НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Глазунова О.Г., д.т.н., професор	Голуб Б.Л., к.т.н., доцент
_____ **202_ р.**	_____ **202_ р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Експертна система дослідження стану атмосферного повітря

Спеціальність _____

(код і назва)

Освітня програма _____

(назва)

Орієнтація освітньої програми _____

(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_____ _____ _____

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

_____ _____ _____

(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Сарабанський О.М.

(підпис) (ПІБ студента)

КИЇВ-2022

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) _____

ЗАТВЕРДЖУЮ

Завідувач кафедри_____

_____

_____ _____ _____

(науковий ступінь, вчене звання) (підпис) (ПІБ)

“ ” _____ 20 ___ року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

_____

(прізвище, ім’я, по батькові)

Спеціальність _____

(код і назва)

Освітня програма _____

(назва)

Орієнтація освітньої програми_____

(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи_____

_____

затверджена наказом ректора НУБіП України від “ ” _____ 20 ___ р. №_____

Термін подання завершеної роботи на кафедру _____

(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи _____

_____

_____

_____

_____

Перелік питань, що підлягають дослідженню:

1. _____
2. _____
3. _____

Перелік графічного матеріалу (за потреби) _____

_____

Дата видачі завдання “ ” _____ 20 ___ р.

Керівник магістерської кваліфікаційної роботи _____ _____

(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ _____

(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП

Актуальність. Агресивне використання природних ресурсів в наслідок технічного перевороту 19 століття, показало що людина неефективно використовує наявні резерви. Відбиток цієї революції ми бачимо дотепер від радіоактивної Чорнобильської зони до забруднень атмосферного повітря в наслідок виробництва у Китаї. Задля попередження небезпек як для людини так і природи в цілому варто почати діяти над вирішенням проблем забруднення і переходу до концепції «Сталого розвитку». Для початку дій що до вирішення наявних екологічних бід, одним із перших пунктів є збір і аналітика даних навколишнього середовища, на основі яких будуть базуватись наступні кроки по вирішенню екологічних проблем. Основою для розробки будь-яких систем оцінок та прогнозування стану довкілля є поточний всебічний аналіз стану навколишнього середовища, впливів різних галузей народного господарства на довкілля. Експертна система зможе пришвидшити етап аналізу даних для кваліфікованих працівників, на основі яких, вони зможуть приймати конкретні рішення для розв’язання конкретних екологічних задач.

Предмет та об’єкт дослідження. Предметом даної роботи є експертна система, спрямована на дослідження стану атмосферного повітря. Об’єктом являється атмосферне повітря.

Мета дослідження. Дослідження основних чинників забруднення атмосферного повітря, задля попередження їх негативного впливу, як на людину так і навколишнє середовище.

Зміст поставлених завдань. Основні завдання даної роботи були:

1. Переосмислення наявної системи, її рефакторинг.
2. Створення архітектури OLAP-системи.
3. Організація сховища даних.

4. Імплементація Data mining алгоритмів.
5. Імплементація додаткових модулів до наявної системи.

Методи дослідження. Перевірка доцільності використання технологій Data Mining для задач виявлення збудників забруднень і прогнозування стану атмосферного повітря.

Наукова новизна. У даній роботі було запропоновано використання і удосконалення алгоритмів обробки інформації Data Mining до такої предметної області як забрудненість атмосферного повітря.

Апробація. За темою дипломної роботи відбувся виступ «Підсистема аналізу даних системи моніторингу екологічних параметрів навколишнього середовища» на конференції «Теоретичні та прикладні аспекти розробки комп'ютерних систем 2021» та опубліковані тези у збірнику «Теоретичні та прикладні аспекти розробки комп'ютерних систем».

Структура пояснювальної записки. У першому розділі було проведено системний аналіз предметної області, проаналізовано наявні рішення, поставлено завдання для проведення магістерського дослідження. У другому розділі було змодельовано систему з використанням об'єктно-орієнтованого підходу, побудовано діаграми прецедентів, послідовності і активності. У третьому розділі описується архітектура експертної системи, розглянуто усі складові і представлено архітектури підсистем. Четвертий розділ показує результати дослідження, апаратні та програмні вимоги, обговорення отриманих результатів.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Атмосферне повітря і його забруднювачі

Забруднення атмосферного повітря одна з найболючіших проблем сучасності. Ще століття назад, склад атмосфери фактично, був не змінним на протязі останніх 300—400 років. Однак бурхливий ріст промисловості, стрибкоподібний вибух автомобільного транспорту, авіації, промислового виробництва нафтохімічних продуктів, побутових хімічних засобів, обробка сільськогосподарських угідь з літаків, сміттєзвалища, привели до прогресуючого збільшення забруднення атмосферного повітря, і ця тенденція стрімко продовжується і в 21 сторіччі.

Атмосферні забруднювачі — це речовини, які накопичуються в повітрі до такого ступеня, що є шкідливим для живих організмів або матеріалів, що знаходяться в повітрі. Звичайні забруднювачі повітря включають дим, смог і такі гази, як окис вуглецю, оксиди азоту та сірки, а також вуглеводневі пари. У той час як газоподібні забруднювачі зазвичай невидимі, тверді або рідкі забруднювачі в димі та смогі легко побачити

Одна особливо шкідлива форма забруднення повітря виникає, коли оксиди сірки та азоту поєднуються з атмосферною вологою з утворенням сірчаної та азотної кислот. Коли кислоти приносяться на Землю у вигляді кислотних дощів, шкода завдається озерам, річкам, рослинності, будівлям та іншим об’єктам. Оскільки оксиди сірки та азоту можуть переноситися на великі відстані в атмосфері, перш ніж вони будуть видалені з опадами, пошкодження можуть виникнути далеко від джерел забруднення.

Дим є давнім забруднювачем навколишнього середовища, але збільшення використання викопного палива в останні століття посилило його серйозність. Дим може посилити симптоми астми, бронхіту та емфіземи, а тривалий вплив може призвести до раку легенів. Токсичність диму зростає, коли разом із ним

вдихаються пари діоксиду сірки, який зазвичай виділяється під час спалювання вугілля. Один особливо серйозний випадок із забрудненням повітря стався в Лондоні в 1952 році, коли приблизно 4000 смертей стали результатом високого рівня диму та діоксиду сірки, що накопичився в столичній зоні під час атмосферної інверсії[1].

Науково підтверджено кореляційний зв'язок між станом забруднення атмосферного повітря та захворюваністю. Всесвітня організація охорони здоров'я (ВООЗ) констатує, що забруднення повітря призводить до збільшення захворюваності та смертності в світі. За даними цієї ж організації, забруднення атмосферного повітря є пріоритетним чинником ризику для здоров'я населення, при цьому понад 80% захворювань тією чи іншою мірою залежать від якості повітря [2].

Для адекватної оцінки використовується «Індекс якості повітря» у кожної країни є свій показник індексу відповідно до різних національних стандартів. При великому індексі, велика частина населення може зіткнутись з проблемами здоров'я. Цей показник здебільшого вираховується на основі шести базових забруднювачів повітря, а саме:

- діоксиду сірки (SO2);
- твердих часток (PM10);
- дрібних твердих часток (PM2.5)
- діоксиду азоту (NO2)
- оксиду вуглецю (CO)
- озону (O3)

Діоксид сірки(SO2) - за звичайних умов являє собою безбарвний газ з різким задушливим запахом. Токсична дія може виявлятися вже в малих концентраціях (20-30 мг/м3) і створює неприємний смак у роті, дратує слизові оболонки очей і дихальних шляхів. Знижує опірність до респіраторних захворювань. При виливі

на організм подразнює верхні дихальні шляхи, викликаючи запалення слизових оболонок носоглотки, бронхів. Високі концентрації оксиду сірки в повітрі викликають у людини задишку, можуть призвести до утрати свідомості. За даними головного управління статистики Харківської області на у розрахунку на одну людину, обсяг викидів становить 10,1 кілограм [2]. Основними збудниками цієї речовини є викиди при виробничих процесах (див. рис. 1) і спалювання сірковмістних видів палива.

Рис. 1 Промислові викиди

PM10 це частинки речовин діаметр яких становить від 10 мікрометрів і менше, PM2.5 це частинки речовин діаметром 2.5 мікрометра і менше. Пил у більшості складається з частинок цих розмірів, завдяки своєму розміру вони потрапляють глибоко у легені на відміну від частинок більших розмірів, які зупиняються в межах носу, роту і горла. Склад цього пилу може варіюватися від місцезнаходження і залежати від таких факторів як близкість до заводів, доріг, будівництв [3]. Основне джерело: цвіль, спори, пилок, дим, бруд і пил із заводів та сільського господарства. Вигляд міста з високою концентрацією PM10\PM2.5 можна побачити знизу.

Рис. 2 Вигляд міста з високою концентрацією PM10\PM2.5

Діоксид азоту впливає переважно на дихальні шляхи і легені, а також змінює склад крові, зокрема зменшує вміст у крові гемоглобіну, особливо у людей з серцево-судинними захворюваннями. Вплив діоксиду азоту на організм людини знижує її опірність до захворювань, призводить до кисневого голодування тканин, особливо у дітей, підсилює дію канцерогенних речовин, сприяючи виникненню злоякісних новоутворень .

Оксид вуглецю (чадний газ) – отруйна речовина, що не має кольору і запаху є результатом неповного згоряння палива, який міститься у вихлопних газах. Вступаючи в реакцію із гемоглобіном крові, оксид вуглецю утворює стійке з'єднання – карбоксигемоглобін, яке утруднює процес газообміну в клітинах, що призводить до кисневого голодування (спорідненість гемоглобіну з оксидом вуглецю приблизно у 210 разів вище його спорідненості з киснем). За його впливу порушується центральна нервова система, уражується дихальна система, знижується гострота зору. Перевищення норм концентрації CO особливо небезпечно для людей з серцевосудинними захворюваннями. За вмісту в повітрі 0,05% CO через годину у людини спостерігається слабке отруєння, за вмісту 1%

– людина втрачає свідомість після декількох вдихів [4]. Основними збудниками цієї речовини є виробництва і автомобілі (див. рис. 3).

Рис. 3 Автомобільні викиди

Озон – це блідно-голубий токсичний газ, який має гострий запах, який нагадує хлор. В високих шарах стратосфери концентрація цього газу формує озоновий шар, який захищає все живе від ультрафіолетового випромінювання. Але у тропосфері цей газ є небезпечним для людини, оскільки при концентрації більше ніж 0,1 мг/м^3 він здатен пошкоджувати слизисту і систему дихання.

1.2 Стан атмосферного повітря України

За даними Державної служби статистики, у 2020 році викиди забруднюючих речовин в атмосферу від стаціонарних джерел забруднення складали 2238,6 тис. т. або на 220,9 тис. т (на 9,0%) менше ніж у минулому році [5]. Але це буде наївно складати думку про позитивну екологічну динаміку. Враховуючи роки пандемії COVID економічна ситуація дещо погіршилась, тому виробництва дещо сповільнились. Найбільші викиди від стаціонарних джерел у 2020 році спостерігались у Донецькій області 750,9 тис. т, Дніпропетровській –

534,6 тис. т та Запорізькій області – 155,4 тис. т.[5] З цих даних можна зробити припущення що основними збудниками є виробництва і заводи, оскільки дані області є промисловими центрами України.

Антропогенне і техногенне навантаження на атмосферне повітря в Україні у кілька разів перевищує відповідні показники у розвинутих країнах світу. Основними забруднювачами атмосферного повітря залишаються підприємства добувної і переробної промисловості, постачання електроенергії, газу, пари та кондиційованого повітря, викиди забруднюючих речовин яких складають понад 90% від загального обсягу викидів в атмосферне повітря в Україні (див. табл. 1).

Обсяги викидів забруднюючих речовин	тис.т	% до загального підсумку
1	2	3
Усього	***2238,6***	***100,0***
Сільське, лісове та рибне господарство	64,1	2,8
Добувна промисловість і розроблення кар`єрів	365,5	16,3
Переробна промисловість, у т.ч.	868,8	38,8
металургійне виробництво	729,8	32,6
Постачання електроенергії, газу, пари та кондиційованого повітря	849,2	37,9
Водопостачання; каналізація, поводження з відходами	17,3	0,7
Будівництво	2	0,08
Транспорт, складське господарство, поштова та кур'єрська діяльність	45,5	2

Табл. 1 Розподіл обсягу викидів по галузям

Стан атмосферного повітря в Україні зазначається як незадовільний, а у деяких регіонах (наприклад, Маріуполь, Кривий Ріг, Запоріжжя та ін.) - вкрай загрозливий. Такий стан обумовлений преш за все структурною деформацією економіки, коли перевага надається розвитку сировинно-видобувних і металоливарних (металургійних, гірничорудних, хімічних), досить брудних і надзвичайно екологічно небезпечних галузей промисловості.

Економіці України властива також висока питома вага ресурсних та енергоємних технологій, впровадження і нарощування яких у промисловості та

сільському господарстві здійснювалося найбільш «дешевим» способом - без будівництва відповідних очисних споруд.

Роки безконтрольної експлуатації природних ресурсів призвели до того, що у багатьох районах забруднення повітря у десятки разів перевищує гранично допустимі норми.

Головним джерелом забруднення атмосферного повітря в Україні від викидів стаціонарних джерел є підприємства паливно-енергетичного комплексу - 36% від загального обсягу викидів, підприємства обробної - 35% та видобувної промисловості - 25%. Основними забруднюючими речовинами є оксиди вуглецю, азоту, діоксиди сірки, аміак, феноли, формальдегід, бензапірен. Хоч обсяги викидів забруднюючих речовин останнім часом, передусім через зупинку багатьох підприємств, зменшилися, проте в деяких промислових регіонах (особливо - в Донецько-Придніпровському) вони і нині значно перевищують гранично допустимі норми [6].

1.3 Огляд інформаційних джерел та існуючих рішень

В ході аналізу функціональних можливостей, було знайдено декілька систем з схожими характеристиками:

1. Ресурс airly.org
2. Ресурс saveecobot.com.

Веб-ресурс airly.org створений приватною польсько-американською компанією Airly. Інтерфейс сайту зображено на рисунку 4.

Рис. 4 Інтерфейс сайту airly.org

Переваги сайту:

- прогнози
- наявність генератору звітів

Мінуси сайту:

- платно;
- відсутність багатьох українських станцій;

Сайт https://www.saveecobot.com/ створений українськими розробниками і націлений на моніторинг станцій місцезнаходження яких в Україні. Сайт також має свій чат-бот у додатках Messenger, Telegram і Viber. У веб-ресурсу є можливість переглядати графіки індексу якості повітря, місцезнаходження станції і її останні параметри. Інтерфейс сайту зображений на рисунку 5.

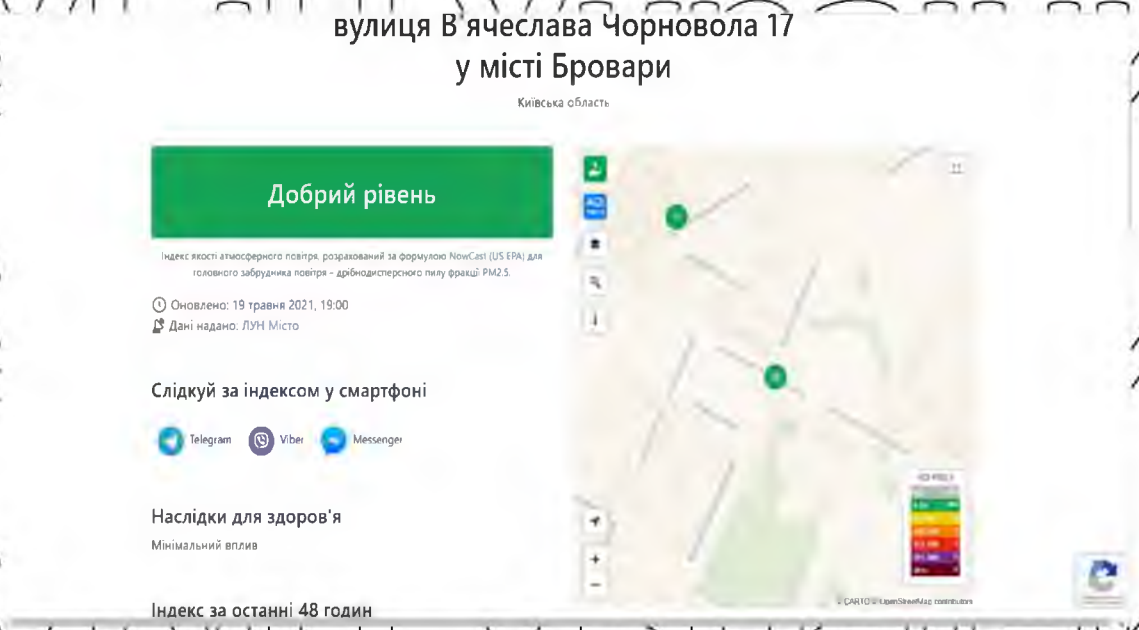

Рис. 5 Інтерфейс сайту saveecobot.com

Переваги веб-ресурсу:

- зручний сайт;
- кількість станцій;
- експорт даних;
- наявність власного чат-боту.

