetime.today(),
            'conclusion': testResult
        }
        form = TestForm(data)
        if form.is_valid():
            form.save()
        else:
            print(form.errors)

class NNBuilder:

    def buildRnnNet():
        loss = 'mean_squared_error'
        method = keras.Sequential()
        method.add(keras.layers.Embedding(input_dim=10,
output_dim=5))
        method.add(keras.layers.LSTM(units=25,
return_sequences=False))
        method.add(keras.layers.Dense(5, activation='softmax'))
        method.compile(optimizer='adam', loss=loss,
metrics='accuracy')
        method.summary()
        return method

    def buildPerceptronNet(output_dim):

```

```

method = keras.Sequential()
method.add(keras.layers.Dense(10, activation='relu',
input_shape=(10,)))
    method.add(keras.layers.Dense(1000, activation='relu'))
    method.add(keras.layers.Dense(100, activation='relu'))
    method.add(keras.layers.Dense(output_dim,
activation='softmax'))
    method.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics='accuracy')
    method.summary()
    return method

def buildRnnNet_ForSummary(output_dim):
    method = keras.Sequential()
    method.add(keras.layers.Embedding(input_dim=10,
output_dim=10))
    method.add(keras.layers.LSTM(units=100,
return_sequences=False))
    method.add(keras.layers.Dense(output_dim,
activation='softmax'))
    method.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics='accuracy')
    method.summary()
    return method

def buildPerceptronNet_ForForecast():
    method = keras.Sequential()
    method.add(keras.layers.Dense(10, activation='relu',
input_shape=(10,)))
    method.add(keras.layers.Dense(500, activation='relu'))
    method.add(keras.layers.Dense(50, activation='relu'))
    method.add(keras.layers.Dense(5, activation='softmax'))
    method.compile(optimizer='adam', loss='mean_squared_error',
metrics='accuracy')
    method.summary()
    return method

def buildAutoencoder_ForSummary(output_dim):
    obj = AutoencoderModel(10, 100, output_dim)
    obj.compile(optimizer='adam', loss='mean_squared_error',
metrics='accuracy')
    return obj

def buildAutoencoder_ForForecast():
    return AutoencoderModel(10, 100, 5)

class AutoencoderModel(Model):

    def __init__(self, values_size, latent_dim, output_dim):
        super(AutoencoderModel, self).__init__()
        self.latent_dim = latent_dim

```

```

        self.encoder = tf.keras.Sequential()

    self.encoder.add(keras.layers.InputLayer(input_shape=(values_size,
)))
        self.encoder.add(keras.layers.Dense(latent_dim,
activation="relu"))
        self.encoder.add(keras.layers.Dense(int(latent_dim / 2),
activation="relu"))
        self.decoder = tf.keras.Sequential()

    self.decoder.add(keras.layers.InputLayer(input_shape=(int(latent_dim
/ 2), )))
        self.decoder.add(keras.layers.Dense(int(latent_dim / 2),
activation="relu"))
        self.decoder.add(keras.layers.Dense(output_dim,
activation="relu"))

    def call(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return decoded

class NNBBase:

    def __init__(self, model_id='2', modelName='my_rnn',
dbName='meteodata_meteodata'):
        self._correctWeatherDict = {}
        self._neuralNetObject = []
        self._modelName = modelName
        self._dbName = dbName
        self._epochs = 10
        self._weatherList = {...}
        tempWeatherList = []
        for item in self._weatherList.values():
            if item not in tempWeatherList:
                tempWeatherList.append(item)
        i = 1.0
        for item in tempWeatherList:
            self._correctWeatherDict.update({item: i})
            i = i + 1.0
        if model_id != '':
            self.loadNet(model_id)

    # def __del__(self):
    #     tmp = subprocess.check_output('pwd',
shell=True).decode().rstrip()
    #     subprocess.check_output("rm -rv
{}/media/models/{}".format(tmp, self._modelName), shell=True)

    def loadDataSet(self):
        if self._dbName == 'meteodata_meteodata':

```

```

        return list(Meteodata.objects.all().order_by('-
datetime').values())
    else:
        return list(ForecastMeteodata.objects.all().order_by('-
datetime').values())

def loadNet(self, net_id):
    print('start load net: ', net_id)
    obj = NeuralNet.objects.get(pk=net_id)
    f = obj.file_data
    self._modelName = obj.name
    wayToFile = subprocess.check_output('pwd',
shell=True).decode().rstrip()
    if not os.path.exists('{}/media/models/{}'.format(wayToFile,
self._modelName)):
        subprocess.check_output("file-roller -h
{}/media/{}".format(wayToFile, str(f)), shell=True)
    self._neuralNetObject =
keras.models.load_model('{}/media/models/{}'.format(wayToFile,
self._modelName))
    self._neuralNetObject.summary()
    print('finish load')

