МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Комп'ютерна система аналізу та візуалізації даних

Освітній ступінь "бакалавр"
Спеціальність 123 – "комп'ютерна інженерія"

Керівник проекту:	_____ (підпис)	Рязанцев О.І. (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Критська Я.О. (ініціали, прізвище)
Здобувач вищої освіти:	_____ (підпис)	Горобинська Н. О. (ініціали, прізвище)
Група:		КІ-15з

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітній ступінь бакалавр
Напрям підготовки _____
(шифр і назва)
Спеціальність 123 – комп'ютерна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри КНІ
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Горобинській Надії Олександрівни
(прізвище, ім'я, по батькові)

1. Тема роботи Комп'ютерна система аналізу та візуалізації даних

керівник проекту (роботи) Рязанцев Олександр Іванович, д.т.н., проф.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "13" 05 2019 р. № 84/15.15
2. Термін подання студентом роботи 16.06.2019
3. Вихідні дані до роботи Розробка програмного застосунку для аналізу та візуалізації даних з використанням Python, перелік використовуваних програмних засобів: теоретичні відомості про методи класифікацій та аналізу даних, мови програмування Python, JavaScript

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Огляд основних технологій використаних для розробки додатку, методи аналізу даних, реалізація методів кластерізації, охорона праці

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст. викл. Критська Я.О.		

7. Дата видачі завдання 30.04.2019

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз завдання та робота з літературою	05.05.2019 - 13.05.2019	
2	Аналіз технічних засобів	14.05.2019 - 22.05.2019	
3	Розробка алгоритму	22.05.2019 - 02.06.2019	
4	Програмна реалізація	02.06 .2019- 11.06.2019	
5	Оформлення пояснювальної записки та	11.06.2019 - 16.06.2019	
	електронних плакатів		

Здобувач вищої освіти _____ Горобинська Н.О.
(підпис) (прізвище та ініціали)

Керівник _____ Рязанцев О.І.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка дипломної роботи: 81с., 20 рис., 1 додаток, 24 джерел.

Робота присвячена вирішенню проблеми розробки програмного додатку для аналізу та візуалізації даних з використанням Python. Метою додатку є надання зручного інтерфейсу для роботи з задачами в сфері аналізу даних. Більшість методів представлених в рамках дипломної роботи являються класичними задачами DATA MANING та можуть бути застосовані на практиці. Частина додатку що відповідає за виконання методів аналізу та класифікації даних реалізована на мові програмування Python, що демонструє оптимальність рішень такого роду задач. Додаток не обмежений функционалом та має велику перспективу в майбутньому, він зручний в розробці та його легко розвивати, реалізовуючи нові методи аналізу даних та їх візуалізації.

Ключові слова: ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАННИХ, КЛАССИФІКАЦИЯ, КРІТЕРІЙ СХОЖСТІ, ДАТАСЕТ, РОЗМІРНІСТЬ.

Умови одержання дипломного проекту: СНУ ім. В. Даля, пр. Центральний 59-А,. м. Сєвєродонецьк, 93400с.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

REST - Representational State Transfer.

SPA – Single Page Application

KNN - K Nearest Neighbor

MIME - Multipurpose Internet Mail Extensions

МНК –Метод найменших квадратов

FCM - Fuzzy C Means

AJAX - Asynchronous Javascript and XML

ВСТУП

До галузі глибинного аналізу даних зазвичай відносяться три основні сфери: класифікація, моделювання та прогнозування.

Класифікація - відношення об'єктів, спостережень або подій до одного з заздалегідь відомих класів.

Прикладом може слугувати класифікація тварин, рослин, предметів, даних и т. д. В класифікації головне наявність загального критерію, за яким буде виконуватись класифікація. Метою класифікації може бути виявленню схожості або систематизації різних видів даних. Іноді класифікація може бути корисною в прогнозуванні подій, в рамках даних предметної області. Одним із таких видів класифікації, наприклад, може бути метод найближчого сусіду. На його основі можна виявити спроби шахрайства. Нові випадки шахрайства можуть бути схожі на ті, які відбувалися колись в минулому. Передбачення відгуку клієнтів. Можна визначити відгук нових клієнтів за даними з минулого та багато чого іншого.

Ця робота спрямована на дослідження методів аналізу даних, їх реалізація з використанням мови програмування Python та надання зручного інтерфейсу для роботи.

Метою цієї роботи є отримання готового додатку, демонструючого коректну роботу методів аналізу даних, на основі якого буде можливо детальне вивчення та розуміння суті роботи того чи іншого методу та використовування таких методів на практиці.

Провідними критеріями оцінки додатку є точність виконання методів їх оптимальність, швидкодія та зручність в доступі. Огляд основних технологій використаних для розробки затсосунку.

1 ОГЛЯД ОСНОВНИХ ТЕХНОЛОГІЙ ВИКОРИСТАНИХ ДЛЯ РОЗРОБКИ ДОДАТКУ

1.1 Мова програмування Python

Все більшого поширення набувають задачі що вимагають швидкодії, роботи з великим об’ємом даних, класифікацій, прогнозування, з’явилася потреба в зручному інструменті, який би міг швидко та зручно задовольнити потреби в вирішенні такого роду задач. Протягом всього існування комп'ютерних технологій людство мало такий інструмент як математика, яка очевидно використовується для оптимізації алгоритмів в програмуванні. Математики це дуже потужний інструмент в програмуванні, але розробка також прагне потужності в роботі, елегантності та зручності, в написанні та читабельності коду. У випадку з високорівневими мовами програмування вдається досягти певної потужності, але часто для написання чогось окремого, треба було написати купу коду якого вимагала сама мова програмування. Саме тут є сенс почати говорити про Python, ця мова поєднала в собі всі фактори продуктивної розробки .

Python являється інтерпретуємою, високорівневою, алгоритмічною, об'єктно-орієнтованою мова програмування з динамічною типізацією. Мова орієнтована на підвищення продуктивності розробки та читання коду. Python дуже мінімалістична мова, в її синтаксисі не існує фігурних скобок та вона не потребує крапки з комою для позначення кінця строки, іншими словами, пишучи на Python можна одразу писати робочий код не витрачаючи час на синтаксис при цьому мати велику швидкість виконання.

Трансляція Python організована дуже схожим чином з мовою JAVA. Початковий код компілюється в байт-код, та наступним кроком байт код виконується віртуальною машиною. Різниця помітна на моменті коли байт-код записується на диск. Наприклад, у мові Java алгоритм виконання коду наступний: запускається компілятор, йому подається початковий код, компілятор генерую байт-код та записує його в файл. Потім запускається

віртуальна машина на яку подається файл з байт-кодом, яка вже потм виконує код. У випадку з Python байт код не записується на диск. Запуск програми виконується приблизно так: подається початковий код інтерпритатору, той генерує байт код, але залишає його в пам’яті, потім передає віртуальній машина яка в свою чергу являється частиною інтерпритатору.

Такий алгоритм в рази прискорює процес запуску програм за рахунок відсутності необхідності записувати байт-код на диск. Не дивлячись на те що при запускі програми Python виконую не малий стек задач, інтерпритатор намагається зберегти байт-код щоб при наступному разі загрузка модулю виконувалась скоріше.

За рахунок такої схожості в запускі програм за Java, Python має дуже велике схожість в швидкодії. Саме цьому одна із основних причин чому Python вважається алгоритмічною мовою.

Ще однією важливою особливістю Python є - автоматичне управління пам’яттю. В Python немає new та delete, пам'ять відводиться і звільняється автоматично. Алгоритм складання сміття як би "двошаровий": по-перше, сам інтерпретатор реалізує reference counting (видаляючи об'єкти, на які ніхто не посилається), і по-друге, є час від часу запускається garbage collector, що працює за більш хитрим, але більш швидким і надійним алгоритмам (наприклад, reference counting не видалить два об'єкти, що посилаються один на одного, навіть якщо на них більше ніхто не посилається).

Зручною властивістю інтерпретатора пайтон є наявність REPL (read-eval-print loop), тобто можливості вводити мовні конструкції з консолі та одразу отримувати результат. Це часто використовується для перевірки роботи коду або для налагодження.

1.2 Архітектура REST

REST – це архітектурний стиль для забезпечення стандартів між комп'ютерними системами в Інтернеті, що полегшує взаємозв'язок між системами. Системи, сумісні з REST, часто називаються RESTful системами, ці системи характерні своєю незалежністю між клієнтом та сервером. Розглянемо що означають ці терміни та чому вони являються корисними для веб додатку.

В архітектурному стилі REST реалізація клієнта та реалізація сервера можуть бути виконані незалежно, без знання існування один про одного. Це означає, що код на стороні клієнта може бути змінений в будь-який час, не впливаючи на роботу сервера, а код на стороні сервера може бути змінений, не впливаючи на роботу клієнта.

Поки кожна сторона знає, який формат повідомлень надсилати іншому, вони можуть зберігатися модульними та окремими. Відокремлюючи особливості, пов'язані з інтерфейсом користувача, з особливостями зберігання даних, ми покращуємо гнучкість інтерфейсу між платформами та покращуємо масштабованість, спрощуючи компоненти сервера. Крім того, поділ дозволяє кожному компоненту здатність розвиватись самостійно.

Використовуючи інтерфейс REST, різні клієнти відправляють ті самі запити на REST, виконують ті самі дії та отримують ті самі відповіді.

Системи, що слідують за парадигмою REST, є незалежними, що означає, що серверу не потрібно нічого знати про стан клієнта і навпаки. Таким чином, сервер і клієнт можуть зрозуміти будь-яке отримане повідомлення, незважаючи на попередні повідомлення. Це обмеження незалежності здійснюється за рахунок використання ресурсів, а не команд. Ресурси описують будь-який об'єкт, документ або речі, які вам, можливо, доведеться зберігати або надсилати до інших служб.

Оскільки системи REST взаємодіють через стандартні операції з ресурсами, вони не займаються реалізацією інтерфейсів.

Такі обмеження допомагають REST-програмам досягти надійності, продуктивності та масштабованостіі, як компонентів, якими можна керувати, оновлювати та повторно використовувати, не впливаючи на систему в цілому, навіть під час роботи системи.

В архітектурі REST клієнти надсилають запити на отримання або зміну ресурсів, а сервера відправляють відповіді на ці запити. Розглянемо стандартні подання запитів та відповіді на них.

Посилання запиту, REST вимагає, щоб клієнт подавав запит на сервер, щоб отримати або змінити дані на сервері. Запит, як правило, складається з:

– типу HTTP запиту, який визначає, який тип операції виконувати;

– заголовок, який дозволяє клієнту передавати інформацію про запит;

– шлях до ресурсу;

– необов'язкове тіло повідомлення, що містить дані.

Існує 4 основних типу HTTP запитів, які ми використовуємо в запитах для взаємодії з ресурсами в системі REST:

– GET - отримати певний ресурс (за ідентифікатором) або набір ресурсів;

– POST - створити новий ресурс;

– PUT - оновити конкретний ресурс (за ідентифікатором);

– DELETE - видалити певний ресурс за ідентифікатором.

У заголовку запиту, клієнт надсилає тип вмісту, який він може отримати від сервера. Такий вміст називається полем Accept, і це гарантує, що сервер не надсилає дані, які неможливо зрозуміти або обробляти клієнтом. Варіанти типів – це типи MIME (Multipurpose Internet Mail Extensions)

Типи MIME, використовуються для вказання типів вмісту у полі Accept, складаються з типу та підтипу. Вони розділені косою рискою “ / ”. Текстовий файл, що містить HTML, буде вказано за допомогою типу text/html. Якщо цей текстовий файл містив CSS, він буде вказаний як text/css. Загальний текстовий файл буде позначено як text/plain. Однак це значення за

замовчуванням, text/plain, не є загальним. Якщо клієнт чекає text/css і отримує text/plain, він не зможе розпізнати вміст.

Наприклад, запит для доступу до ресурсу з ідентифікатором 23 у ресурсі статей на сервері, може надіслати запит GET таким чином рисунок 1.1.

```
GET /articles/23
Accept: text/html, application/xhtml
```

Рисунок 1.1 - Приклад запита до ресурсу с ідентифікатором 23 та полем прийняття

Поле заголовка Accept в даному випадку говорить про те, що клієнт буде приймати вміст у text/html або application/xhtml.

Запити повинні містити шлях до ресурсу, на який слід виконати операцію. У RESTful API, шляхи повинні бути розроблені, щоб допомогти клієнту знати, що відбувається. Традиційно перша частина шляху повинна бути множинною форм ресурсу. Це зберігає вкладені шляхи, прості у читанні та легко зрозумілі. Шлях, який називається /customers/223/orders/12 його ясність в тому, що він вказує, навіть якщо ви ніколи раніше не бачили цього конкретного шляху, оскільки він є ієрархічним та описовим. Ми бачимо, що ми отримуємо доступ до замовлення з ідентифікатором 12 для клієнта з ідентифікатором 223.

Шляхи повинні містити інформацію, необхідну для пошуку ресурсу з необхідною мірою специфіки. Коли звертається до списку або збірки ресурсів, то немає необхідності додавати ідентифікатор до запиту POST на шлях /customers, тому він не потребує додаткового ідентифікатора, оскільки сервер генерує ідентифікатор нового об'єкта.

Якщо ми намагаємось отримати доступ до одного ресурсу, нам потрібно буде додати ідентифікатор до шляху. Наприклад: GET /customers/:id - вилучає елемент у ресурсі клієнтів із вказаним ідентифікатором. DELETE

/customers/:id - видаляє елемент у ресурсі клієнтів із вказаним ідентифікатором.

У випадках, коли сервер надсилає клієнту корисну інформацію, сервер повинен включити типовий вміст (content-type) у заголовок відповіді. Це поле заголовка вмісту сповіщає клієнта про тип даних, які він надсилає в тілі відповіді. Ці типи вмісту - це типи MIME, так само, як вони знаходяться у полі прийому заголовка запиту. Тип контенту, який сервер відправляє назад у відповідь, повинен бути одним із варіантів, який клієнт вказав у полі прийому запиту.

Наприклад, коли клієнт отримує доступ до ресурсу з ідентифікатором 23 у ресурсі статей (рисунку 1.1). Сервер може відправити вміст із заголовком відповідей рисунок 1.2.

```
HTTP/1.1 200 (OK)
Content-Type: text/html
```

Рисунок 1.2 - Приклад вмісту відповіді з серверу

Це означає, що вміст, який запитується, повертається в тілі відповіді, використовуючи тип вмісту text/html, який клієнт всказав, що він зможе прийняти.

1.3 Django Rest Framework

В світі Python існує достатньо широкий набір фреймворків та бібліотек, що реалізують роботу у веб просторі. Одним з таких є Django REST Framework. Взагалі Django REST це похідна від іншого однойменного фреймворку Django, з тією лише відмінністю що другий більш розроблений для розробки під REST архітектуру, яка більш детально розглянута раніше. Іншими словами, розробляючи на Django ми пишемо додаток який реалізує серверну частину додатку та інтерфейс,

пишучі на DJANGO REST ми пишемо API. У випадку з Django дуже зручно писати весь застосунок, не виходячи за рамки одного проекту та ми піклуємося про весь інтерфейс одразу на сервері, але в цьому випадку стає дуже важкою реалізація динаміки на клієнтській частині. З Django REST вся відповідальність за динаміку та важкість клієнтської частини кладеться безпосередньо на клієнта.

Django REST це яскравий приклад реалізації парадігми REST, що містить зручний функціонал серіалізації, що підтримує ORM і не-ORM джерела даних. Можливість повної і детальної настройки - можна використовувати звичайні уявлення-функції, якщо немає потреби в потужному функціоналі. Незважаючи що Django REST це фреймворк виключно для розробки серверної частини додатку, він має дуже зручний веб інтерфейс для посилання та тестування запитів[8].

1.4 Мова програмування JavaScript

Саме тому що програмний додаток дипломної роботи реалізований на основі REST технологій, важливо розповісти про клієнтську частину проекту. В сучасному світі, коли йдеться мова про web розробку неможливо не говорити про JavaScript. Саме на мові JavaScript побудовано 90 відсотків клієнтської частини.

На початку свого існування мова використовувалась для надання інтерактивності веб-сторінці: реакція по натисканню кнопки, анімацій, змін стилю. Згодом потенціал JavaScript-а зростав, з'явився широкий набір бібліотек та фреймворків для роботи з JavaScript. Зараз неможливо уявити веб-додаток без використання JavaScript.

Технологія вивела розробку на новий рівень, завдяки можливості сучасних комп'ютерів та мобільних телефонів, стало зручно перенести більшу частину логіки додатку на клієнтську частину, при цьому підвищити швидкодію та добавити велику кількість динамічності.

Сучасний JavaScript - це «безпечна» мова програмування загального призначення. Вона не надає низькорівневих засобів роботи з пам'ятю, процесором, так як з самого спочатку був орієнтований на браузери, в яких це не потрібно.

