Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп’ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2024 р

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «Комп'ютерна модель прогнозування обсягу пам'яті_графічного процесора»

Захищено на засіданні
Атестаційної комісії № 42
протокол № __ від __.06.2024 р.
Оцінка _____ /
_____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**

Виконав:
студент 4 курсу, групи КІ–41
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 – Комп’ютерна інженерія.
Богданов Євгеній Сергійович

Керівник: к.т.н., професор, доцент кафедри теоретичної та прикладної системотехніки
БАКУМЕНКО Ніна Станіславовна

Рецензент:
д.т.н., доцент, професор закладу вищої освіти кафедри теоретичної та прикладної інформатики факультету математики і інформатики Руккас Кирило Маркович

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 51 сторінок, із яких 35 сторінки основної частини з 9 рисунками, 1 таблицею, 2 формулами, 12 найменуваннями списку використаних джерел та чотирма додатками.

Метою даної роботи є розробка комп'ютерної моделі прогнозування обсягу пам'яті графічного процесора в залежності від його технічних характеристик.

Об'єктом дослідження є прогнозування характеристик комп'ютерних систем за допомогою методів машинного навчання.

Предметом дослідження є регресійні методи моделювання та прогнозу.

Практична значимість результатів полягає у використанні розробленої моделі прогнозування в процесі аналізу ринку відеокарт.

Ключові слова: Python, Graphene, Pandas, регресійні моделі, метод найменших квадратів, реалізація проєкту, програмне забезпечення, апаратне забезпечення, кодова база.

ЗМІСТ

ВСТУП **4**

РОЗДІЛ 1 ОГЛЯД АРХІТЕКТУРИ ГРАФІЧНИХ ПРОЦЕСОРІВ **5**

1.1 Визначення актуальності роботи 5

1.2 Огляд основних технологій графічних прискорювачів 6

1.3 Принципи роботи графічних прискорювачів 8

1.4 Огляд роботи 3D прискорювачів та їхні основні функції. 11

1.5 Огляд методів прогнозування значень цільових змінних 13

Висновки по розділу 1 16

РОЗДІЛ 2 МЕТОДИ МНОЖИННОЇ ЛІНІЙНОЇ РЕГРЕСІЇ ДЛЯ ПРОГНОЗУВАННЯ ПАРАМЕТРІВ КОМП'ЮТЕРНИХ СИСТЕМ **17**

2.1 Постановка задачі та визначення мети 17

2.2 Вибір методів реалізації 19

2.3 Вибір технології комп'ютерної реалізації 21

Висновки по розділу 2 23

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ МОДЕЛІ ПРОГНОЗУВАННЯ ОБСЯГУ ПАМ'ЯТІ ГРАФІЧНОГО ПРОЦЕСОРА **24**

3.1 Реалізація обробки даних 24

3.2 Реалізація моделі лінійної регресії 25

3.3 Візуалізація результатів та аналіз прогнозів 28

Висновки по розділу 3 30

РОЗДІЛ 4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ ТА ОЦІНКА ЯКОСТІ МОДЕЛІ **31**

4.1 Дослідження вхідних змінних 31

4.2 Вибір предикторів 31

4.3 Побудова комп'ютерної моделі 32

4.4 Оцінка якості моделі 35

Висновки по розділу 4 38

ВИСНОВКИ **39**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **40**

Додаток А 41

Додаток Б 43

Додаток В 46

Додаток Г 50

ВСТУП

У сучасному світі, де технологічний прогрес змінює наше оточення з неймовірною швидкістю, роль графічних прискорювачів стає все більш критичною та різноманітною. Започаткування величезної кількості графічних додатків, від відеоігор до професійних програм для рендерингу, моделювання та обробки відео, створює попит на надійні та продуктивні графічні рішення.

Зараз, коли сучасні графічні прискорювачі відводяться для виконання навіть більшого різноманіття завдань, від відображення складної візуалізації до здійснення складних обчислень у реальному часі, їх важливість стає неоціненною. Відеокарти стали незамінним інструментом для рендерингу відео, створення тривимірних моделей, а також для енкодингу та декодингу відео.

Зокрема, геймерська спільнота вимагає все більше від графічних прискорювачів, щоб забезпечити найвищий рівень графічної якості та продуктивності під час гри. Такі технології також знайшли застосування у сферах відеопродукції, від створення вражаючих візуальних ефектів до оптимізації часу обробки великих обсягів даних.

Тому вивчення та розробка моделей для прогнозування обсягу пам'яті графічного процесора стає надзвичайно актуальною та важливою задачею в контексті сучасних технологічних вимог. Розуміння цього аспекту дозволить оптимізувати використання графічних прискорювачів у різних сферах діяльності та підвищити ефективність їх використання.

РОЗДІЛ 1
ОГЛЯД АРХІТЕКТУРИ ГРАФІЧНИХ ПРОЦЕСОРІВ

1.1 Визначення актуальності роботи

Тема дипломного проекту зараз є дуже актуальною в контексті стрімкого розвитку сучасних технологій та зростання потреб у високоякісній графічній обробці. З кожним роком збільшується обсяг графічної інформації, що потребує обробки, від відеоігор та фільмів до наукових досліджень та візуалізації даних у різних сферах. Графічні прискорювачі стають необхідними складовими для забезпечення високої продуктивності та ефективності в обробці графічної інформації. Такі прискорювачі використовуються не лише в ігровій індустрії, але й у багатьох інших галузях, таких як відео монтаж, тривимірне моделювання, медицина, наукові дослідження та штучний інтелект. Тому розробка комп'ютерних моделей для прогнозування обсягу пам'яті графічного процесора має велике значення і потенціал вплинути на подальший прогрес у цій області.

Однією з основних причин, яка робить цю тему актуальною, є постійна зміна вимог до обсягу пам'яті в сучасних програмах та іграх. Завдяки новим технологіям у галузі віртуальної реальності, штучного інтелекту, обробки великих обсягів даних тощо, сучасні програми вимагають все більше ресурсів від графічних процесорів. Тому важливо мати засоби для прогнозування того, який обсяг пам'яті буде необхідний для ефективної роботи програм або ігор у майбутньому.

Крім того, індустрія комп'ютерних ігор є однією з основних сфер використання графічних процесорів, і вона відома своєю постійною потребою в нових технологіях та ресурсах. Розробники ігор постійно шукають способи підвищення якості графіки та реалізму, що вимагає більш потужних графічних процесорів з більшим обсягом пам'яті. Прогнозування цих вимог допоможе розробникам вибирати оптимальні конфігурації обладнання та вирішувати питання щодо планування розвитку нових ігор.

Додатково, у сфері штучного інтелекту з'являються нові методи та алгоритми, які вимагають значних обчислювальних ресурсів. Використання графічних процесорів для прискорення обчислень у цій галузі також підвищує вимоги до обсягу пам'яті на графічних пристроях.

Таким чином, розробка комп'ютерної моделі прогнозування обсягу пам'яті графічного процесора має великий практичний і науковий інтерес у зв'язку зі зростаючою потребою в ефективному використанні графічних ресурсів у сучасних програмах та іграх, а також у зв'язку зі швидким розвитком нових технологій, що вимагають ще більшого обсягу пам'яті та продуктивності від графічних процесорів.

1.2 Огляд основних технологій графічних прискорювачів

Технології CUDA та OpenCL є важливим у контексті нашої дипломної роботи через те, що ці технології широко використовуються для програмного забезпечення, яке вимагає використання потужностей графічного апарату. CUDA від NVIDIA та OpenCL від AMD є основними інструментами для паралельного програмування на графічних прискорювачах. Вони надають можливість використовувати обчислювальні ресурси графічного процесора для виконання різноманітних завдань, включаючи обробку даних, штучний інтелект, наукові обчислення та інші.

CUDA (Compute Unified Device Architecture) є технологією розробки програмного забезпечення для графічних прискорювачів від NVIDIA. Вона надає програмістам можливість використовувати різні функції графічних прискорювачів для обчислення загального призначення. CUDA дозволяє програмістам використовувати потужності графічних прискорювачів для виконання складних обчислень у широкому спектрі застосувань, від наукових досліджень до штучного інтелекту.

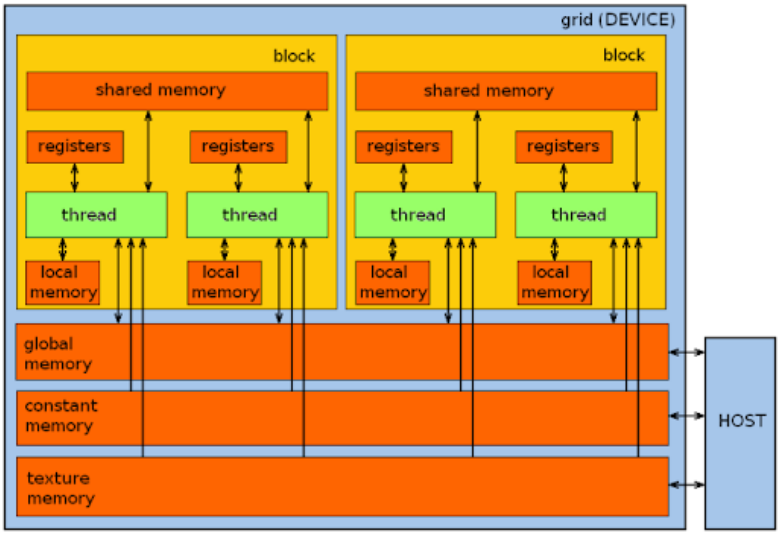

Рисунок 1.1 — Графічне відображення роботи CUDA

OpenCL (Open Computing Language) розроблений як відкритий стандарт для програмування різноманітних гетерогенних систем, включаючи графічні прискорювачі від AMD. Він надає стандартизований інтерфейс для програмування гетерогенних систем, що дозволяє розробникам писати код, який може виконуватися на різних пристроях, таких як центральні процесори, графічні прискорювачі та інші. OpenCL дозволяє ефективно використовувати потужності графічних прискорювачів для виконання різноманітних обчислень, від обробки відео до машинного навчання.

