Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп’ютерних інформаційних технологій

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до проведення лабораторних робіт
з дисципліни «Машинне навчання на основі хмарних технологій»

спеціальність: 123 - Комп’ютерна інженерія
галузь знань: 12 – Інформаційні технології
освітній рівень «Бакалавр»

2025

Піцун О.Й. Методичні рекомендації до проведення лабораторних робіт з дисципліни «Машинне навчання на основі хмарних технологій». Тернопіль: ЗУНУ, 2025. 40 с.

Укладачі: О.Й. Піцун, к.т.н., доцент кафедри КІ.

Відповідальний за випуск: Піцун О.!., к.т.н., доцент

Рецензенти:

Цмоць І.Г. - д.т.н., професор, професор кафедри АСУ НУ «Львівська політехніка»

Яцків В.В, - д.т.н., професор, завідувач кафедри кібербезпеки ЗУНУ

Методичні вказівки розглянуто та рекомендовано до друку на засіданні кафедри комп’ютерної інженерії протокол № 7 від 08.03.2025 р.

Методичні рекомендації затверджено на засіданні групи забезпечення спеціальності за напрямом підготовки "Комп’ютерна інженерія" протокол № 5 від 19.03.2025 р.

ЗМІСТ

ВСТУП

Курс «Машинне навчання на основі хмарних сервісів» призначений для розширення компетентностей випускників спеціальності 123 - Комп'ютерна інженерія в галузі прикладного застосування комп’ютерних систем штучного інтелекту в наукових дослідженнях та на виробництві. Введення курсу в навчальний план дозволяє надати студентам додаткові знання та практичні навички, які вони зможуть застосовувати як при подальшому навчанні, так і в майбутній професійній діяльності.

Дані методичні рекомендації призначені для студентів денної і заочної форм навчання освітнього рівня «Бакалавр».

Лабораторна робота №1

Тема: Виконання алгоритми машинного навчання на сервісі kaggle.

Мета: Навчитись налаштовувати та запускати скрипти на мові програмування Python у середовищі kaggle, що вміщують алгоритми машинного навчання.

Питання для обговорення:

1. Python, Jupyter Notebook, Kaggle
2. Лінійна регресія, опорні векторні машини
3. Персептрон

1. Теоретичні відомості

Машинне навчання — це галузь штучного інтелекту, яка фокусується на розробці алгоритмів і статистичних моделей, що дозволяють комп'ютерам навчатися, спираючись на дані, робити прогнози або приймати рішення.

Простим прикладом алгоритму машинного навчання є потоковий музичний сервіс Spotify. Для того, щоб Spotify міг прийняти рішення про те, які нові пісні або виконавців вам порекомендувати, алгоритми машинного навчання пов'язують ваші уподобання з іншими слухачами, які мають схожі музичні смаки. Цей метод, який часто називають просто штучним інтелектом, використовується в багатьох сервісах, що пропонують автоматизовані рекомендації.

Інший приклад — програмне забезпечення для розпізнавання мови, яке дозволяє перетворювати голосові повідомлення в текст. Або безпілотні автомобілі та функції допомоги водієві, такі як виявлення сліпих зон та автоматична зупинка.

Етапи машинне навчання

Збір даних. Інформація може надходити з таких джерел, як бази даних, датчики або інтернет.

Попередня обробка даних. Після того, як дані зібрані, їх потрібно обробити, щоб перевірити якість і придатність для аналізу.

Навчання моделі. Алгоритм потрібно навчити робити прогнози або приймати рішення на основі вхідних даних.

Відбір та розробка функцій. Модель машинного навчання відбирає із вхідних даних найбільш релевантні ознаки, які матимуть значний вплив на її продуктивність.

Оцінювання та оптимізація моделі. Після того, як модель навчена, її потрібно оцінити, щоб визначити, наскільки вона відповідає бажаним критеріям.

Розгортання та моніторинг. Після успішного навчання та оцінки модель розгортають в реальних застосуваннях машинного навчання.

Методи машинного навчання

Кероване машинне навчання

Модель тренують на вже помічених даних та готових відповідях. Алгоритм має обрати відповідь на гіпотезу: правильно чи неправильно, а людина — проконтролювати результат. Це відносно простий спосіб ML, який підходить для класифікації даних.

Некероване машинне навчання

Для навчання беруть інформацію без поміток. Алгоритм сам має зрозуміти закономірності та ознаки, які відрізняють об'єкти. Підходить для прогнозування та автоматизованого очищення вхідних даних, розпізнавання та розуміння мови людини.

Напівкероване машинне навчання

Використовуються марковані та немарковані дані. Решту розмітки виконує сам алгоритм за заданими параметрами. Таке навчання корисне для обробки об’ємних файлів.

Машинне навчання з підкріпленням

Алгоритм навчається, використовуючи метод спроб і помилок. Цей спосіб грунтується на системі балів, які модель отримує залежно від дій. Це схоже на систему винагород у грі, де за правильну дію гравець отримує бонус.

Переваги та недоліки машинного навчання

Переваги:

Ефективність. ML може автоматизувати повторювані завдання, що призводить до підвищення ефективності та дозволяє людині зосередитися на складніших та творчих завданнях.

Послідовність. Машини можуть виконувати завдання послідовно, не втомлюючись, що призводить до меншої кількості помилок порівняно з ручними процесами.

Інсайти на основі даних. Алгоритми ML можуть аналізувати великі обсяги даних, щоб виявити закономірності та ідеї, які можуть бути пропущені людиною.

Адаптивність. Моделі можуть адаптуватися до нових даних, вдосконалюючись з часом.

Зменшення помилок. ML може досягти більшої точності та достовірності, ніж традиційні методи, зменшуючи помилки в таких сферах, як медична діагностика або фінансове прогнозування.

Недоліки:

Якість даних. Ефективність моделей ML значною мірою залежить від якості та кількості даних. Неякісні дані можуть призвести до неточних моделей.

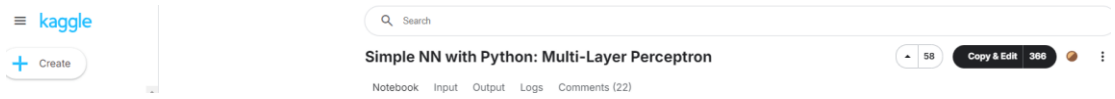
Конфіденційність. Робота з великими масивами даних, особливо з персональними даними, викликає значні занепокоєння щодо конфіденційності та безпеки.

Витрати. Навчання моделей ML може бути ресурсомістким, вимагаючи значних обчислювальних потужностей і часу.

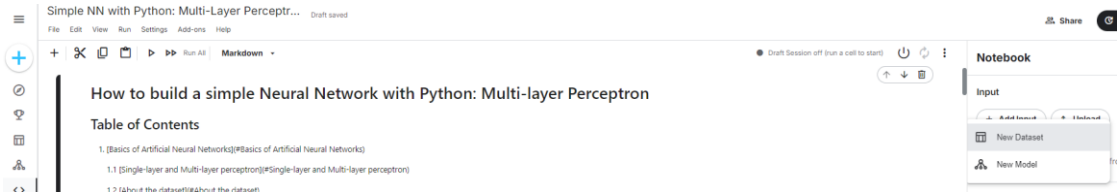
Фахівці. Без допомоги фахівця складно правильно інтерпретувати результати й усунути невизначеність.

Хід роботи

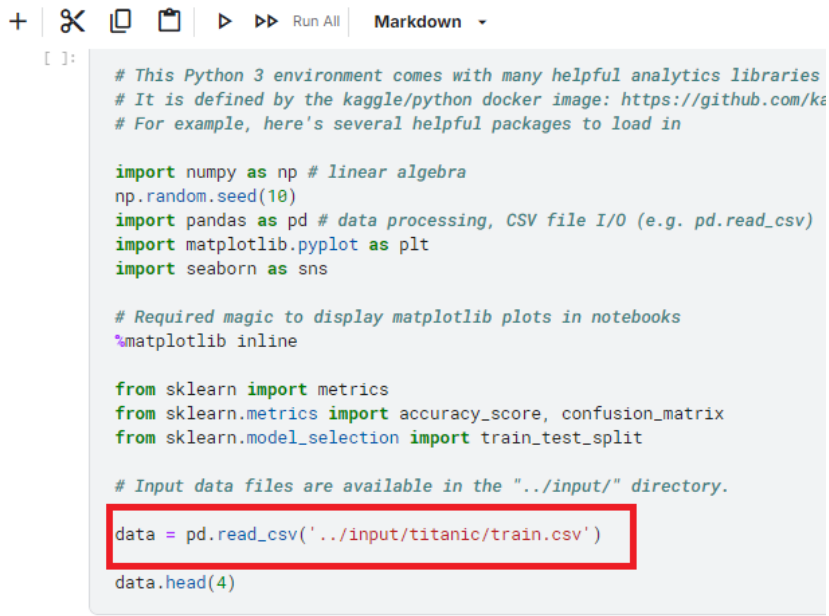
1. Зареєструватись у сервісі kaggle https://www.kaggle.com/ для можливості застосування python скриптів для машинного навчання.
2. Перейти за посиланням https://www.kaggle.com/code/androbomb/simple-nn-with-python-multi-layer-perceptron та виконати «Copy&Edit» для можливості запуску власної копії проекту.