Що стосується інших можливостей - вони залежать від оточення, в якому запущений JavaScript. У браузері JavaScript вміє робити все, що відноситься до маніпуляцій з сторінкою, взаємодії з клієнтом та, в якійсь мірі, з сервером:

– створювати нові HTML-теги, виділяти існуючі, змінювати стилі елементів, ховати, показувати елементи;

– реагувати на дії відвідувача, обробляти кліки миші, переміщення курсору, натискання на клавіатуру і та багато іншого;

– посилати запити на сервер і завантажувати дані без перезавантаження сторінки;

– отримувати і встановлювати cookie, запитувати данні, виводити повідомлення;

– та багато іншого.

JavaScript підтримують не тільки браузери. Бази даних типу MongoDB та CouchDB використовують його в якості скриптової мови та мови запитів. Є кілька платформ для десктопів та серверів, найбільш відома з яких Node.js, що надають потужне оточення для програмування за межами браузера.

JavaScript – швидка та потужна мова, але браузер накладає на його виконання деякі обмеження. Це зроблено для безпеки користувачів, щоб зловмисник не міг за допомоги JavaScript отримати особисті дані або якось нашкодити комп'ютеру користувача.

Цих обмежень немає там, де JavaScript використовується поза браузером, наприклад на сервері. Крім того, сучасні браузери надають свої механізми по установці плагінів і розширень, які володіють розширеними можливостями, але вимагають спеціальних дій по установці від користувача

Більшість можливостей JavaScript в браузері обмежена поточним вікном та сторінкою.

JavaScript не може читати/записувати довільні файли на жорсткий диск, копіювати їх або викликати програми. Він не має прямого доступу до операційної системи.

Сучасні браузери можуть працювати з файлами, але ця можливість обмежена спеціально виділеної Директорією - «пісочницею». Можливості щодо доступу до пристроїв також опрацьовуються в сучасних стандартах і частково доступні в деяких браузерах[9].

JavaScript, що працює в одній вкладці, не може спілкуватися з іншими вкладками і вікнами, за винятком випадку, коли він сам відкрив це вікно або декілька вкладок з одного джерела (однаковий домен, порт, протокол).

Є способи це обійти, і вони розкриті в підручнику, але вони вимагають спеціального коду на обидва документи, які знаходяться в різних вкладках або вікнах. Без нього, з міркувань безпеки, залізти з однієї вкладки в іншу за допомогою JavaScript можна.

Існує як мінімум три чудових особливості JavaScript:

– повна інтеграція з HTML / CSS;

– прості речи робляться просто;

– підтримується всіма поширеними браузерами та включений автоматично.

Цих трьох речей одночасно немає жодна браузерна технологія.

Тому JavaScript і є найпоширенішим засобом створення браузерних інтерфейсів.

1.5 Платформа Node.js

Node.js представляє собою середу виконання коду на JavaScript, яка побудована на основі двигуна JavaScript Chrome V8, який дозволяє транслювати код написаний на мові JavaScript в машинний код. Node.js перш

за все призначений для створення серверних додатків на мові JavaScript. Хоча також існують проекти по написанню десктопних додатків і навіть створення коду для мікроконтролерів. Але перш за все Node.js, це платформа для створення веб застосунків.

Ядро Node.js розроблено згідно з деякими принципами, один з який полягає в наданні мінімального набору функціональних можливостей, залишаючи реалізацію всіх додаткових можливостей за модулями екосистеми, що не входять в ядро. Цей принцип має великий вплив на платформу Node.js, оскільки дає розробникам повну свободу експериментувати в області застосування призначених для користувача модулів, не нав'язуючи якесь одне, повільно розвивається рішення, вбудоване в жорстко контрольоване стабільне ядро. Зведення основного набору функцій до мінімуму зручно не тільки з точки зору обслуговування, але і корисно з точки зору позитивного впливу на розвиток всієї екосистеми.

Платформа Node.js використовує ідею модуля як основний засіб структурування програмного коду. Модулі є будівельними блоками додатків і бібліотек, часто званих пакетами. Один з найбільш активно просуваються принципів платформ ми Node.js полягає в розробці невеликих модулів, не тільки з точки зору розміру, але і, що більш важливо, з точки зору охоплених можливостей. Цей принцип виходить з філософії Unix, зокрема з двох наступних її заповідей:

– чим менше, тим краще;

– будь-яка програма повинна робити щось одне, але робити це добре.

Платформа Node.js підносить ці ідеї на новий рівень. Офіційний диспетчер пакетів платформи Node.js допомагає вирішити проблему залежності, гарантуючи кожному встановленому пакету наявність власного окремого набору залежності, що дозволяє програмі використовувати безліч пакетів без появи конфліктів. Підхід Node забезпечує оптимальний рівень повторного використання коду, оскільки при його застосуванні додатки складаються з великого числа малих, чітко спрямованих залежності. Хоча це

вважається непрактичним або навіть абсолютно не здійсненним в інших платформах, для розробки на платформі Node.js рекомендується саме така методика. Як наслідок npm-пакети не часто містять менше 100 рядків коду і призначені для реалізації тільки однієї єдиної функції. Крім того, природні переваги багаторазово використовуваних невеликих модулів полягають в наступному:

- простота розуміння і використання;
- легкість тестування і підтримки;
- оптимальність при обміні з браузером.

Наявність невеликих, вузькоспеціалізованих модулів забезпечує можливість спільного багаторазового використання фрагментів коду. Це дозволяє підняти принцип «не повторювати» на абсолютно новий рівень[10].

Крім того що модулі Node.js повинні мати невеликий розмір і конкретну спрямованість, вони зазвичай експортують мінімальний набір функціональних можливостей. Основною перевагою є підвищення зручності і чіткості програмного інтерфейсу, що зменшує ймовірність неправильного використання. Як правило, користувачам компонента потрібно дуже обмежений, конкретний набір функцій, а не розширення його функціональності або занурення в його особливості.

У платформі Node.js широко використовується шаблон визначення модулів, які експортують тільки одну функціональну можливість, наприклад функцію або конструктор, при цьому доступ до більш складних аспектів або додатковим віз можливостям здійснюється через властивості, що експортуються цією функцією або конструктором. Це допомагає користувачеві розділити важливі і вторинні можливості. Часто модулі експортують тільки одну функцію і нічого крім неї, забезпечуючи явну єдину точку входу.

Ще однією характерною рисою багатьох модулів платформи Node.js є їх призначення для використання, а не для розширення. Приховування внутрішнього зі тримання модулів, що забороняє будь-яке його розширення,

може здатися позбавленим гнучкості підходом, але він забезпечує скорочення варіантів використання, спрощує реалізацію, полегшує обслуговування і підвищує зручність використання.

1.6 Односторінковий додаток (SPA)

Односторінковий застосунком називають застосунок, що працює в браузері, та не вимагає перезавантаження сторінки під час використання. Такий тип програм останнім часом набув все більшої популярності. Наприклад Gmail, Google Maps, Facebook або GitHub. SPA - це все, що стосується обслуговування видатного UX, намагаючись імітувати "природне" середовище в браузері - перезавантаження сторінок не вимагає додаткового часу очікування. Це лише одна веб-сторінка, яку ви відвідуєте, а потім завантажує весь інший вміст за допомогою JavaScript, від якого вони сильно залежать.

Односторінковий додаток запрошує розмітку та дані незалежно, та відображую сторінки прямо в браузері. Такі застосунки реалізуються завдяки вдосконаленим системам JavaScript, таких як AngularJS, Ember.js, Meteor.js, Knockout.js. Сайти на одній сторінці допомагають зберігати користувача в одному, зручному веб-просторі, де вміст представлений користувачеві просто, зручно та практично.

Плюси для застосування односторінкового додатку:

– SPA є швидким, оскільки більшість ресурсів (HTML + CSS + Scripts) завантажуються лише один раз протягом усього терміну дії програми. Тільки дані передаються назад і вперед;

– розробка спрощена та упорядкований. Не потрібно писати код для відтворення сторінок на сервері. Набагато простіше почати роботу, оскільки зазвичай ви можете запустити розробку з файлового file://URI, не використовуючи жодного сервера взагалі;

– SPA можна легко налагодити за допомогою браузеру, оскільки ви можете відстежувати операції в мережі, досліджувати елементи сторінки та пов'язані з ним дані;

– простіше зробити мобільну програму, оскільки розробник може повторно використовувати той же код сервера для веб-додатки та власної мобільної програми;

– може ефективно кєшувати локальне сховище. Програма надсилає лише один запит, зберігає всі дані, потім може використовувати ці дані і працює навіть у режимі офлайн.

Мінуси односторінкового додатку:

– складне і нелегке завдання зробити оптимізацію SEO одним додаванням сторінки. Його вміст завантажується за допомогою AJAX (Asynchronous JavaScript and XML) - методів обміну даними та оновлення в додатку без оновлення сторінки;

– завантажується повільно, оскільки важливі клієнтські рамки потрібно завантажувати до клієнта;

– для цього потрібен JavaScript, щоб він був присутній та включений. Якщо будь-який користувач відключить JavaScript у своєму браузері, він не зможе правильно завантажити програму та її дії;

– у порівнянні з "традиційним" застосуванням, SPA є менш безпечним. завдяки Cross-Site Scripting (XSS) це дозволяє шахраям вставляти сторонні скрипти в веб додаток іншим користувачам;

– витік пам'яті JavaScript може навіть призвести до сповільнення потужної системи.

1.7 Технологія Webpack

В індустрії web-розробки спостерігається постійне зростання односторінкових додатків (SPA), багато автономних додатків, які можна

встановити на комп'ютер, що пропонують повноцінний функціонал для користувача. На жаль, як відомо, чим більше функцій додається, тим більше пишеться коду, враховуючи середню складність SPA, в кінці розробка досягає точки, де код може складатися з сотен, якщо не тисяч javascript, css та багатьох інших файлів.

Тому виникає проблема: як впоратися з кодовою базою такого розміру. Як управляти порядком завантаження файлів/модулів, які роблять додаток працездатним, коли їх така велика кількість. У минулому це було значно легше. Використовувалися теги script в певному порядку в тілі html. Так, наприклад, якщо в файл B повинен бути завантажений після файла A і файл C повинен бути завантажений після A і B рисунок 1.3.

```
<html>
    <body>
        …
        …
        <script type='text/javascript' src='path/to/A.js'></script>
        <script type='text/javascript' src='path/to/B.js'></script>
        <script type='text/javascript' src='path/to/C.js'></script>
    </body>
</html>
```

Рисунок 1.3 - Приклад порядку завантаження файлів

З часом додатки стали складніші, кількість цих файлів буде рости та контроль буде складнішим. Наприклад якщо файл B потрібно загрузити після іншого файлу D, який, в свою чергу, повинен бути завантажений після іншого файлу E, і файл A також повинен бути після завантаження файлу D. Таким чином маючи всього 5 файлів з’являється складність в правильному порядку завантаження.

Якщо уявити відносні залежності між цими файлами. Ця конструкція нагадує граф залежності, дерево, яке описує порядок завантаження файлів.

Ведення цього графу самостійно – важке завдання. При створенні нового файлу, потрібно знати, де повинно бути завантаження в ланцюгу

завантажень скриптів, а також від яких зовнішніх файлів він залежить, і в решті, де повинен знаходитись тег `<script ...></script>` в `.html`.

Крім того важливо створити єдиний .js файл з усіх цих окремо взятих A.js, B.js, C.js і так далі, щоб уникнути витрат на кілька HTTP запитів завантаження для кожного .js файлу. Цей процес називається збірка, він об'єднує різні файли в один, зберігаючи порядок їх залежностей. Роблячи це вручну, потрібно багато копіювати і вставляти, кожен раз коли файл було створюєте або змінено, одночасно ризикуючи допустити помилку.

На допомогу приходить Webpack, інструмент, основна мета якого полягає в тому, щоб зібрати всі .js файли в будь-яку потрібну кількість пакетів (bundles), а також переконатися, що в зібраному пакеті є всі .js файли проекту в правильному порядку.

Для того, щоб використовувати Webpack є одна велика вимога: ваш JavaScript код повинен бути організований у модульному форматі. Це означає, що кожен файл повинен явно вимагати всі інші .js файли, від яких він залежить, і потенційно зробити деякі елементи доступними для інших .js файлів.

У сучасному світі існує 3 різних способи зробити це:

– CommonJS;

– AMD modules;

– ES6 modules.

На даний момент у відповідності зі специфікацією ES6 шаблони CommonJS і AMD вважаються в основному застарілими, тому рекомендується стандартна підтримка модулів ES6.

Як говорилося раніше, кожен файл повинен явно описувати те, що йому потрібно в якості вхідних даних, і те, що він зробить доступним для інших файлів. Все, що не зроблено доступним для інших файлів, залишається в межах файлу і ніколи не буде розкрито.

Отже, стає питання загрузки файлів. Перш за все, потрібно відмовитись від всіх тегів *<script ...> </ script>* в .html файлах. Після цього потрібно буде повідомити всі .js файли про пріоритет завантаження.

У світі модулів вимоги файлів перетворюються в вимоги змінних, функцій, компонентів, об'єктів і так далі.

Приклад синтаксису ES6 рисунок 1.4.

```
// add.js
export default function add (a,b) {
    return a + b
};

// numbers.js
export const number1 = 3;
export const number2 = 5;

// whatever.js
import add from './path/to/add.js';
import { number1, number2 } from './path/to/numbers.js';

const X = add(number1, number2);

export const X;
```

Рисунок 1.4 - Пиклад загрузки модулів в ES6

Вищевказані інструкції визначають add.js і variables.js як залежність для whatever.js. Маючи це на увазі, файли .js мають залежності, які повністю описують, що потрібно завантажувати, обчислювати або запускати, перш ніж вони самі можуть бути завантажені або запущені. Webpack приймає ці інструкції і намагається створити спрямований ациклічний граф це означає, що він намагається підтягти всі залежності без будь-яких циклів. Наявність циклу означало б, що файл А залежить від файлу В, а файл В залежить від файлу А, що робить зв’язування неможливим. Якщо ж циклів не виявлено, Webpack може взяти вміст всіх цих файлів і вставити їх в один .js файл в правильному порядку[10].

Але як Webpack дізнається, які файли додавати в фінальний файл - bundle, коли немає тегів <script> . Проблема полягає в тому, що просто так

це дізнатися неможливо. Ось чому потрібно оголосити файл, який буде працювати як точка входу.

Такою точкою входу буде файл .js, що містить основний/базовий компонент додатку. У більш старому сценарії: файл, який повинен бути завантажений після завантаження всіх інших js файлів.

Останній крок, треба добавити тільки один тег <script>, котрий буде вказувати на збірний .js файл, зібраний Webpack-ом.

1.8 Бібліотека React

Перше про що слід розповісти про React, React це бібліотека, не фреймворк. Саме поняття бібліотеки дуже схоже на фремворк, вона так само надає набір готового функціоналу для вирішення різних завдань, але на відміну від фреймверка, бібліотека не обмежує в свободі дій та синтаксису.

Бібліотека React пропонує абстракцію уявлень, засновану на ідеї компонентів, де компонентом може бути кнопка, поле введення, простий контейнер, такий як HTML-тег div або будь-який інший елемент призначеного для користувача інтерфейсу. Ідея бібліотеки полягає в наданні розробникам можливості створювати призначені для користувача інтерфейси за допомогою простого визначення і з'єднання багаторазово використовуваних компонентів, що мають конкретні обов'язки. Від інших реалізацій бібліотека React відрізняється відсутністю прив'язки до моделі DOM. Справді, вона забезпечує високорівневу абстракцію, що носить називання віртуальна модель DOM, яка добре сумісна з Веб, але також може використовуватися в інших контекстах, наприклад в мобільних додатках, для моделювання тривимірних оточень і навіть для визначення взаємодій між апаратними компонентами.

Основна причина інтересу до бібліотеки React в контексті розробки на універсальному JavaScript - в тому, що вона дозволяє реалізувати

відображення уявлень на стороні сервера або клієнта за допомогою практично одного й того ж коду.

Іншими словами, за допомогою React можна створити HTML-код сторінки на сервері, а потім, після завантаження сторінки, додати додаткові інтерактивні елементи безпосередньо в браузері. Це дозволяє створювати односторінкові додатки, де велика частина дій виконується в браузері і оновлюється тільки та частина сторінки, яку потрібно змінити. При такому підході початкове уявлення завантажується з сервера, що створює ефект (здається) швидкої за загрузки і значно спрощує індексацію вмісту пошуковими системами. Варто також відзначити, що віртуальна модель DOM бібліотеки React спосіб на оптимізувати відображення змін. Тобто модель DOM не відображається повністю після кожної зміни, замість цього бібліотека React використовує хитрий алгоритм, який розраховує мінімально необхідний обсяг змін в DOM для оновлення зображення на екрані. Це дуже ефективний механізм швидкого відображення в браузері, він є ще однією причиною, чому бібліотеці React все частіше віддається перевага при наявності безлічі інших бібліотек і фреймворків

Як уже згадувалося раніше, бібліотека React - це високорівнева програмний інтерфейс для створення віртуальної моделі DOM і управління нею. Сама по собі ідея моделі DOM чудова, і цю модель можна без особливих проблем уявити в форматі XML або HTML, але динамічне маніпулювання її деревом, наприклад на рівні вузлів, батьків і нащадків, дуже швидко стає занадто громадилися ким. Для подолання даної проблеми бібліотека React надає мову JSX як проміжний формат для опису віртуальної моделі DOM і управління нею.

