НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УДК 004.4:004.8

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	Завідувач кафедри
Інформаційних технологій	Комп'ютерних систем, мереж та кібербезпеки
_____ Болбот І.М., д.т.н, проф.	_____ Касаткін Д.Ю., к. пед.н, доц.
підпис ПІБ, вчене звання і ступінь	підпис ПІБ, вчене звання і ступінь
«___»_____2024 р.	«___»_____2024 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему: «Дослідження захищеної інформаційної системи з елементами штучного інтелекту на основі цифрових слідів»

Спеціальність: 123 «Комп’ютерна інженерія»

Освітня програма: Комп’ютерні системи і мережі

Орієнтація освітньої програми: Освітньо-професійна

Гарант освітньої програми

_____ _____ _____

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник дипломного проекту

професор, д.т.н. _____ Лахно В.А.

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Макаєв Вадим Вікторович

(підпис) (ПІБ студента)

КИЇВ – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

«ЗАТВЕРДЖУЮ»
завідувач кафедри
комп'ютерних систем, мереж та кібербезпеки
_____/ Касаткін Д.Ю., к.п.н., доц. /
підпис ПІБ, вчене звання і ступінь
«__» _____ 20__ р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТУ

Макаєв Вадим Вікторович
(прізвище, ім'я, по батькові)
Спеціальність (напрям підготовки): 123 «Комп'ютерна інженерія»
Тема кваліфікаційної бакалаврської роботи: «Дослідження захищеної інформаційної системи з елементами штучного інтелекту на основі цифрових слідів»
затверджена наказом ректора НУБіП України від "___" _____ _____ № _____
Термін подання завершеної роботи на кафедру _____
Вихідні дані до кваліфікаційної бакалаврської роботи _____
Перелік питань, що підлягають розробці:
1. Цифрові сліди, їх використання для виявлення атака, аналіз попередніх досліджень
2. Методологія дослідження, виявлення аномалій з використання нейронних мереж
3. Розроблення штучної нейронної мережі для виявлення аномалій в ІС

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання "_____"_____

Керівник бакалаврської роботи _____ Лахно В.А, професор, д.т.н
(підпис) (прізвище та ініціали)
Завдання прийняв до виконання _____ Макаєв В.В.
(підпис) (прізвище та ініціали студента)

РЕФЕРАТ

Пояснювальна записка: 60 сторінок, 34 рисунки, 3 таблиці, 8 лістингів коду, 3 додатки, 32 джерела

ІНФОРМАЦІЙНА СИСТЕМА, ЦИФРОВІ СЛІДИ, ЛОГИ, ЛОГ-АНАЛІЗ, АНОМАЛІЇ, АНАЛІЗ ЧАСОВИХ РЯДІВ, ПРОГНОЗУВАННЯ, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

Об’єкт дослідження – захищена інформаційна система університету.

Мета – дослідити та розробити методи аналізу журналів логування, прогнозування та виявлення аномалій у захищеній інформаційній системі університету з використання елементів штучного інтелекту.

Робота складається з трьох розділів:

У першому розділі досліджено використання цифрових слідів та їх використання для виявлення атак на інформаційні та інші системи, проведено аналіз попередніх досліджень прогнозування часових рядів з використанням статистичних методів аналізу.

У другому розділі розглянуто існуючі методи виявлення аномалій у часових рядах та проведено аналіз можливості використання штучних нейронних мереж для їх виявлення.

У третьому розділі розроблено метод прогнозування та виявлення аномальної активності у часовому ряді з використанням нейронної мережі з довгою короткостроковою пам'яттю (LSTM).

ЗМІСТ

ВСТУП

Більшість сучасних комп'ютерних систем створюють файли журналів. До лог-файлів часто звертаються для аналізу операційних аспектів роботи системи, оскільки вони містять дуже цінну інформацію про функціонування системи та діяльність користувачів у системі. При роботі з інформаційними системами навчальних закладів, особливо в глобальній перспективі переходу до онлайн-навчання, сотні користувачів – студентів і викладачів – майже щодня відвідують систему залишаючи там цифровий слід.

Цифрові сліди, які залишають користувачі під час взаємодії з цими системами, відіграють ключову роль. Вони не просто дозволяють ідентифікувати потенційні загрози та вразливості, але й слугують інструментом для глибшого розуміння поведінкових моделей користувачів, ефективності системних процесів та взаємодій між компонентами системи. Вони є важливим джерелом інформації для досліджень, оскільки також дають уявлення про зацікавленість користувачів у використанні електронних ресурсів у процесі навчання. Їх аналіз може виявити потенційні проблеми у використанні електронних ресурсів для покращення навчального процесу.

Через дуже велику кількість записів у журналах, що створюються в деяких системах, дуже складно шукати потрібну інформацію в файлах при лог-аналізу. Тому комп'ютерні методи аналізу логів з використанням методів штучного навчання є незамінними для пошуку потрібних даних.

Таким чином, актуальність роботи обумовлена не тільки необхідністю забезпечення безпеки в масштабах всієї системи, а й важливістю постійного аналізу та вдосконалення інформаційних систем.

1 АНАЛІЗ ПОПЕРЕДНІХ ДОСЛІДЖЕНЬ

1.1. Цифрові сліди та їх використання для виявлення атак

Цифрові або електронні сліди є записами дій користувачів в мереж Інтернет: від простих переходів по веб-сторінках до складних взаємодій з різними цифровими платформами. Майже кожна онлайн діяльність залишає свої відбитки, які можуть бути як видимими, так і прихованими. Деякі з них очевидні: наприклад, публікації в соціальних мережах, тоді як інші менш помітні, наприклад, файли cookie для відстеження користувачів на сайті [1][2].

Цифровий слід залишається назавжди після того, як дані стають загальнодоступними, і власнику цих даних стає складно контролювати їх подальше використання. Розуміння та керування своїми цифровими слідами стає вкрай важливим для особистої цифрової безпеки та конфіденційності. Знання процесів формування цифрових слідів та найбільш уразливих аспектів допомагає більш ефективно керувати цією інформацією та знижувати ризики, пов'язані з можливими витоками даних або кібератаками.

Загалом цифрову інформацію про користувачів в Інтернеті можна класифікувати за різними параметрами. Її можна досліджувати з погляду її походження – навмисного чи випадкового – і навіть доступності іншими особами. Зазвичай виділяють два види основних видів цифрових слідів: активні та пасивні [4].

Пасивний цифровий слід створюється під час взаємодії користувача безпосередньо з мережею Інтернет. У минулому історія пошукових запитів та відвіданих сайтів зберігалася за IP-адресою комп'ютера. З розвитком технологій власникам ПК доводиться реєструватися з особистими даними та IP-адресою перед використанням пристроїв. Таким чином, підключення до мережі та відвідування сайтів формують цифровий слід користувача. Університети та інші освітні установи

використовують системи автентифікації з використанням імен користувачів та паролів для відстеження дій студентів та викладачів. У цьому контексті пасивні цифрові сліди завжди пов’язані як з конкретним користувачем, так і з його IP-адресою [3][5]. Щоразу при виході до мережі студент залишає за собою цифровий слід: час відвідування сайту навчального закладу, тривалість онлайн-сесії, сторінки перегляду, а також всі дії на сайті, наприклад, завантаження файлів або відправка форм.

Активний цифровий слід можна визначити, як набір даних, який користувач створює свідомо: публікації в соціальних мережах, коментарі на сайтах, електронні повідомлення або інші форми активного спілкування [5]. Користувачі самостійно контролюють свої активні цифрові сліди, вирішуючи, що і коли публікувати та з ким ділитися цією інформацією. Важливо пам'ятати про те, що інші особи можуть використовувати ці цифрові сліди та навіть зловживати ними, якщо вони не забезпечені належними заходами безпеки. Такі сліди мають потенціал до серйозного впливу на репутацію користувача, оскільки можуть відображати його особисті інтереси, політичні переконання та соціальні зв'язки [3].

В контексті цифрової безпеки електронні сліди відіграють ключову роль. Так, вони використовуються у наступних напрямках [1][6]:

- Виявлення та реагування на інциденти безпеки, де цифрові сліди дозволяють вчасно виявляти підозрілі дії чи відхилення від норми, які можуть свідчити про можливі загрози безпеці даних: несанкціонований доступ до інформації, наявність шкідливого програмного забезпечення та інші небезпеки.
- Проведення розслідування після інциденту, так звана форензік-діагностика, що дозволяє уважно вивчити цифрові сліди та з'ясувати обставини того, як стався інцидент, хто за ним стоїть і які дані могли бути під ризиком витоку.

− Звіти та аудит, де цифрові сліди відіграють важливу роль у демонстрації відповідності вимогам під час перевірок та надання підтверджень про вжиті заходи безпеки.

− Аналіз загроз, де вивчення цифрових слідів проводиться з метою виявлення потенційних джерел загрози, методів їхньої атаки та цілей.

− Профілактика загроз, що допомагає організаціям виявляти слабкі місця у своїх системах та процесах, завдяки чому підприємства можуть активно вживати заходів для запобігання потенційним атакам у майбутньому.

При проведенні такого роду аналізу використовують лог-файли або файли журналів [7], згенеровані програмним забезпеченням, які містять інформацію про операції, дії та моделі використання додатків, серверів або IT-систем. Вони містять записи про всі процеси, події та повідомлення, а також додаткові описові дані, такі як тимчасові мітки, які показують, що і коли сталося всередині системи [8].

Журналюванням подій називається процедура збереження повідомлень про події в журналі, а журналом подій – файл, який модифікується у процесі додаванням повідомлень про них.

Подія – це запис, який фіксує специфічну дію або зміну стану, що відбулася в системі, програмному забезпеченні або пристрої. Ці записи генеруються автоматично системами логування для документування важливих чи підозрілих подій, які можуть включати широкий спектр дій, від простих запитів користувачів до системних помилок, транзакцій баз даних, змін конфігурації та інше [9][13].

Хоча точний формат може варіюватися в залежності від системи або програмного забезпечення, більшість подій у лог-файлах містить наступну інформацію [9]:

− Часова мітка. Дата та час, коли подія відбулася. Часові мітки дозволяють відстежувати коли саме сталася подія і є критично важливими для аналізу логів, особливо при налагодженні системи або розслідуванні інцидентів безпеки.

− Рівень серйозності. Вказує на важливість події. Рівні можуть варіюватися від DEBUG (інформація для розробників) до INFO (загальна інформація про роботу системи), WARN (попередження про потенційні проблеми), ERROR (помилки, що не зупиняють роботу системи), і CRITICAL або FATAL (критичні помилки, що можуть призвести до зупинки системи) [12].

− Джерело події. Ідентифікує компонент системи або програму, де відбулася подія. Це може бути назва програми, модуля, сервісу або навіть конкретного пристрою.

− Опис події. Детальний опис того, що сталося, включаючи будь-які специфічні для події дані, такі як повідомлення про помилку, ідентифікатори транзакцій, імена файлів тощо.

− Ідентифікатор користувача або сесії. Якщо подія пов'язана з діями користувача, лог може включати інформацію про користувача або сесію, під час якої відбулася подія.

Такі дані надають уявлення про продуктивність і відповідність вимогам додатків і систем. Майже кожен компонент системи та мережі генерує різні типи даних, і кожен компонент збирає ці дані у власному журналі. Через це існує багато типів лог-файлів [11][13] (таблиця 1.1).

Таблиця 1.1 – Типи лог-файлів

Тип лог-файлу	Призначення
Журнал подій	записує інформацію про мережевий трафік і використання мережі, наприклад, спроби входу в систему, невдалі спроби введення пароля та події програми
Журнал сервера	містить записи про дії, пов'язані з певним сервером за певний період часу

Продовження таблиці 1.1

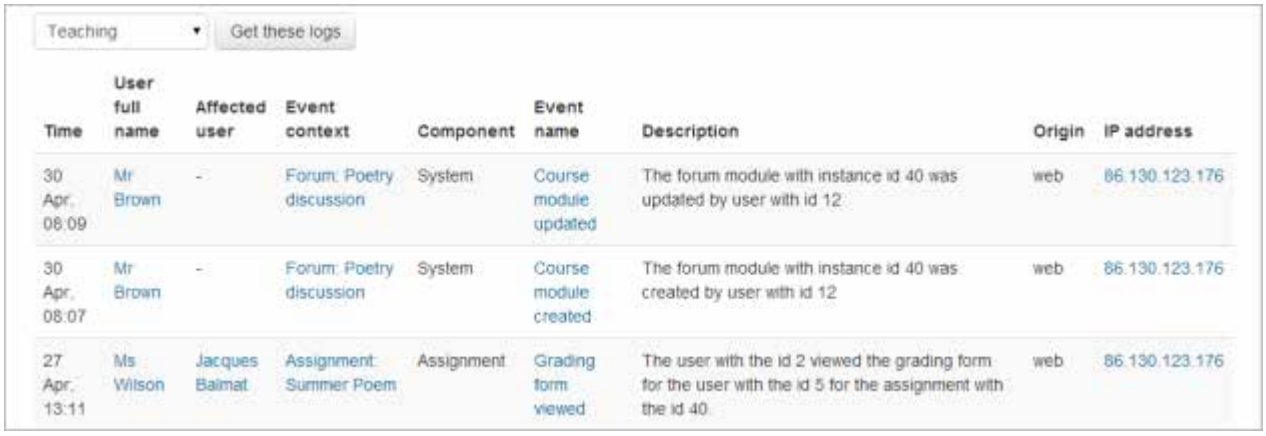
Тип лог-файлу	Призначення
Системний журнал	запис подій операційної системи, що включає в себе повідомлення про запуск, системні зміни, несподівані вимкнення, помилки та попередження, а також інші важливі процеси
Журнал подій	записує інформацію про мережевий трафік і використання мережі, наприклад, спроби входу в систему, невдалі спроби введення пароля та події програми
Журнал сервера	містить записи про дії, пов'язані з певним сервером за певний період часу
Системний журнал	запис подій операційної системи, що включає в себе повідомлення про запуск, системні зміни, несподівані вимкнення, помилки та попередження, а також інші важливі процеси

Лог-файли діють як сховища для записів, що документують події в системі чи програмному забезпеченні. Цей процес передбачає збір повної інформації про всю взаємодію та операції в системі, де кожна подія автоматично фіксується в файлі журналювання зі специфічними характеристиками цифрового сліду [9][16]:

- Час доступу до системи або сервісу – дата та час, коли користувач виконав вхід або вихід з системи.
- Дії користувача – операції, які користувач виконав у системі, наприклад, відкриття файлів, запуск програм, виконання транзакцій тощо.
- IP-адреси та місцезнаходження – інформація про IP-адресу, з якої користувач доступав до сервісу, що може дати змогу визначити його приблизне місцезнаходження.

– Інформація про пристрій та браузер – дані про тип пристрою, операційну систему, браузер та його версію, які користувач використовував для доступу до сервісу.

Більшість програм зберігають дані, що містять інформацію, де кожна подія описується набором користувачів, які виконували дії над об'єктами в певний момент часу або протягом певного інтервалу [13]. Наприклад, Moodle, одна з найпопулярніших програмних платформ для електронного навчання, створює лог-файл, що містить кожну дію, виконану користувачем (вхід, читання, завантаження тощо) із зазначенням його ролі, IP-адреси та багатьох інших параметрів [14][15] (рисунок 1.1).


Teaching ▾ Get these logs

Time	User full name	Affected user	Event context	Component	Event name	Description	Origin	IP address
30 Apr, 08:09	Mr Brown	-	Forum: Poetry discussion	System	Course module updated	The forum module with instance id 40 was updated by user with id 12	web	86.130.123.176
30 Apr, 08:07	Mr Brown	-	Forum: Poetry discussion	System	Course module created	The forum module with instance id 40 was created by user with id 12	web	86.130.123.176
27 Apr, 13:11	Ms Wilson	Jacques Balmat	Assignment: Summer Poem	Assignment	Grading form viewed	The user with the id 2 viewed the grading form for the user with the id 5 for the assignment with the id 40	web	86.130.123.176

Рисунок 1.1 – Приклад лог-файлу, створеного навчальною платформою Moodle

Студенти та викладачі, які чи не щодня відвідують та взаємодіють зі сторінками навчальних платформ та дотичними до них ресурсами , самі того не усвідомлюючи, залишають значну кількість цифрових слідів під час користування ними (таблиця 1.2). Вони можуть бути цінним джерелом інформації для дослідження, оскільки дозволяють отримати глибоке розуміння поведінки користувачів та їх залученості у використанні електронних ресурсів у процесі навчання, а їх аналіз може допомогти визначити потенційні проблеми, які виникають під час користування ними задля покращення навчального процесу [15].

Таблиця 1.2 – цифрові сліди для аналізу навчальних інформаційних систем

Пасивні ЦС	Активні ЦС
дані, отримані від спеціально виділених учасників заходів, які виконують функцію фіксації цифрового сліду	дані, отримані від учасників освітньої діяльності, які надходять від учня у формі анкет, опитувань тощо
дані месенджерів, засобів організації онлайн-конференцій і соціальних мереж, задіяних у процесі навчання	дані, що є формальним свідченням виконаної активності у формі оцінок, відміток відвідуваності тощо
дані, що генеруються в середовищах розроблення або спільної організації робіт над проектами	дані, зібрані від студентів через мобільні додатки навчального закладу, включаючи використання освітніх додатків, відслідковування участі в навчальних заходах та поведінкові дані
дані цифрових платформ онлайн-навчання, таких, як Moodle	дані соціальної взаємодії студентів, такі як участь в дискусійних форумах, відгуки на лекції та семінари, а також залученість у студентські групи і спільноти в соціальних мережах

Аналіз логів – це комплексна процедура, яка включає збір даних і впорядкування їх у потрібному форматі для детального вивчення та виявлення тенденцій і аномалій, а також важливих подій, які впливають на продуктивність системи чи можуть бути використані для її вдосконалення з метою безпеки [16].

Аномалії — це, по суті, точки даних, які відрізняються від більшості даних і виявляють поведінку, яка не є типовою або очікуваною в рамках тенденцій наборів даних і моделей поведінки, які вважаються прийнятними або стандартними. Виявлення аномалій або викидів включає точне визначення точок даних, які

відхиляються від очікуваних моделей у наборі даних [17]. Так, до деяких випадків аномальної активності, можна віднести: велику кількість спроб вхожу з певних IP-адрес, що може сигналізувати про потенційну спробу злому, або раптове збільшення кількості системних збоїв, сплеск мережевої активності внаслідок непередбачуваного збільшення обсягу трафіку чи поведінка користувачів, яка є нетиповою для нього.

У наступному розділі буде розглянуто та продемонстровано результати попередніх досліджень, що були проведені задля аналізу лог-файлів даних користувачів інформаційної системи Національно університету біоресурсів та природокористування України з метою виявлення сталих тенденцій та підозрілої активності впродовж начального року з використанням статистичних методів аналізу даних.

1.2. Результати попередніх досліджень при здійсненні лог-аналізу інформаційної системи університету

Перед тим, як перейти до результатів попередньої оцінки при дослідженні інформаційної системи університету, варто трохи детальніше заглибитись у розуміння того, що таке часові ряди, метод яких був застосований для аналізу активності студентів.

Часовий ряд – це послідовність чисел, де кожен елемент являє собою значення деякого процесу, що відбувається у послідовні рівні проміжки часу та дозволяє відстежувати зміни в часі, надаючи цінну інформацію про тенденції, закономірності та взаємозв’язки. Дані часових рядів поширені в різних сферах, включаючи фінансові ринки, дані датчиків, аналітика та веб-аналітика, Інтернет речей, соціальні мережі та інші інформаційні системи. Це дає змогу отримувати інформацію з необхідних систем в режимі реального часу, дозволяючи аналізувати їх та прогнозувати потенційні проблеми [18][19].

Дані часових рядів можуть набувати різних форм залежно від характеру спостережень. Два основних типи даних часових рядів – безперервні та дискретні.

Безперервні дані часових рядів збираються безперервно в часі. Прикладами можуть бути вимірювання температури, що реєструються щогодини, або біржові ціни, які оновлюються щосекунди. У межах безперервних часових рядів існують різні підтипи, які можна дослідити далі. Наприклад, періодичні часові ряди – це дані, які повторюються через певний проміжок часу, наприклад, щоденні коливання температури або щотижнева відвідуваність веб-сайтів [19].

Дискретні дані часових рядів збираються і записуються через певні проміжки часу. Наприклад, щомісячний звіт про продажі або річний темп зростання ВВП є формами дискретних часових рядів. З іншого боку, нерегулярні дані часових рядів не відповідають певному шаблону і можуть мати випадкові коливання або аномалії. Наприклад, дані про події можна вважати нерегулярними даними часових рядів, вони стосуються записів подій, які відбуваються в певні моменти часу, часто без передбачуваної закономірності. Це призводить до того, що часові позначки не відповідають регулярному інтервалу, що робить їх нерегулярними. Прикладами можуть бути дії користувача на веб-сайті, сповіщення датчиків або журнали транзакцій. Кожна подія реєструється, коли вона відбувається, створюючи часовий ряд з різними інтервалами між точками даних. Дискретні дані часових рядів також можна класифікувати на різні підтипи залежно від часових інтервалів, в які збираються дані. Деякі приклади включають щоденні, щотижневі, щомісячні, щоквартальні або річні дані. Кожен тип дискретних часових рядів має свої унікальні характеристики і може вимагати різних аналітичних підходів [19].