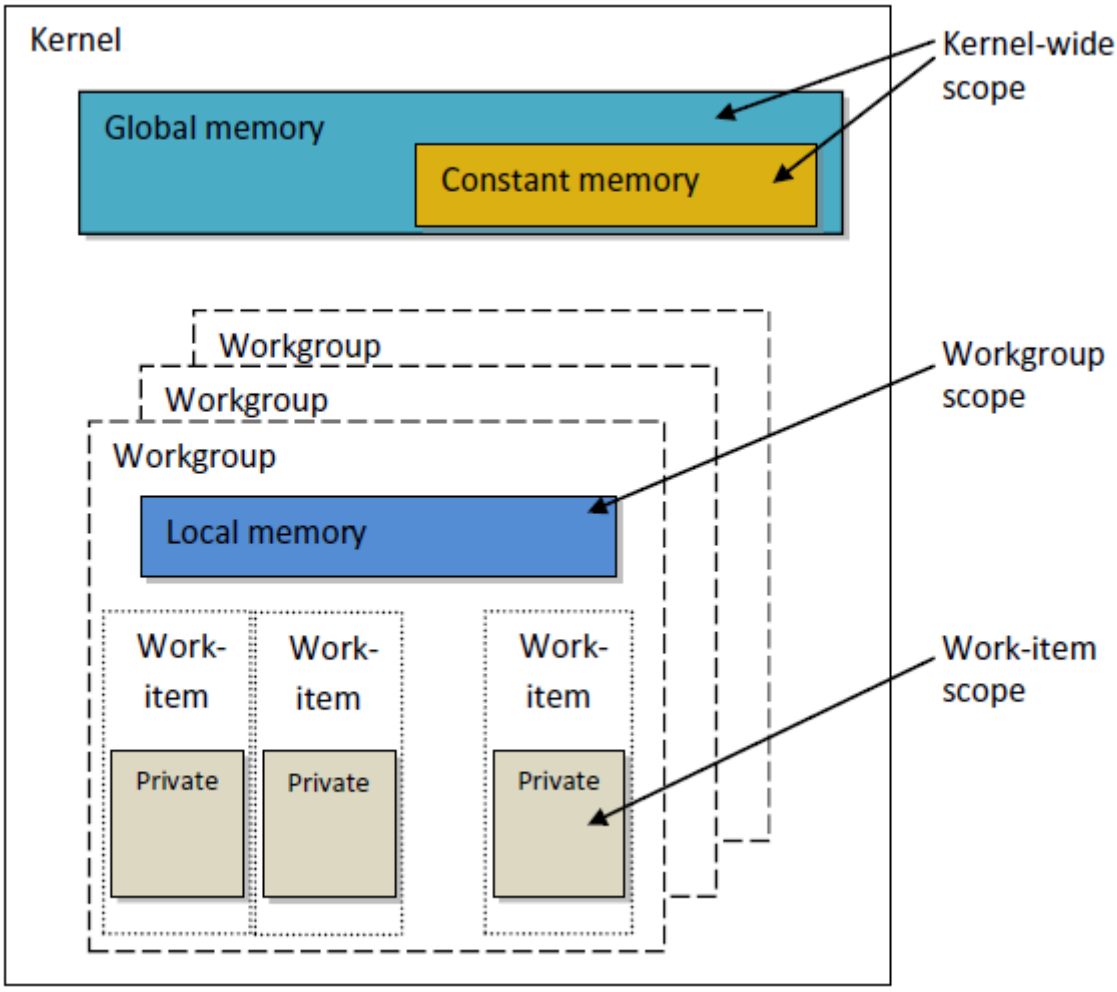

Рисунок 1.2 — Графічне відображення роботи OpenCL

1.3 Принципи роботи графічних прискорювачів

Графічні прискорювачі стали важливою складовою сучасних комп'ютерних систем не лише для відображення графіки у відеоіграх та відеоредакторах, але і для виконання різноманітних обчислювальних завдань. Розгляд розрахункових потужностей графічного прискорювача відображає його можливості у сфері паралельних обчислень та наукових досліджень.

Одним із ключових аспектів розрахункових потужностей є їхні можливості у паралельному обчисленні. Графічні прискорювачі володіють великою кількістю обчислювальних ядер, які можуть працювати паралельно над великою кількістю

завдань. Наприклад, технологія CUDA від NVIDIA та OpenCL від AMD надають інструменти для розробки програм, які використовують потужності графічного прискорювача для виконання обчислень загального призначення.

Крім того, важливо враховувати ефективність обчислень графічного прискорювача. Це включає в себе швидкість виконання обчислень та точність результатів. Графічні прискорювачі зазвичай мають високу продуктивність у виконанні паралельних завдань, а також можуть бути ефективними для обробки великих обсягів даних у реальному часі.

Графічний процесор розбиває вхідні дані на менші фрагменти і розподіляє їх між обчислювальними ядрами для паралельної обробки. Кожне обчислювальне ядро виконує певні обчислення над своїм фрагментом даних, що дозволяє прискорити обробку великої кількості інформації.

Після обробки кожен фрагмент даних відправляється на вивідний пристрій, який відповідає за відображення графіки на екрані монітора. Зазвичай це може бути монітор комп'ютера, телевізор або інший візуальний пристрій

Загальний огляд розрахункових потужностей графічного прискорювача допомагає зрозуміти його можливості у сфері наукових досліджень, обчислювального моделювання, штучного інтелекту та інших областях, де вимагається велика кількість обчислень у короткий час.

Шина пам'яті графічного прискорювача є ключовою складовою, яка впливає на його продуктивність і можливості. У цьому розділі ми розглянемо роль шини пам'яті у роботі графічних прискорювачів та її вплив на швидкість обчислень та передачу даних.

Шина пам'яті є критично важливою складовою графічних прискорювачів, відіграючи ключову роль у їхній роботі та продуктивності. Враховуючи те, що графічні прискорювачі використовуються для обробки великих обсягів інформації та відображення складних сцен у реальному часі, розгляд ролі шини пам'яті дозволяє нам краще зрозуміти, як саме відбувається обмін даними між компонентами прискорювача та системи в цілому. Шина пам'яті відповідає за

передачу даних між графічним процесором, оперативною пам'яттю та іншими компонентами системи. Вона визначає пропускну здатність та швидкість доступу до пам'яті, що впливає на швидкодію графічних обчислень та завантаження текстур та моделей.

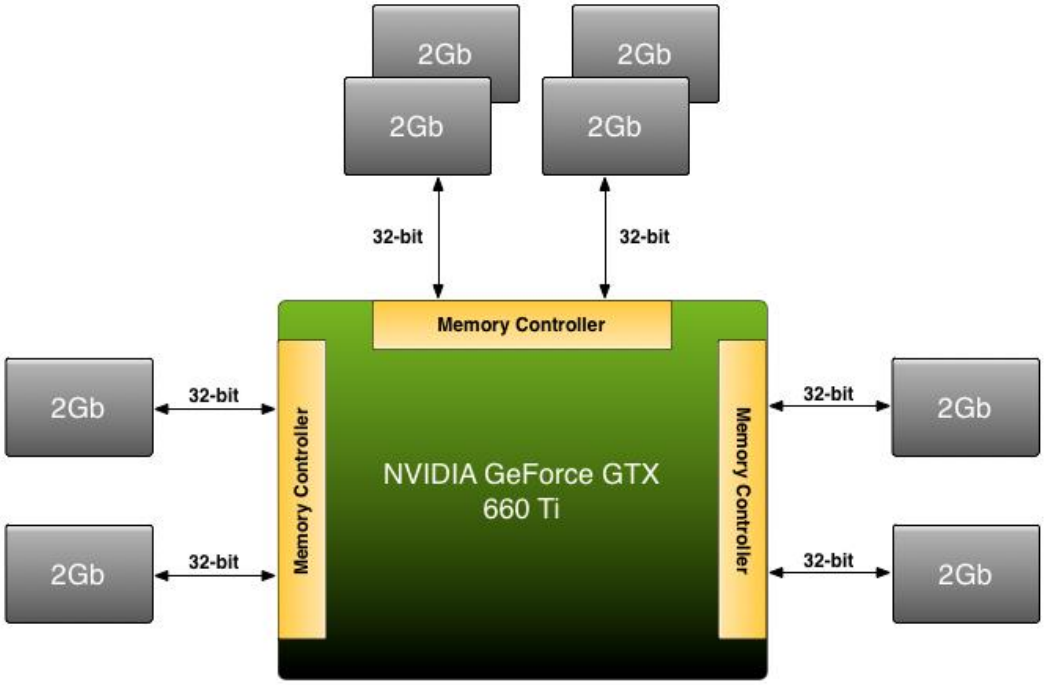

Рисунок 1.3 — Графічне відображення роботи шини пам’яті

Одним з важливих аспектів шини пам'яті є її ширина каналу передачі даних. Широкий канал дозволяє передавати більше даних одночасно, що покращує продуктивність при роботі з великими обсягами графічної інформації. Також важливою є швидкість передачі даних, яка визначає час, необхідний для завантаження текстур, обчислень ефектів освітлення та інших графічних операцій.

Крім того, шина пам'яті може підтримувати різні технології оптимізації доступу до пам'яті, такі як кешування та передбачення відвідувань. Ці технології дозволяють покращити швидкість доступу до даних та зменшити час затримки при обробці графічних запитів.

У підрозділі, ми розглянемо конкретні аспекти роботи шини пам'яті, її вплив на продуктивність графічних прискорювачів та можливість оптимізації цього компонента для підвищення ефективності роботи графічної підсистеми комп'ютера.

1.4 Огляд роботи 3D прискорювачів та їхні основні функції.

3D прискорювачі є невід'ємною частиною багатьох сучасних комп'ютерних систем, забезпечуючи високу продуктивність та різноманітні можливості для візуалізації та обробки графіки. У цьому розділі ми розглянемо популярні професійні 3D прискорювачі та їхні основні функції, а також зосередимося на значенні швидкості відеопам'яті для різних видів завдань.

На сьогоднішній день на ринку існує багато професійних 3D прискорювачів, призначених для використання у великих проектах з відеообробки, 3D моделювання, архітектурного дизайну та інших галузях. Популярні моделі включають:

— NVIDIA Quadro;

— AMD Radeon Pro для рендеру;

— AMD Instinct та Nvidia HGX AI для штучного інтелекту.

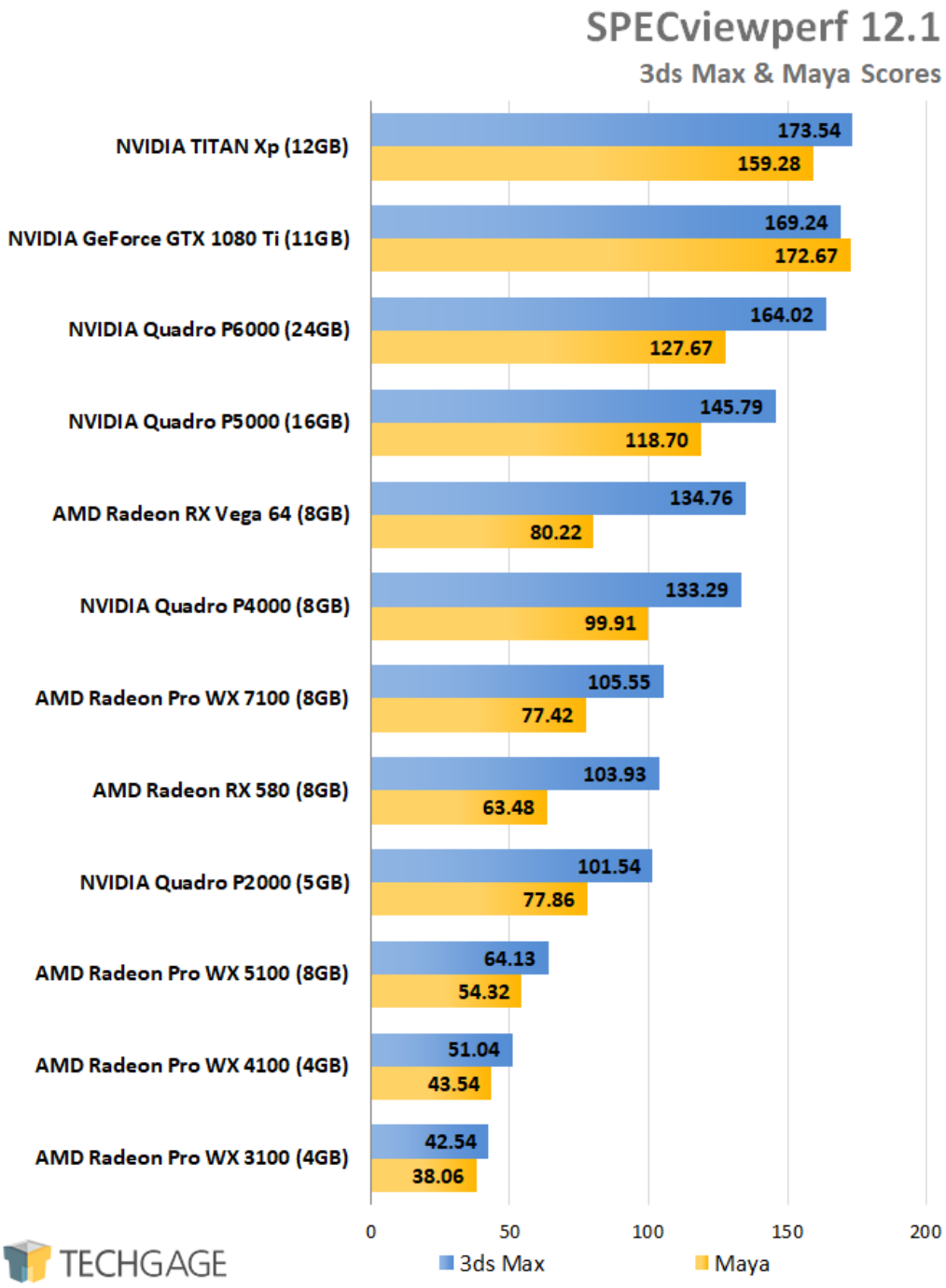

Рисунок 1.4 — Порівняння графічних прискорювачів та об’єму їх пам’яті

Ці прискорювачі відрізняються від звичних “ігрових” моделей тим, що вони мають спеціалізоване програмне забезпечення та оптимізовані для роботи з великими обсягами даних та складними алгоритмами.