Насправді JSX не є самостійною мовою - це лише надмножина мови JavaScript, яке потрібно транслювати в звичайну мову JavaScript перед виконанням. Однак ця надбудова дозволяє розробникам використовувати XML-синтаксис в коді на JavaScript. При розробці клієнтського коду мову JSX використовується для опису розмітки HTML, яка визначає веб-

компоненти, як було показано в попередньому прикладі. Причому HTML-теги можна розміщувати безпосередньо в JSX-коді, як якщо б вони були частиною розширеного синтаксису мови JavaScript. Цей підхід забезпечує істотну перевагу, оскільки тепер HTML код динамічно перевіряється під час складання з видачею повідомлень про помилки, наприклад якщо забути закрити один з тегів.

1.9 Технологія Redux

В ході розвитку односторінкових веб застосунків, з’явився сенс в зберіганні даних на клієнтській частині програми . У минулому такого завдання не представлялось тому як сервера віддавали готові html файли. Дані які повинні відображатися на сторінці були визначені статично в сторінці. І при будь-якій взаємодії клієнта на сторінці відбувалось її перезавантаження.

З приходом JavaScrpt в веб розробку такий підхід став вважатись не оптимальним.

Чому такий підхід не є оптимальним ? У будь-якому випадку ми робимо запит на сервер, отримуємо відповідь і відображаємо результат.

Однак у випадку з старим підходом до розробки веб додатків ми робили запити занадто часто, і якщо потрібно змінити лише одну одиницю даних на сторінці змінювалось вся сторінка, хоча нам потрібно змінити лише одне значення. Через це страждала швидкість роботи веб додатку.

У сучасній реалізації все змінилось, і я вважаю змінилось на краще. Стало як мінімум логічніше, ми просимо у сервера дані, і ми отримуємо виключно дані без купи рядків html коду.

З іншого боку не оптимальність, це невелика плата за зручність, ми отримували у відповідь html який можемо відразу показувати клієнту. У випадку з односторінковими додатками, нам взагалі не доводиться перезавантажувати html, ми завантажуємо його всього один раз.

Сучасні додатки використовують JavaScript фреймворки різні бібліотеки, в яких найчастіше реалізований функціонал додавання даних безпосередньо з коду JavaScript.

Отже, в результати технологія нам дає, оптимальність запитів, завантаження html коду всього один раз, залишилось тільки додавати дані на сторінку, також дані потрібно десь зберігати, оновлювати, видаляти і отримувати. Тут до нас на допомогу приходить Redux.

Redux - інструмент управління станом даних на клієнтській частині веб додатку. У загальному випадку Redux це деревоподібний об’єкт названий сховищем, в зберігаються дані згідно певної логіки. Один з найважливіших моментів в розумінні Redux-у, це те що сховище існує тільки для читання. Саме сховище має невелике API, що складається всього з чотирьох методів:

– store.dispatch(action);
– store.subscribe(listener);
– store.getState();
– replaceReducer(nextReducer).

На першому етапі використання Redux в своєму додатку ми визначаємо логіку зберігання. Уявимо що у нас є маленький додаток який зберігає дані про користувача, логіка максимально проста, це об'єкт з одним полем (рис. 5)

```
let store  = {
    user: {}
};
```

Рисунок 1.5 - Приклад сховища данних з полем user

Значення за змовчанням задано як порожній об’єкт. Для того щоб клієнт який зайде на наш додаток мав можливість переходити між сторінками і при цьому не втрачати персональні дані, потрібно скористатись одним з методів Api Redux store.dispatch (action). Action представляє собою об’єкт з полями type і payload. В поле type зберігається тип дії, найчастіше це

збережений в окремий файл константа, по суті це просто рядок наприклад GET_USER_INFORMATION, в полі payload передаються дані рисунок 1.6.

```
store.dispatch({
    type: 'GET_USER_INFORMATION',
    paylod: {
        name: 'Vlad',
        age: 21
    }
})
```

Рисунок 1.6 - Приклад передачі даних до сховища Redux

Другим важливим моментом в розумінні Redux це те що Redux не перезаписує дані, а повертає новий стан об’єкту. Як вже сказано раніше, неможливо безпосередньо змінювати сховище, ми можемо тільки повенути новий стан. Наприклад якщо користувач вирішить змінити свій рік, то він не змінить значення user.age він поверне новий стан, який буде переданий в сховище як {name: 'Vlad', age: 22}.

Таким чином в нашому сховищу вже зберігаються дані про користувача, данні готові до використання в інтерфейсі, залишається тільки передати сховище потрібного компоненту та можна їх відображати.

1.10 Технологія Ajax

AJAX, більш детальніше, асинхронний Javascript та Xml – це технологія для взаємодії з сервером без перезавантаження сторінок. За рахунок AJAX зменшується час відгуку веб-додатків та збільшується інтерактивність.

Технологія AJAX - асинхронна, тобто браузер, відіславши запит, може робити що завгодно, наприклад, показувати повідомлення про очікування відповіді, прокручувати сторінки, і т.д.

Технологія AJAX використовує комбінацію з:

– (X) HTML, CSS для подачі та стилізації інформації;

– DOM-модель, операції над якою використовується javascript на стороні клієнта, щоб забезпечити динамічне відображення та взаємодію з інформацією;

– XMLHttpRequest для асинхронного обміну даними з веб-сервером. Інколи замість XMLHttpRequest використовується IFrame, SCRIPT-тег або інший аналогічний транспорт;

– JSON часто використовується для обміну даними, однак будь-який формат підходить, включаючи HTML, текст, XML;

– типовий AJAX-додаток складається як мінімум з двох частин;

– перша виконується в браузері і написана, як правило, на JavaScript, а друга – знаходиться на сервері і написана, наприклад, на Ruby, Java, Python або PHP. Між цими двома частинами відбувається обмін даними через XMLHttpRequest.

AJAX - в інтерактивності та швидкості часу відгуку.

– динамічне завантаження даних з сервера;

Наприклад, дерево, вузли якого завантажуються в міру розкриття.

– непомітні для користувача дії;

Наприклад, при редагуванні статті - кожні 10 хвилин результати записуються на сервері.

– невпинна загрузка інформації з сервера.

Гарний приклад - чат. У вікно постійно надходять все нові повідомлення, невпинно поступаючи з сервера. Так само через AJAX, без перезавантаження сторінки, користувач може відіслати повідомлення на сервер. Ще один вдалий приклад – біржові застосунки, які постійно оновлюються.

У синхронній моделі браузер відправляє запит на сервер та чекає, поки той зробить всю необхідну роботу. Сервер виконує запит до баз даних,

згортає відповіді в необхідний формат та віддає його. Браузер отримавши відповідь, викликає функцію показу.

Всі процеси виконуються послідовно, один за іншим.

Клієнт не може займатись чимось іншим, поки відбувається синхронний обмін даними.

У асинхронній моделі запит відсилається, і можна займатись іншими справами. Коли запит виконався - запускається заздалегідь підготовлена програмістом функція показу повідомлення сервера.

Тут сервер відразу ж повідомляє браузер про те, що запит прийнятий в обробку і звільняє його для подальшої роботи. Коли відповідь буде готовий - сервер перешле його та в браузері буде викликана відповідна функція показу, але поки ця відповідь формується та пересилається - браузер вільний.

Користувач може писати коментарі, заповнити та відіслати форму і т.п.В той же момент можуть відбуватися нові асинхронні запити.

Асинхронна модель характеризується майже миттєвою реакцією на дії користувача, так що створюється враження зручного та швидкого додатку.

1.11 Постановка задачі

Актуальним завданням є вирішення задач стосовно аналізу даних та їх візуалізації. Тому ставиться завдання розробки додатку для реалізації методів аналізу даних з використанням мови програмування Python для безпосереднього виконання математичних методів та роботи з даними та мови програмування JavaScript для реалізації роботи методів на стороні клієнту. Саме тому що додатку доведеться мати справу з даними та частіше з їх великим об'ємом, дуже важливим моментом є оптимальність роботи додатку, в такому випадку є сенс в розробці REST архітектури, односторінкового додатку та іншими, раніше розглянутими технологіями.

Отже для реалізації додатку необхідно вирішити такі завдання:

– розробити методи класифікацій даних;

- налаштувати серверну частину в архітектурному стилі REST;
- реалізувати клієнтський односторінковий застосунок.

2 МЕТОДИ АНАЛІЗУ ДАНИХ

2.1 Огляд методів кластерізації даних

Під задачею кластерізації зазвичай вважають розбиття множини об'єктів на групи, які називаються кластерами. Усередині кожної групи повинні виявитися «схожі» об'єкти, а об'єкти різних групи повинні бути якомога більш відмінні. Головна відмінність кластеризації від класифікації полягає в тому, що перелік груп чітко не заданий і визначається в процесі роботи методу.

Застосування кластерного аналізу в загальному вигляді зводиться до наступних етапів:

– відбір вибірки об'єктів для кластеризації;

– визначення множини змінних, за якими будуть оцінюватися об'єкти у вибірці. При необхідності - нормалізація значень змінних;

– обчислення значень міри схожості між об'єктами;

– застосування методу кластерного аналізу для створення груп схожих об'єктів (кластерів);

– представлення результатів аналізу.

Після отримання та аналізу результатів можливе корегування обраної метрики і методу кластеризації до отримання оптимального результату.

Міри відстаней

Як визначається «схожість» об'єктів? Для початку складається вектор характеристик для кожного об'єкта - як правило, це набір числових значень, наприклад, зріст або вага людини. Однак існують також алгоритми, що працюють з категорійними характеристиками.

Після того, як був визначений вектор характеристик, можна провести нормалізацію, щоб всі компоненти давали однаковий внесок при розрахунку «відстані». У процесі нормалізації все значення приводяться до деякого діапазону, наприклад, [-1, -1] або [0, 1].

Наступним кроком, для кожної пари об’єктів вимірюється відстань між ними – ступінь схожості. Існує велика варіація метрик. Приклади деяких з них:

Евклідова відстань.

Найбільш поширена функція відстані. Являє собою геометричним відстанню в багатовимірному просторі:

$$p(x, x') = \sqrt{\sum_i^n (x_i - x_i)^2}. \quad (2.1)$$

Квадрат евклідової відстані.

Застосовується для додання більшої ваги більш віддаленим один від одного об'єктів. Це відстань обчислюється таким чином:

$$p(x, x') = \sum_i^n (x_i - x_i)^2. \quad (2.2)$$

Відстань міських кварталів (Манхеттенська відстань).

Ця відстань є середнім різниць по координатах. У більшості випадків ця міра відстані приводить до таких же результатів, як і для звичайного відстані Евкліда. Однак для цього заходу вплив окремих великих різниць (викидів) зменшується (тому що вони не зводяться в квадрат). Формула для розрахунку манхеттенської відстані:

$$p(x, x') = \sum_i^n |x_i - x_i| \quad (2.3)$$

Відстань Чебишева.

Це відстань може виявитися корисним, коли потрібно визначити два об'єкти як різні, якщо вони розрізняються за якоюсь однією координатою. Відстань Чебишева обчислюється за формулою:

$$p(x, x') = \max (|x_i - x'_i|). \quad (2.4)$$

Степенна відстань.

Застосовується в разі, коли необхідно збільшити або зменшити вагу, що відноситься до розмірності, для якої відповідні об'єкти сильно відрізняються. Степенна відстань обчислюється за такою формулою:

$$p(x, x') = \sqrt[r]{\sum_i^n (x_i - x_i)^p}. \quad (2.5)$$

де r і p - параметри, що визначаються користувачем. Параметр p відповідальний за поступове зважування різниць за окремими координатами, параметр r відповідальний за прогресивне зважування великих відстаней між об'єктами. Якщо обидва параметри - r і p - дорівнюють двом, то це відстань збігається з відстанню Евкліда.

Вибір метрики повністю повністю залежить від розробника, оскільки результати кластеризації можуть суттєво відрізнятися при використанні різних заходів.

Класифікація методів.

Взагалі існує дві основні класифікації методів кластеризації:

– ієрархічні та плоскі;

Ієрархічні методи (також звані методи таксономії) будують не одне розбиття вибірки на непересічні кластери, а систему вкладених розбитих кластерів. Тобто на виході ми отримуємо дерево кластерів, коренем якого є вся вибірка, а листям - найбільш дрібні кластера.

Плоскі методи будують одне розбиття об'єктів на кластери.

– чіткі та нечіткі;

Чіткі (або непересічні) методи в яких кожному об'єкту вибірки ставлять у відповідність номер кластера, тобто кожен об'єкт належить тільки одному кластеру. Нечіткі (або пересічні) методи в яких кожному об'єкту ставлять у відповідність набір речових значень, що показують ступінь відносини об'єкта до кластерів. Тобто кожен об'єкт відноситься до кожного кластеру з певною ймовірністю.

Об'єднання кластерів.

У разі використання ієрархічних алгоритмів постає питання, як об'єднувати між собою кластера, як обчислювати «відстані» між ними. Існує кілька метрик:

– одиночний зв'язок або відстані найближчого сусіда;

У цьому методі відстань між двома кластерами визначається відстанню між двома найбільш близьким об'єктами в різних кластерах. Результуючі кластери мають тенденцію об'єднуватися в ланцюжки.

– повний зв'язок або відстань найбільш віддалених сусідів;

У цьому методі відстані між кластерами визначаються найбільшою відстанню між будь-якими двома об'єктами в різних кластерах (тобто найбільш віддаленими сусідами). Цей метод зазвичай працює дуже добре, коли об'єкти походять з окремих груп. Якщо ж кластери мають подовжену форму або їх природний тип є ланцюговим, то цей метод непридатний.

– незважене попарне середнє;

У цьому методі відстань між двома різними кластерами обчислюється як середня відстань між усіма парами об'єктів в них. Метод ефективний, коли об'єкти формують різні групи, проте він працює однаково добре і в випадках протяжних ланцюгових кластерів.

– зважене попарне середнє;

Метод ідентичний методу невиваженого попарного середнього, за винятком того, що при обчисленнях розмір відповідних кластерів (тобто число об'єктів, що містяться в них) використовується в якості вагового коефіцієнта. Тому даний метод повинен бути використаний, коли передбачаються нерівні розміри кластерів.

– незважений центроїдний метод;

У цьому методі відстань між двома кластерами визначається як відстань між їх центрами тяжкості.

– зважений центроїдний метод;

Цей метод ідентичний попередньому, за винятком того, що при обчисленнях використовуються ваги для обліку різниці між розмірами кластерів. Тому, якщо є або підозрюються значні відмінності в розмірах кластерів, цей метод виявляється переважно попереднього.

Методи ієрархічної кластеризації.

Серед алгоритмів ієрархічної кластеризації виділяються два основних типи: зростаючі та спадаючі алгоритми. Спадаючі алгоритми працюють за принципом «зверху-вниз»: на початку всі об'єкти поміщаються в один кластер, який потім розбивається на все більш дрібні кластери. Більш поширені зростаючі алгоритми, які на початку роботи поміщають кожен об'єкт в окремий кластер, а потім об'єднують кластери в все більші, поки всі об'єкти вибірки не будуть міститися в одному кластері. Таким чином будується система вкладеного розбиття. Результати таких алгоритмів зазвичай представляють у вигляді дерева - дендрограмми. Класичний приклад такого дерева - класифікація тварин і рослин.

Для обчислення відстаней між кластерами частіше все користуються двома відстанями: одиночній зв'язком або повним зв'язком.

До недоліку ієрархічних алгоритмів можна віднести систему повних розбиття, яка може бути зайвої в контексті розв'язуваної задачі.

Алгоритми квадратичної помилки.

Завдання кластеризації можна розглядати як побудова оптимального розбиття об'єктів на групи. При цьому оптимальність може бути визначена як вимога мінімізації середньоквадратичної помилки розбиття:

$$e^2(X, L) = \sum_{j=1}^{K} \sum_{i=1}^{n_i} ||x_i^{(j)} - c_j||^2, \qquad (2.6)$$

де c_i - центр мас кластера j (точка з середніми значеннями характеристик для даного кластера). Алгоритми квадратичної помилки відносяться до типу плоских алгоритмів. Найпоширенішим алгоритмом цієї категорії є метод k-середніх. Цей алгоритм будує задане число кластерів,

розташованих якнайдалі один від одного. Робота алгоритму ділиться на кілька етапів:

– випадково вибрати k точок, які є початковими центрами мас кластерів;

– віднести кожен об'єкт до кластеру з найближчим центром мас;

– перерахувати центри мас кластерів відповідно до їх поточним складом;

– якщо критерій зупинки алгоритму не задоволений, повернутися до другого пункту;

Як критерій зупинки роботи алгоритму зазвичай вибирають мінімальне зміна среднеквадратичної помилки. Так само можливо зупиняти роботу алгоритму, якщо на кроці 2 не було об'єктів, що перемістилися з кластера в кластер.

До недоліків даного алгоритму можна віднести необхідність задавати кількість кластерів для розбиття.

Нечіткі алгоритми.

