МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____
"_____"_____202_ р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: _____Засіб моніторингу технічного стану авіаційного обладнання

_____

_____

Виконавець: _____Подвойський О. І.

Керівник: _____Масловський Б. Г.

Консультанти з окремих розділів пояснювальної записки:

_____

_____

_____

Нормоконтролер: _____Тупота Є.В.

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук та технологій

Кафедра комп'ютеризованих систем управління

Спеціальність 123 «Комп'ютерна інженерія»

Спеціалізація «Системне програмування»

ЗАТВЕРДЖУЮ

Завідувач кафедри КСУ

Литвиненко О.Є.

« » 2022 р.

ЗАВДАННЯ

на виконання дипломної роботи (проекту)

Подвойського Олександра Ігоровича

(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема проекту (роботи): **Засіб моніторингу технічного стану авіаційного обладнання** затверджена наказом ректора від " 28 " серпня 2023 року № 1494/ст.

2. Термін виконання проекту (роботи): з 02.10.2023 до 31.12.2023

3. Вихідні дані до проекту (роботи): Правила збирання та перетворення визначальних параметрів. Правила використання мов програмування. Стандарти ДСТУ. Положення про дипломні роботи.

4. Зміст пояснювальної записки (перелік питань, що підлягають розробці):

1) Аналіз існуючих засобів і методів моніторингу визначальних параметрів.

2) Система моніторингу авіаційного обладнання.

3) Програмна реалізації алгоритмів роботи програми. Тестування отриманої програми

5. Перелік обов'язкового графічного матеріалу:

1) Структура системи моніторингу технічного стану АО;

2) Алгоритм роботи моніторингу стану АО;

3) Алгоритму навчання та зберігання даних моделі;

4) Діаграми трирівневого архітектурного різновиду ситеми;

5) Скріншоти тестування програми;

6. Календарний план

№ пор.	Етапи виконання дипломної роботи	Термін виконання етапів
1	Ознайомитись з постановкою задачі дипломного проектування	02.10.2023 – 04.10.2023
2	Вивчити спеціальну літературу і технічну документацію	05.10.2023 – 10.10.2023
3	Проведення аналізу існуючих засобів і методів моніторингу АО.	11.10.2023 – 25.10.2023
4	Проведення моделювання структури, алгоритмів роботи сайту.	26.10.2023 – 02.11.2023
5	Розробка алгоритмів роботи програми моніторингу	03.11.2023 – 17.11.2023
6	Проведення програмної реалізації розроблених алгоритмів. Тестування розроблених програм.	18.11.2023 – 01.12.2023
7	Оформлення пояснювальної записки т обов’язкового графічного матеріалу	02.12.2023 – 15.12.2023
8	Підготовка доповіді та презентації	16.12.2023 – 18.12.2023

7. Дата видачі завдання _____

Керівник дипломної роботи _____Масловський Б. Г.
(підпис) (П.І.Б.)

Завдання прийняв до виконання _____Подвойський О. І.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Засіб моніторингу технічного стану авіаційного обладнання»: 113 с., 46 рис., 2 дод., 17 літературних джерел.

Ключові слова: АВІАЦІЙНЕ ОБЛАДНАННЯ, ТЕХНІЧНИЙ СТАН, МОНІТОРИНГ, ДІАГНОСТИКА, АВТОМАТИЗОВАНІ СИСТЕМИ МОНІТОРИНГУ, СИСТЕМИ ПРОГНОЗУВАННЯ ВІДМОВИ ОБЛАДНАННЯ.

Актуальність: Актуальність даної робботи полягає у необхідності розробки нових методів та технологій для моніторингу технічного стану авіаційного обладнання, які б забезпечували більш високий рівень безпеки, надійності та економічної ефективності.

Мета: розробка оптимізованого та надійного засобу моніторингу технічного стану авіаційного обладнання, що забезпечує підвищення рівня безпеки польотів, економічної ефективності та відповідність міжнародним стандартам.

Об'єкт дослідження: процес моніторингу технічного стану авіаційного обладнання.

Предмет дослідження: є методи та алгоритми, які використовуються в процесі моніторингу для оцінки та прогнозування технічного стану авіаційного обладнання.

Методи дослідження: аналітичний, математичне моделювання, статистичний аналіз, алгоритми машинного навчання, експериментальний метод.

Результати магістерської роботи рекомендується використовувати під час проведення наукових досліджень та в практичній діяльності фахівців з технічного обслуговування авіаційної техніки.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ. ОДИ–НИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АО – Авіаційне обладнання.

ШД – швидке дерево (Fast Tree), алгоритм машинного навчання, який використовується для задач класифікації та регресії.

ШІ – штучний інтелект, галузь комп'ютерних наук, що займається створенням машин, здатних виконувати завдання, які зазвичай потребують людського інтелекту.

C# – об'єктно–орієнтована мова програмування, розроблена компанією Microsoft як частина платформи .NET.

Fast Tree – алгоритм машинного навчання для роботи з задачами класифікації та регресії, відомий своєю швидкістю та ефективністю.

ML – машинне навчання (Machine Learning), наукова дисципліна, яка займається дослідженням та розробкою алгоритмів, здатних навчатися та робити прогнози на основі даних.

ML.NET – фреймворк машинного навчання для .NET, що дозволяє розробникам легко інтегрувати функціональність машинного навчання у свої .NET додатки.

SQL – мова програмування структурованих запитів (Structured Query Language), використовується для управління та маніпуляції даними в базах даних.

VS – Visual Studio, інтегроване середовище розробки від компанії Microsoft, використовуване для створення програмного забезпечення.

ВСТУП

Авіаційна промисловість відіграє критичну роль в сучасному світі, стаючи сектором, де надійність, безпека та ефективність є абсолютними пріоритетами. Моніторинг технічного стану авіаційного обладнання є ключовою задачею, яка безпосередньо впливає на безпеку польотів та економічну віддачу авіакомпаній. Зростання обсягів авіаційних перевезень, посилення міжнародних та національних стандартів безпеки, а також збільшення складності авіаційних систем зробили цю проблему ще більш актуальною.

Існуючі системи моніторингу, як правило, є реактивними і здатні виявляти проблеми лише після їх виникнення. Такий підхід може призвести до небезпечних ситуацій, аж до катастрофічних наслідків, таких як аварії чи життєво небезпечні інциденти. Проактивні методи прогнозування та виявлення можливих збоїв на ранніх етапах, хоча і обіцяють певні переваги, часто є фінансово недоступними або технічно незрілими для широкого внедрення в авіаційній галузі.

Враховуючи ці обмеження, дослідження та розробка нових, більш ефективних та економічно виправданих методів моніторингу є першочерговим завданням. Це не тільки підвищить рівень безпеки авіаційних перевезень, але і забезпечить конкурентні переваги для авіакомпаній, сприяючи їх стабільному економічному розвитку. Нові методики можуть також послужити основою для оновлення національного та міжнародного законодавства в області авіаційної безпеки, що є особливо важливим для України в контексті її амбіцій щодо модернізації авіаційної галузі та інтеграції в міжнародні авіаційні системи.

Актуальність даної теми полягає у необхідності розробки нових методів та технологій для моніторингу технічного стану авіаційного обладнання, які б забезпечували більш високий рівень безпеки, надійності та економічної ефективності. Це стає особливо важливим для України, яка активно розвиває свою авіаційну інфраструктуру та прагне інтегруватися в міжнародні авіаційні стандарти.

Незважаючи на численні наукові дослідження та практичні розв'язки в цій галузі, існують численні прогалини в системах моніторингу, які вимагають

подальших наукових розробок. Порівняння із відомими розв'язаннями показує, що багато з них або занадто специфічні, або не забезпечують необхідний рівень універсальності та адаптивності до змінних умов експлуатації.

У контексті національної безпеки та розвитку авіаційної промисловості України, проблема моніторингу технічного стану авіаційного обладнання набуває стратегічного значення. Це обумовлено не тільки зростаючими обсягами пасажирських та вантажних перевезень, але і потребою в сучасних і надійних системах моніторингу для забезпечення відповідності міжнародним стандартам безпеки.

Метою даного дослідження є розробка оптимізованого та надійного засобу моніторингу технічного стану авіаційного обладнання, що забезпечує підвищення рівня безпеки польотів, економічної ефективності та відповідність міжнародним стандартам.

Для досягнення вищевказаної мети необхідно вирішити наступні задачі:

– аналіз існуючих методів і систем моніторингу. Вивчення та критичний аналіз існуючих методологій та технічних рішень у галузі моніторингу технічного стану авіаційного обладнання;

– оцінка параметрів та критеріїв безпеки. Визначення ключових параметрів та критеріїв, які впливають на безпеку польотів та ефективність моніторингових систем;

– розробка математичної моделі. Створення математичної моделі для оцінки та прогнозування технічного стану авіаційного обладнання на основі зібраних даних та параметрів;

– розробка алгоритмів аналізу даних. Розробка та тестування алгоритмів для аналізу даних, отриманих від сенсорів та інших джерел;

– прототипування системи. Створення прототипу засобу моніторингу на основі розроблених моделей та алгоритмів;

– тестування та валідація. Проведення комплексних тестів для перевірки надійності, точності та ефективності пропонованого засобу моніторингу.

Об'єктом дослідження є процес моніторингу технічного стану авіаційного обладнання. Цей процес є комплексним та багатоаспектним, охоплюючи такі елементи як збір даних з сенсорів, аналіз цих даних, прогнозування можливих відмов та збоїв, а також впровадження коректних дій для підтримки оптимального технічного стану обладнання.

Предметом дослідження є методи та алгоритми, які використовуються в процесі моніторингу для оцінки та прогнозування технічного стану авіаційного обладнання. Це у свою чергу включає математичну модель для аналізу даних з сенсорів, алгоритми машинного навчання для прогнозування потенційних відмов, системи для автоматичного виявлення аномалій, а також інтерфейси та протоколи для взаємодії між різними компонентами системи моніторингу.

Для досягнення поставленої мети в роботі було використано комплексний підхід, що включає в себе ряд методологічних інструментів:

– аналітичний метод. Використано для систематизації та критичного аналізу наукової літератури, статистичних даних, технічної документації та інших джерел інформації для вивчення проблеми моніторингу технічного стану авіаційного обладнання;

– метод математичного моделювання. Використано для розробки математичної моделі, яка описує динаміку та характеристики авіаційного обладнання;

– статистичний аналіз. Використано для обробки емпіричних даних, зокрема для визначення закономірностей, трендів та аномалій в даних, отриманих з сенсорів та інших систем моніторингу;

– алгоритми машинного навчання. Використано для аналізу великих обсягів даних та прогнозування потенційних відмов та збоїв в роботі обладнання;

– експериментальний метод. Включає в себе створення прототипу системи, її тестування та валідація в контрольованих умовах.

Комбінація цих методів дослідження дозволила розробити комплексний та багатогранний підхід до проблеми моніторингу технічного стану авіаційного обладнання, що відповідає актуальним вимогам та стандартам.

В результаті проведеного дослідження було розроблено ряд нових положень і рішень, які представляють наукову новизну:

– оптимізована математична модель для моніторингу. Вперше була розроблена математична модель, яка враховує динамічні характеристики авіаційного обладнання та покращує точність прогнозування їх технічного стану;

– адаптивні алгоритми машинного навчання. Удосконалено алгоритми машинного навчання для аналізу даних з сенсорів, що забезпечують адаптивність до змінних умов експлуатації та підвищують ефективність виявлення аномалій.

Отримані в ході дослідження результати мають високе практичне значення і можуть бути застосовані в ряді ключових областей авіаційної промисловості:

– підвищення рівня безпеки польотів. Розроблена система моніторингу може бути впроваджена в цивільній та військовій авіації для забезпечення вищого рівня безпеки через раннє виявлення та прогнозування технічних збоїв;

– економічна ефективність. Введення економічного модуля оцінки ефективності дозволяє авіакомпаніям оптимізувати витрати на обслуговування та ремонт, знижуючи при цьому ризики несправностей;

– стандартизація та уніфікація. Розроблені методи і рекомендації можуть слугувати основою для створення національних та міжнародних стандартів у галузі моніторингу технічного стану авіаційного обладнання;

– наукове використання. Методи та алгоритми, розроблені в ході дослідження, можуть бути використані в академічних дослідженнях для подальшого розвитку теорії моніторингу технічних систем;

– промислове впровадження. Розроблені методики та засоби можуть бути безпосередньо впроваджені на підприємствах авіаційної промисловості для оптимізації процесів обслуговування та управління якістю.

РОЗДІЛ 1
СУЧАСНІ МЕТОДИ ТА ЗАСОБИ МОНІТОРИНГУ АВІАЦІЙНОГО ОБЛАДНАННЯ

1.1 Класифікація систем для моніторингу авіаційного обладнання

Системи моніторингу технічного стану авіаційного обладнання відіграють ключову роль в забезпеченні безпеки польотів, ефективності використання ресурсів і оптимізації експлуатаційних витрат [1]. Для реалізації цих завдань необхідно детально класифікувати обладнання, що підлягає моніторингу, та визначити специфікації для кожного типу обладнання (рис. 1.1).

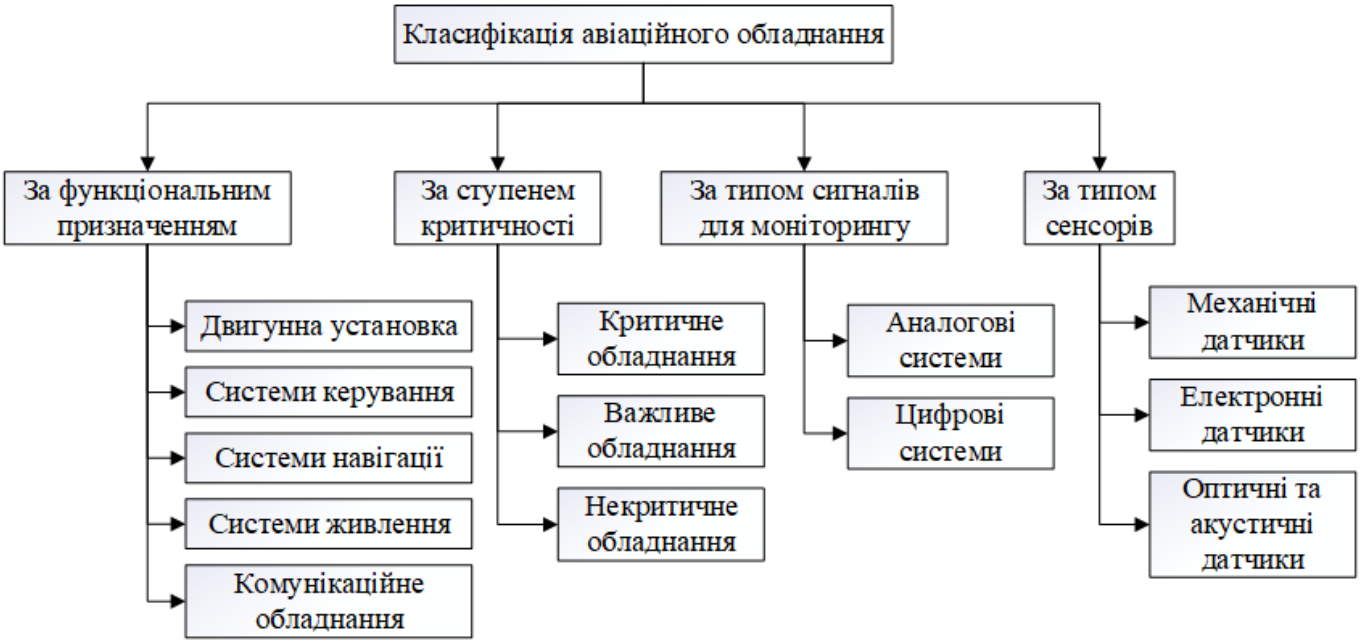

Рис 1.1. Класифікація авіаційного обладнання

Як видно із рис. 1.1 класифікація авіаційного обладнання для систем моніторингу виглядає наступним чином:

За функціональним призначенням

Функціональне призначення є одним з критеріїв, на основі якого проектуються, виробляються та експлуатуються технічні системи, в тому числі авіаційне обладнання. Це поняття визначає основні завдання, які система або її компонент

повинен виконувати для досягнення поставлених цілей [2]. Для авіаційної техніки функціональне призначення може бути дуже різнорідним: від забезпечення тяги і управління літаком до навігації, комунікації та забезпечення живлення всіх бортових систем. Розуміння функціонального призначення дозволяє оптимізувати процеси обслуговування, підвищити надійність та безпеку експлуатації.

До даного критерію відносяться:

– двигунна установка. Двигунна установка є однією з найважливіших складових літального апарату, що включає в себе не лише основні двигуни, але й турбіни, вентилятори, системи охолодження, паливні системи та інші супутні елементи. Моніторинг цих компонентів вимагає високої точності та оперативності, оскільки будь–який збій або несправність може мати критичні наслідки [3]. Датчики високого температурного та тискового діапазону, а також системи аналізу вібрацій, зазвичай використовуються для контролю стану двигунної установки;

– навігаційне обладнання. Системи навігації в авіації включають в себе не тільки традиційні GPS–системи, але й радари, датчики висоти, швидкості, кутової орієнтації та інші [4]. Ці системи відіграють ключову роль у забезпеченні безпеки польотів, дозволяючи екіпажу та автоматичним системам керування отримувати актуальну інформацію про положення літака в просторі та його взаємодію з іншими об'єктами;

– керувальні системи. Системи керування літаком є складними інтегрованими комплексами, що включають гідро – та електросистеми, системи стабілізації та автопілоти. Ці системи надзвичайно взаємозалежні і потребують високої ступені координації для забезпечення безпеки та ефективності польоту [5]. Системи моніторингу в цій категорії часто використовують алгоритми машинного навчання для аналізу великих обсягів даних та виявлення потенційних проблем на ранніх стадіях;

– системи живлення. Системи живлення в авіації включають електричні генератори, батареї та системи зарядки. Ці системи забезпечують енергію для всіх електричних систем на борту, від освітлення та кондиціонування до навігаційного та

комунікаційного обладнання [6]. Моніторинг їх стану включає в себе відстеження параметрів як напруги, струму, а також стану батарей;

– комунікаційне обладнання. Комунікаційне обладнання в авіації включає в себе різноманітні системи, від базових радіостанцій до складних транспондерів і систем зв'язку з наземними станціями. Це обладнання є критично важливим для забезпечення безпеки польотів, оскільки воно дозволяє літаку взаємодіяти з наземними службами та іншими літальними апаратами [7].

Кожна з цих категорій має свої унікальні вимоги до моніторингу та діагностики, що вимагає специфічного підходу при розробці систем моніторингу технічного стану.

За ступенем критичності

Класифікація авіаційного обладнання за ступенем критичності дозволяє встановити пріоритети у процесах обслуговування та ремонту, а також при виборі методів діагностики. Обладнання класифікується як критичне, важливе та некритичне з огляду на можливий вплив його відмови на безпеку польотів та ефективність експлуатації. Від цієї класифікації безпосередньо залежать також стратегії реагування на можливі інциденти та розробка планів профілактичних заходів. Розуміння ступеня критичності кожного компоненту є необхідним для оптимізації ресурсів та підвищення загальної надійності авіаційних систем.

До даного критерію відносяться:

– критичне обладнання. Критичне обладнання в авіаційній техніці – це те, відмова якого може мати безпосередньо катастрофічні наслідки для польоту. Це включає, наприклад, двигунні установки, системи керування та навігаційне обладнання. Таке обладнання вимагає найвищого рівня надійності та безпеки, а моніторинг його стану здійснюється з використанням найбільш передових технологій [8]. Відмова такого обладнання вимагає негайної активізації систем аварійного реагування;

– важливе обладнання. Важливе обладнання – це компоненти, відмова яких може не призвести до катастрофи, але все ж вимагає негайної реакції з боку екіпажу або автоматичних систем. Приклади включають деякі електричні системи, системи кондиціонування повітря, та деяке комунікаційне обладнання. Хоча таке обладнання

і не є критичним для безпосередньої безпеки польоту, його відмова може серйозно ускладнити управління літаком або зробити неможливим виконання певних маневрів.

– некритичне обладнання. Некритичне обладнання в авіації – це те, відмова якого не має прямого впливу на безпеку польоту. Це може включати обладнання для зручності пасажирів, таке як системи розваги, або допоміжне обладнання, наприклад, системи освітлення в кабіні. Хоча відмова такого обладнання не є критичною, правильна його робота сприяє загальному комфорту та задоволенності пасажирів, що в кінцевому підсумку може мати економічну цінність для авіакомпаній.

Класифікація обладнання за ступенем критичності дозволяє авіакомпаніям та сервісним організаціям більш точно планувати процеси обслуговування, встановлювати пріоритети в рамках програм моніторингу стану, а також розробляти стратегії для оптимізації експлуатаційної ефективності та безпеки.

За типом сигналів для моніторингу

Класифікація авіаційного обладнання за типом сигналів для моніторингу є важливим аспектом в організації системи контролю технічного стану. Залежно від вимог до точності, швидкості реакції та інших оперативних параметрів, можуть використовуватися аналогові або цифрові системи [9]. Аналогові системи, зазвичай, простіші та надійні, але можуть бути обмежені в плані аналітичних можливостей. Натомість, цифрові системи надають більш широкі можливості для аналізу та прогнозування, включаючи використання алгоритмів машинного навчання. Вибір типу сигналів для моніторингу напряму впливає на ефективність системи контролю та може бути критичним для забезпечення безпеки польотів.

До типів сигналів для моніторингу відносяться:

– аналогові системи. Аналогові системи моніторингу в авіації часто використовують термодатчики, датчики тиску та інші сенсори, які вимірюють фізичні параметри в неперервному діапазоні значень [10]. Ці системи зазвичай є простішими за своєю конструкцією, але можуть вимагати додаткового обладнання для перетворення аналогових сигналів в цифровий формат для подальшого аналізу та обробки. Хоча аналогові системи можуть бути менш витонченими в плані

діагностичних можливостей, вони зазвичай дуже надійні та стійкі до різних видів збоїв та відмов;

– цифрові системи. Цифрові системи моніторингу базуються на використанні мікроконтролерів, програмного забезпечення та інших високотехнологічних рішень [11]. Вони здатні обробляти великі обсяги даних, здійснювати комплексний аналіз стану обладнання та навіть використовувати алгоритми машинного навчання для прогнозування можливих відмов. Цифрові системи дозволяють інтегрувати різноманітні джерела інформації та надають можливість дистанційного моніторингу та управління.

Обидва типи систем – аналогові та цифрові – мають свої переваги та недоліки, і вибір між ними залежить від конкретних вимог до надійності, точності моніторингу та інших оперативних факторів. У багатьох сучасних авіаційних системах використовується комбінація обох типів сигналів для забезпечення найвищого рівня ефективності та безпеки.

За типом сенсорів

Класифікація авіаційного обладнання за типом сенсорів є ключовим елементом в системах моніторингу технічного стану. Залежно від конкретних експлуатаційних вимог та параметрів, що вимірюються, можуть застосовуватися механічні, електронні, оптичні або акустичні датчики [12]. Механічні датчики зазвичай використовуються для вимірювання фізичних величин, таких як температура або тиск, і є відносно простими в установці та експлуатації. Електронні та оптичні датчики, з іншого боку, можуть надавати більш деталізовану інформацію та інтегруватися з цифровими системами для комплексного аналізу. Вибір типу сенсорів може суттєво вплинути на точність, надійність та ефективність системи моніторингу.

До типів сенсорів відносяться:

– механічні датчики. Механічні датчики в авіаційних системах моніторингу включають в себе датчики тиску, температури, вібрації та інші, які здійснюють пряме вимірювання фізичних величин. Ці датчики можуть бути вбудовані в двигунні установки, системи керування, гідравлічні системи та інше обладнання. Вони

відзначаються високою надійністю та можуть працювати в екстремальних умовах, таких як високі температури чи сильні вібрації;

– електронні датчики. Електронні датчики спеціалізуються на моніторингу електричних параметрів, таких як напруга, струм, опір. Вони можуть бути використані для контролю стану електричних систем, таких як генератори, батареї та системи живлення. Ці датчики можуть бути інтегровані в цифрові системи моніторингу для реалізації комплексного аналізу та діагностики;

– оптичні та акустичні датчики. Оптичні та акустичні датчики використовуються для моніторингу параметрів за допомогою оптичних та акустичних сигналів. Наприклад, оптичні датчики можуть бути використані для визначення якості масла в двигуні або для виявлення тріщин на поверхні матеріалів. Акустичні датчики можуть бути використані для виявлення аномалій у роботі двигуна чи інших механічних систем через аналіз звукових характеристик.