def saveNet(self, name='Rnn'):
    d = str(date.today())
    t = ttime.localtime()
    t = ttime.strftime("%M%S", t)
    fullName = '' + d + t + '--' + name
    tmp = subprocess.check_output('pwd',
shell=True).decode().rstrip()
    # subprocess.call('mkdir ' + tmp +
'/media/models/{}'.format(fullName))
    self._neuralNetObject.save(tmp +
'/media/models/{}'.format(fullName))

def write_weights(obj, afterTrain):
    f = open('logs.txt', 'a')
    strL = []
    if afterTrain:
        strL.append('train weights')
    else:
        strL.append('load weights')
    s = ''
    for i in range(len(obj.layers)):
        s = s + 'layer #' + str(i)
        for w in obj.layers[i].get_weights():
            if isinstance(w, float):
                s = s + '-{}-'.format(str(w))
            else:
                for x in w:
                    s = s + '_{}_'.format(str(x))
    strL.append(s)

```

```

        strL.append('end weights')
        f.writelines(strL)
        f.close()

    def getHyperparametersLabels(self):
        return [
            'year',
            'month',
            'day',
            'hour',
            'place',
            'temperature',
            'wind_way',
            'wind_speed',
            'air_pressure',
            'water_pressure'
        ]

    class ForecastSummaryModel(NNBase):

        def __init__(self, model_id='3', modelName='my_fullconnect',
                     dbName='meteodata_forecastmeteodata'):
            NNBase.__init__(self, model_id=model_id,
                           modelName=modelName, dbName=dbName)

        def predict(self, data):
            result = []
            tmpData = self.encode(data)
            result = self._neuralNetObject.predict(tmpData, verbose=1)
            return self.decode(result)

        def decode(self, data):
            val = list(self._correctWeatherDict.values())
            keys = list(self._correctWeatherDict.keys())
            resultList = []
            for row in data:
                maxValueIndex = 0
                if isinstance(row, float):
                    maxValueIndex = row - 1
                else:
                    maxValueIndex = 0
                    for i in range(len(row)):
                        if row[i] > row[maxValueIndex]:
                            maxValueIndex = i
                for i in range(len(val)):
                    if maxValueIndex + 1 == val[i]:
                        resultList.append(keys[i])
            return resultList

        def _decode(self, data):
            val = list(self._correctWeatherDict.values())
            keys = list(self._correctWeatherDict.keys())

```

```

resultList = []
for row in data:
    maxValueIndex = 0
    if isinstance(row, float):
        maxValueIndex = row - 1
    else:
        maxValueIndex = 0
        for i in range(len(row)):
            if row[i] > row[maxValueIndex]:
                maxValueIndex = i
        resultList.append(maxValueIndex)
return resultList

def encode(self, data):
    values = data # TODO make good normalisation func?
    resultList = []
    for i in range(len(values)):
        tmp = []
        d = values[i]['datetime']
        tmp.append(d.year)
        tmp.append(d.month)
        tmp.append(d.day) # 0 - date
        tmp.append(d.hour)
        tmp.append(float(values[i]['place'])) # 1 - place
        tmp.append(float(values[i]['temperature'])) # 3 -
temperature
        tmp.append(float(values[i]['wind_way'])) # 4 - wind_way
        tmp.append(float(values[i]['wind_speed'])) # 5 -
wind_speed
        tmp.append(float(values[i]['air_pressure'])) # 6 -
air_pressure
        tmp.append(float(values[i]['water_pressure'])) # 7 -
water_pressure
        resultList.append(tmp)
    return resultList

def _encode(self, data):
    values = data
    labels = []
    for i in range(len(values)):
        weather = values[i]['weather']
        labels.append(weather)
    for i in range(len(labels)):
        try:
            temp = labels[i]
            if temp.isdigit():
                labels[i] = "Het"
            else:
                labels[i] = self._weatherList[labels[i]]
        except:
            labels[i] = "Het"
    for i in range(len(labels)):

```