Найбільш популярним алгоритмом нечіткої кластеризації є алгоритм c-середніх (c-means). Він являє собою модифікацію методу k-середніх. Кроки роботи алгоритму:

– вибрати початкове нечітке розбиття n об'єктів на k кластерів шляхом вибору матриці приналежності U розміру n x k;

– використовуючи матрицю U, знайти значення критерію нечіткої помилки;

– перегрупувати об'єкти з метою зменшення цього значення критерію нечіткої помилки;

– повертатися до 2 кроку до тих пір, поки зміни матриці U не стануть незначними.

Цей алгоритм може не підійти, якщо заздалегідь невідомо число кластерів, або необхідно однозначно віднести кожен об'єкт до одного кластеру.

Методи основані на теорії графів.

Суть таких алгоритмів полягає в тому, що вибірка об'єктів представляється у вигляді графа G = (V, E), вершинам якого відповідають об'єкти, а ребра мають вагу, рівний «відстані» між об'єктами. Перевагою графових методів кластеризації є наочність, відносна простота реалізації і можливість внесення різних удосконалень, засновані на геометричних міркуваннях. Основними алгоритмами є алгоритм виділення зв'язкових компонент, алгоритм побудови мінімального покриває остовного дерева і методу пошаровим кластеризації.

Метод виділення зв'язкових компонентів.

В методі виділення зв'язкових компонент задається вхідний параметр R і в графі видаляються всі ребра, для яких відстані більше R. Сполученими залишаються тільки найбільш близькі пари об'єктів. Сенс алгоритму полягає в тому, щоб підібрати таке значення R, що лежить в діапазон всіх відстаней, при якому граф розвалиться на кілька зв'язкових компонент. Отримані компоненти і є кластери.

Для підбору параметра R зазвичай будується гістограма розподілів попарних відстаней. У завданнях з добре вираженою кластерної структурою даних на гістограмі буде два піки - один відповідає внутрікластерним відстаням, другий - міжкластерним відстані. Параметр R підбирається із зони мінімуму між цими піками. При цьому управляти кількістю кластерів за допомогою порога відстані досить важко[2-7].

2.2 Метод К-найближчих сусідів

Зазвичай, люди зустрічаючи нові завдання на своєму шляху, звикли згадувати аналогічні ситуації в минулому. Це перших логічний крок на шляху в рішенні якої небудь задачі. Про властивості нових об'єктів ми судимо, покладаючись на схожі та знайомі спостереження. Наприклад, зустрівши іноземця на вулиці, ми можемо здогадатися про його походження з

мови, жестів і зовнішності. Для цього необхідно згадати найбільш схожого людини на нього, походження якого відомо.

Так, подібно до наведеного вище прикладу, схожість об'єктів лежить в основі методу k-найближчих сусідів (k-nearest neighbor algorithm, KNN). Метод здатний виділити серед всіх спостережень k відомих об'єктів (k-найближчих сусідів), схожих на новий невідомий раніше об'єкт. На основі класів найближчих сусідів виноситься рішення щодо нового об'єкта. Важливим завданням даного методу є підбір коефіцієнта k - кількість записів, які будуть вважатися близькими. Метод KNN широко застосовується в Data Mining [1].

Розглянемо задачу класифікації. Нехай є m спостережень, кожному з яких відповідає запис в таблиці. Всі записи належать якомусь класу. Необхідно визначити клас для нового запису.

На першому кроці методу слід задати число k - кількість найближчих сусідів. Якщо прийняти k = 1, то метод втратить узагальнюючу здатність (тобто здатність видавати правильний результат для даних, що не зустрічалися раніше в методі) так як нового запису буде присвоєно клас близькому до неї. Якщо встановити занадто велике значення, то багато локальних особливостей не буде виявлено.

На другому кроці знаходяться k записів з мінімальним відстанню до вектора ознак нового об'єкта (пошук сусідів). Функція для розрахунку відстані повинна відповідати наступним правилам:

- $d(x,y) \geq 0$, $d(x,y) = 0$ тоді и тільки тоді, коли $x = y$;
- $d(x,y) = d(y,x)$;
- $d(x,z) \leq d(x,y) + d(y,z)$, при цьому то чки x, y, z не належать одній прямій.

x, y, z – вектори ознак порівнюваних об’єктів.

Для впорядкованих значень атрибутів знаходиться Евклідова відстань:

$$D_E = \sqrt{\sum_i^n (x_i - y_i)^2}, \quad (2.7)$$

де n - кількість атрибутів.

При знаходженні відстані іноді враховують значимість атрибутів. Вона визначається експертоми або аналітиками суб'єктивно, покладаючись на власний досвід. У такому випадку при знаходженні відстані кожен і-ий квадрат різниці в сумі множиться на коефіцієнт Zi. Наприклад, якщо атрибут A в три рази важливіший за атрибут B (Z(A) = 3, Z(B) = 1), то відстань буде перевірюватись в таким способом:

$$D_E = \sqrt{3(x_A - y_A)^2 + (x_B - y_B)^2}. \quad (2.8)$$

Подібний прийом називають розтягуванням осей, що дозволяє знизити помилку класифікації[1].

Варто зробити зазначити, що якщо для запису B найближчим сусідом являється A, то зовсім не означає, що B – найближчий сусід A.

На наступному кроці, коли знайдені записи, найбільш схожі на нову, необхідно вирішити, як вони впливають на клас нового запису. Для цього використовується функція поєднання. Одним з основних варіантів такої функції є просте незважене голосування.

Просте незважене голосування.

Спочатку, задавши число k, ми визначили, скільки записів буде мати право голосу при визначенні класу. Потім ми виявили ці записи, відстань від яких до нової виявилося мінімальним. Тепер можна приступити до простого невиважені голосуванню.

Відстань від кожного запису при голосуванні тут більше не грає ролі. Всі мають рівні права у визначенні класу. Кожна вільна позиція голосує за клас, до якого належить. Нового запису присвоюється клас, який набрав найбільшу кількість голосів.

В ході роботи методу існують випадки, коли декілька класів набрали рівну кількість голосів. Цю проблему знімає зважене голосування.

Зважене голосування.

У такій ситуації враховується також і відстань до нового запису. Чим менше відстань, тим більш значущий внесок вносить голос. Голоси за клас знаходяться за наступною формулою:

$$votes(class) = \sum_{i=1}^{n} \frac{1}{d^2(X,Y_i)}. \quad (2.9)$$

де $d^2(X, Y_i)$ - квадрат відстані від відомої записи Y_i до нової X, n - кількість відомих записів класу, для якого розраховуються голос, class - найменування класу.

Новий запис співвідноситься з класом, який набрав найбільшу кількість голосів. При цьому ймовірність того, що кілька класів наберуть однакові голоси, набагато нижче. Цілком очевидно, що при k = 1 нового запису присвоюється клас самого найближчого сусіда.

Області застосування методу KNN

Алгоритм k-найближчих сусідів має широке застосування. наприклад:

– виявлення шахрайства. Нові випадки шахрайства можуть бути схожі на ті, які відбувалися колись в минулому. Алгоритм KNN може розпізнати їх для подальшого розгляду;

– передбачення відгуку клієнтів. Можна визначити відгук нових клієнтів за даними з минулого;

– медицина. Алгоритм може класифікувати пацієнтів за різними показниками, на підставі даних минулих періодів.

Переваги та недоліки методу KNN.

Переваги:

– алгоритм стійкий до аномальних викидів, тому що ймовірність попадання такого запису в число k-найближчих сусідів мала. Якщо ж це

сталося, то вплив на голосування (особливо зважене) (при k> 2) також, швидше за все, буде незначним, і, отже, малим буде і вплив на підсумок класифікації;

– програмна реалізація алгоритму відносно проста;

– результат роботи алгоритму легко піддається інтерпретації. Експертам в різних областях цілком зрозуміла логіка роботи алгоритму, заснована на знаходженні схожих об'єктів;

– можливість модифікації алгоритму, шляхом використання найбільш придатних функцій поєднання і метрик дозволяє підлаштовувати алгоритм під конкретну задачу.

Алгоритм KNN володіє і рядом недоліків. По-перше, набір даних, який використовується для алгоритму, повинен бути репрезентативним. По-друге, модель не можна "відокремити" від даних: для класифікації нового прикладу потрібно використовувати всі приклади. Ця особливість сильно обмежує використання алгоритму

2.3 Метод найменших квадратів

Регресія та Метод найменших квадратів

Перш ніж почати, слід зауважити що Метод найменших квадратів дуже тісно зв’язаний з поняттям регресійний аналіз, тому перш за все треба визначитись в понятті, що таке регресійний аналіз та лінійна регресія.

Регресійний аналіз – це статистичний метод дослідження впливу однієї або декількох незалежних змінних { X(1), X(2) … X(n) } на залежну змінну { Y }. У випадку з лінійною регресією, вплив змінних виражається через лінійну функцію . Незалежні змінні інакше називають регресором, а залежні змінні - критеріальними. Термінологія залежних і незалежних змінних відображає математичну залежність змінних.

Методу найменших квадратів належить одному із базових методів регресійного аналізу даних, для оцінювання невідомих параметрів

регресійних моделей по вибірковим даним. МНК виконує функцію знаходження оптимальних параметрів лінійної регресії, таких щоб сума квадратів похибки була мінімальною. Суть методу полягає в мінімзації евклідового простору між двома векторами - вектором відновлених значень залежної змінної і вектором фактичних значень залежної змінної.

Суть методу найменших квадратів.

Нехай x - набір n невідомих змінних (параметрів), $F_i(x), i = 1, \dots, m$, $m > n$, де *n* сукупність функцій від цього набору змінних. Завдання полягає в підборі таких значень *x*, щоб значення цих функцій були максимально близькі до деяких значень y_i . По суті мова йде про «рішення» перевизначення системи рівнянь $f_i(x) = y_i, i = 1, \dots, m$ в зазначеному сенсі максимальній близькості лівою та правої частини системи. сутність МНК полягає у виборі в якості «заходи близькості» суми квадратів відхилень лівих і правих частин $|f_i(x) - y_i|$. Таким чином, сутність МНК може бути виражена таким чином:

$$\sum_i e_i^2 = \sum_i \big(y_i - f_i(x)\big)^2 \to \min_x. \qquad (2.10)$$

У разі, якщо система рівняння має рішення, то найменше значення суми квадратів дорівнюватиме нулю і можуть бути знайдені точні рішення системи рівнянь аналітично або, наприклад, різними чисельними методами оптимізації. Якщо система перевизначена, тобто кажучи, кількість незалежних рівнянь більше кількості шуканих змінних, то система не має точного рішення і метод найменших квадратів дозволяє знайти певний «оптимальний» вектор *x* в сенсі максимальній близькості векторів *y* і *f(x)* або максимальній близькості вектора відхилень *e* до нуля (близькість мається на увазі евклидова відстань)[1].

Метод найменших квадратів в регресивному аналізі

Нехай існує *n* значень деякої змінної *y* (це можуть бути результати спостережень, експериментів і т. д.) та відповідних змінних *x*. Завдання

полягає в тому, щоб взаємозв'язок між *y* і *x* апроксимувати деякою функцією *f (x, b)*, відомої з точністю до деяких невідомих параметрів b, тобто фактично знайти найкращі значення параметрів b, максимально наближають значення

f(x, b) до фактичних значенням *y*. Фактично це зводиться до випадку «рішення» перевизначення системи рівнянь щодо b:

$$f(x_t, b) = y_t, t = 1, \ldots, n. \quad (2.11)$$

У регресійному аналізі і зокрема використовуються імовірнісні моделі залежності між змінними $y_t = f(x_t, b) + \varepsilon_t$, де ε_t - так звані випадкові помилки моделі.

Відповідно, відхилення спостережуваних значень у від модельних *f(x,b)* передбачається вже в самій моделі. Сутність МНК (класичного) полягає в тому, щоб знайти такі параметри b, при яких сума квадратів відхилень (помилок, для регресійних моделей їх часто називають залишками регресії) e_t буде мінімальною:

$$b_{OLS} = \arg\min_b RSS(b), \quad (2.12)$$

де RSS - Residual Sum of Squares визначається як:

$$RSS(b) = e^t e = \sum_{t=1}^{1} e_t^2 = \sum_{t=1}^{n} (y_t - f(x_t, b))^2. \quad (2.13)$$

У загальному випадку рішення цього завдання може здійснюватися чисельними методами оптимізації (мінімізації). У цьому випадку говорять про нелінійну МНК. У багатьох випадках можна отримати аналітичне рішення. Для вирішення завдання мінімізації необхідно знайти стаціонарні точки функції RSS (b), продифференцировав її з невідомими параметрами b, прирівнявши похідні до нуля і вирішивши отриману систему рівнянь:

$$\sum_{t=1}^{n}\left(y_t - f(x_t, b)\right)\frac{\partial f(x_t,b)}{\partial b} = 0. \quad (2.14)$$

2.4 Метод нечіткої кластерізації С середніх

Метод кластеризації організовує елементи в групи на основі критеріїв подібності. Метод нечітких кластерізацій це метод, де кожен елемент може належать до декількох груп, де ступінь членства для кожного елементу визначається розподілом ймовірності по кластерам.

Метод нечіткої кластерізації корисний, коли потрібна кількість кластерів, заздалегідь визначена, таким чином, алгоритм намагається поставити кожен з даних точок в один з кластерів. Це робить метод особливим тому що він не визначає абсолютне членство точки даних для даного кластера, замість цього він обчислює ймовірність (ступінь членства), що точка даних буде належати до цього кластера. Отже, в залежності від точності кластеризації, що вимагається на практиці, можлива відповідна толерантності відносно місця розташування. Оскільки абсолютне членство не розраховується, метод може бути надзвичайно швидким, оскільки кількість ітерацій, необхідних для досягнення конкретної кластеризації, відповідає необхідній точності.

У кожній ітерації алгоритму FCM наступна цільова функція *J* мінімізується:

$$J = \sum_{i=1}^{N}\sum_{j=1}^{C}\delta_{ij}\left|\left|x_i - c_j\right|\right|^2. \quad (2.15)$$

де N – кількість даних точок , C – кількість необхідних кластерів, C_j– центральний вектор кластера j, а δ_{ij}- ступінь членства для i-ої точки x_i даних

в кластері j. Норма, $|x_i - c_i|$ заходи подібність (або близькість) точки x_i даних до центру вектора c_j кластера j. Важливо пам’ятати, що в кожному

ітерація, алгоритм підтримує вектор центру для кожного з кластерів. Ці точки даних обчислюються як середньозважене значення точок даних, де ваги визначаються ступенями членства.

Ступінь членства

Для заданої точки x_i, ступінь його членства в кластері *j* розраховується наступним чином

$$\delta_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{||x_i - c_j||}{||x_i - c_k||}\right)^{\frac{2}{m-1}}}, \quad (2.16)$$

де, m - коефіцієнт нечіткості, а вектор центру cj обчислюється наступним чином:

$$c_j = \frac{\sum_{i=1}^{N} \delta_{ij}^{m} * x_i}{\sum_{i=1}^{N} \delta_{ij}^{m}}. \quad (2.17)$$

У рівнянні 2.16 вище, δ_{ij} - значення ступеня членства, розраховане на попередній ітерації. на початку алгоритму ступінь членства для точки даних *i* в кластері j ініційовано з випадковим значенням θ_{ij} , $0 \leq \theta_{ij} \leq 1$ при цьому

$\sum_{j}^{C} \delta_{ij} = 1.$

Коефіцієнт нечіткості.

У рівняннях 2.15 та 2.16 коефіцієнт нечіткості m, де 1 <m <∞, вимірює належність необхідного кластеру. Це значення визначає, скільки кластерів може перекриватися один з одним. Це значення визначає, наскільки кластери можуть перекривати один одного. Чим вище значення m, тим більше перекриваються кластери. Іншими словами, чим вище коефіцієнт нечіткості, який використовує метод, тим більше число точок даних потрапить всередину групи "нечітка", де ступінь членства не є ні 0, ні 1, а десь посередині[12].

Умови виключення

Необхідна точність ступеня членства визначає номер ітерацій, виконаних методом нечіткої кластерізації. Цей показник точності розраховується за ступенем членство від однієї ітерації до наступної, беручи найбільше з цих значень у всіх точках даних, які враховують всі кластери. Якщо ми представляємо міру точності між ітерацією k і $k + 1$ з ϵ, ми обчислимо

його значення наступним чином: $\epsilon = \Delta_i^N \Delta_j^C |\delta_{ij}^{K+1} - \delta_{ij}^K|$, де δ_{ij}^k та δ_{ij}^{k+1} складають відповідно ступінь членства на ітерації k та $k + 1$, а оператор Δ, коли він поставляє вектор значень, повертає найбільше значення в цьому векторі.

3 РЕАЛІЗАЦІЯ МЕТОДІВ КЛАСТЕРІЗАЦІЇ

3.1 Реалізація методу К-найближчих сусідів

Програмний застосунок, демонструє комфортний інтерфейс для роботи з методами аналізу даних та її візуалізації. Додаток розроблений згідно з архітектурою REST, отже вся розрахункова частина виконується на стороні серверу для покращення швидкодії методів. Почнемо з методу К найближчих сусідів. Для того щоб зрозуміти принципи роботи методу та використовувати його на практиці, інтерфейс додаток представляє основний функціонал для роботи в рамках методу.

