**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ “КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВИЙ ПРОЕКТ**

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: «Моніторингова система ресурсів з пошуку оголошень про продаж нерухомості(квартир)»

**Студентка**

**групи** ​КП-72 Городченко Анна Володимирівна \_\_\_\_\_\_\_\_\_ (ПІБ) (підпис)

**Викладач**

**к.т.н, доцент кафедри**

**СПіСКС** **Петрашенко А.В.**  \_\_\_\_\_\_\_\_\_

(підпис)

Захищено з оцінкою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ – 2020

# **Анотація**

Метою розробки даного курсового проекту є набуття практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з постреляційними базами даних, а також здобуття навичок оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. У результаті виконання курсового проекту було опановано навички розробляти програмне забезпечення для постреляційних баз даних, володіння основами використання СУБД, а також інструментальними засобами аналізу великих обсягів даних.

Темою даного курсового проекту є створення моніторингової системи ресурсів з пошуку оголошень про продаж квартир в Україні. У документі викладена актуальність та проблематика аналізу великого обсягу даних, аналіз використаного інструментарію (опис мови програмування, використаних бібліотек та СУБД), описана структура бази даних, опис розробленого програмного забезпечення (загальний, опис модулів та основних алгоритмів роботи), аналіз функціонування засобів масштабування, та опис результатів проведеного аналізу.

Результатами даного проекту стали діаграми та графіки, що зображають результати аналізу оголошень про продаж квартир в Україні. З ними можна ознайомитися в додатку А.

# **Зміст**

[**Анотація** 2](#_Toc40960364)

[**Зміст** 3](#_Toc40960365)

[**Вступ** 4](#_Toc40960366)

[**Аналіз інструментарію для виконання курсового проекту** 5](#_Toc40960367)

[**Аналіз СУБД** 5](#_Toc40960368)

[**Обґрунтування вибору мови програмування** 8](#_Toc40960369)

[**Обґрунтування вибору бібліотек і фреймворків** 9](#_Toc40960370)

[**Структура бази даних** 10](#_Toc40960371)

[**Аналіз функціонування засобів масштабування** 11](#_Toc40960372)

[**Тестування масштабування** 14](#_Toc40960373)

[**1.** **Тестування відмовостійкості шляхом зупинки декількох процесів із репліки сету одного із шардингу** 15](#_Toc40960374)

[**2.** **Тестування відмовостійкості системи шляхом зупинки декількох процесів із репліки сети серверів конфігурації** 17](#_Toc40960375)

[**3.**  **Перевірка того, яку частину даних ми втратимо, в разі відмови однієї реплікації серверів шардингу** 18](#_Toc40960376)

[**Опис результатів аналізу предметної галузі** 19](#_Toc40960377)

[**Висновки** 20](#_Toc40960378)

[**Література** 21](#_Toc40960379)

[**Додаток А** 22](#_Toc40960380)

[**Додаток Б** 23](#_Toc40960381)

# **Вступ**

Data Science – галузь інформатики, що вивчає проблеми аналізу, обробки і представлення даних у цифровій формі. Говорячи простіше, це наука про методи обробки великих масивів даних і вилучення з них цінної інформації, завдяки чому можна більш ефективно приймати рішення. Все це стало можливо завдяки появі хмарних сервісів для зберігання даних, зростання обчислювальних здібностей комп'ютерів, розвитку технологій машинного навчання і нейромереж.

Сучасний Data Science спеціаліст (дослідник даних) має оперувати великими обсягами даних та вміти виокремити з них приховані залежності, на основі яких зробити прогноз про те, як будуть надалі відбуватися ті чи інші явища. Дослідники даних використовують свої [дані](https://uk.wikipedia.org/wiki/Дані) та [аналітичні здібності](https://uk.wikipedia.org/wiki/Аналітичні_здібності) для пошуку та інтерпретації великих джерел даних; керують великими обсягами даних безвідносно до [апаратного](https://uk.wikipedia.org/wiki/Апаратне_забезпечення) та [програмного](https://uk.wikipedia.org/wiki/Програмне_забезпечення) забезпечення і обмежень пропускної здатності; об'єднують джерела даних; забезпечують [цілісність](https://uk.wikipedia.org/wiki/Цілісність) наборів даних; створюють [візуалізації](https://uk.wikipedia.org/wiki/Візуалізація) для кращого розуміння даних; з використанням даних будують [математичні моделі](https://uk.wikipedia.org/wiki/Математична_модель); надають тлумачення даних та висновки.

Метою даного курсового проекту був аналіз оголошень про продаж квартир в Україні. Мета дослідження полягає в прогнозуванні ціни на квартири відповідно до площі, кількості кімнат та району, де розташовані квартири. Також метою даного курсового проекту є набуття навичок з масштабування високонавантажених системи, роботою з Big Data, науковими бібліотеками мови програмування Python3 а також набуття навичок регресійного аналізу.

## **Аналіз інструментарію для виконання курсового проекту**

## **Аналіз СУБД**

Під час виконання курсового проекту виникла потреба зберігати велику кількість даних. Найкращий варіант для зберігання великої кількості даних-використання СУБД.

В якості СУБД були розглянуті варіанти: PostgreSQL, MongoDB. З порівняльною характеристикою цих СУБД можна ознайомитися в таблиці 1.

*таблиця 1. Порівняльна характеристика СУБД*

|  |  |  |
| --- | --- | --- |
| Критерій порівняння | Назва СУБД | |
| MongoDB | PostgreSQL |
| Має відкритий вихідний код | так | так |
| Схема даних | динамічна | статична і динамічна |
| Підтримка ієрархічних даних | так | так ( з 2012) |
| Реляційні дані | ні | так |
| Транзакції | ні | так |
| Атомарністі операцій | всередині документа | по всій БД |
| Мова запитів | JSON/JavaScript | SQL |
| Найлегший спосіб масштабування | горизонтальний | вертикальний |
| Підтримка шардингів | так | так ( важка конфігурація) |
| Приклад використання | Великі дані (мільярди записів) з великою кількістю паралельних оновлень, де цілісність і узгодженість даних не потрібно. | Транзакційні і операційні програми, вигода яких в нормалізованому формі, об'єднаннях, обмеження даних і підтримки транзакцій. |
| Наявність бібліотек для мови програмування Python 3 | так | так |
| Підтримка реплікації | так, автоматичне переобрання головного процесу | За принципом master-slave |
| Засіб збреження та відновлення даних | mongodump | pg\_dump |
| Форма збереження даних | документи JSON | таблиця |

За результатами порівняння цих СУБД було прийнято рішення зупинитися на NoSQL рішеннях. Оскільки вони чудово поєднують в собі переваги неструктурованих баз даних та простоту використання горизонтального масштабування. Крім цього, класичним прикладом використання NoSQL СУБД є системи збору та аналізу даних, до яких можна застосувати індексування за первинними та вторинними ключами.

NoSQL база даних є об’єктно орієнтованою та дозволяє зберігати великі масиви неструктурованих даних. На відміну від SQL баз даних ми можемо зберігати дані у “сирому” об’єктному вигляді, який використовується програмою та є більш близьким за структурою до моделі даних, яку буде використовувати ПЗ написане з використанням мови програмування Python. Це пришвидшить збір, збереження та отримання даних програмним забезпеченням. Оскільки MongoDB є представником NoSQL баз даних, вона не потребує жорсткої схеми даних, що дозволяє пришвидшити процес розробки та зробити його більш гнучким. Окрім цього дана СУБД підтримує горизонтальне масштабування за допомогою шардингу з метою зменшення навантаження на кожен окремий вузол шляхом розподілення навантаження між ними.

