МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»

Завідувач кафедри

_____ /В. І. Шинкаренко/

«_____» _____ 20____р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань **12 Інформаційні технології**

Спеціальність **121 Інженерія програмного забезпечення**

Тема **Дослідження сервісів технологій Data Mining**

Theme **Researching of services of Data Mining technology**

Керівник дипломної роботи доц. _____ О. П. Іванов

Нормоконтролер доц. _____ О.С. Куроп'ятник

Студентка групи ПЗ1921 _____ Д. В. Васильєва

Student Vasylieva Daria

Дніпро – 2020

Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Факультет Комп'ютерних технологій і систем
кафедра Комп'ютерні інформаційні технології
Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ проф. Шинкаренко В.І.
(підпис)
«___»_____2020 р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС Магістр
(освітньо-кваліфікаційнийрівень)

студента групи ПЗ1921 Васильєва Дар'я Віталіївна
(номер групи) (ПІБ)

1 Тема дипломної роботи: Дослідження сервісів технології Data Mining
затверджена наказом по університету від «10» жовтня 2019 р. № 879.

2 Термін подання студентом закінченої роботи 21 грудня 2020 р.

3 Вихідні дані до дипломної роботи _____

4 Зміст пояснювальної записки (перелік питань до розробки) проведення аналізу ефективності сервісів технології Data Mining і зокрема таких як Orange та Rapid Miner.

5 Перелік демонстраційного матеріалу презентація дослідження сервісів технології Data Mining, результати експериментів, висновки, відео демонстрація роботи розробленого ПО.

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Техніко-економічні розрахунки	доц. Гненний М.В.	30.10.20	10.12.20
Охорона праці та безпека в надзвичайних ситуаціях	ст. викладач Музикін М.І.	05.11.20	16.11.20

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва розділів дипломного проекту	Термін виконання розділів проекту (роботи)	Примітка
1	Вступ	01.09.2020 – 10.09.2020	
2	Огляд літератури	10.09.2020 – 15.09.2020	
3	Постановка задачі, технічне завдання	15.09.2020 – 30.09.2020	
4	Створення тестової програми	01.10.2020 – 15.10.2020	
5	Перші тестування	15.10.2020 – 22.10.2020	
6	Аналіз результатів	30.10.2020 – 11.11.2020	
7	Розрахунок економічних показників	11.11.2020 – 14.11.2020	
8	Охорона праці	14.11.2020 – 18.11.2020	
9	Оформлення пояснювальної записки	18.11.2020 – 01.12.2020	
10	Демонстраційні матеріали	07.11.2020 – 16.12.2020	

Дата видачі завдання «10» жовтня 2019 р.

Керівник дипломного проекту _____ (підпис) Іванов О.П. (ПІБ)

Завдання прийняв до виконання _____ (підпис) Васильева Д.В. (ПІБ)

РЕФЕРАТ

Об’єкт дослідження: функції кластеризації, пошуку викидів, прогнозування та пошуку моделі лінійної регресії, що реалізовані сервісами Rapid Miner та Orange.

Предметом дослідження є результати отримані під час обробки даних різними сервісами.

Метою дослідження є визначення впливу інструменту, що використовується для обробки даних, на кінцеві результати при дослідженні даних.

Методи дослідження: порівняння часу виконання функцій сервісів, порівняння оцінок якості для отриманих результатів, аналіз графіку часової ефективності програмних засобів Rapid Miner та Orange.

Результати дослідження дозволяють зробити висновки щодо впливу обраного інструменту на кінцеві результати глибинного вивчення інформації.

Пояснювальна записка складається зі вступу, 5 розділів, висновків, бібліографічного списку та 4 додатків:

- вступ описує актуальність обраної теми, складається із 2 сторінок;
- перший розділ складається з 13 сторінок і описує сучасний стан технологій;
- другий розділ складається з 18 сторінок і містить обґрунтування експериментального методу дослідження;
- третій розділ складається з 9 сторінок, описує архітектуру системи;
- четвертий розділ складається з 30 сторінок де описані експерименти;
- п’ятий розділ розкриває питання охорони праці, складається з 8 сторінок.
- додатки містять технічне завдання й робочий проект;
- тези на XIV Міжнародну науково-практичну конференцію «Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті»;
- стаття підготовлена до друку.

Усього робота містить: таблиць – 22, рисунків – 33, бібліографія – 28 джерел. Ключові слова: Data Mining, глибинний аналіз, показники якості, кластеризація, викиди, лінійна регресія, передбачення.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАК, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

LOF – (англ. local outlier factor) локальний рівень викидів

MSE – (англ. mean squared error) середня квадратична похибка

MAE – (англ. mean absolute error) середня абсолютна похибка

RMSE – (англ. root mean squared error) корінь середньої квадратичної похибки

ПК – персональний комп’ютер

ПЗ – програмне забезпечення

ВСТУП

Стрімкий розвиток інформаційних технологій спричинює появу великих об’ємів даних, що потребують ефективної обробки. На ряду з цією проблемою постає задача здобуття нової, раніше невідомої інформації з вже існуючих даних, цей процес називається data mining, або глибинний аналіз даних. З підвищенням актуальності здобуття нових даних, з’являється все більше і більше програмних засобів, що дозволяють автоматизувати опрацювання даних. Не дивлячись на те, що сервіси Data Mining використовують одні й ті самі алгоритми, отримані результати можуть кардинально відрізнятися, що можна пояснити особливостями реалізації ПО. Тому актуальним є дослідження різниці у поведінці сервісів, з метою забезпечення ліпшої якості отриманих результатів та ефективності під час обробки даних. Для того, щоб проаналізувати різницю в поведінці сервісів і оцінити якість їх роботи, експерт повинен витратити велику кількість часу, адже на даний момент для таких цілей не існує спеціалізованого ПО.

Об’єктом дослідження є робота сервісів Rapid Miner та Orange, що реалізують функціонал для вирішення задач Data Mining. В рамках дослідження порівнюються та аналізуються функції кластеризації, знаходження викидів та розрахунку коефіцієнтів рівняння лінійної регресії.

Предметом дослідження є різниця результатів отриманих під час обробки даних різними сервісами. Порівнюються наступні критерії: час обробки інформації, ріст часу виконання функції в залежності від об’ємів даних, показники якості здобутих даних.

Метою дослідження є визначення впливу інструменту, що використовується для обробки даних, на кінцеві результати при дослідженні даних.

Методи дослідження: порівняння часу виконання функцій сервісів, порівняння оцінок якості для отриманих результатів, аналіз графіку часової ефективності програмних засобів Rapid Miner та Orange.

Результати та їх новизна: дослідження робить внесок у аналіз ефективності існуючих сервісів, що вирішують задачі Data Mining. Результати дослідження

дозволяють зробити висновки щодо впливу обраного інструменту на кінцеві результати глибинного вивчення інформації.

Практичне значення одержаних результатів полягає у використанні розробленої програми для вибору кращого сервісу Data Mining для подальшого аналізу даних.

Апробація результатів дослідження та публікації.

Основні положення магістерської роботи доповідалися та були схвалені на засіданнях кафедри КІТ ДНУЗТ (03.12.2019 р., 12.03.2020 р., 07.10.2020 р.).

За результатами роботи опубліковано наукову працю – тези доповідей на науковій конференції.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ДОСЛІДЖЕННЯ СЕРВІСІВ DATA MINING

1.1 Розкриття теми Data mining

Data Mining [1] (або глибинний чи інтелектуальний аналіз даних) – термін який використовується для опису методології і процесу знаходження в великих масивах даних раніше невідомих, нетривіальних, корисних та доступних для інтерпретації знань, що необхідні для прийняття рішень в різних сферах людської діяльності. Під поняттям Data Mining розуміють сукупність великої кількості різних методів знаходження знань. Задачі, що вирішуються методами Data Mining можна умовно поділити на декілька видів:

- класифікація;
- регресія;
- кластеризація;
- асоціація;
- послідовні шаблони;
- аналіз викидів.

Data Mining носить мультидисципліонарний характер, оскільки включає в себе елементи числових методів, математичної статистики та теорії ймовірностей, теорії інформації і математичної логіки, штучного інтелекту та машинного навчання.

Не дивлячись на те, що область Data Mining не є новою (оскільки перше згадування датується 1989 р. [2]), все ще існує багато невирішених питань, які виникають при обробці даних. Для того, щоб отримати дійсно якісні знання використовуючи складні алгоритми інтелектуального аналізу потрібно дати відповідь на наступні запитання:

- На скільки якісні здобуті знання?
- Чи завжди один й той самий метод дає однакові знання?
- Чи потрібні різні інструменти для різних сфер застосування?
- Які фактори впливають на продуктивність інструменту?
- Які когнітивні фактори впливають на результат?

На рисунку 1.1 зображена схема розподілення основних проблем Data Mining

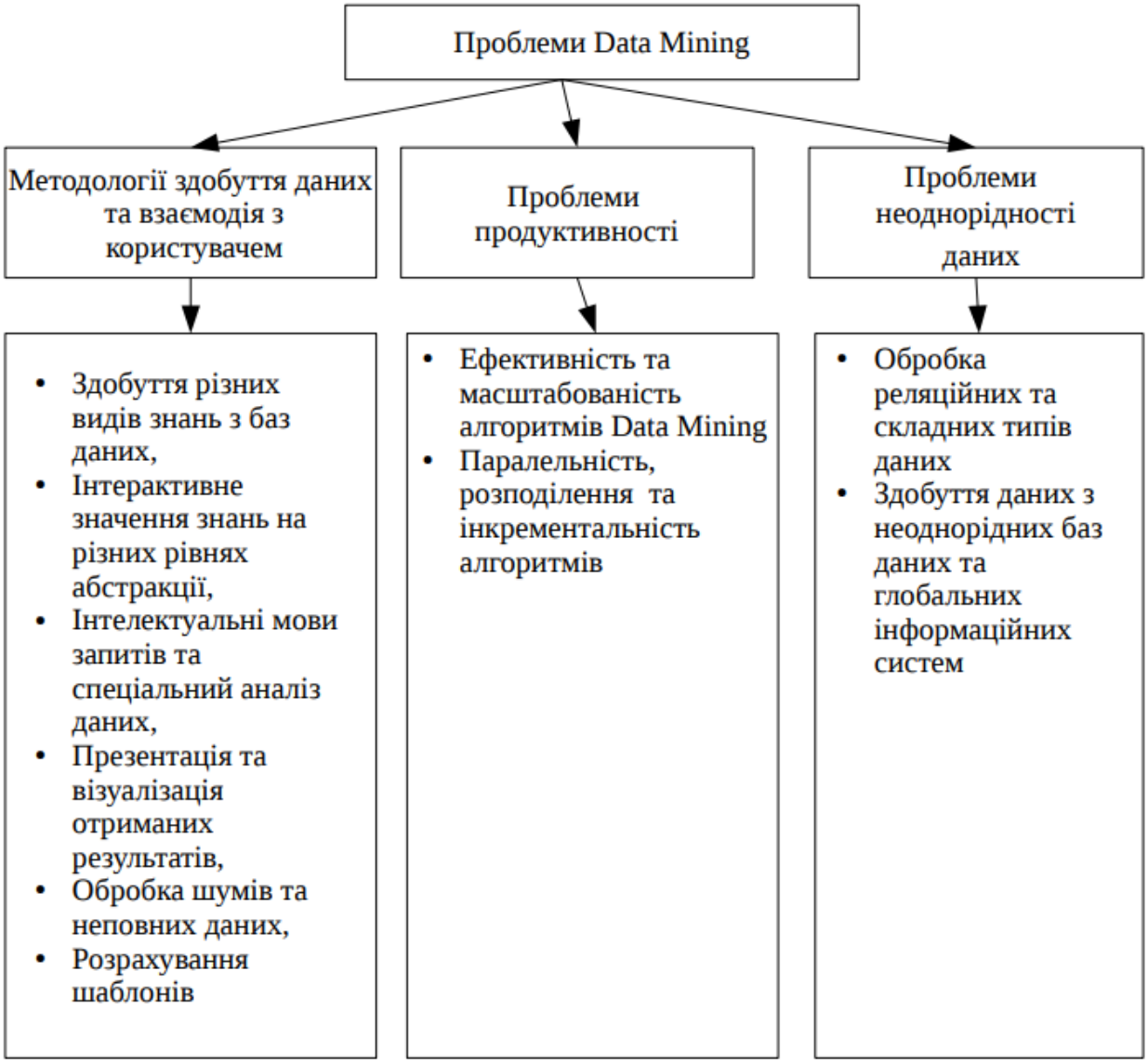


Рисунок 1.1 – схема розподілення проблем Data Mining

Процес знаходження корисних даних можна звести до чотирьох основних етапів:

– визначення проблеми, чітка постановка цілей і вимог. На цьому етапі необхідно визначити проблему, що потребує вирішення;
– сбір і підготовка до аналізу. До цього етапу також можна віднести аналіз отриманих даних, видалення шумів, знаходження шаблонів;
– побудова моделі аналізу і оцінки. Етап включає аналіз співвідношення отриманої моделі і проблеми яку вона має вирішити;
– використання отриманих знань на практиці.

Процес знаходження даних є безкінечним. Схема процесу Data Mining зображена на рисунку 1.2.

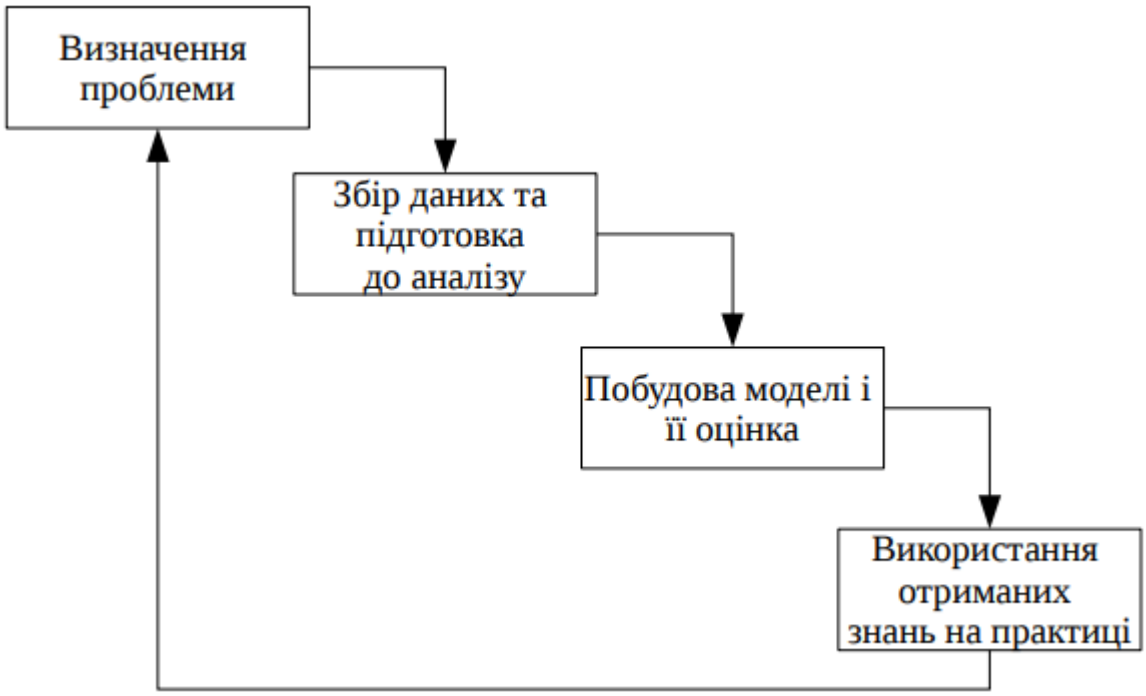


Рисунок 1.2 – Схематичне зображення процесу здобуття корисних даних

У даній дипломній роботі підіймається питання стосовно різниці в здобутих даних, що були отримані за допомогою різних сервісів. При здобутті даних дотримано описаного вище процесу знаходження даних.

1.2 Задачі Data Mining

У главі 1.1 були згадані основні задачі, що вирішуються методами Data Mining. В цьому розділі більш детально описані ті функції, що найбільш широко застосовуються в сучасних дослідженнях.

1.2.1 Регресія

Регресія – це метод, що використовується для моделювання та аналізу відношення між змінними. Також для того, щоб зрозуміти, як ці змінні разом впливають на отримання певного результату. Є декілька видів регресії, серед них:

- лінійна регресія;
- поліноміальна регресія;
- регресія за методом «гребінь»;
- регресія за методом «ласо».

Вид регресії для аналізу даних обирається в залежності від залежностей, що присутні. Одним із базових алгоритмів для вирішення задач у багатьох сферах є лінійна регресія.

Лінійна регресія відноситься до виду регресійної моделі, що складаються з взаємопов'язаних змінних. Існує парна (проста) та множинна лінійна регресія. Останній вид лінійної регресії передбачає встановлення лінійної залежності між множиною вхідних незалежних змінних та однієї вихідної залежної змінної. Така модель залишається лінійною з тієї причини, що вихідні дані є лінійною комбінацією вхідних змінних. Варто зазначити, що такий вид аналізу підходить не для всіх наборів даних, для того щоб підібрати правильний спосіб знаходження регресії, можна скористатися візуальним аналізом досліджуваного набору даних. Також варто зазначити деякі важливі факти, щодо лінійної регресії:

- вона легко моделюється і є особливо корисною при створенні не дуже складної залежності, а також при невеликій кількості даних;
- позначення є інтуїтивно зрозумілими;
- є чутливою до викидів.

Функція лінійної регресії обчислюється за формулою 1.1:

$$f(x, b) = b_0 + b_1 x_1 + b_2 x_2 + \cdots b_k x_k\,, \tag{1.1}$$

де b_j – коефіцієнти, параметри лінійної регресії,x_j – фактори моделі, k – кількість факторів моделі.

Після того, як рівняння лінійної регресії було знайдено, постає питання оцінки якості отриманої моделі. Для цього підраховують коефіцієнт детермінації R^2, середню квадратичну похибку MSE, середню абсолютну похибку MAE, та інші варіації знаходження помилок.

1.2.2 Передбачення

Передбачення (англ. prediction) в Data Mining – різновид аналітичної задачі де існуючі данні використовуються для передбачення невідомих даних. Зазвичай, отримання нових даних базується на відношенні об’єктів в середині існуючих.

Більшість методів передбачення використовує на математичні моделі, серед них:

- прості математичні моделі, такі як регресія;
- нелінійна статистика, наприклад: степеневі ряди;
- нейронні мережі.
- Процес створення моделі для прогнозувань складається з наступних кроків:
- очистити данні від викидів та опрацювати відсутні данні;
- попередньо обробити дані у формі, що є придатною для обраного алгоритму моделювання;
- вказати підмножину даних, що буде використана для навчання моделі;
- натренувати або оцінити параметри моделі використовуючи тестові данні;
- провести тести продуктивності моделі, перевірити адекватність моделі;
- перевірити точність передбачень на даних, які не використовуються для калібрування моделі.

1.2.3 Кластеризація

Задача кластеризації полягає у розбитті заданої групи об’єктів, на підгрупи, які ще називають кластерами. Кожний кластер повинен складатися з об’єктів, що схожі між собою. Об’єкти, що знаходяться в різних кластерах повинні істотно відрізнятися один від одного. На рисунку 1.3 зображений приклад кластеризації об’єктів для відомого набору даних «Iris» [3], у якості показників для яких розраховані кластери, були обрані ширина та висота чашолистка. Кожний кластер позначений своїм кольором.

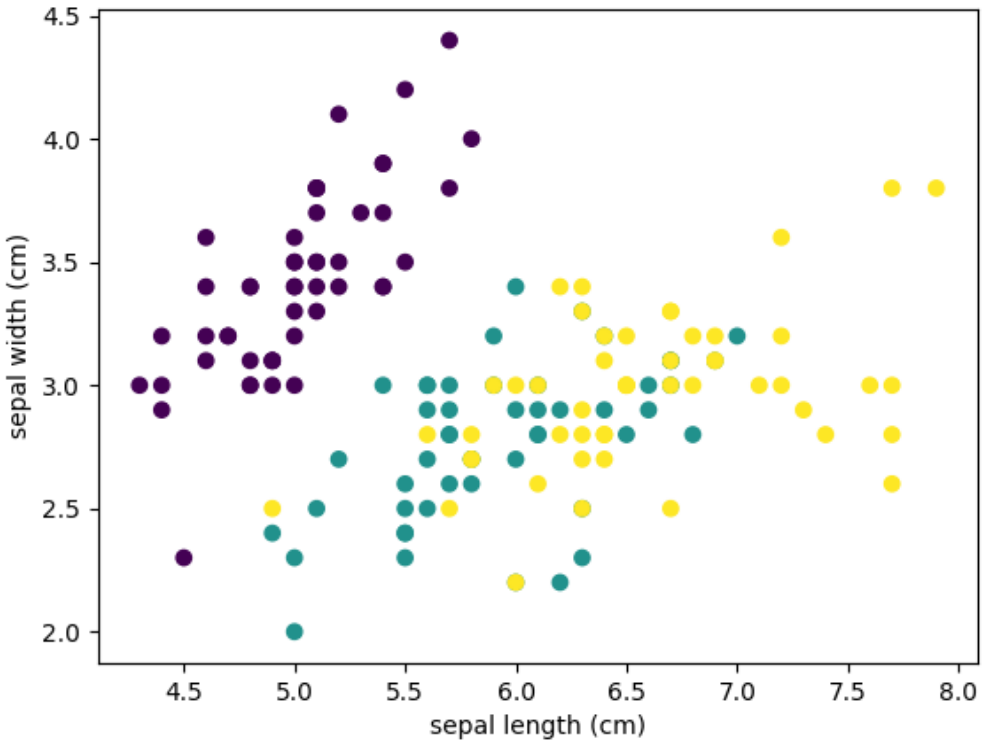


Рисунок 1.3 – Кластеризація ірисів

Методів кластеризації існує безліч, оскільки поняття кластера в різних алгоритмах має різні властивості. Виділяють наступні моделі кластеризації:

– моделі зв’язності;
– статистичні моделі;
– центроїдні моделі;
– моделі засновані на щільності об’єктів у кластері;
– групові моделі;
– графові моделі;
– нейронні моделі.

Розглянемо більш детально метод кластеризації k-середніх, що належить до центроїдної моделі.

Кластеризація методом k-середніх полягає у розділенні n спостережень на k кластерів, таким чином щоб кожне спостереження належало до кластера з найближчим до нього середнім значенням. Даний метод базується на мінімізації суми квадратів відстаней між кожним спостереженням та центром його кластера.

1.2.4 Знаходження викидів

Знаходження викидів в даних – аналітична задача, що полягає в виявленні рідкісних даних, подій або спостережень, які суттєво відрізняються від більшості даних. До викидів відносять: аномалії, відхилення, винятки або шуми в даних. На рисунку 1.4 графічно зображений приклад викидів в даних, аномальні екземпляри позначені червоним кольором.

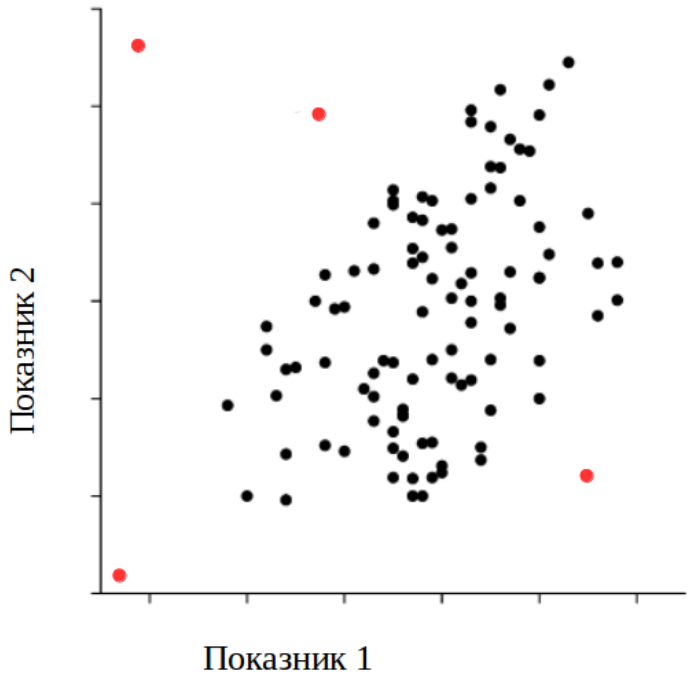


Рисунок 1.4 – Викиди в наборі даних

Алгоритми для виявлення аномалій в залежності від типу їх навчання можна умовно поділити на три групи: автоматичне навчання, напівавтоматичне навчання і контрольоване навчання. Вид навчання залежить від типу даних. Перечислимо найбільш поширені види алгоритмів для знаходження викидів:

- k-найближчих сусідів (k-NN);
- Локальний рівень викидів (LOF);
- ізоляційний ліс (isolation forest);
- штучні нейронні мережі;
- SVM;
- Elliptic envelope і статичні методи.

Розглянемо більш детально алгоритм локального рівня викидів. Даний метод базується на концепції локальної щільності, де локальність задається k-найближчими сусідами, відстань до яких використовують для оцінки щільності. Порівнюючи локальну щільність об’єкту з локальною щільністю його сусідів, можна визначити області з аналогічною щільністю і точки, що мають суттєво меншу щільність, ніж їх сусіди. Дані точки називають викидами. Локальна щільність визначається відстанню до сусідніх точок.

1.3 Огляд та аналіз актуальності сервісів Data Mining

Стрімкий розвиток інформаційних технологій, зокрема в області інтелектуальної обробки даних, дозволив багатьом організаціям збирати великі масиви даних, які необхідно аналізувати. Об’єми даних стають настільки великими. що ресурсів експертів не вистачає. Дана проблема послугувала створенню сервісів для обробки даних, що мають в своєму арсеналі автоматизовані системи вирішення сучасних задач Data Mining. Таким чином основною метою сервісів Data Mining є мінімізація зусиль експерта, що приймає рішення в процесі аналізу даних і налаштування алгоритмів аналізу. Більшість систем інтелектуального аналізу даних дозволяє не тільки вирішувати класичні задачі з прийняття рішень, але й мають змогу виявляти причинно-наслідкові зв’язки, скриті закономірності в системі яка аналізується. Варто зазначити, що з збільшенням даних постає питання ефективності їх обробки алгоритмами інтелектуального аналізу.

На ринку представлена велике різноманіття інструментів для здобуття даних. Серед них як універсальні додатки, що слугують допоміжним засобом для аналітиків, так і вузько спеціалізовані програми для аналізу певного спектру даних (наприклад програми для аналізу продаж інтернет-магазину).

Серед безкоштовних універсальних систем Data Mining варто виділити наступні:

- Rapid Miner;
- Orange;
- Weka;
- Sisense;

- Revolution;
- Qlik;
- SAS Data Mining;
- Teradata;
- InetSoft;
- Dundas.

Серед вище зазначених сервісів лише Rapid Miner і Orange мають API, або бібліотеку для доступної роботи з їх алгоритмами. Такий функціонал надає можливість стороннім програмам використовувати їх реалізації алгоритмів глибинного аналізу, досліджувати, або навіть порівнювати їх між собою. Розглянемо ці сервіси більш детально у наступних параграфах.

1.3.1 Огляд сервісу Rapid Miner та його можливостей

Rapid Miner – програмне забезпечення, розроблене однойменною компанією, що забезпечує інтеграційну середу для підготовки даних, машинного навчання, глибокого вивчення даних, здобуття даних з тексту і прогнозування даних. Активно використовується в різних сферах, зокрема: бізнес, навчання, освіта, наука і розробка додатків. Програма автоматизує всі кроки процесу машинного навчання, включаючи підготовку даних, візуалізацію результатів, перевірку моделей на адекватність та оптимізацію роботі. Мова розробки – Java. Процес користування інтуїтивно зрозумілий і не потребує написання коду. Для початку роботи з аналізу даних потрібно лише створити процес, і перемістити необхідні віджети на робоче вікно, після чого запустити процес.

Система складається з декількох компонентів, частина з яких може бути використана як самостійне рішення. Для більш повного розуміння системи розглянемо її компоненти в таблиці 1.1.

Таблиця 1.1 – Компоненти Rapid Miner

Компонент	Опис
Rapid Miner Studio	Програма для побудови та редагування аналітичних процесів. Rapid Miner Studio та Rapid Miner Server підключаються та взаємодіють один з одним використовуючи стандартні протоколи. Для кожного з об’єктів Rapid Miner Server можна підключити один або декілька клієнтів.
Rapid Miner Server	Служить для запуску процесів створених у Rapid Miner Studio.
Rapid Miner Job Agent	Агент відповідальний за керування процесами на вузлі, де він був встановлений. Після того, як агент був запущений, він починає по черзі виконувати Job Containers.
Rapid Miner Job Container	Компонент, що є відповідалструктури в іоносфері. "Погані" повернення - це ті, які цього не роблять; їх сигнали ьним за виконання процесу. Життєвий цикл компонента контролює Job Agent.
Rapid Miner Server repository	Даний компонент відповідає за зберігання процесів і даних. Всі дані зберігаються в домашній директорії Rapid Miner.
Data sources	Індивідуальні дані користувача, щ обули створені під час використання програми.

Під час роботи інтеграції Rapid Miner з власною програмою відбувається взаємодія з такими компоненти системи як Rapid Miner Studio і Data sources. Для взаємодії процесів Rapid Miner з програмою існує Python API. Дане API являє собою бібліотеку з відкритим кодом, написану на мові Python, за допомогою якої можна запускати попередньо створені процеси і отримувати результати їх виконання.

На рисунку 1.4 представлений скріншот програми з робочим процесом.

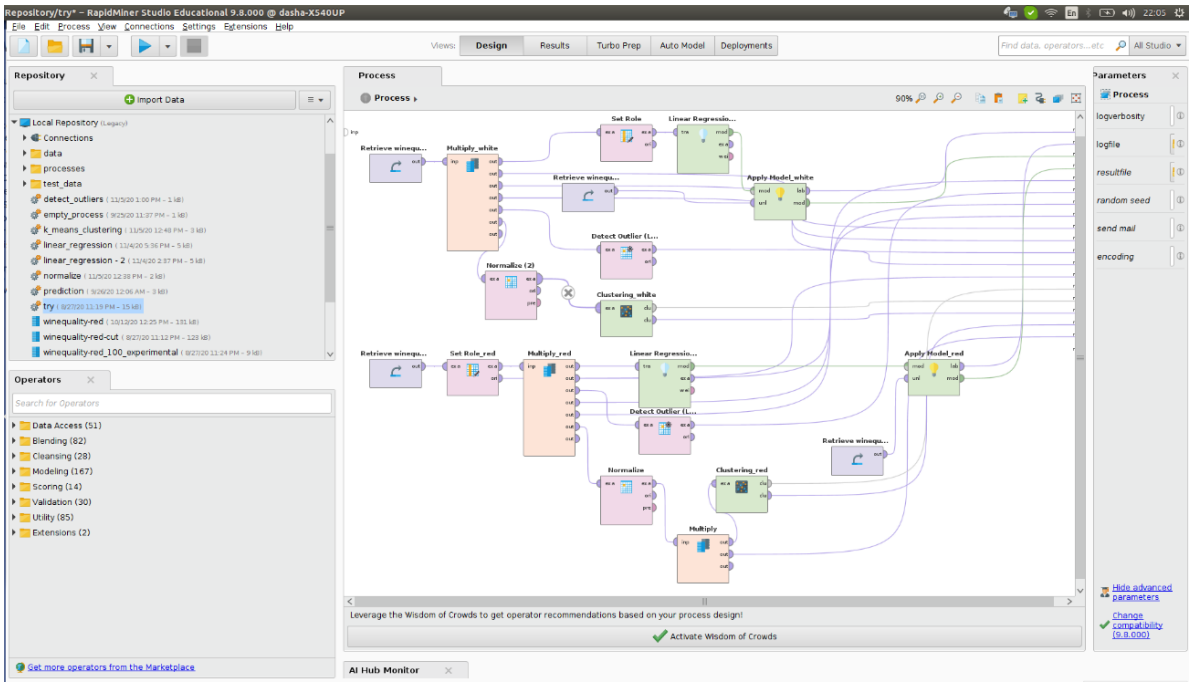


Рисунок 1.4 – Приклад створення робочого процесу в Rapid Miner

Підсумовуючи, виділимо позивні та негативні особливості даного сервісу. До переваг можна віднести наступні:

– наявність великої кількості процедур, особливо це стосується області відбору атрибутів і знаходження аномалій;
– повний набор інструментів для оцінки моделей, що використовує кросс-валідацію;
– більше ніж 1500 методів для інтеграції даних, трансформації даних, аналізу, візуалізації і т.д.;
– підтримує інтеграцію з іншими сервісами;
– дуже гнучкий та масштабований, може опрацьовувати великі дані (є можливим через інтеграцію з Hadoop).
– До недоліків програми Rapid Miner можна віднести наступні пункти:
– обмежені можливості розділення для даних на тестові дані і дані для навчання;
– для виконання деяких задач потрібно знання sql.

1.3.2 Сервіс Orange. Огляд функціоналу

Orange – компонентно-орієнтований програмний пакет для візуалізації даних, машинного навчання, аналізу даних та їх інтелектуальної обробки. Компоненти Orange називаються віджетами і варіюються від простих візуалізацій даних, обрання підмножин і попередньої обробки даних до емпіричних оцінок алгоритмів навчання і моделей прогнозування. Візуальне програмування розроблено через інтерфейс в якому робочі процеси створені за допомогою зв'язаних віджетів. Для досвідчених користувачів є можливість використовувати Orange у якості Python бібліотеки для маніпуляцій даними. Мова розробки – C++, з використанням обгортки на Python. Для більш повного розуміння роботи з програмою на рисунку 1.5 зображений скріншот робочого процесу.

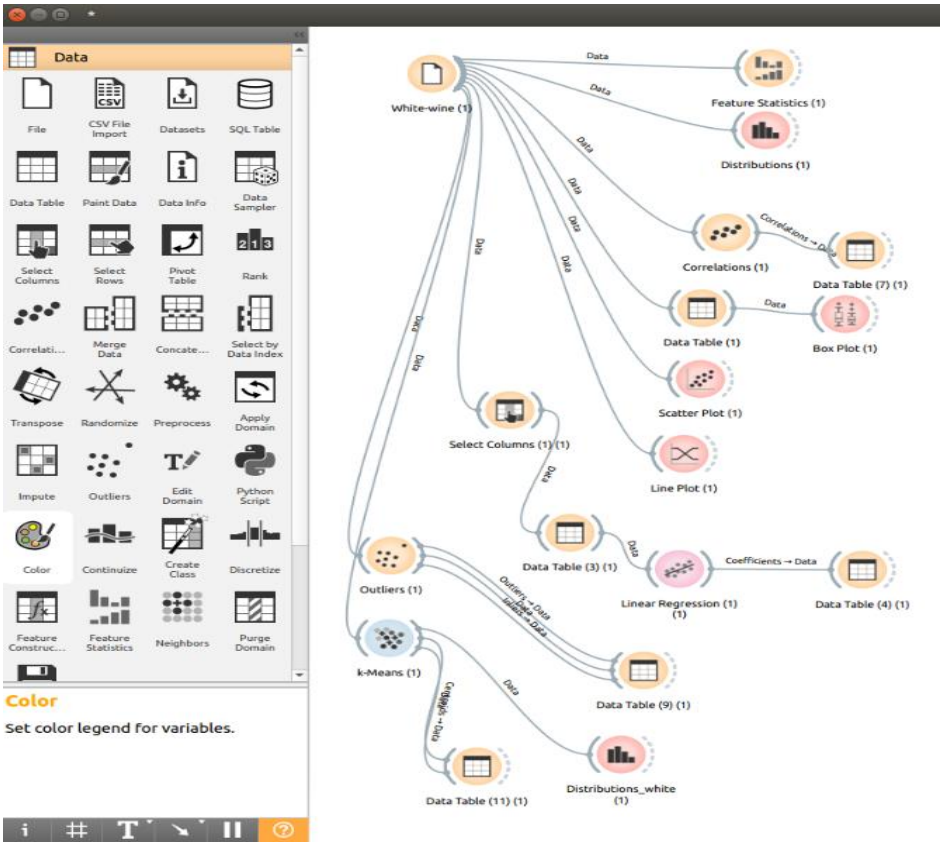


Рисунок 1.5 – Приклад створення робочого процесу в Orange

Виділимо переваги даного сервісу:

– швидка обробка даних, оскільки всі складні операції реалізовані на мові C++;
– простий у використанні;

– швидкий для вивчення, оскільки має добру документацію і достатню кількість онлайн курсів.

До недоліків програми Orange можна віднести наступні пункти:

– не має інтеграції з сервісами Big Data, що ставить під сумнів його ефективність при роботі з великими об’ємами даних;
– не має інтеграції з іншими сервісами Data Mining.

1.4 Опис проблеми та актуальність дослідження

На даний момент ринок Data Mining пропонує безліч інструментів для автоматизації вирішення задач. Існує велика кількість програмних рішень, що різняться між собою за ціною, призначенням та рівнем автоматизації глибинного аналізу даних. Сьогодні найбільш поширеними є ті сервіси Data Mining, що пропонують універсальну середу розробки власних процесів для інтелектуального аналізу даних. Актуальність їх дослідження полягає у тому що, при використанні одних і тих самих алгоритмів обробки даних різними сервісами, отримані результати можуть суттєво відрізнятися, що в подальшому може значно вплинути на кінцеві результати при вирішенні задачі. Швидкість обробки даних також може відрізнятися, даний аспект при роботі з великими даними є вирішальним. Причиною різниці у поведінці сервісів можуть бути модифікації алгоритмів, попередня обробка даних, неефективна робота з ресурсами, що сповільнює дію програми і т.д.

1.5 Огляд програмних аналогів

Програмний продукт розроблений у рамках дипломної роботи для порівняння і дослідження сервісів Data Mining не має близьких аналогів, тому спробуємо розглянути його віддалені аналоги – програми для аналізу сайтів.

Netpeak Spider – програмне рішення для регулярного аудиту, швидкого пошуку помилок і системного аналізу сайту. Добре підходить для аналізу великих сайтів (більше 100 000 сторінок). Можливий експорт отриманих даних про помилки в наступних форматах: CSV, PDF, Google Drive.

PR-CY – онлайн сервіс, що дозволяє перевірити сайт на велику кількість SEO параметрів сайтів. Функціонал сервісу є дуже обширним, також присутній аудит сайтів на помилки, оптимізацію.

Site Analyzer – програмне рішення, що дозволяє сканувати сайти і перевіряти їх основні технічні і SEO-параметри на предмет помилок і ефективно їх виправляти.

Xenu – програма для аудиту, що інсталюється на комп'ютер. Використовуючи URL ресурсу вона проходить по всім посиланням сайту. Аналізує биті посилання, статус-код сторінки, об'єм тексту на сторінці, рівень вкладеності, кодування.

Недоліками перелічених додатків є відсутність можливості порівняти 2 або більше сайтів між собою. Цей момент буде враховано при розробці програми для порівняння сервісів Data Mining.

Висновки до першого розділу

У даному розділі була розглянута тема глибинного аналізу даних, виконано короткий огляд основних задач Data Mining та сервісів для автоматизації їх виконання.

Аналіз предметної сфери показав відсутність інструментів для оцінки програмних рішень інтелектуального аналізу даних. На даний момент такі оцінки можуть бути виконані лише експертом-аналітиком. станом на сьогодні близьких аналогів програми для порівняння сервісів не існує. Таким чином, завдання з розробки програмної системи «Data Mining Service Analyzer» є актуальним.

2 ПОРІВНЯЛЬНІ ОЦІНКИ ФУНКЦІОНАЛУ СЕРВІСІВ DATA MINING

Оскільки метою дипломної роботи є створення програмного продукту для порівняння сервісів глибинного аналізу даних, однією з супутніх проблем є підбір ефективних метрик та параметрів для їх оцінки. Розглядаючи характеристики аналітичних сервісів, варто зазначити, що важливою характеристикою є ефективність виконання поставлених сервісу задач. З огляду на це, основні зусилля були спрямовані на оцінку ефективності виконання кожного з чотирьох алгоритмів поведінку яких порівнює сервіс «Data Mining Service Analyzer», а саме: знаходження аномалій, пошук коефіцієнтів лінійної регресії, кластеризація і передбачення. Для кожної з задач були підібрані індивідуальні метрики оцінки якості отриманих результатів.

2.1 Оцінка виявлених аномалій. Метрика AUC-ROC

Для оцінки виявлених аномалій використовується метрика AUC-ROC [4]. Цей підхід має суттєвий недолік, оскільки для його роботи потрібно попередньо знати які з експериментальних даних є аномальними. Згаданий вище факт означає, що таким чином можна оцінити ефективність роботи сервісів лише для тестових даних, де викиди були попередньо знайдені експертом.

Метрика AUC-ROC базується на побудові ROC-кривої [5] (англ. receiver operating characteristic). ROC-крива представляє собою графік для оцінки якості бінарної класифікації. Крива відображає співвідношення між часткою об’єктів від загальної кількості носіїв ознаки що були вірно класифіковані до частки об’єктів від загальної кількості об’єктів, що були не вірно класифіковані. Для того, щоб отримати кількісну інтерпретацію ROC використовують показник AUC (англ. area under ROC curve). Вище згаданий показник це площа обмежена ROC-кривою і віссю частки помилкових позитивних класифікацій.

На рисунку 2.1 зображений приклад побудови ROC-кривої, зазначимо, що для обох осей діапазон значень складає від 0 до 1.

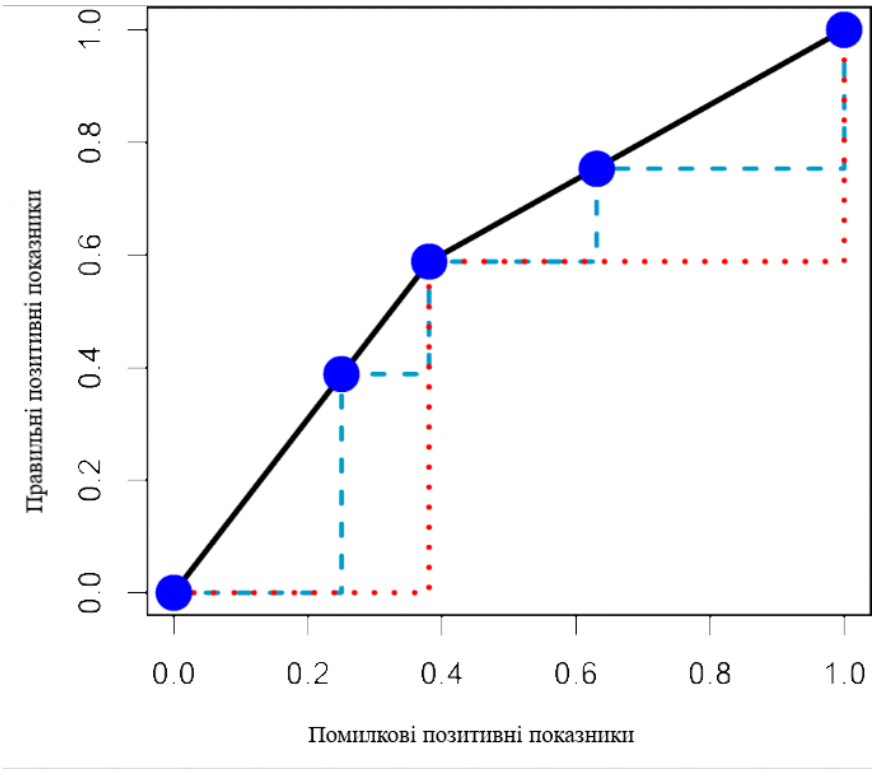


Рисунок 2.1 – Приклад побудови ROC-кривої

Метрика AUC-ROC розраховується за наступною формулою 2.2:

$$\mathrm{AUCROC} = \frac{1 + \mathrm{TPR} - \mathrm{FPR}}{2}, \tag{2.2}$$

де показники TPR і FPR обчислюються за формулами 2.3 і 2.4 відповідно:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{2.3}$$

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \tag{2.3}$$

$$\mathrm{FRP} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}. \tag{2.4}$$

Для використання вище приведених формул необхідно попередньо обчислити наступні показники:

1. TP – кількість нормальних експериментів, які алгоритм вважає не аномальними.

2. FP – кількість аномальних експериментів, які алгоритм вважає нормальними
3. TN – кількість аномальних експериментів, які алгоритм вважає викидами.
4. FN – кількість нормальних експериментів, які алгоритм вважає викидами.

Якщо при підрахунку показника AOC було отримано 1, класифікація вважається ідеальною, у випадку, якщо під час підрахунків було отримано значення 0.5, обраний метод класифікації вважається непридатним, оскільки це відповідає випадковому вгадуванню.

У якості реалізації алгоритму підрахунку метрики AOC-ROC була використана відповідна функція бібліотеки sklearn.

Варто додати, що для порівняння результатів отриманих різними сервісами була розроблена функція їх оцінки за часом виконання.

2.2 Критерії оцінки якості регресійної моделі

Якістю регресійної моделі називається адекватність побудованої моделі відповідно до початкових даних. Вона перевіряється на основі аналізу залишків регресії. Аналіз залишків дозволяє зрозуміти наскільки добре була підібрана сама модель і наскільки добре підібраний метод оцінки коефіцієнтів.

Залишки регресії – це різниці між значеннями, що досліджуються і значеннями, передбаченими моделлю регресії, що вивчається. Чим краще модель узгоджена з даними, тим менше величина залишків. При розробці програмного продукту «Data Mining Service Analyzer» були використані наступні метрики для оцінки отриманої моделі регресії [6]:

1. коефіцієнт детермінації
2. середня квадратична помилка
3. корінь середньої квадратичної помилки
4. середня абсолютна помилка

В наступних параграфах розглянемо переваги і недоліки кожної з них окремо.

2.2.1 Коефіцієнт детермінацій

Коефіцієнт детермінацій R^2 [7] характеризує частку дисперсії результативної ознаки, що пояснюється регресією, в загальній дисперсії. Даний коефіцієнт

розраховується для оцінки якості отриманого рівняння регресії. Для допустимих моделей вважається, що коефіцієнт детермінації повинен бути не менше 0.5. Моделі для яких коефіцієнт детермінації вище 0.8 вважаються добре підібраними. У разі, якщо $R^2 = 1$, дані мають функціональну залежність. R^2 розраховується за формулою 2.5.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \acute{y})^2}, \tag{2.5}$$

де n – кількість експериментів,

y_i – значення досліджуваної змінної,

$\hat{y}$ – значення досліджуваної змінної, отримані через рівняння лінійної регресії,

$\acute{y}$ – середнє значення досліджуваної змінної, що розраховується за формулою 2.6

$$\acute{y} = \sum_{i=y}^{n} \frac{y_i}{n}. \tag{2.6}$$

Переваги даного показника порівняно з іншими метриками полягають у тому що його значення не є масштабованим, оскільки лежить у діапазоні $(-\infty; 1]$.

2.2.2 Середня квадратична помилка

Середня квадратична помилка MSE [8] (англ. mean squared error) – величина, що характеризує відхилення окремого показника від його справжнього значення. Формула за якою розраховується показник наведена нижче (2.7).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{2.7}$$

На рисунку 2.8 зображено схематичний вигляд підрахунку MSE, де середній квадратичній помилці відповідає середня відстань між залишками і точками що лежать на лінії регресії піднесена до квадрату.

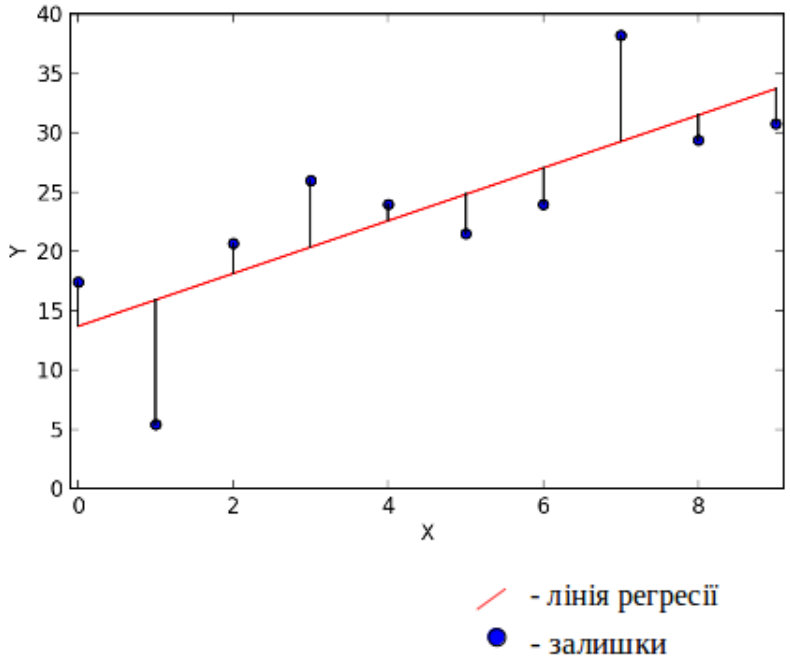


Рисунок 2.8 – Підрахунок MSE

Як вже було зазначено вище, MSE вимірює середньоквадратичну помилку отриманих прогнозів. Для кожної точки вимірюється квадратна різниця між значенням, що було отримане в результаті підрахунку рівняння регресії і актуальними даними, після чого знаходиться їх середнє значення. Чим показник вищий, тим гірше підібрана модель, для ідеальної моделі значення близьке до 0.

Недоліками даної метрики є її чутливість до окремих невдалих прогнозів, оскільки, у разі якщо один із експериментів виявився з великою похибкою, то різниця між значеннями, піднесена до квадрату, зробить великий вплив на результуючий показник. Таким чином, дуже важливо прибирати шуми з даних перед розрахунком даного показника.

2.2.4 Середня абсолютна помилка

Середня абсолютна помилка MAE [10] (англ. mean absolute error) – показник, що розраховується як середнє значення між цільовими значеннями і прогнозами. Формула для розрахунку показника наступна 2.10:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \qquad (2.10)$$

Даний коефіцієнт має два недоліки, що проявляються лише у випадку аналізу прогнозованої точності моделей:

- значення важко інтерпретувати, оскільки його діапазон лежить в інтервалі [0;+∞);
- при порівнянні моделей ефективний лише у тому разі, якщо моделі порівнюються за одним рядом даних.

2.2.5 Застосування метрик оцінки лінійної регресії в програмному продукті

У програмному продуктів «Data Mining Service Analyzer» реалізована можливість оцінки моделі регресії, що була отримана в наслідок використання функцій сервісу Rapid Miner або Orange. Після того, як користувач вказав шлях до набору даних і ввів ім’я цільового стовпчика відносно якого буде розраховано рівняння лінійної регресії, програма виводить результати з підрахованими коефіцієнтами та метриками. У даному випадку метрики особливо ефективні, оскільки дозволяють порівняти моделі регресії отримані для одного набору даних різними сервісами Data Mining. Приклад робочого вікна з розрахунками наведено на рисунку 2.11.