```

        labels[i] = (self._correctWeatherDict[labels[i]])
    resultList = []
    for i in range(len(values)):
        tmp = []
        d = values[i]['datetime']
        tmp.append(d.year)
        tmp.append(d.month)
        tmp.append(d.day) # 0 - date
        tmp.append(d.hour)
        tmp.append(float(values[i]['place'])) # 1 - place
        tmp.append(float(values[i]['temperature'])) # 3 -
temperature
        tmp.append(float(values[i]['wind_way'])) # 4 - wind_way
        tmp.append(float(values[i]['wind_speed'])) # 5 -
wind_speed
        tmp.append(float(values[i]['air_pressure'])) # 6 -
air_pressure
        tmp.append(float(values[i]['water_pressure'])) # 7 -
water_pressure
        resultList.append(tmp)
    values = resultList
    train_values, test_values, train_labels, test_labels =
train_test_split(values, labels, test_size=0.20)
    d = {'values': values, 'labels': labels, 'train_values':
train_values, 'test_values': test_values, 'train_labels':
train_labels, 'test_labels': test_labels}
    return d

    def test(self):
        ForecastSummaryModel.write_weights(self._neuralNetObject,
False)
        # testData = self._encode(self.loadDataSet())
        testData =
self._encode(list(Meteodata.objects.all().order_by(
'-datetime').values()[:500000]))
        predictResult =
self._neuralNetObject.predict(testData['test_values'], verbose=1)
        predictResult = self._decode(predictResult)
        test_result = ''
        acc = Accuracy()
        acc.update_state(testData['test_labels'], predictResult)
        test_result = test_result + ' Accuracy: {}'
'.format(float(f'{acc.result().numpy():.2f}'))
        mse = MeanSquaredError()
        mse.update_state(testData['test_labels'], predictResult)
        test_result = test_result + ' MeanSquaredError: {}'
'.format(float(f'{mse.result().numpy():.2f}'))
        mae = MeanAbsoluteError()
        mae.update_state(testData['test_labels'], predictResult)
        test_result = test_result + ' MeanAbsoluteError: {}'
'.format(float(f'{mae.result().numpy():.2f}'))
        rmse = RootMeanSquaredError()

```

```

        rmse.update_state(testData['test_labels'], predictResult)
        test_result = test_result + ' RootMeanSquaredError: {}'
    '.format(float(f'{rmse.result().numpy():.2f}')))

    return test_result

class MainMenu:

    Miner = MeteodataMiner()
    SummaryModel = ForecastSummaryModel()
    ForecastModel = SimpleForecastModel()
    Tester = Tester()
    rows_count = 40

    def get_context():
        context = {}
        return context

    def data_update():
        print('data_update')
        MainMenu.Miner.updateMeteodata()

    def forecast_update():
        print('forecast_update')
        print('prepare data for forecast')
        date = datetime.now()
        lastdate = ForecastMeteodata.objects.all().order_by(
            '-datetime').values_list('datetime').last()
        if not lastdate:
            lastdate =
Meteodata.objects.all().order_by('datetime').values_list('datetime')
        .first()
        print('lastdate: ', lastdate)
        dataFull = list(Meteodata.objects.order_by(
            '-datetime').filter(datetime__gte=str(lastdate[0])).values()[:10000])
        print('start forecast prediction: ', str(datetime.now() -
date))
        resultMeteo = MainMenu.ForecastModel.predict(dataFull)
        print(resultMeteo[:10])
        for i in range(len(resultMeteo)):
            resultMeteo[i].update({
                dataFull[i]['datetime'],
                dataFull[i]['place']
            })
        print('start summary prediction: ', str(datetime.now() -
date))
        resultSummary = MainMenu.SummaryModel.predict(resultMeteo) #
TODO check result dict assembling
        result = []
        for i in range(len(resultSummary)):
            result.append({
                'datetime': dataFull[i]['datetime'],

```

```

        'place': dataFull[i]['place'],
        'place_name': dataFull[i]['place_name'],
        'temperature': resultMeteo[i]['temperature'],
        'wind_way': resultMeteo[i]['wind_way'],
        'wind_speed': resultMeteo[i]['wind_speed'],
        'air_pressure': resultMeteo[i]['air_pressure'],
        'water_pressure': resultMeteo[i]['water_pressure'],
        'weather': resultSummary[i]
    })
print('finish forecast creation: ', str(datetime.now() - date))
MainMenu._save_forecast(result)

def make_test(examiner_id):
    print('make_nets_test')
    MainMenu.Tester.makeFullTest(examiner_id)

def _save_forecast(data):
    print('forecast saving started')
    date = datetime.now()
    for row in data:
        form = ForecastForm(row)
        if form.is_valid():
            form.save()
        else:
            print(form.errors)
    print('forecast saving finished: ', str(datetime.now() - date))