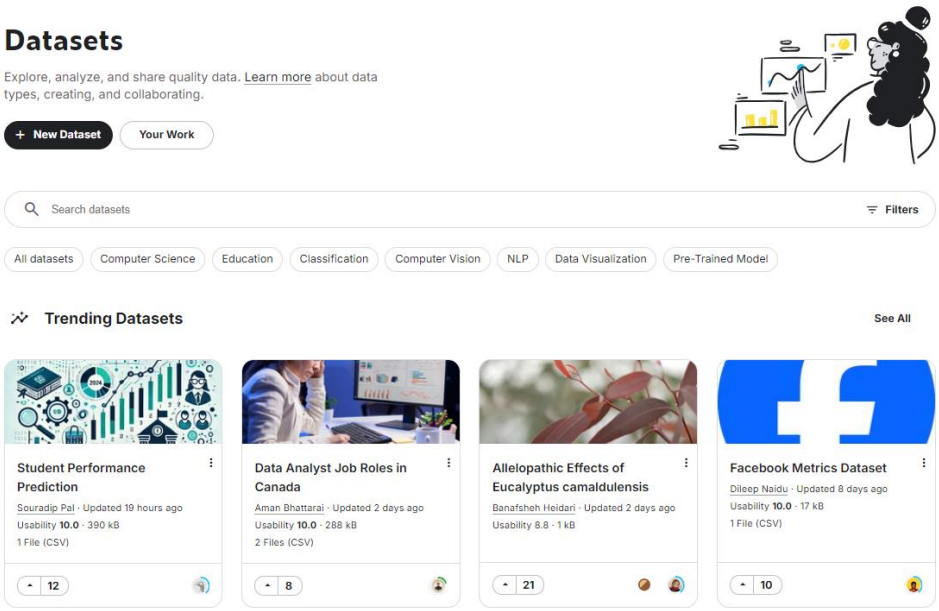
3. Запустити усі наявні блоки коду в Notebook
4. Вибрати або власноруч сформувати власний датасет та превести дослідження. Потрібно звернути увагу на назви полів в датасеті та зміни їх в програмному коді. «Upload -> New Datasets»



5. Змінити шлях до датасету

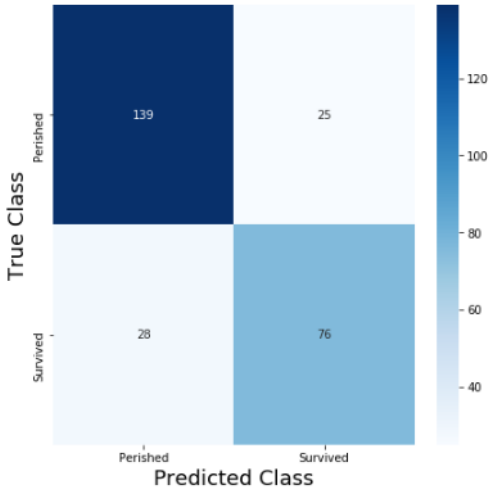


6. Приклади існуючих датасетів зберігаються у kaggle https://www.kaggle.com/datasets або можна сформувати власноруч.



7. Сформувати відношення навчальної вибірки до тестової
8. Реалізувати вивід confusion matrix для результатів дослідження

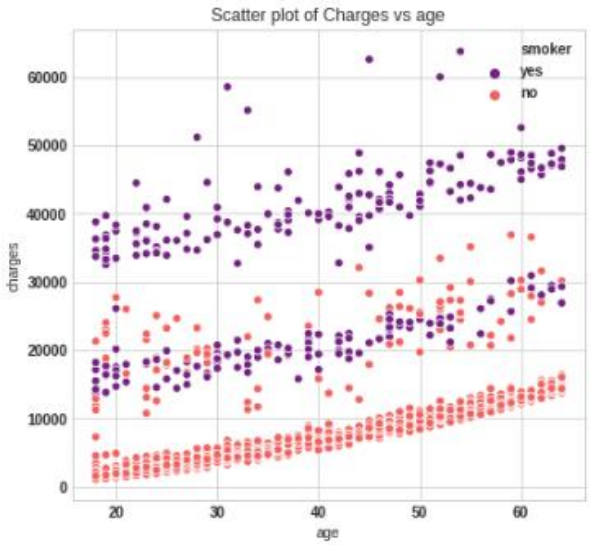
```
# Plot the confusion matrix
cm = confusion_matrix(Y_test, predictions)

df_cm = pd.DataFrame(cm, index = [dict_live[i] for i in range(0,2)], columns = [dict_live[i] for i
in range(0,2)])
plt.figure(figsize = (7,7))
sns.heatmap(df_cm, annot=True, cmap=plt.cm.Blues, fmt='g')
plt.xlabel("Predicted Class", fontsize=18)
plt.ylabel("True Class", fontsize=18)
plt.show()
```

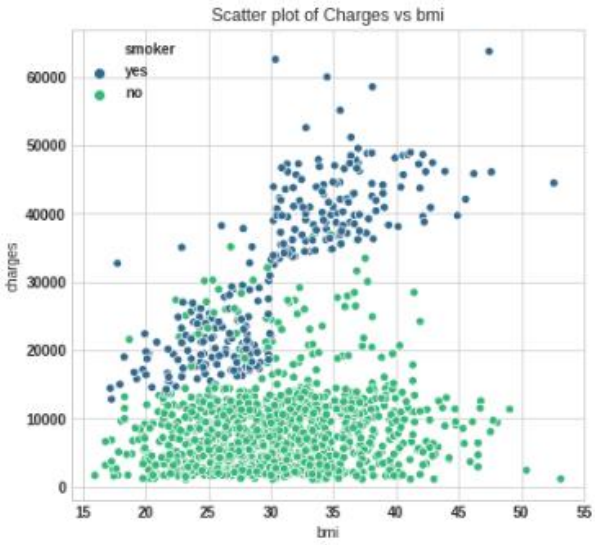


9. Визначити точність роботи програми.

Лінійна регресія

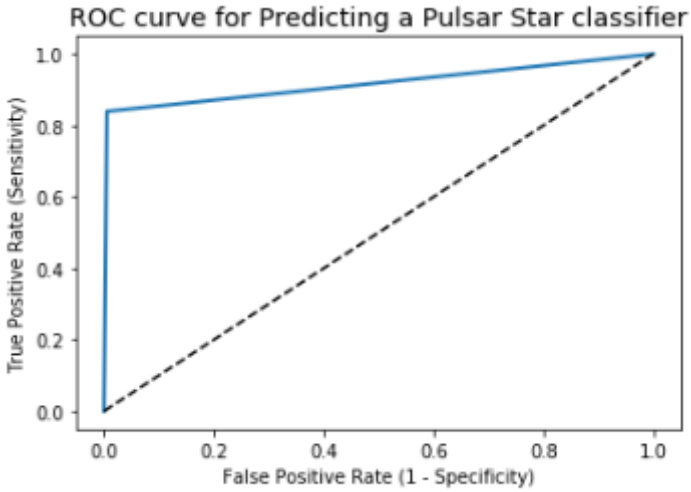
1. Створити копію проекту https://www.kaggle.com/code/sudhirnl7/linear-regression-tutorial у власному профілі
2. Змінити датасет на власний
3. Навести графіки результатів досліджень, для прикладу





Метод опорних векторів

1. Створити копію проекту https://www.kaggle.com/code/prashant111/svm-classifier-tutorial у власному профілі
2. Змінити датасет на власний
3. Результат роботи навести у вигляді ROC – кривих, приклад



Завдання

1. Відтворити кроки наведені у розділі «Хід роботи»

2. Підібрати датасет відповідно до варіанту
3. Результати досліджень оформити у вигляді звіту.

Контрольні питання

1. Основні бібліотеки для роботи з алгоритмами машинного навчання.
2. Основні функції сервісу Kaggle.
3. Поняття персептрону.
4. Лінійна регресія.
5. Метод опорних векторів.

Лабораторна робота №2

Тема: Використання згорткових нейронних мереж для класифікації зображень з використання Google Colab.

Мета: Навчитись розробляти архітектури згорткових нейронних мереж для класифікації зображень

Питання для обговорення:

1. Класифікація нейронних мереж
2. Шар згортки
3. Субдискретизуючий шар

1 Теоретичні відомості

Згорткові нейронні мережі (Convolutional neural networks - CNNs) подібні до звичайних штучних нейронних мереж. Вони також складаються з нейронів, які мають певні ваги і зміщення. Проте вхідними даними для них є зображення. Тобто вхідні дані можна подати у трьох вимірах – висота, ширина та кількість каналів. В той час як звичайні нейронні мережі приймають на вхід одномірний вектор. Саме через це CNN-мережі стали такими популярними. Оскільки такий підхід значно зменшує кількість параметрів в самій мережі і, відповідно, зменшує час навчання.

Типова згорткова нейронна мережа складається із таких шарів: згорткові, субдискретизуючі та повнозв’язні шари. Приклад такої мережі зображено на рисунку 1.1.