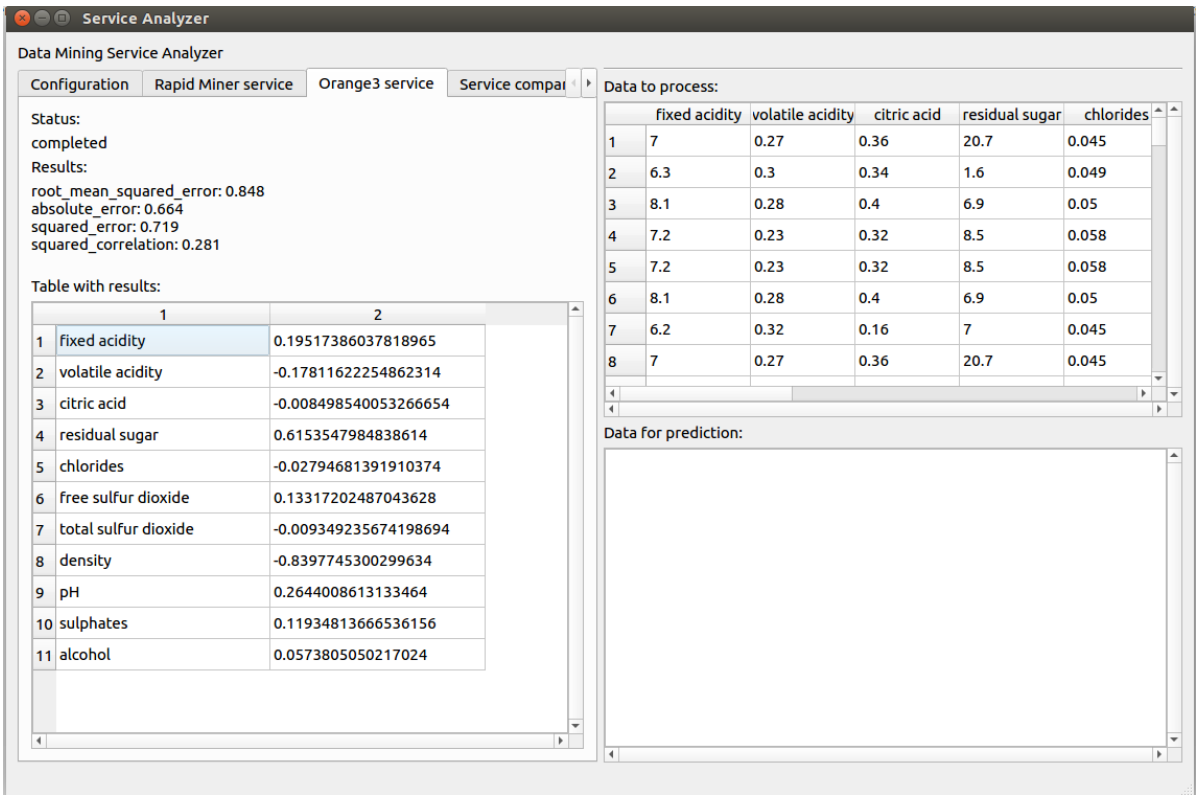


Рисунок 2.11 – Використання метрик оцінки лінійної регресії

Усі метрики були розраховані за допомогою функцій сервісів, що досліджувалися.

2.2.3 Корінь середньої квадратичної помилки

Корінь середньої квадратичної помилки RMSE [9] (англ. root mean squared error) це квадратний корінь MSE. Оскільки значення MSE складно інтерпретувати, був введений квадратний корінь для того, щоб масштаб помилок був таким самим як і масштаб цілей. Для підрахунку показника можна скористатися наступною формулою 2.9:

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}. \qquad (2.9)$$

Математичні переваги полягають у використанні показника при аналізі ефективності лінійної регресії, оскільки він дозволяє розділити варіацію у наборі даних на варіації, що об’єднані моделлю і варіації, що об’єднані випадковістю.

2.3 Метрики оцінки кластеризації

Для оцінки якості побудованих алгоритмом кластерів можна скористатися відповідними метриками. Можна виділити декілька основних видів валідації кластерів: зовнішня валідація, відносна валідація і внутрішня.

Зовнішня валідація – полягає у порівнянні результатів кластерного аналізу з попередньо відомими результатами. Очевидний недолік даного підходу полягає у не можливості оцінки кластеризації, якщо мітки кластерів не відомі. До зовнішніх мір оцінки якості відносяться:

- індекс Rand;
- індекс Adjusted Rand;
- індекс Жаккара;
- індекс Phi.

Відносна валідація – оцінює структуру кластерів, змінюючи різні параметри певного алгоритму. Використовується для таких методів кластеризації як k-means для підбору оптимального значення k.

Внутрішня валідація – використовує лише внутрішню інформацію, оцінюючи якість структури кластерів. До метрик внутрішньої валідації відносять:

- компактність кластерів;
- відокремленість кластерів;
- індекс Данна;
- показник Сілуетте;
- індекс Calinski–Harabasz;
- індекс Девіса–Болдуіна.

Після аналізу існуючих метрики, були зроблені висновки щодо доцільності використання метрик внутрішньої валідації при оцінці кластерів інструментом «Data Mining Service Analyzer». Вибір на користь даного виду валідації був зроблений через те, що він якнайкраще відповідає поставленим задачам, метрики зовнішньої валідації не можуть бути застосовані, оскільки для реальних даних, як правило, попередньо не відомий розподіл об’єктів до кластерів. Серед усіх метрик зовнішньої валідації були обрані наступні: індекс Девіса–Болдуіна [11], індекс Калінскі-Харабаш [12], показник

Сілуетте [13]. В наступних параграфах вище згадані метрики будуть розглянуті більш детально.

2.3.1 Показник Сілуетте (Silhouette)

Показник Сілуетте вимірює наскільки добре досліджуванні дані кластеризовані і розраховує середню відстань між кластерами. Графік Сілуетте показує міру наскільки близько кожна точка кластеру знаходиться відповідно до сусідніх кластерів. Даний метод дає графічне представлення про те, наскільки добре кожний об'єкт був класифікований. Приклад графіку побудований даним методом зображений на рисунку 2.12, він показує індекс Сілуетте для кожно з трьох видів ірисів. На графіку можна побачити аномальні дані для груп iris-versicolor і iris-virginica.

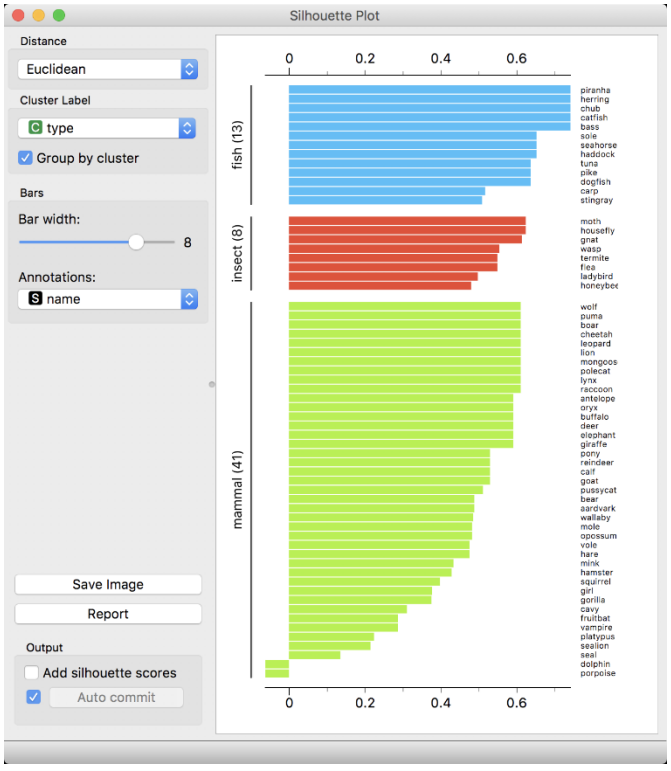


Рисунок 2.12 – графік Сілуетте для набору даних «Iris»

Для того, щоб розрахувати показник необхідно виконати наступні дії:

1. Для кожної точки $i \epsilon C_i$ ($де C_i$–кластер) визначити середню відстань до кожною точки всередині класу за формулою 2.13

$$a(i) = \frac{1}{|c_i| - 1} \sum_{i \epsilon c_i, i \neq j} d(i, j), \tag{2.13}$$

де $d(i,j)$ – відстань між точкою i та j,

$a(i)$ – можна інтерпретувати як показник того, наскільки добре для точки був підібраний кластер, чим менше значення, тим краща якість.

2. Наступним кроком, необхідно розрахувати несхожість точки i для кластеру C_k де ($C_k \neq C_i$. Для цього необхідно знайти середню відстань від i до всіх точок кластеру C_k за формулою 2.14.

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \neq C_k} d(i,j) \,. \tag{2.14}$$

3. Далі необхідно розрахувати показник Сілуетте для кожного з об'єктів за формулою 2.15.

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{if} |C_i| > 1 \,, \tag{2.15}$$
$$S(i) = 0, \text{if} |C_i| = 0 \,.$$

4. Останній етап полягає у отриманні коефіцієнта Сілуетте для оцінки отриманих кластерів, він розраховується за формулою 2.16:

$$SC = \max_k \tilde{S}(k) \,, \tag{2.16}$$

– де $\tilde{s}(k)$ – середнє значення всіх раніше підрахованих s(i).

Переваги метрики:

– зручна для інтерпретації, оскільки значення близьке до -1 свідчить про неправильну кластеризацію, натомість значення близькі до 1 сигналізують високу щільність кластеризації, у разі якщо показник близький до 0 – кластери перекриваються;
– показник вище, коли кластери щільні і добре відокремлені, що відноситься до стандартної концепції кластеру.

Недоліки даної метрики:

– як правило, показує кращий результат для випуклих кластерів [14];
– висока обчислювальна складність O(n2).

2.3.2 Індекс Калінскі-Харабаш (Calinski–Harabasz)

Індекс Калінського-Харабаша, також відомий як критерій коефіцієнта дисперсії – це відношення суми міжкластерної дисперсії та дисперсії всередині кластера для всіх кластерів, чим вище значення, тим кращі показники.

Для того, щоб розрахувати індекс припустимо, що в нас є вибірка даних E розміром n, яка була розподілена на k кластерів, далі скористаємося формулою 2.17:

$$\mathrm{CHI} = \frac{\frac{\mathrm{tr}(B_k)}{\mathrm{tr}(W_k)} * n - k}{k - 1}, \tag{2.17}$$

де $\mathrm{tr}(W_k)$ – матриця внутрішньої дисперсії, що розраховується за формулою 2.18,

$\mathrm{tr}(B_k)$ – матриця зовнішньої дисперсії, для розрахунку може бути використана формула 2.19

$$W_k = \sum_{i=1}^{k} \sum_{x \epsilon C_i} (x - c_i)(x - c_i)^T, \tag{2.18}$$

$$B_k = \sum_{i=1}^{k} n_i (c_i - C_E)(c_i - C_E)^T. \tag{2.19}$$

Для формул 2.18 і 2.19 C_i – множина точок, що належать кластеру i, c_i – центр кластера i,C_E – центр набору даних E, n_i – кількість точок для кластера i.

Переваги даної метрики:

– показник вище, коли кластери щільні і добре відокремлені, що відповідає стандартній концепції кластеру;
– показник швидкий для обчислення.

Недоліки метрики: як правило, показує кращий результат для випуклих кластерів;

2.3.3 Індекс Девіса–Болдуіна (Davies-Bouldin)

Індекс Девіса-Болдуіна означає середню «подібність» між кластерами, де подібність є мірою, яка порівнює відстань між кластерами з розміром самих кластерів. Нижчий індекс Девіса-Боулдіна відноситься до моделі з кращим розділенням між кластерами.

Індекс визначається як середня подібність між кожним кластером C_i для i=1,..,k і найбільш подібним до нього кластером C_j, де i≠j. В контексті цього індексу, подібність визначається як значення R_{ij}, що визначається наступними показниками:

– s_i, середня відстань між кожною точкою кластеру і і центром цього кластеру;
– dij , відстань між центрами кластерів і та j.

Формула для підрахунку простого показника R_{ij} вказана під номером 2.20.

$$R_{ij} = \frac{(s_i + s_j)}{d_{ij}}. \quad (2.20)$$

–

Показник Девіса-Боулдіна визначається за формулою 2.21

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} R_{ij} . \quad (2.21)$$

Переваги даної метрики:

– обчислення індексу Девіса-Боулдіна набагато простіше ніж обчислення показника Сілуетте;
– індекс обчислюється лише за ознаками, властивими набору даних.

Недоліки метрики:

– використання центроїдної відстані обмежує метрику відстані до евклідового простору;
– як правило, показує кращий результат для випуклих кластерів;

2.3.4 Застосування метрик оцінки кластеризації в програмному продукті

Програмний продукт «Data Mining Service Analyzer» реалізує функціонал для оцінки кластерів отриманих в результаті роботи сервісів Orange3 і Rapid Miner. У якості метрик оцінки якості кластеризації були використані вище згадані внутрішні метрики. Приклад підрахунку показників якості для кластерів зображений на рисунку 2.22.

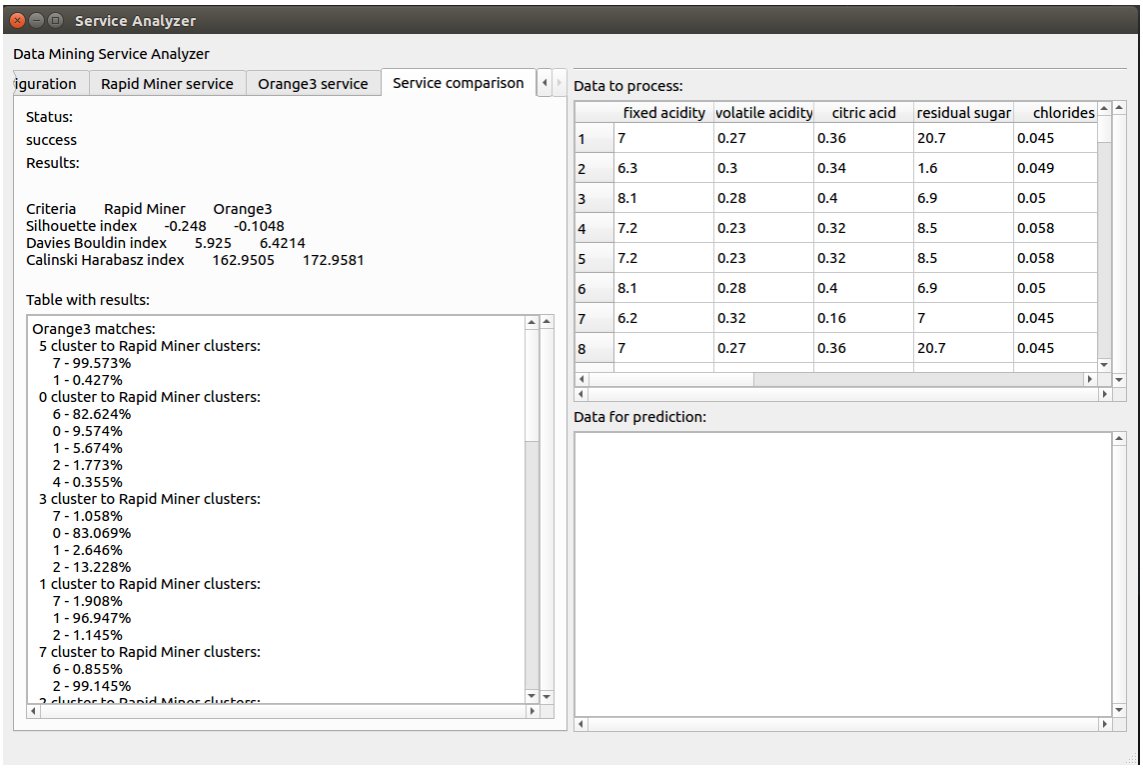


Рисунок 2.22 – Отримання оцінок кластеризації

Для отримання індексів були використані функції бібліотеки sklearn []

2.3.4 Методи ефективного порівняння кластерів для двох сервівсів

Оскільки однією з головних задач програми «Data Mining Service Analyzer» є надання функціоналу для зручного порівняння роботи сервісів, виникає проблема при аналізі кластерів отриманих від двох сервісів. Під час експериментів було виявлено, що у більшості випадків сервіси видають різні кластери для одного й того самого набору даних. Звичайно, якість кластеризації можна виявити порівнявши оцінки метрик для обох сервісів, але для деяких задач, цікаво дізнатися як кластери, отриманні різними сервісами, перетинаються між собою. Для вирішення даної задачі був розроблений відповідний функціонал. На рисунку 2.23 зображено приклад знаходження перетину кластерів різних сервісів для набору даних «Іриси».

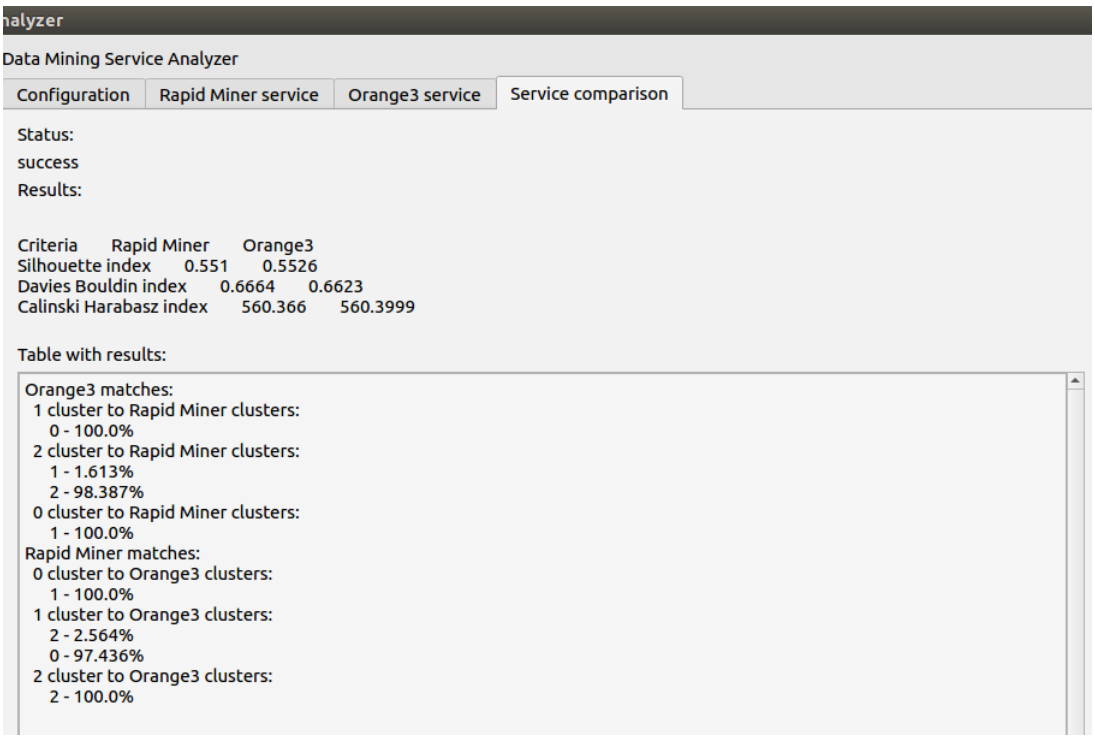


Рисунок 2.23 – Знаходження перетину кластерів

Як видно з рисунку 2.23, об’єкти ірисів були розподілені між трьома кластерами, де 1-й кластер отриманий від сервісу Rapid Miner на 100% подібний до 0-го кластеру отриманому від сервісу Orange. Другий кластер сервісу Orange перетинає перший кластер Rapid Miner на 1.6%, і його другий кластер на 98.4%.

2.3.5 Візуальне порівняння результатів кластеризації

Для візуального порівняння результатів кластеризації у програмному продукті «Data Mining Service Analyzer» була розроблена можливість побудови простої стовпчастої діаграми. На рисунку 2.24 і 2.25 зображені приклади побудови графіків кластеризації для сервісів Orange і Rapid Miner відповідно, по осі абсцис відображена кількість об’єктів у кластерів, по осі ординат розташовані кластери за номерами. Перевага використання стовпчастих діаграм для аналізу кластеризації полягає у можливості для користувача швидко порівняти кластери отримані різними сервісами і зробити висновки щодо їх подібності.

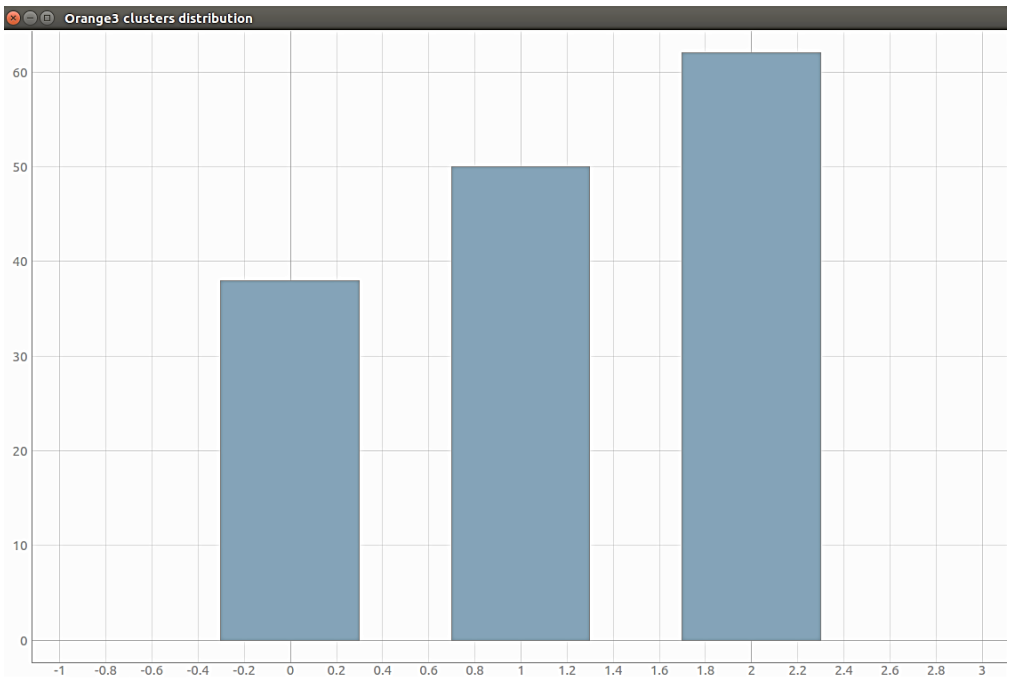


Рисунок 2.24 – Діаграма розподілення класів для сервісу Orange

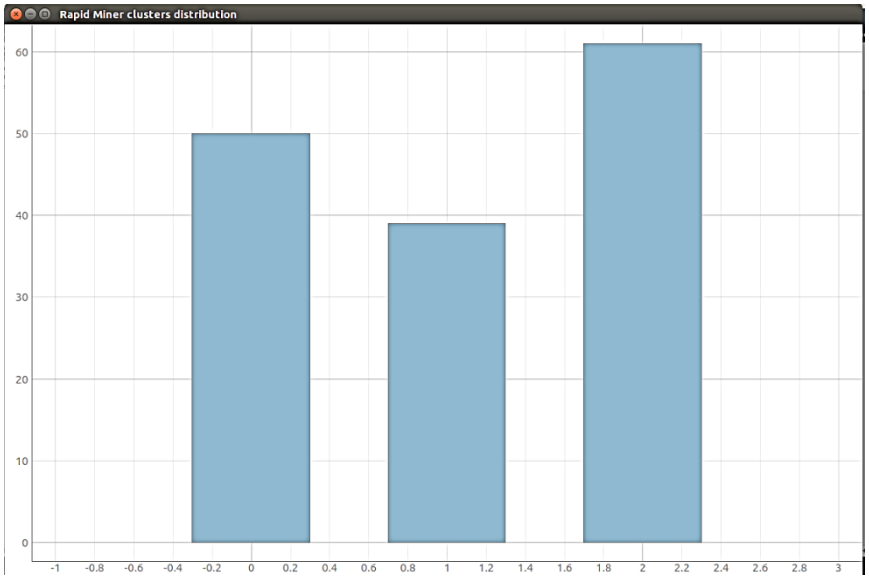


Рисунок 2.25 – Діаграма розподілення класів для сервісу Rapid Miner

2.4 Метрики оцінки передбачення

Оскільки метод передбачення використовує модель лінійної регресії, для оцінки якості прогнозів використовуються ті ж самі метрики: коефіцієнт детермінації, середня абсолютна помилка, середня квадратична помилка і корінь від середньої квадратичної помилки. Дані метрики дозволяють дізнатися відхилення результатів, що були передбачені, від дійсних даних. У разі, якщо відхилення дуже великі, модель підібрана для передбачень, вважається не коректною.

2.5 Часова залежність алгоритму від об’єму даних

Однією з головних задач Data Mining є ефективна робота з великими даними. Для того, щоб перевірити, як буде себе поводити будь-який з запропонованих алгоритмів (кластеризація, знаходження викидів, знаходження моделі лінійної регресії або передбачення) на збільшених даних, був розроблений функціонал для побудови графіку залежності часу виконання програми від об’єму даних. Користувач має змогу обрати кількість експериментів і вказати дані для дослідженні, результатом роботи програми буде відображення графіку залежності (рис. 2.26).

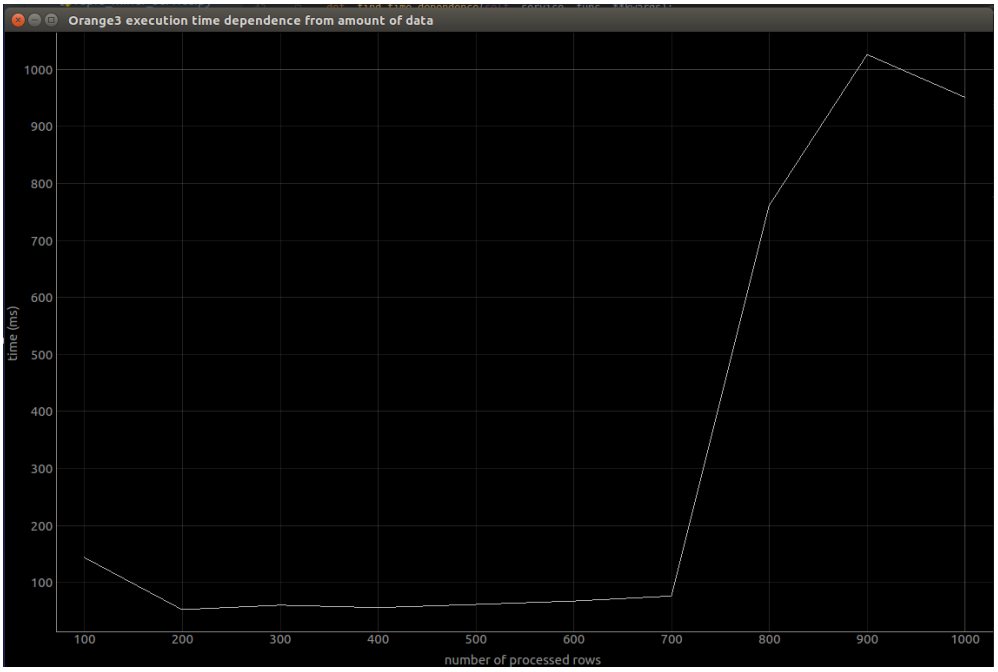


Рисунок 2.26 – Графік залежності

Побудова графіку залежності є корисною функцією для тих випадків, коли необхідно передбачити час виконання алгоритму на великих об’ємах даних.

2.8 Системні обмеження для аналізу сервісів

При аналізі сервісів користувач може стикнутися з наступними системними обмеженнями:

– брак оперативної пам’яті при опрацюванні великих об’ємів даних. У такому випадку рекомендується завчасно перевірити відповідність ПК системним вимогам програми. До можливих негативних наслідків можна віднести зависання системи.

– брак пам’яті на жорсткому диску при побудуванні графіка залежності часу від об’єму даних. Користувач повинен заздалегідь попіклуватися про достатній об’єм пам’яті на диску перед початком експерименту.

2.9 Опис апаратних та програмних засобів

Для функціонування системи аналізу сервісів, необхідно використовувати апаратні засоби з наступними мінімальними характеристиками:

– два ядра;
– 2 гГц частота процесору;
– більше 1 Гб вільного простору на диску;
– 1280x1024 розширення монітору.

Операційні системи, що підтримуються:64 бітна система, Ubuntu 16.04.

Висновки до другого розділу

Під час вибору метрик для оцінки сервісів Data Mining з метою подальшого порівняння алгоритмів було вирішено оцінювати кожну функцію сервісу окремо. Таким чином були проаналізовані та підібрані відповідні метрики для кожної з наступних задач:

– кластеризація методом k-середніх;
– прогнозування даних на основі моделі лінійної регресії;
– побудова моделі лінійної регресії;
– знаходження викидів;

Такий метод оцінки сервісів дозволяє дізнатися який з них дає меншу похибку при виконанні алгоритму.

Окрім стандартних метрик, був розроблений допоміжний функціонал, що дозволяє візуально порівнювати результати роботи сервісів. Варто також зазначити, що для оцінки часової ефективності сервісів були впроваджені заміри виконання функцій і можливість побудови графіка часової залежності від об’єму даних.

3 РОЗРОБКА ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ ДОСЛІДЖЕННЯ СЕРВІСІВ DATA MINING

3.1 Формалізація задачі

Опис взаємодії користувача з програмою відображений через діаграму прецедентів. Діаграма була розроблена для того, щоб забезпечити більш точне розуміння того, як повинна працювати система з точки зору користувача. Під актором на схемі розуміється користувач програми, який взаємодіє з сервісом. Під варіантами використання розуміється опис послідовності дій, які може здійснювати система у відповідь на зовнішні дії користувачів. Варіанти використання системи взаємодіє зображені на рис. 3.1.

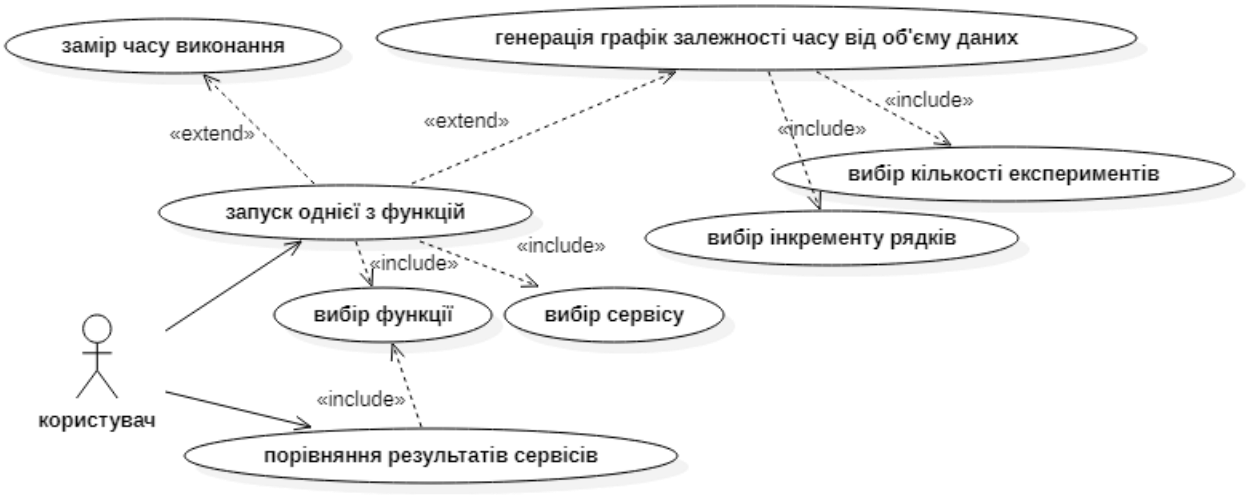


Рисунок 3.1. – Взаємодія користувача з системою

3.2 Базова архітектура системи

У ході розробки інструменту для дослідження сервісів Data mining була створена система, що складається з трьох основних компонентів:

– інтерфейс користувача;
– основна частина програми, де здійснюються обчислювальні процеси;
– модуль для взаємодії інтерфейсу користувача з основною системою.

Графічний інтерфейс користувача було розроблено в інтегрованому середовищі розробки QtDesigner. Для того щоб інтегрувати інтерфейс з основною частиною програми його було переведену в код написаний за допомогою фреймворку PyQt5. Даний фреймворк є оболонкою на мові програмування Python для бібліотеки

Qt5. Основна частина програми написана на інтерпретованій об’єктно-орієнтованій мові програмування – Python. Обрана парадигма програмування – об’єктно-орієнтована. В процесі проектування класів програми було дотримано основних принципів об’єктно-орієнтованого програмування та дизайну SOLID [15]. Для більш повного відображення взаємодії модулів в системі була розроблена діаграма компонентів на мові UML. Дана діаграма відображає компоненти, залежності та зв’язки між ними. Діаграма зображена на рисунку 3.2.

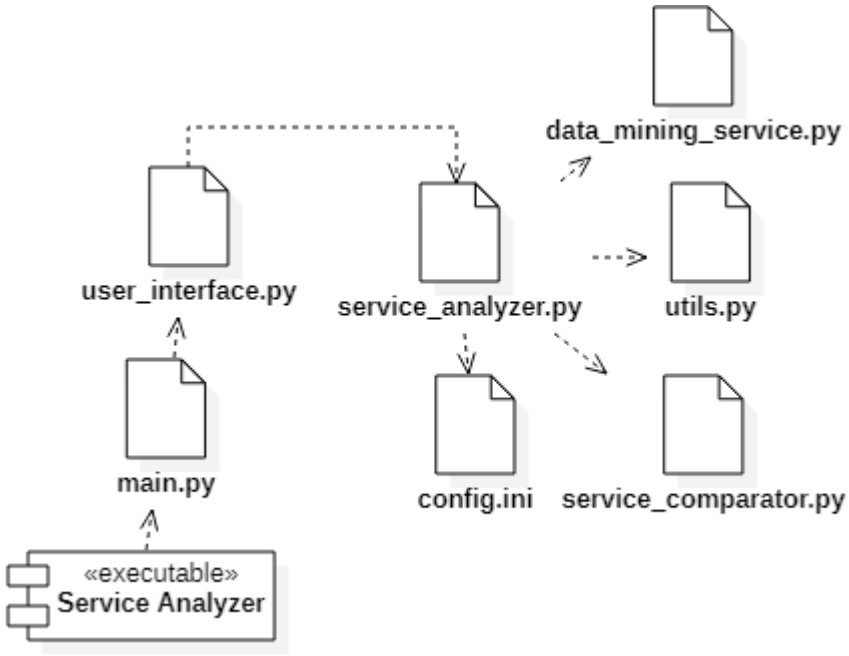


Рисунок 3.2 – Взаємодія компонентів в системі

3.3 Внутрішнє проектування

3.3.1 Вибір мови програмування

У якості основної мови програмування була обрана мова Python3. Перевага саме цій мові була надана через ряд причин:

- код написаний на Python має високу читабельність, адже є близьким до англійської мови;
- швидкість написання коду є більшою порівняно з більш низькорівневими мовами програмування, наприклад такими як C/C++;
- має велику кількість бібліотек для інтелектуальної обробки даних, зокрема: pandas, numpy;

- основні сервіси Data Mining мають бібліотеки написані на мов Python для взаємодії з ними;
- є крос-платформною мовою програмування;
- має велику кількість фреймворків, що полегшує і прискорює процес розробки застосунків;
- має свої стандарти написання коду PEP8 [16] та забезпечений детальною документацією.

В таблиці 3.1 представлено порівняння мови програмування Python з іншими популярними мовами.

Таблиця 3.1 – Порівняння мов програмування

Мова програмування	Основне призначення	Типи парадигм програмування			Стандарти
		Об'єктно-орієнтоване	функціональне	Процедурне	
1	2	3	4	5	6
Python	Додатки загального застосування, web, скрипти, штучний інтелект, наукове програмування	+	+	+	PEP

Продовження таблиці 3.1

1	2	3	4	5	6
C#	Додатки, програми клієнти, програми-сервери, web додатки, бізнес додатки	+	+	+	2000, ECMA, ISO
C++	Системні програми	+	+	+	1998. ISO/IEC 2003, ISO/IEC 2011,ISO/IEC 2014,ISO/IEC 2017
GoLang	Web додатки, сервера	-	-	+	Go Language Specification
Java	Додатки, клієнт-серверні сервіси, web, бізнес додатки	+	+	+	Java Language Specification

Аналізуючи таблицю можна прийти до висновку, що мова Python якнайкраще підходить для дослідження та вирішення задач інтелектуальної обробки даних.

3.3.2 Технологічна платформа

Під час реалізації системи використовувалася така технологічна платформа.

PyQt – це набір Python бібліотек для створення графічного інтерфейсу на базі платформи Qt5. Є крос-платформною бібліотекою, яка працює на всіх основних операційних системах, в тому числі таких як: Unix, Windows та Mac OS.

Qt — інструмент розробки програмного забезпечення реалізований мовою програмування C++. Має велику кількість компонентів для реалізації графічного інтерфейсу. Оскільки основна частина програми написана на мові Python, її графічний інтерфейс був розроблений у IDE QtDesigner і потім переведений на фреймворк PyQt5 за допомогою утиліти pyuic5.

3.3.3 Ієрархія та взаємодія класів системи

Для відображення ієрархії та взаємодії класів системи була розроблена діаграма класів зображена на рис. 3.3.

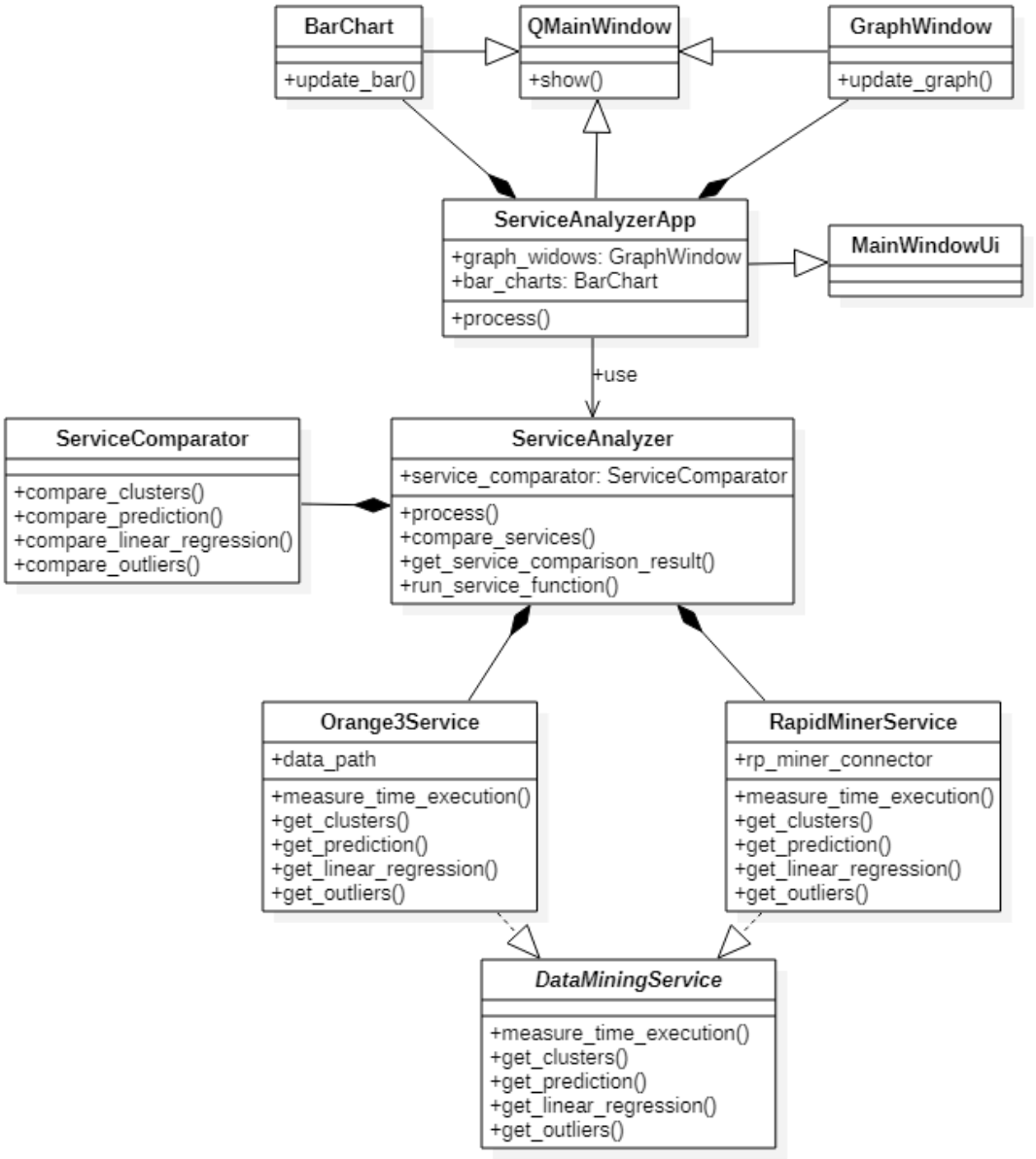


Рисунок 3.3 – Діаграма класів

3.3.4 Використані принципи проектування

При проектуванні системи було дотримано п’яти основних принципів об’єктно-орієнтованого програмування SOLID, а саме:

– принцип єдиної відповідальності – класи створюються для виконання лише однієї задачі, яку вони повністю інкапсулюють;

- принцип відкритості та закритості – класи та функції розроблені відкриті для розширення та закриті для змін;
- принцип заміщення Лісков – розроблені ієрархії класів для можливого заміщення;
- принцип розділення інтерфейсу – інтерфейси мають вузьку направленість;
- принцип інверсії залежностей – модулі вищого рівня не залежать від модулів нижчого рівня.

3.4 Розробка інтерфейсу користувача

Програмне забезпечення для аналізу сервісів Data Mining має декілька вікон графічного інтерфейсу для користувача. Програма представляє собою універсальний інструмент для проведення експериментів з метою виявлення ефективності обробки заданих даних одним чи обома сервісами. Після запуску програми перед користувачем з’являється головне вікно (рис.3.4).

Вікно містить необхідні параметри для налаштування та запуску експерименту. Перш за все, необхідно обрати розташування файлу з тестовим набором даних формату csv. Після того, як файл було обрано, його зміст буде відображено в правій верхній частині робочого вікна. За замовчуванням у якості експериментальних використовується файл з даними про хімічний склад червоного вина. Даний файл поставляється разом з програмою. Після того, як було вказано розташування даних, необхідно обрати одну з запропонованих функцій: знаходження викидів, знаходження коефіцієнтів лінійної регресії, кластеризація або передбачення. В залежності від того, яка функція буде обрана, можуть активуватися такі вікна для вводу даних як:

- розташування файлу з даними для передбачень – активізується якщо обрана функція передбачення. Після того, як файл було обрано, дані будуть завантажені в таблицю в правому нижньому кутку;
- ім’я цільової колонки – активізується якщо була обрана функція регресійного аналізу або передбачення;

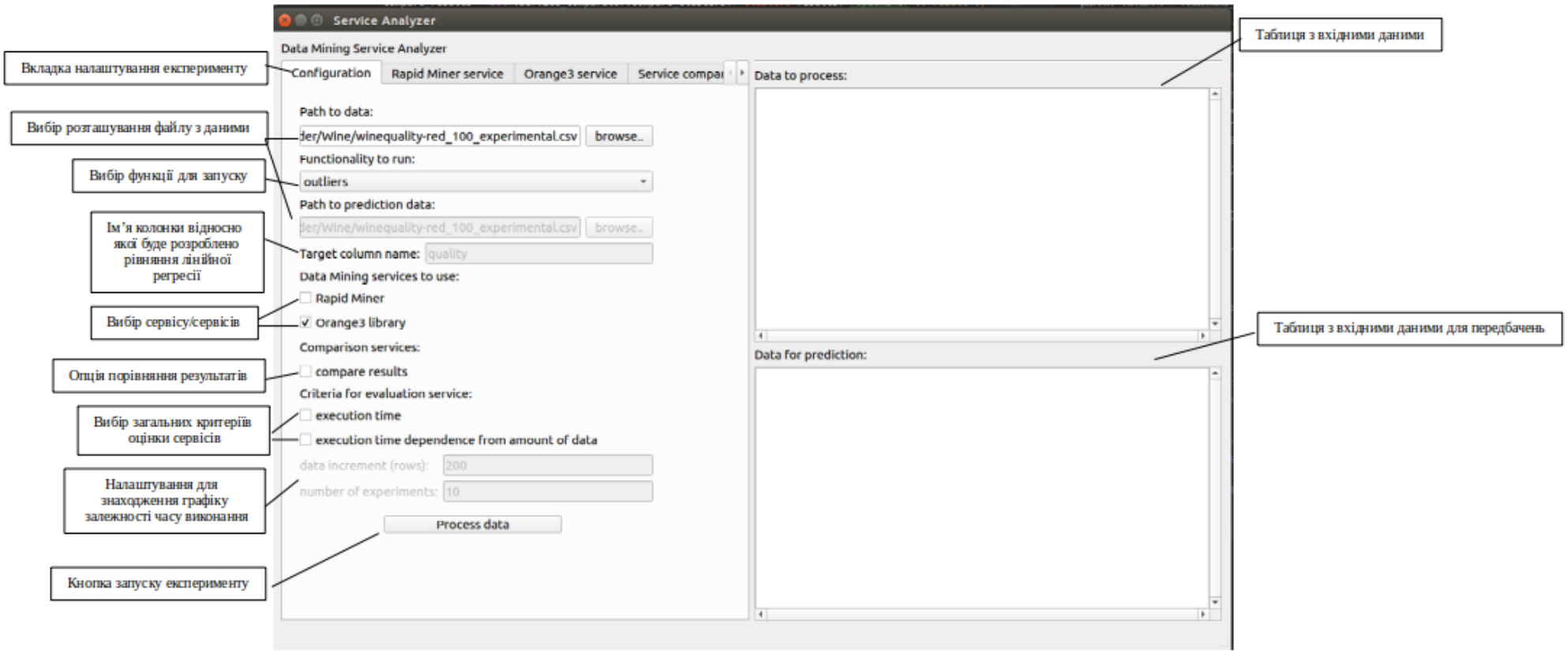


Рисунок 3.4 – Головне вікно програми

На рисунку 3.5 зображено головне вікно з завантаженими таблицями даних.

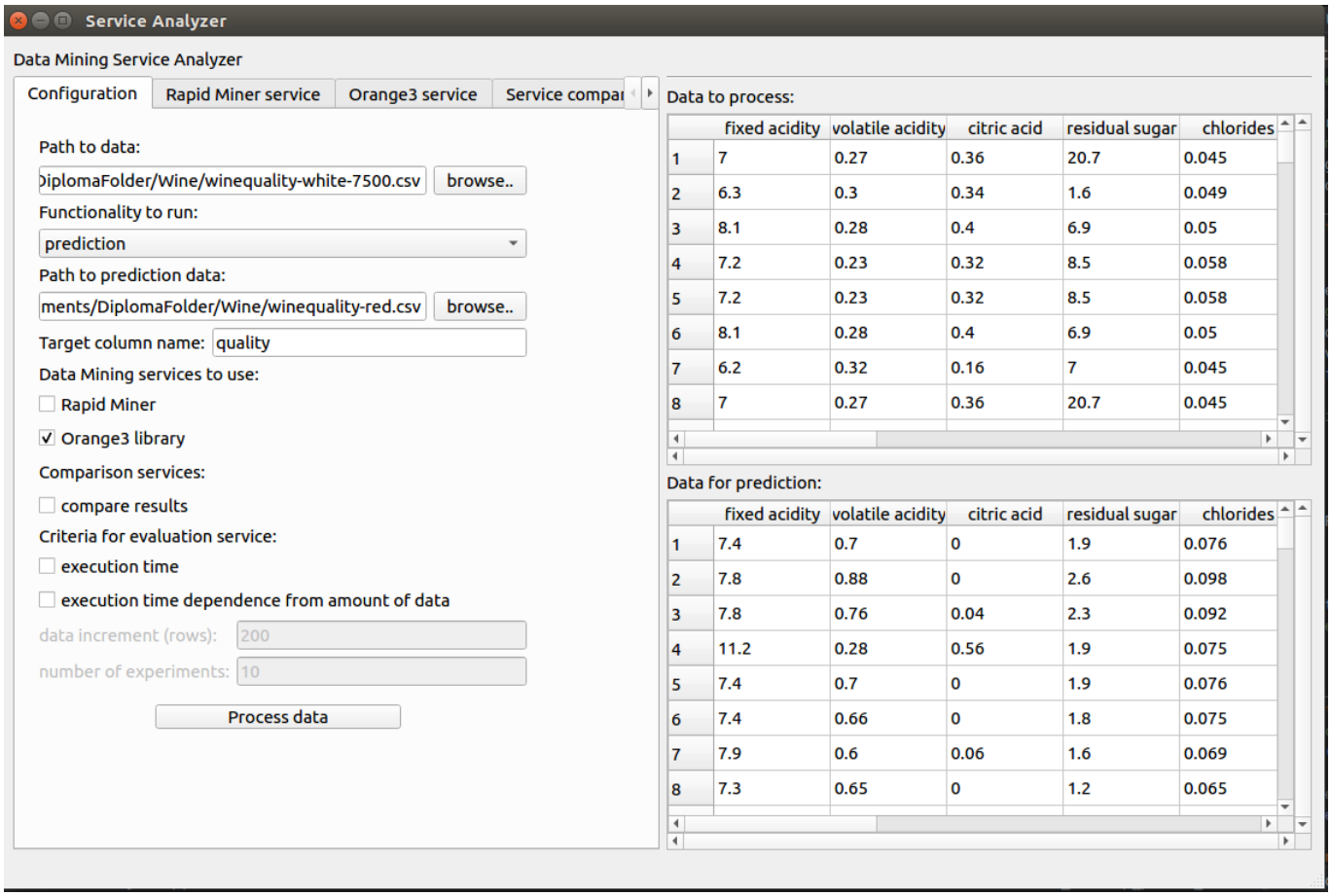


Рисунок 3.5 – Завантаження даних у таблиці

Наступним кроком у налаштуванні експерименту є вибір сервісів, результати яких цікавлять користувача. Для того, щоб обрати необхідний сервіс необхідно виставити галочку напроти нього.

Після того, як було обрано сервіси необхідно виставити критерії оцінки сервісів, вони є опціональними і невибрані за замовчуванням. До критеріїв оцінки сервісів належить:

– час за який сервіс виконує задану функцію;

– графік залежності часу виконання від об’єму даних.

За умови, якщо користувач вибрав генерацію графіку наступні поля для вводу активізуються:

– кількість експериментів;

– число на яке буде збільшуватись кількість рядків з кожним експериментом.

Коли всі дані для проведення експерименту були налаштовані, користувач повинен натиснути на кнопку «Process». Після обробки даних результати кожного з сервісів будуть відображені у відповідній вкладці (див. рис 3.6, 3.7). У разі якщо була обрана опція «Compare service», у вікні з результатами порівняння сервісів будуть виведені результати. Макет вікна представлений на рисунку 3.8. На рисунку 3.9 зображено вікно з результатами порівняння функції передбачення.