## **Обґрунтування вибору мови програмування**

Мовою програмування для ПЗ було обрано Python 3.8.Ключовою особливістю Python є широкий інструментарій засобів розробки систем збору та аналізу даних. Фактично ця мова є стандартом у світі математичних розрахунків та обробки даних у реальному часі. Тож під час виконання курсового проекту було відносно легко знайти необхідну документацію та приклади роботи із цією мовою.

## **Обґрунтування вибору бібліотек і фреймворків**

Використані бібліотеки та фреймворки:

* ***numpy*** - математична бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами.
* ***matplotlib*** - це бібліотека Python 2D, яка представляє числові дані у різноманітних форматах та інтерактивних середовищах на різних платформах
* ***scikit-learn*** - бібліотека машинного навчання на мові програмування Python з відкритим вихідним кодом. Містить реалізації практично всіх можливих перетворень, і нерідко її однією вистачає для повної реалізації моделі. У бібліотеці також є основні алгоритми машинного навчання: *лінійної регресії* і її модифікацій Лассо, гребньовій регресії, опорних векторів вирішальних дерев і лісів та інше.
* ***scrappy*** - це швидкий веб-фреймворк з відкритим вихідним кодом, написаний на Python, який використовується для отримання даних з веб-сторінки за допомогою селекторів на основі XPath.
* ***scipy*** - є відкритим вихідним кодом для Python, поширюваним в рамках ліцензованої бібліотеки BSD для виконання математичних, наукових та інженерних обчислень. Бібліотека SciPy створена для роботи з масивами NumPy і надає безліч зручних і ефективних чисельних методів, таких як процедури чисельної інтеграції та оптимізації.
* ***pymongo*** - є дистрибутивом Python, що містить інструменти для роботи з MongoDB, і є рекомендованим способом роботи з MongoDB від Python.
* ***flask*** - фреймворк для створення веб-додатків на мові програмування Python

# **Структура бази даних**

База даних складається з однієї колекції, в якій зберігаються відомості про оголошення. Загальна структура документа в базі даних приведена у таблиці 2.

*таблиця 2. Опис властивостей документа у базі даних*

|  |  |  |
| --- | --- | --- |
| Назва властивості | Тип | Опис |
| \_id | ObjectId | Ідентифікатор запису |
| region | String | Район, де знаходиться квартира |
| price | Integer | Ціна квартири |
| number\_of\_rooms | Integer | Кількість кімнат в квартирі |
| area | Integer | Загальна площа квартири |
| source | String | Назва сайту, де викладено оголошення |

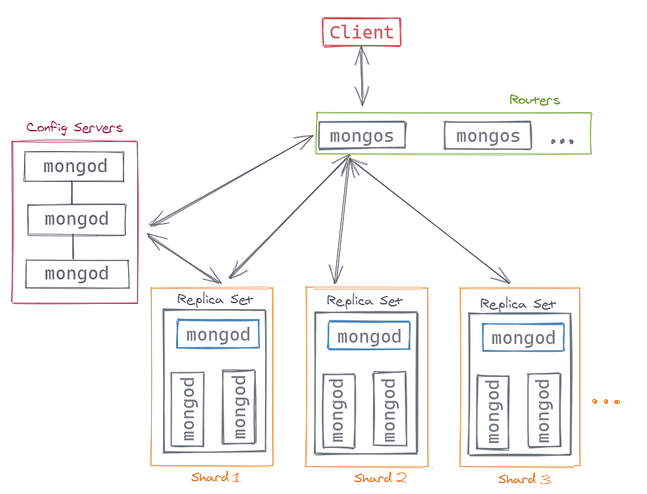
# **Аналіз функціонування засобів масштабування**

В якості засобів масштабування було обрано реплікацію та шардинг.

**Реплікація** - це процес синхронізації даних на декількох серверах. Даний механізм зменшує витрати ресурсів і збільшує доступність даних, копії яких зберігаються на різних серверах. Реплікація захищає базу даних від втрати єдиної сервера і дозволяє зберегти дані в разі технічної несправності на одному з серверів. У MongoDB реплікація досягається шляхом використання набору копій (replica set). Це група примірників mongod, який зберігають однакові набори даних. У копії один вузол - це ключовий вузол, який отримує всі операції запису. Всі інші вузли - вторинні, приймають операції з першого, таким чином, зберігаючи такі ж записи, як і первинний вузол. Набір копій може мати тільки один первинний вузол.

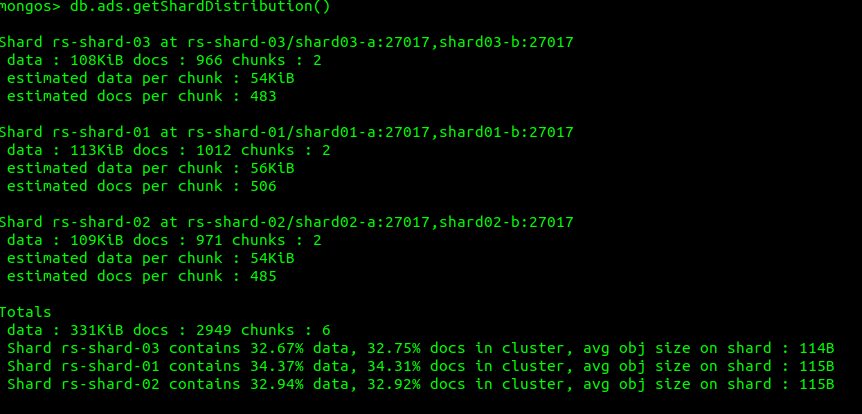
**Шардінг** - це процес зберігання документів на декількох серверах і це спосіб, яким MongoDB справляється з великими даними. З ростом кількості даних, один сервер не може зберігати всі даних, ні записувати їх, ні давати до них доступ. Шардінг вирішує проблему шляхом горизонтального масштабування. Завдяки даному механізму ми можемо підключати додаткові сервери для зберігання, записи і читання даних.

Для емуляції існування багатьох серверів із СКБД було використано Docker машину. Конфігурація docker-compose наведено у Додатку А.



*рис 1.Схема використаної моделі шардингу MongoDB у даному ПЗ*

В якості ключа для розбивання даних на партіції було використано поле *price* оскільки воно зберігає дані із більш-менш лінійним розподілом. Тож в якості критерія до партиціювання було використано хеш - індекс за властивістю *price.* Із результатами розбиття даних на партиції можна ознайомитися на рис 2.



*рис 2. Розподіл даних між шардами в рамках однієї колекції*

### 

1. Запустити на виконання один або декілька процессів mongod, які стануть вузлами шардингу за допомоги команди у таблиці 3. Також необхідно запустити як мінімум один конфігураційний сервер.

*таблиця 3. Скрипт запуску необхідних вузлів шардингу*

|  |
| --- |
| mongod --dbpath data/shard1 --port 27000 --fork --syslog mongod --dbpath data/shard2 --port 27001 --fork --syslog mongod --configsvr --dbpath data/config --port 27002 --fork --syslog mongos --configdb localhost:27002 --port 27100 --fork --syslog |

1. Дочекатися їх ініцілізації, під’єднатися до процесу mongos сервера-роутера та виконати команду наведену у таблиці 4. Ця команда спочатку прив’яже два сервера-шарди, створить індекс price, по якому дані будуть розподілятися між шардами, а після цього розшардує колекцію по цьому індексу.

