КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЙ УПРАВЛІННЯ

Спеціальність 122 – «Комп'ютерні науки»
Освітня програма «Інформаційна аналітика та впливи»

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

АНАЛІТИКА ТА ПРОГНОЗУВАННЯ ЗАБРУДНЕННЯ НАВКОЛИЩНЬОГО СЕРЕДОВИЩА

Виконав:

студент групи ІАВ-21 Пасічний Богдан Віталійович

Керівник:

доктор технічних наук,
доцент кафедри технологій управління Хлевна Юлія Леонідівна

Попередній захист:

(Висновок: "До захисту в Державній екзаменаційній комісії")

Завідувач кафедри
технологій управління

(підпис) *(прізвище, ініціали)* *(дата)*

Київ-2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітньо-кваліфікаційний рівень Магістр
Спеціальність 122 - Комп'ютерні науки
Програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Морозов В.В.
"___" _____ 20___ року

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент Пасічний Богдан Віталійович

Група ІАВ-21

1. **Тема кваліфікаційної роботи** АНАЛІТИКА ТА ПРОГНОЗУВАННЯ ЗАБРУДНЕННЯ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Затверджена наказом по від "___" _____ 20__ р. № ___.

2. **Строк подання студентом готової роботи** - "___" _____ 20___ р.

3. **Цільова установка та вихідні дані до роботи** математичні моделі оцінки рівня забрудненості.

4. **Зміст роботи (перелік питань, що їх потрібно розробити)**
Проаналізував існуючі види забруднень навколишнього середовища, обґрунтував вибір одного з них та розглянув існуючі методи моніторингу та прогнозування його рівня;
Проаналізував існуючі підходи до прогнозування фізичних величин, зокрема рівнів забруднення, та обґрунтував вибір одного з методів для створення математичної моделі оцінки рівня забрудненості;
Розробив відповідну математичну модель та здійснити її реалізацію, провів тестування та дослідження отриманого продукту, оцінити його ефективність.

5. Календарний план виконання роботи:

№ з/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1.	Вивчення літературних джерел з предмету дослідження	15	5.09.22-1.11.2	5.11.22
2.	Збір і вивчення матеріалів досліджуваного підприємства	5	5.12.2-10.12.2	15.12.22
3.	Складання розгорнутого плану кваліфікаційної роботи	5	11.12.22-20.12.22	15.12.22
4.	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін.	10	20.12.22	21.12.22
5.	Підготовка розділу 1	15	10.01.22-25.01.22	20.01.22
6.	Підготовка розділу 2	20	25.01.18-07.02.18	20.02.22
7.	Підготовка розділу 3	15	07.03.22-22.04.22	22.04.22
8.	Оформлення кваліфікаційної роботи	5	01.05.22	16.05.22
9.	Передача кваліфікаційної роботи рецензенту для рецензування		17.05.22	17.05.22
10.	Передача кваліфікаційної роботи науковому керівникові		17.05.22	17.05.22
11.	Попередній захист кваліфікаційної роботи		18.05.22	18.05.22

Дата видачі завдання “____” _____ 20__ р.

Керівник роботи доктор технічних наук, доцент Хлевна Юлія Леонідівна

(підпис)

Завдання прийняв до виконання студент групи ІАВ-21 Пасічний Богдан Віталійович

(підпис)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ASP	Active Server Pages
CSS	Cascading Style Sheet
CSV	Comma Separated Values
GUI	Graphics User Interface
HTML	HyperText Markup Language
JSP	Java Server Pages
PC	Personal Computer
PHP	Personal Home Page, **P**HP **H**ypertext **P**reprocessor
SQL	Structured Queries Language
WWW	World Wide Web
XML	eXtensible Markup Language
БД	База даних
ІТ	Інформаційні технології Об’єктно-
ОО	орієнтований (-на, -не) Об’єктно-
ООП	орієнтоване програмування
ПЗ	Програмне забезпечення
ПК	Персональний комп’ютер
СУБД	Система управління базами даних

ВСТУП

Незважаючи на бурхливий розвиток технологій, людина як була кілька тисяч років тому назад, так і залишається досить ніжним створінням, яке для свого нормального існування потребує цілого комплексу безпечних, а краще, комфортних умов життя. В першу чергу, виставляються вимоги до (по мірі зменшенні нагальності, оперативності забезпечення):

– чистоти повітря, яке вона вдихає;

– чистоти води, яка використовується для пиття;

– біологічної чистоти рослин та тварин, які використовуються у їжу.

Менш очевидним, але безперечно необхідним є контроль наступних параметрів:

– хімічного складу ґрунтів, які використовуються для сільськогосподарських цілей;

– чистоти підземних, ґрунтових вод;

– радіаційної чистоти навколишнього середовища;

– глобального теплового забруднення місцевості, тощо.

Незважаючи на те, що усі вказані умови є важливими, все ж на першому місці знаходиться чистота повітря, для оцінки якої доцільно як розширювати існуючі мережі відповідних контролюючих пристроїв, так і розробляти нові. Сучасні датчики не тільки мають вимірюючий елемент, а й дозволяють використовувати автономний режим роботи, для якого необхідно вбудовувати у них комунікаційні підсистеми, розраховані на відповідні параметри передачі даних. Наприклад, сучасні датчики споживання електроенергії, встановлені у багатоквартирних будинках, можуть мати модуль Wi-Fi, за допомогою якого дані у повністю автоматичному режимі передаються на центральний блок, що встановлений зазвичай на даху будинку, який в свою чергу передає дані до дата-центру по GPRS з’єднанню. Описана система має усі ознаки «інтелектуальності»: автономний, повністю автоматичний режим роботи,

наявність постійного комунікаційного зв’язку між елементами системи з можливістю передачі змістовних даних, а також певних управлінських команд. Системи, що працюють за описаними принципами, називають інтелектуальними, розумними, або англомовним терміном SMART. Зважаючи на вищенаведене, можна стверджувати, що використання та (за відсутності підходящих варіантів) побудова систем оцінки чистоти повітря навколишнього середовища є **актуальною задачею** галузі сучасної інженерної науки та інформаційних технологій, зокрема.

Відповідно до описаної актуальності можна сформулювати **мету дослідження** – розробити систему аналізу та прогнозування забрудненості навколишнього середовища, що дозволить у випадку її реального впровадження більш адекватно реагувати (готуватися) до більших, чи менших рівнів забруднення.

Для досягнення такої мети необхідно вирішити певні **завдання дослідження**:

- проаналізувати існуючі види забруднень навколишнього середовища, обґрунтувати вибір одного з них та розглянути існуючі методи моніторингу та прогнозування його рівня;

- проаналізувати існуючі підходи до прогнозування фізичних величин, зокрема рівнів забруднення, та обґрунтувати вибір одного з методів для створення математичної моделі оцінки рівня забрудненості;

- розробити відповідну математичну модель та здійснити її реалізацію, провести тестування та дослідження отриманого продукту, оцінити його ефективність.

Об’єкт дослідження – процес оцінки рівня забрудненості навколишнього середовища.

Предмет дослідження – математичні моделі оцінки рівня забрудненості та їх реалізації.

Наукова новизна дослідження полягає у розробці математичної моделі для оцінки загального рівня забрудненості повітря даної місцевості на базі окремих показників забрудненості місцевості, що враховує різні ваги таких показників, та базується на використанні спеціалізованих інтелектуальних датчиків.

Практичне значення роботи полягає у створенні програмного продукту для оцінки інтегрального рівня забрудненості повітря на основі даних окремих датчиків, встановлених у даній місцевості.

Методи, застосовані в роботі: математичне моделювання, математична статистика (регресійний аналіз), технології програмування.

В перспективі математична модель, створена у роботі, може бути покращена шляхом залучення додаткових параметрів, окрім тих, що вже враховуються програмою.

1 АНАЛІЗ ПРОБЛЕМИ ЗАБРУДНЕННЯ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

1.1 Види забруднення навколишнього середовища та формалізація питань про їх чисельний опис

Безпека життя і здоров’я людини є однією з найвищих цінностей суспільства в сучасному світі. Нажаль, існує чимало факторів, що загрожують цим беззаперечним важливим складовим нашого життя. Цілий багатогранний комплекс утворюють різноманітні уражаючі фактори, пов’язані із забрудненням [1].

Забруднення є багатоплановим процесом і в галузі безпеки життєдіяльності та охорони праці виділяють наступні різні його типи:

а) хімічне – при неконтрольованому поширенні хімічної речовини у:

1) газоподібній фазі (забруднення повітря, яке може передаватися за рахунок інтенсивної у газах дифузії та конвективних потоків);

2) рідкій фазі (у 99,9% випадків несучою рідиною є вода; в цьому випадку також основну роль грають дифузія та конвекція, хоча не виключений варіант чисто механічного перенесення рідкого забруднювача з одного місця на інше за допомогою ще одного якогось носія);

3) твердому стані – за рахунок механічного перенесення (силою повітря, як при ураганах, води – як при повені, механічним перенесенням на якомусь носії).

б) фізичне – при розповсюдженні підвищених значень (у порівнянні із незбуреним фоном) різних фізичних величин. Для цього випадку можна виділити наступні підвиди:

1) радіаційне (виникає при поширенні у навколишньому середовищі радіоактивної речовини, що, відповідно, підлягає одному із трьох варіантів розпаду: a, b або g; надзвичайно небезпечне);

2) теплове (зважаючи, що більшість об'єктів оточуючого середовища мають суттєві теплоємності, цей вид забруднення досить довго не зникає навіть при повному зникненні джерел забруднення; крім того, нагріті маси повітря, води можуть переміщуватися на порівняно великі відстані);

3) шумове забруднення (його специфіка полягає у дуже малій інерційності: при зникненні джерела воно також миттєво пропадає, тому говорити у цьому випадку про стійке забруднення немає сенсу; крім того шумове забруднення не може поширюватися самовільно на невизначені відстані від джерела);

4) магнітне (також чітко прив'язане до джерела і пропадає при прибиранні самих магнітів, самовільно не поширюється);

5) електростатичне (при переміщенні статичних зарядів може змінювати свою конфігурацію, але зазвичай, у невеликих межах);

6) світлове (аналогічно неінерційне, самовільно не поширюється);

7) вібраційне (аналогічно).

в) біологічне забруднення – не тільки може поширюватися далеко від джерела, а й самовільно підвищувати власну інтенсивність, причому у досить широких межах (яка може характеризуватися, наприклад, масою живої речовини бактерій, або інших шкідливих організмів, на одиницю площі);

г) механічне забруднення великогабаритними відходами життєдіяльності людини (як-то при організації сміттєзвалищ). Характеризується тим, що самовільному поширенню не підлягає, може зростати лише контрольовано, «під наглядом» людини.

Аналіз особливостей усіх типів забруднень показує, що деякі з них підлягають самовільному поширенню, до того ж являються суттєвими

небезпечними та шкідливими факторами. Так, особливу увагу слід приділяти питанням хімічного, радіаційного та біологічного забруднення – як небезпечним факторам для здоров'я (та тим більше, життя людини). Із них найбільш швидкоплинними, «мінливими» є процеси забруднення повітря. Дійсно, механічне забруднення взагалі не може виникнути і рухатися само по собі, забруднення води може поширюватися лише повільно (як, наприклад, відбувається рух нафтових плям по поверхні води), так само як і забруднення грунту, а ось у випадку зараженого повітря ситуація є протилежною. В атмосфері майже завжди присутні досить суттєві процеси руху повітряних мас, в результаті чого рівень їх забрудненості у певній конкретній місцевості може різко змінюватися не тільки у різні дні, а й іноді – протягом однієї доби [2]. Важливу роль також грає середній за великі періоди часу рівень забрудненості повітря (наприклад, при виборі нового міста для переїзду, або при виборі розміщення майбутнього дитячого табору, і т.п.). Відповідно, бажаною є наявність інструменту, засобу, який би дозволяв проводити оцінку поточного та, особливо, майбутніх рівнів забрудненості на основі певної вхідної інформації.

Очевидно, що усі згадані процеси забруднення (як і майже будь-які процеси реального світу) можуть бути промодельовані на комп'ютері [3] – звичайно, за умови наявності відповідного програмного забезпечення. Результатами такого моделювання, наприклад, можуть бути масштаби забруднення при аваріях, катастрофах, чи інших нештатних ситуаціях (причому в динаміці, прив'язуючись до дат та, навіть, часу доби) – в першу чергу, питання територіального поширення забруднення. Такі відомості можуть допомогти спрогнозувати доцільну зону евакуації, зберігаючи людські життя та майно. Економічна обґрунтованість розробки відповідного програмного забезпечення є очевидною (як і його великий соціальний ефект). З точки зору математичної реалізації такої задачі, теоретично можливим є опис усіх трьох типів забруднення (хімічного, радіаційного та біологічного), однією

системою рівнянь (можливо, із зануленням деяких коефіцієнтів у різних випадках із трьох згаданих).

Однак і задача прогнозування «неаварійних» рівнів забрудненості також є вкрай актуальною [4], оскільки дозволить вносити корективи у повсякденне функціонування суспільства з метою мінімізації шкоди для здоров'я людей. Наприклад, при високих рівнях забрудненості повітря діти (організми яких, як відомо, є найбільш вразливими для поганих умов навколишнього середовища) можуть проводити більше часу у кондиціонованих приміщеннях дитячих садків та шкіл, а у моменти чистого повітря на вулиці, очевидно більший час краще проводити на відкритому просторі. При високих рівнях забрудненості повітря (які на сьогодні досить часто спостерігаються, наприклад, на території промислових районів Китаю – рис. 1.1) доцільно переносити (звичайно, за наявності такої можливості) масові заходи, роботи на відкритому повітрі, і т.п., [5].

Рис. 1.1 Фото смогу (не туману!) в Пекіні у грудні 2015 р., який часто спостерігається у Китаї в різних провінціях по кілька тижнів без перерви

Зважаючи на вищесказане, можна зробити висновок, що задача моделювання процесу поширення забруднень в цілому є надзвичайно актуальною для сучасної галузі ІТ, а задача моделювання саме забруднення повітря (як, наприклад, описано у [6]) взагалі має стратегічний характер і пов'язана зі здоров'ям усієї людської цивілізації в майбутньому, саме тому вона і буде вирішуватися у даному дослідженні.

1.2 Аналіз існуючих методів та програмних продуктів для аналітики та прогнозування забруднення навколишнього середовища

При створенні нових наукових та інженерних розробок довільного призначення, по-перше, слід проаналізувати наявні у відкритому доступі аналогічні рішення та підходи.

Дуже близьким до даної роботи по своїй суті видається ресурс [7], стартова сторінка якого наведена на рис. 1.2.

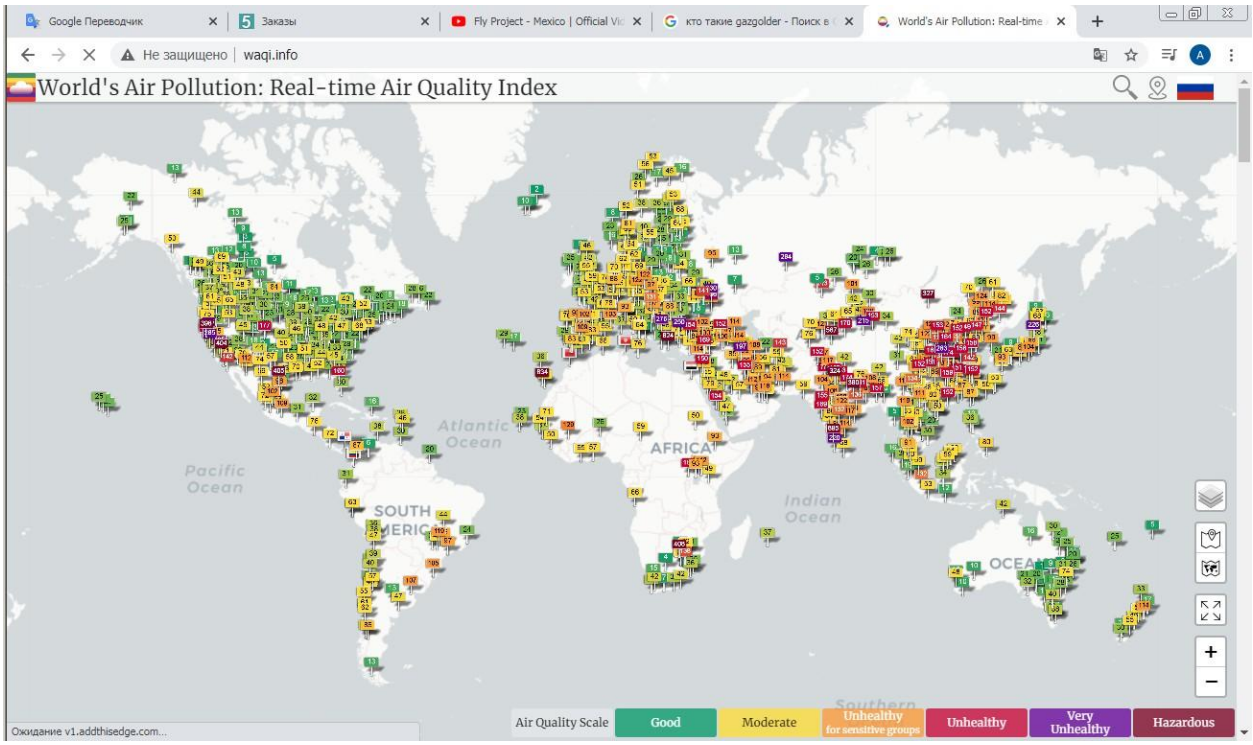


Рис. 1.2 Стартова сторінка ресурсу World's Air Pollution: Real-time Air Quality Index, присвяченого визначенню чистоти повітря

Ця система у реальному часі надає інформацію про параметри чистоти повітря по всьому світові (тобто це глобальна система), а саме у точках де встановлені спеціальні контролюючі станції, що популяризує даний сайт. На даному ресурсі є можливість передивлятися забруднення повітря в цілому, за значенням загального єдиного показника AQI (Air Quality Index), а також окремо по кільком важливим частинним показникам, серед яких є:

- $PM_{2,5}$ – характеризує ступінь забруднення повітря малими частинками пилу (бруду), а саме діаметром до 2,5 мкм;

- PM_{10} – характеризує ступінь забруднення повітря великими частинками, а саме діаметром до 10 мкм;

- O_3 – концентрація озону від промислових джерел;

- NO_2 – концентрація оксиду азоту;

- SO_2 – концентрація оксиду сірки;

- CO – концентрація чадного газу (моно оксиду вуглецю).

