Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи машинного навчання у Big Data”

КбПЗ_2024

Виконав здобувач вищої освіти
ІІ курсу, групи КН-23М
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Ланецький В.С.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Марченко К.М.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Ланецькому Владиславу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи машинного навчання у Big Data*

2. Керівник роботи *Марченко Костянтин Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 18-13 від 07.08.2024 року

3. Строк подання студентом роботи до захисту *2.12.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи машинного навчання у Big Data*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Маркетингове та економічне обґрунтування ІТ-проєкту.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

КОПІЯ 2024

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Доренська А.О.	05.10.2024	14.11.2024
Охорона праці	Марченко К.М., к.т.н., доцент	06.10.2024	16.11.2024

7. Дата видачі завдання « 6 » вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2024 р.	
3.	Розробка моделі компонента	20.10.2024 р.	
4.	Розробка структур даних	25.10.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2024 р.	
6.	Програмування алгоритмів	10.11.2024 р.	
7.	Розрахунок економічної ефективності	13.11.2024 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2024 р.	
9.	Оформлення ПЗ	17.11.2024 р.	
10.	Попередній захист роботи	2.12.2024 р.	

КОПЗ_2024

Дата видачі завдання
« 6 » вересня 2024 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2024 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Ланецький В.С. Дослідження та програмна реалізація системи машинного навчання у Big Data. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи машинного навчання у Big Data.

Метою розробки є дослідження та програмна реалізація системи машинного навчання у Big Data.

Об'єктом дослідження є процес машинного навчання у Big Data.

Предметом дослідження є методи машинного навчання у Big Data.

Методи дослідження базуються на методах машинного навчання та обробки Big Data, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи машинного навчання у Big Data.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: Комп'ютерні науки, машинне навчання, Big Data

КОПЗ – 2024

ABSTRACT

Lanetskyi V.S. Research and software implementation of a machine learning system in Big Data. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the second (master's) level of higher education, software has been developed, which is intended for a machine learning system in Big Data.

The purpose of the development is the research and software implementation of a machine learning system in Big Data.

The object of the research is the process of machine learning in Big Data.

The subject of the research is machine learning methods in Big Data.

The research methods are based on machine learning methods and Big Data processing, methods of mathematical statistics, and methods of software development.

The result of the work is a software implementation of a machine learning system in Big Data.

In the process of working on the software model, an analysis of existing hardware and software tools was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs with Windows 10/11.

The program is developed in the Python environment.

Keywords: Computer Science, Machine Learning, Big Data

КОПЗ_2024

ЗМІСТ

КОПЗ_2024

КбПЗ_2024

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

EOM	–	електронно-обчислювальна машина
ATM	–	Asynchronous Transfer Mode – асинхронний спосіб передачі даних
BER	–	Bit Error Rate – імовірність ушкодження одного біта
CBWFQ	–	class based weighted fair queueing
DiffServ	–	Differentiated Services – механізм який залежно від вимог до якості обслуговування записує в IP заголовки пакетів спеціальні мітки
DSCP	–	DiffServ CoSde Point
ICMP	–	Internet CoSntrol Message Protocol – міжмережний протокол управляючих повідомлень
ISP	–	Internet Service Provider – провайдер
LLQ	–	low latency queueing
MPLS-TE	–	Multiprotocol Label Switching Traffic Engineering
PQ	–	priority queueing
RIO	–	RED with Input Output
RTT	–	Round Trip Time – час між відправкою запиту та отриманням відповіді
STM	–	Synchronous Transfer Mode – синхронний спосіб передачі даних
QoS	–	Quality of Service – якість обслуговування
WFQ	–	Weighted Fair Queuing – механізм планування пакетних потоків даних
WRED	–	Weighted random early detection – взвішане значення довжини черги, у якості фактора, визначаючого ймовірність відкидання пакета

КПЗ – 2024

ВСТУП

Актуальність теми. Один з можливих сценаріїв застосування машинного навчання (МН) у системах зберігання даних (СЗД) – забезпечення якості обслуговування (QoS) на рівні додатків.

Однією з перспективних областей застосування машинного навчання можна назвати зберігання даних. Розумні пристрої зберігання усе ще залишаються концепцією майбутнього, тієї самої Next Big Thing, що так чекають експерти. Такі пристрої зможуть «розуміти» поводження додатків на основі специфічних факторів, визначати рівень вимог до доступності й надійності систем зберігання й автоматично підбудовуватися під необхідний клас рішень.

Говорячи про машинне навчання (МН), багато хто мають на увазі цілий ряд «забавних» сценаріїв, від яких дуже мало практичної користі. Як правило, мова йде про генератори текстів, які на перевірку виявляються нісенітницею, про автоматизовані сценарії для кіно, імітації стилю живопису старих майстрів і інших сумнівних проектів. Інформаційний шум навколо такої активності формує неправильне уявлення про машинне навчання й штучний інтелект, у те час як існують цілком конкретні області їхнього застосування в науці, техніку й в інформаційних технологіях.

У першу чергу машинне навчання допомагає інженерам автоматизувати ту або іншу інтелектуальну діяльність, адже комп'ютер здатний обробляти великий обсяг даних за менший час і швидше приймати конкретні рішення. Безумовно, скасувати посаду охоронця правопорядку й замінити її штучним інтелектом не вдасться, але цілком можливо застосувати потрібний алгоритм для точного розпізнавання осіб підозрюваних у великому потоці людей на жвавій вулиці.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи машинного навчання у Big Data.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем машинного навчання у Big Data.

КПІ – 2024

– Дослідження системи машинного навчання у Big Data.

– Програмна реалізація системи машинного навчання у Big Data.

Об'єктом дослідження є процес машинного навчання у Big Data.

Предметом дослідження є методи машинного навчання у Big Data.

Методи дослідження базуються на методах машинного навчання та обробки Big Data, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод машинного навчання у Big Data.

– Розроблено вітчизняний продукт машинного навчання у Big Data, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі машинного навчання у Big Data.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2024 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи машинного навчання у Big Data, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

КбПЗ_2024

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

За кілька років дослідники істотно просунулися в напрямку машинного навчання (ML; Machine Learning) і глибинного навчання (Deep Learning) – і, що важливіше, суспільство стало готове до нових технологій.

На жаль, спекулюючи на популярній темі машинного навчання, журналісти зосередилися на зовсім непотрібних людству областях його застосування: генерації текстів і сценаріїв для божевільних фільмів, написанні картин у стилі відомих художників і т.д. Частина подібних статей і зовсім скачується до панічних настроїв начебто “Незабаром ми все залишимося без роботи”.

Такий підхід створює в суспільстві невірне подання про машинне навчання й сценарії його використання в реальних завданнях. На практиці ML дозволяє нам автоматизувати той або інший вид розумової діяльності. Машина здатна приймати рішення так само, як людина, і навіть швидше або точніше, ніж людина, – за рахунок обробки більшої кількості даних. Проте важливо розуміти, що алгоритм здатний добре виконувати тільки одне певне завдання.

Ми не можемо замінити комп'ютером поліцейського, але ми можемо навчити комп'ютер розпізнавати в потоці людей особи, схожі на особу злочинця. Вся міць алгоритмів висихає тоді, коли наявних даних стає недостатньо або відбувається подія, не схожа на попередню – відомі системі. Машина не здатна на творчість у незнайомій ситуації.

КПЗ-2024

1.2 Область застосування

Вертаючись до теми систем зберігання даних, задамося питанням: як підходи ML можуть застосовуватися у світі твердих і твердотільних дисків?

Настроювання параметрів зберігання на лету

Почнемо зі зміни параметрів роботи всієї системи в цілому і її окремих компонентах.

Алгоритм, вивчаючи безліч даних і стежачи за характеристиками пристроїв, може змінювати параметри поводження системи зберігання. Так, кілька років назад почали дослідницький проект “RAIDIX Autopilot”, метою якого була автоматична зміна параметрів роботи СЗД за рахунок вибору оптимальних настроювань для поточного типу навантаження. Надалі проект розгалузився на кілька напрямків.

Були спроби реалізації подібного «автопілоту» для flash-накопичувачів. Слід зазначити, що розроблювачам пристроїв на основі NAND Flash завжди доводиться йти на компроміси, вибираючи між обсягом, надійністю й максимальною кількістю перезаписів. У цьому випадку можливо змінювати параметри пристрою, налаштовуючи поводження контролера через виставляння параметрів регістрів. Таких регістрів може бути 50-100 у планарної пам'яті й більше 1000 в 3D NAND.

Яскравий приклад машинного навчання для зміни параметрів різних регістрів демонструє стартап NVMdurance.

Предіктивна аналітика

Інше важливе завдання пов'язана з аналізом поводження СЗД і предіктивной аналітикою. Алгоритм може аналізувати журнали й історію подій і пророкувати потенційні проблеми с кластером зберігання. Так, більшу роботу на шляху до “розумного дата-центру” проробили з Nimble Storage.

Розвиток мобільних технологій і поведінкові характеристики нового покоління користувачів приведуть до того, що через 5-7 років засоби,

КОПЗ_2024

використовувані для керування інфраструктурою, будуть розуміти природна мова. Уже зараз з'являються аналоги Microsoft IFTTT, які розуміють завдання начебто “Створити посилання в Twitter для всіх моїх відновлень в Facebook”.

Рано або пізно системи на підприємствах зможуть по простому запиті надати інформацію про стан інфраструктури, виділити потрібні ресурси або провести аналіз навантаження.

Частина того, про що я написав, уже існує й використовується в реальних продуктах, інше з'явиться через кілька років. Ясно одне – підхід до керування інфраструктурою підприємства зміниться й зміниться кардинально.

QoS (якість обслуговування) на рівні додатків

Третій важливий випадок застосування ML – забезпечення QoS для певних додатків. Очевидно, що старий підхід “один додаток – один LUN” більше не працює. Ми неодноразово зіштовхувалися із ситуацією, коли з тим самим томом з того самого хосту працюють різні додатки. При цьому багато хто з них зовсім не критичні для бізнесу, але дуже вимогливі до ресурсів.

Для рішення цієї проблеми реалізували проект QoSmic. Фактично, ПЗ навчило СЗД розпізнавати додатки й навантаження по характерних ознаках IO.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи машинного навчання у Big Data, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

КОПЗ-2024

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Відповідь на питання "Які алгоритми машинного навчання використовувати?" завжди звучить так: "Це залежить від ряду обставин". Це залежить від розміру, якості й природи даних. Це залежить від того, що потрібно зробити з відповіддю. Це залежить від того, як математичний алгоритм був перетворений в інструкції для вашого комп'ютера. І це залежить від того, скільки часу у вас є. Навіть самі досвідчені фахівці з даних не зможуть визначити найкращий алгоритм, не спробувавши його.

Пам'ятка по алгоритмах машинного навчання

Пам'ятка по алгоритмах машинного навчання для Студії машинного навчання Microsoft Azure допоможе вибрати правильний алгоритм із бібліотеки алгоритмів машинного навчання Microsoft Azure для рішень прогнозної аналітики. У цій статті описано, як користуватися пам'яткою.

Це пам'ятка розрахована на дуже вузьку аудиторію: початківців фахівців з даних, які намагаються вибрати алгоритм для початку роботи зі Студією машинного навчання Microsoft Azure. Це означає, що в пам'ятці використовуються деякі узагальнення й спрощення, але вона направить вас у вірному напрямку. Це також означає, що існує безліч алгоритмів, які не описані в пам'ятці. У міру розвитку машинного навчання Azure і поповнення списку доступних методів ми будемо додавати їх у пам'ятку.

Ці рекомендації являють собою об'єднані відгуки й поради від великої кількості фахівців з даних і експертів по машинному навчанню. Ми дійшли згоди не у всьому, але спробували прийти до єдиної думки по складних питаннях.

КПЗ_2024

Більшість інструкцій, у яких виникли розбіжності, починаються зі слів "Це залежить від..."

Як користуватися пам'яткою

Позначення шляху й алгоритму на схемі варто читати як "для <позначення шляху> використовуйте <алгоритм>". Наприклад, "для швидкості використовуйте двокласну логістичну регресію". Іноді може використовуватися більше однієї галузі алгоритму. Іноді жодна з галузей алгоритму не підходить ідеально. Ці рекомендації наближені, тому не потрібно турбуватися про те, що вони не є точними. Деякі фахівці з даних, з якими я спілкувався, говорили, що єдиний надійний спосіб визначити найкращий алгоритм – спробувати їх усе.

Різновиди машинного навчання

Контрольоване

Контрольовані алгоритми навчання виконують прогнозування на основі набору прикладів. Наприклад, вартість акцій у минулому дозволяє припустити вартість акцій у майбутньому. Кожний приклад, використовуваний для навчання, позначається значенням, що цікавить нас, – у цьому випадку це вартість акцій. Контрольований алгоритм навчання виконує пошук шаблонів у цих значеннях. Він може використовувати будь-які відповідні відомості – день тижня, пора року, фінансові дані компанії, тип галузі, наявність важливих геополітичних подій – і кожний алгоритм шукає шаблони різних типів. Після того як алгоритм виявив найкращий шаблон, він може на основі цього шаблона пророчити нерозмічені перевірочні дані – завтрашні ціни.

Це популярний і корисний тип машинного навчання. За одним виключенням всі модулі машинного навчання Azure є контрольованими алгоритмами навчання. У машинному навчанні Azure існує кілька типів контрольованих алгоритмів навчання: класифікація, регресія й виявлення аномалій.

– Класифікація**.** Якщо дані використовуються для прогнозування категорії, контрольоване навчання також називається класифікацією. Це

КПЗ-2024

відбувається, коли рисунок визначається як зображення "кішки" або "собаки". При наявності тільки двох варіантів така класифікація називається двокласної або біноміальної. При наявності декількох категорій, як при прогнозуванні переможця турніру NCAA March Madness, класифікація називається багатокласовою**.**

– Регресія. При прогнозуванні значення, наприклад вартості акцій, контрольоване навчання називається регресією.

– Виявлення аномалій**.** Іноді завдання полягає в ідентифікації точок даних, які просто є незвичайними. Наприклад, при виявленні шахрайства із кредитною картою підозрілими є будь-які незвичайні операції оплати. Кількість можливих варіантів так велико, а кількість відомих прикладів так мало, що навчитися визначати шахрайські дії дуже важко. Підхід, що використовується при виявленні аномалій, полягає в тому, щоб просто вивчити, як виглядають нормальні дії (за допомогою журналу нормальних транзакцій), і визначати всі дії, які істотно відрізняються від нормальних.

Неконтрольоване

При неконтрольованому навчанні точкам даних не привласнюються мітки. Замість цього мета алгоритму неконтрольованого навчання – певне впорядкування даних або опис їхньої структури. Це може означати угруповання даних у кластери або пошук різних способів аналізу складних даних для їхнього спрощення або поліпшення їхньої організації.

Навчання з підкріпленням

У навчанні з підкріпленням алгоритм вибирає дія у відповідь на кожну точку даних. Алгоритм навчання також незабаром одержує сигнал, що сповіщає про успіх, що дає зрозуміти, наскільки вдало було ухвалене рішення. На основі цього алгоритм змінює свою стратегію для досягнення кращого результату. На даний момент у машинному навчанні Azure модулі алгоритмів навчання з підкріпленням відсутні. Навчання з підкріпленням широко поширено в робототехніці, де набір показань датчиків в один момент часу являє собою точку

КОПЗ – 2024

даних і алгоритму необхідно вибрати наступну дію робота. Крім того, воно природно підходить для додатків з Інтернету речей.

Рекомендації з вибору алгоритму

Точність

Одержання найбільш точної відповіді необхідно не завжди. Іноді досить наближеної відповіді залежно від того, для чого він використовується. У цьому випадку можна значно скоротити час обробки, дотримуючись більше наближених методів. Ще однією перевагою більше наближених методів є те, що вони природно прагнуть уникнути надмірно високої точності.

Час навчання

Кількість хвилин або годин, необхідних для навчання моделі, сильно відрізняється для різних алгоритмів. Час навчання часто тісно пов'язане з точністю – одне звичайно супроводжує іншому. Крім того, деякі алгоритми більше чутливі до кількості точок даних, чим інші. Коли час обмежений, це може вплинути на вибір алгоритму, особливо з більшим набором даних.

Лінійність

Лінійність використовується в багатьох алгоритмах машинного навчання. Алгоритми лінійної класифікації припускають, що класи можуть бути розділені прямою лінією (або її аналогом для більшого числа вимірів). До них ставляться логістична регресія й метод опорних векторів (у тому виді, у якому це реалізовано в машинному навчанні Azure). Алгоритми лінійної регресії припускають, що тренди даних випливають прямої лінії. Ці припущення припустимі для ряду завдань, але для інших завдань вони приводять до зниження точності.

Границя нелінійного класу – використання алгоритму лінійної класифікації приведе до низької точності.

Дані з нелінійним трендом – використання алгоритму лінійної класифікації приведе до появи набагато більшої кількості помилок, чим необхідно

КПЗ – 2024

Незважаючи на їхню небезпеку, лінійні алгоритми дуже популярні на першій лінії атаки. Звичайно вони алгоритмічно прості й швидко освоюються.

Кількість параметрів

Параметри є тими регуляторами, які фахівець із даних повертає при настроюванні алгоритму. Це числа, які впливають на поводження алгоритму, наприклад чутливість до помилок або кількість ітерацій, або на варіанти поводження алгоритму. Час навчання й точність алгоритму іноді можуть бути дуже чутливими до точності завдання параметрів. Як правило, алгоритми з більшим числом параметрів вимагають великої кількості проб і помилок, щоб визначити вдале сполучення параметрів.

Крім того, у машинному навчанні Azure є блок модулів коректування параметрів , що автоматично пробує всі сполучення параметрів з будь-якою обраною деталізацією. Хоча це відмінний спосіб переконатися в тім, що ви заповнили простір параметрів, зі збільшенням кількості параметрів експоненціально зростає час, необхідне для навчання моделі.

Плюсом є те, що наявність великої кількості параметрів звичайно означає, що алгоритм має більшу гнучкість. Це часто дозволяє домагатися дуже гарної точності. За умови, що ви можете знайти правильну комбінацію параметрів.

Кількість функцій

Для деяких типів даних кількість функцій може бути дуже більшим у порівнянні з кількістю точок даних. Це часто відбувається з генетичними або текстовими даними. Велика кількість функцій для деяких алгоритмів навчання може привести до того, що вони загрузнуть і час навчання стане неприпустимо більшим. Для цього варіанта особливо добре підходить метод опорних векторів.

Особливі випадки

Деякі алгоритми навчання роблять певні припущення про структуру даних або бажаних результатів. Якщо ви зможете знайти той алгоритм, що відповідає вашим потребам, з ним ви зможете одержати більше точні результати, більше точні прогнози й скоротити час навчання.

КПЗ–2024

Linear regression

Як згадувалося раніше, лінійна регресія зв'язує з набором даних лінію (або площина або гіперплощина). Це швидка й проста "робоча конячка", але вона може бути зайво простій для деяких завдань. Посібник з лінійної регресії можна знайти тут.

Дані з лінійним трендом

Логістична регресія

Незважаючи на слово, що вводить в оману, "регресія" у назві, логістична регресія насправді є потужним інструментом двокласовой і багатокласовой класифікації. Це швидкий і простий метод. Той факт, що в ньому замість прямої лінії використовується S-образна крива, дозволяє природно використовувати його для поділу даних на групи. Логістична регресія приводить до появи лінійних границь класів, тому при її використанні переконаєтеся, що вам комфортно з лінійною апроксимацією.

Логістична регресія для двокласових даних із усього однією функцією – границя класу є точкою, у якій логістична крива рівноудалена від обох класів.

Дерева, ліси й джунглі

Лісу рішень (регресійні, двокласові й багатокласові), джунглі рішень (двокласові й багатокласові) і дерева, що збільшуються, рішень (регресійні й двокласові) засновані на деревах рішень, базової концепції машинного навчання. Існує безліч варіантів дерев рішень, але всі вони роблять те саме: підрозділяють простір функцій на області з переважно однаковими мітками. Це можуть бути області з однаковою категорією або однаковим значенням залежно від того, чи проводиться класифікація або регресія.

Дерево рішень підрозділяє простір функцій на області приблизно с однаковими значеннями.

Так як простір функцій можна підрозділити на регіони довільного розміру, легко представити такий поділ, при якому в одному регіоні буде тільки одна

КПЗ_2024

точка даних – це крайній приклад надмірно високої точності. Щоб цього уникнути, більші набори дерев створюються з особливою математичною обережністю, щоб дерева не корелювали один з одним. Середнім для цього "лісу рішень" є дерево, що дозволяє уникнути надмірно високої точності. Лісу рішень можуть використовувати великий обсяг пам'яті. Джунглі рішень – це варіант, що використовують менший обсяг пам'яті за рахунок невеликого збільшення часу навчання.