Для аналізу активності студенів університету, було використано дані з журналу логів навчальної платформи Moodle за 2021 рік (таблиця 1.3). Параметри журналу Moodle включають журнали активності в курсі, журнали активності на сайті, журнали активності в реальному часі, налаштування адміністрування сайту

та можливості перегляду журналів – хоча до уваги бралися лише журнали активності в курсі.

Таблиця 1.3 – Набір логів, отриманих з системи Moodle про активність студентів

Атрибут	Опис атрибуту
Time	Мітка часу кожного користувача в конкретному модулі, з дискретним типом даних, наприклад, «30/06/18, 22:22»
User full name	Зареєстроване ім'я користувача в Moodle при вході в систему
Affected user	Зареєстрований користувач, коли він завантажує будь-який файл у Moodle
Event context	Діяльність, до якої зареєстрований користувач отримує доступ у Moodle, з типом номінальних даних, наприклад, «File: Week4»
Component	Компонент, до якого зареєстрований користувач має доступ у Moodle, з аномальним типом даних, наприклад, «System/File»
Event name	Конкретна подія, що викликається в Moodle зареєстрованим користувачем, з аномальним типом даних, наприклад, «Course module viewed»
Description	Опис дії, виконаної в Moodle зареєстрованим користувачем, з номінальним типом даних, таким як «The user with id ‘4357’ viewed the course with id ‘158’»
Origin	Походження дії, виконаної в Moodle зареєстрованим користувачем, з номінальним типом даних типу «web»
IP address	Адреса інтернет-протоколу зареєстрованого користувача, який отримав доступ до Moodle, з номінальним типом даних «192.168.99.3»

Лістинг коду 1.1 – Візуалізація активності студентів з використанням часового ряду

```
import pandas as pd
import matplotlib.pyplot as plt

activity_data = pd.read_csv(r"data/Moodle Module 4 SP18.csv", encoding='ISO-8859-1')

# Перетворення стовпця 'Time' у формат datetime
activity_data['Time'] = pd.to_datetime(activity_data['Time'], format='%d/%m/%y, %H:%M')

# Створення нового стовпця для дати без часу
activity_data['Date'] = activity_data['Time'].dt.date

# Групування подій за днями та підрахунок кількості подій на кожен день
events_by_day = activity_data.groupby('Date').size()

# Перетворення індексу на формат datetime
events_by_day.index = pd.to_datetime(events_by_day.index)

events_by_day.plot()

plt.title('Кількість подій на день', fontsize=20)
plt.xlabel('Дата', fontsize=16)
plt.ylabel('Кількість подій', fontsize=16)
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()
```

Для візуалізації використовувалась мова програмування Python з використанням бібліотеки Pandas для аналізу великих наборів даних та Matplotlib для візуалізації даних (лістинг 1.1). Графік нижче (рисунок 1.2) наглядно демонструє загальний вигляд часового ряду щоденної активності студентів університету в період з березня по липень 2021 року, де вісь «Кількість подій» відображає кількість звернень до системи Moodle: перегляд сторінки курсу, оновлення даних на сторінці курсу, завершення курсу тощо.

Рисунок 1.2 – Візуалазація часового ряду

Щоб краще пояснити характеристики даних часових рядів, їх можна розбити на чотири складові [19][21]:

- Тренд T(t) – плавно змінювана, не циклічна компонента, що описує чистий вплив довготривалих чинників, ефект яких позначається поступово. Він показує загальний спад або зростання за певний період часу.
- Сезонність S(t) – це регулярні періодичні явища, що відбуваються протягом заданого проміжку часу (року, місяця, тижня, дня). Сезонні дані демонструють коливання, зафіксовані за величиною, напрямком і часом.
- Залишковість R(t) – охоплює короткострокові нерегулярні коливання, шум або залишкову мінливість даних, які не можуть бути пояснені іншими компонентами. Вона включає непередбачувані та хаотичні відхилення після врахування циклічності, сезонності та тенденцій.
- Циклічність C(t) – це повторювані коливання, які не мають фіксованого періоду, тривають недостатньо довго, щоб вважатися тенденціями (але довше, ніж нерівномірності), і не мають постійної тривалості або амплітуди.

Взаємозв’язок цих компонент одна з одною переважно залежить від типу моделі часового ряду: адитивної чи мультиплікативної [21][22].

$$Y(t) = T(t) + S(t) + R(t) + C(t), \quad (1.1)$$

$$Y(t) = T(t) \times S(t) \times R(t) \times C(t), \quad (1.2)$$

де: *Y(t)* – спостережуване значення часового ряду на момент часу t;

T(t) – тренд;

S(t) – сезонність;

R(t) – залишковість;

C(t) – циклічність.

У адитивній моделі усі складові додаються один до одного (формула 1.1), оскільки зміни в одній із компонент не впливають на іншу, тобто сезонність чи залишковість однакові на всіх рівнях тренду, що підходить для випадків, коли сезонність постійна та не змінюється у залежності від рівня тренду, мультиплікативна модель навпаки підходить для даних, де сезонність та залишковість збільшуються разом зі збільшенням тренду, тобто коли одна із складових змінюється, її вплив пропорційно змінює інші складові (формула 1.2), що продемонстровано на графіках нижче [21][22] (рисунок 1.3-1.4). Нижче наведено код для декомпозиція часових рядів з використанням бібліотеки Statsmodels для розбиття рядів на компоненти: тренд, сезонність та залишковість (лістинг 1.2).

Для прогнозування та аналізу часових рядів використовуються три моделі [21]: ARMA (авторегресійне середнє ковзне), ARIMA (авторегресійне інтегроване середнє ковзне) та SARIMA (сезонне ARIMA).

Авторегресійна модель робить прогнози на основі попередніх спостережень, які можна виразити як AR(p), де p вказує на кількість попередніх точок даних, які потрібно розглянути [21].

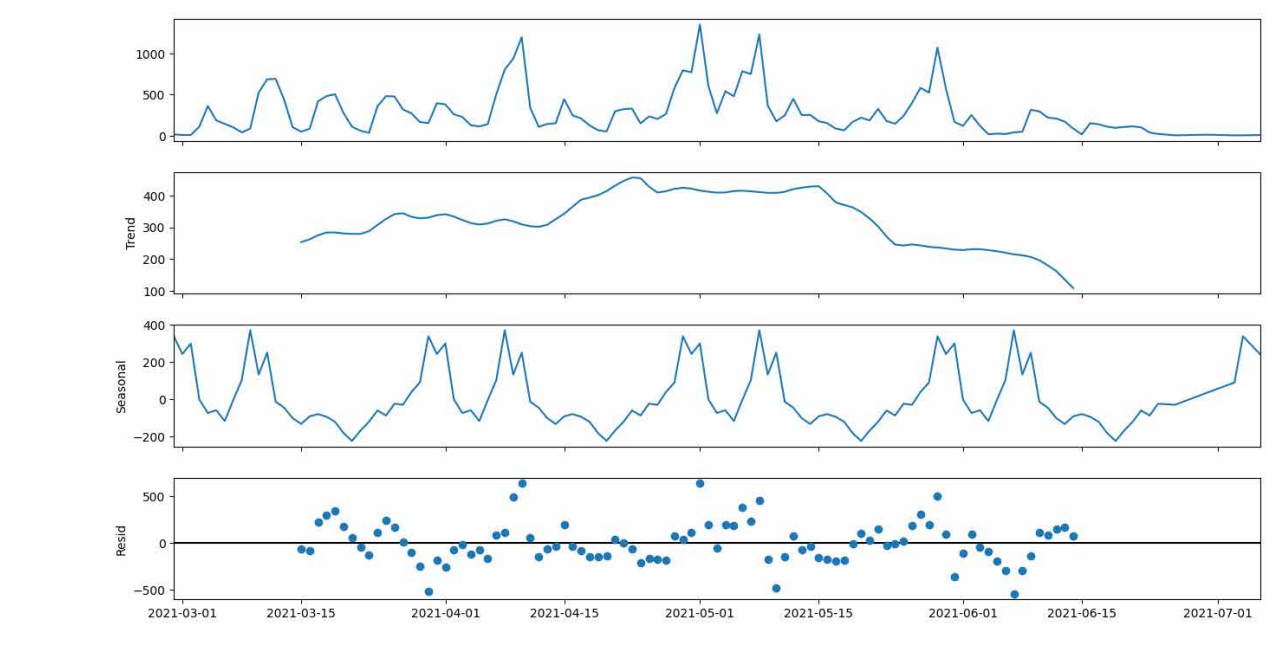


Рисунок 1.3 – Розбиття часового ряду на складові з використанням адитивної моделі

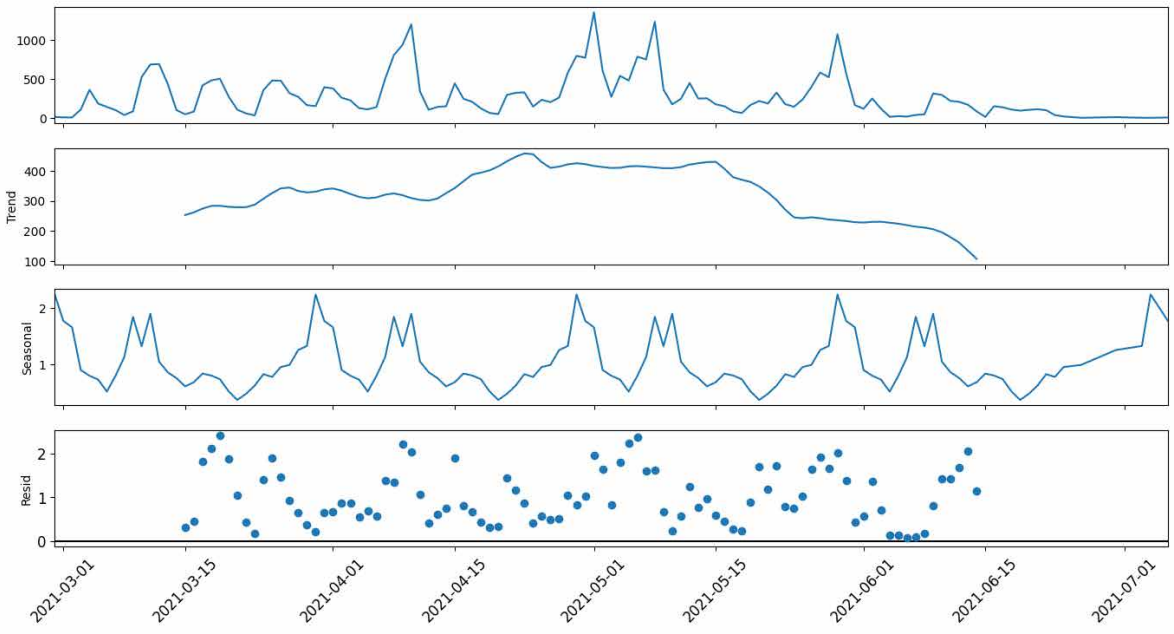


Рисунок 1.4 – Розбиття часового ряду на складові з використанням мультиплікативної моделі

Лістинг коду 1.2 – Декомпозиція часового ряду

```
from statsmodels.tsa.seasonal import seasonal_decompose as sm

decomposition_additive = sm(events_by_day, model='additive', period=30)
fig_additive = decomposition_additive.plot()

decomposition_multiplicative = sm(events_by_day, model='multiplicative',
period=30)
fig_multiplicative = decomposition_multiplicative.plot()

plt.tight_layout()
plt.xticks(rotation=45, fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```

Модель авторегресії порядку p матиме вигляд [20][(формула 1.3)]:

$$AR(X_t,p) = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p}, \quad (1.3)$$

де: X_t – значення змінної у момент часу t;

$\phi_1, \phi_2, ..., \phi_p$ – вагові коефіцієнти моделі;

p – порядок авторегресії, кількість попередніх значень, що використовуються для прогнозу.

Зазвичай для визначення значення p використовується функція часткової автокореляції PACF (рисунок 1.5). Для даного спостереження в часовому ряді X_t, значення p корелювати із запізнілим спостереженням X_{t-1}, на яке також впливають його попередні значення. PACF візуалізує прямий внесок попередніх спостережень на поточні, виключаючи вплив проміжних лагів.

Наприклад, наведений нище PACF при лазі 1 становить близько 0.75, що вказує на сильну кореляцію між поточним спостереженням X_t та попереднім X_{t-1}. На наступних лагах з 2 по 5 автокореляція знижується, що вказує на те, що вплив попередніх значень на поточні зменшується і вони не мають великого впливу на модель. Значення p для моделі AR(p) визначається тоді, коли PACF вперше різко

опускається нижче значущого порогу, тому можна говорити про те, що оптимальне значення для моделі буде рівна p = 1 [21].

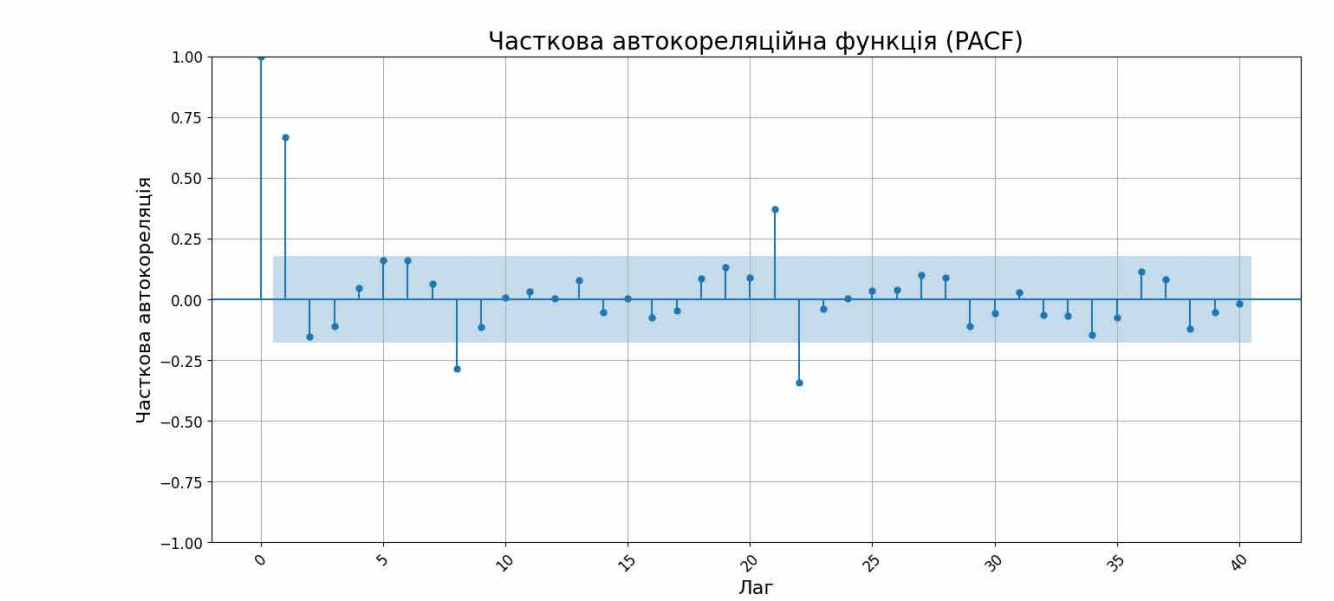


Рисунок 1.5 – Визначення параметра p з використанням PACF

Модель ковзного середнього MR(q) налаштовує модель на основі середніх помилок прогнозів з попередніх q спостережень для прогнозування майбутніх значень. Модель ковзного середнього можна позначити наступним рівнянням [20] (формула 1.4):

$$MA(X_t,q) = e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + ... + \theta_q e_{t-q}, \quad (1.4)$$

де: X_t – значення змінної у момент часу t;

e_t – випадкова похибка у момент часу t;

$\theta_1, \theta_2, ..., \theta_p$ – вагові коефіцієнти моделі;

q – порядок авторегресії, кількість похибок, що використовуються для прогнозу.

Для визначення параметра q використовується автокореляційна функція ACF (рисунок 1.6). На графіку видно, що після першого лагу значення автокореляції починають спадати і знаходяться в межах довірчого інтервалу, тому оптимальне значення q так само буде рівне q = 1 [21].

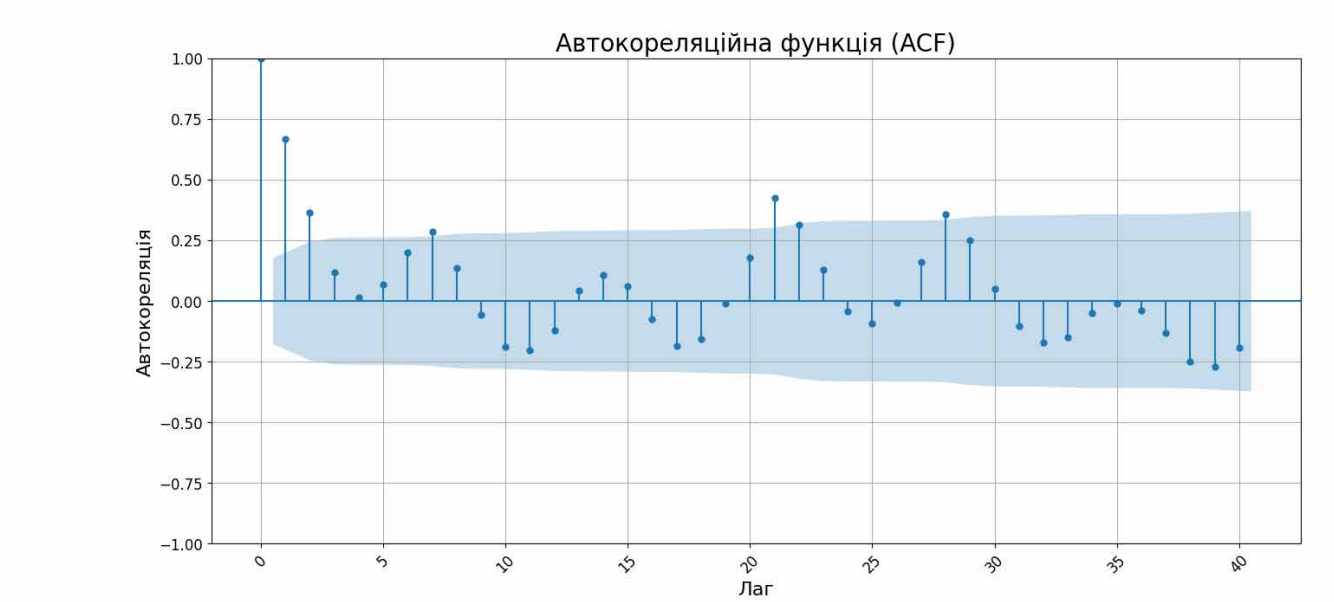


Рисунок 1.6 – Визначення параметра q з використанням ACF

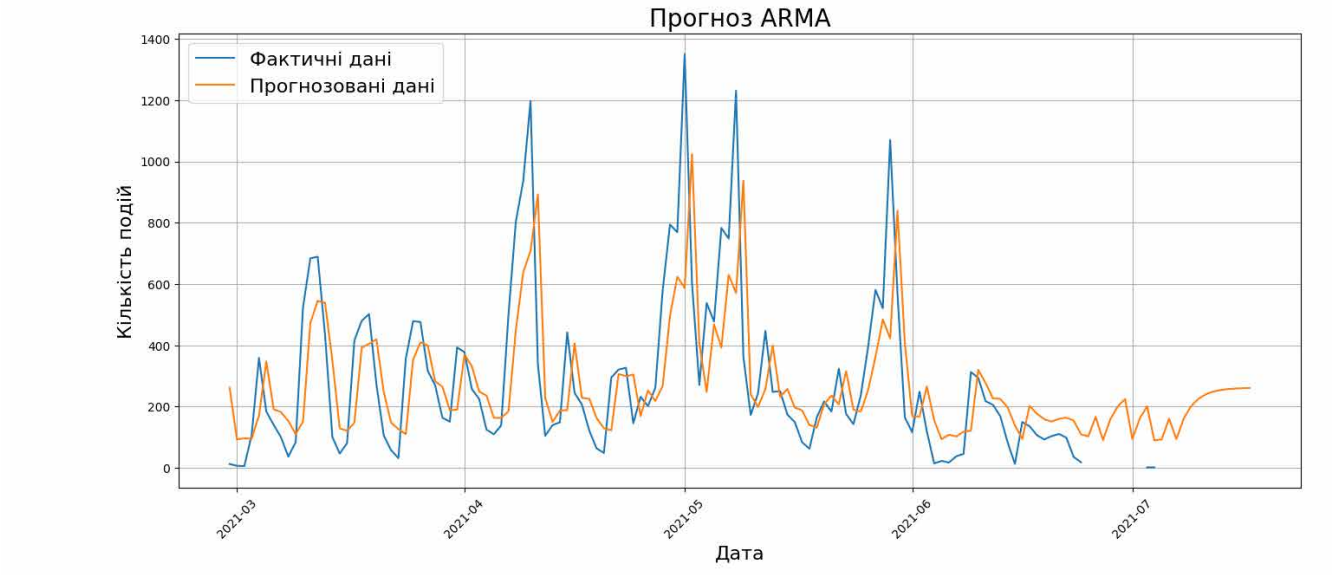


Рисунок 1.7 – Прогнозування часового ряду з використанням моделі ARMA

Лістинг коду 1.3 – Побудова моделі ARMA

