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 кліматичних показників та якості повітря»

Студент

групи КП-72 Абу Шамала Амір Махер _____

(підпис)

Викладач

к.т.н, доцент кафедри

СПіСКС **Петрашенко А.В.** _____

(підпис)

Захищено з оцінкою _____

Київ – 2020

Анотація

У даній курсовій роботі була створена моніторингова система кліматичних показників та якості повітря.

Галуззю застосування даної розробки є метеорологія та екологія.

Були отримані реальні дані про погодні умови та якість повітря із відкритого ресурсу[1].

У основному модулі програми відбувається збір, фільтрація та аналіз кліматичних показників, якості повітря різних країн, міст та дат з метою прогнозування забруднення повітря або інших кліматичних показників.

Також у основному модулі наявне візуальне представлення залежності динаміки забруднення повітря із іншими показниками (швидкість вітру, тиск, вологість та ін.).

Результатами даного проекту стали діаграми та графіки, що зображають результати аналізу погодних умов та забруднення повітря у світі. З ними можна ознайомитися в додатку А.

Зміст

Вступ

Була створена моніторингова система кліматичних показників та забруднення повітря.

Важко недооцінити важливість впливу якості повітря на здоров’я населення. Сучасний світ має проблеми з екологією та відносно нову науку Data Science, тому актуально розробити таку моніторингову систему, щоб спростити аналіз ситуації як локально у місті або країні так і у світі глобально.

Даний проект призначений для обробки інформації про погодні показники світу та їх аналіз. За допомогою розробленого програмного забезпечення можно вести статистику про зміну погодних умов та якості повітря, відслідкувати країни з гіршим або кращим станом повітря, аналізувати кореляцію між різними показниками для подальшого прогнозування різних умов.

Також метою даного курсового проекту є набуття навичок з масштабування високонавантажених системи, роботою з Big Data, науковими бібліотеками мови програмування Python 3 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: MSSQL, MongoDB.

З порівняльною характеристикою цих СУБД можна ознайомитися в таблиці 1.

Таблиця 1. Порівняльна характеристика СУБД

Властивість\Критерій	Назва СУБД	
	MongoDB	MSSQL
Реляційні дані	ні	так
Форма збереження даних	документи JSON	таблиці
Підтримка ієрархічних даних	так	так
Транзакції	ні	так
Атомарність операцій	всередині документа	по всій БД
Мова запитів	JSON/JavaScript	SQL
Підтримка шардингів	так	так (важка конфігурація)
Наявність бібліотек для мови	так	так

програмування Python 3		
Реплікація	так, автоматичне переобрання головного процесу	за принципом Master Slave

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 3.6. 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.

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
temp	Integer	Температура повітря за цельсієм
humidity	Integer	Відносна вологість повітря, %
wind_speed	Float	Швидкість вітру, км/год
pressure	Integer	Тиск, мілібар
aqi	Integer	Індекс якості повітря, PM2.5
country	String	Країна вимірюваних показників
city1	String	Місто, штат або інш. одиниця вимірюваних показників
city2	String	Місто, штат або інш. одиниця вимірюваних показників
date	String	Дата актуальності даних показників
origin_url	String	Посилання на сторінку зібраних даних

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 локальну машину з різними портами. Конфігурацію та скрипти наведено у Додатку А.

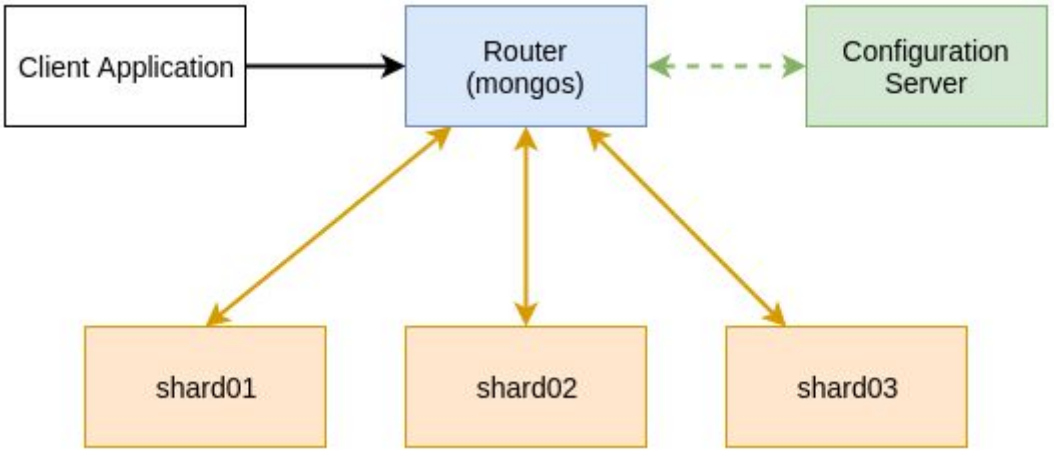

Рис 1.Схема використаної моделі шардингу MongoDB у даному ПЗ

Оскільки дані у БД можуть сильно відрізнятись було обрано в якості ключа для розбивання даних на партиції звичайне хешування за полем "_id". Із результатами розбиття даних на партиції можна ознайомитися на рис 2.

```
mongos> db.air_quality_data.getShardDistribution()

Shard Shard1AQIMonitoringReplSet at Shard1AQIMonitoringReplSet/localhost:27001,localhost:27002
 data : 4.53MiB docs : 24239 chunks : 2
 estimated data per chunk : 2.26MiB
 estimated docs per chunk : 12119

Shard Shard2AQIMonitoringReplSet at Shard2AQIMonitoringReplSet/localhost:27004,localhost:27005
 data : 4.57MiB docs : 24443 chunks : 2
 estimated data per chunk : 2.28MiB
 estimated docs per chunk : 12221

Totals
 data : 9.11MiB docs : 48682 chunks : 4
 Shard Shard1AQIMonitoringReplSet contains 49.77% data, 49.79% docs in cluster, avg obj size on shard : 196B
 Shard Shard2AQIMonitoringReplSet contains 50.22% data, 50.2% docs in cluster, avg obj size on shard : 196B


mongos>
```

Рис 2. Розподіл даних між шардами в рамках однієї колекції

1. Автоматичний запуск шардингу локально або вручну з пункту 2.