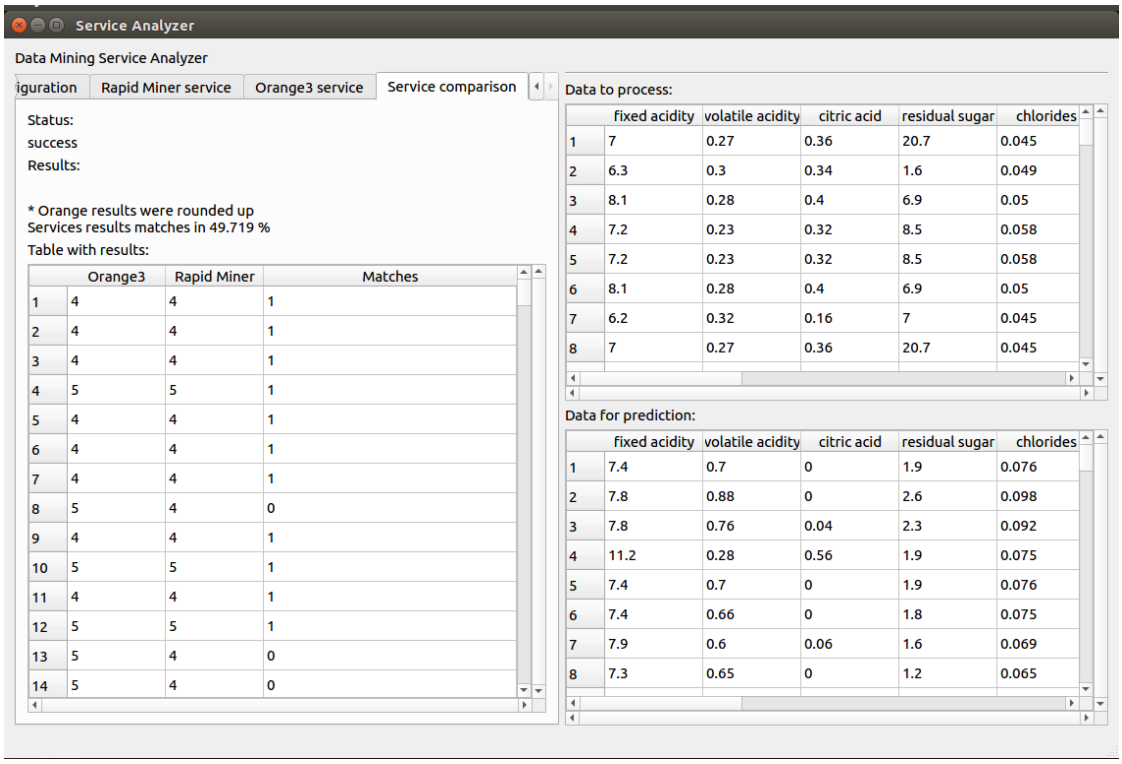
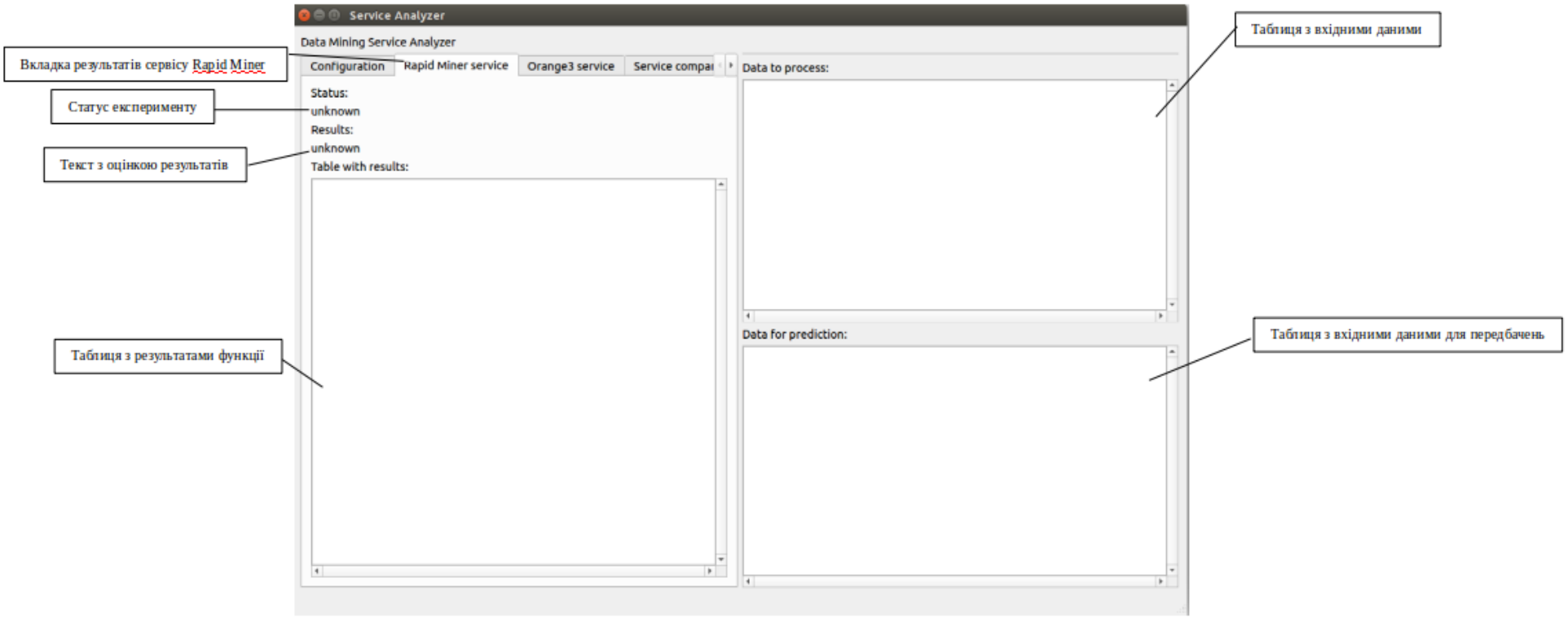


Рисунок 3.9 – Вікно з результатами порівняння сервісів

У більшості випадків усі результати порівняння сервісів відображаються у вкладці «Service comparison». Виняток становить запуск програми з увімкненою опцією «execution time dependence from amount of data», даний критерій порівняння активізує створення нових вікон з графіками.



3.6 – Вікно з результатами сервісу Rapid Miner

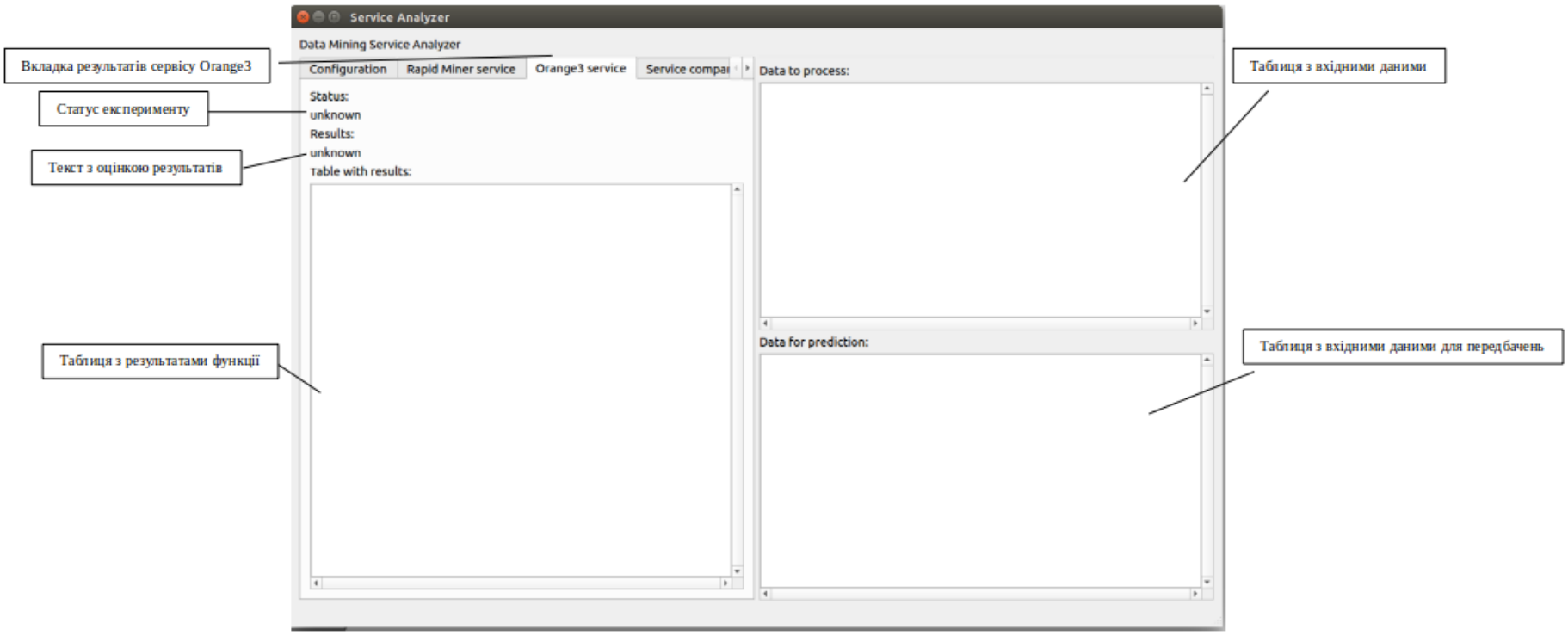


Рисунок 3.7 – Вікно з результатами сервісу Orange3

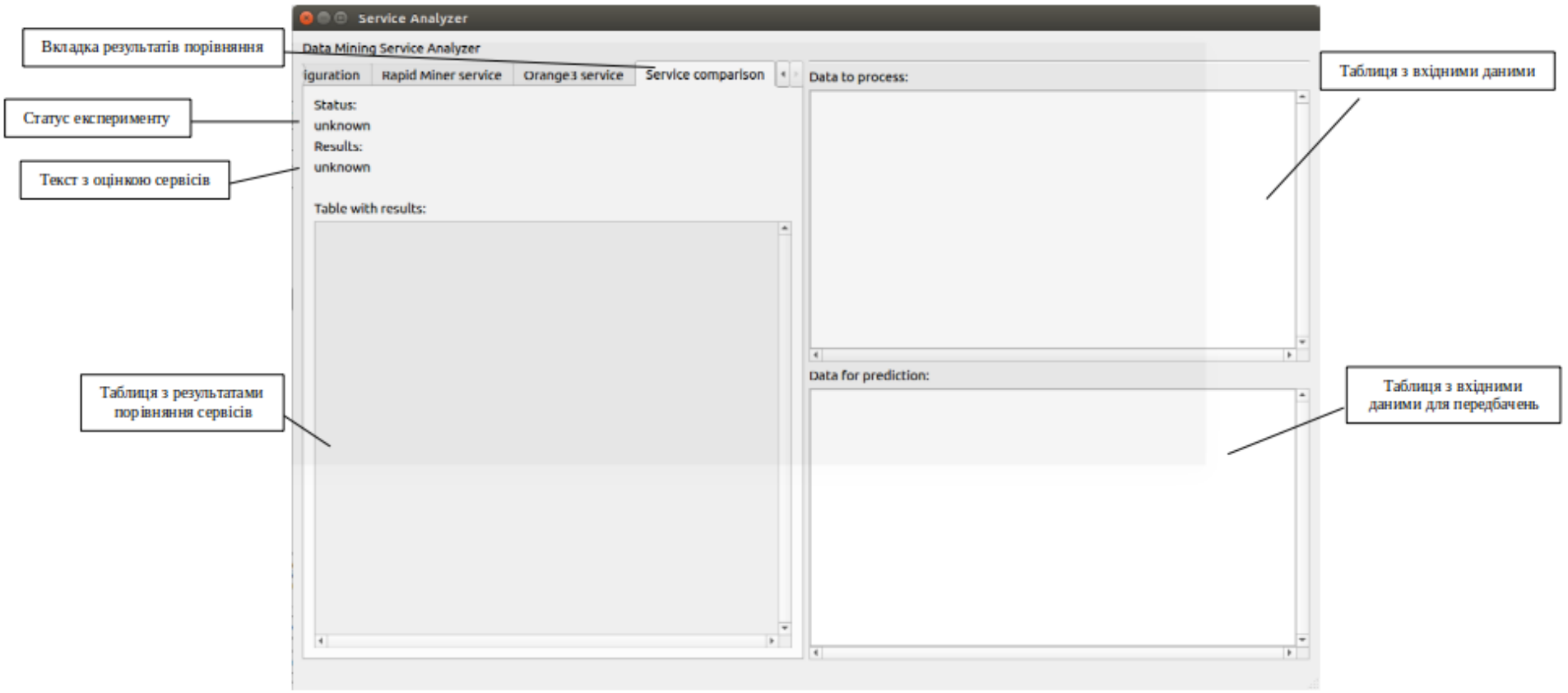


Рисунок 3.8 – Вікно з порівнянням результатів сервісів

Приклад вікна з відображенням графіка залежності часу виконання функції від об’єму даних для сервісу Orange3 представлений на рисунку 3.10.

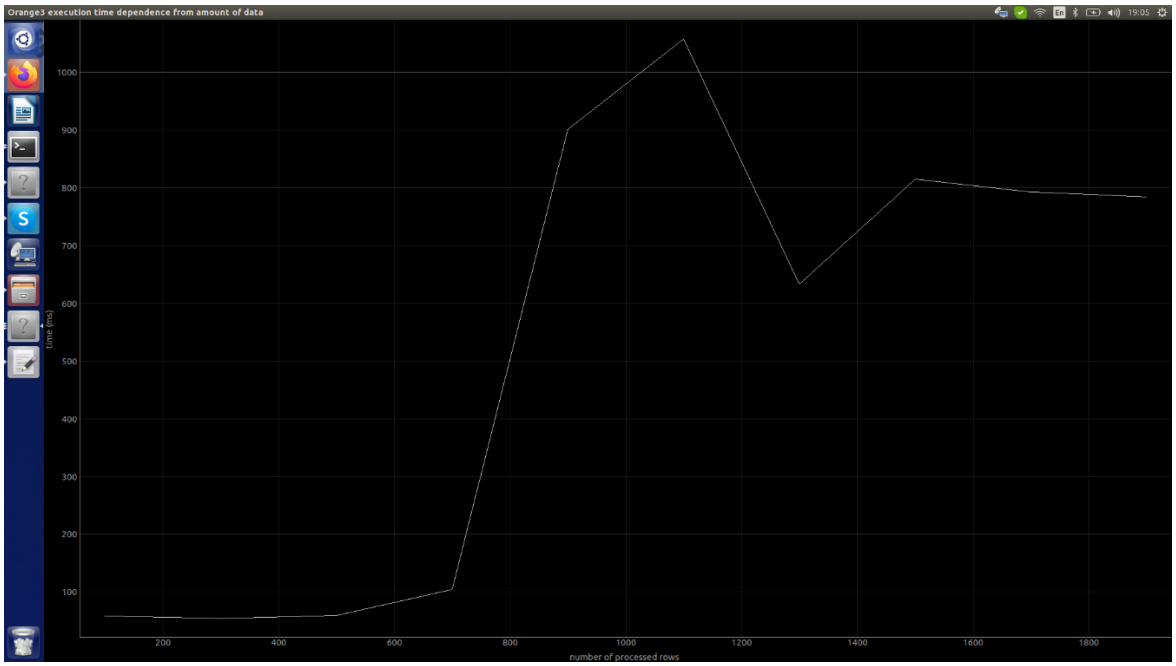


Рисунок 3.10 – Вікно з відображенням графіку часової задежності

Для візуального аналізу результатів кластеризації програма створює нові вікна в яких виводить користувачу звичайну стовпчасту діаграму. Приклад вікна з відображенням діаграми представлений на рисунку 3.11.

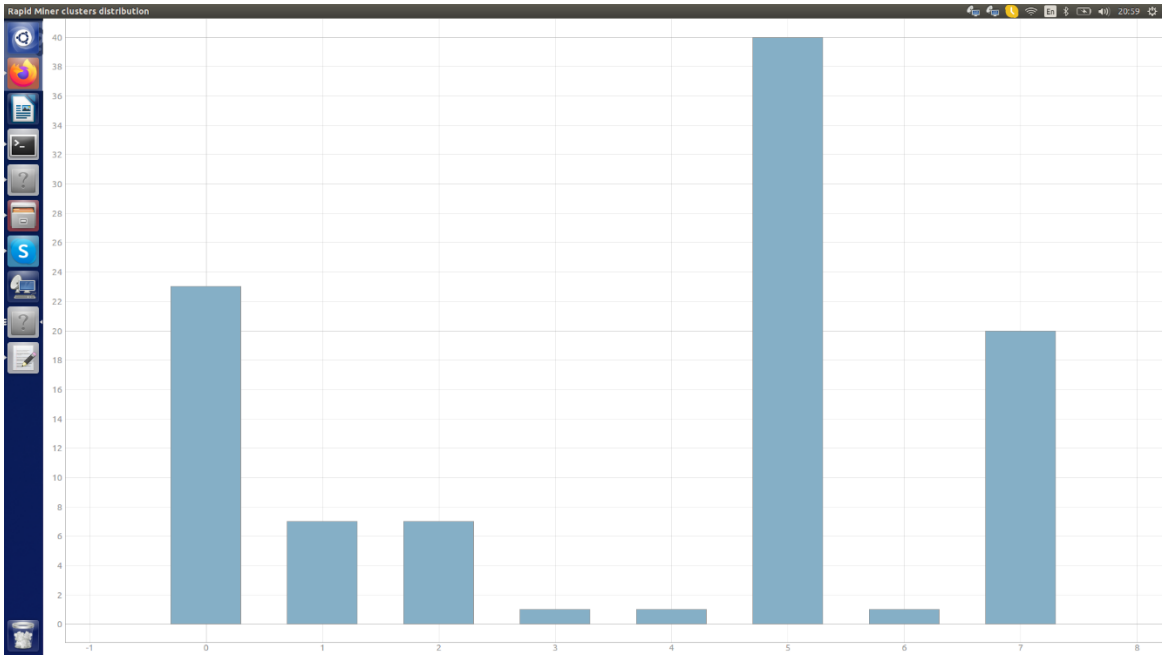


Рисунок 3.11 – Вікно з діаграмою розподілення кластерів

Висновки до третього розділу

Виконано проектування архітектури інструменту для дослідження сервісів інтелектуальної обробки даних для чотирьох головних задач Data Mining: знаходження викидів, кластеризація даних, регресійний аналіз та передбачення значень для обраної колонки даних. У ході проектування архітектури системи було передбачено можливість розширення системи за рахунок її розбиття на слабко пов’язані модулі. У разі подальшого розвитку програми програміст зможе швидку додати роботу з новими сервісами Data Mining та впровадити нові аналіз нових функцій. Відповідно до SOLID-принципів, кожен клас системи має тільки один обов’язок, розроблені програмні інтерфейси відкриті для розширення, але закриті для змін. Це дозволить повторно використовувати розроблені класи та модулі, розширювати функціонал.

Завдяки використанню загальновідомих шаблонів проектування, будь-якому розробнику буде легко зрозуміти принципи функціонування системи та виконувати її підтримку.

У розділі наведено опис проектування інтерфейсу користувача. Інтерфейс був розроблений максимально простим та інтуїтивно зрозумілим. Для розробки інтерфейсу використовувалися знайомі користувачу елементи; увесь інтерфейс системи умовно поділено на блоки для вкладки, що значно полегшує процес користування та навігацію програмою.

Система готова до впровадження та використання.

4 ПОРІВНЯННЯ ЕФЕКТИВНОСТІ СЕРВІСІВ DATA MINING

Під час порівняння сервісів технології Data Mining було проведено ряд незалежних експериментів на різних видах даних, взятих з відкритих ресурсів. Метою проведення експериментів було виявлення різниці у поведінці сервісів при вирішенні однакових задач. Для кожної з функцій сервісів були окремо підібрані такі експерименти, що у повній мірі відображають її ефективність або неефективність для конкретних наборів даних. Процес дослідження проводився на ПК з наступними апаратними характеристиками:

- об’єм оперативної пам’яті – 12 Гб;
- об’єм жорсткого диску – 500 Гб;
- двохядерний процесор i5-7200U 2.5 ГГц.

Для перевірки часової ефективності була налаштована віртуальна машина з наступними характеристиками:

- об’єм оперативної пам’яті – 6 Гб;
- об’єм жорсткого диску – 70 Гб;
- одноядерний процесор.

Перед проведенням експерименту були налаштовані наступні програмні компоненти:

- встановлено програмне забезпечення Rapid Miner Studio 9.8;
- встановлена python3 бібліотека Orange версії 3.27.1;
- встановлена python3 бібліотека sklearn для проведення додаткових розрахунків при обчисленні метрик якості;
- налаштована і встановлена програма «Data Mining Service Analyzer».

Також були попередньо знайдені відповідні набори даних для подальшого тестування.

4.1 Експериментальні дослідження функції LOF для знаходження викидів

Для оцінки ефективності функції LOF програмне забезпечення «Data Mining Analyzer» використовує наступні критерії якості:

- метрика AUC-ROC;
- час обробки даних.

При проведенні експериментів були знайдені і проаналізовані показники для наступних видів даних:

- нормалізовані дані;
- ненормалізовані дані;
- збільшені нормалізовані дані;
- збільшені ненормалізовані дані;

Для кожного виду даних було підраховано час за який сервіс виконує функцію. Також аналізу часової ефективності сервісів був побудований графік залежності часу обробки від об’ємів даних на двохядерному та одноядерному процесорі.

4.1.1 Експеримент №1 – набір даних «Ionosphere»

Опис набору тестових даних

У якості тестових даних для знаходження викидів був взятий набір «Ionosphere» з сайту UCI Machine Learning Repository. Набір даних «Ionosphere» представляє собою радіолокаційні дані, що були зібрані системою в Гус-Бей, Лабрадор. Мішенями радарів були вільні електрони в іоносфері. Усі сигнали поділяються на хороші і погані, поганий сигнал є викидом. Набір має 34 атрибути, останній з яких вказує на те чи є сигнал хорошим. У таблиці описано 351 експеримент, викиди складають 92 експерименти, що становить 26% від загальної кількості даних.

Результати експерименту

Результати експериментів були занесені до таблиці 4.1.

Таблиця 4.1 – Результати експерименту для набору даних «Ionosphere».

№	Тип даних	Rapid Miner		Orange	
		AUC-ROC	Час виконання (мс)	AUC-ROC	Час виконання (мс)
1	2	3	4	5	6
1	не нормалізовані дані	0,341	212	0,809	83

Продовження таблиці 4.1

1	2	3	4	5	6
2	нормалізовані дані	0,83	251	0,821	47
3	не нормалізовані збільшені дані	0,5	103072	0,494	15441
4	нормалізовані збільшені дані	0,5	11069	0,79	19284

Була досліджена залежність швидкості обробки даних від їх об’єму на двоядерному (рис. 4.1) та одноядерному процесорі (рис. 4.2)

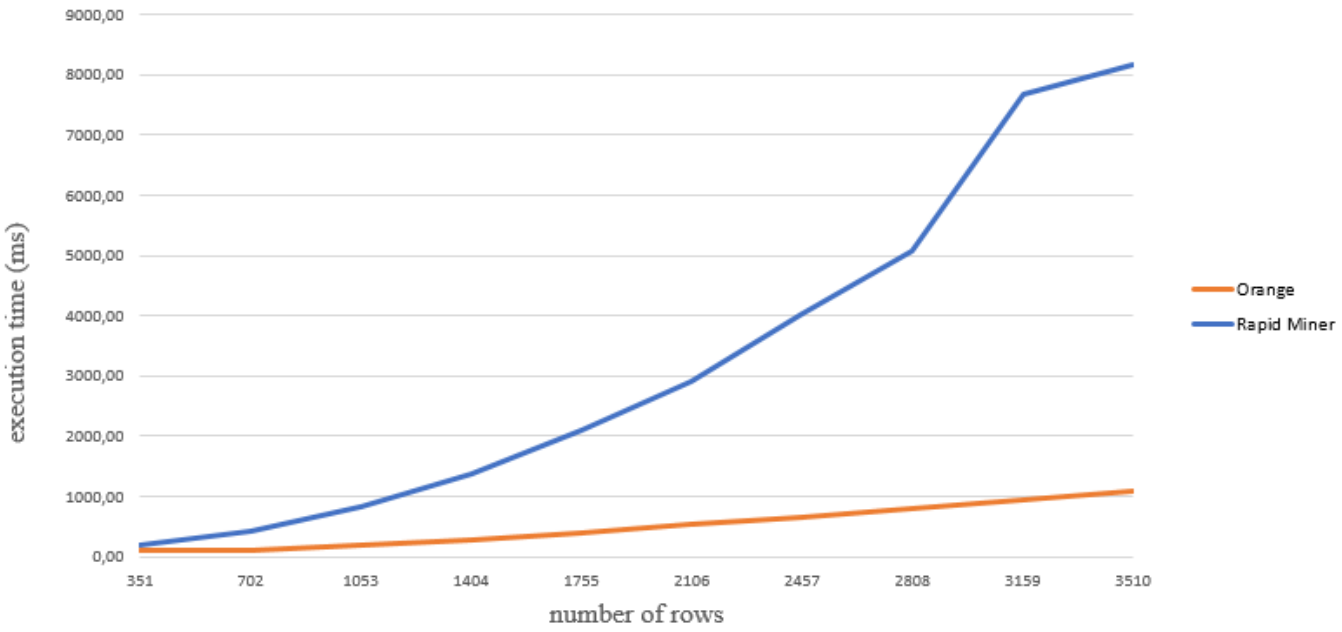


Рисунок 4.1 – Графік залежності часу виконання на двоядерному процесорі

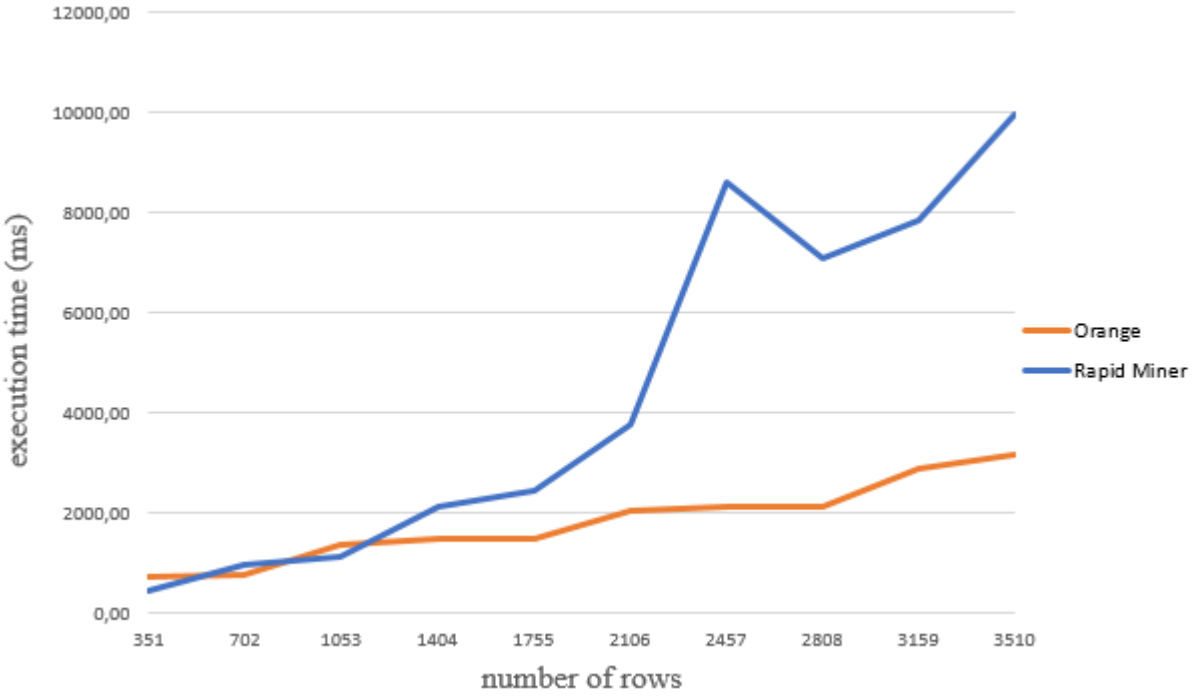


Рисунок 4.2 – Графік залежності часу виконання на одноядерному процесорі

Інтерпретація результатів

Аналізуючи час виконання експериментів різними сервісами, можна прийти до висновку, що програмне забезпечення Orange працює набагато швидше (щонайменше вдвічі) ніж система Rapid Miner. Таку поведінку можна обґрунтувати вибором мови реалізації функцій, оскільки Orange використовує C++ для оптимізації результатів, у той час як весь функціонал Rapid Miner реалізований на мові Java.

Досліджуючи вплив нормалізації на показник AUC-ROC можна припустити, що нормалізація даних не дає великого приросту на оцінку результатів сервісу Orange, порівняно з оцінкою, що була отримана на ненормалізованих даних. Однак, нормалізація значно впливає на результат отриманий від програми Rapid Miner, оскільки до нормалізації AUC-ROC дорівнював 0,3, а після став 0,8. Така різниця у поведінці може бути пояснена тим, що сервіс Orange скоріш за все має вбудовану систему нормалізації для даних, що обробляються функцією LOF. На противагу йому, Rapid Miner такого функціоналу не має, тому для нього важливо явно обробляти дані перед знаходженням викидів в них.

Обробка великих даних, негативно впливає на показник AUC-ROC, можна зробити висновки, що дана функція знаходження викидів в даних погано масштабується.

Підсумовуючи, з графіків 4.1, 4.2 і таблиці 4.1 можна зробити висновки що з даним набором даних функція LOF сервісу Orange впоралась краще, оскільки вона має ліпші показники AUC-ROC для майже всіх експериментів і набагато швидше виконує роботу.

4.1.2 Експеримент №2 – набір даних «Arrhythmia»

Опис набору тестових даних

Набір даних «Arrhythmia» містить інформацію про людей, що мають серцево-судинні захворювання. Кількість атрибутів – 279, кількість об’єктів – 453. Хворі класифікуються по 16-м групам, аномальними вважаються класи з номерами 3-9 и 14-15. Доля аномалій складає 15%.

Результати експерименту

Обчислення показника AUC-ROC для сервісів Rapid Miner та Orange при виконанні різних експериментів наведені у таблиці 4.2.

Таблиця 4.2 – Результати експерименту для набору даних Arrhythmia.

№	Тип даних	Rapid Miner	Orange
1	не нормалізовані дані	0,737	0,581
2	нормалізовані дані	0,571	0,665
3	не нормалізовані збільшені дані	0,519	0,497
4	нормалізовані збільшені дані	0,581	0,539

Графік залежності часу виконання функції LOF від кількості об’єктів в даних побудований на двоядерному процесорі зображено на рисунку 4.3

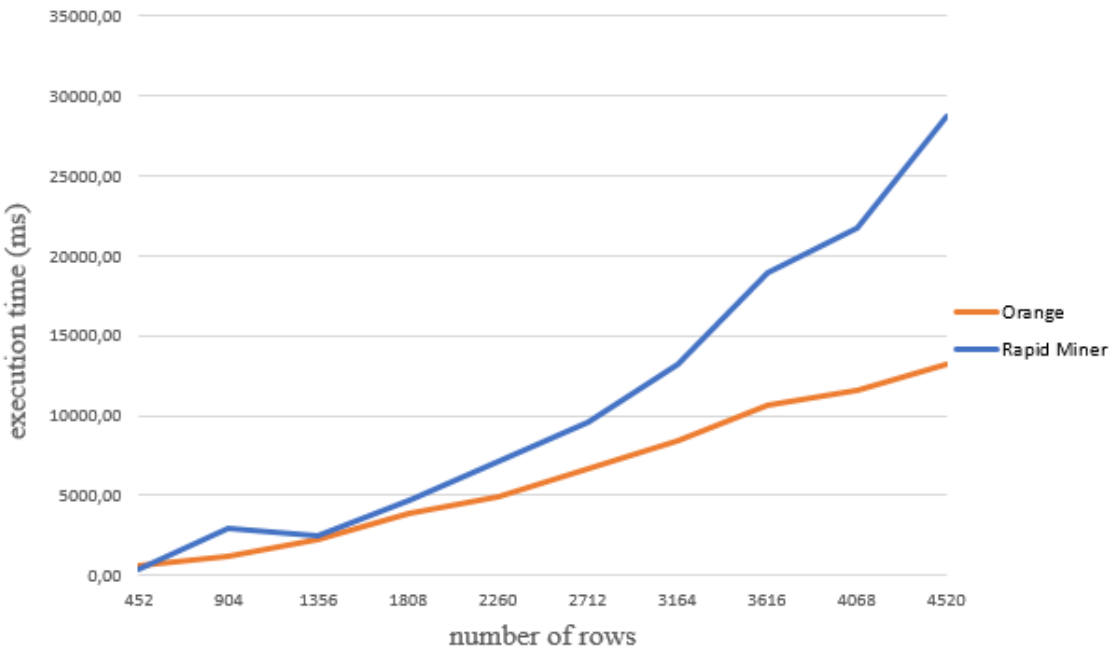


Рисунок 4.3 – Графік залежності побудований на двоядерному процесорі.

Подібний графік було побудовано на одноядерному процесорі для перевірки того, як сервіси можуть виконувати обчислення паралельно.

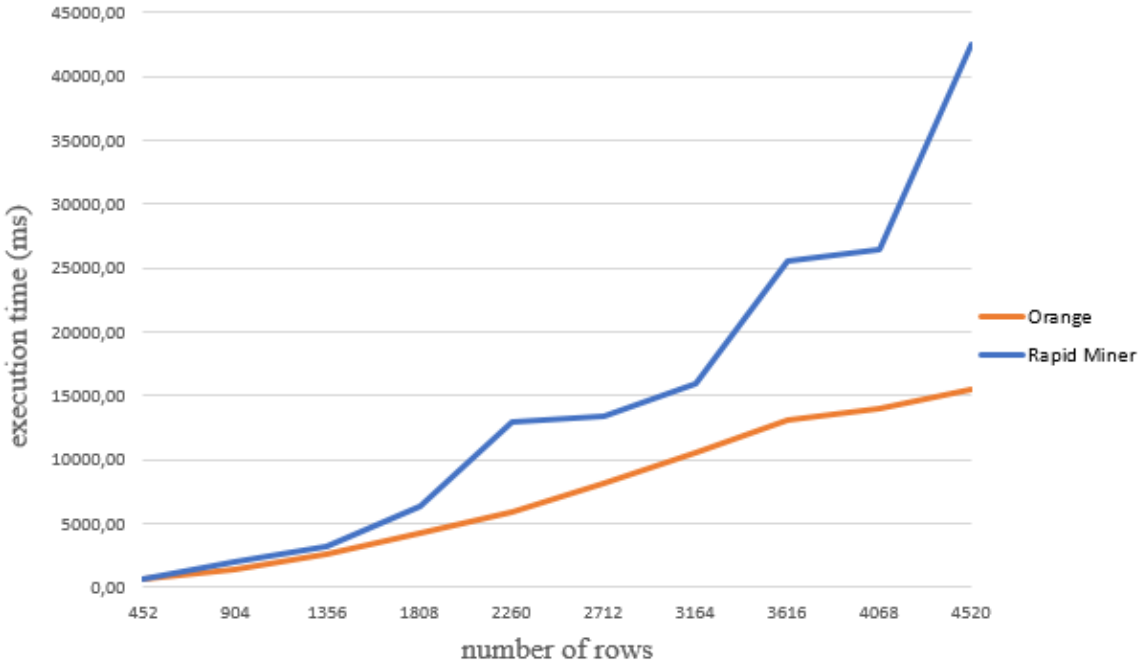


Рисунок 4.4 – Графік залежності побудований на одноядерному процесорі.

Інтерпретація результатів

Порівнюючи результати обох сервісів, отримані для збільшеного об’єму даних варто зазначити, що якісний показник AUC-ROC суттєво зменшився порівняно з оцінкою, що була отримана для початкового набору даних. Це свідчить про неефективність застосування даного алгоритму на великих об’ємах даних.

Аналізуючи вплив нормалізації на якість знаходження викидів в даних також варто відмітити, що для даного експерименту вона позитивно вплинула на всі показники AUC-ROC, окрім випадку її застосування на нормальному об’ємі даних для сервісу Rapid Miner. Порівнюючи дослідження №1 і №2 можемо помітити, як суттєво знизився показник якості для нормалізованих даних. Якщо дотримуватися теорії про те, що сервіс Orange на відміну від сервісу Rapid Miner, завжди використовує нормалізацію при обробці даних, то можна припустити, що у даному випадку вона не потрібна, оскільки негативно впливає на результати.

Виходячи з графіків 4.3 і 4.4 швидкість обробки даних більша для програмного забезпечення Orange. На одноядерному процесорі швидкість незначною мірою погіршується.

4.1.3 Експеримент №3 – набір даних «Breast cancer Wisconsin»

Опис набору тестових даних

Набір даних “Breast cancer Wisconsin” містить дані щодо діагностики і класифікації новоутворень. Таблиця містить десять стовпців з інформацією про стан клітин і останній одинадцятий стовпець, що свідчить про тип ракових клітин. Аномальними значеннями для цього стовбця вважається четвертий клас, що свідчить про злоякісні утворення. Кількість об’єктів у наборі – 699, процент викидів становить 34.5%.

Результати експерименту

Обчислення показника AUC-ROC для сервісів Rapid Miner та Orange при виконанні різних експериментів наведені у таблиці 4.3.

Таблиця 4.3 – Результати експерименту для набору даних «Breast cancer Wisconsin».

№	Тип даних	Rapid Miner	Orange
1	не нормалізовані дані	0,032	0,032
2	нормалізовані дані	0,662	0,662
3	не нормалізовані збільшені дані	0,037	0,037
4	нормалізовані збільшені дані	0,696	0,696

Для аналізу залежності часу виконання функції LOF від кількості об’єктів в даних було побудовано графіки 4.5 і 4.6. Експерименти проводилися на двоядерному та одноядерному процесорах відповідно.

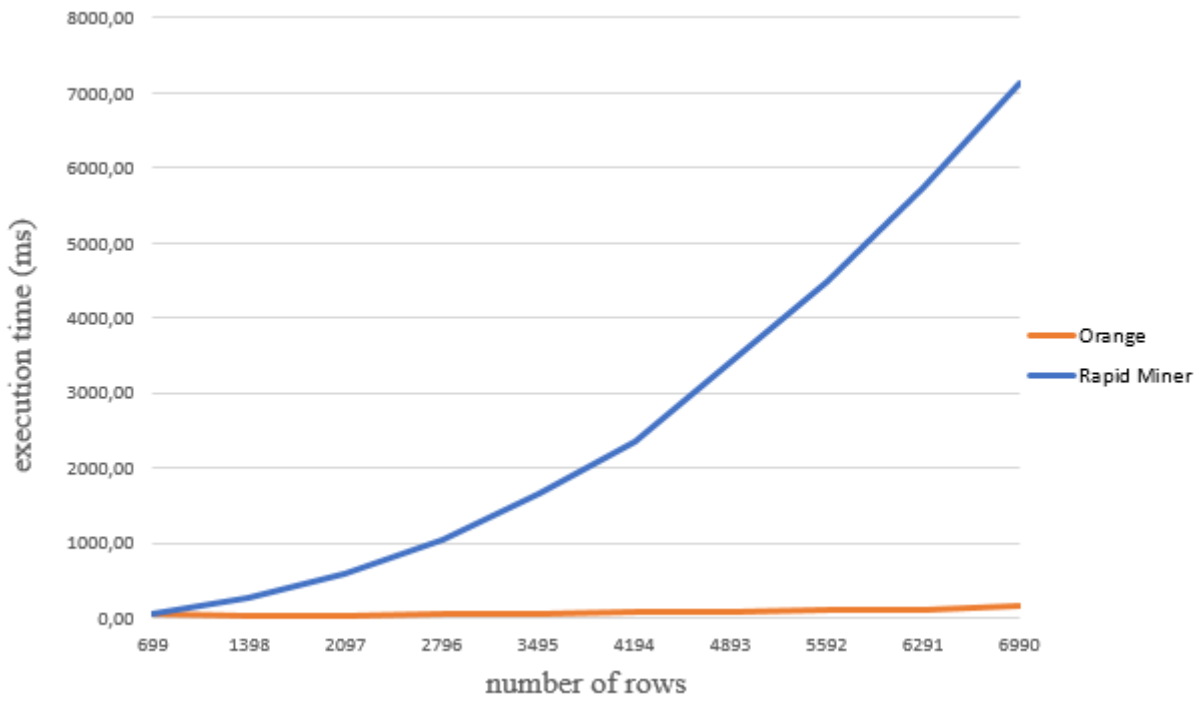


Рисунок 4.5 – Графік залежності часу виконання для даних «Breast cancer Wisconsin» для двоядерного процесору

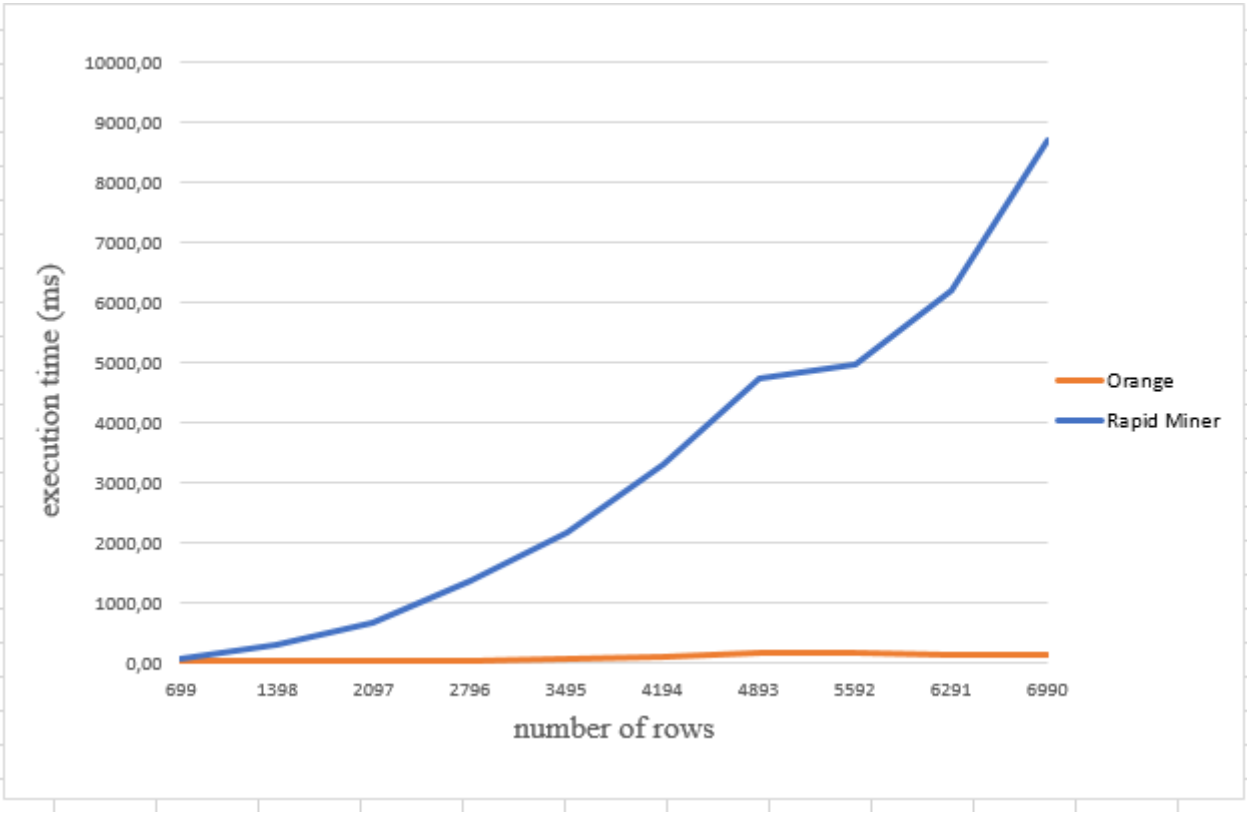


Рисунок 4.6 – Графік залежності часу виконання для даних «Breast cancer Wisconsin» для одноядерного процесору

Інтерпретація результатів

З таблиці 4.3 видно, що для даного набору даних попередня нормалізація негативно вплинула на результати сервісу Orange для всіх досліджень. У випадку з сервісом Rapid Miner ситуація кардинально інша, оскільки як для великих даних так і для звичайних оцінка AUC-ROC після нормалізації стрімко збільшилася. Даний експеримент ще раз підтверджує теорію про те, що Orange має вбудовану стандартизацію даних.

Розглядаючи різницю в якості отриманих викидів для великих даних і звичайних, варто відзначити, що оцінка несуттєво зменшилась для сервісу Orange і так само несуттєво збільшилась на 0,03 для сервісу Rapid Miner.

Для набору даних «Breast cancer Wisconsin» знаходження викидів триває надзвичайно довго при використанні Rapid Miner, це видно з діаграм 4.5 і 4.6.

4.1.4 Експеримент №4 – набір даних «Diabetes»

Опис набору тестових даних

Набір даних «Diabetes» містить інформацію щодо діагностики діабету. Таблиця складається з восьми стовпців, останній з яких містить інформацію щодо аномальності об’єкта. Загалом набір налічує 767 об’єктів, 35% з яких є аномальними.

Результати експерименту

Обчислення показника AUC-ROC для сервісів Rapid Miner та Orange при виконанні різних експериментів наведені у таблиці 4.4.

Таблиця 4.4 – Результати експерименту для набору даних «Diabetes».

№	Тип даних	Rapid Miner	Orange
1	не нормалізовані дані	0,458	0,498
2	нормалізовані дані	0,501	0,493
3	не нормалізовані збільшені дані	0,49	0,505
4	нормалізовані збільшені дані	0,513	0,498

Графік залежності часу виконання функції LOF від кількості об’єктів в даних зображено на рисунку 4.7

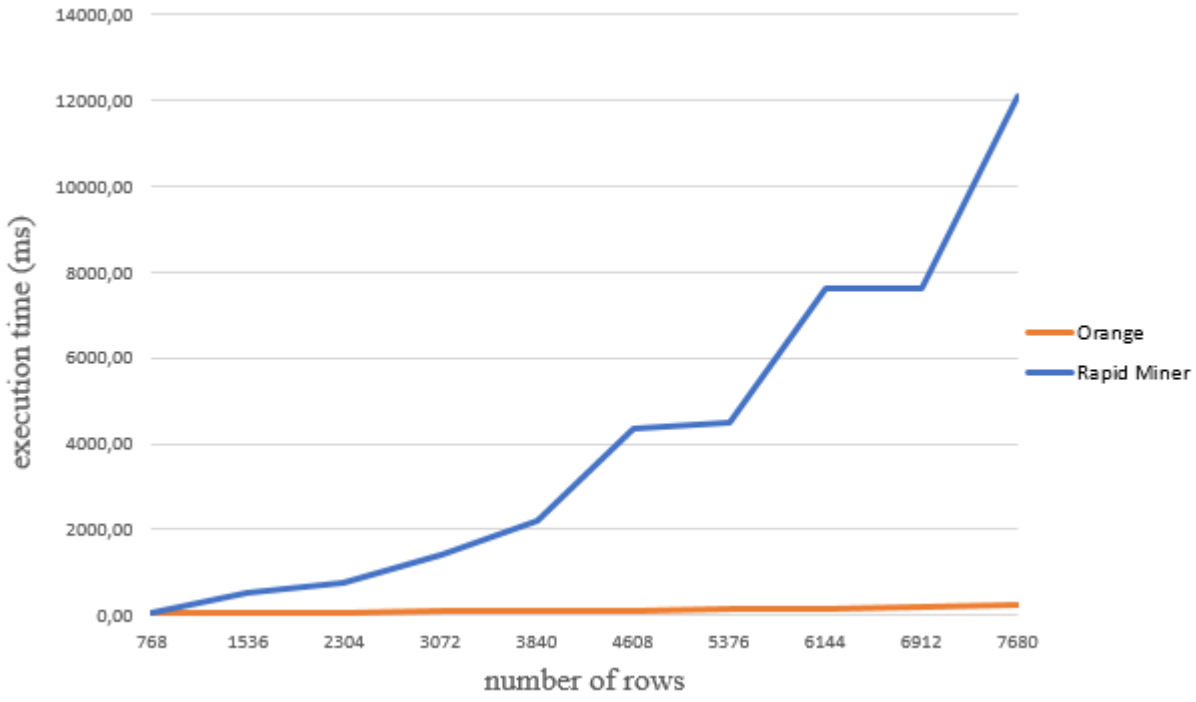


Рисунок 4.7 – Графік залежності часу виконання для даних «Diabetes» для двоядерного процесору.

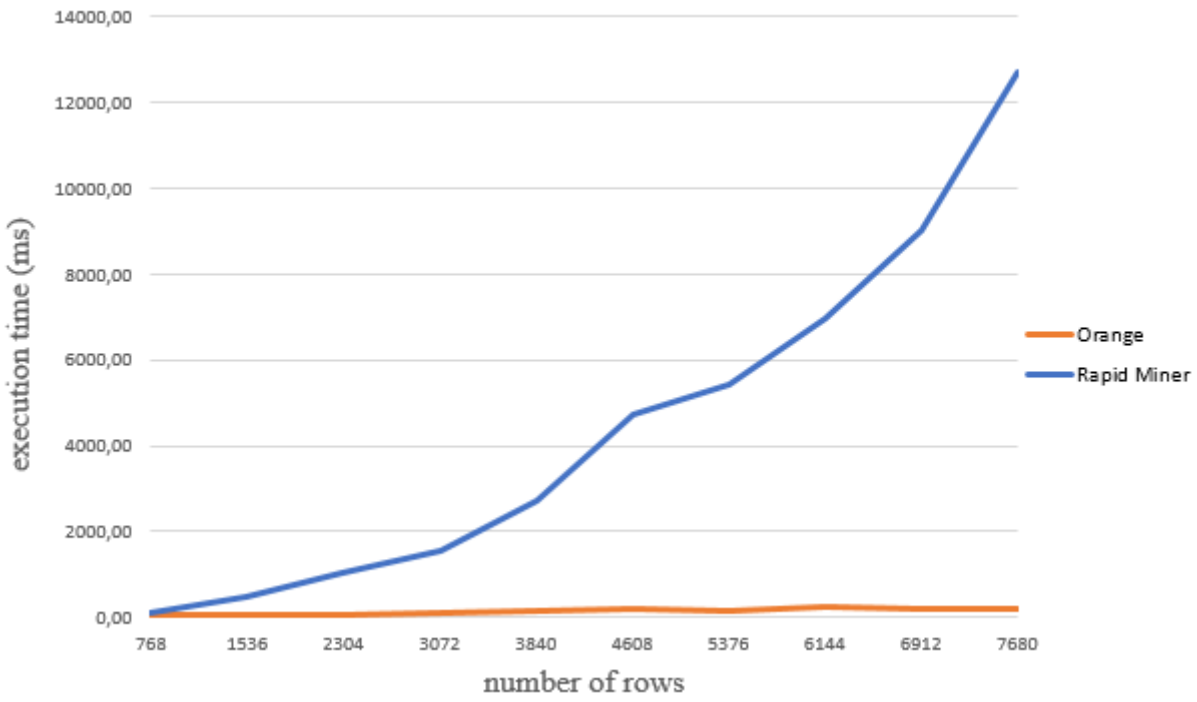


Рисунок 4.8 – Графік залежності часу виконання для даних «Diabetes» для одноядерного процесору.