Згаданий вище загальний показник AQI обчислюється як зважена сума перелічених частинних показників:

$$AQI = \frac{\sum_{i=1}^{N}(w_i c_i)}{\sum_{i=1}^{N} w_i},$$

(1.1)

де c_i – концентрація i-тої забруднюючої речовини;

w_i – вага даної забруднюючої речовини, що обчислюються за формулою [8]:

$$w = \begin{cases} w^*, w^* > W_{\min}, w^* = \dfrac{c_{\min}}{c_{\max}}, \\ W_{\min}, w^* < W_{\min} \end{cases} \qquad (1.2)$$

де w^* - перше можливе значення ваги, яке рівне відношенню мінімальної концентрації $c_{\min}$ забруднюючої речовини до максимального значення $c_{\max}$, і це значення береться у якості ваги, якщо його значення більше порогу $W_{\min}$;

$W_{\min}$ – друге можливе значення ваги, яке є постійним, і це значення береться у якості ваги, якщо w^* менше, за $W_{\min}$ (тобто $W_{\min}$ – порогове значення, нижче за яке вага w не може бути рівною).

Формула (1.2) побудована так, що мінімальне значення ваги рівне $W_{\min}$, а саме $W_{\min}$ приймається рівним 0,5. Звичайно це є недоліком моделі, оскільки деякі частинні показники забрудненості повітря не є дуже суттєвими, але з іншого боку легко вимірюються, тому бажано було би включити їх у кінцевий результат. При підході (1.2), прийнятому при обчисленні AQI по (1.1), врахувати у малій мірі якийсь показник неможливо, тому доцільним є перегляд процедури вибору вагових коефіцієнтів (1.2). Взагалі ж ваговий коефіцієнт в (1.2) залежить від розкиду значень мінімальної $c_{\min}$ та максимальної $c_{\max}$ концентрацій забруднюючої речовини.

ГДК – традиційна для екології абревіатура, яка означає гранично допустиму концентрацію якоїсь речовини.

Прив’язка до порівняння з діоксидом сірки звичайно є недоліком моделі, оскільки кожна небезпечна (шкідлива) речовина має свою специфіку, дії по-різному, уражає різні органи і системи, тому доцільність порівняння з одним конкретним забруднювачем є як мінімум сумнівною.

Вказані приклади свідчать про те, що доцільним є розробка власної математичної моделі для виведення одного підсумкового показника забруднення на основі багатьох окремих частинних показників.

Також розглянемо апаратні засоби, на основі яких доцільно проводити зведення систем оцінки та прогнозування чистоти повітря. Для побудови системи оцінки забрудненості повітря, як уже зазначалося вище, доцільно використати не застарілі датчики, що потребують ручного (візуального) зчитування показів, а сучасні сенсори із розвинутими комунікаційними можливостями, що дозволяють автоматичну передачу власних показів до відповідних дата-центрів. Для цього можуть бути використані наступні рішення.

В першу чергу, поширеним та досить небезпечним забруднювачем є звичайний вуглекислий газ CO_2. Для визначення його рівня можна застосувати одне з численних рішень, присутніх на ринку, наприклад, від компанії Smart-MAC – рис. 1.3. Він дозволяє вимірювати концентрацію вуглекислого газу від 0 до 5000 ppm (кількість часток CO_2 на мільйон часток повітря), або у інших одиницях виміру – від 0 до 9129 мг/м^3. Похибка вимірювання концентрації складає ±50 ppm (або 91 мг/м^3), чого цілком достатньо для побутових потреб щодо вимірювання рівня забруднення повітря. Даний датчик працює від джерела живлення 5 В, що дозволяє використати компактні автономні акумуляторні джерела живлення, а, отже, встановлювати датчик у будь-яких місцях, незалежно від наявності мережі 220 В поблизу. Цьому також сприяє досить широкий температурний режим роботи (0…+50 °С) та відносної вологості повітря (0…90 %). Комунікаційні можливості даного датчика полягають у його локальному підключенні до аналогового входу розумного

лічильни ка імпульсів smart-MAC D105 – рис. 1.4. Цей пристрій, в свою чергу, має вості бездротової комунікації по мережі Wi-Fi, тобто утворює можли азом із самим датчиком дійсно розумний SMART-пристрій.

Рис. 1.3 Зовнішній вигляд датчику рівня вуглекислого газу компанії Smart- AC
M

Рис. 1.4 Зовнішній вигляд розумного лічильника імпульсів smart-MAC D105

Більш небезпечною, хімічно схожою із вуглекислим газом речовиною, є чадний газ CO, для аналізу наявності якого також можна використати один із численних пристроїв, доступних на ринку, наприклад сигналізатор Honeywell XC100D – рис. 1.5. Даний пристрій забезпечує вимірювання від 0 до 300 ppm, що у перекладі на масову концентрацію складає 349 мг/м^3. Точність вимірювання, що заявляється виробником, складає 1 ppm, що є достатнім для побутових пристроїв (слід сказати, що чадний газ є надзвичайно отруйною речовиною, тому навіть малі його концентрації до 10 ppm можуть негативно впливати на стан здоров’я людини при тривалій дії. Перевагою даного пристрою є використання вбудованої батареї, яка забезпечує термін безперервної роботи пристрою 10 років. Власне кажучи, він розрахований на однократне встановлення у верхніх частинах (адже чадний газ трохи легше повітря, тому концентрується у верхніх частинах приміщень) контрольованих приміщень та автономну роботу протягом вказаного періоду. Температурний діапазон (–10…+45 °C) та відносна допустима вологість повітря (25…95 %) сприяють широкому використанню також даного датчику.

Рис. 1.5 Зовнішній вигляд сигналізатора чадного газу CO Honeywell XC100D

Для визначення частинок пилу (показники PM2,5 та PM10) можна використати датчики на зразок PMS5003 – рис. 1.6, що має провідний інтерфейс підключення UART (RS485), але по ньому може бути під’єднаний або до мікроконтролеру з можливістю подальшої бездротової передачі, або безпосередньо до модуля бездротової передачі даних Wi-Fi. Діапазон вимірювання приладу складає 0-1000 мкг/м^3 (у ppm не переводиться, оскільки мова йде не про газ, а про тверду речовину). Точність його вимірювання складає 1 мкг/м^3. У якості елемента живлення використовується джерело на 4,5…5,5 В, що дозволяє використовувати комплект із 3 звичайних батарейок (чи акумуляторів) стандарту АА на 1,5 В. Діапазон робочих температур є навіть більшим, ніж у попередніх пристроїв, і складає –10…+60 °С, що дозволяє використовувати його не тільки у побутових приміщеннях, а й у більшості виробничих. Аналогічну властивість забезпечує і потрібна відносна вологість повітря, що для роботи даного датчика повинна знаходитися у межах 0…99 %.

Рис. 1.6 Зовнішній вигляд датчика пилу PMS5003

Також небезпечною речовиною, що активно виділяється промисловими технічними установками є озон. Виконуючи надзвичайно корисну функцію захисту нашої планети від космічного випромінювання та жорсткого ультрафіолету у верхніх шарах атмосфери, унизу він є досить небезпечним. Навіть невеликі концентрації озону можуть бути шкідливі для здоров'я. Ранні ознаки впливу підвищених концентрацій цього газу включають роздратування очей, сухість у роті, кашель, утруднення дихання, хрипи, задишку, біль в грудях і інші ознаки. Діти, а також люди, які страждають на астму, бронхіт, серцеві захворювання, особливо уразливі відносно дії цієї речовини.

Деякі виробники обладнання вказують, що озон знищує віруси, бактерії і цвіль. Це так, але проблема в тому, що його концентрація, при якій газ починає деактивувати шкідливі біологічні об'єкти, в 5-10 разів більше рівня ГДК. Це означає, що піддавати озонуванню можна виключно порожні (без людей) приміщення, після чого концентрація озона має бути зменшена в рази для перебування у них людей. Особливо небезпечним є тривалий вплив озону на організм людини, який може призводити до серйозних захворювань, включаючи хвороби легенів.

Це означає, що озон також слід контролювати у рамках створення комплексного показника чистоти повітря, для чого можна використати, наприклад, «розумний» датчик EnergoM-3001-O3 - рис. 1.7. Діапазон вимірювань його складає 0-100 ppm, тобто 0-116 мг/м^3. Точність вимірювання складає 1 ppm. Живлення цього пристрою реалізовано на не дуже зручному рівні 12-24 В, що вимагає використання спеціального блока живлення (перетворювача 220 В змінного струму у 12-24 В постійного струму), який поставляється разом із датчиком. Дані вимірювання датчик передає по інтерфейсу RS485, що дозволяє за необхідності підключати до одного кабелю

декілька пристроїв (до 16 штук). Робоча температура знаходиться у межах –30…+50 °С, а необхідна відносна вологість повітря складає 15…95 %.

Рис. 1.7 Датчик промислового озону EnergoM-3001-O3

Сірчистий газ, безперечно, є одним із найнебезпечніших домішок, що може бути присутнім у повітрі біля великих підприємств, особливо пов’язаних із хімічною промисловістю. Для його контролю може бути використаний наприклад, датчик "Дозор-С" – рис. 1.8. Він визначає небезпечну речовину у діапазоні 0-100 мг/м^3, що еквівалентно 0-38 ppm. Точність вимірювання складає 1 мг/м^3. Діапазон робочих температур складає –40…+50 °С, а необхідна відносна вологість повітря складає 10…95 %. Пристрій має розвинуті комунікаційні властивості і дозволяє підключатися до комп’ютерної техніки за інтерфейсом RS232 або RS485. Відповідно до документації виробника, особливості електричної схеми пристрою забезпечують можливість утворення лінії зв’язку довжиною до 1200 м. Живлення даного датчика може відбуватися від джерела постійного струму 24 В, а також від джерела змінного струму 220 В.

Рис. 1.8 Аналізатор газів «Дозор-С»: варіант для визначення сірчистого газу

Таким чином, можна констатувати, що на ринку присутні численні пристрої (причому із можливістю вибору: або придбати одразу більш дороге рішення із бездротовою передачею через Інтернет, або із більш дешевим дротовим підключенням до інших місцевих пристроїв, що вже мають можливості комунікації), серед яких є можливості вибору необхідних рішень, які у повній мірі відповідають вимогам кожного конкретного випадку проектування.

1.3 Вибір невирішених частин проблеми, що розглядається, та постановка задачі на дане дослідження

Таким чином, беручи до уваги вищенаведену інформацію, можна сформулювати наступну уточнену задачу дослідження.

Зважаючи на те, що забруднення повітря, яким дихає людина, є одним із найбільш небезпечних видів забруднення (через його швидке поширення, негайну дію на організм та неможливість виведення багатьох типів забруднювачів із легенів), у даній роботі необхідно провести зведення системи оцінки та прогнозування забруднення саме повітря. Існуючі математичні

моделі (такі, як AQI, ІЗА) мають певні недоліки, які обумовлюють необхідність розробки власного способу виведення інтегрального ступеня забрудненості повітря на основі частинних показників. Важливим елементом роботи є спирання на покази інтелектуальних сучасних датчиків окремих забруднюючих речовин, які розглядаються у даному дослідженні.

Створену математичну модель необхідно реалізувати програмно однією із мов програмування загального призначення (вибір засобів розробки має бути обґрунтований) та провести тестування отриманого продукту. В роботі також має бути наявною оцінка ефективності розробленого рішення.

Важливою вимогою, що висувається для проектованого додатку є простота інтерфейсу користувача, що важливо, оскільки користуватися ним будуть непрофесіонали у галузі інформаційних технологій, а прості користувачі, що бажають дізнатися рівень забруднення повітря (відповідно, при розробці інтерфейсу слід орієнтуватися на самого мінімально кваліфікованого користувача).

1.4 Висновки по розділу

Таким чином, у даному розділі розглянуто основні аспекти проблеми оцінки забруднення навколишнього середовища. В першу чергу описані різні типи забруднень та встановлено, що саме забруднення повітря є найнебезпечнішим, оскільки людина постійно і безперервно потребує повітря (отже, якщо воно заражене, то дія на організм починається негайно), і, крім того, забруднення у ньому можуть поширюватися із дуже великою швидкістю (на відміну, наприклад, від водних або ґрунтових). Наступним кроком став розгляд існуючих показників загального рівня забруднення повітря, до яких необхідно переходити, оскільки різних забруднюючих речовин існує багато, а пересічну людину цікавить ступінь небезпеки повітря в цілому, взагалі. Вияснено, що існуючі показники (AQI, ІЗА) мають певні недоліки, пов’язані

із складністю вибору вагових коефіцієнтів моделей для окремих забруднювачів. Серед таких забруднювачів обрано основні та розглянуто прилади для вимірювання їх рівнів, що одночасно мають розвинуті інформаційно-комунікаційні можливості, необхідні для передачі їх показів через мережні інтерфейси. На основі матеріалу даного розділу можна переходити до наступних досліджень, пов’язаних із проектуванням відповідної системи.

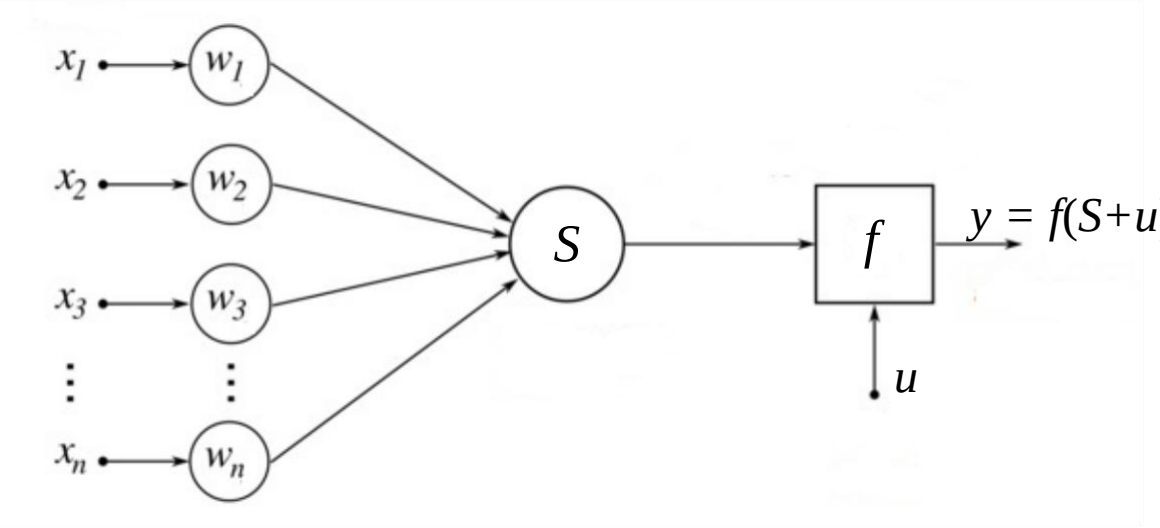
2 РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ ПО СТВОРЕННЮ СИСТЕМИ АНАЛІТИКИ ТА ПРОГНОЗУВАННЯ ЗАБРУДНЕННЯ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

2.1 Обґрунтування вибору базового методу аналізу та прогнозування забруднень

Задача оцінки рівня забрудненості повітря може бути віднесена до класу задач прогнозування, адже мова йде про вирахування значень забруднення, що на часовій шкалі відносяться до майбутньої найближчої перспективи. Саме прогнозування можна проводити як на основі попередніх значень забрудненості, так і з урахуванням наявних та, можливо, прогнозованих значень інших показників, що впливають на забрудненість повітря. Такі задачі часто вирішують методами математичної статистики, а також штучного інтелекту, тому розглянемо їх докладніше.

Іншим класом таких систем, які в принципі можуть бути використані для завдань прогнозування, є інтелектуальні рішення на основі нейронних мереж, які за останні роки досягли найбільшої відносної частки прогнозування серед усіх наукових ресурсів. Така ситуація багато в чому пояснюється певним «бумом», який спостерігається у сфері штучного інтелекту та суміжних областях навколо концепції нейронних мереж. Цей об’єкт широко використовується для задач розпізнавання, класифікації, апроксимації в [9], а отже, і для передбачення.

Як ми всі знаємо, штучна нейронна мережа (ШНМ) — це система, яка приймає вектор вхідних величин і виробляє вектор вихідних величин за складними (нелінійними) внутрішніми законами, що є більш-менш задовільним рішенням проблемна програма.



Елементарною складовою ШНМ є один нейрон – рис. 2.1.

Це об’єкт, що має певну кількість входів (на рисунку – *n* входів), які називають дендритами та один вихід, що називається аксоном [9].

Рис. 2.1 Структура штучного нейрону

Вхідні сигнали x1, x2,…, xn надходять на входи нейронів, які є виходами попередніх нейронів, розташованих раніше в сигнальному процесі. Кожен вхідний сигнал xi множиться на деяке число wi, яке називається вагою (або синаптичною вагою) відповідного входу. Усі зважені таким чином вхідні сигнали надходять на суматор, який створює зважену суму S у вигляді:

$$S = \sum_{i=1}^{n} w_i x_i . \qquad (2.1)$$

До цієї зваженої суми вхідних сигналів може додаватися зсув або зміщення *u* (у поширеному частинному випадку зміщення відсутнє, тобто $u = 0$) і від цієї суми виробляється певна функція *f*, що називається активаційною; так і формується вихідний сигнал нейрону *y*, що передається далі по нейронній мережі:

$$y = f(S + u). \qquad (2.2)$$

Відповідно комбінуючи (2.1) та (2.2) маємо функцію, що здійснює штучний нейрон [10]:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + u\right). \qquad (2.3)$$

При аналізі такого способу завдання функції, що здійснює нейрон, як залежність (2.3), бачимо, що важливу роль для кожного нейрона відіграють три речі (особливо перші дві):

– набір конкретних значень ваг w_i, що описують кожен вхід нейрона;

– вид активаційної функції f даного нейрона;

– наявність зсуву або порогу активації u (який може бути як додатним, так і від'ємним).