def get_top_labels(modelObj, exeptionList=[]):
    labels = []
    fields = modelObj._meta.get_fields()
    for row in fields:
        if str(row.verbose_name) not in exeptionList:
            labels.append('' + row.verbose_name)
    return labels

```

ДОДАТОК Б

Слайди презентації

Тема кваліфікаційної роботи:

Розробка системи оцінки ефективності нейромереж у задачах обробки та аналізу метеоданих

Виконав студент:

Носов Максим Борисович

Рисунок Б1 – Титульний слайд презентації

Мета розробки інформаційної системи

-
- Оцінка ефективності вирішення метеорологічних задач алгоритмами машинного
 - Розробка системи оцінки ефективності алгоритмів машинного навчання
 - Оцінка роботи нейромереж класичними метриками
 - Оцінка впливовості гіперпараметрів на результат вирішення задачі

Рисунок Б2 – Слайд описом мети та описом розроблених діаграм

Розробка інформаційної системи

- Оновлення метеоданих
 - Відображення даних
 - Завантаження нейромереж
-
- Класичний алгоритм оцінки нейромереж завдяки метрикам
 - Алгоритм факторного аналізу нейромережі для оцінки впливовості гіперпараметрів

Рисунок Б3 – Слайд з визначеннями функціями системи

Діаграма класів

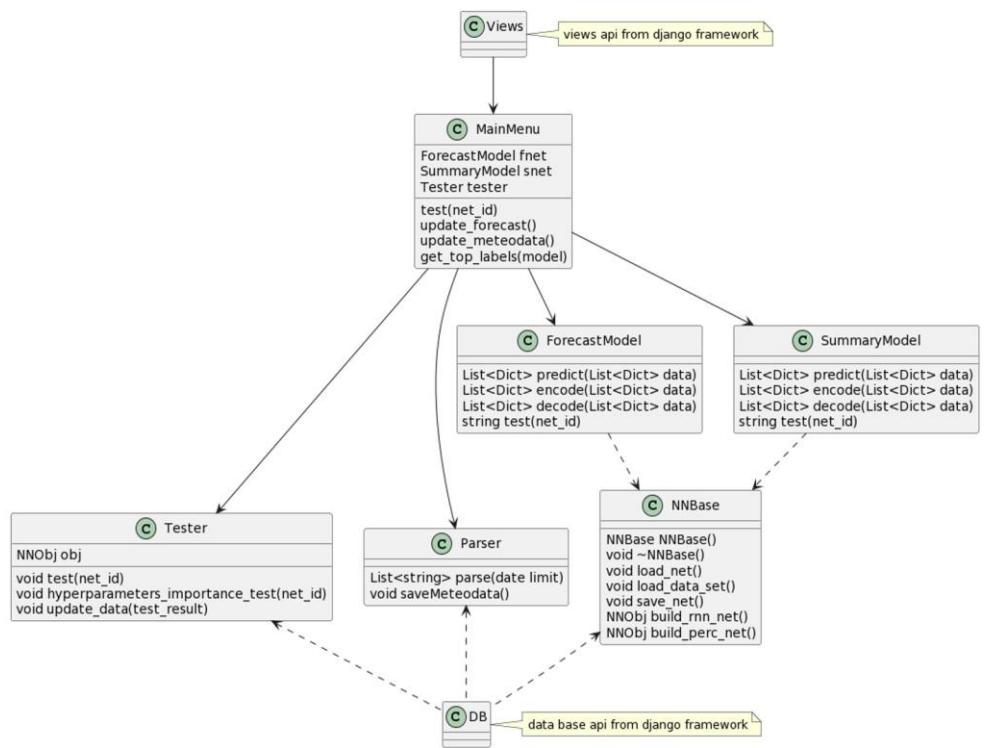


Рисунок Б4 = Слайд з диаграмою класів

Використані архітектури алгоритмів машинного навчання:

- ▶ Перцептрон
- ▶ Рекурентна
- ▶ Автоенкодер

Кожна з варіантів нейромереж була використана для вирішення задач прогнозування і класифікації метеоданих

Рисунок Б5 – Слайд зі списком використаних в роботі нейромереж

Перцептрон

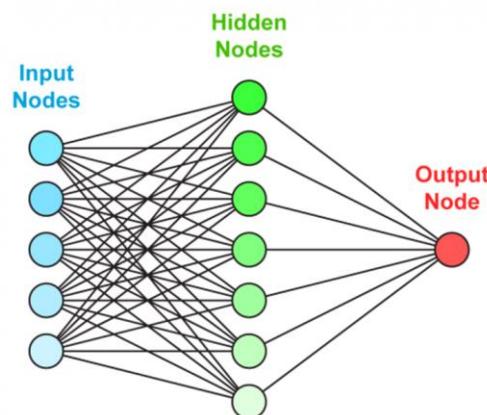


Рисунок Б6 – Слайд з прикладом перцептрону

Рекуррентна

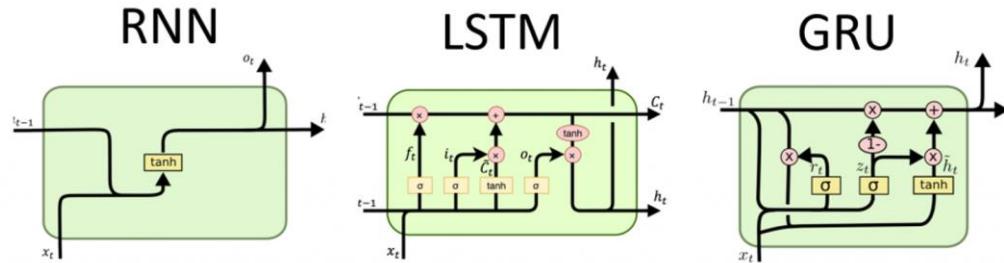


Рисунок Б7 – Слайд з прикладом рекуррентної мережі

Автоенкодер

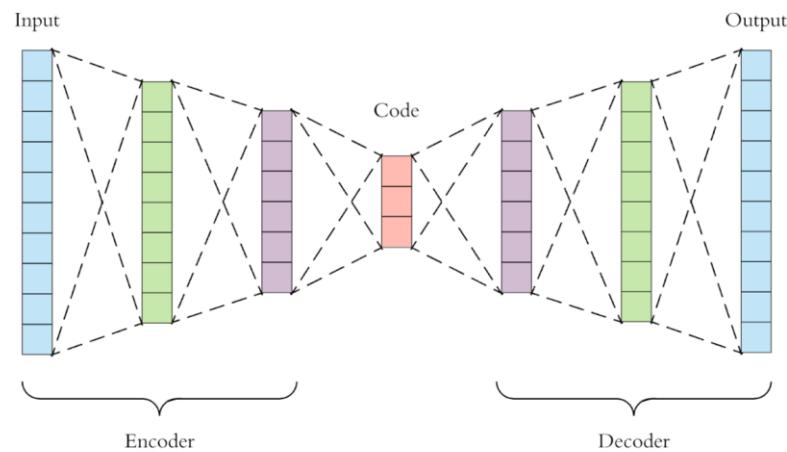


Рисунок Б8 – Слайд з прикладом автоенкодера

Результати дослідження

Perceptron Forecast	Accuracy: 71% Precision: 71.3
RNN Forecast	Accuracy: 88% Precision: 88.0
Autoencoder Forecast	Accuracy: 69% Precision: 69.7
Perceptron Summary	Accuracy: 72% MeanAbsoluteError: 1.98
RNN Summary	Accuracy: 64% MeanAbsoluteError: 2.13
Autoencoder Summary	Accuracy: 71% MeanAbsoluteError: 1.99

Найбільш впливові гіперпараметри

- атмосферний тиск
- місце запису метеоданих

Найбільш не впливові гіперпараметри.

- температура повітря
- рік запису метеоданих

Рисунок Б9 – Слайд результатів дослідження

Використані програмні засоби та бібліотеки:

- ▶ Мова програмування Python3
- ▶ Середовище для розробки Anaconda
- ▶ Веб-фреймворк Django3
- ▶ СУБД SQLite3
- ▶ Бібліотека Tensorflow2 і BeautifulSoup4

Рисунок Б10 – Слайд з визначенням засобів реалізації роботи

Під час написання кваліфікаційної роботи, було виконано:

- ▶ Аналіз особливостей метеорологічних задач
- ▶ Аналіз специфіки машинного навчання
- ▶ Аналіз можливостей сучасних засобів розробки програмного забезпечення з використанням алгоритмів машинного навчання
- ▶ Постановка завдання регресії і класифікації, порівняльний аналіз та вибір алгоритму регресії і класифікації