```
from statsmodels.tsa.arima.model import ARIMA

# Побудова ARMA моделі з порядком (1, 0, 1)
ARMA_model = ARIMA(events_by_day, order=(1, 0, 1)).fit()
print(ARMA_model.summary())

# Прогнозування значень
forecast_steps = len(events_by_day) + 10
df_pred = ARMA_model.predict(start=0, end=forecast_steps)
```

ARIMA базується на авторегресійній моделі та моделі ковзного середнього (MA), вводячи ступінь відмінності компонентів, що задається параметром d – ARIMA (p, d, q). Це робиться для випадків, коли в даних часового ряду спостерігається очевидна тенденція. При використанні моделі ARMA цей параметр був рівний 0 [21].

Лістинг коду 1.4 – Перевірка ряду на стаціонарність з використанням ADF-тесту

```
from statsmodels.tsa.stattools import adfuller

adf_test = adfuller(events_by_day)
print(f'ADF Statistic: {adf_test[0]}')
print(f'p-value: {adf_test[1]}')
```

Оскільки ARIMA включає диференціювання в процес побудови моделі, вона не вимагає, щоб навчальні дані були стаціонарними. Для того, щоб модель ARIMA працювала добре, слід вибрати відповідний ступінь диференціювання, щоб часовий ряд був перетворений на стаціонарні дані після вилучення тренду. Для цього можна використати ADF-тест, щоб визначити, чи дані вже стаціонарні: якщо дані стаціонарні, то диференціювання не потрібне, у таких випадках d = 0 [21]. При виконанні зазначеного коду (лістинг 1.4), ADF-тест показує значення p-значення рівним 0.032. Якщо p-value ADF-тесту менше значення 0.05, то виконувати потреба у виконанні диференціюванні відсутня.

Для побудови моделі ARIMA можна використати ту саму функцію, що й для моделі ARMA (лістинг 1.3), і хоча після проведених тестів було визначено, що для даного часового ряду виконувати диференціювання не потрібно і параметр d = 0, однак для фактичної перевірки параметра d буде набувати значення d = 1 [21]. З результатів на графіку (рисунок 1.8) можна побачити, що точність прогнозування майже не змінилась, що підтверджує описані вище твердження.

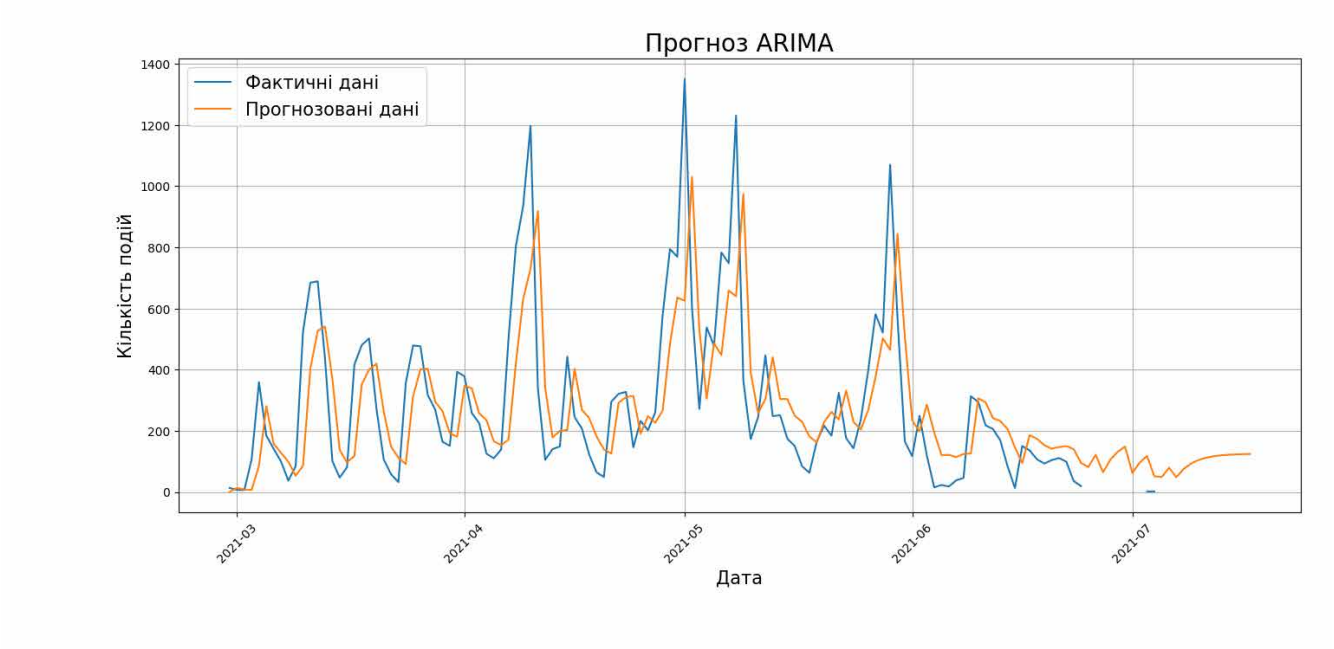


Рисунок 1.8 – Прогнозування часового ряду з використанням моделі ARIMA

Модель SARIMA є розширеною моделлю ARIMA та дозволяє розглядати періодичну закономірність, що спостерігається у часовому ряді. SARIMA включає як не-сезонні компоненти ARIMA (p, d, q), так і сезонні компоненти (P, D, Q, s), де P, D, Q – це члени авторегресії, диференціювання та ковзного середнього сезонного порядку відповідно, а s – кількість спостережень у кожному періоді [21]. Щоб визначити параметр s, можна скористатись графіком автокореляційної функції ACF (рисунок 1.6). На графіку можна помітити незначні піки автокореляції на лагових значеннях 7, 14, 21 та 28. І хоча піки не виражені яскраво, спостерігається певна

щотижнева активність, тому можна припустити, що параметр s для SAMIRA-моделі буде дорівнювати s = 7. Лістинг коду із застосування моделі SARIMA наведено нижче (лістинг 1.5).

Лістинг коду 1.5 – Визначенні функції для побудови моделі SARIMA

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

SARIMA_model = SARIMAX(events_by_day, order=(1, 0, 1), seasonal_order=(1, 0, 1,
7)).fit()
```

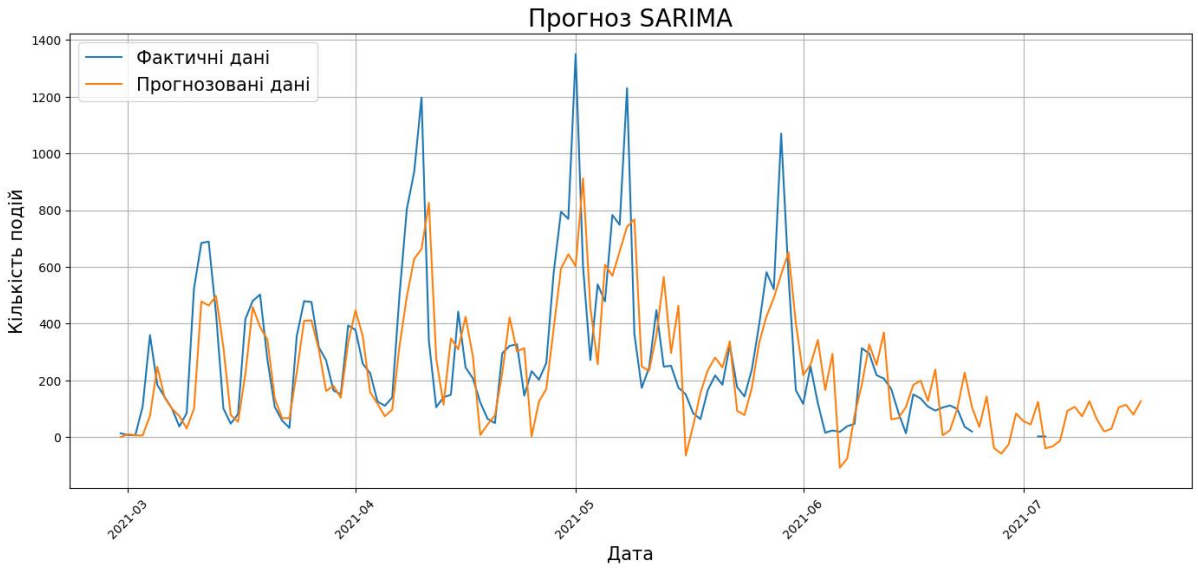


Рисунок 1.9 – Прогнозування часового ряду з використанням моделі SARIMA

На графіку прогнозування з використанням SAMIRA (рисунок 1.9) можна побачити, що прогнозована пряма хоч і повторює основні тренди, однак має розбіжності з фактичними даними. При детальному розгляді результатів підсумку аналізу моделі (рисунок 1.10) можна побачити, що значення коефіцієнту ma.L1 має p-значення, що рівне -0.0667, і оскільки воно перевищує рівень значущості 0.05, то параметр ковзного середнього MA(q) є статично незначущим, тому не має суттєвого впливу на модель.

SARIMAX Results

Dep. Variable:	y	No. Observations:	129
Model:	SARIMAX(1, 0, 1)x(1, 0, 1, 7)	Log Likelihood	-807.552
Date:	Tue, 01 Oct 2024	AIC	1625.105
Time:	23:56:57	BIC	1639.404
Sample:	02-28-2018	HQIC	1630.915
	- 07-06-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.7589	0.078	9.678	0.000	0.605	0.913
ma.L1	-0.0667	0.138	-0.483	0.629	-0.337	0.204
ar.S.L7	0.9389	0.052	17.985	0.000	0.837	1.041
ma.S.L7	-0.7311	0.103	-7.076	0.000	-0.934	-0.529
sigma2	3.107e+04	2985.354	10.406	0.000	2.52e+04	3.69e+04

Ljung-Box (L1) (Q):	0.01	Jarque-Bera (JB):	66.02
Prob(Q):	0.94	Prob(JB):	0.00
Heteroskedasticity (H):	0.58	Skew:	0.77
Prob(H) (two-sided):	0.08	Kurtosis:	6.15

Рисунок 1.10 – Результати аналізу моделі SAMIRA

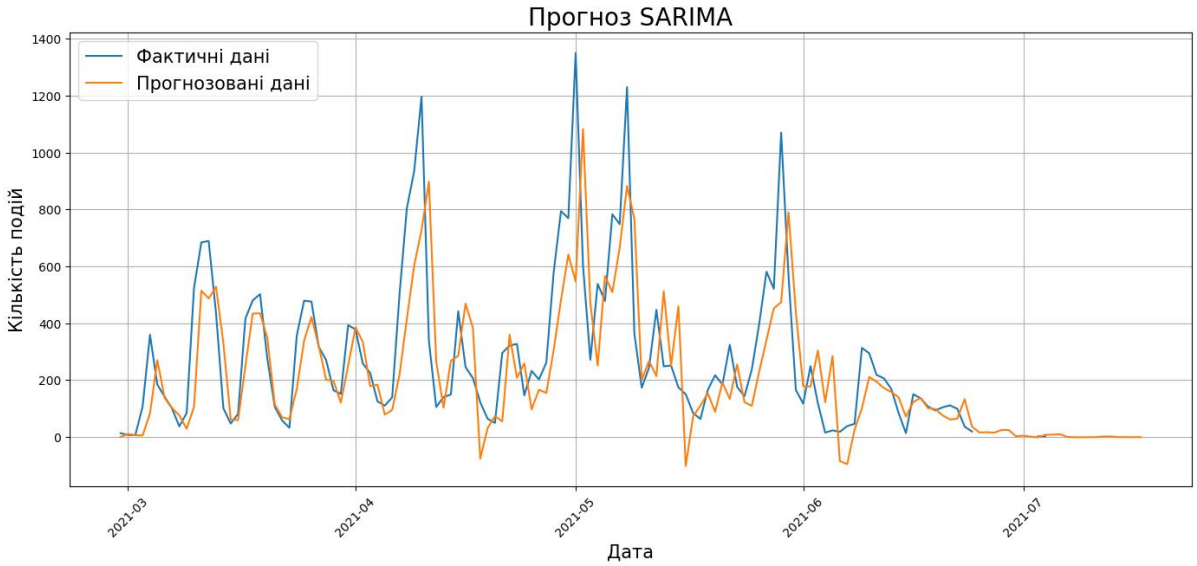


Рисунок 1.11 – Прогнозування часового ряду з використанням моделі SARIMA після виключення параметра ковзного середнього

При встановленні параметра q = 0, можна помітити, наскільки покращились прогнозування моделі (рисунок 1.11). З отриманих результатів видно, що модель гарно передбачає сезонність та циклічні коливання, однак не повністю охоплює різкість змін часового ряду, що можна побачити по тому, як піки прогнозованих даних не досягають фактичних.

Хоча графічно може здатись, що модель відпрацьовує доволі точно, однак корінь середньоквадратичної похибки RMSE набуває великих значень. Середня похибка між фактичними та прогнозованими значеннями моделі становить 138.28, що говорить про недоліки моделі чи її вибору. При аналізі часових рядів дані повинні бути однорідними та сезонними, тобто мати повторювані патерни, за якими модель може виконувати прогнозування. Оскільки дані, на яких виконувався аналіз, містили лише записи за чотири місяці 2021 навчального року, то можна припустити, що вони були недостатньо однорідними для навчання моделі та прогнозу.

1.3. Постановка цілей та завдань роботи

Хоча використання моделей ARMA, ARIMA та SAMIRA у рамках поточного дослідження залученості студентів у процес навчання дозволило провести початковий аналіз і прогнозування, однак недостатня однорідність даних, нерегулярні патерни та змінна сезонність виявили обмеження цих моделей, через що результати виявились недостатньо точними.

У наступних розділах магістерської роботи буде проведено агрегацію та вибірку даних про активність студентів з логів електронної навчальної системи Moodle для підвищення точності прогнозування, а також використано інші, більш сучасні, методи аналізу даних з використанням штучного інтелекту, щоб спробувати покращити якість прогнозування при використанні постійно змінюваних і нестаціонарих даних.

2 МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ

2.1. Аномалії та їх виявлення із застосуванням методів машинного навчання

Виявлення аномалій або відхилень відноситься до проблеми пошуку шаблонів у даних, які не відповідають очікуваній поведінці. Ці невідповідні шаблони часто називають аномаліями, викидами, неузгодженими спостереженнями, відхиленнями або забрудненнями в різних сферах застосування. Цей процес знайшов широке застосування в найрізноманітніших галузях, наприклад: у фінансовій сфері для виявлення шахрайства, у виробництві для виявлення дефектів або несправностей обладнання, у кібербезпеці для виявлення незвичної мережевої активності і в охороні здоров'я для виявлення аномальних станів пацієнтів. Різні наукові галузі використовують різні визначення, однак широкого визнання серед них усіх отримав опис, запропонований Хокінсом: «Аномалія – це спостереження, яке настільки різко відрізняється від інших, що викликає питання про механізм, який його спричинив» [24].

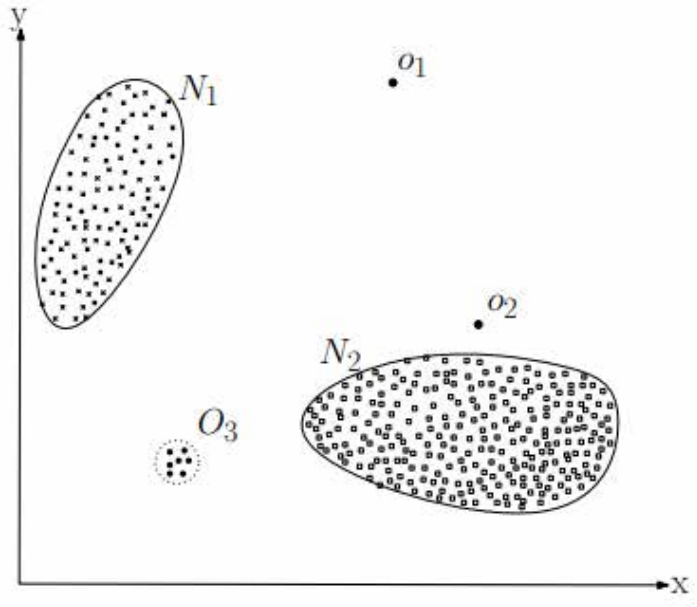


Рисунок 2.1 – Приклад точкової аномалій

Ключовим аспектом будь-якого методу виявлення аномалій є характер вхідних даних. Вхідні дані, як правило, являють собою набір екземплярів даних, які можна класифікувати за наступними трьома категоріями: точкові, контекстні та групові.

Якщо окремий екземпляр даних можна вважати аномальним по відношенню до решти даних, то такий екземпляр називається точковою аномалією. Це найпростіший тип аномалії, на якому зосереджено більшість досліджень з виявлення аномалій. На рисунку вище (рисунок 2.1) проілюстровано аномалії у простому двовимірному наборі даних. Дані мають дві нормальні області, N_1 і N_2, оскільки більшість спостережень лежать у цих двох областях. Точки o_1 і o_2, а також точки в області O_3 лежать поза межами нормальних областей, а отже, є точковими аномаліями, оскільки відрізняються від нормальних точок даних [17].

Коли екземпляр даних є аномальним у певному контексті, то його називають контекстною аномалією. Кожен такий екземпляр даних визначається за допомогою наступних двох наборів атрибутів:

- Контекстні атрибути. Контекстні атрибути використовуються для визначення контексту для цього екземпляра. Наприклад, у часових рядах даних час є контекстним атрибутом, який визначає позицію екземпляра у всій послідовності.
- Поведінкові атрибути. Поведінкові атрибути визначають неконтекстні характеристики екземпляра. Наприклад, у наборі даних, що описує середню кількість опадів у всьому світі, кількість опадів у окремому місці є поведінковим атрибутом.

Екземпляр даних може бути контекстною аномалією, але ідентичний екземпляр даних з точки зору поведінкових атрибутів може вважатися нормальним в іншому контексті. Наприклад, у часовому ряді, який відображає щомісячну температуру впродовж чотирьох років (рисунок 2.2), температура в момент часу t_1 є такою ж, як і в момент часу t_2, але відбувається в іншому контексті і тому не вважається аномалією [17].

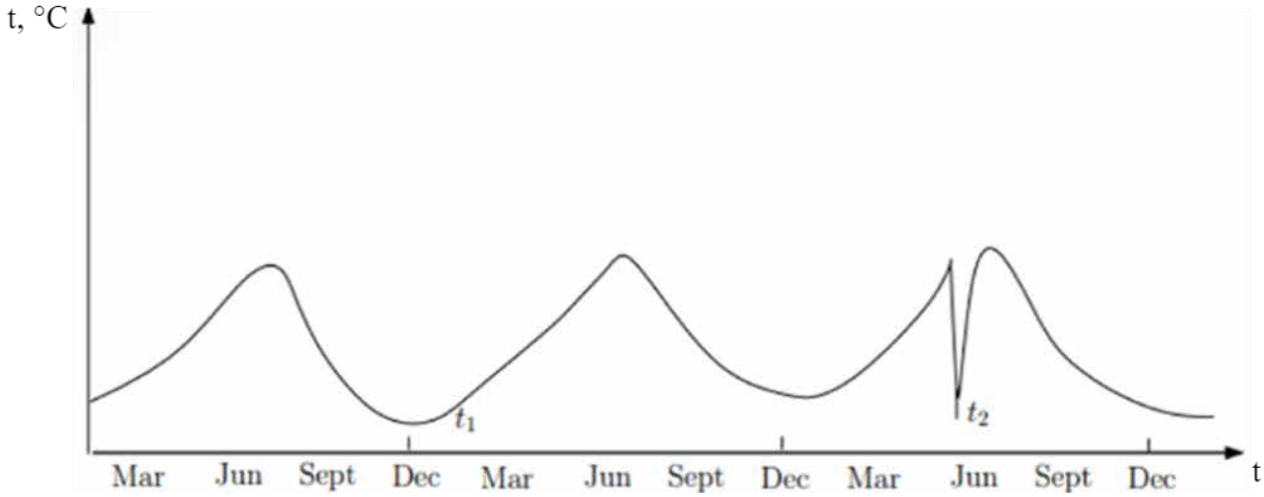


Рисунок 2.2 – Приклад контекстної аномалії

Колективною аномалією називається сукупність пов'язаних екземплярів даних є аномальною по відношенню до всього набору даних. Окремі екземпляри даних у колективній аномалії можуть не бути аномаліями самі по собі, але їхня поява разом є аномальною. Наприклад, на вихідній електрокардіограми людини (рисунок 2.3) виділена область позначає аномалію, оскільки одне і те ж низьке значення існує протягом аномально тривалого часу [17].