Враховуючи, що типова ШНМ складається щонайменше з кількох нейронів (частіше – з десятків або навіть сотень нейронів), і кожен нейрон має багато вхідних даних, проблема вибору ваг wi за своєю складністю стає дуже серйозною [11]. Насправді, тривалий процес встановлення конкретних «правильних» значень для цих ваг називається процесом навчання нейронної мережі. Існують різні алгоритми навчання нейронних мереж, які можна розділити на дві групи: «з учителем» і «без викладача». Загалом навчання нейронних мереж є настільки складним процесом, що різні його варіанти необхідно досліджувати в одному великому масиві інформації, який не може бути розглянутий у цій роботі.

Тип функції активації f, очевидно, також є дуже важливим моментом функції нейрона, хоча й значно меншою мірою, ніж вибір синаптичних ваг [12]. на рис. 2.2 показано найпоширеніші типи функцій активації, які найчастіше використовуються на практиці при роботі з нейронними мережами.

З точки зору обчислювальної складності більш кращим є використання порогових функцій, що фактично являють собою константну залежність (або хоча б лінійних, але не сигмоїдних).

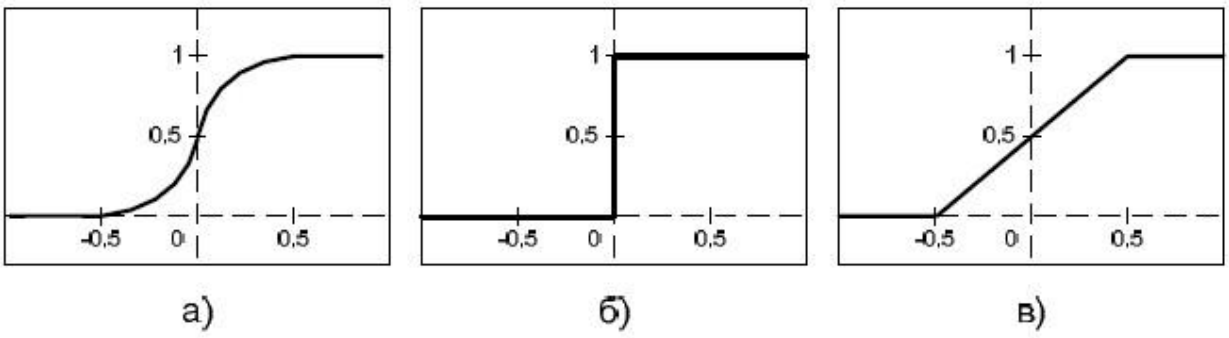


Рис. 2.2 Найбільш поширені типи функцій активації: а – сигмоїдна, б – порогова, в – лінійна

Крім властивостей окремих нейронів, при побудові всієї нейронної мережі важливо, як вони з'єднані (а також їх загальна кількість, як нейрони організовані в групи, шари тощо).

Найпростішим і найпоширенішим способом з'єднання нейронів у конкретній ШНМ є персептрон, оскільки ШНМ з такою внутрішньою структурою можуть вирішувати широкий спектр завдань [12].

Слід відмітити, що нейронні мережі мають недолік, який полягає у не повній передбачуваності результатів їх висновків. Існують повідомлення [13] про різку зміну висновку нейронної мережі при дуже незначних змінах вхідних величин. Зважаючи на це, використовувати нейронну мережу у даній роботі не будемо.

Дуже популярним методом отримання остаточних рішень у складних задачах моделювання [14], особливо прогнозування в області навколишнього середовища, є використання нечіткої логіки, яка є дуже перспективним механізмом і також широко використовується в інтелектуальному аналізі даних (IAD). Перевага таких систем полягає в тому, що вони можуть обробляти нечислові, розпливчасті або навіть якісні вирази та лінгвістичні дані. Процес постановки вхідної інформації, який використовує лінгвістичні змінні, що дозволяє вам зіставляти елементи, описані набором тексту, а потім використовувати набір простих і зрозумілих правил нечіткого продукту, щоб

прийняти остаточне рішення в нечіткій, нечіткій формі. Після застосування засобу для видалення накипу отримуємо конкретне значення прогнозованого значення. Основна перевага цього підходу полягає в тому, що він не вимагає створення складних математичних моделей (як правило, реалізованих на основі багатовимірного регресійного аналізу, інших статистичних методів, які потребують нетривіальних математичних основ), а також використовує прості правила, які можна розробити на основі про систематичний логічний аналіз. або процес процесу) будь-яка здорова людина, яка має принаймні досвід спостереження або використання системи. Правила мають структуру на зразок «якщо концентрація вуглекислого газу в кімнаті є високою, то потужність вентилятора фільтру є малою», «якщо концентрація технічного озону є низькою, то ступінь використання високовольтного агрегату є високою», і т.п.

Зважаючи на широку поширеність систем нечіткого виводу, розглянемо докладніше весь процес їх використання (зокрема, для задач прогнозування, як у даній роботі) [15]. Він полягає у циклічному виконанні системи наступних кроків всього процесу, показаних на рис. 2.3:

– фазифікація вхідних даних, тобто надання ним нечіткості;

– агрегування різних підумов для тих правил, що мають складені умови (антецеденти);

– активізація окремих під висновків, які можуть зачіпати одну й ту саму вихідну змінну;

– акумулювання висновків (потрібно, якщо одна й та сама вихідна змінна входить до консеквентів різних правил нечітких продукцій, що були задіяні на попередньому етапі активізації);

– дефазифікація вихідних величин.

Рис. 2.3 UML-діаграма діяльності процесу нечіткого виводу

Нечіткі системи мають недолік складності контролю кінцевого результату моделювання традиційними математичними методами, а сам цей результат сильно залежить від вибору функцій належності (який, взагалі

кажучи, є вільним). Через ці причини нечіткі множини у даній роботі застосовувати не будемо.

Окрім розглянутих методів із галузі штучного інтелекту, які з'явилися порівняно нещодавно (у другій половині XX ст.), також для описаних вище задач можна застосовувати методи математичної статистики, розроблені дещо раніше, а саме більше 100 років тому (початок XX ст.). Серед усього блоку відомостей про математичну статистику для задач прогнозування особливо виділяється розділ «Аналіз часових рядів».

Аналіз часових рядів - сукупність математико-статистичних методів аналізу, призначених для виявлення структури часових рядів і для їх прогнозування. Сюди відносяться, зокрема, методи регресійного аналізу.

Виявлення структури часового ряду необхідно для того, щоб побудувати математичну модель того явища, яке є джерелом аналізованого часового ряду. Прогноз майбутніх значень часового ряду використовується для ефективного прийняття рішень. Прогнозування часових рядів має широке застосування: від прогнозу температури до передбачення значення біржових котирувань і валютних пар. Очевидно, що цей підхід можна використовувати і для прогнозування забруднення повітря.

Часові ряди складаються з двох елементів:

- періоду часу, за який або за станом на який наводяться числові значення;

- числових значень того чи іншого показника, званих рівнями ряду.

Часові ряди класифікуються за такими ознаками:

а) за формою подання рівнів:

- ряди абсолютних показників;

- відносних показників;

- середніх величин.

б) за кількістю показників, для яких визначаються рівні в кожен момент часу: одномірні і багатовимірні тимчасові ряди;

в) за характером тимчасового параметра: моментні і інтервальні часові ряди. У моментних часових рядах рівні характеризують значення показника станом на певні моменти часу. В інтервальних рядах рівні характеризують значення показника за певні періоди часу. Важлива особливість інтервальних часових рядів абсолютних величин полягає в можливості підсумовування їх рівнів. Окремі ж рівні моментного ряду абсолютних величин містять елементи повторного рахунку. Це робить безглуздим підсумовування рівнів моментних рядів;

г) по відстані між датами і інтервалами часу виділяють рівновіддалені - коли дати реєстрації або закінчення періодів йдуть один за одним з рівними інтервалами і неповні (не рівновіддалені) – коли принцип рівних інтервалів не витримується;

д) за наявністю пропущених значень: повні і неповні часові ряди;

е) часові ряди бувають детермінованими і випадковими: перші отримують на основі значень деякої невипадковою функції (ряд послідовних даних про кількість днів у місяцях); другі є результат реалізації деякої випадкової величини.

є) залежно від наявності основної тенденції виділяють стаціонарні ряди, в яких середнє значення і дисперсія постійні, і нестаціонарні, які містять основну тенденцію розвитку.

Часові ряди, як правило, виникають в результаті вимірювання деякого показника. Це можуть бути як показники (характеристики) технічних систем, так і показники природних, соціальних, економічних та інших систем (наприклад, погодні дані). Типовим прикладом тимчасового ряду можна назвати біржовий курс, при аналізі якого намагаються визначити основний напрям розвитку (тенденцію або тренд).

Найбільш поширеним методом аналізу часових рядів є регресійний аналіз. При його застосуванні метою є встановлення виду та параметрів функції, яка відповідає законові зміни однієї вихідної контрольованої

величини від одного (одновимірний регресійний аналіз та регресія) або декількох (багатовимірний регресійний аналіз та багатовимірна регресія) вхідних параметрів. При цьому у більшості випадків залежність, що встановлюється, обирається лінійною (отже, визначенню підлягають коефіцієнти такої лінійної залежності), хоча в окремих випадках (коли відомо про суттєву не лінійність моделі) застосовують і вищі степені апроксимуючого поліному (квадратична регресія, і т.д.). У дуже рідких випадках використовують якісь інші функціональні залежності, окрім поліномів. Конкретний вид регресійної залежності буде обґрунтований у наступному підрозділі.

Окрім наведених вище, для аналізу даних щодо забруднення можна було би епізодично застосовувати і інші методи аналізу даних, наприклад, кластерний аналіз, кореляційний аналіз, дисперсійний аналіз, тощо.

Розглядаючи суть кластерного аналізу, в першу чергу, слід зазначити, що існують дві споріднені задачі розбиття сукупності об’єктів на класи: класифікації та кластеризації. Відмінність полягає в тому, що при класифікації проводиться циклічний перебір всіх наявних об’єктів і віднесення кожного з них до одного з заздалегідь відомих (наперед заданих, ще до розгляду об’єктів даної сукупності) класів. При кластеризації навпаки: ніяких заздалегідь визначених класів не існує, а сама суть завдання якраз і полягає в їх (класів) оптимальному виділення (найбільш ефективному для даної предметної області). При такій постановці іноді число класів, які потрібно виділити, задано наперед, іноді – ні, що визначає алгоритмічні особливості даного процесу.

Ускладнюючим аспектом задачі кластеризації (з точки зору її алгоритмічної та обчислювальної складності) є те, що одержувані розбиття на класи можуть істотно змінюватися для однієї і тієї ж сукупності об’єктів, при використанні різних наборів даних, що описують кожен об’єкт. Додавання (або вилучення) навіть однієї властивості з набору характеристик, за

допомогою яких можна описувати об’єкти сукупності, може призводити до повного їх перегруповування у порівнянні з початковим розбиттям (і навіть до зміни доцільного числа класів *N*). Отже, вибір структури набору вхідних даних для кластеризації об’єктів є одним з ключових етапів всього процесу.

Завдання кластеризації вирішується на наборах даних про однотипні об’єкти навколишнього світу (реального або цифрового). Наприклад, це може бути:

- опис особливостей кількох тисяч покупок, здійснених в певному Інтернет-магазині (описується в який час користувач зайшов на сайт, скільки часу всього провів на сайті, скільки переглянув сторінок, які групи товарів переглядав і в якій послідовності, і, нарешті, які в підсумку зробив покупки);

- опис десятків таких небезпечних природних явищ, як торнадо (де сформувалися, скільки тривали, максимальна сила вітру, атмосферний тиск на кордоні і в епіцентрі, і т.п.);

- набір числових характеристик певного технічного процесу (наприклад, експлуатації та виходу з ладу певної електронної плати, які можна описати часом безперебійної експлуатації, середньою температурою експлуатації, якістю напруги і частоти живлять ланцюгів, і т.д.).

В усіх наведених прикладах, які при їх попарному порівнянні здаються досить далекими один від одного, присутні такі особливості:

- явища, які описуються, є досить складними, у всякому разі відсутні їх прості загальноприйняті моделі;

- існує необхідність поділу всієї сукупності на кілька класів (кластеризація), причому віднесення кожного наступного явища до того чи іншого класу (класифікація) дозволить найбільш ефективним чином взаємодіяти з цим явищем, отримуючи максимальну вигоду (в разі, якщо явище несе позитивний сенс) або мінімізувати збитки (якщо явище шкідливе);

- існує набір даних (дата-сет), що описує велику кількість (очевидно, це число обмежується циклічністю, повторюваністю процесу, але чим більше -

тим краще) реально спостережуваних раніше таких явищ. Зазвичай в такий набір включається хоча б кілька параметрів явища, тобто не одна характеристика, і майже ніколи - дві. Зазвичай число параметрів починається мінімум від 3 і вище, а взагалі ж, чим більше параметрів процесу розглядається, тим ефективніше можна побудувати систему класів, на які описуються явища будуть розділятися.

Так, наприклад, чим більше параметрів профілю в соціальній мережі буде приведено в дата-сеті, тим вище ймовірність побудови ефективної системи класифікації користувачів соцмереж на базі попередньої кластеризації зазначеної сукупності.

Отже, беручи до уваги усі особливості процесу кластеризації, можна сказати, що види та варіанти забруднення навколишнього світу можливо і було би доцільно кластеризувати за деякими розділами показників на окремі групи, виділяючись в певні класи. Однак, для цілей саме даної роботи ця властивість є занадто специфічною і використовувати її у роботі не будемо.

Ще одним популярним видом статистичних досліджень є кореляційний аналіз, що дозволяє встановлювати наявність зв'язку між двома величинами, які міняються в часі (тобто між двома функціональними залежностями). Найбільш просто це виконується для послідовностей двох величин, заданих дискретними наборами чисел. При цьому обрахуванню підлягає коефіцієнт кореляції *R*, за величиною якого і можна встановити наявність зв'язку двох величин: $R = 1$ означає, що дві величини утворюють детерміновану функціональну залежність, а $R = 0$ свідчить про те, що жодного зв'язку між двома величинами не існує. Формула для обчислення має вид:

$$R = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 (y_i - \bar{y})^2}}.$$

У даній роботі можна було би застосувати кореляційний аналіз, якби був у наявності певний типовий паттерн процесу забруднення, з яким можна було

би порівнювати поточну динаміку цього процесу, а оскільки такого прикладу в наявності немає, то цей метод аналізу даних не застосовуватимемо.

Існують і інші варіанти статистичних досліджень, які можна застосувати для задач аналізу забруднень (дисперсійний аналіз, факторний аналіз, дискримінантний аналіз, тощо), однак їхні дослідження виходять за рамки даної роботи через її обмежений обсяг та наявність методів, що дозволяють добре вирішити поставлені задачі.

2.2 Розробка алгоритмів роботи проектованої системи аналізу та прогнозування

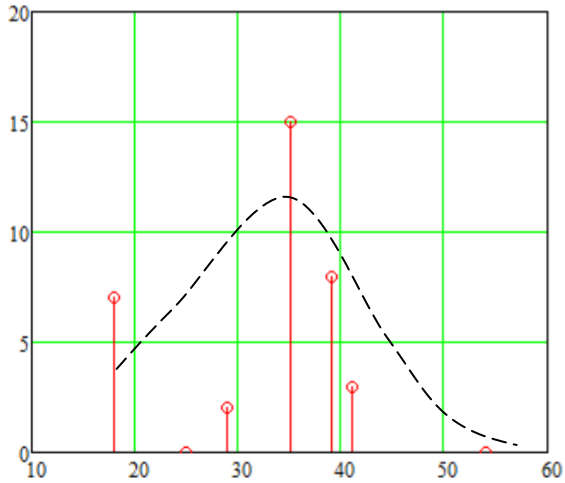
Отже, у якості базового методу прогнозування обрано регресійний аналіз, для практичного застосування якого слід розробити алгоритмічне підґрунтя. Для цього слід по-перше, детальніше розглянути суть самого цього методу.

Задача відтворення функціональної залежності між однією величиною (аргументом) та іншою (функцією), так само як і між кількома аргументами та функцією, є типовою для розділу математичної статистики, що називається регресійним аналізом. Сам вид цієї функціональної залежності називається регресією (або рівнянням регресії). Іноді назву уточнюють словом «одновимірна», якщо треба підкреслити, що вихідна величина залежить тільки від однієї вхідної, або «багатовимірна», якщо вхідних величин декілька.

Дійсно, в загальному випадку предметом пошуку у регресійному аналізі може бути не тільки функція однієї змінної. З математичного аналізу добре відомі функції багатьох змінних, які також широко використовуються у регресійному аналізі, якщо необхідно встановити залежність однієї вихідної величини від декількох вхідних аргументів. Така задача називається множинною регресією (або встановленням рівняння множинної регресії). Розглянемо ці задачі докладніше.

Як зазначено вище, у практичних дослідженнях часто виникає необхідність встановлення залежності між однією вихідною величиною та набором вхідних параметрів, які впливають на значення цієї величини. Результатом є функція багатьох змінних:

$$y = f(x_1, x_2, \ldots, x_n).$$



У найпростішому і найбільш поширеному випадку розглядається залежність вихідної змінної від одного аргументу:

$$y = f(x). \qquad (2.5)$$

Функція (2.5) може бути більш чи менш складною, залежно від явища, що розглядається.

Рис. 2.4 Приклад побудови регресії для залежності середньої кількості опадів (у мм), залежно від часу (у днях), що пройшов від початку сезону дощів (нелінійна регресія)

Наприклад, експериментально досліджується залежність середньої кількості опадів (у мм), що випадають у певній місцевості, від кількості днів, що пройшли від початку сезону дощів. На основі реальних опадів побудовано графік, на якому кожна точка відповідає одному дощу: по осі абсцис

відкладається номер дня від початку сезону дощів, на який спостерігався дощ, а по осі ординат – кількість мм води, що випала під час цього дощу – рис. 2.4. Шляхом регресійного аналізу можна приблизно побудувати функціональну залежність, за допомогою якої для інших днів можна спрогнозувати кількість опадів, які в середньому випадатимуть у даній місцевості.

Очевидно, залежність рис.2.4 є досить складною з математичної точки зору. У найпростішому випадку функція (2.5) представляється лінійною залежністю:

$$y = f(x). \tag{2.6}$$

де коефіцієнти регресії a і b мають бути визначені на основі експериментальних даних про процес, що моделюється.