Мінуси сайту:

- графіки лише по Air Quality Index;
- закупівля станцій від одного вендора по одній ціні.

При аналізі цих ресурсів було виявлено декілька функціональних недостатків, а саме неможливість обробки даних при відключенні від глобальної мережі інтернет і відсутність можливості сформувати звіт у окремий файл.

1.4 Постановка завдання

Основні задачі поставлені впродовж етапу системного аналізу:

1. Переосмислення наявної системи, її рефакторинг.
2. Створення архітектури OLAP-системи.
3. Організація сховища даних.
4. Імплементація Data mining алгоритмів.

Одною з головних задач є організація сховища даних і імплементація алгоритмів Data Mining на основі даних з OLAP-системи.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Загальні поняття технології Data Mining

Інтелектуальний аналіз даних (ІАД, Data Mining), або розвідка даних - термін, що застосовується для опису здобуття знань у базах даних, дослідження даних, обробки зразків даних, очищення і збору даних. Це процес виявлення кореляції, тенденцій, шаблонів, зв'язків і категорій.

Термін Data Mining дістав назву від двох понять: дані - data і переробка сирого матеріалу (гірської руди) - mining.

Data Mining - предметна область" що виникла і розвивається на базі таких наук, як прикладна статистика, розпізнавання образів, штучний інтелект, теорія баз даних тощо.

Виникнення і розвиток Data Mining зумовлені різними факторами, серед яких вирізняємо основні: вдосконалення програмно-апаратного забезпечення; вдосконалення технологій зберігання і запису даних; накопичення великої кількості ретроспективних даних; вдосконалення алгоритмів обробки інформації.

Сутність і мету технології Data Mining можна описати так: це технологія, призначена для пошуку у великих інформаційних масивах даних неочевидних, об'єктивних, корисних на практиці закономірностей. ІАД здійснюється за допомогою використання технологій розпізнавання шаблонів, а також статистичних і математичних методів.

При розвідці даних багаторазово виконуються операції і перетворення над "сирими" даними (відбір ознак, стратифікація, кластеризація, візуалізація і регресія), що призначені для знаходження:

- о структур, які інтуїтивно зрозумілі для людей і краще розкривають суть бізнес-процесів, що лежать в основі їх протікання;

- о моделей, які можуть передбачити результат або значення певних ситуацій, використовуючи історичні або суб'єктивні дані.

Інтелектуальний аналіз даних - процес автоматичного пошуку прихованих закономірностей або взаємозв'язків між змінними у великих масивах необроблених даних, що поділяється на задачі класифікації, моделювання і прогнозування. Класичне визначення цього терміна дав у 1996 р. один із засновників цього напряму Г. П'ятецький-Шапіро.

Data Mining - це процес виявлення у необроблених даних раніше невідомих нетривіальних, практично корисних і доступних інтерпретацій знань, необхідних для прийняття рішень у різних сферах діяльності.

За визначенням SAS Institute, Data Mining - це процес виділення, дослідження і моделювання великих обсягів даних для виявлення невідомих до цього структур (patterns) з метою досягнення переваг у бізнесі.

За визначенням Gartner Group, Data Mining - це процес, мета якого - виявляти нові кореляції, зразки і тенденції у результаті просіювання великого обсягу даних з використанням методик розпізнавання зразків і статистичних та математичних методів.

В основу технології Data Mining покладено концепцію шаблонів (patterns), що є закономірностями, які властиві вибіркам даних і можуть бути подані у формі, зрозумілій людині.

Задачі Data Mining:

- 1. Класифікація (Classification) - виявляються ознаки, які характеризують групи об'єктів досліджуваного набору даних - класи; за цими ознаками новий об'єкт можна віднести до того або іншого класу. Для вирішення задач класифікації можуть використовуватися методи: найближчий сусід (Nearest Neighbor); к-найближчий сусід (k-Nearest Neighbor); байє-совські

мережі (Bayesian Networks); індукція дерев рішень; нейронні мережі (neural networks).

- 2. Кластеризація (Clustering) - результатом її є поділ об'єктів на групи.
- 3. Асоціація (Associations) - знаходять закономірності між пов'язаними подіями у наборі даних. Найбільш відомий алгоритм рішення задачі пошуку асоціативних правил - алгоритм Apriori.
- 4. Послідовність (Sequence), або послідовна асоціація (sequential association), - дає можливість знайти часові закономірності між транзакціями. Завдання послідовності подібне до асоціації, але її метою є встановлення закономірностей між подіями, пов'язаними за часом, тобто послідовність визначається високою ймовірністю ланцюжка пов'язаних за часом подій.
- 5. Прогнозування (Forecasting) - на основі особливостей історичних даних оцінюються майбутні значення показників. Застосовуються методи математичної статистики, нейронні мережі тощо.
- 6. Визначення відхилень (Deviation Detection), аналіз відхилень або викидів - виявлення й аналіз даних, що найбільше відрізняються від загальної чисельності даних, виявлення нехарактерних шаблонів.
- 7. Оцінювання (Estimation) - зводиться до прогнозу безперервних значень ознак.
- 8. Аналіз зв'язків (Link Analysis) - задача знаходження залежностей у наборі даних.
- 9. Візуалізація (Visualization, Graph Mining) - створюється графічний образ аналізованих даних. Для вирішення задач візуалізації використовуються графічні методи, що показують наявність закономірностей в даних.
- 10. Підбивання підсумків (Summarization) - опис конкретних груп об'єктів за допомогою аналізованого набору даних.

Зазначені вище задачі поділяються за призначенням на описові і предиктивні.

Описові, або дескриптивні (descriptive), задачі пов'язані з поліпшенням розуміння аналізованих даних. Ключовий момент у таких моделях - простота і прозорість результатів для сприйняття людиною. До такого типу задач належать кластеризація і пошук асоціативних правил.

Рішення предиктивних (predictive), або прогнозуючих, задач поділяється на два етапи. На першому етапі на підставі набору даних з відомими результатами будується модель. На другому етапі вона використовується для прогнозу результатів на підставі нових наборів даних. Вимагається, щоб побудовані моделі працювали максимально точно. До цього типу задач відносять задачі класифікації і регресії. Сюди можна віднести і задачу пошуку асоціативних правил, якщо результати її рішення можуть бути використані для прогнозу появи деяких подій.

За способами рішення задачі поділяють на такі, що вирішують за допомогою вчителя і без його допомоги. Категорія навчання з учителем представлена такими задачами Data Mining: класифікація, оцінка, прогнозування, категорія навчання; без учителя - задачею кластеризації.

У випадку рішення з допомогою вчителя задача аналізу даних розв'язується у кілька етапів. Спочатку за допомогою конкретного алгоритму Data Mining будується модель аналізованих даних - класифікатор. Потім класифікатор піддається навчанню. Іншими словами, перевіряється якість його роботи і, якщо вона незадовільна, відбувається додаткове навчання класифікатора. Так продовжується доти, доки не буде досягнуто необхідного рівня якості або не стане зрозуміло, що обраний алгоритм не працює коректно з даними, або дані не мають структури, здатної проявитися. До цього типу задач відносять задачі класифікації і регресії.

Рішення без допомоги вчителя об'єднує задачі, що виявляють описові моделі, наприклад, закономірності в часових рядах макропоказників. Очевидно, якщо ці закономірності існують, то модель має їх проявити. Перевагою цих задач є можливість їх рішення без будь-яких попередніх знань про дані аналізу. До них належать кластеризація і пошук асоціативних правил[6].

2.2 Огляд інструментарію для реалізації задач Data Mining

Для побудови OLAP-кубів та проведення OLAP-аналізу використовувалося середовище Visual Studio, оскільки воно має такі переваги:

- засоби Business Intelligence;
- робота з базами даних SQL Server;
- зручний графічний інтерфейс.

Для створення БД використовувалося середовище Microsoft SQL Server Management Studio за рахунок таких переваг:

- зручний графічний інтерфейс;
- багаті можливості;
- висока продуктивність;
- надійність.

Як середовище розробки програми для наповнення БД була використана Visual Studio. Вибір даного програмного забезпечення обумовлений такими можливостями:

- робота з об'єктами баз даних в браузері об'єктів SQL Server;
- визначення таблиць у новому конструкторі таблиць;
- запис асинхронного коду простим та інтуїтивно зрозумілим способом;
- отримання відомостей про об'єкт, що викликає, які допомагають з трасуванням та налагодженням.

Як мову програмування була обрана мова C#, так як вона має наступні переваги:

- об'єктно-орієнтованість;
- орієнтація на безпеку коду;
- багатофункціональність;
- відносна простота.

2.3 Структура джерела інформації для проведення інтелектуального аналізу

Сховище даних (англ. Data Warehouse) — предметно-орієнтована інформаційна база даних, спеціально розроблена та призначена для підготовки звітів та бізнес-аналізу з метою підтримки прийняття рішень в організації. Будується на базі систем управління базами даних та систем підтримки прийняття рішень. Дані, що надходять до сховища даних, зазвичай доступні тільки для читання.

Дані з OLTP-системи копіюються в сховище даних таким чином, щоб при побудові звітів та OLAP-аналізі не використовувалися ресурси транзакційної системи та не порушувалася її стабільність. Є два варіанти оновлення даних у сховищі:

- повне оновлення даних у сховищі. Спочатку старі дані видаляються, потім відбувається завантаження нових даних. Процес відбувається з певною періодичністю, при цьому актуальність даних може відставати від OLTP-системи;
- інкрементальне оновлення – оновлюються лише ті дані, які змінилися на OLTP-системі.

Основні принципи створення сховища даних включають в себе:

- Проблемно-предметна орієнтація. Дані об'єднуються у категорії та зберігаються відповідно до областей, які вони описують, а не з додатками, які вони використовують.
- Інтегрованість. Дані об'єднані так, щоб вони задовольняли всі вимоги підприємства в цілому, а не єдину функцію бізнесу.
- Некоректованість. Дані у сховищі даних не створюються: тобто надходять із зовнішніх джерел, не коригуються та не видаляються.
- Залежність від часу. Дані в сховищі точні та коректні лише в тому випадку, коли вони прив'язані до деякого проміжку або часу[7].

Діаграма прецедентів – це діаграма де показано відношення між клієнтами системи і варіантами використання, являється одією з видів діаграм UML, призначених для моделювання динамічних систем.[6]

Діаграми прецедентів – основний вид діаграм при моделюванні поведінки системи, підсистеми чи класу. Кожна з них показує набір варіантів користування і діючих осіб в їх взаємодії.

Діаграми прецедентів використовуються для моделювання представлення системи з точки зору варіантів використання. В більшій степені це означає моделювання контексту системи, підсистеми чи класу або є моделювання вимог до цих елементів.

Діаграма прецедентів важлива для візуалізації, специфікування і документування поведінки елементу. Вона забезпечує доступність і зрозумілість систем, підсистем і класів за рахунок зовнішнього представлення того, як ці елементи можуть бути використані в контексті. Крім того, такі діаграми важливі для тестування працюючих систем за допомогою прямого проєктування і для забезпечення їх розуміння з допомогою зворотнього проєктування [6].

Діаграма прецедентів, на основі якій створювалось сховище даних зображена на рисунку 6.

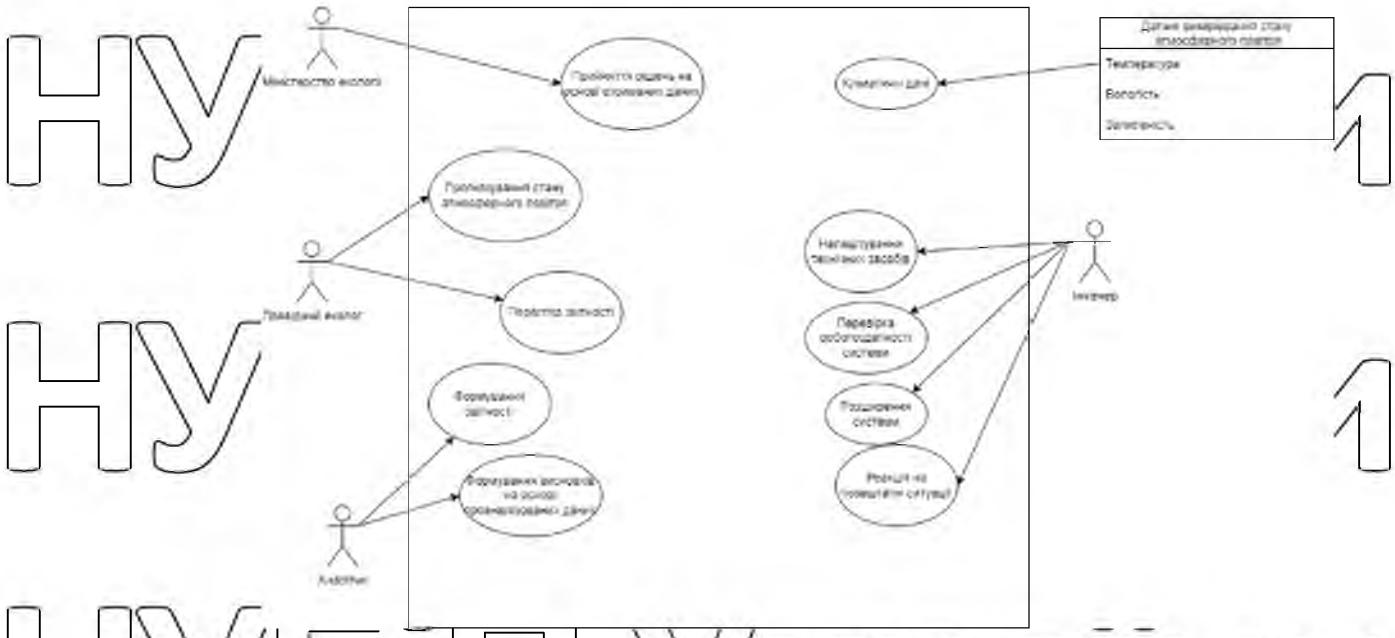

Рис. 6 Діаграма прецедентів

Збереження даних атмосферного повітря для подальшого їх аналізу було забезпечено СД, структура якого представлена на рис.7.