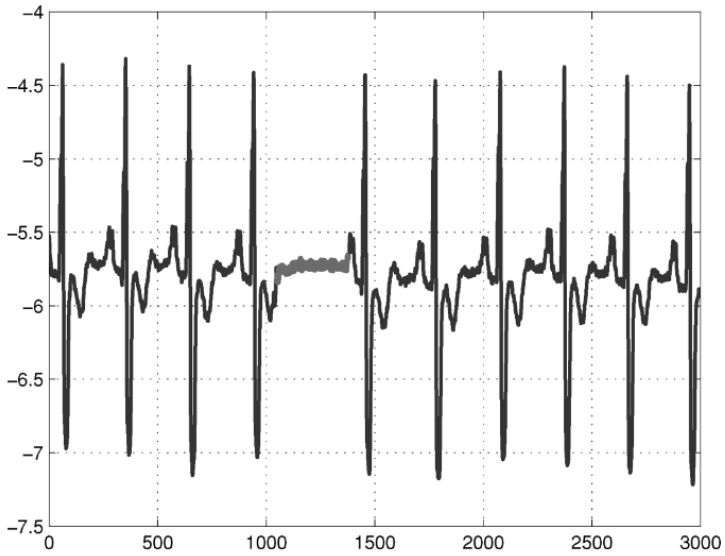


Рисунок 2.3 – Приклад колективної аномалії

Аномалії в часових рядах можна класифікувати як тимчасові, міжметричні або тимчасово-міжметричні аномалії, а також можна порівнювати або з сусідами (локальні), або з усім часовим рядом (глобальні), і вони мають різні форми залежно від їхньої поведінки [23].

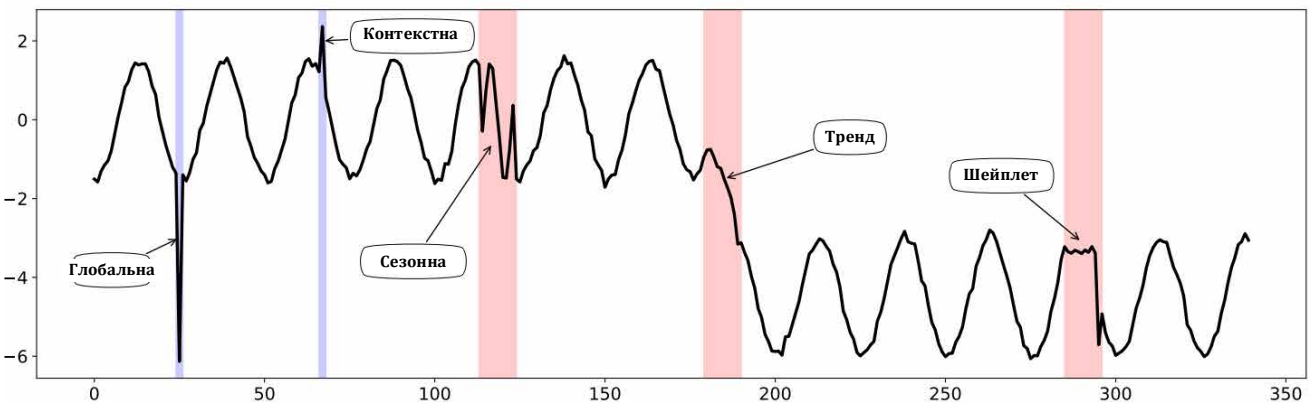


Рисунок 2.4 – Різні типи аномалій на одновимірному часовому ряді

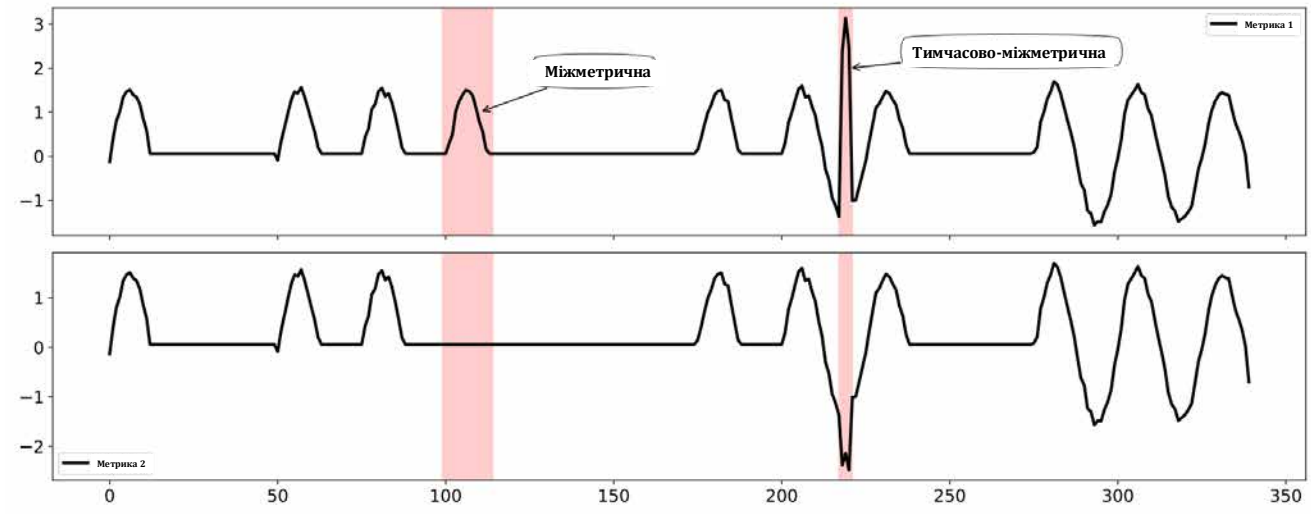


Рисунок 2.5 – Різні типи аномалій на багатовимірному часовому ряді

Тимчасові аномалії відображають незначну поведінку в часовому ряді, яка може бути помітні лише на одному або декількох тимчасових кроках. До них відносяться локальні та глобальні аномалії. Перші виділяються у контекстні

найближчих сусідніх значень, другі – не відповідають загальному шаблону поведінки усього тимчасового ряду, наприклад, несподіване велике відхилення на часовому ряді, що можна побачити на рисунку 2.4, є глобальною аномалією. Прикладом такої аномалії може бути несподіваний платіж клієнта у звичайний день, який не відповідає його типовій поведінці.

Важливо зазначити, що такі характеристики часового ряду як тренд та сезонність також можуть піддаватись аномальній поведінці. Як показано на першому червоному діленні на рисунку 2.4, сезонна аномалія змінює частоту зростання та падіння даних у певному сегменті. Незважаючи на нормальну форму і тенденції часових рядів, їхня сезонність є незвичайною порівняно із загальною сезонністю. Прикладом може слугувати кількість відвідувачів у ресторані протягом тижня. Такий ряд має чітку тижневу сезонність, тому має сенс шукати відхилення в цій сезонності та обробляти аномальні періоди окремо. Різка зміна тренду, який спричиняє постійний зсув даних до їхнього середнього значення і спричиняє перехід у тенденції часового ряду, на другому червоному діленні також може бути такою. Хоча ця аномалія зберігає циклічність і сезонність, вона різко змінює нахил тренду, іноді тренди можуть змінювати напрямок, тобто переходити від зростання до спаду і навпаки. Коли виходить нова пісня та стає популярною на деякий час, а потім зникає з чартів, де тренд змінюється і вважається аномалією тренду, але цілком ймовірно, що в майбутньому тренд відновиться.

Шейплет – це особливий шаблон або підпослідовність, яка відрізняється від звичайної структури, що спостерігається в решті послідовності, як на останньому червоному діленні на рисунку 2.4. Вони часто використовуються для виявлення аномалій або змін у даних, оскільки поява послідовності, яка значною мірою не відповідає загальному шаблону часового ряду, може свідчити про аномальну подію. Зазвичай серцевий ритм має регулярний шаблон і якщо в певний момент часу з'являється короткочасний сплеск або зниження, що сильно відрізняється від основного шаблону, це може бути шейплетом, який вказує на можливе порушення.

Міжметричні аномалії характерні для багатовимірних часових рядів, які містять багато пов’язаних змінних, наприклад вологість та температуру, та виникають, коли взаємозв’язок між цими метриками порушується. Коли незвичайна поведінка зустрічається як у часі, так і зі зміною залежних метрик, ймовірніше за все, це тимчасово-міжметрична аномалія. Наприклад, у фінансових даних це може означати одночасну зміну вартості акцій та обсягів торгів, які не відповідають історичним тенденціям. Такі аномалії вносять додаткову складність і проблеми у виявлення аномалій, однак, іноді полегшують виявлення в різних часових вимірах через одночасне порушення декількох залежностей, як це показано рисунок 2.5.

Для виявлення аномалій використовують декілька підходів: розпізнавання аномалій з вчителем та без. Підхід виявлення аномалій із вчителем вимагає наявності повноцінної навчальної вибірки, що включає достатньо екземплярів нормального й аномального класів значень. Спершу відбувається навчання на даних, на яких вручну позначені нормальні та аномальні точки, потім відбувається розпізнавання, коли на основі побудованої моделі класифікуються нові дані. Зазвичай передбачається, що статистичні властивості моделі не змінюються з плином часу, і така зміна часто вимагає повторного навчання. Основна складність таких підходів – формування даних для навчання. Часто аномальний клас до того ж гірше представлений, ніж нормальний, що може призводити до неточностей в отриманій моделі. У деяких випадках дані для навчання можуть представляти лише нормальний клас, тоді варто говорити про часткове виявлення аномалій з вчителем. Навчена на нормальному класі модель може визначити приналежність даних до нього методом виключення, визначаючи аномальні дані. Розпізнавання в режимі без учителя виходить із припущення, що аномальні дані зустрічаються досить рідко. Тому аномальними позначаються тільки найбільш віддалені від середніх значень. Застосування цієї методики на потокових даних ускладнене, оскільки для

гарної оцінки середнього та очікуваних відхилень необхідно мати уявлення про весь масив даних [17][25].

Серед методів виявлення аномалій можна виокремити статистичні та методи машинного навчання. У свою чергу, методи машинного навчання підрозділяють на методи з учителем (дерево рішень, SVM, LSTM) і методи без учителя (K-середніх, ієрархічна кластеризація, DBSCAN) [25]. Методи з учителем потребують навчальної вибірки для застосування моделі, що відрізняє норму від аномалії. Методи машинного навчання можуть знаходити складніші аномалії, ніж статистичні методи. Так, за допомогою нейронних мереж можливе виявлення і класифікація мережевих атак.

До статистичних методів виявлення аномалій можна віднести інтегровану ковзну середню ARIMA, яка застосовувалась при попередніх дослідженнях. Статистичні методи ґрунтуються на ідеї, що нормальні часові ряди генеруються зі статистичного процесу, а значення, які не відповідають цьому процесу, вважаються аномальними. Ключовою складовою цих методів є вивчення параметрів статистичного процесу на навчальному часовому ряді й оцінювання відповідності тестового часового ряду отриманим параметрам.

З методів машинного навчання, які використовуються для виявлення аномалій, можна виділити наступні:

- Нейронні мережі. Вони передбачають поведінку різних користувачів і метрик у системах. За умови правильного проектування та впровадження, нейронні мережі здатні вирішити багато проблем, з якими стикаються підходи, що базуються на правилах. Основною перевагою ШНМ є їхня толерантність до неточних даних і невизначеної інформації, а також здатність знаходити рішення на основі даних, не маючи попередніх знань про закономірності в них. Це, у поєднанні з їхньою здатністю узагальнювати навчальні дані, зробило їх чи не найліпшим підходом до ідентифікації аномальної активності. Для того, щоб застосувати цей підхід до ідентифікації, дані, що представляють звичайну активність та аномальну, повинні

бути введені в ШНМ для автоматичного налаштування коефіцієнтів на етапі навчання [25].

- SVM (метод опорних векторів) – метод класифікації, що переводить вихідні вектори, які представляють значення, у простір вищої розмірності та знаходить роздільну гіперплощину з максимальним проміжком у цьому просторі, що відокремлює нормальні значення від аномальних. SVM спочатку відображає вхідний вектор у вимірний простір ознак, а потім отримує оптимальну розділювальну гіперплощину у вимірному просторі. Межа рішення, тобто розділова гіперплощина, визначається опорними векторами, а не всією навчальною вибіркою, і, таким чином, є надзвичайно стійкою до викидів. SVM-класифікатор зокрема призначений для бінарної класифікації. Тобто для розділення набору навчальних векторів, які належать до двох різних класів, при цьому опорними векторами є навчальні вибірки, близькі до межі прийняття рішення. SVM також надає користувачеві параметр, який називається штрафним коефіцієнтом. Він дозволяє користувачеві зробити компроміс між кількістю зразків неправильної класифікації та шириною межі прийняття рішення [25].

- Метод K-середніх – алгоритм кластеризації, який належить до методу машинного навчання без вчителя, ідеєю якого є об'єднання даних у кластери навколо головних точок, що мають кількість точок, які є головною. Він розділяє дані на k кластерів і гарантує, що дані в межах одного кластера схожі, тоді як дані в різних кластерах мають низьку схожість. Спочатку алгоритм випадковим чином вибирає K даних як початковий центр кластера, для решти даних додає їх до кластера з найбільшою схожістю відповідно до відстані до його центру, а потім перераховує центр кластера для кожного з кластерів. Цей процес повторюється до тих пір, поки центр кожного кластера не зміниться. Таким чином дані розбиваються на K кластерів. На жаль, кластеризація за методом K-середніх чутлива до викидів, і набір об'єктів, розташованих ближче до центроїда, може бути порожнім, і в цьому випадку центроїди не можуть бути оновлені [25].

– Дерево рішень – потужні та дуже поширені інструменти для класифікації та прогнозування. Дерево рішень – це дерево, яке має три основні компоненти: вузли, дуги та листя. Кожен вузол позначається атрибутом ознаки, який є найбільш інформативним серед атрибутів, які ще не розглядалися на шляху від кореня. Кожна дуга, що виходить з вузла, позначається значенням характеристики вузла, а кожне листя – категорією або класом. Дерево рішень можна використовувати для класифікації точки даних, починаючи з кореня дерева і рухаючись по ньому, поки не буде досягнуто листкового вузла, який забезпечує класифікацію точки даних [25].

2.2. Використання ШНМ для підвищення ефективності прогнозування та виявлення аномалій

2.2.1. Модель штучного нейрона

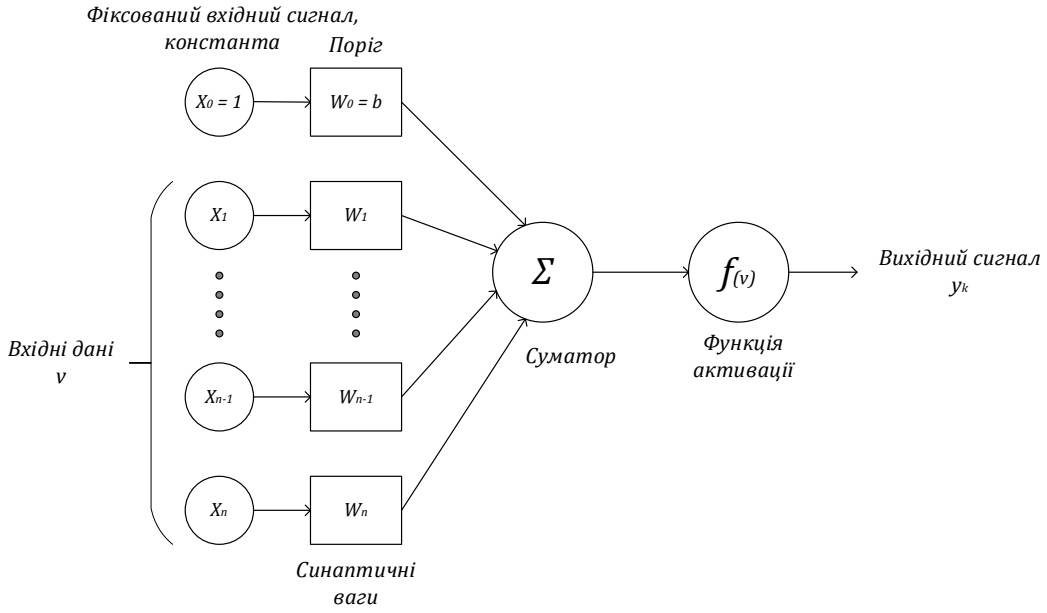


Рисунок 2.6 – Структура штучного нейрона

Нейронні мережі, також відомі як штучні нейронні мережі (ANN) або імітаційні нейронні мережі (SNN), є галуззю машинного навчання в штучному інтелекті та основою алгоритмів глибокого навчання. Їх назва та структура натхненні людським мозком, імітуючи спосіб, у який біологічні нейрони передають сигнали один одному [25]. Штучні нейронні мережі – це обчислювальні структури, які використовують у своїй архітектурі спеціалізовані обчислювальні елементи, що називаються штучними нейронами, які з'єднані між собою і працюють паралельно, передаючи сигнали один одному для виконання певних обчислювальних завдань.

Модель штучного нейрона (рисунок 2.6), складається з трьох компонентів – вагових коефіцієнтів, суматора та функції активації – які визначають, як нейрон обробляє інформацію, маніпулюючи та перетворюючи вхідні дані на результати [25].

Вагові коефіцієнти x_j – це числа, що використовуються для масштабування вхідних значень. Вони відіграють ключову роль в обчисленнях, які робить нейрон. Як правило, ці коефіцієнти ініціалізуються випадковим чином та налаштовуються під час навчання за допомогою процесу зворотного поширенням помилки.

Суматор Σ – це компонент, що обчислює суму вхідних значень. Кожен штучний нейрон містить набір ваг w_j, які відповідають вхідним з'єднанням x_j, що являють собою числові коефіцієнти, що визначають силу впливу кожного вхідного сигналу на нейрон. Кожне вхідне значення x_j множиться на відповідну вагу w_j і потім сумується. Ця сума далі проходить через функцію активації $f(v)$.

Функція активації $f(v)$ приймає взважену суму від суматора та перетворює її на скалярне вихідне значення нейрона. Зазвичай нормалізований діапазон амплітуди нейрона лежить у інтервалах [0, 1] або [-1, 1].

Біологічні нейрони мають певний поріг активації, після якого вони вважаються активованими і не можуть бути активованими постійно. Для цього використовується функція активації. Функція активації контролює значення v і вирішує, чи визнає зовнішній зв'язок цей нейрон активним, чи його можна

ігнорувати. Існує декілька основних функцій активації: порогова, сигмоїдальна логістична та гіперболічний тангенс.

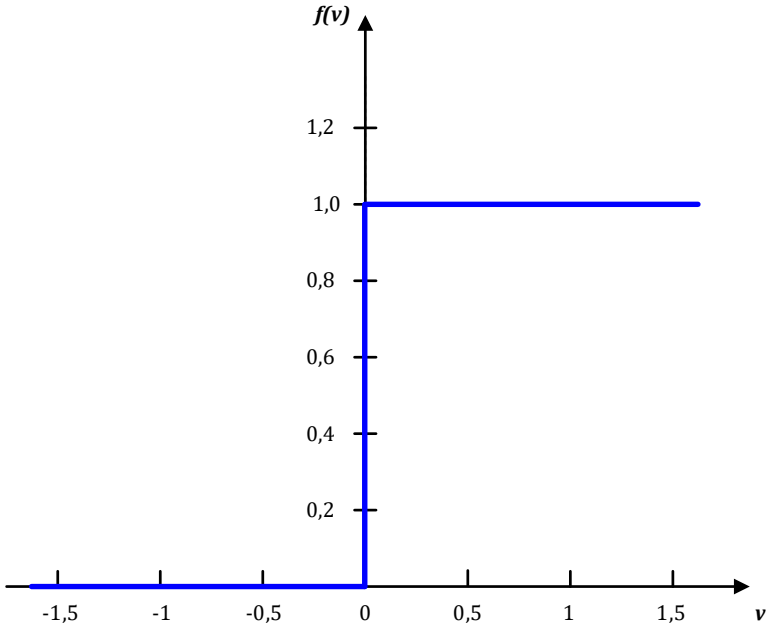


Рисунок 2.7 - Порогова функція активації

Порогова функція активації, або функція Хевісайда, показана на рисунку 2.7, приймає значення в діапазоні [0, 1] (формула 2.1). Якщо значення *v* більше деякого порогового значення, нейрон вважається активованим, у іншому разі – неактивний.

$$f(v) = \begin{cases} 1, \text{ якщо } v \geq 0; \\ 0, \text{ якщо } v < 0. \end{cases} \quad (2.1)$$

Функція Хевісайда має серйозний недолік, який проявляється, коли мережа працює з великою кількістю нейронів. Порогова функція є стрибкоподібною, де значення стану 1 означає що нейрон активований, а 0 деактивований. Це ускладнює оптимізацію та навчання мережі і перешкоджає плавним переходам між ними. Вона також не надає додаткової інформації про ступінь активації нейронів, що унеможливлює розв'язання складних задач або обробку великих обсягів даних. Однак головною проблемою цієї функції є відсутність похідних, що ускладнює

навчання нейронних мереж за допомогою градієнтного спуску або інших методів оптимізації. Тому щоб покращити результати навчання, замість неї застосовуються інші функції активації, такі як сигмоїдальна, гіперболічний тангенс та інші функції, які мають гладкість, нелінійність та можливість обчислення похідних, що полегшує процес навчання та оптимізацію мережі [25].