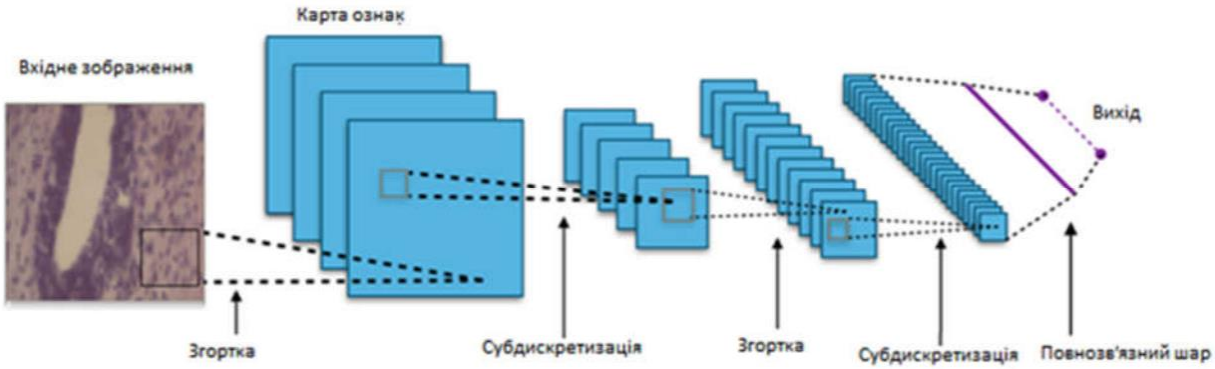


Рисунок 1.1 – Загальна структура згорткової нейронної мережі

Згортковий шар. Основними гіперпараметрами даного шару є розмір ядра (k - kernel size), кількість вхідних та вихідних фільтрів (f - filters), крок (s - stride) та доповнення нулями (p - padding). Принцип роботи полягає в наступному: пікселі ядра поелементно перемножуються із вхідними пікселями зображення та сумуються. Ядро переміщується із кроком, що дорівнює параметру stride, поки не «пройде» все вхідне зображення. Вихідний об’єм обчислюється за формулою O = (W-K+2P)/S+1.

Приклад. Нехай маємо вхідне зображення розміром 32x32x3 (W - ширина, H - висота, C - кількість каналів). Застосуємо згортковий шар k=3, s=1, p=0, f=96. Тоді отримаємо вихідне зображення розміром 30x30x96.

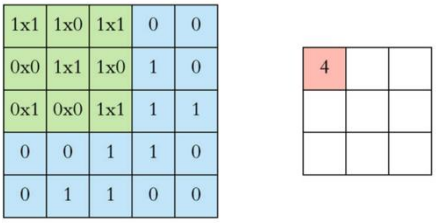


Рисунок 1.2 – Згортка зображення

Субдискретизація. Зазвичай цей шар використовується для зменшення розмірності зображення. В більшості випадків використовують максимальний пулінг, який вибирає максимальні значення із вхідного об’єму. Гіперпараметрами є розмір ядра та крок. Вихідний об’єм обчислюється за формулою O = (W-K)/S+1.

Приклад. Зображення 30x30x96 пікселів після застосування пулінгу буде мати об’єм 14x14x96 пікселів.

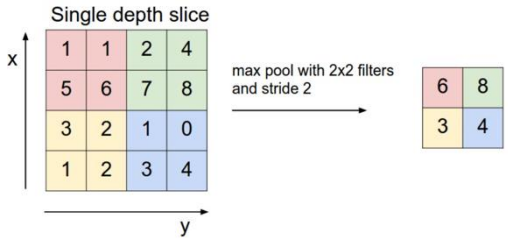


Рисунок 1.3 – Максимальний пулінг

Повнозв'язний шар. Цей шар застосовується в якості вихідного шару в задачах класифікації.

Детальнішу інформацію можна переглянути за посиланням: https://cs231n.github.io/convolutional-networks

2. Хід роботи.

Google Colab є потужним хмарним середовищем для роботи з Jupyter Notebook, яка дозволяє програмістам і дослідникам зручно проводити обчислення, аналіз даних, машинне навчання та інші завдання прямо в хмарі, не витрачаючи час на налаштування власної інфраструктури. Однією з ключових переваг Google Colab є безкоштовна доступність графічних процесорів, що дозволяє прискорити виконання складних обчислень та моделей машинного навчання. Крім того, Colab підтримує популярні бібліотеки Python, має зручний інтерфейс та надає можливість спільної роботи над проектами. Для програмістів та дослідників Google Colab стає незамінним інструментом для підвищення ефективності роботи та спрощення процесу розробки та досліджень.

Для того щоб розпочати роботу з Google Colab, першим кроком буде вивчення, що таке колаб та створення нового Colab ноутбука. Для цього потрібно зайти на сайт colab.research.google.com, де можна створити новий ноутбук, натиснувши кнопку "Файл" і вибравши "Новий ноутбук". Після цього відкриється нова вкладка з порожнім ноутбуком, де можна почати писати та запускати код на Python.

Для виконання коду в Colab необхідно в кожному осередку ноутбука вказати тип коду – Python

Для виконання коду в комірці необхідно натиснути кнопку у вигляді трикутника праворуч від комірки або використовувати комбінацію клавіш Shift + Enter. Після виконання коду результат буде відображено нижче за комірку.

Для імпорту даних та бібліотек у Google Colab ноутбуках існує кілька способів. Один із найпоширеніших способів завантаження даних – це використання методів з бібліотеки Pandas. Для завантаження даних із файлів CSV, Excel, SQL та інших форматів можна скористатися функціями Pandas

```
import tensorflow as tf
```

Также в Google Colab удобно использовать специальные команды, начинающиеся с «!» для установки дополнительных библиотек прямо из ноутбука:

```
!pip install numpy
import numpy as np
```

Google Colab надає чудові можливості для спільної роботи та обміну інформацією між користувачами. Однією з основних функцій, яка робить роботу в Colab зручною та ефективною є можливість спільної роботи в режимі реального часу.

За допомогою Colab можна легко запросити інших користувачів для спільної роботи над одним і тим самим ноутбуком. Це означає, що кілька людей можуть одночасно вносити зміни, коментувати код, обговорювати стратегії та

рішення, що суттєво прискорює процес розробки та покращує комунікацію всередині команди.

Для ефективного використання Google Colab є ряд кращих практик і корисних порад, які допоможуть оптимізувати роботу та забезпечити безпеку даних.

Рекомендується використовувати GPU та TPU для прискорення обчислень – це дозволить значно знизити час виконання коду, особливо під час роботи з великими обсягами даних або складними моделями машинного навчання. Також корисно регулярно зберігати результати роботи, використовуючи функцію збереження у хмарі або експорту та завантаження файлів – це допоможе уникнути втрати даних під час перезапуску середовища виконання. Важливо також стежити за безпекою даних, не розголошувати конфіденційну інформацію та використовувати аутентифікацію двофакторної перевірки, щоб захистити обліковий запис від несанкціонованого доступу.

Дотримуючись цих рекомендацій, ви зможете максимально ефективно працювати в Google Colab, забезпечуючи безпеку та збереження ваших даних.

3. Завдання.

- Скопіювати Colab Notebook за посиланням: https://colab.research.google.com/drive/1cx95kKLN8-k60NYhPORBfvGopNYBQLxQ?usp=sharing
- Експериментальним шляхом дослідити як змінюється точність класифікації при додаванні/видаленні шарів, коригуванні гіперпараметрів. Порівняти час навчання на CPU та GPU.

4. Структура звіту лабораторної роботи.

- Титульна сторінка.
- Тема та мета роботи.
- Код програми розв’язку індивідуального завдання.
- Копії екранів роботи програми.
- Висновки.

5. Контрольні запитання

1. Що таке CNN?
2. Які типи шарів використовуються в CNN та як вони працюють?
3 Відомі архітектури згорткових мереж.
4. Від чого залежить точність класифікації?
5. Що таке overfitting, underfitting та як з ними боротися?

Лабораторна робота №3

Тема: Реалізація CI/CD підходу з використанням github.

Мета: Навчитись працювати із системами контролю версій коду та освоїти принципи CI/CD

Питання для обговорення:

1. Мерження віток в GIT
2. Принципи CI
3. Принципи CD

2. Теоретичні відомості

Безперервна інтеграція (Continuous Integration, CI) і безперервне постачання (Continuous Delivery, CD) є культурою, набором принципів і практик, які дозволяють розробникам частіше і надійніше розгортати зміни програмного забезпечення.

CI/CD – це одна з DevOps-практик. Вона також відноситься і до agile-практик: автоматизація розгортання дозволяє розробникам зосередитися на реалізації бізнес-вимог, якості коду та безпеки.

Безперервна інтеграція — це методологія розробки та набір практик, за яких код вносяться невеликі зміни з частими комітами. І оскільки більшість сучасних додатків розробляються з використанням різних платформ та інструментів, то виникає потреба в механізмі інтеграції та тестуванні змін, що вносяться.

З технічної точки зору, мета CI — забезпечити послідовний та автоматизований спосіб складання, пакування та тестування додатків. При налагодженому процесі безперервної інтеграції розробники з більшою ймовірністю робитимуть часті комміти, що, у свою чергу, сприятиме покращенню комунікації та підвищенню якості програмного забезпечення.