Кожен тип сенсорів має свої специфічні переваги та області застосування, і вибір конкретного типу залежить від ряду факторів, включаючи вимоги до точності, надійності та специфіки експлуатаційних умов.

Проведена класифікація дозволяє ефективно організувати процес моніторингу технічного стану авіаційного обладнання, вибирати оптимальні методи аналізу та прогнозування для різних типів обладнання, а також розробляти цільові рекомендації для підвищення надійності та безпеки авіаційних систем.

1.2 Основні параметри та характеристики технічного стану авіаційного обладнання

Моніторинг технічного стану авіаційного обладнання вимагає системного підходу до вимірювання та аналізу широкого спектру параметрів і характеристик (рис. 1.2). Ці параметри та характеристики відіграють ключову роль у забезпеченні

високого рівня безпеки польотів, ефективності роботи авіаційних систем та їх надійності в експлуатації.

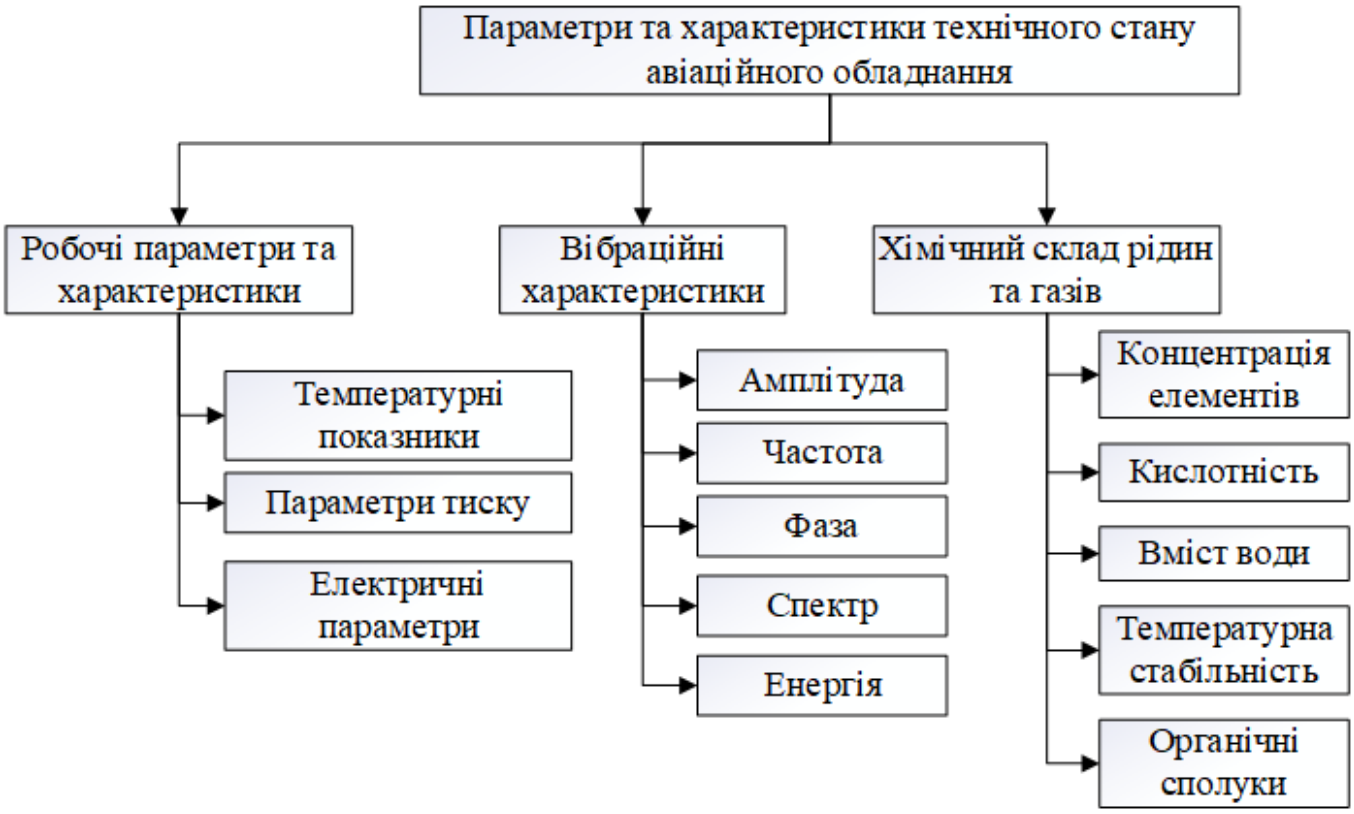

Рис 1.2. Параметри та характеристики технічного стану авіаційного обладнання

Для забезпечення цілісності даного процесу, розглядаються різні типи параметрів, до яких можна віднести, зокрема:

Робочі параметри та характеристики:

Основні робочі параметри та характеристики є інтегрованою сукупністю ключових показників, які визначають стан та ефективність роботи авіаційного обладнання. Ця категорія включає в себе такі параметри, як температура, тиск, електричні характеристики, швидкість, вібрація та інші, які безпосередньо або опосередковано впливають на безпеку, надійність та ефективність польотів [13]. Моніторинг цих параметрів є важливим для виявлення і виправлення потенційних проблем або несправностей перед тим, як вони призведуть до серйозних інцидентів. Основні робочі параметри та характеристики служать як фундамент для аналітичних моделей та систем прогнозування, що дозволяють автоматизувати процеси діагностики та управління технічним станом.

До основних робочих параметрів та характеристик можна віднести:

– температурні показники. Температура виступає як один з критичних параметрів в системах моніторингу технічного стану авіаційного обладнання. Вона безпосередньо впливає на робочі характеристики двигунів, систем охолодження, гідравлічних систем та інших механічних компонентів. Неконтрольовані або аномальні зміни температурних показників можуть сигналізувати про ряд потенційних проблем, включаючи знос матеріалів, недостатнє охолодження, або навіть можливість виникнення пожежі чи вибуху. Тому системи моніторингу температури мають бути здатні забезпечувати точні та оперативні дані для своєчасного виявлення та усунення аномалій;

– параметри тиску. Моніторинг параметрів тиску є не менш важливим аспектом оцінки технічного стану авіаційного обладнання. Це особливо актуально для гідравлічних систем, систем паливоподачі, а також для пневматичних систем. Нестабільність тиску може привести до неправильної роботи важливих систем та компонентів, що, у свою чергу, може загрожувати безпеці польоту. Отже, сенсори та системи моніторингу тиску повинні бути високочутливими, точними та надійними для забезпечення найвищого рівня безпеки;

– електричні параметри. Електричні параметри такі, як напруга, струм та опір, відіграють важливу роль в стабільності та надійності авіаційного обладнання. Ці параметри є ключовими для систем електроживлення, які живлять критичні для безпеки системи, такі як навігаційні системи, системи керування та екстрені системи. Аномалії або відхилення в електричних параметрах можуть вказувати на серйозні несправності в електричних кіл або компонентах, які можуть привести до системних збоїв. Тому, електричні системи моніторингу повинні мати високу точність, швидкість відгуку та можливість інтеграції з іншими системами для всебічного аналізу.

Вібраційні характеристики

Вібраційні характеристики є однією з ключових категорій параметрів, яка забезпечує деталізований аналіз стану механічних систем та компонентів в авіаційному обладнанні. Спеціалізовані датчики вібрації здатні вимірювати амплітуду, частоту та фазу вібраційних процесів. Ці дані є вкрай важливими для

раннього виявлення проблем в двигунах, турбінах, гідравлічних насосах та інших обертових механізмах. Наприклад, зміна вібраційних характеристик може вказувати на незбалансованість, знос лопаток, дефекти підшипників або інші механічні несправності, що вимагають негайного втручання.

Для адекватного моніторингу вібраційних характеристик авіаційного обладнання важливо зосередитися на декількох ключових параметрах, які можуть надати всебічне розуміння стану механічних систем [14]:

– амплітуда вібрації. Це максимальне відхилення об'єкта від його рівноважного положення під час одного циклу вібрації. Амплітуда є важливим показником інтенсивності вібрацій;

– частота вібрації. Це кількість циклів вібрації за одиницю часу. Вона часто вимірюється в герцах (Hz) і може служити індикатором різних типів механічних проблем;

– фаза вібрації. Це параметр, який описує зсув у часі між вібрацією об'єкта та певною довільно вибраною початковою точкою. Фазові зсуви між різними частинами механізму можуть вказувати на проблеми в їх взаємодії;

– спектр вібрації. Це розподіл амплітуди та фази вібрації в залежності від частоти. Аналіз спектру може виявити специфічні частотні складові, які вказують на певні типи несправностей;

– енергія вібрації. Це інтегрований показник, який враховує амплітуду та частоту вібрації для оцінки загальної енергії вібраційного процесу. Збільшення енергії вібрації може сигналізувати про збільшення навантаження на механічну систему.

Ці параметри можуть бути виміряні та аналізовані за допомогою спеціалізованих вібраційних датчиків та аналітичного програмного забезпечення, що дозволяє оперативно ідентифікувати аномалії та планувати профілактичні дії.

Хімічний склад рідин та газів

Моніторинг хімічного складу рідин та газів в авіаційному обладнанні є іншою важливою ділянкою, яка може надати важливу інформацію про технічний стан систем. Аналіз хімічного складу, зокрема масел, палива, гідравлічних рідин та

охолоджувальних агентів, може виявити присутність забруднюючих речовин, окислів, води або інших небажаних елементів. Ці забруднення можуть призвести до корозії, зносу матеріалів або навіть до хімічних реакцій, які можуть бути небезпечними. Системи моніторингу хімічного складу рідин та газів мають бути високочутливими та точними, щоб своєчасно виявляти такі забруднення та запобігти можливим негативним наслідкам.

Декількома ключовими параметрами у цьому контексті можуть бути:

– концентрація елементів. Вимірювання концентрації окремих хімічних елементів, таких як сірка, фосфор, залізо, може допомогти виявити знос металічних компонентів або наявність забруднюючих речовин;

– кислотність (pH). Зміни рівня pH можуть вказувати на хімічні реакції у системі, які можуть призвести до корозії або інших форм зносу;

– вміст води. Високий вміст води в оливах та паливі може призвести до корозії та інших проблем;

– температурна стабільність. Деякі хімічні компоненти можуть розкладатися при високих температурах, випускаючи шкідливі або корозійні продукти;

– органічні сполуки. Наявність органічних сполук, таких як бензол або толуол, може вказувати на забруднення або на витік палива чи мастила;

– окислення та антиоксидантна активність. Рівень окислення рідини може вказувати на її старіння або наявність окислювачів, що можуть призвести до швидкого зносу.

Ці параметри можуть бути виміряні за допомогою різних методів хімічного аналізу, таких як хроматографія, спектроскопія, титрометрія та інші. Вони надають цінну інформацію для раннього виявлення проблем, що можуть призвести до несправностей або аварій, і дозволяють проводити своєчасне технічне обслуговування.

Кожний з цих параметрів та характеристик має свої методи вимірювання та діагностики, і їх адекватний моніторинг є ключовим для своєчасного виявлення та усунення потенційних проблем. Таким чином, системи моніторингу технічного стану

авіаційного обладнання повинні бути комплексними, інтегрованими та здатними до аналізу великої кількості різноманітних даних для забезпечення найвищого рівня безпеки та ефективності експлуатації.

1.3 Сучасні методи та засоби моніторингу авіаційного обладнання

Обслуговування технічних засобів «за станом» – це сучасний підхід до управління технічним обслуговуванням, який базується на постійному моніторингу та аналізі визначальних параметрів обладнання [15]. Цей метод дозволяє визначати реальний технічний стан обладнання та виконувати технічне обслуговування або ремонт лише тоді, коли це дійсно необхідно, замість традиційних планових профілактичних робіт. На рис. 1.3 зображено алгоритм перевірки тех. засобів «за станом»

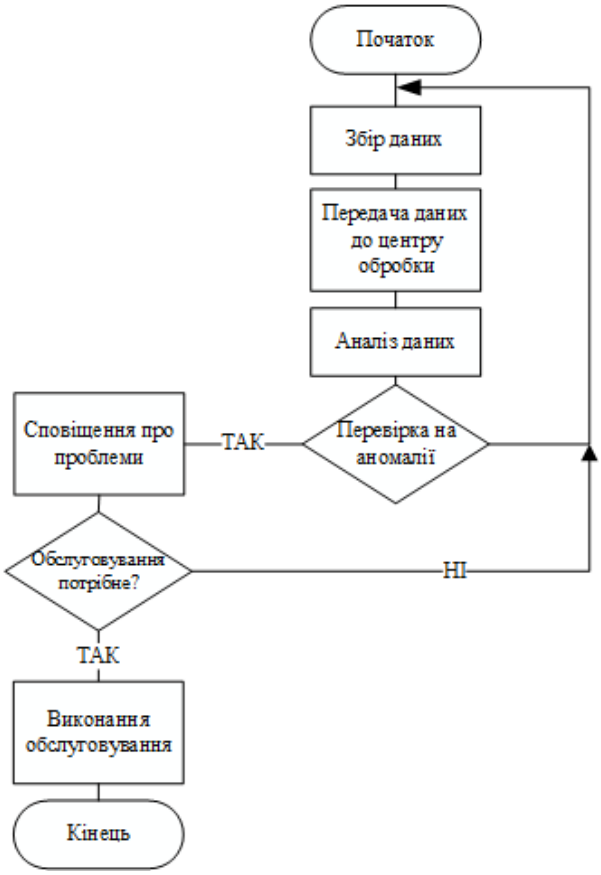

Рис 1.3. Алгоритм перевірки тех. засобів «за станом»

Обслуговування технічних засобів «за станом» починається зі збору даних про стан обладнання за допомогою датчиків. Ці дані передаються до центру обробки, де вони аналізуються на предмет виявлення аномалій або проблем. Якщо аномалії виявлені, система генерує сповіщення для персоналу. На основі цих сповіщень персонал приймає рішення про необхідність обслуговування чи ремонту. Якщо обслуговування необхідне, планується та виконується відповідна робота. Після завершення робіт або якщо обслуговування не потрібне, система повертається до збору даних, продовжуючи цикл моніторингу. Алгоритм завершується при відключенні системи або за інших специфічних умов.

Цей алгоритм дозволяє забезпечити постійний контроль стану обладнання та своєчасно реагувати на будь–які проблеми, мінімізуючи простої та збільшуючи надійність технічних засобів.

Для реалізації такого підходу необхідно впровадити систему моніторингу, яка здатна збирати, обробляти та аналізувати дані про визначальні параметри обладнання. Ці параметри можуть включати, наприклад, температуру, вібрації, енергоспоживання, а також інші індикатори, що можуть свідчити про стан обладнання.

Система моніторингу має включати датчики для збору даних, обчислювальні можливості для обробки та аналізу цих даних та інтерфейси для звітності та сповіщення. Використання технологій машинного навчання та штучного інтелекту може значно підвищити ефективність такої системи, дозволяючи прогнозувати потенційні несправності та оптимізувати графіки обслуговування.

Переваги обслуговування за станом полягають у зниженні витрат, підвищенні надійності та доступності обладнання, а також у покращенні безпеки. Оскільки обслуговування проводиться на основі реального стану обладнання, це дозволяє уникати непотрібних перерв у роботі та зменшувати ризики виникнення несподіваних збоїв.

Такий підхід вимагає інтегрованого підходу до проектування систем моніторингу, а також постійного оновлення та вдосконалення алгоритмів аналізу даних для забезпечення максимальної точності та ефективності системи.

1.4 Сучасні методи та засоби моніторингу авіаційного обладнання

Сучасні методи та техніки моніторингу є комплексними та багаторівневими, вони забезпечують високий рівень надійності та безпеки в авіаційному секторі. Це, в свою чергу, є важливою складовою для забезпечення безпеки польотів, ефективності використання ресурсів та стратегічного розвитку авіаційної індустрії в цілому.

1.4.1 Переваги та недоліки сучасних систем моніторингу

Перед розробкою програми доцільним є вивчення аналогічних програмних рішень.

Atennea Air

Atennea Air є методологією та набором технологій для моніторингу та аналізу даних польотів з метою підвищення рівня безпеки авіаційних операцій. Важливо зазначити, що FDM не є конкретним продуктом чи системою, а загальним підходом, який може бути реалізований за допомогою різних технологій та програмних рішень.

Розробниками цієї систем є такі компанії як Honeywell, GE Aviation та Teledyne. Ці компанії спеціалізуються на авіаційних технологіях і часто пропонують комплексні рішення для моніторингу даних польотів.

FDM широко використовується в комерційній авіації для аналізу показників безпеки польотів, для оцінки ефективності пілотажу, для дотримання стандартів та регулятивних вимог, а також для загального підвищення якості авіаційних операцій. Крім комерційних авіаліній, системи FDM також використовуються в корпоративній авіації, військовій авіації, а також в загальній авіації для навчання та дослідження.

Зазвичай «Atennea Air» система збирає дані з бортових датчиків та інших систем моніторингу під час польоту, а потім передають ці дані на наземну станцію для подальшого аналізу. Тут може відбуватися детальний аналіз даних, включаючи

виявлення аномалій, оцінку показників безпеки та рекомендації щодо можливих поліпшень.

Інтерфейс «Atennea Air» представлено на рис. 1.3.

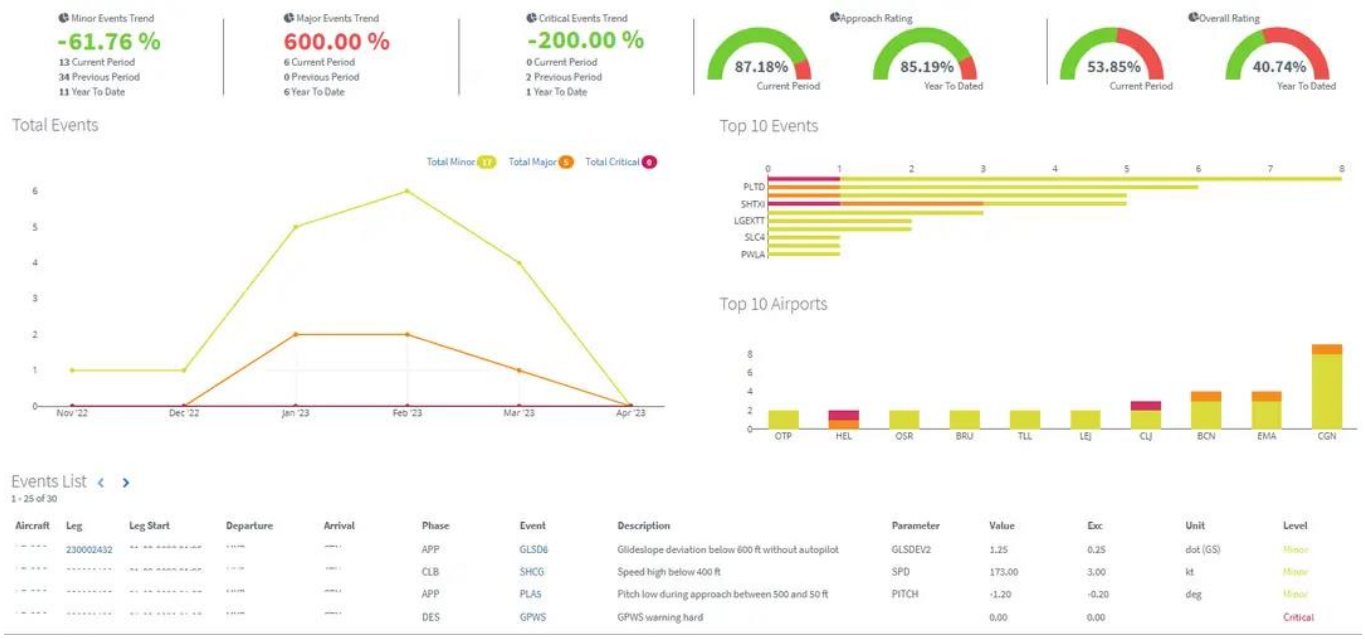

Рис 1.3. Інтерфейс користувача системи «Atennea Air»

Переваги системи «Atennea Air» [16]:

– підвищення безпеки польотів. Завдяки збору та аналізу даних про роботу авіаційного обладнання та дії екіпажу, система дозволяє ідентифікувати потенційні ризики та проблеми до того, як вони призведуть до серйозних інцидентів;

– оптимізація експлуатаційної ефективності. Аналіз даних може допомогти авіакомпаніям в підвищенні ефективності польотів, включаючи зниження витрат на паливо та оптимізацію маршрутів;

– забезпечення дотримання стандартів та регуляцій. Система може автоматично виявляти порушення авіаційних стандартів та регулятивних вимог, що спрощує процеси аудиту та контролю;

– об'єктивність та точність даних. На відміну від суб'єктивних оцінок, система надає об'єктивну інформацію, засновану на реальних даних, що підвищує точність діагностичних та прогностичних моделей.

Недоліки системи:

– високі витрати на впровадження та обслуговування. Первинна установка та подальше обслуговування FDM–систем може бути дороговартісним, особливо для малих авіакомпаній;

– проблеми з приватністю та конфіденційністю. Збір та аналіз даних може включати чутливу інформацію, яка, якщо не буде належним чином захищена, може бути використана недобросовісно;

– обмежена сумісність. Не сумісна з великим спектром авіаційного обладнання та програмного забезпечення, що може ускладнити її інтеграцію;

– потенційна складність в інтерпретації даних. Великий обсяг зібраних даних потребує висококваліфікованих аналітиків для їх правильної інтерпретації, що може бути викликом для авіакомпаній з обмеженими ресурсами.

Отже, система «Atennea Air» є незамінним інструментом в сучасній авіаційній промисловості для підвищення безпеки, оптимізації робочих процесів та забезпечення дотримання регулятивних вимог. Завдяки збору та аналізу деталізованих даних польотів, ці системи дозволяють авіакомпаніям оперативно реагувати на потенційні ризики та вносити корективи в експлуатацію авіаційного обладнання. Проте необхідно враховувати високі витрати на впровадження та обслуговування, можливі проблеми з сумісністю та конфіденційністю даних, а також потребу в кваліфікованих аналітиках для ефективного її використання.

SkyAnalyst FDM

«SkyAnalyst FDM» є продуктом, що розробляється та розповсюджується компанією Skytrac, яка є відомим постачальником рішень для моніторингу та комунікацій в авіаційній галузі.

«SkyAnalyst FDM» широко використовується в комерційній, корпоративній та навіть військовій авіації [17]. Система забезпечує аналіз даних польотів для підвищення рівня безпеки, для оцінки ефективності пілотажу, а також для дотримання стандартів та регулятивних вимог.

Програма може збирати, аналізувати та відображати різноманітні параметри, такі як швидкість, висота, положення руля, тиски в системах, стани двигунів і багато

інших. Зазвичай, дані можуть бути відображені в реальному часі та/або зберігатися для подальшого аналізу.

«SkyAnalyst FDM» має інтуїтивно зрозумілий користувацький інтерфейс (рис. 1.4), можливість кастомізації дашбордів та генерації звітів, а також ряд додаткових інструментів для глибокого аналізу даних.