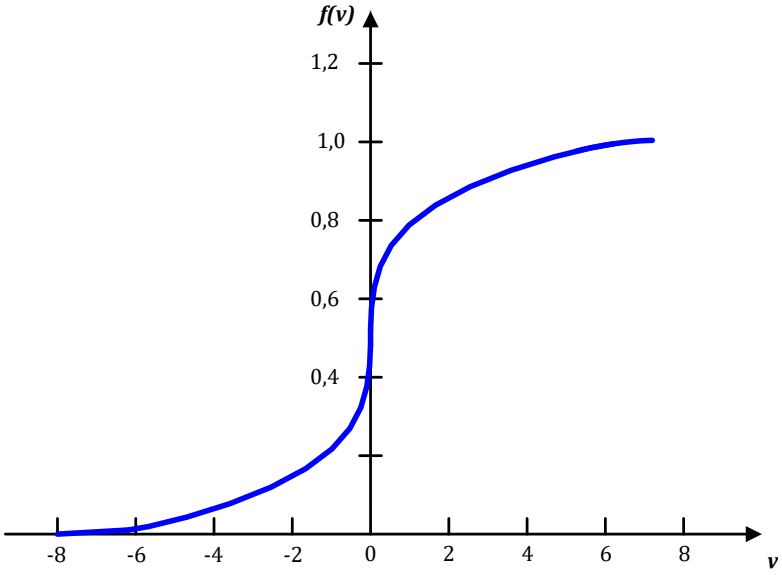


Рисунок 2.8 – Сигмоїдальна функція активації

Сигмоїдальна функція (рисунок 2.8) є найпоширенішою функцією, що використовується для створення штучних нейронних мереж. Вона є нелінійною та диференційованою і може набувати будь-яких значень у діапазоні [0, 1]. Прикладом сигмоїдальної функції може слугувати логістична функція (формула 2.2), що задається наступним відновленням:

$$f(v) = \frac{1}{1 + e^{-\alpha v}}, \tag{2.2}$$

де: α – параметр нахилення сигмоїдальної функції, змінюючи який можна будувати функції з різним рівнем крутизни.

Основна перевага сигмоїдальної функції активації полягає в тому, що весь її діапазон є гладким і диференційованим. Це означає, що їх можна використовувати для операцій, які вимагають обчислення похідних, таких як алгоритми оптимізації градієнтного спуску. Однак сигмоїдальні функції активації мають і недоліки. Вони можуть призвести до так званої проблеми «вмираючого нейрона». Нейрони майже завжди перебувають у стані 0 або 1, або взагалі не змінюються. Це відбувається тому, що сигмоїдальна функція насичується на краях, де її градієнти стають занадто малим [25].

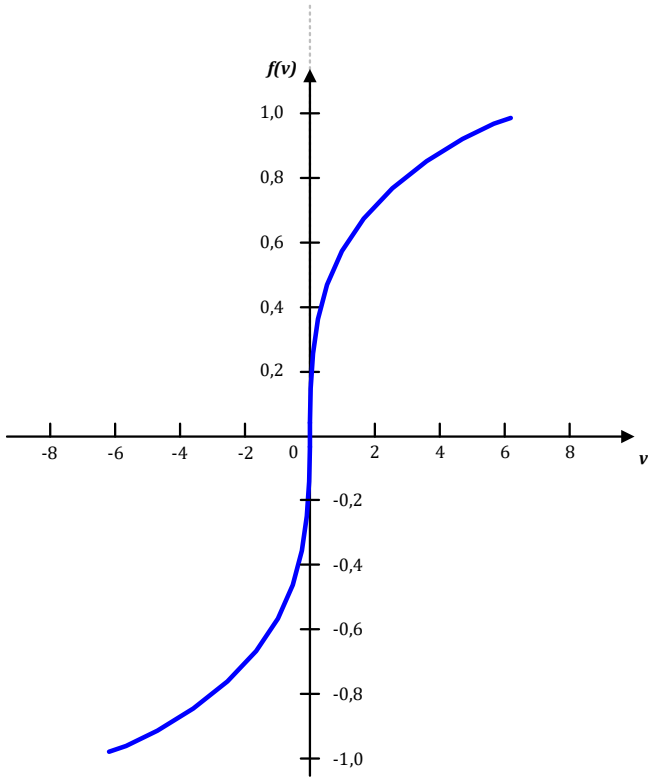


Рисунок 2.9 – Функція активації гіперболічного тангенсу

Гіперболічний тангенс (рисунок 2.9) є подібним до сигмоїдальної функції та має подібні характеристики до сигмоїди, однак відрізняється тим, що вихідне значення варіюється у діапазоні значень [-1, 1] та визначається наступною логістичною функцією (формула 2.3):

$$f(v) = th(v) = \frac{e^{\alpha v} - e^{-\alpha v}}{e^{\alpha v} + e^{-\alpha v}}\ , \qquad (2.3)$$

де: α – параметр нахилення функції гіперболічного тангенсу, змінюючи який можна будувати функції з різним рівнем крутизни.

Основними перевагами цієї функції активації є те, що вона має високу центрованість навколо нуля, що призводить до більш ефективного навчання нейронної мережі і може давати від'ємні значення. Як і сигмоїдальна, ця функція є гладкою та диференційованою по всьому діапазону, однак вона має ті ж недоліки. Ця функція також може спричинити проблему «вмираючого нейрона», коли нейрони досягають насичення і майже завжди видають -1 або 1, що може сповільнити або навіть зупинити процес навчання [25].

Крім того, модель нейрона включає додатковий пороговий елемент b, що дорівнює синаптичній вазі w_0, який контролює активність нейрона і визначає, чи збільшується або зменшується вхідний сигнал на функцію активації. Замість того, щоб розглядатися окремо, пороги включені безпосередньо в модель нейрона і представлені як додаткові величини з фіксованими вагами [25]. Постійний вхідний сигнал $x_0 = 1$ є способом представлення цього порогу в математичній моделі нейрона і спрощує математичну модель нейрона, оскільки постійне значення може бути безпосередньо включене в загальний ефект нейронів і ваг.

Вихідний сигнал нейрона y_k (формула 2.4) може бути описана наступним співвідношенням:

$$y_k = f(\Sigma(x_j \times w_j) + b), \qquad (2.4)$$

де: y_k – вихідний сигнал нейрона;

$f(v)$ – функція активації;

x_j – значення вхідних аргументів;

w_j – синаптичні ваги нейрона k;

b – значення порогу.

Ця формула описує сумування вхідних значень, помножених на відповідні ваги w_j та поріг b. Ваги та поріг є параметрами, які навчаються під час процесу тренування нейронної мережі. Однак як зазначалося раніше, оскільки в деяких нейронних моделях вхідний сигнал може бути представлений як сума ваг, помножених на відповідні входи, де один з них приймає значення, що дорівнює 1, а його вага еквівалентна порогу, вихідний сигнал нейрона y_k може бути представлено наступним співвідношенням (формула 2.5):

$$y_k = f(\Sigma(x_j \times w_j)), \tag{2.5}$$

де: y_k – вихідний сигнал нейрона;

$f(v)$ – функція активації;

x_j – значення вхідних аргументів, де $x_0 = 1$;

w_j – синаптичні ваги нейрона k, де w_0 приймає значення порогу b.

Це дозволило трансформувати модель нейрона до вигляду, який показано на рисунку 2.6, та полегшити роботу з нейронними мережами, надавши низку переваг [25]:

– Обчислювальна ефективність. Цей підхід дозволяє об'єднати обчислення в одну операцію. Замість окремого віднімання порогу, можна просто додати ще одну вагу, яка множиться на вхід з фіксованою величиною, що дозволяє збільшити швидкість обчислень.

– Універсальність. Цей підхід дозволяє використовувати однаковий механізм для оновлення всіх ваг, включаючи поріг. Це полегшує написання коду та розробку алгоритмів оптимізації.

− Гнучкість. Поріг дозволяє моделі робити більш гнучкі прогнози. Якщо у всіх входів нейрона є нульові значення, модель все ще може активуватися, якщо порогове значення відповідно високе. Це може бути корисним для вирішення певних задач.

− Можливість зсуву функції активації. Поріг діє як зсув для функції активації, дозволяючи нейрону активуватися при менш високих значеннях вхідного сигналу. Це дозволяє нейронній мережі краще пристосовуватися до різних видів даних і завдань.

2.2.2. Архітектура штучних нейронних мереж

Архітектура нейронної мережі визначає її структуру. Кожен шар містить кілька шарів і певну кількість нейронів, а також те, як вони з'єднані між собою за допомогою зв'язків, що визначаються ваговими коефіцієнтами. Вхідні нейрони отримують вхідний вектор аналізованих даних або сигналів і передають його на наступний шар нейронної мережі, фактично не виконуючи жодних обчислень. Наступний шар нейронної мережі, або прихований шар мережі, який містить проміжні нейрони, виконує більшу частину обчислень і перетворює переданий сигнал у форму, яка може бути сприйнята та інтерпретована вихідними нейронами. Після того, як результати передаються на наступний шар мережі, що складається з вихідних нейронів, вони перераховуються відповідно до значень, отриманих від нейронів попереднього шару. Для обчислення використовується вихідна функція нейрона (формула 2.4).

Будь-яке перетворення даних, що виконується нейронною мережею, визначається не тільки властивостями окремих нейронів, але й властивостями її архітектури. Структура нейронної мережі визначає її здатність вирішувати різні завдання різного рівня складності, а різні архітектури мереж використовують різні структури і конфігурації нейронів для вирішення поставленого завдання. З точки

зору топології розрізняють три основні класи нейронних мереж: одношарові, багатошарові та рекурентні нейронні мережі.

Одношарова нейронна мережа (рисунок 2.10) або перцептрон являє собою найпростішу форму нейронної мережі прямого розповсюдження, у якій вихідний шар безпосередньо проектується на вихідний шар нейронів, що виступають у цій мережі обчислювальними вузлами [25].

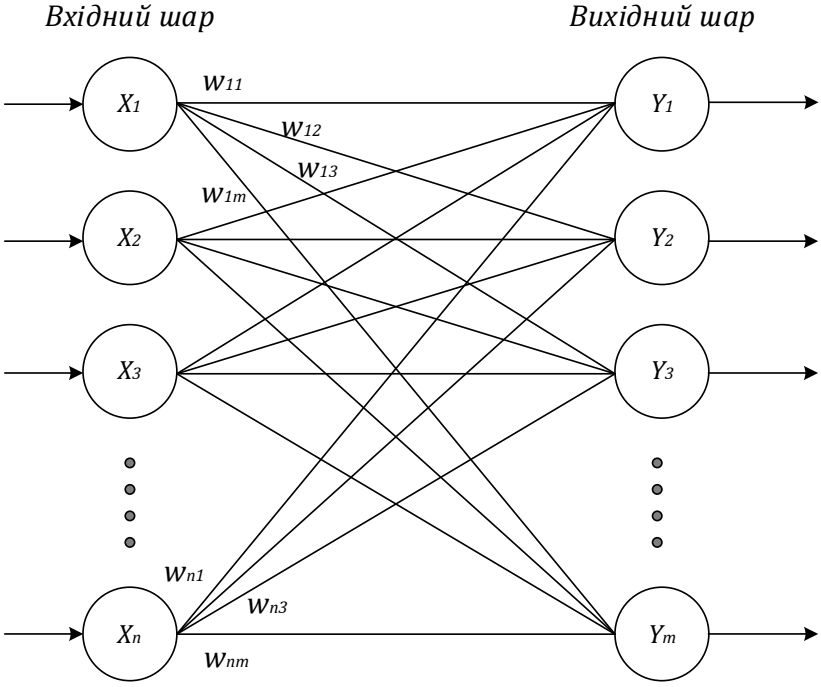


Рисунок 2.10 – Одношарова нейронна мережа

Під час обробки інформації у такій одношаровій мережі властиві наступні процеси:

– кожен нейрон X_n, де n – номер вхідного нейрона, отримує вхідний сигнал, це можуть бути вхідні дані, які потрібно обробити;

– кожен вхідний сигнал потім множиться на відповідний ваговий коефіцієнт w_{nm}, де n – номер вхідного нейрона, m – номер вихідного нейрона, ваги визначають важливість кожного вхідного сигналу для обчислення вихідного сигналу;

– обчислені добутки потім сумуються для кожного нейрона у вихідному шарі Y_m, де m – номер вихідного нейрона;

– отримана сума передається через функцію активації, яка визначає вихідний сигнал кожного нейрона Y_m.

– вихідний сигнал потім може використовуватися як вхід для іншого шару в більш складній мережі, або він може бути кінцевим вихідним сигналом мережі.

Багатошарові нейронні мережі або багатошарові перцептрони відрізняються від мереж прямого поширення наявністю додаткового прихованого шару, який називається прихованими нейронами [25]. Ця частина мережі невидима для входів і виходів нейронної мережі, але функція прихованих нейронів полягає в корисному втручанні між цими шарами: додавши один або кілька прихованих шарів, мережа може отримати статистику вищого порядку з вхідних даних.

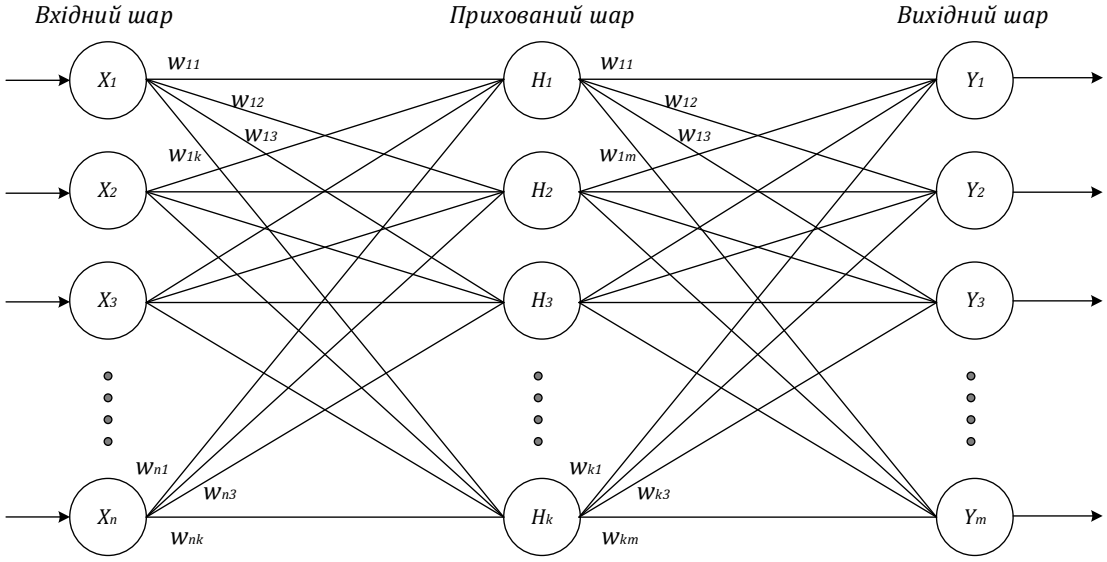


Рисунок 2.11 – Багатошарова нейронна мережа

На рисунку 2.11 показано багатошарову нейронну мережу. Можна побачити, що всі вузли кожного окремого шару пов’язані з усіма вузлами наступного, що робить таку мережу повнозв’язною. Вихідні вузли вхідного шару мережі подають і

застосовують відповідні елементи вхідного вектора до нейронів наступного прихованого шару. Як правило, нейрони в кожному шарі мережі мають вихідні сигнали тільки з попереднього рівня. Сукупність вихідних сигналів нейронів останнього рівня мережі представляє загальну реакцію мережі на вхідний шаблон активації, згенерований вузлами-джерелами у вхідному шарі.

Алгоритм обробки сигналів подібний до того, що описаний для одношарової нейронної мережі з тією лише відмінністю, що процес повторюється більшу кількість разів у залежності від кількості прихованих шарів H_k мережі.

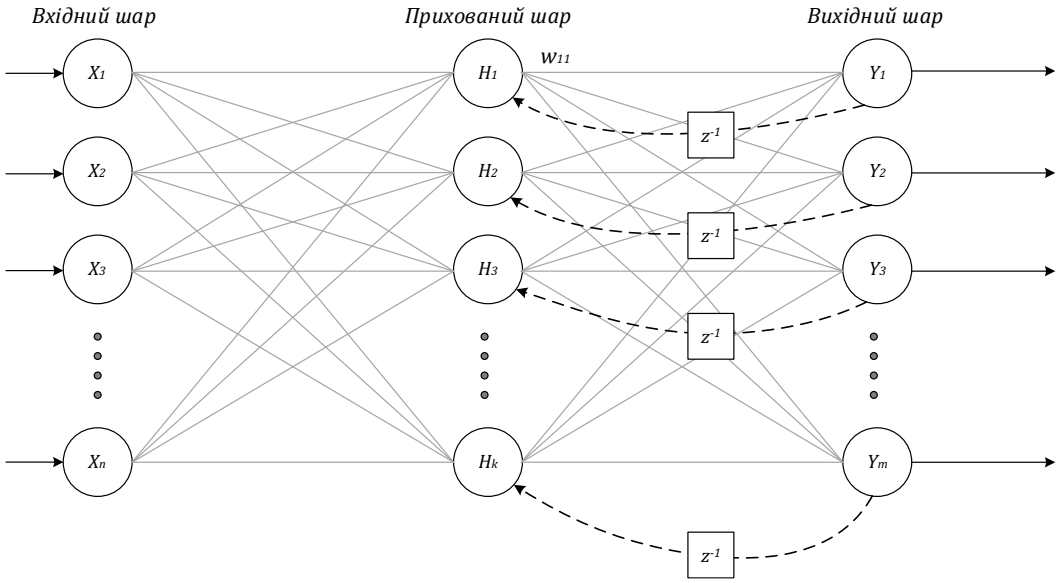


Рисунок 2.12 – Рекурентна нейронна мережа

Рекурентні нейронні мережі (RNN) відрізняються від попередніх двох тим, що мають принаймні один ланцюг зворотного зв'язку. Рекурентні нейронні мережі складаються лише з одного шару нейронів, але кожен нейрон може спрямовувати свій вихідний сигнал на входи всіх інших. Наявність таких зв'язків дає можливість навчатися, використовуючи інформацію з попередніх шарів, що важливо для врахування контексту та випадковості [25]. При застосування зворотнього зв’язку у рекурентній мережі (рисунок 2.12) використовується оператор одиничної

затримки z^{-1} для моделювання зв'язку між послідовними часовими кроками, що дозволяє їй враховувати не тільки поточні вхідні дані, але й інформацію, отриману на попередніх часових кроках та цим забезпечує «пам'ять» для мережі.

2.2.3. Проблематика навчання штучних нейронних мереж

Як зазначалось раніше, можна виділити два основних алгоритми навчання, що використовуються в тому числі для навчання штучних нейронних мереж: навчання з учителем та навчання без учителя.

Навчання без вчителя використовує алгоритми машинного навчання для аналізу та кластеризації немаркованих наборів необроблених даних. Групуючи схожі вхідні вектори, мережа намагається знайти структури та взаємозв'язки у наборі вхідних даних без попередніх вказівок [25]. Параметри можуть бути встановлені таким чином, щоб мінімізувати відстань між векторами в одному кластері та максимізувати відстань між різними кластерами. Тоді вхідні вектори, які потрапляють в один кластер, вважаються схожими за певними критеріями.

Навчання з учителем використовується у тих випадках, коли є чітко визначені вхідні та вихідні дані. Використовуючи позначені входи та виходи, модель може вимірювати свою точність та навчитися розпізнавати об’єкти та ознаки, які їх класифікують. Вхідні вектори даних представляють собою навчальні пари, що складаються з прикладу вхідних даних та відповідного цільового вихідного значення. Під час навчання мережа передбачає вихід на основі вхідних даних і потім порівнює свій прогноз з реальним цільовим значенням. Різниця між прогнозом мережі та реальним цільовим значенням визначає помилку, яка потім використовується для коригування ваг мережі з метою її зменшення [25]. Це процес, зазвичай, відбувається крок за кроком для кожного навчального прикладу в наборі даних, і повторюється декілька разів, поки загальна помилка на наборі даних не знизиться до прийнятного рівня. Цей алгоритм навчання нейронних мереж є

алгоритмом зворотного поширення помилки, який використовується в багатошарових мережах [25][26].

На основі вбудованих знань вчитель може генерувати бажаний зворотний зв'язок, що відповідає вектору вхідних даних, і надсилати його мережі для отримання бажаного результату. Вихід мережі порівнюється з бажаним результатом і мережею буде отримано сигнал помилки (формула 2.12) – різницю між бажаним результатом та поточним зворотним відгуком нейронної мережі [26].

$$e_k(n) = d_k(n) - y_k(n), \tag{2.12}$$

де: $e_k(n)$ – сигнал помилки нейрона k з номером кроку ітераційного процесу або дискретним часом n;

$d_k(n)$ – бажаний результат, який потрібно отримати;

$y_k(n)$ – вихідний сигнал або зворотній відгук нейронної мережі.