Безперервне постачання починається там, де закінчується безперервна інтеграція. Вона автоматизує розгортання додатків у різні оточення: більшість розробників працюють як із продакшн-оточенням, так і з середовищами розробки та тестування.

Інструменти CI/CD допомагають налаштовувати специфічні параметри оточення, які конфігуруються під час розгортання. А також CI/CD-автоматизація виконує необхідні запити до веб-серверів, баз даних та інших сервісів, які можуть потребувати перезапуску або виконання якихось додаткових дій під час розгортання програми.

Безперервна інтеграція та безперервне постачання потребують безперервного тестування, оскільки кінцева мета — розробка якісних додатків. Безперервне тестування часто реалізується у вигляді набору різних автоматизованих тестів, які виконуються у CI/CD-конвеєрі.

Безперервне постачання

Безперервне постачання — це автоматичне розгортання програми на цільове оточення. Зазвичай розробники працюють з одним або декількома оточеннями розробки та тестування, в яких програма розгортається для тестування та ревю. Для цього використовуються такі CI/CD-інструменти як Jenkins, CircleCI, AWS CodeBuild, Azure DevOps, Atlassian Bamboo, Travis CI.

Типовий CD-конвеєр складається з етапів збирання, тестування та розгортання. Більш складні конвеєри включають наступні етапи:

- Отримання коду із системи контролю версій та виконання складання.
- Налаштування інфраструктури, що автоматизується через підхід “інфраструктура як код”.
- Копіювання коду у цільове середовище.
- Налаштування змінних оточення для цільового середовища.
- Розгортання компонентів програми (веб-сервери, API-сервіси, бази даних).

- Виконання додаткових дій, таких як перезапуск сервісів або виклик сервісів, необхідних працездатності нових змін.
- Виконання тестів та відкіт змін оточення у разі провалу тестів.
- Логування та надсилання сповіщень про стан поставки.

CI/CD-конвеєри

CI/CD-конвеєри призначені для організацій, яким необхідно часто вносити зміни до додатків з надійним процесом постачання. Крім стандартизації складання, розробки тестів та автоматизації розгортань ми отримуємо цілісний виробничий процес з розгортання змін коду. Використання CI/CD дозволяє розробникам зосередитися на покращенні програм і не витрачати сили на його розгортання.

CI/CD є однією з DevOps-практик, оскільки спрямована на боротьбу з протиріччями між розробниками, які хочуть часто вносити зміни та експлуатацією, яка потребує стабільності. Завдяки автоматизації, розробники можуть вносити зміни частіше, а команди експлуатації, у свою чергу, набувають більшої стабільності, оскільки конфігурація оточень стандартизована і в процесі постачання здійснюється безперервне тестування. Також налаштування змінних оточення відокремлено від програми та присутні автоматизовані процедури відкату.

Життєвий цикл DevOps

Життєвий цикл DevOps розділений на сім різних фаз безперервної розробки, які спрямовують процес розробки програмного забезпечення від початку до кінця. Щоб зрозуміти DevOps, важливо знати кожен етап життєвого циклу, а також процеси та вимоги кожного етапу.

Continuous development and delivery

Розробка програмного забезпечення починається з планування та кодування. У DevOps це робиться через процес регулярної доставки з метою постійного вдосконалення.

Спираючись на основні цінності Agile, DevOps заохочує регулярні часті випуски програмного забезпечення. Стандартним способом досягнення цього є автоматизація інтеграції та розгортання коду, процес, який називається безперервною інтеграцією/безперервним розгортанням (CI/CD).

Під час розробки, будь то на етапі попереднього чи постпродакшну, команди використовують відгуки, щоб виявити проблеми та висунути гіпотезу про рішення під час планування.

Після фази планування життєвого циклу DevOps починається створення вихідного коду та активів з метою забезпечення просування виробництва. Незалежно від того, яка мова кодування використовується, підтримка кодової бази за допомогою інструментів керування вихідним кодом є пріоритетом.

Continuous integration

Безперервна інтеграція (CI) — це практика розробки, яка вимагає від розробників інтегрувати код у спільне сховище кілька разів на день. Потім кожна перевірка або гілка перевіряється автоматизованою збіркою, що дозволяє командам виявляти проблеми на ранніх стадіях, забезпечуючи життєздатність і готовність основної гілки коду.

CI розроблений для підтримки багатьох невеликих повторюваних змін, а не меншої кількості великих змін. Це допомагає командам масштабувати автоматизовані робочі процеси для створення коду, тестування, об’єднання та перевірки в спільних сховищах.

Кінцевою метою безперервної інтеграції є надання кращого коду швидше. Завдяки меншим частим змінам у поєднанні з автоматизацією команди можуть швидше знаходити та усувати помилки та скорочувати час, витрачений на перевірку та випуск нових оновлень.

Continuous testing

Постійне тестування йде рука об руку з постійною інтеграцією. Конвеєри CI/CD залежать від автоматизованих тестів, а не від перевірки коду вручну. Це зроблено для того, щоб переконатися, що те, що розгортається, є якісним і не призведе до помилок, які порушують гру, до випуску.

DevOps покладається на усунення якомога більшої кількості ручних процесів. Чим більше ручних, виснажливих процесів на місці, тим більше часу витрачається даремно та більша ймовірність помилок. Мета безперервного тестування інструментів DevOps полягає не лише у виявленні помилок, а й у тому, щоб знайти їх якомога швидше, щоб їх не потрібно було виправляти на етапі виробництва за допомогою патча чи виправлення, що стає набагато складнішим і забирає багато часу. .

Автоматичні тести налаштовуються перед випуском збірки, а також перед виробництвом. Команди можуть додати перевірку вручну як останній крок перед виробництвом і після завершення автоматизованого тестування.

Continuous monitoring

Постійний моніторинг гарантує належну підтримку життєвого циклу DevOps з кінцевою метою забезпечення чудової взаємодії з користувачем. Оновлення та використання програмного забезпечення ретельно відстежуються, а зібрані дані використовуються для забезпечення правильної роботи програмного забезпечення.

Під час фази безперервного моніторингу команди намагаються якнайшвидше виявляти та усувати системні помилки. Автоматизоване відстеження помилок тут є важливим. Автоматизація також може забезпечити видимість інших областей, таких як загальна продуктивність програмного забезпечення, поведінка користувачів, стабільність інфраструктури розробки тощо.

Окрім нагляду за автоматизацією, ваша команда DevOps відповідає за забезпечення відповідності всіх аспектів конвеєра стандартам безпеки. Під час цієї фази також відбувається ручна обробка керування випуском.

Continuous feedback

Безперервний зворотний зв'язок вимагає впровадження циклу зворотного зв'язку для збору інформації про продуктивність програмного забезпечення від вашої внутрішньої команди та користувачів. Потім відгуки надсилаються команді DevOps, щоб допомогти в ітерації продукту. Джерелами можуть бути опитування, анкети, фокус-групи, соціальні мережі, форуми тощо.

Цей процес полягає не лише в тому, щоб визначити, чи ваше програмне забезпечення працює належним чином, але й у вимірюванні загальної задоволеності клієнтів, щоб керувати бізнес-стратегією та забезпечити найкращі результати. Необхідно використовувати постійний зворотний зв'язок, щоб скерувати план розвитку продукту та допомогти вам задовольнити бажання, потреби та очікування вашої аудиторії.

Continuous deployment

Безперервне розгортання працює в парі з безперервною інтеграцією, завершуючи цикл автоматизації та мінімізуючи або припиняючи втручання людини в процес розгортання. Автоматизовані інструменти DevOps відстежують оновлення вихідного коду та автоматично розгортають їх у робочому середовищі, щойно вони пройдуть фазу тестування, заощаджуючи час і підвищуючи задоволеність користувачів.

Безперервне розгортання прискорює зв'язок із користувачами завдяки автоматизації. Методи також можна розгортати для окремого розгортання для випуску, або приховуючи їх від користувачів (темні випуски), або вмикаючи їх для конкретних користувачів для тестування нових функцій і отримання відгуків (перемикачі функцій).

Оскільки код випускається невеликими партіями, це мінімізує ваші ризики, пов’язані з великими змінами коду – усе з мінімальними зусиллями завдяки автоматизації.

Continuous operations

Безперервна робота має на меті мінімізувати час простою та запобігти неприємним збоям у роботі користувачів. Ця фаза життєвого циклу DevOps зосереджена на оптимізації програм і середовищ для стабільності та продуктивності. Він також завершує цикл життєвого циклу DevOps, доповнюючи фазу планування безперервної розробки звітами про помилки та відгуками користувачів щодо покращень.

Завдяки безперервній співпраці між командами та користувачами помилки, відгуки та проблеми з безпекою можна постійно передавати, оцінювати та повторювати через конвеєр DevOps.