Наприклад, виконано декілька експериментальних замірів величини y_i при заданій величині x_i – рис. 2.5, що показані точками.

.

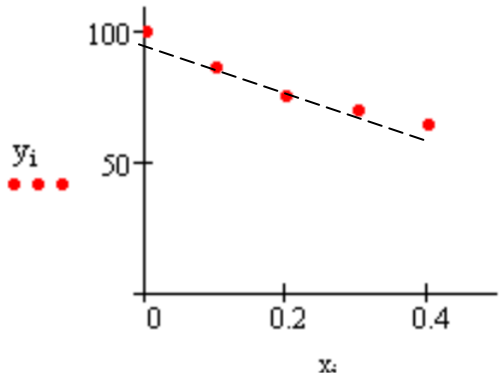


Рис. 2.5 Функціональна залежність, близька до лінійної

Після виконання експериментів постає задача відтворення функціональної залежності величини y від x. Як уже сказано вище, можливо виконати апроксимацію лінійною залежністю (2.6) (і яка показана пунктиром на рис. 2.5). Можливо, більш адекватно відображувала би явище квадратична залежність (або які-небудь інші варіанти функціональної залежності), але з математичної точки зору, як найпростіша, узята саме лінійна залежність (2.6).

Обрати конкретні коефіцієнти можна за методом найменших квадратів: слід розглянути відхилення $\varepsilon_i = (y_i - (ax_i + b))$ та знайти мінімум суми квадратів таких нев'язок:

$$\Sigma\varepsilon_i^2 \to \min.$$

Із цієї вимоги слідує система двох рівнянь, з якої можна визначити коефіцієнти лінійної регресії:

$$\begin{cases} a\sum x_i^2 + b\sum x_i = \sum x_i y_i \\ a\sum x_i + bn = \sum y_i \end{cases} \quad \to \quad a = \frac{\frac{\sum x_i y_i}{n} - x_{cp} y_{cp}}{\frac{\sum x_i^2}{n} - (x_{cp})^2}; \qquad b = y_{cp} - a x_{cp}. \quad (2.7)$$

де x_i, y_i – пара значень відповідно незалежної змінної та функції, що відповідають i-тому вимірюванню;

x_{cp}, y_{cp} – середні арифметичні усіх значень x_i та y_i відповідно;

n – кількість вимірювань, тобто пар x_i, y_i.

Аналогічну до (2.7) систему можна отримати і для коефіцієнтів квадратичної регресії:

$$\begin{cases} a\sum x_i^4 + b\sum x_i^3 + c\sum x_i^2 = \sum x_i^2 y_i \\ a\sum x_i^3 + b\sum x_i^2 + c\sum x_i = \sum x_i y_i \\ a\sum x_i^2 + b\sum x_i + cn = \sum y_i. \end{cases}$$

Невідомі коефіцієнти a, b, c можна знайти, розв'язуючи отриману систему лінійних алгебраїчних рівнянь, наприклад, методом Крамера (звичайно, можливим є застосування і інших підходів):

$$a = \frac{D_a}{D} = \frac{\begin{vmatrix} \sum x_i^2 y_i & \sum x_i^3 & \sum x^2 \\ \sum x_i y_i & \sum x_i^2 & \sum x_i \\ \sum y_i & \sum x_i & n \end{vmatrix}}{\begin{vmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 \\ \sum x_i^3 & \sum x_i^2 & \sum x_i \\ \sum x_i^2 & \sum x_i & n \end{vmatrix}},$$

$$b = \frac{D_b}{D} = \frac{\begin{vmatrix} \sum x_i^4 & \sum x_i^2 y_i & \sum x_i^2 \\ \sum x_i^3 & \sum x_i y_i & \sum x_i \\ \sum x_i^2 & \sum y_i & n \end{vmatrix}}{\begin{vmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 \\ \sum x_i^3 & \sum x_i^2 & \sum x_i \\ \sum x_i^2 & \sum x_i & n \end{vmatrix}},$$

$$c = \frac{D_c}{D} = \frac{\begin{vmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 y_i \\ \sum x_i^3 & \sum x_i^2 & \sum x_i y_i \\ \sum x_i^2 & \sum x_i & \sum y_i \end{vmatrix}}{\begin{vmatrix} \sum x_i^4 & \sum x_i^3 & \sum x_i^2 \\ \sum x_i^3 & \sum x_i^2 & \sum x_i \\ \sum x_i^2 & \sum x_i & n \end{vmatrix}}.$$

Зважаючи на складність (непрактичність) формул квадратичної регресії, в роботі будемо застосовувати лінійну регресію, розмірність якої встановимо нижче.

Переходячи до питання розмірності, можна згадати, що на практиці одна контрольована величина може бути функцією багатьох вхідних параметрів, як в принципі і є у даному дослідженні:

$$y = f(x_1, x_2, \ldots, x_m).$$

Форма вказаної залежності може бути досить складною (залежно від конкретного процесу, що досліджується), але з математичної точки зору найпростішим є відтворення рівняння лінійної множинної регресії:

$$y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_mx_m. \tag{2.8}$$

Рівняння (2.8) називають гіперплощиною, зважаючи на те, що таке рівняння із двома незалежними аргументами у правій частині визначає реальну площину у тривимірному фізичному просторі.

Задача багатовимірного регресійного аналізу полягає у визначенні коефіцієнтів a_i на основі доступних експериментальних даних. Під експериментальними даними мається на увазі таблиця відповідності

виміряних значень вхідних параметрів x_{ij} та значення контрольованої величини y_i, загальний вид якої показано як табл. 2.1.

Таблиця 2.1 - Загальний вид таблиці експериментальних висхідних даних для зведення рівняння множинної регресії.

№ експ	x_1	x_2	…	x_m	y
1	x_{11}	x_{12}	…	x_{1m}	y_1
2	x_{21}	x_{22}	…	x_{2m}	y_2
…	…	…	…	…	…
n	x_{n1}	x_{n2}	…	x_{nm}	y_n

Для виконання конкретних математичних операцій працюють з трьома матрицями, що описуються нижче.

Першою записується регресійна матриця:

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1m} \\ 1 & x_{21} & x_{22} & \dots & x_{2m} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n1} & x_{n2} & \dots & x_{nm} \end{pmatrix}.$$

Далі береться матриця-стовпець значень контрольованої величини:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}.$$

Останньою формується матриця-стовпець шуканих коефіцієнтів:

$$A = \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_m \end{pmatrix}.$$

Використовуючи метод найменших квадратів, операції диференціювання за компонентами вектора, та виконуючи необхідні перетворення, отримують наступну формулу для знаходження невідомих коефіцієнтів рівняння (2.8) множинної лінійної регресії:

$$A = (X^T X)^{-1} X^T Y, \qquad (2.9)$$

де літерою T позначена операція транспонування (заміна рядків на стовпці, або навпаки);

степенем мінус 1 позначена обернена матриця.

Як бачимо, залежність (2.9) є порівняно складною, оскільки вимагає перемноження матриць, а головне, знаходження оберненої матриці. Для цих операцій доцільним є використання спеціалізованого математичного програмного забезпечення, але можна і використати мови програмування загального призначення. Іншим варіантом є вирішення системи рівнянь, які отримуються методом найменших квадратів за допомогою метода Крамера (а не матричного):

$$\begin{cases} na_0 + a_1 \sum x_{1i} + a_2 \sum x_{2i} + \dots + a_m \sum x_{mi} = \sum y_i \\ a_0 \sum x_{1i} + a_1 \sum x_{1i}^2 + a_2 \sum x_{1i} x_{2i} + \dots + a_m \sum x_{1i} x_{mi} = \sum x_{1i} y_i \\ a_0 \sum x_{2i} + a_1 \sum x_{2i} x_{1i} + a_2 \sum x_{2i}^2 + \dots + a_m \sum x_{2i} x_{mi} = \sum x_{2i} y_i \\ \dots \\ a_0 \sum x_{mi} + a_1 \sum x_{mi} x_{1i} + a_2 \sum x_{mi} x_{2i} + \dots + a_m \sum x_{mi}^2 = \sum x_{mi} y_i \end{cases} \qquad (2.10)$$

При цьому головний визначник системи матиме вид:

$$\mathrm{D} = \begin{vmatrix} n & \sum x_{1i} & \sum x_{2i} & \dots & \sum x_{mi} \\ \sum x_{1i} & \sum x_{1i}^2 & \sum x_{1i} x_{2i} & \dots & \sum x_{1i} x_{mi} \\ \sum x_{2i} & \sum x_{2i} x_{1i} & \sum x_{2i}^2 & \dots & \sum x_{2i} x_{mi} \\ \dots & \dots & \dots & \dots & \dots \\ \sum x_{mi} & \sum x_{mi} x_{1i} & \sum x_{mi} x_{2i} & \dots & \sum x_{mi}^2 \end{vmatrix}. \qquad (2.11)$$

Міняючи за методом Крамера відповідний стовпець на стовпець вільних членів системи (праві частини кожного рівняння із (2.10)), отримуватимемо частинні визначники D_i, на основі яких за формулами Крамера визначатимемо шукані коефіцієнти рівняння множинної лінійної регресії (2.8).

Для розрахунків корисним буде спочатку визначити усі суми, присутні у системі (2.10). Саме ці залежності є основою для побудови конкретної математичної моделі процесу оцінки забруднення повітря на основі відомостей про температуру, тиск та рух повітряних мас, що буде зводитися у наступному викладенні.

Виділимо у складі даної роботи два типи алгоритмів: алгоритм роботи з програмою та алгоритм математичних розрахунків значення забруднення.

Програма призначена для отримання одного підсумкового значення ступеня забрудненості повітря, яке вираховується виключно на основі вхідних даних. Такий режим роботи може бути названий пакетним і алгоритм роботи з програмою тоді буде наступним – рис. 2.6.

Рис. 2.6 Алгоритм роботи з програмою оцінки забруднення повітря

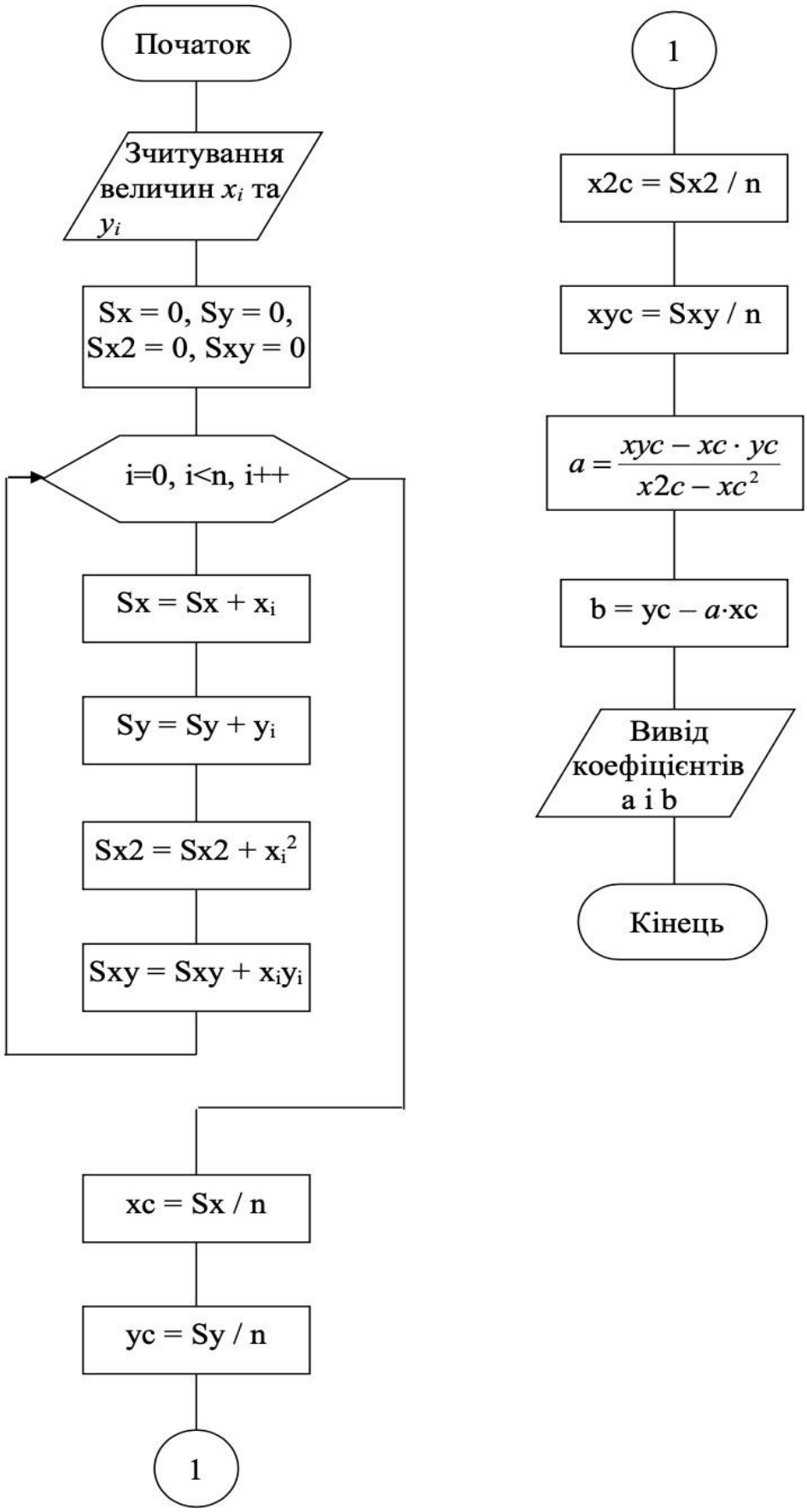
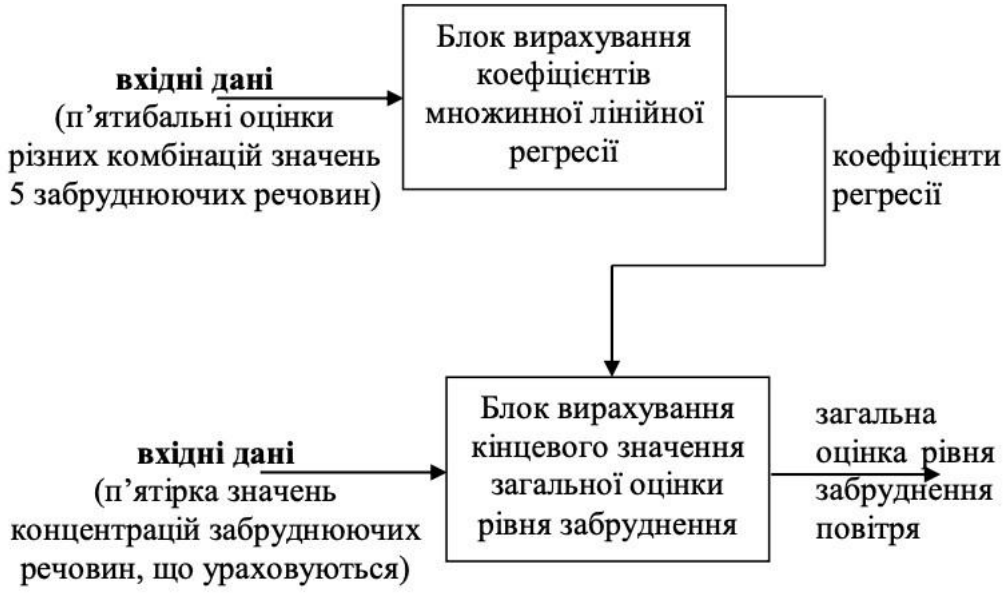


Рис. 2.7 Алгоритм розрахунку лінійної регресії

На рис. 2.6 пункт 4 – «здійснення розрахунку» втілює усі математичні формули розрахунку коефіцієнтів множинної регресії, записані у попередньому підрозділі. Для лінійної регресії алгоритм має вид рис. 2.7.

Для задачі множинної регресії алгоритм є схожим за тим виключенням, що в його основі будуть лежати формули (2.8)-(2.11).



Керуючись наведеними алгоритмами, можна передбачити наступну структуру математичної моделі системи, яка показана на рис. 2.8.

Рис. 2.8 Структура математичної моделі системи оцінки забрудненості навколишнього середовища

2.3 Проектування інформаційної моделі системи та відповідного сховища даних

Ще одним важливим питанням, яке має бути вирішене під час розробки алгоритмічної складової, є отримання масиву вхідних даних для роботи алгоритму лінійної регресії. Його пропонується вирішувати за допомогою експертного опитування спеціалістів, які залучені до вирішення проблем

забруднення навколишнього середовища, зокрема повітря. Процедуру опитування опишемо докладніше.

Кожному експертові пропонувалося декілька десятків наборів із п’яти чисел кожний, де:

- перше число відповідає концентрації чадного газу CO;

- друге число відповідає концентрації вуглекислого газу CO_2;

- третє число є концентрацією озону O_3;

- четверте є концентрацією сірчистого газу SO_2;

- п’яте – вміст частинок пилу у повітрі, а саме, показник $PM_{2,5}$.

У шостому порожньому полі експерт повинен був проставити число від 1 до 5, де:

- оцінка «5» на його думку відповідає відмінній якості повітря із даними показниками, яким можна дихати без будь-яких обмежень по тривалості, під час спортивних занять, інтенсивної фізичної роботи, особам із захворюваннями дихальної системи, і т.п.;

- оцінка «4» відповідає хорошому повітрю, яким може без обмежень по тривалості дихати здорова людина, але яке може викликати негативні реакції у осіб із проблемами дихальної системи, а також у деяких здорових осіб під час інтенсивної дихальної активності;

- оцінка «3» відповідає задовільному повітрю, яким без негайних наслідків для здоров’я можна дихати протягом середніх за тривалістю інтервалів часу порядку тижнів (після яких мають наступати періоди покращення), і яке викликає негативні реакції у осіб із вадами дихальної системи, а також може викликати хвороби дихальної системи у первинно здорових осіб при тривалому диханні (протягом місяців та, тим більше, років);

- оцінка «2» відповідає поганому, шкідливому повітрю, вдихання якого протягом середніх по тривалості інтервалів часу порядку тижнів може спричинити хвороби дихальної системи у початково здорової людини, а також

негайно викликає серйозні порушення дихання у осіб, які вже мають проблеми з дихальною системою.