*таблиця 4. Скрипт створення нових вузлів шардингу*

|  |
| --- |
| sh.addShard("127.0.0.1:27000")  sh.addShard("127.0.0.1:27001")  use covid  db.cases.ensureIndex({price: 1})  use admin  db.runCommand({shardCollection: "ads.ads", key {price: 1}}) |

**Загальний алгоритм додавання нового вузла реплікації серверів конфігурації**

Запустити на виконання один або декілька процессів mongod, які стануть вузлами реплікації.

1. Дочекатися їх ініцілізації, під’єднатися до процесу mongod одного з них та виконати команду наведену у таблиці 6.

*таблиця 5. Скрипт додавання нового серверу конфігурації до реплікації*

|  |
| --- |
| rs.add( { host: "<hostnameNew>:<portNew>", priority: 0, votes: 0 } ) |

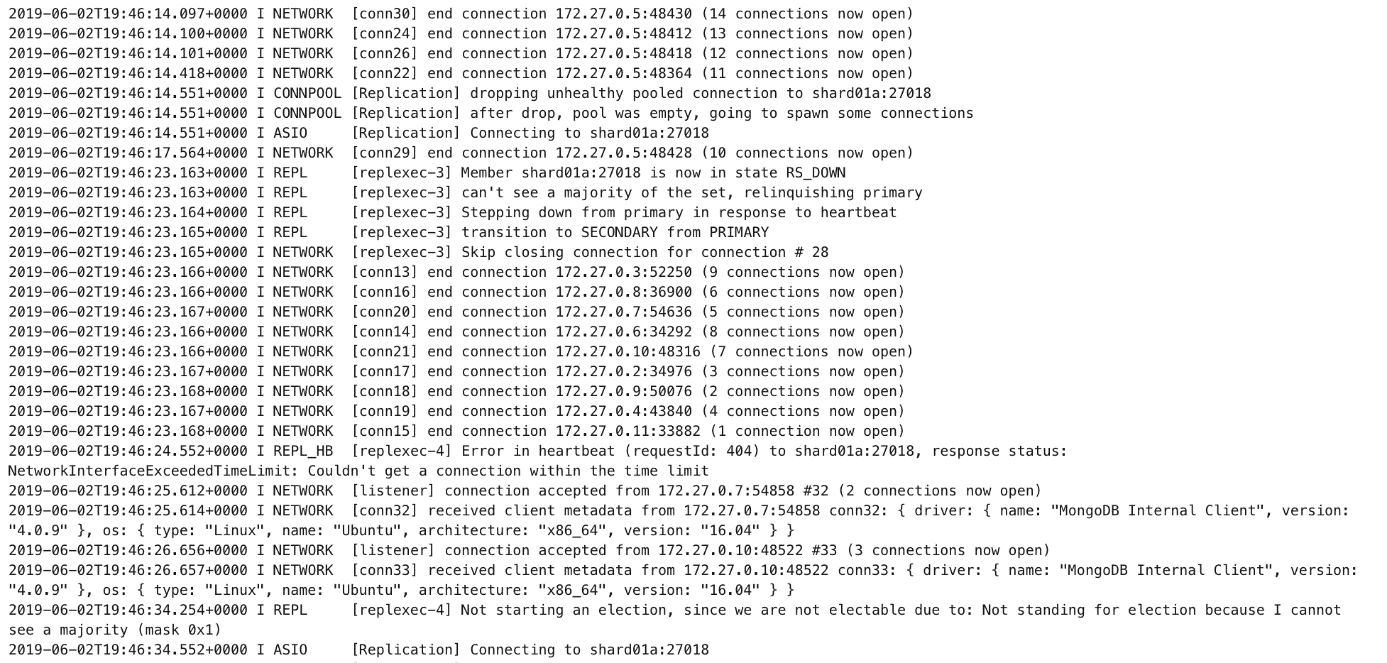
### **Тестування масштабування**

В процесі перевірки здатності системи масштабуватися, було проведено наступні тести:

1. Тестування відмовостійкості шляхом зупинки декількох процесів із репліки сету одного із шардингу
2. Тестування відмовостійкості системи шляхом зупинки декількох процесів із репліки сети серверів конфігурації
3. Перевірка того, яку частину даних ми втратимо, в разі відмови однієї реплікації серверів шардингу

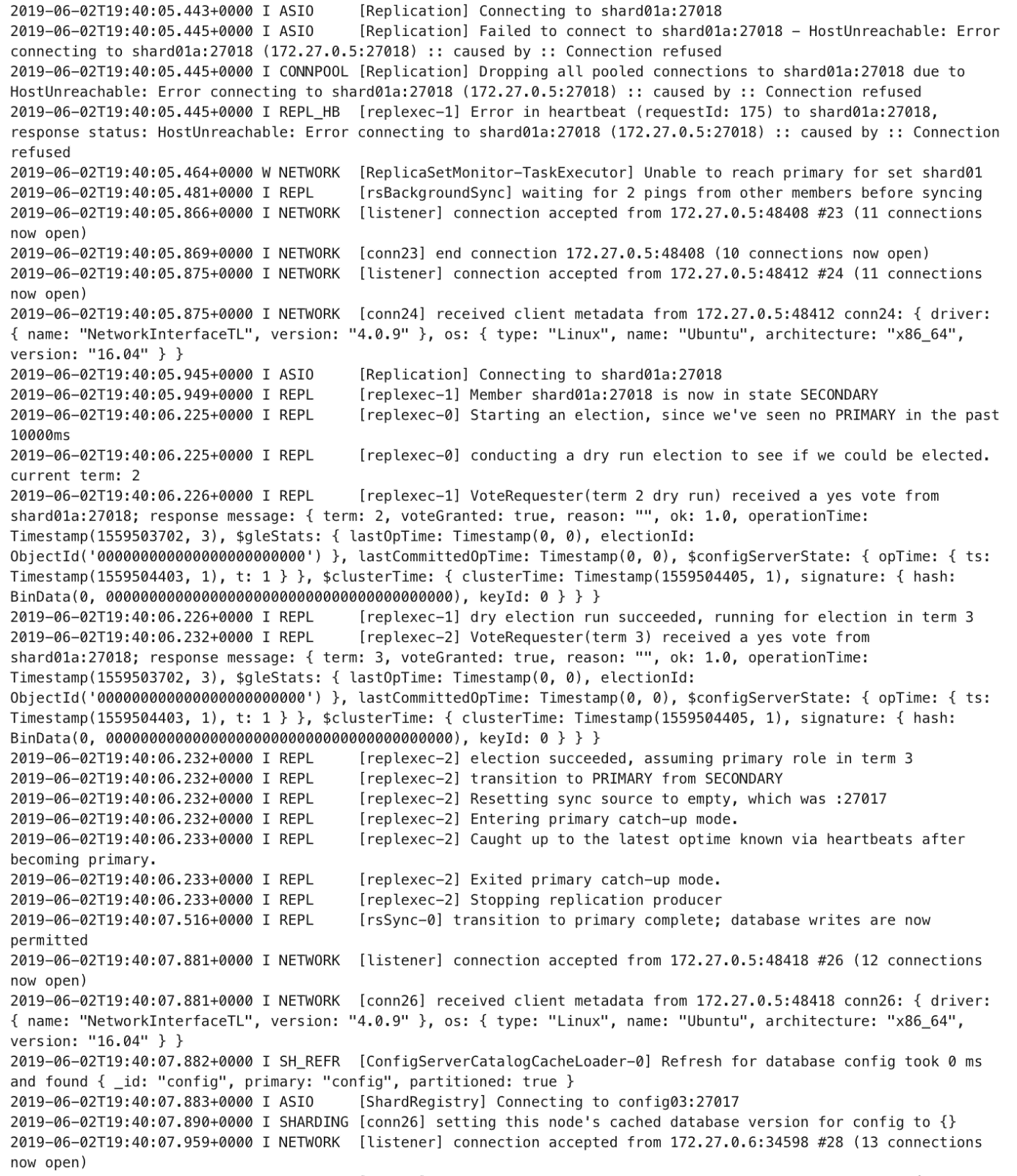
### **Тестування відмовостійкості шляхом зупинки декількох процесів із репліки сету одного із шардингу**

Нами було зупинено процес shard1a в результаті чого, ми очікували, що система перенаправить усі запити, що йшли до нього до його реплікації shard1b, та обере його головним, в разі якщо він не був таким. Вивід процесу mongos серверу маршрутизації наведено на рис. 3.