Як вже було сказано раніше метод К-найближчих сусідів залежить від початкового стану даних, тобто для того щоб метод працював необхідно мати вже існуючі данні на основі яких буде виконуватись подальша кластерізація.

На рисунку 3.1 показаний перший етап роботи - загрузка даних для роботи методу.

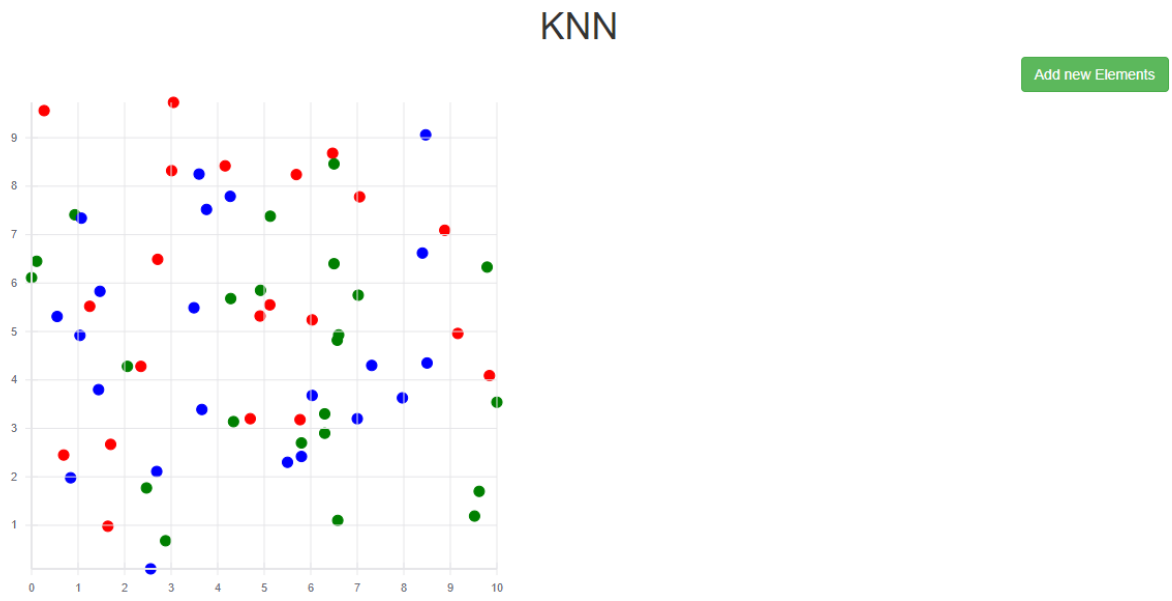


Рисунок 3.1 – Загрузку тестових даних

Як ми можемо бачити данні розташовані випадковим чином. В наборі даних існує три основних класи – червоний, зелений та синій. Та кожний елемент вибірки вже має свою координати відносно площини.

Функціонал додатку дозволяє читати базову інформацію кожного елементу при наведені курсору на нього. Інформація про елемент за початкового стану не несе в собі особливої інформації про логіку роботи методу, тому що вона не має сусідів завдяки яким елемент був кластерізований. В такому випадку можна побачити тільки точні координати та клас якому елемент належить рисунок 3.2.

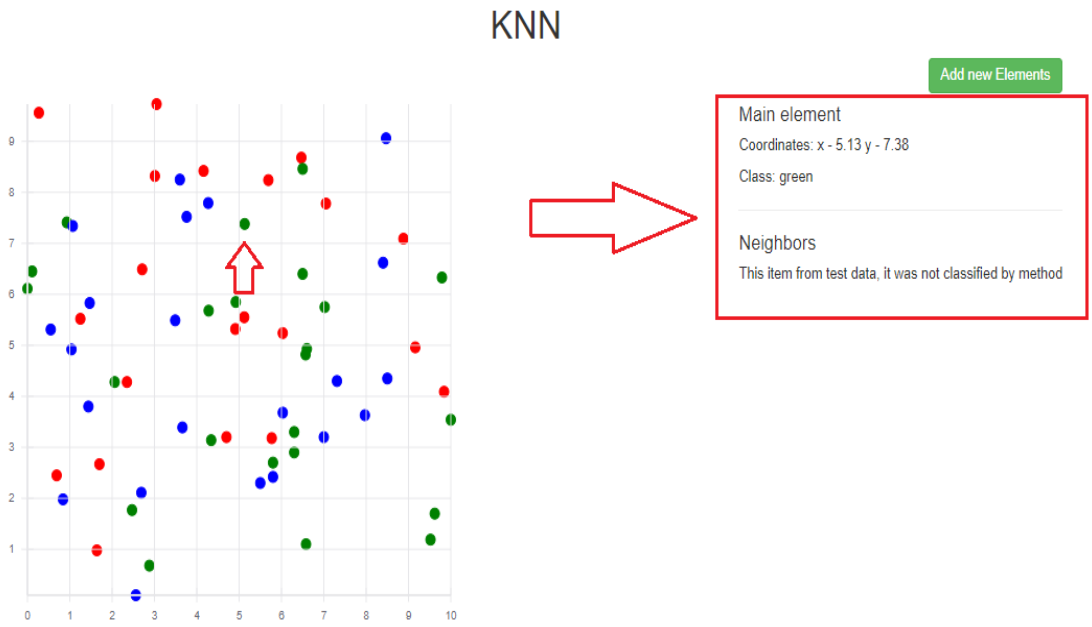


Рисунок 3.2 – Приклад демонстрації інформації елементу

Наступним кроком роботи методу є кластерізація нового елементу. Для цього необхідно викликати форму додавання нового елементу «Add new Elements». Функціонал дозволяє кластерізувати одночасно скільки завгодно елементів. На першому етапі додавання елементів указується кількість К найближчих сусідів які будуть приймати рішення що до якого майбутнього класу елементу (рис. 3.3).

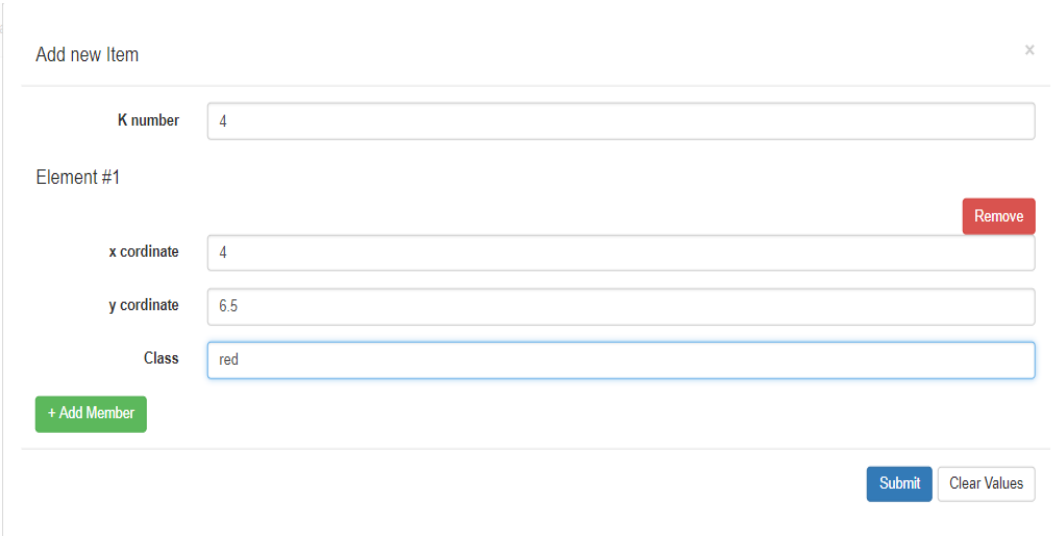


Рисунок 3.3 – Створення нового елементу

На цьому етапі відбувається основна робота методу к найближчих сусідів. Після того я к користувач натискає кнопку відправки форми, серверна частина додаток приймає данні в форматі JSON (рис. 3.4).


```
{
  "k":"4",
  "elements":[
    {
      "x":4,
      "y":6.5,
      "classElement":"red"
    }
  ]
}
```


Рисунок 3.4 – Приклад вигляду тіла запиту в форматі JSON

Після того як сервер додатку отримав данні починається процес кластерізації. На основі формули евклідової відстані (формула 2.1) знаходяться найближчі елементи відносно координат об’єкту кластерізації.

Завдяки тому, чіткій кількості вказаних К сусідів, метод зупиняє пошук коли кількість сусідніх елементів рівна числу К.

Наступним кроком йде етап зважування, як говорилось раніше, цей етап може бути різним, самим стандартним методом зважування це пошук класу кількість яких більше в множині більше.

Після всіх кроків методу додаток вертає результат з новим кластрізованим елементом. При наведенні курсору на щойно кластерізований елемент можна чітко зрозуміти чому саме об’єкт був віднесений то того чи іншого класу рисунок 3.5.

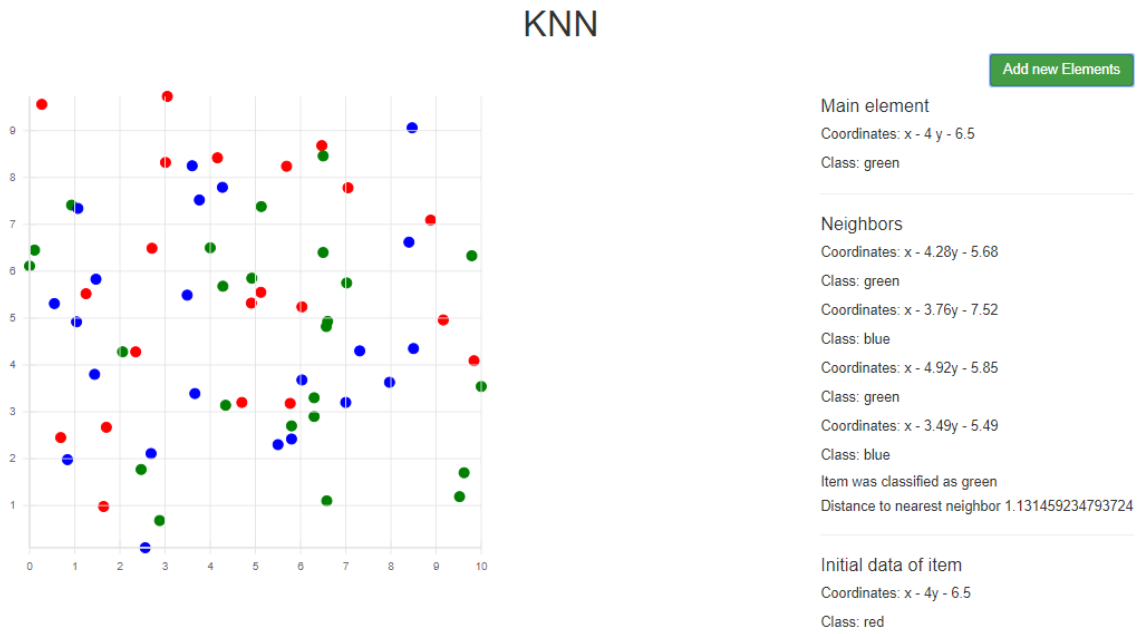


Рисунок 3.5 – Приклад демонстрації кластерізованого елементу

В даному випадку видно що елемент з координатами x – 4, y – 6.5 був присвоєний клас green на основі 4 найближчих сусідів, також показана відстань до найближчого сусіду.

3.2 Реалізація методу нечіткої кластерізації С середніх

Принципи нечіткої логіки можуть бути використані для кластерізації багатовимірних даних, присвоюючи кожній точці членство в кожному кластерному центрі з ймовірністю від 0 до 100 відсотків. Це може бути дуже

потужним в порівнянні з традиційними кластеризаціями з жорстким порогом, де кожній точці присвоюється чітке, точне позначення.

Задача додатку, в рамках реалізації методу С середніх, знайти найоптимальнішу кількість центральних точок кластерізації, та досягти максимального нечіткого коефіцієнту розподілу, який залежить від числа С середніх точок.

Отже, задача додатку найти кількість С середніх точок при яких данні будуть кластерізовані краще.

Для досягнення цієї цілі, на першому етапі роботи додатку генеруються випадкові данні, відносно трех центрів (рис. 3.6).

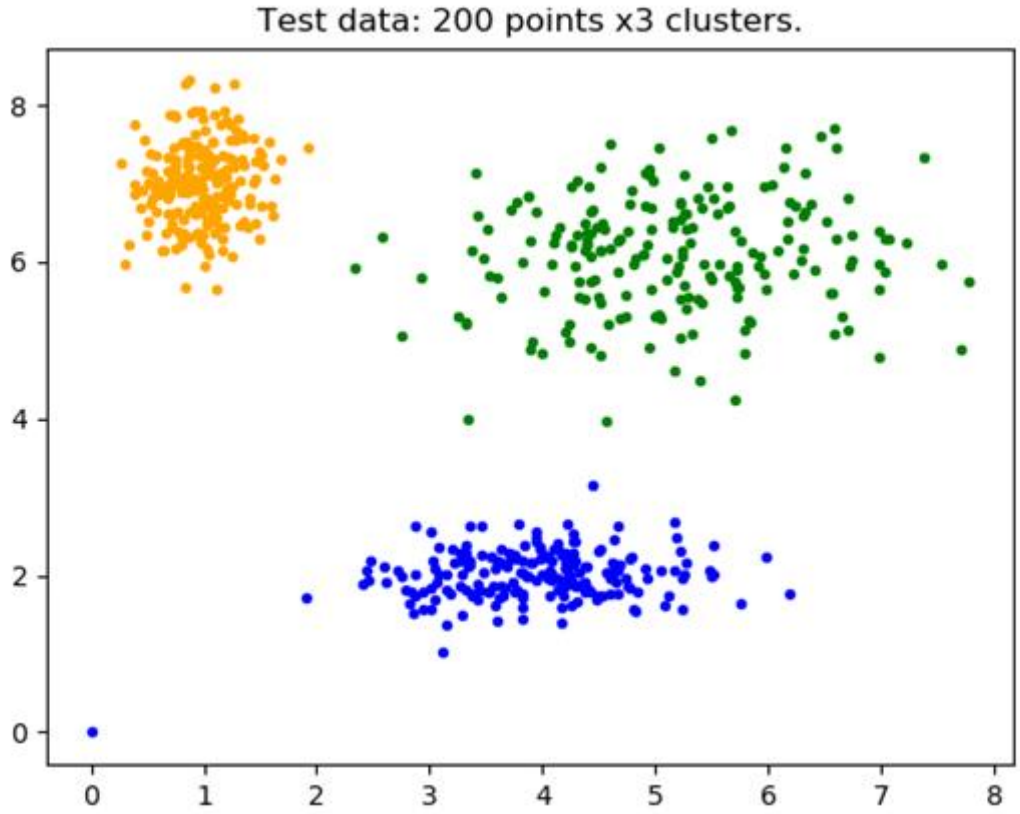


Рисунок 3.6 – Генерація тестових даних

Наступним кроком виконується безпосередньо сама кластерізація. На рисунку 3.6 видно кластерізовані дані відносно трьох точок. Кожних об’єкт з певною ймовірністю належить до всіх точок на площині одночасно, саме це визначає положення кожного об’єкту у просторі. В даному випаду, ми знаємо кількість точок, це дає нам інформацію про коефіцієнт розподілу, але що як

би кількість точок була невідома, можливо в цьому випаду коефіцієнт був би більшим.

Наступним етапом виконання додатку є – пошук найбільшого коефіцієнта розподілу даних. Для цього генерується 10 можливих варіантів розподілу з різною кількістю центрів (рис. 3.7).