- ▶ Розробка концепції попередньої обробки та аналізу даних
- ▶ Розробка проекту інформаційної системи
- ▶ Опис структури та бази даних інформаційної системи
- ▶ Дослідження результатів застосування алгоритмів машинного навчання при вирішенні задач обробки та аналізу метеоданих

Рисунок Б11 – Слайд з висновками до роботи

ДОДАТОК В

Апробація результатів роботи

Інформатика, інформаційні системи та технології

Державний заклад
«ПІВДЕННОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ
ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
імені К. Д. УШИНСЬКОГО»



ОДЕСЬКИЙ
НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ імені І. І. МЕЧНИКОВА

ДЕВ'ЯТНАДЦЯТА ВСЕУКРАЇНСЬКА КОНФЕРЕНЦІЯ
СТУДЕНТІВ І МОЛОДИХ НАУКОВЦІВ

**ІНФОРМАТИКА, ІНФОРМАЦІЙНІ
СИСТЕМИ ТА ТЕХНОЛОГІЇ**

29 квітня 2022 р.

Одеса – 2022

Рисунок В.1 – Скановане зображення титульного аркушу збірника тез

додано можливість обліку бонусів та знижок. Таким чином таке програмне застосування можна масштабувати та збільшувати користь чи комфорт для користувачів згідно їх потребам.

Література

1. Фінансова грамотність, фінансова інклузія та фінансовий добробут в Україні [Електронний ресурс]. – Режим доступу: http://www.fst-ua.info/wp-content/uploads/2019/06/Financial-Literacy-Survey-Report_June2019_ua.html
2. Do budgeting apps really work? [Електронний ресурс]. – Режим доступу: <https://www.ft.com/content/6dff9670-cf26-11e9-b018-ca4456540ea6>
3. 10 Best Practices to Enhance Your Mobile App User Experience [Електронний ресурс]. – Режим доступу: <https://clearbridgemobile.com/best-practices-to-enhance-your-mobile-app-user-experience>

ПРОЕКТ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ЗБОРУ, ОБРОБКИ ТА АНАЛІЗУ МЕТЕОДАНИХ

Рудніченко М. Д., Носов М. Б., Шведін О. В.

Міжрегіональна академія управління персоналом

Анотація: в даній роботі було розглянуто проект клієнт-серверної інформаційної системи для збору, обробки та аналізу метеоданих.

Ключові слова: інформаційні системи, метеодані, збір даних, аналіз даних.

Сучасна метеорологія є добрим напрямком для впровадження сучасних інформаційних технологій [1]. Це пов'язано з актуальним використанням засобів комунікації та інформаційного обміну між метеостанціями та метеоцентрими [2].

Для ефективного виконання завдань метеорології використане програмне забезпечення повинно мати наступні властивості [3]:

- Швидкодія - так як даних які потрібно обробити дуже багато і сам розрахунок прогнозу досить складний алгоритми використовуються в програмі повинні бути дуже швидкими, як і інші функції програми, інакше необхідний розрахунок ризикує не бути проведений вчасно.

- Зручність використання - чим простіше і зрозуміліше для користувача інтерфейс програми тим легше йому буде з ним працювати і тим швидше будуть виконуватися його повсякденна робота.

- Точність – з урахуванням того що обчислення робить програ ма а не людина то було б дуже корисно максимально підвищити точність прогнозів.