У деревах, що збільшуються, рішень щоб уникнути надмірно високої точності обмежуються кількість повторних розподілів і мінімальна кількість точок даних у кожному регіоні. Алгоритм створює послідовність дерев, кожне з яких учиться компенсувати помилки, залишені попереднім деревом. У результаті ми одержуємо дуже точний механізм навчання, що, як правило, використовує великий обсяг пам'яті.

Квантильна регресія швидкого лісу є різновидом дерева рішень для особливого випадку, у якому ви хочете знати не тільки типове (середнє) значення даних у межах регіону, але його розподіл у вигляді квантилей.

Нейронні мережі й сприйняття

Нейронні мережі – це алгоритми навчання, натхненні пристроєм людського мозку, які охоплюють багатокласові, двокласові й регресійні завдання. Розмаїтість нейронних мереж дуже велико, але в машинному навчанні Azure всі вони мають форму спрямованого ациклічного графа. Це означає, що вхідні функції передаються вперед (і тільки вперед) по послідовності шарів, після чого перетворюються у вихідні дані. У кожному шарі вхідні функції зважуються в різних сполученнях, підсумуються й передаються на наступний рівень. Таке сполучення простих обчислень дає можливість як вивчати границі складних класів і тренди даних. Багатошарові мережі такого типу виконують "глибоке навчання", що так широко представлене в технічних звітах і науково-фантастичній літературі.

КПЗ_2024

Але така висока продуктивність має й зворотну сторону. На навчання нейронних мереж може йти тривалий час, особливо для більших наборів даних з більшою кількістю функцій. Вони також мають більше параметрів, чим більшість алгоритмів, що означає, що коректування параметрів значно подовжує час навчання. А для тих, хто хоче перевищити власні досягнення й визначити власну структуру мережі, можливості нейронних мереж невичерпні.

Границі, досліджувані нейронними мережами, можуть бути складними й нестандартними.

Двокласне усереднене сприйняття – відповідь нейронних мереж на величезне збільшення часу навчання. У ньому використовується структура мережі, що надає лінійні границі класу. Вона майже примітивна за сьогоднішніми стандартами, але має довгу історію надійної роботи й досить мала для швидкого вивчення.

Методи опорних векторів

Методи опорних векторів знаходять границю, що розділяє класи з як можна більшою шириною. Якщо два класи не можна чітко розділити, алгоритми знайдуть найкращу границю, що зможуть. Як записано в Машинному навчанні Azure, двокласові методи опорних векторів роблять це тільки для прямої лінії. (У термінах методів опорних векторів, вони використовують лінійне ядро.) Так як це виконується за допомогою лінійної апроксимації, час виконання досить мало. Ці методи дійсно проявляють себе з даними з більшою кількістю функцій, такими як текст або генетичні дані. У цих випадках методи опорних векторів дозволяють розділити класи швидше й з меншою надлишковою точністю, чим більшість інших алгоритмів; крім того, вони використовують невеликий обсяг пам'яті.

Типова границя класу для методу опорних векторів збільшує ширину границі, що розділяє два класи.

Інший продукт Microsoft Research, двокласовий локально глибокий метод опорних векторів , являє собою нелінійний варіант методу опорних векторів, що зберігає більшу частину швидкості й ефективного використання пам'яті лінійної

КПЗ_2024

версії методу. Він ідеально підходить для випадків, у яких лінійний підхід не дає досить точних результатів. Розроблювачі зберегли швидкість методу шляхом розбивки завдання на кілька малих завдань методу опорних векторів. Докладніше довідатися про те, як вони це зробили, можна в повному описі .

За допомогою ефективного розширення нелінійних методів опорних векторів однокласовий метод опорних векторів проводить границю, що щільно відокремлює весь набір даних. Це зручно для виявлення аномалій. Всі нові точки даних, які виходять далеко за цю границю, досить незвичайні, щоб звернути на них увага.

Методи Байєса

Методи Байєса мають одну дуже коштовну якість: вони не приводять до надмірного збільшення точності. Для цього вони попередньо роблять деякі припущення про ймовірний розподіл відповіді. Іншим побічним ефектом цього підходу є те, що в цих методів дуже мало параметрів. У машинному навчанні Azure є обидва алгоритми Байєса для класифікації (двокласна точкова машина Байєса) і регресії (лінійна регресія Байєса). Зверніть увагу, що в них передбачається, що дані можна розбити прямою лінією або зіставити їй.

З історичної точки зору точкові машини Байєса були розроблені в Microsoft Research.

Спеціалізовані алгоритми

При наявності дуже конкретної мети вам може повезти зі спеціалізованим алгоритмом. У колекції Машинного навчання Azure є алгоритми, які спеціалізуються на прогнозуванні ранжирування (порядкова регресія), кількісних прогнозах (регресія Пуассона) і виявленні аномалій (на основі аналізу основних компонентів і на основі методу опорних векторів), а також одиночний алгоритм кластеризації (K-середні).

Виявлення аномалій на основі аналізу первинних компонентів – більшість даних попадає в стереотипний розподіл; підозрілими вважаються точки, які значно відхиляються від цього розподілу

КПЗ-2024

Набір даних групується в 5 кластерів з використанням K-середніх.

Також існує багатокласова класифікація "один-всі", що розбиває проблему класифікації класу N на проблеми двокласової класифікації класу N-1. Точність, час навчання й властивості лінійності визначаються використовуваними двокласовими класифікаторами.

Два двокласових класифікатори поєднуються разом для одержання трикласового класифікатора.

Машинне навчання Azure також включає доступ до потужної платформи машинного навчання за назвою Vowpal Wabbit. Ця платформа зневажає наведеною тут класифікацією, так як може вирішувати як класифікаційні, так і регресійні завдання й навіть навчатися на основі частково нерозмічених даних. Її можна настроїти для використання будь-якого алгоритму навчання, будь-якої функції втрати й будь-якого алгоритму оптимізації. Ця система споконвічно розроблялася як ефективна, паралельна й дуже швидка. Вона обробляє величезні набори функцій з мінімальними зусиллями. Запущена й керована Джоном Ленгфордом з Microsoft Research, Vowpal Wabbit – це "формула один" серед інших алгоритмів. За допомогою Vowpal Wabbit можна вирішити не всі завдання, але якщо система підходить для вашого завдання, можливо, варто витратити час на вивчення інтерфейсу системи. Система також доступна у вигляді автономного відкритого вихідного коду кількома мовами.

КБПЗ-2024

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його

ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи машинного навчання у Big Data.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

КОПЗ_2024

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

КбПЗ_2024

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Алгоритми машинного навчання усе ширше використовуються для збільшення продуктивності гібридних систем зберігання даних. Класифікації, регресії й аналіз часових рядів допомагають вибрати тип сховища й упорядкувати в ньому розміщення файлів.

У число чотирьох головних експериментів, проведених на Великому адронному коллайдері в ЦЕРН, входить LHCb. Його детектори й системи моделювання фізичних процесів щорічно генерують 15 екзабайт сирих даних, які після обробки містяться в розподілену систему зберігання на жорстких дисках і магнітних стрічках. Зрозуміло, що диски швидше стрічок, але дорожче, тому важливо оптимізувати розміщення файлів на різних носіях. Система керування дисковою пам'яттю для LHCb заснована на статистичному аналізі історії звертання до даних, що поряд з алгоритмами машинного навчання дає можливість прогнозувати популярність файлів, що дозволяє визначити, які файли можна видалити з диска. А застосовуючи алгоритми регресійного аналізу й аналізу часових рядів, вдається визначити оптимальне число копій файлів на диску для забезпечення їхнього надійного зберігання, економії дискового простору й зменшення часу доступу до даних.

Алгоритми машинного навчання вже давно застосовуються для поліпшення продуктивності алгоритмів роботи з кеш-пам'яттю, наприклад в [1] для прогнозування популярності даних використовувалися ланцюги Маркова, а опираючись на цей прогноз, удалося домогтися збільшення продуктивності гібридної (SSD + HDD) системи зберігання. В [2] прогноз популярності файлів виконувався за допомогою нейронної мережі й дозволив визначити оптимальне число копій файлів.

КПЗ-2024

Особливість системи керування дисковою пам'яттю для LHCb полягає в тому, що історія звертань до файлів сильно розріджена – часовий ряд, що відбиває факти звертань до дисків, має багато нульових значень. Більшість файлів використовуються рідко, але інтенсивно, тому такі методи, як штучні нейронні мережі [3], ARMA, ARIMA [4], демонструють погані результати через сильну розрідженість даних. Обчислення додаткових ознак для кожного файлу, що характеризують особливості їхньої історії звертань, дозволяє розділити файли на популярні й непопулярні.

Вхідними даними для алгоритму керування дисковою пам'яттю для LHCb є історія звертань до файлів і метадані, до яких ставляться: джерело даних, конфігурація детектора, тип файлу, тип даних (отримані по методу Монте Карло або реальні дані), тип події, час створення файлу, час першого/останнього обігу, розмір однієї копії, загальний розмір файлу на диску, число копій на диску й ін. Наприклад, можна використовувати історію звертання до файлів за два роки з розбивкою по тижнях і представити її у вигляді часового ряду з 104 точок, кожна з яких – це число звертань до файлу за один тиждень.

Для прогнозу популярності файлів використовується класифікатор – алгоритм машинного навчання із учителем, у результаті роботи якого кожний файл позначається як популярний або непопулярний. Як ми вже відзначали, часові ряди історії звертань до файлів сильно розріджені, тому показники за останні півроку (26 тижнів) історії звертань служать для розмітки даних – якщо файл протягом цього періоду не використовувався, то він позначається як непопулярний і одержує мітку «1», у противному випадку позначається як «0».

Метадані використовуються як вхідні ознаки для класифікатора, однак для уточнення алгоритму були обчислені нові ознаки, які використовувалися разом з існуючими. Ці нові ознаки описують форму часового ряду для історії звертань до файлу – наприклад, якщо останні півроку історії звертань використовуються для розмітки файлів, то для обчислення нових ознак беруться тільки перші 78 тижнів. Такий вибір часових інтервалів обумовлений тим, що дані зберігаються й

КПЗ – 2024

використовуються роками, а між двома послідовними звертаннями до файлу можуть пройти тижні й місяці.

Один з ознак – кількість тижнів, протягом яких були звертання до файлу, іншої – число тижнів з тих пор, як до файлу зверталися востаннє. Також обчислювалися максимальне значення, середнє значення й стандартне відхилення числа тижнів між послідовними тижнями з ненульовим числом звертань і їхні відносини. Ще одна ознака – центр мас часового ряду файлу, у якому маса вказує на число звертань до файлу за кожній тиждень, а координата – на номер тижня. Всі ці ознаки істотно підняли якість навчання класифікатора.

У ролі класифікатора використовувався градієнтний бустинг над деревами (алгоритм машинного навчання, заснований на застосуванні ансамблю дерев рішень), що дозволяє без перенавчання швидко одержати високу якість класифікації. Для навчання використовувався метод перехресної перевірки з 10 частинами (k-fold cross-validation). Це означає, що всі файли розбиваються на 10 частин, при цьому класифікатор навчається на дев'яти частинах даних, а потім використовується для пророкування ймовірності одержання мітки «1» для десятої частини даних. Процедура повторюється 10 разів. У підсумку одержуємо прогноз імовірності мітки «1» для кожної з 10 частин. Передвіщена ймовірність перетвориться в популярність так, щоб популярність для класу з міткою «1» була розподілена рівномірно. Чим ближче популярність до 1, тим вище ймовірність того, що файл не буде використаний у майбутньому. Тобто обчислена популярність визначає антипопулярність файлу. Такий вибір зроблений для того, щоб файли, що є першими кандидатами на видалення, мали більшу величину метрики.

Популярність даних, рівна 1, відбиває ймовірність того факту, що файл буде марний у майбутньому, а другою важливою характеристикою є інтенсивність звертання до файлу. Існує безліч алгоритмів аналізу часових рядів для пророкування їхніх майбутніх значень, але знову, з огляду на, що більшість часових рядів сильно розріджено, складні параметричні методи, такі як

КПЗ-2024

поліноміальна регресія, авторегресія, ARMA, ARIMA, штучні нейронні мережі й інші, не підходять. Тому для прогнозу інтенсивностей звертань до файлів використовувалися дві непараметричні моделі.

Спочатку часові ряди згладжувалися по формулі ядерного згладжування Надарая-Ватсона на основі RBF-ядра (один з найпростіших видів непараметричної регресії), а ширина вікна згладжування була взята в 30 тижнів.

Найпростіший спосіб пророчити майбутнє значення часового ряду – це взяти в його якості значення з останньої точки спостереження (статична модель прогнозу в роботі).

Знаючи популярність і передвіщене значення інтенсивності звертань до файлів, можна визначити, які файли повинні зберігатися на жорстких дисках і скільки копій вони повинні мати. При цьому небажано видаляти з диска файли, які будуть затребувані в майбутньому. Більше того, має сенс зберігати найбільш затребувані файли з більшим числом копій, щоб зменшити середній час доступу до даних. Наявність додаткових копій дозволяє забезпечити декільком користувачам швидкий доступ до даних зі збереженням пропускної здатності, так як копії можуть бути розміщені фізично близько до місць, де в них бідують.

Для оптимізації розподілу даних використовувалася функція втрат, що дозволяє оцінити вартість зберігання всіх даних на дисках і стрічках, а також вартість відновлення даних з магнітних стрічок у випадку помилки.

Оптимальне число копій файлів визначається по передвіщеній інтенсивності звертань до файлів – чим вище інтенсивність звертань, тим більше копій має конкретний файл.

Якщо антипопулярність файлу вище граничного значення, то файл віддаляється з диска. Оптимізація функції втрат полягає в тому, щоб знайти такі граничні значення популярності даних і оптимальні значення копій файлів, які дають мінімум функції втрат.

Можна зрівняти запропонований алгоритм роботи рекомендаційної системи з алгоритмом Least Recently Used (LRU), що дивиться на останні

КПЗ – 2024

спостереження в історії звертань до файлу й ухвалює рішення щодо видалення файлів з диска.

«Відношення часу доступу» – це відношення часу доступу (завантаження) до файлів, отримане після застосування алгоритму, до первісного часу доступу. Колонка «Число помилок» показує, скільки файлів було вилучено з жорсткого диска, але потім знадобилося в роботі [2]. «Альфа» – коефіцієнт пропорційності між числом копій файлу й квадратним коренем інтенсивності звертання до нього. Як видно з таблиці, обидва алгоритми дозволяють оптимізувати обсяг дискового простору, однак LRU допускає більше помилок.

Як показує досвід, для багатьох сховищ характерно неоптимальне розміщення даних по різних рівнях. Дійсно, важко заздалегідь передбачити, як будуть використовуватися файли, у результаті затребувані дані можуть виявитися, наприклад, на стрічці. Запропонований алгоритм керування дисковою пам'яттю в гібридній системі зберігання, побудований на основі машинного навчання, допускає невелике число помилкових видалень файлів, дозволяє домогтися економії дискового простору й знизити середній час доступу до даних.

3.2 Розробка структурної схеми

На жаль або на щастя, про самостійну творчість машин у незапрограмованій заздалегідь ситуації мова не йде. Система виявляється марною, якщо наявних даних недостатньо або зафіксована подія не співвідноситься з якими-небудь попередніми аналогами. До того ж універсальних алгоритмів не існує – кожний з них вирішує одне певне завдання.

Розумні пристрої зберігання

Однієї з перспективних областей для застосування машинного навчання є зберігання даних. Перераховуючи потенційні напрямки нового технологічного прориву в сфері СЗД, експерти називають технології RAIN, SSD, NVMe, хмарні інфраструктури й програмно обумовлене зберігання (SDS), але навіть не

КПЗ_2024

згадують машинне навчання в СЗД. Імовірно, це пов'язане з тим, що відповідні методи ще мало вивчені й не до кінця освоєні лідерами ринку.

Поширення флеш-накопичувачів і розвиток програмно обумовленого зберігання – важливі фактори, але вони аж ніяк не новина. На відміну від названих технологій, розумні пристрої зберігання багато в чому залишаються концепцією майбутнього, тої самої Next Big Thing, що так чекають експерти.

Що ж будуть робити розумні пристрої? Вони покликані не тільки зберігати інформацію, але й виконувати інтелектуальні аналітичні дії, забезпечуючи роботу мікросервісів і еластичну масштабованість. В ідеалі такі пристрої повинні мати здатність міняти свої ролі залежно від контексту й при необхідності поєднуватися для виконання спільного завдання.

Розумні пристрої зможуть «розуміти» поводження додатків на основі специфічних факторів, визначати рівень вимог до доступності й надійності систем зберігання й автоматично підбудовуватися під необхідний клас рішень. Пристрою нового типу допоможуть вирішити цілий ряд проблем, що виникають у світі розумних міст, підприємств і просторів.

Які основні сценарії використання машинного навчання вже зараз і розумних пристроїв зберігання в майбутньому? Зупинимося на них більш докладно.

Сценарії застосування

Перший і один з головних – предіктивна аналітика на основі аналізу поводження СЗД. Відповідний алгоритм покликаний здійснювати аналіз системних журналів і історії подій і попереджати про можливі проблеми с кластером зберігання. Рано або пізно буде досить простого голосового запиту до системи підприємства, щоб моментально одержати дані про стан інфраструктури, проаналізувати поточне навантаження, виділити й розподілити потрібні для співробітників ресурси.

Не виключено, що при збереженні поточних темпів розвитку мобільних технологій і автоматизації через п'ять-сім років засобу керування

КПЗ-2024

інфраструктурою навчаться розуміти природна мова. Уже сьогодні доступні аналоги Microsoft IFTTT, які адекватно реагують на завдання «Транслювати всі мої повідомлення з Twitter в Facebook». Якщо говорити про конкретні приклади, то на шляху до розумного центра обробки даних істотно просунулася компанія Nimble Storage із продуктом InfoSight.

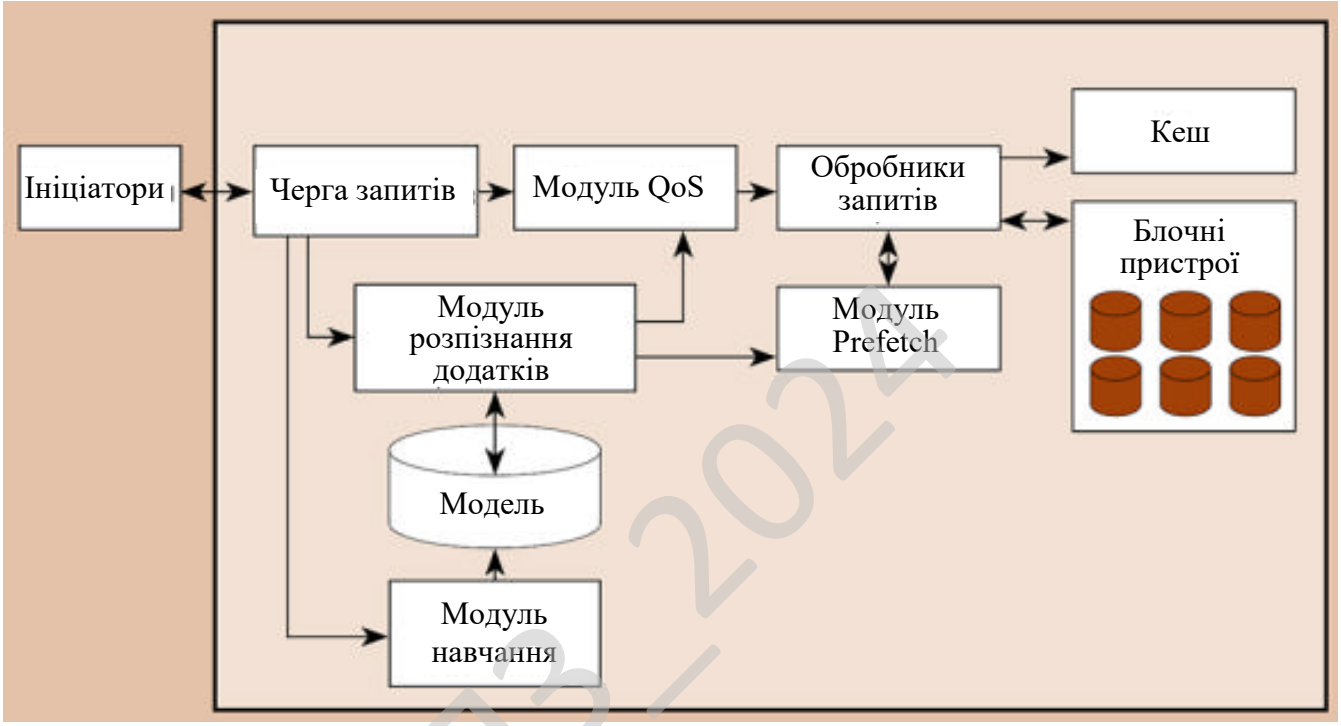

Рисунок 3.1 – Структурна схема системи

КОПЗ 2024

Другий важливий сценарій – зміна параметрів зберігання на лету з модифікацією настроювань СЗД у цілому й окремих компонентах зокрема. Багато хто вендори з різним успіхом розробляли свого роду «автопілот», що на основі аналізу наявних даних і поведінкових характеристик міг би автоматично змінювати параметри системи зберігання, наприклад, вибираючи оптимальні настроювання для переважного типу навантаження.