Основні функції професійних 3D прискорювачів включають високопродуктивне рендеринг, підтримку різних форматів файлів та відображення, а також можливості для розробки та виконання складних обчислень у сферах наукових досліджень та інженерії. Ці прискорювачі також часто використовуються для обробки великих обсягів даних у реальному часі, що робить їх незамінними для задач штучного інтелекту та машинного навчання.

Щодо швидкості відеопам'яті, для професійних завдань, таких як 3D моделювання або великомасштабна відеообробка, важливо мати велику кількість оперативної пам'яті на прискорювачі. Це дозволяє зберігати та обробляти великі обсяги даних без затримок та забезпечує плавну роботу програм, що працюють з великими моделями чи відеофайлами. У той час як для геймерських прискорювачів може бути важливою висока швидкість пам'яті для досягнення більш високої кількості кадрів на секунду у відеоіграх, професійні користувачі зазвичай більше цінують кількість пам'яті для оптимальної роботи з великими проектами.

1.5 Огляд методів прогнозування значень цільових змінних

Прогнозування є невід’ємною складовою аналізу в усіх галузях. В залежності від специфіки та типу задачі використовують різна підходи в побудові моделей прогнозування.

Наприклад для вирішення задач в галузі економіки використовують економетричні методи. Ці методи базуються на статистичних моделях, які дозволяють вивчати та аналізувати взаємозв'язки між різними змінними в економічному контексті. Основними економетричними методами є моделі лінійної та нелинійної регресії, моделі часових рядів, які включають ARIMA (Autoregressive Integrated Moving Average), VAR (Vector Autoregression) та інші. Економетричні методи дозволяють враховувати економічну теорію та розробляти моделі, які можуть бути використані для прогнозування майбутніх значень економічних показників, оцінки впливу різних факторів на економічні процеси, а також для проведення економічного аналізу та прийняття управлінських рішень.

За останні роки чітко прослідковується стрімкий розвиток нейромережевих методів прогнозування. З розвитком продуктивності обчислювальної техніки стала можлива побудова високопродуктивних штучних нейронних мереж. Одними з найпоширеніших нейромережевих методів прогнозування є Feedforward Neural Networks (нейронні мережі з прямим зв'язком), які складаються зі шарів нейронів, де кожен нейрон пов'язаний з нейронами попереднього і наступного шарів. Ці мережі здатні вивчати складні нелинійні залежності між вхідними та вихідними даними, що робить їх ефективними для прогнозування великих обсягів даних. Проте нейромережеві методи прогнозування накладають свої обмеження на розробника. Нейромережеві методи прогнозування вимагають великої кількості даних для навчання та оптимізації моделі, а також ефективних обчислювальних ресурсів для навчання складних мереж. Однак, вони дозволяють моделювати складні залежності в даних і досягати високих результатів у прогнозуванні, що робить їх популярними в багатьох галузях, таких як фінанси, маркетинг, медицина та інші. Враховуючи, що на ринку графічних прискорювачів не представлено достатньої кількості моделей для ефективного навчання нейромережної моделі прогнозування доцільним буде використати регресійний метод.

Регресійні методи є потужним інструментом аналізу даних, спрямованим на моделювання залежностей між змінними та прогнозування значень цільових змінних на основі інших вхідних змінних. Основними видами регресійних моделей є лінійна регресія, множинна лінійна регресія, логістична регресія та нелінійні регресійні моделі.

Логістична регресія є методом, який використовується для розв'язання задачі бінарної класифікації, де цільова змінна приймає значення "так" або "ні". Основна ідея полягає в тому, щоб прогнозувати ймовірність належності об'єкта до певного класу шляхом застосування логістичної функції до лінійної комбінації пояснювальних змінних.

Модель логістичної регресії виглядає наступним чином:

$$f(x) = \frac{1}{1 + e^{-(\beta 0 + \beta 1 x 1 + \cdots + \beta p x p)}} \quad (1.1)$$

Тут $f(x)$ - ймовірність того, що об'єкт з вхідними ознаками x=(x1,x2,…,xf)x=(x1,x2,…,xf) належить до класу "так", а β0,β1,…,βpβ0,β1,…,βp - параметри моделі, що підлягають оцінці.

Оцінка параметрів моделі здійснюється шляхом максимізації функції правдоподібності на основі навчальних даних. Для цього часто використовується метод максимальної правдоподібності або метод градієнтного спуску. Після отримання оцінок параметрів моделі, їх можна використовувати для прогнозування ймовірності належності нових об'єктів до класу "так" або "ні".

Логістична регресія широко застосовується в медичній діагностиці, фінансовому аналізі, маркетингових дослідженнях та інших галузях, де необхідно прогнозувати ймовірність відбуття події на основі вхідних ознак. Цей метод є ефективним інструментом для класифікації, особливо коли дані мають лінійно роздільні класи.

Враховуючі особливості предметної області доцільним буде використати множинну лінійно-регресійну модель прогнозування.

Множинна лінійна регресія є розширенням простої лінійної регресії і використовується для моделювання залежностей між цільовою змінною і багатьма пояснювальними змінними. Цей метод дозволяє враховувати одночасно вплив декількох факторів на цільову змінну, що часто більш реалістично відображає складність реальних ситуацій.

Рівняння множинної лінійної регресії регресії наведено нижче:

$$Y = \beta 0 + \beta 1 * X1 + \beta 2 * X2 + \ldots + \beta n * Xn + \varepsilon \quad (1.2)$$

де Y – залежна змінна;

β0 – константа;

β1, β2, ..., βn – коефіцієнти регресії;

X1, X2, ..., Xn – незалежні змінні;

ε – випадкова похибка

Множинна лінійна регресія дозволяє враховувати одночасно багато факторів, що впливають на цільову змінну, що робить цей метод корисним у багатьох галузях, включаючи економіку, соціологію, маркетинг, фінанси та інші. Важливо проводити аналіз статистичної значущості параметрів моделі та враховувати можливі обмеження та припущення при застосуванні множинної лінійної регресії для прогнозування і пояснення взаємозв'язків у даних.

Висновки по розділу 1

Метою даної роботи є розробка комп’ютерної моделі прогнозування обсягу пам’яті графічного процесора в залежності від його технічних характеристик.

Для досягнення поставленої мети необхідно реалізувати такі **завдання**:

- здійснити порівняльний аналіз методів моделювання залежностей;
- розробити математичну модель для рішення задачі прогнозування о обсягу пам’яті графічного процесора за допомогою множинної регресії;
- розробити програмно-алгоритмічну модель прогнозування;
- провести тестування і аналіз отриманих результатів.

Об’єктом дослідження є прогнозування характеристик комп’ютерних систем за допомогою методів машинного навчання.

Предметом дослідження є регресійні методи моделювання та прогнозу.

Практична значимість результатів полягає у використанні розробленої моделі прогнозування в процесі аналізу ринку відеокарт.

РОЗДІЛ 2
МЕТОДИ МНОЖИННОЇ ЛІНІЙНОЇ РЕГРЕСІЇ ДЛЯ ПРОГНОЗУВАННЯ ПАРАМЕТРІВ КОМП'ЮТЕРНИХ СИСТЕМ

2.1 Постановка задачі та визначення мети

Основною метою кваліфікаційної роботи є розробка комп'ютерної моделі прогнозування об'єму пам'яті графічного процесора (GPU) на основі його характеристик. У ході виконання цієї роботи буде проведено детальний аналіз предметної області, що включатиме вивчення архітектури та функціональних особливостей графічних прискорювачів, а також дослідження взаємозв'язків між обсягом відеопам'яті та іншими технічними параметрами GPU.

Першим етапом реалізації даної мети буде проведення ґрунтовного аналізу предметної області. Це включає дослідження сучасних графічних процесорів, їх архітектури, функціональних можливостей та технічних характеристик. Зокрема, буде розглянуто, які характеристики впливають на обсяг відеопам'яті, такі як кількість обчислювальних ядер, тактова частота, пропускна здатність пам'яті, енергоспоживання та інші параметри. Важливим аспектом цього аналізу є розуміння того, як ці характеристики взаємодіють між собою та впливають на продуктивність і ємність відеопам'яті.

Другим етапом буде аналіз методів побудови моделей прогнозування. На цьому етапі буде вивчено різні підходи та алгоритми машинного навчання, які можуть бути використані для створення прогнозуючої моделі. Особлива увага буде приділена методам регресії, які дозволяють моделювати залежність цільової змінної (об'єму відеопам'яті) від набору вхідних характеристик GPU. Серед розглянутих методів будуть метод найменших квадратів (OLS), методи регуляризації (Lasso, Ridge), а також більш складні методи, такі як випадковий ліс і градієнтний бустинг. Метою цього етапу є вибір оптимального методу, який забезпечить високу точність і надійність прогнозів.

Після вибору методу буде здійснено розробку комп’ютерної моделі. Цей процес включає збір та підготовку даних, вибір необхідних характеристик, налаштування моделі та її тренування на навчальних даних. Важливою частиною цього етапу є забезпечення якості даних, що використовуються для тренування моделі, оскільки якість вхідних даних безпосередньо впливає на точність прогнозів. Буде розглянуто питання попередньої обробки даних, включаючи нормалізацію, обробку пропущених значень та усунення мультиколінеарності між характеристиками.

Після розробки моделі буде проведено її тестування та оцінка якості. На цьому етапі модель буде протестовано на тестових даних, які не використовувалися під час тренування, для оцінки її здатності робити точні прогнози на нових даних. Будуть використані різні метрики якості, такі як середньоквадратична помилка (MSE), коефіцієнт детермінації (R^2) та інші, для оцінки ефективності моделі. Також буде проведено аналіз помилок для виявлення можливих недоліків моделі та шляхів їх усунення.