*рис 3. Вивід процесу shard1b після того, вимкнення shard1a*

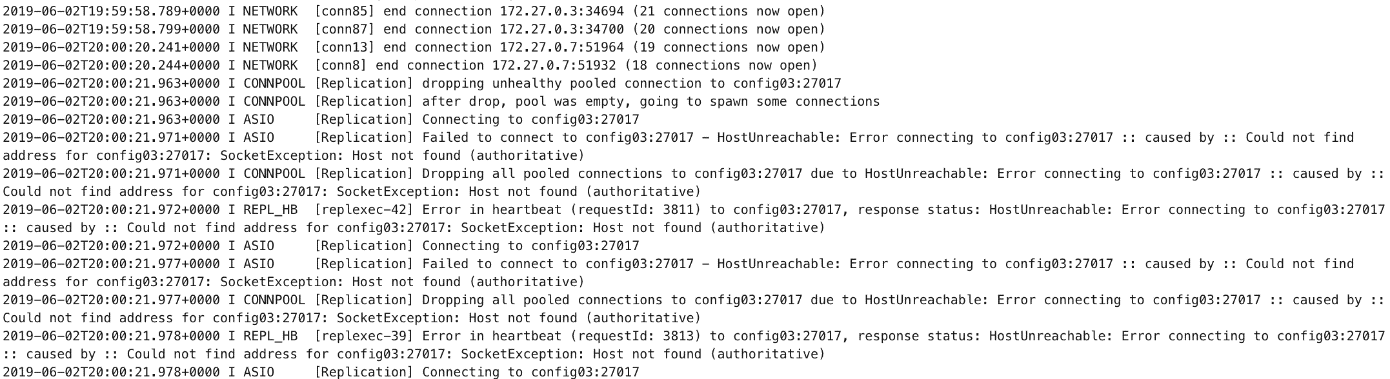
Як видно із виводу, сервер *shard1b* не зміг отримати відповідь від своєї реплікації *shard1a* на запит перевірки “серцебиття” (*en. heartbeat*). В результаті чого він почав надсилати цей запит повторно кожну секунду та взяв на себе відповідальність бути лідером у реплікації із сервером *shard1a.* При цьому усі дані, які зберігалися у цьому кластері шардингу ще доступні на запис та читання. Після відновлення процесу *shard1a*, нами отримано наступний вивід процесу *shard1b,* наведений на рис 4. З нього стає зрозуміло що сервер *shard1b* зміг отримати відповідь від *shard1a* та запустив процес синхронізації та обрання нового лідера реплікації.



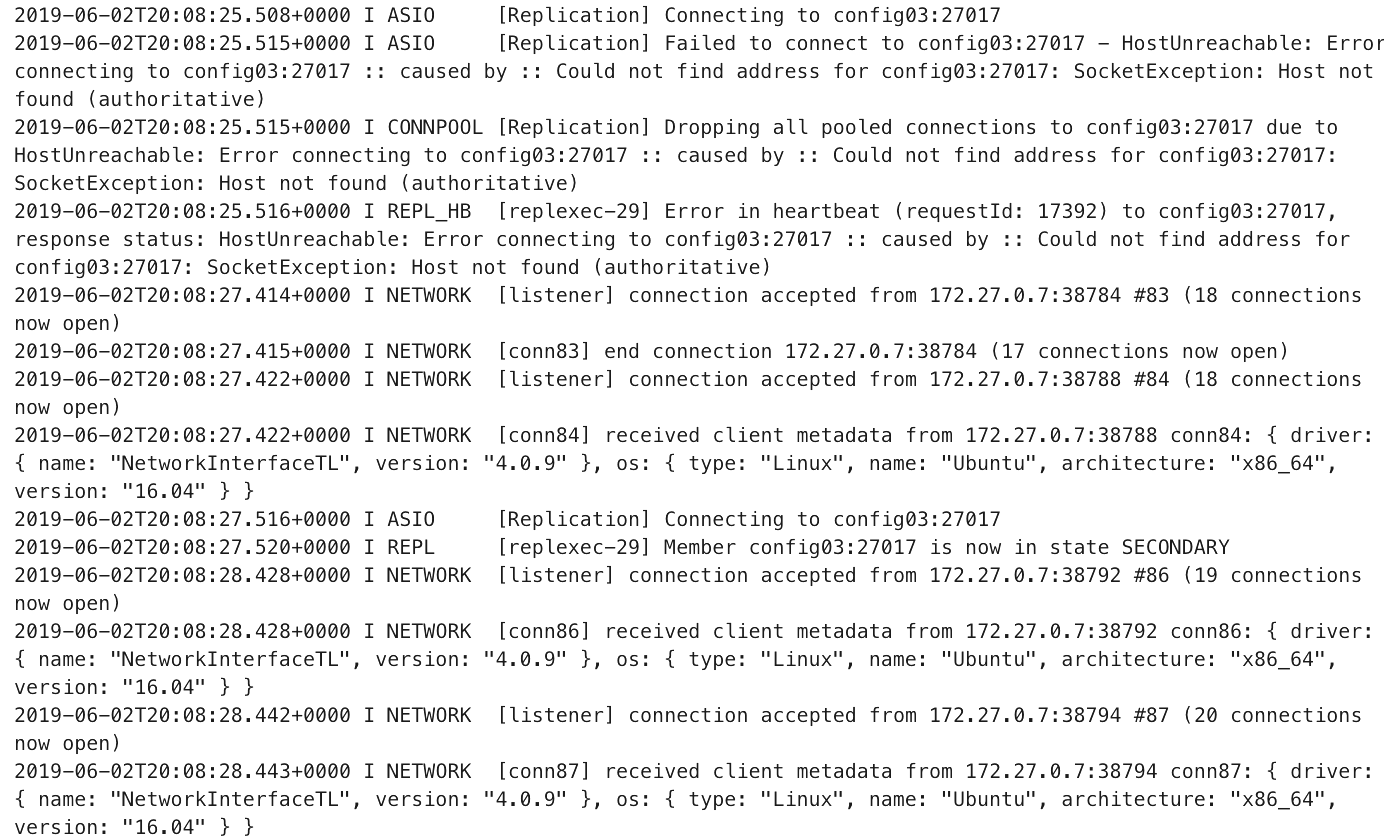
*рис 4. Вивід процесу shard1b після відновлення процесу shard1a*