Полем для експериментів по реалізації такого алгоритму стали флеш-накопичувачі. Через специфіку NAND Flash, при розробці рішень для SSD в одному такому рішенні неможливо сполучити великий обсяг, високу надійність і

максимальне число перезаписів, через що доводиться йти на певні компроміси. Настроювання пристрою і його поводження можна змінити шляхом завдання параметрів регістрів (регістрів може бути більше 1000 в 3D NAND і до 100 у планарній пам'яті). Удалий приклад машинного навчання для автоматичної модифікації параметрів різних регістрів – рішення технологічного стартапа NVMdurance.

Третій приклад застосування МН в СЗД – забезпечення якості обслуговування (QoS) на рівні додатків. Усе більше розроблювачів доходять висновку, що традиційний підхід «один додаток для одного LUN» не оптимальний і вимагає перегляду. Дійсно, нерідкі випадки, коли з одним томом з одного хосту взаємодіють багато додатків, різні по функції й навантаженню. Левова частина цих додатків не критична для діяльності підприємства, тобто ресурси системи витрачаються нераціонально.

Як не заблудитися в «випадковому лісі»

Для рішення цієї проблеми були реалізовані проекти, у рамках яких розпізнавання додатків і їхнього навантаження здійснюється по характерних операціях уведення-виводу (IO). У сфері Data Mining є кілька алгоритмів, придатних для подібного завдання ідентифікації – наприклад, Random Forest, або «випадковий ліс». Цей метод заснований на побудові великого числа (ансамблю) «дерев» рішень (їхня кількість є параметром методу). У порівнянні з іншими аналогічними алгоритмами Random Forest має наступні переваги:

– висока швидкість навчання;

– неітеративне навчання (алгоритм завершується за фіксоване число операцій);

– масштабованість (здатність обробляти більші обсяги даних);

– висока якість одержуваних моделей (порівнянне з нейронними мережами й ансамблями нейронних мереж);

– мала кількість параметрів, що налаштовуються.

КПЗ - 2024

Random Forest – імовірнісний алгоритм. Коли до системи зберігання звертається невідомий додаток, він оцінює ймовірність збігу даного додатка з раніше виявленими в системі. При цьому в алгоритму є й недоліки. Один з них – схильність до перенавчання на зашумлених даних. Втім, ця проблема вирішується за допомогою застосування спеціалізованих фільтрів на зразок FCBF.

Реалізацію функції QoS зроблено в модулю QoSmic. Принцип його функціонування досить простий. Всі запити до СЗД проходять через модуль розпізнавання додатків. Модуль працює у двох режимах:

– Навчання: система зберігання вивчає характеристики нового додатка, що планується розпізнавати з метою забезпечення QoS або читання, що попереджає.

– Розпізнавання: додатки ідентифікуються в режимі реального часу, а інформація про їх використовується модулями QoS і Prefetch.

Протягом певного періоду часу (наприклад, 20 с) запити накопичуються в модулі розпізнавання, далі журнал аналізується й на підставі зібраних відомостей будуються сигнатури вводу-виводу. У режимі навчання сигнатури позначаються ім'ям додатка, а в режимі розпізнавання передаються у відповідний модуль для ідентифікації.

Для ідентифікації додатка досить знати чотири характеристики: довжину запиту, тип запиту (читання або запис), зсув (адресний простір), час надходження запиту. На підставі цих параметрів і будуються сигнатури I/O.

Споконвічно при ідентифікації додатків акцент робився на пошуку відомих «паттернів» (послідовності довжин запитів, що йдуть один за іншим). Даний підхід відмінно зарекомендував себе раніше у випадку порівняльних тестів (benchmark), ПЗ для резервного копіювання й антивірусів. Тестувались різні алгоритми машинних навчань, спочатку вдавалося ідентифікувати додатка з низькою ймовірністю, але з додаванням у сигнатуру уведення-виводу спеціальних атрибутів точність ідентифікації вдалося довести до 99,9%.

КПЗ-2024

У деяких системах головними критеріями для ідентифікації є адресний простір. У випадку QoSmic передбачається, що з одним простором можуть працювати кілька додатків; таким чином, розпізнавання по місцю розташування не здійснюється. Ключовим же критерієм для ідентифікації виявився розподіл довжин запитів, що надходять від конкретного додатка.

Розроблений метод ідентифікації має високу точність і швидкістю, що дозволяє використовувати його для автоматичного завдання пріоритету критично важливим додаткам і гарантувати їм необхідну пропускну здатність поза залежністю від навантаження з боку інших працюючих додатків. Високий рівень точності досягається завдяки певним атрибутам, тобто параметрам у сигнатурі уведення-виводу, що дозволяє формувати досить точний статистичний профіль різних додатків, з високою точністю виявляти працююче й відрізняти його від низькопріоритетного.

Безумовно, адміністратор повинен брати до уваги необхідність періодичного перенавчання системи. Особливо цим не варто зневажати, якщо часто видається повідомлення «не вдалося визначити» або додатки визначаються неправильно. У процесі навчання сам алгоритм може підказати, що отриманих сигнатур недостатньо для точної ідентифікації.

Від навчання до роботи

Автоматичне визначення працюючого додатка на ініціаторі дозволяє застосовувати технологію QoSmic у будь-якій системі зберігання, де є модуль QoS, відповідальний за різні рівні якості обслуговування. Сам же спосіб забезпечення QoS може бути реалізований по-різному. Наприклад, у рішеннях компаній Fujitsu або Oracle рівень QoS визначається адміністратором вручну. Ці СЗД створюють профілі на основі співвідношення операцій читання-запису й типів навантаження, при цьому розпізнавання конкретних додатків не відбувається.

Схожа ідея застосування машинного навчання викладена в патенті US8762583 компанії EMC. Вона припускає використання нейронної мережі, для

КБПЗ-2024

того щоб визначити оптимальний спосіб обробки вхідних запитів вводу-виводу. Даний підхід може бути використаний і в модулі QoS для балансування навантаження на систему.

Таким чином, методи машинного навчання можуть із успіхом застосовуватися в СЗД, не викликаючи додаткових затримок і втрат продуктивності. У результаті вузьким місцем у системі зберігання виявляється не обчислювальний модуль, а швидкість читання із самих дисків, тобто обмеження встаткування.

Завершуючи огляд прикладів впровадження машинного навчання в системах зберігання даних, відзначимо, що роль «науки про даний» і самих фахівців з аналізу даних (data scientist) в адмініструванні IT-інфраструктури може згодом істотно зменшитися. Можливо, у майбутньому не залишиться адміністраторів як таких, оскільки через подальший розвиток алгоритмів машинного навчання й штучного інтелекту залученість людини в керування інфраструктурою буде зведена до мінімуму.

3.3 Розробка функціональної схеми

В основі рекомендацій для забезпечення необхідної якості обслуговування потоків лежить оптимальний вибір параметрів для алгоритмів керування чергами WFQ і WRED. Дослідимо можливості механізмів WRED і WFQ і вплив конфігурації цих алгоритмів на якість послуг. Для підбора параметрів використовуються результати експериментальних вимірів у тестовому каналі й у програмному забезпеченні, що розроблено в результаті виконання магістерської роботи. Для рішення завдання аналізувалися різні моделі доставки пакетів у рамках протоколів ICMP, UDP і TCP. Для кожного потоку задавався набір параметрів: T_1 T_2, p_c, w_q, λ/μ. Для кожного субпотоку задавався відсоток доступної смуги пропущення (настроювання WFQ). В експериментах будувалися залежності різних параметрів QoS для системи машинного навчання у Big Data

КПЗ - 2024

(втрат, затримок RTT) при зміні одного з параметрів WFQ або WRED. Після виміру й порівняння всіх можливих випадків робиться вивід про оптимальне настроювання параметрів WRED і WFQ.

Опишемо два механізми віддаленої діагностики локальних мереж. Проблема забезпечення якості обслуговування багатогранна, і в ранніх дослідженнях вивчалося питання про виявлення проблемних ділянок мережі, у яких відбуваються втрати пакетів, досліджувався механізм нагромадження втрат пакетів. У рамках даного дослідження було розроблено два алгоритми віддаленої діагностики локальних мереж. Ці алгоритми, випробувані на практиці [5,6,7], вони дозволяють знаходити в мережі проблемні ділянки, які можуть сильно впливати на якість передачі даних через ці ділянки.

Перший з розглянутих нами алгоритмів дозволяє за допомогою однієї робочої станції, у реальному масштабі часу, цілодобово оцінювати роботу віддалених мережних сегментів. Як параметр, що характеризує якість роботи сегмента, використовується відсоток пакетів, загублених у сегменті.

Другий розглянутий алгоритм ґрунтується на залежності ймовірності втрати пакета від довжини пакета. Його можна вважати корисним діагностичним засобом при дослідженні властивостей каналів і виявленні частки втрат, сполучених з перекручуваннями переданих кадрів. Ця методика дозволяє виявити ділянки мережі або каналу, що містять джерела наведень і перекручувань. Корисним може бути порівняння залежності ймовірності втрати пакета від його довжини, отриманої для того самого каналу в різні моменти часу.

Тестова ЕОМ перебуває в сегменті, підключеному до мережного пристрою N1. На ЕОМ установлена програма, що послідовно посилає ICMP запити до підмножин A, B, . . . машин із сегментів N1, N2, ..., Nn. Т же тестова ЕОМ обробляє ICMP відгуки, що прийшли із сегментів. ICMP відгуки мають ту ж довжину, що й ICMP запити. Уважається, що ймовірності втрати в проміжному пристрої для запиту й відгуку під час експерименту не міняються й рівні між собою.

КПЗ-2024

Якщо на вхід каналу в N1 надійшло *S* пакетів, спрямованих у субмережу n, то з N1 в N2 надійде $S_*(1-\alpha_1)$, де α_1 – імовірність втрати пакета в активному пристрої N1.

За аналогією одержимо вираження для кількості пакетів, що дійшли до сегмента n і повернулися назад відправникові – тестової ЕОМ:

$$R = S\cdot(1-\alpha_1)^2\cdot(1-\alpha_2)^2\cdot(1-\alpha_{n-1})^2\cdot(1-\alpha_n),$$

де α_n – процент загублених пакетів в віддаленному сегменті n.

Ступінь 2 виникає через те, що пакети можуть губитися в мережних пристроях як по шляху туди, так і назад. При цьому передбачається, що ймовірності втрати пакета на шляху туди й назад рівні. При відправленні відгуку з машини сегмента *n* втрата неможлива, із цієї причини співмножник $(1-\alpha_n)$ представлений у першому ступені, На практиці виміряються величини $(1-\alpha_1)^2$, $(1-\alpha_2)^2$,…...

З бази даних береться інформація про приналежність тієї або іншої IP – адреси ЕОМ до сегмента, що представляє інтерес, і виробляється зондування всіх машин субмережі. Кожна ЕОМ зондується послідовно серіями по три пакета. Якщо всі три пакети не дали відгуку, машина вважається виключеною, і ці дані не враховуються при оцінці ймовірності втрати пакета. Такий підхід приводить до певного заниження ймовірності втрат, але ефект незначний, тому що α_3 навіть при ймовірності втрати $\alpha = 0{,}1$ становить лише 0,1%, і його впливом можна зневажити. При вимірах не відбувається завищення ймовірності втрат пакетів за рахунок відключених у цей момент ЕОМ.

Розглянутий алгоритм дозволяє віддалено виміряти ймовірність втрати пакетів для мережного сегмента, якщо відомий список IP – адрес цього сегмента й у кожному з мережних пристроїв по шляху є інтерфейс, де можна оцінити втрати в цьому конкретному мережному пристрої. Така ситуація звичайно легко

КПЗ_2024

реалізується усередині локальної мережі, а також в опорних мережах сервіс провайдерів.

Другий алгоритм дозволяє виявити причини втрати пакетів у транспортному мережному каналі. Причинами втрати можуть бути переповнення буферів у мережних пристроях по шляху проходження й ушкодження пакетів при транспортуванні.

Якщо втрати сполучені з перекручуваннями пакетів при транспортуванні, то залежність імовірності втрат P_L від довжини L пакета повинна характеризуватися залежністю $P_L = L(1 - p(1-p)^L)$, де p – імовірність ушкодження одного біта (BER – Bit Error Rate), L – кількість біт у пакеті, і має розмірність штук.

З обліком того, що BER лежить у діапазоні $10^{-3} - 10^{-10}$, а L>512 біт, при малих L має місце наближена формула $P_L = (1 - p(1 - L \cdot p + 0{,}5 * p^3 L(L-1)))$.

Таким чином, можна приблизно очікувати квадратичну залежність імовірності втрат від довжини пакета для малих L. Цю обставину можна використовувати для аналізу властивостей каналів і причин виміру ймовірності втрат.

Вимірюючи залежність імовірності втрат від довжини пакета й, апроксимуючи її методом найменших квадратів, можна ідентифікувати причини втрат. Були проведені дослідження втрат пакетів для двох маршрутів. Були проведені виміри ймовірності втрати для довжин пакетів 64, 128, 256, 512, 1024 і 1450 байт. Для кожної з довжин було послано по 2000 ICMP – пакетів.

У загальному виді має місце твердження: $C_{path} = P_L + F(L)$, де C_{path} – повна ймовірність втрати пакета на всьому маршруті, L – число біт у пакеті.

Апроксимація поліномом другого ступеня по методу найменших квадратів дає наступну залежність імовірності втрати пакета від його довжини: $C_{path} = (14{,}33 \pm 6) + (0.298 \pm 0.04) \cdot L - \cdot (0.00012 \pm 3 \cdot 10^{-5}) L^2$

КОПЗ_2024

Для другого випадку залежність повної ймовірності втрати пакета від його довжини може бути наближено наступною формулою: $C_{path} = (25.56 \pm 4.42) + (0.00167 \pm 0.01) \cdot L - (9.46 \cdot 10^{-7} \pm 1.23 \cdot 10^{-5}) \cdot L^2$

Значення ймовірності втрати пакетів для пакетів довжини, що цікавить, можна одержати, поділивши число загублених пакетів на загальне число відправлених пакетів (для кожної довжини відправлялося 2000 пакетів).

Порівняння результатів показує, що помітний вплив на втрати робить ушкодження пакетів у шляху. Також імовірність втрати визначається переповненням буфера, ємність якого задається в сегментах.

Звідси можна зробити вивід, що дослідження залежності ймовірності втрати від довжини пакета можна вважати корисним діагностичним засобом при дослідженні властивостей каналів і виявленні втрат, сполучених з перекручуваннями переданих кадрів. Ця методика дозволяє виявити ділянки мережі або каналу, що містять джерела наведень і перекручувань. Корисним може бути порівняння залежностей імовірності втрати пакета від його довжини, отриманих для того самого каналу в різний час.

Опишемо результати експериментів по забезпеченню QoS для системи машинного навчання у Big Data у середовищі моделювання програмного забезпечення, що розроблено в результаті виконання магістерської роботи. Ціль даного моделювання – розробка додаткових засобів виміру й аналізу параметрів QoS для системи машинного навчання у Big Data. Ми також хотіли досліджувати можливість застосування модельних методів оцінки параметрів якості обслуговування:

1. Проведено дослідження алгоритмів RIO-C і WRED для різних моделей призначення порогів. Був також досліджений вплив алгоритмів – диспетчерів на надавану якість обслуговування. Установлено оптимальні границі роботи алгоритмів WRED і RIO, при яких використання цих алгоритмів є найбільш ефективним.

2. До вбудованих в програмному забезпеченні, що розроблено в результаті виконання магістерської роботи, функцій були додані утиліти для розрахунку й

КПЗ-2024

відображення значень середньозважених черг і ймовірності відкидання пакетів. Це було необхідно для вивчення й інтерпретації поводження нашої тестової мережі.

3. Отримано залежності довжин черг від часу, довжин середньозважених черг від часу й імовірності відкидання пакетів від часу. Виконаний також аналіз цих залежностей і вироблені рекомендації з установки параметрів алгоритмів WRED і RIO для досягнення оптимальної роботи (для моделі).

4. Зроблено підбір параметрів моделі віртуальної мережі таким чином, щоб вона адекватно відбивала роботу тестової мережі. Було зроблено повномасштабне моделювання для широкого набору параметрів, були отримані умови, при яких модель і експеримент на реальній мережі поводяться подібним чином.

Моделювання показало, що при некоректному виборі параметрів, наприклад фактора усереднення можна одержати досить широкі варіації довжини черги, які у свою чергу можуть у кілька разів збільшити дисперсію RTT. Завданням моделювання було виявлення області параметрів керування чергою, при яких осцилляції довжини черги мінімальні, а усереднення прийнятно. Показане моделювання продемонструвало наслідку неоптимального підбора параметрів і наші можливості спостереження за цими параметрами в середовищі моделювання NS – 2.

Усереднення довжини черги є важливим компонентом алгоритму керування процесом буферізації. Без усереднення процес буферізації був би підданий сильному впливу випадкових флуктуацій вхідного потоку пакетів. Але саме усереднення є причиною виникнення осцилляції довжини черги. Адже залежність ухвалення рішення про відкидання того або іншого пакета визначається значенням усередненої довжини черги, що може сильно відрізнятися від поточного значення довжини черги. Амплітуда варіації поточного значення довжини черги істотно більше усередненого. Розрахунки показують, що при певних параметрах поточна довжина черги може досягати в

КПІ_2024

максимумі повного обсягу буфера, а в мінімумі нуля (тобто буфер уже порожній, а відкидання пакетів триває).

Звідси видно, що усереднені значення довжини черги на початковій ділянці залежності уступають поточній довжині більш, ніж у два рази. У розрахунках вхідний потік λ і вихідний μ задавалися в бітах у секунду. В області від 0 до T_1 ріст довжини черги визначається добутком $(\lambda - \mu)t$. Після досягнення рівня T_1 швидкість росту довжини черги вповільнюється, тому що частина пакетів відкидається, залежність стає квадратичною. Припинення росту й початок спаду Q відбувається в момент, коли Q досягає рівня T_2. Розрахунки проводилися із залученням пакета програм NS – 2. Значення T_1 і T_2 задавалися в пакетах.

Оптимальний вибір параметрів алгоритму WRED дозволяє збільшити ефективність використання буферів маршрутизатора й, як наслідок, підняти пропускну здатність або поліпшити рівень QoS для системи машинного навчання у Big Data. Подібним чином проводилися варіації всіх параметрів WRED.

З отриманих даних був зроблений вивід, що прийнятний набір параметрів з погляду осцилляції довжини черги відповідає: p_c<0,4; 1,2<λ/μ<1,5, і w_q >0,003.

Функціональна схема розробленої системи зображена на рисунку 3.2. Існує не занадто багато способів забезпечення QoS для системи машинного навчання у Big Data. Найпростіший з них – збільшення смуги пропущення мережі за рахунок нарощування апаратних можливостей устаткування. Можна використовувати й такі прийоми, як завдання пріоритетів даних, організація черг, запобігання перевантажень і формування трафіку. Керування мережею за заданими правилами в перспективі повинне об'єднати всі ці способи в єдину автоматизовану систему, що буде гарантувати якість послуг абсолютно на всіх ділянках мережі.

Збільшення апаратної потужності, безсумнівно, є найбільш ефективним засобом реалізації QoS для системи машинного навчання у Big Data. Тиск із боку конкурентів, необхідність підвищення ефективності виробництва, поява нових технологій, що дозволяють оснащувати спеціалізовані мікросхеми (ASIC) найрізноманітнішими функціями, – все це змушує постачальників комутаційного

КОПЗ_2024

встаткування для локальних мереж викидати на ринок усе більше швидкодіючі пристрої за цінами, порівнянним з вартістю моделей колишнього покоління.

Малоймовірно, що в доступному для огляду майбутньому даний підхід до підтримки QoS для системи машинного навчання у Big Data перестане бути пріоритетним. Оскільки у Big Data вдається забезпечити гарантовану якість послуг, не прибігаючи до дорогої модернізації усього встаткування й серйозних змін у системі керування мережею, мережні адміністратори будуть звертати увагу й на програмні засоби, що дозволяють реалізувати QoS для системи машинного навчання у Big Data.