Інтерпретація результатів

Беручи до уваги результати, описані в таблиці 4.4 можна підсумувати, що для даного набору даних при використанні сервісу Orange попередня нормалізація не потрібна. Для досліджень виконаних сервісом Rapid Miner попередня нормалізація трохи підвищила показник якості. На великих даних якість знаходження викидів залишилась приблизно такою самою.

Аналізуючи графіки 4.7 і 4.8 можна прийти до висновку, що на одноядерному процесорі обробка виконується трохи повільніше, тенденція зросту часу залишається тією самою.

4.1.5 Висновки до аналізу функції знаходження викидів

Під часу аналізу функції LOF сервісів Orange і Rapid Miner були зроблені наступні висновки:

- швидкість обробки даних сервісом Rapid Miner більш ніж вдвічі гірша за швидкість обробки даних сервісом Orange;
- на одноядерному процесорі дані обробляються повільніше ніж на двоядерному, але різниця в часі не велика;

- якість обробки інформації падає на великих об’ємах даних (від 1 Мб);
- попередня нормалізація негативно впливає на результати сервісу Orange і позитивно на якість результатів сервісу Rapid Miner;
- найвищу оцінку якості при знаходженні викидів можна досягнути попередньо нормалізувавши дані для сервісу Rapid Miner;

Підсумовуючи, можна стверджувати, що показники ефективності функції LOF для сервісу Orange є більш стабільними, оскільки дають задовільні оцінки для різних експериментів. Також швидкість обробки даних сервісом Orange є набагато вищою.

4.2 Порівняння функції лінійної регресії і передбачення

Оскільки програма «Data Mining Service Analyzer» використовує функцію передбачення, що базується на моделі лінійної регресії, то досліджуватися буде саме якість отриманих результатів для функції лінійної регресії.

Для порівняння результатів функцій лінійної регресії програмним забезпеченням «Data Mining Service Analyzer» були підраховані метрики для оцінки якості регресійної моделі:

- коефіцієнт кореляції;
- середня квадратична помилка;
- корінь середньої квадратичної помилки;
- абсолютна помилка.

Також були зроблені заміри часу виконання функції і побудовані графіки для відтворення залежності часу виконання функції від об’єму даних на двоядерному та одноядерному процесорах.

Показники були розраховані для наступних даних:

- нормалізований набір даних;
- нормалізований набір даних з видаленням викидів;
- набір даних з видаленням викидів;
- не нормалізований набір даних без видалення викидів;

Ті ж самі дії були виконані на наборі даних, що є збільшеним від початкового у 20 разів.

Для порівняння сервісів було проведено чотири експерименти на різних даних. Кожний з наборів даних був поділений на дві вибірки: навчальна та контрольна. Після того, як модель була навчена на вибірці навчальних даних, були підраховані показники якості отримані для контрольної вибірки. У наступних розділах описані експерименти і їх результати.

4.2.1 Експеримент №1 – набір даних «Fish»

Опис набору тестових даних

Набір даних «Fish» – містить інформацію про розмір риби:

- вертикальна довжина;
- довжина по діагоналі;
- довжина поперек;
- висота;
- ширина;
- ширина по діагоналі;
- вага риби.

Усі дані окрім ваги подані у сантиметрах. Загалом таблиця містить 7 атрибутів і 160 сутностей. Половина даних з вибірки використана в якості навчальних даних, інша половина – контрольні дані.

Результати експерименту

Результати експерименту для початкового набору даних наведені у таблиці 4.5.

Таблиця 4.5 – Результати для набору даних «Fish».

	R2		RMSE		MSE		MAE	
	O.	RP.	O.	RP.	O.	RP.	O.	RP.
1	2	3	4	5	6	7	8	9
нормалізований набір даних	0.952	0.952	0.219	0.218	0.048	0.047	0.161	0.159

Продовження таблиці 4.5.

1	2	3	4	5	6	7	8	9
нормалізований набір даних з видаленням викидів	0.952	0.941	0.219	0.239	0.048	0.057	0.161	0.158
набір даних з видаленням викидів	0.952	0.941	0.285	0.154	0.082	0.024	0.21	0.119
не нормалізований набір даних без видалення викидів	0.952	0.952	0.285	0.286	0.082	0.082	0.21	0.209

Результати експерименту для набору даних збільшеного у 300 разів (23400 рядків) наведені у таблиці 4.6.

Таблиця 4.6 – Результати для збільшеного набору даних «Fish»

	R2		RMSE		MSE		MAE	
	O.	RP.	O.	RP.	O.	RP.	O.	RP.
нормалізований набір даних	0,959	0.952	0,203	0,219	0,041	0,048	0,15	0,16
нормалізований набір даних з видаленням викидів	0,959	0,952	0,203	0,219	0,041	0,048	0,15	0,16
набір даних з видаленням викидів	0,959	0,952	0,256	0,286	0,07	0,082	0,196	0,209
не нормалізований набір даних без видалення викидів	0,959	0,952	0,256	0,286	0,07	0,082	0,196	0,209

Було побудовано графік залежності часу виконання функції від об’єму даних для сервісів Orange і Rapid Miner на двоядерному процесорі (рис. 4.9) та одноядерному процесорі (рис. 4.10).

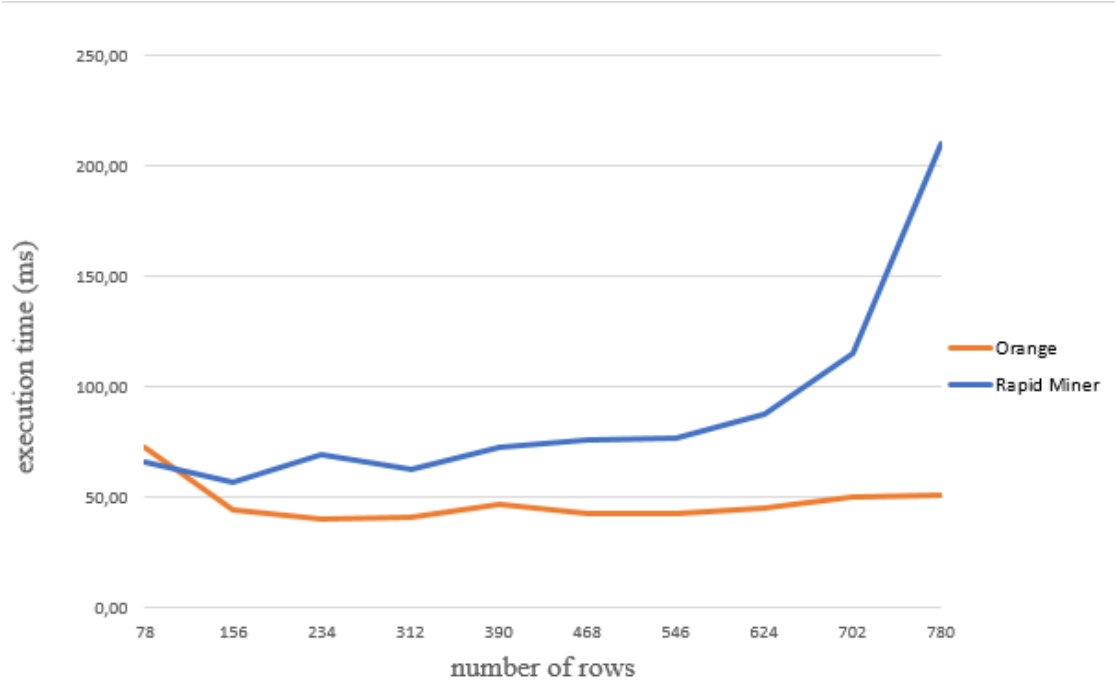


Рисунок 4.9 – Графік часової залежності для двоядерного процесору

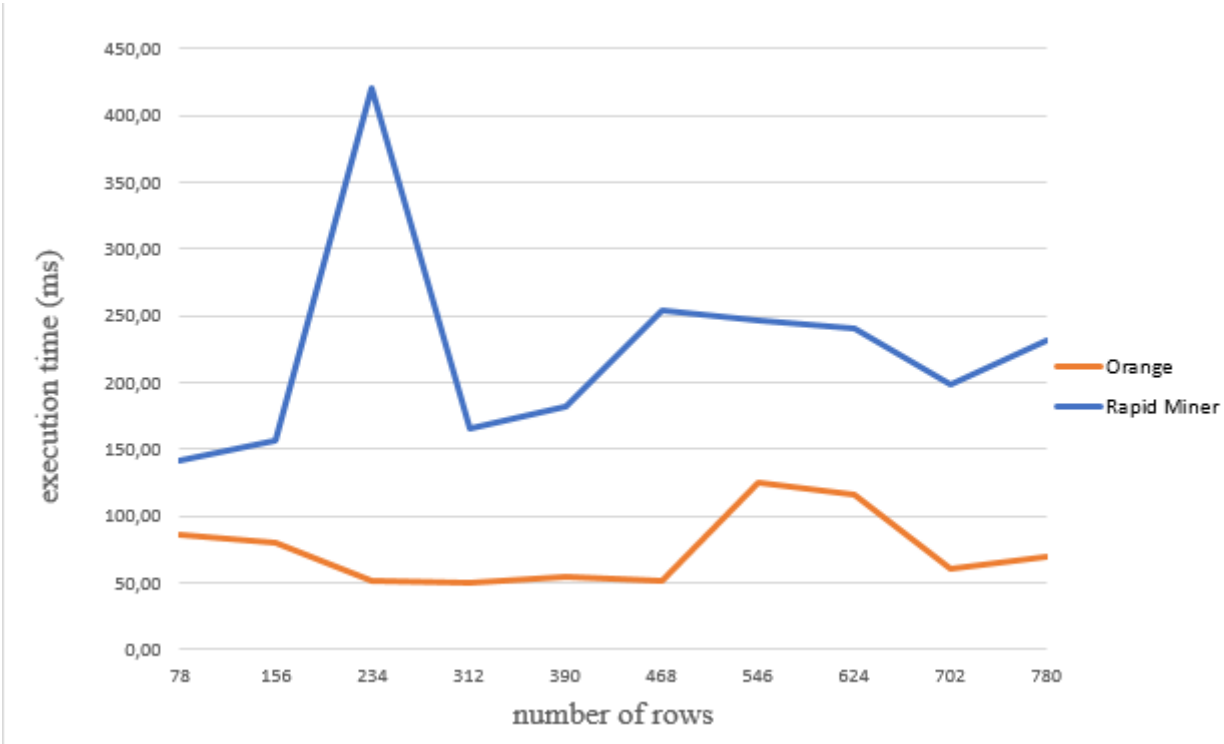


Рисунок 4.10 – Графік часової залежності для одноядерного процесору

Інтерпретація результатів

В ході аналізу показників було виявлено, що для даного набору даних правдиві наступні твердження:

- нормалізація даних і видалення викидів не впливає на якість результатів, отриманих сервісом Orange;
- нормалізація даних і видалення викидів негативно впливає на якість результатів, отриманих сервісом Rapid Miner;
- збільшення даних у наборі не впливає на якість рівняння лінійної регресії;
- часові показники сервісу Orange суттєво ліпші за показники Rapid Miner для обох типів процесорів.

4.2.2 Експеримент №2 – набір даних «Red Wine»

Опис набору тестових даних

Набір даних «Red Wine» – містить інформацію про хімічний склад вина і його оцінку за десяти бальною шкалою. Таблиця містить 12 атрибутів і 1599 сутностей. Половина даних з вибірки використана в якості навчальних даних, інша половина – контрольні дані. Мета задачі полягає у знаходженні залежності якості вина від його хімічного складу.

Результати експерименту

Результати експерименту для початкового набору даних наведені у таблиці 4.7.

Таблиця 4.7 – Результати для набору даних «Red Wine».

	R2		RMSE		MSE		MAE	
	O.	RP.	O.	RP.	O.	RP.	O.	RP.
1	2	3	4	5	6	7	8	9
нормалізований набір даних	0.347	0.36	0.808	0.8	0.347	0.36	0.808	0.8
нормалізований набір даних з видаленням викидів	0.356	0.359	0.803	0.8	0.356	0.359	0.803	0.8

Продовження таблиці 4.7

1	2	3	4	5	6	7	8	9
набір даних з видаленням викидів	0.356	0.359	0.649	0.648	0.356	0.359	0.649	0.648
не нормалізований набір даних без видалення викидів	0.652	0.646	0.505	0.501	0.652	0.646	0.505	0.501

Результати експерименту для набору даних збільшеного у 10 разів (15990 рядків) наведені у таблиці 4.8.

Таблиця 4.8 – Результати для збільшеного набору даних «Red Wine»

	R2		RMSE		MSE		MAE	
	O.	RP.	O.	RP.	O.	RP.	O.	RP.
нормалізований набір даних	0.36	0.361	0.8	0.8	0.64	0.36	0.361	0.8
нормалізований набір даних з видаленням викидів	0.364	0.361c	0.798	0.799	0.636	0.364	0.361c	0.798
набір даних з видаленням викидів	0.364	0.361	0.64	0.645	0.409	0.364	0.361	0.64
не нормалізований набір даних без видалення викидів	0.36	0.361	0.646	0.646	0.417	0.36	0.361	0.646

Також було побудовано графік залежності часу виконання функції від об’єму даних на двоядерному процесорі (рис. 4.11) та одноядерному процесорі (рис. 4.12).

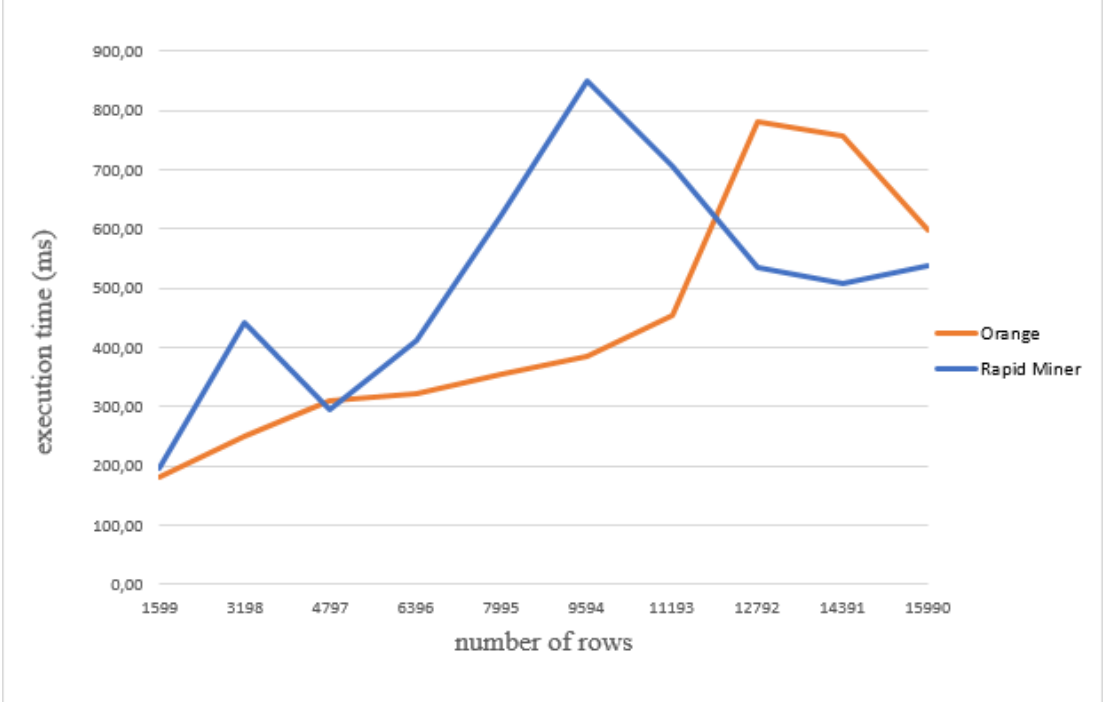


Рисунок 4.11 – Графік часової залежності для двоядерного процесора

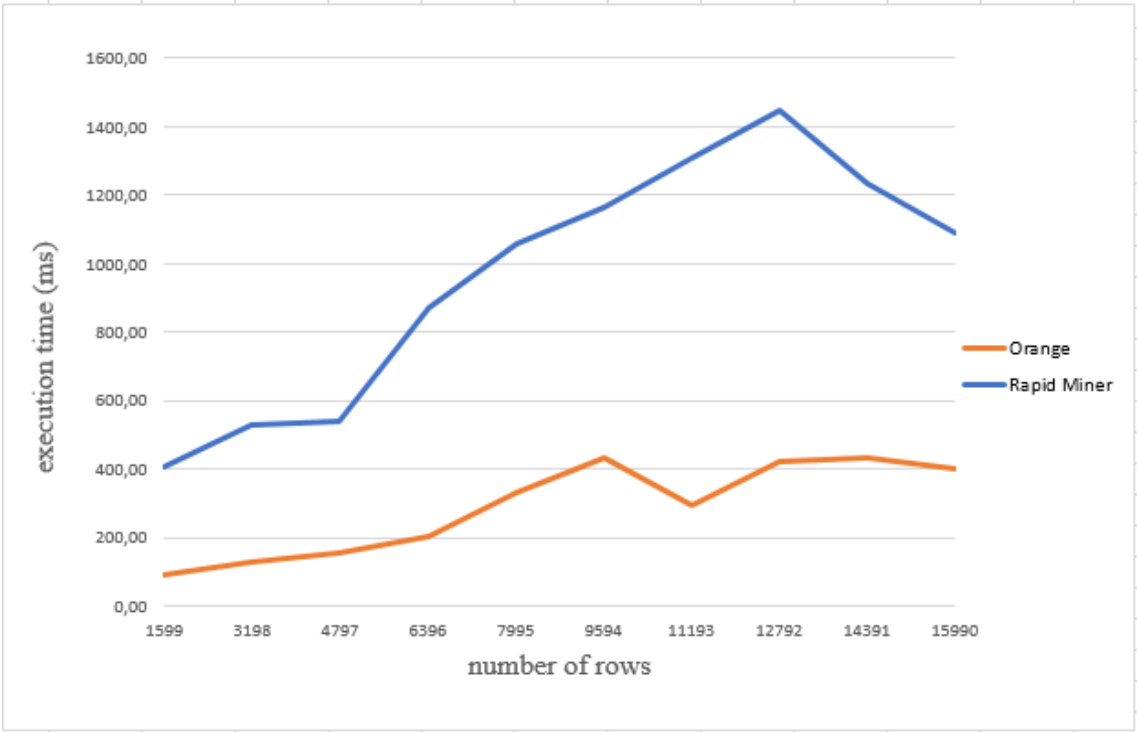


Рисунок 4.12 – Графік часової залежності для одноядерного процесору

Інтерпретація результатів

Дані приведені в таблицях 4.6 і 4.7 свідчать про наступні особливості поведінки сервісів на наборі даних «Red wine»::

– нормалізація даних і видалення викидів не впливає на якість результатів, отриманих сервісом Orange;
– нормалізація даних і видалення викидів негативно впливає на якість результатів, отриманих сервісом Rapid Miner;
– збільшення даних у наборі не впливає на якість рівняння лінійної регресії;
– якість регресійної моделі вдвічі гірша на великих даних;

Аналізуючи графіки на рисунку 4.11 і 4.12 варто зазначити, що для одноядерного процесору дані обробляються повільніше.

4.2.3 Експеримент №3 – набір даних «Breast Cancer»

Опис набору тестових даних

Набір даних «Breast Cancer» вже був описаний у попередніх експериментах при дослідженні ефективності функції LOF. Мета задачі полягає у виявлені типу новоутворення керуючись наведеними в таблиці даними.

Результати експерименту

Результати експерименту для початкового набору даних наведені у таблиці 4.9.

Таблиця 4.9 – Результати для набору даних «Breast Cancer»

	R2		RMSE		MSE		MAE	
	O.	RP.	O.	RP.	O.	RP.	O.	RP.
1	2	3	4	5	6	7	8	9
нормалізований набір даних	0.832	0.843	0.41	0.396	0.168	0.832	0.843	0.41
нормалізований набір даних з видаленням викидів	0.836	0.586	0.405	0.634	0.164	0.836	0.586	0.405
набір даних з видаленням викидів	0.836	0.586	0.386	0.375	0.149	0.836	0.586	0.386

Прдовження таблиці 4.9

1	2	3	4	5	6	7	8	9
не нормалізований набір даних без видалення викидів	0.832	0.843	0.391	0.378	0.153	0.832	0.843	0.391

Результати експерименту для набору даних збільшеного у 10 разів (15990 рядків) наведені у таблиці 4.10.

Таблиця 4.10 Результати для збільшеного набору даних «Breast Cancer»

	R2		RMSE		MSE		MAE	
	O.	RP.	O.	RP.	O.	RP.	O.	RP.
нормалізований набір даних	0.79	0.804	0.459	0.442	0.21	0.195	0.79	0.804
нормалізований набір даних з видаленням викидів	0.788	0.573	0.46	0.642	0.212	0.413	0.788	0.573
набір даних з видаленням викидів	0.788	0.593	0.459	0.389	0.211	0.151	0.788	0.593
не нормалізований набір даних без видалення викидів	0.79	0.804	0.457	0.441	0.209	0.195	0.79	0.804

Нижче представлено графік залежності часу виконання функції від об’єму даних на двоядерному процесорі (рис. 4.13) та одноядерному процесорі (рис. 4.14).

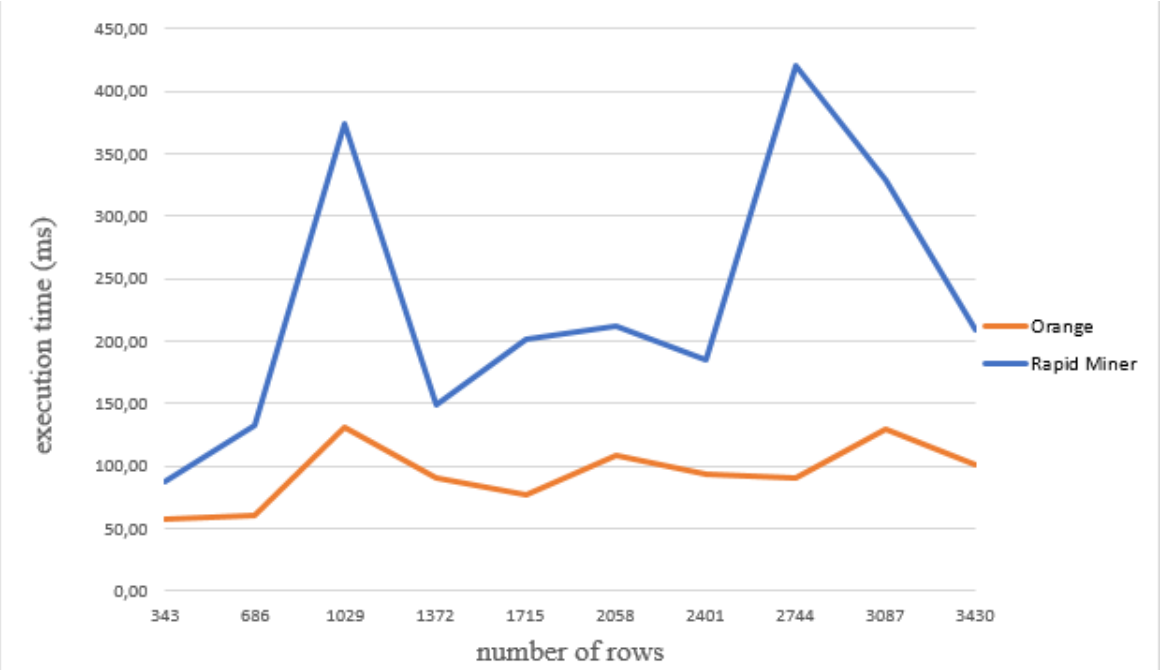


Рисунок 4.13 – Графік часової залежності для двоядерного процесору

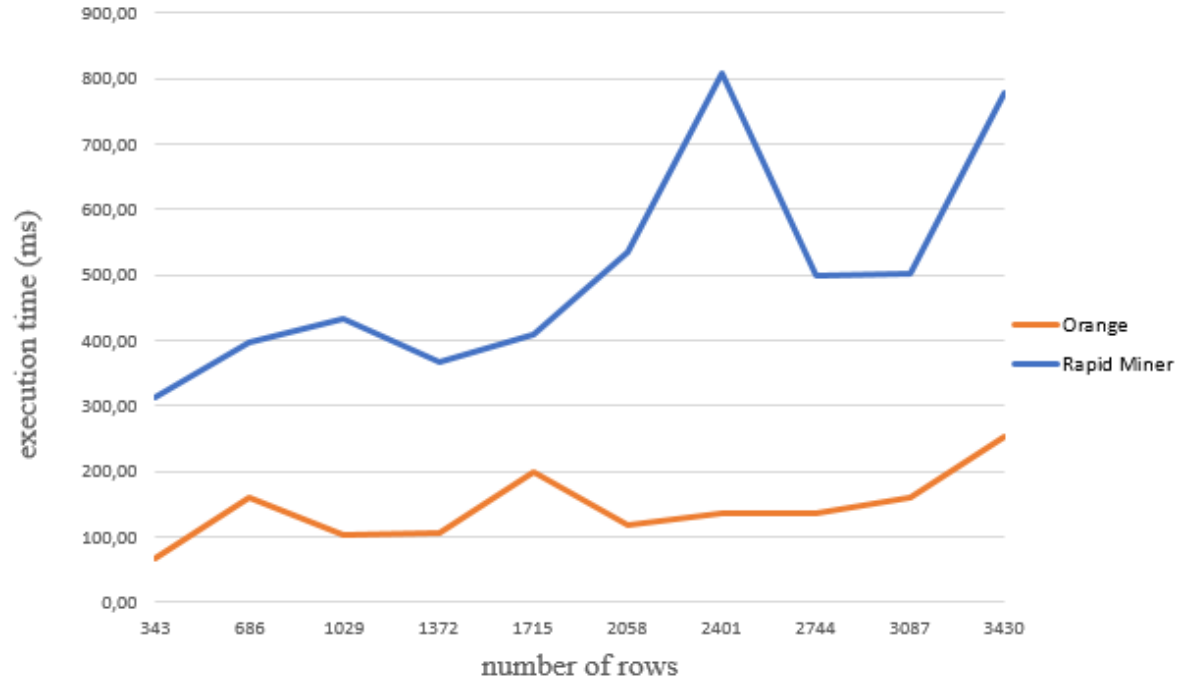


Рисунок 4.13 – Графік часової залежності для одноядерного процесору

Інтерпретація результатів

Після аналізу даних наведених у таблиці 4.9 і 4.10 можна зробити наступні висновки:

– функція сервісу Orange стійкі до викидів і ненормалізованих даних;
– показники сервісу Orange кращі за показники сервісу Rapid Miner;
– видалення викидів негативно впливає на показники Rapid Miner;

– показники якості незначною мірою погіршуються на великих об’ємах даних.

Аналізуючи графіки 4.13 та 4.14 варто зазначити наступне:

– сервіс Orange має кращі показники часової ефективності на всіх наборах даних;
– швидкість обробки даних на одноядерному процесорі менша, тенденція залишається тією самою.

4.2.4 Експеримент №4 – набір даних «Machine»

Опис набору тестових даних

Набір даних «Machine» містить інформацію про CPU. Мета задачі полягає у виявлені залежності продуктивності CPU від характеристик процесора. Таблиця складається з 8 атрибутів і 209 екземплярів. Вибірка поділена навпіл на контрольну і навчальну.

Результати експерименту

Результати експерименту для початкового набору даних «Machine» наведені у таблиці 4.11.

Таблиця 4.11 – Результати для набору даних «Machine»

	R2		RMSE		MSE		MAE	
	O.	RP.	O.	RP.	O.	RP.	O.	RP.
1	2	3	4	5	6	7	8	9
нормалізований набір даних	0.899	0.952	0.317	0.217	0.101	0.047	0.168	0.142
нормалізований набір даних з видаленням викидів	0.975	0.955	0.157	0.212	0.025	0.045	0.119	0.139
набір даних з видаленням викидів	0.975	0.954	13.851	38.155	191.86	1455.779	10.472	25.299

Продовження таблиці 4.11

1	2	3	4	5	6	7	8	9
не нормалізований набір даних без видалення викидів	0.899	0.952	50.116	34.433	2511.639	1185.637	26.55	22.618

Результати експерименту для набору даних збільшеного у 100 разів (20900 рядків) наведені у таблиці 4.12.

Таблиця 4.12 – Результати для збільшеного набору даних «Machine»

	R2		RMSE		MSE		MAE	
	O.	RP.	O.	RP.	O.	RP.	O.	RP.
нормалізований набір даних	0.959	0.959	0.202	0.201	0.041	0.041	0.129	0.129
нормалізований набір даних з видаленням викидів	0.959	0.959	0.202	0.201	0.041	0.041	0.129	0.129
набір даних з видаленням викидів	0.959	0.959	31.244	31.087	976.176	966.423	19.906	19.852
не нормалізований набір даних без видалення викидів	0.959	0.959	31.244	31.087	976.176	966.423	19.906	19.852

Під час експерименту побудовано графік залежності часу виконання функції від об’єму даних на двоядерному та одноядерному процесорах (рис. 4.15, 4.16).

Рисунок 4.15 – Графік часової залежності для двоядерного процесору

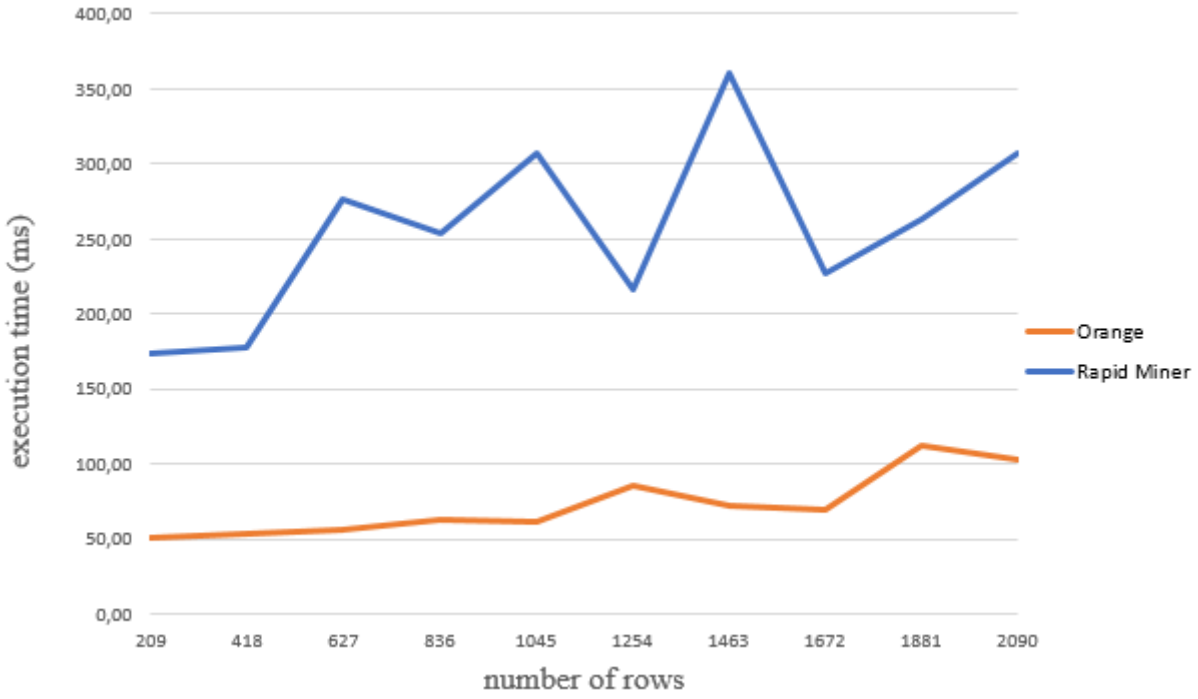


Рисунок 4.16 – Графік часової залежності для одноядерного процесору

Інтерпретація результатів

Аналізуючи данні наведені у таблиці 4.11 і 4.12 можна прийти до наступних висновків:

– видалення викидів позитивно впливає на якість отриманих результатів від Orange;

– якість результатів несуттєво зменшилась на великих даних;

– без нормалізації даних похибка є дуже високою.

Графіки зображені на рисунках 4.15 і 4.16 свідчать про наступне:

– сервіс Orange має кращі показники часової ефективності ніж Rapid Miner.
– час обробки даних збільшується для одноядерного процесору.

4.2.5 Висновки до аналізу функції функцій передбачення і лінійної регресії

Під час аналізу функції для знаходження коефіцієнтів рівняння лінійної регресії сервісів Orange і Rapid Miner були зроблені наступні висновки:

– швидкість обробки даних набагато краща у сервіса Orange;
– нормалізація даних та видалення викидів може як позитивно так і негативно впливати на отримані результати, користь від використання даних опцій залежить від набору даних;
– показники якості рівняння знижуються на великих даних;
– оцінки якості рівнянь лінійної регресії обох сервісів є приблизно однаковими для ненормалізованих даних і суттєво різняться для даних що були попередньо оброблені.

4.3 Аналіз функцій для знаходження кластерів методом k-середніх

Для порівняння реалізацій функцій сервісів Orange і Rapid Miner були підібрані наступні критерії оцінки якості кластерів:

– індекс Девіса-Болдуіна;
– показник Сілуетте;
– індекс Калінські-Харабаш.

Додатковим параметром оцінки ефективності функції сервісу є швидкість обробки даних з різними об’ємами.

Експерименти для порівняння сервісів на ефективність функції k-means складаються з наступних етапів:

– порівняння показників на нормалізованих даних;
– порівняння показників на не нормалізованих даних;
– порівняння показників на нормалізованих збільшених даних;
– порівняння показників на не нормалізованих збільшених даних;

– порівняння часу обробки даних на різних об’ємах даних та на двоядерному, одноядерному процесорах.

Наведені вище етапи експериментів дозволять дати відповідь на наступні питання:

– який з сервісів дає кращі кластери;
– як об’єм даних впливає на показники якості кластерів і який з сервісів здійснює кращу обробку на великих даних;
– чи реалізована в функції нормалізація даних, і як вона впливає на результати;
– який з сервісів виконує кластеризацію швидше.

Описані вище дії будуть засновані на трьох наборах даних, що дозволить зробити об’єктивні висновки.

4.3.1 Експеримент №1 – набір даних «Iris»

Опис набору тестових даних

Набір даних «Iris» містить інформацію про розміри квітки іриса. У наборі представлено 3 класи ірисів метою кластеризації є поділити об’єкти на класи за розмірами квітки. Таблиця складається з 150 об’єктів і 4 атрибутів.

Результати експерименту

Підрахунки коефіцієнтів для набору даних «Iris» наведені у таблиці 4.13.

Таблиця 4.13 – коефіцієнти якості для набору даних «Iris»

	індекс Девіса-Болдуіна		показник Сілуетте		індекс Калінські-Харабаш	
	O.	RP.	O.	RP.	O.	RP.
нормалізований набір даних	0.739	0.764	0.506	0.497	504	484
не нормалізований набір даних	560	560	0.662	0.666	0.553	0.551

Підрахунки коефіцієнтів для збільшеного у 30 разів набору даних «Iris» наведені у таблиці 4.14.

Таблиця 4.14 – коефіцієнти якості для збільшеного набору даних «Iris»

	індекс Девіса-Болдуіна		показник Сілуетте		індекс Калінські-Харабаш	
	О.	RP.	О.	RP.	О.	RP.
нормалізований набір даних	0.739	0.764	0.515	0.506	30901	29671
не нормалізований набір даних	0.662	0.666	0.561	0.56	34298	34296

Під час проведення експерименту було побудовано графік залежності часу виконання функції кластеризації від об’єму даних для набору «Iris». Дослідження проводились на двоядерному (рис. 4.17) та одноядерному (рис.4.18).

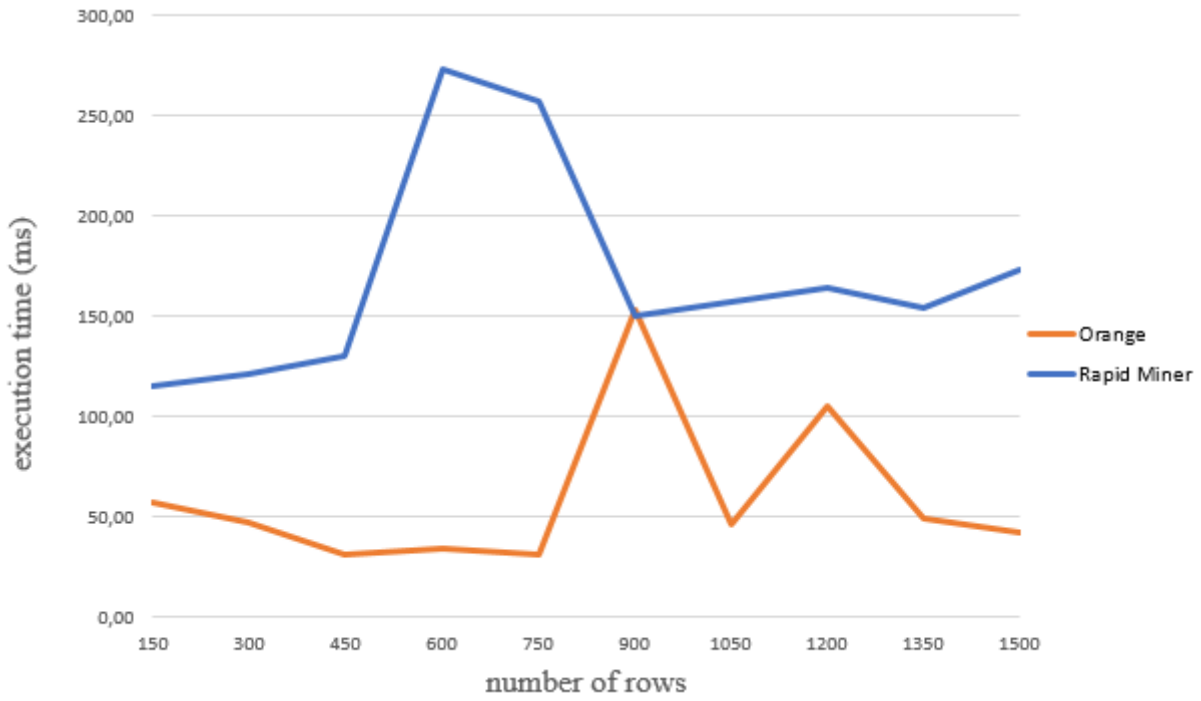


Рисунок 4.17 – Графік часу виконання функції k-means на двоядерному процесорі

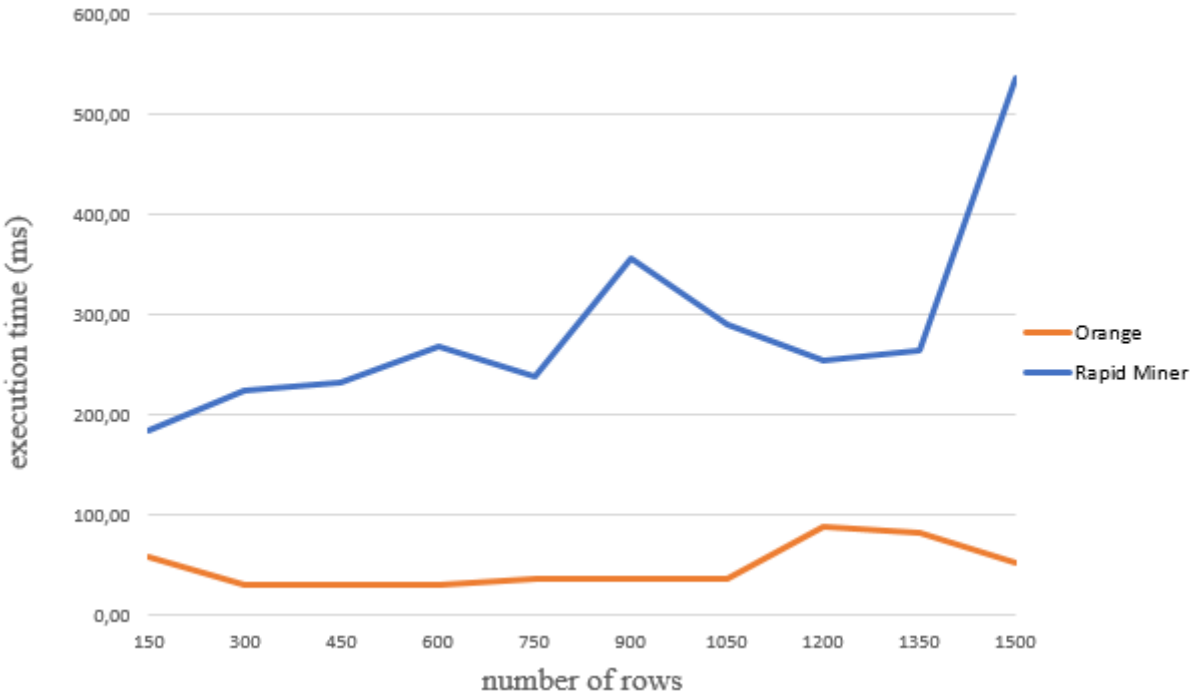


Рисунок 4.18 – Графік часу виконання функції k-means на одноядерному процесорі

Інтерпретація результатів

Проаналізувавши показники наведені у таблицях 4.13 і 4.14 можна прийти до наступних висновків:

- показники якості кластеризації кращі на малих даних;
- нормалізація даного набору даних негативно впливає на якість отриманих кластерів;
- метрики дають кращі результати для сервісу Orange.
- швидкість обробки даних у сервіса Rapid Miner значно нижча.

4.3.2 Експеримент №2 – набір даних «Wine quality»

Опис набору тестових даних

Набір даних «Wine quality» вже був описаний раніше при проведенні експериментів на знаходження лінійної регресії. Набір містить інформацію про хімічний склад вина, кожний екземпляр має свою якість. Задача полягає у розподілені вина на кластери за якістю, усього – 6 кластерів.

Результати експерименту

Підрахунки показників якості для набору даних «Wine quality» наведені у таблиці 4.15.

Таблиця 4.15 – коефіцієнти якості для набору даних «Wine quality»

	індекс Девіса-Болдуіна		показник Сілуетте		індекс Калінські-Харабаш	
	O.	RP.	O.	RP.	O.	RP.
нормалізований набір даних	4.051	3.99	-0.036	-0.037	320	322
не нормалізований набір даних	0.65	0.65	0.44	0.44	3103	3103

Підрахунки коефіцієнтів для збільшеного у 10 разів набору даних «Wine quality» наведені у таблиці 4.16.

Таблиця 4.16 – коефіцієнти якості для збільшеного набору даних «Wine quality»

	індекс Девіса-Болдуіна		показник Сілуетте		індекс Калінські-Харабаш	
	O.	RP.	O.	RP.	O.	RP.
нормалізований набір даних	3.98	4	-0.034	-0.033	3201	3211
не нормалізований набір даних	0.652	0.652	0.44	0.44	31135	31136

Під час проведення експерименту було побудовано графік залежності часу виконання функції кластеризації від об’єму даних. Дослідження проводились на двоядерному (рис. 4.19) та одноядерному (рис.4.20).

Інтерпретація результатів

Показники наведені у таблицях 4.15 і 4.16 свідчать про наступні особливості сервісів:

- показники якості кластеризації отримані на малих і великих даних несуттєво відрізняються;
- нормалізація даного набору даних значно погіршує якість отриманих кластерів;

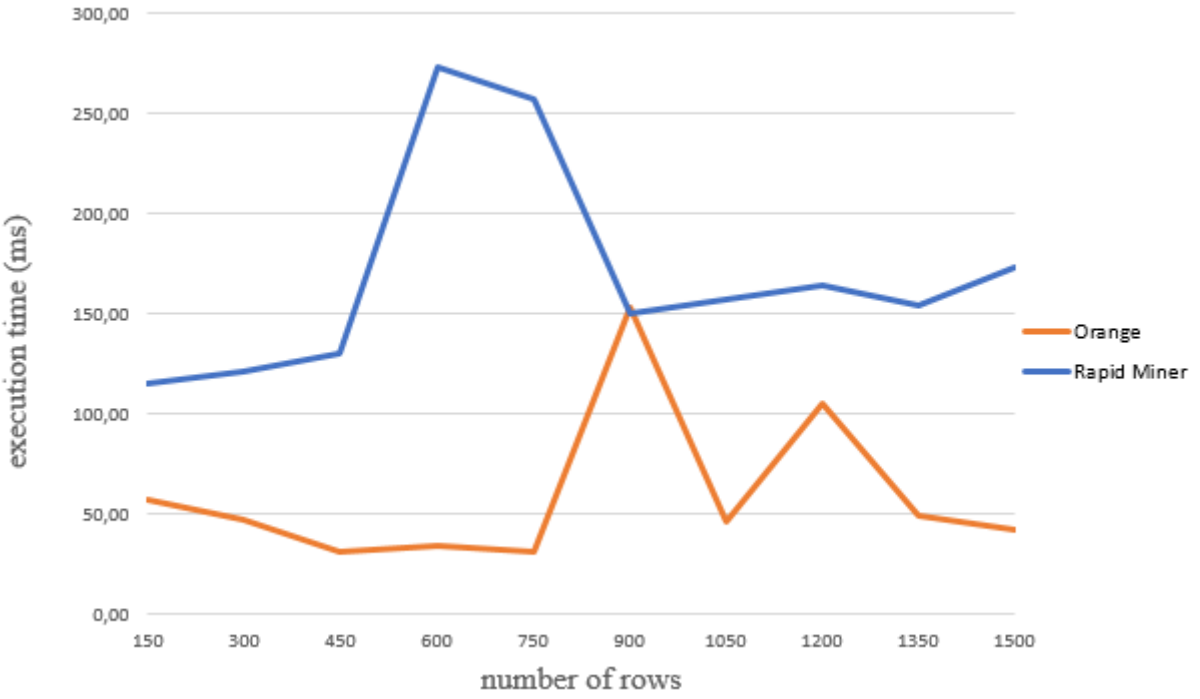


Рисунок 4.19 – Графік часу виконання кластеризації для двох ядер

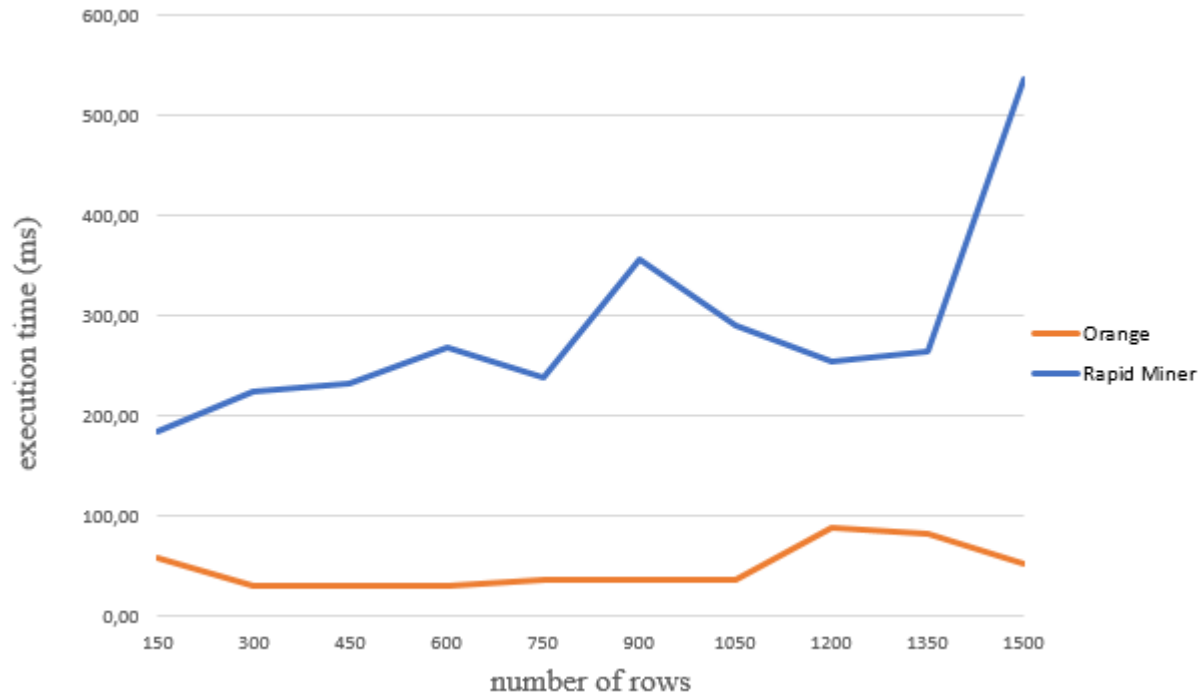


Рисунок 4.20 – Графік часу виконання кластеризації для одного ядра

– метрики дають приблизно однакову оцінку якості результатів для сервісу Orange і Rapid Miner:

– швидкість обробки даних у сервіса Orange є лінійною, в той час як сервіс Rapid Miner може необґрунтовано довго обробляти дані.

4.3.3 Експеримент №3 – набір даних «Wine»

Опис набору тестових даних

Набір даних «Wine» містить інформацію про хімічний склад вина трьох різних сортів. Таблиця містить 12 атрибутів в 179 екземплярів. Задача полягає у розподілені вина на три кластери за сортами.

Результати експерименту

Підрахунки коефіцієнтів для набору даних «Wine» наведені у таблиці 4.17.

Таблиця 4.17 – коефіцієнти якості для набору даних «Wine»

	індекс Девіса-Болдуіна		показник Сілуетте		індекс Калінські-Харабаш	
	O.	RP.	O.	RP.	O.	RP.
нормалізований набір даних	1.52	1.21	0.19	0.2	199	172
не нормалізований набір даних	0.53	0.55	0.57	0.56	562	497

Підрахунки коефіцієнтів для збільшеного у 60 разів набору даних «Wine» наведені у таблиці 4.18.

Таблиця 4.18 – коефіцієнти якості для збільшеного набору даних «Wine»

	індекс Девіса-Болдуіна		показник Сілуетте		індекс Калінські-Харабаш	
	O.	RP.	O.	RP.	O.	RP.
нормалізований набір даних	1.52	1.52	0.2	0.2	12159	12159
не нормалізований набір даних	0.53	0.55	0.58	0.57	34277	30322

Під час проведення експерименту було побудовано графік залежності часу виконання функції кластеризації від об’єму даних для набору «Wine». Дослідження проводились на двоядерному (рис. 4.21 та одноядерному (рис.4.22).