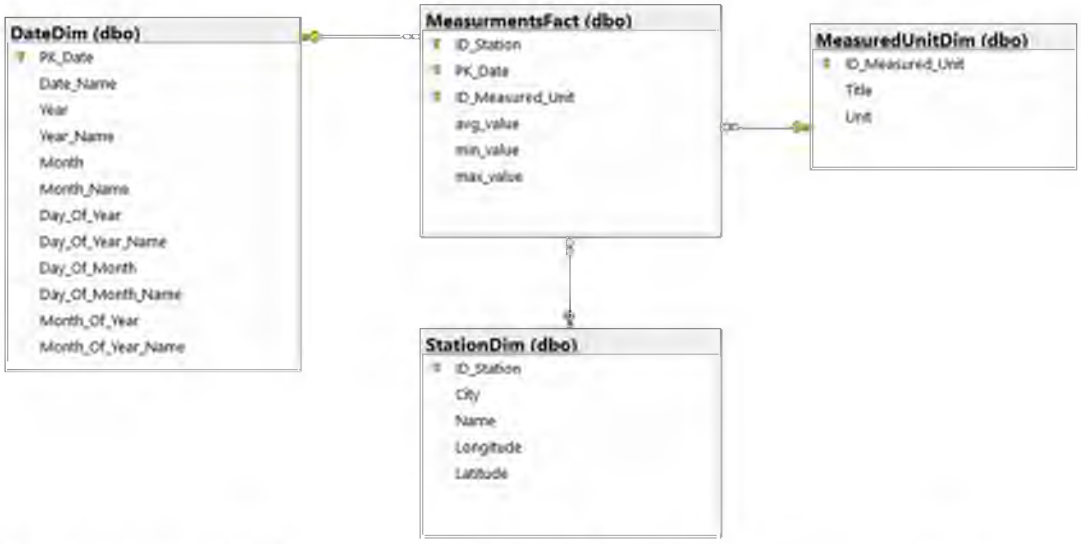

Рисунок 7. Структура сховища даних

Загальні поняття з напрямку OLAP-технології. Оперативні дані збираються з БД різних с/г об'єктів, очищаються, трансформуються і «складаються» в сховище даних.

Сховище представляє дані в більш зрозумілій для аналізу структурі. Користувач (аналітик) отримує інтуїтивно зрозумілу модель даних, у вигляді

багатовимірних кубів. Куб є структурою даних, яка забезпечує можливість швидкого аналізу даних за рамками обмежень реляційних баз даних. Куби здатні відображати і підсумовувати великі обсяги даних, також надаючи користувачам доступ до будь-яких точках даних з можливістю пошуку. Таким чином, дані можуть бути зведені, фрагментовані і оброблені в міру необхідності для вирішення найбільш широкого спектра питань, що відносяться до сфери використання системи. Аналітик може отримувати зведені (наприклад, по роках) або, навпаки, детальні (по тижнях) відомості та здійснювати інші маніпуляції в процесі аналізу. Інструментом, який забезпечує необхідні для аналізу маніпуляції над даними, є OLAP (Online Analytical Processing, оперативний аналіз даних).

За допомогою технологій OLAP користувач має можливість сформувати звіти, та зробити висновки по роботі відповідного с/г об'єкту, які в подальшому будуть представлені керівнику підприємства для допомоги в прийнятті управлінських рішень.

Вимір – це безліч об'єктів одного або декількох типів, організованих у вигляді ієрархічної структури і забезпечують інформаційний контекст числового показника (факту). Для збереження необхідних даних були розроблені такі таблиці вимірів:

- MeasuredUnitDim – містить дані про область, де знаходиться господарство, а саме назва регіону та кліматичної зони;
- StationDim – містить інформацію про господарство;
- DateDim – часовий вимір (Рік – місяць – день).

Факт – це величина (зазвичай числова), яка є предметом аналізу. Таблиця фактів представленого сховища:

MeasurmentDim, містить інформацію у розрізі часу, станції і виміру інформацію про:

- Середнє значення виміру;
- Максимальне значення виміру;
- Мінімальне значення виміру;

Таблиці вимірів є батьківськими щодо таблиці фактів, тож первинні ключі таблиць вимірів є зовнішніми ключами таблиці фактів. Первинний ключ таблиці фактів є складеним і складається з усіх зовнішніх ключів. Тип схеми *СД* – крижинка, оскільки наявна деталізована таблиця вимірів.

Архітектура рішення зображена в додатку А.

3 РОЗРОБКА СИСТЕМИ

3.1 Організаційна структура програмного забезпечення

Основою для побудови проєкту було обрано мову C#, яка входить в платформу .NET. Основні причини для вибору даної мови були: наявність конкретних фреймворків для вирішення задач дослідження, наявність збірника «сміття».

.NET це програмна платформа створена компанією Microsoft, основою якої є Common Language Runtime на якій засновуються високорівневі мови .NET. Основний причина у використанні шару CLR є кероване середовище, яке автоматично збирає сміття, керує доменами додатків і використання одних і тих самих наборів бібліотек що дозволить скоротити об'єм додатку. З основних плюсів платформи можна виділити: підтримку великої кількості мов програмування, спільний двигун для запуску додатків всіх мов, інтеграція мов (наприклад клас написаний на мові VB може використовуватись у C# і навпаки), величезна бібліотека класів, просте розгортання [8].

.NET є платформою яка підтримує багато мов програмування, але для вирішення задач дослідження було використано C# оскільки вона об'єктно-орієнтована, підтримує фреймворки WPF і ML.NET.

Для розробки експертної системи було використано систему WPF.

Windows Presentation Foundation була створена на заміну Windows Forms оскільки у остання не зберігалися симетрія для створення візуально яскравих додатків. Тобто для створення тривимірної графіки програмісту потрібно було взаємодіяти з DirectX API напряму, для двовимірної графіки програміст використовував GDI+, для стримінгу відео використовувався Windows Media Player API. Ці проблеми з уніфікації програмних інтерфейсів і вирішував WPF. Також WPF вирішив проблему поділу відповідальності, фреймворк Windows Forms мав недоліки у тому, що візуальна частина також була частиною коду що

не дозволяло програмісту гнучко налаштувати вигляд інтерфейсу. У системі WPF ця проблема була вирішена завдяки мові розмітки XAML. XAML це мова розмітки створена корпорацією Microsoft, яка заснована на мові XML. Оскільки у WPF код інтерфейсу не залежить від коду логіки додатку вигляд додатку можна налаштовувати гнучкіше ніж у Windows Forms. Також відмінністю від Forms є те, що WPF для рендерингу всіх вікон, анімацій, елементів управління використовує тільки DirectX API. Тому плюсами системи WPF є [9]:

- велика кількість менеджерів макетів, що дозволяє гнучко налаштовувати розташування елементів у вікні;
- використання зв'язування даних, що дозволяє використовувати MVVM модель;
- вбудований двигун стилів, що надає гнучко налаштувати тему додатка;
- використання векторної графіки, завдяки якій контент автоматично розширюється і позиціонується;
- підтримка 2D і 3D графіки, анімацій, відео і аудіо;
- широка підтримка документів різного розширення в тому числі XML Paper Specification, фіксованих документів тощо;
- зворотня підтримка з старішими моделями GUI в тому числі activex, Windows Forms тощо.

При створенні продукту були використані сучасні сучасні підходи програмування, для цього були використані фреймворки.

Фреймворк — інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проєкту та написання коду. Одна з головних переваг, при використанні каркасних застосунків, полягає в тому, що такі програми мають стандартну структуру. Каркаси застосунків стали популярними з появою

елементів інтерфейсу, які мали тенденцію до реалізації стандартної структури для додатків. З їх використанням стало набагато простіше створювати засоби для автоматичного створення графічних інтерфейсів, оскільки структура внутрішньої реалізації коду програми стала відома заздалегідь. Для забезпечення каркаса, зазвичай, використовують підходи об'єктно-орієнтованого програмування, наприклад, частини програми можуть успадковуватися від базових класів фреймворку.

Тобто фреймворк задає основну структуру програмного продукту і змушує програміста писати «правильний» код. Для створення підсистеми аналізу я використав фреймворки Prism і Entity framework.

На рисунку 8 зображено діаграму пакетів підсистеми. У ній ми бачимо що на стороні клієнту міститься вся основна логіка, користувацький інтерфейс і ORM який зв'язується з базою даних яка знаходиться на сервері.

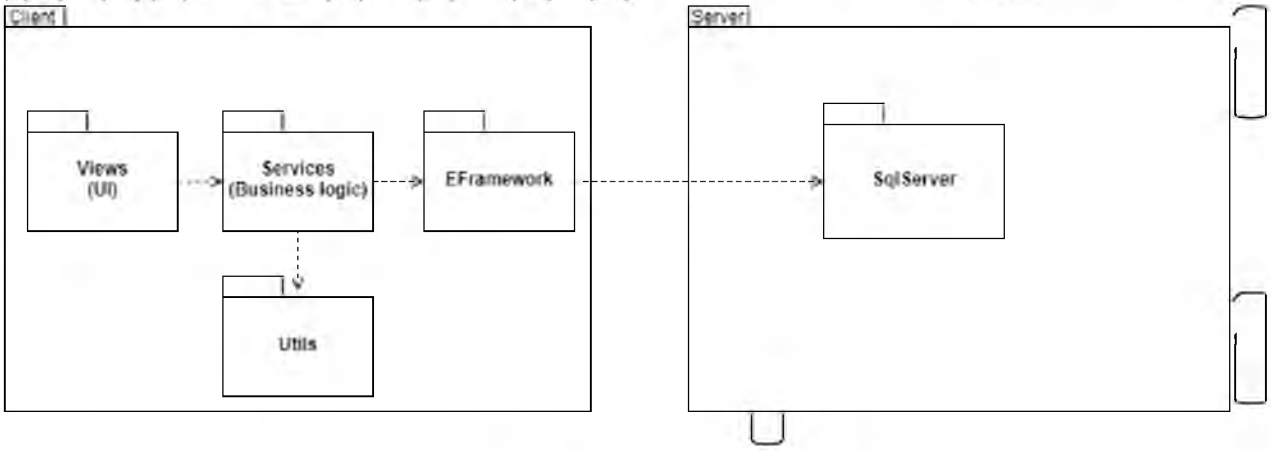

Рис. 8 Діаграма пакетів

Діаграма розміщення показує архітектуру майбутньої системи. Користувач взаємодіє з комп'ютером на якому встановлений додаток у вигляді артефакту. Головний артефакт містить у собі модулі конвертування даних у xls формат, бібліотеку створення графіків і DataAccessLayer, який використовує EntityFramework. З допомогою Unity контейнера вирішуються залежності при виборі режиму додатку офлайн або онлайн. Через конектори здійснюється обмін даними. В БД яка знаходиться в хмарному середовищі Azure надходять дані з

серверу який збирає дані з відкритих джерел і дані з сервера по MQTT протоколу. MQTT сервер збирає дані з станцій з допомогою GSM модуля. Зображена діаграма розміщення у додатку А.

3.2 Вибір інструментарію для створення прикладного програмного забезпечення

Для реалізації програмного продукту використовувалась IDE Visual Studio 2019. Вона має безкоштовну версію під назвою Community Edition. Її основним плюсом являється широка інтеграція з платформою .NET, а саме компілятором Roslyn. Цей компілятор є вбудованим у Visual Studio він автоматично аналізує код і надає список можливих виправень.

Для тестування підключення до бази даних і виконання запитів використовувався програмний продукт від компанії Microsoft під назвою Azure Data Studio. Цей редактор має вбудовану підтримку IntelliSense, інтеграцію з системами контролю версій і інтегрованим терміналом.

Prism – це фреймворк призначений для побудови слабко-зв'язних, легко підтримуваних додатків у системах WPF і Xamarin Forms. Фреймворк надає колекцію шаблонів проєктування які допомагають у створенні добре структурованих додатків наприклад. MVVM, ін'єкція залежностей, команди, агрегатор подій тощо. Бібліотека доступна на .NET Standard, .Net Framework і .NET Core.

Entity Framework - це набір технологій в ADO.NET, які підтримують розробку програмно-орієнтованих на дані програмних додатків. Архітектори та розробники програм, орієнтованих на дані, зазвичай борються з необхідністю досягнення двох дуже різних цілей. Вони повинні моделювати сутності, взаємозв'язки та логіку бізнес-проблем, які вони вирішують, а також повинні працювати з движками даних, що використовуються для зберігання та отримання даних. Дані можуть охоплювати кілька систем зберігання, кожна з

яких має власні протоколи; навіть програми, які працюють з єдиною системою зберігання, повинні збалансувати вимоги системи зберігання та вимоги щодо написання ефективного та ремонтопридатного коду програми [14].

LINQ компонент Microsoft .NET Framework, який додає нативні можливості виконання запитів даних до мов, що входять у .NET. Хоча порти існують для PHP (PHPLinq), JavaScript(linq.js), TypeScript (linq.ts), і ActionScript (ActionLinq), - жоден з них не є абсолютно еквівалентним LINQ в C# (де LINQ - не просто додаткова бібліотека, а частина мови).

LINQ розширює можливості мови, додаючи до неї вирази запитів, що є схожими на твердження SQL та можуть бути використані для зручного отримання та обробки даних масивів, XML документів, реляційних баз даних та сторонніх джерел.

LINQ також визначає набір імен методів (що називаються стандартними операторами запитів, або стандартними операторами послідовностей), а також правила перекладу, що має використовувати компілятор для перекладу текучих виразів у звичайні, використовуючи їх назву, лямбда-вирази та анонімні типи [15].

Інверсія керування це принцип побудови програми, при якому її частини отримують потік керування із загальної спільновикористовуваної бібліотеки. Це ніби звичайне процедурне програмування вивернуте навиворіт (inversed).

Однією з реалізацій IoC є , що використовується в багатьох фреймворках, вони називаються IoC контейнери. Використовуються в таких об'єктно-орієнтованих мовах програмування, як Smalltalk, C++, Java, PHP або мови платформи .NET [10].

Для реалізації проєкту використовувався IoC контейнер Unity. Unity контейнер це легкий, розширюваний контейнер впровадження залежностей. Він полегшує створення слабкозв'язного коду і надає розробникам такі можливості як:

- полегшене створення об’єктів, особливо для ієрархічних об’єктних структур;
- абстрагування вимог завдяки яким можна специфікувати залежність

ML.NET – безкоштовна відкрита бібліотека із засобами машинного навчання для мов програмування C# та F#. Вона також підтримує моделі на Python під час використання спільно з NimbusML. Попередній випуск ML.NET включав рішення для конструювання ознак (наприклад, створення N-грам), двійкової та мультикласової класифікацій, регресійного аналізу. Пізніше було додано додаткові завдання машинного навчання: виявлення аномалій та рекомендаційні системи.

3.3. Алгоритмізація та програмування програмних модулів

Підсистема була поділена на такі модулі:

- EcoSensors;
- ComparisonWindow;
- Login;
- Map ;
- Menu;
- SensorsInfo;
- MainScreen;
- Core;
- DAL;
- DAL.Models.
- ML.Models
- MLScreen

Більшість модулів системи були вже імплементовані в минулій роботі. Була оновлена кодова база модулів Map, SensorsInfo. EcoSensors я початковою точкою

і площиною для розміщення інших модулів. Після запуску відбувається перехід на вікно входу де користувач вводить свої дані і обирає в якому режимі він буде працювати. Після цього користувача переносить на вікно меню і інформації по датчикам.

Для створення ПЗ використовувався шаблон Model-View-ViewModel.

Model-View-ViewModel — це шаблон проєктування, що застосовується під час проєктування архітектури застосунків (додатків). Публічно вперше був представлений Джоном Госсманом (John Gossman) у 2005 році як модифікація шаблону Presentation Model [11]. Шаблон графічно зображено на рисунку 9.

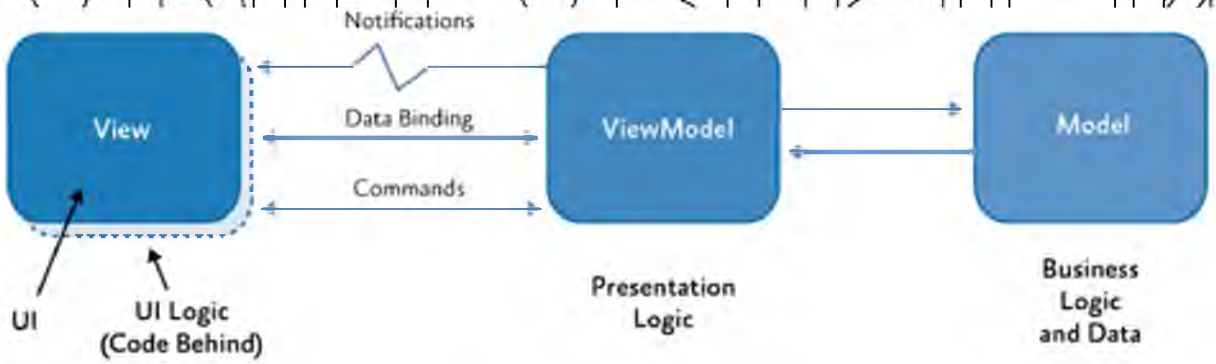

Рис. 9 MVVM шаблон

MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки (бек-енд логіки), відомої як модель (можна також сказати, що це відокремлення представлення від моделі).