Отже, найбільше поширення, швидше за все, одержить комбінований підхід. Деякі виробники висловлюються на його користь, затверджуючи, що найкраще збільшувати пропускну здатність мережі не прямо, а за рахунок інтелектуальних можливостей устаткування, що має засоби забезпечення QoS для системи машинного навчання у Big Data. Правда, виробники мережних пристроїв навряд чи можуть бути об'єктивними в цьому питанні, тому що вони зацікавлені в збуті тих самих продуктів, які підтримують гарантовану якість послуг.

У глобальних мережах нарощування апаратних потужностей використовується рідше. Звичайно, зниження вартості смуги пропущення зробило б передачу даних по глобальних мережах доступною для більше широкого кола користувачів (і навіть трохи знизило б актуальність впровадження гарантованої якості послуг). Але в найближчому майбутньому вартість смуги пропущення в глобальних мережах буде залишатися досить високою, тому й нарощування апаратної потужності не стане настільки популярним, як у локальних мережах.

З рисунку видно, що розроблена система складається з наступних функціональних частин:

– Блок інтерфейсу користувача.

– Блок визначення параметрів QoS для системи машинного навчання у Big Data.

КПЗ – 2024

– Блок примусового завдання параметрів QoS для системи машинного навчання у Big Data.

– Блок моделювання завантаженості мережі для системи машинного навчання у Big Data.

– Блок моніторингу мережі.

– Блок дослідження можливостей механізмів WRED.

– Блок дослідження можливостей механізмів WFQ.

– Блок призначення пріоритетів.

– Блок організації та обслуговування черг.

– Блок управління навантаженням.

– Блок формування трафіка.

Розглянемо ці блоки більш детально.

Блок інтерфейсу користувача

Призначений для реалізації взаємодії користувача, або дослідника з системою.

Блок визначення параметрів QoS для системи машинного навчання у Big Data

Призначений для визначення існуючих параметрів якості обслуговування (QoS для системи машинного навчання у Big Data). До них відносяться:

– Bandwidth (BW) – смуга пропущення, описує номінальну пропускну здатність середовища передачі інформації, визначає ширину каналу. Виміряється в bit/s (bps), kbit/s (kbps), mbit/s (mbps).

– Delay – затримка при передачі пакета.

– Jitter – коливання (варіація) затримки при передачі пакетів.

– Packet Loss – втрати пакетів. Визначає кількість пакетів, що відкидаються мережею під час передачі.

Блок примусового завдання параметрів QoS для системи машинного навчання у Big Data

Призначений для примусового завдання одного, або декількох параметрів якості обслуговування (QoS для системи машинного навчання у Big Data). До них

КБПЗ_2024

відносяться:

– Bandwidth (BW) – смуга пропущення, описує номінальну пропускну здатність середовища передачі інформації, визначає ширину каналу. Виміряється в bit/s (bps), kbit/s (kbps), mbit/s (mbps).

– Delay – затримка при передачі пакета.

– Jitter – коливання (варіація) затримки при передачі пакетів.

– Packet Loss – втрати пакетів. Визначає кількість пакетів, що відкидаються мережею під час передачі.

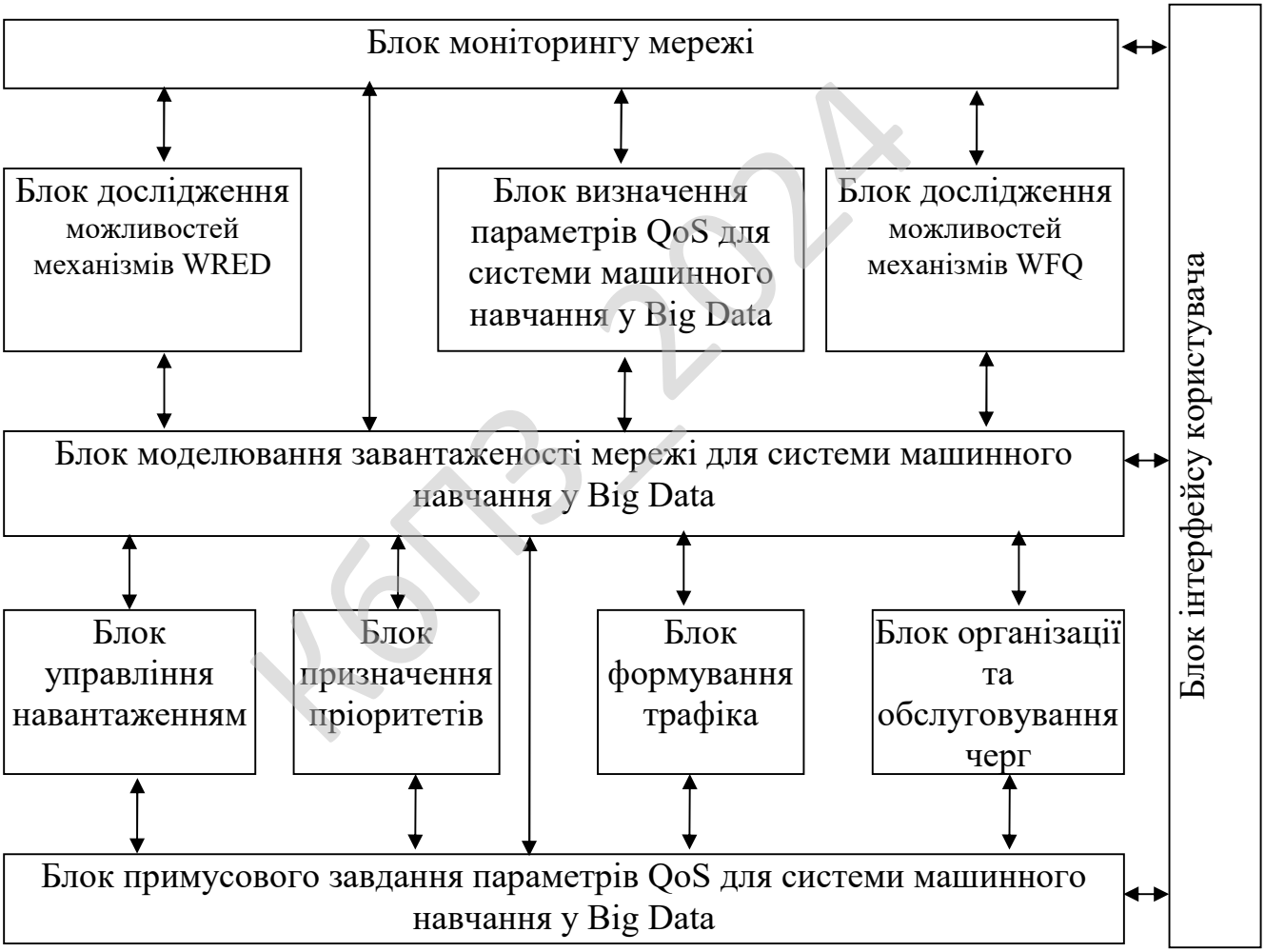

Рисунок 3.2 – Функціональна схема системи

КБПЗ-2024

Блок моделювання завантаженості мережі для системи машинного навчання у Big Data

Призначений для моделювання завантаженості мережі з визначеним трафіком та заданими параметрами якості обслуговування (QoS для системи машинного навчання у Big Data).

Блок моніторингу мережі

Призначений для аналізу поточного стану мережі.

Блок дослідження можливостей механізмів WRED

Одним з методів QoS для системи машинного навчання у Big Data, призначених для забезпечення необхідних вимог до різних потоків даних – запобігання перевантажень (congestion avoidance). Він заснований на обмеженні розмір черги, сигналізуючи джерелам даних про необхідність зменшити швидкість передачі інформації (WRED – Weighted random early detection).

Блок дослідження можливостей механізмів WFQ

Другим з методів QoS для системи машинного навчання у Big Data, призначених для забезпечення необхідних вимог до різних потоків даних – керування перевантаженням (congestion management). Він заснований на присвоєнні квот і пріоритетів потокам, і у випадку перевантаження, потоки одержують якість, обмежену їхньою квотою й пріоритетом (WFQ – Weighted Fair Queuing).

Блок призначення пріоритетів

Нарівні з нарощуванням апаратного забезпечення мережі для реалізації QoS для системи машинного навчання у Big Data застосовуються й засобу типу завдання пріоритетів даних і організації черг. Маршрутизатори підтримують ці механізми протягом багатьох років, як і деякі з нових комутаторів для каналів Gigabit Ethernet.

Способи пріоритезації даних можна умовно підрозділити на явні й неявні.

При неявному призначенні пріоритетів маршрутизатор або комутатор автоматично привласнює послугам відповідні рівні, виходячи із заданих

КбПЗ-2024

адміністратором мережі критеріїв (наприклад, типу додатка для застосовуваного протоколу передачі або адреси джерела). Кожний вхідний пакет аналізується (фільтрується) на відповідність цим критеріям. Механізм неявної пріоритезації підтримують практично всі маршрутизатори.

Деякі комутатори теж здатні задавати пріоритети, але мають обмежений набір функцій. Так, комутатори можуть забезпечувати пріоритезацію даних по типу віртуальної локальної мережі, адресі джерела або адресата, але не використовують інформацію більш високого рівня (протокол передачі або тип додатка). Розроблювальні в цей час системи керування мережею за заданими правилами дозволять реалізувати більше зроблені схеми пріоритезації даних при роботі з такими комутаторами.

При явній пріоритезації даних користувач або додаток запитує певний рівень служби, а комутатор або маршрутизатор намагається задовольнити запит. Імовірно, самим популярним механізмом явної пріоритезації стане протокол IP Precedence (протокол старшинства), що одержав другу назву IP TOS (IP Type Of Service), – один з розділів четвертої версії протоколу IP.

IP TOS резервує спеціальне поле в заголовку пакета, де можуть бути зазначені ознаки QoS для системи машинного навчання у Big Data, що визначають час затримки, швидкість пропущення й рівень надійності передачі пакета. Однак найдеться небагато популярних додатків – за винятком мультимедійного ПЗ, – у які реалізована підтримка протоколу IP TOS.

Зараз розробляється протокол резервування ресурсів RSVP, що передбачає більше складний, чим в IP TOS, механізм передачі від додатка до машрутизатору запиту на гарантовану якість послуг. Як і IP TOS, протокол RSVP поки не одержав широкої підтримки розроблювачів – він реалізований лише в окремих типах маршрутизаторів. Поширення RSVP стримується через те, що не вирішені деякі питання, пов'язані із сумісністю різних мереж. До того ж застосування RSVP значно збільшує навантаження на маршрутизатори й може привести до зниження швидкодії цих пристроїв.

КПЗ-2024

Видимо, у доступному для огляду майбутньому неявна пріоритезація, не потребуюча серйозних обчислювальних потужностей маршрутизатора, залишиться більше популярною, чим явна. Крім того, при явному завданні пріоритетів значно ускладнюється керування мережею. Кінцеві користувачі, швидше за все, будуть набудовувати своє програмне забезпечення на запит найвищого з можливих рівнів послуг. Відповідно, адміністраторові мережі прийдеться розробляти правила керування користувачами й, можливо, навіть побудовувати служби з гарантованою якістю для кожного користувача окремо.

Блок організації та обслуговування черг

Після того як переданим по мережі даним призначені відповідні пріоритети (за допомогою явних або неявних методів), потрібно визначити порядок передачі цих даних, задавши алгоритм обслуговування черг із необхідною якістю (рівнем QoS для системи машинного навчання у Big Data). По суті, черги являють собою області пам'яті комутатора або маршрутизатора, у яких групуються пакети з однаковими пріоритетами передачі. Алгоритм обслуговування черги визначає порядок, у якому відбувається передача пакетів, що зберігаються в ній. Зміст застосування всіх алгоритмів зводиться до того, щоб забезпечити найкраще обслуговування трафіку з більш високим пріоритетом за умови, що й пакету з низьким пріоритетом гарантується відповідна увага.

При використанні способів завдання явних і неявних пріоритетів алгоритм обробки черг визначає порядок їхнього обслуговування. Відповідно до цього алгоритму на кожні два пакети, переданих у мережу із черги 1 (з високим пріоритетом) доводиться по одному пакету із черг 2 і 3. Пакети з однаковими пріоритетами передаються за принципом FIFO (”першим прийшов – першим вийшов”).

Якщо в мережі виникає перевантаження, служба черг не гарантує своєчасного досягнення пункту призначення найбільш важливими даними. Гарантується лише те, що ці пакети будуть передані раніше, ніж ті, що мають більш низький пріоритет.

КОПЗ-2024

Сучасні служби QoS для системи машинного навчання у Big Data вирішують таке завдання за рахунок резервування смуги пропущення. Кожній із черг (або їхніх груп) виділяється заздалегідь задана величина смуги пропущення, що гарантує певну смугу пропущення для черги з більш високим пріоритетом. Для критичних ситуацій, коли обсяг даних у черзі перевищує розміри смуги пропущення, в алгоритмах обслуговування звичайно передбачається передача трафіку з високим пріоритетом на смугу пропущення, "приналежну" чергам з низьким пріоритетом, і навпаки. Найпростіші алгоритми обслуговують кожну чергу за принципом FIFO. При цьому передача кадрів великого розміру, що мають високий пріоритет, може приводити до затримок трафіку іншого додатка з настільки ж високим пріоритетом, але меншим обсягом.

У більше складних алгоритмах уживає спроба "справедливої" обробки черг. Наприклад, алгоритм рівномірного пропорційного (або зваженого) обслуговування (WFQ – Weighted Fair Queuing), розроблений компанією Cisco, підрозділяє додатки на потребуючі великої й малої ширини смуги пропущення, а сама смуга пропущення розподіляється між всіма додатками нарівно. Слід зазначити, що основні виробники маршрутизаторів самі розробляють алгоритми обслуговування черг і використовують для їхнього опису власну термінологію.

Істотним недоліком сучасних маршрутизаторів і комутаторів є те, що вони підтримують мале число черг. Найчастіше виробники організують служби QoS для системи машинного навчання у Big Data, що використовують чотири черги, хоча чим більше черг, тим більше різних пріоритетів можна привласнити переданим пакетам і тим "справедливіше" розподілити смугу пропущення між додатками. Наприклад, адміністратор у стані задати пріоритети таким чином, щоб перевага при передачі віддавалося пакетам, адресованим на більше віддалені вузли.

Блок управління навантаженням

Служба QoS для системи машинного навчання у Big Data дає можливість використовувати для керування мережею два важливих механізми – керування в

КПЗ_2024

умовах перевантаження й запобігання перевантажень. Перший з них дозволяє кінцевій станції відразу знижувати швидкість передачі даних, коли в мережі починається втрата пакетів. У протоколах TCP/IP і SNA цей механізм підтримується вже протягом декількох років. І хоча сам по собі він не гарантує якості передачі, при його використанні разом з механізмом запобігання перевантажень результати виявляються набагато кращими. У мережах TCP/IP механізм запобігання перевантажень застосовується досить давно, але лише в останні роки він стає стандартом "де-факто" для маршрутизаторів телекомунікаційних мереж і Internet.

Стандартним способом запобігання перевантажень у мережі стало застосування механізму випадкового виділення пакетів (Random Early Detection, RED). При заповненні черг вище певної критичної оцінки цей механізм змушує маршрутизатор вибирати із черги за випадковим законом деякі пакети й "втрачати" їх. Швидкість передачі даних станціями-відправниками знижується, що й дозволяє уникнути переповнення черги.

Механізм пропорційного випадкового виділення пакетів – WRED (Weighted RED) – можна вважати наступною, більше зробленою "версією" RED. Він передбачає, що вибір пакетів, які повинні "втратитися", буде відбуватися з обліком їх пріоритезації згідно IP TOS.

Блок формування трафіка

Формування трафіку – це загальний термін, яким прийнято позначати різні способи маніпулювання даними для підвищення якості їхньої передачі. Один їхній таких способів – сегментація пакетів. У мережах ATM гарантовано високий рівень QoS для системи машинного навчання у Big Data досягається в тому числі й за рахунок малого розміру переданих пакетів (осередків – у термінології ATM). Максимальний час затримки при передачі будь-якого пакета мережі ATM – це час передачі одного осередку.

Запозичаючи корисні механізми технології ATM, виробники маршрутизаторів і комутаторів починають забезпечувати у своїх продуктах можливість сегментації пакетів. Деякі пристрої, призначені для мереж frame relay,

КПІ_2024

сегментують пакети, передані по каналах глобальних мереж, щоб гарантувати конкретний час передачі й мінімізувати затримки.

Ще один спосіб формування трафіку – його “вирівнювання”. Для таких протоколів, як наприклад, AppleTalk, характерна нерівномірна передача пакетів, що часом приводить до появи в мережі послідовностей або ланцюжків пакетів, а отже – до її перевантаження. Процедура вирівнювання трафіку дозволяє розчленувати ланцюжки шляхом розміщення пакетів у буфері перед їхньою передачею в мережу. Для забезпечення більше рівномірної передачі даних можна також вирівнювати трафік кінецевих вузлів мережі.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється. Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

КПЗ-2024

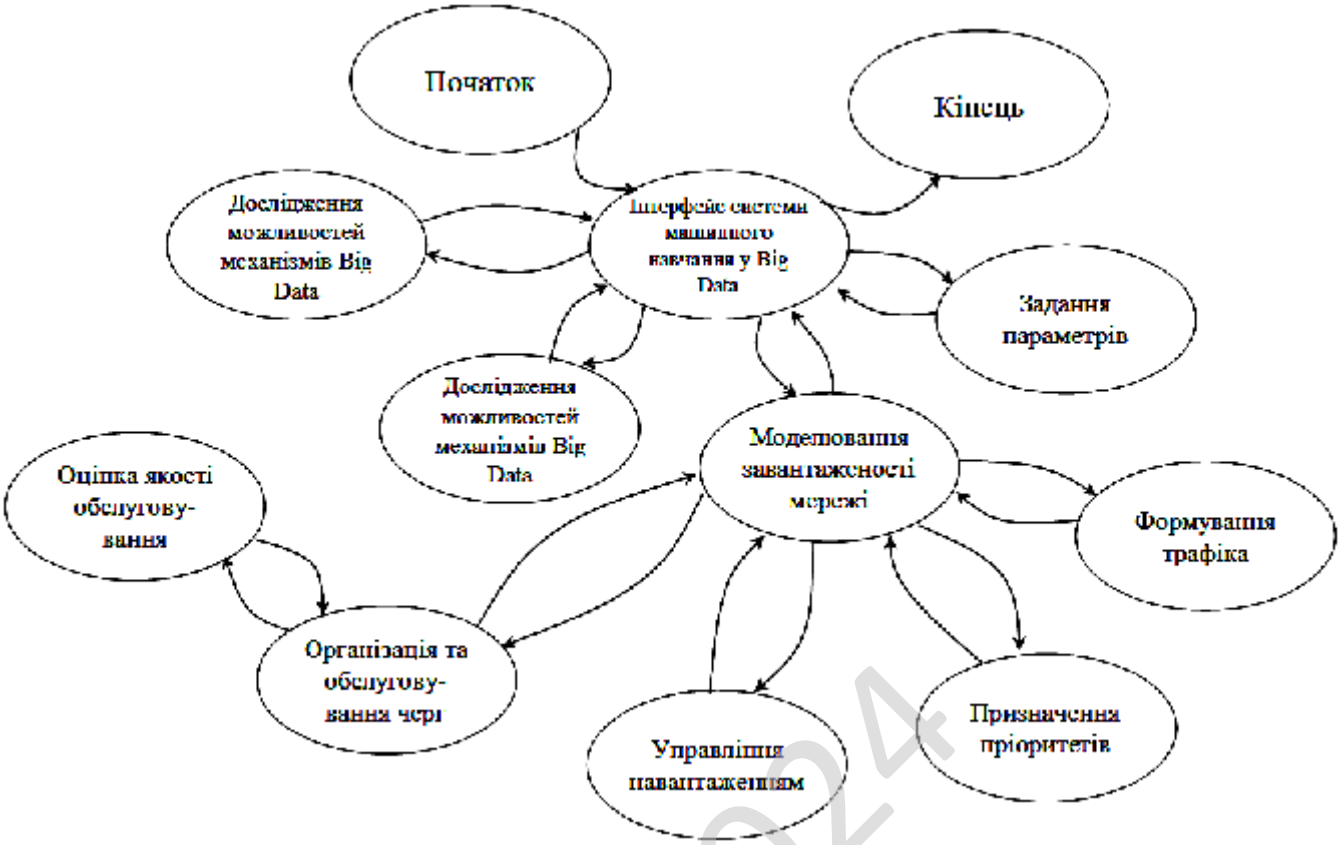

Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

КОПЗ_2024

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схеми є основою ПЗ. Тому від точності і детальності проробки блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації, також те, що при розробці програми слід надати особливу увагу модулю машинного навчання у Big Data.