Однією з найважливіших елементів в сфері метеорології є самі метеостанції, на даний момент, в більшості випадків, вони представляють собою майданчик з різним набором аналогових детекторів. При переробки метеостанції

Інформатика, інформаційні системи та технології

в вид повністю електронної системи розробляється програмний продукт повинен відповідати наступним вимогам [4]:

- Малий розмір – чим менше місця в пам'яті пристрою буде займати програмний продукт тим більше можна заощадити на пам'яті пристрою що в цілому дозволить спростити сам пристрій і зробити його дешевше.

- Автономність – цей критерій застосовний як до програмного забезпечення так і до самого приладу. Реалізується фактор автономності оптимальними алгоритмами передачі даних і наявністю систем отримання і накопичення енергії в пристрой.

- Інтернет – чим краще буде реалізована логіка передачі даних тим меньше буде втрат даних на відстані та зручніше збирати їх на сервері.

Основне призначення проекту, що розробляється в рамках даної роботи, полягає в забезпеченні зручного інструмента для збору, обробки та аналізу метеоданих

ІС повинна забезпечити виконання наступних функціональних можливостей:

1. Створення бази даних на сервері.
2. Збір даних з різних баз даних або з метеорологічних станцій
3. Операції по роботі з базою даних, перегляд, видалення, тощо.
4. Автоматична обробка зібраних даних.
5. Аналіз даних на аномалії.
6. Прогнозування метеорологічних явищ.
7. Прогнозування показників метеорологічних .
8. Відображення загальної статистики по даним.

В базі даних зберігаються зібрані з метеостанцій та інших джерел дані.

Для роботи системи потрібно запустити сервер програми, наприклад локальний, та веб-браузер для відображення сторінки користувача.

Після першого запуску в поточній директорії буде створено файл бази даних програми. Система може бути встановлена на операційні системи Linux\Windows\MacOS. Для роботи системи потрібні: python v3.9+, Django 3, Tensorflow 2 та SQLite 3. В ході роботи з ІС, користувач може створювати нові і відкривати вже існуючі БД. При запуску програми автоматично відкривається остання використовувана БД.

Висновки. Розроблений проект інформаційної системи є логічно послідовним та цілісним, він може бути реалізований на базі використання сучасних засобів програмної розробки та імплементований у подальших дослідженнях з обраної тематики.

Література

1. Антоні С. Опануй самостійно програмування за 21 день / С. Антоні. – Москва: Вильямс, 2017. – 562 с.
2. Аронов І.З. Сучасні проблеми безпеки технічних систем і аналізу ризику / І.З. Аронов. – Москва: Інтуїт. – 451 с.
3. Дронов В. Django З практика створення веб-сайтів на python / В. Дронов – СПб.: «БХВ-Петербург», 2019 – 712с
4. Багриновский К.А. Нові інформаційні технології / К.А. Багриновский, Е.Ю. Хрустальов. – Москва: ЕКО, 2015. – 212 с.

**АНАЛІЗ МЕТОДІВ СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ
ПРИЙНЯТТЯ РІШЕНЬ ПО РЕАГУВАННЮ НА ВПЛИВИ
ЗОВНІШНЬОГО СЕРЕДОВИЩА**