Фреймворк Prism використовує MVVM шаблон при реалізації програмних продуктів. Фреймворк автоматично створює дві папки з назвами ViewModels і Views, де зберігаються моделі виглядів і вигляди відповідно. Для автоматичного зв'язування вигляду з моделлю вигляду використовується налаштування AutoWireViewModel. Модель вигляду містить команди і презентаційну логіку додатку, модель містить лише бізнес логіку, а вигляд містить XAML код з

графікою вікна і посилання на команди в контексті(моделі вигляду) вони виконуються. Вигляд структури папок зображений на рисунку 10

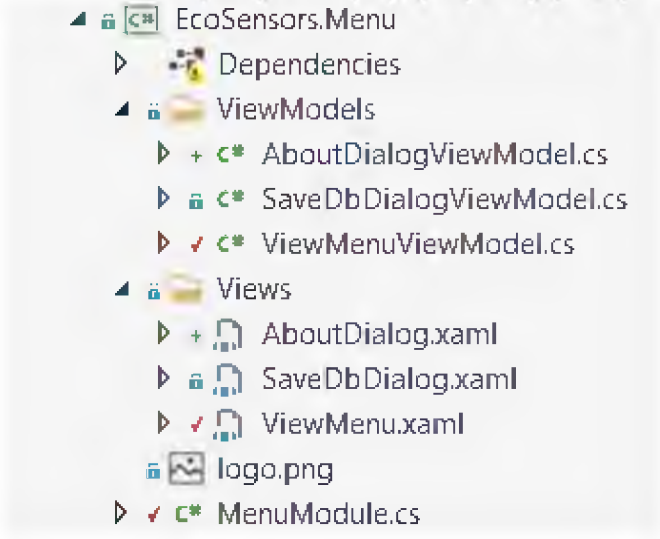

Рис. 10 структура папок MVVM моделі

Модель представлення є частиною, яка відповідає за перетворення даних для їх подальшої підтримки і використання. З цієї точки зору, модель представлення більше схожа на модель, ніж на представлення і обробляє більшість, якщо не всю, логіку відображення даних. Модель представлення може також реалізовувати шаблон медіатор, організовуючи доступ до бек-енд логіки навколо множини правил використання, які підтримуються представленням [12]

Шаблон MVVM ділиться на три частини.

Модель (Model), як і в класичному шаблоні MVC, Модель являє собою фундаментальні дані, що необхідні для роботи застосунку.

Вид/(Вигляд) (View) як і в класичному шаблоні MVC, Вигляд — це графічний інтерфейс, тобто вікно, кнопки тощо.

Модель вигляду (ViewModel, що означає «Model of View») з одного боку є абстракцією Вигляду, а з іншого надає обгортку даних з Моделі, які мають зв'язуватись. Тобто вона містить Модель, яка перетворена до Вигляду, а також

містить у собі команди, якими може скористатися Вигляд для впливу на Модель. Фактично ViewModel призначена для того, щоб

- Здійснювати зв'язок між моделлю та вікном
- Відслідковувати зміни в даних, що зроблені користувачем
- Відпрацьовувати логіку роботи View (механізм команд)

Для створення логіки використовувався шаблон команда. У фреймворку Prism є клас по створенню команди під назвою DelegateCommand. Цей клас реалізує сам шаблон без необхідності імплементувати всю логіку вручну. Конструктор об'єкту приймає два методи, перший метод під назвою Execute повинен не повертати значення і являється логікою команди яка змінює або оброблює дані, другий метод має назву CanExecute повертає булеве значення чи може команда активуватись. Метод CanExecute є свого роду валідатором. Також у ці методи можна передавати значення інших елементів. Наприклад на рисунку 11 передається SelectedDatesCollection, яка є колекцією обраних дат користувача

```
0 references
public XlsConverterDialogViewModel(IUnityContainer container)
{
    uow = container.Resolve<IUnityOfWork>();
    XlsDownloadButtonClicked = new DelegateCommand<SelectedDatesCollection>(XlsDownloadButton, (x) => (x != null) && (x.Count != 0));
```

Рис. 11 Команда

UML діаграма логіки шаблону команда зображено на рисунку 12.

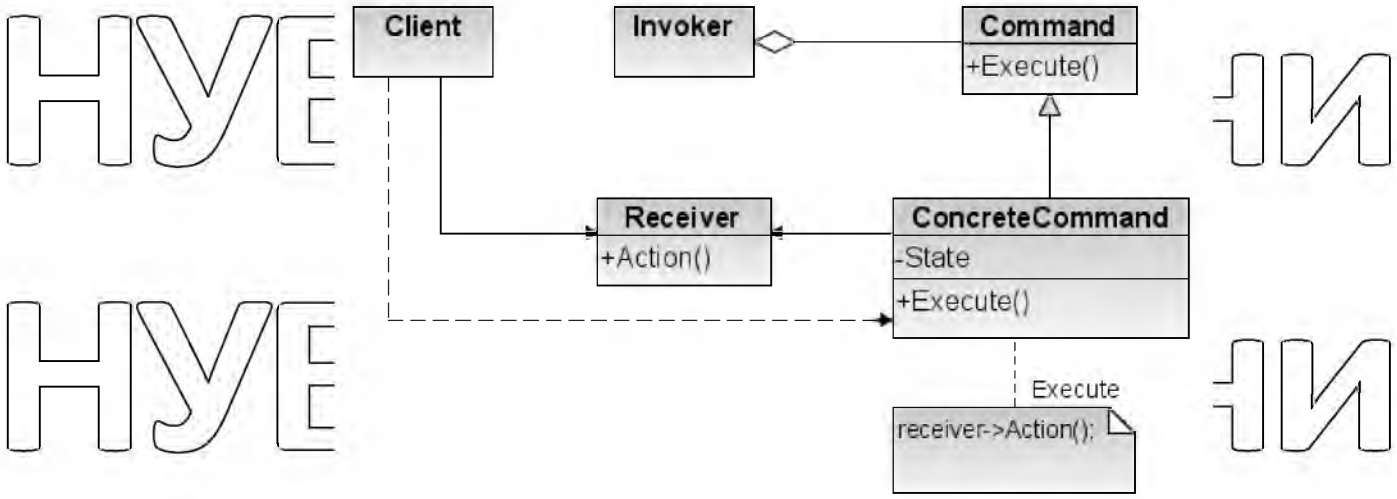

Рис. 12 UML діаграма опису поведінки «Команди»

- Command — команда: оголошує інтерфейс для виконання операції;
- ConcreteCommand — конкретна команда: визначає зв'язок між об'єктом-отримувачем Receiver та дією, реалізує операцію Execute шляхом виклику відповідних операцій об'єкта Receiver;
- Client — клієнт: створює об'єкт класу ConcreteCommand та встановлює його отримувача;
- Invoker — викликач: звертається до команди щоб та виконала запит;
- Receiver — отримувач: має у своєму розпорядженні усю інформацію про способи виконання операцій, необхідних для задоволення запиту. У ролі отримувача може виступати будь-який клас [13].

3.4. Реалізація інтерфейсу користувача

Для створення інтерфейсу користувача використовувалась мова розмітки XAML, вбудовані можливості Visual Studio 2019 і бібліотека MaterialDesign.

XAML – це основана на мові XML декларативна мова розмітки, створена компанією Microsoft. XAML включає в себе 4 основні категорії елементів: панелі, елементи управління, елементи документа і графічні фігури.

Для візуалізації карт використовувалась бібліотека з відкритим вихідним кодом Mapsui. Mapsui – це компонент мапи для додатків WPF і Xamarin. Розповсюджується дана бібліотека по ліцензії MIT.

Для створення графіків використовувалась відкрита крос-платформна бібліотека OxyPlot.

Oxyplot це крос-платформна бібліотека для створення графіків для платформи .NET. Кодова база має ліцензію MIT, ця ліцензія має мало заборон і є корпоративно-дружньою. Основою бібліотеки є Portable Class Library яка може використовуватись на різних платформах. Містить багато контролів для багатьох систем таких як WPF, Windows 8, Windows Phone, Windows Phone Silverlight, Windows Forms, Silverlight, GTK#, Xwt, Xamarin.IOS, Xamarin.Android, Xamarin.Forms і Xamarin.Mac.

Бібліотека містить багато типів осей і рядів, також при потребі є можливість збільшити функціональність бібліотеки, при наслідуванні класів або перезавантаженні коду відповідального за рендеринг. Самі графіки можуть експортуватися в такі формати як png, pdf і svc.

3.5 Забезпечення інтерфейсу з базою даних

Основним засобом з'єднання з базою даних є ORM Entity Framework яка внутрішньо побудована на ADO.NET. База даних sqlite не потребує рядка підключення. Для SqlServer було використано шаблон проєктування «фабричний метод» для створення контексту бази даних. У опціях ми вказуємо стратегію повторного підключення при помилці.

Ми обираємо з уявлення MeasurmentsStationsUnits, ідентифікатор, час, назву виміру і його значення створюючи анонімний об'єкт. Функція Where фільтрує всі дані по часу, ідентифікатору і назвою виміру. OrderBy сортує за часом. Ці вкладені запити функцій транслюються Entity Framework у запит БД. Цей LINQ запит зображено на рисунку 13.

```
var listOfPoints = _uow.dbContext.MeasurmentStationUnits.Select( x =>
    new
    {
        IdStation = x.IdStation,
        Time = x.Time,
        UnitTitle = x.UnitTitle,
        Value = x.Value
    }
    )
    .Where(x => x.IdStation == stationId &&
    collection.Max().Date >= x.Time.Date &&
    collection.Min().Date <= x.Time.Date &&
    x.UnitTitle == _selectedValCheckBox.MeasurmentName).OrderBy(x => x.Time);
```

Рис. 13 Запит до бази даних

Насправді при присвоєнні запиту LINQ ми отримуємо не запит а змінну типу IQueriable яка вже потім при виклику буде виконувати запит.

Також EF дозволяє запускати «сирі» запити. В випадку нижче ми виконуємо процедуру Get_Last_Measurment. Оскільки цей запит буде типу IQueriable цей запит також буде виконуватись при виклику перебору в змінній, якщо не вказано явне перетворення. Ви можете використовувати метод розширення FromSqlRaw, щоб розпочати запит LINQ на основі необробленого SQL-запиту. FromSqlRaw можна використовувати лише в кореневих структурах, тобто безпосередньо на DbSet. FromSqlInterpolated схожий на FromSqlRaw, але дозволяє використовувати синтаксис рядкової інтерполяції. Так само, як FromSqlRaw, FromSqlInterpolated можна використовувати лише в кореневих структурах. Значення перетворюється на DbParameter і не є вразливим для ін'єкції SQL. Запит до збереженої процедури з допомогою методу FromSqlInterpolated зображено на рисунку 14.

```
var data = uow.dbContext.MeasurmentStationUnits.FromSqlInterpolated($"Get_Last_Measurment @ID_Station = {station.Id}");
foreach (var measurment in data)
{
    station.Measurments.Add(
        new MeasurmentModel(
            measurment.UnitTitle,
            measurment.Value,
            CheckOptimalValue(measurment.Value, measurment.UnitTitle),
            measurment.Unit));
}
```

Рис. 14 Прямий запит до БД без використання LINQ

3.6 Data Mining, алгоритми регресії та класифікації

Для генерації моделей експертна система використовує ML.NET і вбудований Model Builder. Це розширення для Visual Studio дозволяє побудувати найбільш ефективну модель яка підходить для різних типів задач таких як регресії, класифікації, кластеризації тощо.

В ML.NET присутні такі алгоритми багатокласової класифікації як:

- LightGbmMulticlass
- SdcaMaximumEntropyMulticlass
- SdcaNonCalibratedMulticlass
- LbfgsMaximumEntropyMulticlass
- NaiveBayesMulticlass
- OneVersusAll
- PairwiseCouplingTrainer
- І такі алгоритми регресії:
-
- LbfgsPoissonRegressionTrainer
- LightGbmRegressionTrainer
- SdcaRegressionTrainer
- OlsTrainer
- OnlineGradientDescentTrainer
- FastTreeRegressionTrainer
- FastTreeTweedieTrainer
- FastForestRegressionTrainer
- GamRegressionTrainer

Регресійний аналіз — це набір статистичних методів, які використовуються для оцінки зв'язків між залежною змінною та однією або

кількома незалежними змінними. Його можна використовувати для оцінки міцності зв'язку між змінними та для моделювання майбутнього зв'язку між ними.

Класифікаційний аналіз — це завдання аналізу даних у інтелектуальному аналізі даних, яке визначає та призначає категорії колекції даних, щоб забезпечити більш точний аналіз. Метод класифікації використовує такі математичні методи, як дерева рішень, лінійне програмування, нейронні мережі та статистика.

Розширення ModelBuilder для ML.NET автоматично вибирає найбільш ефективну модель при генерації проекту. Процес побудови моделі складається з вибору сценарія, тобто який ми результат очікуємо від даних. Вибір сценаріє регресії це Value prediction (див. рис. 15).

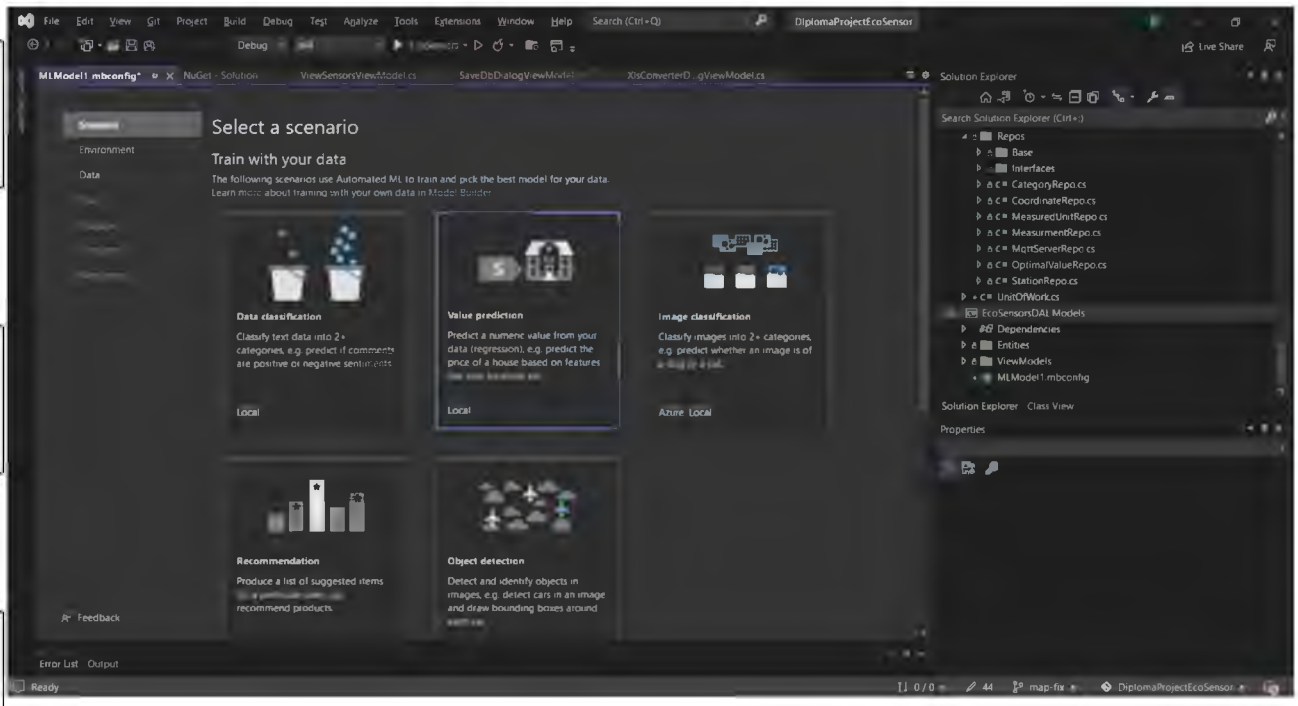

Рис. 15 Вибір сценарію регресії

Далі (див. рис. 16) йде вибір машини на якій буде здійснюватись тренування моделі. Також є можливість проводити тренування у хмарі або на відеокарті (для моделей розпізнавання образів).