Завершальним етапом роботи буде документування отриманих результатів та розробка рекомендацій щодо застосування розробленої моделі у практичних умовах. Буде розглянуто, як модель може бути інтегрована у системи планування та оптимізації ресурсів для GPU, а також можливі напрями подальшого вдосконалення моделі для підвищення її точності та надійності.

Таким чином, кваліфікаційна робота не лише спрямована на розробку моделі прогнозування об'єму пам'яті графічного процесора, але й на забезпечення комплексного підходу до аналізу та вирішення проблеми, що включає вивчення теоретичних аспектів, практичну реалізацію моделі та її оцінку, що дозволить досягти високої ефективності прогнозування та практичної цінності отриманих результатів.

2.2 Вибір методів реалізації

Побудова регресійних моделей є ключовим етапом аналізу даних і прогнозування величини цільової змінної на основі інших змінних. Регресійні моделі дозволяють вивчати залежності між змінними та встановлювати паттерни у даних, що є фундаментальним для багатьох галузей науки та бізнесу. Основне завдання при побудові регресійних моделей полягає у визначенні оптимальних параметрів моделі, які найкращим чином описують взаємозв'язки між змінними. Для цього використовуються різні методи, такі як метод найменших квадратів (OLS), метод максимальної правдоподібності, методи регуляризації (наприклад, Lasso або Ridge), а також методи оптимізації, які дозволяють знаходити оптимальні значення параметрів моделі.

Правильний вибір методу побудови регресійної моделі залежить від природи даних, кількості та типів змінних, а також поставленої задачі прогнозування чи аналізу. Наприклад, для задач з високою кількістю змінних може бути доцільним використання методів регуляризації, які допомагають уникнути перенавчання і зменшують складність моделі. Крім того, важливо проводити аналіз якості моделі, враховуючи статистичну значущість параметрів, адекватність моделі і можливість узагальнення результатів на нові дані.

Один із найпоширеніших і ефективних підходів до побудови регресійних моделей - це метод найменших квадратів (OLS). Цей метод спрямований на знаходження параметрів моделі, які мінімізують суму квадратів відхилень між спостереженими значеннями цільової змінної і значеннями, передбаченими моделлю. Основна ідея полягає в тому, щоб знайти такі значення параметрів (коефіцієнтів), які роблять прогнози моделі якомога більш точними відносно реальних даних.

Метод найменших квадратів широко використовується в статистиці та машинному навчанні через його простоту та ефективність. Він дозволяє отримати оптимальні оцінки параметрів моделі і провести інференцію щодо впливу пояснювальних змінних на цільову змінну. При застосуванні методу найменших

квадратів важливо враховувати передумови моделі, такі як нормальність залишкових (випадкових) складових і відсутність мультиколінеарності між пояснювальними змінними. Відповідність цим передумовам забезпечує точність і надійність оцінок параметрів, що є критично важливим для адекватної інтерпретації результатів.

Крім методу найменших квадратів, що є ефективним для побудови лінійних моделей, існують інші потужні методи, такі як метод випадкового лісу. Метод випадкового лісу є одним з потужних методів машинного навчання, який використовується для розв'язання задач як регресії, так і класифікації. Він базується на ідеї ансамблю, де декілька дерев прийняття рішень об'єднуються для отримання більш точного та стійкого прогнозу. Кожне дерево випадкового лісу навчається на випадковій підмножині даних та випадковому підмножині ознак, що дозволяє зменшити велику дисперсію, характерну для одного дерева прийняття рішень. Такий підхід дозволяє покращити якість прогнозів та зменшити ризик перенавчання моделі.

Також варто приділити увагу ще одному потужному методу побудови моделей, а саме логістичній регресії. Логістична регресія використовується для моделювання залежності між дискретною цільовою змінною та набором пояснювальних змінних. Цей метод є особливо ефективним у задачах класифікації і дозволяє отримати ймовірнісні оцінки належності спостережень до різних класів.

У порівнянні з іншими методами, такими як випадковий ліс або логістична регресія, метод OLS може бути менш складним у використанні та інтерпретації, особливо коли маємо справу зі звичайними лінійними залежностями між змінними. Враховуючи специфіку задачі прогнозування об'єму відеопам'яті, де важливо чітко інтерпретувати коефіцієнти моделі, метод OLS може бути оптимальним вибором для досягнення необхідної точності й прозорості результатів.

Таким чином, вибір методів реалізації регресійних моделей є комплексним процесом, що вимагає врахування багатьох факторів. Ретельний аналіз даних, визначення оптимального методу побудови моделі, а також постійний контроль

якості моделі є ключовими елементами для досягнення успішних результатів у задачах прогнозування та аналізу даних.

2.3 Вибір технології комп’ютерної реалізації

Побудова регресійних моделей включає використання різних технологій та інструментів, які допомагають аналізувати дані та прогнозувати значення цільових змінних на основі інших змінних. Однією з найпоширеніших і найбільш ефективних технологій для побудови регресійних моделей є використання мов програмування, таких як Python, разом з відповідними бібліотеками та інструментами для аналізу даних та машинного навчання. Python має широке сприйняття у спільноті дослідників і аналітиків через свою простоту використання, багату екосистему бібліотек та інструментів, а також швидкість розробки моделей.

Python є однією з найпопулярніших мов програмування для задач аналізу даних і машинного навчання завдяки своїй гнучкості і можливостям розширення. Вона підтримує багато бібліотек, таких як NumPy, pandas, scikit-learn, TensorFlow, і Keras, які надають потужні інструменти для обробки даних, візуалізації, та побудови і тренування моделей машинного навчання. Бібліотека scikit-learn, наприклад, є однією з найвідоміших і широко використовуваних бібліотек для побудови регресійних моделей, оскільки вона містить велику кількість реалізацій різних алгоритмів машинного навчання, включаючи лінійну регресію, регуляризовані регресійні методи, і ансамблеві методи.

Крім Python, існують інші технології та мови програмування, які також використовуються для побудови регресійних моделей. Наприклад, R є популярною мовою серед статистиків і дослідників, оскільки вона має багаті статистичні і аналітичні можливості. R забезпечує потужні інструменти для статистичного аналізу і візуалізації даних через бібліотеки, такі як ggplot2, dplyr, і caret. MATLAB також використовується для розв'язання складних задач у чисельних обчисленнях, включаючи побудову регресійних моделей. MATLAB надає інтегроване

середовище для математичних обчислень, що робить його корисним для задач, які вимагають високої точності і швидкості обчислень.

Однак, Python залишається однією з найбільш зручних та універсальних мов програмування для побудови регресійних моделей завдяки своїй гнучкості, розширюваності та активному співтовариству, що постійно розробляє нові інструменти та рішення для аналізу даних. Python дозволяє легко виконувати завдання від обробки даних та візуалізації до побудови складних моделей машинного навчання, включаючи регресійні моделі, що робить його чудовим вибором для багатьох досліджень та застосувань у галузі аналітики та прогнозування.

Використання Python для побудови регресійних моделей забезпечує кілька важливих переваг. По-перше, це висока продуктивність розробки завдяки простому синтаксису мови та великій кількості доступних бібліотек. По-друге, Python підтримує інтерактивні середовища, такі як Jupyter Notebook, що дозволяє дослідникам і аналітикам зручно виконувати аналіз даних і візуалізацію результатів в режимі реального часу. По-третє, велика спільнота користувачів Python сприяє швидкому вирішенню проблем та постійному оновленню інструментів.

Окрім зазначених мов програмування, для побудови регресійних моделей можуть використовуватися спеціалізовані програмні платформи та інструменти. Наприклад, платформи для автоматизованого машинного навчання (AutoML), такі як H2O.ai, Google Cloud AutoML, і Azure Machine Learning, дозволяють автоматизувати процес побудови моделей, включаючи вибір алгоритмів, налаштування параметрів та оцінку якості моделей. Ці платформи надають користувачам можливість швидко і ефективно створювати високоякісні моделі без глибоких знань в області програмування і машинного навчання.

Таким чином, вибір технології комп’ютерної реалізації для побудови регресійних моделей є критичним етапом, який впливає на ефективність та точність аналізу даних. Python, завдяки своїй універсальності, гнучкості та широкому

спектру інструментів, є однією з найкращих мов програмування для реалізації таких моделей. Проте, залежно від специфіки задачі та вимог до аналізу, можуть бути використані й інші мови та інструменти, такі як R, MATLAB, або платформи AutoML, що забезпечують високу продуктивність і точність результатів.

Висновки по розділу 2

В ході виконання другого розділу було проаналізовано методи побудови регресійних моделей, визначено їх переваги та обрано оптимальний підхід для вирішення задачі. В нашому випадку було обрано метод найменших квадратів за його простоту в використанні зі звичайними лінійними залежностями.

Інструментом реалізації було обрано мову програмування Python з бібліотекою pandas, адже це найпопулярніше рішення для побудови нейронних моделей прогнозування.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ МОДЕЛІ ПРОГНОЗУВАННЯ ОБСЯГУ ПАМ'ЯТІ ГРАФІЧНОГО ПРОЦЕСОРА

3.1 Реалізація обробки даних

У цьому пункті ми розглянемо кожен з етапів розробки програмного коду для обробки даних у контексті побудови моделі прогнозування обсягу пам'яті графічного процесора.

3.1.1 Зчитування даних з Excel файлу за допомогою бібліотеки Pandas:

Для початку необхідно встановити бібліотеку Pandas, яка дозволить нам працювати з даними у форматі Excel. Далі, за допомогою функції read_excel ми зможемо зчитати дані з Excel файлу і завантажити їх у об'єкт DataFrame.

```
import pandas as pd
# Зчитування даних з Excel файлу
data = pd.read_excel('input.xlsx')
```

Після виконання цього коду, дані з Excel файлу будуть завантажені в об'єкт DataFrame під назвою data, який ми будемо використовувати для подальшої обробки та моделювання.

3.1.2 Визначення вхідних та вихідних змінних для моделі:

Після завантаження даних, необхідно визначити, які колонки (змінні) будуть використовуватись як вхідні (предиктори), а які - як вихідна змінна (цільова).

```
# Визначення вхідних та вихідних змінних
X = data[['Shader Cores', 'Memory Bus Width', 'Memory Bandwidth']]
y = data['Video Memory Size']
```

У цьому прикладі колонки 'Shader Cores', 'Memory Bus Width' і 'Memory Bandwidth' визначені як вхідні змінні, а 'Video Memory Size' - як вихідна змінна.