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні блоки можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірки поточного стану та поверненням на початок схеми чи з завершенням роботи розробленого ПЗ.

КбПЗ_2024

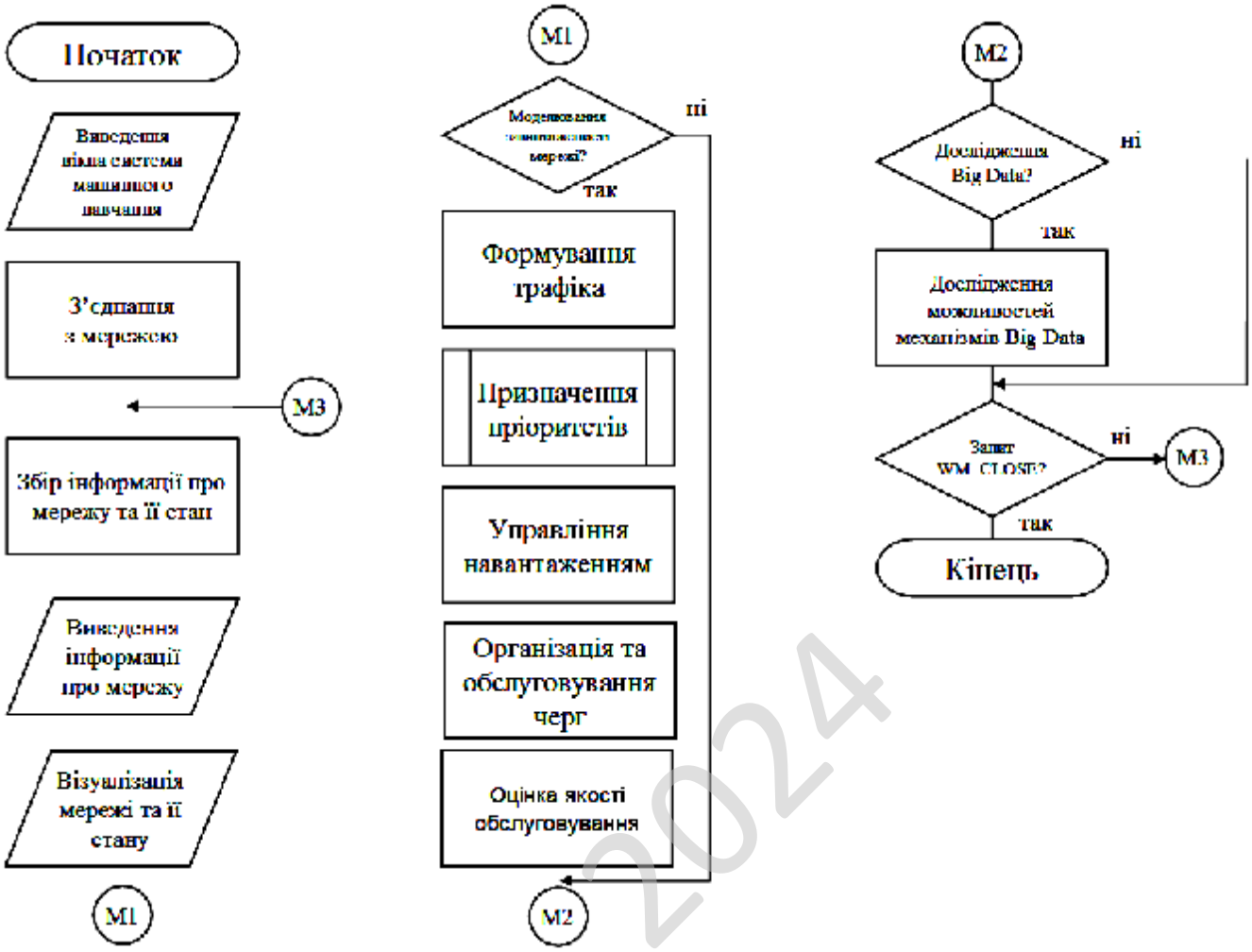

Рисунок 4.1 – Блок-схема основної програми

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

КПЗ_2024

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація

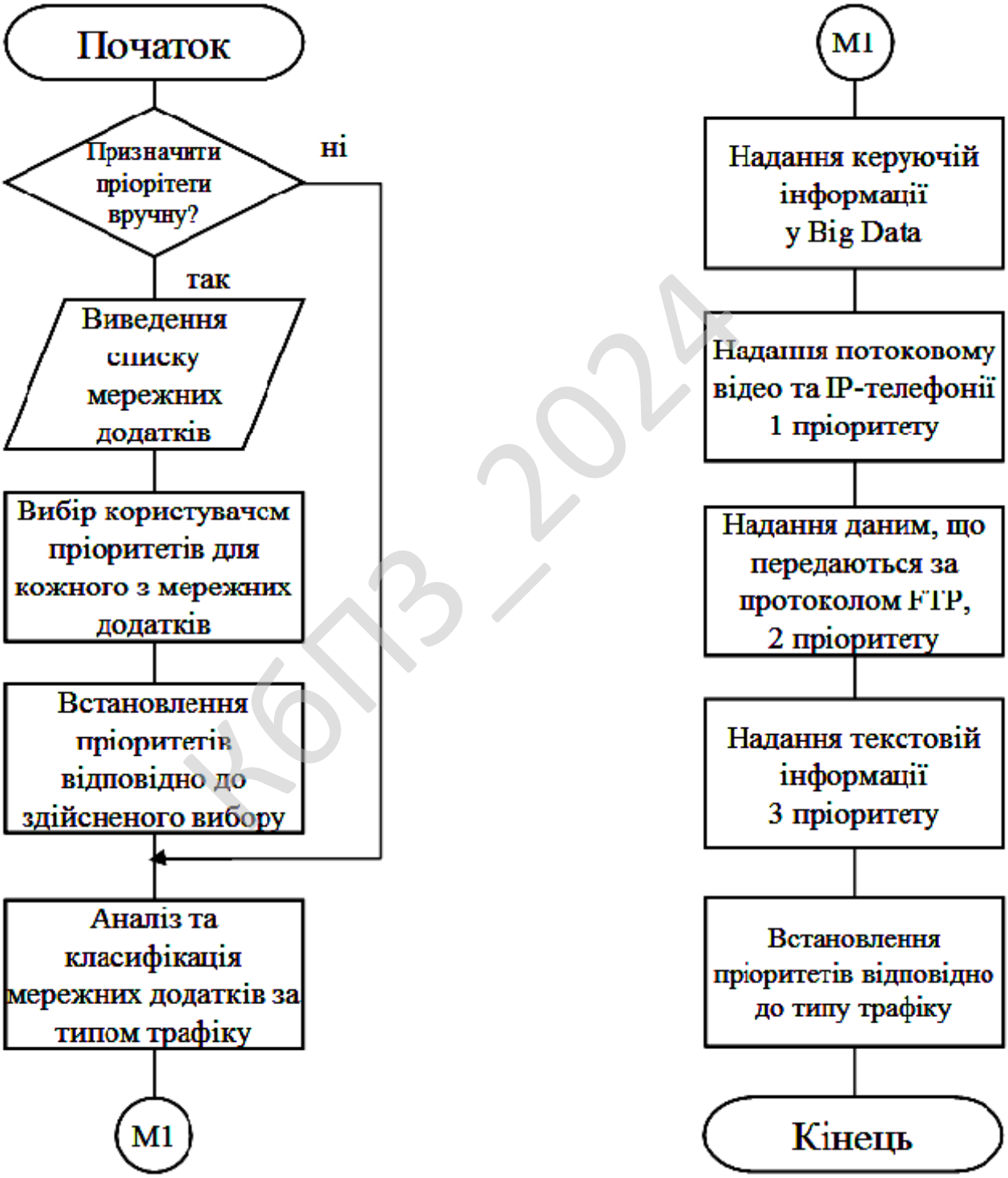

Рисунок 4.2 – Блок-схема роботи підпрограми

КБПЗ_2024

Розглянемо використані технології та їх основні компоненти що підтверджують правильність використаних проектних рішень.

Peer-to-peer (рівний до рівного) – варіант архітектури системи, в основі якої стоїть мережа рівноправних вузлів.

Комп'ютерні мережі типу peer-to-peer (або P2P) засновані на принципі рівноправності учасників і характеризуються тим, що їх елементи можуть зв'язуватися між собою, на відміну від традиційної архітектури, коли лише окрема категорія учасників, яка називається серверами може надавати певні сервіси іншим.

Фраза «peer-to-peer» була вперше використана у 1984 році Парбауелом Йохнухуйтсманом (Parbawell Yohnuhuitsman) при розробці архітектури Advanced Peer to Peer Networking фірми IBM.

В чистій «peer-to-peer» мережі не існує поняття клієнтів або серверів, лише рівні вузли, які одночасно функціонують як клієнти та сервери по відношенню до інших вузлів мережі. Ця модель мережевої взаємодії відрізняється від клієнт-серверної архітектури, в якій зв'язок відбувається лише між клієнтами та центральним сервером.

Така організація дозволяє зберігати працездатність мережі при будь-якій конфігурації доступних її учасників. Проте практикується використання P2P мереж які все ж таки мають сервери, але їх роль полягає вже не у наданні сервісів, а у підтримці інформації з приводу сервісів, що надаються клієнтами мережі.

В P2P системі автономні вузли взаємодіють з іншими автономними вузлами. Вузли є автономними в тому сенсі, що не існує загальної влади, яка може контролювати їх. В результаті автономії вузлів, вони не можуть довіряти один одному та покладатися на поведінку інших вузлів, тому проблеми масштабування та надмірності стають важливішими ніж у випадку традиційної архітектури.

Сучасні P2P-мережі набули розвитку завдяки ідеям, пов'язаними з обміном інформацією, які формувалися у руслі того, кожен вузол може надавати та

КОПЗ_2024

отримувати ресурси які надаються будь-якими іншими учасниками. У випадку мережі Napster, це був обмін музикою, в інших випадках це може бути надання процесорного часу для пошуку інопланетних цивілізацій (SETI@home) або ліків від раку (Folding@home).

Переваги P2P

Розподіл/зменшення вартості. Централізовані системи, які обслуговують багато клієнтів, зазвичай складають більшість вартості системи. Коли, ця вартість стає дуже великою, архітектура P2P може допомогти розподілити вартість серед користувачів. Наприклад, серед систем файлообміну Napster дозволив розподілити вартість зберігання файлів і міг підтримувати індекс, потрібний для сумісного використання. Економія коштів, здійснюється за допомогою використання та об'єднання ресурсів, які в іншому випадку не використовуються (наприклад SETI@home). Оскільки вузли зазвичай є автономними, важливо розподіляти витрати справедливо.

Об'єднання ресурсів. Децентралізований підхід веде до об'єднання ресурсів. Кожен вузол в системі P2P приносить певні ресурси як наприклад обчислювальна потужність або пам'ять. У програмах, які потребують величезну кількість цих ресурсів, як наприклад intensive моделювання або розподілені файлові системи, природно використовувати P2P, щоб залучити ці ресурси. Розподілені обчислювальні системи, як наприклад SETI@Home, distributed.net, і Endeavours – очевидні приклади цього підходу. Об'єднуючи ресурси тисяч вузлів, вони можуть виконувати важкі з точки зору кількості обчислень функції. Файлобмінні системи, як наприклад Napster, Gnutella, і так далі, також об'єднують ресурси. У цих випадках, це дисковий простір, щоб зберігати дані, та пропускна спроможність, щоб їх передавати.

Вдосконалена масштабованість/надійність. З відсутністю сильної центральної влади по відношенню до автономних вузлів, важливою метою є покращення масштабованості і надійності.

КПЗ_2024

Масштабованість і надійність визначаються в традиційному для розподілених систем сенсі, як наприклад використання пропускної спроможності – скільки вузлів можуть бути досягнуті від одного вузла, скільки вузлів може підтримуватися, скільки користувачів може підтримуватися. Розподілена природа peer-to-peer мереж також збільшує помилкостійкість у разі невдач, шляхом дублювання даних поміж багатьох вузлів, і – в чистих системах P2P – надаючи можливість вузлу знайти дані без залежності від єдиного централізованого індексного сервера. У останньому випадку, немає ніякої єдиної критичної точки в системі.

Збільшена автономія. У багатьох випадках, користувачі розподіленої системи не бажають залежати від будь-якого централізованого постачальника послуг. Натомість, вони воліють, щоб всі дані та призначена для них робота виконувалась локально. Системи P2P підтримують цей рівень автономії, тому що вони вимагають, щоб кожен вузол робив необхідну для нього частину праці.

Анонімність/конфіденційність. Пов'язаним із автономією є поняття анонімності і конфіденційності. Користувач, можливо, не хоче, щоб кого-небудь або будь-який постачальник послуг знав про нього або про його роль у системі. З центральним сервером, гарантувати анонімність важко, тому що сервер зазвичай зможе ідентифікувати клієнта, як мінімум через його адресу в Інтернет. Використовуючи структуру P2P, в якій дії виконуються локально, користувачі можуть уникати необхідності передавати будь-яку інформацію про себе до кого-небудь іншого.

FreeNet – яскравий приклад того, як анонімність може вбудуватися в додаток P2P. Він пересилає повідомлення через інші вузли, щоб забезпечити неможливість вистежування початкового автора. Це збільшує анонімність, використовуючи ймовірнісні алгоритми таким чином, щоб походження не можливо було легко відстежити аналізуючи трафік у мережі.

Динамічність. Системи P2P припускають, що оточення надзвичайно динамічне.

КПЗ – 2024

Тобто, ресурси, як наприклад вузли, з'являються та зникають із системи безперервно. У випадках комунікації, як наприклад мережі для обміну повідомленнями, використовуються так званий «список контактів», щоб інформувати користувачів, коли їхні друзі стають доступними. Без цього, потрібно було би, щоб користувачі «опитували» партнерів, посилаючи періодичні повідомлення. У випадку розподілених обчислень, як наприклад distributed.net і SETI@home, система повинна пристосуватись до заміни учасників. Тому вони повинні повторно видавати завдання для обчислення іншим учасникам, щоб гарантувати, що робота не втрачена, якщо попередні учасники відпадають від мережі, поки вони виконували крок обчислення.

Класифікація P2P систем

За функціями:

1. Розподілені обчислення. Обчислювальна проблема розподіляються на невеликі незалежні частини. Обробка кожної з частин робиться на індивідуальному ПК і результати збираються на центральному сервері. Цей центральний сервер відповідальний за розподілення елементів роботи серед окремих комп'ютерів в Інтернеті. Кожен із зареєстрованих користувачів має клієнтське програмне забезпечення. Воно користується періодами бездіяльності в ПК (часто це характеризується часами активації скрінсейверів), щоб виконувати деяке обчислення, надане сервером. Після того, як обчислення закінчене, результат посилається назад до сервера, і нова робота передається для клієнта.

2. Файлообмін. Зберігання та обмін даними – це одна з областей, де технологія P2P була найуспішнішою. Мультимедійні дані, наприклад, вимагають великих файлів. Napster і Gnutella використовувались користувачами, щоб обійти обмеження пропускної спроможності, які роблять передачу великих файлів неприйнятними.

3. Співпраця. Природа технології P2P робить її добре придатною для забезпечення співпраці між користувачами. Це може бути обмін повідомленнями, онлайн ігри, сумісна робота над документами в бізнесі, освіті та дома.

КПЗ-2024

За ступенем централізації:

1. Чисті peer-to-peer системи. Вузли є рівними, поєднуючи ролі серверу та клієнту. Не існує центрального сервера, що керує мережею. Прикладами таких систем є Gnutella та Freenet

2. Гібридні peer-to-peer системи. Мають центральний сервер, що зберігає інформацію про вузли та відповідає на запити відносно цієї інформації. Вузли займаються забезпеченням ресурсами (тому що центральний сервер їх не має), повідомленням сервера про наявність цих ресурсів надання ресурсів іншим вузлам, які бажають ними скористатися.

В залежності від того, як вузли з'єднуються один з одним можна поділити мережі на структуровані та неструктуровані:

1. Неструктурована мережа P2P формується, коли з'єднання встановлюються довільно. Такі мережі можуть бути легко сконструйовані, оскільки новий вузол, який хоче приєднатися до мережі, може скопіювати існуючі з'єднання іншого вузла, а вже потім почати формувати свої власні. У неструктурованій мережі P2P, якщо вузол бажає знайти певні дані в мережі, запит доведеться передати майже через всю мережу, щоб охопити так багато вузлів, як можливо. Головним недоліком таких мереж є те, що запити, можливо, не завжди вирішуються. Скоріш за все популярні дані будуть доступні в багатьох вузлів та пошук швидко знайде потрібне, але якщо вузол шукає рідкісні дані, наявні лише в декількох інших вузлів, то надзвичайно малоймовірно, що пошук буде успішним. Оскільки немає ніякої кореляції між вузлами та даними, що вони зберігають, немає ніякої гарантії, що запит знайде вузол, який має бажані дані.

2. Структурована мережа P2P використовує єдиний алгоритм, щоб гарантувати, що будь-який вузол може ефективно передати запит іншому вузлу, який має бажаний файл, навіть якщо файл надзвичайно рідкісний. Така гарантія потребує структуровану систему з'єднань. У наш час найпопулярнішим типом структурованої мережі P2P є розподілені хеш-таблиці, в яких хешування використовується для встановлення зв'язку між даними та конкретним вузлом,

КПЗ-2024

який за них відповідає.

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка

КбПЗ_2024

робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

КПЗ-2024

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан.

У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок.

Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

КПЗ-2024

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок.

Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

КПЗ-2024

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму UMAC (код автентифікації повідомлення на основі універсального гешування) – один з видів коду автентичності повідомлень (MAC).

Швидка «універсальна» функція використовується, для того, щоб гешувати вхідне повідомлення M у короткий рядок. До цього рядка потім застосовується функція XOR із псевдовипадковим значенням, у результаті чого ми одержуємо тег UMAC:

$$Tag=H_{K1}(M)\oplus F_{K2}(Nonce)$$

де K1 і K2 – секретні випадкові ключі, які мають одержувач і відправник.

Звідси видне, що безпека UMAC залежить від того, яким випадковим способом відправник і одержувач вибрали таємну геш-функцію й псевдовипадкову послідовність. При цьому значення Nonce міняється кожний такт. Через використання Nonce, приймач і передавач повинні знати час відправлення повідомлення й принцип створення значення Nonce. Замість цього можна використовувати в якості Nonce будь-яке інше неповторюване значення, наприклад порядковий номер повідомлення. При цьому даний номер не зобов'язано бути секретним, головне щоб він не повторювався.

UMAC розрахований на використання 32-х, 64-х, 92-х, і 128-бітових тегів, залежно від необхідного рівня безпеки. UMAC звичайно використовується разом з алгоритмом шифрування AES.

Функція створення ключа й псевдовипадкової послідовності

Створення псевдовипадкових байтів необхідно для роботи UHASH і при створенні тегів

КОПЗ-2024

Вибір блокового шифру

Для своєї роботи UMAC використовує блоковий шифр, вибір якого визначають наступні константи:

– BLOCLEN – довжина, у байтах, блоку з яким працює блоковий шифр.

– KEYLEN – довжина, у байтах, ключа блокового шифру.

При цьому використовується функція

– ENCIPHER(K,P) – зашифрувати рядок P з BLOCLEN байтів, використовуючи ключ K.

Приклад: якщо використовується AES з 16-байтним ключем, то BLOCLEN буде рівним 16(тому що AES працює з 16-байтними блоками).

KDF – функція створення ключа

Ця функція генерує послідовність псевдовипадкових байтів, використовуваних для ключових геш-функцій.

Вхід:

– K – рядок довжиною KEYLEN байт. / / Ключ блокового шифру.

– Index – ненегативне ціле число менше, чим 2 ^ 64.

– Numbytes – ненегативне ціле число менше, чим 2 ^ 64.

Вихід:

– Y – рядок довжини numbytes байт.

PDF: функція створення псевдовипадкового числа

Ця функція ухвалює ключ і даний час і повертає псевдовипадкове число для використання його в тегу покоління. За допомогою цієї функції можуть бути отримані числа довжиною 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– Nonce – рядок довжиною від 1 до BLOCKLEN байт.

– Taglen – ціле число 4, 8, 12 або 16.

Вихід:

– Y – послідовність байтів довжини taglen.

КПЗ 2024

Генерація UMAC-тегів

Генерація UMAC-тегів відбувається за допомогою UHASH функції при використанні Nonce значенні й отриманої до цього рядка. Їхня довжина може бути 4, 8, 12 або 16 байт.