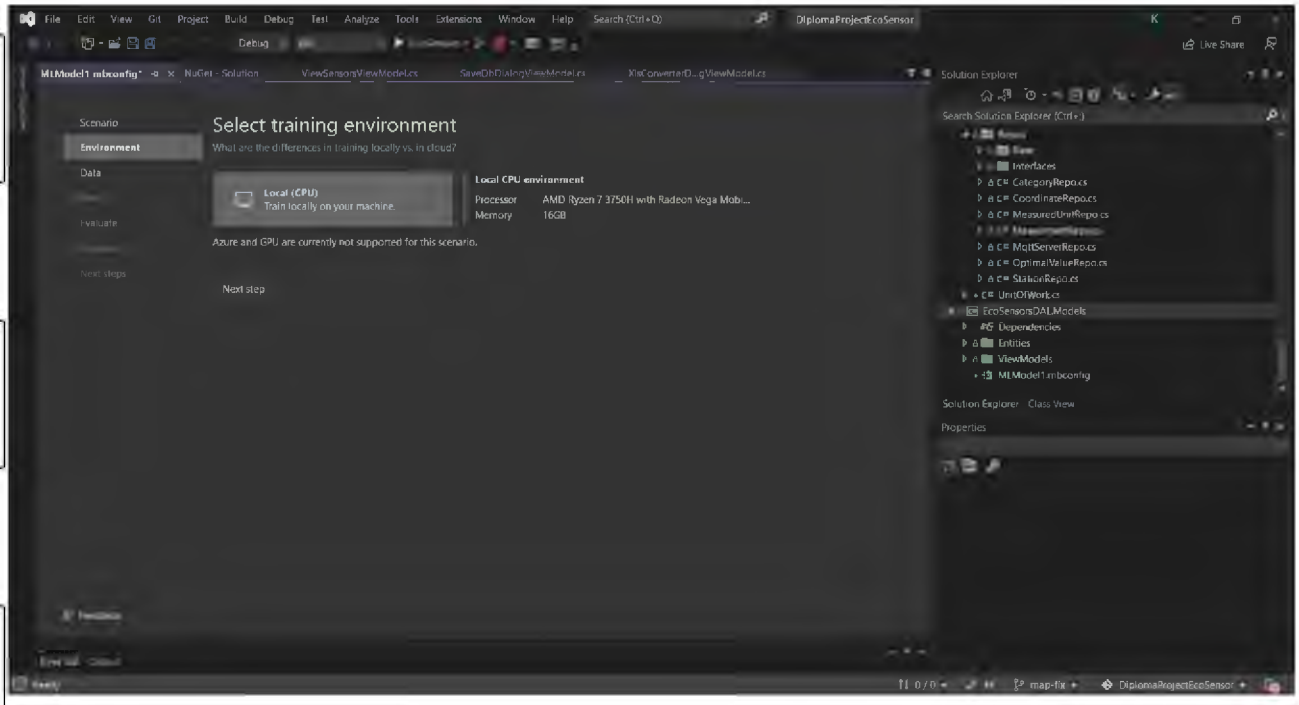

Рис. 16 Вибір середовища тренування

Після цього (див. рис. 17) йде вибір джерела даних, файл або сервер бази даних. В нашому випадку ми обираємо сервер сховища даних. Після того як ми підключились до сховища, ми обираємо таблицю на основі якої ми будем тренувати модель і значення яке потрібно спрогнозувати.

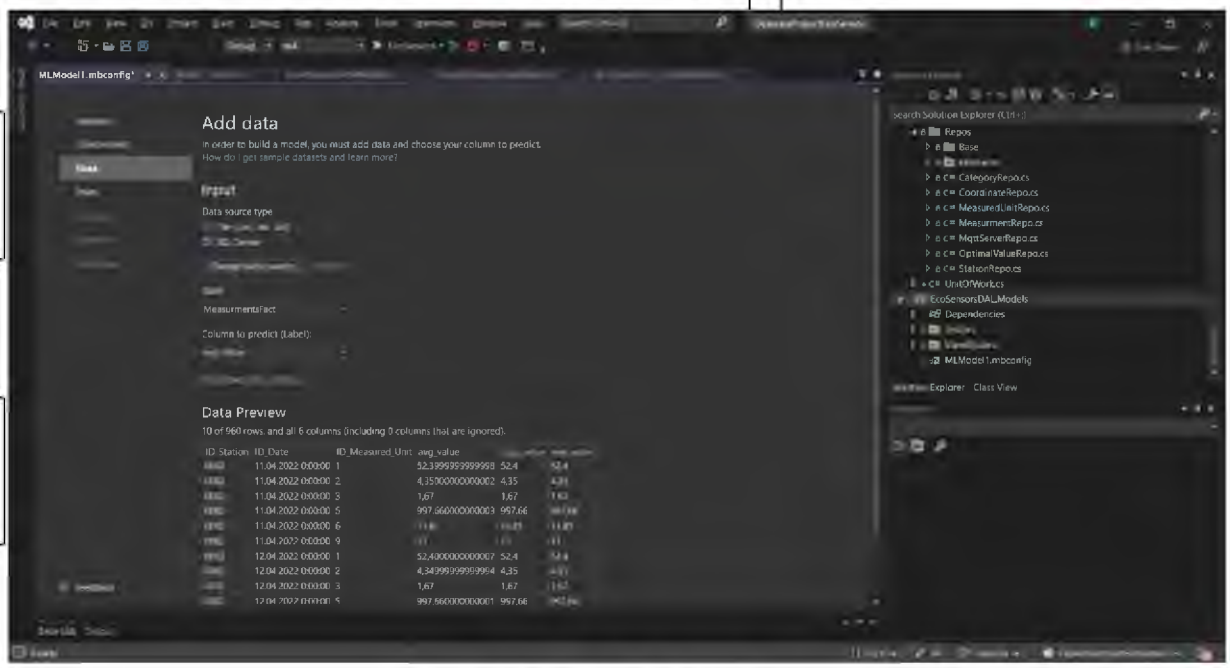

Рис. 17 Вибір вхідних даних

Далі йде (див. рис 18) вибір наскільки довго буде тренуватись модель. І процес тренування моделі.

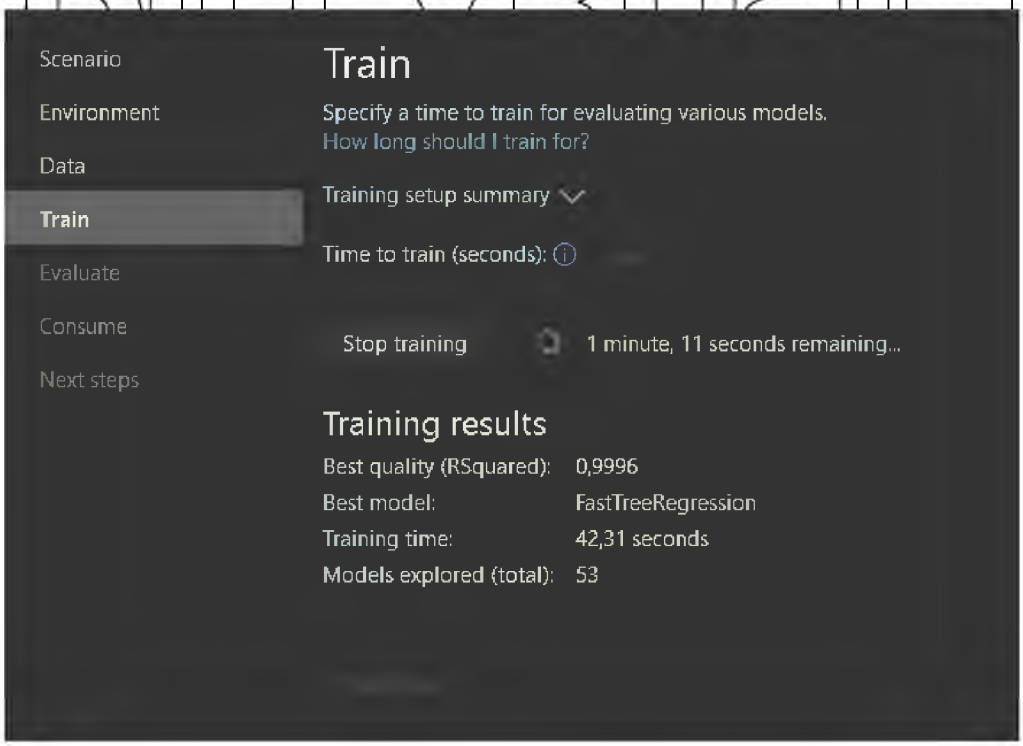

Рис. 18 Етап тренування

Напалі (див. рис. 19) можна вибрати найкращу модель і випробувати її Як ми бачимо найкращою моделлю є FastTreeRegression.

FastTree — це ефективна реалізація алгоритму посилення градієнта MART. Збільшення градієнта – це техніка машинного навчання для задач регресії. Він будує кожне дерево регресії поетапно, використовуючи попередньо визначену функцію втрат для вимірювання помилки на кожному кроці та виправляє її на наступному.

Рис. 19 Випробовування моделі

Ті ж самі процедури були проведені над класифікаційною моделлю.

3.7 Класифікація методом наївного Байєса

Створюємо структуру інтелектуального аналізу за допомогою Data Mining Wizard (див. рис. 20):

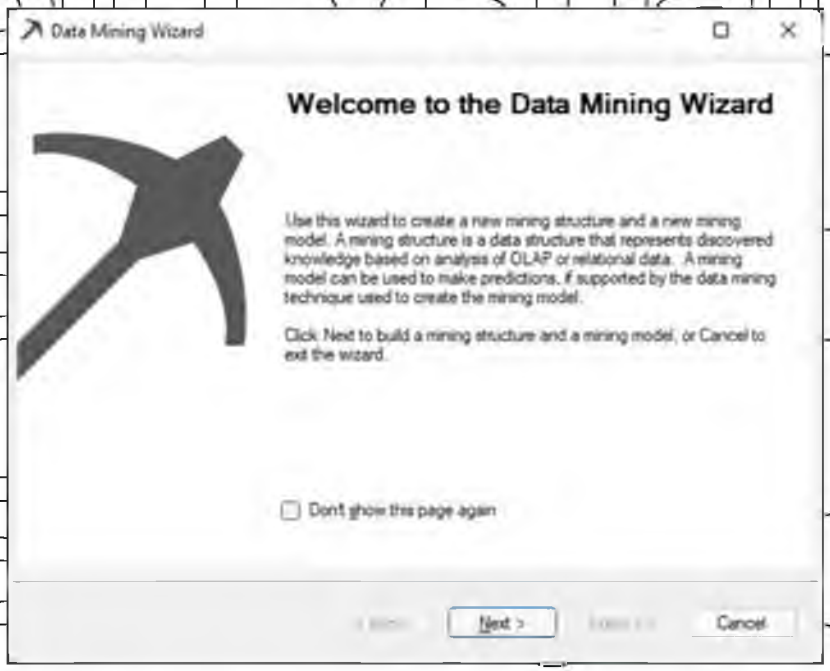

Рис. 20 Початок роботи з Wizard

Спочатку необхідно обрати джерело, на основі якого буде будуватися структура інтелектуального аналізу. Wizard пропонує два варіанти: куб, який сформований в проекті, будь яка інша реляційна БД або СД. Для роботи обираємо куб, який сформований в проекті(див. рис. 21).

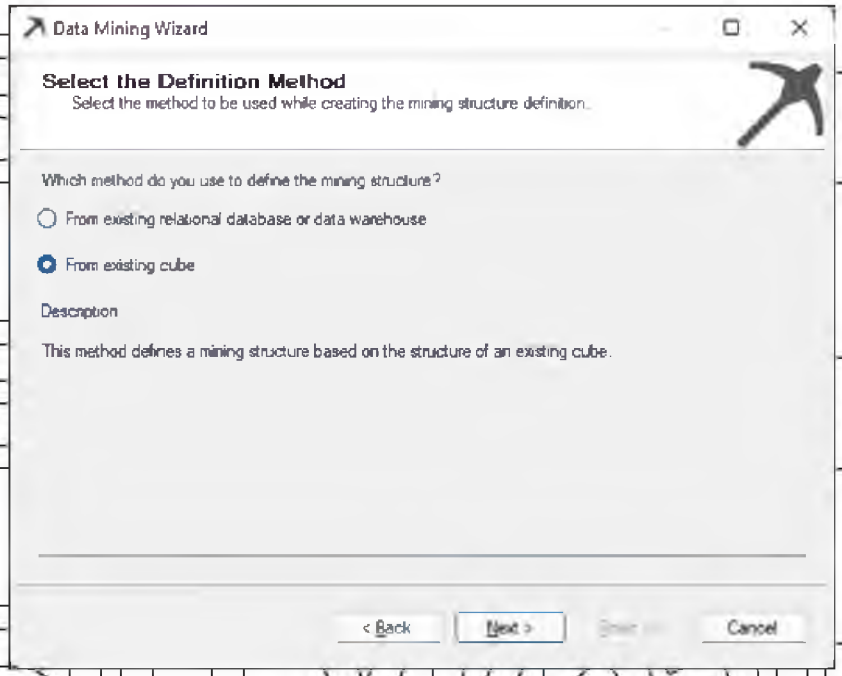

Рис. 21 Вибір вже створеного в проекті кубу

Далі обираємо метод наївного Байєса з переліку доступних технологій інтелектуального аналізу (див. рис. 22).

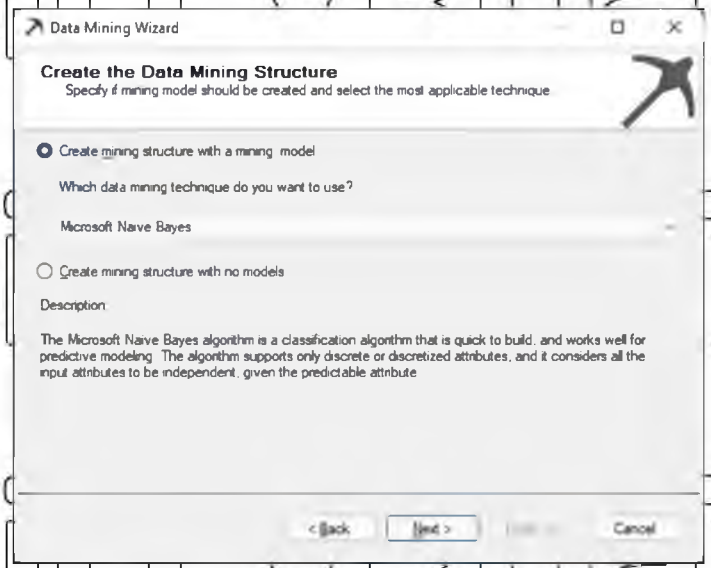

Рис. 22 Вибір технології аналізу даних

На наступному кроці обираємо таблицю вимірів, відповідно до яких буде показано та передбачено зміну фактичних показників. В задачі, яка вирішується, такою таблицею буде StationDim, що зберігає інформацію про станцію(рис. 23).