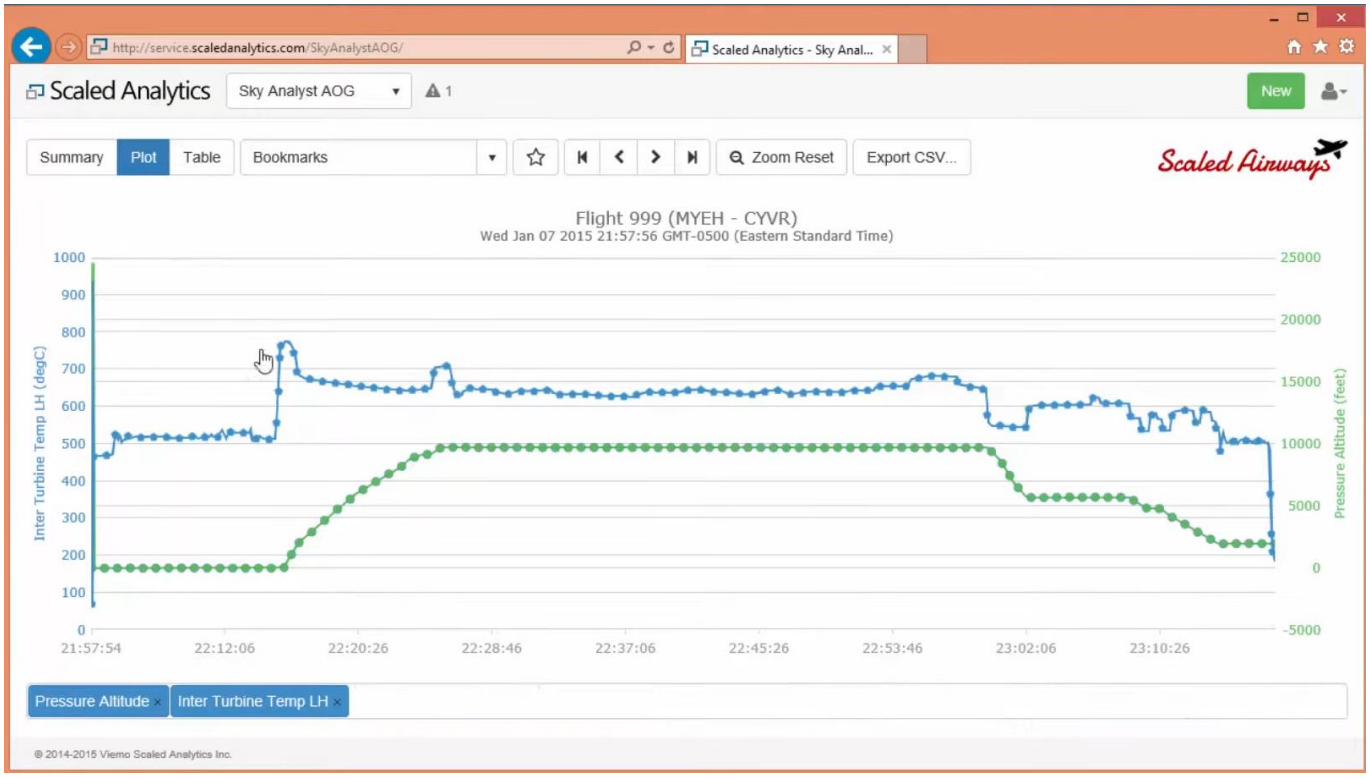

Рис 1.4. Інтерфейс користувача системи «SkyAnalyst FDM»

Переваги «SkyAnalyst FDM»:

– глибокий аналіз даних. Дозволяє проводити комплексний аналіз різних параметрів польоту, від швидкості та висоти до стану двигунів та систем;

– висока адаптивність. Система має інтуїтивний користувацький інтерфейс і можливість кастомізації, що дозволяє адаптувати її під конкретні потреби авіакомпанії;

– дотримання регулятивних вимог: Завдяки автоматизованому моніторингу, програма спрощує процес дотримання стандартів та регулятивних вимог в авіаційній сфері;

– мультиплатформеність. SkyAnalyst FDM часто сумісна з різними видами авіаційного обладнання, що робить її універсальним рішенням для різних типів літаків.

Недоліки системи:

– високі вартості ліцензування та підтримки. Ініціальні та повторні витрати на систему можуть бути високими, особливо для малих та середніх авіакомпаній;

– складність впровадження. Залежно від інфраструктури авіакомпанії, процес впровадження та налаштування може бути часомістким;

– потреба в кваліфікованому персоналі. Для ефективного використання системи потрібні спеціалісти, які розуміють як аналізувати авіаційні дані;

– можливі проблеми з безпекою даних. Як і з будь–якою системою, що працює з великими об'ємами даних, існує ризик несанкціонованого доступу або витоку інформації, якщо не забезпечена належна кібербезпека.

Отже, «SkyAnalyst FDM» є потужним інструментом для моніторингу та аналізу авіаційних даних, розробленим компанією Skytrac. Ця система відзначається високою адаптивністю, глибоким аналізом даних та можливістю дотримання регулятивних вимог. Проте вона має і свої слабкі сторони, такі як високі витрати на ліцензування та підтримку, складність впровадження та потреба в кваліфікованих аналітиках. Загалом, «SkyAnalyst FDM» може стати важливим елементом в системі управління безпекою для авіакомпаній різного рівня.

Aerobytes FDM

«Aerobytes FDM» є одним із відомих рішень для моніторингу даних польоту. Цей продукт розроблено компанією Aerobytes Ltd., яка спеціалізується на авіаційних безпекових рішеннях.

«Aerobytes FDM» використовується у різних сферах авіації, включаючи комерційні авіалінії, корпоративну авіацію, медичні служби авіації, а також у військовій сфері. Ця система спрямована на підвищення рівня безпеки польотів, ефективності пілотажу та дотримання авіаційних стандартів та регуляцій.

Програма забезпечує збір, аналіз та візуалізацію даних польоту. Вона може відслідковувати ряд ключових параметрів, таких як висота, швидкість, кут нахилу літака, тиск в різних системах, стан двигунів та інше.

Система «Aerobytes FDM» включає розширений набір інструментів для аналізу, таких як створення звітів, алерти в реальному часі, а також можливість інтеграції з

іншими авіаційними системами. Користувацький інтерфейс даного рішення представлено на рис. 1.5.

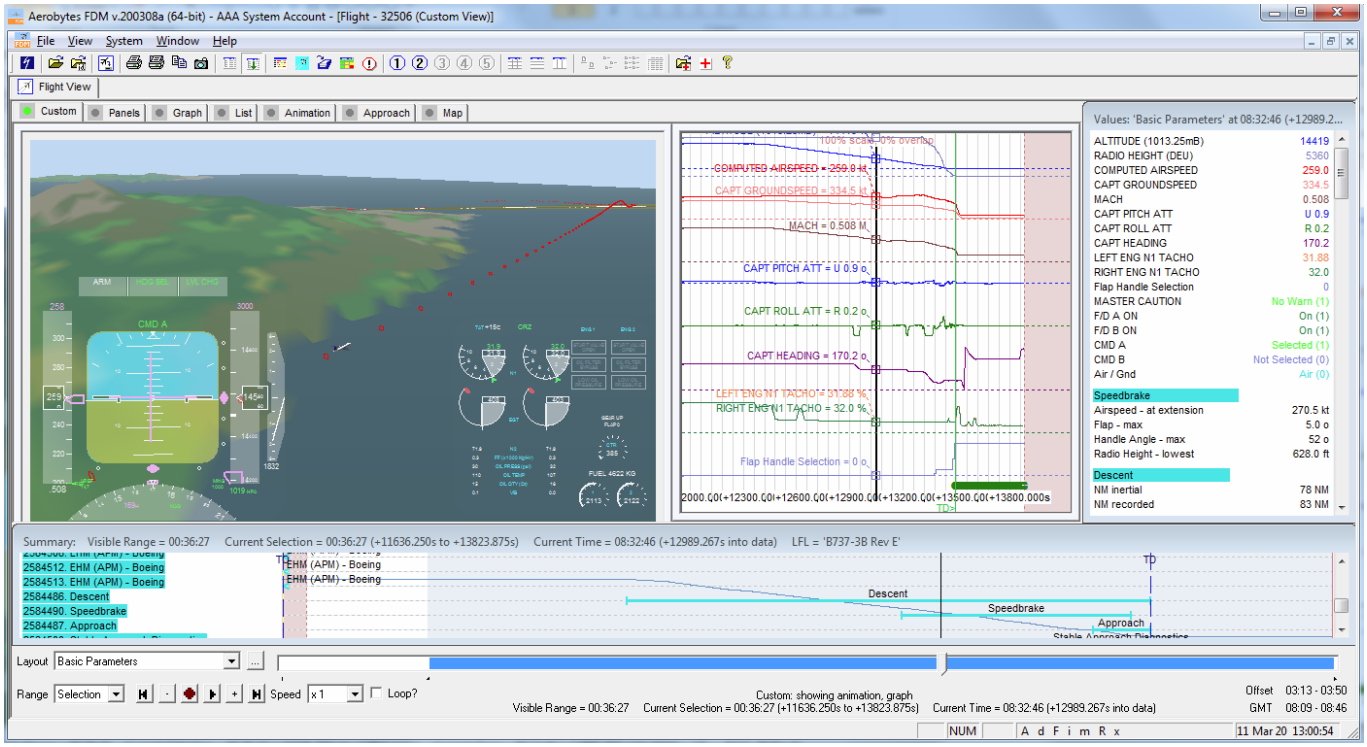

Рис 1.5. Інтерфейс користувача системи «Aerobytes FDM»

Переваги рішення «Aerobytes FDM» [18]:

– комплексний аналіз. Система дозволяє проводити глибокий аналіз різних параметрів польоту, що значно підвищує рівень безпеки авіаційних операцій;

– гнучкість та кастомізація. Має високу ступінь кастомізації, що дозволяє адаптувати систему до специфічних потреб авіакомпанії;

– багатоплатформеність. Система сумісна з різними видами авіаційного обладнання, що робить її універсальною;

– інтеграція з іншими системами. Здатність інтегруватися з іншими системами управління та моніторингу створює додаткову цінність, що полегшує управління ресурсами.

Недоліки системи:

– вартість. Витрати на систему можуть бути високими, що є критичним для малих авіакомпаній;

– складність впровадження. Залежно від інфраструктури та особливостей авіакомпанії, процес впровадження може бути тривалим та трудомістким;

– потреба в спеціалізованому персоналі. Ефективне використання системи вимагає наявності кваліфікованих спеціалістів, які здатні правильно інтерпретувати зібрані дані;

– обмеженість функціональності. Незважаючи на широкий спектр можливостей, система має обмеження в плані деталізації аналізу певних типів даних або в плані швидкості обробки інформації.

Отже, Aerobytes FDM представляє собою потужний інструмент для моніторингу та аналізу даних польоту, який може значно підвищити рівень безпеки та ефективності авіаційних операцій. Ця система відрізняється високою гнучкістю, можливістю кастомізації та інтеграції з іншими системами. Однак її впровадження може бути коштовним та трудомістким, особливо для малих та середніх авіакомпаній. Також система вимагає наявності кваліфікованого персоналу для ефективної експлуатації. Загалом, необхідно враховувати як переваги, так і недоліки при виборі даного рішення для конкретної авіакомпанії.

1.4.2 Принципи побудови систем моніторингу

В умовах сучасної авіації, де фактори безпеки, надійності та ефективності виступають як невід'ємні складові успішної експлуатації, роль систем моніторингу технічного стану авіаційного обладнання стає все більш значущою. Зростаюча складність авіаційної техніки, збільшення обсягів повітряних перевезень, а також суворі міжнародні норми і стандарти, посилюють необхідність вдосконалення методів та технік моніторингу. В цьому контексті, принципи побудови систем моніторингу виступають як фундаментальна база, на якій ґрунтується весь процес збору, обробки та інтерпретації даних. Ці принципи формуються на основі теоретичних досліджень, практичного досвіду, а також актуальних потреб авіаційної галузі. Вони враховують не тільки технічні аспекти роботи обладнання, але й організаційні, економічні та соціальні фактори. За допомогою дотримання цих

ключових принципів можливе створення ефективних, надійних та безпечних систем моніторингу, що спроможні задовольнити найвищі вимоги сучасної авіаційної промисловості.

На рис. 1.6 представлено базову схему сучасної системи моніторингу авіаційного обладнання.

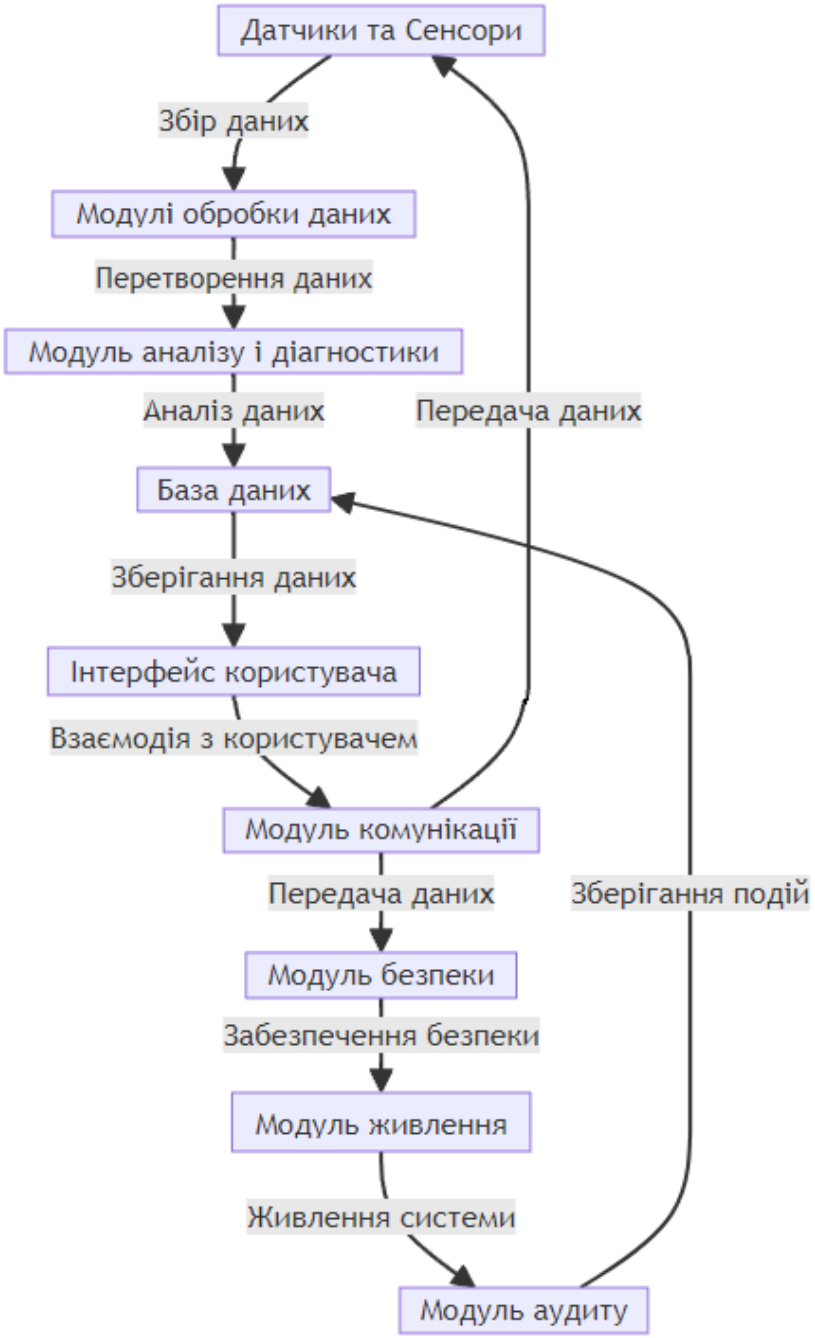

Рис 1.6. Схема системи моніторингу авіаційного обладнання

Діаграма ілюструє систему моніторингу технічного стану авіаційного обладнання, яка є високо інтегрованою та інтердисциплінарною. Система починає свою роботу з датчиків та сенсорів, які збирають первинні дані про технічний стан

обладнання, такі як температура, вібрація, швидкість обертання тощо. Ці дані потім передаються до модулів обробки даних, де вони перетворюються в корисну інформацію.

Після обробки, дані направляються до модуля аналізу і діагностики. Тут вони піддаються аналізу за допомогою спеціалізованих алгоритмів, які визначають стан обладнання. Результати аналізу зберігаються в базі даних, де вони можуть бути використані для подальших аналізів та алгоритмів машинного навчання.

Інтерфейс користувача відображає стан обладнання та надає можливість взаємодії з системою. Модуль комунікації забезпечує передачу даних між різними компонентами системи, включаючи повернення даних до датчиків та сенсорів для подальшого моніторингу.

Модуль безпеки забезпечує конфіденційність, цілісність та доступність інформації, тоді як модуль живлення гарантує стабільне живлення всіх компонентів системи. В кінці, модуль аудиту зберігає інформацію про всі події в системі для подальшого аналізу. Таким чином, кожен компонент системи взаємодіє в гармонійний спосіб, щоб забезпечити ефективний моніторинг та управління технічним станом авіаційного обладнання.

Отже, система моніторингу технічного стану авіаційного обладнання є комплексною інтегрованою структурою, яка об'єднує ряд підсистем та компонентів. Вона забезпечує збір, обробку та аналіз даних про стан обладнання, взаємодію з користувачем, а також гарантує безпеку та стабільність роботи всієї системи. Кожен компонент відіграє ключову роль в цілісному функціонуванні системи, що дозволяє ефективно моніторити та управляти технічним станом авіаційного обладнання.

1.5 Висновки до розділу

У даному розділі проведено аналіз існуючих систем моніторингу, таких як *Atennea Air, SkyAnalyst FDM ma Aerobytes FDM*, їх переваг та недоліків. Що дозволяє зробити обґрунтований вибір системи для конкретних задач та умов експлуатації.

Окрім цього, були вивчені та сформульовані принципи побудови систем моніторингу, які беруть до уваги всі раніше згадані аспекти – від класифікації до конкретних параметрів та характеристик. Ці принципи дозволяють не лише створити ефективну систему моніторингу, але й забезпечити її гнучкість та адаптивність до змінюваних умов експлуатації.

Таким чином, проведений аналіз показав що для розробки теми дипломного проекту необхідно наступне:

1) Визначити математичну модель програми;

2) Реалізувати основні алгоритми, такі як:

- Алгоритм навчання та зберігання даних моделі;
- Алгоритм моніторингу стану авіаційного обладнання;
- Алгоритм авторизації користувача;
- Алгоритм процесу додавання нової категорії прогназування та ін.;

3) Реалізувати архітектуру системи моніторингу, а саме:

- Визначити діаграму класів рівня даних;
- Визначити діаграму класів користувацького інтерфейсу;
- Визначити діаграму класів бізнес логіки;

4) Створити структуру системи моніторингу технічного стану авіаційного обладнання;

5) Провести тестування розробленої програми.

РОЗДІЛ 2
СИСТЕМА МОНІТОРИНГУ АВІАЦІЙНОГО ОБЛАДНАННЯ

2.1 Формалізація задачі моніторингу авіаційного обладнання

Формалізація задачі моніторингу технічного стану авіаційного обладнання з використанням технологій машинного навчання передбачає створення складної моделі, яка аналізує різноманітні аспекти технічних даних обладнання. Ця модель здатна виявляти потенційні несправності, враховуючи історичні дані про стан обладнання, а також різні показники його роботи. Процес формалізації включає наступні етапи:

Вхідні дані

Вхідні дані, які включають характеристики технічного стану обладнання, можна представити у вигляді вектору вхідних даних:

$$X = \{X_1, X_2 \dots X_n\} \qquad (2.1)$$

де X_i відображає кожну характеристику стану обладнання, зокрема:

X_1 – температура обладнання, вказує на тепловий стан та потенційні перегріви;

X_2 – тиск у системі, що може вказувати на проблеми з гідравлікою або ущільнювачами;

X_3– рівень вібрацій, що вказує на стан механічних компонентів;

X_4 – рівень масла, критичний для змащування рухомих частин;

X_5 – кількість годин роботи, важлива для планування технічного обслуговування;

X_6 – дата останнього технічного обслуговування, що вказує на частоту обслуговування;

X_7 – рівень зносу обладнання, важливий для оцінки його довговічності.

Кожна з цих характеристик несе важливу інформацію про технічний стан обладнання і використовується для виявлення можливих проблем та планування технічного обслуговування. Алгоритм аналізує ці вхідні дані для побудови моделі,

яка може ефективно виявляти потенційні несправності та потребу в технічному обслуговуванні на основі існуючих показників стану та історичних даних.

Конкатенація

Процес конкатенації полягає у злитті різних ознак (характеристик обладнання) у єдиний вектор ознак, який потім використовується у моделі для аналізу та прогнозування. Математично, конкатенація цих ознак може бути виражена як формування вектора ознак:

$$\text{Features} = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7\} \qquad (2.2)$$

де, $X_1, X_2, X_3, X_4, X_5, X_6, X_7$ відповідно є значеннями для температури обладнання, тиск у системі, рівню вібрації, рівню масла, кількості годин роботи, даті останнього технічного обслуговування та рівню зносу обладнання.

У системі моніторингу, мітка класу (Label) використовується для позначення стану обладнання – чи воно несправне чи ні. Цей процес можна виразити як просте присвоєння:

$$\text{Label} = IsFaulty \qquad (2.3)$$

де «IsFaulty» є булевим значенням, що вказує на наявність несправності в обладнанні.

Після виконання всіх трансформацій, загальний вектор ознак для моделі буде виглядати наступним чином:

$$Total\ Features = [Features] \qquad (2.4)$$

де «Total Features» включає всі числові ознаки, зібрані з даних про технічний стан обладнання. Цей вектор ознак є фундаментом для подальшого навчання моделі та її здатності прогнозувати можливі несправності в авіаційному обладнанні.

Оцінка моделі

Оцінка моделі моніторингу технічного стану авіаційного обладнання важлива для забезпечення її точності та надійності. Для цього використовуються статистичні метрики, що дозволяють оцінити, наскільки добре модель відповідає реальним даним. Нижче наведено декілька основних метрик для оцінки моделі:

1. Точність (Accuracy)

Точність вимірює відсоток випадків, коли модель правильно класифікувала дані:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (2.5)$$

де *TP* – істинно позитивні результати, *TN* – істинно негативні результати, *FP* – хибно позитивні результати, *FN* – хибно негативні результати.

2. Площа під кривою ROC (AUC – Area under the receiver operating characteristics curve):

AUC вимірює здатність моделі розрізняти між класами. Чим вища AUC, тим краще модель у розрізненні між позитивними та негативними класами.

$$AUC = \int_0^1 TPR(t)dFPR(t) \qquad (2.6)$$

де *TPR(t)* – істинно позитивний рівень при порозі *t*, *FPR(t)* – хибно позитивний рівень при порозі *t*.

3. F1 Score

F1 Score – це гармонійне середнє точності та повноти, що дозволяє збалансувати ці дві метрики (Precision, Recall), особливо в ситуаціях з нерівномірним розподілом класів. F1 Score розраховується за формулою:

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad (2.7)$$

де:

$$Precision = \frac{TP}{TP+FP} \qquad (2.8)$$

$$Recall = \frac{TP}{TP+FN} \qquad (2.9)$$

Ці метрики допомагають оцінити різні аспекти профілю точності класифікаційної моделі, включаючи її здатність правильно ідентифікувати класи, збалансувати помилки класифікації та розрізняти між класами.

Вихідні дані моделі

Вихідні дані моделі моніторингу технічного стану авіаційного обладнання можна виразити як функцію, яка відображає вхідні дані в передбачення стану обладнання. Цю функцію можна формулювати наступним чином:

$$\hat{y} = f(X) \qquad (2.10)$$

де:

$\hat{y}$ – представляє передбачений стан обладнання (наприклад, чи є обладнання несправним);

f – функція прогнозування, яка представляє модель;

X – вектор вхідних характеристик, який може включати такі параметри, як температура, тиск, рівень вібрацій, рівень масла, кількість годин роботи, дата останнього технічного обслуговування та рівень зносу обладнання.

Прогнозований стан обладнання $\hat{y}$ є основною вихідною характеристикою, яка визначається як:

$$IsFaulty = \hat{y} \qquad (2.11)$$

де, *IsFaulty* є передбаченням моделі щодо стану обладнання.

Оцінка якості цих передбачень здійснюється за допомогою метрик, які включають точність (*Accuracy*), площу під кривою *ROC* (*AUC*) та *F1 Score*. Ці метрики дозволяють визначити точність та надійність моделі в контексті прогнозування технічного стану авіаційного обладнання.

2.2 Математична модель

Алгоритм швидкого дерева є одним з варіантів алгоритмів дерев рішень у машинному навчанні. Основна ідея цього підходу полягає у створенні моделі, яка дозволяє класифікувати дані шляхом рішень, взятих на основі атрибутів цих даних.

У спрощеному описі алгоритму можна виходити з припущення, що всі атрибути є ненумеричними, і кожен атрибут зустрічається рівно один раз на кожному шляху від листа до кореня дерева. Однак, згодом можна вказати, як справлятися з нумеричними атрибутами. В аналізі алгоритму припускається, що кількість класів та кількість значень кожного атрибуту значно менші за m (розмір навчальних даних), і тому ці параметри можна ігнорувати. Також припускається, що всі навчальні дані завантажені у головну пам'ять.

При вирощуванні дерева евристичний принцип має вирішальне значення для визначення як ефективності класифікації, так і обчислювальних витрат. Більшість сучасних алгоритмів навчання дерев рішень використовують евристичний принцип, заснований на (не)чистоті, який суттєво вимірює чистоту результуючих підмножин після застосування атрибуту розбиття для розділення навчальних даних. Широко використовується інформаційний виграш (Information Gain, IG), визначений наступним чином:

$$IG(S, X) = Entropy(S) - \sum_x \frac{|S_x|}{|S|} Entropy(S_x) \qquad (2.12)$$

де S представляє набір даних, X – атрибут, за допомогою якого відбувається розбиття, S_x – підмножина даних, що відповідає певному значенню атрибута X, а *Entropy(S)* – міра ентропії набору даних S. Формула для розрахунку ентропії набору даних S виглядає наступним чином:

$$Entropy(S) = -\sum_{i=1}^{|C|} P_s(c_i) \log\left(P_s(c_i)\right) \qquad (2.13)$$

де $P_s(c_i)$ – це ймовірність класу c_i у наборі даних S, оцінена як відсоток випадків, що належать до c_i у S, і $|C|$ – це кількість класів. Ентропія для підмножини даних S_x розраховується аналогічно.