Вхід:

– K – рядок довжиною KEYLEN байт.

– M – рядок довжиною менше 267 біт.

– Nonce – випадкове число від 1 до BLOCKLEN байт.

– Taglen – ціле 4, 8, 12 або 16.

Вихід:

– Тег, послідовність байтів довжиною taglen.

Алгоритм обчислення тегів:

Hashedmassage = UHASH(K, M, Taglen)

Pad = PDF(K, nonce, Taglen)

Tag = Pad xor Hashedmassage

UMAC-32 UMAC-64 UMAC-96 UMAC-128

Дані позначення містять у своїй назві певне значення довжини тегу:

– UMAC-32 (K, M, Nonce) = UMAC (K, M, Nonce, 4).

– UMAC-64 (K, M, Nonce) = UMAC (K, M, Nonce, 8).

– UMAC-96 (K, M, Nonce) = UMAC (K, M, Nonce, 12).

– UMAC-128 (K, M, Nonce) = UMAC (K, M, Nonce, 16).

Універсальна функція гешування(UHASH)

UHASH – універсальна функція гешування, серцевина алгоритму UMAC. UHASH – функція працює в три етапи. Спочатку до вхідного повідомлення застосовується L1-HASH, потім до цього результату застосовується L2-HASH і, нарешті, до результату застосовується L3-HASH . Якщо при цьому довжина вхідного повідомлення не більш 1024 біт, то L2-HASH не використовується. Тому що функція L3-hash повертає тільки слово довжини 4 байта, те якщо

КБПЗ–2024

потрібно одержати геш довжини більше 4 байт, здійснюється кілька ітерацій даної трирівневої схеми.

Універсальна функція

Нехай функція гешування вибирається із класу геш-функцій H, які відображають повідомлення в D, набір усіляких образів повідомлення. Цей клас називається універсальним, якщо для яких-небудь окремих пар повідомлень, існує на безлічі H/D функцій, функція, яка відображає їх в елемент D. Зміст цієї функції в тому, що якщо третя сторона прагне замінити одне повідомлення іншим, але при цьому вважає, що геш-функція була обрана абсолютно випадково, те ймовірність не виявлення підміни стороною, що ухвалює, прагне до 1/D.

L1-hash – перший етап

L1-hash розбиває повідомлення на шматки з 1024 байт і до кожного шматка застосовує алгоритм гешування називаний NH. Вихідний результат алгоритму NH в 128 раз менше вхідного.

L2-hash – другий етап

L2-hash працює з виходом L1-hash, використовує поліноміальний алгоритм POLY. Другий етап гешування використовується, тільки якщо довжина вхідного повідомлення більше 16 мегабайт. Використання алгоритму POLY потрібно для того, щоб уникнути тимчасову атаку. На виході з алгоритму POLY виходить 16 байтне число.

L3-hash – третій етап

Цей етап потрібно для того щоб з вихідних 16 байтів алгоритму L2-hash одержати 4-байтне значення.

КОПЗ – 2024

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ системи машинного навчання у Big Data яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Файл; Моніторинг мережі; Моделювання завантаженості; Параметри; Довідка.

– Блоку обробки запитів (моніторингу).

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

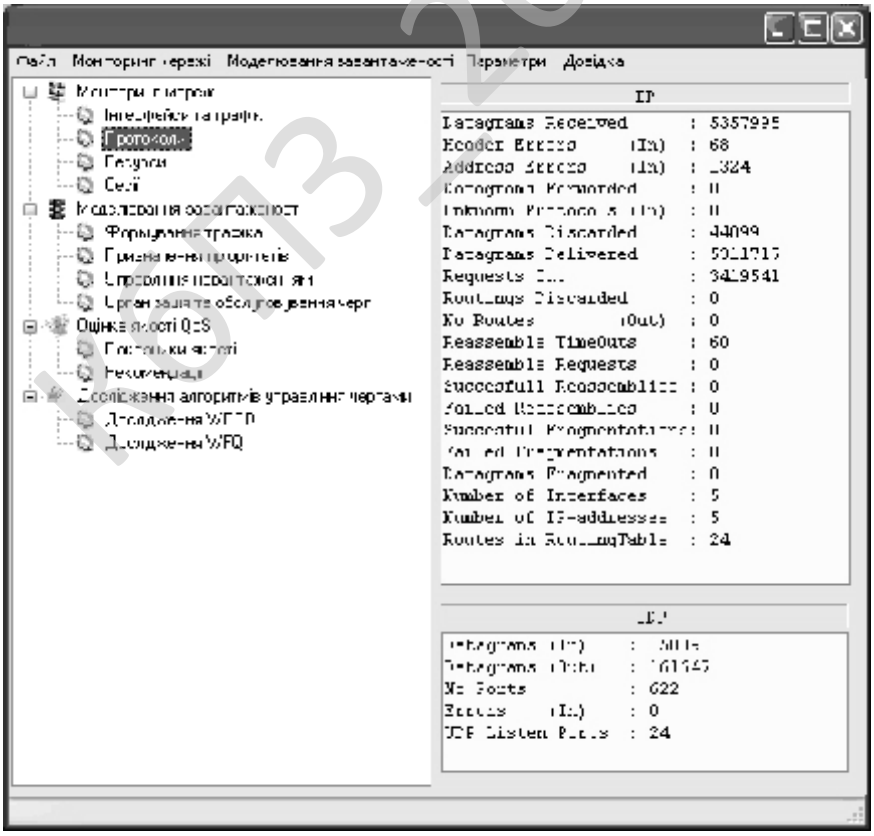

Рисунок 5.1 – Інтерфейс головного вікна

КПЗ - 2024

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

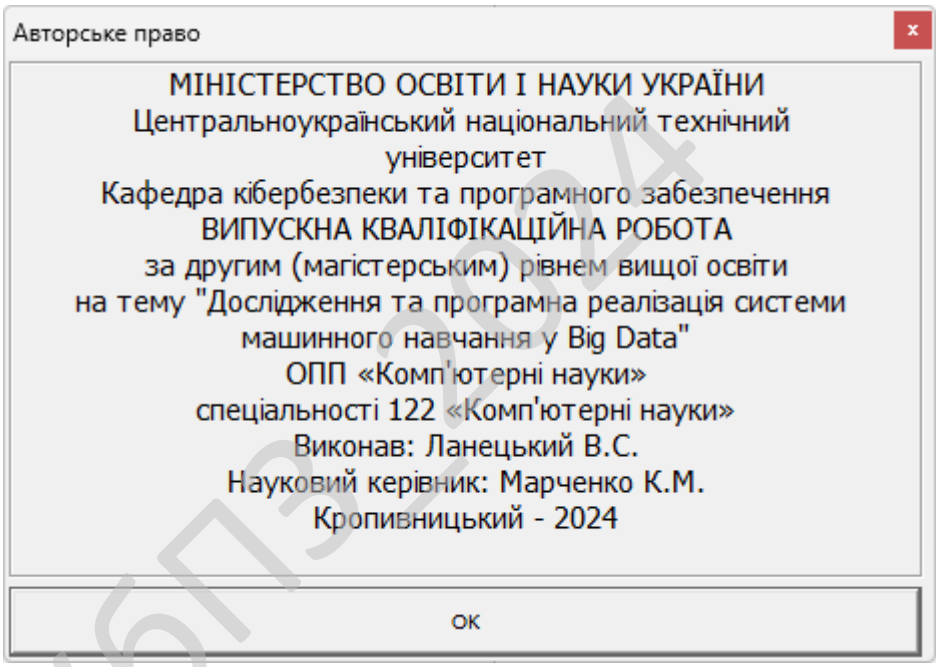

Рисунок 5.2 – Авторське право

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

КБПЗ–2024

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

КПЗ-2024

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

КПЗ-2024

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

КПЗ – 2024

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи машинного навчання у Big Data.

Метою розробки є дослідження та програмна реалізація системи машинного навчання у Big Data.

Об'єктом дослідження є процес машинного навчання у Big Data.

Предметом дослідження є методи машинного навчання у Big Data.

Методи дослідження базуються на методах машинного навчання та обробки Big Data, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод машинного навчання у Big Data.

– Розроблено вітчизняний продукт машинного навчання у Big Data, який має більш широкі можливості, на відміну від існуючих аналогів.

КбПЗ-2024

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та програмної реалізації системи машинного навчання у Big Data можуть бути цікавими для різних груп та секторів. Ось кілька прикладів:

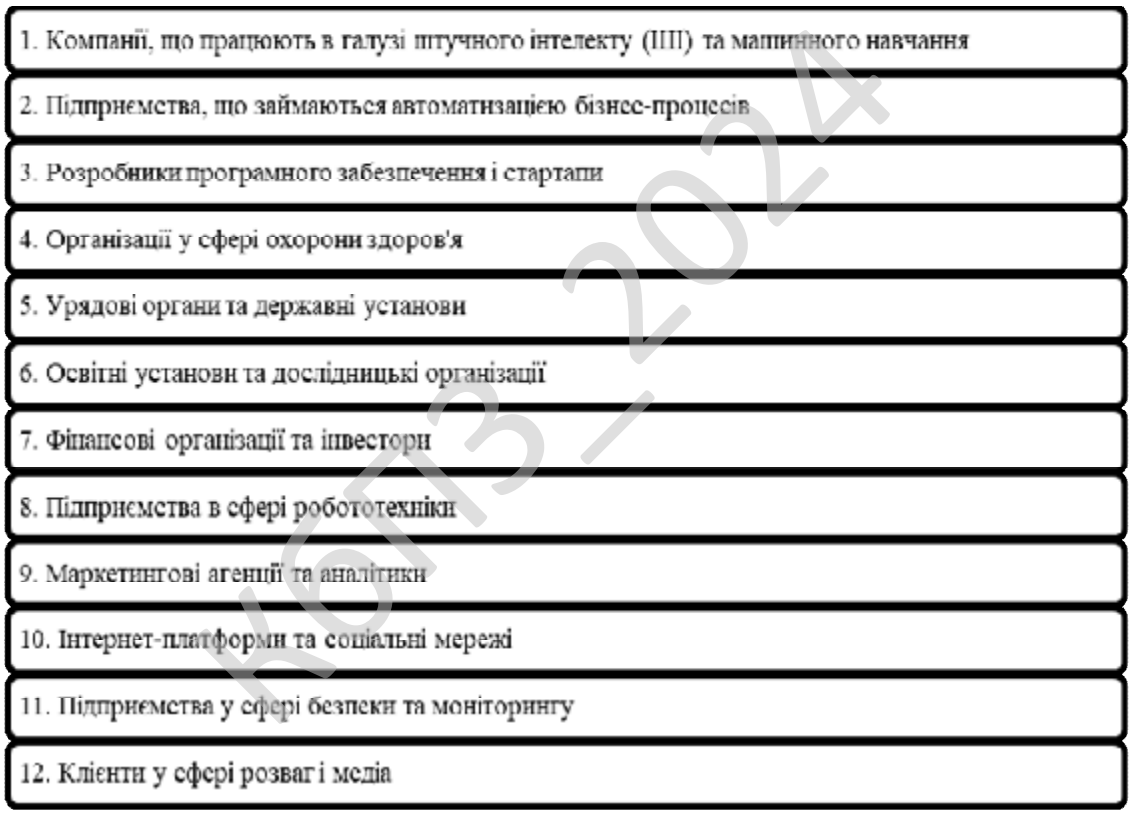

Рисунок 7.1 – Цільова аудиторія

Всі ці групи можуть отримати значну вигоду від дослідження та впровадження програмних систем, що імітують людське прийняття рішень, оскільки вони забезпечують більш ефективні, точні та швидкі рішення в різних сферах діяльності.

КОПІЯ – 2024

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Оцінка привабливості програмної реалізації системи машинного навчання у Big Data через методи експертних оцінок. Для оцінки привабливості проекту доцільно залучити експертів з різних галузей: IT та розробка програмного забезпечення (експерти, що мають досвід у створенні систем автоматичного прийняття рішень), бізнес-аналітики та економісти (для оцінки економічної ефективності та вигод від впровадження), фахівці з штучного інтелекту та машинного навчання (для оцінки технічних можливостей системи), представники потенційних користувачів (це можуть бути бізнес-користувачі або клієнти, які використовуватимуть систему).

Далі визначаємо ключові критерії, за якими експерти будуть оцінювати привабливість програмної реалізації системи (рисунок 7.2).

Рисунок 7.2 – Критерії оцінювання проєкту експертами

КПІ 2024

Оцінку можна проводити за допомогою шкали, наприклад, від 1 до 5 або 1 до 10, де: 1 – дуже низький рівень, 5 або 10 – високий рівень привабливості для кожного критерію.

Кожен експерт оцінює кожен критерій, після чого отримуються середні бали для кожного критерію, які зводимо до таблиці 7.1.

Таблиця 7.1 – Зведені результати оцінювання

Критерій	Експерт 1	Експерт 2	Експерт 3	Середній бал
Технічна ефективність	8	9	8	8.33
Інноваційність	7	8	9	8.00
Економічна доцільність	6	7	6	6.33
Легкість інтеграції	7	7	8	7.33
Масштабованість	8	8	9	8.33
Потенціал для комерціалізації	7	8	7	7.33

КПЗ – 2024

Загальний середній бал для кожного критерію допомагає зрозуміти, як експерти оцінюють привабливість проекту. Вищий бал у категоріях, таких як технічна ефективність, масштабованість та інноваційність, свідчить про те, що система має потенціал для успішного впровадження та розвитку. Якщо виявляються слабкі місця, наприклад, в економічній доцільності або легкості інтеграції, це може бути сигналом для коригування проєкту.

Кількісні та якісні оцінки, отримані від експертів, дозволяють зробити висновки щодо того, наскільки перспективним і привабливим є проект з погляду технічного та економічного потенціалу. Цей підхід допомагає приймати обґрунтовані рішення щодо подальшого розвитку, фінансування та впровадження програмної реалізації системи.

Цей метод експертних оцінок надає структурований підхід до оцінки привабливості проекту, що дозволяє мінімізувати суб'єктивність та забезпечити прийняття обґрунтованих рішень.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості програмної реалізації системи машинного навчання у Big Data доцільно застосувати комплексний підхід. Пропонуємо розглядати варіант використання методу аналогів та експертних оцінок.

1. Метод порівняння з аналогами (Market-based method)

- Цей метод полягає в оцінці вартості на основі порівняння з іншими подібними проектами або продуктами на ринку. Для цього проводиться аналіз аналогічних систем автоматичного прийняття рішень, зокрема в галузях штучного інтелекту та машинного навчання.
- **Переваги**: оцінка вартості на основі реальних даних з ринку; допомагає зрозуміти рівень конкурентоспроможності та можливості для монетизації.
- **Недоліки**: може бути важко знайти точні аналоги для специфічних або інноваційних проектів; зміни на ринку можуть вплинути на точність порівняння.

2. Метод експертних оцінок (Expert judgment method)

- Цей метод полягає в тому, щоб залучити експертів для оцінки вартості проекту на основі їх досвіду, знань і попередніх проектів. Експерти можуть використовувати кілька критеріїв, таких як складність задачі, рівень технологій, доступність ресурсів, і прогнозовані витрати.
- **Переваги**: здатність оцінити складні чи нові проекти, де відсутні чіткі ринкові аналоги; може враховувати не тільки фінансові аспекти, але й інші чинники, такі як технічний ризик.
- **Недоліки**: суб'єктивність оцінки, що може призвести до похибок, потребує залучення висококваліфікованих експертів.

КОПІЯ

Рисунок 7.3 – методи оцінки вартості ПЗ

Кількість методів та їх вид остаточно залежить від етапу розвитку проекту, характеру витрат і типу інвестицій.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Економічна ефективність від впровадження системи машинного навчання у Big Data може бути оцінена на основі різних факторів, таких як зниження витрат, підвищення продуктивності, скорочення часу на прийняття рішень та підвищення точності результатів. Ось приклад того, як можна оцінити економічну ефективність цього проєкту.

Крім прямих фінансових вигод, є і додаткові переваги:

– покращення якості обслуговування клієнтів завдяки швидшим та точнішим рішенням;

– підвищення конкурентоспроможності на ринку;

– зниження ризику людських помилок у критичних ситуаціях.

Впровадження системи машинного навчання у Big Data може призвести до значних економічних вигод, зокрема за рахунок зменшення витрат на персонал, збільшення продуктивності та зменшення кількості помилок. Це дозволяє суттєво підвищити економічну ефективність проекту.

Таблиця 7.2 – Основні показники впровадження проєкту

1. **Передумови для впровадження системи:** Проект стосується автоматизації прийняття рішень в компанії, яка працює в галузі фінансових послуг. Раніше рішення приймалися вручну, що вимагало залучення значної кількості співробітників та часу. Система автоматичного прийняття рішень повинна імітувати людське прийняття рішень, базуючись на аналізі великих обсягів даних і виявленні патернів.

КПЗ–2024

Продовження таблиці 7.2

2. **Витрати на впровадження системи:** **Інвестиції в розробку та впровадження системи**: 500 000 грн. **Підготовка персоналу (навчання, сертифікація)**: 50 000 грн. **Обслуговування та оновлення системи (щорічно)**: 40 000 грн.
3. **Очікувані вигоди:** **Зниження витрат на персонал**: До впровадження системи 10 аналітиків приймали рішення вручну, кожен із яких отримував зарплату 50 000 грн на рік. Після автоматизації цей персонал можна скоротити на 70%, що дозволить зекономити 350 000 грн на рік. **Збільшення продуктивності**: Система дозволяє приймати рішення в 3 рази швидше, що дозволяє збільшити кількість оброблених справ (якщо до цього компанія обробляла 1 000 справ на місяць, тепер зможе обробляти 3 000). Це збільшить доходи від наданих послуг на 150 000 грн на рік. **Зменшення помилок**: Система зменшує кількість помилок у прийнятті рішень, що дозволяє уникнути штрафів та репутаційних втрат. Очікувана економія від цього складає 100 000 грн на рік.
4. **Розрахунок економічної ефективності:** Для розрахунку ефективності можна використовувати показник **чистої теперішньої вартості (NPV)** або **показник внутрішньої норми прибутковості (IRR)**. **NPV**: Початкові витрати: 500 000 грн (інвестиції) + 50 000 грн (навчання) = 550 000 грн. Щорічні вигоди: 350 000 грн (економія на персоналі) + 150 000 грн (збільшення доходу) + 100 000 ГРН (економія від зменшення помилок) = 600 000 грн на рік. Строк оцінки: 5 років. Вартість капіталу (ставка дисконту): 10% на рік. NPV=(1+0.1)1600000+(1+0.1)2600000+(1+0.1)3600000+(1+0.1)4600000+(1+0.1)5600000−550000

КПІ_2024

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Для ефективного просування проєкту програмної реалізації системи машинного навчання у Big Data важливо розробити детальний алгоритм, який охоплюватиме як технічні, так і маркетингові аспекти (рисунок 7.4).

Рисунок 7.4 – Алгоритм просування проєкту

Цей алгоритм охоплює всі етапи просування проєкту, від розробки до масштабування, і дозволяє ефективно просувати програмну реалізацію системи машинного навчання у Big Data на ринку.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Оптимізація каналів збуту та шляхів реалізації проєкту програмної реалізації системи машинного навчання у Big Data вимагає інтегрованого підходу, який включає як технологічні, так і маркетингові стратегії (рисунок 7.5).