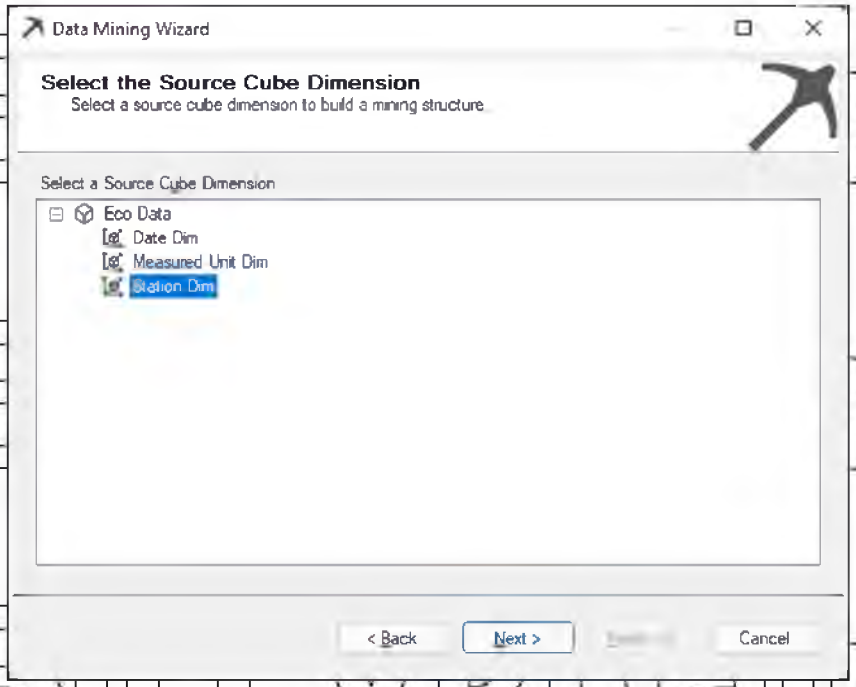

Рис. 23 Вибір таблиці вимірів

Далі необхідно обрати ключове поле. В якості ключового поля обираємо поле «Код станції» (рис. 24).

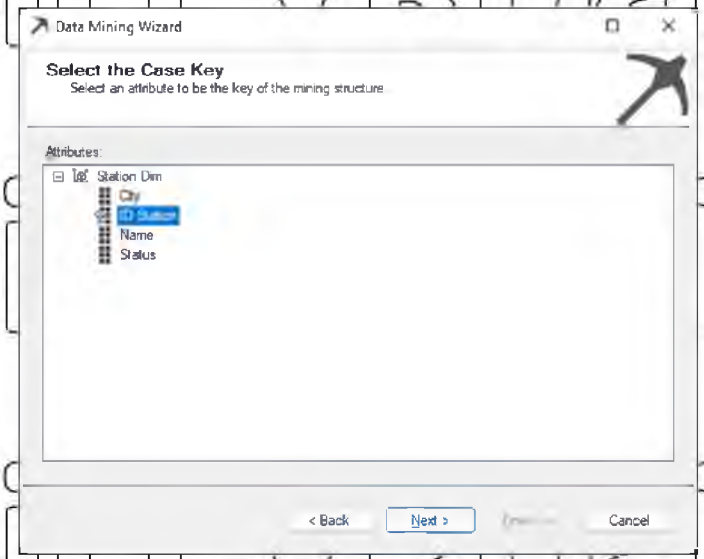

Рис. 24 Вибір ключового поля

Далі BI пропонує обрати атрибути, які будуть відображатися в моделі інтелектуального аналізу окрім ключового. Оскільки об'єкти необхідно

розподілити по класам за забрудненістю, обираємо обчислювальну міру, яка розраховує середнє значення вимірюваної величини(рис.25).

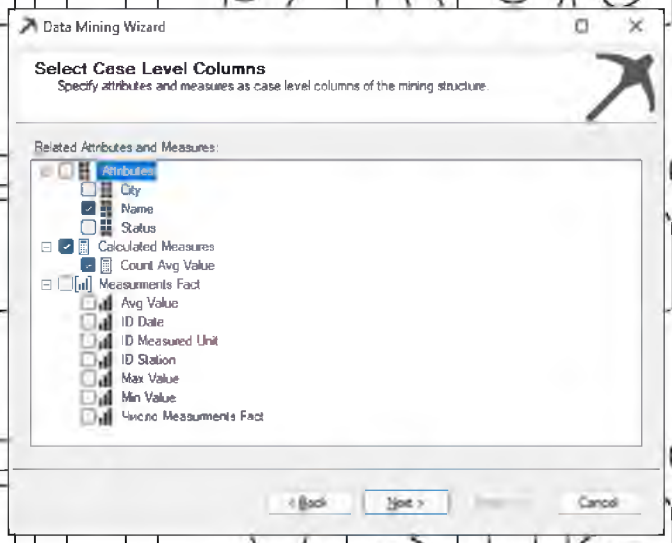

Рис. 25 Вибір атрибутів

Далі для обраних атрибутів необхідно визначити налаштування (рис.26).

- Input – вхідна зміна, яка значною мірою впливає на перебіг процесу, який досліджується (Станція);
- Predict – зміна, значення якої будуть передбачатися (Середнє значення вимірюваної величини).

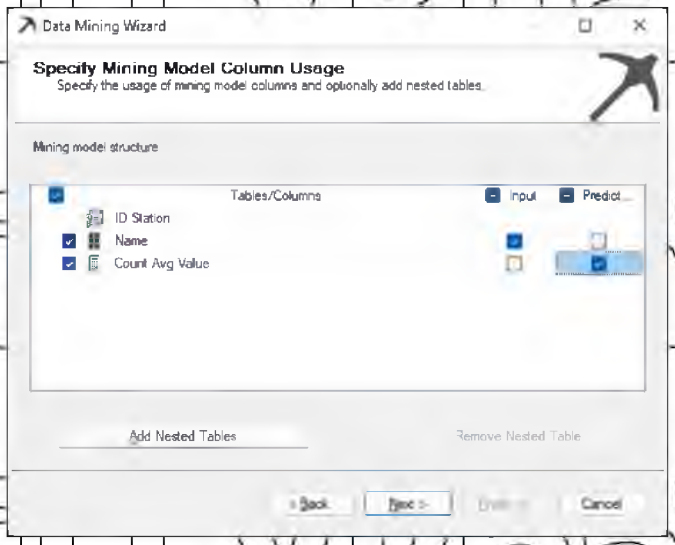

Рис. 26 Налаштування атрибутів

Структура моделі, яка була отримана в результаті наведених кроків зображена на рис. 27.

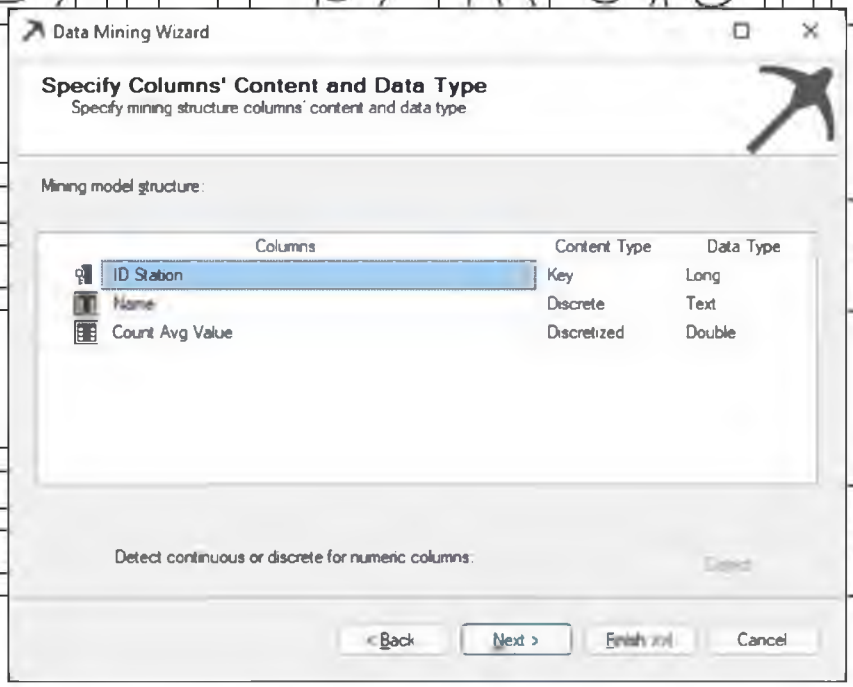

Рис. 27 Структура моделі

В результаті проведеного аналізу дані було розбито на 5(які можна об'єднати в 3) класи (рис. 28). У вікні є можливість обрати клас, відповідно до якого будуть відображатися дані. На рис.29 показано відображення даних, які відносяться до класу «Дуже низька забрудненість». Усі дані відображено за показником запиленості **PM10**

Рис. 28 Класи, які було отримано в результаті аналізу

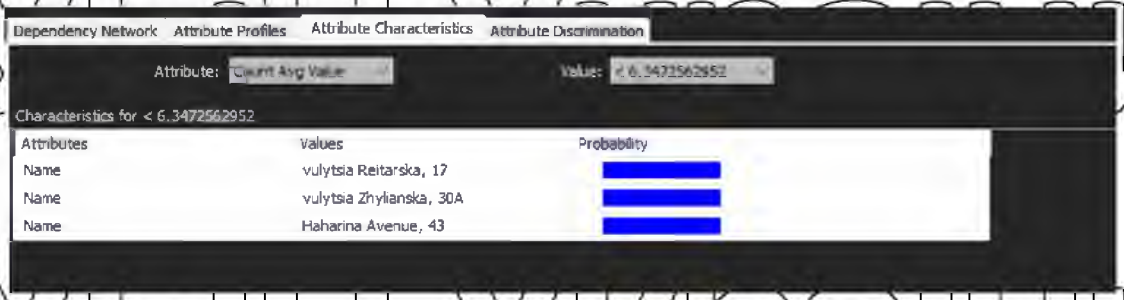

Рис. 29 Відображення результатів

3 Дослідження використання методу асоціативних правил

Афінітивний аналіз (affinity analysis) — один з розповсюджених методів Data Mining. Його назва походить від англійського слова affinity, що у перекладі означає «близькість», «подібність». Ціль даного методу — дослідження взаємного зв'язку між подіями, які відбуваються спільно. Різновидом афінітивного аналізу є аналіз ринкового кошика (market basket analysis), ціль якого - виявити асоціації між різними подіями, тобто знайти правила для кількісного опису взаємного зв'язку між двома або більше подіями. Такі правила називаються асоціативними правилами (association rules)[14].

Базовим поняттям у теорії асоціативних правил є транзакція — деяка множина подій, що відбуваються спільно. Типова транзакція - покупка клієнтом товару в супермаркеті. У переважній більшості випадків клієнт купує не один товар, а набір товарів, що називається ринковим кошиком. При цьому виникає питання: чи є покупка одного товару в кошику наслідком або причиною покупки іншого товару, тобто, чи пов'язані дані події? Цей зв'язок і встановлюють асоціативні правила. Наприклад, може бути виявлене асоціативне правило, котре стверджує, що клієнт, який купив молоко, з імовірністю 75 % купить і хліб.

Для проведення практичного завдання буде використовуватись середовище “Visual Studio SQL Server Data Tools”. Для роботи необхідно розгорнути сховище даних, визначити виміри, міри та побудувати гіперкуб.

На початку формування структури аналізу даних пропонується вибрати структуру даних, на якій будуватимуться метод пошуку асоціативних правил. Джерелом даних можна визначити існуючу базу даних чи сховище або використати існуючий гіперкуб. Далі у діалоговому вікні пропонується вибрати модель методу data mining. Оскільки у завданні розглядається пошук асоціативних правил (рис. 30).

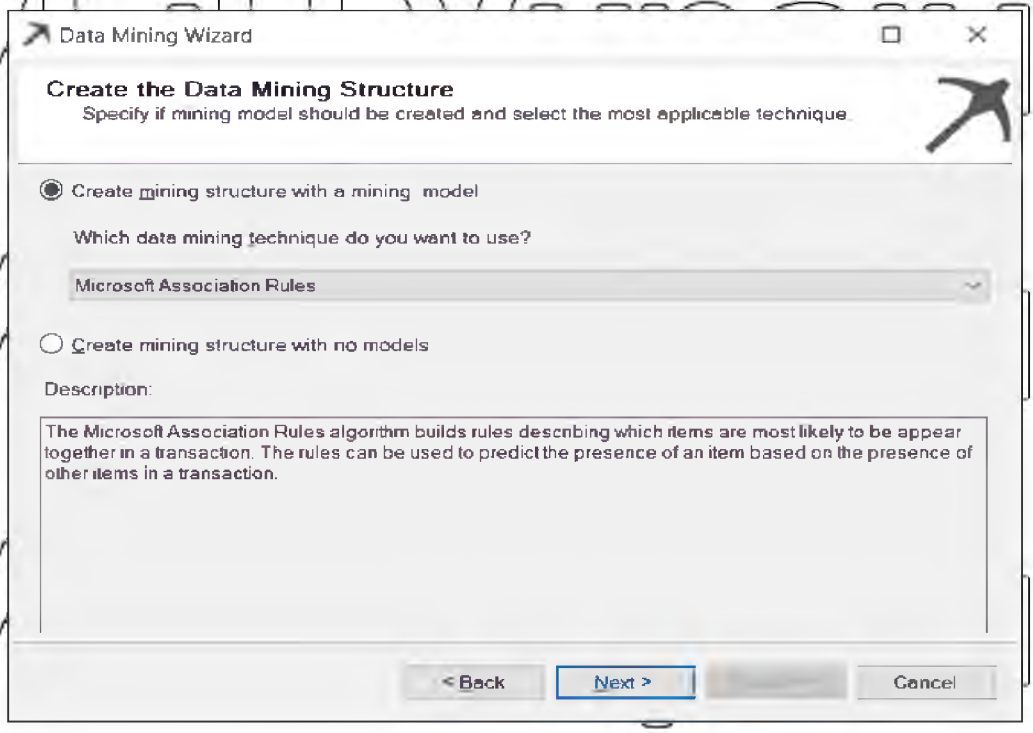

Рис. 30 Вибір моделі обробки даних.

Наступним кроком є визначення виміру (dimension), який буде використовуватись при пошуку правил. У даному вимірі для пошуку визначаємо вимір “StationDim” (рис. 31).

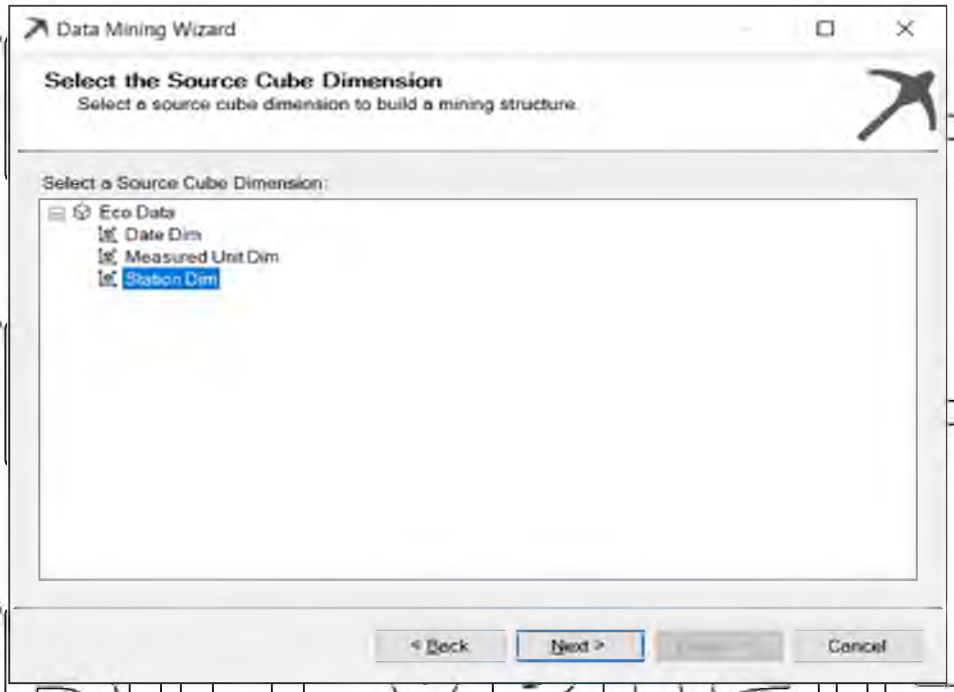

Рис. 31 Вибір виміру даних.