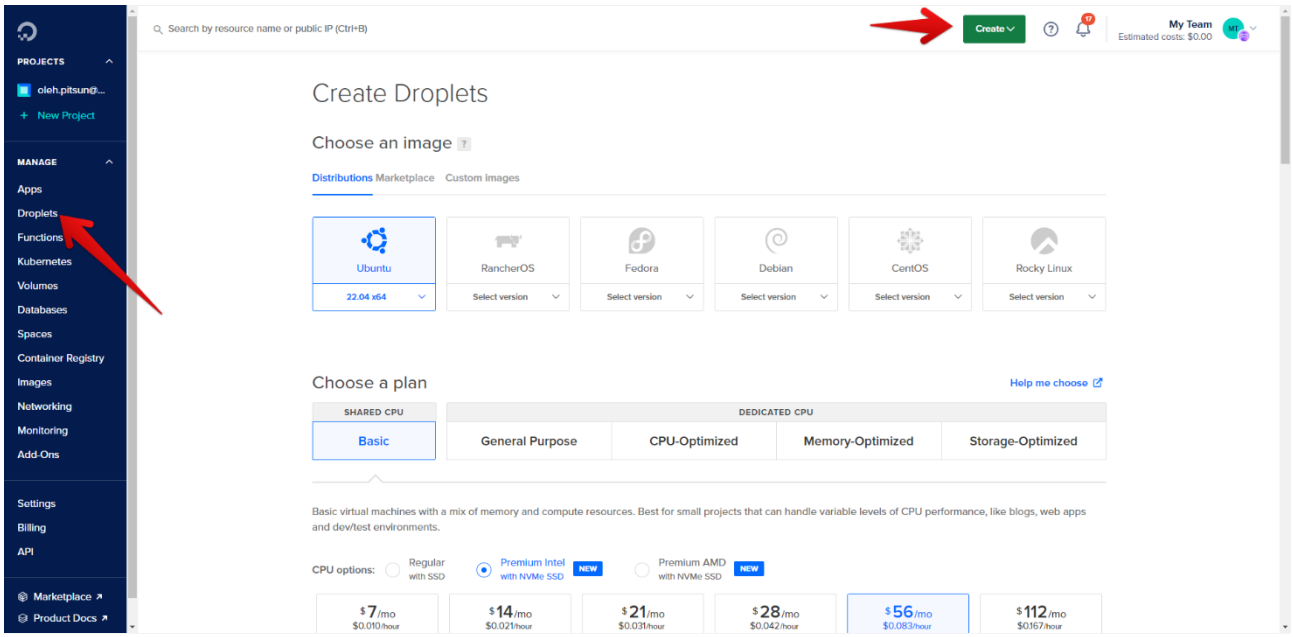
Хід роботи

Розглянемо яким чином можна реалізувати деплой проекту на DigitalOcean з використанням GithubAction базуючись на підходах CI/CD. Детальніше описано у цій статті https://apptest.ai-tern.in.ua/deploying-to-digitalocean-with-github-actions-ci-cd

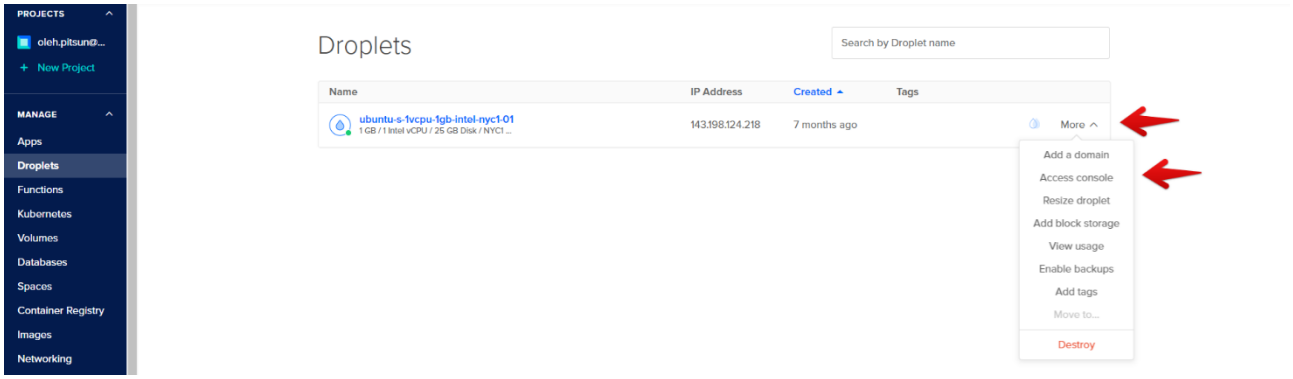
Створюємо новий droplet на digitalocean

На стороні digitalocean створюємо новий droplet (аналог віртуальної машини). У моєму випадку droplet не прив’язаний до домену, оскільки на даному етапі у цьому немає потреби. Створений droplet ви можете використовувати як серверне середовище для веб-сайту.

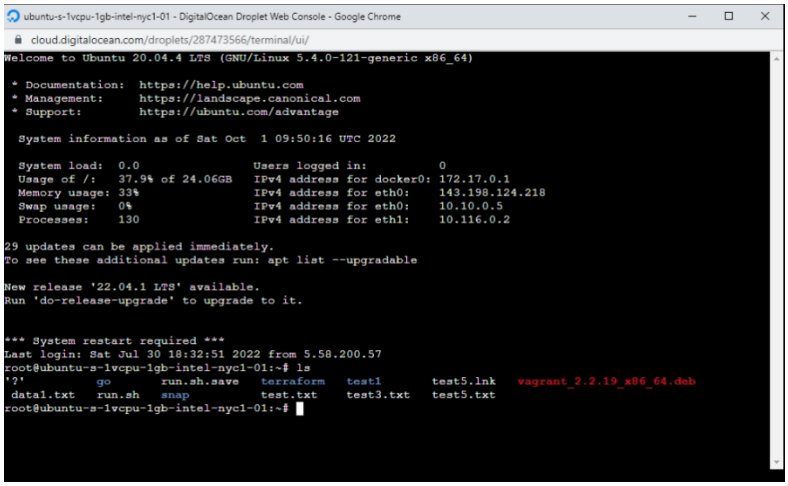
В якості операційної системи вибрано Ubuntu, оскільки вона є однією із найпопулярніших для розгортання веб – серверів, до того ж найкраще підходить для навчальних цілей.



Тепер потрібно отримати доступ до нашої системи. Для можна підключитись по SSH або підключитись використовуючи графічний інтерфейс digitalocean

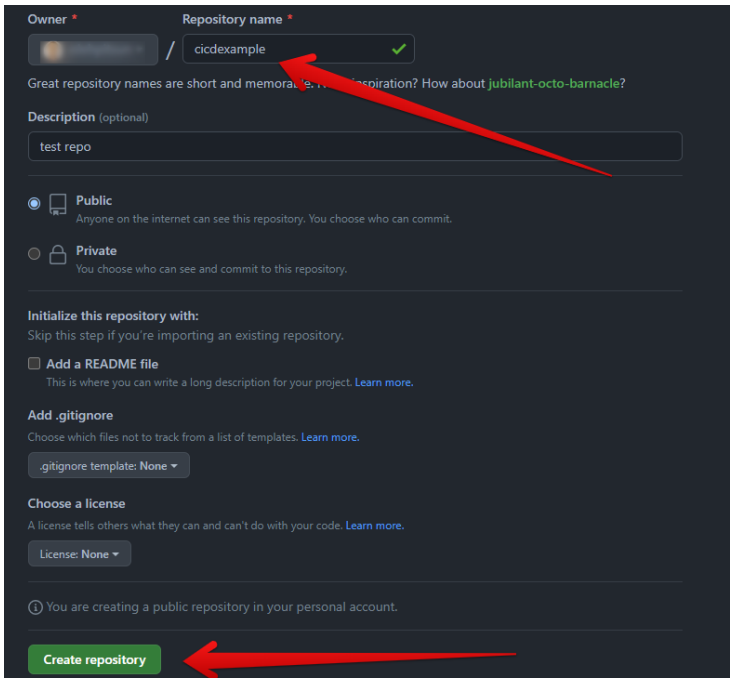


В результаті – отримуємо доступ до сервера з допомогою терміналу. Тепер ми можемо працювати на сервері та виконувати всі команди ОС Ubuntu з root – доступами.



Створення github – репозиторію

У нас є сервер на якому ми можемо розмістити наш проект, однак тепер потрібно створити репозиторій в якому буде зберігатись наш проект (код). Для цього потрібно мати акаунти на github та перейти на вкладку Repositories.