Рудніченко М. Д., Венгерович І. М., Отрадська Т. В.

Державний університет «Міжрегіональна Академія Управління персоналом»

Анотація: в даній роботі було розглянуто технології реалізації інтелектуальної системи прийняття рішень здатної реагувати на середовище.

Ключові слова: навчання з підкріпленням, мультиагентні системи, немарківські процеси.

Для того щоб реалізувати системи прийняття рішень, яка здатна відповідно реагувати на різні ситуації з середовища, необхідно скласти мультиагентну систему, в якій елементом навчання для агента виступатиме навчання з підкріпленням, а елементом прийняття рішень буде немарківський процес, представлений у вигляді нейронної мережі [1]. Таким чином, для вирішення задачі виділяються такі ключові теми:

- нейромережа;
- навчання з підкріпленням;
- немарківський процес;
- мультиагентну систему.

Нейронна мережа — спроба за допомогою математичних моделей відтворити роботу людського мозку для створення машин, які мають штучний інтелект.

Штучна нейронна мережа зазвичай навчається з учителем. Це означає наявність навчального набору (датасету), який містить приклади з істинними значеннями: тегами, класами, показниками.

Навчання з підкріпленням (Reinforcement Learning) – це метод машинного навчання, у якому наша система (агент) навчається методом спроб та помилок. Ідея полягає в тому, що агент взаємодіє із середовищем, паралельно навчаючись, і отримує винагороду за виконання дій [2].

Тези доповідей 57-ої конференції молодих дослідників НУОП “Сучасні інформаційні технології” // Одеса: ДУОП, 2022 вип. 57.С.115-117

ПРОЕКТ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ЗБОРУ, ОБРОБКИ ТА АНАЛІЗУ МЕТЕОДАНИХ

Рудніченко М.Д., Носов М.Б., Шибаєв Д.С.

Приватне акціонерне товариство «Вищий навчальний заклад «Міжрегіональна академія управління персоналом»»

Анотація: в даній роботі було розглянуто проект клієнт-серверної інформаційної системи для збору, обробки та аналізу метеоданих.

Ключові слова: інформаційні системи, метеодані, збір даних, аналіз даних.

Призначення проекту, що розробляється в рамках даної роботи, полягає в забезпеченні зручного інструмента для збору, обробки та аналізу метеоданих

Розроблена система пропонує користувачу наступний об'єм функцій, рисунок 1.

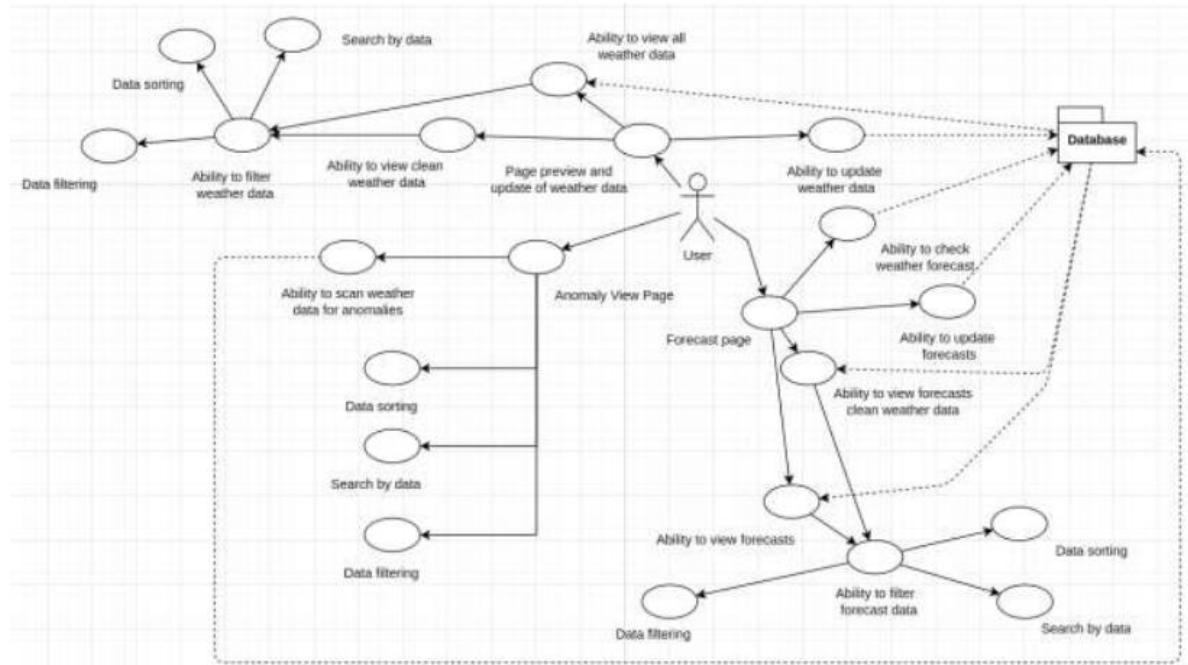


Рисунок 1 - Діаграма варіантів використання

З найголовнішої функції можна визначити функції пушку аномалій та розрахування метеорологічного прогнозу.

Рисунок В.5 – Скановане зображення першої сторінки публікації

Тези доповідей 57-ої конференції молодих дослідників НУОП “Сучасні інформаційні технології” // Одеса: ДУОП, 2022 вип. 57.C.115-117

Оновлення прогнозу. На сторінці перегляду прогнозів користувач має можливість оновити зроблений системою прогноз, ця функція потрібна для тестування алгоритму прогнозування, та використовується коли нових даних не має або сервер оновив нейромережі які працювали з даними.