Для пошуку правил необхідно вказати ключ в структурі виміру. За замовчуванням встановлюється ключ виміру, проте можна вибрати будь-який інший атрибут. У даному випадку нас влаштовує обраний за замовчуванням ключ виміру “ID Station” (рис. 32).

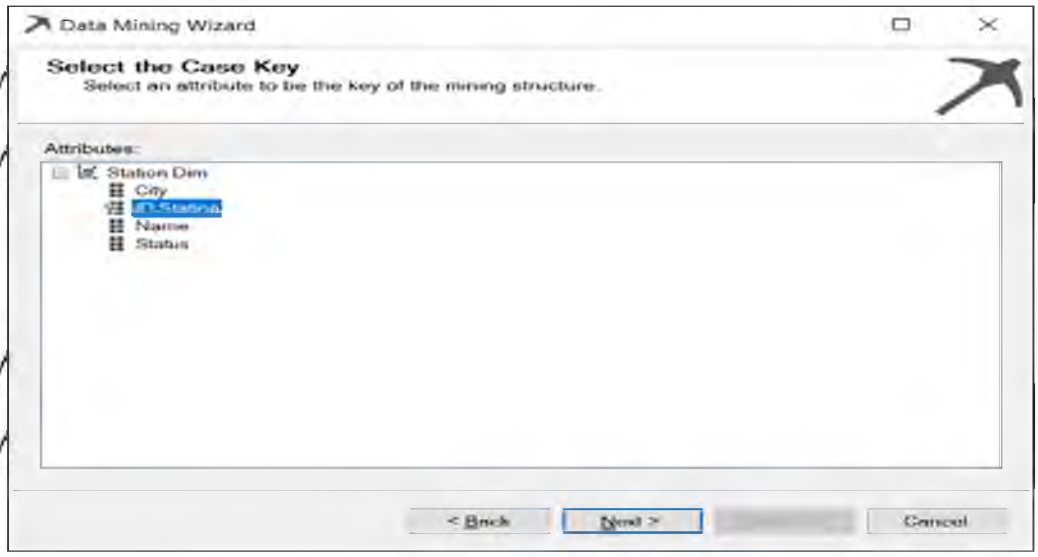

Рис. 32 Вікно вибору ключа виміру.

На рис. 33 показано обрані атрибути виміру, які цікавлять нас як вхідні поля до пошуку правил: city, name, avg value, min value, max value.

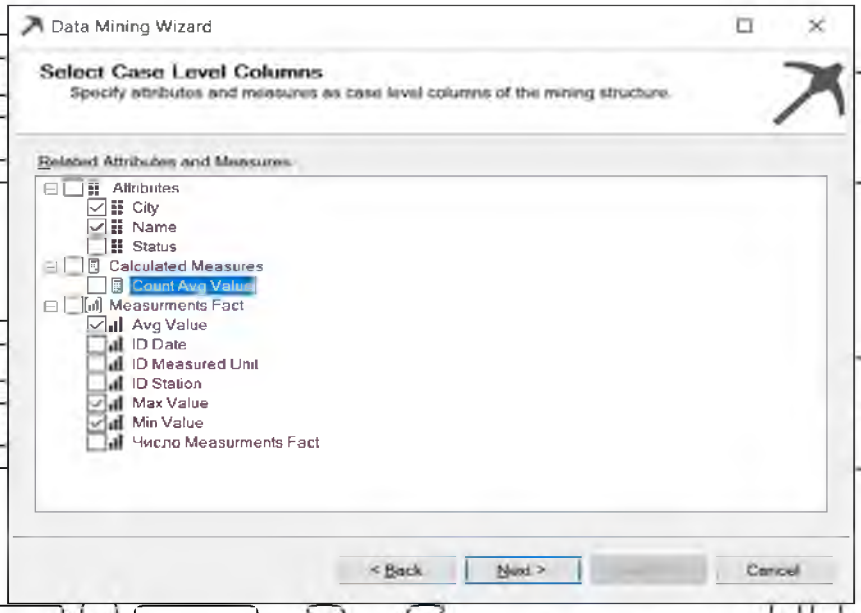

Рис. 33 Вікно вибору атрибутів і фактів

Наступним етапом є визначення передбачуваних полів (predictable) та вхідних полів (input). За замовчуванням усі поля вже є вхідними, проте полів для передбачення не обрано. Для прикладу в якості полів передбачення обираємо поля фактів Avg value, Max value, Min value (рис. 34).

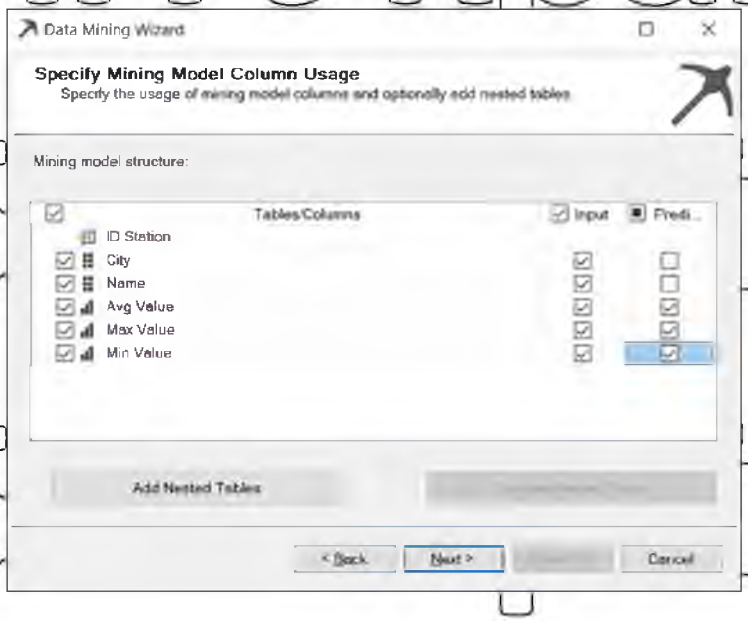

Рис. 34 Вікно визначення вхідних та передбачуваних полів (з визначеними полями).

Після цього на наступному вікні відображається інформація про поля в даній моделі, а саме який тип даних у поля та який тип поля у моделі (рис.22).

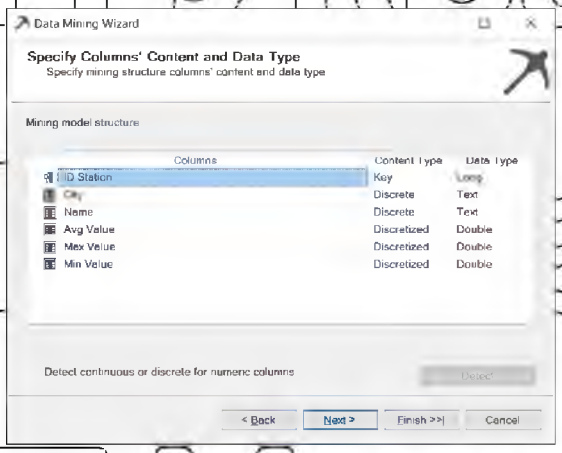

Рис. 35 Інформаційне вікно про тип полів у моделі

В наступних вікнах пропонується ввести вимоги для тренування моделі та налаштування тестування моделі. Можна покращити результат роботи моделі шляхом правильного налаштування вимог для тренування, є можливість визначити процент даних для тестування та максимальне число випадків в тестувальному наборі. У нашому випадку кількість даних у моделі незначна, тож залишаємо всі налаштування за замовчуванням (рис. 36).

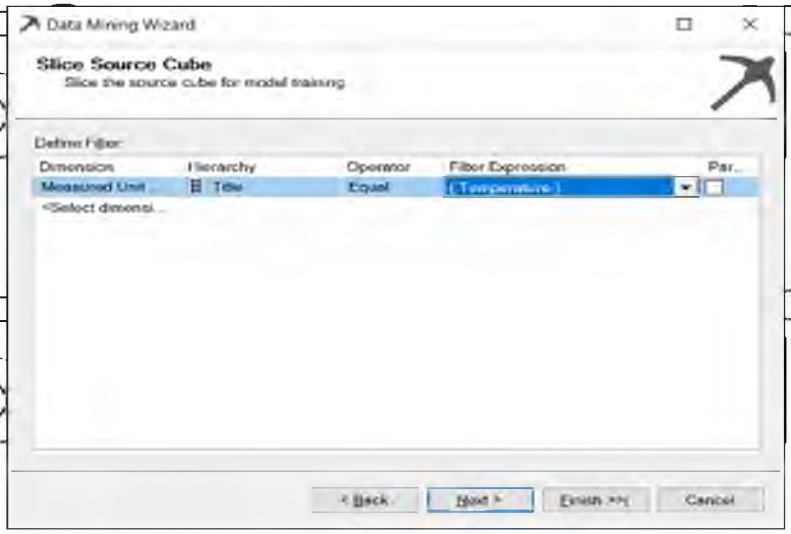

Рис. 36 Вікно налаштування вимог для тренування моделі та налаштування тестування моделі

Для перегляду результату аналізу даних необхідно запустити обробку моделі. У вкладці “Mining Model Viewer” після обробки моделі можемо побачити результат аналізу даних у вигляді правил, для кожного правила автоматично визначена ймовірність та важливість(рис. 37). Надалі аналітик повинен визначити правила, на яких можна побудувати гіпотезу, яку надалі може спростувати або підтвердити OLAP-аналіз.

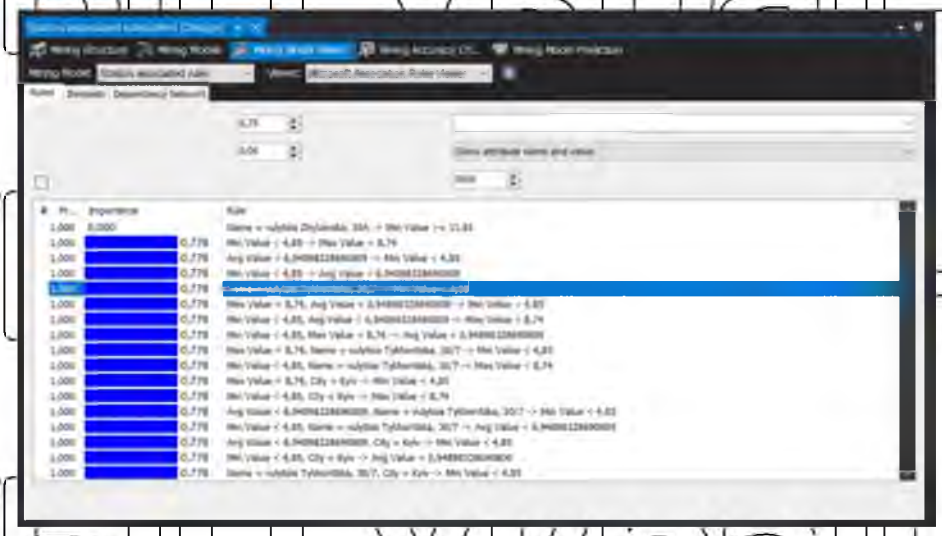

Рис. 37 Відображення знайдених правил під час обробки моделі

У вкладці “Mining accuracy chart” (рис.38) можна побачити графік, який зображує відношення правильних до загальних популяцій. Також модель зрівнюється з так званою “ідеальною моделлю”. У нашому випадку відсоток правильних 33, а в ідеальній моделі він дорівнює 33.

Рис.38 Графік порівняння моделі з ідеальною моделлю

На підвкладці “Itemsets” (рис. 39) користувач може побачити значення підтримки кожного набору елементів та на вкладці “Dependency network” (рис.40) – графічне відображення зв’язків між наборами елементів.

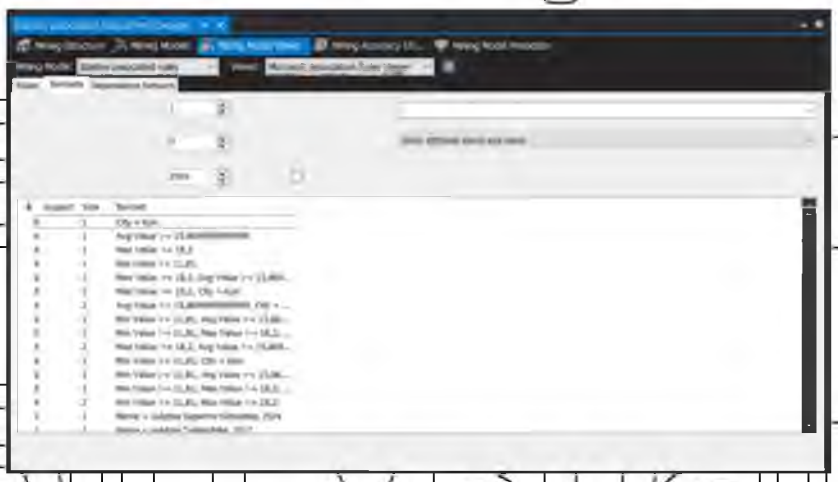

Рис. 39 Вікно відображення наборів елементів

Рис. 40 Вікно графічного відображення зв’язків між наборами елементів

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1 Тестування системи

Для тестування системи було створено набір юніт тестів і проведене візуальне тестування.

При неправильному вводу паролю виводиться помилка. Зверху зліва знаходиться Combo Box з можливістю вибору поведінки додатку, ввімкнення оффлайн чи онлайн логіки. Приклад вікна входу зображено на рисунку 20.

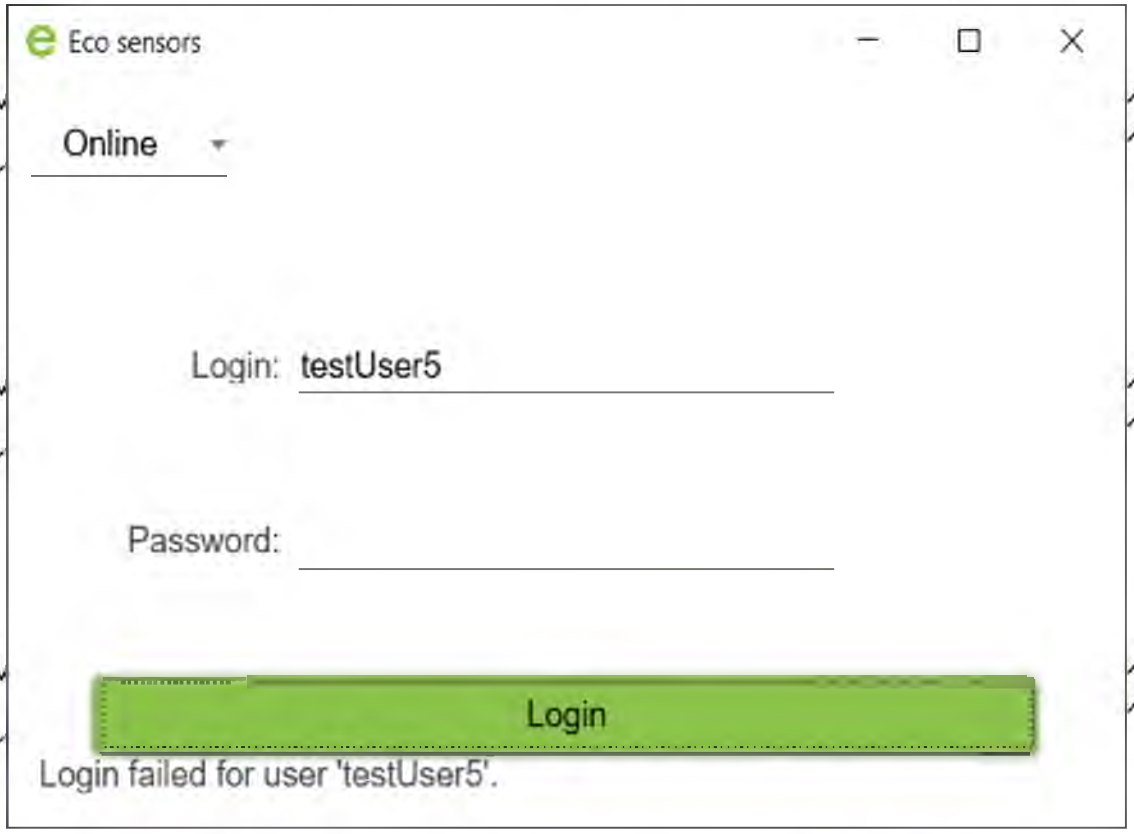

Рис. 20 Повідомлення про помилку входу

Демонстрація роботи основного вікна і вкладки Stations information наведено на рисунку 21. Зліва показується меню з лого додатку, і кнопками переходу до модулів програми, а саме Stations information, Comparsion, Map, Save to locatl db і About. Посередині видно регіон з основною логікою. Зверху є стрічка пошуку станцій по їх іменам. Кожна станція має свій контрол в якому відображаються основні дані, а саме ідентифікатор, ім’я станції, індикатор стану, останні

характеристики з мітками оптимального стану, кнопки перегляду даних окремої станції і завантаження звіту.