- оцінка «1» відповідає надзвичайно небезпечному повітрю, вдихання якого без захисних засобів (типу респіратора) практично зразу викликає негативні реакції і може спричинити серйозні захворювання навіть у здорових людей, що невеликі інтервали часу порядку годин вдихають таке повітря, а також загострення (аж до загибелі) у початкового вже хворих осіб.

Результатом опитування стала таблиця виду 2.2 із численним набором вхідних даних, наведена у Додатку 1, яку далі було використано для формування рівняння лінійної множинної регресії.

Таблиця 2.1. Приклад заповнення таблиці вхідних даних для пошуку рівняння множинної лінійної регресії.

№	CO, ppm	CO_2, ppm	O_3, ppm	SO_2, ppm	$PM_{2,5}$, мг/м^3	Оцінка експерта (1-5)
22	…	…	…	…	…	…
23	18	620	0,05	1,8	0,02	**3**
24	…	…	…	…	…	…

Зважаючи на те, що вхідних параметрів у моделі буде всього 5, головний визначник системи (2.11) прийматиме вид:

$$D = \begin{vmatrix} n & \sum x_{1i} & \sum x_{2i} & \sum x_{3i} & \sum x_{4i} & \sum x_{5i} \\ \sum x_{1i} & \sum x_{1i}^2 & \sum x_{1i}x_{2i} & \sum x_{1i}x_{3i} & \sum x_{1i}x_{4i} & \sum x_{1i}x_{5i} \\ \sum x_{2i} & \sum x_{2i}x_{1i} & \sum x_{2i}^2 & \sum x_{2i}x_{3i} & \sum x_{2i}x_{4i} & \sum x_{2i}x_{5i} \\ \sum x_{3i} & \sum x_{3i}x_{1i} & \sum x_{3i}x_{2i} & \sum x_{3i}^2 & \sum x_{3i}x_{4i} & \sum x_{3i}x_{5i} \\ \sum x_{4i} & \sum x_{4i}x_{1i} & \sum x_{4i}x_{2i} & \sum x_{4i}x_{3i} & \sum x_{4i}^2 & \sum x_{4i}x_{5i} \\ \sum x_{5i} & \sum x_{5i}x_{1i} & \sum x_{5i}x_{2i} & \sum x_{5i}x_{3i} & \sum x_{5i}x_{4i} & \sum x_{5i}^2 \end{vmatrix}. \qquad (2.12)$$

Розрахунки цього та шести частинних визначників дозволили вирішити задачу пошуку невідомих коефіцієнтів у рівнянні множинної лінійної регресії (2.8), де $m = 5$. В результаті шукане рівняння отримано у вигляді:

$$y = 4{,}717 - 0{,}089x_1 - 0{,}00004x_2 - 0{,}502x_3 - 0{,}085x_4 - 0{,}0031x_5. \quad (2.13)$$

З інформаційними цілями нагадаємо діапазони концентрацій, що їх вимірюють датчики відповідних забруднюючих величин:

- датчик CO_2: від 0 до 5000 ppm;
- датчик CO: від 0 до 300 ppm;
- датчик O_3: від 0 до 100 ppm;
- датчик SO_2: від 0 до 38 ppm;
- датчик частинок пилу PM2,5: 0-1000 мкг/м^3.

На основі математичної моделі (2.13) можна проводити програмну реалізацію, що і буде виконуватися у наступному розділі. За необхідності можна виконувати коригування моделі, вводячи до неї нові дані від експертів. Таким чином, зважаючи на те, що необхідні розрахунки рівняння регресії виконано, то підключати певну базу даних із великими обсягами інформації немає сенсу. Отже, уся інформація, що необхідна для роботи програми, зводиться до простої текстової форми, у якій подаються звичайні дробові числа.

2.4 Висновки по розділу

Таким чином, у даному розділі проведено розробку основних проектних рішень щодо створення системи аналітики та прогнозування забруднення навколишнього середовища. Докладно розглянуто особливості методів регресійного аналізу та розроблено відповідні алгоритми їх застосування для задачі прогнозування рівня забруднення повітря. Конкретно, модель будувалася у вигляді рівняння множинної регресії із п’ятьма вхідними показниками, якими є рівні забруднення по обраним частинним

забруднюючим речовинам. Зважаючи на загальну математичну складність роботи із множинною регресією, ступінь відповідного рівняння обрано рівним 1 (лінійна множинна регресія). Також у розділі розроблено алгоритми, на основі яких далі будуватиметься програмна реалізація системи, що досліджується. Прийняті рішення дозволяють перейти до проведення програмної реалізації, що і буде виконуватися у наступному розділі.

3 РЕАЛІЗАЦІЯ СИСТЕМИ АНАЛІТИКИ ТА ПРОГНОЗУВАННЯ ЗАБРУДНЕННЯ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

3.1 Обгрунтування вибору інструментальних засобів розробки проекту

Першочерговим питання, яке постає перед розробниками при виборі інструментальних засобів практично будь-якого програмного забезпечення, є вибір технології програмування (методики, парадигми). На сьогоднішній день активно використовуються у реальній практичній діяльності об’єктно-орієнтоване програмування (часто у формі компонентного програмування – коли програміст використовує готові обширні бібліотеки класів, написаних сторонніми фахівцями замість того, щоби у кожному проекті «вигадувати велосипед») та структурне програмування (коли програма складається із окремих блоків типу підпрограм). Під підпрограмою на сьогоднішні день мається на увазі функція, оскільки поняття процедур майже вийшли із використання. Структурне програмування доцільно застосовувати для невеликих або максимум середніх за розмірами проектів. Для великих проектів даний підхід буде неефективним, оскільки будь-хто (один єдиний розробник або ціла команда програмістів) буде зубитися у великій кількості функцій, не структурованих у більш крупні сутності. Об’єктно-орієнтоване програмування якраз і дозволяє розробнику застосувати певну агрегацію таких структурних одиниць програми, як функції, до систем більш високого рівня, які називаються класами. Окремі функції при цьому стають методами (членами) даного класу.

Таким чином, головним критерієм у застосування тієї, чи іншої методики програмування виступає масштаб програмного забезпечення. Також важливим є можливе розширення програми у майбутньому, адже якщо для середнього проекту застосувати структурний підхід до програмування, а у майбутньому він почне розширюватися, то програмістам доведеться

переписувати абсолютно весь проект, щоби перейти до об'єктно-орієнтованого варіанту. Очевидно, що структурний підхід слід застосовувати з обережністю, тільки у випадках, коли роль та місце програми у комплексі засобів автоматизації компанії чи суспільства взагалі є чітко визначеними. У даній роботі проектується продукт для аналізу забруднення навколишнього середовища, який за своїми масштабами можна віднести до проміжних значень між невеликим та середнім. Отже, грунтуючись на розмірі та інших перерахованих особливостях двох існуючих методик програмування, вибираємо структурний підхід як більш простий, що відповідає невеликим (не промисловим) масштабам проектованого ПЗ, а також вимогам до складності (програма не повинна бути складною через цільову аудиторію користувачів, якими є не спеціалісти з галузі ІТ, а екологи чи «домашні» користувачі, які піклуються про чистоту повітря, що вдихають).

Після вибору технології програмування наступним кроком, необхідним для вибору інструментальних засобів розробки, є обгрунтування типу (архітектури) додатку, що може забезпечити заявлену функціональність та інші особливості, що вимагаються завданням на розробку. З точки зору програмної архітектури додаток для аналітики забруднення навколишнього середовища може бути реалізований досить різними шляхами – рис. 3.1.

Рис. 3.1 Можливі типи програмних архітектур програми аналітики забруднення навколишнього середовища

Настільна програма зазвичай вимагає окремої установки, хоча в простих випадках може поширюватися і як портативна версія, що не вимагає інсталяції на ПК. Цей варіант зазвичай прив'язаний до певної програмної платформи, яка визначається типом встановленої на ПК операційної системи. Всі настільні програми можна розділити на два великі класи: програми для операційних систем сімейства Windows і такі, що відповідають міжплатформенному стандарту POSIX, тобто мають бути сумісними між собою. Очевидно, що протягом останніх не менше 20 років на більшості персональних комп'ютерів встановлені різновиди ОС Windows, тому при розробці настільних програм для масового використання (а саме до таких рішень відносяться програми для аналізу забруднень навколишнього середовища) краще орієнтуватися саме на цю систему.

У той же час аналіз Інтернет-статистики за 2021 рік свідчить, що більшість користувачів отримує доступ до глобальної мережі з використанням мобільних пристроїв типу смартфонів. Зазначимо, що частка Windows-сумісних мобільних пристроїв (зокрема, на основі щодо сучасної системи Windows 8.1, яка спочатку позиціонувалася як мобільна), вкрай мала і становить не більше 1 %. В основному на цьому ринку присутні пристрої на базі Android від корпорації Google (близько 75%) та iOS від Apple (біля 25%). Таким чином, можна констатувати відсутність однієї конкретної операційної системи, для якої потрібно розробляти прикладне програмне забезпечення на сьогоднішній день. Ще одним недоліком настільних систем є прив'язка до конкретного комп'ютера, за яким повинна здійснюватися робота з системою. Мобільні та веб-додатки позбавлені цього недоліку і доступ до них може здійснюватися користувачем з будь-якої точки світу (де є, звичайно, Інтернет-з'єднання).

Мобільний додаток, виходячи з самої своєї суті, запускається на пристроях, що володіють невеликим екраном, і тільки одна ця обставина є серйозною перешкодою для використання даного типу архітектури в системах

аналітики та прогнозування. Справді, як вказувалося раніше, основною функцією подібних систем має бути зручне та комфортне відображення певних, не дуже малих обсягів інформації, що неможливо зробити на екрані малого розміру. Крім того, для введення інформації в смартфон або планшет застосовується тільки сенсорний екран, що значно менш зручно при перегляді великих обсягів інформації, порівняно з використанням миші та клавіатури, як при роботі на ПК.

Нарешті, також існує можливість створення ПЗ у вигляді веб-додатку, який надає наступну низку переваг:

- повністю кросплатформенний варіант (якісні веб-програми однаково функціонально працюють у будь-яких сучасних веб-браузерах, незалежно від того, на якій операційній системі, і навіть на якому апаратному пристрої вони запущені);

- дозволяє використовувати обчислювальні ресурси сервера, які завжди значно потужніші, ніж у настільних ПК, а значить алгоритми роботи програми можуть бути складнішими з обчислювальної точки зору та за обсягами необхідної пам'яті;

- можуть мати досить швидкий та зручний інтерфейс користувача, при умові використання в них сучасних веб-технологій (типу AJAX, кешування, локального доопрацювання даних, передзавантаження сторінок тощо);

- можуть запускатися на пристроях з великими (стандартними для ПК) екранами та зручними засобами введення типу миші та клавіатури.

Саме ці особливості роблять веб-додатки, тобто, фактично веб-сайти із розширеною логікою найбільш зручним засобом для проведення аналітики забруднення навколишнього середовища для використовування широкими масами людей.

Отже, перейдемо до вибору такого інструментального засобу, як мова програмування або комплексу мов – за умови, наприклад, написання програмного продукту у вигляді розподіленого клієнт-серверного додатку, а

також при деяких інших умовах). Отже обрано веб-розробку тому давайте розберемо інструменти, які можна використовувати в даному випадку.

Сучасні інструменти для розробки Інтернет-ресурсів настільки великі, що розбити їх усі в цій роботі неможливо. Тому в процесі відбору ми зосередимося на об’єктивних факторах (наприклад, підтримка сучасних методів програмування) і суб’єктивних факторах, але тих, які набули широкого впливу, як-от особисті схильності до певних мов або, навіть стиль програмування, в свою чергу призвело до масової популярності пов'язаних інструментів розробки.

Тому ми розглянемо найпоширеніші інструменти розробки, які зараз популярні в цій галузі.

Загалом весь процес веб-розробки традиційно ділиться на дві частини: front-end і back-end - графіки. 3.2.

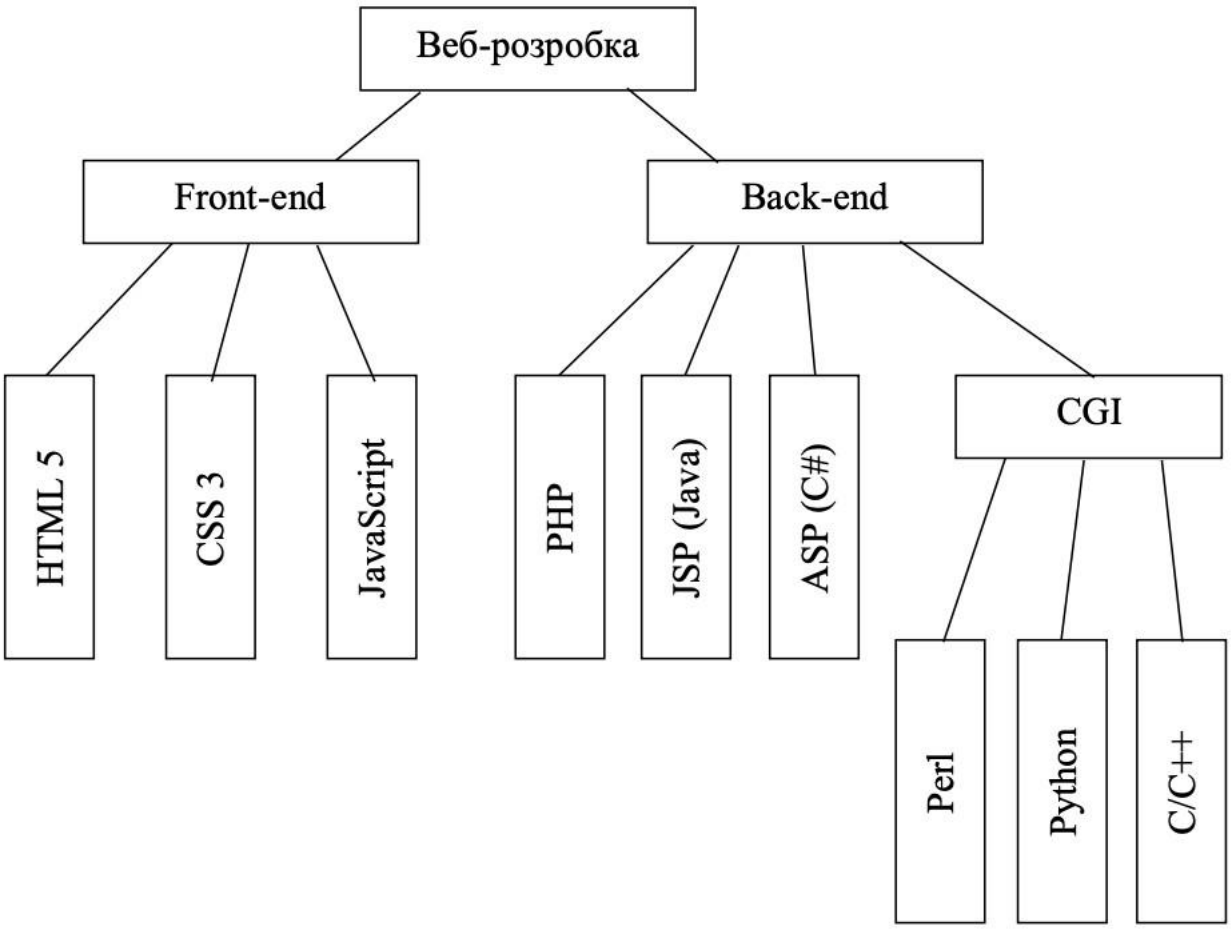


Рис. 3.2 Найпопулярніші засоби розробки сучасних Інтернет-ресурсів

Зважаючи на те, що у даній роботі не буде використовуватися база даних, то використання бек-енд компонентів взагалі кажучи не є потрібним, а розглядати слід лише фронт-енд рішення, можливостей яких буде цілком достатньо для досягнення цілей даної роботи.

Простіше кажучи, "фронт-енд" -- інтерфейс -- у програмуванні ми маємо на увазі те, що бачить користувач. Для настільних систем це означає інтерфейс, тоді як бекенд відноситься до логіки програми, яка в теорії (особливо часто це робиться в системах Linux) зазвичай може бути реалізована як окрема програма з текстовим інтерфейсом консолі. Завдання на передньому плані дозволяють широкому колу користувачів збирати всі вхідні дані, необхідні для виконання значущих дій за допомогою серверної частини консольної утиліти. У багатьох випадках передня і задня частина об'єднані в одне програмне забезпечення. Це стосується настільних систем.

Точніше, що стосується веб-програмування, ситуація максимально поляризована (навіть більше, коли один програміст пише термінальну програму для серверної утиліти, створеної іншим розробником), тому що користувач працює і «бачить» систему через браузер на один віддалений комп’ютер (на комп’ютері, який називається клієнтом), логіка програми та майже вся пов’язана з нею інформація може бути розміщена на абсолютно іншому комп’ютерному сервері.

Таким чином, весь програмний код і вихідний текст, що виконується в браузері, тобто клієнті, належать до інтерфейсної частини. З іншого боку, все, що працює на сервері, відноситься до компонентів бекенда. Без глибокого аналізу ми можемо просто сказати, що набір інструментів для розробки front-end набагато менше, ніж для back-end. в тому числі:

- HTML - п'яте видання (HTML5), найсучасніший стандарт для мови розмітки гіпертексту, де гіпертекст традиційно відноситься до тексту з

гіперпосиланнями на інші частини документа, можливо, в супроводі зображень, звуків, відео тощо. .р.;

- CSS - Third Edition (CSS3) - Правила стилізації різних елементів веб-сторінки, а точніше - Каскадні таблиці стилів;

- JavaScript - найпотужніший інструмент розробки front-end, який дозволяє вивести ваші html сторінки на наступний рівень майже зрілого додатка, такого як робочий стіл, активно реагуючи на дії користувача, передаючи та одержуючи від нього інформацію. Ці прийоми незмінні, тому давайте розглянемо їх докладніше. Hyper Text Markup Language (HTML) – мова розмітки гіпертексту – призначена для написання усіх документів в мережі World Wide Web – WWW.

Документи HTML – це текстові файли зі спеціальними тегами, які передаються від комп'ютера-сервера до клієнтського браузера за запитом клієнта і використовуються клієнтом для відображення вмісту файлу на екрані комп'ютера.