Сигнал помилки запускає спеціальні механізми управління, які коригують синаптичні ваги нейронів. Ці налаштування спрямовані на поступове наближення вихідного сигналу $yk(n)$ до бажаного значення сигналу $dk(n)$ і досягаються шляхом мінімізації функції втрат (рівняння 2.13), яка є мірою того є мірою того, наскільки прогнози нейронної мережі недооцінюють бажаний результат, та показників продуктивності [26].

$$E(n) = \frac{e_k^2(n)}{2}, \tag{2.13}$$

де: $E(n)$ – поточне значення функції вартості. Процес буде продовжуватись до моменту, поки синаптичні ваги w_k нейрона k не будуть стабілізовані.

Мінімізація витрат $E(n)$ виконується за дельта-правилом Відроу-Хоффа [26], яке каже, що коригування, що застосовується до синаптичної ваги нейрона, є прямопропорційним добутку сигналу помилки та вхідний сигнал, що її викликав

(формула 2.14). Після визначення змінної $\Delta w_{kj}(n)$ визнається нове значення для наступного кроку дискретизації (формула 2.15) .

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n), \tag{2.14}$$

$$w_{kj}(n + 1) = w_{kj}(n) + \Delta w_{kj}(n), \tag{2.15}$$

де: $w_{kj}(n)$ – поточне значення синаптичних ваг нейрона k;

η – додатна константа, що визначає швидкість навчання та використовується при переході від одного кроку процесу до іншого;

$e_k(n)$ – сигнал помилки нейрХона k;

$x_j(n)$ – відповідний елемент вектору $x(n)$;

n – крок дискретизації.

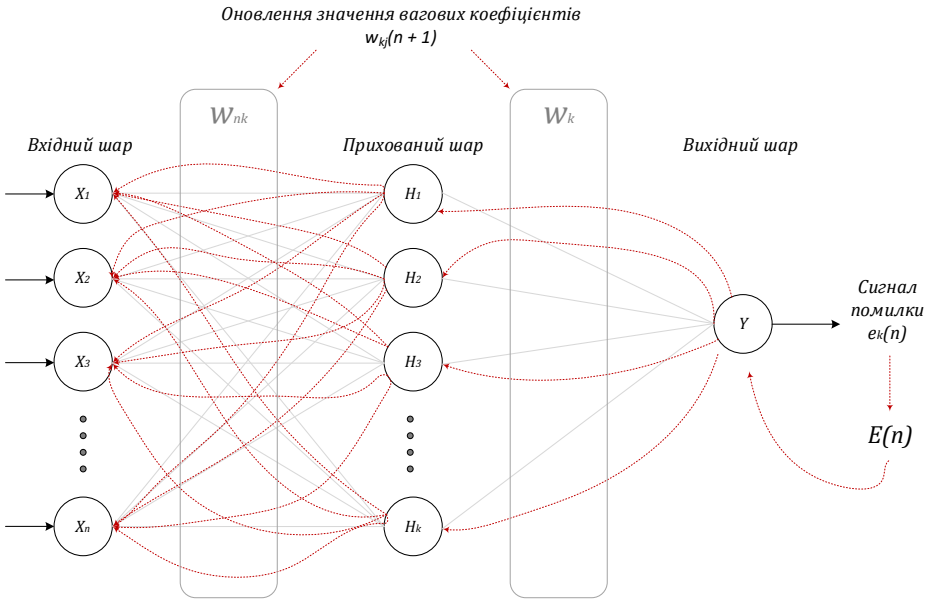


Рисунок 2.13 – Навчання нейронних мереж з алгоритмом поширення помилки

Алгоритм навчання нейронної мережі методом зворотного поширення (рисунок 2.13) перетворює навчальні дані в пакети малого розміру, де кожен

навчальний зразок надсилається на вхідний шар нейронної мережі, та визначає допущені мережею помилки, порівнюючи виходи та мітки за допомогою функції витрат [26]. Після чого обчислюється, скільки нейронів у вихідному шарі зробили внесок у помилку, використовуючи правило ланцюгових обчислень (формула 2.16).

$$\frac{dE(n)}{dw_{kj}(n)} = \frac{dE(n)}{de_k(n)} \frac{de_k(n)}{dy_k(n)} \frac{dy_k(n)}{dv_k(n)} \frac{dv_k(n)}{dw_{kj}(n)}, \qquad (2.16)$$

де: $dE(n)/dw_{kj}(n)$ – фактор чутливості, що визначає наскільки сильно потрібно змінити зміщення для мінімізації помилки.

Після цього алгоритмом обчислюється, наскільки попередній прихований шар впливає на помилку, зроблену вихідним шаром, визначаючи градієнт помилки в кожному нейроні, за допомогою яких ваги та зміщення оновлюються відповідно за допомогою алгоритму градієнтного спуску [26] (формула 2.17).

$$\Delta w_{kj}(n) = \eta \delta_k(n) y_j(n), \qquad (2.17)$$

де: $\delta_k(n)$ – локальний градієнт, що представлено наступним виразом (формула 2.18):

$$\delta_k(n) = - \frac{dE(n)}{dv_j(n)} = \frac{dE(n)}{de_k(n)} \frac{de_k(n)}{dy_k(n)} \frac{dy_k(n)}{dv_k(n)}. \qquad (2.18)$$

Локальний градієнт $\delta_k(n)$ при диференціації даного виразу, визначається за допомогою наступного рівняння (формула 2.19):

$$\delta_k(n) = e_k(n) \varphi`(v_k(n)), \qquad (2.19)$$

де: $e_k(n)$ – сигнал про помилку;

$\varphi`(v_k(n))$ – похідна, що відповідає функції активації.

Метод зворотного розповсюдження помилки є досить ефективним при навчання штучних нейронних мереж, але він має свої недоліки. Іноді може виникати проблема зникаючих градієнтів, коли у процесі передачі через мережу вони стають дуже малими [26]. Це може призвести до того, що ваги в нижніх шарах мережі майже не оновлюються, що призводить до повільного навчання або навчання зовсім не відбувається. При зникненні співвідношення між поточним і колишнім прихованим станом дуже мале, і значення прихованого наближається до 0 протягом багатьох обчислень похідних, фактично обнуляючи прихований стан і змушуючи мережу «забути» довгострокові залежності, які вона моделювала. Протилежною проблемою зникаючих градієнтів є проблема вибухаючих градієнтів, коли співвідношення між поточним і минулим прихованими станами дуже велике, що призводить до нестабільного навчання і великих коливань в вагах. За велику кількість часових кроків і обчислень похідних значення в прихованому стані стають дуже великими, що призводить до неточних результатів роботи мережі. Через зникаючі градієнти не рекомендується використовувати звичайні ШНМ для моделювання довгострокових залежностей у послідовностях. У такому випадку використовується довга короткочасна пам'ять або LSTM [27][29].

2.2.4. Нейронні мережі з довгою короткостроковою пам’яттю

Для виявлення аномалій у часових рядах можливе використання методів прогнозування. Прогнозування за допомогою нейронних мереж передбачає роботу з вхідними даними, що представляють послідовність, тому важливо, щоб нейронна мережа, обробляючи черговий показник у певний момент часу, вміла ґрунтуватися на значеннях у попередні моменти часу, з чим можуть впоратись рекурентні нейронні мережі, однак вони мають один недолік – при міри отримання нових

значень показників вплив попередніх значень послаблюється. Щоб нівелювати цю проблему, використовують так звані мережі з довгостроковою пам’яттю (LSTM) [29].

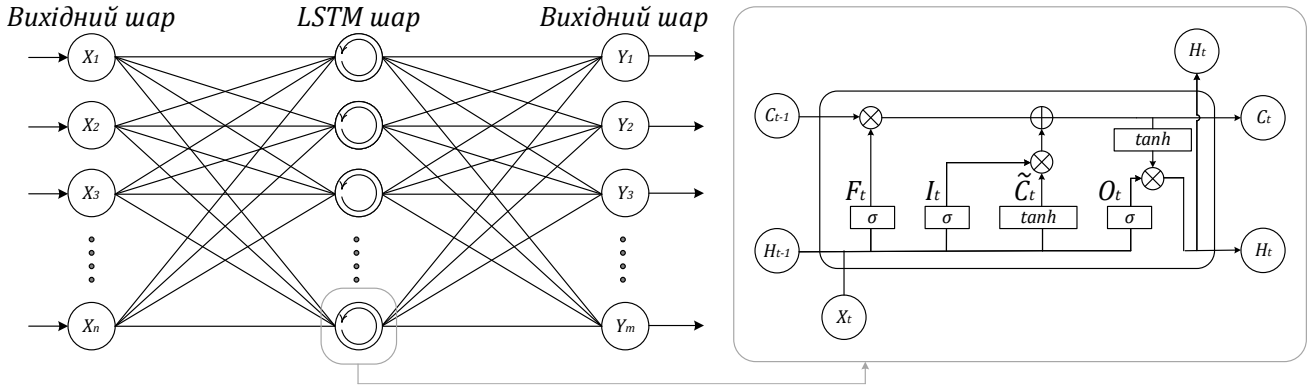


Рисунок 2.14 – Архітектура нейронної мережі LSTM

LSTM є розширенням рекурентних нейронних мереж RNN. Кожен нейрон в LSTM – це комірка з «пам'яттю», яка може зберігати інформацію, підтримуючи свій стан, на відміну від RNN, які просто приймають поточний вхід з попереднього прихованого стану, щоб вивести новий прихований стан. LSTM розроблений для подолання проблеми зникаючого градієнта в традиційних RNN, яка виникає, коли градієнти стають занадто малими під час зворотного розповсюдження і змушують мережу забувати довгострокові залежності [28][29].

LSTM складається з декількох комірок пам'яті, кожна з яких містить три компоненти: вхідні ворота I_t, ворота втрати F_t та вихідні ворота O_t [28]. Архітектуру мережі, а також структуру самих нейронів показано на рисунку 2.14. Архітектура LSTM може бути складена в декілька шарів, що дозволяє мережі вивчати більш складні патерни у вхідній послідовності. Вхідні ворота I_t контролюють кількість нової інформації, яка потрапляє в комірку, в той час як ворота втрати F_t контролюють кількість старої інформації, яка зберігається в комірці, вихідні ворота O_t контролюють кількість інформації, яка виводиться з комірки.

На першому етапі LSTM необхідно визначити, яку інформацію вносити у стан комірки. Протягом кожного часового кроку LSTM отримує вхідний вектор X_t і прихований вектор H_{t-1} стану з попереднього часового кроку. Вхідний вектор X_t спочатку множиться на вагову матрицю W_F і додається до попереднього прихованого вектора стану H_{t-1}, враховуючи значення зміщення b_F. Остання використовується для зважування суми вхідних значень перед застосуванням функцій активації, дозволяючи моделі краще підлаштовуватись під дані та коригувати вихідні значення [28]. Отриманий вектор застосовується на сигмоїдальну функцію активації σ шару воріт утрати F_t (формула 2.10), що видає значення у діапазоні [0; 1] для кожного стану комірки C_{t-1}, де 0 означає «забути», а 1 – «зберегти» [30].

$$F_t = f(W_F \times [H_{t-1}, X_t] + b_F), \tag{2.10}$$

де: F_t – вихідне значення воріт втрати;

$f(v)$ – сигмоїдальна функція активації σ;

W_F – вагова матриця для воріт забування;

H_{t-1} – прихований стан з попереднього часового кроку;

X_t – вхідний вектор з попереднього часового кроку;

b_F – значення зміщення для воріт втрати.

На наступному кроці відбувається визначення того, яка інформація повинна бути збережена у стані комірки пам’яті [28]. Аналогічно до попереднього розрахунку, початку сигмоїдальна функція шару воріт входу I_t визначає, які значення потрібно оновити (формула 2.11), після чого застосовується гіперболічна функція активації *tanh* для створення нових векторів-кандидатів $\tilde{C}_t$, які пропускаються через гіперболічну функцію активації *tanh* для нормалізації значень у проміжку [-1; 1] [30] (формула 2.12).

$$I_t = f(W_I \times [H_{t-1}, X_t] + b_I), \tag{2.11}$$

$$\tilde{C}_t = th(W_C \times [H_{t-1}, X_t] + \tilde{b}_C), \quad (2.12)$$

де: I_t – вихідне значення воріт входу;

$\tilde{C}_t$ – вихідне значення векторів-кандидатів;

th(v) – гіперболічна тангенс-функція активації *tanh*;

W_I, W_C – вагові матриця для воріт входу та кандидатного стану;

b_I, $\tilde{b}_C$ – значення зміщення для вхідних воріт та кандидатного стану.

Для отримання нового стану комірки вихід вхідних воріт I_t множиться на вектор-кандидат $\tilde{C}_t$, що містить нову інформацію, а значення воріт втрати F_t – на значення попереднього стану комірки C_{t-1} що призводить до видалення непотрібної інформації з комірки [28]. Після цього відбувається оновлення стану комірки з C_{t-1} на C_t [30] (формула 2.13).

$$C_t = F_t \times C_{t-1} + I_t \times \tilde{C}_t, \quad (2.13)$$

де: C_t – нове значення комірки;

F_t – вихідне значення воріт втрати, що регулює, яку інформацію з попереднього стану комірки C_{t-1} зберегти;

I_t – вихідне значення воріт входу, що регулює, яку нову інформацію з кандидатного стану $\tilde{C}_t$ потрібно додати;

C_{t-1} – попереднє значення стану комірки;

$\tilde{C}_t$ – вихідне значення векторів-кандидатів.

Потім шар вихідних воріт O_t застосовує сигмоїдальну функцію активації до зваженої суми вхідного вектора X_t і поточного прихованого стану H_{t-1} (формула 2.14), яка вирішує, які частини стану комірки виводити. Після цього значення вихідне значення O_t множиться на гіперболічний тангенс *tanh* кута нахилу вектора поточного стану комірки (формула 2.15), щоб розмістити усі значення у проміжку

[-1, 1]. В результаті отримуємо вихідне значення комірки прихованого стану H_t нейронної мережі LSTM [30].

$$O_t = f(W_O \times [H_{t-1}, X_t] + b_O), \tag{2.14}$$

$$H_t = O_t \times th(C_t), \tag{2.15}$$

де: O_t – вихідне значення воріт виходу;

$th(v)$ – гіперболічна тангенс-функція активації *tanh*;

W_O – вагова матриця для воріт виходу;

b_O – значення зміщення для воріт виходу;

H_t – вихідне значення прихованого стану комірки.

2.3. Постановка завдань для практичної реалізації

У наступному розділі буде запропоновано використання моделей штучних нейронних мереж, у тому числі мереж LSTM, які через свою здатність відмінно оброблювати часові послідовності, фіксуючи довготривалі залежності, що є важливим для аналізу даних та вдосконалення алгоритмів обробки, запропонованих при попередньому аналізі даних статистичними методами ARMA, ARIMA та SARIMA, відмінно підходять для аналізу часових рядів.

Застосування моделей ШНМ для аналізу лог-журналів на основі штучного інтелекту зможе допомогти краще прогнозувати нормальну поведінку та виявляти відхилення, відслідковуючи підозрілу активність у системі, активно виявляючи аномалії та надаючи прогностичні висновки, що допоможе підсилити загальний рівень безпеки інформаційних систем, вказуючи на потенційні загрози.

3 ПРАКТИЧНА ЧАСТИНА

3.1. Програмна реалізація нейронної мережі LSTM

Аналогічно до моделей ARMA, ARIMA та SARIMA, програмна реалізація моделі LSTM працює з тими ж записами журналів логування навчальної інформаційної системи Moodle, записаних у csv-файл, що використовувалися для аналізу та прогнозування при попередньому аналізі даних (таблиця 1.3).

Щоб дані були придатні для навчання, на основі масиву створюється матриця, у якій кожен рядок містить фрагмент часового ряду завдовжки 20, що визначається змінною *time_steps*. Кожен наступний рядок у цій матриці містить зсув на 1 значення відносно попереднього (рисунок 3.1).

```
          0         1         2         3         4         5         6   \
0 -0.980741 -0.989630 -0.991111 -0.841481 -0.468148 -0.727407 -0.791111
1 -0.989630 -0.991111 -0.841481 -0.468148 -0.727407 -0.791111 -0.851852
2 -0.991111 -0.841481 -0.468148 -0.727407 -0.791111 -0.851852 -0.945185
3 -0.841481 -0.468148 -0.727407 -0.791111 -0.851852 -0.945185 -0.875556
4 -0.468148 -0.727407 -0.791111 -0.851852 -0.945185 -0.875556 -0.223704

          7         8         9         10        11        12        13  \
0 -0.851852 -0.945185 -0.875556 -0.223704  0.013333  0.020741 -0.355556
1 -0.945185 -0.875556 -0.223704  0.013333  0.020741 -0.355556 -0.850370
2 -0.875556 -0.223704  0.013333  0.020741 -0.355556 -0.850370 -0.930370
3 -0.223704  0.013333  0.020741 -0.355556 -0.850370 -0.930370 -0.880000
4  0.013333  0.020741 -0.355556 -0.850370 -0.930370 -0.880000 -0.383704

          14        15        16        17        18        19
0 -0.850370 -0.930370 -0.880000 -0.383704 -0.288889 -0.256296
1 -0.930370 -0.880000 -0.383704 -0.288889 -0.256296 -0.595556
2 -0.880000 -0.383704 -0.288889 -0.256296 -0.595556 -0.842963
3 -0.383704 -0.288889 -0.256296 -0.595556 -0.842963 -0.914074
4 -0.288889 -0.256296 -0.595556 -0.842963 -0.914074 -0.952593
```

Рисунок 3.1 – Створення матриці часового ряду

Лістинг коду 3.1 – Нормалізація значень з використанням MinMaxScaler

```
values = events_by_day.values.reshape(-1, 1)
scaler = MinMaxScaler(feature_range=(-1, 1))
scaled_data = scaler.fit_transform(values)
```

Далі проводиться нормалізація значень за допомогою функції MinMaxScaler пакета scikit-learn, щоб вони перебували в проміжку [-1, 1] (лістинг 3.1).

Для реалізації нейронної моделі було використано бібліотеку Keras. Мережа має два шари LSTM, які безпосередньо використовується для навчання та кожен з яких з'єднаний з відповідним шаром Dropout, який випадковим чином деактивує певний відсоток нейронів для уникнення перенавчання моделі, у реалізованій моделі цей параметр *dropout_rate* становить 20%. Останній шар Dense являє собою окремий нейрон, який об'єднує виходи шару LSTM в одне кінцеве значення, яке вважається результатом прогнозу. Схематичну архітектуру мережі (рисунок 3.2) та її програмну реалізацію (лістинг 3.2) представлено нижче.

Як параметри було використано значення *time_steps* = 20, *batch_size* = 10, *epochs* = 400, *n_neurons* = 50. Кожен з параметрів позначає наступні значення:

- *time_steps* – довжина часового відрізка, за яким передбачають наступне значення;
- *batch_size* – розмір пакета вхідної послідовності, після опрацювання якого відбувається оновлення ваг;
- *epochs* – кількість епох нейронної мережі;
- *n_neurons* – кількість елементів у векторі стану, що в розгорнутому вигляді LSTM означало б кількість нейронів.