3.1.3 Розділення загального набору даних на навчальний та тестовий для подальшого моделювання

Перед тим як навчити модель, ми повинні розділити наші дані на дві частини: навчальний набір, на якому модель буде навчатись, і тестовий набір, на якому ми перевіримо її ефективність.

```
from sklearn.model_selection import train_test_split
# Розділення на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Тут test_size=0.2 вказує, що 20% даних будуть використовуватись для тестування, а решта - для навчання моделі. random_state=42 встановлює початкове значення для генерації випадкових чисел, щоб кожного разу отримувати однаковий результат при розділенні даних.

3.2 Реалізація моделі лінійної регресії

Реалізація моделі лінійної регресії полягає в побудові та навчанні моделі на вхідних даних. Для цього ми використовуємо бібліотеки Scikit-learn та NumPy, які надають зручні та потужні інструменти для роботи зі статистичними моделями та аналізу даних.

Почнемо з імпорту необхідних бібліотек:

```
import pandas as pd  # Для роботи з даними
from sklearn.linear_model import LinearRegression  # Для побудови лінійної регресії
from sklearn.metrics import mean_squared_error  # Для оцінки точності моделі
from math import sqrt  # Для обчислення квадратного кореня
from sklearn.model_selection import train_test_split  # Для розділення даних на навчальний та тестовий набори
import matplotlib.pyplot as plt  # Для побудови графіків
import numpy as np  # Для роботи з числами та масивами
```

Зчитування даних та підготовка їх до аналізу є першим кроком у реалізації моделі. Ми використовуємо бібліотеку Pandas для завантаження даних з Excel файлу та подальшої роботи з ними:

```
# Зчитування даних з Excel файлу
data = pd.read_excel('/content/predictors.xlsx')
```

Після завантаження даних ми визначаємо вхідні та вихідні змінні для моделі:

```
# Визначення вхідних (предикторів) та вихідної (цільової) змінних
X = data[['Shader Cores', 'Memory Bus Width', 'Memory Bandwidth']]
y = data['Video Memory Size']
```

Потім ми розділяємо наші дані на навчальний та тестовий набори для подальшого моделювання:

```
# Розділення даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Тепер ми готові побудувати модель лінійної регресії та навчити її на навчальних даних. Використовуючи клас LinearRegression з бібліотеки Scikit-learn, ми створюємо об'єкт моделі та викликаємо метод fit() для навчання моделі:

```
# Побудова моделі лінійної регресії
model = LinearRegression()
model.fit(X_train, y_train)
```

Далі ми можемо отримати коефіцієнти моделі та оцінити її точність на тестових даних. Коефіцієнти моделі вказують на вагу кожного вхідного предиктора, а оцінка точності дає нам уявлення про те, наскільки добре модель працює на незалежних тестових даних:

```
# Виведення коефіцієнтів моделі
print('Коефіцієнти моделі:')
print('Coefficients:', model.coef)
```

Після побудови моделі та оцінки її точності, ми можемо перевірити, наскільки добре прогнозується вихідна змінна (обсяг відеопам'яті) за допомогою лінійної регресії. Для цього побудуємо графік, де порівняємо фактичні значення обсягу відеопам'яті (y_test) з прогнозованими значеннями (y_pred):

```
# Побудова графіку прогнозів та реальних значень на тестовому наборі
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], linestyle='--', color='red', label='Actual')
plt.title('Actual vs. Predicted Video Memory Size (Test Set)')
plt.xlabel('Actual Video Memory Size (GB)')
plt.ylabel('Predicted Video Memory Size (GB)')
plt.legend()
plt.show()
```

Цей графік дозволяє нам візуально порівняти прогнозовані значення з фактичними. Якщо модель працює ефективно, всі точки будуть розташовані вздовж прямої лінії з підйомом 45 градусів, що відповідає ідеальній узгодженості прогнозованих та фактичних значень.

На завершення, ми можемо використати побудовану модель для прогнозування обсягу відеопам'яті для нових даних. Для цього ми створюємо DataFrame з новими значеннями предикторів та викликаємо метод predict() моделі:

```
# Приклад прогнозів для нових даних
new_data = pd.DataFrame({
    'Shader Cores': [10752, 10496, 8704],
    'Memory Bus Width': [384, 384, 384],
    'Memory Bandwidth': [1036, 936, 760]
})
predicted_memory_new = model.predict(new_data)
```

Потім ми можемо візуалізувати ці прогнози, порівнявши їх з вхідними даними:

```
# Виведення прогнозів для нових даних і відображення на графіку
порівняння з вхідними даними
plt.figure(figsize=(10, 6))
# Додавання вхідних даних на графік
plt.scatter(X['Memory Bandwidth'], y, color='black', label='Input
Data')
# Додавання прогнозів для нових даних на графік
plt.scatter(new_data['Memory Bandwidth'], predicted_memory_new,
color='green', label='Predicted New Data')

plt.title('Video Memory Size Predictions')
plt.xlabel('Memory Bandwidth')
plt.ylabel('Video Memory Size (GB)')
plt.legend()
plt.show()
```

Це дозволяє нам бачити, які прогнозовані значення відеопам'яті ми отримуємо для нових даних з використанням побудованої моделі лінійної регресії.

3.3 Візуалізація результатів та аналіз прогнозів

Для візуалізації результатів та аналізу прогнозів спочатку побудуємо графік, що порівнює фактичні та прогнозовані значення обсягу пам'яті графічного процесора на тестовому наборі даних:

```
    # Побудова графіка для порівняння фактичних та прогнозованих значень обсягу пам'яті графічного процесора
    plt.figure(figsize=(10, 6))
    plt.scatter(y_test, y_pred, color='blue', label='Predicted')
    plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], linestyle='--', color='red', label='Actual')
    plt.title('Actual vs. Predicted Video Memory Size (Test Set)')
    plt.xlabel('Actual Video Memory Size (GB)')
    plt.ylabel('Predicted Video Memory Size (GB)')
    plt.legend()
    plt.show()
```

Цей графік дозволяє нам візуально оцінити ефективність прогнозування моделі. Якщо всі точки розташовані вздовж прямої лінії з підйомом 45 градусів, це свідчить про те, що прогнози моделі добре узгоджуються з фактичними значеннями.

Після цього ми можемо відобразити прогнози для нових даних на тому ж графіку порівняння з вхідними даними:

```
    # Відображення прогнозів для нових даних на графіку порівняння з вхідними даними
    plt.figure(figsize=(10, 6))
    plt.scatter(X['Memory Bandwidth'], y, color='black', label='Input Data')
    plt.scatter(new_data['Memory Bandwidth'], predicted_memory_new, color='green', label='Predicted New Data')
    plt.title('Video Memory Size Predictions')
    plt.xlabel('Memory Bandwidth')
    plt.ylabel('Video Memory Size (GB)')
    plt.legend()
    plt.show()
```

Цей графік дозволяє нам порівняти прогнозовані значення для нових даних з вхідними даними, що може допомогти в оцінці адекватності моделі для нових спостережень.

В заключенні, проведемо аналіз результатів та виявимо можливі шляхи вдосконалення моделі на основі візуальних та числових даних. Для цього порівняємо прогнозовані значення з фактичними за допомогою показників точності моделі, таких як середньоквадратична помилка (RMSE). Враховуючи ці

результати, можна розглянути можливі покращення моделі, наприклад, врахування додаткових предикторів або застосування інших методів моделювання.

Для аналізу точності моделі можемо використати середньоквадратичну помилку (RMSE), яка вимірює середньоквадратичне відхилення між фактичними та прогнозованими значеннями. Чим менше значення RMSE, тим краще модель узгоджується з даними.

```
    # Оцінка точності моделі за допомогою середньоквадратичної помилки (RMSE)
    rmse = sqrt(mean_squared_error(y_test, y_pred))
    print(f'Root Mean Squared Error (RMSE) на тестовому наборі: {rmse:.2f} GB')
```

Аналізуючи значення RMSE, ми можемо зробити висновки про ефективність моделі. Наприклад, якщо RMSE низьке, це означає, що модель достатньо точно прогнозує значення, що підтверджує його використання для прогнозування обсягу пам'яті графічного процесора.

Звісно, важливо також враховувати візуальні результати аналізу. Графіки порівняння фактичних та прогнозованих значень дозволяють швидко оцінити здатність моделі до прогнозування на реальних даних. Аналізуючи ці графіки, можна виявити, чи є які-небудь систематичні відхилення між фактичними та прогнозованими значеннями, а також виявити зони найбільшої невідповідності.

На основі цього аналізу можна здійснити подальші кроки для вдосконалення моделі. Наприклад, можна спробувати додаткові функції або функціональні форми, врахувати додаткові фактори, які можуть впливати на обсяг пам'яті графічного процесора, або застосувати інші методи моделювання, які краще підходять для конкретного набору даних.

В цілому, аналіз результатів моделювання допомагає зрозуміти її сильні та слабкі сторони, а також надає відомості про те, як можна поліпшити її ефективність.

Висновки по розділу 3

В ході виконання практичного розділу було виконано практичну реалізацію комп’ютерної моделі прогнозування обсягу пам’яті графічного процесора з допомогою мови Python та бібліотеки pandas. Детально описано кожен етап роботи програми, що реалізує модель прогнозування.

РОЗДІЛ 4
АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ ТА ОЦІНКА ЯКОСТІ МОДЕЛІ

4.1 Дослідження вхідних змінних

Для побудови ефективної моделі прогнозування був використаний набір даних, який містить інформацію про кількість обчислювальних ядер (Shader Cores), ширину шини пам'яті (Memory Bus Width) та пропускну здатність пам'яті (Memory Bandwidth). Попереднє дослідження цих змінних дозволило визначити їх вплив на обсяг відеопам'яті графічного процесора. Нижче наведено фрагмент вхідних даних з відомостями про кожен графічний процесор.

Таблиця 4.1 - Фрагмент вхідних даних

Графічний процесор	Shader Cores	Memory Bus Width	Memory Bandwidth	Video Memory Size
GPU_1	3072	192	336	4
GPU_2	3840	256	448	8
GPU_3	4352	352	484	12

Аналіз цих даних допоміг виявити взаємозв'язок між характеристиками графічного процесора та обсягом його відеопам'яті. На основі цього аналізу було вирішено використовувати лінійну регресію для побудови моделі прогнозування.

4.2 Вибір предикторів

Для побудови ефективної моделі прогнозування прогнозування обсягу відео пам’яті необхідно визначити список змінних які будуть предикторами. Продуктивність усіх графічних прискорювачів можна поділити на два напрямки, а саме продуктивність графічного процесору та продуктивність підсистеми пам’яті. Розробники здебільшого притримуються балансу в розвитку двох цих напрямків. Тому буде доцільним опиратися в прогнозах на перші та другі показники.

Продуктивність відеокарти в аспекті потужності графічного процесора можна оцінити за кількістю шейдерних блоків. Кількість шейдерів у відеокарті визначає обчислювальну потужність і здатність пристрою обробляти графічні

операції. Більша кількість шейдерів дозволяє виконувати складніші обчислення, які використовуються в сучасних іграх та програмах для обробки графіки. Таким чином, кількість шейдерів може впливати на потребу в обсязі відеопам'яті для оптимальної роботи пристрою. З іншого боку потужність підсистеми пам’яті можна відслідкувати в пропускній спроможності пам’яті та шириною її шини.

Пропускна спроможність пам'яті і ширина шини пам'яті також впливають на продуктивність графічного прискорювача. Ці параметри визначають швидкість передачі даних між пам'яттю і графічним процесором. Висока пропускна спроможність і широкий канал пам'яті дозволяють ефективно використовувати великі обсяги даних, що є важливим для оптимальної роботи відеокарти в сучасних графічних програмах і іграх.

Отже, список ключових предикторів для прогнозування обсягу відеопам'яті графічного процесору включатиме такі важливі характеристики:

1. Кількість шейдерів (Shader Cores);
2. Об'єм відеопам'яті (Video Memory Size);
3. Пропускна спроможність пам'яті (Memory Bandwidth);
4. Ширина шини пам'яті (Memory Bus Width);

Ці характеристики визначають обчислювальні можливості та ефективність використання графічного процесору при обробці графічних операцій. Більші значення цих параметрів вказують на більшу продуктивність і здатність до роботи з великими обсягами даних. Прогнозування обсягу відеопам'яті на основі цих предикторів допоможе визначити оптимальну відеокарту для виконання конкретних завдань, таких як відеоігри або обробка графіки.