Важливо зазначити, що процес вирощування дерева є рекурсивним процесом розділення навчальних даних, при цьому S представляє навчальні дані, асоційовані з поточним вузлом. Тоді $P_s(c_i)$ насправді є $P_s(c_i|x_p)$ на всьому наборі навчальних даних, де X_p – це набір атрибутів на шляху від поточного вузла до кореня (атрибути шляху), а x_p –це набір значень для змінних у X_p . Аналогічно, $P_{S_x}(c_i) \epsilon P(c_i|x_p, x)$ на всьому наборі навчальних даних.

У процесі вирощування дерева кожен потенційний атрибут (атрибути, які не входять до X_p) аналізується за допомогою зазначеної формули, і той, що має найвищий інформаційний виграш, вибирається як атрибут розбиття. Найбільш обчислювально витратною частиною цього процесу є оцінка $P(c_i|x_p, x)$ для розрахунку ентропії S_x Це вимагає проходження через кожен екземпляр у S_x, для кожного з яких ітерується через кожен потенційний атрибут X. Це призводить до часової складності $O(|S| \cdot n)$. Об'єднання підмножин на кожному рівні дерева є весь

навчальний набір даних розміром m, і тому часова складність для кожного рівня становить $O(m \cdot n)$. Таким чином, стандартний алгоритм навчання дерев рішень має часову складність $O(m \cdot n^2)$.

У алгоритмі швидкого дерева ключовим спостереженням є те, що не потрібно проходити через весь набір даних S для кожного потенційного атрибуту, щоб оцінити $P(c_i|x_p, x)$. Згідно з теорією ймовірностей, отримується:

$$P(c_i|x_p, x) = \frac{P(c_i|x_p)P(x|x_p,c_i)}{P(x|x_p)} = \frac{P(c_i|x_p)P(x|x_p,c_i)}{\sum_{i=1}^{|C|} P(c_i|x_p)P(x|x_p,c_i)} \qquad (2.14)$$

Припускаючи, що кожен потенційний атрибут є незалежним від призначення атрибуту шляху x_p, враховуючи клас:

$$P(X|x_p, C) = P(X|C), \qquad (2.15)$$

отримується:

$$P(c_i|x_p, x) \approx \frac{P(c_i|x_p)P(x|c_i)}{\sum_{i=1}^{|C|} P(c_i|x_p)P(x|c_i)} \qquad (2.16)$$

Інформаційний виграш, отриманий за допомогою цієї формули і формули 1, називається в цій роботі незалежним інформаційним виграшем (IIG). Зауважте, що в формулі 2.16 $P(x|c_i)$ є відсотком випадків з X=x і C= c_i на всьому навчальному наборі даних, який може бути попередньо розрахований і збережений з часовою складністю $O(m \cdot n)$ перед процесом вирощування дерева з додатковим збільшенням простору на $O(n)$, а $P(c_i|x_p)$ – це відсоток випадків, що належать до класу c_i у S, який можна розрахувати, пройшовши через S один раз, займаючи $O(|S|)$. Таким чином, на кожному рівні часова складність для розрахунку $P(c_i|x_p, x)$ за допомогою формули 3.16 становить $O(m)$.

Для поточного вузла $IIG(S, X)$ повинен бути розрахований для кожного потенційного атрибуту, що займає $O(n)$ для кожного вузла. Загальна часова складність для вибору атрибуту розбиття на всьому дереві тоді становить $O(k \cdot n)$, де k – кількість внутрішніх вузлів на дереві. У найгіршому випадку кожен лист дерева містить один випадок, і кількість внутрішніх вузлів є $O(m)$. Таким чином, загальна часова складність для вибору атрибуту розбиття становить $O(m \cdot n)$.

$$P(X_1, X_2, \ldots X_n|C) = \prod_{i=1}^{n} P(X_i|C) \qquad (2.17)$$

У даному алгоритмі робиться припущення про умовну незалежність, яке схоже на відоме припущення умовної незалежності в наївному баєсовому класифікаторі. У наївному баєсовому класифікаторі припускається, що всі атрибути незалежні за умови наявності певного класу, і це визначається таким чином:

Однак, припущення в рівнянні 2.15 слабше, ніж припущення в рівнянні 2.17. По–перше, рівняння 2.17 визначає припущення на всьому просторі екземплярів (глобальне припущення про незалежність). Але рівняння 2.15 визначає припущення на підпросторі, визначеному конкретним призначенням x_p. Це, по суті, контекстно–специфічна незалежність, яку описали Фрідман і Гольдшмідт у 1996 році [19]. Ясно, що контекстно–специфічна незалежність має більш тонке розмежування, ніж глобальна незалежність, у тому сенсі, що деякі атрибути не є глобально незалежними, але вони є незалежними у певному контексті. По–друге, в рівнянні 2.15 не робиться припущення, що кандидатські атрибути, які не входять до X_p, є незалежними. На відміну від цього, наївний баєсівський класифікатор припускає, що всі атрибути є незалежними, включаючи кандидатські атрибути.

Це розрізнення між умовною незалежністю в алгоритмі швидкого дерева та наївному баєсівському класифікаторі має важливі наслідки для практичного застосування. Контекстно–специфічна незалежність дозволяє алгоритму бути більш гнучким та точним при аналізі даних, що містять складні залежності між атрибутами, які не можна виявити за допомогою глобальних припущень незалежності. Така гнучкість та точність є особливо важливими при розробці систем моніторингу технічного стану авіаційного обладнання. Авіаційне обладнання характеризується високою складністю та великою кількістю параметрів, які потрібно відстежувати, включаючи температуру, тиск, швидкість обертання двигунів, стан гідравлічних систем та багато іншого. Кожен з цих параметрів може впливати на інші в різних умовах, тому алгоритм, здатний розуміти та адаптуватися до цих залежних відношень, є надзвичайно корисним.

Система моніторингу технічного стану, розроблена на основі алгоритму швидкого дерева, може бути ефективною у передбаченні потенційних несправностей та необхідності технічного обслуговування. Така система може аналізувати великі

об'єми даних у реальному часі, виділяючи важливі закономірності та аномалії, які можуть свідчити про зниження ефективності або пошкодження обладнання. Це, в свою чергу, дозволяє авіаційній компанії своєчасно реагувати на потенційні проблеми, мінімізуючи ризики для безпеки польотів та оптимізуючи графік технічного обслуговування.

Використання алгоритму швидкого дерева в таких системах може не тільки підвищити безпеку і надійність авіаційного обладнання, але й знизити витрати на його обслуговування, передбачаючи проблеми до того, як вони призведуть до серйозних збоїв або дорогих ремонтних робіт.

2.3 Реалізація основних алгоритмів

Розробка алгоритмів для системи моніторингу технічного стану авіаційного обладнання є критично важливим етапом, який залежить від глибокого розуміння основних алгоритмічних принципів та вимог до системи. Ці алгоритми становлять основу для виконання широкого спектру завдань, від збору та обробки даних до їх аналізу та виведення прогнозів. Основною метою алгоритмів у такій системі є виявлення потенційних проблем або несправностей обладнання на ранніх етапах, що дозволяє запобігти серйозним поломкам та оптимізувати процеси технічного обслуговування.

Алгоритми машинного навчання у цьому контексті відіграють ключову роль, дозволяючи системі не тільки класифікувати поточний стан обладнання, але й прогнозувати його майбутній стан на основі аналізу великих обсягів даних. Це включає розробку моделей, які можуть вивчати складні закономірності в роботі обладнання, виходячи з історичних даних. Алгоритми використання навчених моделей включають застосування цих моделей до реальних даних для оцінки поточного стану обладнання та надання рекомендацій щодо можливих дій.

У сукупності, ці алгоритми формують систему, яка може динамічно адаптуватися до різних умов роботи обладнання, надаючи точні та своєчасні дані для підтримки процесів технічного обслуговування та управління авіаційним обладнанням. Вони також сприяють підвищенню безпеки та ефективності використання авіаційної техніки, надаючи важливі дані для прийняття рішень у сфері авіаційної промисловості.

На рис. 2.1 представлено схему алгоритму для навчання моделі.

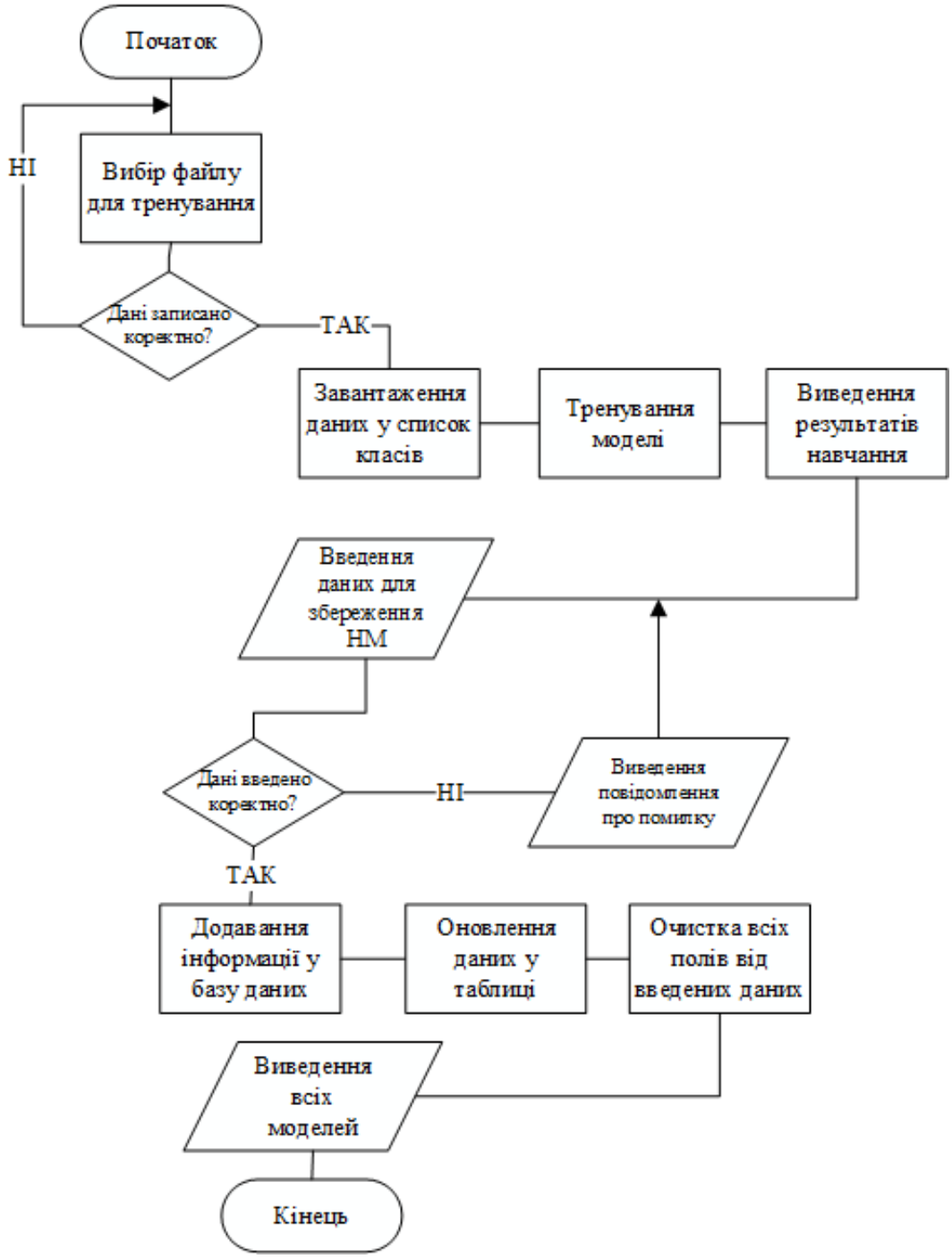

Рис 2.1. Алгоритм навчання та зберігання даних моделі

Процес тренування моделі у системі моніторингу технічного стану авіаційного обладнання починається з вибору файлу для тренування, де користувач визначає вхідні дані, такі як різноманітні показники стану обладнання. Алгоритм перевіряє ці дані на коректність, і у разі помилок пропонує повернутися до попереднього кроку. Після підтвердження коректності, дані завантажуються у систему для подальшого аналізу.

Наступним кроком є сам процес тренування, де алгоритм використовує вхідні дані для «навчання» моделі розпізнавати шаблони та закономірності в даних. Цей процес включає в себе багаторазові ітерації, завдяки яким модель адаптується та оптимізує свою здатність до прогнозування.

Після завершення тренування алгоритм виводить результати, включаючи метрики якості моделі, такі як точність та F–міра, що дозволяє оцінити ефективність навчання. Далі користувач має можливість ввести всю необхідну інформацію для збереження моделі, включаючи її назву та місце зберігання.

У разі помилок при введенні даних, алгоритм надає повідомлення про помилку, вимагаючи корекції. Після успішного введення та перевірки даних, інформація про модель додається до бази даних, а таблиця з даними про навчені моделі оновлюється. На завершальному етапі всі поля для введення даних очищаються, готуючи систему до наступного використання, і алгоритм виводить загальну інформацію про всі навчені моделі, демонструючи загальні результати та статистику тренувань.

Рис. 2.2 відображає схему алгоритму роботи навченої моделі у реальному часі.

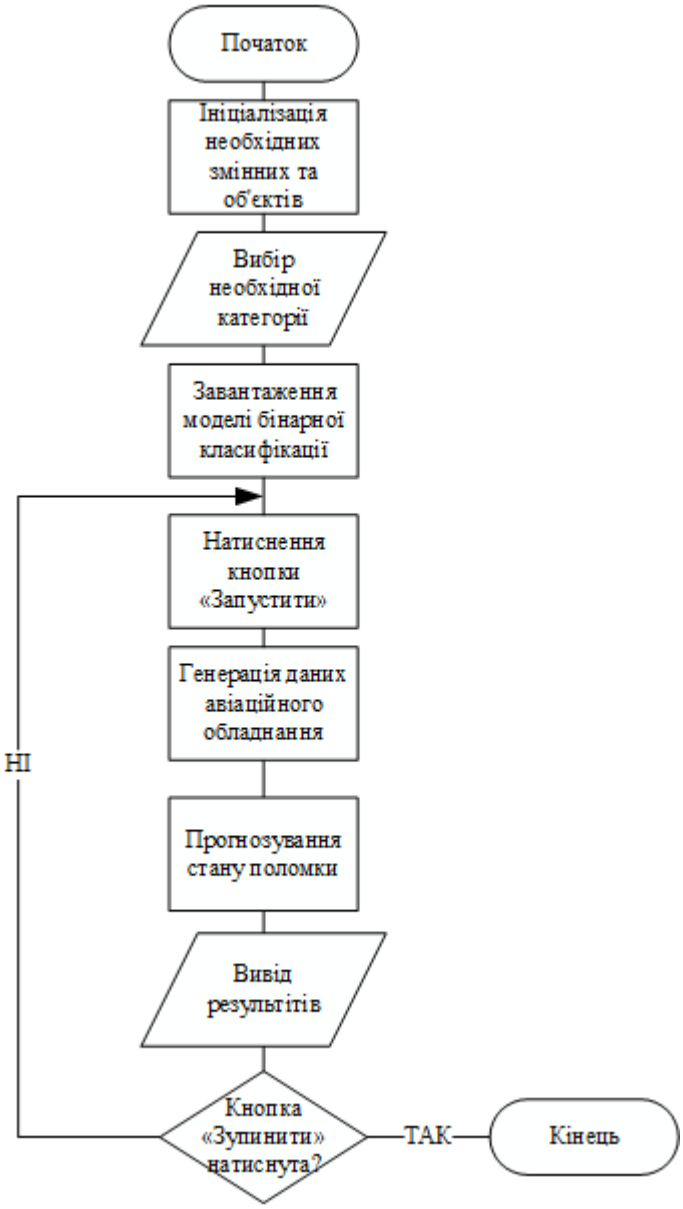

Рис 2.2. Алгоритм моніторингу стану авіаційного обладнання

У програмі для моніторингу технічного стану авіаційного обладнання використовується взаємодія з користувачем та машинне навчання для аналізу й прогнозування стану обладнання. Програма ініціюється завантаженням необхідних категорій, що дозволяє користувачу вибрати конкретну категорію для аналізу. Після вибору категорії програма завантажує відповідну модель, готову до аналізу даних.

Коли користувач натискає кнопку «Запустити», активується таймер, який генерує випадкові дані, імітуючи реальні вимірювання стану обладнання. Ці дані включають температуру, тиск, вібрації, рівень олії, кількість годин роботи, час від останнього технічного обслуговування та рівень зносу. Згенеровані дані передаються в нейронну мережу, яка виконує прогнозування стану обладнання.

Прогноз містить інформацію про ймовірність несправності, яку програма інтерпретує та відображає у текстовому полі. Якщо модель прогнозує потенційну несправність, користувачу відображається повідомлення про можливу поломку, а також вказується ймовірність такої поломки. При цьому програма забезпечує можливість користувача зупинити процес аналізу в будь–який момент, змінюючи текст кнопки управління та стан таймера.

Весь процес роботи програми спрямований на забезпечення оперативного та точного моніторингу технічного стану обладнання, що дозволяє запобігти можливим поломкам або неполадкам і вжити відповідних заходів з технічного обслуговування. Завдяки цьому програма є важливим інструментом у підтримці надійної та безпечної експлуатації авіаційного обладнання.

2.4 Архітектурні особливості системи моніторингу АО

Для розробки системи моніторингу технічного стану АО обрано трирівневий архітектурний різновид, враховуючи її здатність ефективно розділяти різні функціональні аспекти системи. Такий підхід до архітектурної побудови забезпечує ясне відокремлення логіки користувацького інтерфейсу від бізнес–логіки та шару доступу до даних, що є критично важливим для забезпечення масштабованості, гнучкості та легкості обслуговування високоспеціалізованої системи.

На презентаційному рівні цей архітектурний різновид дозволяє створити інтуїтивно зрозумілий та зручний інтерфейс, через який користувачі можуть спостерігати за станом обладнання, отримувати повідомлення про виявлені несправності та управляти налаштуваннями системи моніторингу. Цей рівень ізольований від безпосереднього доступу до даних, що збільшує безпеку та стабільність роботи.

Бізнес–логічний рівень виконує обробку даних, використовуючи алгоритми машинного навчання та аналітичні інструменти для аналізу зібраної інформації та

виведення прогнозів. Відділення цього шару дозволяє легко адаптувати або замінювати алгоритми обробки без зміни інших частин системи.

На рівні доступу до даних забезпечується взаємодія з базами даних або іншими зовнішніми системами для зберігання, оновлення та відновлення інформації, необхідної для роботи моніторингу. Ця модульність управління даними дозволяє системі гнучко реагувати на зміни в структурі або обсязі даних.

Таким чином, трирівнева архітектура є оптимальним рішенням, яке дозволяє досягти високого рівня спеціалізації кожного компоненту системи, гарантуючи при цьому її надійність, масштабованість та легкість у супроводі.

Спочатку було створено шар доступу до даних (рис. 2.3), що представляє собою абстракцію, яка ізолює бізнес–логіку застосунку від безпосереднього доступу до бази даних. Це дозволяє змінювати структуру бази даних без впливу на інші частини системи та надає гнучкість для масштабування та підтримки.

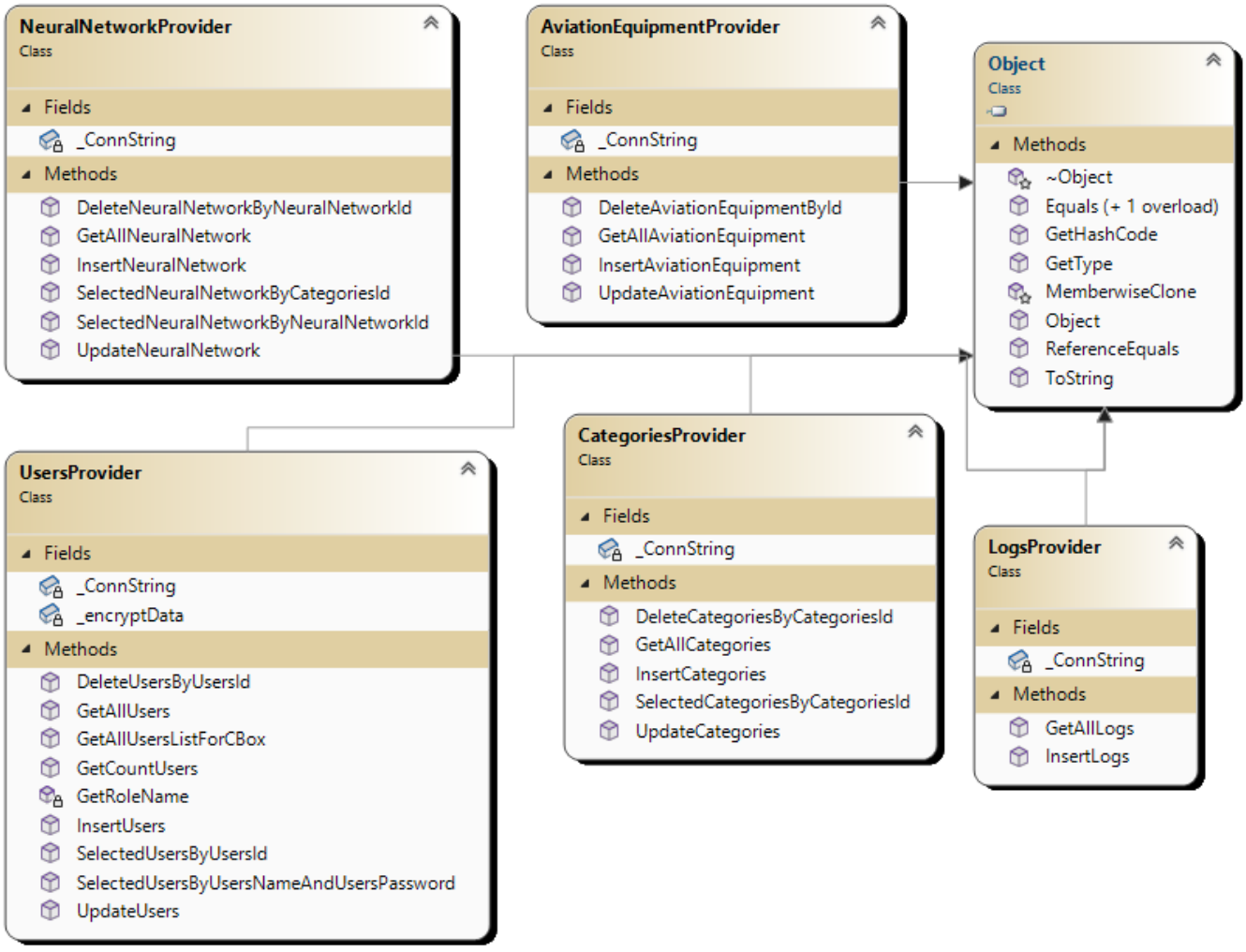

Рис. 2.3. Діаграма класів рівня даних

Діаграма даного рівня складається із таких класів:

– клас AviationEquipmentProvider відповідає за отримання та управління даними про авіаційне обладнання. Функції класу включають завантаження детальної інформації про обладнання, його технічні характеристики, історію роботи та дані про його стан. Він також включає методи для збереження нових даних, оновлення існуючих та видалення застарілих записів;

– клас CategoriesProvider відповідає за управління категоріями моделей, які використовуються для класифікації та аналізу різних типів авіаційного обладнання. Він містить методи для створення нових категорій, отримання списку всіх доступних категорій та їх описів, а також оновлення та видалення існуючих категорій;

– клас LogsProvider займається збором та управлінням логами системи, які можуть містити інформацію про виконання різних процесів та дії користувачі. Він включає функціонал для реєстрації нових записів логів, пошуку та фільтрації за певними критеріями, а також архівування та видалення старих логів;

– клас NeuralNetworkProvider призначений для управління нейронними мережами, які використовуються в системі для аналізу даних.

– Функції включають завантаження навчених моделей, оновлення їх параметрів, відстеження їх ефективності та вибір найбільш підходящих моделей для конкретних завдань аналізу;

– клас UsersProvider відповідає за управління даними користувачів системи. Він включає методи для реєстрації нових користувачів, автентифікації, управління ролями та дозволами, а також відстеження активності користувачів в системі.