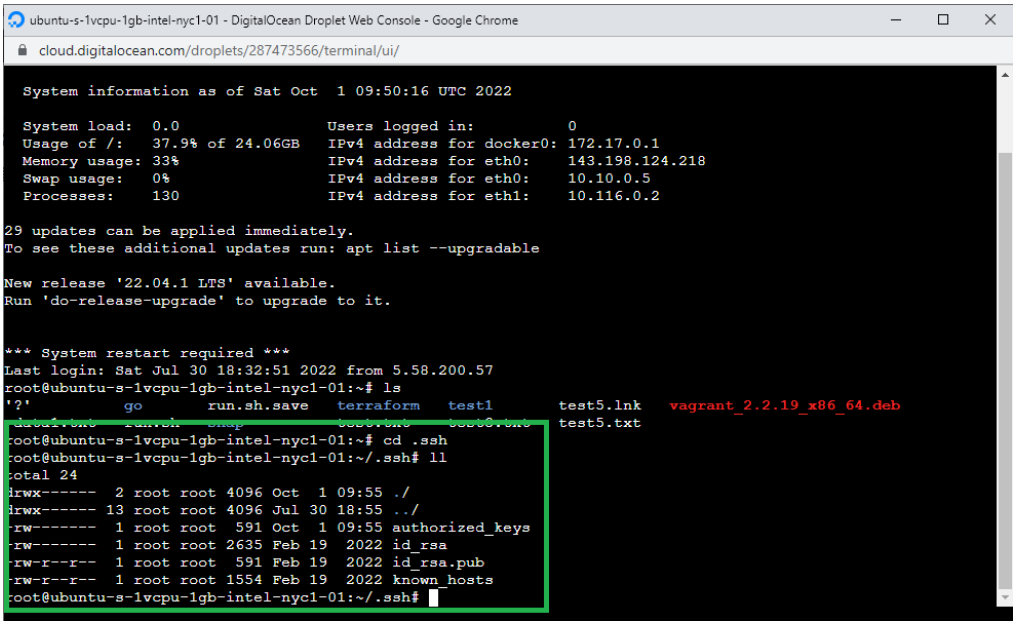
У даній репозиторії снаходитиметься код нашого проекту (наприклад, php проект, node.js – проект тощо) та конфігурації файли для реалізації ci/cd.

Генерування SSH ключів на стороні сервера

Для реалізації можливості CI/CD необхідно налаштувати звязок між github – репозиторієм та сервером. Для цього переходимо до терміналу та запускаємо команду:

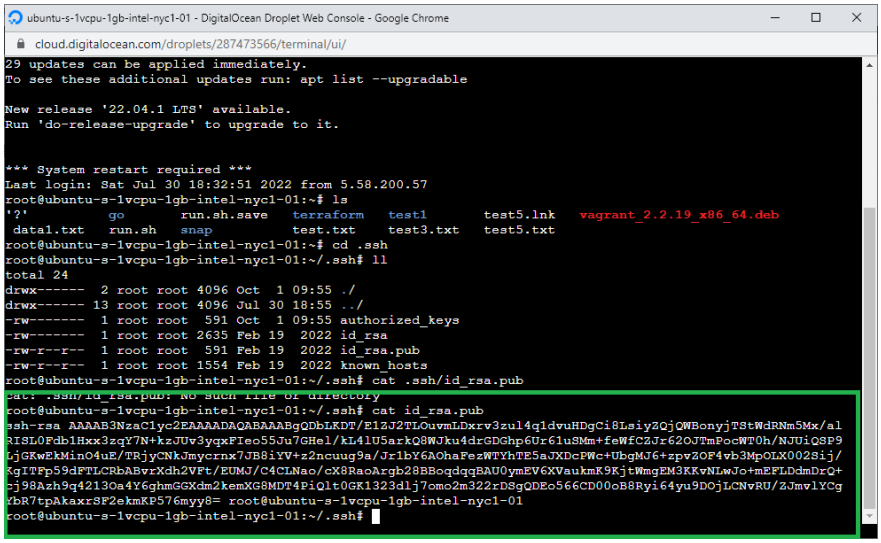
ssh-keygen

В результаті виконання команди, в директорії .ssh згенерувались два ключі приватний та публічний



Щоби переглянути вміст приватного ключа

cat .ssh/id_rsa



Додавання ключів та SECRETS параметрів у Github

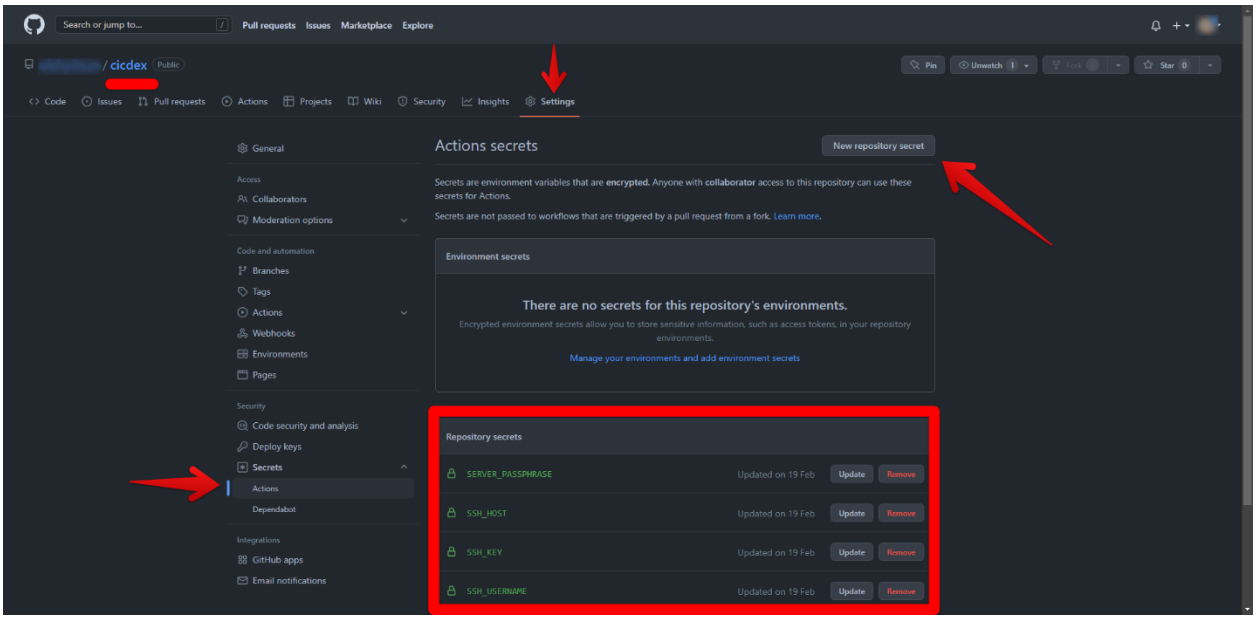
На стороні Github створимо Secrets – константи. Для цього перейдемо у необхідний репозиторій на github -> вкладку Settings -> Secrets -> Action

SERVER_PASSPHRASE

SSH_HOST – IP адреса сервера (в нашому випадку digitalocean)

SSH_KEY – приватний ключ, згенерований вище. Для перегляду ключа необхідно в терміналі виконати команду cat .ssh/id_rsa та скопіювати в буфер обміну. Після чого вставити на стороні Github

SSH_USERNAME – користувач (наприклад, root)



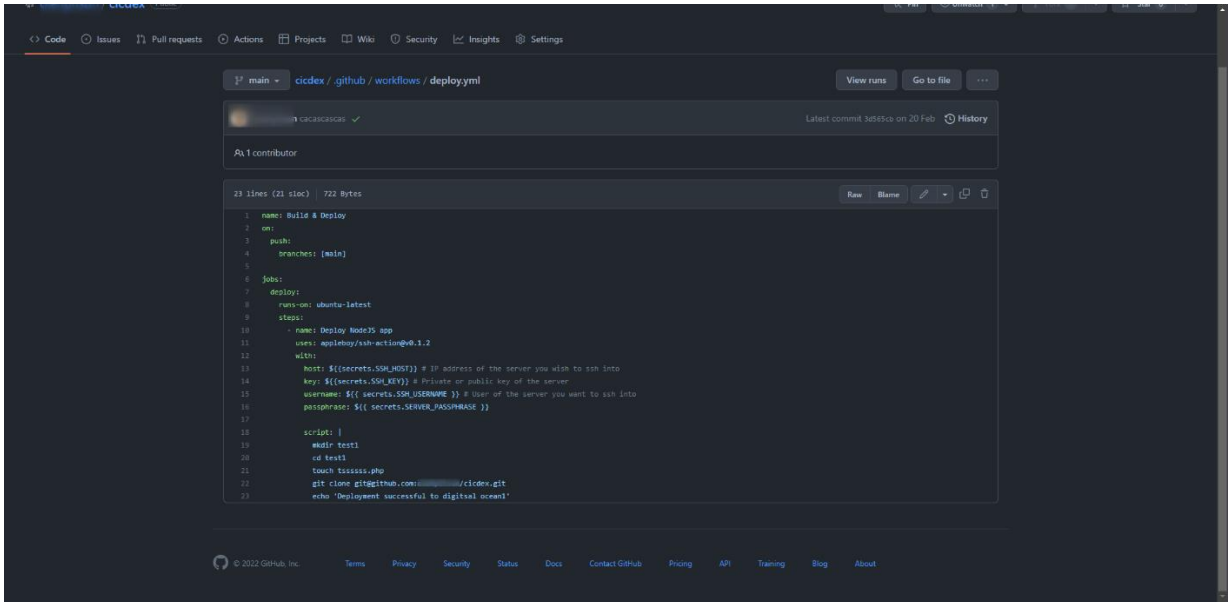
Створення deploy.yml файлу

Для реалізації ci/cd підходу необхідно розробити правила за якими наш проект буде розгортатись на сервері і таким чином впроваджуватись у production mode.