4.3 Побудова комп’ютерної моделі

Для побудови комп’ютерної моделі необхідно підготувати вибірку даних для навчання. Приклад навчальної вибірки наведено на рисунку 4.1

	A	B	C	D	E	F
1	Shader Co	Memory B	Memory B	Video Memory Size		
2	10752	384	1036	24		
3	10496	384	936	24		
4	8704	384	760	12		
5	8704	384	760	10		
6	5888	256	560	8		
7	5888	256	560	8		
8	4864	256	480	8		
9	3584	192	360	12		
10	5120	384	760	16		
11	5120	256	520	16		
12	3840	256	512	16		
13	4096	256	448	12		
14	3600	128	288	8		
15	3584	192	360	12		
16	2560	128	240	4		
17	2560	128	240	8		
18	3072	128	224	8		
19	2560	128	224	4		
20	4352	352	660	11		
21	4352	384	496	8		
22	2944	384	480	8		
23	3584	352	520	11		
24	2560	256	480	8		
25	2048	256	336	8		
26	1280	192	288	6		
27	2304	256	336	8		
28	2048	256	288	8		
29	2816	384	680	6		
30	2048	384	560	4		
31	1664	256	320	4		
32	2816	512	576	8		

Рисунок 4.1 — Навчальна вибірка

Зчитування даних з Excel файлу було першим кроком у процесі побудови моделі прогнозування. Використання бібліотеки Pandas дозволило ефективно завантажити дані з Excel-файлу в об'єкт DataFrame, що є основним типом даних у бібліотеці Pandas. Цей крок відображає важливість попередньої обробки даних перед подальшим аналізом і моделюванням.

Після завантаження даних було визначено вхідні та вихідні змінні для моделі. Вхідні змінні (предиктори) включали такі параметри, як кількість шейдерів, шина пам'яті та пропускна спроможність, які можуть бути корисними у прогнозуванні обсягу відеопам'яті. Вихідна змінна (цільова) була обсягом відеопам'яті, яку модель буде намагатися прогнозувати на основі цих вхідних даних.

У бібліотеці Scikit-learn для використання методу найменших квадратів в побудові лінійної регресії використовується клас LinearRegression. Цей клас автоматично здійснює оптимальну підгонку лінійної моделі до даних шляхом розв'язання задачі оптимізації за допомогою МНК(методу найменших квадратів). Модель намагається знайти оптимальні коефіцієнти (ваги) для вхідних змінних, які найкраще підлаштовуються під спостереження.

Основний принцип МНК полягає в тому, щоб знайти такі значення параметрів моделі (ваги), які мінімізують суму квадратів відхилень (різниць) між фактичними значеннями вихідної змінної і прогнозованими значеннями моделі.

На рисунку 4.1 відображено відношення між фактичними та прогнозованими значеннями з використанням лінійної регресії.

Після завантаження даних визначені вхідні та вихідні змінні, необхідні для побудови моделі прогнозування. Дослідження вхідних змінних дозволило встановити їх вплив на обсяг відеопам'яті графічного процесора. Використання методу лінійної регресії з використанням бібліотеки Scikit-learn дозволило побудувати модель, яка знаходить оптимальні параметри для прогнозування обсягу пам'яті. Навчання моделі на навчальній вибірці та її подальша перевірка на тестовій вибірці є важливими етапами в розробці ефективної моделі прогнозування.

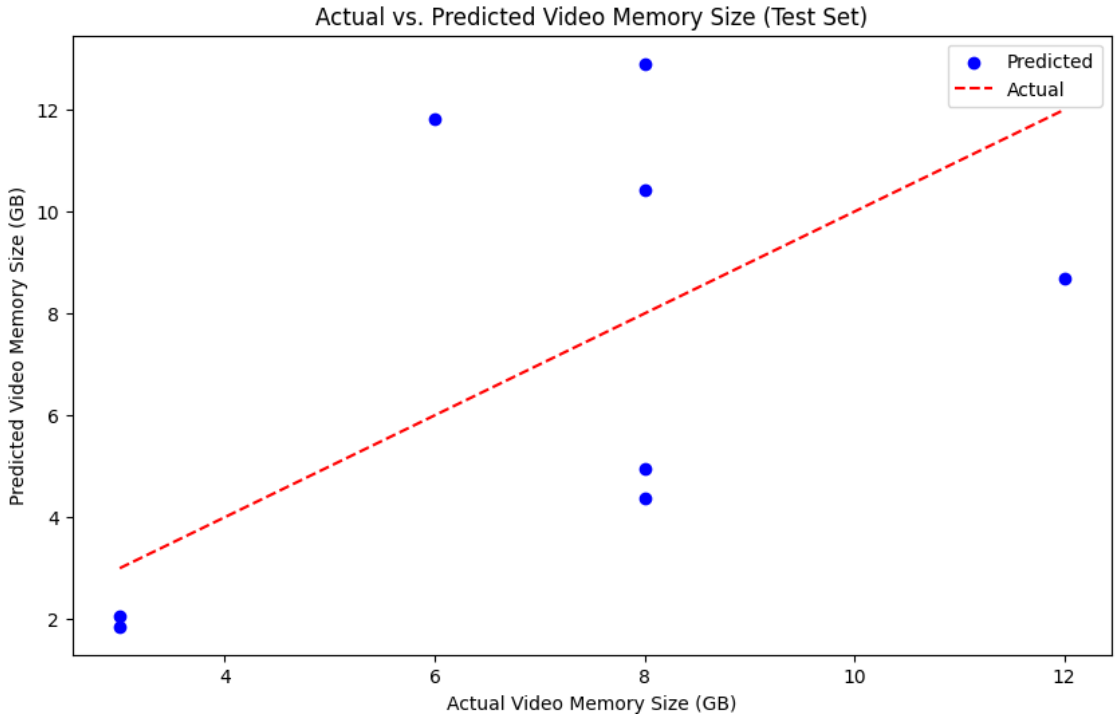

Рисунок 4.2 — Порівняння очікуваних та прогнозованих значень

Графік демонструє, як відхилення (епсілон) між фактичними та прогнозованими значеннями змінюється зі збільшенням значень пропускної здатності пам'яті графічного процесора. Чим більше значення memory bandwidth, тим менше відхилення між фактичними та прогнозованими значеннями, що вказує на кращу точність моделі прогнозування. Такий аналіз допомагає зрозуміти, які параметри найбільше впливають на точність прогнозування обсягу пам'яті графічного процесора і може вказати на можливі шляхи вдосконалення моделі.

4.4 Оцінка якості моделі

Отриманий результат моделювання необхідно перевірити на точність та дати оцінку його ефективності. У процесі побудови моделі з використанням МНК у бібліотеці Scikit-learn, основною задачею є мінімізація середньоквадратичної помилки (RMSE) або інших метрик відхилення для забезпечення найкращого підгону моделі до даних. Цей підхід є ефективним для розв'язання задач регресії,

де необхідно знайти лінійну залежність між вхідними та вихідною змінною. Оцінка коефіцієнтів моделі зображено на рисунку 4.3.

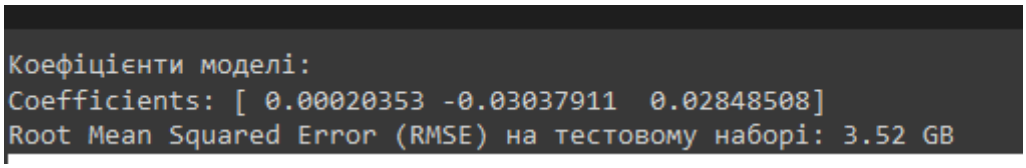

Рисунок 4.3 — Коефіцієнти впливу параметрів та середньоквадратична помилка моделі

Після оцінки ефективності моделі лінійної регресії можна зробити наступні висновки. Коефіцієнти моделі, які отрималися після підгонки до навчальних даних, вказують на залежність між вхідними характеристиками (кількість шейдерів, ширина шини пам'яті, пропускна здатність) і обсягом відеопам'яті. За цими коефіцієнтами можна зробити наступні спостереження:

- Коефіцієнт для кількості шейдерів (0.00020353) вказує на те, що збільшення кількості шейдерів слабо впливає на обсяг відеопам'яті згідно з цією моделлю.
- Коефіцієнт для шини пам'яті (-0.03037911) показує, що збільшення ширини шини пам'яті призводить до зменшення обсягу відеопам'яті згідно з моделлю.
- Коефіцієнт для пропускної здатності (0.02848508) показує, що збільшення пропускної здатності впливає на збільшення обсягу відеопам'яті згідно з моделлю.

Крім того, значення середньоквадратичної помилки (RMSE) на тестовому наборі складає 3.52 GB. Це значення оцінює точність моделі: чим нижче значення RMSE, тим краще модель прогнозує цільові значення. У даному випадку RMSE в 3.52 GB свідчить про те, що модель має помірну точність у прогнозуванні обсягу відеопам'яті на основі вхідних характеристик. Для кращого візуального

відображення результату прогнозування в порівнянні з вхідними даними відобразимо їх на графіку залежності об’єму відеопам’яті до найбільш вагомого параметру, тобто пропускної здатності рисунок 4.4.

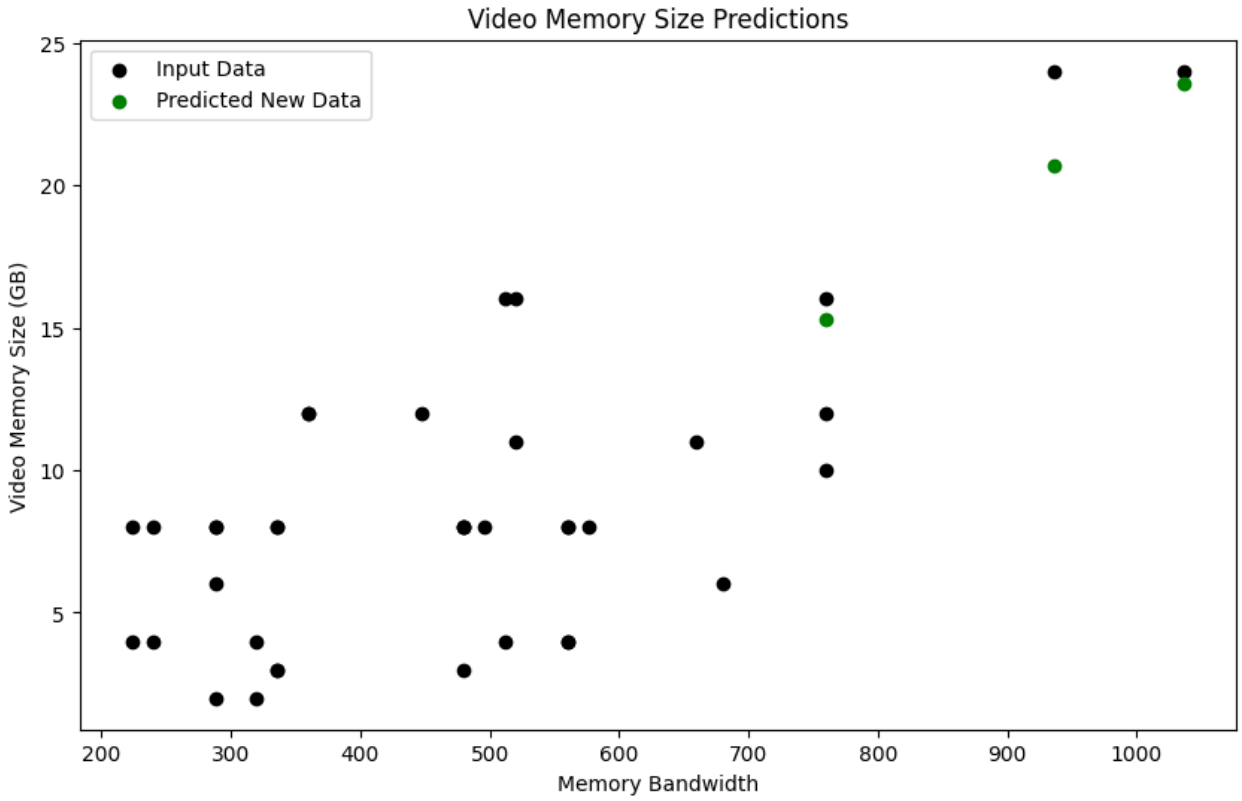

Рисунок 4.4 — Відношення об’єму пам’яті до пропускної здатності в навчальній та прогнозованій виборках

Як бачимо на зображенні зеленим відображені результати прогнозування, а навчальні дані помічені чорним.