В архітектурі системи було вирішено виділити інтерактивний рівень користувацького інтерфейсу. Розробка цього сегмента зосереджена на створенні інтуїтивно зрозумілих форм, які інтегрують різноманітні графічні елементи управління, такі як кнопки, текстові поля та таблиці для відображення даних. Ці компоненти забезпечують основу для двосторонньої взаємодії: вони дозволяють користувачам не тільки подавати інформацію, але й отримувати зворотній зв'язок та результати операцій.

Інтерфейс спроектований таким чином, щоб максимально спростити введення даних та навігацію по системі. Кожна взаємодія користувача з графічними елементами передає управління обробникам подій, які в свою чергу комунікують з бізнес–логікою та рівнем даних, щоб забезпечити виконання запитів і повернення результатів (рис. 2.4).

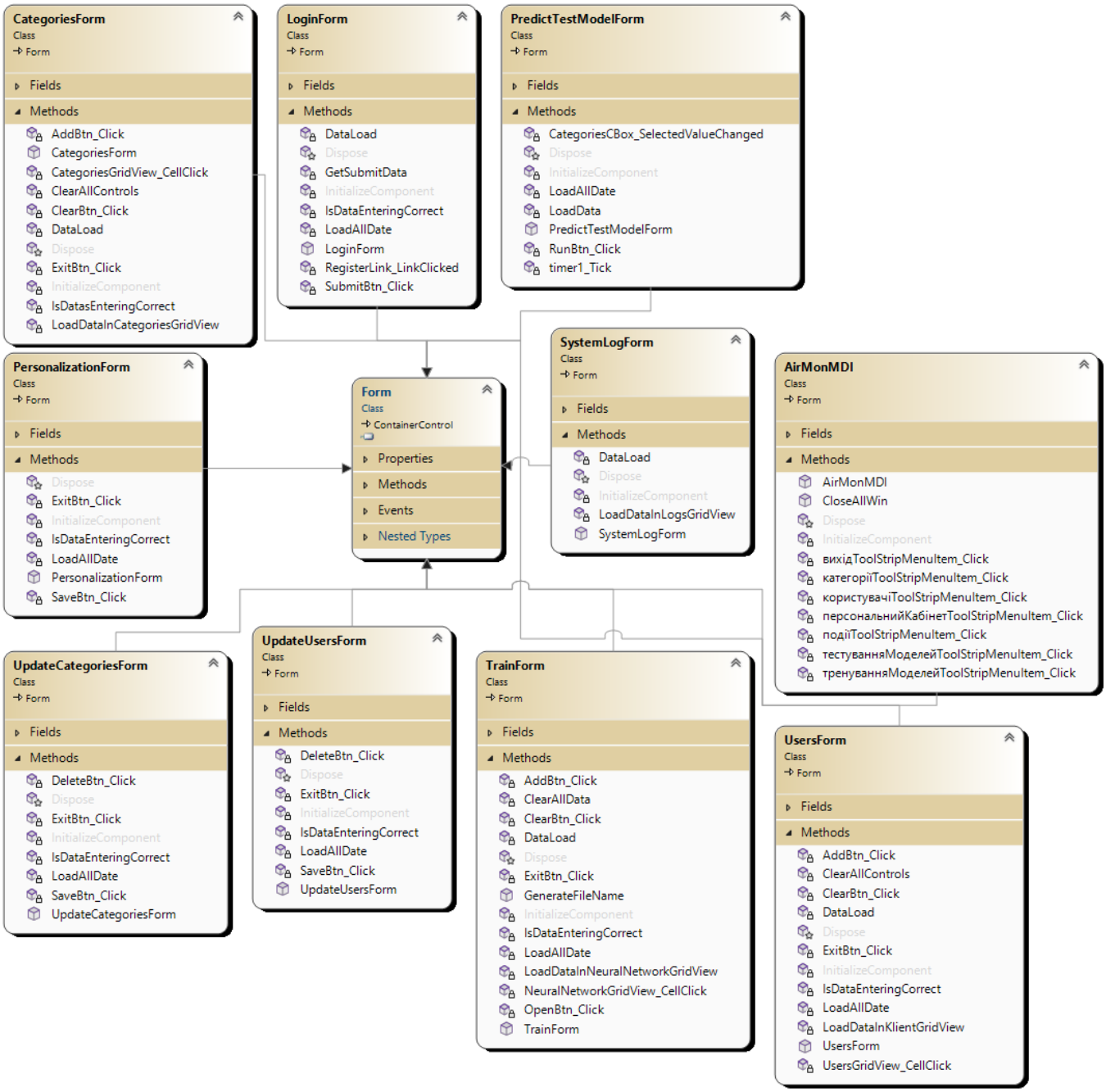

Рис. 2.4. Діаграма класів користувацького інтерфейсу

Діаграма даного рівня складається з десяти класів рівня UI та є похідними від класу Form, тобто мають графічний інтерфейс:

– клас AirMonMDI – це основне вікно програми, яке виступає як багатодокументний інтерфейс (MDI). Він також включає функціональність для навігації між різними формами системи.

– клас PredictTestModelForm використовується для виконання прогнозування стану обладнання за допомогою навчених моделей. Користувач може ініціювати процес тестування моделі, вибираючи певні параметри та категорії обладнання, для якого потрібно виконати прогноз. Форма також включає інтерактивний елемент для відображення вхідних даних, кнопку для запуску процесу прогнозування та область для виведення результатів;

– клас CategoriesForm використовується для управління категоріями обладнання у системі. Цей інтерфейс дозволяє користувачам створювати, редагувати та видаляти категорії, до яких належить обладнання, забезпечуючи організацію даних за категоріями для легшого доступу та аналізу. Він також надає можливість перегляду та управління існуючими категоріями, відображаючи деталі та характеристики кожної категорії;

– клас TrainForm призначений для навчання або тренування моделей на основі тренувальних наборів даних. Після завершення навчання форма може відображати статистику та метрики ефективності моделі, такі як точність, втрати та інші релевантні показники;

– клас UpdateCategoriesForm використовується для оновлення інформації про категорії обладнання. Він дозволяє користувачам здійснювати зміни до властивостей та атрибутів існуючих категорій, включаючи назви категорій, описи та пов'язані з ними параметри;

– клас LoginForm – це форма авторизації, яка використовується для входу користувачів в систему. Цей клас управляє процесом перевірки облікових даних користувача та надає доступ до системи після успішної автентифікації. Він також містить заходи безпеки, такі як шифрування паролів та захист від несанкціонованого доступу;

– клас PersonalizationForm дозволяє користувачам налаштувати індивідуальні параметри облікових даних;

– клас SystemLogForm призначений відображення системних логів, що надає інформацію про події в системі, такі як вхід користувачів, помилки, попередження та інші системні повідомлення;

– клас UpdateUsersForm надає можливість оновлення інформації про користувачів, таку як їхні ролі, статуси та доступи. Використовується адміністраторами для управління обліковими записами та зміни правил доступу до системи.

– клас UsersForm відповідає за управління обліковими записами користувачів, надаючи інтерфейс для їх створення, редагування, перегляду та видалення.

Розробка бізнес–логіки стала ключовим моментом у створенні архітектури застосунку, оскільки вона забезпечує основу для усіх бізнес–процесів системи. Цей елемент архітектури, що інкапсулює всі бізнес–правила та алгоритми, є вирішальним для забезпечення адекватної бізнес–функціональності та становить осердя внутрішніх операцій додатку. Організація бізнес–логіки в окремі модулі сприяє гнучкості системи, дозволяючи їй легко масштабуватися та адаптуватися до змінних вимог з мінімальними змінами в коді.

Ізоляція бізнес–правил від інших рівнів застосунку, таких як взаємодія з базою даних чи користувацький інтерфейс, дозволяє розробникам фокусуватися на розробці логіки без необхідності вникати в деталі реалізації бази даних або презентації даних користувачеві. Це спрощує розробку, підтримку та розширення коду, робить систему більш гнучкою та відкритою для різноманітних технологічних інтеграцій.

Візуальне представлення бізнес–логіки на діаграмі чітко демонструє її взаємозв'язок з іншими компонентами системи, підкреслюючи межі відповідальності та забезпечуючи зрозуміле уявлення про структуру та дизайн застосунку. (рис. 2.5).

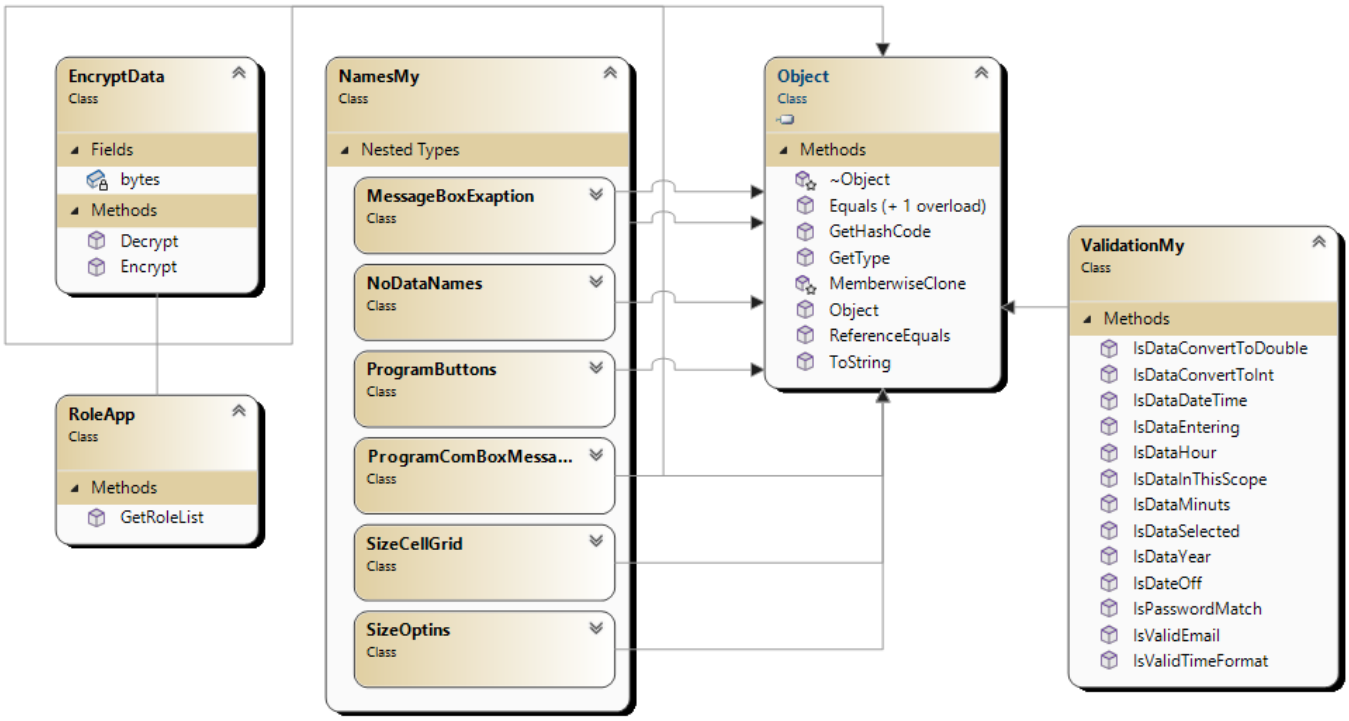

Рис. 2.5. Діаграма класів бізнес логіки

У системі моніторингу авіаційного обладнання, класи бізнес–логіки відіграють важливу роль у застосуванні бізнес–правил і алгоритмів, забезпечуючи безпеку, управління даними та валідацію. Основні класи бізнес–логіки складаються із:

– клас EncryptData використовується для шифрування чутливої інформації, що обробляється системою моніторингу. Він містить методи для шифрування та розшифрування даних, використовуючи сучасні алгоритми криптографії. Шифрування даних забезпечує безпеку інформації, що передається між різними компонентами системи та зберігається в базах даних;

– клас NamesMy призначений для управління номенклатурою та налаштуванням іменувань об'єктів у системі. Він включає функціональність для створення, зберігання та виклику змінних імені, що дозволяє уніфікувати та стандартизувати назви компонентів системи для полегшення їх ідентифікації та використання;

– клас RoleApp відповідає за управління ролями користувачів у програмі. Він містить методи для призначення ролей користувачам, визначення рівнів доступу та дозволів, а також відстеження та забезпечення дотримання політики безпеки. Це

забезпечує контрольований доступ до функцій системи залежно від визначених ролей та обов'язків користувачів;

– клас ValidationMy призначений для валідації даних, що вводяться користувачами або отримуються з зовнішніх джерел. Він включає набір правил та перевірок, які забезпечують точність та відповідність даних до вимог системи перед їх обробкою або зберіганням. Функції валідації важливі для попередження помилок та неполадок у системі, які можуть виникнути через некоректні дані.

Кожен з цих класів відіграє визначену роль у загальному процесі обробки даних, забезпечуючи надійність та ефективність системи моніторингу. Вони допомагають створити структуровану та безпечну обробку інформації, що є необхідною для забезпечення безпеки та ефективності авіаційного обладнання.

2.5 Висновки до розділу

У рамках даного розділу було розроблено та детально описано систему моніторингу технічного стану авіаційного обладнання, яка використовує технології машинного навчання. Спочатку була виконана формалізація задачі моніторингу, що передбачала створення складної моделі для аналізу різноманітних аспектів технічних даних обладнання. Модель здатна виявляти потенційні несправності, враховуючи історичні дані про стан обладнання та різні показники його роботи.

Далі було розглянуто математичну модель, зокрема, алгоритм швидкого дерева, що дозволяє ефективно класифікувати дані, виявляти важливі закономірності та аномалії в даних, що можуть свідчити про зниження ефективності або пошкодження обладнання. Це забезпечує авіаційній компанії можливість своєчасно реагувати на потенційні проблеми, мінімізуючи ризики для безпеки польотів та оптимізуючи графік технічного обслуговування.

Реалізація основних алгоритмів у системі моніторингу була критично важливою. Алгоритми машинного навчання відіграли ключову роль, дозволяючи не

тільки класифікувати поточний стан обладнання, але й прогнозувати його майбутній стан. Це забезпечує точні та своєчасні дані для підтримки процесів технічного обслуговування та управління авіаційним обладнанням.

Важливим аспектом роботи було створення трьохрівневої архітектури системи моніторингу, що забезпечує ефективне розділення різних функціональних аспектів системи. Цей підхід до архітектури забезпечує ясне відокремлення логіки користувацького інтерфейсу від бізнес–логіки та шару доступу до даних, що є критично важливим для забезпечення масштабованості, гнучкості та легкості обслуговування високоспеціалізованої системи.

РОЗДІЛ 3
ПРОГРАМНО–АПАРАТНИЙ КОМПЛЕКС СИСТЕМИ МОНІТОРИНГУ

3.1 Структура системи моніторингу авіаційного обладнання

Система моніторингу технічного стану авіаційного обладнання використовує комплексний підхід, що охоплює важливі етапи збору, обробки та аналізу даних. Ця система спроектована для забезпечення безперебійної та надійної роботи авіаційного обладнання, зниження ризику виникнення аварійних ситуацій та підвищення загальної безпеки авіаційних операцій.

Центральним елементом цієї системи є датчики, які розташовані на різних частинах авіаційного обладнання. Ці датчики здійснюють постійний моніторинг параметрів, таких як температура, тиск, вібрація та інші фізичні величини, які можуть вказувати на потенційні проблеми або знос обладнання.

Зібрані дані передаються контролеру системи. Контролер виконує функції обробки та фільтрації даних, щоб визначити, чи відповідає стан обладнання нормальним параметрам. Якщо виявляються відхилення або аномалії, контролер автоматично генерує сповіщення та інформацію для подальшого аналізу.

Отримані дані також передаються на сервер системи, де проводиться подальший аналіз та зберігання інформації. Аналіз може включати в себе порівняння поточних даних з історичними даними, розрахунок прогнозних показників технічного стану обладнання та визначення потреби в плановому технічному обслуговуванні.

На рис. 3.1 представленої схеми можна побачити, як взаємодіють основні компоненти системи. Від датчиків, які збирають первинні дані, до сервера, що координує процеси обробки та аналізу даних. Аналіз даних здійснюється за допомогою методу швидкого дерева, що забезпечує ефективне виявлення та аналіз потенційних проблем. Результати аналізу відображаються в застосунку, який надає

користувачу зручний інтерфейс для моніторингу стану обладнання. Таким чином, схема демонструє замкнений цикл взаємодії між компонентами системи, що забезпечує надійний моніторинг та управління технічним станом авіаційного обладнання.

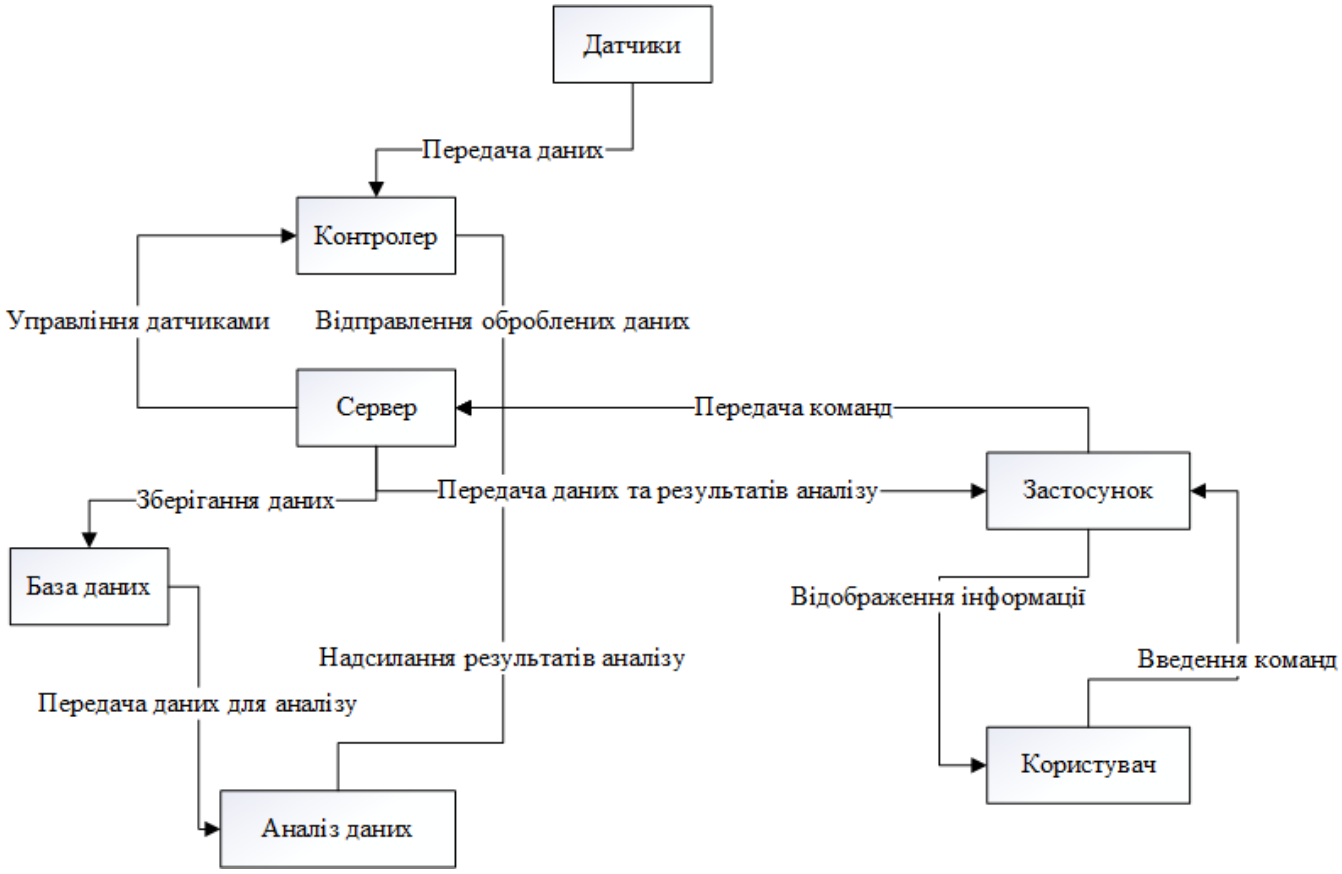

Рис. 3.1. Система моніторингу технічного стану авіаційного обладнання

Представлена графічна схема на рис. 3.1 ілюструє взаємодію компонентів у системі моніторингу технічного стану авіаційного обладнання:

– Датчики → Контролер. Датчики збирають необхідні дані з авіаційного обладнання. Ці дані включають параметри, такі як температура, тиск, вібрації тощо. Після збору даних, вони передаються до контролера;

– Контролер → Сервер. Контролер обробляє отримані дані. Це включає фільтрацію, нормалізацію та інші види обробки для підготовки даних до подальшого аналізу. Після обробки, дані відправляються на сервер;

– Сервер → База даних. Сервер отримує оброблені дані від контролера і зберігає їх у базі даних. База даних служить для накопичення, систематизації та зберігання даних для подальшого використання;

– База даних → Аналіз даних. Дані з бази даних передаються для аналізу. У цьому контексті використовується метод швидкого дерева, який дозволяє ефективно аналізувати та інтерпретувати зібрані дані;

– Аналіз даних → Сервер. Після завершення аналізу, результати надсилаються назад на сервер;

– Сервер → Застосунок. Сервер передає дані та результати аналізу до застосунку. Спеціалізований програмний інтерфейс дозволяє користувачам переглядати та інтерпретувати результати моніторингу;

– Застосунок → Користувач. Застосунок відображає інформацію для користувача. Користувач може переглядати зібрані дані та аналітичні звіти, це все допомагає зрозуміти стан авіаційного обладнання;

– Користувач → Застосунок. Користувач вводить команди або запити в застосунок. Це може включати запити на отримання додаткової інформації, навчання нових моделей для прогнозування технічного стану авіаційного обладнання, налаштування параметрів моніторингу тощо;

– Застосунок → Сервер. Застосунок передає команди або запити користувача на сервер для обробки;

– Сервер → Контролер. Сервер передає команди або інструкції від користувача до контролера. Це включає команди на зміну параметрів моніторингу, запити на збір додаткових даних тощо;

– Контролер → Датчики. Контролер управляє датчиками на основі команд, отриманих від сервера. Це включає зміну режимів роботи датчиків, частоти збору даних тощо.

Схема відображає замкнений цикл взаємодії між компонентами системи моніторингу, що забезпечує збір, обробку, аналіз та візуалізацію даних, а також взаємодію з користувачем для управління процесом моніторингу.

3.2 Алгоритми роботи

Задача розробки застосунку для моніторингу технічного стану авіаційного обладнання полягає в створенні зручного та простого у використанні застосунку, який допоможе вести моніторинг технічного стану авіаційного обладнання, створювати категорії прогнозування, тренувати моделі для прогнозування поломок, тощо. Для цього необхідно розробити основні алгоритми, що забезпечать роботу додатку та забезпечать користувачів необхідною інформацією та можливістю взаємодії із авіаційним обладнанням. У даному підрозділі буде розглянуто розробку ключових алгоритмів, необхідних для роботи застосунку « Моніторинг технічного стану авіаційного обладнання».

Перше, із чим користувач буде взаємодіяти – форма авторизації. На рис. 3.2 показано блок–схему алгоритму, яка демонструє процес авторизації користувача.

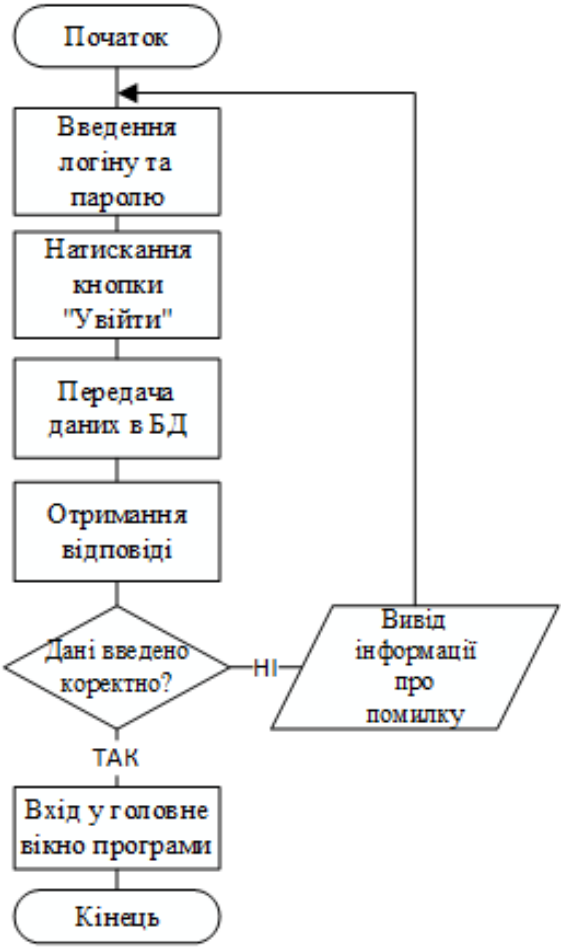

Рис. 3.2. Алгоритм авторизації користувача

Авторизація користувача в інформаційних системах відіграє ключову роль у забезпеченні безпеки та контролі доступу. Процес авторизації розпочинається з ініціації користувачем входу в систему, де він зобов'язаний ввести свої аутентифікаційні дані: логін та пароль. Ці дані є основними кредeнціалами, які використовуються для ідентифікації користувача та забезпечення його унікальності в системі.