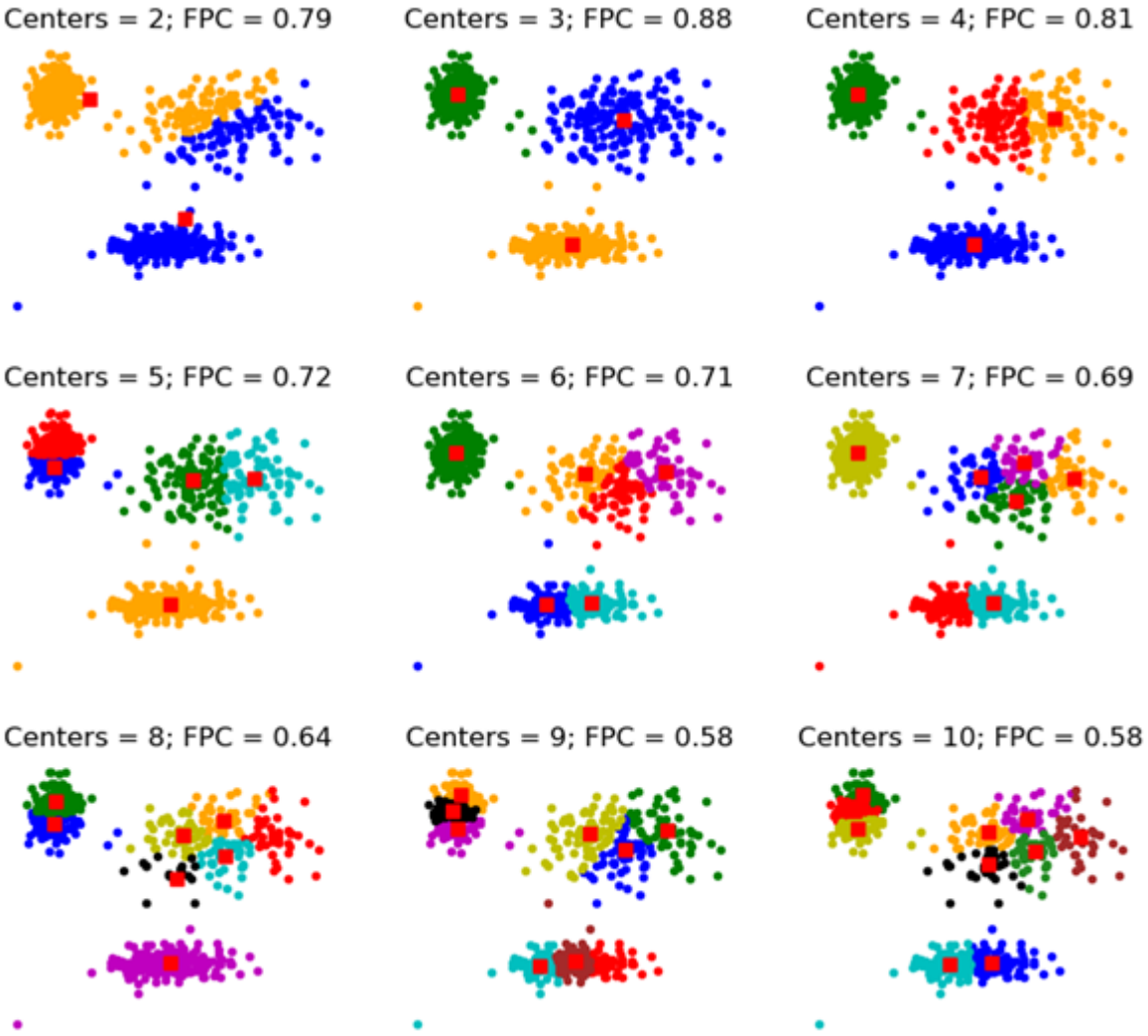


Рисунок 3.7 – Варіанти кластирізації з різною кількістю центрів

Нечіткий коефіцієнт розподілу визначається в діапазоні від 0 до 1, де 1 це найкращий. Це метрика, яка розповідає, наскільки чисто дані описуються певною моделлю. Далі збирається набір даних, який, як відомо, має три кластери, кілька разів, з 2 до 9 кластерів. Результати кластерізації видно на графіку рисунку 3.8

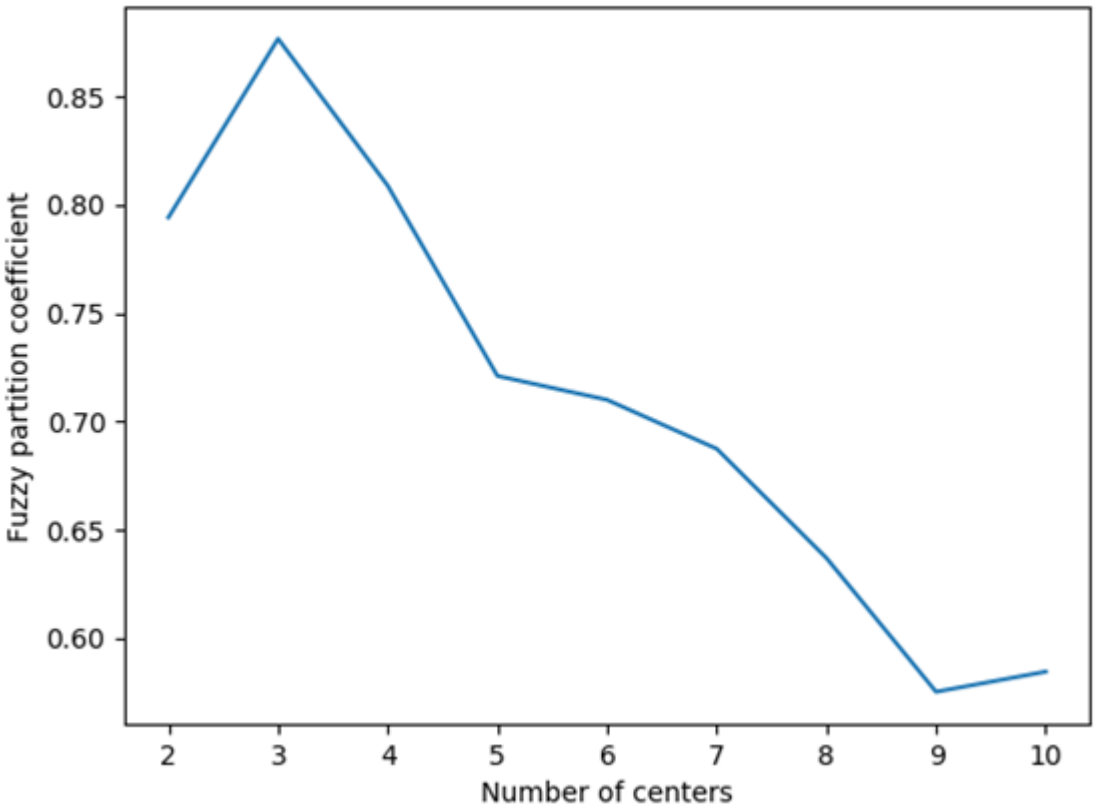


Рисунок 3.8 – Демонстрація результатів кластеризації

Як ми бачимо, ідеальна кількість центрів дорівнює 3. Наявність нечіткого коефіцієнту розподілу може бути дуже корисною, коли структура даних не зрозуміла.

Слід зауважте, аналіз починався з двох центрів, а не з одного; кластеризація набору даних з єдиним кластерним центром є тривіальним рішенням і буде за визначенням рівне 1.

Наступним кроком э додавання нових об’єктів до існуючої моделі. Такий процес називається прогнозом. Прогноз вимагає наявності кластерізованої моделі та нових даних для класифікації.

Як вже відомо найкраща модель має три кластерні центри. Надалі буде використовуватись ця модель для виконання прогнозування. Додаток генерує нові уніфіковані дані та прогнозує, до якого кластеру нележить кожна нова точка даних (рис. 3.9).

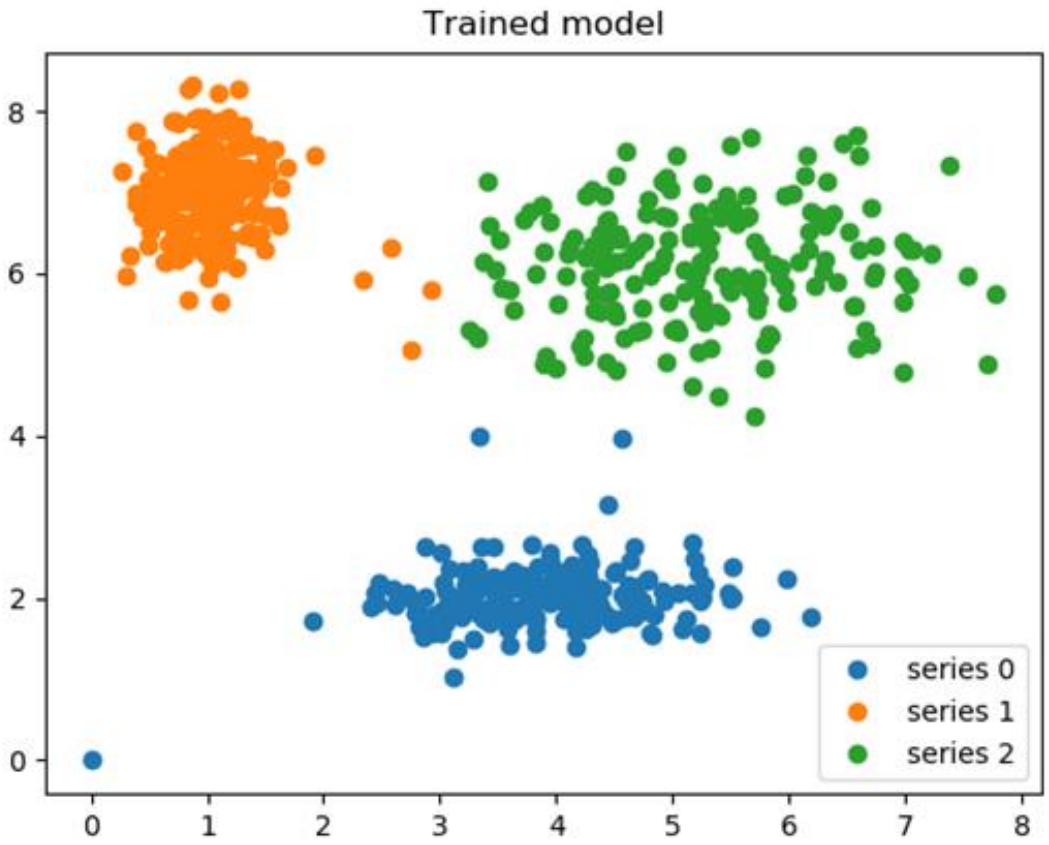


Рисунок 3.9 – Демонстрація даних для прогнозування

Останнім кроком роботи додатку є прогнозування даних. Додаток генерує дані в діапазоні від 0 до 10, та виконує процес прогнозування в рамках відомої моделі рисунок.

Результат роботи прогнозування представлено на рисунку3.10.

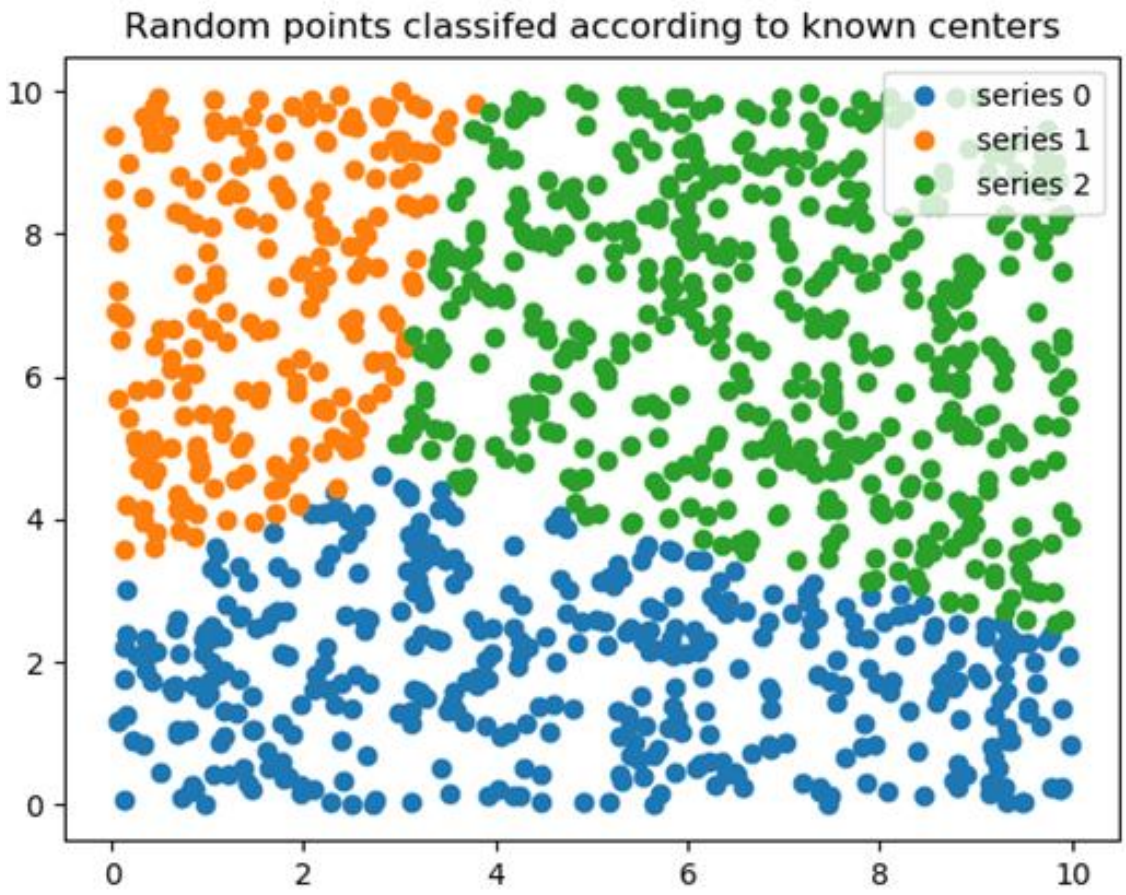


Рисунок 3.10 – Результат прогнозування

3.3 Реалізація методу найменших квадратів

В рамках програмного додатку реалізований метод найменших квадратів.

На першому етапі роботи додаток генерує довільний набір даних для подальшої роботи методу. Результат готових даних несе в собі інформацію про положення точок відносно осі координат рисунок 3.11.

Задача додатку полягає в тому щоб розрахувати координати прямої так щоб сума квадратів довжин відстаней від усіх точок до цієї прямої мінімальною.

Результат робити додатку демонструє пряму яка має найоптимальніші координати відносно усіх точок на площині рисунок 3.12.

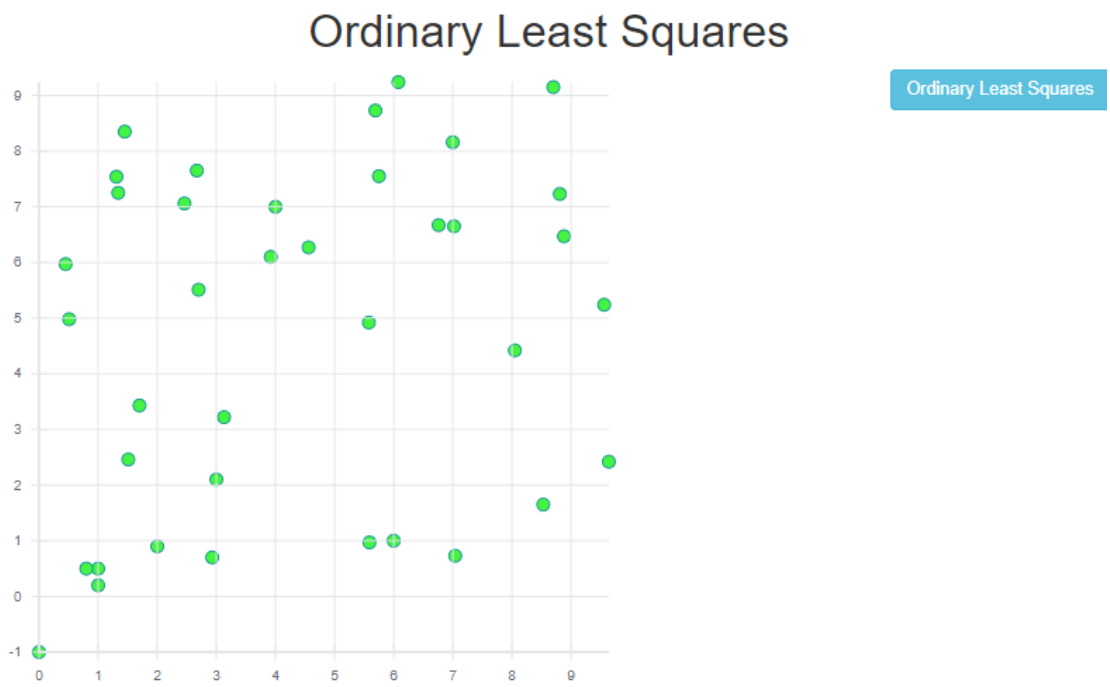


Рисунок 3.11 – Візуалізація даних

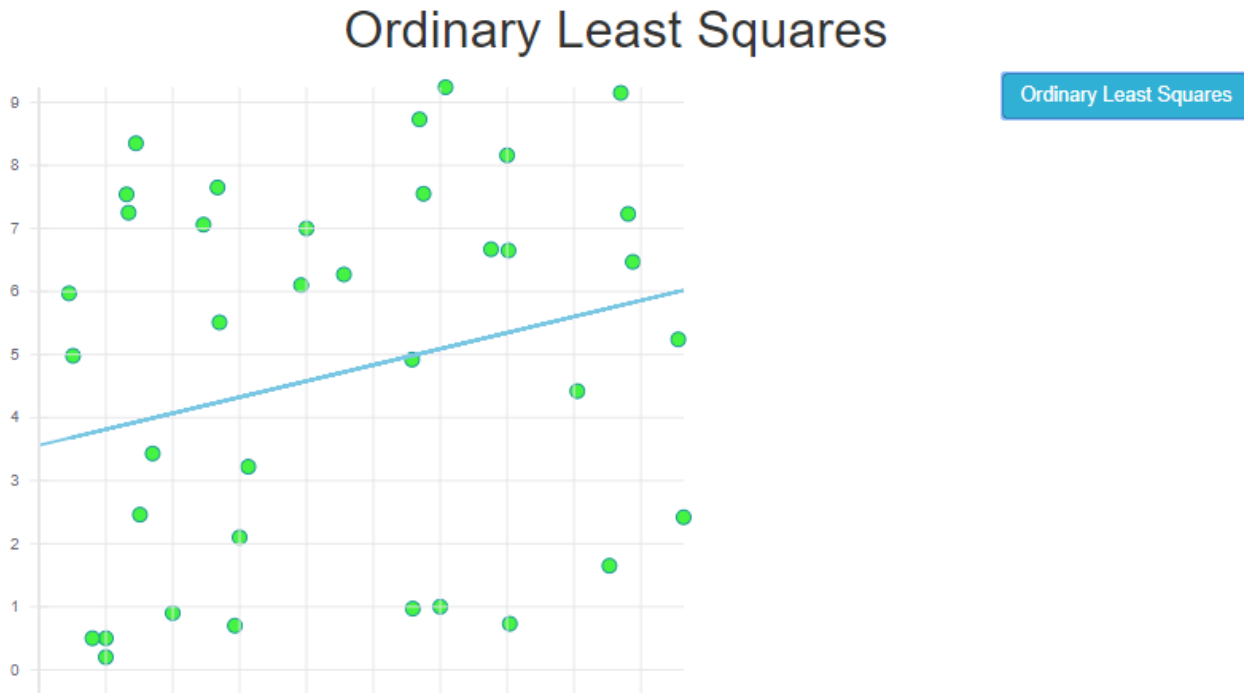


Рисунок 3.12 – Демонстрація роботи метод найменших квадратів

4 ОХОРОНА ПРАЦІ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

4.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання [12], які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне елект-рообладнання, зокрема персональні комп’ютери (ПК) та периферійні пристрої.