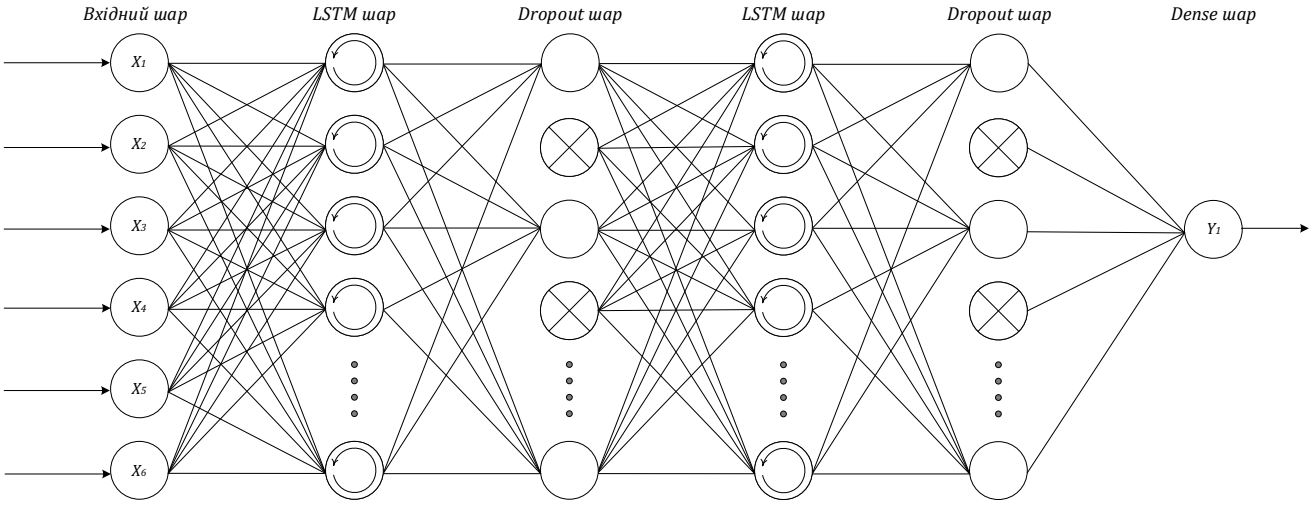


Рисунок 3.2 – Архітектура розробленої мережі LSTM

Лістинг коду 3.2 – Програмна реалізація моделі LSTM

```
def build_lstm_model(time_step, n_neurons, dropout_rate, output_neurons):
    # Побудова LSTM моделі
    model = Sequential()
    model.add(LSTM(n_neurons,return_sequences=True,input_shape=(time_step, 1)))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(n_neurons, return_sequences=False))
    model.add(Dropout(dropout_rate))
    model.add(Dense(output_neurons))
    model.compile(optimizer='adam' , loss='mean_squared_error')
    return model

def train_model(model, X_train, Y_train, X_test, Y_test, epochs, batch_size):
    # Навчання моделі
    model.fit(
        X_train, Y_train,
        epochs=epochs,
        batch_size=batch_size,
        validation_data=(X_test, Y_test),
        verbose=1,
        shuffle=False
    )
```

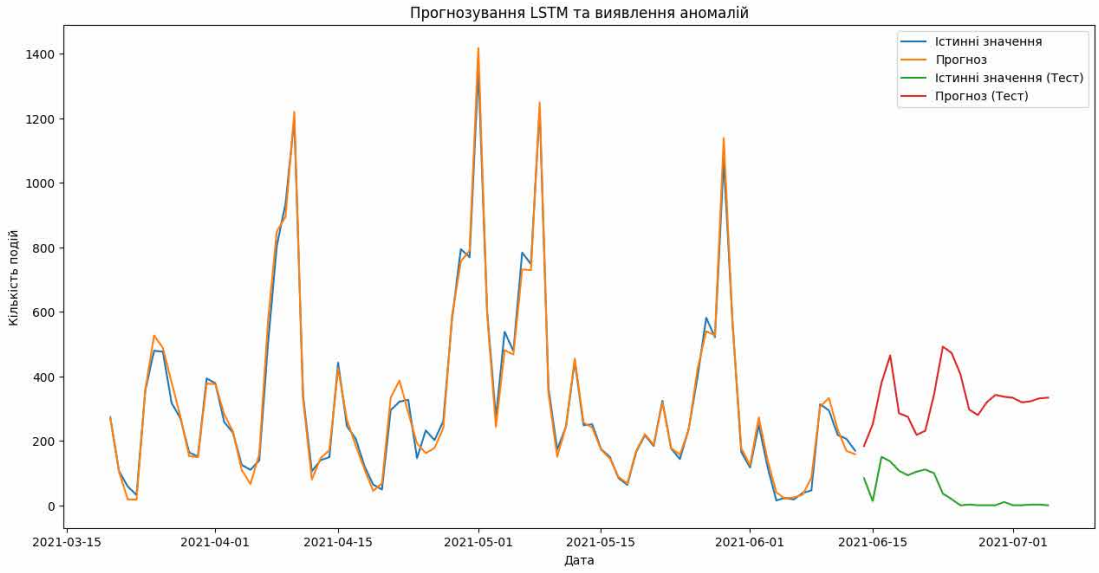


Рисунок 3.3 – Результати роботи моделі

Після здійснюється поділу даних на навчальну і тестову вибірки, де тестова вибірка складається зі значень, число яких задано у змінній *train_split_ratio* у відсотковому відношенні і рівне 20% від усього набору даних. Це останні рядки матриці, в яких потрібно виконати передбачення останнього стовпчика. Модель після навчання використовується для отримання останніх стовпців матриці на тестовій вибірці. Передбачення виконують за частинами, що мають кількість рядків, яка дорівнює *batch_size*, далі мережа навчається із новими значеннями. Після отримання всіх значень із тестової вибірки виконується зворотне перетворення з нормалізованих значень.

На отриманих результатах прогнозування розробленої моделі (рисунок 3.3) можна побачити, що модель набагато краще передбачає тренувальні далі, ніж попередні статистичні моделі, використані раніше, що свідчить про її здатність виявляти закономірності та тренди, забезпечуючи більш точне прогнозування. Середньоквадратична похибка для тренувального набору даних склала 39.24, що є набагато кращим результатом, ніж при аналізу за допомогою алгоритму SARIMA, де вона склала 138.28.

Однак через різку зміну активності у системі на тестовій вибірці модель починає відхилятись від фактичних значень, хоча і зберігає загальний напрямок тренду, який вона вивчила раніше. Це може бути зумовленим коротким проміжком часу, який використовувався для аналізу, який складав один семестр навчального року, чого може бути недостатньо для вивчення довгострокових закономірностей.

Для визначення точності моделі та її можливості адаптуватись під різні типи даних, для подальшого аналізу та прогнозування було використано новий набір даних активності у навчально-інформаційній системі Moodle в періоди з травня 2022 до травня 2024 навчальних років. На результатах роботи моделі (рисунок 3.4) можна побачити, що тренувальний прогноз доволі точно відображає фактичні значення, модель уловлює сплески та падіння активності, а у тестовому наборі

модель слідує тенденціям, однак знову не може спрогнозувати різки дуже різні зміни активності.

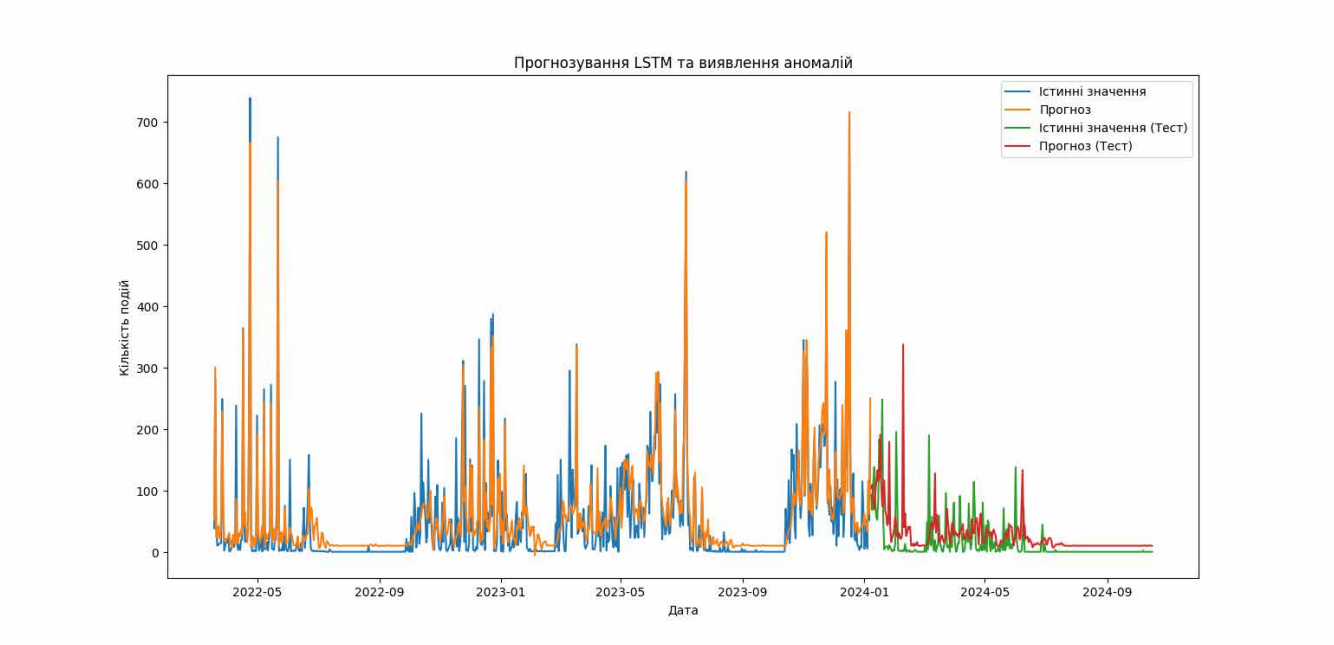


Рисунок 3.4 – Навчання моделі на більшому наборі даних

Для виявлення аномалій (рисунок 3.5) у таких тестових рядах використовуються значення, які найбільше відхиляються від прогнозованих. Їх виявлення відбувається за допомогою обчислення адаптивного порогу на основі середнього значення і стандартного відхилення помилки прогнозування (лістинг 3.3). Такий підхід до виявлення аномалій припускає, що помилки з тренувального набору мають розподіл, подібний до нормального. Вибір порогового значення в 2 стандартних відхилення передбачає охоплення 95% варіацій для нормальних похибок, і будь-які значення похибок вище цього порогового значення вважаються аномаліями.

Щоб визначати причини та умови, за яких виникають аномалії, модель враховує інші параметри з набору даних, такі як: «Event name», «User ID» з атрибута «Description» та «IP address». Усі атрибути даних відповідають попереднім, що

зазначені у таблиці 1.3. Щоб інтерактивно виводити контекст аномалії (рисунок 3.6), для побудови графіків замість бібліотеки Matplotlib використано Plotly.

Лістинг коду 3.2 – Визначення аномалій

```
# Обчислення помилок прогнозування
train_errors = np.abs(Y_train.flatten() - train_predict.flatten())
test_errors = np.abs(Y_test.flatten() - test_predict.flatten())

# Виявлення аномалій з адаптивним порогом
mean_error = np.mean(train_errors)
std_error = np.std(train_errors)
threshold = mean_error + 2 * std_error

anomalies = test_errors > threshold
```

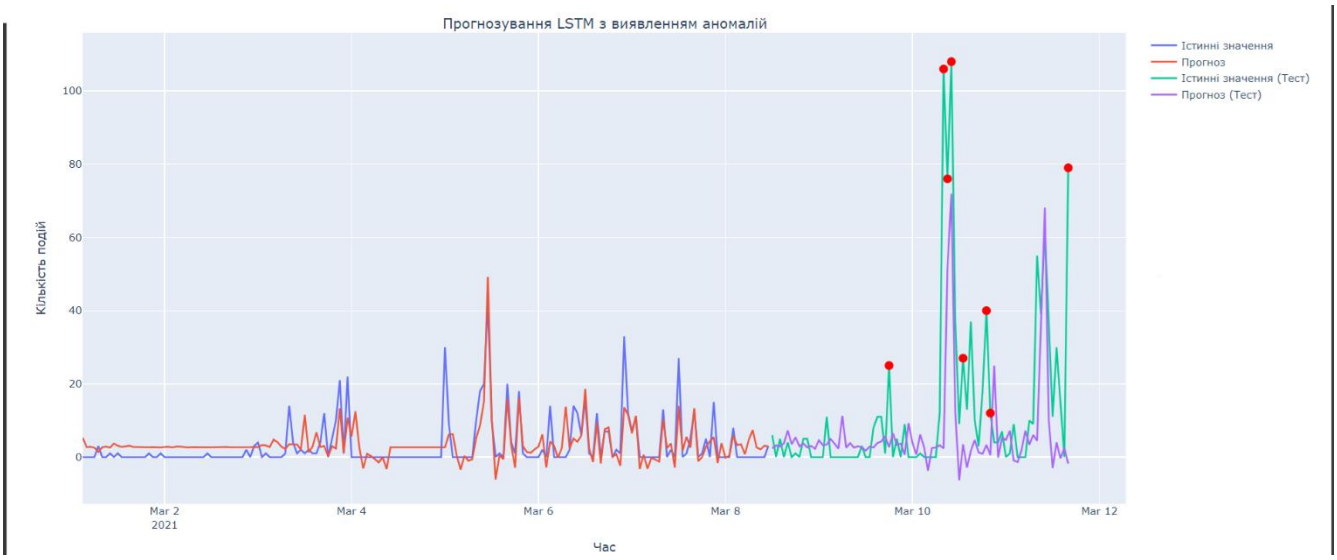


Рисунок 3.5 – Виявлення аномалій з використанням LSTM

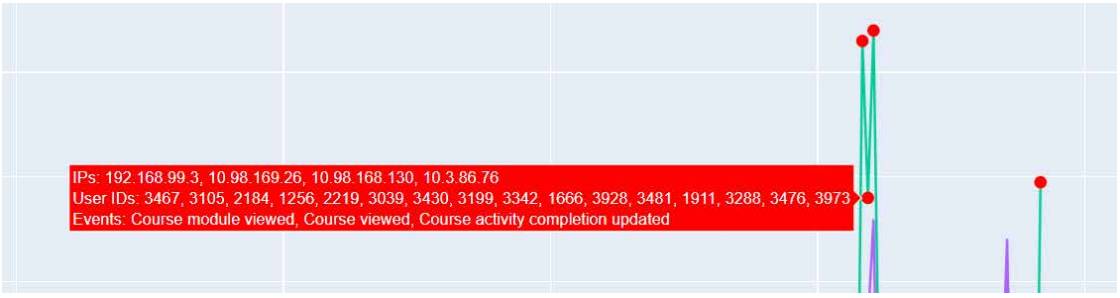


Рисунок 3.6 – Контекст аномалії

3.2. Перспективи вдосконалення системи

Незважаючи на те, що система в цілому добре визначає аномалії на часових рядах, вона все ще може мати значну похибку на тренувальних даних, як це було показано на рисунку 3.3. Переважно це зумовлено неоднорідністю часового ряду, оскільки активність студентів в різні семестри та роки суттєво відрізняється одна від одної, через що моделі доводиться постійно вивчати та оброблювати постійні сплески та спади активності у системі, що призводить до того, що вона не встигає вивчати деякі закономірності або навіть перенавчається. Для підвищення точності прогнозування моделі та виявлення аномалій можна використати так звані LSTM-автокодера.

Автокодер (AE) – це нейронна мережа, по архітектурі схожа на персептрон, яка копіює вхідні дані на вихід. Автокодери стискають вхідні дані для представлення їх у прихований простір, а потім віднавлюють з цього представлення вихідні дані. Мета полягає у тому, щоб отримати на вихідному шарі результат, найближчий до вхідного [28].

Основна ідея автокодера полягає в тому, щоб вивчити стиснене представлення вхідних даних шляхом зменшення їх розмірності, а потім відновити вихідні дані на основі цього стисненого представлення. Автокодери складаються з двох основних компонентів: кодувальної мережі, яка стискає вхідні дані до представлення меншої розмірності, і декодувальної мережі, яка відновлює вихідні дані з цього представлення. Кодувальна мережа приймає вхідні дані і застосовує до них низку перетворень, щоб зменшити їхню розмірність. Зазвичай це робиться за допомогою серії повністю з'єднаних шарів або згорткових шарів з нелінійними функціями активації. Вхідний шар передає вхідні дані до прихованого шару на вихідний шар, де вони декодуються і максимально реконструюються (рисунок 3.7). Кількість прихованих шарів в такій ШНМ може бути довільною, з умовою, що для кожної частини мережі, наприклад, кодера, кожен наступний прихований шар

повинен мати менше нейронів, ніж попередній. Вихід мережі – це стиснене представлення вхідних даних, яке фіксує їх найважливіші характеристики. Під час навчання автокодер намагається мінімізувати різницю між оригінальними вхідними даними та реконструйованими вихідними даними. Це робиться шляхом оптимізації функції втрат, такої як середньоквадратична помилка або двійкова перехресна ентропія. Параметри мереж кодера і декодера налаштовуються за допомогою зворотного поширення, щоб мінімізувати цю функцію втрат [28].

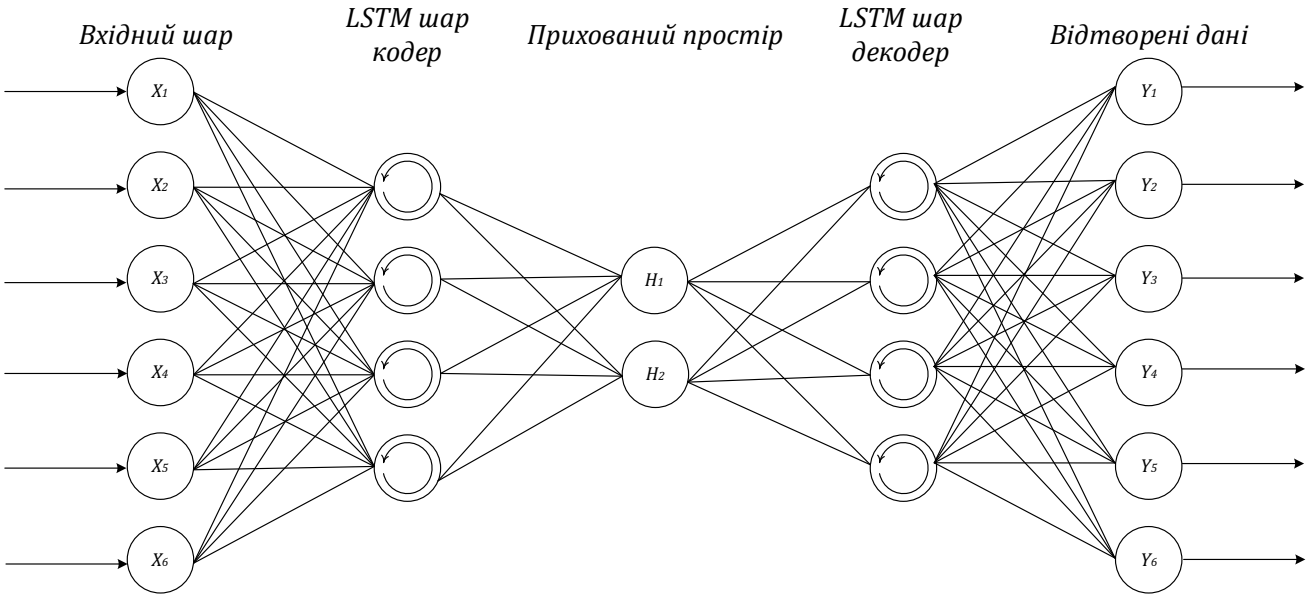


Рисунок 3.7 – Архітектура простого LSTM-кодера

Автокодер LSTM, навчений на нормальних даних часових рядів, дуже ефективно кодує дані у своїх внутрішніх представленнях. Однак, коли в мережу подаються аномальні дані, декодер не зможе правильно реконструювати ці дані, оскільки кодер не бачив таких залежностей під час навчання. Якщо помилка реконструкції перевищує поріг для будь-якої точки даних, модель позначає її як аномальну. Як зазначено у [31], завдяки здатності моделювати часові залежності та реконструювати їх LSTM-автокодер краще навчається на даних та з більшою

точністю виявляє аномалії, однак через складну архітектуру час навчання моделі може зрости.

Окрім вищезазначеного, розроблена система може бути корисною не лише для виявлення аномальної активності у системі, однак і для процесу викладання та навчання, аналізуючи успішність навчання самих студентів. Набори даних, які збирають навчальні системи типу Moodle, можуть бути корисними для проведення порівняльних досліджень поведінки студентів і закономірностей, пов'язаних з онлайн навчальними середовищами, що може допомогти сформувати модель інтелектуального аналізу освітніх даних. Крім того, можуть бути застосовані методи відбору ознак, які можуть забезпечити кращу точність прогнозування академічної успішності студентів.

У дослідженні [32] для аналізу використовується набір даних, який містить дані про активність студентів в продовж навчального року, які позначають наступні атрибути:

- академічні дані студентів – містять 15 атрибутів, включаючи «ModuleCode», «CGPA», «AttemptCount», «RemoteStudent», «HighRisk», «CW1», «CW2», «ESE» та «Result» для оцінки успішності студентів;
- логи активності Moodle – «Online C» (активність у кампусі) та «Online O» (активність поза кампусом).
- Перегляд навчальних відеоматеріалів через систему eDify – «Played», «Paused», «Likes» та «Segment».