### **Тестування відмовостійкості системи шляхом зупинки декількох процесів із репліки сети серверів конфігурації**

Сервери конфігурація, які знаходять у реплікації поводять себе так само, як і при тестуванні реплікації кластерів шардів. Після невдачі отримати відповідь на запит перевірки серцебиття (рис 5) від серверу конфігурації (*config1*), кожен з процесів серверу конфігурації, що ще онлайн робить запит до вимкненого сервера кілька раз на секунду. Крім цього відбувається обрання нового лідера реплікації. Після поновлення *config1* відбувається обрання нового лідера реплікації та система переходить до звичайного режиму роботи (рис 6)



*рис 5. Спроби отримати відповідь від вимкненого сервера конфігурації config1*



*рис 6. Створення нового пулу з’єднань із config1 після його поновлення та обрання нового лідера репліки*

### **3. Перевірка того, яку частину даних ми втратимо, в разі відмови однієї реплікації серверів шардингу**

Нами було зупинено процеси *shard1a* та *shard2a*. В результаті чого частина даних, яка зберігалися на цих серверах, повинна бути втрачена, а після поновлення хоча б одного процесу shard1Х вони знову повинні бути доступними. Внаслідок розірвання зв’зку між серверами конфігурації та кластерами шардингу, ми не могли запустити запит на виконання до бд, що стосувався даних у колекції, яка була розбита на партиції. Після поновлення процесів *shard1a* та *shard2a* цілісність даних було відновлено і ми знову змогли виконати запит до колекції.

# **Опис результатів аналізу предметної галузі**

В результаті виконання курсового проекту було проаналізовано дані про оголошення квартир в Україні. Було отримано наступні дані:

1. Згідно із рисунком Б1

Знайдено лінійну залежність ціни квартири від *загальної площі кватрири в м.Києві.*

1. Згідно із рисунком Б2

Знайдено криволінійну залежність ціни квартири від *загальної площі квартири в м.Києві.*

1. Згідно із рисунком Б3

Знайдено лінійну залежність ціни квартири від *кількості кімнат в квартирі в м.Києві.*

1. Згідно із рисунком Б4

Знайдено криволінійну залежність ціни квартири від *кількості кімнат в кватрирі в м.Києві.*

1. Згідно із рисунком Б5

Знайдено лінійну залежність ціни квартири від *кількості кімнат в квартирі в м.Києві.*

1. Згідно із рисунком Б6

Знайдено криволінійну залежність ціни квартири від *кількості кімнат в квартирі в м.Києві.*

1. Згідно із рисунком Б7

Передбачено ціну квартири *в м.Києві відповідно площі, к-сті кімнат та району.*

# **Висновки**

В процесі виконання даного курсового проекту було отримано практичні навички обробки великих масивів даних за допомогою мови програмування Python 3 та СУБД MongoDB.

Було проаналізовано сучасні методи та інструменти для роботи із великими даними, знайдено гарно працюючу комбінацію із мови програмування Python 3 та бібліотек до нього: Scrappy, Scikit-learn, Mathplotlib, Numpy. Було складено порівняльну таблицю із двох баз даних, які найчастіше використовують. На основі цих даних було обрано в якості СУБД NoSQL рішення MongoDB.

Для забезпечення горизонтального масштабування було використано засоби MongoDB, такі як: реплікація та шардинг. Практичним шляхом було знайдено спосіб партиціювання даних часових рядів на основі хеш-функції від мітки часу. Це дозволило розподілити дані, які зберігаються в рамках однієї колекції між багатьма серверами кластерів шардингу. За допомогою реплікації було досягнуто відмовостійкість системи, в результаті чого, під час тестування ми впевнилися, що система продовжує функціонування після виходу із ладу кількох реплік.

На основі зібраних даних було проаналізовано ринок продажу нерухомості (квартир) України. Знайдено кореляції та залежності між різними показниками (кількості кімнат в квартирі, площі та району, де знаходиться квартира). Дані аналізу приведені у Додатку Б.

В ході виконання даного курсового проекту було досягнуто поставленої мети: було набуто практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з NoSQL базами даних, а також були здобуті навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. У результаті виконання курсового проекту я навчилася писати програмне забезпечення для NoSQL баз даних, володіти основами використання СУБД, а також інструментальними засобами аналізу великих обсягів даних, а саме відкритими бібліотеками мови Python.

# **Література**

1. MongoDB от теории к практике. Руководство по установке кластера mongoDB [Електронний ресурс].

– Режим доступу до ресурсу: <https://m.habr.com/ru/post/217393/>;

1. Почему Python так хорош в научных вычислениях [Електронний ресурс].

– Режим доступу до ресурсу:<https://habr.com/post/349482/>;

1. Scrapy: собираем данные и сохраняем в базу данных [Електронний ресурс].

– Режим доступу до ресурсу: <https://habr.com/ru/post/308660/>

1. Python [Електронний ресурс].

– Режим доступу до ресурсу:<https://uk.wikipedia.org/wiki/Python>;

1. Scikit-learn [Електронний ресурс].

– Режим доступу до ресурсу:<https://en.wikipedia.org/wiki/Scikit-learn>;

1. Matplotlib [Електронний ресурс].

– Режим доступу до ресурсу:<https://en.wikipedia.org/wiki/Matplotlib>;

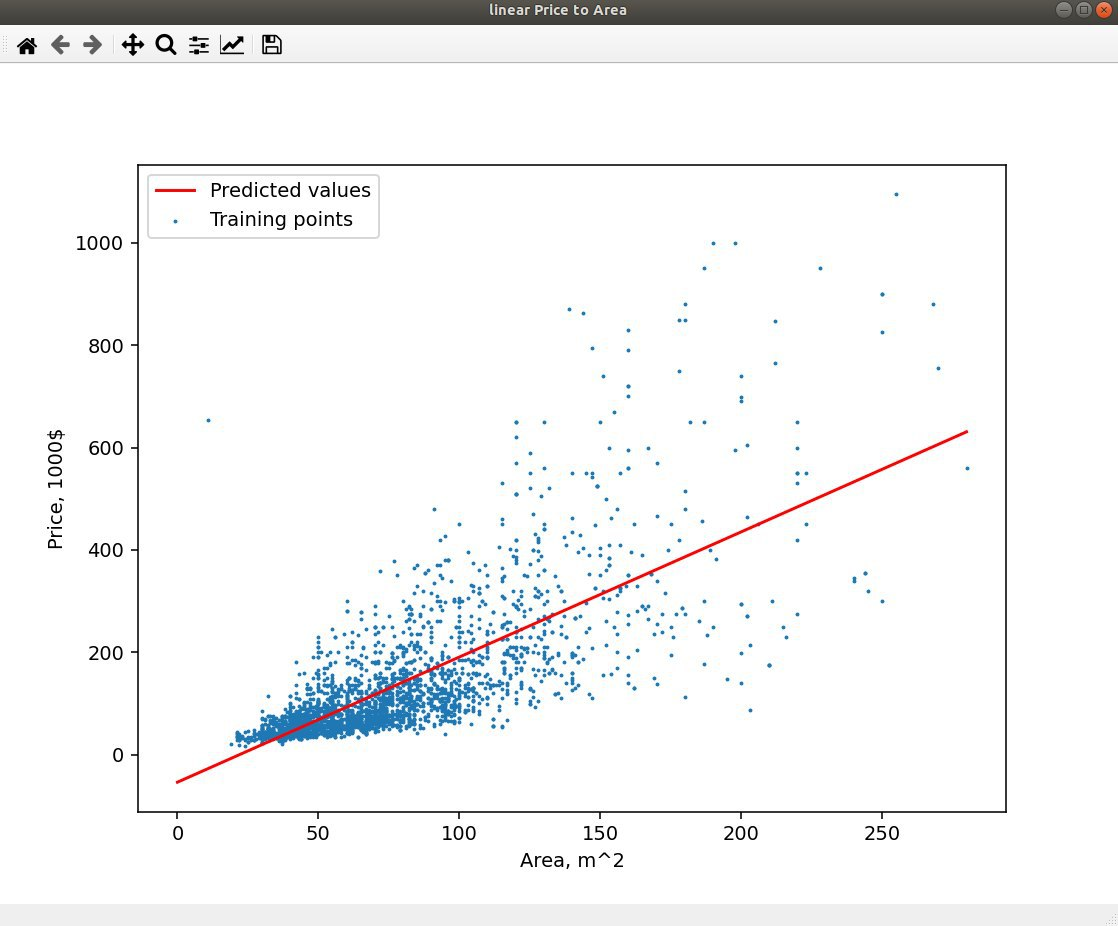
1. NumPy [Електронний ресурс].