Створимо таку структуру директорій і файлів у проекті

.github/workflows/deploy.yml

У файл deploy.yml внесемо такий вміст

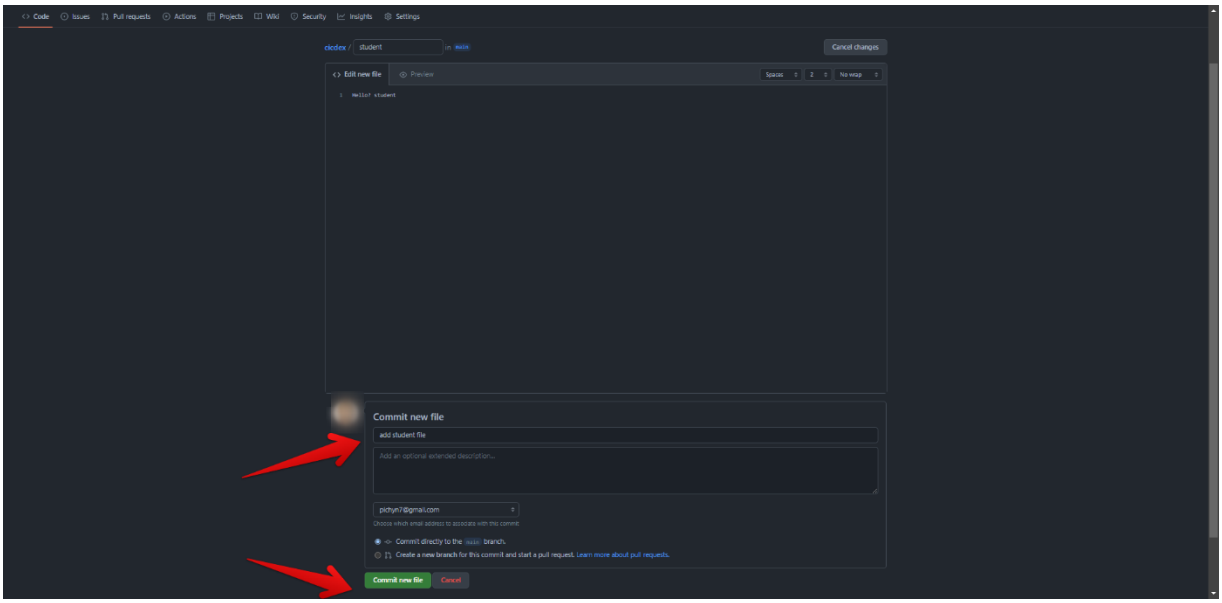


Відповідно до вищенаписаного ми вказуємо константи, що відповідають за під'єднання та авторизацією на сервері, копіюємо наш проект у певну директорію на сервері та з допомогою команд ОС Linux – створюємо додаткові директорії та файли.

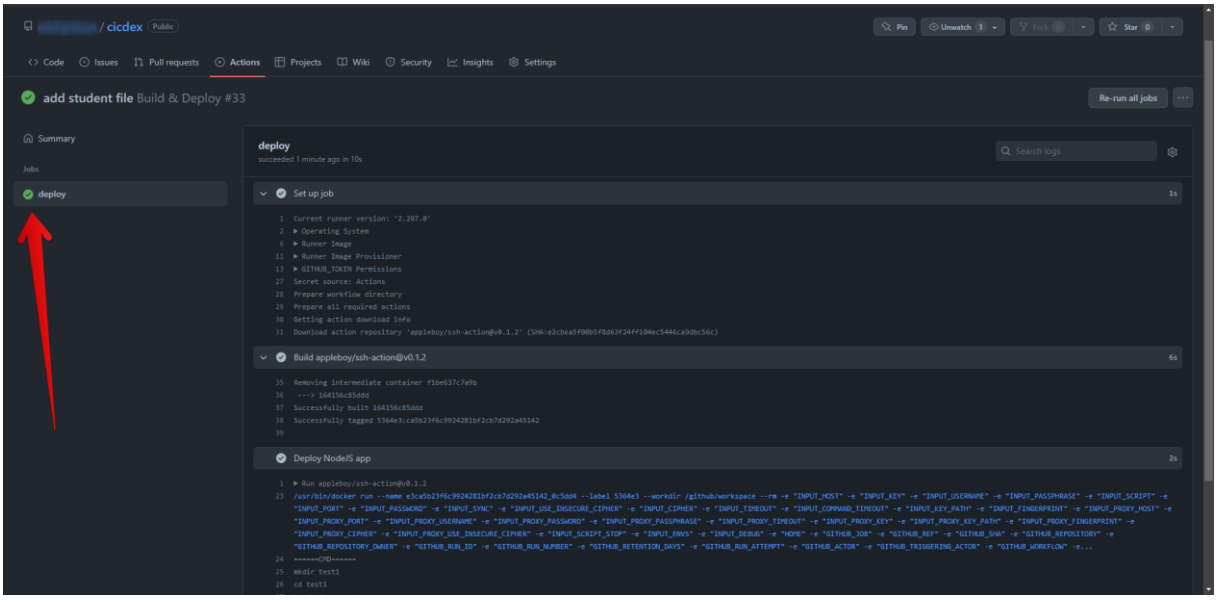
Результат

При створенні нового коміта у вітку main буде відбуватись розгортання проекту на сервері використовуючи методологію ci/cd

Створимо новий файл та закомітимо його.

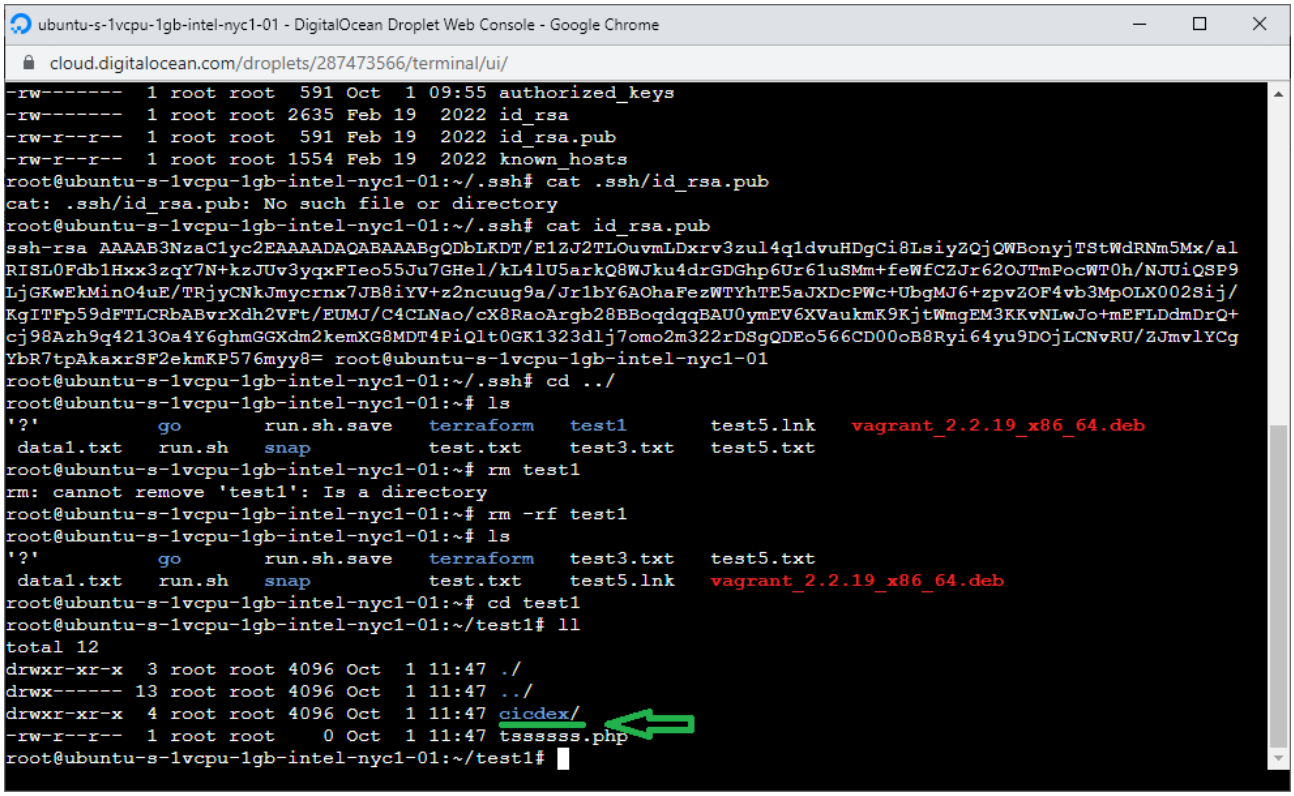


Переходимо у вкладку Actions



В результатів бачимо повідомлення про успішне виконання деплою.

Що ж відбувається на нашому сервері?



В результаті виконання деплою ми скопіювали останню версію проекту у папку cicdex та створили новий файл. Таким чином, у нас сервері знаходиться актуальна версія ПЗ.

Завдання

Реалізувати механізм CI/CD , використовуючи власний github репозиторій та ключі від клауд – сервера, надані викладачем.

Лабораторна робота №4

Тема: Генерування зображень з використанням GAN мереж.