Основними робочими характеристиками персонального комп'ютера є наступні:

- робоча напруга U = +220В ± 5%;
- робочий струм I = 2А;
- споживана потужність P = 350 Вт.

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з ві-зуальними дисплейними терміналами електронно-

обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [13].

За умов роботи з ПК виникають наступні небезпечні та шкідливі чинники: несприятливі мі-крокліматичні умови, освітлення, електромагнітні випромінювання, забруднення повітря шкідливими речовинами (джерелом, яких можуть бути: принтер, сканер та інші джерела виділення багатьох хімічних речовин - напр., озону, оксидів азоту та аерозолів високодисперсних частинок тонера), шум, вібрація, електричний струм, електростатичне поле, напруженість трудового процесу та інше.

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 4.1).

Таблиця 4.1 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, принтерів, сканерів чи/або серверного обладнання для роботи	2	[14]
- підвищений рівень шуму на робочому місці	-//-	2	[15]
- підвищена або знижена рухливість повітря	-//-	1	[12]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[17]

Продовження таблиці 4.1

1	2	3	4
- підвищена напруженість електричного поля	-//-	2	[17]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[18]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	[18]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[13]
- фізичні (статичне – сидіння)	порушення умов праці (організації місця праці-сидіння користувача,) та організації робочого часу - безпервна робота)	2	[13]

4.2 Гігієнічні вимоги до параметрів виробничого середовища

4.2.1 Мікроклімат

Оптимальні значення для температури, відносної вологості й рухливості повітря для зазначеного робочого місця відповідають [14] і наведені в табл. 4.2:

Таблиця 4.2 – Норми мікроклімату робочої зони об’єкту

Період року	Категорія робіт	Температура С0	Відносна вологість %	Швидкість руху повітря, м/с
Холодна	легка-1 а	22 - 24	40 – 60	0,1
Тепла	легка-1 а	23 - 25	40 – 60	0,1

Дане приміщення обладнане системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до [14]. Рівні позитивних і негативних іонів у повітрі мають відповідати [14]. Для забезпечення оптимальних параметрів мікроклімату в приміщенні проводяться перерви в роботі співробітників, з метою його провітрювання. Існують спеціальні системи кондиціонування, які забезпечують підтримання в приміщенні балансу оптимальних параметрів мікроклімату. Контроль параметрів мікроклімату в холодний і теплий період року здійснюється не менше 3-х разів на зміну (на початку, середині, в кінці).

4.2.2 Освітлення

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає ДБН В. 2.5-28:2018 [18]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для будівель виробництв світловий коефіцієнт приймається в межах 1/6 - 1/10:

$$\sqrt{a^2 + b^2} \cdot S_b = (1/8 \div 1/10) \cdot S_n \qquad (4.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$$S_n = a \cdot b = 5 \cdot 5 = 25 \text{ м}^2$$

$$S_{вік} = 1/8 \cdot 25 = 3{,}125 \text{ м}^2$$

Приймаємо 2 вікна площею $S = 1{,}6 \text{ м}^2$ кожне.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 5 м, ширина 5 м, світильниками ЛПО2П, оснащеними лампами типа ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників п виробляється по формулі (4.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M} \qquad (4.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м2; S = 25 м2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.2), отримуємо:

$$n = \frac{300 \cdot 25 \cdot 1,1 \cdot 1,5}{5400 \cdot 0,575 \cdot 2} \approx 2.$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.2.3 Шум та вібрація, електромагнітне випромінювання

Рівень шуму, що супроводжує роботу користувачів персональних комп'ютерів (зумовлений як роботою системних блоків, клавіатури, так і друкуванням на принтерах, а також зовнішніми чинниками), коливається у межах 50–65 дБА [3]. Шум такої інтенсивності на тлі високого ступеня напруженості праці негативно впливає на функціональний стан користувачів. Тому на практиці рекомендують знижувати фактичний рівень шуму у приміщеннях, де створюють комп'ютерні програми, виконують теоретичні та творчі роботи, проводять навчання до 40 дБА, а в приміщеннях, де виконують роботу, що потребує зосередженості, — до 55 дБА. У залах опрацювання інформації та комп'ютерного набору рівні шуму не повинні перевищувати 65 дБА.

Шум часто є причиною зниження рівня працездатності, підвищення рівня загальної та професійної захворюваності, частоти виробничих травм. Шум є загальнобіологічним подразником, який негативно впливає на всі органи і системи організму. У разі тривалого систематичного впливу шуму

може виникнути патологія з переважним ураженням слуху, центральної нервової і серцево- судинної систем.

Для зниження шуму на шляху його поширення передбачається розміщення в приміщенні штучних поглиначів. Для зниження рівня шуму стелю або стіни вище 1.5 - 1.7 метра від підлоги повинні облицьовуватися звукопоглинальним матеріалом з максимальним коефіцієнтом звукопоглинання в області частот 63-8000 Гц. Додатковим звукопоглинанням в КВТ можуть бути фіранки, підвішені в складку на відстані 15-20 см. Від огорожі, виконані з щільної, важкої тканини. У приміщенні з ЕОМ коректований рівень звукової потужності не перевищує 45 дБА. Оскільки рівень шуму не перевищує гранично допустимих величин, які встановлені санітарними нормами, заходи для зниження шуму не проводяться.

Віброізоляцію можливо здійснювати за допомогою спеціальної прокладки під системний блок, який послаблює передачу вібрацій робочого столу. Вібрація на робочому місці в приміщенні, що розглядається, відповідає нормам [15]. Допустимий рівень вібрацій на робочому місці: для 1 ступеня шкідливості до 3 дБ; для 2-3 - 1-6 дБ; для 3 - більше 6 дБ.

Для захисту від електромагнітного випромінювання передбачаються наступні заходи:

1) застосування нових плазмових моніторів, LG W2271TC,

2) віддалення робочого місця не менше, ніж на 0,4-0,5 м, оскільки напруженість електричного поля зменшується при віддаленні від джерела поля,

3) встановлення раціональних режимів роботи персоналу (обмеження часу перебування),

4) раціональне розміщення в робочому приміщенні устаткування, що випромінює електромагнітну енергію.

4.2.4 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти) і установки в віконному отворі автономного кондиціонера БК-2000. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП (30 м3 на годину на одного працюючого).

Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

4.3 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Загальний опір захисного заземлення визначається за формулою:

$$R_{ззп} = \frac{R_з \cdot R_п}{R_п \cdot n \cdot \eta_з + R_з \cdot \eta_п}, \quad (4.3)$$

де Rз - опір заземлення, якими когут бать труби, опори, кути і т.п., Ом;

Rш - опір опори, яке з'єднує заземлювачі, Ом;

п - кількість заземлювачів;

ηз - коефіцієнт екранування заземлювача; приймається в межах 0,2 ÷ 0,9; ηз = 0,7

ηш - коефіцієнт екранування сполучної стійки; приймається в межах 0,1 ÷ 0,7; ηш = 0,5;

Опір заземлення визначається за формулою:

$$R_з = \frac{\rho}{2\pi \cdot l} \cdot \left(\ln \frac{2 \cdot l}{d} + \frac{1}{2} \ln \frac{4 \cdot t + l}{4 \cdot t - l} \right), \quad (4.4)$$

де ρ - питомий опір ґрунту, залежить від типу грунту, Ом·м;

для піску - 400 ÷ 700 Ом·м; приймаємо ρ = 400 Ом·м;

l - довжина заземлювача, м; для труб - 2-3 м; l = 3 м;

d - діаметр заземлювача, м; для труб - 0,03-0,05 м; d = 0,05 м;

t - відстань від середини забитого в ґрунт заземлювача до рівня землі, м; t = 2 м.

$$R_з = \frac{400}{2 \cdot 3{,}14 \cdot 3} \left(\ln \frac{2 \cdot 3}{0{,}05} + \frac{1}{2} \ln \frac{4 \cdot 2 + 3}{4 \cdot 2 - 3} \right) = 110 \text{, Ом}$$

Опір смуги, що з'єднує заземлювачі, визначається за формулою:

$$R_ш = \frac{\rho}{2\pi \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot t^1}, \quad (4.5)$$

де L - довжина смуги, що з'єднує заземлювачі (м) і приблизно дорівнює периметру будівлі: Пбуд. = 42·2 + 38·2 = 160 м; L = 160 м;

b - ширина смуги, м; b = 0,03 м;

t1 - глибина заземлення від рівня землі, м; t1 = 0,5 м.

$$R_п = \frac{400}{2 \cdot 3{,}14 \cdot 160} \cdot \ln \frac{2 \cdot 160^2}{0{,}03 \cdot 0{,}5} = 5{,}99 \text{, Ом}$$

Кількість заземлювачів захисного заземлення визначається за формулою:

$$n = \frac{2 \cdot R_з}{4 \cdot \eta_з}, \quad (4.6)$$

де 4 - допустимий загальний опір, Ом;

2 - коефіцієнт сезонності.

Визначаємо загальний опір захисного заземлення:

$$R_{ззп} = \frac{110 \cdot 5{,}99}{5{,}99 \cdot 79 \cdot 0{,}7 + 110 \cdot 0{,}5} = 1{,}7 \text{ Ом}$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: Rззп < 4 Ом.

3) При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

- іскри і дуги коротких замикань;
- перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої перевантаження та наявність перехідного опору;
- іскри при розмиканні і розмиканні ланцюгів;
- розряди статичної електрики;
- необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

Важливу увагу слід звернути на пожежну безпеку підприємства в цілому і окремих його приміщень. В приміщеннях не повинно накопичуватися сміття, непотрібний папір, мотлох та ін. речі, які не використовуються у виробничому процесі. Наявний вільний аварійний вихід за межі приміщення в разі пожежі, бути передбачені вогнегасники. Вони повинні бути в робочому стані і перевірятися згідно з нормами. У

приміщеннях повинна бути пожежна сигналізація, вогнегасник. У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки по запобіганню можливих наслідків та усуненню пожежі.

ВИСНОВКИ

У рамках дипломної роботи був реалізований програмний додаток для аналізу та візуалізації даних, розроблений згідно з архітектурою REST з використанням технології Python. Програмний додаток був успішно протестований з великим обсягом даних.

Додаток надає зрозумілий інтерфейс для роботи з методами аналізу та кластирізації даних. Використання мови Python в додатку оправдала очікування, додаток показую гарні результати стосовно швидкодії. Що стосовно візуальної частини додатку, асинхронні запроси та технологія односторінкового додатку показали гарні результати в плавності та швидкодії роботи.

Розроблений застосуноук має велику перспективу розвитку. Розробляючи більш складні методи для аналізу даних, можна реалізовувати іх в рамках даної роботи розраховуючи на швидкодію та зручність в використанні.

Ця робота дає підстави для досліджень та розробки більш складних застосунків, що можуть реалізовувати та візуалізовувати методи аналізу даних. Методи використані у цій роботі можуть стати у основу нових більш складних маніпуляцій з даними.

Для отримання результату було застосовано навички роботи з даними, знання мов програмування Python, JavaScript та чималий досвід роботи з різними видами фреймворків та бібліотек.

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в дипломній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника. Приведені рекомендації щодо організації робочого місця, а також важливу інформацію

щодо пожежної та електробезпеки. Була наведена схема, розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Hastie, T. The elements of Statistical Learning [Text] / T. Hastie, R. Tibshirani, J. Friedman – Second edition, 2006 – pp. 9 – 135.

2) Воронцов, К.В. Алгоритми кластерізації та багатовимірного шкалювання [Текс] / Воронцов К.В. - Курс лекций. МГУ, 2007. – 14с.

3) Jain, A. Data clustering: Review [Text] / A. Jain, M. Murty, P. Flynn – ACM Computing Surveys Vol. 31, 1999 –pp. 540 - 597.

4) Котов, А. Кластерізація даних [Текс] / А. Котов, Н. Красільніков. – 206. – 295с.

5) Мендаль, І.Д. Кластерний аналіз [Текст] / І.Д. Мендаль - М.: Фінанси та статистика, 1988. – 255 с.

6) Айвазян, С.А. Прикладна статистика: класифікація та зниження розмірності [Текст] / С.А. Айвязан, В.М. Бухштабер, І.С. Енюков, Л.Д. Мешалкін – М.: Фінанси та статистика, 1998. – 195 с.

7) Чубаков, І.А. Курс лекцій Data mining [Текст] / І.А. Чубаков – 2006. – 20 с.

8) Greenfeld, D.R. Two scopes [Text] / D.R. Greenfeld, A.R. Greenfeld – 1.11:Best part of the django web framework, 2007. – pp. 556.

9) Haverbeke, M. Eloquent JavaScropt [Text] / M. Haverbeke – 3rd edition 2016. –pp. 590.

10) Каскиаро, М. Шаблоны проектирования Node.js [Текст] / М. Каскиаро, Л. Маммино ; пер. с англ. А.Н. Киселева – М.: ДМК Пресс, 2017. - 396 с.

11) Weston, J. Multiclass support vector machines [Text] / J. Weston, M. Verleysen, C. Watkins - Proceedings of ESANN99, D. Facto Press, Brussels. 1999 – pp. 650.

12) Yaikhom, G. Implementing the Fuzzy c-Means Algorithm [Text] / G. Yaikhom. 2005 – pp. 15.