Критерій Акаїке (AIC) є одним з методів порівняння моделей на основі їхньої адекватності та складності. Вперше запропонований японським статистиком Хіроюкі Акаїке, цей критерій зазвичай використовується для вибору найкращої моделі серед конкуруючих моделей, які оцінюють один і той же набір даних.

Основна ідея полягає в тому, що модель, яка найкраще відповідає даним, повинна мати найменший AIC. AIC об'єднує у собі як точність моделі (залежність від даних), так і складність моделі (кількість параметрів), що дозволяє збалансувати

між ними. Модель з меншим значенням AIC вважається більш адекватною для опису даних.

Формула для обчислення AIC має вигляд:

$$AIC = nlog(RMSE^2) + 2k \tag{4.1}$$

де:

n – кількість спостережень у тестовій вибірці;

RMSE – середньоквадратична помилка моделі;

k – кількість параметрів у моделі (включаючи константу).

На рисунку 4.5 зображено результат визначення критерію Акаїке (AIC).

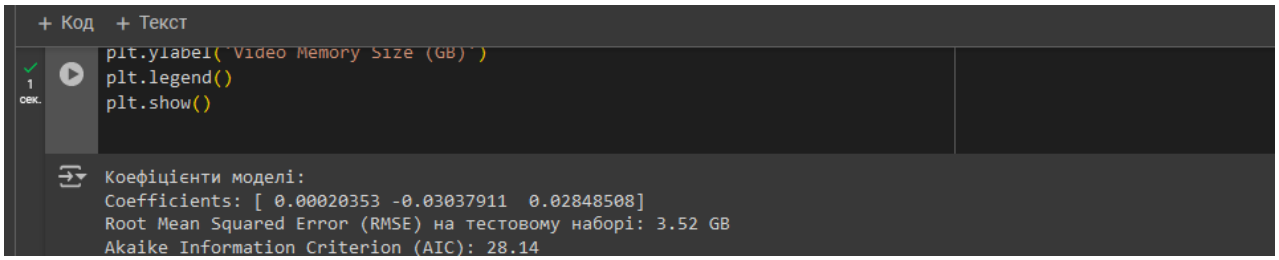

Рисунок 4.5 — Критерій Акаїке (AIC)

Високе значення AIC вказує на те, що модель неадекватна або надто складна для даних, тоді як низьке значення AIC свідчить про те, що модель є більш адекватною та простішою для опису даних. AIC може бути використаний для порівняння різних моделей та вибору найкращої серед них.

Висновки по розділу 4

В ході виконання четвертого розділу було підготовлено навчальну вибірку та протестовано роботу моделі з оцінкою її якості. Визначено коефіцієнти впливу параметрів та середньоквадратична помилка моделі, а також критерій Акаїке (AIC).

ВИСНОВКИ

В ході виконання дипломної роботи було проаналізовано предметну область для розробки комп’ютерної моделі прогнозування обсягу пам’яті графічного процесора.

В ході аналізу предметної області було визначено основні тенденції розвитку графічних процесорів, особливості їх роботи та проведено огляд методів побудови моделей прогнозування.

В проектній частині було визначено мету та задачі роботи, обрано та проаналізовано технології реалізації, а також метод комп’ютерного моделювання. Для реалізації комп’ютерної моделі прогнозування обсягу пам’яті графічного процесора було обрано метод найменших квадратів.

В практичній частині дипломної роботи проаналізували та визначили перелік змінних для побудови комп’ютерної моделі. Проведено оцінку якості прогнозування готової комп’ютерної моделі, визначено середньоквадратичну похибку та коефіцієнти вхідних даних.

В результаті виконання дипломної роботи була розроблена комп’ютерна модель прогнозування обсягу пам’яті графічного процесора, яка допоможе потенційним користувачам зацікавленим в виборі графічного процесора обрати модель з оптимальним відносно ринка об’ємом відеопам’яті.

Лістинг основних модулів розробленого програмного додатку представлено у додатку Г.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gorelick M., Ozsvald I. High Performance Python: Practical Performant Programming for Humans. O'Reilly Media, 2020.

2. Chollet F. Deep Learning with Python. Manning Publications, 2017.

3. Segaran T. Programming Collective Intelligence. O'Reilly Media, 2007.

4. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. O'Reilly Media, 2017.

5. Sanders J., Kandrot E. CUDA by Example: An Introduction to General-Purpose GPU Programming. Addison-Wesley, 2010.

6. Howse J., Minichino J. Learning OpenCV 4 Computer Vision with Python. Packt Publishing, 2020.

7. VanderPlas J. Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media, 2016.

8. Wesolowski L., Grzeszczyk M., Bryll D. Prediction of GPU Memory Requirements using Machine Learning Methods. Proceedings of the 2019 Federated Conference on Computer Science and Information Systems. 2019. Vol. 18, pp. 331-338.

9. Chetlur S., Woolley C., Vandermersch P., Cohen J., Tran J., Catanzaro B., Shelhamer E. cuDNN: Efficient Primitives for Deep Learning. arXiv:1410.0759, 2014.

10. Oliphant T. E. A guide to NumPy. Trelgol Publishing, 2006.

11. Numba: A High-Performance Python Compiler. [Електронний ресурс] - Режим доступу до ресурсу: https://numba.pydata.org/

12. Pytorch Memory Management. [Електронний ресурс] - Режим доступу до ресурсу: https://pytorch.org/tutorials/recipes/recipes/tuning_guide.html

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **бакалавр**
Галузь знань: **12 – Інформаційні технології**
Спеціальність: **123 – Комп'ютерна інженерія.**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.
«21» грудня 2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

БОГДАНОВА ЄВГЕНІЯ СЕРГІЙОВИЧА
_____
(прізвище, ім'я, по батькові студента)

1. Тема роботи **«КОМП'ЮТЕРНА МОДЕЛЬ ПРОГНОЗУВАННЯ ОБСЯГУ ПАМ'ЯТІ ГРАФІЧНОГО ПРОЦЕСОРА»**

керівник роботи Бакуменко Ніна Станіславівна, професор, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "___"_____2024року №_____

2.Строк подання студентом роботи 31 травня 2024року

3. Перелік питань, які потрібно розробити

1. Постановка задачі аналізу та прогнозування обсягу пам'яті графічного процесору.

2. Критерії оцінки ефективності системи.
3. Модель прогнозування обсягу пам’яті графічного процесора.
4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз предметної області	21.12.2023 - 25.01.2024
2	Аналіз методів побудови регресійних моделей	19.12.2023 - 2.01.2024
3	Підбір методу прогнозування обсягу пам’яті графічного процесора	2.01.2024 - 2.02.2024
4	Реалізація розробленого методу	2.01.2024 - 2.02.2024
5	Тестування та апробація комп’ютерної моделі.	3.02.2024 - 30.03.2024
6	Розробка пояснювальної записки.	31.03.2024 - 27.05.2024

5. Дата видачі завдання 21.12.2023

Студент Є. В. Богданов _____
ініціали, прізвище підпис

Керівник роботи Н.С. Бакуменко _____
ініціали, прізвище підпис

Додаток Б

Затверджую
_____
«_____»_____2024 р.

Технічне завдання
на розробку програмного виробу «Комп'ютерна модель прогнозування обсягу пам'яті графічного процесора**»**

1.	Введення	1.1. Назва: Комп'ютерна модель прогнозування обсягу пам'яті графічного процесора 1.2. Галузь застосування: статистика, комп'ютерні технології.
2.	Підстава для розробки	2.1. Навчальний план за спеціальністю 123 – Комп'ютерна інженерія 2.2. Завдання на кваліфікаційну роботу бакалавра № від 21.12.2023.
3.	Призначення розробки	3.1. Мета розробки: є розробка комп'ютерної моделі прогнозування обсягу пам'яті графічного процесора в залежності від його технічних характеристик. 3.2. Призначення розробки: розробка повинна сформувати навички створення програмних продуктів, використовуючи знання з різних дисциплін, і продемонструвати вміння формувати пакет документації, а також представляти результати виконаного проекту. Тема проекту: «Комп'ютерна модель прогнозування обсягу пам'яті графічного процесора». 3.3. Вихідні дані розробки: статистичні експериментальні дані
4.	Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик:немає 4.2. Вимоги до надійності: забезепчувати точність та достовірність визначення інформативності параметрів стану не гіршу за існуючі методи та програмне забезепечення 4.3.Вимоги до умов експлуатації: немає 4.4. Вимоги до складу і параметрів технічних засобів: звичайне обчислювальне обладнання, ПК, тощо 4.5. Вимоги до інформаційної та програмної сумісності: немає 4.6. Вимоги до маркування та упаковки: немає

<table>
<tr><td></td><td></td><td colspan="2">4.7. Вимоги до транспортування і зберігання: на звичайних носіях інформації
4.8. Спеціальні вимоги: немає.</td></tr>
<tr><td>5.</td><td>Вимоги до програмної документації</td><td colspan="2">Програмною документацією до виробу «НАЗВА ВАШОЇ РОБОТИ» вважати:
1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Методику розрахунку інформативності змінних стану (у вигляді глав 4.4 пояснювальної записки до кваліфікаційної роботи).</td></tr>
<tr><td>6.</td><td>Вимоги до техніко-економічних показників</td><td colspan="2">Програмною документацією до виробу «Метод аналізу інформативності змінних стану при діагностиці систем з використанням інформаційних критеріїв» вважати:
1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Методику розрахунку інформативності змінних стану (у вигляді глав 3.3 та 3.4 пояснювальної записки до кваліфікаційної роботи).
3) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)</td></tr>
<tr><td>7.</td><td>Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td></td><td></td><td>грудня 2023

від 25 грудня 2023
до 16 січня 2024

від 2 січня 2024
до 2 лютого 2024

від 3 січня 2024
до 30 березня 2024</td><td>Аналіз практики предметної області.

Аналіз існуючого науково-методичного апарату визначення показників інформативності стану та обгрунтування актуальності наукової задачі.