КПЗ-2024

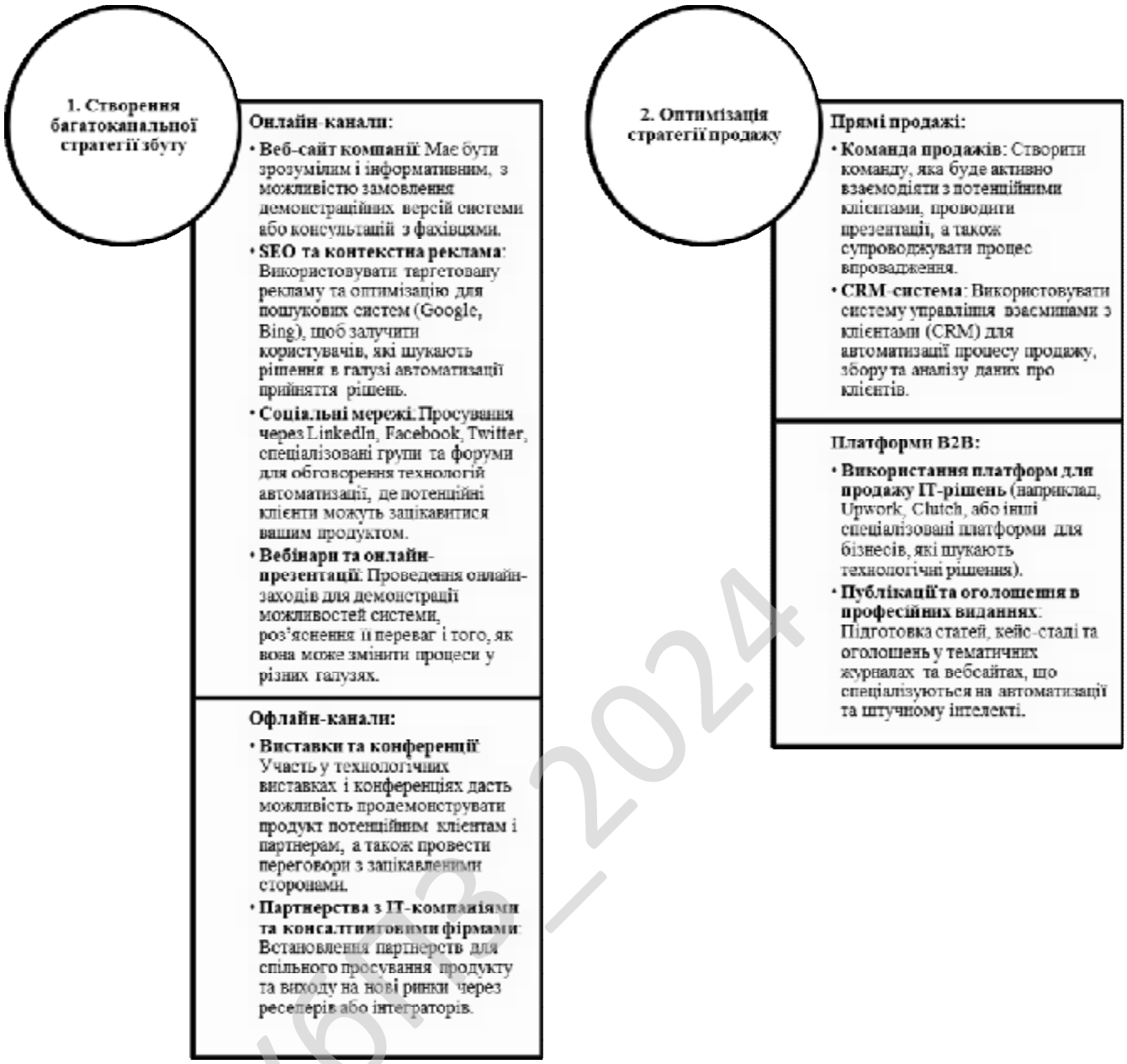

Рисунок 7.5 – Напрями оптимізації каналів збуту

Ці стратегії допоможуть ефективно оптимізувати канали збуту та шляхи реалізації проєкту, забезпечуючи стійке зростання продажів і максимальну вигоду для обох сторін – розробника і клієнтів.

КБ-3_2024

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключовими факторами успіху проєкту програмної реалізації системи машинного навчання у Big Data є кілька критичних аспектів, які забезпечують ефективне впровадження та стабільну роботу такої системи (рисунок 7.6).

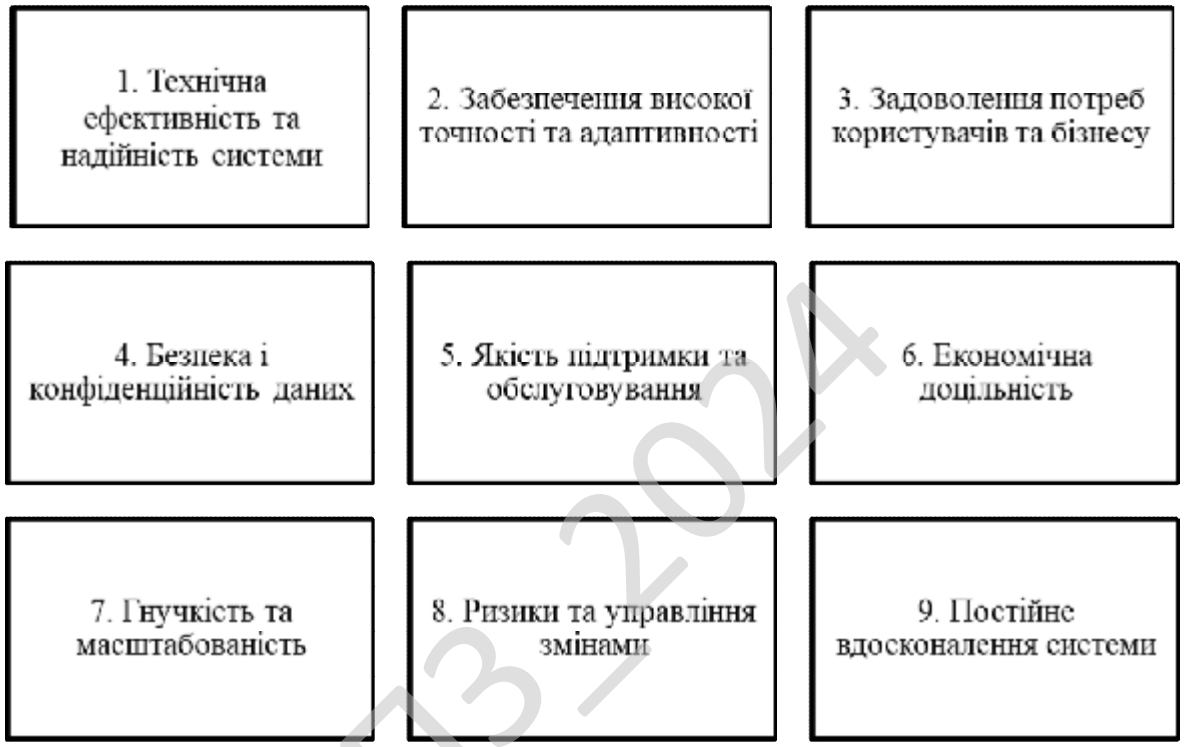

Рисунок 7.6 – Ключові фактори успіху проєкту

КПІ 2024

Забезпечення успіху проєкту програмної реалізації системи машинного навчання у Big Data потребує збалансованого підходу до технічних, економічних і організаційних аспектів. Успішна реалізація такого проєкту забезпечить значні переваги для бізнесу, включаючи підвищення ефективності, швидкості прийняття рішень та зниження витрат.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

У сучасному житті електронно-обчислювальна машина (ЕОМ) займає важливе місце. Нині ми можемо уявити свою професійну діяльність без такого обов'язкового елемента нашого побуту. Зараз за допомогою ЕОМ керуються життєво важливі процеси. Це і управління різними електростанціями, аеропортами, постачання та логістика, документообіг, комерція та розробки, управління виробничими процесами тощо.

Але ЕОМ має і зворотний бік, а саме – небезпеку. Адже вона є складним електронним пристроєм, у якому є безліч процесів, які можуть стати джерелом небезпеки. З ЕОМ потрібно поводитися обережно та знати, як убезпечити себе від негативних наслідків його використання. Оскільки захворювання можуть бути спричинені надмірним фізичним або розумовим навантаженням, через велику нервово-емоційну напругу, або через виробниче середовище. Також програмісти у процесі роботи отримують негативний вплив на органи зору та руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаження. Ці шкідливі фактори можуть привести до професійних захворювань. В даному розділі магістерської роботи проведемо аналіз основних чинників при роботі програміста.

Законом України “Про охорону праці” [7] регламентуються загальні положення державної політики в галузі охорони праці, а також конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [8], яким затверджено нормативно-правовий

КОПЗ–2024

акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Законодавством України чітко врегульовано норми та вимоги до використання комп'ютерної техніки на підприємстві, безпосередньо й охорона праці на підприємстві при роботі за комп'ютером., зокрема «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», затверджені наказом Мінсоцполітики від 14.02.2018 № 207 [1], «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98. [2].

Загальні вимоги пожежної безпеки під час експлуатації комп'ютерної техніки визначають «Правила пожежної безпеки в Україні» (затверджені наказом МВС від 30.12.2014 № 1417) [3], комп'ютерних класів – пункт 3 розділу VIII «Правил пожежної безпеки для навчальних закладів та установ системи освіти України» (затверджені наказом МОН від 15.08.2016 № 974). [4] та інші державні стандарти, що регламентують експлуатування комп'ютерної техніки як радіоелектронної апаратури.

КБПЗ-2024

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Можна виділити наступні основні фактори, що впливають на стан здоров'я людей, які працюють за комп'ютером:

– сидяче положення на протязі тривалого періоду;

– вплив електромагнітного випромінювання монітора;

– втома очей, навантаження на зір;

– перевантаження суглобів кистей;

– стрес при втраті інформації або виникненні критичних помилкою.

У кожному з цих випадків ступінь ризику прямо пропорційний часу, що проводиться за комп’ютером і поблизу від нього. В сучасних умовах взаємодія людини з технікою значно ускладнилась, що вимагає комплексного підходу, який передбачає розгляд людини, технічних засобів праці та виробничого середовища, як взаємозв'язаних елементів єдиної системи. Все вищесказане в повній мірі відноситься й до системи «людина–комп'ютер–середовище».

Вагомий вплив на працездатність та здоров’я користувачів комп’ютерів здійснює виробниче середовище. Це середовище у виробничих приміщеннях (офісах), в основному, визначається мікрокліматом, освітленням, наявністю шкідливих речовин у повітрі, рівнем шуму та випромінювання.

Для того, щоб об’єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам та запропонувати заходи щодо зменшення негативного впливу комп’ютера на організм людини необхідно скласти санітарно-гігієнічну характеристику умов працівника, який працює з програмним продуктом.

КБПЗ – 2024

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці користувача ПК

Розглянемо приміщення в якому працює користувач ПК з даним програмним продуктом.

Приміщення має одностороннє природне освітлення і загальне штучне освітлення. Стіни і стеля обклеєні світлими шпалерами, підлога вкрита темним ламінатом. У приміщенні відсутні сильні вібрації та шкідливі речовини. Склад повітря відповідає нормі. У кімнаті знаходиться ПК з 4-ядерним процесором і 23-дюймовим IPS-монітором, а також меблі.

Приміщення має довжину 4м, ширину 3,5м, висоту стелі 2,7м. Кількість робочих місць – одне. Площа –14 $м^2$, об'єм –37,8 $м^3$. Виходячи з цього, отримано дані, наведені в таблиці 8.1.

Таблиця 8.1 – Фактичні та нормативні значення параметрів приміщення

Параметр	Норма *	Реальні параметри
Площа, S	не менше 6 м^2	14 м^2
Об'єм, V	не менше 20 м^3	37,8 м^3

*Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

За даними, які наведено у табл. 8.1, можна зробити висновок, що отримані показники, площа та об'єм приміщення у розрахунку на одно робоче місце користувача ПК відповідає чинним нормам і вимогам.

Щодо мікроклімату, то згідно з ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [5] роботу з ПК можна віднести до категорії легка – 1а. Джерелами тепла в цьому приміщенні є люди, електроустаткування, освітлювальні прилади в темний час доби і система опалювання взимку. Оператором виділяється до 120 ккал теплової енергії за годину. Оптимальні та фактичні значення параметрів мікроклімату приведені в таблиці 8.2.

Таблиця 8.2 – Значення параметрів мікроклімату

Період року	Параметр	Оптимальний*	Фактичний
Теплий	Температура	23 – 25 ^{0}C	24^0 C
	Вологість	40 – 60%	50%
	Швидкість повітря	< 0,1м/с	
Холодний	Температура	22 – 24^0 C	23^0 C
	Вологість	40 – 60%	55%
	Швидкість повітря	< 0,1м/с	

*ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень»

КБПЗ_2024

По отриманим замірам параметрів мікроклімату можна зробити висновок, що всі показники задовольняють вимогам, зазначеним для робіт категорії легка 1а і є задовільними для здоров'я людини.

Щодо освітлення , то згідно з ДБН В.2.5-28:2006 «Природне і штучне освітлення» [6] ця робота відноситься до Va розряду зорових робіт. Передбачається використання природного, штучного і змішаного освітлення.

Природне освітлення здійснюється за допомогою вікна, площа якого складає S' = 1,8х1,5 = 2,7 м^2 і є бічним освітленням. У світильниках місцевого і загального освітлення використовуються світлодіодні лампи потужністю 20 Вт із світловим потоком однієї лампи 900 лм. Згідно замірів рівень освітлення в даному приміщенні і на робочому місці складає в межах 350 -500 лк, що відповідає нормованому значенню.

Джерелом шуму в приміщенні є комп'ютер. Вентилятори (кулери) системного блоку, процесора, відеокарти і блоку живлення є сучасними і мають низький рівень шуму. Згідно з технічною документацією шум, зумовлений кулером в блоці живлення складає 25 дБ, кулером процесора – 30 дБ, загальний – 34 дБ. Враховуючи незначний рівень шуму від персонального комп'ютера і незначний рівень фонового шуму від іншого устаткування, можна стверджувати, що сумарний рівень шумового забруднення приміщення не перевищує максимально допустимий рівень коригованої звукової потужності і складає не більше 50 дБА, що відповідає рівню шуму для приміщень з комп’ютерною технікою згідно Державних санітарних правил і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98. У приміщенні відсутні джерела інфрачервоного, ультрафіолетового і електромагнітного випромінювання, бо монітор ПК вироблений на основі рідкокристалічної матриці, підсвітка якої здійснюється неоновою лампою, які не мають сильного електромагнітного випромінювання і сертифіковані в Україні.

Блок живлення є екранованим і не випускає вищезазначених видів випромінювання.

КОПЗ_2024

8.4 Розробка заходів з умов поліпшення охорони праці

Перерахуємо проведені заходи щодо забезпечення умов праці на робочому місці користувача ПК.

З точки зору забезпечення електробезпеки до цих заходів можна віднести: устаткування розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв; періодична перевірка всіх приладів і пристроїв; щорічна здача іспитів з охорони праці.

З точки зору забезпечення оптимальних умов мікроклімату, рівня звуку і освітленості до цих заходів можна віднести: організацію природної вентиляції, за допомогою дефлектора, для забезпечення необхідного повітрообміну в приміщенні вузла; організацію системи центрального опалювання, для підтримки оптимальної температури в холодний період року; організацію штучного загального освітлення, для забезпечення необхідних умов зорової роботи, що відповідають, оформлення паспорта на приміщення вузла, з занесенням в нього вимірювань освітленості і рівня звуку, проведених відділом охорони праці.

Крім рекомендацій щодо конкретного приміщення, де було проведено дослідження умов праці, існують загальні вимоги, які зарекомендовані відповідними нормативними документами.

Правильна організація робочих місць запобігає передчасній втомлюваності користувача і сприяє збереженню здоров'я. Організація робочого місця передбачає:

– правильне розміщення робочого місця у виробничому приміщенні;

– вибір ергономічного обґрунтованого робочого положення, виробничих меблів з урахуванням характеристик людини;.

– раціональне компонування обладнання на робочих місцях;

– урахування характеру й особливостей трудової діяльності. Стосовно робочих місць користувача ВДТ, то організація робочого має забезпечуватися відповідно до ДСанПіН 3.3.2-007-98. Для запобігання перевтомленню необхідно

КОПІЯ 2024

виконувати вправи для очей та дотримуватись розпорядку роботи та відпочинку. На робочому місці реалізовувався режим відпочинку: кожні дві години – перерва для виконання фізичних вправ для м‘язів очей.

8.5. Протипожежний захист

Пожежі в приміщеннях з оргтехнікою становлять особливу небезпеку, бо поєднані з великими матеріальними збитками. Пожежа може виникнути при взаємодії горючих речовин і джерел запалювання. Горючими речовинами є будівельні та опоряджувальні матеріали, пластмасові корпуси техніки, шнури тощо. Джерелами запалювання можуть бути електронні схеми комп'ютерів, принтерів, пристроїв електроживлення, де внаслідок різних порушень виникає перегрівання елементів, утворюються електричні іскри та дуги, здатні спричинити займання горючих матеріалів.

При обслуговуванні, ремонтних та профілактичних роботах використовуються різні легкозаймисті рідини, прокладаються тимчасові електропровідники, здійснюється паяння. Виникає додаткова пожежна небезпека, яка потребує відповідних заходів пожежного захисту.

До засобів гасіння пожежі, призначених для локалізації невеликих займань, належать вогнегасники, сухий пісок, азбестові ковдри. Приміщення, в якому встановлено комп'ютери і де немає необхідності влаштування систем автоматичного пожежогасіння, необхідно оснащувати переносними вуглекислотними вогнегасниками з розрахунку 2 шт. на кожні 20 $м^2$ в приміщеннях.

Звуковбирне облицювання стін, стель приміщень треба виконувати з негорючих та важко горючих матеріалів.

З метою виявлення початкової стадії займання необхідно використовувати пристрої систем автоматичного пожежогасіння там, де цього вимагають Правила пожежної безпеки.

КПЗ-2024

З точки зору забезпечення пожежної безпеки до цих заходів можна віднести наявність схеми евакуації з приміщення у випадку пожежі, прикріплену на вхідні двері.

8.6 Розрахункова частина

В приміщенні (де відсутні джерела виділення шкідливих речовин) працює двоє людей. Робота пов'язана з використанням ПЕОМ. Розміри приміщення: А = 4 м, В = 7 м, Н = 2.8 м, устаткування займає 15% об'єму. Визначимо найменшу необхідну кількість повітря для вентиляції.

Для приміщень, в яких відсутні виділення шкідливих речовин у повітрі, розрахунок вентиляції здійснюється залежно від кількості працюючих.

Необхідна кількість повітря ($м^3$/год.), яка забезпечує відповідність параметрів повітря робочої зони нормованим значенням, визначається за наступною формулою:

$$L = L' \cdot N,$$

де L' – нормативна кількість повітря на одного працюючого, яка залежить від питомого об'єму приміщення, $м^3$/(год.-люд.);

N – кількість працюючих.

Питомий об'єм приміщення $V_п$, ($м^3$ /люд.), визначається за формулою:

$$V_п = V/N,$$

де V – об'єм приміщення, $м^3$.

Визначаємо вільний об'єм приміщення

$$V = A \cdot B \cdot H \cdot 0,85 = 4 \cdot 7 \cdot 2,8 \cdot 0,85 = 66,4 \text{ м}^3.$$

Питомий вільний об'єм складає

$$V' = V / N = 66,4 / 2 = 33,2 \text{ м}^3/\text{люд.} > 20 \text{ м}^3/\text{люд.}$$

Нормована кількість повітря на одну людину при V' > 20 $м^3$ /люд. становить 30 $м^3$ /(год.·люд.).

КОПЗ-2024

Висновки до розділу

У даному розділі магістерської роботи проведено аналіз умов працівника робота якого пов’язана з комп’ютерною технікою. Проведено аналіз основних санітарно – гігієнічних показників в заданому приміщені , де працівник зайнятий постійною роботою за комп’ютером.. Створені умови повинні забезпечувати комфортну роботу. На підставі вивченої літератури з даної проблеми, були зазначені оптимальні параметри мікроклімату, освітлення, допустимі рівні шуму та іонізуючого випромінювання при роботі з ПЕОМ, а також розраховано найменшу необхідну кількість повітря для вентиляції.

Дотримання умов, що визначають оптимальну організацію робочих місць працівників, дозволить зберегти гарну працездатність протягом усього робочого дня, підвищить як в кількісному, так і в якісному відносинах продуктивність їх праці.

КбПЗ_2024

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи машинного навчання у Big Data.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів машинного навчання у Big Data.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем машинного навчання у Big Data.

– Досліджена система машинного навчання у Big Data.

– На основі отриманих результатів досліджень створена програмна реалізація системи машинного навчання у Big Data.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання машинного навчання у Big Data.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

КПЗ-2024

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм UMAC.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

КбПЗ-2024

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ланецький В.С. Дослідження та програмна реалізація системи машинного навчання у Big Data // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2024.

2. Mueller J.P., Massaron L. Machine Learning For Dummies. – Wiley, 2016. – 714 p.

3. Teofili T. Deep Learning for Search. – Manning, 2019. – 695 p.

4. Rungta K. TensorFlow in 1 Day: Make your own Neural Network. – Publishdrive, 2019. – 587 p.

5. Weidman S. Deep Learning from Scratch: Building with Python from First Principles. – O'Reilly. 2019. – 252 p.

6. Rajasekaran S., Vijayalakshmi Pai G.A. Neural networks, fuzzy logic, and genetic algorithms: synthesis and applications (with cd-rom) Kindle Edition. – PHI, 2013. – 628 p.

7. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

8. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

9. Smirnov O., Fedorov E., Neskorodieva A., Neskorodieva T. «Intellectual Classification method of Gymnastic Elements Based on Combinations of Descriptive and Generative Approache». *CEUR Workshop Proceedings* Volume 3664, 2024, Pages 11-23.