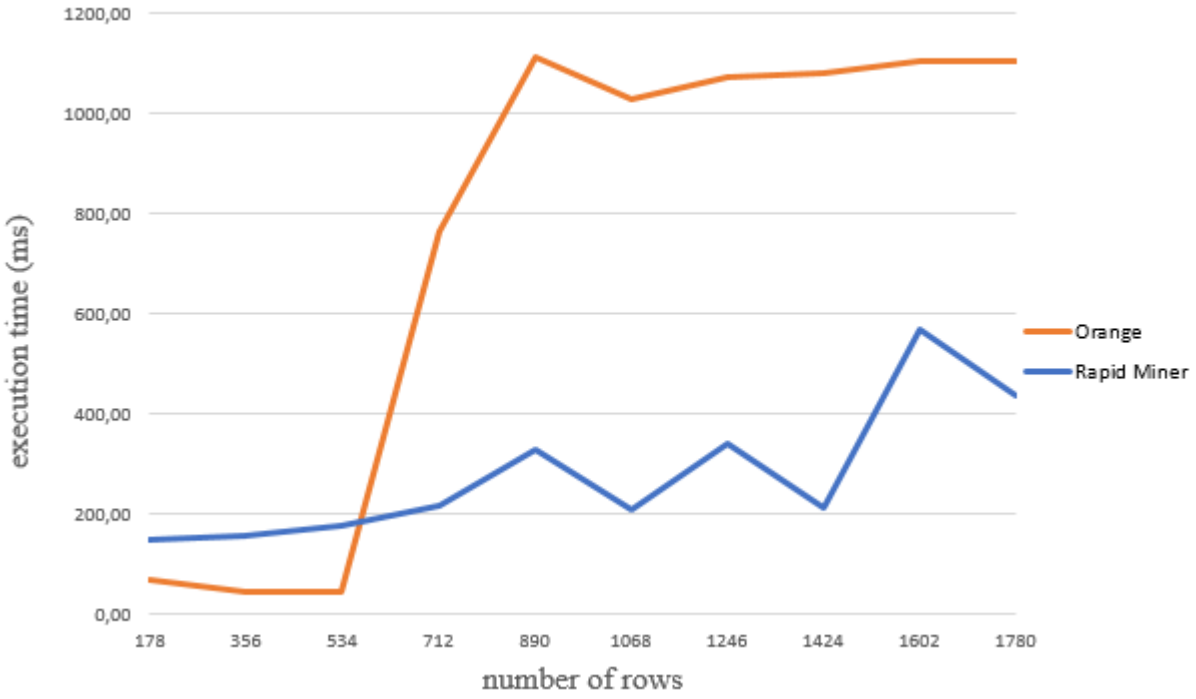


Рисунок 4.21 – Графік часу виконання кластеризації для набору «Wine» на двоядерному процесорі

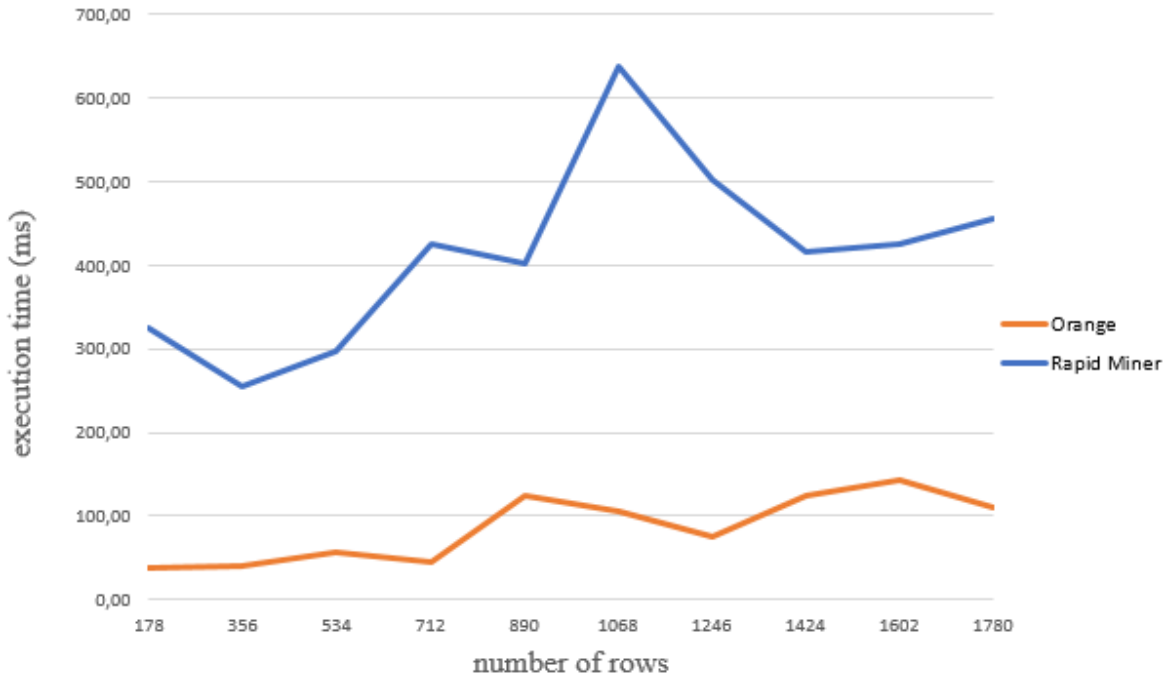


Рисунок 4.22 – Графік часу виконання кластеризації для набору «Wine» » на одноядерному процесорі

Інтерпретація результатів

Показники наведені у таблицях 4.17 і 4.18 свідчать про наступні особливості сервісів:

- показники якості кластеризації отримані на малих і великих даних не відрізняються;
- нормалізація даного набору даних значно погіршує якість отриманих кластерів;
- оцінки якості для сервісів Rapid Miner і Orange суттєво не відрізняються.

Аналізуючи графіки 4.21 і 4.22 варто зазначити наступне:

- швидкість обробки даних у ПЗ Orange є кращою ніж у системи Rapid Miner Studio при використані одноядерного процесору;
- час обробки даних сервісом Orange погіршується на великих даних при використані двоядерного процесору, це може свідчити про деякі труднощі при розпаралелюванні процесів.

4.3.4 Висновки до аналізу функції кластеризації

Проаналізувавши результати отримані під час проведення експериментів були зроблені наступні висновки щодо сервісів Orange і Rapid Miner:

- якість кластеризації погіршується на великих даних;
- при використанні попередньої нормалізації якість кластерів знижується;
- у більшості випадків сервіс Orange здійснює кластеризації швидше за сервіс Rapid Miner. Єдиний випадок, коли часова ефективність сервісу Orange є гіршою – кластеризація набору «Wine» на двоядерному процессорі;
- якість кластерів для сервісу Orange і Rapid Miner приблизно однакова.

Висновки до четвертого розділу

Для дослідження і порівняння сервісів технології Data Mining були проаналізовані результати наступних функцій:

- функція для знаходження викидів LOF;
- функція для знаходження коефіцієнтів лінійної регресії;
- функція для кластеризації об’єктів методом k-середніх.

На підставі проведених експериментів були знайдені наступні особливості сервісів:

- на великих об’ємах даних якість отриманих результатів погіршується;
- сервіс Orange виконує обробку даних набагато швидше за сервіс Rapid Miner;
- при знаходженні викидів сервіс Orange стійкий до нестандартизованих даних;
- для більшості експериментів сервіс Orange надає кращі результати.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Вимоги безпеки при виконанні робіт на робочому місці

У даному розділі розглядається питання безпечної життєдіяльності в середовищі де розроблювалася програма. Аналіз щодо належного стану умов проводиться з огляду до санітарних норм України.

На робочому місці працівника можуть виникнути шкідливі та небезпечні ситуації, наприклад: висока напруга електричної мережі, шкідливі речовини в повітрі, поганий рівень освітлення, високий рівень шуму та інше. Робота з комп'ютером також супроводжується високим рівнем напруженості трудового процесу. Важливо правильно організувати робоче місце, тому що неправильна організація сприяє напрузі верхніх кінцівок, розвитку остеохондрозу, та погіршенню самопочуття робітника в цілому.

Основою безпеки при виконанні робіт на робочому місці є дотримання правил охорони праці, техніки безпеки та протипожежної безпеки при роботі з комп'ютерною технікою. Для забезпечення охорони праці інженера-програміста існують наступні нормативно-правові акти:

- Закон України «Про охорону праці» прийнятий від 14 жовтня 1992 року № 2695-12 [17];
- Типове положення про порядок проведення навчання і перевірки знань з питань охорони, НПАОП 0.00-4.12-05, початок дії від 14.04.2017 [18];
- Державні санітарні норми виробничої загальної та локальної вібрації, ДСН 3.3.6.039-99, початок дії від 01.12.1999 [19];
- Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, ДСанПін 3.3.2.007-98. затверджено постановою головного санітарного лікаря України від 10 грудня 1998 року [20];
- ДСТУ 7299:2013 Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки, затверджено та введено в дію наказом міністерства економічного розвитку і торгівлі України 14.10.2013 № 1231[21];

–ДСТУ ISO 9241-1:2003 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення [22];

–ДСТУ ISO 9241-6:2004 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 6. Вимоги до робочого середовища, введено в дію з 01.01.2006 [23];

–Державні будівельні норми України «Природне і штучне освітлення» ДБН В.2.5-28:2018, затверджені наказом Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України 03.10.2018 № 264, введено в дії з 01.03.2019 [24];

–ДСН 3.3.6.042-99 Державні санітарні норми мікроклімату виробничих приміщень, затверджені Постановою головного санітарного лікаря України № 42 від 1 грудня 1999 року [25];

–Типове положення про службу охорони праці, затверджене наказом Державного комітету України з нагляду за охороною праці від 15.11.2004 № 255 і зареєстроване у Міністерстві юстиції України 01.12.2004 за № 1526/10125 [26];

–НАПБ А.01.001-2014 Правила пожежної безпеки в Україні, затверджений наказом Міністерства внутрішніх справ України від 30.12.2014 № 1417 і зареєстрований у Міністерстві юстиції України 05.03.2015 за № 252/26697 [27] ;

–Наказ про «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою», зареєстрований в Міністерстві юстиції України 7 липня 2014 р.за № 775/25552 [28]

5.2 Шкідливі виробничі фактори на робочому місці

Основним фактором, що впливає на продуктивність праці людей, які працюють за ПЕОМ та відеодисплейними терміналами (далі – ВДТ) є комфортні та безпечні умови праці. Умови праці користувача, що працює з персональним комп’ютерно визначаються:

–особливостями організації робочого місця;

–умовами виробничої середи (освітленням, мікрокліматом, шумом, електромагнітними полями і т.д.);

–характеристиками інформаційної взаємодії людини з ПЕОМ.

Основні шкідливі виробничі фактори на робочому місці для інженера-програміста пов’язані з роботою з персональними електронними обчислювальними машинами. Приведемо перелік найбільш розповсюджених факторів:

–підвищена температура поверхонь ПК;

–нервово-емоційне напруження;

–монотонність трудового процесу;

–напруження зору;

–підвищена контрастність;

–відсутність або брак природнього світла;

–підвищена або низька температура повітря робочої зони;

–підвищений рівень електромагнітних випромінювань.

–підвищений рівень статичної електрики;

–підвищене значення напруги в електричному колі, замикання;

–недостатня штучна освітленість робочої зони;

–підвищена яскравість;

–невідповідність ергономічних показників робочого місця діючим вимогам;

–ризик виникнення пожеж;

Далі у розділі будуть розглянуті рекомендації щодо організації робочого місця таким чином, щоб знизити шкідливий вплив при роботі з ПЕОМ. Також розглянемо відповідність робочого місця, де розроблялась програма, параметрам мікроклімату, освітленості, шуму та вібрації.

5.2.1. Характеристика робочого місця

Робоче місце, де була розроблена програма, має наступні характеристики:

–кількість працюючих: 1 людина;

–довжина приміщення: 6 м;

–ширина приміщення: 4 м;

–висота приміщення: 2.6 м.

За підрахунками отримаємо наступні показники:

–загальна площа дорівнює площі на одне робоче місце: 24м2;

–об’єм приміщення дорівнює об’єму на одне робоче місце: 62.4 м3

Відповідно до вимог робочого приміщення [20] – обсяг приміщення не менше 20 м3 на людину; - площа приміщення не менш 6 м2 на людину. Приміщення відповідає стандартам.

В приміщенні відсутні умови, які можуть створювати підвищену або особливо підвищену небезпеку, тому воно відноситься до класу звичайних приміщень.

5.2.2. Освітлення

Штучне освітлення не має шкідливого впливу на стан робітника за умови, якщо воно правильно спроектоване. У такому випадку воно покращує умови зорової роботи, знижує стомлюваність, сприяє підвищенню продуктивності праці, благотворно впливає на виробниче середовище, надаючи позитивну психологічну дію на працюючого, підвищує безпеку праці і знижує травматизм.

У разі якщо освітлення в робочому місці було спроектовано неправильно, або його недостатньо, людина під час роботи може страждати від напруги зору, ослаблення уваги, настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань, тому такий важливий правильний розрахунок освітленості.

Освітленість в приміщенні обчислювальних центрів нормована ДБН В.2.5-28-2018 [24]. При виконанні роботи використовувалося природне одностороннє бокове й штучне освітлення. Виходячи з того, що по розряду зорової роботи робота за дисплеєм ПЕОМ відноситься до III розряду, при загальному освітленні освітленість робочого місця повинна становити від 200 до 400 лк. Фактична освітленість на робочому місці становить 194,8 лк. Згідно з ДБН В.2.5-28-2018 [24] в приміщенні використовується природне і штучне освітлення. Відповідно до виконаних підрахунків, існуючих джерел світла достатньо для роботи з дисплеєм.

5.2.3 Мікроклімат

Для постійних робочих місць операторів ПК, встановлені оптимальні параметри мікроклімату, при неможливості їх дотримання використовують допустимі параметри. В таблиці 5.1. наведені оптимальні параметри мікроклімату в приміщеннях, де виконуються роботи операторського типу.

Таблиця 5.1. – Параметри мікроклімату для приміщень з ПК

Період року	Параметри мікроклімату	Величина
Холодний	Температура повітря в приміщенні	22 – 24° С
	Відносна вологість	40-60%
	Швидкість руху повітря	До 0.1 м/с
Теплий	Температура повітря в приміщенні	23 - 25° С
	Відносна вологість	40-60%
	Швидкість руху повітря	До 0.1 – 0.2 м/с

Виміряні за допомогою приладів температура та вологість у приміщені відповідають вказаним у таблиці для теплого періоду року. Для того, щоб нормалізувати параметри мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції. Норми подачі свіжого повітря наведені у табл. 5.2.

Таблиця 5.2 – Норми подачі свіжого повітря в приміщення з ПК

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщення, м3 на одну людину в годину
Об'єм до 20м3 на людину	не менше 30
20 - 40 м3 на людину	не менше 20
Більше 40 м3 на людину	може бути використана природна вентиляція

Єдиний ПК розташований у приміщенні є джерелом тепловиділень, крім того для підтримання оптимальних параметрів мікроклімату у приміщенні в холодний період року використовуються нагріті поверхні опалювальної системи.

5.2.4 Шум та вібрація

Шум та вібрація негативно впливають на умови праці надаючи шкідливу дію на організм людини. Люди, що працюють в умовах тривалої шумової дії випробовують дратівливість, головні болі, запаморочення, зниження пам'яті, підвищену стомлюваність, зниження апетиту, біль у вухах і т.д. Також під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється втома у зв'язку з підвищеними енергетичними витратами і нервово-психічним напруженням, погіршується мовна комутація. Все це є причинами зниження працездатності і продуктивності людини.

У приміщенні, де розроблювалась програма, причинною шуму і вібрації являються комп’ютер, вентилятор та кондиціонер. При їхній роботі рівень вібрації не вище 33 дБ, рівень шуму не повинен перевищувати 50 дБА, що є нормою для даного виду діяльності відповідно до ДСН 3.3.6.039-99 [19]. Для зниження рівня шуму стіни і стеля приміщень, де встановлені комп'ютери, можуть бути облицьовані звукопоглинальними матеріалами. Рівень вібрації в приміщеннях обчислювальних центрів може бути понижений шляхом встановлення устаткування на спеціальні віброізолятори.

5.3 Дії працівників в надзвичайних ситуаціях

Існує безліч видів надзвичайних ситуацій з якими можуть стикнутися інженери-програмісти на робочому місці, одна з них – отримання електротравми. Керуючись наказом Міністерства охорони здоров'я України «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою» [28] були розроблені наступні інструкції до виконання у разі нещасного випадку:

- переконатися у відсутності небезпеки. Негайно відключити електроінструмент, переносні споживачі, обладнання та повідомити керівника;
- якщо постраждалий перебуває під дією електричного струму, при можливості припинити його дію: вимкнути джерело струму, відкинути електричний провід за допомогою сухої дерев'яної палиці чи іншого електронепровідного засобу;
- у разі необхідності викликати бригаду екстреної (швидкої) медичної допомоги;
- якщо потерпілий без свідомості, його слід зручно рівно покласти, розстебнути одяг, забезпечити приплив свіжого повітря.
- при поганому диханні слід робити штучне дихання та масаж серця до приїзду швидкої допомоги;
- накласти на місця опіку чисті, стерильні пов'язки;
- у разі погіршенні стану постраждалого до приїзду бригади екстреної медичної допомоги необхідно повторно зателефонувати диспетчеру екстреної медичної допомоги.

Ніколи не слід відмовлятися від надання першої допомоги потерпілому, навіть якщо він здається мертвим через відсутність дихання, серцебиття та інших ознак життя.

Висновки до п’ятого розділу

Для запобігання і усунення можливих шкідливих впливів на робітника був проведений аналіз робочого місця відповідно до державних норм і правил облаштування робочого середовища користувача ПЕОМ. У ході розрахунків було визначено, що освітлення в приміщенні відповідає Державним будівельним нормам України ДБН В.2.5-28:2018 [24], природного освітлення та штучного освітлення достатньо для безпечної та ефективної роботи за комп’ютером. Також було розроблено інструкцію щодо поведінки робітників у разі отримання електротравми.

ЗАГАЛЬНІ ВИСНОВКИ

У дипломній роботі було досліджено особливості функціоналу сервісів Data Mining, зокрема приділялась увага оцінці виконання програмами таких задач як кластеризація, знаходження викидів, побудова моделі лінійної регресії і передбачення.

В ході виконання роботи реалізовано програмне середовище для аналізу і оцінки якості результатів обробки даних, що були отримані від сервісу Orange або Rapid Miner. Також розроблена можливість для порівняння двох сервісів в рамках виконання однієї аналітичної задачі.

У ході розробки проекту використовувалися сучасні технології. Завдяки дотриманню парадигми об’єктно-орієнтованого програмування, система є масштабованою та доступною для вдосконалювати.

Інтерфейс користувача розроблено максимально простим і інтуїтивно зрозумілим, що значною мірою пришвидшує і полегшує процес дослідження сервісів.

Під час аналізу предметної середи було висвітлено основні існуючі проблеми Data Mining та детально розглянуто сервіси, що призначені для автоматизації вирішення цих задач. Виконано огляд існуючого програмного забезпечення.

Проаналізовано метрики для оцінки якості досліджуваних задач, впроваджено автоматичний розрахунок вибраних метрик програмним продуктом «Data Mining Service Analyzer».

Велику увагу було приділено експериментальному дослідженню ефективності сервісів на різних даних і для різних задач. На підставі проведених експериментів було встановлено, що для більшості випадків функції сервісу Orange дають кращі показники якості. Також при тестуванні алгоритмів на різних об’ємах даних, було виявлено наступне:

- якість отриманих результатів незначною мірою падає для великих даних, що свідчить про погану масштабованість програмного забезпечення;

- зменшення кількості операційних ядер при дослідженнях не значною мірою впливає на швидкість оброки даних;

Було виконано математичні розрахунки, що свідчать про економічну доцільність розробки та впровадження нового програмного продукту

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Визначення терміну Data Mining [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://ru.wikipedia.org/wiki/Data_mining (дата звернення: 06.09.2020).

2 Історія виникнення терміну Data Mining [Електронний ресурс]. Стаття з web-сайту History of data mining. – Режим доступу: https://hackerbits.com/data/history-of-data-mining (дата звернення: 06.09.2020).

3 Опис набору даних Іриси Фішера [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://ru.wikipedia.org/wiki/%D0%98%D1%80%D0%B8%D1%81%D1%8B_%D0%A4%D0%B8%D1%88%D0%B5%D1%80%D0%B0 (дата звернення: 06.10.2020).

4 Значення метрики AUC-ROC[Електронний ресурс]. Стаття з web-сайту Understanding AUC - ROC Curve, Sarang Narkhede, Jun 26, 2018. – Режим доступу: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 (дата звернення: 07.10.2020).

5 ROC-крива [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/ROC-%D0%BA%D1%80%D0%B8%D0%B2%D0%B0 (дата звернення: 07.10.2020).

6 Показники для оцінки якості лінійної регресії [Електронний ресурс]. Стаття з web-сайту: Критерії якості лінійної регресії. – Режим доступу: https://studfile.net/preview/5513221/page:6 (дата звернення: 09.10.2020).

7 Коефіцієнт детермінації Mining [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B5%D1%84%D1%96%D1%86%D1%96%D1%94%D0%BD%D1%82_%D0%B4%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D1%96%D0%BD%D0%B0%D1%86%D1%96%D1%97 (дата звернення: 09.10.2020).

8 Середня квадратична помилка [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%94_%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B5_%D0%B2%D1%96%D0%B4%D1%85%D0%B8%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%8C%D0%BE%D0%B3%D0%BE_%D0%B0%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D0%B3%D0%BE (дата звернення: 09.10.2020).

9 Корінь середньої квадратичної помилки [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://en.wikipedia.org/wiki/Root-mean-square_deviation (дата звернення: 09.10.2020).

10 Середня абсолютна похибка [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://en.wikipedia.org/wiki/Mean_absolute_error (дата звернення: 09.10.2020).

11 Індекс Девіса-Болдуіна [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://en.wikipedia.org/wiki/Davies%E2%80%93Bouldin_index (дата звернення: 10.10.2020).

12 Показник Сілуетте [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://en.wikipedia.org/wiki/Silhouette_(clustering) дата звернення: 10.10.2020).

13 Індекс Калінскі-Харабаш [Електронний ресурс]. Стаття з web-сайту: Calinski-Harabasz index. – Режим доступу: https://www.oreilly.com/library/view/machine-learning-

algorithms/9781785889622/8dba1062-2dbe-43ce-a9b0-9ea49203ea9a.xhtml (дата звернення: 10.10.2020).

14 Опуклі кластери [Електронний ресурс]. Стаття з web-сайту: Convex Clustering: Model, Theoretical Guarantee and Efficient Algorithm; Defeng Sun, Kim-Chuan Toh, Yancheng Yuan. – Режим доступу: https://arxiv.org/abs/1810.02677 (дата звернення: 10.10.2020).

15 Принципи SOLID [Електронний ресурс]. Матеріал з Вікіпедії. Доступна енциклопедія. – Режим доступу: https://uk.wikipedia.org/wiki/SOLID_(%D0%BE%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) (дата звернення: 01.11.2020).

16 PEP 8 – стандарт кодування на мові Python. [Електронний ресурс]. Стаття з web-сайту: PEP 8 - руководство по написанию кода на Python. – Режим доступу: https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html (дата звернення: 01.11.2020).

17 Закон України «Про охорону праці» прийнятий від 14 жовтня 1992 року № 2695-12

18 Типове положення про навчання з питань охорони праці, ДНАОП 0.00-4.12-99, зареєстрованого в Міністерстві юстиції України 21 квітня 1999 р.

19 Правила охорони праці під час експлуатації електронно-обчислювальної техніки , ДНАОП 0.00-1.31-99 зареєстрованого в Міністерстві юстиції України від 31 лютого 1999 р.

20 Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, ДСанПін 3.3.2.007-98. затверджено постановою головного санітарного лікаря України від 10 грудня 1998 року

21 ДСТУ 7299:2013 Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки, затверджено та введено в дію наказом міністерства економічного розвитку і торгівлі України 14.10.2013 № 1231

22 ДСТУ ISO 9241-1:2003 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 1. Загальні положення

23 ДСТУ ISO 9241-6:2004 Національний стандарт України. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 6. Вимоги до робочого середовища, введено в дію з 01.01.2006

24 Державні будівельні норми України «Природне і штучне освітлення» ДБН В.2.5-28:2018, затверджені наказом Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України 03.10.2018 № 264, введено в дії з 01.03.2019

25 Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99, затверджені Постановою головного санітарного лікаря України № 42 від 1 грудня 1999 року

26 Типове положення про службу охорони праці, затверджене наказом Державного комітету України з нагляду за охороною праці від 15.11.2004 № 255 і зареєстроване у Міністерстві юстиції України 01.12.2004 за № 1526/10125

27 Правила пожежної безпеки в Україні, затверджені наказом Міністерства внутрішніх справ України від 30.12.2014 № 1417 і зареєстровані у Міністерстві юстиції України 05.03.2015 за № 252/26697

28 Наказ про «Порядок надання домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою» зареєстрований в Міністерстві юстиції України 7 липня 2014 р.за № 775/25552.

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Дніпровського
національного університету
залізничного транспорту імені
академіка В. Лазаряна
_____ Боднар Б. Є.

ІНСТРУМЕНТ ДЛЯ ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ СЕРВІСІВ DATA MINING

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

01116130.01182-01-ЛЗ

Завідувач кафедри КІТ
_____ В. І. Шинкаренко

Керівник розробки
_____ О. П. Іванов

Виконавець
_____ Д. В. Васильєва

Нормоконтролер
_____ Куроп'ятник О.С.

2020

ЗАТВЕРДЖЕНО

01116130.01182-01-ЛЗ

ІНСТРУМЕНТ ДЛЯ ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ СЕРВІСІВ DATA MINING

Технічне завдання

01116130.01182-01

Листів 23

2020

АНОТАЦІЯ

Документ 01116130.01182-01 «Інструмент для дослідження та порівняння сервісів Data Mining. Технічне завдання» входить до складу програмної документації до програми, яка реалізує програмне забезпечення «Data Mining Service Analyzer».

У даному документі представлене призначення та область застосування програмного продукту, основні вимоги, стадії та строки виконання проекту, технічні та техніко-економічні показники, що пред’являються до програмного продукту.

ЗМІСТ

ВСТУП

Стрімкий розвиток інформаційних технологій спричинює появу великих об’ємів даних, що потребують ефективної обробки. На ряду з цією проблемою постає задача здобуття нової, раніше невідомої інформації з вже існуючих даних, цей процес називається data mining, або глибинний аналіз даних. З підвищенням актуальності здобуття нових даних, з’являється все більше і більше програмних засобів, що дозволяють автоматизувати опрацювання даних. Не дивлячись на те, що сервіси Data Mining використовують одні й ті самі алгоритми, отримані результати можуть кардинально відрізнятися, що можна пояснити особливостями реалізації ПО. Для того, щоб проаналізувати різницю в поведінці сервісів і оцінити якість їх роботи, експерт повинен витратити велику кількість часу, адже на даний момент для таких цілей не існує спеціалізованого ПО.

Програмний продукт «Інструмент для дослідження та порівняння сервісів Data Mining» призначено для того щоб експериментальним шляхом виявляти ефективність сервісів для виконання тієї чи іншої задачі.

Ключові слова: Data Mining, глибинний аналіз, показники якості, кластеризація, викиди, лінійна регресія, передбачення.

Причинами виникнення є відсутність подібних аналогів.

Область застосування: глибинний аналіз даних.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є навчальний план для спеціальності 121 «Інженерія програмного забезпечення», затверджений ректором Дніпровського національного університету залізничного транспорту імені академіка В. Лазаряна проф. Пшінька О. М. від 20.06.2019 р.

Тема роботи: дослідження сервісів технології Data Mining, керівник розробки доц. Іванов О.П.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення

Функціональним призначенням розробки є:

– автоматизація процесу аналізу якості результатів функцій кластеризації, знаходження викидів, прогнозування та пошук рівняння лінійної регресії. Вище зазначені алгоритми реалізовані сервісами, що досліджуються;
– опрацювання вказаного набору даних вибраним сервісом і алгоритмом;
– відображення отриманих від сервісів результатів;
– візуалізація кластерів;
– розрахунок і відображення графіку залежності часу виконання функції від об’ємів даних.

2.2 Експлуатаційне призначення

Експлуатаційне призначення полягає у можливості пришвидшити та автоматизувати процес аналізу сервісів, тим самим оптимізувати роботу аналітиків, або експертів, що займаються вивченням даних.

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Розроблювана програма розрахована на використання на ЕОМ та спроектована з урахуванням можливості подальшого розвитку методом додавання нових сервісів і додаткових функцій.

Вимоги до функціональних характеристик програмного забезпечення:

- можливість запускати один і більше сервісів;
- можливість генерувати графік залежності часу виконання функції від об’ємів даних;
- можливість обробляти дані з розширенням csv.

Часові характеристики: відповідь програми через 0.01-0.03 сек. – при натисненні кнопок та пунктів меню.

- Вхідні дані наступні:
- шлях до файлу з даними в форматі csv;
- функція для тестування;
- опція використання нормалізації;
- опція видалення викидів;
- вибір сервісу для запуску;
- критерії для оцінки сервісів;
- критерії для порівняння сервісів;
- кількість експериментів для побудови графіка залежності часу виконання функції від об’ємів даних;
- кількість кластерів;
- ім’я колонки для розрахування лінійної регресії;
- шлях до даних з викидами;
- шлях до даних для прогнозу.

Вихідні дані наступні:

- результати обробки даних для вибраних сервісів;
- показники якості результатів, якщо відповідна опція була обрана;

– графік залежності часу виконання функції від об’ємів даних, якщо відповідна опція була обрана;
– графік розподілення кластерів, якщо була обрана функція кластеризації;
– повідомлення про помилки програми — текст англійською мовою;

3.2 Вимоги до надійності

Вимоги до надійності програмної системи не враховують ситуації, при яких у збої винна версія операційної системи, яка установлена на пристрої.

Основні вимоги надійності:

– програма має здійснювати контроль за введеними даними;
– наявність резервної копії програми на електронному носії;
– збой устаткування не впливає на подальшу роботу програми;
– захист від копіювання та несанкціонованого доступу реалізується за рахунок стандартних засобів операційної системи.

3.3 Умови експлуатації

Даний програмний продукт може використовуватись в умовах, які відповідають вимогам документу [1].

Для нормального функціонування програмного продукту необхідно виконання наступних вимог:

– програма повинна експлуатуватись у приміщенні, призначеному для роботи з ЕОМ, з відповідними кліматичними умовами: температура 21° – 25°С та вологість 40 - 60%
– ЕОМ повинні відповідати вимогам чинних в Україні стандартів, нормативних актів з охорони праці [2];
– стан технічних засобів повинен задовольняти відповідним нормам та вимогам;

Для забезпечення надійного функціонування програмного продукту необхідно дотримуватися таких умов:

– програма призначена для роботи на ЕВМ з операційною системою сімейства Ubuntu 16.04 і вище;

– для роботи з програмою достатньо базових навичок роботи з ПК та системами сімейства Linux.

3.4 Вимоги до складу і параметрів технічних засобів

Пристрій, що має щонайменше 20 Gb вільного місця на вбудованому носії інформації та 4096 Мб оперативної пам’яті для коректної роботи. Для зручної роботи з програмою необхідно мати монітор з роздільною здатністю 800*600 і вище, маніпулятор «миша» та клавіатуру. Для встановлення ПЗ необхідна або наявність CD/DVD приводу чи USB роз’єму, або підключення до мережі Інтернет.

3.5 Вимоги до інформаційної та програмної сумісності

Вимоги до інформаційної та програмної сумісності: ОС Ubuntu 16.04 та вище.

3.6 Вимоги до маркування і упаковки

Програмний продукт може маркуватися як зображено на рис. 3.1:

сервіс “Data Mining Service Analyzer”, 1.0
Васильєва Дар’я Віталіївна
кафедра КІТ, ДНУЗТ 2020

Рисунок 3.1 – Маркувальний штамп

3.7 Вимоги до транспортування і зберігання

Транспортувати програмний продукт можна наступними способами:

– через всесвітню систему взаємосполучених комп'ютерних мереж, що базуються на комплекті Інтернет-протоколів – Інтернет;
– за допомогою портативних носіїв інформації – CD/DVD-дисків або USB-накопичувачів.

Термін збереження обумовлений збереженням інформації на носії.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації має входити технічне завдання та робочий проект.

До складу робочого проекту мають входити:

- специфікація;
- текст програми;
- опис програми;
- опис застосування;
- керівництво користувача. Керівництво користування програмою.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів [3].

5 ВИЗНАЧЕННЯ ВИТРАТ НА ПРОЕКТУВАННЯ ПРОГРАМИ

Основна мета розробки техніко-економічного обґрунтування (ТЕО) – дати фінансову оцінку передбачуваних витрат та одержуваного корисного результату, а також оцінити прибутковість проекту і, в кінцевому підсумку, економічну доцільність його розробки та впровадження.

Початковим етапом розрахунку величини трудових витрат розробників є оцінка розміру програмного забезпечення. Основні відмінності методик, що застосовуються в оцінці трудовитрат, полягають у використовуваному типі критерію оцінки якості [4].

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях.

$$E = a \cdot S^b \cdot EAF, \qquad (5.1)$$

де E – витрати праці на проект (в людино-місяцях);

Sb – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05.

Розмір програмного коду було підраховано за допомогою інструменту командного рядка для сканування папок для вихідних файлів коду і підрахунку кількості рядків вихідних кодів в ньому – Pygount (рисунок 5.1)

Language	Files	%	Code	%	Comment	%
Python	9	90.00	774	100.00	50	100.00
__generated__	1	10.00	0	0.00	0	0.00
Sum total	10		774		50	

Рисунок 5.1 – Підрахунок розміру програмного коду

Отже розмір програмного коду становить 774 рядки:

$$E = 2{,}4 \cdot 774^{1{,}05} \cdot 1 = 1{,}84$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 1,84 людино-місяці.

Нижче наведені розрахунки вартості розробки «Автоматизована система оцінки схожості програм».

Основними статтями витрат прийняті:

- основна заробітна плата;
- відрахування на соціальні потреби;
- накладні витрати;
- витрати на персональний комп’ютер і ліцензійні базові програмні засоби.

Основна заробітна плата (ОЗП) оцінює працю інженера-програміста зі створення програмного продукту і визначається виходячи з кількості розробників, часу виконання розробки (годин), а також заробітної плати в розрахунку на одну годину. Розрахунок заробітної платні проводиться по формі табл. 5.1 [5].

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати грн
			чол	місяців	
1	інженер-програміст	13400	1	1,84	24656

Описаний в проекті програмний продукт був розроблений одним програмістом в період з 13.01.20 до 5.03.20, що складає 39 днів або приблизно 8 робочих тижнів. Витрати робочого часу прийняті за 40 годин у тиждень. Погодинна ставка кваліфікованого інженера–програміста складає 79,76 грн/год. Таким чином, витрачено робочого часу:

$$t_{\text{розробки}} = N_{\text{чол}} \times N_{\text{тиж}} \times N_{\text{год}}, \quad (5.2)$$

де $N_{\text{чол}}$ – кількість виконавців, чол;

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, год;

$$t_{\text{розробки}} = 1 \cdot 8 \cdot 40 = 320 \text{ чол/год.}$$

ОЗП визначається за формулою:

$$\text{ОЗП} = t_{\text{розробки}} \cdot N \cdot K_{KB}, \quad (5.3)$$

де $t_{\text{розробки}}$ – витрати праці у чол/год;

N – погодинна ставка;

K_{KB} – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. Коефіцієнт кваліфікації розробника (k) - ступінь підготовленості виконавця до дорученої йому роботі (він визначається залежність від стажу праці та становить:

для працюючих до 2 років - 0,75;

від 2 до 3 років - 1,0;

від 3 до 5 років - 1,1-1,2;

від 5 до 7 років - 1,3-1,4;

понад 7 років - 1,5-1,6.

В даному випадку K_{KB} приймається 0,75.

ОЗП складає:

$$\text{ОЗП} = 320 \cdot 79{,}76 \cdot 0{,}75 = 19142 \text{ грн.}$$

Відрахування на соціальні потреби (ЄСВ) [6] є 22% та встановлюються у відсотках від суми заробітної плати:

$$C_{\text{соц}} = \frac{\text{ОЗП} \times 22\%}{100\%} \quad (5.4)$$

$$C_{\text{соц}} = \frac{19142 \times 22\%}{100\%} = 4211 \text{ грн}$$

Отримані результати за (5.3) та (5.4) підсумовуються. Вони складають 23353 грн. та визначають основні прямі витрати.

Накладні витрати враховують загальногосподарчі витрати по забезпеченню проведення роботи: витрати на опалення, електроенергію, амортизація будівель,

зарплату адміністративного персоналу та інше. Вони визначаються в процентах (30 – 40%) від суми прямих витрат:

$$С_{накл} = \frac{(ОЗП + С_{СОЦ}) \times 35\%}{100\ \%}; \quad (5.5)$$

$$С_{накл} = \frac{(19142 + 4211) \times 35\%}{100\ \%} = 8173 \text{ грн}$$

На протязі усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов’язані з її експлуатацією.

Експлуатаційні витрати на персональний комп’ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп’ютеру. В експлуатаційні витрати входять:

- вартість витратних матеріалів;
- витрати на ремонт;
- заробітна плата ремонтника;
- оренда приміщення;
- додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;
- амортизаційні витрати на персональний комп'ютер і програмне забезпечення;
- витрати на електроенергію ($С_{ел}$),

які визначаються за формулою:

$$С_{ел} = P \times B \times T_{розр}, \quad (5.6)$$

де Р – потужність комп’ютера та допоміжних споживачів електричної енергії, приймається 0,45 кВт/год [7];

B – вартість 1 кВт/годин для непобутових споживачів, складає 0,240 грн [8];

$T_{розр}$ – час роботи з ЕВМ, приймається рівним робочому часу.

Витрати на електроенергію визначаються так:

$$C_{\text{ел}} = 0{,}45 \cdot 2{,}73 \cdot 320 = 393 \text{ грн.}$$

Витрати на витратні матеріали ($\text{С}_{\text{вм}}$) протягом всього терміну експлуатації приблизно 10% від вартості комп'ютеру. Вартість робочої станції приймається 18 000 грн. [9] (ноутбук Asus, ОЗУ 8 Гб, 4 ядра, 256 ГБ пам'яті), термін експлуатації – 5 років [10]. Отже, можна визначити ці витрати за період створення програмного засобу:

$$\text{С}_{\text{вм}} = \text{В}_{\text{ком}} \times \frac{N_{\text{Д}}}{N_{\text{експ}} \times 365} \times \frac{10\ \%}{100\ \%}, \quad (5.7)$$

де $B_{КОМ}$ – вартість персонального комп'ютеру;

$N_{\text{Д}}$ – кількість днів розробки програмного продукту;

$N_{\text{експ}}$ – термін експлуатації персонального комп'ютеру.

Витрати на витратні матеріали визначаються так:

$$C_{\text{вм}} = 18000 \cdot \frac{39}{5 \cdot 365} \cdot \frac{10}{100} = 38 \text{ грн}$$

Заробітна плата ремонтника ($C_{\text{рем}}$) визначена наступним чином: на ремонт 50 комп'ютерів потрібен один інженер-системотехнік. Його середньомісячна заробітна плата приймається 8000 [11] грн. Тоді в перерахунку на один комп'ютер його заробітна плата за період розробки програмного продукту складає:

$$C_{\text{рем}} = \frac{C_{\text{рем}}}{N_{КОМ}} \times T_{\text{міс}}, \quad (5.8)$$

де $C_{\text{рем}}^{\prime}$ – середньомісячна заробітна плата;

$N_{КОМ}$ – кількість комп'ютерів на одного ремонтника.

Тмес – час розробки програмного продукта, міс.

Заробітна плата ремонтника ($C_{\text{рем}}$) буде складати:

$$C_{\text{рем}} = \frac{8000}{50} \cdot 1{,}84 = 294 \text{ грн.}$$

За статистикою витрати на комплектуючи вироби ($С_{КОМ}$) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$C_{КОМ} = C_{ВМ} = 38\text{грн.} \quad (5.9)$$

Амортизаційні відрахування на персональний комп'ютер (АПК) визначені з положення, що амортизаційний період в даний час дорівнює терміну морального старіння обчислювальної техніки і складає 3 роки [6]. Отже, за 3 роки амортизаційні відрахування на персональний комп'ютер дорівнюють вартості комп'ютера. За період проектування амортизаційні відрахування складуть:

$$\text{АКП} = \text{В}_{КОМ} \times \frac{N_{\text{Д}}}{N_{\text{експ}} \times 365}, \quad (5.10)$$

$$\text{АКП} = 18000 \times \frac{1{,}84}{3 \times 12} = 920{,}00$$

Амортизаційні відрахування на програмне забезпечення (АПЗ) залежать від його циклу заміни. Якщо прийняти термін морального старіння для Linux (Ubuntu 16.04) 5 років та PyCharm за 1 рік то амортизаційні відрахування на програмне забезпечення дорівнюють його вартості.

Для функціонування персонального комп'ютера використовувалася операційна система Linux (Ubuntu 16.04), для написання програмного забезпечення - програмне середовище PyCharm.

$$\text{АКП}_{Linux} = 0 \times \frac{1{,}84}{5 \times 12} = 0$$

$$\text{АКП}_{PyCharm} = 5611 \times \frac{1{,}84}{12} = 860{,}35$$

Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.2. Додаткові витрати ($С_{\text{дод}}$): прибирання приміщень, охорона, комунальні послуги важко оцінити точно і прийняти рівними 50% заробітної плати інженера-програміст, тобто 6700 гривень на місяць. Оренду приміщень для однієї людини приймемо рівною 1 050 гривень на місяць (загальна вартість оренди приміщення на 10 інженер-програмістів 10 500 грн. на місяць [11]). Тобто за весь період розробки –

6 300 грн. Сумарні експлуатаційні витрати на один персональний комп'ютер складають:

$$C_{\text{експ}} = C_{\text{ел}} + C_{\text{ВМ}} + C_{\text{рем}} + \text{АПК} + \text{АПО} + C_{\text{ор}} + C_{\text{дод}}; \quad (5.11)$$

$$C_{\text{експ}} = 393 + 38 + 331 + 920{,}00 + 860{,}00 + 5000{,}00 + 6700{,}00 = 14288 \text{ грн}$$

Результати розрахунків зведено у табл. 5.3.

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
Ubuntu 16.05	0	https://ubuntu.com	0
PyCharm	5611	https://www.jetbrains.com/ru/pycharm/buy	860
Всього:	5611	-	860

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	393
Вартість витратних матеріалів	38
Витрати на ремонт	331
Амортизація персонального комп'ютера	920
Амортизація програмного забезпечення	860
Оренда приміщення	5000
Додаткові витрати	6700
Всього	14288

Таким чином, витрати на створення програмного продукту складають:

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}; \quad (5.12)$$

$$C_{\text{розробки}} = 19142 + 4211 + 8173 + 14108 = 46635 \text{ грн.}$$

Розрахунок витрат зведено у табл. 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	19142
Відрахування на соціальні потреби	4211
Накладні витрати	8173
Експлуатаційні витрати	14288
Всього	46993

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для оцінки схожості програм. За результатами розрахунків, приблизна вартість розробки складає 46993 грн.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки приводяться в табл. 6.1.

Таблиця 6.1 – Стадії та етапи розробки

№	Стадії розробки	Етап розробки	Термін
1	Технічне завдання	Постановка задачі	01.10.19 – 30.12.19
		Розробка структур вхідних і вихідних даних	14.01.20 – 30.01.20
		Розробка вимог до програми	03.02.20 – 28.02.20
		Затвердження технічного завдання	02.03.20 - 31.03.20
2	Робочий проект	Розробка і програмування логіки програми	01.04.20 – 30.06.20
		Розробка і програмування користувацького інтерфейсу	01.07.20 –7.08.20
		Відлагодження програми	10.08.20 – 31.08.20
		Розробка програмної документації	01.09.20 – 30.10.20
3	Впровадження	Підготовка і передача програми	01.11.20 – 12.12.20

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙОМУ

Контроль здійснюється за допомогою виконання набору тестів з метою знаходження помилок в програмному продукті та його специфікації. Контроль виконання роботи забезпечується головним керівником розробки.

Прийом програмного продукту здійснюється уповноваженою комісією.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. ДСанПІН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»
2. Закон Міністерства охорони здоров'я України від 09.10.2000 № 247 (у редакції наказу МОЗ від 14.03.2006 № 120) «Про затвердження Тимчасового порядку проведення державної санітарно-гігієнічної експертизи»
3. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. – 38 с.

4 Стаття з Web-сайту Методики оценки трудозатрат по разработке программного обеспечения информационных систем [Електронний ресурс]. http://repository.enu.kz/bitstream/handle/data/12881/METODIKA-TRUDOZATRAT.pdf (дата звернення: 10.10.2020).

5 Огляд статистики зарплатні професії "Інженер-програміст в Україні" [Електронний ресурс]. – Режим доступу: https://ua.trud.com/ua/salary/2/67643.html (дата звернення: 10.10.2020).

6 Сторінка Міністерства Фінансів «Єдиний соціальний внесок в Україні з 2011 по 2020 рр.» [Електронний ресурс]. – Режим доступу: https://index.minfin.com.ua/ua/labour/social/ (дата звернення: 10.10.2020).

7 Стаття з Web-сайту Середня потужність комп'ютера [Електронний ресурс]. – Режим доступу: https://beasthackerz.ru/uk/windows-7/srednyaya-moshchnost-kompyutera-vt-skolko-elektroenergii-potreblyaet.html (дата звернення: 10.10.2020).

8 Стаття з Web-сайту Національна комісія, що здійснює Державне регулювання у сферах енергетики та комунальних послуг [Електронний ресурс]. – Режим доступу: www.nerc.gov.ua/?id=53029 (дата звернення: 10.10.2020).

9 Сторінка інтернет-магазину Comfi з посиланням на обраний ПК [Електронний ресурс]. – Режим доступу: https://comfy.ua/ua/noutbuk-igrovoj-asus-m570dd-dm101-black-harakteristiki.html (дата звернення: 10.10.2020).

10 Витяг з Податкового кодексу України, стаття 145 [Електронний ресурс]. – Режим доступу: https://i.factor.ua/ukr/law-24/section-121/article-698/2014-07-31/ (дата звернення: 10.10.2020).

11 Системний адміністратор, сервісний інженер. Пошук робочих місць. Режим доступу: https://www.work.ua/ru/resumes/425938/ (дата звернення: 10.10.2020).

12 Об'ява «Аренда офис». Режим доступу: https://www.olx.ua/obyavlenie/arenda-ofis-70m2-sklad-salon-dnepr-tsentr-ya-samarskogo-barrikadna-sdayu-IDJadRe.html. (дата звернення: 10.10.2020).

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Дніпровського
національного університету
залізничного транспорту імені
академіка В. Лазаряна
_____ Боднар Б. Є.

ІНСТРУМЕНТ ДЛЯ ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ СЕРВІСІВ DATA MINING

Робочий проект

ЛИСТ ЗАТВЕРДЖЕННЯ

01116130.01182-01-ЛЗ

Завідувач кафедри КІТ
_____ В. І. Шинкаренко

Керівник розробки
_____ О. П. Іванов

Виконавець
_____ Д. В. Васильєва

Нормоконтролер
_____ О. С. Куроп'ятник

2020

ЗАТВЕРДЖЕНО
01116130.01182-01-ЛЗ

ІНСТРУМЕНТ ДЛЯ ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ СЕРВІСІВ DATA MINING

Специфікація

01116130.01182-01

Листів 204

2020

Позначення	Найменування	Примітка
	Документація	
01116130.01182-01-ЛЗ	Лист затвердження	
01116130.01182-01 12 01-ЛЗ	Лист затвердження	
01116130.01182-01 12 01	Текст програми	
01116130.01182 -01 13 01-ЛЗ	Лист затвердження	
01116130.01182 -01 13 01	Опис програми	
01116130.01182 -01 31 01-ЛЗ	Лист затвердження	
01116130.01182 -01 31 01	Опис застосування	
01116130.01182 -01 І3 01-ЛЗ	Лист затвердження	
01116130.01182 -01 І3 01	Керівництво користувача	

ЗАТВЕРДЖЕНО
01116130.01182-01 13 01-ЛЗ

ІНСТРУМЕНТ ДЛЯ ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ СЕРВІСІВ DATA MINING

Опис програми
01116130.01182-01 13 01
Листів 24

2020

АНОТАЦІЯ

Документ 01116130.01182-01 13 01 «Інструмент для дослідження та порівняння сервісів Data Mining “Data Mining Service Analyzer.” Опис програми» входить до складу програмної документації до програми, яка реалізує програмне забезпечення «Data Mining Service Analyzer» з дотриманням основних принципів ООП.

В документі міститься опис програми та її функціональних можливостей. Програма реалізована на мові python у програмному середовищі PyCharm 2019.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Програмний продукт має назву «Data Mining Service Analyzer» він реалізує функціонал необхідний для дослідження і порівняння сервісів технології Data Mining.

До програмних засобів, що потрібні для функціонування даного продукту, слід віднести операційну систему сімейства GNU/Linux з графічним інтерфейсом.

Програма реалізована на мові python у програмному середовищі PyCharm 2019.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Функціональним призначенням розробки є:

- автоматизація процесу аналізу якості результатів функцій кластеризації, знаходження викидів, прогнозування та пошук рівняння лінійної регресії. Вище зазначені алгоритми реалізовані сервісами, що досліджуються;
- опрацювання вказаного набору даних вибраним сервісом і алгоритмом;
- відображення отриманих від сервісів результатів;
- візуалізація кластерів;
- розрахунок і відображення графіку залежності часу виконання функції від об’ємів даних.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1 Алгоритм програми

Алгоритми організації роботи програми з користувачем є очевидним і зрозумілим. Загальний алгоритм програми наступний: отримати налаштування експерименту від користувача, обробити дані та вивести результат. Розглянемо детальний покроковий алгоритм програми:

- До початку обробки даних необхідно отримати від користувача наступні дані:
  - розташування файлу з даними для обробки: якщо було вказано невірний шлях вивести повідомлення про помилку, інакше відобразити дані в правій верхній таблиці;
  - функцію, що буде оброблювати дані, якщо була обрана функція «outliers», розблокувати для вибору пошук метрики AUC-ROC;
  - режим увімкнення або вимкнення попередньої нормалізації даних;
  - режим увімкнення або вимкнення попереднього видалення викидів в даних;
  - вибір сервісів, що будуть досліджуватися (Rapid Miner, Orange або обидва);
  - вибір критеріїв для порівняння сервісів;
  - вибір критеріїв для оцінки сервісів.
- Після того, як користувач натиснув кнопку «Process» здійснюється перевірка введених даних, в залежності від того, які параметри були налаштовані система може запросити у користувача додаткову інформацію, а саме:
  - шлях до даних для генерації прогнозів, якщо була обрана функція «prediction»;

- ім’я цільової колонки відносно якої буде робитися прогноз, або розраховуватися рівняння лінійної регресії, якщо була обрана функція «prediction» або «linear regression»;
- кількість кластерів, якщо була обрана функція «k-means clustering»;
- кількість експериментів якщо була обрана опція «execution time depense from amount of data»;
- шлях до файлу з промаркованими викидами, якщо була обрана опція «AUC-ROC metric».

– Коли всі необхідні дані були зібрані, програма починає опрацювання інформації, алгоритм дій наступний:
  - отримати результати для сервісу Orange з відповідними метриками, якщо цей сервіс був обраний;
  - отримати результати для сервісу Rapid Miner з відповідними метриками, якщо цей сервіс був обраний;
  - виконати порівняння сервісів, якщо ця опція була вказана.
– Якщо обробка даних пройшла успішно, програма розблоковує робочий екран і видає надпис «Finished», у тому разі якщо при обробці даних виникли помилки, програма виведе відповідне повідомлення.