Після введення логіна та пароля користувач натискає кнопку «Увійти», що ініціює процес передачі введених даних до сервера. На сервері відбувається перевірка цих даних шляхом їх порівняння з відповідними записами в базі даних. Важливо відзначити, що цей етап є критичним з точки зору безпеки, оскільки від правильності перевірки залежить легітимність доступу до системи.

Якщо введені дані не відповідають жодному із записів у базі даних, система генерує повідомлення про помилку, повідомляючи користувача про некоректний ввід даних. У цьому випадку процес авторизації переривається, і користувачу пропонується повторно ввести свої дані, починаючи з введення пароля. У випадку ж, коли введені дані коректно ідентифіковані системою, користувач отримує доступ до головного вікна програми, що свідчить про успішне проходження процедури аутентифікації та авторизації. Таким чином, авторизація завершується, і користувач отримує доступ до ресурсів системи відповідно до своїх прав і привілеїв.

Цей процес є стандартним для більшості інформаційних систем і відіграє важливу роль у забезпеченні безпеки інформаційних ресурсів, а також у забезпеченні контролю за доступом до чутливої інформації.

На рис. 3.3 показано блок–схему алгоритму, яка демонструє процес додавання інформації про нову категорію прогнозування у базу даних.

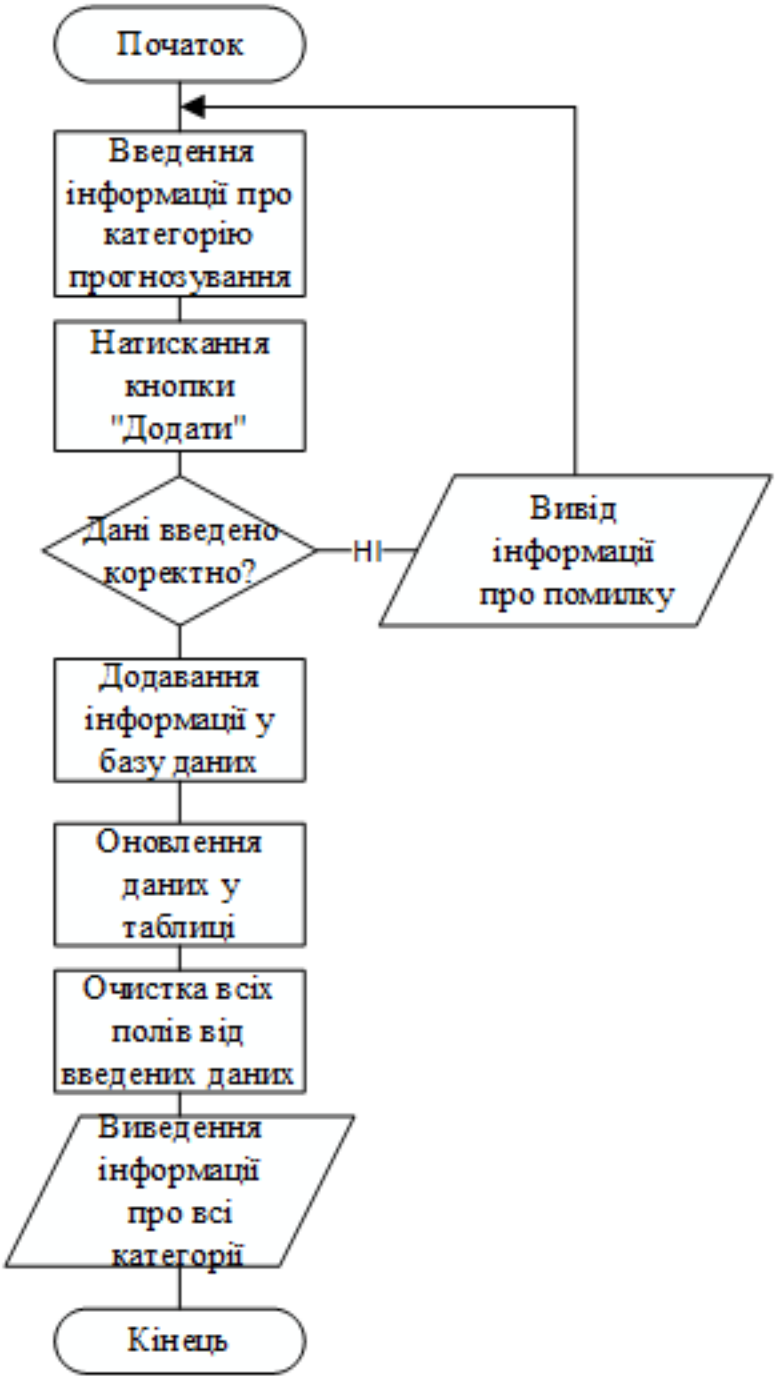

Рис. 3.3. Алгоритм процесу додавання нової категорії прогнозування

Процес додавання нової категорії прогнозування в інформаційну систему починається з моменту, коли користувач вводить необхідну інформацію в спеціально відведені для цього поля. Ця інформація може включати різноманітні деталі, такі як назва категорії, її опис, параметри, за якими буде здійснюватися прогнозування, та інші важливі характеристики.

Після введення всієї необхідної інформації користувач натискає кнопку «Додати», що ініціює перевірку введених даних на предмет їх коректності та повноти.

Якщо в ході перевірки виявляються помилки або неповні дані, система генерує повідомлення про помилку і пропонує користувачу перейти до виправлення помилок. У такому випадку користувач знову повертається до початкового етапу введення інформації про категорію прогнозування.

Якщо ж усі дані введено правильно, інформація про нову категорію прогнозування додається до бази даних. Після успішного додавання відбувається оновлення даних у таблиці, де відображаються усі категорії прогнозування. Це оновлення дозволяє відразу ж побачити нову категорію разом із іншими наявними в системі.

Далі система очищає всі поля від введених раніше даних, що дозволяє користувачу приступити до введення інформації про іншу категорію, якщо це необхідно. Останнім кроком є виведення інформації про всі наявні категорії прогнозування, що дозволяє користувачам системи бачити повний перелік категорій і відповідно орієнтуватися в наявних опціях прогнозування.

Таким чином, процес додавання нової категорії прогнозування є важливим для забезпечення актуальності та повноти інформаційної бази, а також для забезпечення можливості адекватного та ефективного використання системи прогнозування.

На рис. 3.4 показана блок–схема редагування інформації вибраного запису із таблиці та виведення інформації про всі категорії прогнозування, інформація про які збережена у базі даних.

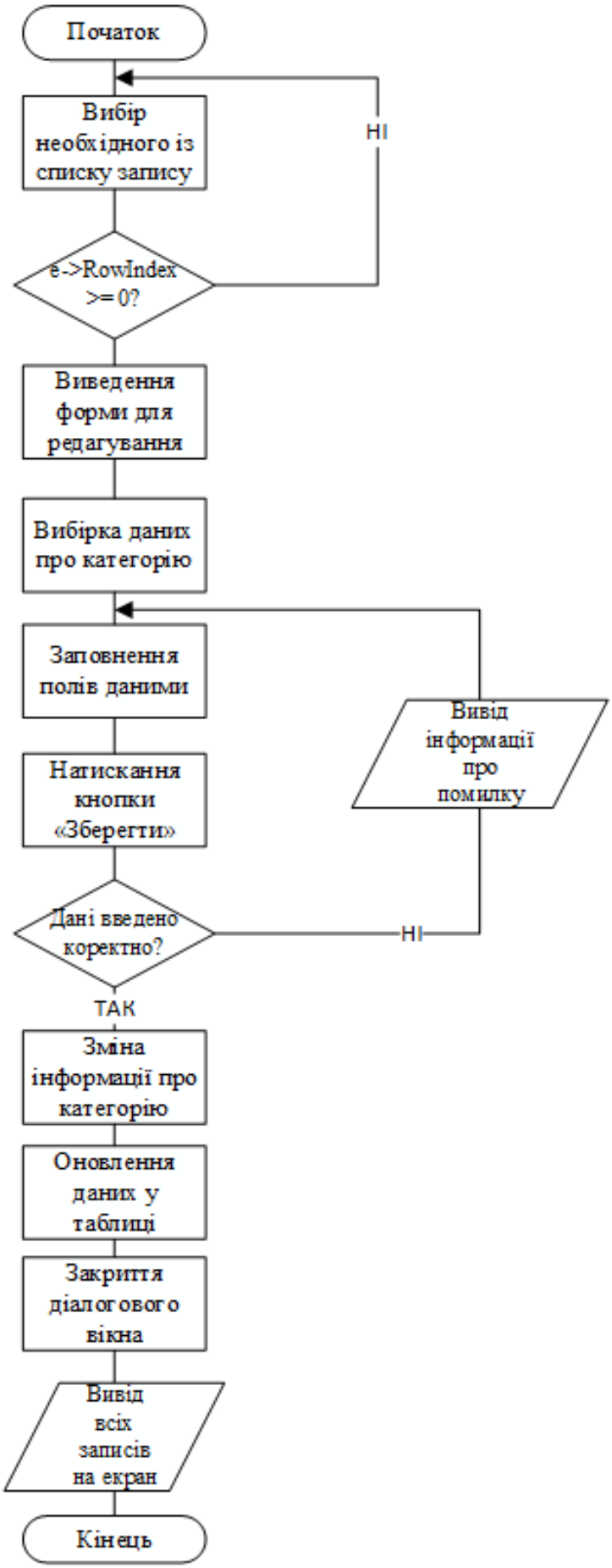

Рис. 3.4. Алгоритм редагування інформації вибраного запису

Алгоритм редагування інформації про обрану зі списку категорію прогнозування розпочинається з вибору потрібного запису користувачем. Користувач переглядає доступний список категорій та вибирає ту, інформацію про яку потрібно відредагувати. Система перевіряє, чи вибір був здійснений коректно (тобто, чи обраний запис дійсно існує в списку), і у випадку невірного вибору пропонує користувачу здійснити вибір знову.

Після вдалого вибору запису система відкриває діалогове вікно для редагування, де користувачу представляється можливість внести зміни. На цьому етапі система здійснює вибірку даних про обрану категорію прогнозування за допомогою переданого ідентифікатора. Інформація, яка вже є в базі даних (наприклад, назва категорії та її опис), автоматично заповнює відповідні поля у формі редагування.

Користувач вносить потрібні зміни до даних та натискає кнопку «Зберегти», ініціюючи процес перевірки коректності введених даних. Якщо виявлено помилки або неповноту даних, система виводить відповідне повідомлення про помилку та пропонує користувачу повернутися до редагування. У випадку коректного введення даних, система здійснює зміну інформації про категорію прогнозування в базі даних.

Після успішного оновлення інформації відбувається автоматичне оновлення даних у таблиці, де відображені всі категорії, що дозволяє користувачам побачити зроблені зміни відразу. Діалогове вікно редагування закривається, і на екран виводиться повний список усіх категорій прогнозування, щоб користувач міг переглянути актуальну інформацію.

Цей алгоритм є важливим елементом управління даними в інформаційній системі, оскільки забезпечує можливість актуалізації та коректування інформації з мінімальними затримками та забезпечує високий рівень точності та актуальності даних, які використовуються для прогнозування.

Рис. 3.5 відображає блок–схему видалення вибраного запису із бази даних та виведення інформації про всі категорії прогнозування, що були збережені.

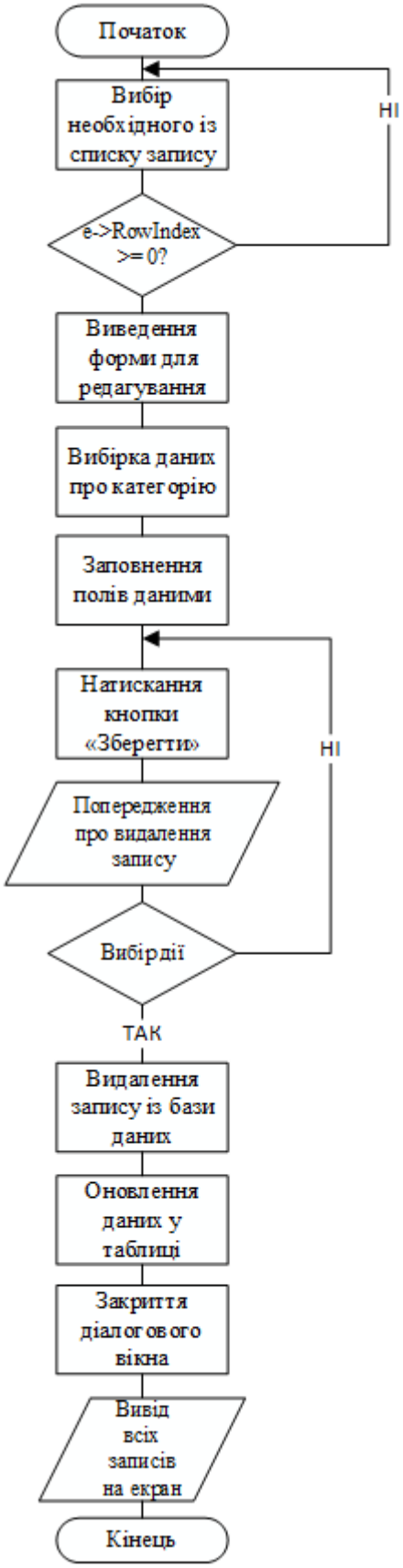

Рис. 3.5. Алгоритм процесу видалення категорії

Алгоритм видалення інформації про обрану зі списку категорію прогнозування ініціюється з вибору користувачем конкретного запису із списку. Користувач

переглядає доступні категорії та вибирає ту, яку бажає видалити. Система перевіряє, чи було зроблено вибір коректно, переконуючись, що користувач обрав існуючий запис.

Після успішного вибору запису система відкриває діалогове вікно, де користувач може підтвердити свій намір видалити обрану категорію. У цьому вікні відображається інформація про категорію, отримана за допомогою переданого ідентифікатора, включаючи назву категорії та її опис.

Далі користувач натискає кнопку «Видалити». Перед самим видаленням запису система виводить попередження про можливі наслідки цієї дії, запитуючи у користувача підтвердження. Якщо користувач відмовляється від видалення, процес повертається до початку, дозволяючи користувачу знову переглянути інформацію або здійснити інші дії. У випадку підтвердження видалення, запис видаляється з бази даних.

Після видалення запису відбувається оновлення даних у таблиці, щоб відображати актуальний стан інформації без видаленої категорії. Діалогове вікно закривається, і на екрані з'являється оновлений список усіх категорій прогнозування.

Цей алгоритм дозволяє забезпечити актуальність даних у системі, видаляючи застарілу або непотрібну інформацію. Важливо, що система надає користувачу достатньо інформації для прийняття обдуманого рішення про видалення, а також запобігає випадковому видаленню важливих даних.

3.3 Програмна реалізація алгоритмів роботи системи

Під час розробки системи «Моніторинг технічного стану авіаційного обладнання» у Visual Studio 2022, на початковій стадії була додана ключова змінна CONNECTING до конфігураційного файлу App.config. Вона містить параметри для підключення до бази даних, що представлені на рис. 3.6.

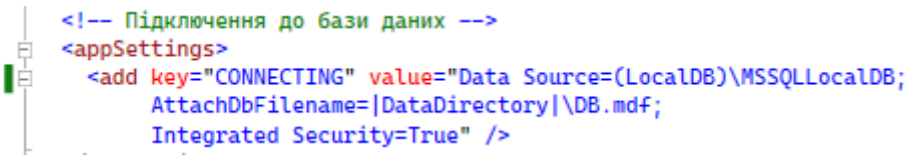

Рис. 3.6. Конфігурація з'єднання з базою даних

В контексті розробки системи було обрано використовувати простір імен System.Data.SqlClient зі стандартної бібліотеки .NET, щоб забезпечити ефективне спілкування з реляційною базою даних. System.Data.SqlClient спеціалізується на взаємодії з SQL Server, надаючи інструменти для управління підключеннями та виконання SQL–запитів. Використання цього простору імен підсилює безпеку та контроль при роботі з базою даних, знижуючи можливість помилок у обробці даних.

Збереження та обробка конфіденційної інформації вимагає високого рівня безпеки, особливо коли йдеться про дані, що зберігаються в базах даних. З метою забезпечення цієї безпеки, розроблено набір методів для шифрування та розшифрування інформації, які використовуються в системі.

Метод шифрування розроблений з використанням алгоритму DES (Data Encryption Standard), який є надійним та перевіреним часом вибором для шифрування. Цей метод приймає вхідний рядок (інформацію, що потрібно зашифрувати) і повертає рядок, закодований у форматі Base64. Перед шифруванням виконується перевірка на відсутність нульових або пустих вхідних рядків, щоб запобігти будь–яким помилкам у процесі шифрування. Цей метод гарантує, що всі дані, перед відправленням до бази даних або перед зберіганням у ній, будуть належним чином захищені. Код даного методу представлено на рис. 3.7.

```
// Метод для шифрування рядка
3 references
public string Encrypt(string originalString) {
  // Перевірка, чи вхідний рядок не пустий або не null
  if (String.IsNullOrEmpty(originalString)) {
    throw new ArgumentNullException("Рядок, який потрібно зашифрувати, не може бути нульовим.");
  }
  // Використання DES-шифрування
  using (DESCryptoServiceProvider cryptoProvider = new DESCryptoServiceProvider()) {
    using (MemoryStream memoryStream = new MemoryStream()) {
      // Створення потоку для шифрування
      CryptoStream cryptoStream = new CryptoStream(memoryStream,
          cryptoProvider.CreateEncryptor(bytes, bytes), CryptoStreamMode.Write);

      // Запис шифрованого рядка у потік
      using (StreamWriter writer = new StreamWriter(cryptoStream)) {
        writer.Write(originalString);
      } // StreamWriter та CryptoStream закриваються автоматично тут через 'using'

      // Повертаємо шифрований рядок у вигляді Base64
      return Convert.ToBase64String(memoryStream.ToArray());
      // Використовуємо ToArray для отримання байтів із MemoryStream
    }
  }
}
```

Рис. 3.7. Код методу для шифрування даних

Метод розшифрування працює у тандемі з методом шифрування. Він приймає зашифрований рядок у форматі Base64 і повертає вихідний, розшифрований рядок. Цей процес також включає перевірку на нульовість та пустоту вхідних даних. Застосування цього методу забезпечує можливість безпечного отримання та використання інформації з бази даних, гарантуючи, що всі дані, які були зашифровані, можуть бути ефективно та безпечно розшифровані для подальшої обробки або відображення. Код методу розшифрування інформації представлено на рис. 3.8.

```
// Метод для дешифрування рядка
4 references
public string Decrypt(string cryptedString) {
  // Перевірка, чи вхідний рядок не пустий або не null
  if (String.IsNullOrEmpty(cryptedString)) {
    throw new ArgumentNullException("Рядок, який потрібно розшифрувати, не може бути нульовим.");
  }
  // Використання DES-дешифрування
  using (DESCryptoServiceProvider cryptoProvider = new DESCryptoServiceProvider()) {
    using (MemoryStream memoryStream = new MemoryStream(Convert.FromBase64String(cryptedString))) {
      // Створення потоку для дешифрування
      using (CryptoStream cryptoStream = new CryptoStream(memoryStream,
          cryptoProvider.CreateDecryptor(bytes, bytes), CryptoStreamMode.Read)) {
        // Читання дешифрованого рядка із потоку
        using (StreamReader reader = new StreamReader(cryptoStream)) {
          return reader.ReadToEnd();
        } // StreamReader та CryptoStream закриваються автоматично тут через 'using'
      }
    }
  }
}
```

Рис. 3.8. Код методу для розшифрування даних

Обидва методи, шифрування та розшифрування, розроблені з урахуванням потреб захисту конфіденційної інформації та вимог до ефективності системи. Вони забезпечують надійний захист даних під час їх передачі та зберігання, мінімізуючи ризики несанкціонованого доступу або витоку інформації. Такий підхід до шифрування є важливою складовою загальної стратегії забезпечення безпеки даних у рамках системи моніторингу технічного стану авіаційного обладнання.

З точки зору користувацького інтерфейсу, важливою частиною системи стала інтеграція компоненту MenuStrip в головний інтерфейс Windows Forms. MenuStrip забезпечує легкість управління та дозволяє створювати графічні меню з різними функціональними можливостями, полегшуючи користувачам навігацію та забезпечуючи прямий доступ до основних функцій системи (рис. 3.9).

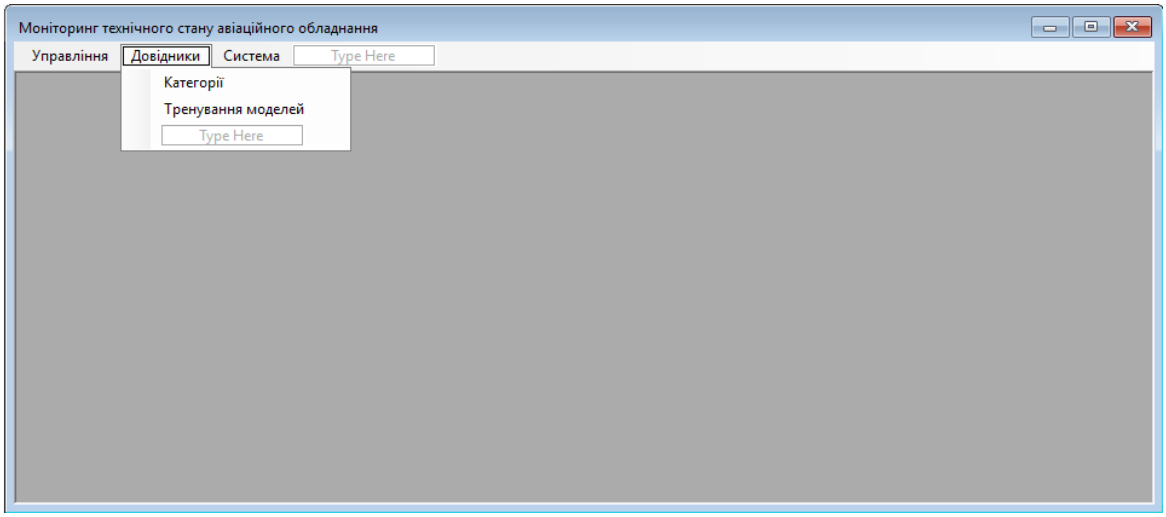

Рис. 3.9. Головна форма системи

Для кожного елемента меню було розроблено унікальну реакцію системи, яка активується при його виборі. Наприклад, як видно на рис. 3.10, вибір опції «Тестування моделей» ініціює відкриття відповідного набору інструментів для тестування та аналізу моделей.

```
1 reference
private void тестуванняМоделейToolStripMenuItem_Click(object sender, EventArgs e) {
    CloseAllWindows();
    PredictorForm predictorForm = new PredictorForm();
    predictorForm.MdiParent = this;
    predictorForm.WindowState = FormWindowState.Maximized;
    predictorForm.Show();
}
```

Рис. 3.10. Активація форми «Тестування моделей»

Також було розроблено спеціальний інтерфейс для тренування моделей машинного навчання, зразок якого можна побачити на рис. 3.11.