КОПЗ – 2024

10. Malyukov V., Bebeshko B., Lakhno V., Smirnov O., Malyukova I., Mohylnyi H. «Managing the Purchase-Sale Process of Digital Currencies Under Fuzzy Conditions». *Lecture Notes in Networks and Systems,* 2023, 729 LNNS, pp. 104–112.

11. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

12. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

13. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

14. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022, pp. 1-12.

15. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

16. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

17. Kuznetsov, A., Oleshko, I., Chernov, K., Bagmut, M., Smirnova, T. «Biometric authentication using convolutional neural networks». Lecture Notes in Networks and Systems. Volume 152, 2021, Pages 85-98.

КОПІЯ 2024

18. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

19. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». CEUR Workshop Proceedings Volume 3101, 2021, Pages 192-207.

20. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

21. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

22. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

23. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

24. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

25. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated

КОПЗ-2024

with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

26. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

27. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

28. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

29. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». CEUR Workshop Proceedings, Vol 2588, P. 215-227, 2019.

30. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

31. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

32. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in

КОПЗ-2024

Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 353-358.

33. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

34. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

35. Smirnov, S., Bulekbaeva, G., Kikvidze, O.G., Lakhno, V., Brzhanov, R., Tabylov, A. «Computer simulation in the MathCAD package of plastic deformation of the deposited layer on the flat surface of the part». Journal of Theoretical and Applied Information Technology Volume 97, Issue 20, 2019, Pages 2467-2484. (Scopus).

36. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

37. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

38. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». Сучасні інформаційні системи. 2021. Т. 5, № 4. С. 79-95

39. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для

КОПЗ – 2024

підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

40. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

41. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

42. Смірнов, С.А., Смірнова, Т.В., Минайленко, Р.М., Доренський, О.П., Сисоєнко С.В. «Хмарна автоматизована система інтелектуальної підтримки прийняття рішень для технологічних процесів». Вісник Черкаського державного технологічного університету. Технічні науки. №4, 2020, С. 84-92.

43. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

44. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

45. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура № 1 (11). с. 48-57, 2019.

46. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу

КОПЗ_2024

відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

47. Смірнова Т.В., Дрєєв О.М., Смірнов О.А. Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням. Системи управління, навігації та зв'язку, № 2 (54). С. 149-154, 2019.

48. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

49. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

50. Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1 (37). – Полтава: ПолтНТУ. – 2016. – С. 98-103

51. Смірнов О.А., Лисенко І.А. Формалізація процесу проектування тестових наборів. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 3 (48). – Харків: ХУПС. – 2016. – С.96-100.

52. Смірнов О.А., Лисенко І.А. Удосконалення методу перевірки коректності таблиць рішень для подання тестових наборів. Збірник наукових праць "Системи обробки інформації". – Випуск 8 (145). – Х.: ХУПС – 2016. – С. 77-80.

КОПЗ-2024

Додаток А
(обов'язковий)

Технічне завдання

Зміст

КПЗ–2024

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи машинного навчання у Big Data.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 18-13 від 07.08.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи машинного навчання у Big Data.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

КБПЗ_2024

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи машинного навчання у Big Data;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

КБПЗ_2024

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

КПЗ – 2024

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати маркетингове та економічне обґрунтування ІТ-проєкту з урахуванням цін на 3 вересня 2024 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз санітарно-гігієнічних умов праці на робочому місці користувача ПК.

КбПЗ_2024

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 97 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Маркетингове та економічне обґрунтування ІТ-проєкту.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 02.12.2024 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 19.12.2024 р.

КПЗ – 2024

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Марченко К.М.

КбПЗ_2024

Дослідження та програмна реалізація системи машинного навчання у Big Data

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 17

Літера: РП

Кропивницький – 2024 року

Основна програма main.py

```
# Імпорт функцій для роботи з великими даними
from big_data_handling import process_large_dataset

# Головна функція
def main():
    # Опція для запуску з великим набором даних
    use_large_data = input("Чи бажаєте працювати з великим набором даних? (y/n): 
").lower() == 'y'

    if use_large_data:
        # Робота з великим набором даних
        process_large_dataset()
    else:
        # Стандартний робочий процес
        data = generate_big_data()

        print("Аугментація набору даних...")
        augmented_data = augment_data(data, factor=3)

        print("Попередня обробка даних...")
        processed_data = preprocess_data(augmented_data)

        X = processed_data[['feature1', 'feature2', 'feature3']]
        y = processed_data['target']
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, 
random_state=123)

        print("Оптимізація та навчання Random Forest...")
        best_rf = optimize_random_forest(X_train, y_train)
        y_pred_rf = best_rf.predict(X_test)
        metrics_rf = evaluate_model(y_test, y_pred_rf, "Random Forest 
(Optimized)")
        log_performance("Random Forest (Optimized)", metrics_rf, 
params=best_rf.get_params())

        print("Оптимізація та навчання Decision Tree...")
        best_dt = optimize_decision_tree(X_train, y_train)
        y_pred_dt = best_dt.predict(X_test)
        metrics_dt = evaluate_model(y_test, y_pred_dt, "Decision Tree 
(Optimized)")
        log_performance("Decision Tree (Optimized)", metrics_dt, 
params=best_dt.get_params())
```

КОПЗ_2024

```
        print("Оптимізація та навчання SVM...")
        best_svm = optimize_svm(X_train, y_train)
        y_pred_svm = best_svm.predict(X_test)
        metrics_svm = evaluate_model(y_test, y_pred_svm, "SVM (Optimized)")
        log_performance("SVM (Optimized)", metrics_svm,
params=best_svm.get_params())

        print("Генерація PDF-звіту...")
        generate_report()

        print("Побудова графіків для аналізу...")
        plot_model_accuracies()
        plot_metric_comparison()

        print("Запуск інтерактивної панелі...")
        run_dashboard()

# Запуск програми
if __name__ == "__main__":
    main()
```

КбПЗ_2024

Файл data_processing.py

```
import pandas as pd
import numpy as np

# Функція для генерації великого набору даних
def generate_big_data():
    # Ініціалізація випадкових даних
    np.random.seed(42)
    data = pd.DataFrame({
        'feature1': np.random.rand(10000),
        'feature2': np.random.rand(10000) * 10,
        'feature3': np.random.choice([0, 1], size=10000),
        'target': np.random.choice([0, 1], size=10000)
    })
    return data

# Функція для попередньої обробки даних
def preprocess_data(data):
    # Нормалізація першої ознаки
    data['feature1'] = (data['feature1'] - data['feature1'].min()) /
(data['feature1'].max() - data['feature1'].min())

    # Масштабування другої ознаки
    data['feature2'] = np.log1p(data['feature2'])

    # Інверсія третьої ознаки
    data['feature3'] = data['feature3'].apply(lambda x: 1 - x)

    return data
```

Файл model_training.py

```
# Тренування Random Forest
def train_random_forest(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=123)
    model = RandomForestClassifier(n_estimators=100, random_state=123)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    return y_pred, y_test

# Тренування Decision Tree
def train_decision_tree(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=123)
    model = DecisionTreeClassifier(max_depth=5, random_state=123)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    return y_pred, y_test

# Тренування SVM
def train_svm(X, y):
```

КбПЗ – 2024

```
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=123)
    model = SVC(kernel='linear', random_state=123)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    return y_pred, y_test
```

КбПЗ_2024

Файл visualization.py

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Функція для візуалізації розподілу даних
def plot_data_distribution(data):
    # Побудова гістограм для кожної ознаки
    plt.figure(figsize=(16, 6))
    for i, column in enumerate(data.columns[:-1]):
        plt.subplot(1, len(data.columns[:-1]), i + 1)
        sns.histplot(data[column], bins=30, kde=True)
        plt.title(f"Розподіл {column}")
        plt.xlabel(column)
    plt.tight_layout()
    plt.show()

# Функція для побудови матриці плутанини
def plot_confusion_matrix(y_true, y_pred, model_name):
    # Генерація матриці
    cm = confusion_matrix(y_true, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(cmap='Blues')
    plt.title(f"Матриця плутанини для {model_name}")
    plt.show()

# Функція для візуалізації залежностей між ознаками
def plot_feature_relationship(data):
    plt.figure(figsize=(8, 6))
    sns.scatterplot(x='feature1', y='feature2', hue='target', data=data,
palette='viridis')
    plt.title("Залежність між Feature1 та Feature2")
    plt.xlabel("Feature1")
    plt.ylabel("Feature2")
    plt.legend(title="Target")
    plt.show()
```

КОПІЯ - 2024

Файл evaluation.py

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

# Функція для оцінки продуктивності моделі
def evaluate_model(y_true, y_pred, model_name):
    # Розрахунок метрик
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='binary')
    recall = recall_score(y_true, y_pred, average='binary')
    f1 = f1_score(y_true, y_pred, average='binary')

    # Вивід метрик
    print(f"Оцінка моделі: {model_name}")
    print(f"Точність: {accuracy:.4f}")
    print(f"Точність (Precision): {precision:.4f}")
    print(f"Повнота (Recall): {recall:.4f}")
    print(f"F1-міра: {f1:.4f}")
    print("-" * 40)

    return {"accuracy": accuracy, "precision": precision, "recall": recall,
"f1": f1}
```

КбПЗ_2024

Файл hyperparameter_tuning.py

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

# Функція для оптимізації Random Forest
def optimize_random_forest(X_train, y_train):
    # Набір параметрів для перебору
    param_grid = {
        'n_estimators': [50, 100, 200],
        'max_depth': [5, 10, 20],
        'min_samples_split': [2, 5, 10]
    }

    # Ініціалізація моделі та GridSearchCV
    grid_search = GridSearchCV(RandomForestClassifier(random_state=123),
param_grid, cv=3, scoring='accuracy', verbose=1)

    # Навчання моделі
    grid_search.fit(X_train, y_train)

    # Вивід найкращих параметрів
    print("Найкращі параметри для Random Forest:", grid_search.best_params_)
    return grid_search.best_estimator_

# Функція для оптимізації Decision Tree
def optimize_decision_tree(X_train, y_train):
    # Набір параметрів для перебору
    param_grid = {
        'max_depth': [3, 5, 10],
        'min_samples_split': [2, 5, 10],
        'criterion': ['gini', 'entropy']
    }

    # Ініціалізація моделі та GridSearchCV
    grid_search = GridSearchCV(DecisionTreeClassifier(random_state=123),
param_grid, cv=3, scoring='accuracy', verbose=1)

    # Навчання моделі
    grid_search.fit(X_train, y_train)

    # Вивід найкращих параметрів
    print("Найкращі параметри для Decision Tree:", grid_search.best_params_)
    return grid_search.best_estimator_

# Функція для оптимізації SVM
def optimize_svm(X_train, y_train):
    # Набір параметрів для перебору
    param_grid = {
        'C': [0.1, 1, 10],
        'kernel': ['linear', 'rbf', 'poly'],
```

КБТВ_2024

```
        'gamma': ['scale', 'auto']
    }

    # Ініціалізація моделі та GridSearchCV
    grid_search = GridSearchCV(SVC(random_state=123), param_grid, cv=3,
scoring='accuracy', verbose=1)

    # Навчання моделі
    grid_search.fit(X_train, y_train)

    # Вивід найкращих параметрів
    print("Найкращі параметри для SVM:", grid_search.best_params_)
    return grid_search.best_estimator_
```

КбПЗ_2024

Файл data_augmentation.py

```
import pandas as pd
import numpy as np

# Функція для створення синтетичних даних
def augment_data(data, factor=2):
    augmented_data = data.copy()

    # Дублювання даних з випадковим шумом
    for _ in range(factor - 1):
        noise = pd.DataFrame({
            'feature1': np.random.normal(0, 0.01, size=len(data)),
            'feature2': np.random.normal(0, 0.5, size=len(data)),
            'feature3': np.random.choice([-1, 1], size=len(data)) * 0.1,
            'target': data['target']
        })

        # Додавання шуму до існуючих даних
        new_data = data[['feature1', 'feature2', 'feature3']] +
noise[['feature1', 'feature2', 'feature3']]
        new_data['target'] = noise['target']
        augmented_data = pd.concat([augmented_data, new_data],
ignore_index=True)

    return augmented_data

# Функція для розширення набору даних шляхом дублювання
def duplicate_data(data, factor=2):
    return pd.concat([data] * factor, ignore_index=True)
```

КбТЗ – 2024

Файл performance_logging.py

```
import json
import os

# Функція для створення/доповнення логу продуктивності
def log_performance(model_name, metrics, params=None,
log_file="performance_log.json"):
    # Формування запису
    log_entry = {
        "model_name": model_name,
        "metrics": metrics,
        "parameters": params
    }

    # Перевірка існування файлу
    if not os.path.exists(log_file):
        # Створення нового файлу
        with open(log_file, 'w') as f:
            json.dump([log_entry], f, indent=4)
    else:
        # Додавання до існуючого файлу
        with open(log_file, 'r') as f:
            logs = json.load(f)

        logs.append(log_entry)

        with open(log_file, 'w') as f:
            json.dump(logs, f, indent=4)

    print(f"Продуктивність моделі {model_name} успішно записано в {log_file}")
```

КбПЗ_2024

Файл report_generation.py

```
import json
from fpdf import FPDF

# Клас для створення PDF-звіту
class PDFReport(FPDF):
    def header(self):
        self.set_font('Arial', 'B', 12)
        self.cell(0, 10, 'Звіт про продуктивність моделей машинного навчання',
0, 1, 'C')
        self.ln(10)

    def footer(self):
        self.set_y(-15)
        self.set_font('Arial', 'I', 8)
        self.cell(0, 10, f'Сторінка {self.page_no()}', 0, 0, 'C')

    def add_model_performance(self, model_name, metrics, params):
        self.set_font('Arial', 'B', 12)
        self.cell(0, 10, f'Модель: {model_name}', 0, 1)
        self.set_font('Arial', '', 11)

        # Додавання метрик
        self.cell(0, 10, 'Метрики:', 0, 1)
        for metric, value in metrics.items():
            self.cell(0, 10, f'{metric}: {value:.4f}', 0, 1)

        # Додавання параметрів
        self.cell(0, 10, 'Параметри моделі:', 0, 1)
        for param, value in params.items():
            self.cell(0, 10, f'{param}: {value}', 0, 1)

        self.ln(10)

# Функція для створення звіту
def generate_report(log_file="performance_log.json",
report_file="performance_report.pdf"):
    # Читання лог-файлу
    if not log_file or not report_file:
        print("Файл логів не знайдено або не вказаний.")
        return

    with open(log_file, 'r') as f:
        logs = json.load(f)

    # Створення PDF
    pdf = PDFReport()
    pdf.add_page()

    for log in logs:
        model_name = log['model_name']
        metrics = log['metrics']
```

КОПЗ_2024

```
        params = log['parameters']
        pdf.add_model_performance(model_name, metrics, params)

# Збереження PDF
pdf.output(report_file)
print(f"Звіт збережено у файл {report_file}")
```

КбПЗ_2024

Файл custom_visualizations.py

```
import matplotlib.pyplot as plt
import json

# Функція для побудови графіку точності моделей
def plot_model_accuracies(log_file="performance_log.json"):
    # Завантаження логів
    with open(log_file, 'r') as f:
        logs = json.load(f)

    # Підготовка даних
    models = [log['model_name'] for log in logs]
    accuracies = [log['metrics']['accuracy'] for log in logs]

    # Побудова графіку
    plt.figure(figsize=(10, 6))
    plt.bar(models, accuracies, color='skyblue')
    plt.title("Точність моделей", fontsize=14)
    plt.xlabel("Моделі", fontsize=12)
    plt.ylabel("Точність", fontsize=12)
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()

# Функція для побудови порівняння метрик
def plot_metric_comparison(log_file="performance_log.json"):
    # Завантаження логів
    with open(log_file, 'r') as f:
        logs = json.load(f)

    # Підготовка даних
    models = [log['model_name'] for log in logs]
    metrics = ['accuracy', 'precision', 'recall', 'f1']
    metric_values = {metric: [log['metrics'][metric] for log in logs] for metric
in metrics}

    # Побудова графіку
    x = range(len(models))
    plt.figure(figsize=(12, 8))
    for metric, values in metric_values.items():
        plt.plot(x, values, marker='o', label=metric)

    plt.xticks(x, models, rotation=45, ha='right')
    plt.title("Порівняння метрик моделей", fontsize=14)
    plt.xlabel("Моделі", fontsize=12)
    plt.ylabel("Значення метрик", fontsize=12)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

КБІЗ_2024

Файл interactive_dashboard.py

```
from dash import Dash, dcc, html, Input, Output
import pandas as pd
import json

# Функція для завантаження даних із лог-файлу
def load_logs(log_file="performance_log.json"):
    with open(log_file, 'r') as f:
        logs = json.load(f)
    return logs

# Ініціалізація Dash-додатку
app = Dash(__name__)

# Завантаження даних
logs = load_logs()

# Підготовка даних для графіків
models = [log['model_name'] for log in logs]
metrics = ['accuracy', 'precision', 'recall', 'f1']
metric_data = {metric: [log['metrics'][metric] for log in logs] for metric in
metrics}

# Макет Dash-додатку
app.layout = html.Div([
    html.H1("Аналіз продуктивності моделей", style={'textAlign': 'center'}),
    dcc.Dropdown(
        id='metric-dropdown',
        options=[{'label': metric.capitalize(), 'value': metric} for metric in
metrics],
        value='accuracy',
        style={'width': '50%', 'margin': 'auto'}
    ),
    dcc.Graph(id='metric-graph', style={'margin-top': '20px'}),
    html.Div(id='model-details', style={'margin-top': '20px', 'textAlign':
'center'})
])

# Оновлення графіка залежно від вибраної метрики
@app.callback(
    Output('metric-graph', 'figure'),
    [Input('metric-dropdown', 'value')]
)
def update_graph(selected_metric):
    y_values = metric_data[selected_metric]
    return {
        'data': [{
            'x': models,
            'y': y_values,
            'type': 'bar',
            'name': selected_metric,
            'marker': {'color': 'blue'}
```

КБГЗ_2024

```
        }],
        'layout': {
            'title': f'Метрика: {selected_metric.capitalize()}',
            'xaxis': {'title': 'Моделі'},
            'yaxis': {'title': selected_metric.capitalize()}
        }
    }

# Оновлення деталей моделі на основі вибраної метрики
@app.callback(
    Output('model-details', 'children'),
    [Input('metric-dropdown', 'value')]
)
def update_model_details(selected_metric):
    best_model_index =
metric_data[selected_metric].index(max(metric_data[selected_metric]))
    best_model = logs[best_model_index]
    details = [
        html.H4(f"Найкраща модель для метрики {selected_metric.capitalize()}:
{best_model['model_name']}"),
        html.P("Параметри моделі:"),
        html.Ul([html.Li(f"{key}: {value}") for key, value in
best_model['parameters'].items()])
    ]
    return details

# Запуск сервера Dash
def run_dashboard():
    app.run_server(debug=True, use_reloader=False)
```

КбПЗ_2024

Файл big_data_handling.py

```
import dask.dataframe as dd
from dask_ml.model_selection import train_test_split as dask_train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Функція для створення великого набору даних за допомогою Dask
def generate_big_data_with_dask(rows=10**7, chunksize=10**6):
    # Генерація випадкових даних
    data = {
        'feature1': np.random.rand(rows),
        'feature2': np.random.rand(rows) * 10,
        'feature3': np.random.choice([0, 1], size=rows),
        'target': np.random.choice([0, 1], size=rows)
    }
    dask_df = dd.from_pandas(pd.DataFrame(data), npartitions=rows // chunksize)
    return dask_df

# Функція для роботи з великими даними
def process_large_dataset():
    # Генерація великого набору даних
    print("Генерація великого набору даних...")
    big_data = generate_big_data_with_dask()

    # Розбиття даних на навчальну та тестову вибірки
    print("Розбиття великого набору даних...")
    X = big_data[['feature1', 'feature2', 'feature3']]
    y = big_data['target']
    X_train, X_test, y_train, y_test = dask_train_test_split(X, y,
test_size=0.3, random_state=123)

    # Навчання моделі на великому наборі даних
    print("Навчання моделі Random Forest на великому наборі даних...")
    model = RandomForestClassifier(n_estimators=100, random_state=123)
    model.fit(X_train.compute(), y_train.compute())

    # Прогнозування та оцінка моделі
    print("Прогнозування та оцінка...")
    y_pred = model.predict(X_test.compute())
    accuracy = accuracy_score(y_test.compute(), y_pred)
    print(f"Точність моделі на великому наборі даних: {accuracy:.4f}")

    return model
```

КОПІЯ 2024