Таблиця 3. Автоматичний запуск шардингу локально

```
cd sharding/
sh start.sh
```

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

Таблиця 4. 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
```

3. Дочекатися їх ініціалізації, 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 5. Скрипт запуску необхідних вузлів шардингу

```
sh.addShard("localhost:27000")
sh.addShard("localhost:27001")
use air_quality_db
db.cases.ensureIndex({_id: "hashed"})
use admin
db.runCommand({shardCollection: "air_quality_db.air_quality_data", key {"_id":
"hashed"}})
```

Загальний алгоритм додавання нового вузла реплікації серверів конфігурації

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

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

Таблиця 6. 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.

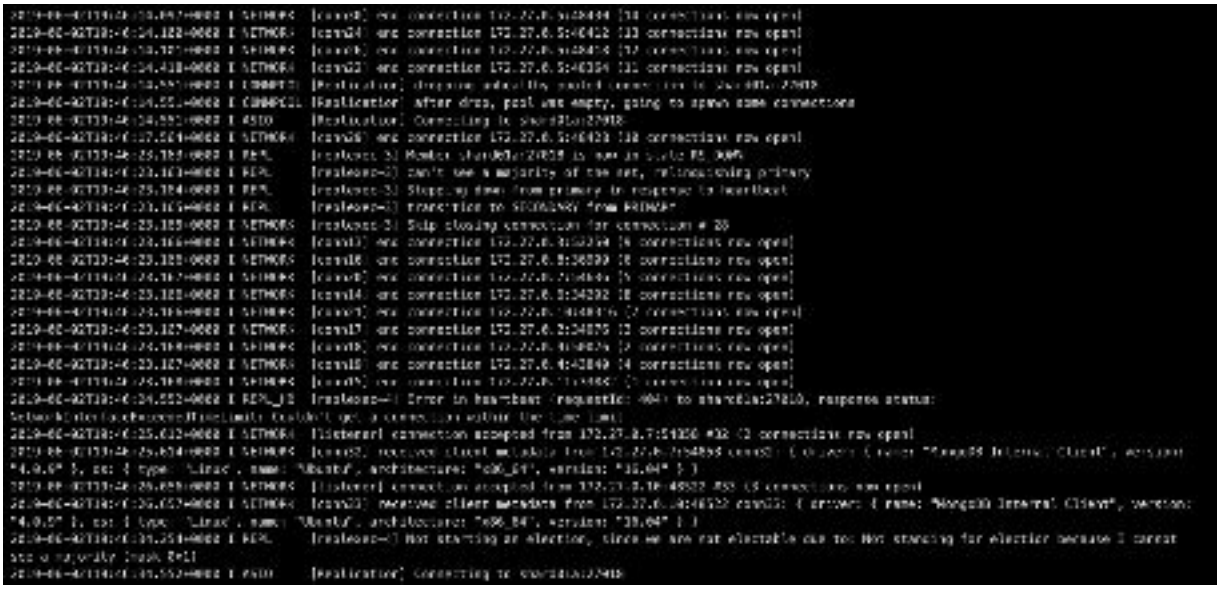

Рис 3. Вивід процесу shard1b після того, вимкнення shard1a

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.

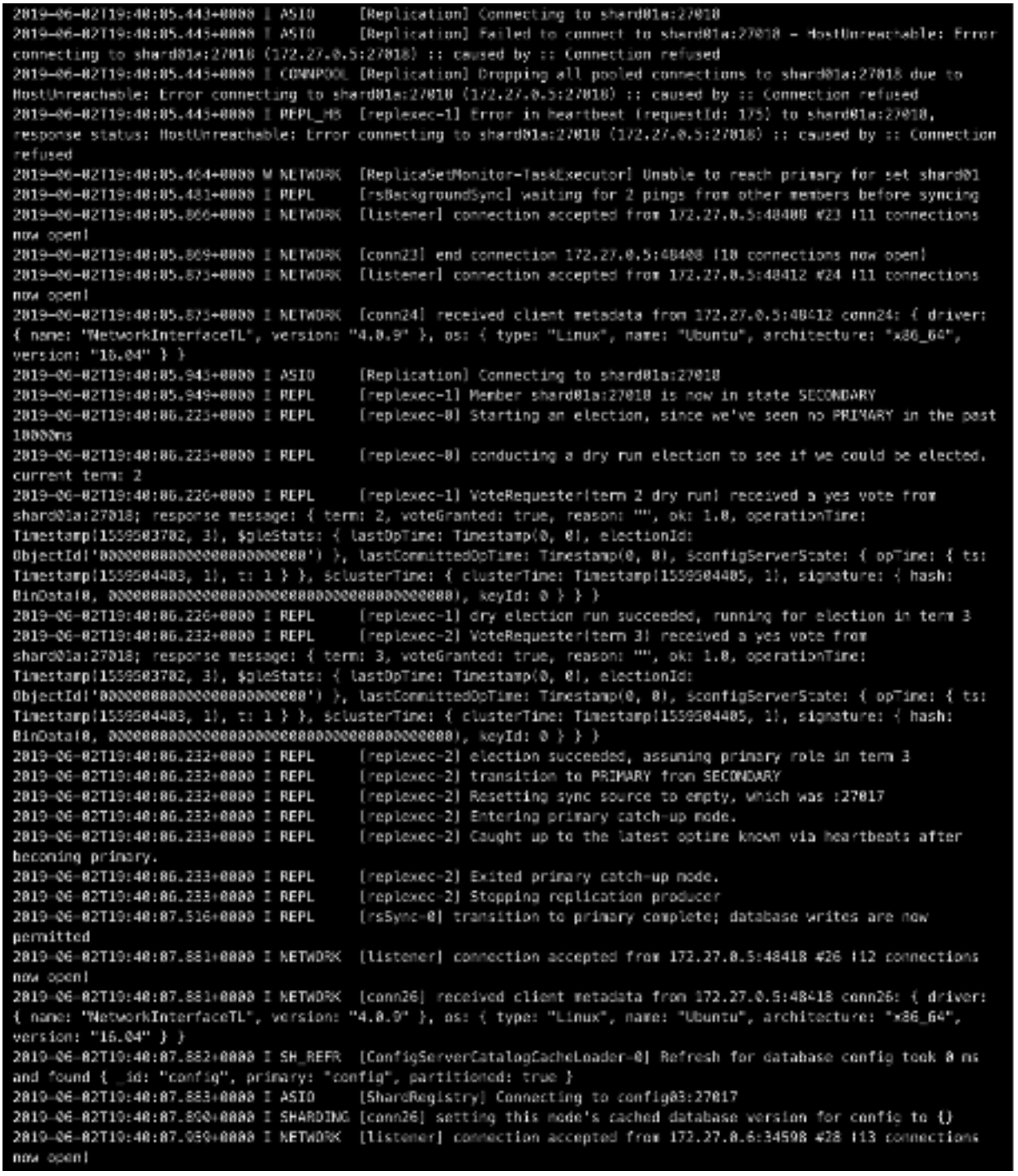

Рис 4. Вивід процесу shard1b після відновлення процесу shard1a

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

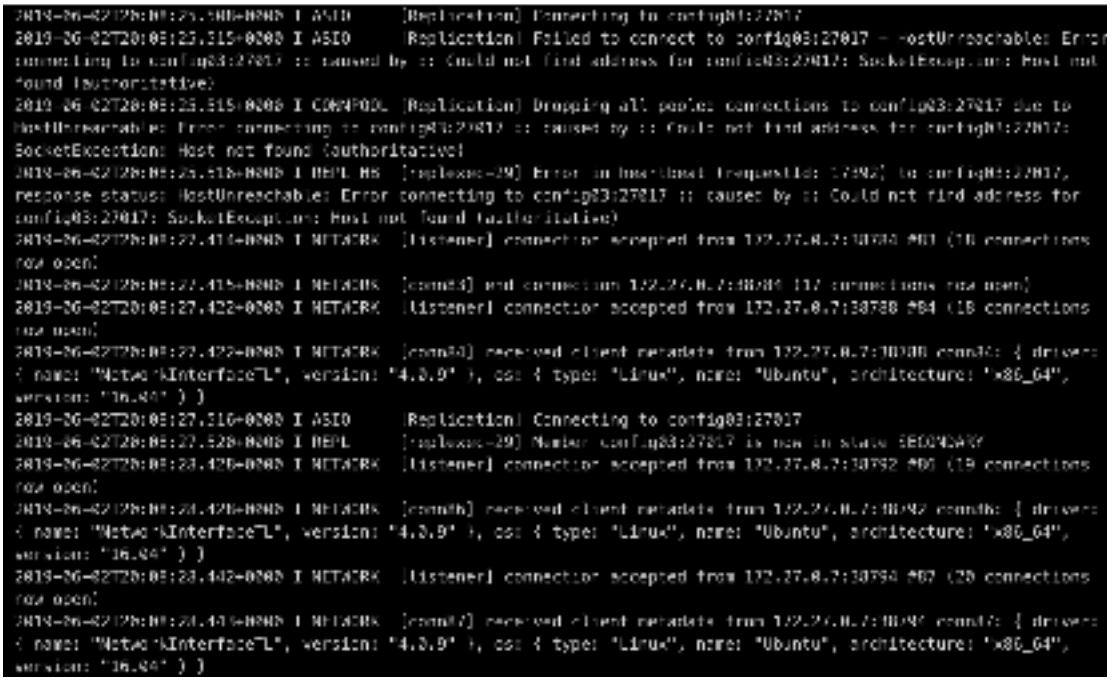

Рис 6. Створення нового пулу з’єднань із config1 після його поновлення та обрання нового лідера репліки

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 shard1X вони знову повинні бути доступними. Внаслідок розірвання зв'язку 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 забруднення повітря та кліматичні показники. Деякі дані неточні, оскільки присутній вплив інших факторів (особливості місцевості і т.д.)

Було отримано наступні дані:

1. Згідно із рисунком Б1

   Знайдено лінійну залежність *індексу якості повітря від тиску*
2. Згідно із рисунком Б2

   Була спроба знайти поліноміальну залежніть *індексу якості повітря від тиску*, але крива некоректна
3. Згідно із рисунком Б3

   Знайдено *слабку* лінійну залежність *індексу якості повітря від вологості*
4. Згідно із рисунком Б4

   Знайдено *слабку* поліноміальну залежність *індексу якості повітря від вологості*
5. Згідно із рисунком Б5

   Було складено топ 10 найчистіших країн у світі за індексом якості повітря
6. Згідно із рисунком Б6

   Було складено топ 10 країн з гіршим індексом якості повітря