Запуск алгоритма пошуку аномалій. На сторінці перегляду аномалій користувач має можливість оновити список знайдених аномалій, ця функція потрібна для тестування алгоритму пошуку аномалій, та використовується коли нових даних не має або сервер оновив нейромережі які працювали з даними.

Можливість проведення тестування ефективності навчання нейромереж на сирих та «чистих» метеоданих. Процес тестування починається з того що нейромережа для пошуку аномалій пробігає данні, формує два списки даних, на цих списках вчаться нейромережі прогнозування метеоданих та погодних явищ, після цього ці нейромережі проходять тестування на сирих та «чистих» даних де і порівнюється їх точність.

Що стусується системи в цілому то вона має MVC архітектуру та складається з декількох програмних модулів. Система представляє собою веб-додаток який написаний мовою Python з використанням фреймворку Django. Даний фреймворк дозволяє зручно організувати структуру веб-сторінок та їх шаблонизацію, а також надає зручні інтерфейси для спілкування з базою даних. Основна структура системи відштовхується від шаблону MVC, оскільки саме він використовується в даному фреймворку. Моделі нейронних мереж були побудовані за допомогою фреймворку TensorFlow 2, він дозволяє зручно розробляти та тестувати створені нейронні мережі.

Ключовими модулями в системі є: модуль збору метеоданих, який відповідає за парсинг різних ресурсів з метою оновлення поточної бази метеоданих у системі, модуль нейронних мереж, що проводить розрахунки на основі зібраних метеоданих, також цей модуль займається фільтрацією аномалій у даних та модуль побудови графіків для наочного відображення характеру наявних даних.

Рисунок В.6 – Скановане зображення другої сторінки публікації

Тези доповідей 57-ої конференції молодих дослідників НУОП “Сучасні інформаційні технології ” // Одеса: ДУОП, 2022 вип. 57.С.115-117

Як конкретний приклад наводяться два класи системи. Клас Parser має вхідний метод у якому організується весь процес збору даних, починаючи з формування веб-запитів, а також використовуючи бібліотеку BeautifulSoup 4 отримує “сирі” дані з вибраних ресурсів і передає їх у метод обробник, який перебираючи отриману інформацію виділяє з неї ті метеорологічні змінні, що цікавлять систему, оформляючи кожен рядок даних у словнику та повертаючи відповідю їх список. Клас ForecastSummaryModel це один із класів описують інтерфейс взаємодії з однією з нейронних мереж, а саме з відповідальною за визначення погодного явища за наявними метеоданими. У даному класі містяться методи завантаження моделі із заздалегідь підготовлених файлів, методи нормалізації та аналізу даних, методи оцінки точності розрахунку нейромережею.

Рисунок В.7 – Скановане зображення третьої сторінки публікації

ДОДАТОК Г

Електронні матеріали

Зміст	Папка	Ім'я файла
Пояснювальна записка до роботи магістра		diploma.docx
Вихідні дані до роботи магістра		readme.txt
Докладний проект програми з кодами і поясненнями у середовищі проектування програмного продукту		project.zip
Демонстраційний ролик програмної системи (слайди, анімація тощо)		demo.pptx