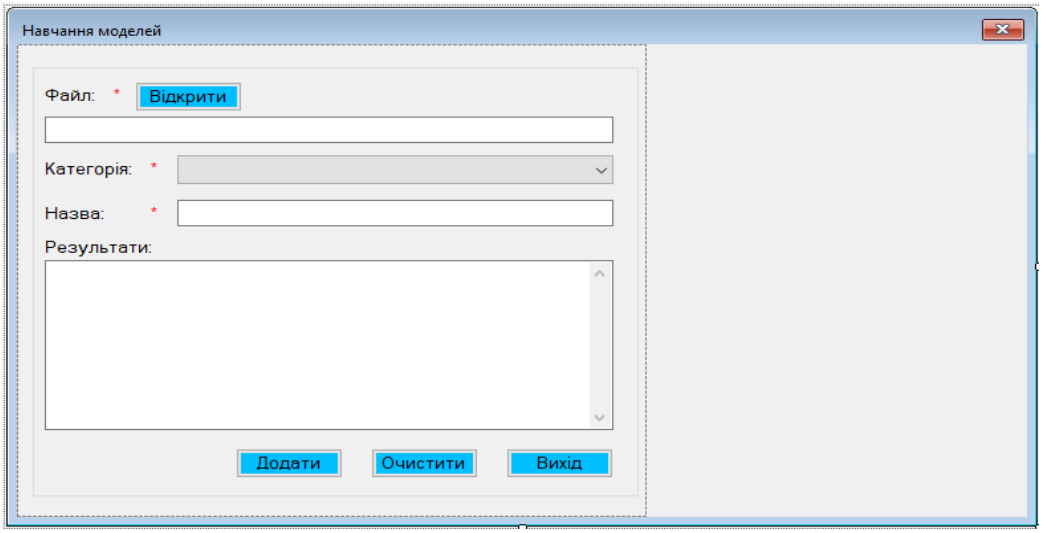

Рис. 3.11. Форма навчання моделей

На рис. 3.12 представлено фрагмент коду, який уможливлює вибір файлу користувачем із його локальної файлової системи. Цей процес також дозволяє програмі отримати доступ до шляху обраного файлу для подальшого його використання.

```
// Відображення діалогового вікна та обробка результату
if (openFileDialog.ShowDialog() == DialogResult.OK) {

  _Path = openFileDialog.FileName;
  FileNameTBox.Text = openFileDialog.FileName;
```

Рис. 3.12. Кодовий фрагмент для відкриття тренувального файлу

Процедура ініціалізації контексту машинного навчання у програмі відображена на рис. 3.13.

```
//Створення контексту
mlContext = new MLContext(seed: 0);
```

Рис. 3.13. Створення контексту

У цій частині коду відбувається створення контексту машинного навчання за допомогою ML.NET, фреймворку від Microsoft для реалізації машинного навчання у .NET середовищі. Застосування конструкції «new MLContext» ініціює створення нового екземпляра контексту машинного навчання, який служить як основа для подальшої роботи з алгоритмами машинного навчання.

Параметр «seed: 0», встановлений у конструкторі MLContext, задає стартове значення для внутрішнього генератора випадкових чисел у контексті. Вказівка конкретного seed значення забезпечує консистентність результатів при кожному тренуванні моделей, що є критично важливим для їх тестування та порівняння.

Після цього проходить завантаження даних із текстового файлу для подальшого використання у машинному навчанні (рис. 3.14).

```
trainingDataView = mlContext.Data.LoadFromTextFile<AviationEquipmentData>(
    path: FileNameTBox.Text,
    separatorChar: ',',
    hasHeader: true);
```

Рис. 3.14. Завантаження даних

Код виконує завантаження даних для тренування з текстового файлу у контекст машинного навчання. Цей процес відбувається за допомогою функції LoadFromTextFile від ML.NET, яка є частиною бібліотеки Microsoft для машинного навчання. Функція приймає кілька параметрів для коректного завантаження та інтерпретації даних:

– тип даних. Функція використовує тип AviationEquipmentData для вказівки на структуру даних, яка відповідає формату рядків у текстовому файлі. Це дозволяє системі правильно інтерпретувати кожен рядок файлу як окремий запис з відповідними полями;

– шлях до файлу. Шлях до файлу передається через змінну FileNameTBox.Text. Це означає, що шлях до файлу береться із текстового поля в графічному інтерфейсі, де користувач може ввести або вибрати потрібний файл;

– роздільник. В якості роздільника даних у файлі використовується символ коми (,). Це стандартний роздільник для CSV–файлів, що дозволяє ефективно розбивати дані на окремі поля;

– наявність заголовку. Параметр hasHeader встановлений як true, що вказує на наявність рядка заголовка у файлі. Завдяки цьому перший рядок файлу розглядається як опис колонок, а не як дані.

Після виконання цього коду, змінна trainingDataView містить дані, завантажені з вказаного файлу, структуровані згідно з визначеним типом AviationEquipmentData.

Ці дані готові до подальшого використання у процесах машинного навчання, таких як тренування моделей.

Дані необхідно провести розділення набору даних на тренувальний і тестовий набори в контексті машинного навчання за допомогою ML.NET, фреймворку від Microsoft (рис. 3.15). Це розділення дозволяє оцінити ефективність моделі на даних, які не використовувались під час тренування.

```
var trainTestSplit = mlContext.Data.TrainTestSplit(trainingDataView);
var trainData = trainTestSplit.TrainSet;
var testData = trainTestSplit.TestSet;
```

Рис. 3.15. Розділення даних із тренувального набору

Спочатку виконується функція TrainTestSplit з ML.NET, яка приймає вхідний набір даних. Ця функція розділяє вхідний набір даних (trainingDataView) на дві частини: набір для тренування моделі та набір для її тестування. Розділення зазвичай виконується таким чином, щоб більша частина даних використовувалася для тренування, а менша – для тестування.

Після розділення, тренувальний набір даних зберігається у змінній trainData. Цей набір включає частину оригінальних даних, яка використовуватиметься для тренування моделі машинного навчання. Аналогічно, тестовий набір даних зберігається у змінній testData. Цей набір містить залишок даних, який не використовувався під час тренування. Він використовуватиметься для оцінки точності та ефективності навченої моделі.

Таким чином, цей код ефективно підготовлює дані для двоетапного процесу машинного навчання: спочатку тренування моделі на одному наборі даних, а потім оцінювання її продуктивності на іншому, незалежному наборі даних. Це забезпечує більш об'єктивну оцінку моделі, оскільки тестування відбувається на даних, які модель раніше не «бачила».

Після цього потрібно підготувати дані для машинного навчання (рис. 3.16).

```
var pipeline = mlContext.Transforms.Conversion.ConvertType(new[] {
          new InputOutputColumnPair("Temperature", "Temperature"),
          new InputOutputColumnPair("Pressure", "Pressure"),
          new InputOutputColumnPair("Vibration", "Vibration"),
          new InputOutputColumnPair("OilLevel", "OilLevel"),
          new InputOutputColumnPair("HoursOfOperation", "HoursOfOperation"),
          new InputOutputColumnPair("LastMaintenanceDate", "LastMaintenanceDate"),
          new InputOutputColumnPair("WearLevel", "WearLevel")
      }, DataKind.Single)
.Append(mlContext.Transforms.Concatenate("Features", new[] {
          "Temperature", "Pressure", "Vibration", "OilLevel",
          "HoursOfOperation", "LastMaintenanceDate", "WearLevel" }))
.Append(mlContext.BinaryClassification.Trainers.FastTree(
    numberOfLeaves: 50, numberOfTrees: 50, minimumExampleCountPerLeaf: 1));
```

Рис. 3.16. Підготовка конвеєру даних для навчання

Спочатку створюється конвеєр обробки даних (pipeline). Цей конвеєр включає кілька етапів, перший з яких – перетворення типів даних. Для ряду входових колонок даних (таких як «Temperature», «Pressure», «Vibration» тощо) виконується конвертація їх у тип Single (що є форматом з плаваючою комою). Це забезпечує уніфікацію типів даних, що важливо для подальшої обробки.

Наступним кроком є об'єднання різних входових колонок у єдину колонку «Features». Це робиться за допомогою методу Concatenate, який збирає окремі колонки в одну для зручності тренування моделі. Таким чином, колонки «Temperature», «Pressure», «Vibration» та інші стають частиною одного набору ознак, який використовується як вхід для моделі машинного навчання.

Останнім етапом у конвеєрі є додавання тренера для бінарної класифікації. У цьому випадку використовується алгоритм FastTree для бінарної класифікації. Алгоритм FastTree є популярним вибором для задач класифікації та регресії і використовує методи ансамблевого навчання. Параметри, такі як numberOfLeaves, numberOfTrees та minimumExampleCountPerLeaf, визначають конфігурацію тренування, впливаючи на кількість листів та дерев у моделі та мінімальну кількість прикладів, необхідних для листа.

Виконавши цей код, отримується готовий до тренування конвеєр, який послідовно перетворює вхідні дані, готує набір ознак і використовує алгоритм ШД для класифікації даних. Це дозволяє ефективно підготувати модель для виявлення патернів або здійснення прогнозувань на основі навчального набору даних.

Після цього потрібно реалізувати підготовку та тренування моделі бінарної класифікації в середовищі машинного навчання ML.NET. Він структурований як послідовність кроків, кожен з яких виконує певну операцію, що є важливою для процесу підготовки та тренування моделі.

Далі необхідно обрати алгоритм машинного навчання (рис. 3.17).

```
//Створення тренера
var trainer = mlContext.BinaryClassification.Trainers.FastTree("Label", "Features");
```

Рис. 3.17. Вибір алгоритму

Процес створення тренера для задач бінарної класифікації використовується алгоритм FastTree, який є популярним вибором для задач класифікації. У коді створюється об'єкт тренера за допомогою методу, який приймає два аргументи:

– перший аргумент «Label» визначає назву колонки в наборі даних, яка містить мітки класу. Мітки класу використовуються для вказівки, до якого класу належить кожен екземпляр у навчальному наборі даних. У задачах бінарної класифікації ці мітки зазвичай мають два можливих значення, які представляють два класи;

– другий аргумент «Features» вказує на назву колонки, яка містить вектор ознак для кожного екземпляра в наборі даних. Ознаки – це вхідні змінні моделі, які використовуються для виконання прогнозів.

В результаті виконання цього коду, отримується тренер моделі, готовий до використання у процесі тренування. Цей тренер використовуватиме дані з вказаних колонок «Label» і «Features» для створення моделі, яка здатна класифікувати нові екземпляри на основі навчених ознак і закономірностей.

Після цього необхідно створити тренувальний пайплайн для машинного навчання, який формується шляхом додавання (append) об'єкту тренера, який був визначений раніше, до пайплайну обробки даних (рис. 3.18).

```
var trainingPipeline = pipeline.Append(trainer)
    .Append(mlContext.Transforms.Conversion.MapKeyToValue("PredictedLabel", "Label"));
```

Рис. 3.18. Створення тренувального пайплайну

У цьому фрагменті коду реалізується створення та розширення конвеєра тренування для машинного навчання. Описана дія є частиною процесу підготовки

моделі в середовищі ML.NET. Основні компоненти та кроки цього процесу включають:

– додавання тренера до конвеєра. Спочатку до вже існуючого конвеєра (визначеного як pipeline), який містить різні етапи передобробки даних, додається тренер моделі (визначений як trainer). Тренер – це компонент, відповідальний за власне навчання моделі, використовуючи вхідні дані, що пройшли через всі етапи передобробки;

– перетворення прогнозованих міток. Після тренера в конвеєр додається етап преобразування, який перетворює ключі прогнозованих міток (представлених моделлю) назад у зрозумілі та читабельні значення. Цей крок виконується за допомогою методу MapKeyToValue. Він приймає назву вихідної колонки з прогнозованими мітками («PredictedLabel») і відображає її значення назад на відповідні значення оригінальних міток, визначених у колонці «Label». Це дозволяє легше інтерпретувати результати моделі після тренування, оскільки прогнозовані мітки тепер будуть відображені у форматі, зрозумілому для користувачів.

В результаті, trainingPipeline стає повноцінним конвеєром, що включає в себе всі необхідні кроки для тренування моделі: від передобробки даних до тренування і трансформації результатів. Цей конвеєр може бути використаний для тренування моделі на наборі даних та отримання прогнозів зі зрозумілими мітками.

Наступним кроком відбувається процес тренування моделі машинного навчання в середовищі ML.NET (рис. 3.19). Ця дія є ключовим кроком у побудові моделі, яка зможе робити прогнози або класифікації на основі навчальних даних.

```
//Навчання моделі
model = pipeline.Fit(trainingDataView);
```

Рис. 3.19. Навчання моделі

Процес тренування моделі відбувається наступним чином:

– використання раніше створеного конвеєра. Конвеєр, який був визначений і сконфігурований раніше (в цьому контексті відомий як pipeline), включає в себе всі етапи обробки даних, включаючи передобробку, функції трансформації та алгоритм тренування моделі;

– застосування конвеєра до набору даних. Функція Fit цього конвеєра застосовується до набору навчальних даних (у цьому випадку, вказаних як trainingDataView). Цей набір даних містить приклади, на основі яких модель буде навчатися;

– процес тренування моделі. Під час виконання функції Fit, конвеєр автоматично виконує всі визначені в ньому кроки обробки даних та тренує модель, використовуючи алгоритм, вказаний в конвеєрі;

– створення тренованої моделі. По завершенню процесу тренування, результатом виконання цього коду є тренована модель (у цьому випадку, збережена в змінній model). Ця модель тепер може бути використана для здійснення прогнозів або оцінки на нових даних, які не входили в навчальний набір.

Отже, виконання цього коду у процесі побудови моделі машинного навчання перетворює підготовлені та передоброблені дані у готову до використання модель.

Після цього відбувається процес оцінки результатів навчання моделі та виведення її метрик у відповідне поле (рис. 3.20).

```
// Оцінка моделі
var predictions = model.Transform(testDataView);
var metr = mlContext.BinaryClassification.Evaluate(predictions, "Label", "Probability");

RaportTBox.Text += ($"Точність: {metr.Accuracy:P2}") + "\r\n";
RaportTBox.Text += ($"AUC: {metr.AreaUnderRocCurve:P2}") + "\r\n";
RaportTBox.Text += ($"F1 Score: {metr.F1Score:P2}") + "\r\n";
RaportTBox.Text += ($"Precision: {metr.PositivePrecision:P2}") + "\r\n";
RaportTBox.Text += ($"Recall: {metr.PositiveRecall:P2}") + "\r\n";
```

Рис. 3.20. Оцінка моделі та виведення її метрик

Цей процес оцінки якості тренованої моделі машинного навчання перевіряє наскільки добре модель працює на даних, які не використовувались під час тренування. Ось основні кроки цього процесу:

застосування моделі до тестових даних. Спочатку, вже тренована модель застосовується до набору тестових даних. Це робиться за допомогою методу Transform, який перетворює тестові дані, використовуючи модель для отримання прогнозів. Результатом цього перетворення є набір прогнозованих значень, збережений у змінній predictions;

виконання оцінки моделі. Далі виконується оцінка моделі за допомогою методу Evaluate. Цей метод аналізує прогнози моделі, порівнюючи їх з фактичними мітками класів у тестовому наборі даних. В результаті отримуються метрики, які демонструють ефективність моделі. Зокрема, використовуються такі метрики, як точність, площа під ROC–кривою (AUC), F1 Score, Precision та Recall;

відображення результатів оцінки. Останнім кроком є відображення отриманих метрик. Результати оцінки (такі як точність, AUC, F1 Score, Precision, Recall) виводяться у текстове поле, імовірно, у графічному інтерфейсі користувача. Для кожної метрики використовується форматування відсотків (наприклад, {metr.Accuracy:P2}), що дозволяє легко інтерпретувати результати.

Код дає можливість зрозуміти, наскільки добре модель здатна прогнозувати результати на нових, раніше невідомих даних, що є важливим для визначення її практичної придатності.

Для збереження даних навчання моделі реалізовано подію «AddBtn_Click», код якої представлено на рис. 3.21.

```
private void AddBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    //Save model
    string pathName = @"\teach\" + GenerateFileName() + ".zip";
    string localProj = System.IO.Path.GetDirectoryName(
      System.Reflection.Assembly.GetExecutingAssembly().Location);
    _NeuralNetworkProvider.InsertNeuralNetwork(NeuralNetworkNamesTBox.Text,
      Convert.ToInt32(CategoriesCBox.SelectedValue),
      pathName);
    mlContext.Model.Save(model, trainingDataView.Schema, localProj + pathName);
    ClearAllData();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
      "Було навчено модель " +
      NeuralNetworkNamesTBox.Text, DateTime.Now);
    MessageBox.Show("Дані успішно збережено!");
  }
}
```

Рис. 3.21. Метод збереження даних моделі

Код є типовим прикладом обробки подій в графічному інтерфейсі користувача, що включає взаємодію з моделлю машинного навчання, збереження результатів, логування подій та взаємодію з користувачем. Основні кроки коду виглядають наступним чином:

– перевірка коректності введених даних. Спочатку виконується перевірка введених даних за допомогою функції IsDataEnteringCorrect(). Ця функція, імовірно,

перевіряє, чи введені користувачем дані є валідними та повними для виконання наступних дій;

– збереження моделі. Якщо дані введені коректно, код продовжує збереження навченої моделі машинного навчання. Спочатку формується шлях та ім'я файлу для збереження моделі. Це включає генерацію назви файлу за допомогою методу GenerateFileName() та додавання розширення «.zip». Далі визначається локальний шлях до проекту. Метод InsertNeuralNetwork класу _NeuralNetworkProvider використовується для додавання інформації про нейронну мережу до якоїсь структури зберігання або бази даних. Нарешті, навчена модель зберігається у файлі за вказаним шляхом за допомогою методу Save;

– очищення даних. Після збереження моделі виконується метод ClearAllData(), який, ймовірно, очищує або скидає форми вводу або інтерфейс;

– запис у логи. Вноситься запис до системи логування про те, що модель було навчено. Це включає ідентифікатор користувача, назву навченої моделі, та час події;

– повідомлення користувачеві. На завершення виводиться повідомлення за допомогою MessageBox.Show, що інформує користувача про успішне збереження даних.

Для тестування навчених моделей було реалізовано форму «Тестування моделей», зовнішній вигляд якої представлено на рис. 3.22.

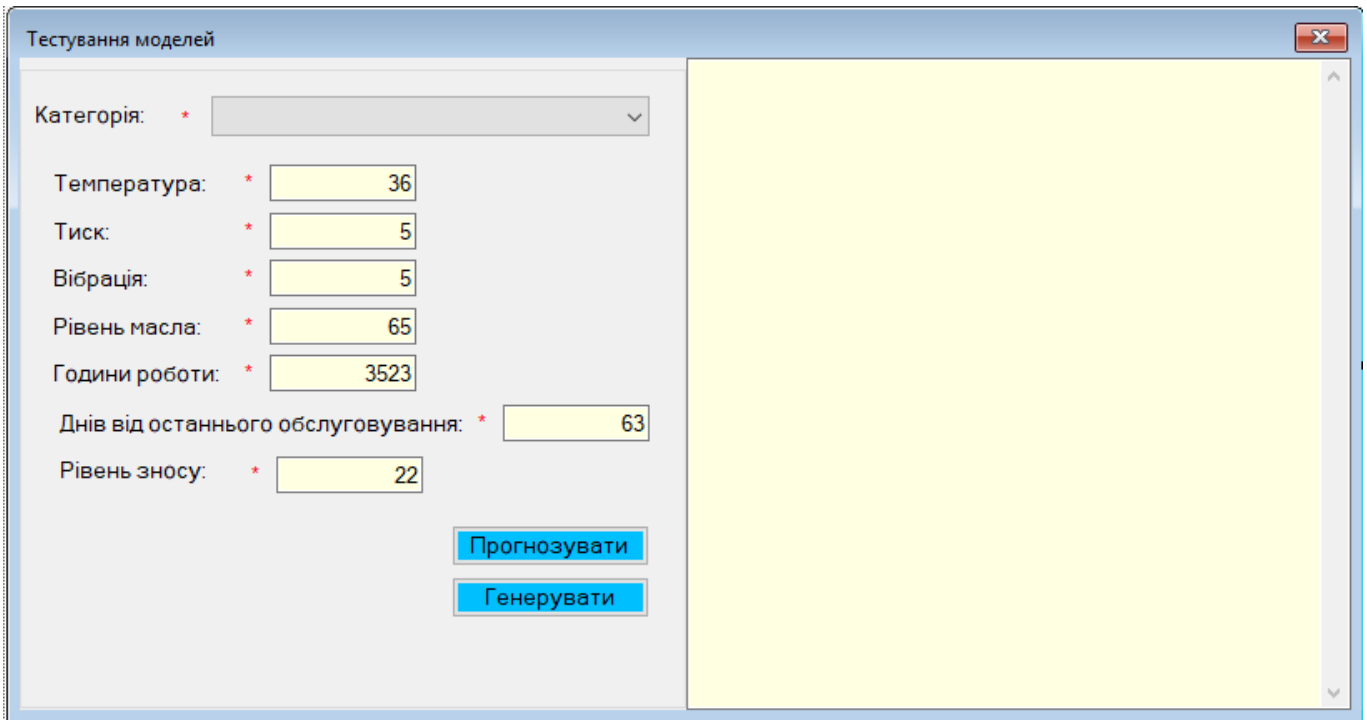

Рис. 3.22. Вигляд форми «Тестування моделей»

Для зміни моделі яку необхідно тестувати у випадаючому списку, реалізовано подію «CategoriesCBox_SelectedValueChanged» (рис. 3.23).

```
private void CategoriesCBox_SelectedValueChanged(object sender, EventArgs e) {
  if (_IsThemesLoad) {
    _SelectedNeural = _NeuralNetworkProvider.SelectedNeuralNetworkByCategoriesId(
      Convert.ToInt32(CategoriesCBox.SelectedValue));
    LoadData(_SelectedNeural.NeuralNetworkFileModel);
  }
}
```

Рис. 3.23. Код події зміни категорії прогнозування

Фрагмент коду представляє обробник події, який активується при зміні вибраного значення в комбо–боксі (CategoriesCBox) в графічному інтерфейсі користувача. Обробник події виконує наступні функції:

– перевірка умови. Початково виконується перевірка логічної змінної _IsThemesLoad. Ця змінна, ймовірно, використовується для визначення, чи були вже завантажені необхідні дані чи теми до цього моменту. Якщо умова виконується (тобто _IsThemesLoad є true), то процес продовжується;

– отримання вибраної нейронної мережі. Якщо умова перевірки виконується, викликається метод SelectedNeuralNetworkByCategoriesId з об'єкта _NeuralNetworkProvider. Цей метод відповідає за отримання інформації про вибрану модель на основі ідентифікатора, обраного в комбо–боксі. Ідентифікатор отримується

за допомогою властивості «SelectedValue», який перетворюється в ціле число. В результаті цієї дії отримується конкретна модель, яка зберігається в змінній «_SelectedNeural»;

– завантаження даних нейронної мережі. Після отримання вибраної нейронної мережі виконується метод «LoadData» з параметром «NeuralNetworkFileModel», що входить до складу об'єкта «_SelectedNeural». Цей метод відповідає за завантаження даних або моделі нейронної мережі для подальшого використання чи відображення в інтерфейсі.

Для реалізації процесу генерації даних тестування моделі або зупинки в залежності від її поточного стану розроблено подію «RunBtn_Click» (рис. 3.24).

```
1 reference
private void RunBtn_Click(object sender, EventArgs e) {
    if (timer1.Enabled) {
        timer1.Enabled = false;
        RunBtn.Text = "Запустити";
    } else {
        timer1.Enabled = true;
        RunBtn.Text = "Зупинити";
    }
}
```

Рис. 3.24. Код події «RunBtn_Click»

Метод описує обробник події, що виконується при натисканні кнопки (RunBtn) у графічному інтерфейсі користувача. Функціональність цього обробника події полягає у включенні або вимкненні таймера, а також зміні тексту на кнопці в залежності від поточного стану таймера. Ось детальний опис процесу:

Перевірка стану таймера: Спочатку код перевіряє, чи таймер (timer1) вже активний (тобто timer1.Enabled має значення true). Ця перевірка дозволяє визначити поточний стан таймера – чи він зараз запущений, чи зупинений.

Зупинка та зміна тексту кнопки: Якщо таймер активний, це означає, що він вже запущений, тому код зупиняє таймер, встановлюючи timer1.Enabled у false. Також змінюється текст на кнопці (RunBtn) на "Запустити", відображаючи, що зараз кнопка може бути використана для запуску таймера.

Запуск таймера та зміна тексту кнопки: Якщо ж таймер не активний на момент натискання кнопки, код активує таймер, встановлюючи timer1.Enabled у true. У цьому

випадку текст на кнопці змінюється на "Зупинити", вказуючи, що тепер натискання кнопки призведе до зупинки таймера.

Цей обробник події дозволяє користувачеві керувати роботою таймера в графічному інтерфейсі, забезпечуючи зручний механізм для запуску та зупинки таймера за допомогою однієї кнопки, яка динамічно змінює свій текст в залежності від стану таймера.