7. Згідно із рисунком Б7

   Знайдено лінійну залежність *індексу якості повітря від швидкості вітру*
8. Згідно із рисунком Б8

   Була спроба знайти поліноміальну залежніть*індексу якості повітря від швидкості вітру*, але крива некоректна
9. Згідно із рисунком Б9

   Знайдено лінійну залежність *вологості повітря від тиску*
10. Згідно із рисунком Б10

    Знайдено поліноміальну залежність *вологості повітря від тиску*

11. Згідно із рисунком Б11

Знайдено лінійну *залежності температури від вологості*

12. Згідно із рисунком Б12

Знайдено поліноміальну *залежності температури від вологості*

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, Matplotlib, Numpy.

Для забезпечення горизонтального масштабування було використано засоби MongoDB, такі як: реплікація та шардинг. Було обрано хешований спосіб партиціювання даних за ідентифікатором запису. Це дозволило розподілити дані, які зберігаються в рамках однієї колекції між багатьма серверами. 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 кліматичні показники та забруднення повітря по всьому світу. Знайдено кореляції та залежності між різними показниками (забруднення повітря та швидкість вітру, вологість та тиск і т.д.). Кореляції слабкі, оскільки дані незбалансовані (по деяким країнам даним набагато більше) та на коректність впливає багато інших факторів (наприклад геопозиція країни, к-ість фабрик та машин і т.д.) Дані аналізу приведені у Додатку Б.

В ході виконання даного курсового проекту було набуто практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з NoSQL БД, а також навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. Я навчився 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. Explore the air quality anywhere in the world
   https://www.iqair.com/
2. Scrapy: An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.
   https://scrapy.org//
3. Python — Wikipedia
   https://uk.wikipedia.org/wiki/Python
4. Scikit-learn — Wikipedia
   https://en.wikipedia.org/wiki/Scikit-learn
5. Matplotlib — Wikipedia
   ttps://en.wikipedia.org/wiki/Matplotlib
6. NumPy — Wikipedia
   https://en.wikipedia.org/wiki/NumPy
7. Sharding — MongoDB Manual
   https://docs.mongodb.com/manual/sharding/

Додаток А

Таблиця А1. Вміст файлу configsvr_start.sh

configsvr_start.sh