3.2 Використані методи

При написанні коду було дотримано методики об’єктно-орієнтованого програмування в мові Python [1]. Були застосовані такі принципи об’єктно-орієнтованого програмування як поліморфізм, наслідування, інкапсуляція [3]. При розробці програми жодних математичних формул застосовано не було.

3.3 Структура програми з описом функцій складових частин і зв'язки

Структуру програми можна представити за допомогою діаграми класів. Діаграма класів складається з множини елементів, які в сукупності відображають декларативні знання про предметну галузь. Ці знання інтерпретуються в базових поняттях мови UML, таких як класи, інтерфейси та відношення між ними. На діаграмі

класів, класи відображаються у вигляді прямокутників, у верхній частині яких відображається назва в середині атрибути, а в нижній – методи цих класів. На рис. 3.1 представлена схема взаємодії основних модулів програми.

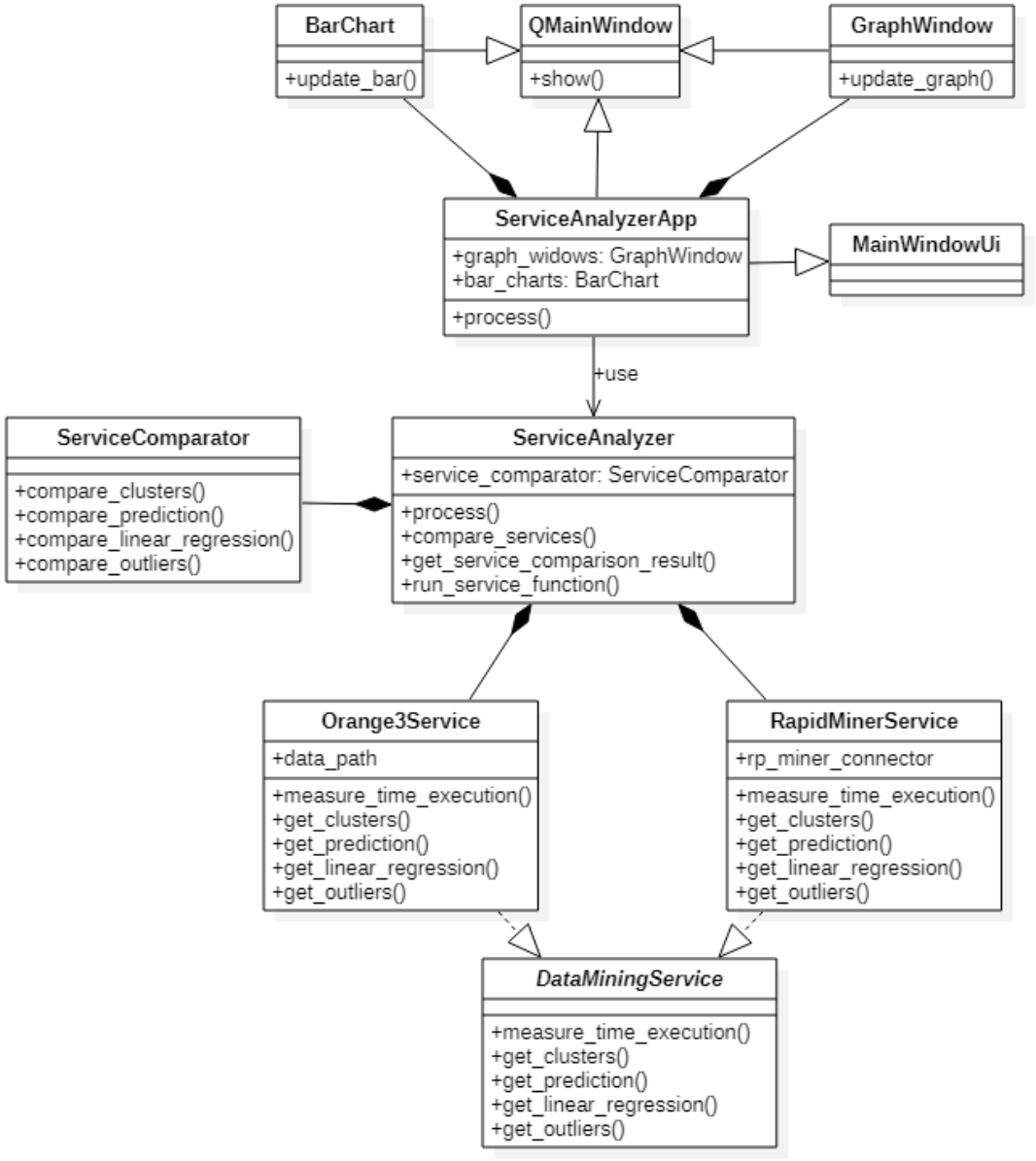


Рисунок 3.1 – Схема взаємодії модулів програми

QMainWindow – клас з бібліотеки PyQT, реалізує стандарте графічне вікно. Всі класи, які відповідають за створення робочого вікна повинні наслідуватися від даного компоненту.

Клас Bar Chart відповідає за побудову та відображення графіків отриманих кластерів. Є нащадком класу QMainWindow, оскільки представляє об’єкт робочого

вікна. Клас має зв’язок композиції з класом ServiceAnalyzerApp, адже цей клас відповідає за його створення і ініціалізацію.

Клас Graph Window, схожий за характеристиками до класу Bar Chart, відповідає за побудову та відображення графіків, на яких відтворюється залежність часу виконання функції від об’ємів даних Є нащадком класу QMainWindow. Клас створюється і ініціалізується в модулі ServiceAnalyzerApp, тому має з ним зв’язок композиції.

ServiceAnalyzerApp – компонент, що відповідає за запуск програми та її функціонування. Є нащадком класу QMainWindow оскільки реалізує головне вікно програми. Також має зв’язок наслідування з класом MainWindow_Ui, для того, щоб використовувати попередньо розроблені віджети для взаємодії з користувачем. Клас ServiceAnalyzerApp отримує дані від користувача, що необхідні для подальшої обробки даних і передає їх до екземпляру класу ServiceAnalyzer. Після того, як ServiceAnalyzer обробив дані, клас ServiceAnalyzerApp відображає результати на робочому вікні.

Модуль ServiceAnalyzer є відповідальним за обробку даних, отриманих від користувача. В залежності від того, які налаштування були вибрані, компонент створює класи Orange3Service, RapidMinerService і викликає у них відповідні функції. Результати отримані від сервісів, передаються на обробку до класу ServiceComparator, де вони порівнюються. Готові дані повертаються до компоненту ServiceAnalyzerApp, де користувач може їх самостійно проаналізувати. Оскільки клас створює допоміжні компоненти Orange3Service, RapidMinerService, ServiceComparator то має з ним зв’язок композиції.

Інтерфейс DataMiningService визначає методи, які повинні бути реалізовані для класів що надають доступ до функцій сервісів Data Mining.

Модуль Orange3Service – відповідає за надання доступу до функцій сервісу Orange3. Реалізує інтерфейс DataMiningService.

Модуль RapidMinerService – відповідає за надання доступу до функцій сервісу RapidMiner. Реалізує інтерфейс DataMiningService.

Модуль ServiceComparator – реалізує порівняння результатів сервісів. Підраховує оцінки якості для отриманих результатів.

3.4 Зв'язки програми з іншими програмами

Розроблюваний програмний продукт має залежність від програми RapidMiner Studio версії 9.8 і не може працювати автономно. Якщо програма RapidMiner Studio не була встановлена, то програмне забезпечення «Data Mining Service Analyzer» не підтримує наступні функції:

- порівняння сервісів;
- запуск будь-якої з задач Data Mining сервісом Rapid Miner;
- побудова графіків для роботи сервісу.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для експлуатації програмного продукту необхідний будь-який пристрій з параметрами, які задовольняють нормальному функціонуванню операційної системи Ubuntu 16.04 або вище та має графічний інтерфейс.

Мінімальна конфігурація пристрою, що забезпечить нормальну роботу програмного продукту:

- щонайменше 20 Gb вільного місця на вбудованому носії інформації;
- щонайменше 4096 Мб оперативної пам’яті;
- монітор з роздільною здатністю 800*600 і вище;
- центральний процесор з тактовою частотою 1 ГГц та більше;
- маніпулятор «миша»;
- стандартна клавіатура.

5 ВИКЛИК ТА ЗАВАНТАЖЕННЯ

Для запуску програмного продукту «Data Mining Service Analyzer» необхідно виконати запуск файлу ServiceAnalyzer з терміналу Linux. Після запуску програми користувачеві представляється головне вікно програми (рис. 5.1).

Об’єм програми у неархівованому вигляді складає 123 Мбайт. Комплекс функціонує у середовищі Ubuntu 16.04 та вище.

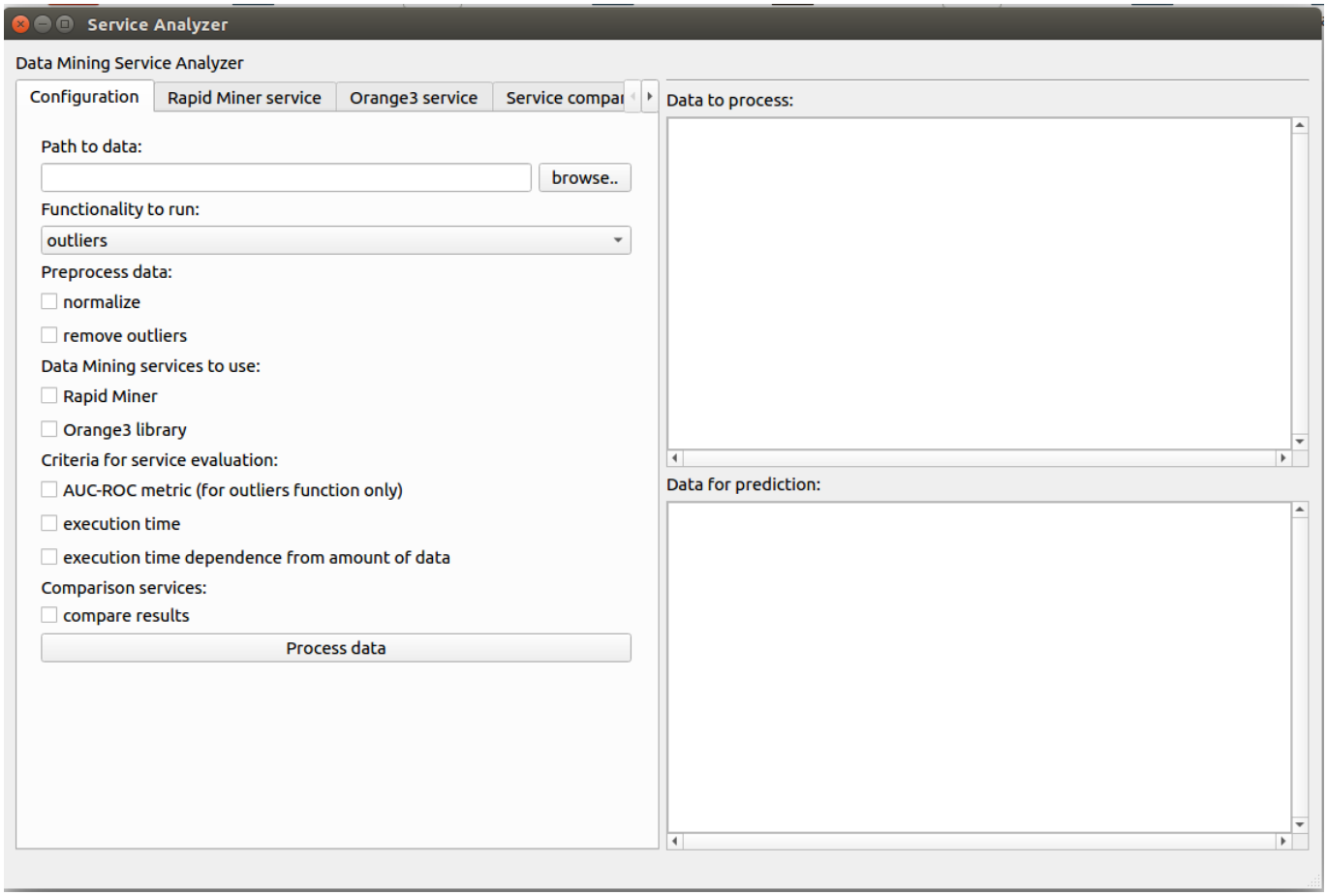


Рисунок 5.1 – Головне вікно програми

6 ВХІДНІ ДАНІ

Вхідними даними програми є:

- шлях до файлу з даними в форматі csv;
- функція для тестування;
- опція використання нормалізації;
- опція видалення викидів;
- вибір сервісу для запуску;
- критерії для оцінки сервісів;
- критерії для порівняння сервісів;
- кількість експериментів для побудови графіка залежності часу виконання функції від об’ємів даних;
- кількість кластерів;
- ім’я колонки для розрахування лінійної регресії;
- шлях до даних з викидами;
- шлях до даних для прогнозу.

7 ВИХІДНІ ДАНІ

Вихідними даними програми є:

- результати обробки даних для вибраних сервісів;
- показники якості результатів, якщо відповідна опція була обрана;
- графік залежності часу виконання функції від об’ємів даних, якщо відповідна опція була обрана;
- графік розподілення кластерів, якщо була обрана функція кластеризації;
- повідомлення про помилки програми — текст англійською мовою.

8 ОПИС ПРИЗНАЧЕНОГО ДЛЯ КОРИСТУВАЧА ІНТЕРФЕЙСУ

8.1 Опис станів програми

Стани, в яких може знаходитись програма наведено у табл. 8.1.

Таблиця 8.1 – Повідомлення користувачу

№ стану	Стан	Опис стану	Рекомендовані дії
1	Завантаження програми	Програма завантажується до оперативної пам'яті	Почекайте, доки програма запуститься
2	Відкрита програма з незаповненими параметрами	Щойно запущена програма з усіма незаповненими полями	Починайте працювати у програмі
3	Програма у процесі отримання параметрів	Користувач поступово обирає параметри дослідження.	Введіть усі параметри у зручному порядку
4	Програма з виставленими параметрами обробки	Система отримала всі дані, необхідні для обробки даних	Натисність кнопку «Process» для початку обробки даних, або змініть параметри.
5	Програма у процесі обробки даних	Програма опрацьовує отримані дані	Дочекайтеся доки програма завершить обробку даних.
6	Програма закінчила обробку даних та вивела результати	Програма готова до проведення нового експерименту	Перегляньте отримані результати. Почніть новий експеримент, або завершіть користування програмою

8.2 Опис переходів між станами програми

Схема переходів між станами програми представлено на рис. 8.1. Вихід з програми допускається з усіх станів, окрім 1.

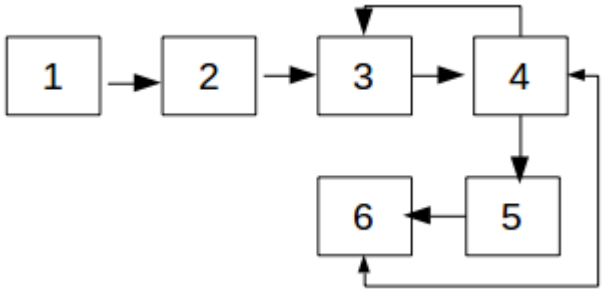


Рисунок 8.1 – Схема переходів між станами

8.3 Опис керування діалогом

Робота в програмі виконується за такими напрямками:

– підготовка вхідних даних для початку проведення експерименту;
– вивід результатів експерименту.

На рис. 5.1 представлений зовнішній вигляд програми після відкриття. Для того, щоб почати експеримент необхідно послідовно виконати наступні дії:

– вказати файлу з даними;
– вибрати функцію з випадаючого списку;
– за бажанням розташування увімкнути попередню нормалізацію даних;
– за бажанням увімкнути попереднє видалення викидів в даних;
– обрати один або більше сервісів, що досліджуються;
– за необхідністю вибрати критерії оцінки результатів сервісу;
– за необхідністю увімкнути автоматичне порівняння сервісів;
– у діалоговому режимі вказати необхідні дані, що можуть бути необхідні в залежності від вибраної функції та параметрів оцінки сервісу.

натиснути кнопку «Process» для початку обробки даних, після чого робоче вікно заблокується для вводу даних і розпочнеться опрацювання інформації. (рис. 8.2).

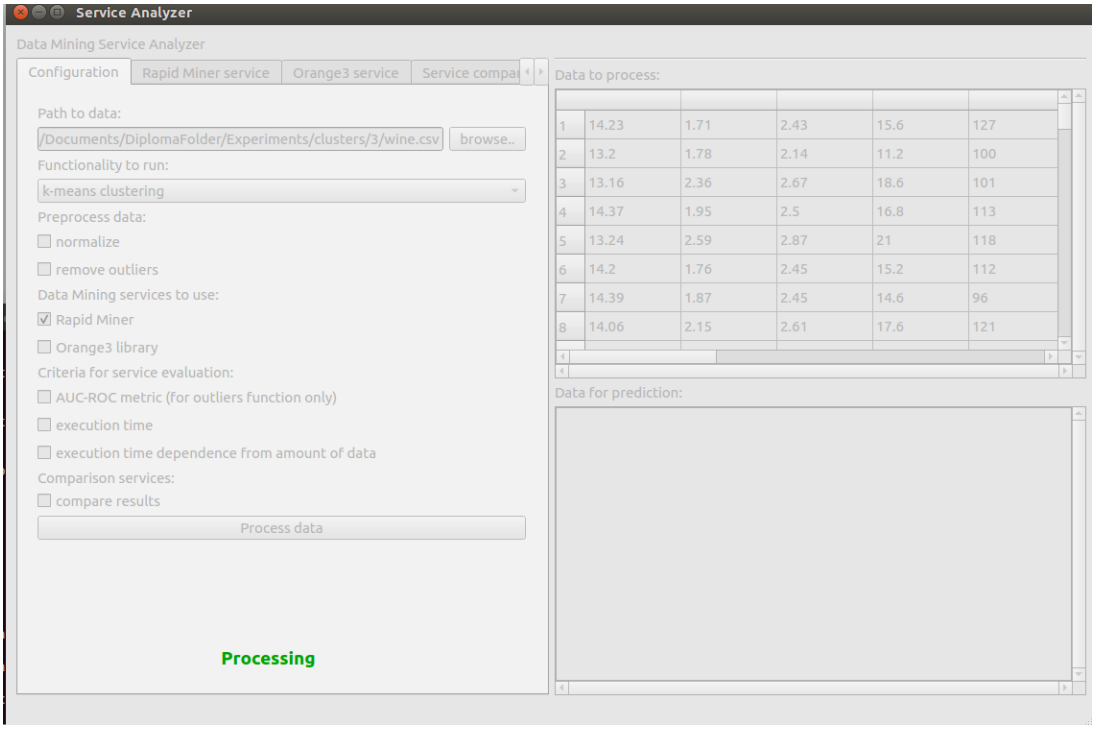


Рисунок 8.2 – Вікно опрацювання даних

Після того, як програма завершила обчислення, робоче вікно буде розблоковано (рис. 8.3).

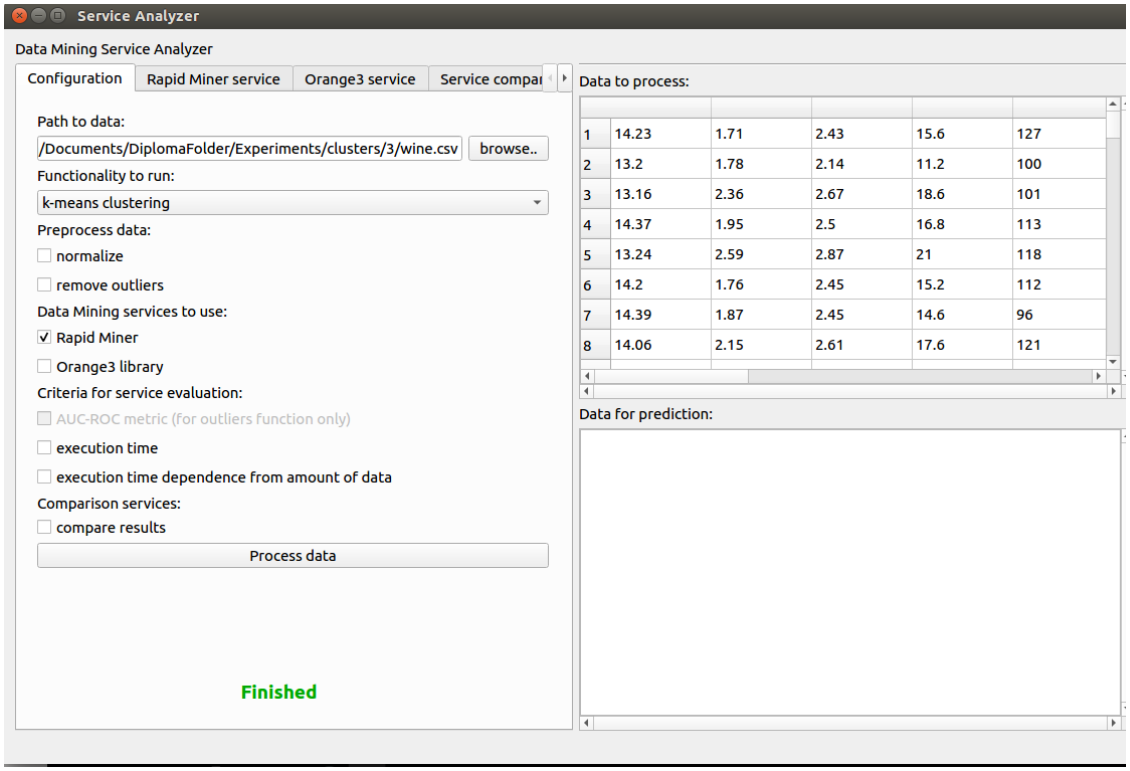


Рисунок 8.3 – Вікно завершення опрацювання даних

У разі якщо під час обчислень програма отримала помилку, з’явиться відповідне вікно з деталями проблеми (рис. 8.4).

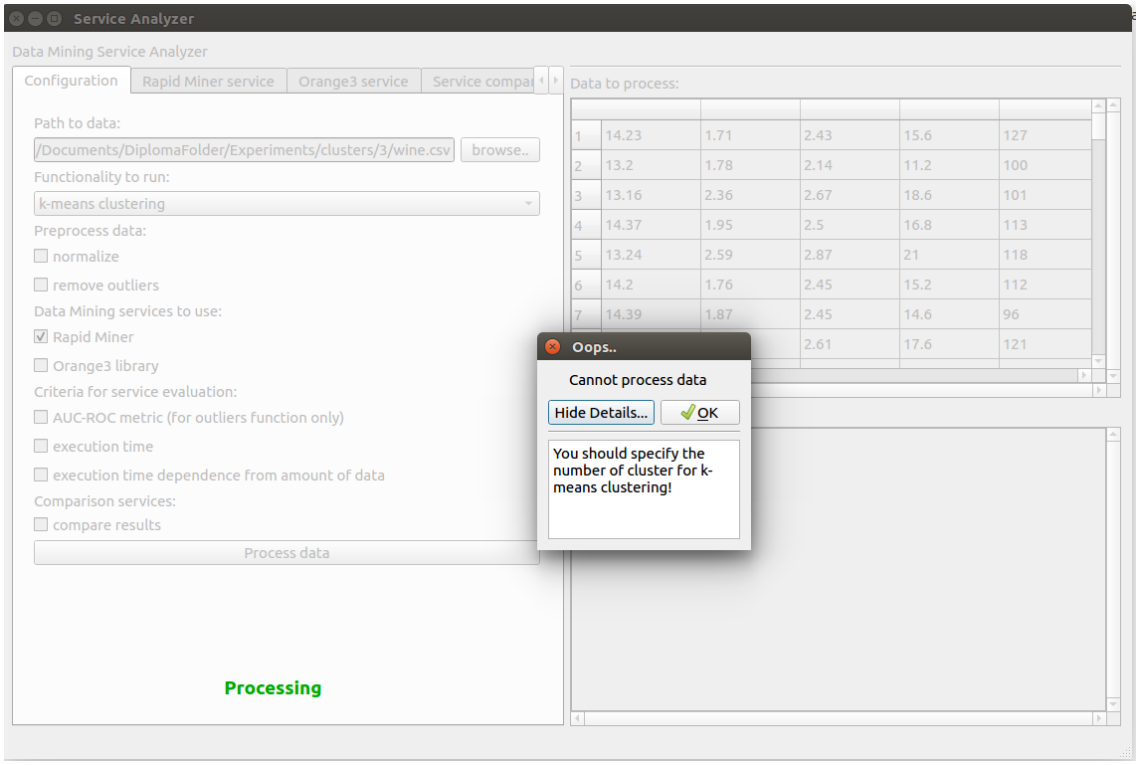


Рисунок 8.4 – Вікно про помилку

Результати отримані від сервісів можна переглянути у відповідних вкладках «Rapid Miner Service» та «Orange3 Service» (рис 8.5, 8.6).

Якщо користувач вибрав опцію «compare results», то у вкладці «Service comparison» будуть відображені результати порівняння сервісів (рис. 8.7).

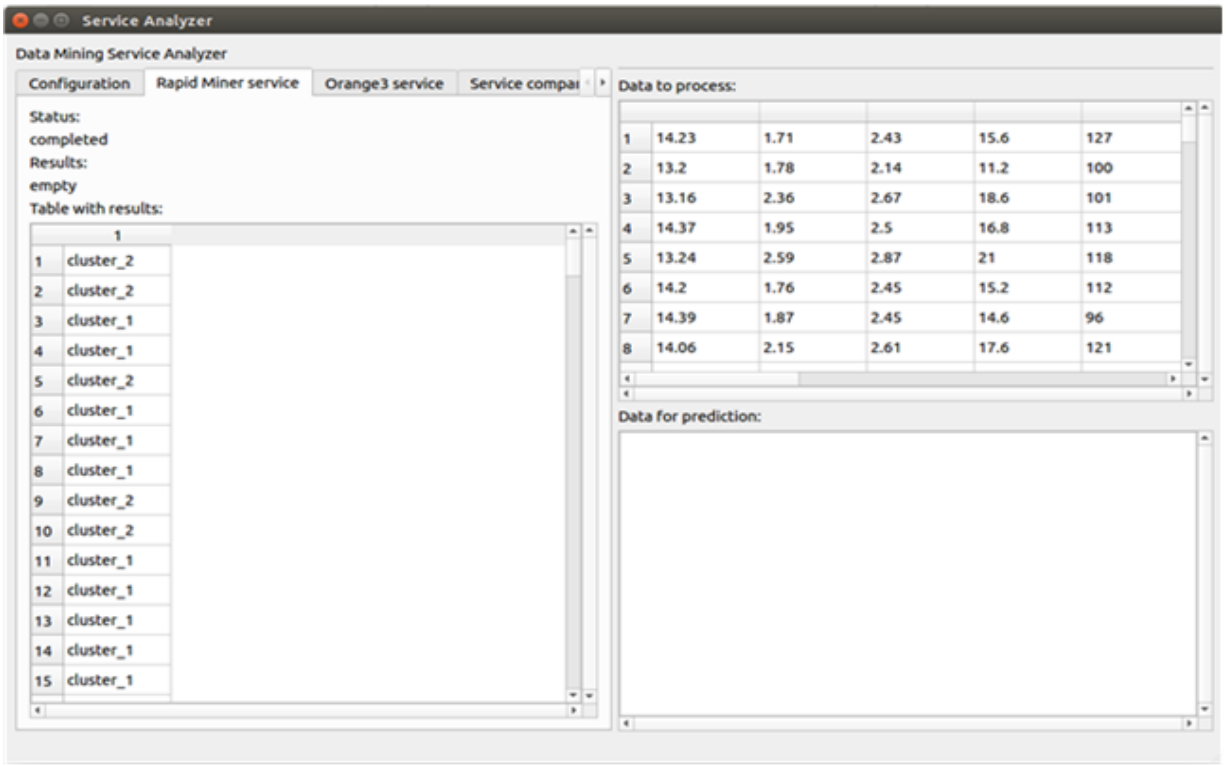


Рисунок 8.5 – Результати сервісу Rapid Miner

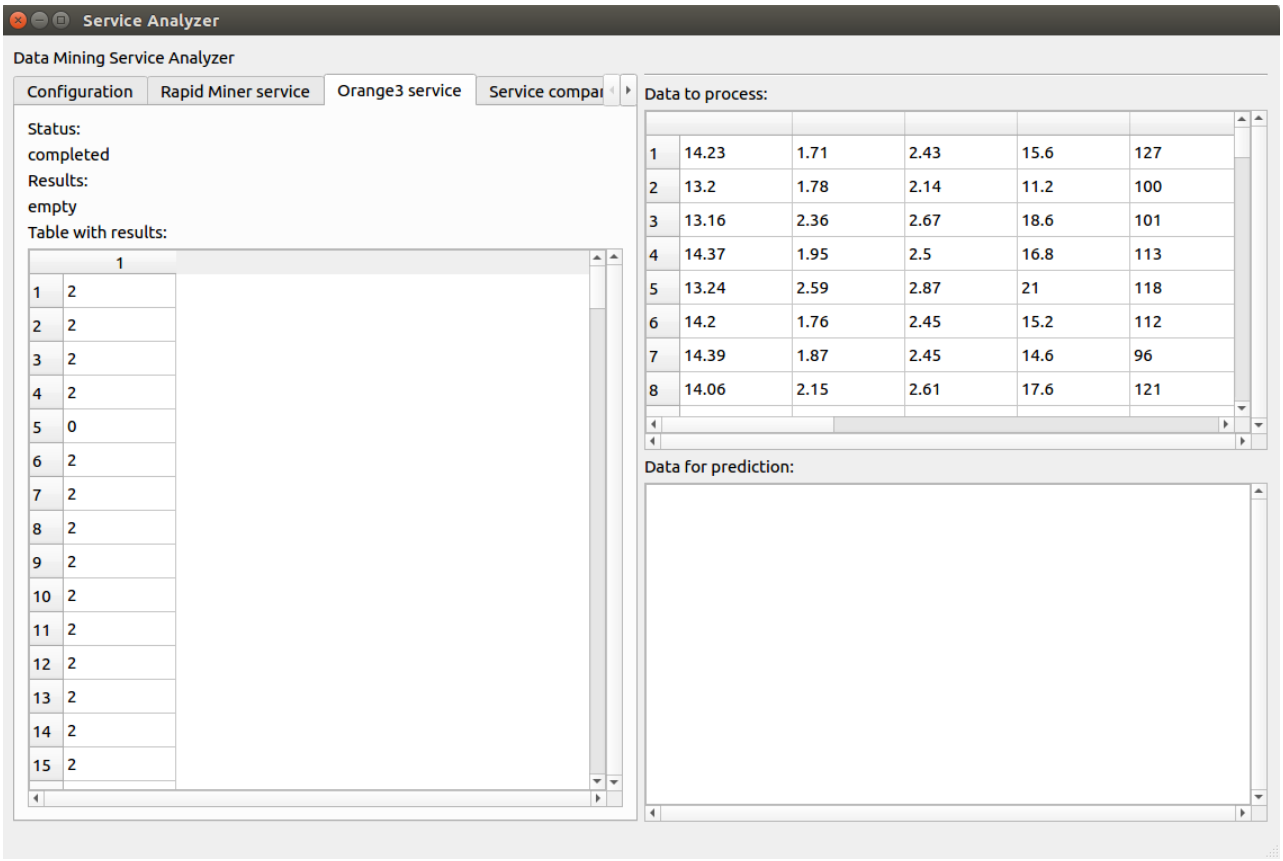


Рисунок 8.6 – Результати сервісу Orange3

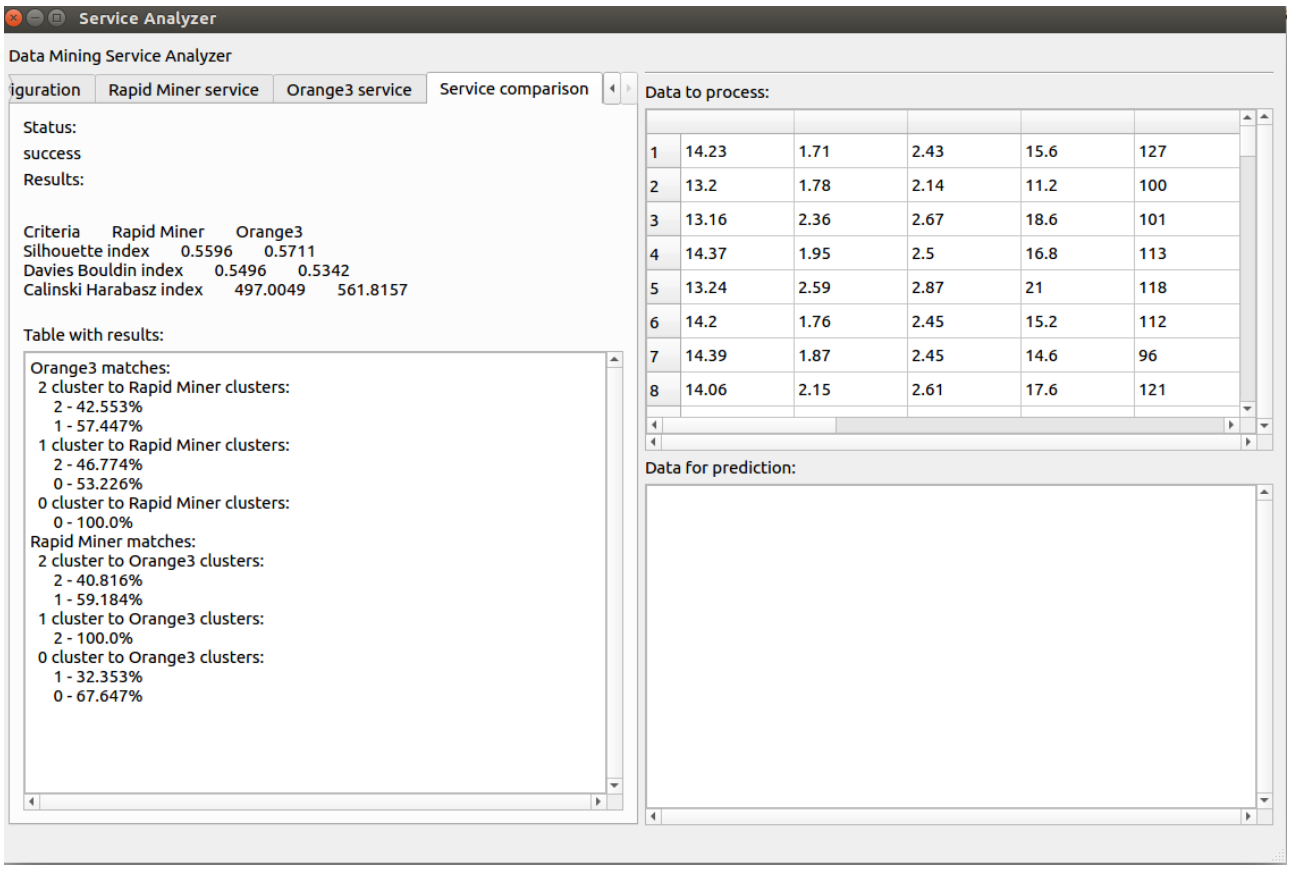


Рисунок 8.7 – Порівняння сервісів

8.4 Формування екранів

Програма має головне вікно для налаштування дослідження, 2 вікна для відображення результатів конкретного сервісу та вікно з результати порівняння сервісів. В залежності від обраних параметрів дослідження, можуть з’являтися додаткові вікна з відображенням графіків.

Головне вікно програми було представлене на рис. 5.1, вікна з результатами роботи сервісів зображено на рис. 8.5-8.6. Вікно з порівнянням сервісів зображено на рис. 8.7.

9 ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

У разі виникнення питань, пов’язаних з продуктом «Data Mining Service Analyzer», можна подати запит за електронною адресою: dmanalyzer@tracking.com. Підтримка з 9:00 до 18:00 по буднях.

10 ПОВІДОМЛЕННЯ

У табл. 10.1 наведені повідомлення користувачу, що можуть з’явитись у процесі роботи з програмою.

Таблиця 10.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
1	2	3
Вкажіть шлях до набору даних для якого необхідно передбачити значення	Була обрана функція «prediction» та натиснута кнопка «Process».	Користувач повинен вказати шлях до набору з даними для якого потрібно виконати прогнозування.
Ви не вказали шлях!	Користувач не вказав шлях до набору з даними для якого потрібно виконати прогнозування.	Натиснути кнопку «Process» та вказати шлях до даних у діалоговому вікні.
Введіть ім’я колонки для передбачення значень	Була обрана функція «prediction», натиснута кнопка «Process» та обраний шлях до набору з даними.	Користувач повинен вказати ім’я колонки для якої потрібно виконати прогнозування.
Ви не вказали ім’я колонки!	Користувач не вказав ім’я колонки для якої потрібно виконати прогнозування.	Натиснути кнопку «Process» та вказати ім’я колонки у діалоговому вікні.
Введіть ім’я колонки відносно якої буде розраховано рівняння.	Була обрана функція «linear regression» та натиснута кнопка «Process».	Ввести ім’я колонки відносно якої будуть розраховані коефіцієнти лінійної регресії.
Вкажіть шлях до файлу з поміченими викидами.	Була обрана функція «outliers», увімкнута опція «AUC-ROC score» та натиснута кнопка «Process».	Необхідно вказати шлях до файлу, де вказані викиди.

Продовження табл. 10.1

1	2	3
Вкажіть число експериментів.	увімкнута опція «execution time depence from amount of data» та натиснута кнопка «Process».	Необхідно ввести кількість експериментів для яких буде розрахований час виконання.
Число експериментів повинно бути вказане	Користувач не вказав кількість експериментів для побудови графіку залежності часу виконання від об’єму даних.	Натиснути кнопку «Process» та вказати кількість експериментів для яких буде розрахований час виконання.
Попереднє видалення викидів дозволено лише для функцій знаходження лінійної регресії та передбачення.	Була обрана функція «linear regression», «k-means clustering» або «prediction», увімкнута опція «remove outliers» та натиснута кнопка «Process».	Вимкнути опцію «remove outliers» та натиснути кнопку «Process».
Помилка при опрацюванні даних!	При опрацюванні даних програмою виникла помилка.	Ознайомитися з деталями помилки та виконати відповідні дії для її усунення.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Пілігрим, М. Dive into Python. [Текст] / М. Пілігрим;2004. – 487 с.
2. Павловська, Т. А. C/C++. Програмування на мові високого рівня / Т. А. Павлівська. – СПб.: Пітер, 2010. – 461 с.

ЗАТВЕРДЖЕНО

01116130.01182 -01 ІЗ 01-ЛЗ

ІНСТРУМЕНТ ДЛЯ ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ СЕРВІСІВ DATA MINING

Керівництво користувача. Керівництво для порівняння сервісів

01116130.01182-01 ІЗ 01

Листів 14

2020

АНОТАЦІЯ

Документ 01116130.01182-01 ІЗ 01 «Інструмент для дослідження та порівняння сервісів Data Mining. Керівництво для порівняння сервісів» входить до складу програмної документації до програми.

Програмний продукт дозволяє виконувати різні варіанти обробки даних сервісами Rapid Miner та Orange. Також представляє функціонал для ефективного порівняння отриманих результатів.

ЗМІСТ

ВСТУП

Інструмент для дослідження сервісів технології Data Mining дозволяє зробити висновки щодо вибору кращого сервісу серед запропонованих для вирішення конкретної задачі на вказаних даних.

Сфера застосування: глибинний аналіз даних.

Користувачі системи для дослідження сервісів технології Data Mining – дослідники, які займаються здобуттям корисної інформації з існуючих даних.

Користувачі системи повинні мати досвід роботи з ПК та використання операційної системи Ubuntu 16.04.

Адміністратор системи повинен володіти досвідом встановлення та налаштування програмних та апаратних засобів у платформі GNU\Linux.

Практичне значення одержаних результатів полягає у використанні розробленої програми для вибору кращого сервісу для подальшого аналізу даних.

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Інструмент для дослідження і порівняння сервісів технології Data Mining призначений для вивчення ефективності сервісів при вирішенні таких задач як: кластеризація, знаходження викидів, пошук рівняння лінійної регресії та прогнозування. Результат дослідження дозволить зрозуміти, який сервіс краще використовувати для рішення конкретної задачі. Дану інформацію можна використовувати для оптимізації процесів глибинного аналізу даних.

Умови застосування системи:

- ЕВМ, що має щонайменше 20 Гб вільного місця на вбудованому носії інформації та 4096 Мб оперативної пам’яті;
- монітор з роздільною здатністю 1366x768;
- маніпулятор «миша» та клавіатуру;
- операційна система Ubuntu.

2 ПІДГОТОВКА ДО РОБОТИ

На дистрибутивному носії даних розміщений виконуваний файл ServiceAnalyzer за допомогою якого можна запустити автоматизовану систему та необхідні для повноцінної роботи зовнішні бібліотеки.

Для запуску програмного продукту «Інструмент для дослідження і порівняння сервісів технології Data Mining» необхідно виконати запуск файлу ServiceAnalyzer з терміналу Linux.

3 ОПИС ОПЕРАЦІЙ

Керівництво користування програмою «Інструмент для дослідження і порівняння сервісів технології Data Mining» складається з наступних операцій:

- вибір набору даних;
- вибір відповідної функції для обробки;
- вибір параметрів для попередньої обробки даних;
- вибір сервісів для обробки даних;
- вибір критеріїв для оцінки результатів сервісу;
- вибір критеріїв для порівняння сервісів;
- запуск обробки даних та введення додаткової інформації;
- аналіз отриманих результатів.

3.1 Вибір набору даних

Для початку проведення експерименту вкажіть шлях до набору даних. Варто зазначити, що дані повинні зберігатися у форматі csv. Для того щоб вказати шлях необхідно введіть його вручну в поле (рис. 3.1), або виберіть через діалогове вікно, натиснувши кнопку «browse..» (рис. 3.2, 3.3). У разі якщо планується обробляти дані функцією «linear regression» або «prediction», таблиця об’язково повинна містити ім’я атрибутів, інакше виникне помилка під час роботи програми.

3.2 Вибір функції для обробки

Після того, як був вказаний шлях до даних, вкажіть функцію, яка буде застосована для нього. Під час одного експерименту можна запускати лише одну із реалізованих нині задач:

- кластеризація методом k-середніх (k-means clustering);
- пошук викидів в даних за алгоритмом LOF (outliers);
- знаходження рівняння лінійної регресії (linear regression);
- передбачення даних (prediction).

В залежності від того, яка функція буде обрана для обробки, в подальшому можуть знадобитися додаткові дані. За замовчуванням обрана функція знаходження викидів.

Імена функцій розташовані у випадаючому списку, для того, щоб переглянути всі варіанти необхідно натиснути лівою клавішею миші на відповідне поле (рис.3.4).

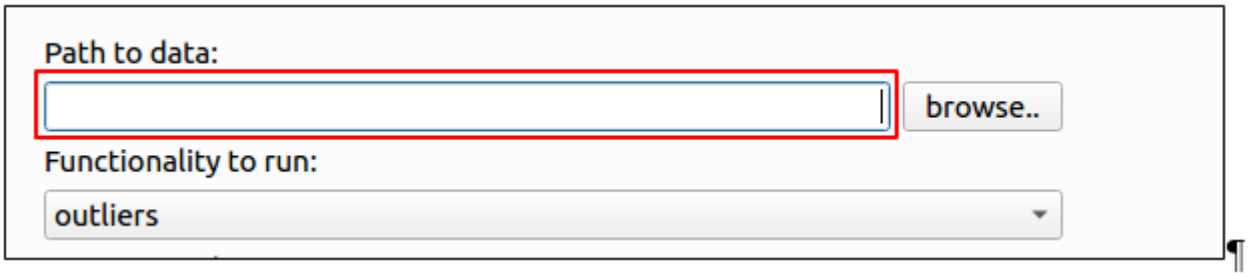


Рисунок 3.1 – Поле вводу шляху до даних

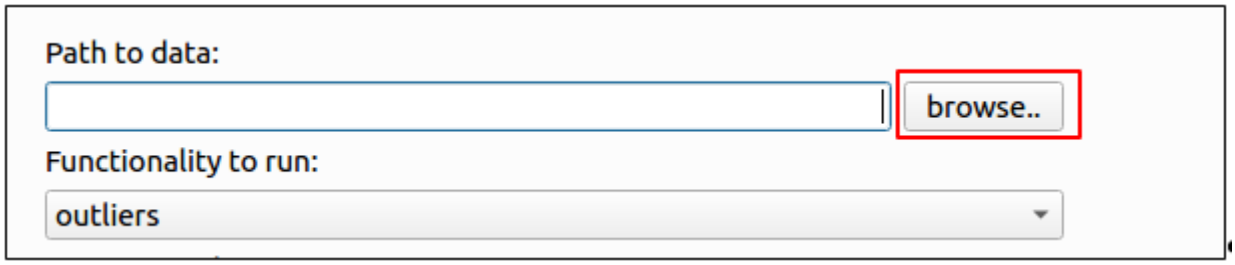


Рисунок 3.2 – кнопка для вибору файлу через діалогове вікно

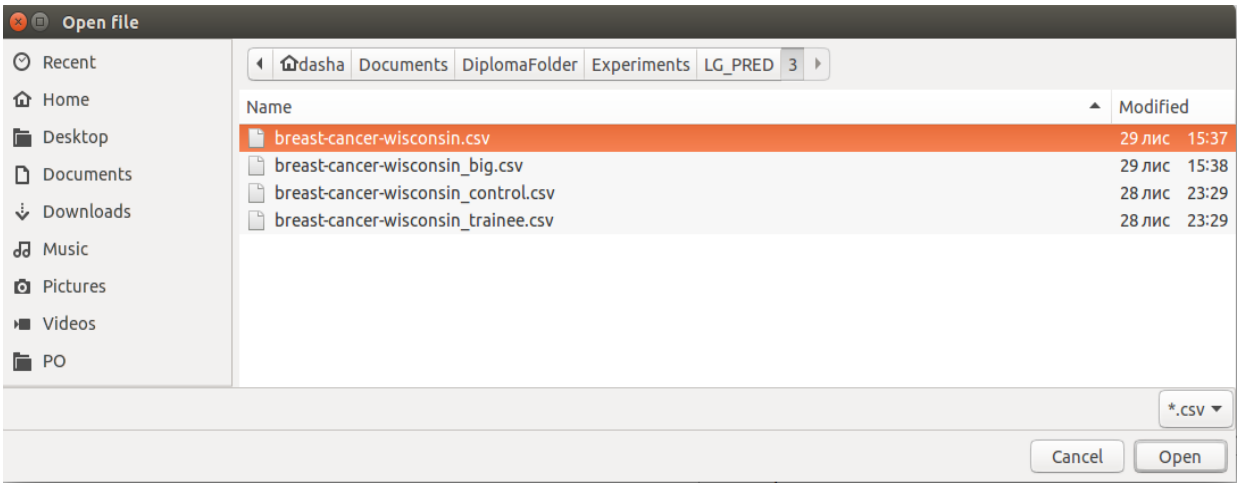


Рисунок 3.3 – Діалогове вікно для вибору файлу формату csv

3.3 Вибір параметрів для попередньої обробки даних

Після того, як було обрано функцію, оберіть операції які будуть застосовані на даних, перед їх обробкою. Дані опції можуть поліпшити або навпаки погіршити якість отриманих результатів в залежності від набору даних та задачі.

Рисунок 3.4 – Випадаючий список для вибору функції

На даному етапі реалізації запропоновані наступні методи попередньої обробки даних:

– нормалізація (normalize);

– видалення викидів (remove outliers).

За замовчуванням ці опції увімкнені (рис. 3.5). Пам’ятайте, що слід вимикати опцію видалення викидів, якщо була обрана функція знаходження викидів.

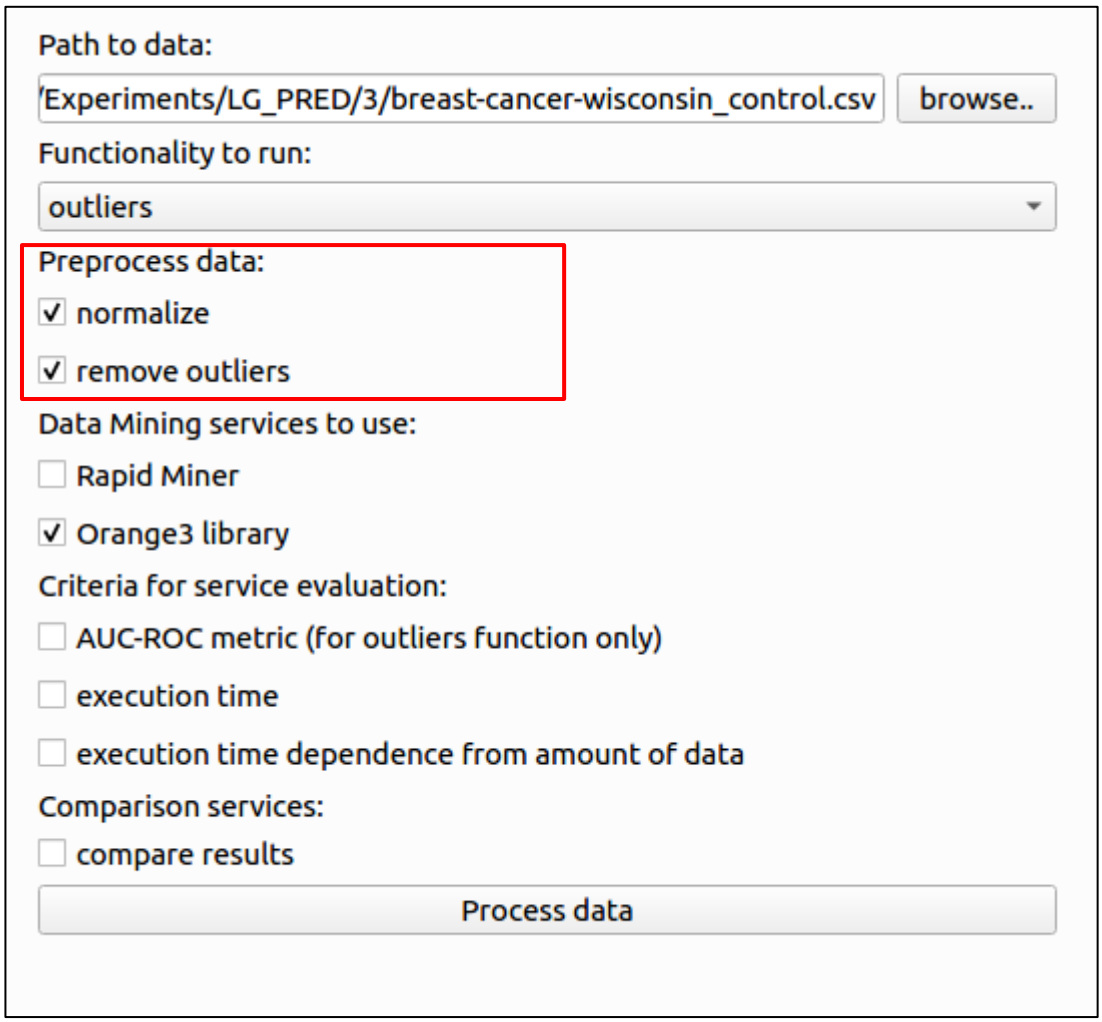


Рисунок 3.5 – Функції попередньої обробки даних

3.4 Вибір сервісів для обробки даних

Виберіть один або два сервіси, які будуть здійснювати обробку даних. За замовчуванням вибраний сервіс Orange. Щонайменше один з запропонованих сервісів повинен бути відмічений галочкою. На даний момент для порівняння доступні наступні:

– Rapid Miner;
– Orange.