Мета: Навчитись програмно реалізовувати модулі для генерування зображень

Питання для обговорення:

1. види генеративних мереж
2. Основні функції
3. Бібліотеки для розробки GAN мереж

Теоретичні відомості

Генеративні змагальні мережі (GAN – Generative Adversarial Nets) це засоби для генеративного моделювання з використанням методів глибокого навчання, таких як в глибоких згорткових нейронних мережах. GAN є генеративною моделлю, яка навчається з використанням двох нейронних мереж. Одна модель називається моделлю «генератор» або «генеративна мережа», яка навчається генерувати нові вірогідні зразки. Інша модель називається «дискримінатором» або «дискримінаційною мережею» і вчиться диференціювати генеровані приклади з реальними прикладами. Вони навчаються одночасно.

Генеративна мережа породжує підробку, а дискримінаційна нейромережа визначає наскільки генерований зразок реалістичний, порівнюючи його з реальним зразком. Генеративна і дискримінаційна нейромережі поводять себе як противники в деякій грі, звідси й назва змагальні нейромережі.

- GAN здатні реалістично генерувати нові людські обличчя, а також змінювати зображення, розфарбовуючи фотографії, зістарювати обличчя, використовуючи високу роздільну здатність. Також можна створити нових анімаційних персонажів або новий логотип. Створення моделей нового одягу або зразків нових меблів також може бути згенеровано за допомогою GAN. Генеративні змагальні мережі використовуються для створення зразків предметів для сцен у комп'ютерних іграх. Відомо, що ці мережі використовуються Facebook. Генеративні змагальні мережі – дуже молода галузь, перша стаття про такі мережі з'явилась в 2014 році. Використовуючи генеративні змагальні мережі, ми можемо навчитися створювати реалістичні підроблені версії майже будь-чого. У статті підібрана колекція сайтів, які з'явилися за останній місяць.

Типи генеративних моделей:

Варіаційні автоенкодери (англ. *variational autoencoders, VAE*) об'єднують елементи автоенкодерів і ймовірнісного моделювання. Їх використовують для генерації нових точок даних і вивчення базового розподілу даних.

Генеративно-змагальні мережі (англ. *generative adversarial network, GAN*) складаються з генератора і дискримінатора, які конкурують один з одним. GAN популярні для створення високореалістичних зображень, музики і тексту.

Рекурентні нейронні мережі (англ. *recurrent neural network, RNN*) використовуються для моделювання послідовних даних і чудово підходять для таких завдань, як генерація тексту, мовне моделювання і написання музики.

Моделі-трансформери, такі як GPT-4, затребувані завдяки здатності генерувати тексти. Їх застосовують для написання різних текстів і навіть коду.

Генеративні AI-фреймворки

TensorFlow — один з найбільш широко використовуваних фреймворків глибокого навчання. Він пропонує кілька бібліотек та інструментів для створення генеративних моделей, зокрема VAE та GAN. Основні переваги TensorFlow — гнучкість і велика документація.

PyTorch відомий своїм динамічним обчислювальним графом, що чудово підходить для досліджень і експериментів. Він підтримує різні генеративні моделі, включно з GAN, а його простий інтерфейс завоював популярність серед дослідників.

Keras, часто використовуваний як високорівневий API поверх TensorFlow, спрощує розробку моделей. Він підтримує GAN та інші генеративні моделі — чудовий вибір для тих, хто шукає простоту використання без втрати продуктивності.

Інструменти для генерації зображень

DALL-E, розроблений OpenAI, генерує зображення за текстовими описами. Він здатний створювати уявні та сюрреалістичні візуальні образи на основі простих промптів.

StyleGAN (розширення GAN) вміє генерувати реалістичні зображення високої роздільної здатності. Він може контролювати різні аспекти створення зображень, наприклад стиль та унікальні особливості.

Хід роботи та завдання

Загалом принцип роботи будь-якої генеративно-змагальної мережі можна описати наступним чином. Дискримінатор повинен відрізнити реальні дані від тих, які синтезовані генератором. Опираючись на певну функцію втрат, дискримінаційна модель повинна максимізувати імовірність реальних даних та мінімізувати імовірність синтезованих. Так само генератор повинен синтезувати дані високої правдоподібності, щоб дискримінатор вважав їх реальними.

- Скопіювати Colab Notebook за посиланням: https://colab.research.google.com/drive/11B3cW8amCcSE5axX6rIXlXO6p710d45r?usp=sharing
- Експериментальним шляхом дослідити як змінюється якість згенерованих зображень при додаванні/видаленні шарів, коригуванні гіперпараметрів.

Структура звіту лабораторної роботи.

- Титульна сторінка.
- Тема та мета роботи.
- Код програми розв’язку індивідуального завдання.
- Копії екранів роботи програми. - Висновки.

Контрольні запитання

1. Що таке GAN мережа та з чого вона складається?

2. Які типи шарів можна використовувати в генераторі для збільшення розмірності зображення?

3 Проблеми навчання GAN.

4. Від чого залежить якість згенерованих зображень?

5. Які приклади застосування GAN ви знаєте?

РЕКОМЕНДОВАНІ ДЖЕРЕЛА ІНФОРМАЦІЇ

[1] M. Testi et al., "MLOps: A Taxonomy and a Methodology," in IEEE Access, vol. 10, pp. 63606-63618, 2022, doi: 10.1109/ACCESS.2022.3181730.

[2] Nazir, Sajid, Diane M. Dickson, and Muhammad Usman Akram. "Survey of explainable artificial intelligence techniques for biomedical imaging with deep neural networks." Computers in Biology and Medicine (2023): 106668.

[3] Dhar, Tribikram, Nilanjan Dey, Surekha Borra, and R. Simon Sherratt. "Challenges of Deep Learning in Medical Image Analysis—Improving Explainability and Trust." IEEE Transactions on Technology and Society 4, no. 1 (2023): 68-75.

[4] Vega, Carlos, Miroslav Kratochvil, Venkata Satagopam, and Reinhard Schneider. "Translational challenges of biomedical machine learning solutions in clinical and laboratory settings." In International Work-Conference on Bioinformatics and Biomedical Engineering, pp. 353-358. Cham: Springer International Publishing, 2022.

[5] Bennetot, Adrien, Ivan Donadello, Ayoub El Qadi, Mauro Dragoni, Thomas Frossard, Benedikt Wagner, Anna Saranti et al. "A Practical guide on Explainable AI Techniques applied on Biomedical use case applications." arXiv preprint arXiv:2111.14260 (2021).

[6] Granlund, Tuomas, Vlad Stirbu, and Tommi Mikkonen. "Towards regulatory-compliant MLOps: Oravizio’s journey from a machine learning experiment to a deployed certified medical product." SN computer Science 2, no. 5 (2021): 342.

[7] Reddy, Manjunatha, Brahmanand Dattaprakash, Sandesh Kammath, Subramanya Kn, Sumathra Manokaran, and Rangaswamy Be. "Application of mlops in prediction of lifestyle diseases." ECS Transactions 107, no. 1 (2022): 1191.

[8] Jain, Archit, Adarsh Malviya, Disha Bajaj, Revati Bhavsar, and Amit Savyanavar. "Brain Tumor Detection using MLops and Hybrid Multi-Cloud." In

2022 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS), pp. 1-6. IEEE, 2022.

[9] Stirbu, Vlad, Tuomas Granlund, and Tommi Mikkonen. "Continuous design control for machine learning in certified medical systems." Software Quality Journal 31, no. 2 (2023): 307-333.

[10] Berezsky, Oleh, Oleh Pitsun, Grygory Melnyk, Yuriy Batko, Bohdan Derysh, and Petro Liashchynskyi. "Application Of MLOps Practices For Biomedical Image Classification." In IDDM, pp. 69-77. 2022.

[11] Berezsky, Oleh, Oleh Pitsun, Grygoriy Melnyk, Tamara Datsko, Ivan Izonin, and Bohdan Derysh. "An Approach toward Automatic Specifics Diagnosis of Breast Cancer Based on an Immunohistochemical Image." Journal of Imaging 9, no. 1 (2023): 12.

[12] Berezsky, Oleh, Oleh Pitsun, Bohdan Derysh, Ihor Pazdriy, Grygory Melnyk, and Yuriy Batko. "Automatic segmentation of immunohistochemical images based on U-net architecture." In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), vol. 1, pp. 29-32. IEEE, 2021.

[13] Nyckel URL: https://www.nyckel.com/

[14] ximilar URL: https://www.ximilar.com/

[15] hasty URL: https://hasty.cloudfactory.com/

[16] levity URL: https://levity.ai/

О.Й. Піцун

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт

з дисципліни «Машинне навчання на основі хмарних технологій»

спеціальність: 123 - Комп’ютерна інженерія

галузь знань: 12 – Інформаційні технології

освітній рівень «Бакалавр»

Підписано до друку 20.03.2025 р.
Формат 60х84/16. Папір офсетний.
Друк на дублікаторі.
Умов.- друк арк. 1.4 Обл.-вид. арк 1.5.
Тираж 25 прим.

Віддруковано ФОП Шпак В.Д.
Свідоцтво про державну реєстрацію
серія В02 №924434 від 11.12.2006 р.
Свідоцтво платника податку: Е № 897220