Ви можете використовувати ці вкладки, щоб виділити заголовки документів, змінити колір, розмір і написання літер, а також вставити графіки та таблиці. Але головною перевагою гіпертексту перед звичайним текстом є можливість додавати гіперпосилання до вмісту документа — спеціальні структури HTML, які дозволяють клацнути, щоб переглянути інший документ. Таким чином, «гіпертекст» означає— розвиток ідей для текстових документів, але з більшими можливостями.

Документи HTML мають дві складові: фактичну текстову інформацію, яка є даними, що складають вміст документа, і теги — спеціальні структури HTML, які розмічають документ і контролюють його відображення. Теги HTML визначають, як відтворюється текст, які компоненти будуть виконувати роль гіперпосилань і які графічні чи мультимедійні об'єкти мають бути включені в документ.

Графічна та звукова інформація, що міститься в документах HTML, зберігається в окремих файлах. Програми перегляду документів HTML називаються браузерами, вони інтерпретують теги розмітки, маніпулюють текстом і графікою та відповідно розміщують їх на екрані. Розширення .Htm або .html використовується для файлів, що містять документи HTML.

У переважній більшості випадків теги використовуються парами. Пара складається з тегу, що відкриває <tag_name>, і тегу, що закриває </tag_name>. Дія будь-якої пари тегів починається там, де стикаються початкові теги, і закінчується там, де стикаються відповідні кінцеві теги. Зазвичай пару відкривають і закриваючих тегів називають контейнером, а частину тексту між відкриваючим і закриваючим тегами називають елементом. Першим тегом HTML є однойменний тег <html>. Він завжди відкриває документ так, ніби тег </html> має бути в останньому рядку.

Підсумовуючи особливості мови HTML, можна сказати, що за допомогою різноманітних тегів можна реалізувати таблиці, форматувати текст, вставляти зображення, відео, аудіофайли тощо у свій документ. Під час його розробки тег <style> і пов’язані з ним каскадні таблиці стилів приділяли все більше уваги.

Каскадні таблиці стилів (CSS) — це мова, яка містить набір властивостей, що визначають зовнішній вигляд документа. Специфікація CSS (CSS3 - поточний) визначає властивості та мову опису для зв'язування властивостей з елементами в документі. Останню інформацію про інновації та елементи, які підтримуються таблицями стилів, можна знайти на веб-сайті консорціуму W3C (www.w3.org).

Таблиця стилів — це абстракція, в якій стиль документа визначається окремо від змісту чи структури. Веб-майстер може додавати таблиці стилів до документа трьома способами – загалом, зі збільшенням рівня складності збільшуються і можливості, що пропонуються, підвищуючи рівень абстракції. Перший спосіб — використовувати вбудовані таблиці стилів. Вбудовані стилі

визначаються безпосередньо в елементі. Другий спосіб — використовувати глобальну таблицю стилів для визначення стилів на початку документа. Третій, найбільш абстрактний і потужний спосіб — використовувати пов’язані таблиці стилів для окремого визначення стилів в іншому документі.

Внутрішні стилі мало відрізняються від традиційного HTML. При використанні внутрішніх стилів зовнішній вигляд документа важко змінити. Перевага цього підходу полягає в тому, що кількість розмітки зменшується, і HTML можна більш повно використовувати для відтворення вмісту презентації. Використання глобальної таблиці стилів дає змогу ефективніше відокремлювати перегляд від вмісту та дозволяє швидко й незалежно змінювати стиль та обробку вашого документа. Використання пов’язаних таблиць стилів надає додаткові переваги, відображаючи вміст у вигляді набору сторінок або використовуючи один файл для визначення цілого веб-сайту. Таким чином, вбудовані стилі, по суті, є таблицями стилів для окремого екземпляра елемента і визначаються безпосередньо в тегу елемента. Внутрішні таблиці стилів визначаються за допомогою атрибута STYLE, а дані атрибута визначаються за допомогою мови таблиць стилів.

Внутрішні стилі допомагають вивчити мову CSS або швидко змінити окремий екземпляр елемента, якщо це необхідно. Однак внутрішні стилі не відповідають ідеї структурованого документа і погано працюють, коли вам потрібно змінити зовнішній вигляд кількох елементів у документі з повним розділенням презентації та вмісту. Щоб відокремити стилі документа від їх структури, таблиці стилів мають бути визначені в заголовку документа або як окремий файл, пов’язаний з документом. Наступною, більш ідеологічно вірною можливістю, є використання окремого тегу <STYLE> для завдання таблиць глобальних стилів, який зазвичай розміщується в заголовку документа. Розміщення усіх стилів документа в одному місці спрощує зміну режиму відтворення документа. Наведений нижче приклад таблиці стилів визначає зовнішній вигляд усіх параграфів в документі. Для зміни режиму

відтворення параграфів потрібно змінити тільки елемент STYLE. Якби використовувалися внутрішні стилі, то довелося б міняти кожен параграф в документі окремо.

Тег <link> розміщується в заголовку документа для визначення зовнішньої таблиці пов'язаних стилів. Його атрибут REL вказує, що пов'язаний файл є таблицею стилів, а атрибут TYPE визначає тип MIME таблиці стилів. Властивість HREF є покажчиком URL-адреси на зовнішню таблицю стилів. Відповідна таблиця стилів повинна містити лише контекстні правила та визначення стилів і не повинна містити HTML-код.

Після короткого розгляду каскадних таблиць стилів ви також повинні зрозуміти головну особливість останнього та дуже потужного інструменту розробки інтерфейсу, мови програмування JavaScript. Це спеціалізована мова програмування, яка традиційно використовується для керування об'єктною моделлю документів у браузері під час перегляду Інтернету (є продукти, які перетворюють JavaScript на мову загального призначення, команди якого виконуються на сервері, наприклад Node.JS; але цей підхід різко суперечить початковій концепції використання цієї мови програмування, а з огляду на велику кількість серверних трансформацій традиційної розробки бекенда, досить багато програмістів також вважає JavaScript абсолютно незручним, тому ми не будемо про це говорити. варіант).

Особливістю JavaScript є те, що його вихідний код інтерпретується, що є дуже гнучким, але повільним рішенням.

Оператори в цій, як правило, C-подібній мові відокремлюються крапкою з комою. Ця мова чутлива до регістру та часто ігнорується початківцями, ймовірно, тому, що HTML, який часто використовується разом із JavaScript, не чутливий до регістру (теги HTML та назви атрибутів можна писати малими та великими літерами). Застосуйте слабкий (динамічний) контроль типу до даних. В операторах з різними типами даних останні автоматично зводяться до потрібного типу. Типи даних можуть бути примітивними і складними.

Примітивні типи містять прості однорідні значення, які можна передавати як аргументи функціям за значенням, а не за посиланням. Складені типи містять різнорідні дані (включаючи складені дані), які передаються функціям лише за посиланням.

JavaScript є об'єктно-орієнтованим, але заснований на прототипах, а не на класах. Є чотири типи об’єктів: вбудовані об’єкти, об’єкти браузера, об’єкти документа та об’єкти користувача (програміста).

Введення-виведення здебільшого обмежується взаємодією з документами та користувачами. За замовчуванням доступ до локальної файлової системи заборонений. Однак браузери можуть надати спеціальні функції, які допоможуть вам працювати з файловою системою користувача, незважаючи на попередження про небезпеку маніпулювання файлами.

Скрипти JavaScript активно взаємодіють з об'єктами, вбудованими у веб-сторінки. Для цього вони фактично створені. Але перш ніж ця взаємодія стане можливою, вам потрібно вставити код сценарію в текст HTML-документа. Існує кілька способів пов’язати HTML-документ із певним сценарієм, але зазвичай вони просто розміщуються в тегу контейнера <script>. Тобто між дескрипторами <script> і </script>.

У цьому випадку контейнер <script> буде розташований безпосередньо в HTML-документі і будь-де. Програмний код записується безпосередньо в HTML-документ або в спеціальний текстовий файл, який можна викликати з основного HTML-документа. Спочатку розглянемо перший варіант. Спочатку браузер знаходить тег <script> в тілі веб-документа і намагається обробити весь наступний текст як код сценарію. І так далі, поки не зустрінеться закриваючий тег </script>. Після цього всі наступні символи будуть розглядатися як текст HTML. Будь-який HTML-документ може містити будь-яку кількість «включає сценарій», але кожен має бути відкритий і закритий з відповідною розміткою. Їхнє положення в тілі HTML-документа іноді може

вплинути на функціональність всієї веб-сторінки, але це буде розглянуто пізніше.

Тег контейнера <script> може містити атрибут SRC, який вказує ім’я або URL-адресу текстового файлу, що містить код сценарію. Цей атрибут необхідний, якщо скрипт розташований не безпосередньо в HTML-документі, а в окремому файлі. Розширення файлу сценарію може бути будь-яким, але зазвичай використовується js.

Якщо сценарій знаходиться в окремому файлі, він, звичайно, не записуватиме теги <script> і </script>. Сценарій, завантажений із зовнішнього файлу, можна просто розглядати як вставлення його в HTML-документ.

Як згадувалося раніше, один файл зазвичай містить бібліотеку функцій (визначень функцій), а також сценарії, які використовуються в кількох документах HTML на одному або кількох сайтах. Сценарій також може бути записаний у вигляді рядка операторів, розділених крапкою з комою. Такий рядок у лапках використовується як значення властивості події.

Виклики функцій та їх визначення можна виконувати в будь-якому порядку, якщо вони знаходяться в одному контейнері <script>. Якщо вони розміщені в різних контейнерах <script>, визначення функції має передувати її виклику. Так само, якщо функції визначені в окремих файлах, їх необхідно завантажити в HTML-документ перед викликом цих функцій.

Розглянувши особливості мови JavaScript (разом із основою у вигляді HTML та CSS) можна із впевненістю сказати, що її доцільно узяти в якості основного засобу для зведення системи оцінки забрудненості навколишнього середовища.

3.2 Кодування програмного продукту, що реалізує проект системи аналітики та прогнозування забруднення навколишнього середовища

При розробці програмного продукту наступним кроком після вибору технології програмування, мови програмування та основного засобу розробки є створення інтерфейсу користувача системи, тобто її зовнішнього вигляду. Він формується набором та розміщенням необхідних елементів управління, які призначені для збору вхідної інформації від користувача та для відображення йому вихідної інформації по роботі програмного продукту.

Беручи до уваги специфіку проектованого програмного продукту (який має надати оцінку загального рівня забрудненості на основі набору частинних показників забрудненості, які є вхідними даними), можна зробити висновок, що у нього має бути 2 основних робочих зони: для збору інформації та для відображення результатів оцінки. Таким чином, якщо класифікувати інтерфейс користувача, то, незважаючи на складність самої задачі, що вирішується, інтерфейс користувача хоча і є графічним, але досить простим і може бути віднесений до пакетних – по типу рис. 3.1.

Кінцевий вигляд інтерфейсу буде наступним – рис. 3.3.

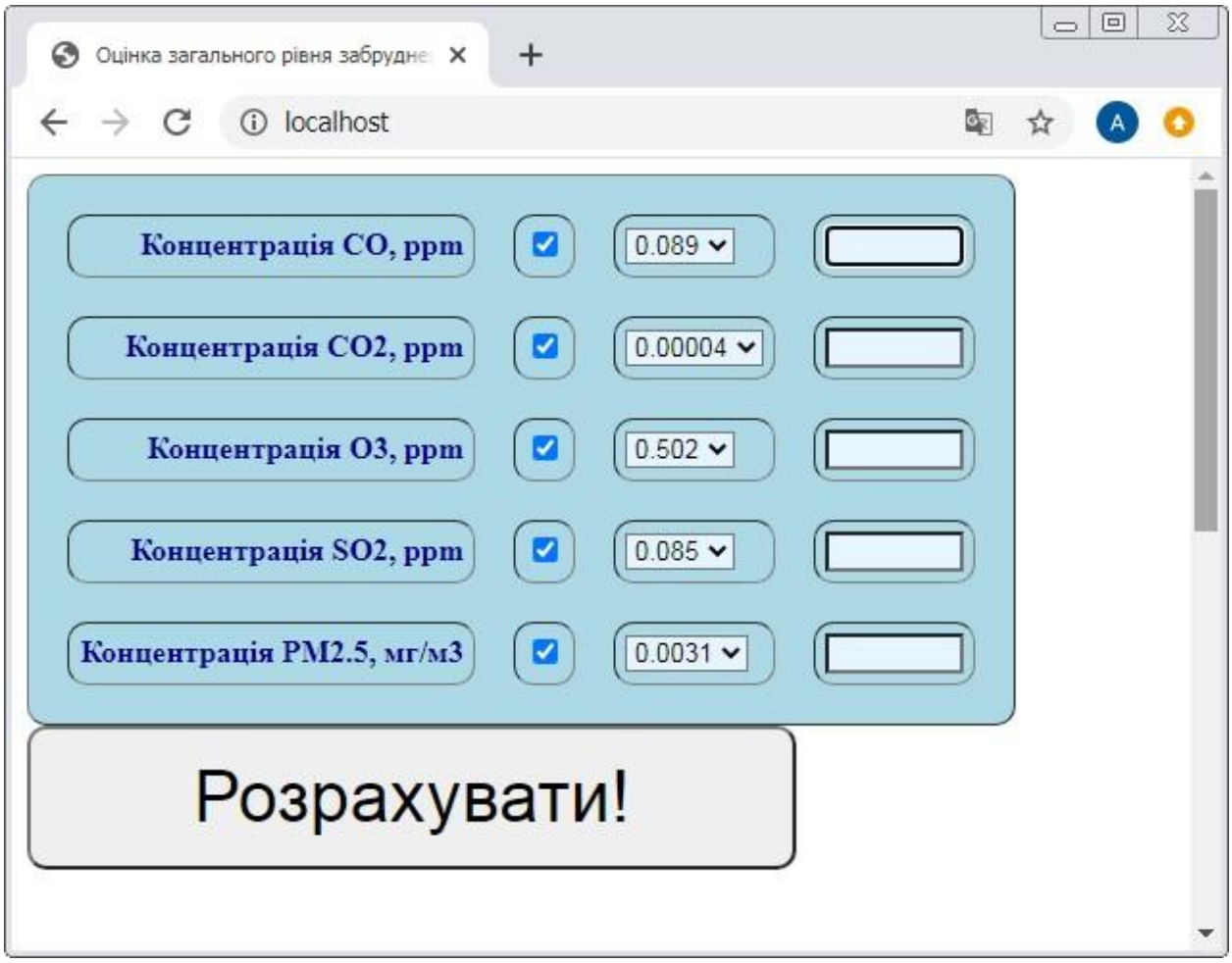


Рис. 3.3 Інтерфейс користувача програми для аналітики та прогнозування рівня забрудненості на основі показів окремих частинних показників забруднення

Програмний продукт, що розробляється, має певні особливості, серед яких, зокрема, виділяється розрахункова процедура, яка виконується за формулою (2.13). Вона, як і багато інших функціональних речей, реалізована функцією мови JavaScript. Наведемо перелік функцій, що використовуються у програмному продукті разом із їхніми короткими описами:

– filldiv() – заповнює основний шар (maindiv), вибудовуючи в ньому таблицю для введення даних потрібної форми та розміру. Ця функція видає у веб-сторінку назви окремих частинних показників та їх ваги (коефіцієнти) у рівнянні лінійної регресії;

– startcalc() – реалізує формулу (2.13) і побудована на базі алгоритмів, розглянутих у попередньому підрозділі;

– checknum() – сервісна функція, яка перевіряє, щоби користувач вводив у поля тільки числові дані, а у протилежному випадку очищує введене (нечислове) значення і видає відповідне попередження;

– bindhandlers() – стандартна для JavaScript функція, яка викликається уже коли всі елементи управління створені, і, отже, для них можна задати обробники методом addEventListener.

Інші особливості розробленого програмного забезпечення можна передивитися безпосередньо у його початковому коді, що наводиться у додатку А.

3.3 Опис процесу застосування розробленої системи

3.3.1 Особливості встановлення розробленого програмного продукту

Розроблений програмний продукт може розміщуватися на сервері, або використовуватися локально на одному комп’ютері. У першому випадку особливості налаштування залежать від програмного середовища сервера і мають здійснюватися адміністратором даного сервера. При локальному використанні найбільш зручно встановити якийсь інтегрований пакет, що включає одразу і веб-сервер, і інтерпретатор мови PHP, і сервер баз даних MySQL. Такими рішеннями, наприклад, є Денвер, який досить давно не оновлювався і застарів, та XAMPP, який навпаки є проектом, що досить динамічно розвивається. Відповідно, на локальний ПК було встановлено цю систему – рис. 3.4.

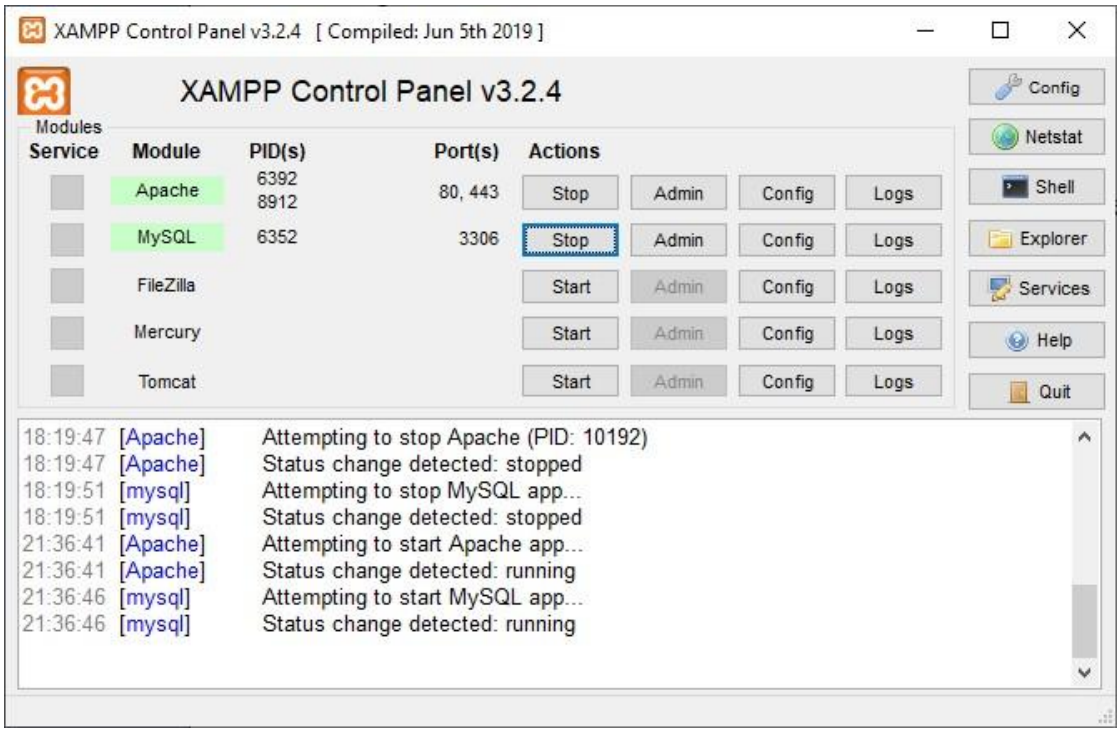


Рис. 3.4 Панель управління пакету для веб-розробки XAMPP

Отже, установка програмного продукту включає наступні дії:

- встановити інтегрований пакет типу XAMPP;

- включити веб-сервер;

- скопіювати у робочу папку XAMPP/htdocs файли, що прикладаються до даної роботи;

- завантажити браузер та перейти за адресою localhost, впевнившись, що продукт працює належним чином;

- програмний продукт готовий до використання.