3.5 Вибір критеріїв для оцінки результатів сервісу

Наступним кроком оберіть параметри, за якими буде оцінюватися сервіс. За замовчуванням жодний із параметрів не обраний. Варто зазначити, що деякі параметри оцінки доступні лише для певних функцій. Таким чином, опція для отримання AUC-ROC метрики є ввімкненою лише якщо вибрана функція знаходження викидів (рис. 3.6).

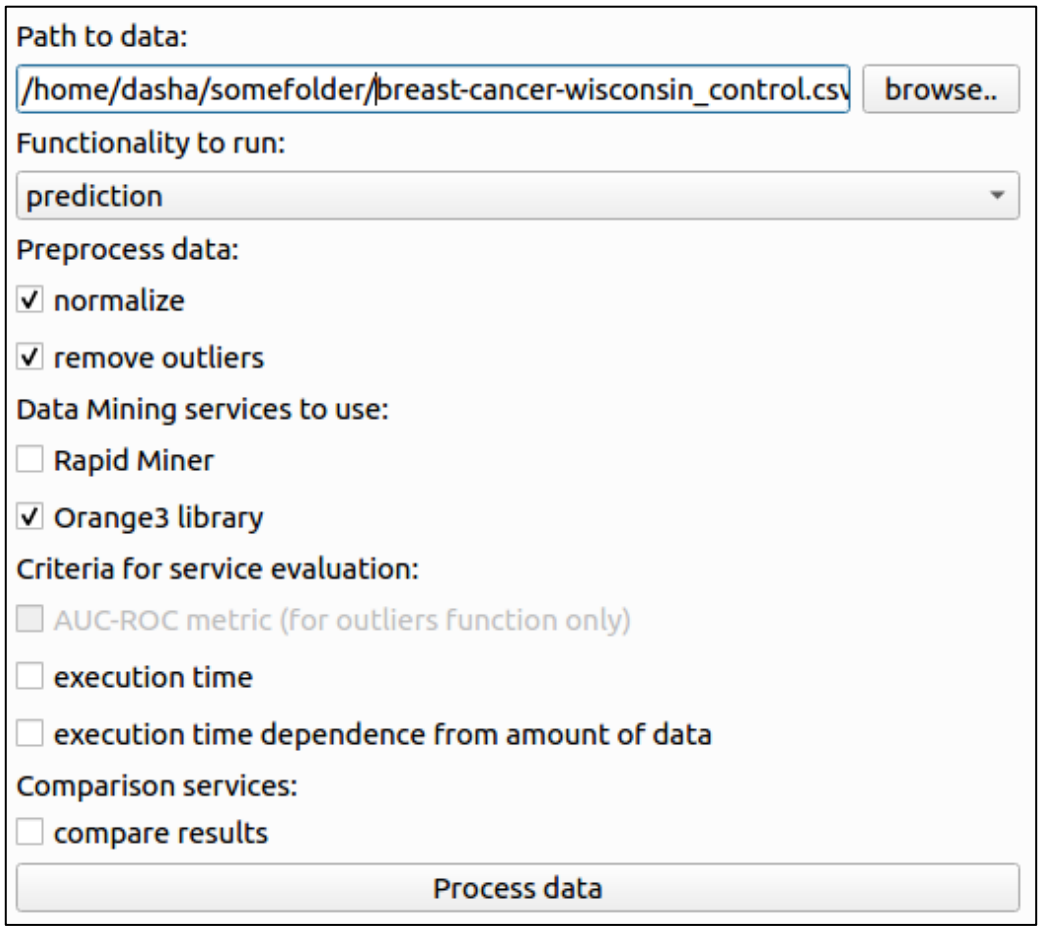


Рисунок 3.6 – опція «AUC-ROC metric» увімкнена лише для функції «outliers»

3.6 Вибір критеріїв для порівняння сервісів

За бажанням виберіть критерії за якими будуть порівнюватися сервіси. На даний момент реалізована лише одна опція порівняння сервісів – порівняння за отриманими результатами. Зауважте, для того, щоб цей функціонал відпрацював необхідно вибрати обидва сервіси, інакше виникне помилка програми.

3.7 Запуск обробки даних та введення додаткової інформації

Після того, як усі параметри були налаштовані, натисніть кнопку «Process». Відповідно до обраних конфігурацій, програма може запросити додаткові дані, що необхідні для початку експерименту (рис.3.7).

Якщо була обрана функція лінійної регресії або передбачення, введіть у діалоговому вікні, що з’явилось, ім’я колонки відносно якої буде створено рівняння лінійної регресії.

Якщо була обрана функція передбачення, після вводу імені колонки, вкажіть шлях до даних, для яких буде створено прогноз.

Якщо була обрана функція кластеризації методом k-середніх, у новому діалоговому вікні введіть кількість кластерів.

Якщо була обрана функція побудови графіка, введіть кількість експериментів, які необхідно провести.

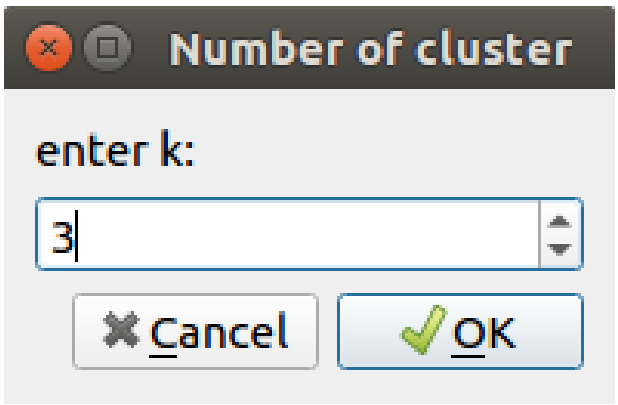


Рисунок 3.7 – Приклад діалогового вікна, що може з’явитися після того, як була натиснута кнопка «Process data»

3.8 Аналіз отриманих результатів.

Після того як інформація була оброблена, ознайомтеся з отриманими результатами. Перегляньте підраховані метрики, та дані, що були отримані в результаті виконання функцій для кожного із сервісів. Інформація стосовно сервісів розташована у вкладках «Rapid Miner Service» та «Orange Service» відповідно. Дані щодо порівняння сервісів розташовані у вкладці «Service Comparison» (рис. 3.8).

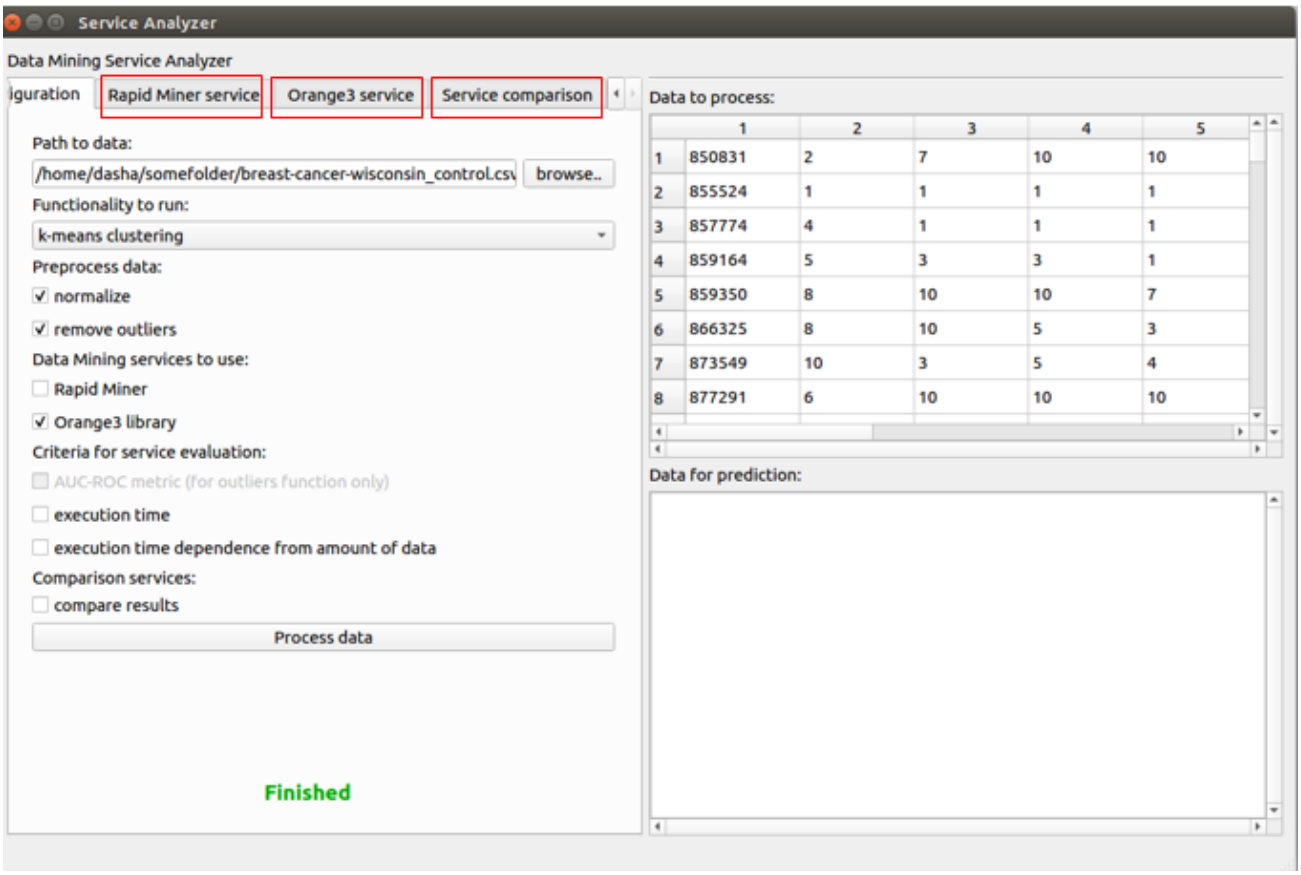


Рисунок 3.8 – Вкладки з результатами обчислень

Щоб провести наступний експеримент, введіть нові параметри та натисніть кнопку «Process data».

4 АВАРІЙНІ СИТУАЦІЇ

Під час експлуатації інструменту для аналізу та дослідження сервісів технології Data Mining можуть виникати аварійні ситуації. Якщо під час роботи системи програма буде завершувати роботу з повідомленням про те, що не вдалося виділити достатній об’єм пам’яті, спробуйте збільшити об’єм вільного місця на жорсткому диску до рекомендованого і повторіть експеримент. У разі зависання програми, або інших проблем, зверніться до технічної підтримки.

5 РЕКОМЕНДАЦІЇ ЩОДО ЗАСВОЄННЯ

Під час проведення експерименту не навантажуйте CPU комп'ютера, оскільки це може призвести до сповільнення роботи програми, або навіть до її зависання.

Контрольний приклад: запустіть програму через термінал Linux, вкажіть шлях до набору даних, оберіть обидва сервіси для обробки даних і натисніть кнопку «Process», після того, як програма опрацює інформацію, перевірте отримані результати у вкладках «Orang3 Service» та «Rapid Miner Service»

ЗАТВЕРДЖЕНО

01116130.01182-01 12 01-ЛЗ

ІНСТРУМЕНТ ДЛЯ ДОСЛІДЖЕННЯ ТА ПОРІВНЯННЯ СЕРВІСІВ DATA MINING

Текст програми

01116130.01182-01 12 01

Листів 29

2020

АНОТАЦІЯ

Документ 01116130.1182-01«Інструмент для дослідження та порівняння сервісів Data Mining. Текст програми» входить до складу програмної документації до програми, яка реалізує програмне забезпечення «Data Mining Service Analyzer».

В документі міститься опис основних модулів програми та їх текст. Програма реалізована на мові python у програмному середовищі PyCharm 2019.

ЗМІСТ

1 СХЕМА ВЗАЄМОДІЇ ПРОГРАМНИХ МОДУЛІВ

На рис. 1.1 представлена схема взаємодії основних модулів програми.

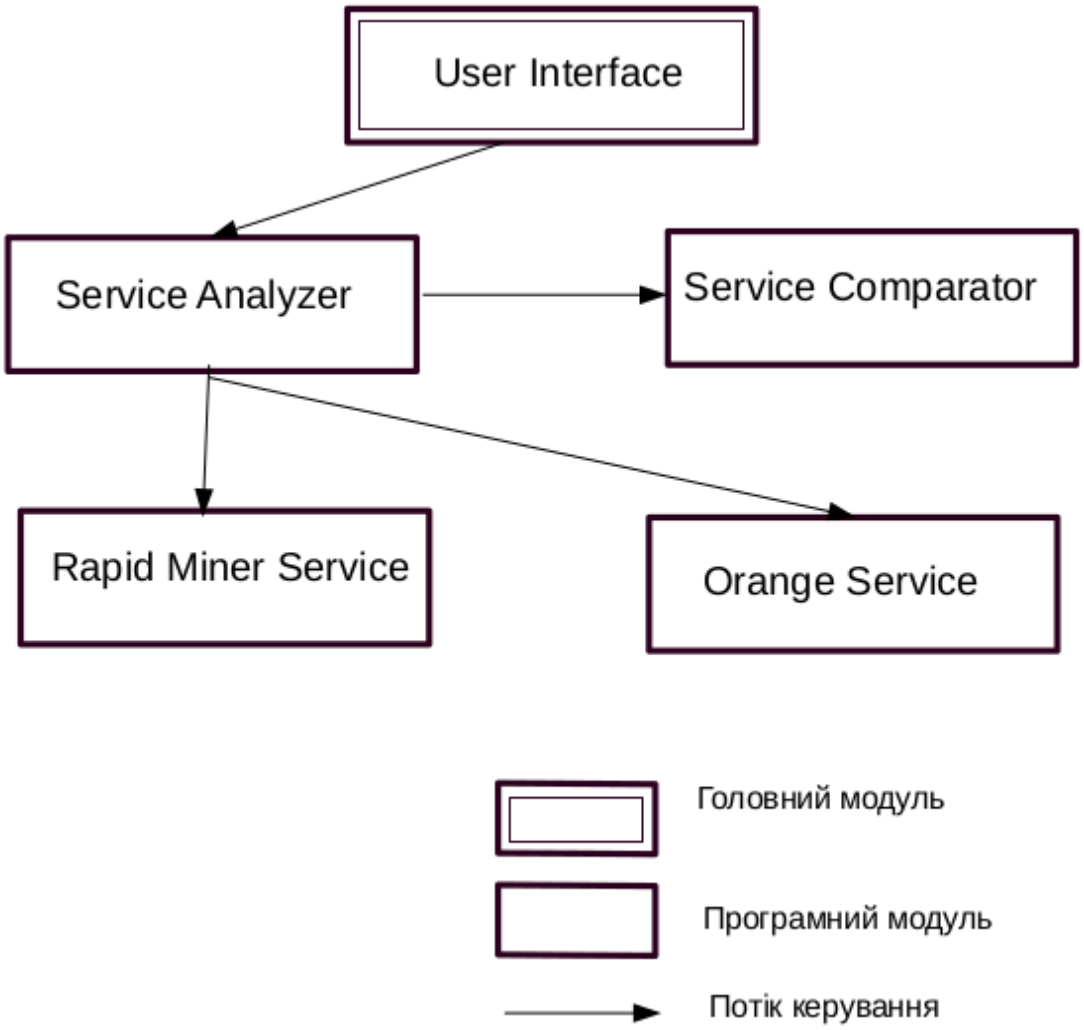


Рисунок 1.1 – Схема взаємодії основних модулів програми

User Interface – є головним модулем програми. Відповідає за отримання налаштувань експерименту від користувача та їх передачу до програми (ServiceAnalyzer) а також демонстрацію даних користувачеві. Відповідає за початок та завершення програми.

Service Analyzer –представляє собою основну логіку програми та відповідає за реалізацію процесу обробки даних. Необхідні дані отримує від модулю User Interface. У якості допоміжних модулів для реалізації експерименту використовує Orange Service, Rapid Miner Service. Процес порівняння сервісів виконує за допомогою модулю Service Comparator.

Service Comparator – реалізує логіку порівняння результатів двох сервісів. Вхідні дані отримує від модулю Service Analyzer.

Rapid Miner Service – модуль Rapid Miner Service є відповідальним за реалізацію функцій сервісом Rapid Miner. */

Orange Service – модуль Orange Service є відповідальним за реалізацію функцій сервісом Orange.

2 ТЕКСТ ПРОГРАМИ

2.1 Модуль User Interface

2.1.1 Текст програми у файлі mainwindow.py

```
# -*- coding: utf-8 -*-
from PyQt5 import QtCore, QtGui, QtWidgets
class Ui_MainWindow(object):
  def setupUi(self, MainWindow):
    MainWindow.setObjectName("MainWindow")
    MainWindow.setEnabled(True)
    MainWindow.resize(1137, 731)
    self.centralWidget = QtWidgets.QWidget(MainWindow)
    self.centralWidget.setObjectName("centralWidget")
    self.verticalLayout_2 = QtWidgets.QVBoxLayout(self.centralWidget)
    self.verticalLayout_2.setContentsMargins(11, 11, 11, 11)
    self.verticalLayout_2.setSpacing(6)
    self.verticalLayout_2.setObjectName("verticalLayout_2")
    self.label_program_name = QtWidgets.QLabel(self.centralWidget)
    self.label_program_name.setObjectName("label_program_name")
    self.verticalLayout_2.addWidget(self.label_program_name)
    self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
    self.horizontalLayout_2.setSpacing(6)
    self.horizontalLayout_2.setObjectName("horizontalLayout_2")
    self.tabWidget = QtWidgets.QTabWidget(self.centralWidget)
    self.tabWidget.setEnabled(True)
    self.tabWidget.setLayoutDirection(QtCore.Qt.LeftToRight)
    self.tabWidget.setStyleSheet("")
    self.tabWidget.setObjectName("tabWidget")
    self.tab_config = QtWidgets.QWidget()
    self.tab_config.setObjectName("tab_config")
    self.verticalLayoutWidget_6 = QtWidgets.QWidget(self.tab_config)
    self.verticalLayoutWidget_6.setGeometry(QtCore.QRect(9, 9, 531, 475))
    self.verticalLayoutWidget_6.setObjectName("verticalLayoutWidget_6")
    self.verticalLayout_0 = QtWidgets.QVBoxLayout(self.verticalLayoutWidget_6)
    self.verticalLayout_0.setContentsMargins(11, 11, 11, 11)
    self.verticalLayout_0.setSpacing(6)
    self.verticalLayout_0.setObjectName("verticalLayout_0")
    self.label_2 = QtWidgets.QLabel(self.verticalLayoutWidget_6)
    self.label_2.setStyleSheet("bold")
    self.label_2.setObjectName("label_2")
    self.verticalLayout_0.addWidget(self.label_2)
    self.verticalLayout_1 = QtWidgets.QVBoxLayout()
    self.verticalLayout_1.setSpacing(6)
    self.verticalLayout_1.setObjectName("verticalLayout_1")
    self.horizontalLayout = QtWidgets.QHBoxLayout()
    self.horizontalLayout.setSpacing(6)
    self.horizontalLayout.setObjectName("horizontalLayout")
    self.edit_data_path = QtWidgets.QLineEdit(self.verticalLayoutWidget_6)
    self.edit_data_path.setMouseTracking(True)
    self.edit_data_path.setLayoutDirection(QtCore.Qt.LeftToRight)
    self.edit_data_path.setInputMethodHints(QtCore.Qt.ImhNone)
    self.edit_data_path.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)
    self.edit_data_path.setObjectName("edit_data_path")
    self.horizontalLayout.addWidget(self.edit_data_path)
    self.btn_browse_path = QtWidgets.QPushButton(self.verticalLayoutWidget_6)
    self.btn_browse_path.setObjectName("btn_browse_path")
    self.horizontalLayout.addWidget(self.btn_browse_path)
    self.verticalLayout_1.addLayout(self.horizontalLayout)
    self.verticalLayout_0.addLayout(self.verticalLayout_1)
    self.verticalLayout = QtWidgets.QVBoxLayout()
    self.verticalLayout.setSpacing(6)
    self.verticalLayout.setObjectName("verticalLayout")
    self.label_3 = QtWidgets.QLabel(self.verticalLayoutWidget_6)
```

```
self.label_3.setObjectName("label_3")
self.verticalLayout.addWidget(self.label_3)
self.list_functionality = QtWidgets.QComboBox(self.verticalLayoutWidget_6)
self.list_functionality.setObjectName("list_functionality")
self.list_functionality.addItem("")
self.list_functionality.addItem("")
self.list_functionality.addItem("")
self.list_functionality.addItem("")
self.verticalLayout.addWidget(self.list_functionality)
self.verticalLayout_10 = QtWidgets.QVBoxLayout()
self.verticalLayout_10.setSpacing(6)
self.verticalLayout_10.setObjectName("verticalLayout_10")
self.verticalLayout_11 = QtWidgets.QVBoxLayout()
self.verticalLayout_11.setSpacing(6)
self.verticalLayout_11.setObjectName("verticalLayout_11")
self.label_13 = QtWidgets.QLabel(self.verticalLayoutWidget_6)
self.label_13.setObjectName("label_13")
self.verticalLayout_11.addWidget(self.label_13)
self.cBx_normalize = QtWidgets.QCheckBox(self.verticalLayoutWidget_6)
self.cBx_normalize.setChecked(True)
self.cBx_normalize.setObjectName("cBx_normalize")
self.verticalLayout_11.addWidget(self.cBx_normalize)
self.cBx_remove_outlier = QtWidgets.QCheckBox(self.verticalLayoutWidget_6)
self.cBx_remove_outlier.setChecked(True)
self.cBx_remove_outlier.setObjectName("cBx_remove_outlier")
self.verticalLayout_11.addWidget(self.cBx_remove_outlier)
self.verticalLayout_10.addLayout(self.verticalLayout_11)
self.verticalLayout.addLayout(self.verticalLayout_10)
self.verticalLayout_0.addLayout(self.verticalLayout)
self.verticalLayout_3 = QtWidgets.QVBoxLayout()
self.verticalLayout_3.setSpacing(6)
self.verticalLayout_3.setObjectName("verticalLayout_3")
self.label_4 = QtWidgets.QLabel(self.verticalLayoutWidget_6)
self.label_4.setObjectName("label_4")
self.verticalLayout_3.addWidget(self.label_4)
self.cBx_rp_service_run = QtWidgets.QCheckBox(self.verticalLayoutWidget_6)
self.cBx_rp_service_run.setChecked(False)
self.cBx_rp_service_run.setObjectName("cBx_rp_service_run")
self.verticalLayout_3.addWidget(self.cBx_rp_service_run)
self.cBx_orange_service_run = QtWidgets.QCheckBox(self.verticalLayoutWidget_6)
self.cBx_orange_service_run.setChecked(True)
self.cBx_orange_service_run.setObjectName("cBx_orange_service_run")
self.verticalLayout_3.addWidget(self.cBx_orange_service_run)
self.verticalLayout_0.addLayout(self.verticalLayout_3)
self.verticalLayout_4 = QtWidgets.QVBoxLayout()
self.verticalLayout_4.setSpacing(6)
self.verticalLayout_4.setObjectName("verticalLayout_4")
self.label_5 = QtWidgets.QLabel(self.verticalLayoutWidget_6)
self.label_5.setObjectName("label_5")
self.verticalLayout_4.addWidget(self.label_5)
self.cBx_calculate_auc_roc = QtWidgets.QCheckBox(self.verticalLayoutWidget_6)
self.cBx_calculate_auc_roc.setEnabled(True)
self.cBx_calculate_auc_roc.setObjectName("cBx_calculate_auc_roc")
self.verticalLayout_4.addWidget(self.cBx_calculate_auc_roc)
self.cBx_find_execution_time = QtWidgets.QCheckBox(self.verticalLayoutWidget_6)
self.cBx_find_execution_time.setObjectName("cBx_find_execution_time")
self.verticalLayout_4.addWidget(self.cBx_find_execution_time)
self.cBx_find_time_dependency = QtWidgets.QCheckBox(self.verticalLayoutWidget_6)
self.cBx_find_time_dependency.setObjectName("cBx_find_time_dependency")
self.verticalLayout_4.addWidget(self.cBx_find_time_dependency)
self.verticalLayout_5 = QtWidgets.QVBoxLayout()
self.verticalLayout_5.setSpacing(6)
self.verticalLayout_5.setObjectName("verticalLayout_5")
self.label_6 = QtWidgets.QLabel(self.verticalLayoutWidget_6)
self.label_6.setObjectName("label_6")
self.verticalLayout_5.addWidget(self.label_6)
```

```
        self.cBx_compare_results = QtWidgets.QCheckBox(self.verticalLayoutWidget_6)
        self.cBx_compare_results.setObjectName("cBx_compare_results")
        self.verticalLayout_5.addWidget(self.cBx_compare_results)
        self.verticalLayout_4.addLayout(self.verticalLayout_5)
        self.verticalLayout_0.addLayout(self.verticalLayout_4)
        self.btn_process_data = QtWidgets.QPushButton(self.verticalLayoutWidget_6)
        self.btn_process_data.setObjectName("btn_process_data")
        self.verticalLayout_0.addWidget(self.btn_process_data)
        self.label_status = QtWidgets.QLabel(self.tab_config)
        self.label_status.setEnabled(False)
        self.label_status.setGeometry(QtCore.QRect(140, 570, 241, 51))
        font = QtGui.QFont()
        font.setFamily("Ubuntu")
        font.setPointSize(14)
        font.setBold(True)
        font.setItalic(False)
        font.setWeight(75)
        font.setStrikeOut(False)
        self.label_status.setFont(font)
        self.label_status.setStyleSheet("color:rgba(0, 170, 0, 0);")
        self.label_status.setAlignment(QtCore.Qt.AlignCenter)
        self.label_status.setObjectName("label_status")
        self.tabWidget.addTab(self.tab_config, "")
        self.tab_rp_service = QtWidgets.QWidget()
        self.tab_rp_service.setEnabled(True)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.tab_rp_service.sizePolicy().hasHeightForWidth())
        self.tab_rp_service.setSizePolicy(sizePolicy)
        self.tab_rp_service.setObjectName("tab_rp_service")
        self.verticalLayout_6 = QtWidgets.QVBoxLayout(self.tab_rp_service)
        self.verticalLayout_6.setContentsMargins(11, 11, 11, 11)
        self.verticalLayout_6.setSpacing(6)
        self.verticalLayout_6.setObjectName("verticalLayout_6")
        self.verticalLayout_16 = QtWidgets.QVBoxLayout()
        self.verticalLayout_16.setSpacing(6)
        self.verticalLayout_16.setObjectName("verticalLayout_16")
        self.label_20 = QtWidgets.QLabel(self.tab_rp_service)
        self.label_20.setEnabled(True)
        self.label_20.setStyleSheet("bold")
        self.label_20.setObjectName("label_20")
        self.verticalLayout_16.addWidget(self.label_20)
        self.label_rp_status = QtWidgets.QLabel(self.tab_rp_service)
        self.label_rp_status.setObjectName("label_rp_status")
        self.verticalLayout_16.addWidget(self.label_rp_status)
        self.verticalLayout_6.addLayout(self.verticalLayout_16)
        self.verticalLayout_17 = QtWidgets.QVBoxLayout()
        self.verticalLayout_17.setSpacing(6)
        self.verticalLayout_17.setObjectName("verticalLayout_17")
        self.label_22 = QtWidgets.QLabel(self.tab_rp_service)
        self.label_22.setEnabled(True)
        self.label_22.setStyleSheet("bold")
        self.label_22.setObjectName("label_22")
        self.verticalLayout_17.addWidget(self.label_22)
        self.label_rp_results = QtWidgets.QLabel(self.tab_rp_service)

self.label_rp_results.setTextInteractionFlags(QtCore.Qt.LinksAccessibleByMouse|QtCore.Qt.TextSelectableByK
eyboard|QtCore.Qt.TextSelectableByMouse)
        self.label_rp_results.setObjectName("label_rp_results")
        self.verticalLayout_17.addWidget(self.label_rp_results)
        self.verticalLayout_6.addLayout(self.verticalLayout_17)
        self.label_7 = QtWidgets.QLabel(self.tab_rp_service)
        self.label_7.setObjectName("label_7")
        self.verticalLayout_6.addWidget(self.label_7)
        self.scrollArea_rp_results = QtWidgets.QScrollArea(self.tab_rp_service)
```

```
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.scrollArea_rp_results.sizePolicy().hasHeightForWidth())
        self.scrollArea_rp_results.setSizePolicy(sizePolicy)
        self.scrollArea_rp_results.viewport().setProperty("cursor", QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.scrollArea_rp_results.setMouseTracking(True)
        self.scrollArea_rp_results.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.scrollArea_rp_results.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.scrollArea_rp_results.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)
        self.scrollArea_rp_results.setWidgetResizable(True)
        self.scrollArea_rp_results.setObjectName("scrollArea_rp_results")
        self.scAreaContent_rp_results = QtWidgets.QWidget()
        self.scAreaContent_rp_results.setEnabled(False)
        self.scAreaContent_rp_results.setGeometry(QtCore.QRect(0, 0, 518, 482))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.scAreaContent_rp_results.sizePolicy().hasHeightForWidth())
        self.scAreaContent_rp_results.setSizePolicy(sizePolicy)
        self.scAreaContent_rp_results.setObjectName("scAreaContent_rp_results")
        self.scrollArea_rp_results.setWidget(self.scAreaContent_rp_results)
        self.verticalLayout_6.addWidget(self.scrollArea_rp_results)
        self.tabWidget.addTab(self.tab_rp_service, "")
        self.tab_orange_service = QtWidgets.QWidget()
        self.tab_orange_service.setObjectName("tab_orange_service")
        self.verticalLayout_7 = QtWidgets.QVBoxLayout(self.tab_orange_service)
        self.verticalLayout_7.setContentsMargins(11, 11, 11, 11)
        self.verticalLayout_7.setSpacing(6)
        self.verticalLayout_7.setObjectName("verticalLayout_7")
        self.verticalLayout_14 = QtWidgets.QVBoxLayout()
        self.verticalLayout_14.setSpacing(6)
        self.verticalLayout_14.setObjectName("verticalLayout_14")
        self.label_16 = QtWidgets.QLabel(self.tab_orange_service)
        self.label_16.setEnabled(True)
        self.label_16.setStyleSheet("bold")
        self.label_16.setObjectName("label_16")
        self.verticalLayout_14.addWidget(self.label_16)
        self.label_orange_status = QtWidgets.QLabel(self.tab_orange_service)
        self.label_orange_status.setObjectName("label_orange_status")
        self.verticalLayout_14.addWidget(self.label_orange_status)
        self.verticalLayout_7.addLayout(self.verticalLayout_14)
        self.verticalLayout_15 = QtWidgets.QVBoxLayout()
        self.verticalLayout_15.setSpacing(6)
        self.verticalLayout_15.setObjectName("verticalLayout_15")
        self.label_18 = QtWidgets.QLabel(self.tab_orange_service)
        self.label_18.setEnabled(True)
        self.label_18.setStyleSheet("bold")
        self.label_18.setObjectName("label_18")
        self.verticalLayout_15.addWidget(self.label_18)
        self.label_orange_criteria_results = QtWidgets.QLabel(self.tab_orange_service)

self.label_orange_criteria_results.setTextInteractionFlags(QtCore.Qt.LinksAccessibleByMouse|QtCore.Qt.TextS
electableByKeyboard|QtCore.Qt.TextSelectableByMouse)
        self.label_orange_criteria_results.setObjectName("label_orange_criteria_results")
        self.verticalLayout_15.addWidget(self.label_orange_criteria_results)
        self.verticalLayout_7.addLayout(self.verticalLayout_15)
        self.label_8 = QtWidgets.QLabel(self.tab_orange_service)
        self.label_8.setObjectName("label_8")
        self.verticalLayout_7.addWidget(self.label_8)
        self.scrollArea_orange_results = QtWidgets.QScrollArea(self.tab_orange_service)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.scrollArea_orange_results.sizePolicy().hasHeightForWidth())
        self.scrollArea_orange_results.setSizePolicy(sizePolicy)
```

```
        self.scrollArea_orange_results.viewport().setProperty("cursor", QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.scrollArea_orange_results.setMouseTracking(True)
        self.scrollArea_orange_results.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.scrollArea_orange_results.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.scrollArea_orange_results.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)
        self.scrollArea_orange_results.setWidgetResizable(True)
        self.scrollArea_orange_results.setObjectName("scrollArea_orange_results")
        self.scAreaContent_orange_results = QtWidgets.QWidget()
        self.scAreaContent_orange_results.setEnabled(False)
        self.scAreaContent_orange_results.setGeometry(QtCore.QRect(0, 0, 518, 482))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.scAreaContent_orange_results.sizePolicy().hasHeightForWidth())
        self.scAreaContent_orange_results.setSizePolicy(sizePolicy)
        self.scAreaContent_orange_results.setObjectName("scAreaContent_orange_results")
        self.scrollArea_orange_results.setWidget(self.scAreaContent_orange_results)
        self.verticalLayout_7.addWidget(self.scrollArea_orange_results)
        self.tabWidget.addTab(self.tab_orange_service, "")
        self.tab_comparison = QtWidgets.QWidget()
        self.tab_comparison.setObjectName("tab_comparison")
        self.verticalLayout_8 = QtWidgets.QVBoxLayout(self.tab_comparison)
        self.verticalLayout_8.setContentsMargins(11, 11, 11, 11)
        self.verticalLayout_8.setSpacing(6)
        self.verticalLayout_8.setObjectName("verticalLayout_8")
        self.verticalLayout_18 = QtWidgets.QVBoxLayout()
        self.verticalLayout_18.setSpacing(6)
        self.verticalLayout_18.setObjectName("verticalLayout_18")
        self.label_24 = QtWidgets.QLabel(self.tab_comparison)
        self.label_24.setEnabled(True)
        self.label_24.setStyleSheet("bold")
        self.label_24.setObjectName("label_24")
        self.verticalLayout_18.addWidget(self.label_24)
        self.label_comparison_status = QtWidgets.QLabel(self.tab_comparison)
        self.label_comparison_status.setObjectName("label_comparison_status")
        self.verticalLayout_18.addWidget(self.label_comparison_status)
        self.verticalLayout_8.addLayout(self.verticalLayout_18)
        self.verticalLayout_19 = QtWidgets.QVBoxLayout()
        self.verticalLayout_19.setSpacing(6)
        self.verticalLayout_19.setObjectName("verticalLayout_19")
        self.label_26 = QtWidgets.QLabel(self.tab_comparison)
        self.label_26.setEnabled(True)
        self.label_26.setStyleSheet("bold")
        self.label_26.setObjectName("label_26")
        self.verticalLayout_19.addWidget(self.label_26)
        self.label_comparison_results = QtWidgets.QLabel(self.tab_comparison)

self.label_comparison_results.setTextInteractionFlags(QtCore.Qt.LinksAccessibleByMouse|QtCore.Qt.TextSele
ctableByKeyboard|QtCore.Qt.TextSelectableByMouse)
        self.label_comparison_results.setObjectName("label_comparison_results")
        self.verticalLayout_19.addWidget(self.label_comparison_results)
        self.verticalLayout_8.addLayout(self.verticalLayout_19)
        self.label_notes = QtWidgets.QLabel(self.tab_comparison)
        self.label_notes.setStyleSheet("color:rgba(0, 170, 0, 0);")
        self.label_notes.setObjectName("label_notes")
        self.verticalLayout_8.addWidget(self.label_notes)
        self.label_9 = QtWidgets.QLabel(self.tab_comparison)
        self.label_9.setObjectName("label_9")
        self.verticalLayout_8.addWidget(self.label_9)
        self.scrollArea_comparison_results = QtWidgets.QScrollArea(self.tab_comparison)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.scrollArea_comparison_results.sizePolicy().hasHeightForWidth())
        self.scrollArea_comparison_results.setSizePolicy(sizePolicy)
```

```
        self.scrollArea_comparison_results.viewport().setProperty("cursor",
QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.scrollArea_comparison_results.setMouseTracking(True)
        self.scrollArea_comparison_results.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.scrollArea_comparison_results.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.scrollArea_comparison_results.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)
        self.scrollArea_comparison_results.setWidgetResizable(True)
        self.scrollArea_comparison_results.setObjectName("scrollArea_comparison_results")
        self.scAreaContent_comparison_results = QtWidgets.QWidget()
        self.scAreaContent_comparison_results.setEnabled(False)
        self.scAreaContent_comparison_results.setGeometry(QtCore.QRect(0, 0, 518, 459))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.scAreaContent_comparison_results.sizePolicy().hasHeightForWidth())
        self.scAreaContent_comparison_results.setSizePolicy(sizePolicy)
        self.scAreaContent_comparison_results.setObjectName("scAreaContent_comparison_results")
        self.scrollArea_comparison_results.setWidget(self.scAreaContent_comparison_results)
        self.verticalLayout_8.addWidget(self.scrollArea_comparison_results)
        self.tabWidget.addTab(self.tab_comparison, "")
        self.horizontalLayout_2.addWidget(self.tabWidget)
        self.verticalLayout_9 = QtWidgets.QVBoxLayout()
        self.verticalLayout_9.setSpacing(6)
        self.verticalLayout_9.setObjectName("verticalLayout_9")
        self.line = QtWidgets.QFrame(self.centralWidget)
        self.line.setFrameShape(QtWidgets.QFrame.HLine)
        self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line.setObjectName("line")
        self.verticalLayout_9.addWidget(self.line)
        self.label = QtWidgets.QLabel(self.centralWidget)
        self.label.setObjectName("label")
        self.verticalLayout_9.addWidget(self.label)
        self.scrollArea_data = QtWidgets.QScrollArea(self.centralWidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.scrollArea_data.sizePolicy().hasHeightForWidth())
        self.scrollArea_data.setSizePolicy(sizePolicy)
        self.scrollArea_data.viewport().setProperty("cursor", QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.scrollArea_data.setMouseTracking(True)
        self.scrollArea_data.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.scrollArea_data.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.scrollArea_data.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)
        self.scrollArea_data.setWidgetResizable(True)
        self.scrollArea_data.setObjectName("scrollArea_data")
        self.scarea_data = QtWidgets.QWidget()
        self.scarea_data.setEnabled(False)
        self.scarea_data.setGeometry(QtCore.QRect(0, 0, 537, 286))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.scarea_data.sizePolicy().hasHeightForWidth())
        self.scarea_data.setSizePolicy(sizePolicy)
        self.scarea_data.setObjectName("scarea_data")
        self.scrollArea_data.setWidget(self.scarea_data)
        self.verticalLayout_9.addWidget(self.scrollArea_data)
        self.label_10 = QtWidgets.QLabel(self.centralWidget)
        self.label_10.setObjectName("label_10")
        self.verticalLayout_9.addWidget(self.label_10)
        self.scrollArea_data_to_predict = QtWidgets.QScrollArea(self.centralWidget)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.scrollArea_data_to_predict.sizePolicy().hasHeightForWidth())
        self.scrollArea_data_to_predict.setSizePolicy(sizePolicy)
```

```
        self.scrollArea_data_to_predict.viewport().setProperty("cursor",
QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.scrollArea_data_to_predict.setMouseTracking(True)
        self.scrollArea_data_to_predict.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.scrollArea_data_to_predict.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
        self.scrollArea_data_to_predict.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContents)
        self.scrollArea_data_to_predict.setWidgetResizable(True)
        self.scrollArea_data_to_predict.setObjectName("scrollArea_data_to_predict")
        self.scarea_data_to_predict = QtWidgets.QWidget()
        self.scarea_data_to_predict.setEnabled(False)
        self.scarea_data_to_predict.setGeometry(QtCore.QRect(0, 0, 537, 285))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.scarea_data_to_predict.sizePolicy().hasHeightForWidth())
        self.scarea_data_to_predict.setSizePolicy(sizePolicy)
        self.scarea_data_to_predict.setObjectName("scarea_data_to_predict")
        self.scrollArea_data_to_predict.setWidget(self.scarea_data_to_predict)
        self.verticalLayout_9.addWidget(self.scrollArea_data_to_predict)
        self.horizontalLayout_2.addLayout(self.verticalLayout_9)
        self.verticalLayout_2.addLayout(self.horizontalLayout_2)
        MainWindow.setCentralWidget(self.centralWidget)
        self.statusBar = QtWidgets.QStatusBar(MainWindow)
        self.statusBar.setObjectName("statusBar")
        MainWindow.setStatusBar(self.statusBar)
        self.retranslateUi(MainWindow)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Service Analyzer"))
        self.label_program_name.setText(_translate("MainWindow", "Data Mining Service Analyzer"))
        self.label_2.setText(_translate("MainWindow", "Path to data:"))
        self.edit_data_path.setText(_translate("MainWindow",
"/home/dasha/Documents/DiplomaFolder/Wine/winequality-red_100_experimental.csv"))
        self.btn_browse_path.setText(_translate("MainWindow", "browse.."))
        self.label_3.setText(_translate("MainWindow", "Functionality to run:"))
        self.list_functionality.setItemText(0, _translate("MainWindow", "outliers"))
        self.list_functionality.setItemText(1, _translate("MainWindow", "linear regression"))
        self.list_functionality.setItemText(2, _translate("MainWindow", "prediction"))
        self.list_functionality.setItemText(3, _translate("MainWindow", "k-means clustering"))
        self.label_13.setText(_translate("MainWindow", "Preprocess data:"))
        self.cBx_normalize.setText(_translate("MainWindow", "normalize"))
        self.cBx_remove_outlier.setText(_translate("MainWindow", "remove outliers"))
        self.label_4.setText(_translate("MainWindow", "Data Mining services to use:"))
        self.cBx_rp_service_run.setText(_translate("MainWindow", "Rapid Miner "))
        self.cBx_orange_service_run.setText(_translate("MainWindow", "Orange3 library "))
        self.label_5.setText(_translate("MainWindow", "Criteria for service evaluation:"))
        self.cBx_calculate_auc_roc.setText(_translate("MainWindow", "AUC-ROC metric (for outliers
function only)"))
        self.cBx_find_execution_time.setText(_translate("MainWindow", "execution time"))
        self.cBx_find_time_dependency.setText(_translate("MainWindow", "execution time dependence
from amount of data"))
        self.label_6.setText(_translate("MainWindow", "Comparison services:"))
        self.cBx_compare_results.setText(_translate("MainWindow", "compare results"))
        self.btn_process_data.setText(_translate("MainWindow", "Process data"))
        self.label_status.setText(_translate("MainWindow", "Processing..."))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_config), _translate("MainWindow",
"Configuration"))
        self.label_20.setText(_translate("MainWindow", "Status:"))
        self.label_rp_status.setText(_translate("MainWindow", "unknown"))
        self.label_22.setText(_translate("MainWindow", "Results:"))
        self.label_rp_results.setText(_translate("MainWindow", "unknown"))
        self.label_7.setText(_translate("MainWindow", "Table with results:"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_rp_service), _translate("MainWindow",
"Rapid Miner service"))
```

```
        self.label_16.setText(_translate("MainWindow", "Status:"))
        self.label_orange_status.setText(_translate("MainWindow", "unknown"))
        self.label_18.setText(_translate("MainWindow", "Results:"))
        self.label_orange_criteria_results.setText(_translate("MainWindow", "unknown"))
        self.label_8.setText(_translate("MainWindow", "Table with results:"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_orange_service), _translate("MainWindow",
"Orange3 service"))
        self.label_24.setText(_translate("MainWindow", "Status:"))
        self.label_comparison_status.setText(_translate("MainWindow", "unknown"))
        self.label_26.setText(_translate("MainWindow", "Results:"))
        self.label_comparison_results.setText(_translate("MainWindow", "unknown"))
        self.label_notes.setText(_translate("MainWindow", "TextLabel"))
        self.label_9.setText(_translate("MainWindow", "Table with results:"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_comparison), _translate("MainWindow",
"Service comparison"))
        self.label.setText(_translate("MainWindow", "Data to process:"))
        self.label_10.setText(_translate("MainWindow", "Data for prediction:"))
```