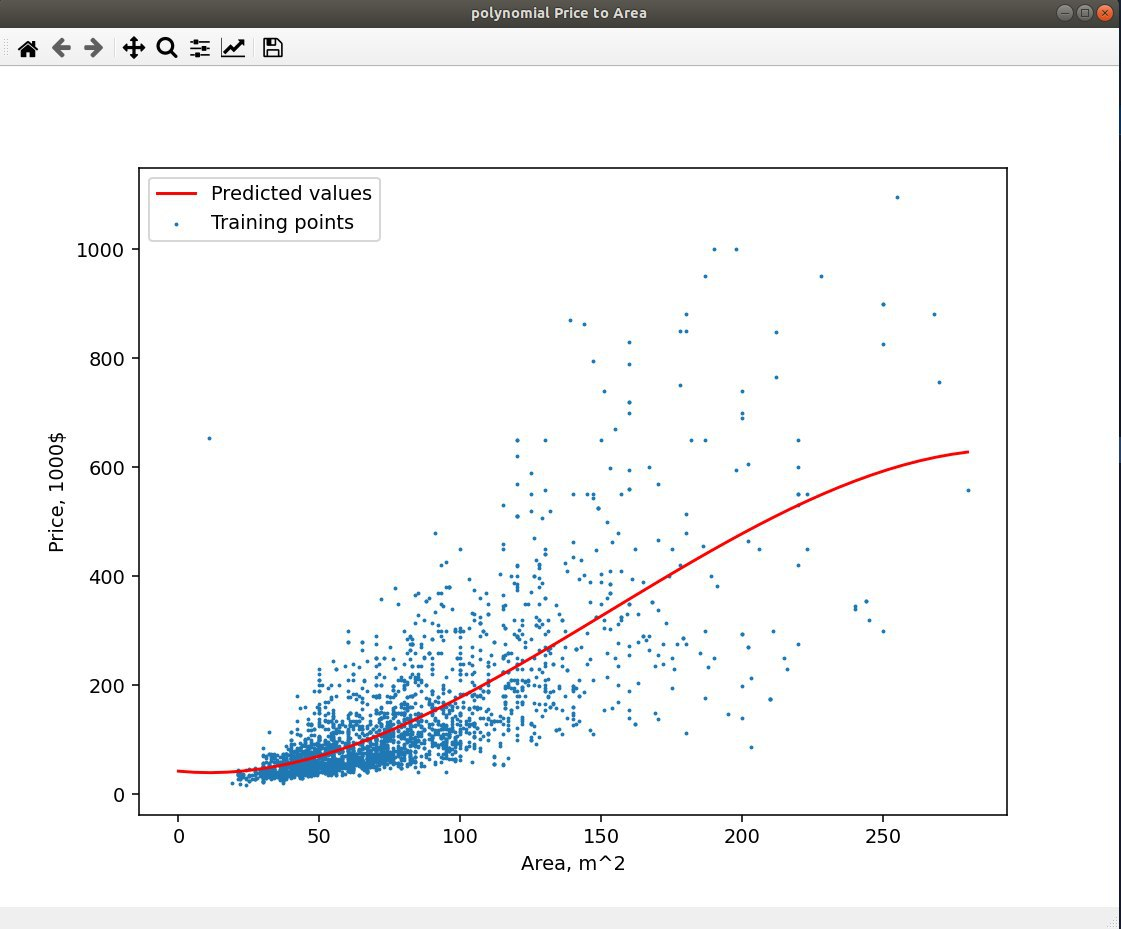
– Режим доступу до ресурсу:<https://en.wikipedia.org/wiki/NumPy>;

**Додаток А**

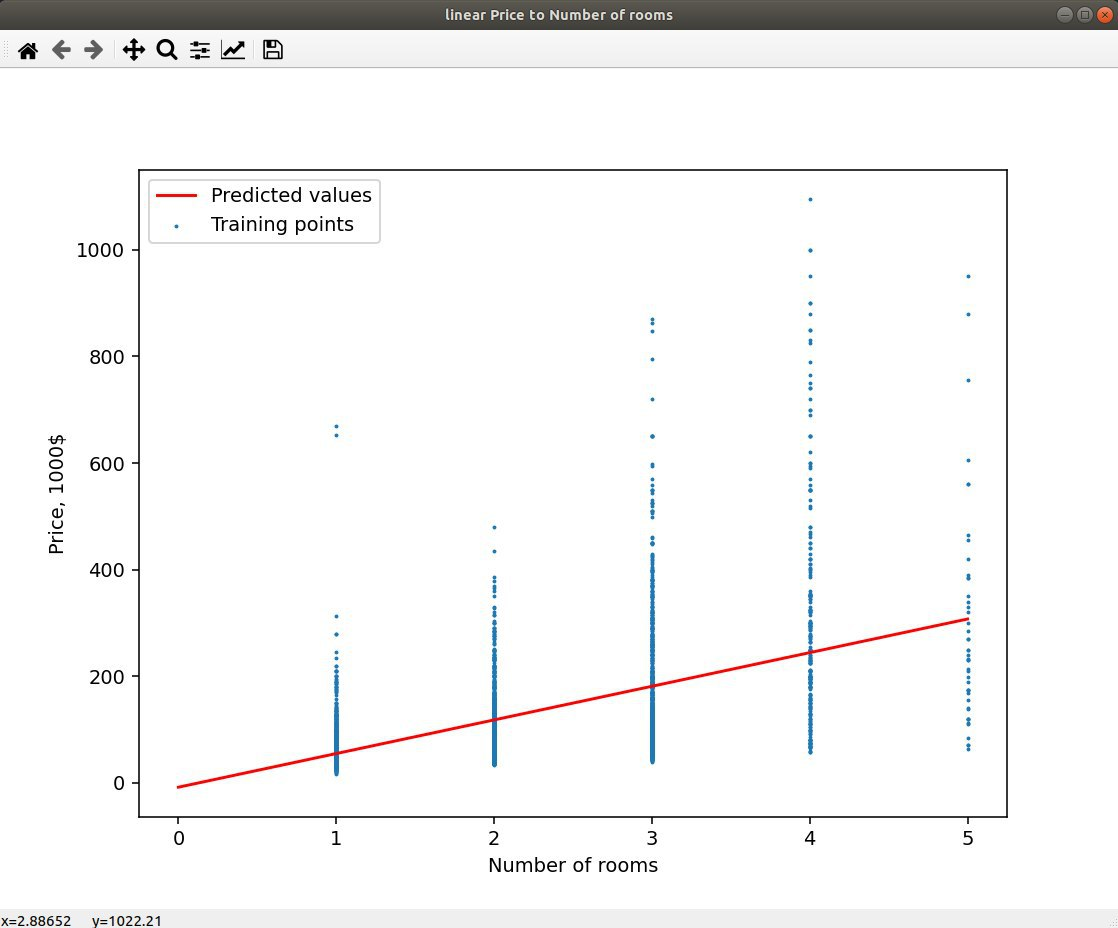
*таблиця А1. Вміст файлу docker-compose.yml*