Аналіз існуючого програмного забезпечення обчислення показників інформативності

Розробка (доопрацювання) методики обчислення інформативності параметрів</td></tr>
</table>

			цільової функції, заданої алгоритмічно
		від 11 лютого 2024 до 27 травня 2024	Розробка (доопрацювання) методики обчислення інформативності параметрів цільової функції, заданої статистичною вибіркою обмеженого обсягу
		від 31 березня 2024 до 27 квітня 2024	Тестування розробленої методики на реальній статистичній вибірці (з медичної практики)
		від 1 травня 2024 до 28 травня 2024	Оформлення результатів. Написання пояснювальної записки.
		30 травня 2024	Представлення кваліфікаційного проєкту керівнику кваліфікаційної роботи та рецензенту.
8.	Порядок контролю і приймання програмного продукту (моделі)	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати в електронному вигляді в 1 примірнику.	

Виконавець
студент групи КІ- 41
БОГДАНОВ Є. С.
_____

Замовник
к.т.н., проф.
БАКУМЕНКО Н.С.
_____

Додаток В

«КОМП’ЮТЕРНА МОДЕЛЬ ПРОГНОЗУВАННЯ ОБСЯГУ ПАМ’ЯТІ ГРАФІЧНОГО ПРОЦЕСОРА»

1 Об'єкт випробувань

1. Назва програмного виробу: «КОМП’ЮТЕРНА МОДЕЛЬ ПРОГНОЗУВАННЯ ОБСЯГУ ПАМ’ЯТІ ГРАФІЧНОГО ПРОЦЕСОРА»
2. Галузь застосування : статистика, комп’ютерні технології.
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

1. працювати на основних операційних системах: Windows, Linux, MacOS;
2. вимоги до надійності;
3. передбачити захист від некоректних дій користувача;
4. сумісність з іншими програмними продуктами;

5. зменшити об’єм програмного коду необхідного для створення веб-додатків;
6. бути легко розширюваною;
7. елементи програми повинні бути ізольовані одне від одного для зменшення їх впливу на роботи програми під час редагування програмного коду;
8. вимоги до складу і параметрів технічних засобів;
9. вимоги до маркування та упаковки (не висуваються);
10. вимоги до транспортування і зберігання (не висуваються).

Спеціальні вимоги (не пред’являються).

5. Вимоги до програмної документації

Програмною документацією до виробу «комп’ютерна модель прогнозування обсягу пам’яті графічного процесора» вважати:

1. Програмною документацією щодо розроблюваного програмного продукту вважати:
2. справжнє технічне завдання на розробку програми (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи);
3. Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи);
4. рекомендацій щодо застосування створеної програмної стандартизації у проєктах (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).
5. Текст програми (представити як Додаток Г до пояснювальної записки до кваліфікаційної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Для проведення випробувань необхідний проєкт для розробки веб-додатку на мові програмування python з використання бібліотеки Pandas.

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:
-ознайомчий (1-й етап);
-випробування програмного виробу (2-й етап).
Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1. Перевірку комплектності програмної документації.
2. Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.
3. Перевірку комплектності складу технічних і програмних засобів.
4. Методику проведення перевірок на 1 етапі випробувань.
5. Якість програмної документації перевіряється на відповідність вимогам стандартів ЕСПД.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1. перевірку відповідності технічних характеристик програми вимогам технічного завдання;
2. перевірку ступеня виконання функціональних вимог до програми;
3. методику проведення перевірок, що входять до переліку по 2 етапу випробувань.

1. Програма працює відповідно до умов експлуатації операційних систем MS Windows, Linux та MacOS.
2. Для роботи необхідний компілятор мови програмування python, версії не нижчої ніж 3.0
3. Порядок проведення випробувань:
   - 3.1. Запуск програми здійснюється за допомогою запуску веб-серверу на мові програмування python: “python <ім’я файлу>”;
   - 3.2. Після запуску програми необхідно у браузері комп’ютеру перейти за посиланням: http://localhost:8000/;
   - 3.3. У вкладці браузеру з’явиться інструмент для тестування Pandas запитів, де необхідно обрати одну з розроблених мутацій, заповнити дані необхідні для виконання запиту та натиснути на кнопку «Play»;
   - 3.4. Після натискання на екрані з’явиться результат відповіді серверу.

   Для проведення випробувань пропонується тест 1, тест 2 та тест 3.

Тест 1

Порівняння очікуваних та прогнозованих значень

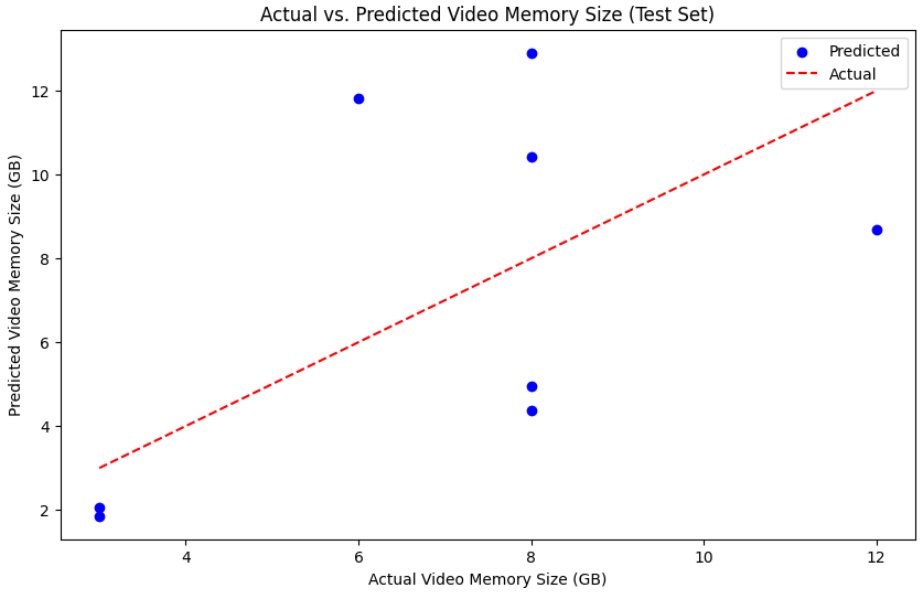

Рис. В.1 Тест 1

Тест 2

1. Відношення об’єму пам’яті до пропускної здатності в навчальній та прогнозованій виборках

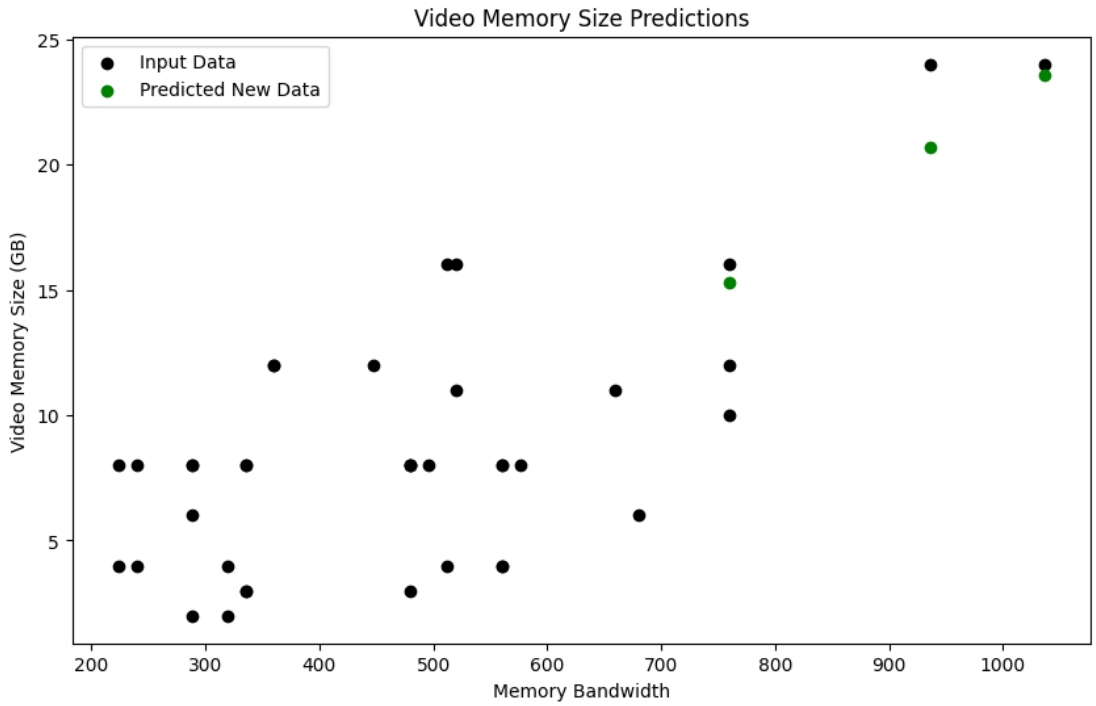

Рис. В.2 Тест 2

Тест 3

Коефіцієнти впливу параметрів та середньоквадратична помилка моделі

```
Коефіцієнти моделі:
Coefficients: [ 0.00020353 -0.03037911  0.02848508]
Root Mean Squared Error (RMSE) на тестовому наборі: 3.52 GB
```

Рис. В.3 Тест 3

Тест вважається пройденим, якщо відбуваються вказані операції і їх відображення у програмному продукті.

Висновки: тест 1 успішно пройшов випробування, тест 2 успішно пройшов випробування і тест 3 успішно пройшов випробування. Випробування пройшло успішно.

Виконавець: студент групи КІ-41, Богданов Є.С.

Додаток Г

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from math import sqrt
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np

# Зчитування даних з Excel файлу (XLSX)
data = pd.read_excel('/content/predictors.xlsx')

# Визначення вхідних (предикторів) та вихідної (цільової) змінних
X = data[['Shader Cores', 'Memory Bus Width', 'Memory Bandwidth']]
y = data['Video Memory Size']

# Розділення даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Побудова моделі лінійної регресії
model = LinearRegression()
model.fit(X_train, y_train)

# Виведення коефіцієнтів моделі
print('Коефіцієнти моделі:')
print('Coefficients:', model.coef_)

# Прогнозування на тестовому наборі
y_pred = model.predict(X_test)
# Оцінка точності моделі за допомогою середньоквадратичної помилки (RMSE)
rmse = sqrt(mean_squared_error(y_test, y_pred))
print(f'Root Mean Squared Error (RMSE) на тестовому наборі: {rmse:.2f} GB')
# Побудова графіку прогнозів та реальних значень на тестовому наборі
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted')
```

```
plt.plot([min(y_test),    max(y_test)],    [min(y_test),    max(y_test)],
linestyle='--', color='red', label='Actual')
plt.title('Actual vs. Predicted Video Memory Size (Test Set)')
plt.xlabel('Actual Video Memory Size (GB)')
plt.ylabel('Predicted Video Memory Size (GB)')
plt.legend()
plt.show()
# Приклад прогнозів для нових даних
new_data = pd.DataFrame({
    'Shader Cores': [10752, 10496, 8704],
    'Memory Bus Width': [384, 384, 384],
    'Memory Bandwidth': [1036, 936, 760]
})
predicted_memory_new = model.predict(new_data)
# Виведення прогнозів для нових даних і відображення на графіку порівняння
з вхідними даними
plt.figure(figsize=(10, 6))
# Додавання вхідних даних на графік
plt.scatter(X['Memory Bandwidth'], y, color='black', label='Input Data')
# Додавання прогнозів для нових даних на графік
plt.scatter(new_data['Memory      Bandwidth'],      predicted_memory_new,
color='green', label='Predicted New Data')

plt.title('Video Memory Size Predictions')
plt.xlabel('Memory Bandwidth')
plt.ylabel('Video Memory Size (GB)')
plt.legend()
plt.show()
```