```
mkdir -p configsvr1 configsvr2 configsvr3

mongod --configsvr --replSet configserver --port 27007 --dbpath configsvr1 --fork
--logpath configsvr1/configsvr1.log
mongod --configsvr --replSet configserver --port 27008 --dbpath configsvr2 --fork
--logpath configsvr2/configsvr2.log
mongod --configsvr --replSet configserver --port 27009 --dbpath configsvr3 --fork
--logpath configsvr3/configsvr3.log

sleep 1s
ps -ef | grep mongo

mongo -port 27007 configsvr_init.js
```

Таблиця А2. Вміст файлу configsvr_init.js

configsvr_init.js

```
config =
    {"_id" : "configserver",
        configsvr: true,
        version: 1,
        members : [
            {"_id" : 0, host : "localhost:27007"},
            {"_id" : 1, host : "localhost:27008"},
            {"_id" : 2, host : "localhost:27009"}
        ]
    };

rs.initiate(config);
rs.status();
```

Таблиця А3. Вміст файлу start.sh

start.sh

```
cd ./shard1
sh shard1_start.sh

cd ../shard2
sh shard2_start.sh

cd ../configsvr
sh configsvr_start.sh

cd ../routerdb
sh routerdb_start.sh
```

Додаток Б

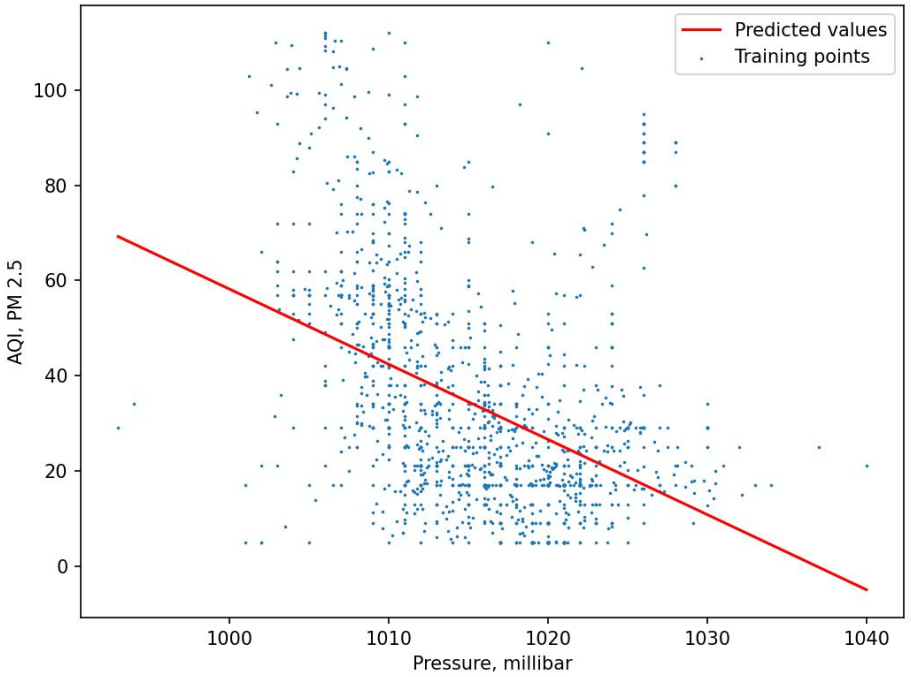

Рис. Б1 Лінійна регресія залежності індексу якості повітря від тиску

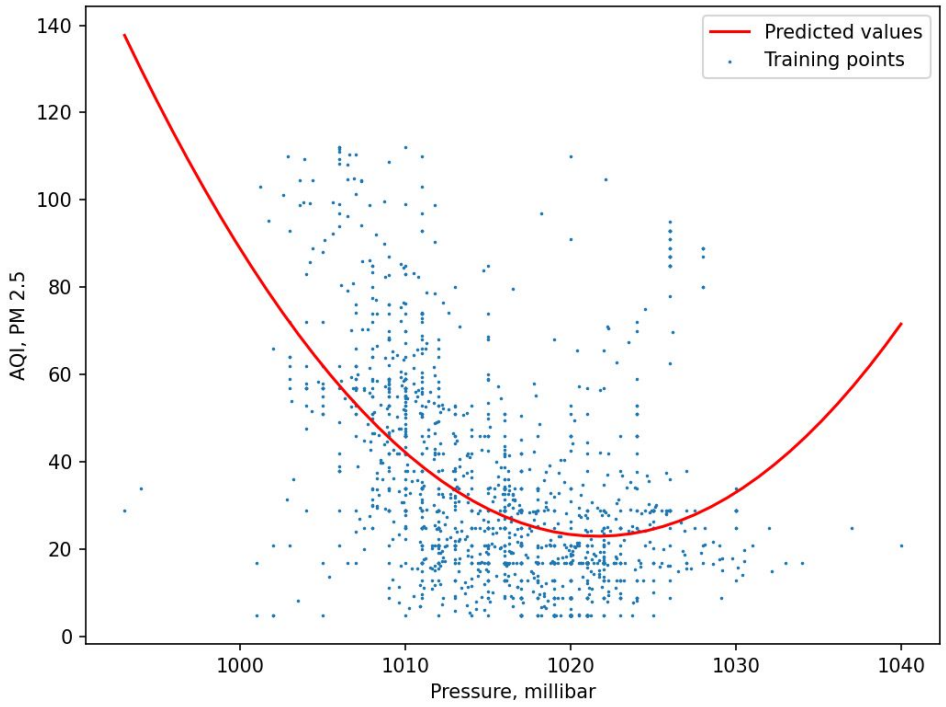

Рис. Б2 Поліноміальна регресія залежності індексу якості повітря від тиску

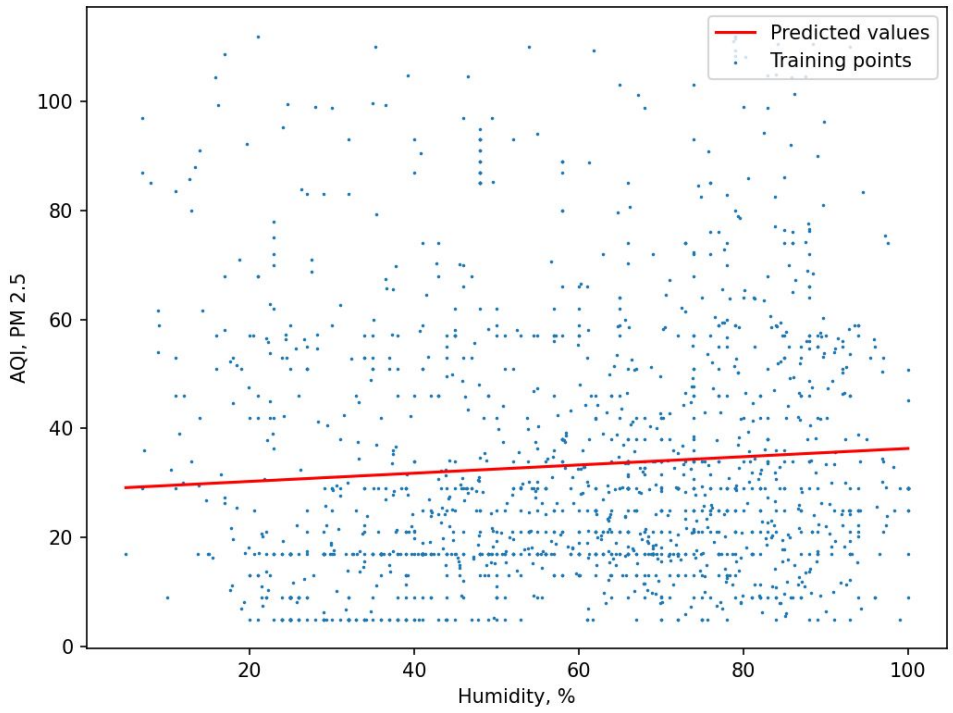

Рис. Б3 Лінійна регресія залежності індексу якості повітря від вологості

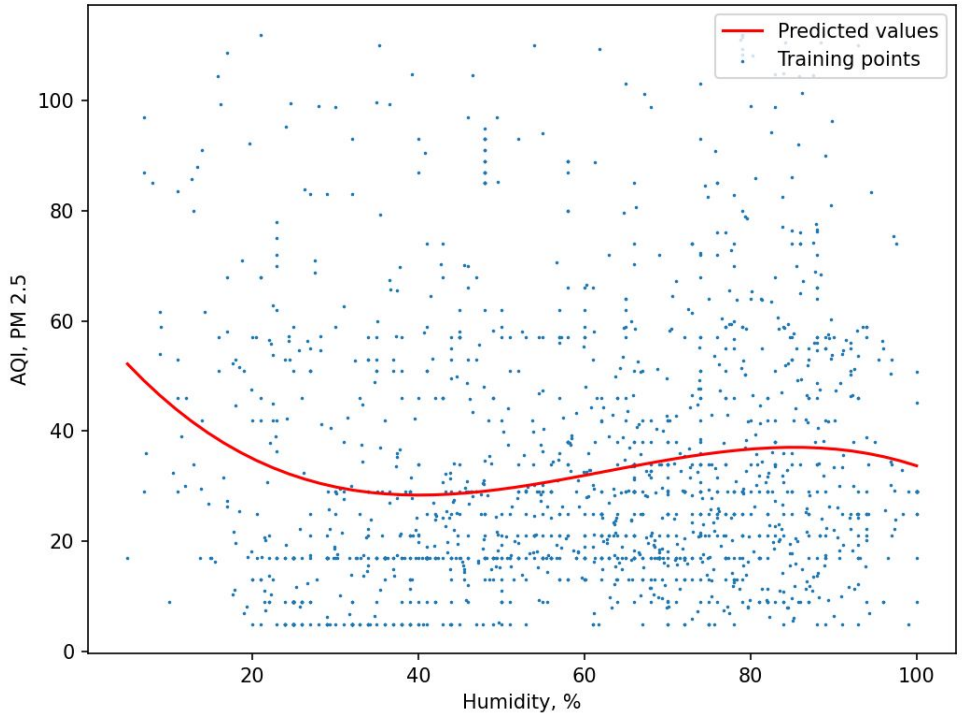