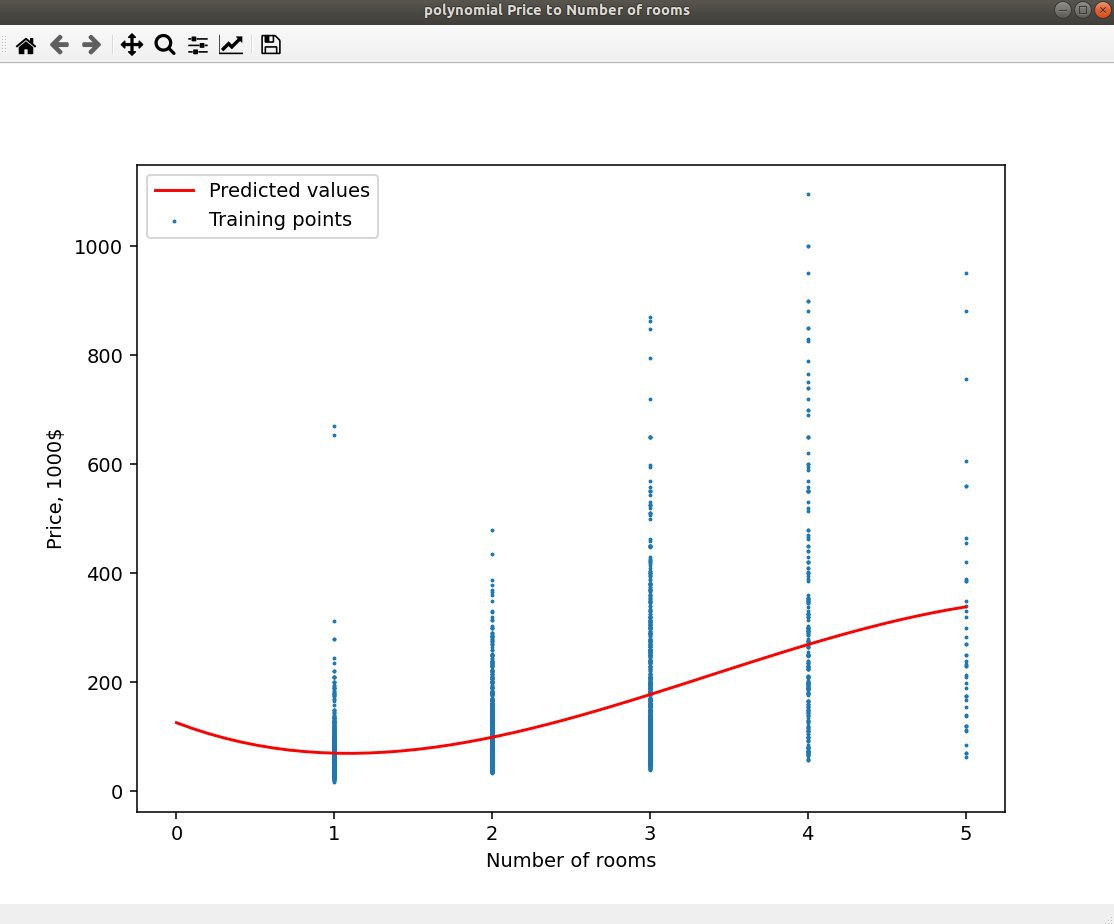
|  |
| --- |
| docker-compose.yml |
| version: '2'  services:  ## Router  router01:  image: mongo:4.0  container\_name: rydell-router-01  command: mongos --port 27017 --configdb rs-config-server/configsvr01:27017,configsvr02:27017,configsvr03:27017 --bind\_ip\_all  ports:  - 27117:27017  volumes:  - ./scripts:/scripts  router02:  image: mongo:4.0  container\_name: rydell-router-02  command: mongos --port 27017 --configdb rs-config-server/configsvr01:27017,configsvr02:27017,configsvr03:27017 --bind\_ip\_all  volumes:  - ./scripts:/scripts  ports:  - 27118:27017  links:  - router01  ## Config Servers  configsvr01:  image: mongo:4.0  container\_name: rydell-mongo-config-01  command: mongod --port 27017 --configsvr --replSet rs-config-server  volumes:  - ./scripts:/scripts  ports:  - 27119:27017  links:  - shard01-a  - shard02-a  - shard03-a  configsvr02:  image: mongo:4.0  container\_name: rydell-mongo-config-02  command: mongod --port 27017 --configsvr --replSet rs-config-server  volumes:  - ./scripts:/scripts  ports:  - 27120:27017  links:  - configsvr01  configsvr03:  image: mongo:4.0  container\_name: rydell-mongo-config-03  command: mongod --port 27017 --configsvr --replSet rs-config-server  volumes:  - ./scripts:/scripts  ports:  - 27121:27017  links:  - configsvr02  ## Shards  ## Shards 01  shard01-x:  image: mongo:4.0  container\_name: rydell-shard-01-node-x  command: mongod --port 27017 --shardsvr --replSet rs-shard-01  volumes:  - ./scripts:/scripts  ports:  - 27122:27017  links:  - shard01-a  - shard01-b  shard01-a:  image: mongo:4.0  container\_name: rydell-shard-01-node-a  command: mongod --port 27017 --shardsvr --replSet rs-shard-01  volumes:  - ./scripts:/scripts  ports:  - 27123:27017  shard01-b:  image: mongo:4.0  container\_name: rydell-shard-01-node-b  command: mongod --port 27017 --shardsvr --replSet rs-shard-01  volumes:  - ./scripts:/scripts  ports:  - 27124:27017  ## Shards 02  shard02-x:  image: mongo:4.0  container\_name: rydell-shard-02-node-x  command: mongod --port 27017 --shardsvr --replSet rs-shard-02  volumes:  - ./scripts:/scripts  ports:  - 27125:27017  links:  - shard02-a  - shard02-b  shard02-a:  image: mongo:4.0  container\_name: rydell-shard-02-node-a  command: mongod --port 27017 --shardsvr --replSet rs-shard-02  volumes:  - ./scripts:/scripts  ports:  - 27126:27017  shard02-b:  image: mongo:4.0  container\_name: rydell-shard-02-node-b  command: mongod --port 27017 --shardsvr --replSet rs-shard-02  volumes:  - ./scripts:/scripts  ports:  - 27127:27017  ## Shards 03  shard03-x:  image: mongo:4.0  container\_name: rydell-shard-03-node-x  command: mongod --port 27017 --shardsvr --replSet rs-shard-03  volumes:  - ./scripts:/scripts  ports:  - 27128:27017  links:  - shard03-a  - shard03-b  shard03-a:  image: mongo:4.0  container\_name: rydell-shard-03-node-a  command: mongod --port 27017 --shardsvr --replSet rs-shard-03  volumes:  - ./scripts:/scripts  ports:  - 27129:27017  shard03-b:  image: mongo:4.0  container\_name: rydell-shard-03-node-b  command: mongod --port 27017 --shardsvr --replSet rs-shard-03  volumes:  - ./scripts:/scripts  ports:  - 27130:27017 |