2.1.2 Текст програми у файлі user_interface.py

```
import pyqtgraph as pg
from math import fabs
from PyQt5.QtWidgets import QMainWindow, QFileDialog, QTableWidget, QTableWidgetItem, QMessageBox,
\
  QSizePolicy, QTextEdit, QInputDialog
import csv
import time
import numpy as np
import mainwindow
import config
import traceback
from service_analyzer import ServiceAnalyzer, Functionality, ComparisonCriteria, EvaluationCriteria
DEFAULT_SOURCE_PATH = config.get_config()['Service']['default_sources_path']
INVISIBLE_STYLE = "QLabel {color : rgba(0, 170, 0, 0); }"
GREEN_STYLE = "QLabel {color : rgb(0, 170, 0); }"
BLACK_STYLE = "QLabel {color : rgb(0, 0, 0); }"
class ServiceStatus:
  unknown = 'unknown'
  in_progress = 'in progress'
  completed = 'completed'
class ServiceAnalyzerApp(QMainWindow, mainwindow.Ui_MainWindow):
  def __init__(self):
    super().__init__()
    self.graph_windows = {'Orange3': GraphWindow(), 'Rapid Miner': GraphWindow()}
    self.barchar_window = {'Orange3': BarChart(), 'Rapid Miner': BarChart()}
    self.setupUi(self)
    self.service_analyzer = ServiceAnalyzer()
    self.btn_browse_path.clicked.connect(self.onBrowseDataPathClicked)
    self.btn_process_data.clicked.connect(self.onProcessBtnClicked)
    self.table_data = QTableWidget()
    self.prediction_table = QTableWidget()
    self.rp_result_table = QTableWidget()
    self.orange_result_table = QTableWidget()
    self.service_compare_table = QTableWidget()
    self.scrollArea_data.setWidget(self.table_data)
    self.scrollArea_data_to_predict.setWidget(self.prediction_table)
    self.scrollArea_orange_results.setWidget(self.orange_result_table)
    self.scrollArea_rp_results.setWidget(self.rp_result_table)
    self.k_means_results = QTextEdit()
    self.list_functionality.currentIndexChanged.connect(self.onFunctionalityChanged)
  def onFunctionalityChanged(self, index):
    enable_outliers = False
    index = index + 1
    if index == Functionality.outliers.value:
      enable_outliers = True
```

```
    self.cBx_calculate_auc_roc.setEnabled(enable_outliers)
def swap_comparison_results_widget(self, show_table=True):
    if show_table:
        self.scrollArea_comparison_results.setWidget(self.service_compare_table)
        self.k_means_results = QTextEdit()
    else:
        self.scrollArea_comparison_results.setWidget(self.k_means_results)
        self.service_compare_table = QTableWidget()
def onBrowseDataPathClicked(self):
    fname = QFileDialog.getOpenFileName(self, 'Open file', filter='*.csv')[0]
    if fname:
        self.edit_data_path.setText(fname)
    self.load_cvs_file_to_table(self.edit_data_path.text(), self.table_data)
def load_cvs_file_to_table(self, fname, table):
    self.clean_table(table)
    with open(fname, newline='') as csvfile:
        reader = csv.reader(csvfile, delimiter=',', quotechar='|')
        line = 0
        for row in reader:
            if not row:
                continue
            j = 0
            if line != 0:
                table.insertRow(table.rowCount())
            table.setColumnCount(len(row))
            for value in row:
                item = QTableWidgetItem(value)
                if line == 0:
                    table.setHorizontalHeaderItem(j, item)
                else:
                    table.setItem(table.rowCount() - 1, j, item)
                j += 1
            line += 1
def clean_table(self, table):
    table.clear()
    while table.rowCount() > 0:
        table.removeRow(0)
def show_error_msg(self, details):
    msgBox = QMessageBox()
    msgBox.setWindowTitle("Oops..")
    msgBox.setText("Cannot process data")
    msgBox.setDetailedText(details)
    msgBox.setStandardButtons(QMessageBox.Ok)
    msgBox.exec_()
def show_service_result_table(self, config, results, service_result_table):
    self.clean_table(service_result_table)
    if not results:
        return
    if config['functionality'] == Functionality.linear_regression:
        results = results[1]
    for r in results:
        service_result_table.insertRow(service_result_table.rowCount())
        if not isinstance(r, list):
            service_result_table.setColumnCount(1)
            item = QTableWidgetItem(str(r))
            service_result_table.setItem(service_result_table.rowCount() - 1, 0, item)
            continue
        service_result_table.setColumnCount(len(r))
        j = 0
        for sub_item in r:
            item = QTableWidgetItem(str(sub_item))
            service_result_table.setItem(service_result_table.rowCount() - 1, j, item)
            j += 1
def _show_chart(self, data, service_name):
    self.graph_windows[service_name].setWindowTitle(
        "{} execution time dependence from amount of data".format(service_name))
```

```
    self.graph_windows[service_name].setGeometry(self.geometry())
    self.graph_windows[service_name].update_plot(data)
    self.graph_windows[service_name].show()
  def show_service_results(self, _config, service_results, label, service_name):
    results = ''
    for c in service_results:
      if c == 'execution_time':
        results = results + "{}: {}\n".format("execution time", service_results['execution_time'])
      elif c == 'time_dependence':
        self._show_chart(service_results['time_dependence'], service_name)
      elif c == 'auc_roc_score':
        results = results + "{}: {}\n".format("AUC-ROC score", service_results['auc_roc_score'])
    if _config['functionality'] == Functionality.linear_regression:
      for criteria in service_results['result'][0]:
        results = results + "{}: {}\n".format(criteria, service_results['result'][0][criteria])
    if not results:
      results = 'empty'
    label.setText(results)
  def _show_outliers_comparison_results(self, results):
    notes = '* Rapid Miner results were normalized to [0;1] range\n' \
        'Services results matches in {} %'.format(results['percent_of_matches'])
    self.label_notes.setStyleSheet(BLACK_STYLE)
    self.label_notes.setText(notes)
    self.swap_comparison_results_widget()
    self.clean_table(self.service_compare_table)
    self._show_matches_table(results)
  def _show_linear_regression_coefs(self, results):
    self.swap_comparison_results_widget()
    self.clean_table(self.service_compare_table)
    columns = 4
    self.service_compare_table.setColumnCount(columns)
    headers = ['titles', 'Orange3', 'Rapid Miner', 'Diff (Orange result - Rapid Miner result)']
    self.service_compare_table.setHorizontalHeaderLabels(headers)
    self.service_compare_table.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
    self.service_compare_table.horizontalHeader().setStretchLastSection(True)
    total_tolerance = 0
    for i in range(len(results['orange'])):
      self.service_compare_table.insertRow(self.service_compare_table.rowCount())
      total_tolerance += fabs(fabs(results['orange'][i]) - fabs(results['rapidminer'][i]))
      row = [results['titles'][i], results['orange'][i], results['rapidminer'][i], results['diffs'][i]]
      self.service_compare_table.setColumnCount(columns)
      for j in range(columns):
        item = QTableWidgetItem(str(row[j]))
        self.service_compare_table.setItem(self.service_compare_table.rowCount() - 1, j, item)
    self.label_comparison_results.setText('Total tolerance %s ' % total_tolerance)
  def _show_matches_table(self, results):
    columns = 3
    self.service_compare_table.setColumnCount(columns)
    headers = ['Orange3', 'Rapid Miner', 'Matches']
    self.service_compare_table.setHorizontalHeaderLabels(headers)
    self.service_compare_table.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
    self.service_compare_table.horizontalHeader().setStretchLastSection(True)
    for i in range(len(results['orange'])):
      self.service_compare_table.insertRow(self.service_compare_table.rowCount())
      row = [results['orange'][i], results['rapidminer'][i], results['matches'][i]]
      self.service_compare_table.setColumnCount(columns)
      for j in range(columns):
        item = QTableWidgetItem(str(row[j]))
        self.service_compare_table.setItem(self.service_compare_table.rowCount() - 1, j, item)
  def _show_prediction_comparison_results(self, results):
    notes = '* Orange results were rounded up\n' \
        'Services results matches in {} %'.format(results['percent_of_matches'])
    self.label_notes.setStyleSheet(BLACK_STYLE)
    self.label_notes.setText(notes)
    self.swap_comparison_results_widget()
    self.clean_table(self.service_compare_table)
```

```
        self._show_matches_table(results)
    def _prepare_clusters_results(self, results):
        headers_objects = {
            'orange': ('Orange3', 'Rapid Miner'),
            'rapidminer': ('Rapid Miner', 'Orange3')
        }
        result_string = ''
        for service in results:
            result_string += '{} matches: \n'.format(headers_objects[service][0])
            for cluster in results[service]:
                print(cluster)
                result_string += '  {} cluster to {} clusters: \n'.format(cluster, headers_objects[service][1])
                for match_cluster in results[service][cluster]:
                    result_string += '      {} - {}% \n'.format(match_cluster,
results[service][cluster][match_cluster])
        return result_string
    def _show_k_means_clustering_results(self, results):
        text = self._prepare_clusters_results(results['clusters_intersection'])
        self.swap_comparison_results_widget(show_table=False)
        self.k_means_results.setText(text)
        evaluation_results_text = 'Criteria       Rapid Miner       Orange3\n'
        space = '        '
        for c in results['evaluation_criteria']:
            evaluation_results_text = evaluation_results_text + c + space + \
                                      str(results['evaluation_criteria'][c]['rapid_miner']) + space + \
                                      str(results['evaluation_criteria'][c]['orange']) + '\n'
        self.label_notes.setStyleSheet(BLACK_STYLE)
        self.label_notes.setText(evaluation_results_text)
        self.barchar_window['Orange3'].update_bar(results['distribution']['orange'])
        self.barchar_window['Rapid Miner'].update_bar(results['distribution']['rapidminer'])
        for service_name in ('Orange3', 'Rapid Miner'):
            self.barchar_window[service_name].setWindowTitle(
                "{} clusters distribution".format(service_name))
            self.barchar_window[service_name].setGeometry(self.geometry())
            self.barchar_window[service_name].show()
    def show_service_comparison_results(self, functionality, results):
        self.label_comparison_status.setText('success')
        self.label_comparison_results.setText('')
        if functionality == Functionality.outliers:
            self._show_outliers_comparison_results(results['service_comparison'])
        elif functionality == Functionality.linear_regression:
            self._show_linear_regression_coefs(results['service_comparison'])
        elif functionality == Functionality.prediction:
            self._show_prediction_comparison_results(results['service_comparison'])
        elif functionality == Functionality.k_means_clustering:
            self._show_k_means_clustering_results(results['service_comparison'])
        else:
            print('Unknown functionality in comparison results ')
    def _gather_info_for_prediction(self):
        filename = QFileDialog.getOpenFileName(self, 'Path to data for prediction', filter='*.csv',
                                               )[0]
        if not filename:
            self.show_error_msg("You should specify file path!")
            raise RuntimeError()
        target, _ = QInputDialog.getText(self, 'Target column is needed for prediction',
                                         'Enter target column name:')
        if not target:
            self.show_error_msg("Prediction function needs target column!")
            raise RuntimeError()
        return filename, target
    def _gather_info_for_linear_regression(self):
        target_name, _ = QInputDialog.getText(self, 'Target column is needed for linear regression',
                                              'Enter target column name:')
        if not target_name:
            self.show_error_msg("Linear regression analysis needs target column!")
            raise RuntimeError()
```

```
        return target_name.rstrip()
    def _gather_info_for_outliers_metric(self):
        filename = QFileDialog.getOpenFileName(self, 'Path to data with marked outliers', filter='*.csv',
                                               )[0]
        if not filename:
            self.show_error_msg("You should specify file path!")
            raise RuntimeError()
        time.sleep(1)
        return filename
    def process(self):
        configuration = {'path': self.edit_data_path.text(),
                         'functionality': 0,
                         'rapidminer': self.cBx_rp_service_run.isChecked(),
                         'orange': self.cBx_orange_service_run.isChecked(),
                         'comparison': [],
                         'evaluation_criteria': [],
                         'target': '',
                         'data_to_predict': '',
                         'num_experiments': '',
                         'clusters': '',
                         'normalization': False,
                         'remove_outliers': False,
                         'file_with_outliers': ''  # for auc-roc metric
                         }
        try:
            self.clean_table(self.prediction_table)
            index = self.list_functionality.currentIndex() + 1
            if index == Functionality.linear_regression.value:
                target = self._gather_info_for_linear_regression()
                configuration['target'] = target
            elif index == Functionality.prediction.value:
                fname, target_value = self._gather_info_for_prediction()
                self.load_cvs_file_to_table(fname, self.prediction_table)
                configuration['data_to_predict'] = fname
                configuration['target'] = target_value
            elif index == Functionality.k_means_clustering.value:
                # get k for experiment
                num, ok = QInputDialog.getInt(self, "Number of cluster", "enter k: ")
                if ok or num:
                    configuration['clusters'] = num
                else:
                    self.show_error_msg('You should specify the number of cluster for k-means clustering!')
                    return
            elif index == Functionality.outliers.value and self.cBx_calculate_auc_roc.isChecked():
                fname = self._gather_info_for_outliers_metric()
                configuration['file_with_outliers'] = fname
            configuration['functionality'] = Functionality(index)
            if self.cBx_compare_results.isChecked():
                configuration['comparison'].append(ComparisonCriteria.results)
            if self.cBx_find_execution_time.isChecked():
                configuration['evaluation_criteria'].append(EvaluationCriteria.execution_time)
            if self.cBx_find_time_dependency.isChecked():
                configuration['evaluation_criteria'].append(EvaluationCriteria.time_dependence_from_data)
                num, ok = QInputDialog.getInt(self, "Building time dependency graphic", "set number of
experiments: ")
                if ok or num:
                    configuration['num_experiments'] = num
                else:
                    self.show_error_msg('Number of experiments must be set!')
                    return
            if configuration['orange']:
                self.label_orange_status.setText(ServiceStatus.in_progress)
            if configuration['rapidminer']:
                self.label_rp_status.setText(ServiceStatus.in_progress)
            if self.cBx_normalize.isChecked():
                configuration['normalization'] = True
```

```
        if self.cBx_remove_outlier.isChecked():
          if index not in (Functionality.linear_regression.value, Functionality.prediction.value):
            self.show_error_msg('Removing of outliers is allowed only for linear regression function'
                                ' or prediction!')
            return
          configuration['remove_outliers'] = True
        results = self.service_analyzer.process(configuration)
        if configuration['orange']:
          self.label_orange_status.setText(ServiceStatus.completed)
          self.show_service_results(configuration, results.get('orange'), self.label_orange_criteria_results,
                                    'Orange3')
          self.show_service_result_table(configuration, results.get('orange')['result'],
self.orange_result_table)
        if configuration['rapidminer']:
          self.label_rp_status.setText(ServiceStatus.completed)
          self.show_service_results(configuration, results.get('rapidminer'), self.label_rp_results,
                                    'Rapid Miner')
          self.show_service_result_table(configuration, results.get('rapidminer')['result'],
self.rp_result_table)
        if configuration['comparison']:
          self.show_service_comparison_results(configuration['functionality'], results)
      except Exception as ex:
        print(ex)
        traceback.print_exc()
        self.show_error_msg(str(ex))
      finally:
        self.setEnabled(True)
        self.label_status.setText("Finished")
    def onProcessBtnClicked(self):
      # gather config
      # run process
      # show results
      self.label_status.setText("Processing")
      self.label_status.setStyleSheet(GREEN_STYLE)
      self.label_notes.setStyleSheet(INVISIBLE_STYLE)
      self.label_comparison_status.setText('unknown')
      self.label_rp_results.setText('')
      self.clean_table(self.service_compare_table)
      self.setEnabled(False)
      self.repaint()
      self.process()
class GraphWindow(QMainWindow):
    def __init__(self, *args, **kwargs):
      super(GraphWindow, self).__init__(*args, **kwargs)
      self.graphWidget = pg.PlotWidget()
      self.setCentralWidget(self.graphWidget)
      self.graphWidget.showGrid(x=True, y=True)
      self.graphWidget.plotItem.getAxis('left').setLabel('time (ms)')
      self.graphWidget.plotItem.getAxis('bottom').setLabel('number of processed rows')
    def update_plot(self, data):
      # plot data: x - num rows y - time values
      self.graphWidget = pg.PlotWidget()
      self.setCentralWidget(self.graphWidget)
      self.graphWidget.showGrid(x=True, y=True)
      self.graphWidget.plotItem.getAxis('left').setLabel('time (ms)')
      self.graphWidget.plotItem.getAxis('bottom').setLabel('number of processed rows')
      time = []
      rows = []
      print('rows')
      for r, _ in data:
        print(int(r))
      print('time')
      for _, t in data:
        print(int(t))
      for r, t in data:
        time.append(t)
```

```
        rows.append(r)
      self.graphWidget.plot(rows, time, clear=None, meta='time dependence from amount of data')
class BarChart(QMainWindow):
  def __init__(self, *args, **kwargs):
    super(BarChart, self).__init__(*args, **kwargs)
    self.graphWidget = pg.plot()
    self.setCentralWidget(self.graphWidget)
    self.graphWidget.showGrid(x=True, y=True)
    self.graphWidget.setBackground((255, 255, 255))
  def update_bar(self, data):
    x = np.arange(len(data))
    bg = pg.BarGraphItem(x=x, height=data, width=0.6, brush=(133, 175, 198))
    self.graphWidget.addItem(bg)
```

2.2 Модуль Service Analyzer

2.2.1 Текст програми у файлі service_analyzer.py

```
import os
import glob
from enum import Enum
import config
from rapid_miner_service import RapidMinerService
from orange3_service import Orange3Service
from utils import read_cvs_file, generate_test_cvs_file, convert_string_list_to_int
from service_comparator import ServiceComparator
class Functionality(Enum):
  outliers = 1
  linear_regression = 2
  prediction = 3
  k_means_clustering = 4
class ComparisonCriteria(Enum):
  results = 1
class EvaluationCriteria(Enum):
  execution_time = 1
  time_dependence_from_data = 2
def simple_wrapper(func, **kwargs):
  def wrapper():
    return func(**kwargs)
  return wrapper
class ServiceAnalyzer:
  def __init__(self):
    self.service_settings = config.get_config()
    self.config = {}
  '''
  :output [(rows, time), ]
  '''
  def _find_time_dependence(self, service, func, **kwargs):
    data_path = service.get_data_path()
    num_experiments = self.config['num_experiments']
    headers, content, num_of_initial_rows = read_cvs_file(data_path)
    generated_path_template = '%s.{}.csv' %
str(self.service_settings['Service']['dir_for_generated_files'] +
                                os.path.sep +
                                os.path.splitext(os.path.basename(data_path))[0])
    rows_increment = num_of_initial_rows - 1
    result = []
    for i in range(num_experiments):
      rows = i * rows_increment + num_of_initial_rows - 1
      file_path = generated_path_template.format(rows)
      if not os.path.exists(file_path):
        generate_test_cvs_file(file_path, content, headers, rows)
      service.set_data_path(file_path)
      _, time = service.measure_time_execution(func, **kwargs)
      result.append((rows, time))
```

```
    return result
  def _run_service_functionality(self, service, service_func, **kwargs):
    results = {}
    for c in self.config['evaluation_criteria']:
      if c == EvaluationCriteria.execution_time:
        results["result"], results["execution_time"] = service.measure_time_execution(service_func,
**kwargs)
      if c == EvaluationCriteria.time_dependence_from_data:
        results["time_dependence"] = self._find_time_dependence(service, service_func, **kwargs)
        service.set_data_path(self.config['path'])
    if not results.get('result'):
      results["result"] = service_func(**kwargs)
    return results
  def _make_preprocess(self, service):
    data_table = None
    if self.config['remove_outliers']:
      data_table = service.remove_outliers()
    if self.config['normalization']:
      data_table = service.normalize(data_table)
    if data_table is None:
      return
    generated_path_template = '%s_pre_processed_{}.csv' % str(
      self.service_settings['Service']['dir_for_generated_files'] +
      os.path.sep +
      os.path.splitext(os.path.basename(self.config['path']))[0])
    generated_path = generated_path_template.format(__class__.__name__)
    service.save_data_to_file(data_table, generated_path)
    service.set_data_path(generated_path)
  def _get_service_results(self, service):
    self._make_preprocess(service)
    functionality = self.config['functionality']
    if functionality == Functionality.outliers:
      results = self._run_service_functionality(service, service.get_outliers)
      if self.config['file_with_outliers']:
        _, outliers_score, _ = read_cvs_file(self.config['file_with_outliers'])
        outliers_score = convert_string_list_to_int(outliers_score)
        results['auc_roc_score'] = service.get_roc_auc_score(outliers_score, results['result'])
    elif functionality == Functionality.linear_regression:
      results = self._run_service_functionality(service, service.get_linear_regression_weights,
                                target_name=self.config['target'])
    elif functionality == Functionality.prediction:
      results = self._run_service_functionality(service,
                                service.get_prediction,
                                data_to_predict_path=self.config['data_to_predict'],
                                target_name=self.config['target'])
    elif functionality == Functionality.k_means_clustering:
      results = self._run_service_functionality(service, service.get_clusters,
                                num_clusters=self.config['clusters'])
    else:
      raise RuntimeError('Unknown functionality')
    return results
  def _get_service_comparison_results(self, results):
    service_comparator = ServiceComparator(self.config['path'])
    functionality = self.config['functionality']
    compare_results = ''
    if functionality == Functionality.outliers:
      compare_results = service_comparator.compare_outliers(r_outliers=results['rapidminer']['result'],
                                      o_outliers=results['orange']['result'])
    elif functionality == Functionality.linear_regression:
      compare_results = service_comparator.compare_linear_regression_coefs(
        r_coefs=results['rapidminer']['result'][1],
        o_coefs=results['orange']['result'][1])
    elif functionality == Functionality.prediction:
      compare_results =
service_comparator.compare_predictions(r_predicts=results['rapidminer']['result'],
                                       o_predicts=results['orange']['result'])
```

```
        elif functionality == Functionality.k_means_clustering:
            compare_results = service_comparator.compare_clusters(r_clusters=results['rapidminer']['result'],
                                                o_clusters=results['orange']['result'])
        else:
            raise RuntimeError('Unknown functionality')
        return compare_results
    def _compare_services(self, results):
        if not self.config['comparison']:
            return results
        for comparison_criteria in self.config['comparison']:
            if comparison_criteria == ComparisonCriteria.results:
                results['service_comparison'] = self._get_service_comparison_results(results)
        return results
    def process(self, configuration):
        files = glob.glob('{}/*'.format(self.service_settings['Service']['dir_for_generated_files']))
        for f in files:
            os.remove(f)
        # configuration = {'path': self.edit_data_path.text(),
        #                  'functionality': 0,
        #                  'rapidminer': self.cBx_rp_service_run.isChecked(),
        #                  'orange': self.cBx_orange_service_run.isChecked(),
        #                  'comparison': [],
        #                  'evaluation_criteria': [],
        #                  'target': ''
        #                  'data_to_predict': '',
        #                  'num_experiments': '',
        #                  'clusters': '',
        #                  'normalization': False,
        #                  'remove_outliers': False,
        #                  'file_with_outliers':''
        #                  }
        self.config = configuration
        results = dict({})
        if configuration['orange']:
            orange = Orange3Service(configuration['path'])
            results['orange'] = self._get_service_results(orange)
        if configuration['rapidminer']:
            rp_miner = RapidMinerService(configuration['path'])
            results['rapidminer'] = self._get_service_results(rp_miner)
        results = self._compare_services(results)
        return results
```

2.3 Модуль Orange Service

2.3.1 Текст програми у файлі data_mining_service.py

```
from abc import ABC, abstractmethod
from sklearn.metrics import roc_auc_score
'''
1. Outliers
2. Linear regression weights
3. Prediction
4. K-means clustering
'''
class DataMiningService(ABC):
    '''
     :return [1,0 ..]
    '''
    @abstractmethod
    def get_outliers(self):
        pass
    '''
     :return [5,6 ..]
    '''
    @abstractmethod
```

```
  def get_prediction(self, data_to_predict_path, target_name):
    pass
  '''
  :return [[column1,w1],[column2,w2]]
  '''
  @abstractmethod
  def get_linear_regression_weights(self, target_name):
    pass
  '''
   :return [cluster_1, ..]
  '''
  @abstractmethod
  def get_clusters(self, num_clusters):
    pass
  @abstractmethod
  def measure_time_execution(self, func, **kwargs):
    pass
  def set_data_path(self, data_path):
    pass
  '''
  actual_results= [1,0..]
  scores = [0,1..] | [0.1, 0.7 ...]
  '''
  @staticmethod
  def get_roc_auc_score(actual_results, scores):
    n_classes = set(actual_results)
    return round(roc_auc_score(actual_results, scores), 3)
```

2.3.2 Текст програми у файлі orange_service.py

```
import datetime
from Orange import clustering, data, classification
from Orange.data import Domain, ContinuousVariable
from Orange.regression.linear import LinearRegressionLearner
from Orange.evaluation import R2, MAE, MSE, RMSE, CrossValidation
from Orange.preprocess import Normalize
from data_mining_service import DataMiningService
from sklearn.preprocessing import StandardScaler
class Orange3Service(DataMiningService):
  def __init__(self, data_path):
    self._data_path = data_path
  def get_outliers(self):
    data_table = data.Table(self._data_path)
    lof = classification.LocalOutlierFactorLearner(metric="chebyshev")
    res = lof(data_table)(data_table)
    return list(res.get_column_view('Outlier')[0])
  def normalize(self, data_table=None):
    if not data_table:
      data_table = data.Table(self._data_path)
    normalizer = Normalize(zero_based=True, norm_type=Normalize.NormalizeBySD,
transform_class=False,
                  center=True, normalize_datetime=False)
    normalized_data = normalizer(data_table)
    return normalized_data
  def remove_outliers(self, data_table=None):
    outliers = self.get_outliers()
    if not data_table:
      data_table = data.Table(self._data_path)
    new_indices = []
    for i in range(len(outliers)):
      if outliers[i] == 1:  # not outlier
        new_indices.append(i)
      else:
        print(i)
    print('finish')
```

```
        new_table = data_table.from_table(data_table.domain, data_table, new_indices)
        return new_table
    def _specify_target_variable(self, table, target_name):
        new_attributes = []
        class_vars = []
        for a in table.domain.attributes:
            if a.name != target_name:
                new_attributes.append(a)
            else:
                class_vars.append(ContinuousVariable(a.name))
        for c in table.domain.class_vars:
            if c.name != target_name:
                new_attributes.append(c)
            else:
                class_vars.append(ContinuousVariable(a.name))
        domain = Domain(new_attributes,
                        class_vars,
                        table.domain.metas)
        return table.transform(domain)
    def get_prediction(self, data_to_predict_path, target_name):
        data_table = data.Table(self._data_path)
        data_table = self._specify_target_variable(data_table, target_name)
        print(data_table.domain)
        mean_ = LinearRegressionLearner()
        model = mean_(data_table)
        data_to_predict_table = data.Table(data_to_predict_path)
        data_to_predict_table = self._specify_target_variable(data_to_predict_table, target_name)
        res = model.predict_storage(data_to_predict_table)
        return res.tolist()
    def get_linear_regression_weights(self, target_name):
        data_table = data.Table(self._data_path)
        data_table = self._specify_target_variable(data_table, target_name)
        print(data_table.domain)
        mean_ = LinearRegressionLearner()
        model = mean_(data_table)
        columns = [d.name for d in data_table.domain.variables]
        columns.remove(target_name)
        result = []
        for i in range(len(columns)):
            result.append([columns[i], model.coefficients[i]])
        res = CrossValidation(data_table, [mean_, ])
        criterias = {'root_mean_squared_error': RMSE(res), 'absolute_error': MAE(res), 'squared_error':
MSE(res),
                     'squared_correlation': R2(res)}
        for c in criterias:
            criterias[c] = round(criterias[c][0], 3)
        return criterias, result
        # columns = [d.name for d in data_table.domain.variables]
        # columns.remove(target_name)
        # new_table = data.Table.from_table(domain=d, source=data_table)
        # mean_ = LinearRegressionLearner()
        # model = mean_(new_table)
    def get_clusters(self, num_clusters):
        data_table = data.Table(self._data_path)
        km = clustering.KMeans(n_clusters=num_clusters)
        return km(data_table).tolist()
    def measure_time_execution(self, func, **kwargs):
        ms_koef = 1000
        print('Orange service.Running function ', func.__name__)
        ts = datetime.datetime.now().timestamp() * ms_koef
        res = func(**kwargs)
        te = datetime.datetime.now().timestamp() * ms_koef
        diff = te - ts
        print('Orange service.{} execution time: {}'.format(func.__name__, diff))
        return res, diff
    def set_data_path(self, data_path):
```

```
    self._data_path = data_path
  @classmethod
  def save_data_to_file(cls, data_table, file_path):
    assert isinstance(data_table, data.Table)
    data_table.save(file_path)
  def get_data_path(self):
    return self._data_path
```

2.4 Модуль Rapid Miner Service

2.4.1 Текст програми у файлі rapid_miner_service.py

```
import os
import os.path
import rapidminer
import pandas
from pandas import DataFrame
import config
from data_mining_service import DataMiningService
from sklearn.preprocessing import StandardScaler
PROCESSES = {
  'outliers': '//Local Repository/detect_outliers',
  'prediction': '//Local Repository/prediction',
  'linear_regression': '//Local Repository/linear_regression',
  'k_means_clusters': '//Local Repository/k_means_clustering',
  'normalization': '//Local Repository/normalize'
}
TEST_DIR_PATH = '//Local Repository/test_data/'
PROGRAM_PATH = os.path.join(os.path.expanduser("~"), 'ServiceAnalyzer')
TIME_EXECUTION_PATH = os.path.join(PROGRAM_PATH, 'execution_time.log')
# We can read logs from file
# All function returns Pandas DataFrame
class RapidMinerService(DataMiningService):
  def __init__(self, data_path):
    rm_home = config.get_config()['RapidMiner']['home']
    self._connector = rapidminer.Studio(rm_home)
    # load dataframe from csv
    self._data_path = data_path
  def remove_outliers(self, data_frame=None):
    outliers = self.get_outliers()
    if data_frame is None:
      data_frame = pandas.read_csv(self._data_path)
    new_indices = []
    for i in range(len(outliers)):
      if outliers[i] >= 1:  # not outlier
        new_indices.append(i)
      else:
        print(i)
    data_frame = data_frame.filter(items=new_indices, axis=0)
    return data_frame
  def normalize(self, data_frame=None):
    if data_frame is None:
      data_frame = pandas.read_csv(self._data_path)
    # z- centered
    normalized_data = self._connector.run_process(PROCESSES['normalization'], inputs=[data_frame])
    return normalized_data
  def get_outliers(self):
    # df = self._connector.read_resource(self._data_path)
    df = pandas.read_csv(self._data_path)
    outliers = self._connector.run_process(PROCESSES['outliers'], inputs=[df])
    return outliers['outlier'].to_numpy().tolist()
  def get_prediction(self, data_to_predict_path, target_name):
    df_to_predict = pandas.read_csv(data_to_predict_path)
    df = pandas.read_csv(self._data_path)
    prediction = self._connector.run_process(PROCESSES['prediction'], inputs=[df, df_to_predict],
```

```
                                macros={'target': target_name})
        return prediction['prediction(%s)' % target_name].values.tolist()
    def get_linear_regression_weights(self, target_name):
        df = pandas.read_csv(self._data_path)
        data = self._connector.run_process(PROCESSES['linear_regression'], inputs=[df],
                                macros={'target': target_name})
        criterias = {}
        for i in range(len(data[1])):
            criterias[data[1]['Criterion'][i]] = round(data[1]['Value'][i], 3)
        return criterias, data[0].values.tolist()
    def get_clusters(self, num_clusters):
        df = pandas.read_csv(self._data_path)
        clusters = self._connector.run_process(PROCESSES['k_means_clusters'], inputs=[df],
                                macros={'num_clusters': num_clusters})
        return clusters['cluster'].values.tolist()
    def _get_logged_time(self):
        if not os.path.exists(TIME_EXECUTION_PATH):
            raise RuntimeError('Cannot get execution time, because %s is absent', TIME_EXECUTION_PATH)
        with open(TIME_EXECUTION_PATH) as f:
            for l in f.readlines():
                print(l)
                if l.startswith('#'):
                    continue
                nums = [float(n.strip()) if 'null' not in n else 0 for n in l.split('\t')]
                return sum(nums)
    def measure_time_execution(self, func, **kwargs):
        print('Rapid miner service. Running function ', func.__name__)
        import time
        ms_koef = 1000
        ts = time.time()
        res = func(**kwargs)
        te = time.time()
        diff1 = (te - ts) * ms_koef
        diff = self._get_logged_time()
        print('Rapid miner service. {} execution time: {}, manual time {}'.format(func.__name__, diff,
diff1))
        return res, diff
    def run_empty_process(self):
        self._connector.run_process('//Local Repository/empty_process')
    def set_data_path(self, data_path):
        self._data_path = data_path
    @classmethod
    def save_data_to_file(cls, data_table, file_path):
        assert isinstance(data_table, DataFrame)
        csv_format = data_table.to_csv(index=False)
        with open(file_path, 'w') as f:
            f.write(csv_format)
    def get_data_path(self):
        return self._data_path
```

2.5 Модуль Service Comparator

2.5.1 Текст програми у файлі service_comparator.py

```
import copy
import math
import pandas as pd
from sklearn import metrics
from sklearn.metrics import pairwise_distances
from sklearn import datasets
import numpy as np
'''
1. Outliers
2. Linear regression weights
3. Prediction
```

4. K-means clustering

```
'''
class ServiceComparator:
  def __init__(self, source_data_path):
    data = pd.read_csv(source_data_path)
    self.sources = data.values.tolist()
  def compare_outliers(self, r_outliers, o_outliers):
    assert len(r_outliers) == len(o_outliers)
    size = len(r_outliers)
    results = {'orange': o_outliers,
          'rapidminer': copy.copy(r_outliers)}
    # Normalize Rapid Miner results [0,1]
    for i in range(len(r_outliers)):
      r = results['rapidminer'][i]
      if r > 1:
        results['rapidminer'][i] = 1
      else:
        results['rapidminer'][i] = 0
    matches = []
    sum_matches = 0
    # find matches
    for i in range(size):
      if results['rapidminer'][i] == results['orange'][i]:
        matches.append(1)
        sum_matches += 1
      else:
        matches.append(0)
    results['matches'] = matches
    results['percent_of_matches'] = round(sum_matches * 100 / len(r_outliers), 3)
    return results
  def compare_linear_regression_coefs(self, r_coefs, o_coefs):
    assert len(r_coefs) == len(o_coefs)
    size = len(r_coefs)
    results = {'orange': [],
          'rapidminer': [],
          'titles': [],
          'diffs': []}
    # find matches
    for i in range(size):
      results['titles'].append(o_coefs[i][0])
      results['diffs'].append(o_coefs[i][1] - r_coefs[i][1])
      results['orange'].append(o_coefs[i][1])
      results['rapidminer'].append(r_coefs[i][1])
    return results
  def compare_predictions(self, r_predicts, o_predicts):
    assert len(r_predicts) == len(o_predicts)
    size = len(r_predicts)
    results = {'orange': [],
          'rapidminer': [],
          'matches': [],
          'percent_of_matches': 0}
    # Round up Orange results
    for i in range(size):
      o = math.ceil(o_predicts[i] - 0.5)
      results['orange'].append(o)
      results['rapidminer'].append(r_predicts[i])
    matches = []
    sum_matches = 0
    # find matches
    for i in range(size):
      if results['rapidminer'][i] == results['orange'][i]:
        matches.append(1)
        sum_matches += 1
      else:
        matches.append(0)
    results['matches'] = matches
```

```
    results['percent_of_matches'] = round(sum_matches * 100 / size, 3)
    return results
  def compare_clusters(self, r_clusters, o_clusters):
    def _convert_rp_cluster_name(name):
      name = name.replace('cluster_', '')
      return int(name)
    assert len(r_clusters) == len(o_clusters)
    size = len(r_clusters)
    results = {'orange': copy.copy(o_clusters),
          'rapidminer': copy.copy(r_clusters),
          'distribution': {
            'orange': [],
            'rapidminer': [],
          }
          }
    r_clusters = results['rapidminer']
    o_clusters = results['orange']
    o_dict = {}
    r_dict = {}
    for i in range(size):
      r_clusters[i] = _convert_rp_cluster_name(r_clusters[i])
      rp_cluster = r_clusters[i]
      if rp_cluster not in r_dict:
        r_dict[rp_cluster] = 1
      else:
        r_dict[rp_cluster] += 1
      if o_clusters[i] not in o_dict:
        o_dict[o_clusters[i]] = 1
      else:
        o_dict[o_clusters[i]] += 1
    # convert dict to list
    o_list = [None for _ in range(len(o_dict))]
    r_list = [None for _ in range(len(r_dict))]
    for i in range(len(o_dict)):
      o_list[i] = o_dict[i]
      r_list[i] = r_dict[i]
    results['distribution']['orange'] = o_list
    results['distribution']['rapidminer'] = r_list
    results['clusters_intersection'] = self._find_clusters_crossing(o_clusters, r_clusters)
    evaluation_criteria = {
      'Silhouette index': {'orange': self._get_silhouette_metric(o_clusters),
                  'rapid_miner': self._get_silhouette_metric(r_clusters)},
      'Davies Bouldin index': {'orange': self._get_davies_bouldin_metric(o_clusters),
                    'rapid_miner': self._get_davies_bouldin_metric(r_clusters)},
      'Calinski Harabasz index': {'orange': self._get_calinski_harabasz_metric(o_clusters),
                     'rapid_miner': self._get_calinski_harabasz_metric(r_clusters)},
    }
    results['evaluation_criteria'] = evaluation_criteria
    return results
  # negative is bad, positive is ok
  def _get_silhouette_metric(self, clusters):
    silhouette_metric = metrics.silhouette_score(self.sources, clusters, metric='euclidean')
    return round(silhouette_metric, 4)
  # the lower the better
  def _get_davies_bouldin_metric(self, clusters):
    davies_bouldin_metric = metrics.davies_bouldin_score(self.sources, clusters)
    return round(davies_bouldin_metric, 4)
  # the higher the better
  def _get_calinski_harabasz_metric(self, clusters):
    calinski_harabasz_metric = metrics.calinski_harabasz_score(self.sources, clusters)
    return round(calinski_harabasz_metric, 4)
  def _find_clusters_crossing(self, or_clusters, rp_clusters):
    # return {1: [0,3,5,6,7]}
    def _fill_dict(c_dict, clusters_list, index):
      cluster = clusters_list[index]
      if not c_dict.get(cluster):
```

```
            c_dict[cluster] = {index}
          else:
            c_dict[cluster].add(index)
      def _percent_of_intersestion(list1, list2):
        # percent of intersection of list1 with list2
        list_len = len(list1)
        intersection_len = len(list1 & list2)
        return round(intersection_len * 100 / list_len, 3)
      r_dict = {}
      o_dict = {}
      for i in range(len(or_clusters)):
        _fill_dict(r_dict, rp_clusters, i)
        _fill_dict(o_dict, or_clusters, i)
      assert len(r_dict) == len(o_dict)
      results = {
        'orange': {},
        'rapidminer': {},
      }
      # Get crossing of clusters
      for r_cluster in r_dict:
        r_set = r_dict[r_cluster]
        results['rapidminer'][r_cluster] = dict()
        for o_cluster in o_dict:
          o_set = o_dict[o_cluster]
          if not results['orange'].get(o_cluster):
            results['orange'][o_cluster] = dict()
          o_intersection = results['orange'][o_cluster]
          r_intersection = results['rapidminer'][r_cluster]
          p1 = _percent_of_intersestion(o_set, r_set)
          p2 = _percent_of_intersestion(r_set, o_set)
          if p1 != 0:
            o_intersection[r_cluster] = p1
          if p2 != 0:
            r_intersection[o_cluster] = p2
      return results
```

2.6 Допоміжні модулі

2.6.1 Текст програми у файлі config.py

```
import configparser
CONFIG_PATH = './config.ini'
config = None
def get_config():
  global config
  if config:
    return config
  config = configparser.ConfigParser()
  config.read(CONFIG_PATH)
  return config
```

2.6.2 Текст програми у файлі utils.py

```
def read_cvs_file(path):
  num_of_initial_rows = 0
  content = []
  with open(path, 'r') as f:
    for l in f:
      if num_of_initial_rows == 0:
        headers = l
      else:
        content.append(l)
      num_of_initial_rows += 1
  return headers, content, num_of_initial_rows
def generate_test_cvs_file(file_path, source_content, source_headers, num_required_rows):
```

```
    rows = 0
    with open(file_path, 'w+') as fout:
      fout.write(source_headers)
      while rows != num_required_rows:
        for r in source_content:
          r = r.strip()
          r = r + '\n'
          if rows == num_required_rows:
            break
          fout.write(r)
          rows += 1
def convert_string_list_to_int(list_):
    new_list = []
    for item in list_:
      new_list.append(int(item.rstrip()))
    return new_list
```

Исследование сервисов технологии Data Mining

Васильева Д. В, Иванов А. П., Днепровский национальный университет
железнодорожного транспорта имени В.Лазаряна, Украина

Интеллектуальный анализ данных (англ. Data mining) – на данный момент одна из самых актуальных отраслей в мире информационных технологий, поскольку с ростом объемов данных становится вопрос об их эффективной обработке. Инструментами Data Mining являются программные средства или целые системы, с помощью которых выполняется подготовка данных, их обработка и дальнейший анализ результатов. Сервисы Data Mining реализуют различные алгоритмы интеллектуального анализа, а также осуществляют процессы машинного обучения. На данный момент рынок программных средств для Data Mining представляет множество решений. При выборе подходящего инструмента, стоит учитывать, что он может существенно повлиять на скорость работы с данными, а также на качество полученных знаний.

Data mining – это процесс обнаружения в сырых данных ранее неизвестных нетривиальных практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. К распространённым примерам использования интеллектуального анализа данных относят задачи классификации, кластеризации, поиска аномалий и прогнозирования. Для решения подобных задач существуют готовые сервисы, библиотеки, которые предоставляют уже реализованные алгоритмы для обработки данных. Наиболее популярными средствами являются такие программные обеспечения как: Rapid Miner, Orange, Weka, Sisense, Revolution, Qlik.

Исходя из того, что на данный момент единственная возможность сравнить эффективность программных средств — это их запуск и оценка вручную, было предложено разработать систему для автоматизированного выполнения вышеупомянутой задачи. Цель исследований состоит в том, чтобы провести сравнительный анализ полученных результатов для двух сервисов. На данном этапе в качестве исследуемых систем были выбраны Orange и Rapid Miner. Для их оценки предлагается рассматривать каждую из реализованных ими функций по отдельности, а именно такие функции как: кластеризация методом k-средних, нахождение выбросов по алгоритму LOF (Local outlier factor), создание линейной регрессионной модели, предсказание значений на основании линейной регрессионной модели. Рекомендуется использовать соответствующие метрики качества для каждого из алгоритмов с целью определения степени удовлетворительности полученных результатов и дальнейшего их анализа. Для оценки временной эффективности сервисов можно использовать графики зависимости времени обработки данных конкретной функцией от объема обрабатываемых данных. Упомянутая возможность позволяет оценить эффективность работы сервисов на больших таблицах, а также сравнить их между собой.

Данный подход к решению задачи по исследованию сервисов технологии Data Mining позволяет ускорить процесс изучения подобных систем. На данном этапе разработано программное обеспечения для сравнения сервисов Orange и Rapid Miner, в дальнейшем оно может быть расширено с целью анализа большого количество сервисов, используемых для решения похожих задач. В перспективе приложение может стать универсальной системой для поиска и подбора программного обеспечения, которое лучше остальных выполняет необходимые аналитику задачи.

Исследование сервисов технологии Data Mining

Васильева Д. В, Иванов А. П., Днепровский национальный университет
железнодорожного транспорта имени В.Лазаряна, Украина

Интеллектуальный анализ данных (англ. Data mining) – на данный момент одна из самых актуальных отраслей в мире информационных технологий, поскольку с ростом объемов данных становится вопрос об их эффективной обработке. Инструментами Data Mining являются программные средства или целые системы, с помощью которых выполняется подготовка данных, их обработка и дальнейший анализ результатов. Сервисы Data Mining реализуют различные алгоритмы интеллектуального анализа, а также осуществляют процессы машинного обучения. На данный момент рынок программных средств для Data Mining представляет множество решений. При выборе подходящего инструмента, стоит учитывать, что он может существенно повлиять на скорость работы с данными, а также на качество полученных знаний.

Data mining – это процесс обнаружения в сырых данных ранее неизвестных нетривиальных практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. К распространённым примерам использования интеллектуального анализа данных относят задачи классификации, кластеризации, поиска аномалий и прогнозирования. Для решения подобных задач существуют готовые сервисы, библиотеки, которые предоставляют уже реализованные алгоритмы для обработки данных. Наиболее популярными средствами являются такие программные обеспечения как: Rapid Miner, Orange, Weka, Sisense, Revolution, Qlik.

Исходя из того, что на данный момент единственная возможность сравнить эффективность программных средств — это их запуск и оценка вручную, было предложено разработать систему для автоматизированного выполнения вышеупомянутой задачи. Цель исследований состоит в том, чтобы провести сравнительный анализ полученных результатов для двух сервисов. Поскольку тема является очень обширной, то на данном этапе в качестве исследуемых систем были выбраны Orange и Rapid Miner. Для их оценки предлагается рассматривать каждую из реализованных ими функций по отдельности. На данный момент были отобраны наиболее актуальные задачи Data Mining, а именно такие функции как: кластеризация методом k-средних, нахождение выбросов по алгоритму LOF (Local outlier factor), создание линейной регрессионной модели, предсказание значений на основании линейной регрессионной модели.

Для достижения указанной выше цели была использована технология оценки полученных результатов с помощью подсчета необходимых метрик. Для каждой из анализируемых функций подобраны соответствующие показатели качества. Предусмотрены опции для замеров времени обработки данных сервисом, как для одиночного эксперимента, так и для серии экспериментов с целью получение графика зависимости времени выполнения функции от объемов обрабатываемых данных.

Модель линейной регрессии

Качество функции поиска уравнения линейной регрессии связывают с адекватностью модели наблюдаемым данным. Проверка адекватности модели регрессии по отношению к наблюдаемым данным проводится на основе анализа остатков. Остатки представляют отклонения фактического значения зависимой переменной от значения данной переменной, полученной расчетным путем. Анализ остатков проходит с использованием следующих показателей:

Коэффициент детерминации – характеризует долю дисперсии результативного признака, объясняемую регрессией, в общей дисперсии признака. Рассчитывается для оценки качества подбора уравнения регрессии. Для приемлемых моделей предполагается, что коэффициент детерминации должен быть хотя бы не меньше 50%.

Средняя квадратическая ошибка – представляет собой среднее квадратическое отклонение наблюдаемых значений результативного признака от рассчитанных значений. Подчеркивает большие ошибки.

Корень средней квадратической ошибки – используется для уменьшения ошибки, в случае если в данных присутствуют выбросы.

Средняя абсолютная ошибка – альтернативный способ для вычисления разницы между наблюдаемыми результатами и рассчитанными. Эффективна при использовании на данных с выбросами.

При исследовании функции нахождения линейной регрессии у сервисов Rapid Miner и Orange, были подобраны специальные наборы данных, имеющие такую зависимость между атрибутами. Выполнены эксперименты по прогнозированию данных на найденных выборках. Стоит заметить, что обучение модели линейной регрессии проводилось на специально отобранных данных, в то время как оценка качества полученной модели была рассчитана для контрольной выборки.

Поиск выбросов

Оценить качество нахождения выбросов в данных не тривиальная задача, поскольку для таких целей сложно найти подходящие метрики. Разработанная программа имеет функционал для оценивания найденных выбросов, который будет полезным, в случае если для указанного набора данных выбросы уже были заранее определены экспертом. Упомянутая функция использует метрику *AUC-ROC* [1]. Таким образом, имея фактические выбросы и выбросы, полученные сервисами, можно проанализировать, как исследуемые программы справляются с решением данной задачи. В дальнейшем ориентируясь на полученные результаты можно выбрать подходящий сервис для нахождения выбросов в похожих данных.

Кластеризация

Существует множество показателей оценки качества кластеризации, которые при расчёте используют внутренние, внешние или гибридные характеристики исследуемого набора данных. Внешние меры оценки качества использую дополнительные знания о кластеризуемом множестве, к примеру: распределение по кластерам, количество кластеров. Внутренние меры опираются только на структуру кластеров. Принимая к вниманию тот, факт, что пользователь не всегда имеет дополнительную информацию о наборе данных, кроме непосредственно той, что представлена в таблице, было принято решение о использовании исключительно внутренних метрик для сравнения и анализа полученных кластеров. Исходя из этого, полученные в результате работы сервисов кластеры, оцениваются по следующим показателям:

Индекс Дэвиса-Боулдина (англ. Davies-Bouldin index) [2] – внутренняя схема оценки, при которой проверка того, насколько хорошо была проведена кластеризация, осуществляется с использованием количеств и характеристик, присущих набору данных. Определяет среднюю «похожесть» между кластерами, где «похожесть» — это мера, которая сравнивает расстояние между кластерами с размером самих кластеров.

Коэффициент Силуэтте (англ. Silhouette) [3] – показывает, на сколько среднее расстояние до объектов текущего кластера отличается от среднего расстояния до объектов ближайшего кластера. Значения коэффициента лежат в диапазоне от минус одного до единицы. Числа, близкие к минус одному, можно интерпретировать как плохую кластеризацию в то время, как значения, близкие к единице, говорят о качественно сформированных кластерах, а значение, близкие к нулю, говорят о том, что кластеры пересекаются и накладываются друг на друга.

Индекс Калински-Харабаш (англ. Calinski–Harabasz) [4] – также известный как критерий коэффициента дисперсии. Представляет отношение суммы межкластерной дисперсии и дисперсии внутри кластера для всех кластеров. Чем выше полученное значение – тем лучше показатели. Коэффициент устойчив к разной размерности кластеров и к наличию близких друг к другу кластеров.

Таким образом, получив вышеописанные показатели качества для каждого из сервисов, пользователь может принять решение, каким результатам следует отдать предпочтение. После проведения множества экспериментов, можно сделать выводы о том, как алгоритмы, реализуемые исследуемым сервисом, справляются с большими объёмами данных, какая из программ предоставляет более качественные кластеры и на каких данных падает эффективность алгоритмов.

Рекомендуется использовать соответствующие метрики качества для каждого из алгоритмов с целью определения степени удовлетворительности полученных результатов и дальнейшего их анализа.

Временная эффективность

Немаловажным показателем качества работы сервиса Data Mining является его временная эффективность. Актуальность данного показателя, объясняется тем, что аналитикам, исследователям приходится обрабатывать всё большие объемы данных, для извлечения из них полезной информации. Очевидно, что скорость обработки данных является важным аспектом при выборе инструмента для анализа. Исходя из этих потребностей, в программе, разработанной для анализа и сравнения сервисов, был реализован функционал замера времени выполнения функции, а также построения графика для неё, где отображается зависимость времени работы сервиса, от объема обрабатываемой информации. Такая возможность может быть особенно полезна, когда необходимо спрогнозировать как поведет себя программа на больших данных.

Таким образом, для того чтобы провести эксперимент с помощью разработанного инструмента, пользователю необходимо выполнить ряд тривиальных действий. Для начала необходимо указать программе путь к данным, стоит заметить, что на текущем этапе в качестве входных данных принимается только формат csv. После того, как данные будут введены, необходимо выбрать функцию и сервисы для исследований. Присутствует возможность выполнить предварительную обработку данных: нормализация и/или удаление выбросов. Нормализированные значения особенно важны при поиске уравнения линейной регрессии. Следующим этапом можно выбрать дополнительные параметры для сравнительной оценки сервисов, а именно: подсчет времени выполнения задачи, построение графика зависимости. Финальная стадия – нажатие на соответствующую кнопку для начала работы инструмента. После завершения обработки данных и расчёта метрик – программа отобразит результаты.

Разработанное ПО в разы ускоряет процесс изучения сервисов технологии Data Mining.

Эксперимент

При исследовании сервисов технологии Data Mining было проведено множество экспериментов, в ходе которых анализировались следующие моменты:

- какой из сервисов составляет дает более качественные результаты для:
  - линейной регрессии;
  - кластеризации методом k-средних;
  - обнаружений аномалий.
- как предварительная нормализация и/или удаление выбросов влияет на качество полученных результатов для Rapid Miner, Orange;
- потеря в качестве при обработке больших данных;
- скорость обработки данных на одноядерном и двухъядерном процессорах.

Все действия проводились исключительно с использованием разработанного инструмента. Найденные и проанализированные в ходе экспериментов результаты позволяют выявить следующие закономерности:

- качество обработки данных не значительно падает при больших объемах информации.
- сервис Orange вероятно использует встроенную предварительную нормализацию при нахождении модели линейной регрессии;
- ПО Rapid Miner обрабатывает данные медленнее чем Orange, поведение актуально как для одноядерного так и для двухъядерного процессора.

На рисунках 1 и 2 приведен пример временных характеристик, полученных на одноядерном и двухъядерном процессорах соответственно.

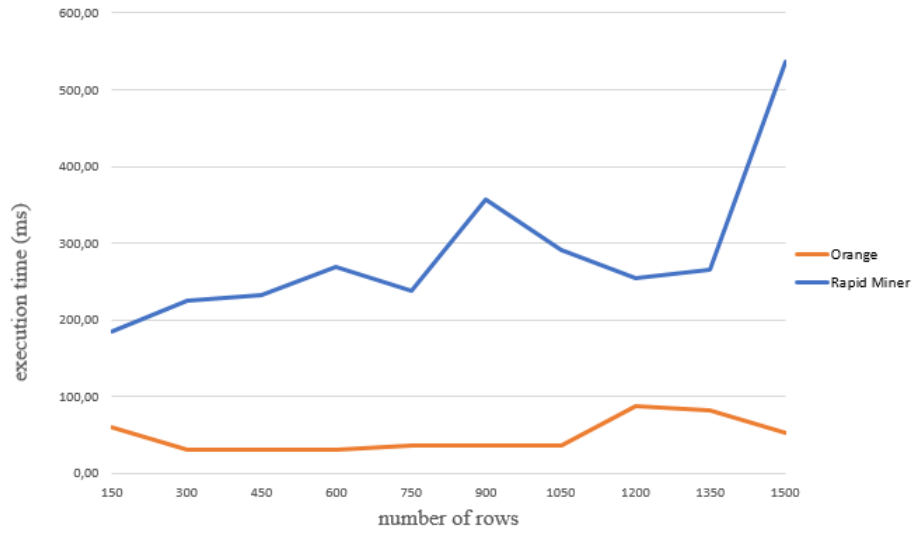


Рисунок 1. Диаграмма зависимости времени выполнения от объема данных на одноядерном процессоре

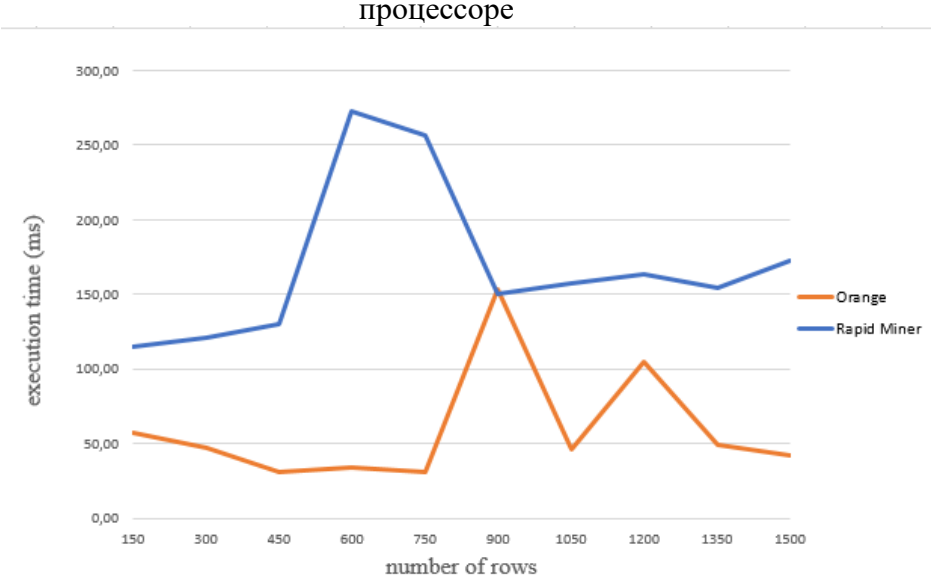


Рисунок 2. Диаграмма зависимости времени выполнения от объема данных на одноядерном процессоре

Подводя итоги вышесказанному, необходимо отметить, что разработанное ПО облегчило процесс исследования сервисов Data Mining. Полученные в ходе экспериментов результаты можно применить для оптимизации процессов исследования данных.

В дальнейшем инструмент можно расширять, добавляя новые сервисы и функции для изучения.

Список литературы:

1. Значение метрики AUC-ROC [Электронный ресурс]. Статья с web-сайта Understanding AUC - ROC Curve, Sarang Narkhede, Jun 26, 2018. – Режим доступа: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 (дата обращения: 07.10.2020).
2. Индекс Девиса-Боулдина [Электронный ресурс]. Материал с Википедии. Доступная энциклопедия. – Режим доступа: https://en.wikipedia.org/wiki/Davies%E2%80%93Bouldin_index (дата обращения: 10.10.2020).
3. Коэффициент Силуэтте [Электронный ресурс]. Материал с Википедии. Доступная энциклопедия. – Режим доступа: https://en.wikipedia.org/wiki/Silhouette_(clustering) (дата обращения: 10.10.2020).
4. Индекс Калински-Харабаш Індекс Калінскі-Харабаш [Електронний ресурс]. Статья с web-сайта Calinski-Harabasz index. – Режим доступа: https://www.oreilly.com/library/view/machine-learning-algorithms/9781785889622/8dba1062-2dbe-43ce-a9b0-9ea49203ea9a.xhtml (дата обращения: 10.10.2020)