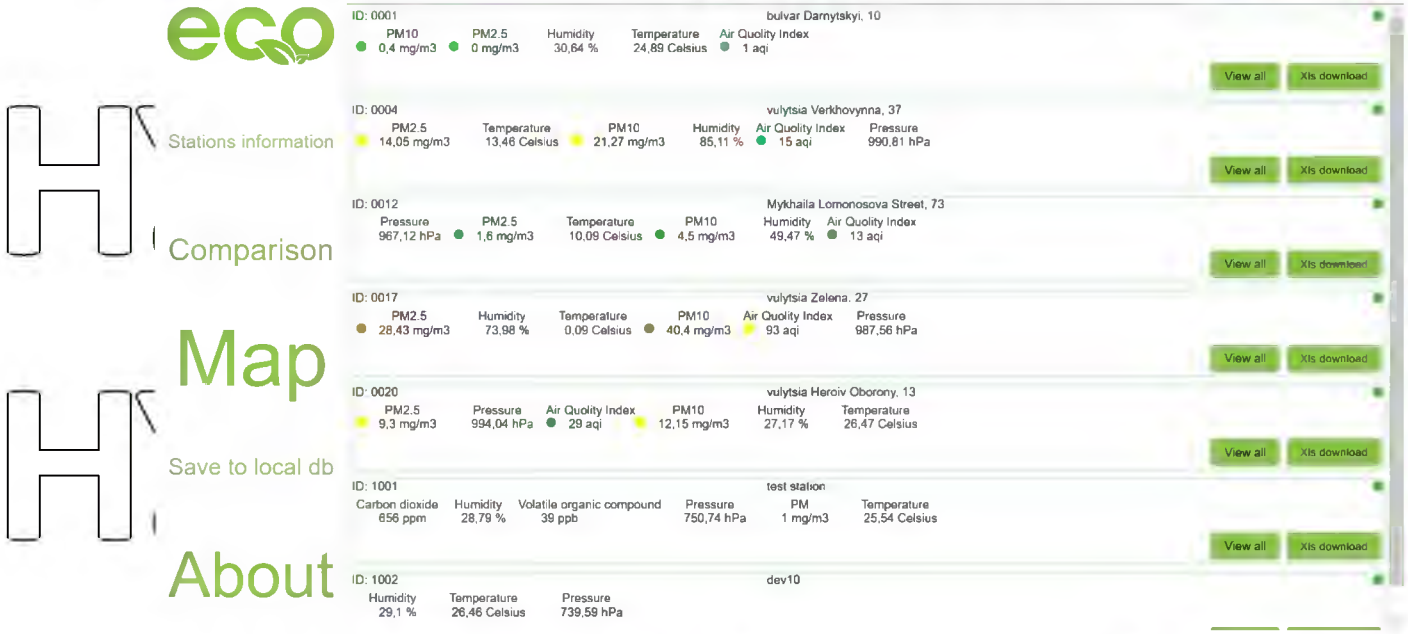

Рис. 21 Головне вікно і вкладка Stations information.

Демонстрація діалогового вікна з даними окремої станції і її графіками. У вікні зверху зліва видно мапу з місцезнаходженням станції. Посередині зверху останні дані зі станції з мітками оптимальних значень і справа зверху календар з можливістю вибору діапазону дат і моментального оновлення графіку. Нижче від місцезнаходження календаря знаходиться Combo Box з характеристиками які вимірює станція, при зміні значення якого також відбувається моментальне оновлення графіку. Вигляд діалогового вікна наведено на рисунку 22.

Рис. 22 – Діалогове вікно окремої станції

На рисунку 23 наведено попарне порівняння станцій відносно місцезнаходження. Посередині видно регіон з графіком. Справа календар з вибраними датами, комбо бокси з вибором станцій і значення, кнопка створення графіку.

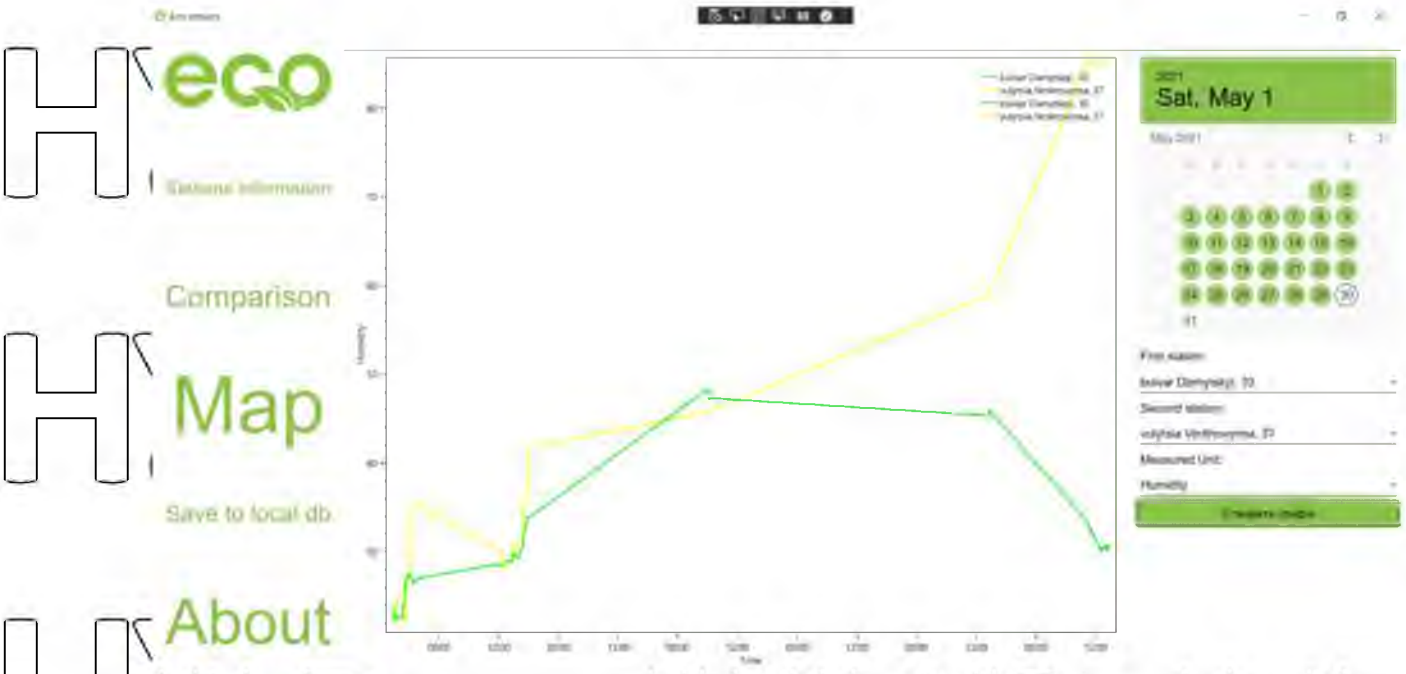

Рис. 23 Вкладка Comparsion

Наступна вкладка Map, для реалізації карти використовувались мапи від Bing. Інтерфейс зображено на рисунку 24.

Рис. 24 Модуль Map

На рисунку 25 зображене діалогове вікно About з кольорами оптимальних значень і їх значеннями.

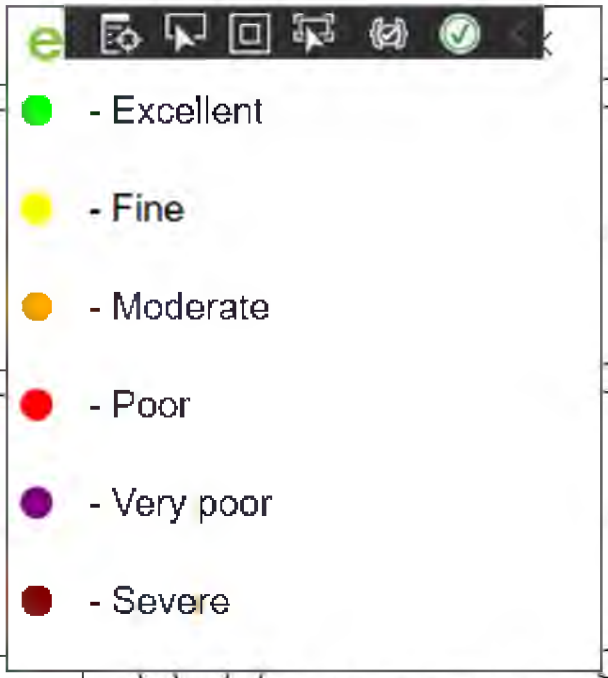

Рис. 25 Діалогове вікно About

4.2 Вимоги до апаратного забезпечення та програмного забезпечення

Апаратні вимоги:

- Процесор Intel Core 2 Duo - з тактовою частотою 2 гігагерц.
- Оперативна пам'ять не менше 4 гігабайт.
- Вільне місце на жорсткому місці – не менше 150 мегабайт.
- Кольоровий дисплей (роздільна здатність 1024x768).
- Маніпулятор типу миша та клавіатура.
- Windows 7 і вище.

4.3 Інсталяційний пакет

Всі файли які розпочинаються як EcoSensors.* і EcoSenosorsDAL відповідають за внутрішню логіку. Файл EPPlus.dll відповідає за бібліотеку EPPlus. MaterialDesignColors.dll і MeaterialDesignThemes.Wpf.dll відповідають за графічну оболонку. Microsoft.Data.*.dll є бібліотеками ADO.NET для SqlServer і Sqlite. Microsoft.EntityFramework.*.dll є бібліотеками EF. Oxyplot.dll і OxyPlot.WPF.dll це файли бібліотеки OxyPlot. Prism.dll це файли бібліотеки Prism. Unity.*.dll це файли IOC контейнера. Також було згенеровано архів з моделями регресії і класифікації. Всього файлів: 84. Розмір додатку: 85 мегабайт, розмір може варіюватися оскільки база даних може містити різну кількість записів.

ВИСНОВКИ

В результаті виконаної магістерської роботи було проаналізовано та змодельовано предметну область експертної системи дослідження стану атмосферного повітря. Було проаналізовано небезпечні речовини і їх вплив на людину. Створено архітектуру програмного забеспечення. Знайдено декілька схожих за функціональністю систем, проаналізовано їх сильні і слабкі сторони. Було написано чітке технічне завдання.

В ході реалізації системи було спроєктовано і розгорнуто сховище даних в хмарному середовищі, створено інтеграційний і аналізаційний проекти в Visual Studio. Створено OLAP куб. Було розглянуто організаційну структуру програмного забезпечення, обрано оптимальний інструментарій. Програмний продукт було поділено на логічні модулі і було створено декілька алгоритмів. Розглянуто Data Mining алгоритми, реалізовано алгоритми регресії і класифікації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Atmospheric (Air) Pollutants. [Електронний ресурс] – Режим доступу: https://www.encyclopedia.com/environment/encyclopedias-almanacs-transcripts-and-maps/atmospheric-air-pollutants
2. Смог у великих містах. [Електронний ресурс]. – Режим доступу: https://karbon-cns.com.ua/uk/smog-u-velikih-mistah.html
3. Вплив діоксид сірки на організм людини та НПС [Електронний ресурс] – Режим доступу: https://cutt.ly/0nf5dg7
4. Что такое PM10 и PM2.5? Чем могут быть опасны тонкодисперсные частицы? [Електронний ресурс] – Режим доступу: https://xn--90aifdm6al.xn--p1ai/blog/chto-takoe-pm10-pm25
5. Вплив автомобільного транспорту на навколишнє середовище в межах транспортної системи вінницької області. [Електронний ресурс] – Режим доступу: http://molodyvcheny.in.ua/files/journal/2018/8/50.pdf
6. Стан атмосферного повітря в Україні. [Електронний ресурс] – Режим доступу: https://www.chiva-planeta.org/ua/pvsus/sapu.html
7. Інтелектуальні технології Data Mining і Text Mining. [Електронний ресурс] – Режим доступу: https://pidru4niki.com/1623021247786/informatika/intelektualni_tehnologiyi_data_mining_text_mining
8. Бази і сховища даних – інформаційний фундамент бухгалтерського обліку та аналізу. [Електронний ресурс] – Режим доступу: http://dspace.wunu.edu.ua/bitstream/316497/4544/1/%D0%90%D0%B4%D0%B0%D0%BC%D0%B8%D0%BA_%D0%9A%D0%BE%D0%BB%20%D0%BC%D0%BE%D0%BD%D0%BE%D0%B3%D1%80_%D0%9F%D0%B0%D0%B7%D0%B8%20%D1%96%20%D1%81%D1%85/

%D0%BF%D0%B2%20%D0%B4%D0%B0%D0%BD%D0%B8%D1%85_16.pdf

9. Что такое приложение UWP? [Документація]. – Режим доступу: https://docs.microsoft.com/ru-ru/windows/uwp/get-started/universal-application-platform-guide
10. Что такое UNIVERSAL WINDOWS PLATFORM(UWP)? - Режим доступу: https://itvdn.com/ru/blog/article/uwp
11. Databinding with WPF. [Документація]. – Режим доступу: https://docs.microsoft.com/en-us/ef/ef6/fundamentals/databinding/wpf
12. Efficient Querying. [Документація]. – Режим доступу: https://docs.microsoft.com/en-us/ef/core/performance/efficient-querying
13. Resource limits for single databases using the DTU purchasing model - Azure SQL Database. [Документація]. – Режим доступу: https://docs.microsoft.com/en-us/azure/azure-sql/database/resource-limits-dtu-single-databases
14. Auditing for Azure SQL Database and Azure Synapse Analytics. [Документація]. – Режим доступу: https://docs.microsoft.com/en-us/azure/azure-sql/database/auditing-overview
15. Overview of WPF windows. [Документація]. Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/windows/?view=netdesktop-5.0
16. Паттерн MVVM. [Електронний ресурс]. Режим доступу: https://professorweb.ru/my/WPF/documents_WPF/level36/36_5.php
17. Будівельник. [Електронний ресурс]. Режим доступу: https://refactoring.guru/uk/design-patterns/builder
18. Команда. [Електронний ресурс]. Режим доступу: https://refactoring.guru/uk/design-patterns/command

19. Предметно-ориентированное проєктирование (DDD): структуризация сложных программных систем. [Книга]. Режим доступу: https://www.yakaboo.ua/ua/predmetno-orientirovannoe-proektirovanie-ddd-strukturizacija-slozhnyh-programmnyh-sistem.html

20. Pro C# 8 with .NET Core 3. Troelsen, Andrew, Japikse, Philip. [Книга]. Режим доступу: https://www.apress.com/gp/book/9781484257562#otherversion=9781484257555

21. Beginning Entity Framework Core 5. Vogel, Eric. [Книга]. Режим доступу: https://studfile.net/preview/4494701/page:2/

22. Azure Arc-Enabled Data Services Revealed. Weissman, Ben, Nocentino, Anthony. [Книга]. Режим доступу: https://www.apress.com/gp/book/9781484842670