Для попереднього дослідження було побудовану модель штучної нейронної мережі, яка використовує бінарну класифікацію для передбачення, чи склав студент модуль чи провалив (класи Pass і Fail). Модель змогла правильно класифікувати дані з точністю 80.3% та має тренд до поступового зменшення втрат на тренувальних даних, що свідчить про її здатність адаптуватись до даних (рисунок 3.8). Модель має гарну точність для класу Pass, що рівна 88% успішних виявлень

від загальної кількості випадків, однак для класу Fail модель передбачає лише 33% (рисунок 3.9)

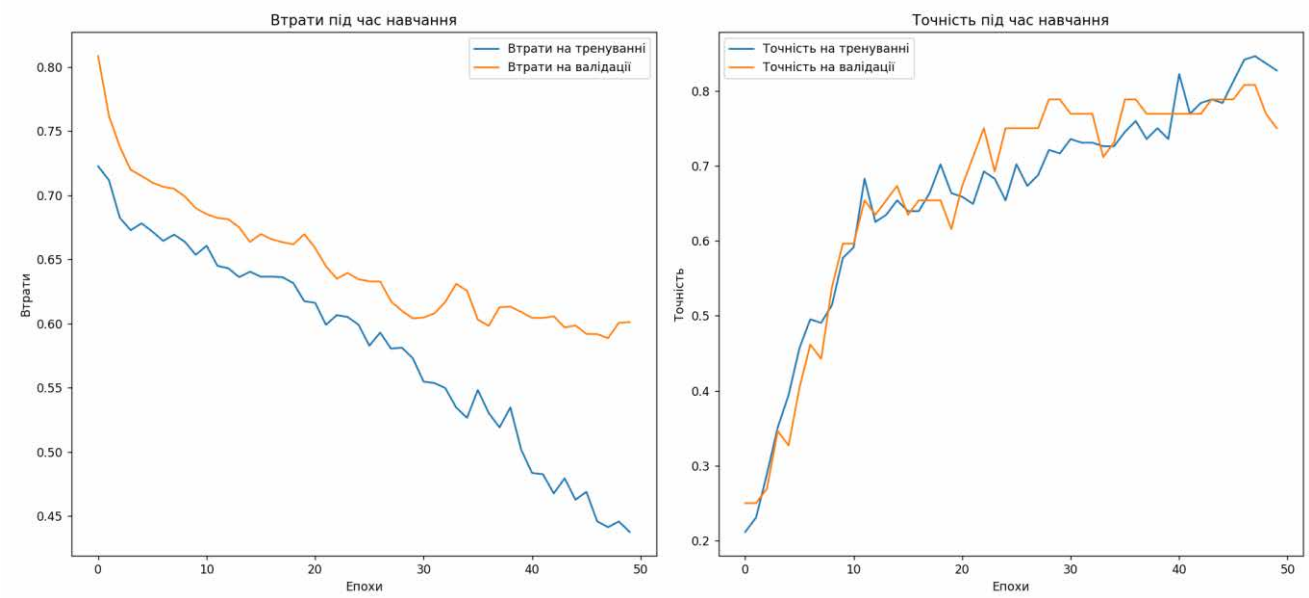


Рисунок 3.8 – Точність навчання моделі ШНМ при поперелнтому аналізі

Точність моделі: 80.30%

	precision	recall	f1-score	support
Fail	0.30	0.33	0.32	9
Pass	0.89	0.88	0.88	57
accuracy			0.80	66
macro avg	0.60	0.61	0.60	66
weighted avg	0.81	0.80	0.81	66

Рисунок 3.9 – Попередній результати оцінки моделі класифікації

ВИСНОВКИ

В рамках даної магістерської кваліфікаційної роботи отримані наступні головні теоретичні та практичні результати:

1. Проведено аналіз важливості використання цифрових слідів для виявлення атак.

2. Здійснено лог-аналіз цифрових слідів, отриманих з бази даних навчальної інформаційної системи університету.

3. Проаналізовано отримані журнали логування за допомогою статистичних методів ARMA, ARIMA та SARIMA для прогнозування часових рядів.

4. Проведено аналіз додаткових методів аналізу часових рядів задля покращення попередніх результатів та виявлення аномальної активності у них.

5. Запропоновано та реалізовано метод аналізу цифрових слідів з використанням штучних нейронних мереж. Використано ШНМ LSTM з довгою короткостроковою пам’яттю, яка призначена для аналізу довгострокових залежностей у часових рядах.

6. Розроблено програмне забезпечення для демонстрації та випробування розробленого методу.

7. В результаті експериментальних досліджень продемонстровано, що з використанням LSTM-моделі результати прогнозування часового ряду значно покращились і мережі змогла виявляти аномальну активність у системі.

8. Запропонований метод може бути використаний для подальшого покращення алгоритмів аналізу захищених інформаційних систем університету.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Alex Mathew. «The Power of Cybersecurity Data Science in Protecting Digital Footprints», 2023.

2. Juanita Blue, Joan Condell, Tom Lunney. «Digital Footprints: Your Unique Identity», 2018.

3. What is a digital footprint? [електронний ресурс] / Режим доступу: https://www.techtarget.com/whatis/definition/digital-footprint

4. What is a digital footprint and why does it matter? [електронний ресурс] / Режим доступу: https://www.allstateidentityprotection.com/content-hub/whats-a-digital-footprint-and-why-does-it-matter

5. Expanding your Active and Passive Digital Footprint [електронний ресурс] / Режим доступу: https://www.sourcr.com/blog/expanding-your-active-and-passive-digital-footprint/

6. What is Security Analytics and What are Its Use Cases? [електронний ресурс] / Режим доступу:https://www.techtarget.com/searchsecurity/definition/security-analytics

7. What Are Log Files? A Guide to Understanding Log Files [електронний ресурс] / Режим доступу: https://www.xcitium.com/log-files/

8. Himal Lalla, Stephen Flowerday, Tendai Sanyamahwe Paul Tarwireyi. «A log file digital forensic model», 2012

9. A. Chuvakin, K. Schmidt, C. Phillips. «A. Logging and Log Management», 2013.

10. N. K. Singh, D. S. Tomar, B. N. Roy. «An Approach to Understand the End User Behavior through Log Analysis», 2010

11. Що таке логи та як з ними працювати [електронний ресурс] / Режим доступу: https://hostiq.ua/wiki/ukr/log/

12. Виявлення проблем у log-файлах за допомогою аналітики [електронний ресурс] / Режим доступу: https://habr.com/ru/companies/otus/articles/781598/

13. Logs: Unveiling the Digital Footprints of Systems and Applications [електронний ресурс] / Режим доступу: https://blog.jirivanek.eu/en/logs-unveiling-the-digital-footprints-of-systems-and-applications/

14. Daniela Rotelli, Giuseppe Fiorentino, Anna Monreale. «Making Sense of Moodle Log Data», 2021

15. Daniela Rotelli, Anna Monreale. «Processing and Understanding Moodle Log Data and Their Temporal Dimension», 2023

16. Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu. «Experience Report: System Log Analysis for Anomaly Detection», 2016

17. Varun Chandola, Arindam Banerjee, and Vipin Kumar. «Anomaly Detection: A Survey», 2007

18. Martin Charlton, Alberto Caimo. «Time Series Analysis», 2012

19. What Is a Time Series and How Is It Used? [електронний ресурс] / Режим доступу: https://www.timescale.com/blog/time-series-introduction/

20. ARIMA & SARIMA: Real-World Time Series Forecasting [електронний ресурс] / Режим доступу: https://neptune.ai/blog/arima-sarima-real-world-time-series-forecasting-guide

21. Time Series Analysis - ARMA, ARIMA, SARIMA [електронний ресурс] / Режим доступу: https://www.visual-design.net/post/time-series-analysis-arma-arima-sarima

22. Vibha Masti. «Unit 3 - Time Series Analysis» / Режим доступу: https://vibhamasti.github.io/notes/cs312/DA%20Unit%203.pdf

23. Zahra Zamanzadeh Darban, Geoffrey I. Webb, Shirui Pan, Charu C. Aggarwal, Mahsa Salehi «Deep Learning for Time Series Anomaly Detection: A Survey», 2024

24. Sergio Trilles, Sahibzada Saadoon Hammad, Ditsuhi Iskandaryan «Anomaly detection based on Artificial Intelligence of Things: A Systematic Literature Mapping», 2024

25. Simon Haykin. «Neural Networks and Learning Machines. Third Edition.», 1999

26. Annette Lopez Davila. «Neural Networks: The Backpropagation Algorithm», 2017

27. Recurrent Neural Networks and LSTM Forecasting [електронний ресурс] / Режим доступу: https://medium.com/@hession520/recurrent-neural-networks-and-lstm-2b15ce45d4bf

28. Aggarwal, Sakshi. «LSTM based Anomaly Detection in Time Series for United States exports and imports», 2023

29. Ralf C. Staudemeyer, Eric Rothstein Morris. «Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks», 2019

30. LSTM — нейронна мережа з високоякісною пам'яттю [електронний ресурс] / Режим доступу: https://neurohive.io/ru/osnovy-data-science/lstm-nejronnaja-set/

31. Lachekhab F., Benzaoui M., Tadjer S.A., Bensmaine A., Hamma H. «LSTM-Autoencoder Deep LearningModel for Anomaly Detection inElectric Motor», 2024

32. Hasan R., Palaniappan, S., Mahmood, S., Abbas, A., Sarker, K.U. «Dataset of Students' PerformanceUsing Student Information System, Moodle and the Mobile Application "eDify"», 2021

Додаток А

ЛІСТИНГ КОДУ ПРОГРАМИ РОЗРОБЛЕНОЇ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ

```
import pandas as pd
import numpy as np
from keras import Sequential
from keras.layers import LSTM, Dense, Dropout # type: ignore
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
import plotly.graph_objs as go
from plotly.subplots import make_subplots

def load_and_preprocess_data(file_path, freq='D'):
    data = pd.read_csv(file_path, encoding='ISO-8859-1')
    data.rename(columns={'ï»¿Time': 'Time'}, inplace=True)
         data['Time']  =  pd.to_datetime(data['Time'],  format='%d/%m/%y,  %H:%M',
errors='coerce')

    if freq == 'H':
        data['Time'] = data['Time'].dt.floor('H')
        events_by_time = data.groupby('Time').size()
        events_by_time.index = pd.to_datetime(events_by_time.index)
        events_by_time = events_by_time.asfreq('H', fill_value=0)
    else:
        data['Date'] = data['Time'].dt.date
        events_by_time = data.groupby('Date').size()
        events_by_time.index = pd.to_datetime(events_by_time.index)
        events_by_time = events_by_time.asfreq('D', fill_value=0)

    return events_by_time, data

def scale_data(data):
    values = data.values.reshape(-1, 1)
    scaler = MinMaxScaler(feature_range=(-1, 1))
    scaled_data = scaler.fit_transform(values)
    return scaled_data, scaler
```

```
def create_dataset(dataset, time_step=1):
    X, Y = [], []
    for i in range(len(dataset) - time_step - 1):
        a = dataset[i:(i + time_step), 0]
        X.append(a)
        Y.append(dataset[i + time_step, 0])
    return np.array(X), np.array(Y)

def build_lstm_model(time_step, n_neurons, dropout_rate, output_neurons):
    model = Sequential()
    model.add(LSTM(n_neurons, return_sequences=True, input_shape=(time_step, 1)))
    model.add(Dropout(dropout_rate))
    model.add(LSTM(n_neurons, return_sequences=False))
    model.add(Dropout(dropout_rate))
    model.add(Dense(output_neurons))
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model

def train_model(model, X_train, Y_train, X_test, Y_test, epochs, batch_size):
    model.fit(
        X_train, Y_train,
        epochs=epochs,
        batch_size=batch_size,
        validation_data=(X_test, Y_test),
        verbose=1,
        shuffle=False
    )

def detect_anomalies(true_values, predicted_values, threshold_factor=3):
    errors = np.abs(true_values.flatten() - predicted_values.flatten())
    mean_error = np.mean(errors)
    std_error = np.std(errors)
    threshold = mean_error + threshold_factor * std_error

    anomalies = errors > threshold
    return anomalies

def get_anomaly_context(context_data, anomaly_indices, timestamps):
```

```
    context_info = []
    for index in anomaly_indices:
            ip_info = context_data[context_data['Time'] == timestamps[index]]['IP
address'].unique()
                            user_ids = context_data[context_data['Time'] ==
timestamps[index]]['Description'].str.extract(r"user with id
'(\d+)'")[0].dropna().astype(str).unique()
          actions = context_data[context_data['Time'] == timestamps[index]]['Event
name'].unique()

        context_info.append({
            'Index': index,
            'IP': ', '.join(ip_info),
            'User IDs': ', '.join(user_ids),
            'Actions': ', '.join(actions)
        })
    return pd.DataFrame(context_info)

def plot_results(events_by_time, Y_train, train_predict, time_step,
                 Y_test=None, test_predict=None,
                 anomalies_test=None, context_data=None):
                                                test_timestamps =
events_by_time.index[time_step+len(Y_train):time_step+len(Y_train)+len(Y_test)]

      fig = make_subplots(rows=1, cols=1, subplot_titles=["Прогнозування LSTM з
виявленням аномалій"])

    # Додавання тренувальних та тестових даних
    fig.add_trace(go.Scatter(x=events_by_time.index[time_step:time_step+len(Y_train
)],
                             y=Y_train.flatten(), mode='lines', name='Істинні значення
(Тренування)'))
    fig.add_trace(go.Scatter(x=events_by_time.index[time_step:time_step+len(train_p
redict)],
                               y=train_predict.flatten(), mode='lines', name='Прогноз
(Тренування)'))
    fig.add_trace(go.Scatter(x=test_timestamps,
                             y=Y_test.flatten(), mode='lines', name='Істинні значення
(Тест)'))
```

```
    fig.add_trace(go.Scatter(x=test_timestamps,
                             y=test_predict.flatten(), mode='lines', name='Прогноз
(Тест)'))

    # Додавання аномалій
    if anomalies_test is not None and context_data is not None:
        test_anomaly_indices = np.where(anomalies_test)[0]
         anomaly_context = get_anomaly_context(context_data, test_anomaly_indices,
test_timestamps)

        for _, row in anomaly_context.iterrows():
            annotation_text = (
                f"IP: {row['IP']}<br>"
                f"User IDs: {row['User IDs']}<br>"
                f"Actions: {row['Actions']}"
            )
            fig.add_trace(go.Scatter(x=[test_timestamps[row['Index']]],
                                     y=[Y_test.flatten()[row['Index']]],
                                     mode='markers',
                                     marker=dict(color='red', size=10),
                                     name='Аномалія (Тест)',
                                     text=annotation_text,
                                     hoverinfo='text'))

         fig.update_layout(title="Прогнозування LSTM з виявленням аномалій",
xaxis_title="Час", yaxis_title="Кількість подій")
    fig.show()

def main():
    n_neurons = 50
    dropout_rate = 0.2
    output_neurons = 1

    epochs = 400
    batch_size = 10

    time_step = 20
    train_split_ratio = 0.7
```

```
    file_path = 'data/moodle_activity_21_logs.csv'
    events_by_time, context_data = load_and_preprocess_data(file_path, freq='D')
    scaled_data, scaler = scale_data(events_by_time)

    X, Y = create_dataset(scaled_data, time_step)
    X = X.reshape(X.shape[0], X.shape[1], 1)

    train_size = int(len(X) * train_split_ratio)
    X_train, X_test = X[:train_size], X[train_size:]
    Y_train, Y_test = Y[:train_size], Y[train_size:]

    model = build_lstm_model(time_step, n_neurons, dropout_rate, output_neurons)
    train_model(model, X_train, Y_train, X_test, Y_test, epochs, batch_size)

    train_predict = model.predict(X_train)
    test_predict = model.predict(X_test)

    train_predict = scaler.inverse_transform(train_predict)
    test_predict = scaler.inverse_transform(test_predict)
    Y_train = scaler.inverse_transform(Y_train.reshape(-1, 1))
    Y_test = scaler.inverse_transform(Y_test.reshape(-1, 1))

    anomalies_test = detect_anomalies(Y_test, test_predict)

    plot_results(events_by_time, Y_train, train_predict, time_step,
                 Y_test, test_predict,
                 anomalies_test=anomalies_test, context_data=context_data)

if __name__ == "__main__":
    main()
```

Додаток Б

ЛІСТИНГ КОДУ ПРОГРАМИ МОДЕЛЕЙ ARMA, ARIMA, SARIMA

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.arima.model import ARIMA as ARMA
from statsmodels.tsa.stattools import adfuller
import numpy as np

def load_and_preprocess_data(file_path):
    data = pd.read_csv(file_path, encoding='ISO-8859-1')
    data['Time'] = pd.to_datetime(data['Time'], format='%d/%m/%y, %H:%M')
    data['Date'] = data['Time'].dt.date
    return data

def aggregate_events_by_day(data):
    events_by_day = data.groupby('Date').size()
    events_by_day.index = pd.to_datetime(events_by_day.index)
    events_by_day = events_by_day.asfreq('D')
    return events_by_day

def build_arima_model(data, order=(1, 1, 1)):
    model = ARIMA(data, order=order).fit()
    return model

def build_sarima_model(data, order=(1, 0, 0), seasonal_order=(1, 0, 0, 7)):
    model = SARIMAX(data, order=order, seasonal_order=seasonal_order).fit()
    return model

def build_arma_model(data, order=(1, 0, 1)):
    model = ARMA(data, order=order).fit()
    return model

def forecast_model(model, data, steps=10):
    forecast_steps = len(data) + steps
```

```
    df_pred = model.predict(start=0, end=forecast_steps)
    return df_pred

def plot_forecast(data, df_pred, title='Прогноз'):
    plt.figure(figsize=(12, 6))
    plt.title(title, fontsize=20)
    plt.plot(data.index, data, label='Фактичні дані', color='#1F77B4')
    plt.plot(df_pred.index, df_pred, label='Прогнозовані дані', color='#FF7F0E')
    plt.legend(fontsize=15, loc='upper left')
    plt.xlabel('Дата', fontsize=15)
    plt.ylabel('Кількість подій', fontsize=15)
    plt.grid(True)
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

def adfuller_test(data):
    result = adfuller(data.dropna())
    print('ADF Statistic:', result[0])
    print('p-value:', result[1])

def main():
    file_path = r"data/moodle_activity_21_logs.csv"
    data = load_and_preprocess_data(file_path)
    events_by_day = aggregate_events_by_day(data)
    arma_model = build_arma_model(events_by_day, order=(1, 0, 1))
    df_pred_arma = forecast_model(arma_model, events_by_day, steps=10)
    plot_forecast(events_by_day, df_pred_arma, title='Прогноз ARMA')
    arima_model = build_arima_model(events_by_day, order=(1, 1, 1))
    df_pred_arima = forecast_model(arima_model, events_by_day, steps=10)
    adfuller_test(events_by_day)
    plot_forecast(events_by_day, df_pred_arima, title='Прогноз ARIMA')
          sarima_model  =  build_sarima_model(events_by_day,  order=(1,  0,  0),
seasonal_order=(1, 0, 0, 7))
    df_pred_sarima = forecast_model(sarima_model, events_by_day, steps=10)
    plot_forecast(events_by_day, df_pred_sarima, title='Прогноз SARIMA')

if __name__ == "__main__":
    main()
```

Додаток В

ЛІСТИНГ КОДУ ПРОГРАМИ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ КЛАСИФІКАЦІЇ

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.utils import class_weight
from keras.models import Sequential # type: ignore
from keras.layers import Dense, Dropout # type: ignore
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix,
ConfusionMatrixDisplay, roc_curve, auc
import matplotlib.pyplot as plt
import numpy as np

data = pd.read_csv('data/students_success_logs.csv')  # Заміни шлях на актуальний

selected_features = [
    'CGPA', 'AttemptCount', 'RemoteStudent', 'Probation', 'HighRisk', 'TermExceeded',
    'AtRisk', 'AtRiskSSC', 'OtherModules', 'PlagiarismHistory', 'CW1', 'CW2', 'ESE',
    'Online C', 'Online O', 'Played', 'Paused', 'Likes', 'Segment'
]
X = data[selected_features]
y = data['Result']

label_encodable_cols = [
    'CGPA', 'AttemptCount', 'CW1', 'CW2', 'ESE', 'Online C', 'Online O'
]
binary_cols = [
    'RemoteStudent', 'Probation', 'HighRisk', 'TermExceeded',
    'AtRisk', 'AtRiskSSC', 'OtherModules', 'PlagiarismHistory'
]

for col in label_encodable_cols:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
```

```
X = pd.get_dummies(X, columns=binary_cols, drop_first=True)

y = LabelEncoder().fit_transform(y)

scaler = MinMaxScaler()
X = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

class_weights = class_weight.compute_class_weight(
    class_weight='balanced',
    classes=np.unique(y_train),
    y=y_train
)
class_weights = dict(enumerate(class_weights))

model = Sequential()
model.add(Dense(units=64, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(units=32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(units=1, activation='sigmoid'))  # Для бінарної класифікації
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

history = model.fit(
    X_train, y_train,
    epochs=50,
    batch_size=32,
    validation_split=0.2,
    class_weight=class_weights,
    verbose=1
)

y_pred = (model.predict(X_test) > 0.5).astype('int32')
print(f"Точність моделі: {accuracy_score(y_test, y_pred) * 100:.2f}%")
print(classification_report(y_test, y_pred, target_names=['Fail', 'Pass'],
zero_division=1))

fig, ax = plt.subplots(1, 2, figsize=(14, 5))
```

```
ax[0].plot(history.history['loss'], label='Втрати на тренуванні')
ax[0].plot(history.history['val_loss'], label='Втрати на валідації')
ax[0].set_title('Втрати під час навчання')
ax[0].set_xlabel('Епохи')
ax[0].set_ylabel('Втрати')
ax[0].legend()
ax[1].plot(history.history['accuracy'], label='Точність на тренуванні')
ax[1].plot(history.history['val_accuracy'], label='Точність на валідації')
ax[1].set_title('Точність під час навчання')
ax[1].set_xlabel('Епохи')
ax[1].set_ylabel('Точність')
ax[1].legend()
plt.tight_layout()
plt.show()

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Fail', 'Pass'])
disp.plot(cmap=plt.cm.Blues)
plt.title('Матриця неточностей')
plt.xlabel('Передбачений клас')
plt.ylabel('Реальний клас')
plt.show()

fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', label=f'ROC-крива (площа = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlabel('Хибнопозитивна частота')
plt.ylabel('Істиннопозитивна частота')
plt.title('ROC-крива')
plt.legend(loc='lower right')
plt.show()
```