13) Державні санітарни норми і правила. ДСанПіН 3.3.2.007-98

«Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» Режим доступу: WWW. URL: https://zakon.rada.gov.ua/rada/show/v0007282-98

14) Державні санітарни норми України. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» Режим доступу: WWW. URL: https://zakon.rada.gov.ua/rada/show/va042282-99

15) Державні санітарни норми України. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» Режим доступу: WWW. URL: https://zakon.rada.gov.ua/rada/show/va037282-99

16) Державні санітарни норми України. ДСН 3.3.6.039-99 «Санітарні норми виробничої загальної та локальної вібрації» Режим доступу: WWW. URL: https://zakon.rada.gov.ua/rada/show/va039282-99

17) Державний стандарт України. ГОСТ 13109-97 «Электрическая энергия. Совместимость технических средств электромагнитных. Нормы качества электроэнергоснабжения общего назначения» Режим доступу: WWW. URL: http://odz.gov.ua/lean_pro/standardization/files/elektromagnitnaja_sovmestimost_2014_03_11_1.pdf

18) Державні будівельні норми України. ДБН В.2.5-28:2018 «Природнє і штучне освітлення» Режим доступу: WWW. URL: https://okna.ua/img_all/oknaua/dbn-V-2-5-28-2018-ed.pdf

19) Нормативно-правові акти з охорони праці. НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів» Режим доступу: WWW. URL: https://zakon.rada.gov.ua/laws/show/z0093-98

20) Державні будівельні норми України. ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціонування» Режим доступу: WWW. URL: https://dnaop.com/html/32609/doc-%D0%94%D0%91%D0%9D_%D0%92.2.5-67_2013

21) Державний стандарт України. ГОСТ 12.1.044-89 «ССБТ. Пожаровзрывоопасность веществ и материалов. Номенклатура показателей и

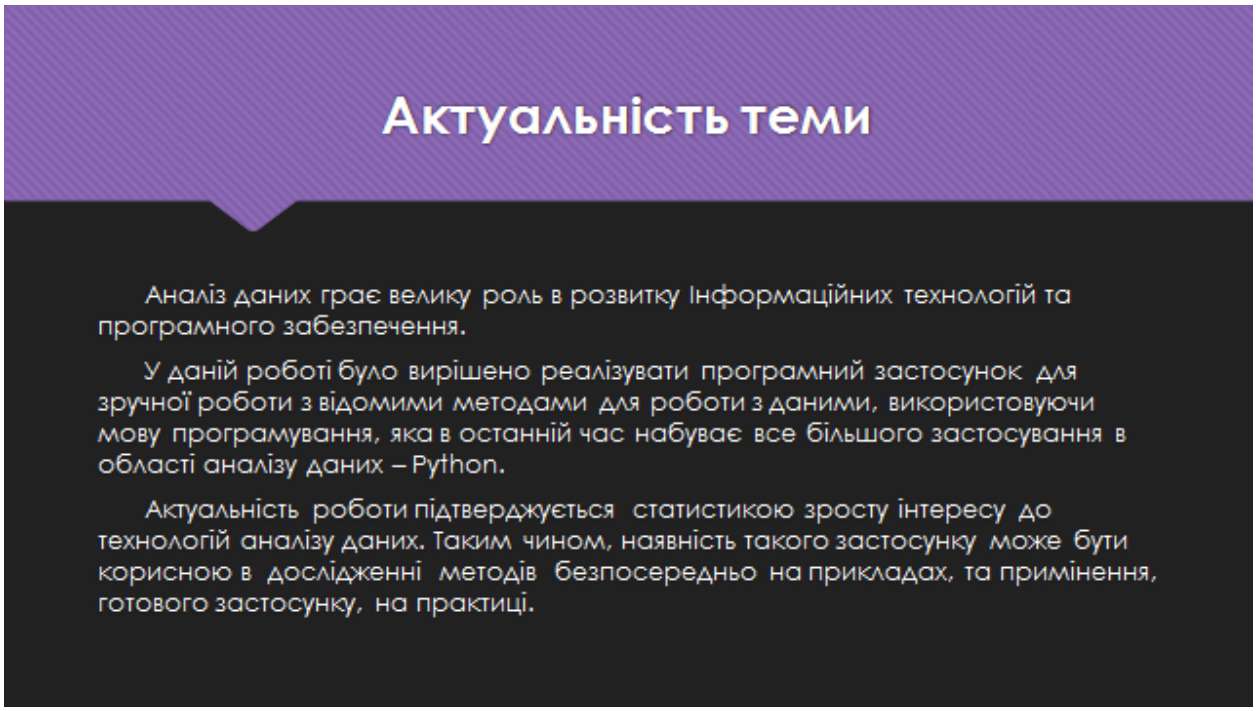
методы их определения» Режим доступу: WWW. URL: http://online.budstandart.com/ru/catalog/doc-page?id_doc=51048

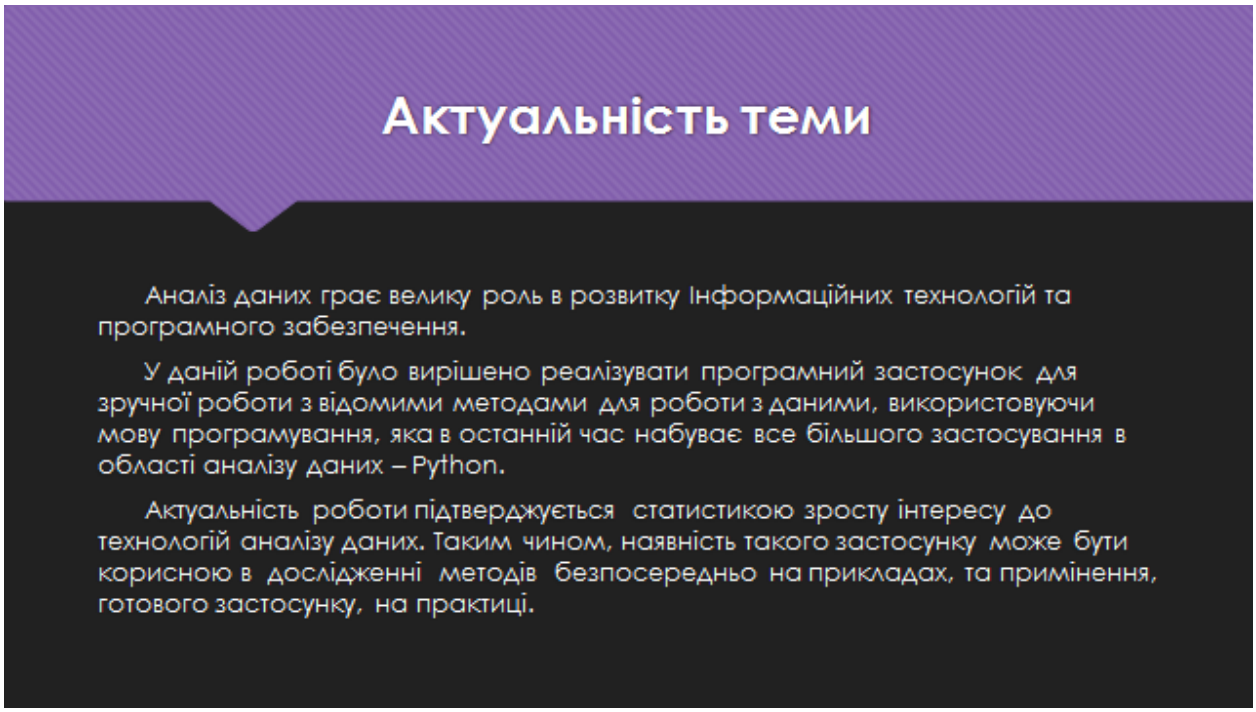
22) НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров`я працівників під час роботи з екранними пристроями Міністерство доходів і зборів України Наказ від 05.09.2013 р. № 443 «Про затвердження Примірної інструкції з охорони праці під час експлуатації електронно-обчислювальних машин» Режим доступу: WWW. URL: http://sop.zp.ua/norm_npaop_0_00-7_15-18_01_ua.php

23) Нормативно-правові акти з охорони праці. НПАОП 0.00-4.15-98 «Про розробку інструкцій з охорони праці» Режим доступу: WWW. URL: http://sop.zp.ua/norm_npaop_0_00-4_15-98_01_ru.php

ДОДАТОК А.

Електронні плакати





Постановка задачі

Розробка застосунку полягає в дослідженні та застосуванні методів аналізу даних. Саме тому що перспектива розвитку та актуіальність значана, реалізація повинна бути гнучкою, з гарними показниками швідкодії та готовий застосунок повинен бути легкий в доступі та використанні. Таким чином щоб забеспечити виконання наступниз вимог потрібно:

- розробити методи класифікацій даних;
- налаштувати серверну частину в архітектурному стилі REST;
- реалізувати клієнтський односторінковий застосунок.

Використані технології

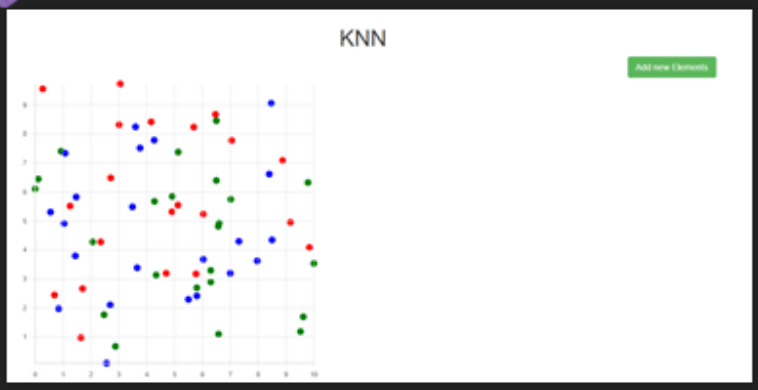
- Python
- Django REST
- JavaScript
- NodeJS
- Webpack
- React
- Redux
- Ajax

Методи аналізу даних

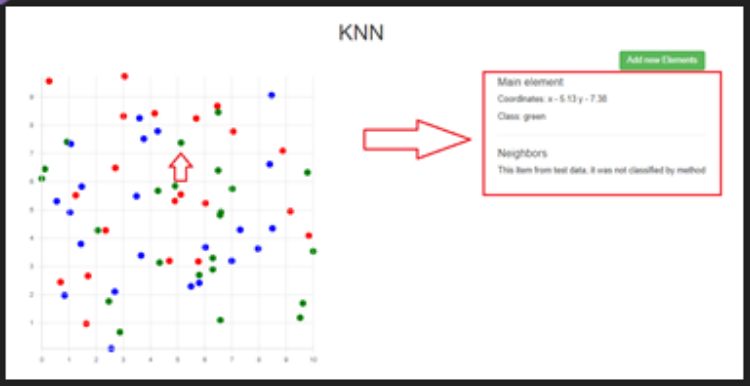
- Метод К найбліжчих сусідів (KNN)
- Метод нечіткої кластерізації С середніх (FCM)
- Метод нечтіких квадратів (МНК)

Метод К найбліжчих сусідів

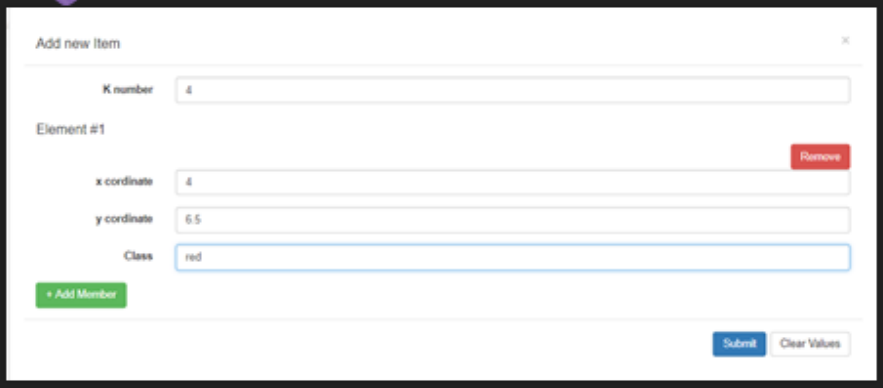
Завантаження даних



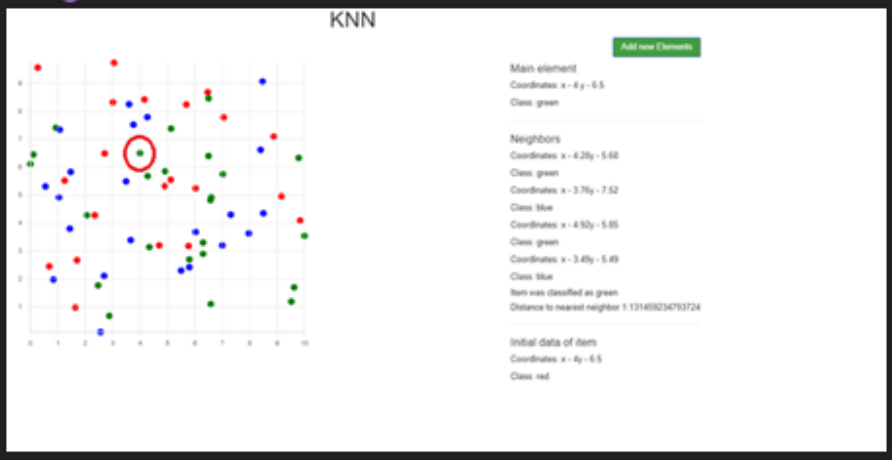
Інформація про об’єкти



Приклад додавання нового елементу

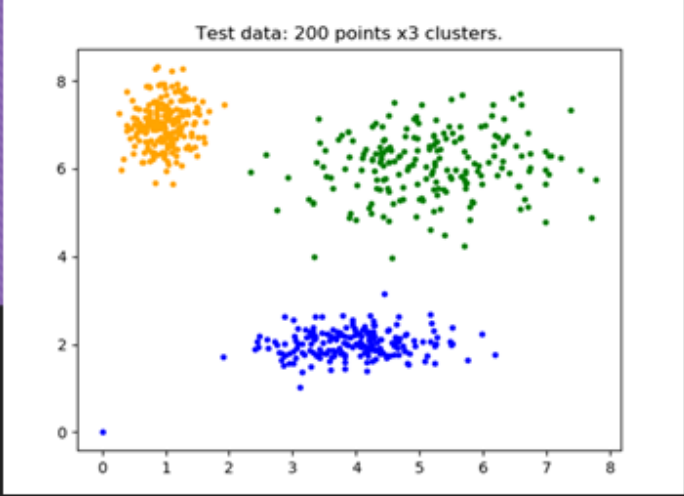


Класифікованих елемент

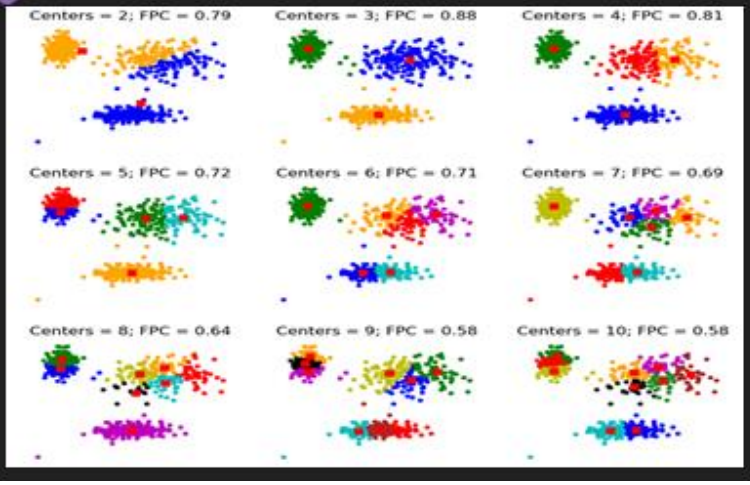


Метод нечіткої кластерізації С середніх

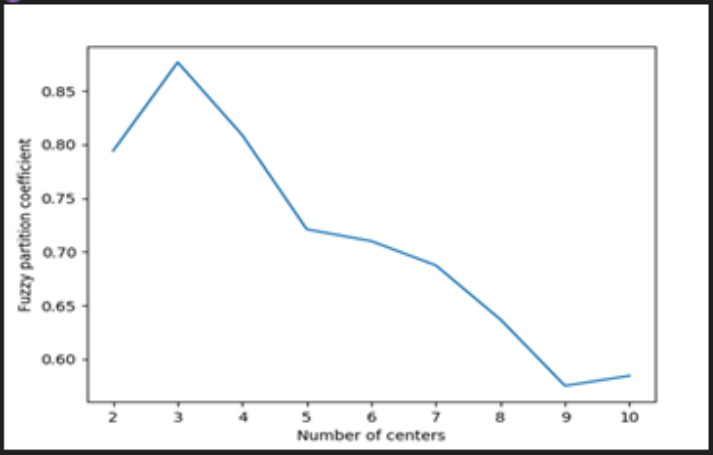
Завантаження вихідних даних



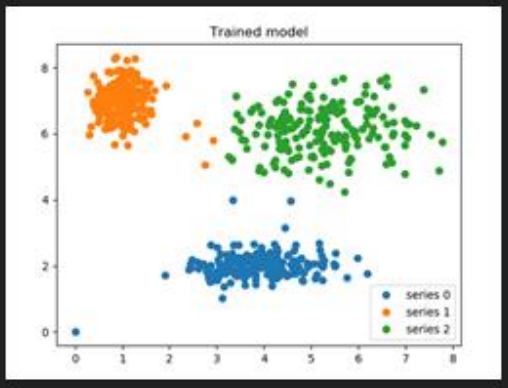
Варіанти кластирізації з різною кількістю центрів



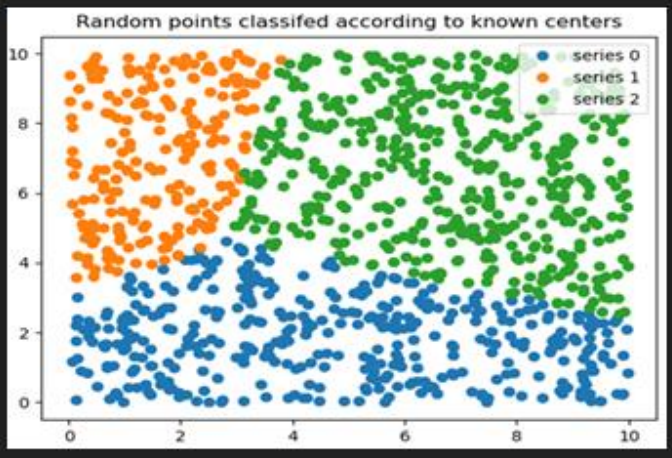
Демонстрація результатів кластеризації



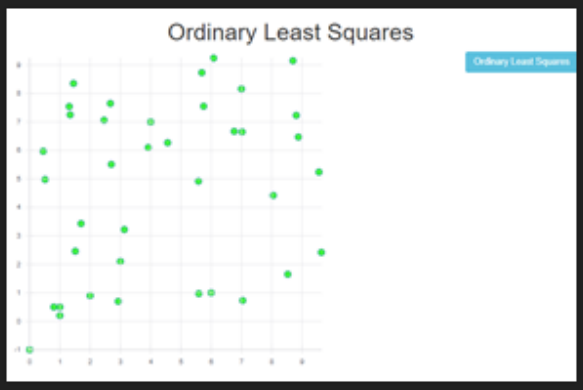
Набір даних для прогнозування



Результат прогнозування



Метод найменших квадратів
завантаження даних



Результат роботи