3.3.2 Особливості експлуатації розробленого програмного продукту

Розроблене програмне забезпечення є спеціалізованим інструментом і в основному призначене для фахівців, що займаються проблемами екології, хоча, звичайно, може бути використане і будь-яким пересічним користувачем мережі Інтернет для оцінки поточного рівня забрудненості повітря по частинним показникам. У будь-якому випадку до програмного забезпечення

має поставлятися комплект документації, з якого обов'язковим елементом є інструкція користувача. Її можна представити пунктами, що слідують нижче.

1) Для початку роботи з програмою зайти у браузер та задати адресу розробленого веб-додатку (локальну localhost – при роботі на локальному сервері, або Інтернет-адресу – за умови, що розроблений продукт буде вміщено у відкритий доступ у Веб). Іншими словами достатньо клацнути по файлу index.html.

2) Після завантаження сторінки користувач має ввести усі частинні показники забрудненості по описаним у програмі показникам: рівень CO, CO2, O3, SO2, PM2.5, причому саме у вказаних у програмі одиницях. Якщо у користувача наявні відомості у інших одиницях, слід скористатися одним із доступних онлайн калькуляторів, наприклад, https://www.gazoanalizators.ru/converter.html

3) Після введення значень усіх наявних у користувача частинних показників, слід впевнитися, що навколо них стоять увімкнені прапорці, а якщо якогось частинного показника немає в наявності, то прапорець біля нього слід зняти, тим самим виключаючи його із розрахунку.

4) Після встановлення усіх необхідних прапорців біля наявних у користувача і введених ним раніше частинних показників, необхідно перевірити ваги усіх таких показників: якщо у користувача наявні нові відомості щодо регресійної моделі, то потрібні числа слід відкоригувати, а інакше – скористатися запропонованою у цій роботі моделлю (формула (2.13)).

5) Коли усі дані будуть введені, слід ще раз усі їх перевірити та натиснути кнопку «Розрахувати!».

6) У новому елементі управління, який відкривається після натискання на кнопку «Розрахувати!», слід подивитися та проаналізувати кінцевий результат. За необхідністю записати його на постійні носії.

7) Для завершення роботи з програмою слід закрити вікно (або вкладку) браузера.

Керуючись цією інструкцією, користувач будь-якої кваліфікації зможе здійснювати оцінку загального рівня забрудненості повітря навколишнього середовища на основі частинних показників забрудненості.

3.4 Оцінка ефективності реалізованої системи

3.4.1 Тестування створеного програмного продукту

Наступним кроком після впровадження програмного забезпечення є тестування продукту. Точніше, альфа-тестування виконується розробниками кілька разів протягом процесу розробки і дозволяє оцінити продуктивність різних частин програмного коду. Весь готовий продукт проходить бета-тестування і також може бути виконаний за різними сценаріями. Щоб мати можливість раціонально підібрати види тестів і методи для цієї розробки, розглянемо особливості цього процесу більш детально.

Тестування програмного забезпечення (надалі - програмне забезпечення) є важливою частиною його розробки і в деяких випадках може зайняти деякий час у порівнянні з часом кодування всього вихідного коду проекту. Поняття тестування можна трактувати в досить широкому діапазоні, а також описувати його можливі варіанти з різних причин. Вся багатогранність цього процесу вимагає детального аналізу та відбору тих чи інших видів тестування програмного забезпечення. Спочатку ми наведемо прапори - графіки, які можна використовувати для класифікації типів тестів. 3.5.

Розглянемо більш детально можливість поділу всіх типів тестів на класи за допомогою різних функцій на малюнку 3. 3.5.

Перший варіант цієї класифікації є найбільш численним, а це означає, що існує багато різних цілей для виконання тестів. Очевидно, що найголовніше в цьому списку – це перевірити, чи програмний продукт виконує належну йому функцію відповідно до ТЗ. Тому цей тип тестування програм

називається функціональним тестуванням. Крім того, ви можете розглянути наступні цілі, пов’язані з тестуванням нефункціонального типу:

Рис. 3.5 Ознаки, за якими може проводитися класифікація видів тестування програмного забезпечення

а) Перевірте продуктивність програми, оскільки реалізація однієї і тієї ж функції може займати різний час залежно від алгоритмічних характеристик продукту, сформульованого програмістом: розробник виконує певну функцію протягом 10 секунд і 10 хвилин. Очевидно, якщо швидкість ЦП фіксована (тобто різні програми запускаються на одному ПК, що є важливою умовою для справедливого порівняння), час виконання деяких функцій програми залежить від кількості команд ЦП, які транслюють вихідний код, написаний Програма. програміст. Кількість інструкцій процесора залежить від якості коду, який пише програміст, і налаштувань компілятора, які він встановлює під час трансляції. Тому найважливішим параметром продуктивності є кількість базових інструкцій процесора, необхідних для виконання певної функції програми. Однак, крім кількості інструкцій процесора (яку також можна

описати як «обчислювальну складність алгоритму»), слід враховувати й інші показники продуктивності. Другим найважливішим з цих показників є обсяг оперативної пам’яті, необхідний для запуску програми. Одній програмі може знадобитися 1 Гб пам’яті для виконання деяких функцій, тоді як іншій програмі може знадобитися лише 10 МБ. Часто перший і другий показники суперечать один одному, і скорочення часу на виконання функції майже завжди призведе до збільшення обсягу необхідної оперативної пам’яті.

Отже, зрозуміло, що продуктивність програми в основному залежить від компонентів алгоритму, які формулюються програмістом на етапах створення автомобіля, проектування та безпосередньої реалізації. Важливо відзначити, що на цьому етапі остаточно уточнюються системні вимоги до майбутніх програмних продуктів. Очевидно, що в технічному завданні вказані мінімальні вимоги до комп’ютерних технологій (або мобільних – якщо ви розробляєте мобільний додаток, чи сервера – якщо ви розробляєте веб-додаток), але практика показує, що іноді (і часто) під час розробки , ці мінімальні системні вимоги можуть бути узгоджені повторно, і ці вимоги не обов’язково будуть збільшуватися або зменшуватися. Тестування продуктивності дозволяє відповісти на питання: які саме системні вимоги до кінцевої версії продукту.

b) Тестовий інтерфейс користувача (User Interface), інтерфейс користувача. При цьому зовнішній вигляд програми та її відповідність загальним правилам побудови графічних інтерфейсів, розробленим за десятиліття, в які будувались віконні додатки. Важливо відзначити, що цей тип тестування призначений для звичайних користувачів. Це буде аналізувати елементи керування: їх тип, розмір, колір тощо, коротше кажучи, «гарний» загальний вигляд програми.

c) Тестувати UX (User Experience), тобто зручність звичайних користувачів використовувати загальні функції програми. Він оцінює чіткість і простоту використання інтерфейсу.

d) Тестування захисту програмного забезпечення, яке оцінює здатність програми протистояти зловмисному впливу на всі можливі типи програм (по-перше, підсистему захисту програмного забезпечення повинна бути перевірена на предмет несанкціонованого використання).

e) Перевірте складність процесу встановлення програмного забезпечення. Метою тут є визначення ймовірності встановлення програмного забезпечення на будь-якому пристрої, який відповідає мінімальним системним вимогам, виконується належним чином і призводить до наступного нормального використання програмного забезпечення.

f) Тестування програми на сумісність для визначення того, що вона коректно працює на різних версіях цільової операційної системи. Звичайно, здебільшого це стосується різних версій Windows (7, 8, 10, 11 або серверних версій), хоча часто можна перевірити сумісність бібліотек за допомогою програм, драйверів та інших опцій додаткового програмного забезпечення, необхідного для запуску програми. . .

g) тестування надійності, призначене для визначення того, що умови використання програми не зміняться з часом. Насправді, можливо, що, наприклад, під час тривалого використання програми накопичилися певні помилки, і коли їх кількість перевищує певну межу, використання програми стає неможливим (або ускладненим).

h) Тестування локалізації, метою якого є перевірка того, що програма виглядає нормально та працює, вибираючи різні мови, надані програмістом.

Тестувати UX (User Experience), тобто зручність звичайних користувачів використовувати загальні функції програми. Він оцінює чіткість і простоту використання інтерфейсу.

Тестування захисту програмного забезпечення, яке оцінює здатність програми протистояти зловмисному впливу на всі можливі типи програм (по-перше, підсистему захисту програмного забезпечення повинна бути перевірена на предмет несанкціонованого використання).

Перевірте складність процесу встановлення програмного забезпечення. Метою тут є визначення ймовірності встановлення програмного забезпечення на будь-якому пристрої, який відповідає мінімальним системним вимогам, виконується належним чином і призводить до наступного нормального використання програмного забезпечення.

Тестування програми на сумісність для визначення того, що вона коректно працює на різних версіях цільової операційної системи. Звичайно, здебільшого це стосується різних версій Windows (7, 8, 10, 11 або серверних версій), хоча часто можна перевірити сумісність бібліотек за допомогою програм, драйверів та інших опцій додаткового програмного забезпечення, необхідного для запуску програми. . .

Тестування надійності, призначене для визначення того, що умови використання програми не зміняться з часом. Насправді, можливо, що, наприклад, під час тривалого використання програми накопичилися певні помилки, і коли їх кількість перевищує певну межу, використання програми стає неможливим (або ускладненим).

Тестування локалізації, метою якого є перевірка того, що програма виглядає нормально та працює, вибираючи різні мови, надані програмістом.

Тестування також поділяється на альфа-тестування і бета-тестування. У першому випадку автором процесу тестування є сам програміст. Хто розробляє програму, тестування проходить в процесі розробки. У другому випадку тестувальники, як правило, є сторонніми, які оцінюють готове програмне забезпечення (яке, на думку розробників, готове до випуску).

Як зазначено в аналізі, знак «2. Ступінь автоматизації» є дуже важливим параметром для автоматизації тестування. Також є можливість виконувати більше перевірок комбінації вхідних даних, ніж ручна робота. Однак є багато труднощів наступного характеру.

По-перше, існує фундаментальна проблема виконання автоматизованого тестування за допомогою спеціально розроблених програм, зазвичай написаних тестувальниками, для врахування деталей програмного забезпечення, що розглядається. Очевидно, необхідно переконатися, що самі тестові програми є безпомилковими, що, можливо, впевнено в їх надзвичайно простих випадках, але сумнівно у випадку складних алгоритмів.

По-друге, сучасні програми мають багато різних застосувань, але для кожного з них завжди можна вибрати деякі вхідні дані із зовнішніх джерел, релевантних для тестованої програми: від користувача, з локальної мережі, з Інтернету тощо. Проблема полягає в тому, щоб адекватно ізолювати ті дійсно важливі вхідні дані, які потребують автоматичного призначення, і відрізати ті, які не вписуються в автоматичне налаштування (і, ймовірно, не повинні бути перевірені взагалі). Особливим випадком цієї проблеми є виділення достатніх діапазонів під час тестування для цих вхідних значень, які вибрані як важливі.

Нарешті, ще однією проблемою автоматизованого тестування може бути складність оцінки кінцевого результату, який часто не обов’язково представляється у зручній числовій формі. Натомість результатом може бути оброблене зображення, неоднорідний набір даних (масиви чи вектори), текстові рядки (навіть великий текст) тощо. Щоб оцінити деякі з цих результатів, необхідно використовувати методи штучного інтелекту, які дозволяють проводити такі оцінки, але складність алгоритмів, які будуть використані, може бути порівнянна зі складністю самого програмного забезпечення, яке буде тестуватися. При цьому проблема доцільності автоматичної перевірки загалом повинна бути уважно розглянута, оскільки в деяких завданнях, які погано набрані, навіть не вдасться розробити відповідний алгоритм.

Також зауважте, що іноді виникають непередбачувані проблеми з автоматизованим тестуванням, наприклад, його висока складність. Здається, що вся суть будь-якої автоматизації процесів полягає в зменшенні складності

деяких ручних операцій, на які насправді впливає автоматизація. Але в цьому випадку (автоматизація процесу тестування ПЗ) кажуть, що в багатьох випадках написати якісну тестову програму не набагато легше, ніж вручну тестувати продукт, тим більше, що повної, 100-відсоткової впевненості немає. Сама тестова програма працює саме так, як і було задумано).

Усі ці труднощі змусили деякі компанії взагалі відмовитися від автоматизованого тестування, вважаючи за краще використовувати дешеву робочу силу дистанційних тестувальників, найнятих із країн, що розвиваються. Тому в цій статті ми проведемо ручне бета-тестування системи за сценарієм білого ящика з позитивним тестом.

3.4.2 Контрольний приклад виконання програмного продукту

Для створеної програмної реалізації було проведено тестування вказаного типу, за результатами якого встановлено наступне:

- програмний продукт працює без виникнення системних помилок, аварійних завершень, зависань, інших порушень доступу до пам’яті чи інших частин системи;

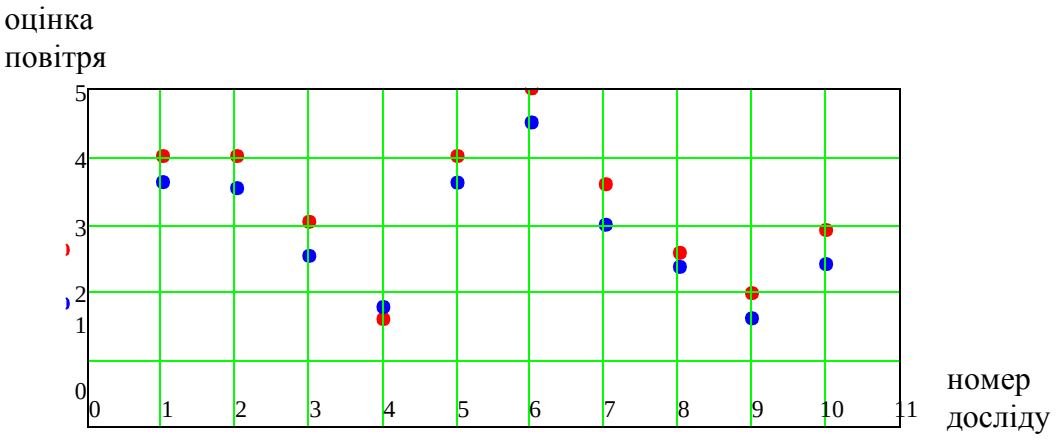
- логіка роботи програми відповідає початково заявленим цілям, що було встановлено в результаті порівняння результатів оцінки рівня забрудненості програмою та експертами.

Для перевірки останньої тези перед формуванням коефіцієнтів (2.13) із початкового набору вхідних даних були випадково вилучені 10 записів, які пізніше використовувалися для оцінки ефективності висновків системи. Порівняння оцінок, наданих системою по цим 10 записам із оцінками експертів показали цілком задовільне середнє відхилення біля 14 % (табл. 3.1), що свідчить про достатній рівень достовірності рішень програми і можливість за допомогою неї ефективно оцінювати загальний рівень забрудненості на основі частинних показників.

Табл. 3.1 Порівняння оцінок, виконаних спроектованою системою та еталонних значень

№	Еталонна оцінка	Оцінка, виставлена системою	Похибка, %
1	4	3,6	10
2	4	3,5	13
3	3	2,5	17
4	1	1,2	20
5	4	3,6	10
6	5	4,5	10
7	3	2,4	20
8	2	1,8	10
9	2	1,7	15
10	3	2,5	17
Середнє	-	-	**14**

Інформація, наведена у таблиці може бути наочно представлена у графічній формі на рис. 3.6. На рис. 3.6, *а* у відповідному масштабі наведено обидва значення оцінки повітря: червоним – еталонна оцінка, а синім – передбачення створеної у даній роботі системи. На рис. 3.6, *б* можна бачити абсолютне значення похибки, що виникає при порівнянні еталонного та отриманого системою значень. Бачимо, що похибка не виходить за абсолютні межі 0,6 (по шкалі від 1 до 5), що у відсотковому вираженні для середнього значення похибки складає розраховане у табл. 3.1 значення 14 %.