Рис. Б4 Поліноміальна регресія залежності індексу якості повітря від вологості

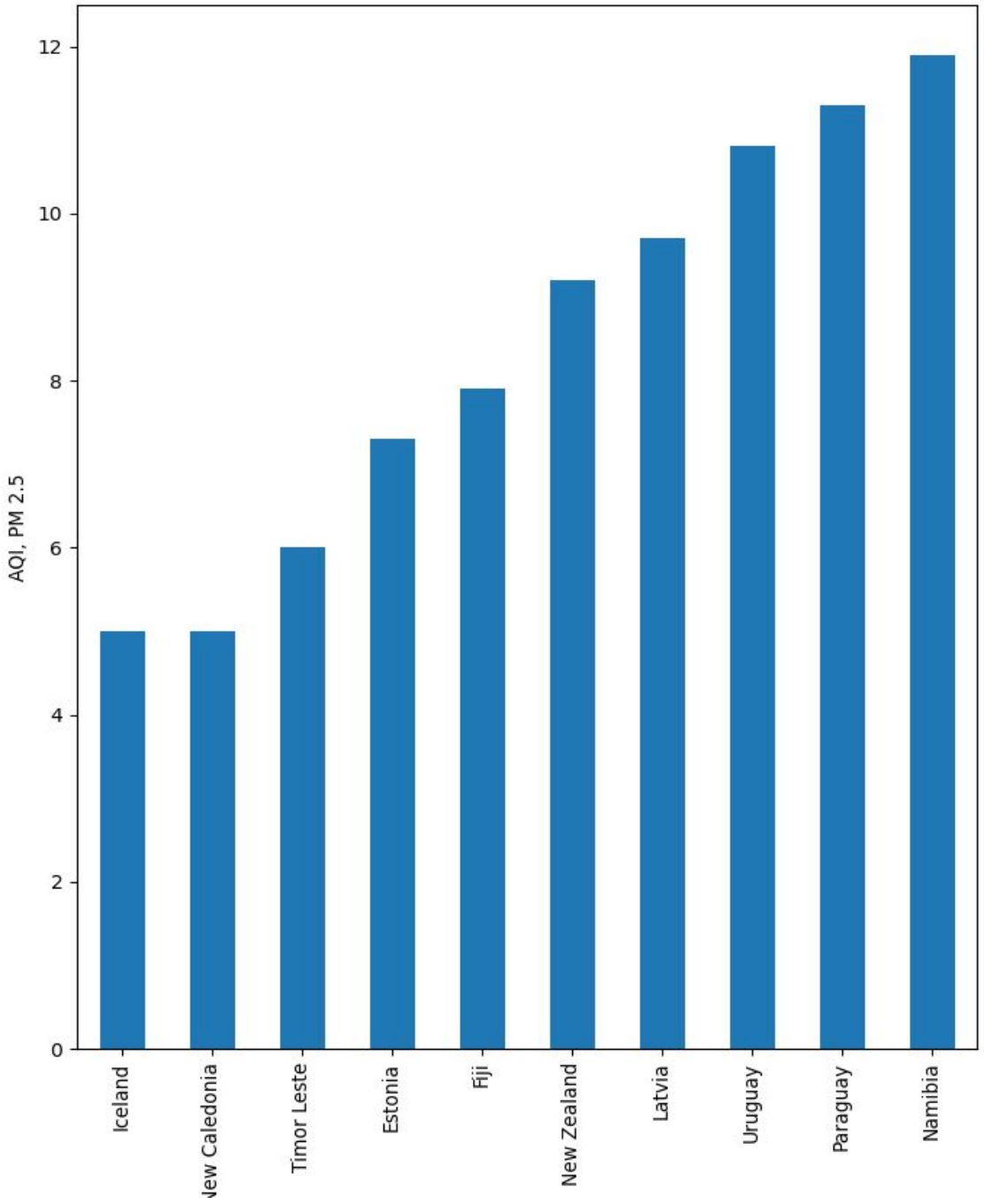

Рис. Б5 Топ 10 найчистіших країн у світі за індексом якості повітря

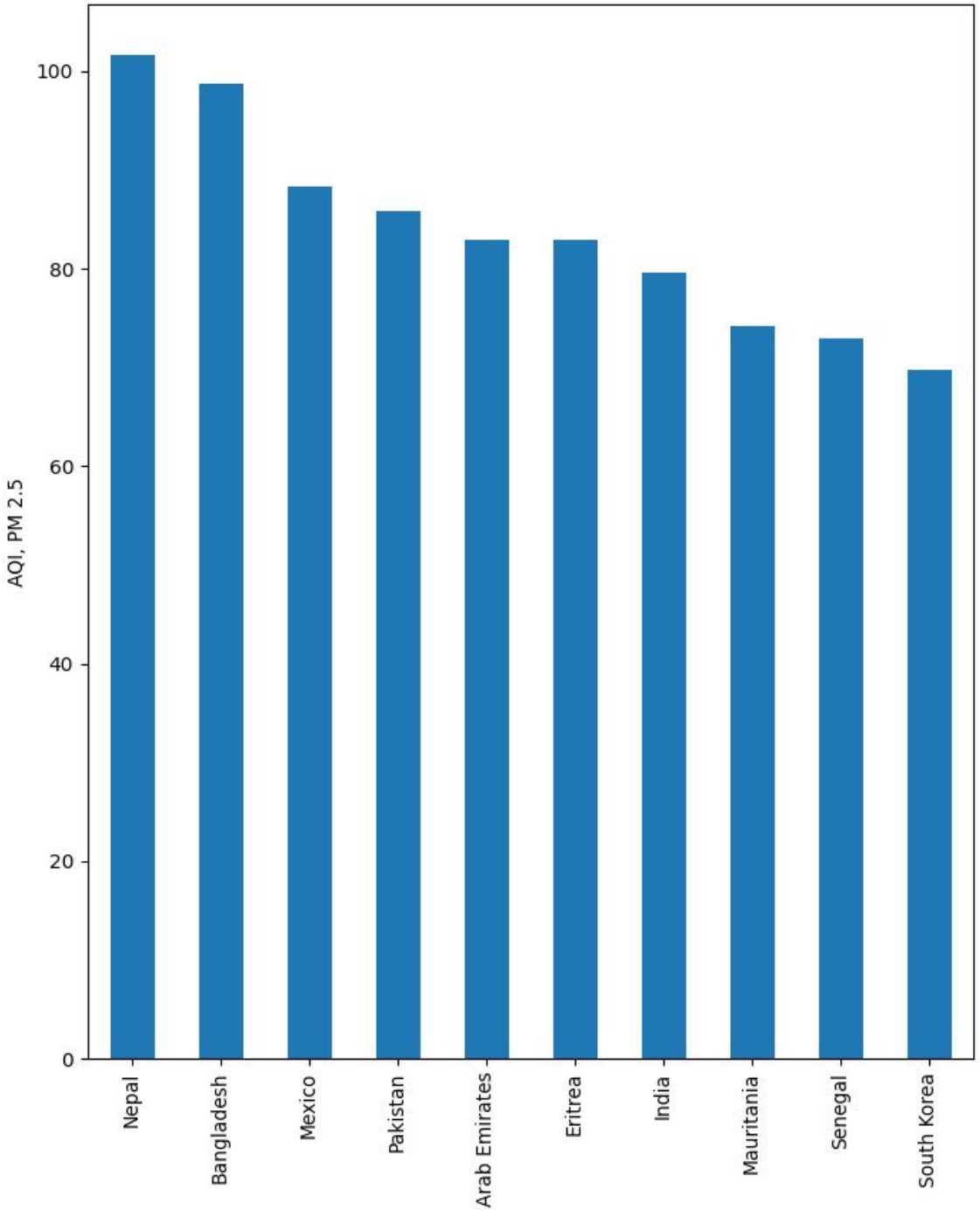

Рис. Б6 Топ 10 країн з гіршим індексом якості повітря

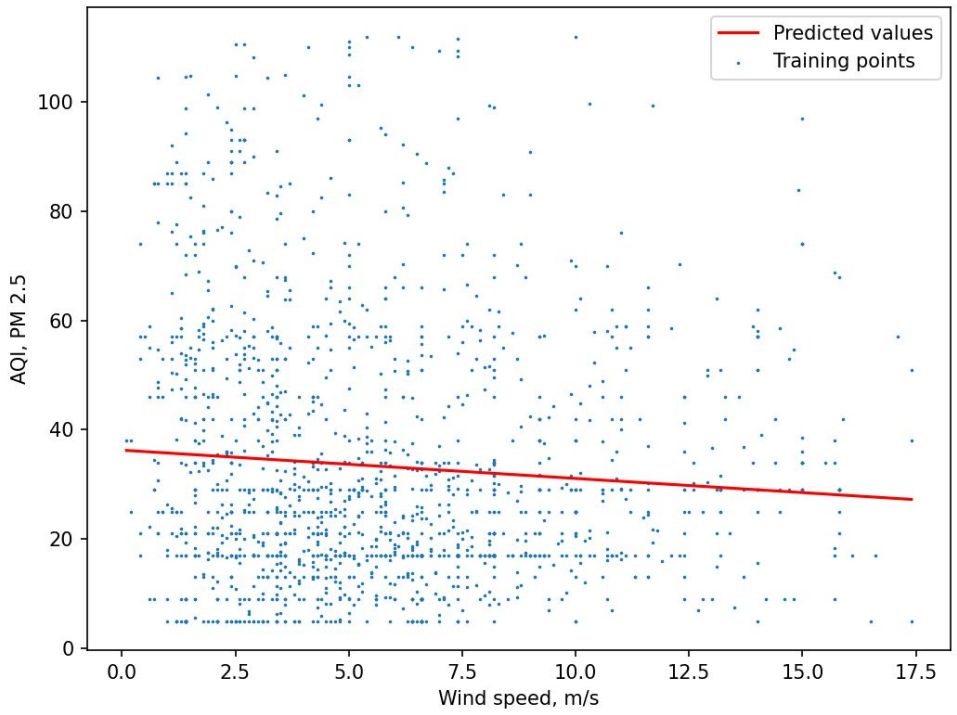

Рис. Б7 Лінійна регресія залежності індексу якості повітря від швидкості вітру

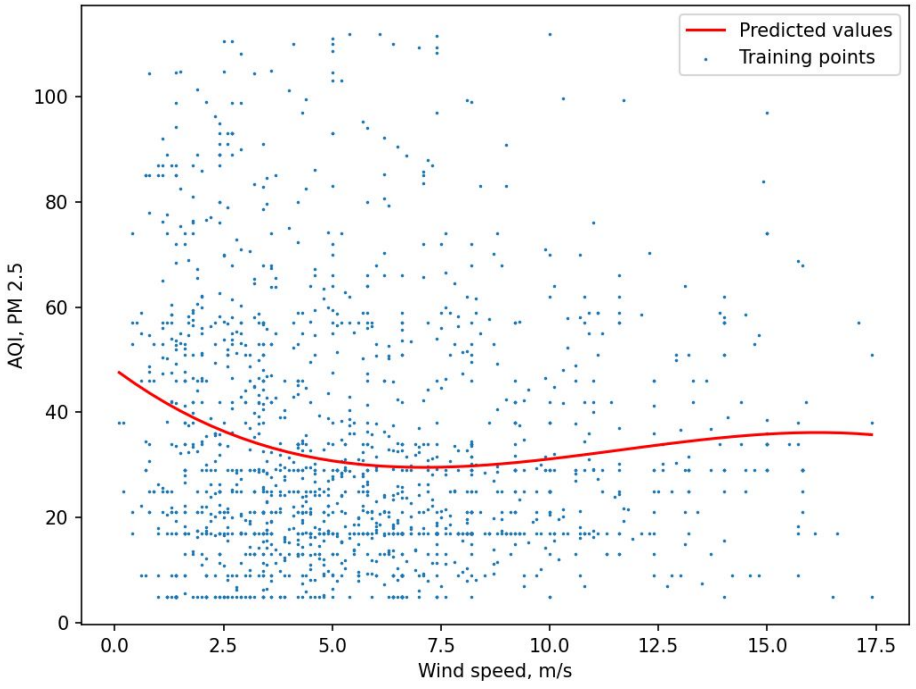