**Додаток Б**

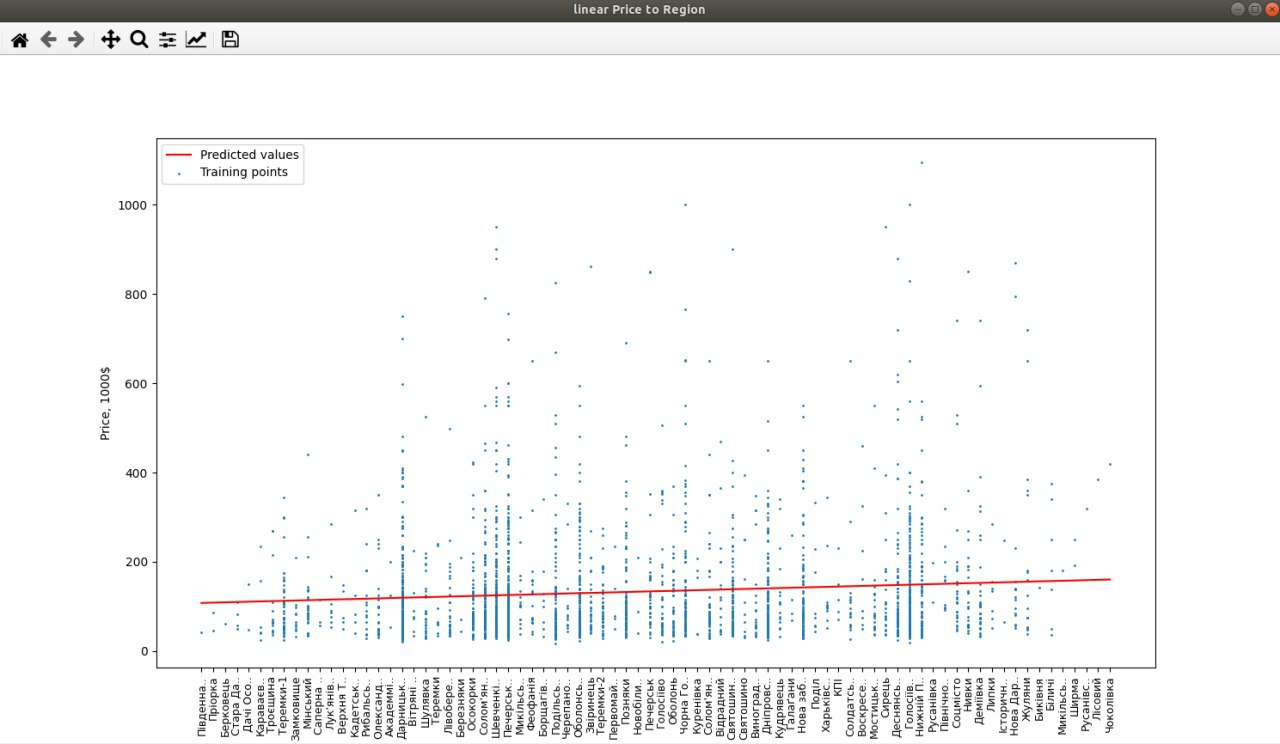
*Рис. Б1 Лінійна регресія залежності ціни квартири від загальної площі кварири*

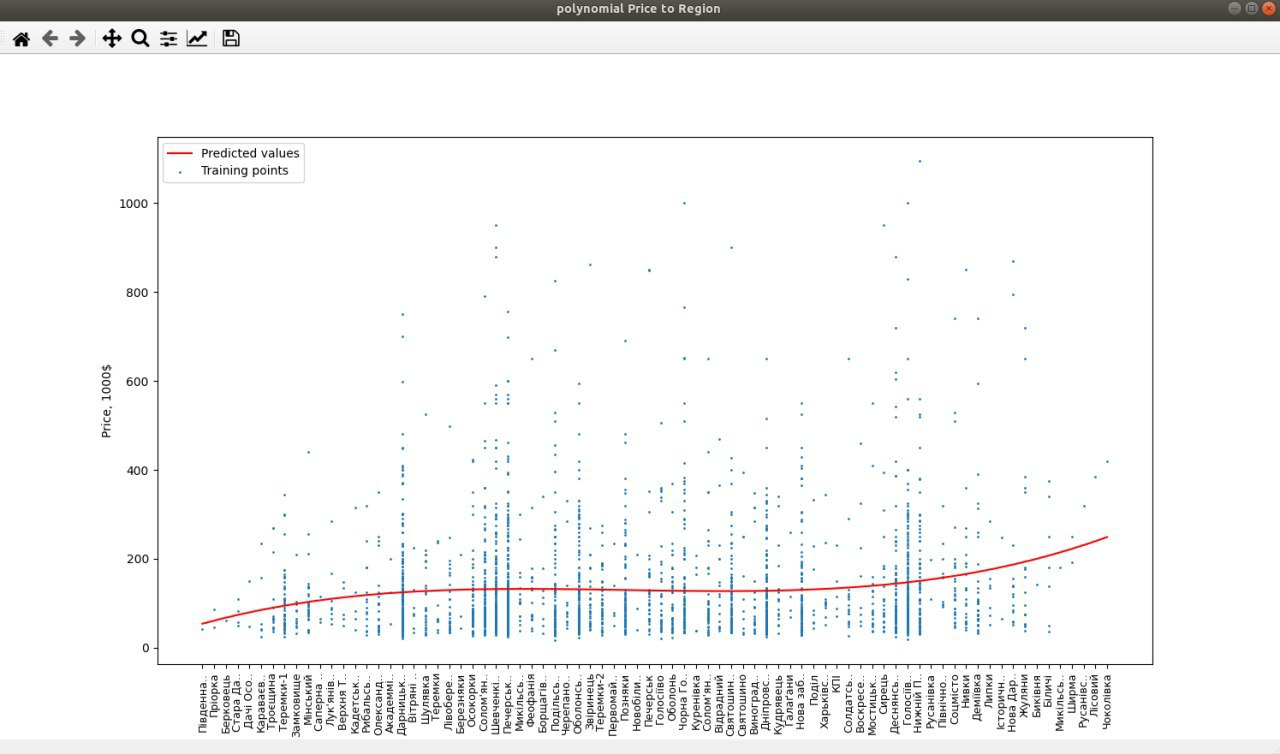
*Рис. Б2 Поліноміальна регресія залежності ціни квартири від загальної площі кварири*

**

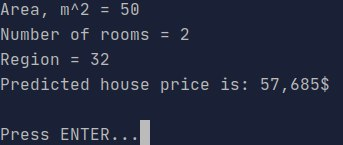
*Рис. Б3 Лінійна регресія залежності ціни квартири від кількості кімнат в кварирі*

*Рис. Б4 Поліноміальна регресія залежності ціни квартири від кількості кімнат в кварирі*



 *Рис. Б5 Лінійна регресія залежності ціни квартири від району*

*Рис. Б6 Поліноміальна регресія залежності ціни квартири від району*



*Рис. Б7 Прогнозування ціни квартири в залежності від її характеристик (площа, к-сть кімнат, район)*