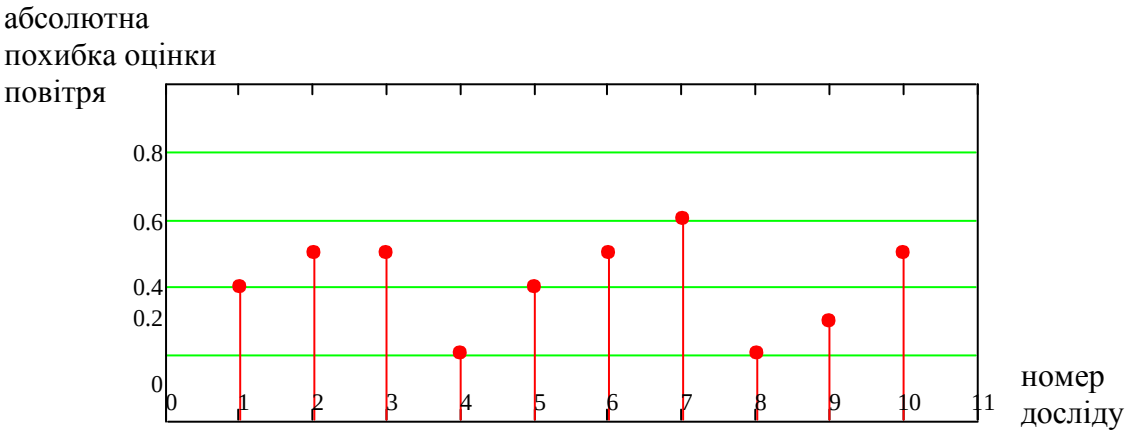


а)

б)

Рис. 3.6 Порівняння еталонних оцінок та оцінок, виставлених системою – *а*; відповідні похибки у абсолютному вираженні – *б*

Таким чином, аналізуючи результати використання системи на тестових контрольних прикладах, бачимо, що точність її роботи може бути оцінена середнім рівнем відхилення результатів, що розраховує програма, та еталонних величин, з середнім значенням порядку 14 %.



3.4.3 Економічна ефективність розробленої системи

Як відомо, будь-яка інженерна (програмна) розробка повинна бути економічно обґрунтованою. Використання нового програмного забезпечення, яке буде створено, має принести користувачеві позитивний економічний ефект. У цьому випадку ефект досягається за рахунок економії часу людей, які бажають визначити кількість забруднення повітря, що можна зробити двома способами: шляхом ручного розрахунку (за допомогою калькулятора) і за допомогою розробленої програми.

Однократний процес розрахунку за (2,13) може бути оцінений на рівні біля D*t* = 3 хв.:

D*t* = 3 хв.

При виконанні цих операцій в електронному вигляді з використанням розробленої програми часи будуть орієнтовно рівними:

$$Dt' = 1 \text{ хв.}$$

Економія на однократному розрахунку складатиме:

$$t = Dt - Dt' = 2 \text{ хв.}$$

Якщо обчислювати чистоту повітря одного разу на день (щодня), то за рік, тобто $N = 365$ днів, економія часу складає:

$$t_д = N \times t = 365 \times 2 = 730 \text{ хв.} \approx 12 \text{ год.}$$

Навіть при мінімальній оплаті праці відповідної людини на рівні $c_{min} = 36{,}11$ грн/год (що встановлено законодавчо в Україні на момент написання даної роботи) річна економія тільки на заробітній платі відповідального працівника складе (і це без урахування додаткових нарахувань на величину заробітної плати, витрати на електроенергію і т.п.):

$$E = 36{,}11 \cdot t_р = 439 \text{ грн.}$$

Зверніть увагу, що всі оцінки проводяться на найнижчому рівні (для найбільш ретельної оцінки). З огляду на те, що програмний продукт є безкоштовним (впровадження здійснюється в рамках університетської освітньої діяльності, а не для комерційного розповсюдження), можна сказати, що використання розробленої програми для користувачів економічно виправдано та економія не менше 12 годин на рік. в їх готівці в еквіваленті не менше 439 грн.

Слід відмітити, що інженерні розробки окрім економічного, можуть мати і інші види ефектів (екологічний, політичний, тощо). В даному випадку розроблене програмне забезпечення має значний соціальний ефект, оскільки дозволяє широким масам населення швидко отримувати інформацію про рівень забруднення повітря у навколишньому середовищі без необхідності докладно розбиратися у концентраціях різних небезпечних забруднюючих величин, що є задачею фахівців з екології. Знання ж рівня забруднення повітря

дуже важливо для того, щоби не піддавати здоров’я людей небезпеці: якщо в даний момент забруднення на вулиці є високим, то прогулянку краще відкласти на пізніше і перейти до видів діяльності всередині приміщень.

3.5 Висновки по розділу

Таким чином, у даному розділі розглянуто особливості програмної реалізації системи оцінки забрудненості навколишнього середовища. По-перше, обґрунтовано вибір програмних засобів для виконання реалізації і для більшої кросплатформенності вирішено використовувати мову JavaScript з основою на HTML. Через це, при виникненні потреби доробки цього продукту, розміщення його в мережі Інтернет буде простим процесом, що не потребує робочого часу через будь-яку переробку створеного ПЗ.

Розроблені вище алгоритми впроваджено у програмних кодах і на основі експертних даних проведено визначення коефіцієнтів рівняння регресії. Створене програмне забезпечення із вбудованою математичною моделлю піддано тестуванню, яке показало його стабільність (відсутність системних помилок при його роботі), а також належне виконання покладеної на нього функціональності: середнє відхилення показів програми відхиляється від еталонних даних на величини до 14%.

ВИСНОВКИ

Таким чином, у даній роботі було розроблено систему для аналізу забрудненості навколишнього середовища, а саме, показників повітря. Використання такої системи дозволяє адекватно реагувати (готуватися) до більших, чи менших рівнів забруднення (переносити вуличні заходи або роботи, планувати дозвілля дітей, і т.п.). Для цього у даній роботі було виконано наступні дії:

- проаналізовано існуючі види забруднень навколишнього середовища, та обґрунтувано, що саме забруднення повітря є найбільш небезпечним і таким, якому слід приділяти максимальну увагу;

- розглянуто існуючі підходи до аналізу та прогнозування рівнів забруднення, та обрано метод математичної статистики із розділу регресійного аналізу, а саме зведення лінійної множинної регресії;

- було розроблено відповідну математичну модель, причому на основі масиву експертних оцінок загального рівня забрудненості повітря по відповідним комбінаціям обраних п’яти частинних показників (CO, CO_2, O_3, SO_2, PM2.5) визначено коефіцієнти моделі;

- обрано засоби та технології розробки (структурний підхід до програмування, найбільш поширена на сьогодні мова програмування JavaScript, що потребує написання базового коду на HTML та, відповідно, CSS) та здійснено програмну реалізацію моделі;

- для розробленого програмного забезпечення проведено ручне системне тестування та встановлено стабільність його функціонуванняого ефективність (шляхом порівняння її оцінок із оцінками експертів, що відповідають невеликій частині початкового масиву вхідних даних, було встановлено середнє розходження порядку 14 %).

Таким чином, створена система може ефективно використовуватися для оцінки загального рівня забрудненості навколишнього середовища, а саме повітря, надаючи результат у зрозумілій та звичній 5-бальній шкалі.

В перспективі можливим видається залучення інших методів вироблення вихідного результату (на додачу до регресійного аналізу), зокрема, із галузі штучного інтелекту (як, наприклад, нейронна мережа прямого поширення сигналу абр нечітка система). Комбінування різних підходів може підвищити точність результатів, які видаються програмою, тому доцільність впровадження нових методів є обгрунтованою. Також, з технічної точки зору можна реалізувати підключення програми до бази даних, що також планується провести у подальшій роботі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Біляєв М. М. Моделювання і прогнозування стану довкілля: підручник для студентів вищих навчальних закладів / М. М. Біляєв, В. В. Біляєва, П. С.; Дніпропетровський національний університет залізничного транспорту імені академіка В. Лазаряна МОН України. - Кривий Ріг: Вид . Р. А. Козлов, 2016. - 207 с.

2 Бисчоков Р. М. Методы проведения анализа и прогнозирования изменения динамики агроклиматических ресурсов Северного Кавказа: монография. – Нальчик: КБГСХА, 2012. – 190 с.

3 Ковальчук П. І. Моделювання і прогнозування стану навколишнього середовища: Навч. Посібник. – К.: Либідь, 2003. – 208 с.

4 Попов О. О. Математичне моделювання розповсюдження техногенного забруднення від підприємств паливної енергетики / О. О. Попов. // Збірник наукових праць Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України. – К.: ІПМЕ ім. Г.Є. Пухова НАН України, 2009. – Вип. 51. – С. 73-84.

5 Олійник Р.В., Никонович С.О., Тарабан С.М. Моделювання оперативного прогнозування забруднення атмосферного повітря автотранспортними потоками // С.95-98.

6 Бідюк П.І., Савенков О.І., Трофимчук О.М.. Моделювання забруднення атмосфери та поверхневих вод в Україні // Екологічна безпека та природокористування, 2011 р. - С.16-34.

7 World's Air Pollution: Real-time Air Quality Index [Електронні дані]. - Режим доступу https: http://waqi.info/. – Заголовок з екрана.

8 A Beginner's Guide to Air Quality Instant-Cast and Now-Cast. [Електронні дані]. - Режим доступу https://aqicn.org/faq/2015-03-15/air-quality-nowcast-a-beginners-guide/]:. – Заголовок з екрана.

9 Марчук Г.И. Математическое моделирование в проблеме окружающей среды. – М.: Наука, 1982. – 319 с.

10 Царегородцев В.Г., Назимова Д.И. Нейросетевые анализ и моделирование современных связей климата и растительности // Современные подходы к интеграции информационных технологий. Труды V Всеросс. семинара "Информационные технологии в энергетике", Иркутск, 2000. Иркутск: ИСЭМ СО РАН, 2001. - С.157-165.

11 Басс Л.П., Кузьмина М.Г., Николаева О.В. Сверточные нейронные сети с глубоким обучением в задачах обработки гиперспектральных спутниковых данных // Препринты ИПМ им. М.В.Келдыша. 2018. № 282. – 32 с.

12 Bengio Y., Lamblin P., Popovici D., and Larochelle H.. Greedy layer-wise training of deep networks, in Proc. Neural Inf. Process. Syst., Cambridge, MA, USA, pp. 153–160, 2007.

13 Gangwar R. K., Mathur A. K., Gohil B. S., and Sujit Basu. Neural network based retrieval of atmospheric temperature profile using AMSU-A observations, International Journal of Atmospheric Sciences, vol. 2014, Article ID 763060, 8 p.

14 Goodfellow, Ian & Shlens, Jonathon & Szegedy, Christian. (2014). Explaining and Harnessing Adversarial Examples. arXiv 1412.6572.

15 Матвеев М. Г., Михайлов В. В., Семенов М. Е., Первезенцев Р. Е. Методические аспекты применения аппарата нечеткой логики при обработке и анализе метеорологической информации // Вестник Военного авиационного инженерного университета. – 2011. – №3 (14). - С. 35–42.

16 Тэрано Т., Асаи К., Сугэно М. Прикладные нечёткие системы. — М.: Мир, 1993. — 368 с.

17 Павловская Т.В. Процедурное и объектно-ориентированное программирование. Учебник. - М.: Питер, 2015. - 496 с.

18 Росс В. С. Создание сайтов: HTML, CSS, PHP, MySQL. Учебное пособие, М.:МГДД(Ю)Т, 2010. Ч.1. 107 с.

19 Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS. 2-е изд //СПб.: Питер. – 2013. – 201 с.

20 Выразительный Javascript / Под ред. Marijn Haverbeke - No Starch Press, 2015. С. 37-41.

21 JavaSpript. Подробное руководство / Под ред. Дэвид Флэнаган. СПб, СимволПлюс, 2008. – 923 с.

22 JavaScript. Шаблоны / Под ред. Стоян Стефанов - СПб, Символ-Плюс, 2011. С. 120-123.

Додаток А. Початковий текст програмного забезпечення для оцінки рівня забрудненості повітря (мовами JavaScript, HTML, CSS).

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Оцінка загального рівня забрудненості повітря</title>
<script>
const params=new Array("Концентрація CO, ppm","Концентрація CO2, ppm","Концентрація O3, ppm","Концентрація SO2, ppm","Концентрація PM2.5, мг/м3");

const startw=new Array(0.089,0.00004,0.502,0.085,0.0031);
const ro=new Array(100,1000,100,1,1);

let nump=10;

function filldiv()
{
      //document.getElementById("nump").value=nump;
   paramlist=document.getElementById("paramlist");
   paramlist.innerHTML="";
   let str="<table border=1 cellpadding=5 cellspacing=20>"
   for(let j=0;j<params.length;j++)
   {
```

```
            str+="<tr><td align=\"right\">"+params[j]+"</td><td><input type=\"checkbox\" id=\"chk"+j+"\" checked title=\"Снимите флажок, если данный показатель учитывать не нужно\"></td>";
            str+="<td><select id=\"sel"+j+"\" title=\"Измените, если считаете нужным, вес данного показателя, заданный по умолчанию\">";
            for(let i=0;i<10;i++)
            {
               str+="<option";
               if(startw[j]==(i+1))
                  str+=" selected";
               str+=">"+startw[j]+"</option>";
            }
            str+="</select></td>";
                     str+="<td>";
                     str+="<input type=\"text\" id=\"par"+j+"\" size=5 class=\"numbertoright\">";
                     str+="</td>";
            str+="</tr>";
         } str+="</table>";
         paramlist.innerHTML=str;
             //for(let i=0;i<nump;i++)

      //document.getElementById("ener"+i).addEventListener('change',checknum, false);
         for(let j=0;j<params.length;j++)

document.getElementById("par"+j).addEventListener('change',checknum,false);
      }
```

```
function startcalc()
{
   let maindiv=document.getElementById("maindiv");
        let sE=0,st=0,stE=0,st2=0,Ei,ti,Es,ts,wi,sw=0;
        let sum1=0,sum2=0,res,Eip1;
        let maxE=0,shift=50,shift2;
   for(let i=0;i<nump;i++)
   {
                Ei=+document.getElementById("ener"+i).value;
                if(Ei>maxE)
                        maxE=Ei;
                ti=(i+1);
                sE+=Ei;
                st+=ti;
                stE+=Ei*ti;
                st2+=ti*ti;
        }
        ts=st/nump;
        Es=sE/nump;
        sum1=(stE/nump-ts*Es)*(ti+1-ts)/(st2/nump-ts*ts)+Es;
        if(sum1>maxE)
                maxE=sum1;
        for(let j=0;j<params.length;j++)
        {
                ui=+document.getElementById("par"+j).value;
      wi=+document.getElementById("sel"+j).selectedIndex+1;
                sw+=wi;
                sum2+=ui/ro[j]*wi;
```

```
} sum2/=sw;
sum2+=1;
res=sum1*sum2;
if(res>maxE)
        maxE=res;

let can=document.getElementById("mycanvas");
let ctx=can.getContext("2d");
ctx.strokeStyle="red";
ctx.beginPath();
ctx.lineTo(100,0+shift);
ctx.lineTo(100,400+shift);
ctx.lineTo(500,400+shift);
ctx.moveTo(90,10+shift);
ctx.lineTo(100,0+shift);
ctx.lineTo(110,10+shift);
ctx.moveTo(490,390+shift);
ctx.lineTo(500,400+shift);
ctx.lineTo(490,410+shift);
ctx.moveTo(90,100+shift);
ctx.lineTo(110,100+shift);
ctx.moveTo(90,200+shift);
ctx.lineTo(110,200+shift);
ctx.moveTo(90,300+shift);
ctx.lineTo(110,300+shift);
ctx.fillText(Math.floor(maxE/4*4)+1,70,10+shift);
ctx.fillText(Math.floor(maxE/4*3)+1,70,110+shift);
ctx.fillText(Math.floor(maxE/4*2)+1,70,210+shift);
```

```
            ctx.fillText(Math.floor(maxE/4)+1,70,310+shift);
            for(let i=0;i<nump;i++)
        {
                ctx.fillText(""+(i+1),85+Math.floor(i*(500-
100)/nump),420+shift);
                ctx.moveTo(100+Math.floor(i*(500-100)/nump), 390+shift);
                ctx.lineTo(100+Math.floor(i*(500-100)/nump), 410+shift);
                Ei=+document.getElementById("ener"+i).value;
                if(i<nump-1)
                    Eip1=+document.getElementById("ener"+(i+1)).value;
                else
                    Eip1=res;
                ctx.moveTo(100+Math.floor(i*(500-100)/nump),          400-
Math.floor(Ei/maxE*400)+shift);
                ctx.lineTo(100+Math.floor((i+1)*(500-100)/nump),      400-
Math.floor(Eip1/maxE*400)+shift);
            }
            if(sum1>res)
                shift2=-5;
            else
                shift2=5;
            ctx.fillText("Традиционная                    модель",520,400-
Math.floor(sum1/maxE*400)+shift2+shift);
            ctx.moveTo(100+Math.floor((nump-1)*(500-100)/nump),   400-
Math.floor(Ei/maxE*400)+shift);
            ctx.lineTo(100+Math.floor(nump*(500-100)/nump),       400-
Math.floor(sum1/maxE*400)+shift);
            ctx.fillText("Предложенная                    модель",520,400-
Math.floor(res/maxE*400)-shift2+shift);
```

```
            ctx.stroke();

            maindiv.innerHTML="Прогнозоване забруднення:
"+Math.round(res)+"<br>Дякуємо за використання нашої системи!";
        }

        function checknum()
        {
           const ener=+this.value;
           if(!ener)
           {
              this.value="";
              alert("Вводіть, будь-ласка, тільки числові дані!");
           }
           else
           {
              if(this.id=="nump")
              {
                 nump=ener;
                 filldiv();
              }
           }
        }

        function fillE(e)
        {
            var E=e.target.responseText.split(';');
            nump=E.length;
            filldiv();
```

```
            for(let i=0;i<nump;i++)
                document.getElementById("ener"+i).value=E[i];
        }

        function ChangedConsumerSelect()
        {
            var index=document.getElementById("cons").selectedIndex;
            ConsumerSelected=document.getElementById("cons").options[index
.text;          ]

            var data=new FormData();
            data.append('consumer',ConsumerSelected);
            var req=new XMLHttpRequest();
            req.addEventListener('load',fillE,false);
            req.open("POST","getE.php",true);
        }       req.send(data);

        function bindhandlers()
        {

document.getElementById("docalc").addEventListener('click',startcalc,false);

document.getElementById("nump").addEventListener('change',checknum,false);
            document.getElementById("cons").addEventListener('change',Chang
e dConsumerSelect,false);
           filldiv();
        }

        window.addEventListener('load',bindhandlers,false);
```

```
</script>
<link rel="stylesheet" href="mycss.css">
</head>
<body>
<div id="maindiv">
<form>
<div id="paramlist">
</div>
<input type="button" value="Розрахувати!" id="docalc">
</form>
</div>
<canvas id="mycanvas" width="600" height="500"></canvas>
</body>
</html>
```