Рис. Б8 Поліноміальна регресія залежності індексу якості повітря від швидкості вітру

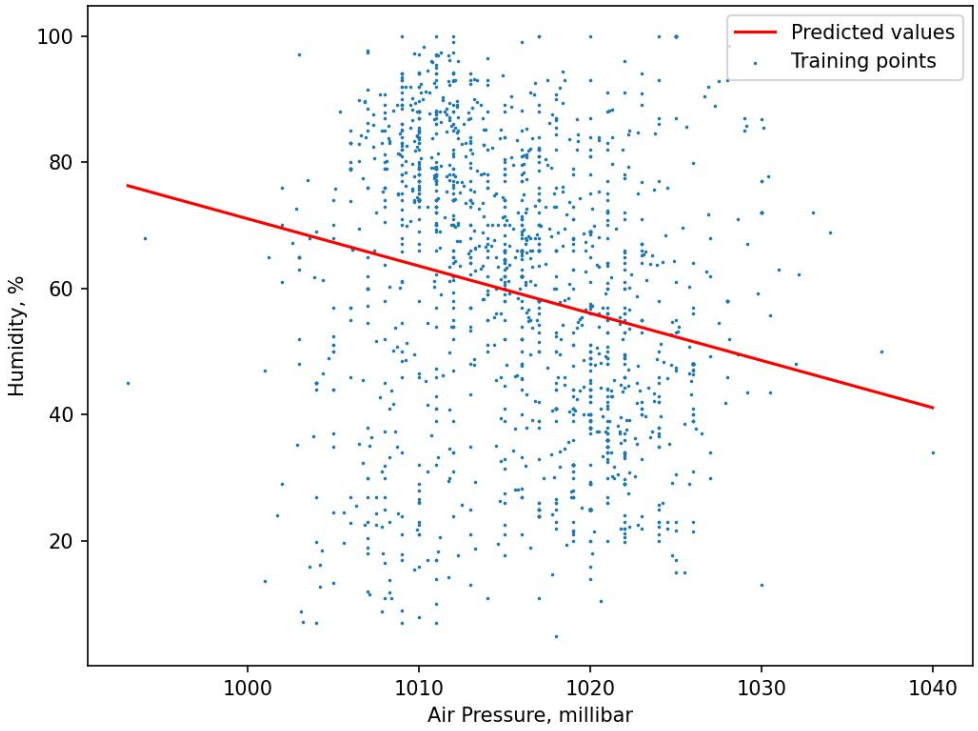

Рис. Б9 Лінійна регресія залежності вологості повітря від тиску

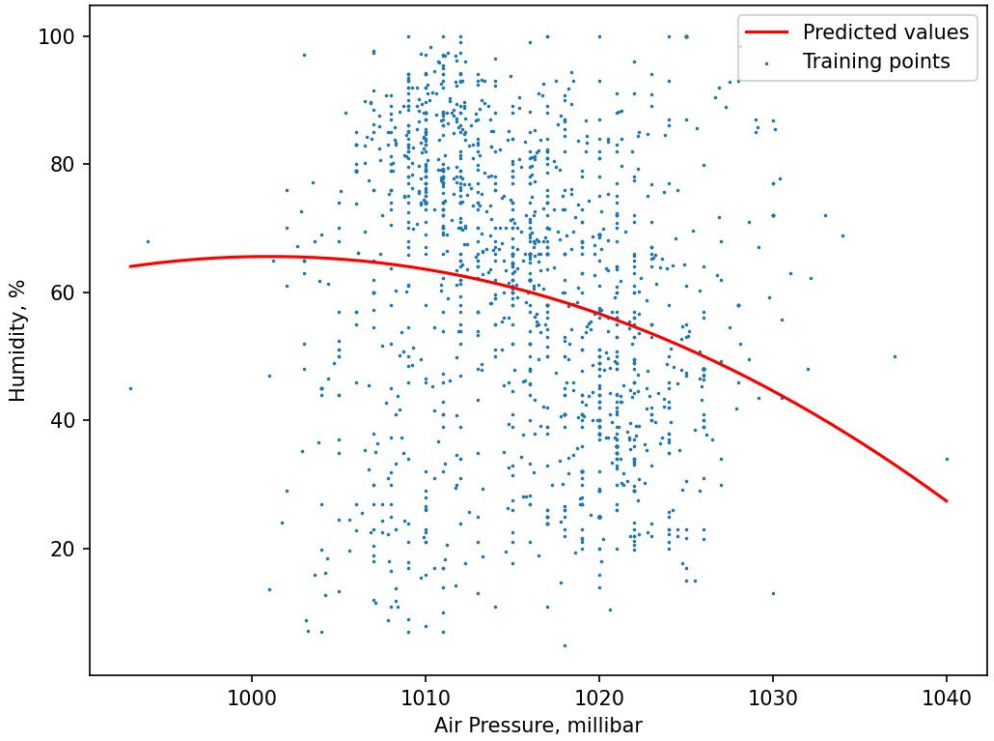

Рис. Б10 Поліноміальна регресія залежності вологості від тиску

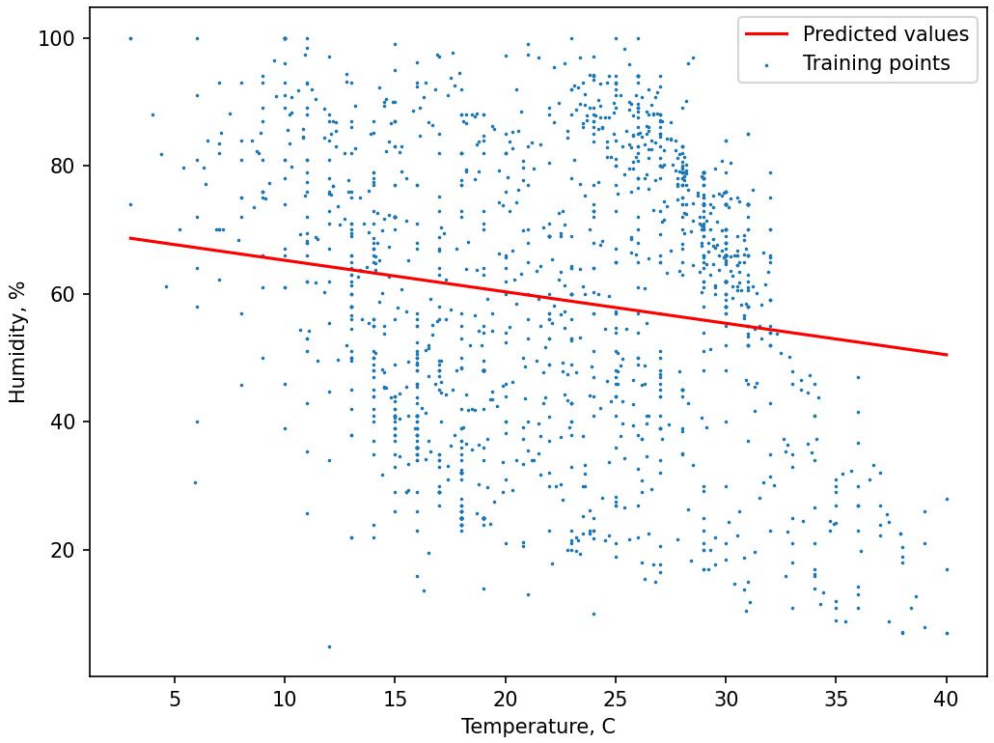

Рис. Б11 Лінійна регресія залежності температури від вологості

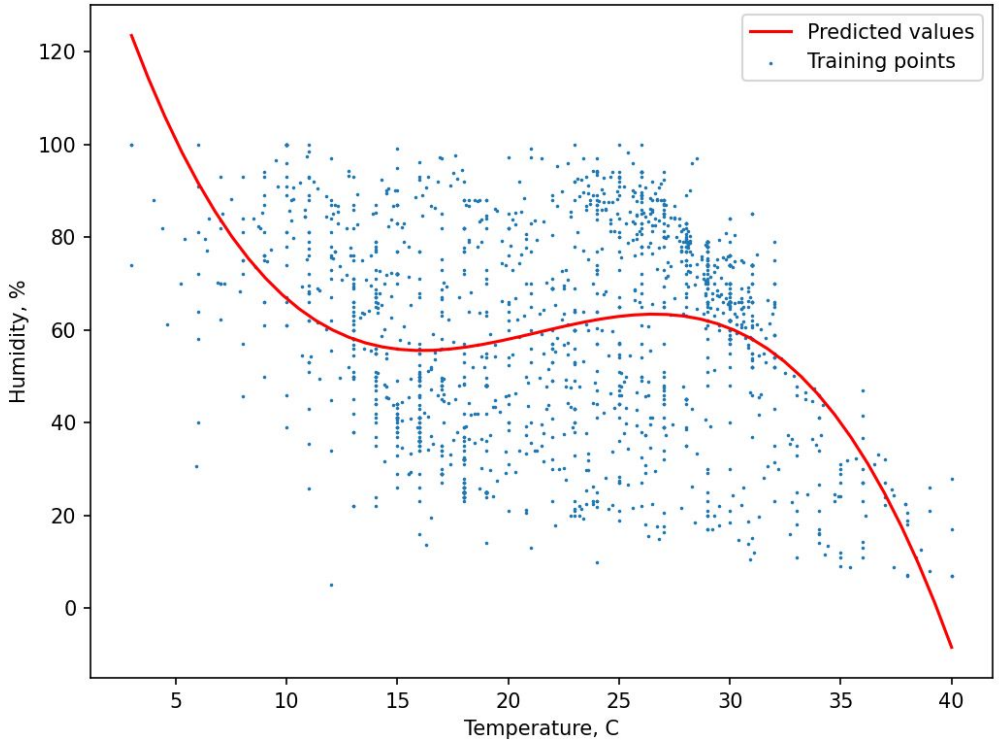

Рис. Б12 Поліноміальна регресія залежності вологості від температури