Також реалізовано метод, який дозволяє тестувати моделі за допомогою введення даних у ручному режимі (рис. 3.25).

```
private void PredictionsBtn_Click(object sender, EventArgs e) {
  if (IsAlAviationEquipmentDataInputCorrect()) {
    var datas = new AviationEquipmentData {
      Temperature = (float)Convert.ToDouble(TemperatureTBox.Text),  Pressure = (float)Convert.ToDouble(PressureTBox.Text),
      Vibration = (float)Convert.ToDouble(VibrationTBox.Text), OilLevel = (float)Convert.ToDouble(OilLevelTBox.Text),
      HoursOfOperation = (float)Convert.ToDouble(HoursOfOperationTBox.Text),
      LastMaintenanceDate = (float)Convert.ToDouble(LastMaintenanceDateTBox.Text),
      WearLevel = (float)Convert.ToDouble(WearLevelTBox.Text) };
    RaportTBox.Text = ($"Дані:\r\n" +
      $"Температура = {datas.Temperature}\r\n" +
      $"Тиск = {datas.Pressure}\r\n" +
      $"Вібрація = {datas.Vibration}\r\n" +
      $"Рівень масла = {datas.OilLevel}\r\n" +
      $"Години роботи = {datas.HoursOfOperation}\r\n" +
      $"Днів від останнього обслуговування = {datas.LastMaintenanceDate}\r\n" +
      $"Рівень зносу = {datas.WearLevel}\r\n");

    var prediction = predictor.Predict(datas);
    // Умовний оператор для перевірки значення IsFaulty
    if (prediction.IsFaulty) {
      // Якщо IsFaulty == true
      RaportTBox.Text += "Прогноз поломки: можлива поломка\r\n";
    } else {
      // Якщо IsFaulty != true
      RaportTBox.Text += "Прогноз поломки: обладнання справне\r\n";
    }
    // Додавання ймовірності поломки
    RaportTBox.Text += $"Ймовірність поломки: {prediction.Probability:P2}\r\n";

  }
}
```

Рис. 3.25. Метод тестування модель за допомогою введених даних

Метод "PredictionsBtn_Click" здійснює прогнозування стану авіаційного обладнання на основі введених даних. Процес прогнозування включає декілька етапів:

– перевірка коректності введених даних. Спочатку виконується перевірка валідності введених даних за допомогою методу IsAlAviationEquipmentDataInputCorrect. Цей метод перевіряє, чи всі необхідні поля були заповнені коректно.

– створення об'єкта з даними обладнання. Якщо дані введено правильно, створюється об'єкт datas класу AviationEquipmentData. Для цього використовуються значення з текстових полів інтерфейсу, перетворені в тип float;

– виведення введених даних у текстове поле. Далі інформація про введені дані виводиться в текстове поле (RaportTBox). Це включає відображення значень температури, тиску, вібрації, рівня масла, кількості годин роботи, днів від останнього обслуговування та рівня зносу;

– виконання прогнозування. Використовуючи модель прогнозування (predictor), виконується прогноз на основі даних обладнання. Метод Predict моделі застосовується до об'єкта datas, отримуючи прогнозовані значення;

– аналіз результатів прогнозування та виведення їх у текстове поле. На основі отриманих прогнозів виводиться інформація про потенційну поломку обладнання. Це включає перевірку поля IsFaulty в об'єкті прогнозу. В залежності від цього значення в текстове поле додається повідомлення про ймовірну поломку або про те, що обладнання справне. Також виводиться ймовірність поломки з відповідним форматуванням.

3.4 Тестування та верифікація програми

При аналізі методів прогнозування технічного стану авіаційного обладнання, особливо важливими є два підходи до тестування – метод покриття діагностичних параметрів та метод аналізу типових помилок обладнання.

Метод покриття діагностичних параметрів зосереджується на визначенні, які параметри обладнання перевіряються під час діагностичних тестів. Цей підхід дозволяє оцінити, який відсоток необхідних параметрів, таких як температура, тиск, вібрація тощо, був включений у процес тестування. Це допомагає ідентифікувати потенційно проблемні області, які можуть вимагати додаткового моніторингу або обслуговування, хоча не завжди може гарантувати виявлення всіх можливих

дефектів, оскільки високий рівень покриття параметрів не завжди вказує на відсутність проблем у обладнанні.

Метод аналізу типових помилок обладнання, з іншого боку, базується на ідентифікації та виявленні частих дефектів або зносу, які можуть виникати в авіаційному обладнанні. Тести в цьому випадку розробляються таким чином, щоб цілеспрямовано виявити ці звичні проблеми, дозволяючи ефективніше знаходити потенційні слабкі місця в технічному стані обладнання. Цей метод вимагає глибокого розуміння типових проблем та особливостей специфічного для авіації обладнання.

Обидва ці методи є критично важливими для всебічного процесу тестування та оцінювання авіаційного обладнання, і їх ефективність може варіюватися залежно від типу обладнання та специфічних умов експлуатації. Найчастіше оптимальним варіантом є комбінація цих та інших методів для детального аналізу та забезпечення надійності та безпеки авіаційного обладнання.

Тестування стану та надійності авіаційного обладнання є ключовою частиною його технічного обслуговування, забезпечуючи безпеку та ефективність його використання. У цьому проекті було приділено особливу увагу модульному тестуванню, яке дозволяє ефективно перевіряти функціональність окремих компонентів обладнання. Застосування спеціалізованих інструментів тестування дало можливість розробити та провести тести, оцінюючи ключові аспекти роботи системи. Тестування включало в себе відтворення сценаріїв використання обладнання, таких як моніторинг параметрів роботи, оцінка зносу важливих деталей та аналіз даних про технічне обслуговування. Це гарантувало, що критичні функції обладнання працюють належним чином і відповідають вимогам безпеки.

На рис. 3.26 представлено результати цих тестів, що відображають ефективність виконаного тестування у забезпеченні якості та надійності авіаційного обладнання.

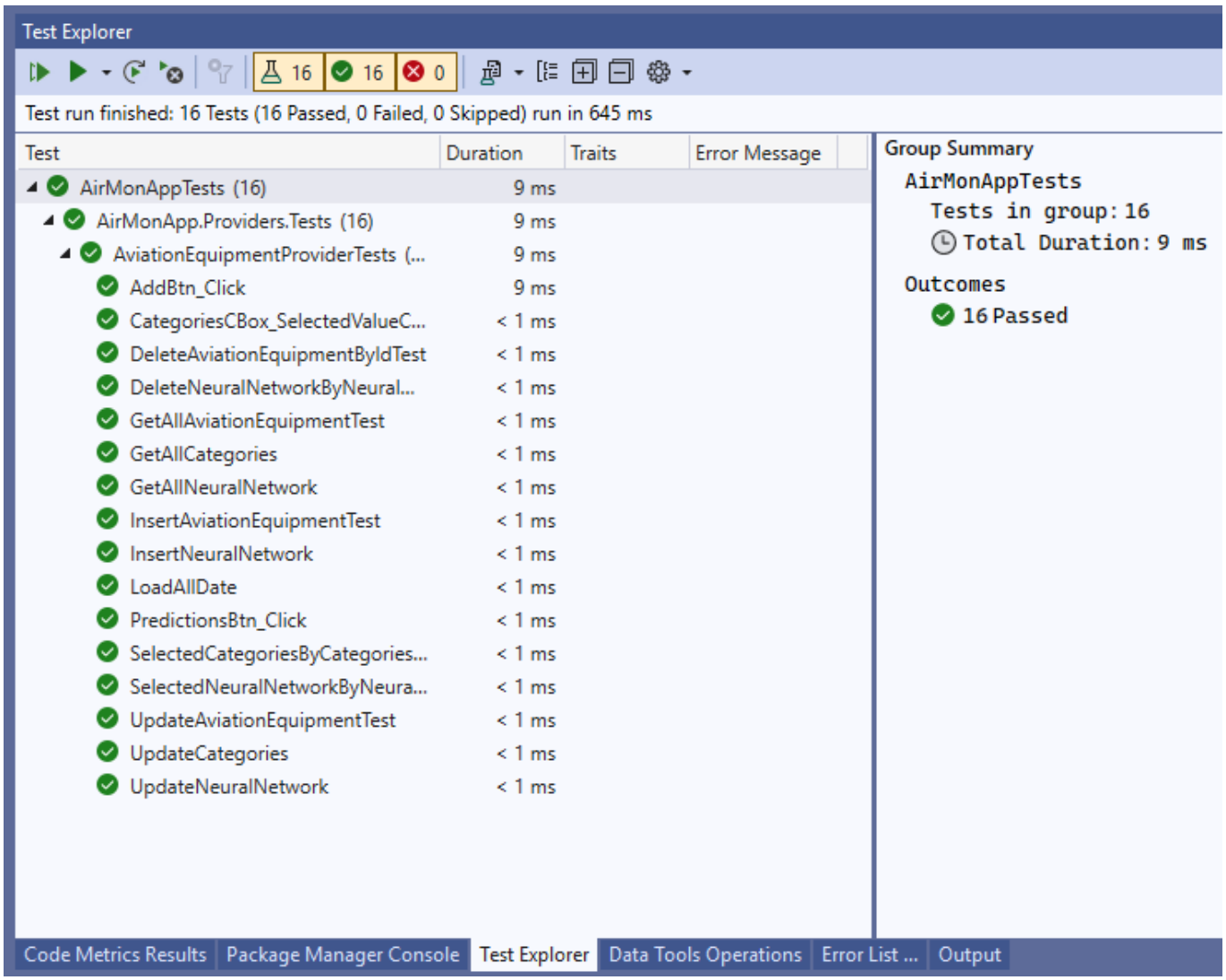

Рис.3.26. Результати проведеного модульного тестування

Ці дії забезпечили глибокий і детальний підхід до тестування, чий результат було узагальнено та представлено у вигляді візуальних звітів. Ці звіти, відображені на зазначеному рисунку, ілюструють обсяг тестування критичних компонентів системи та ефективність виявлення потенційних неполадок або дефектів.

В контексті тестування технічного стану авіаційного обладнання, методом припущення про похибку були створені сценарії, які імітують типові помилки, які можуть виникати під час моніторингу та аналізу даних обладнання.

Сценарій №1 «Перевірка коректності введених даних технічного стану»

Мета: Оцінити реакцію системи на введення некоректних або неповних технічних даних обладнання.

Процедура: Спроба внесення неповних або помилкових технічних параметрів обладнання, таких як температура, тиск або години роботи, без повного набору необхідних даних.

Очікуваний результат: Система відображає помилку, вказуючи на неправильне введення даних, і блокує подальше оброблення до виправлення цих помилок.

Сценарій №2 «Тестування реакції на видалення записів обладнання»

Мета: Перевірка, чи правильно система обробляє запит на видалення записів невикористовуваного або застарілого обладнання.

Процедура: Вибір запису обладнання зі списку та натискання кнопки «Видалити», після чого очікується запит на підтвердження видалення.

Очікуваний результат: Система видаляє обраний запис обладнання після підтвердження видалення та оновлює список, вилучаючи видалений елемент.

У контексті тестування системи моніторингу стану авіаційного обладнання, були розроблені додаткові тестові сценарії для методу припущення про похибку:

Сценарій №3 «Валідація вибору сценарію»

Мета: Забезпечити, що система коректно реагує на спробу використання сценарію, для якого не створено відповідної моделі бінарної класифікації.

Процедура: Виконання вибору сценарію зі списку, який не має відповідної моделі, з наступною спробою запустити симуляцію за цим сценарієм.

Очікуваний результат: Система відображає повідомлення про відсутність доступної моделі для обраного сценарію та запобігає реалізації симуляції.

Сценарій №4 «Проведення симуляції з випадковими даними»

Мета: Перевірити здатність системи до правильного прогнозування на основі даних, згенерованих випадковим чином.

Процедура: Створення випадкового набору даних за допомогою вбудованого генератора та ініціація процесу симуляції для отримання прогнозу.

Очікуваний результат: Система ефективно обробляє введені випадкові дані та надає прогноз щодо потенційних поломок, результати якого доступні для перегляду на панелі результатів.

Реалізовані тестові сценарії надали цінну можливість перевірки того, як інтерфейс користувача відповідає заданим критеріям та виявлення слабких місць у його логіці. Вони забезпечили ключову перевірку виконання симуляцій сценаріїв відповідно до очікувань, а також гарантували адекватну відповідь системи на непередбачені обставини, такі як відсутність необхідних даних для певного сценарію. Додатково, тестування з використанням випадково згенерованих даних дозволило оцінити точність та надійність прогнозів моделі, а також забезпечило ефективне введення даних та їх правильне відображення у системі.

3.5 Перевірка та валідація програми

У ході дослідження з моніторингу технічного стану авіаційного обладнання, ключовою була робота з даними високої якості та актуальності. Для цього було вибрано набір даних, доступний з відкритих джерел, який містив суттєві характеристики, пов'язані з технічним обслуговуванням, включаючи параметри такі як температура, тиск, рівень вібрації, години роботи, остання дата обслуговування та інші важливі індикатори стану обладнання. Ці дані давали змогу аналізувати різні аспекти, що впливають на технічний стан обладнання.

Перед проведенням експерименту було здійснено ретельну підготовку даних. Спершу було перевірено наявність аномалій і вилучено застарілі або пошкоджені записи. Після цього проведена нормалізація даних для забезпечення однакових масштабів вимірювань та усунення непропорційного впливу окремих параметрів на результати аналізу. Також було вирівняно формати даних для забезпечення точності їх обробки машинним навчанням.

Наступним кроком було тренування моделі, під час якого вона вчилася виявляти потенційні несправності та недоліки. Використовуючи репрезентативні приклади з набору даних, модель навчилася розпізнавати різноманітні типи можливих проблем. На рис. 3.27 показана частина інтерфейсу, де представлені дані,

підготовлені для тренування моделі, включаючи візуалізацію вхідних даних та їх підготовку до подачі у модель, що дозволило візуально оцінити якість та структуру даних перед їх застосуванням у процесі машинного навчання.

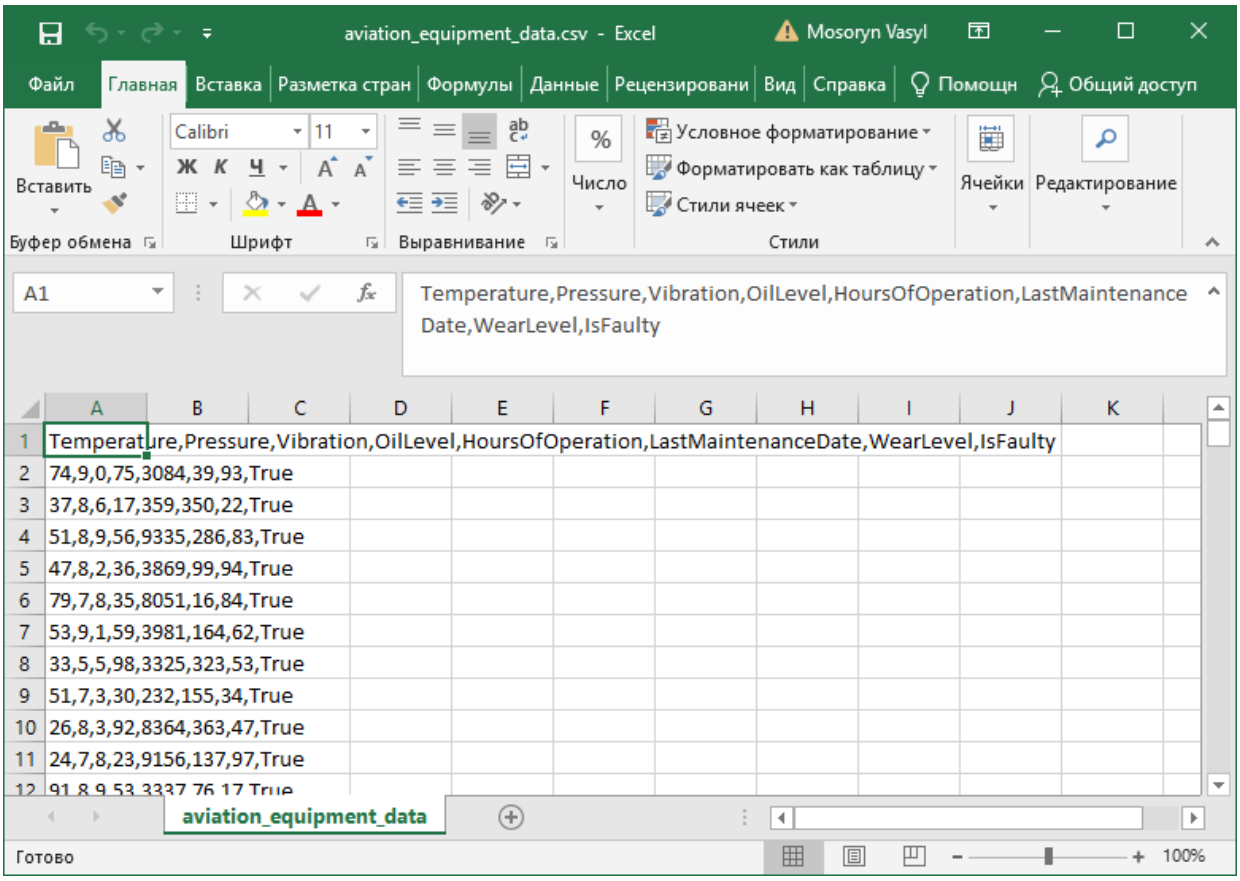

Рис. 3.27. Фрагмент підготовлених даних навчання моделі

Для тренування моделі у застосунку було створено категорію прогнозування із назвою «Категорія №1» (рис. 3.28).

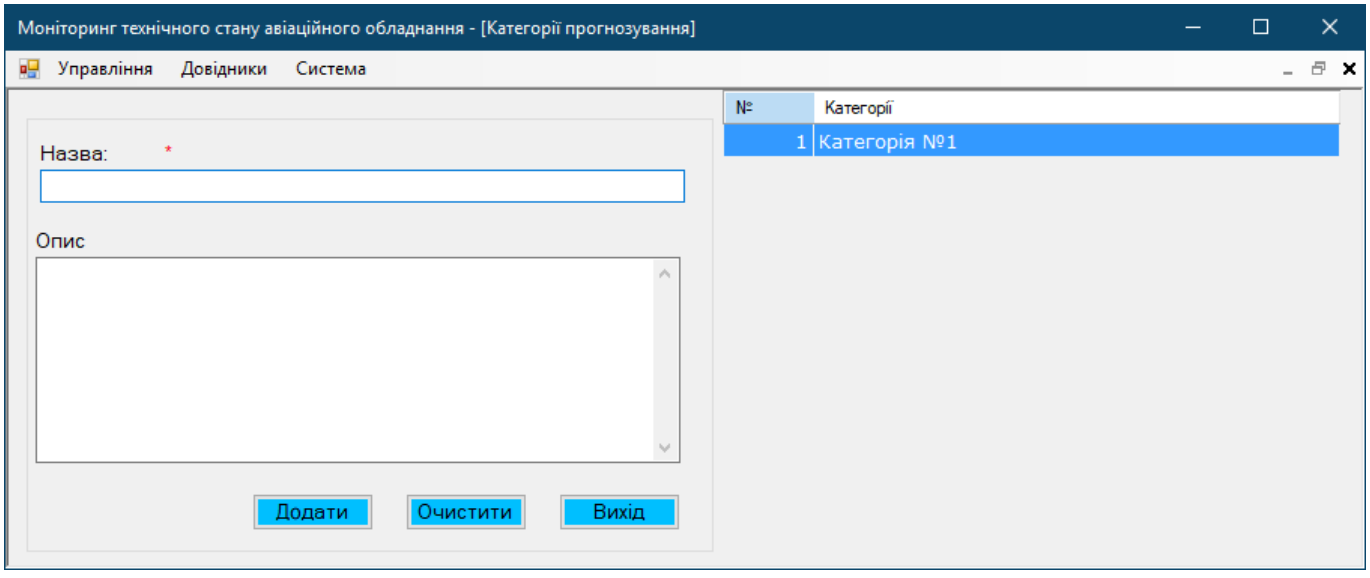

Рис. 3.28. Створення категорії прогнозування

Для тренування нової моделі створеної категорії згідно підготовленого датасету, було здійснено перехід по меню програми «Довідники» –> «Навчання моделей» та обрано створену категорію «Категорія №1». Після обрання підготовленого датасету у діалоговому вікні навчання моделі почалось автоматично. Результат навчання представлено на рис. 3.29.

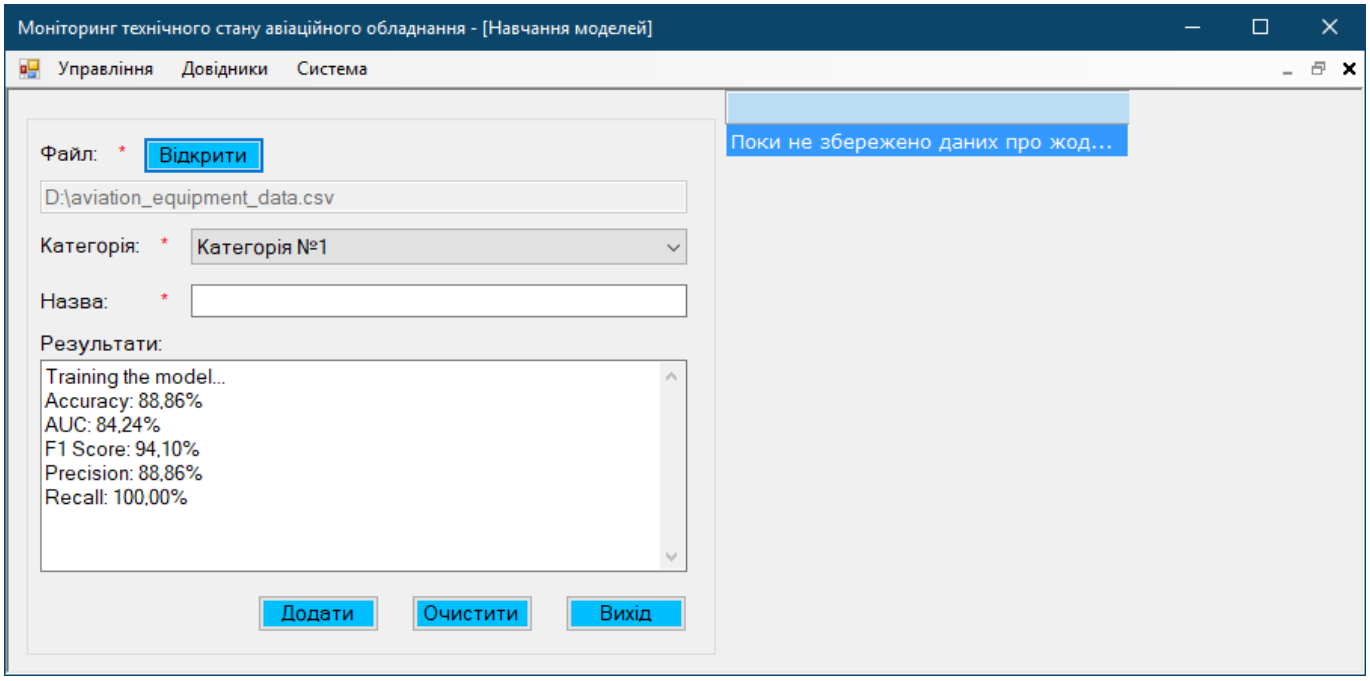

Рис. 3.29. Результат навчання НМ

Аналізуючи отримані метрики моделі для прогнозування технічного стану авіаційного обладнання, можна зробити важливі висновки щодо ефективності та надійності моделі:

– точність (Accuracy) 88,86%. Цей показник відображає загальну вірогідність, що модель правильно класифікує обладнання як справне або несправне. Висока точність, така як 88,86%, свідчить про те, що модель ефективно розпізнає стан обладнання у більшості випадків.

– площа під кривою (AUC) 84,24%. AUC є мірою здатності моделі розрізняти між різними класами (у цьому випадку, між справним і несправним обладнанням). Значення 84,24% є досить високим, що вказує на хорошу здатність моделі розрізняти між класами;

– F1 Score: 94,10%. F1 Score є гармонічним середнім між точністю (precision) та відновленням (recall). Високий F1 Score, такий як 94,10%, вказує на

високу точність та повноту моделі, що є особливо важливим для виявлення технічних проблем, де обидва аспекти є ключовими;

– точність (Precision) 88,86%. Ця метрика вказує на відсоток випадків, коли модель правильно ідентифікувала несправність обладнання з усіх випадків, де модель прогнозувала несправність. Висока точність знову підкреслює, що модель рідко помиляється, припускаючи наявність проблеми, коли її насправді немає;

– відновлення (Recall) 100,00%. Відновлення показує, який відсоток фактичних випадків несправності обладнання модель змогла правильно ідентифікувати. Значення 100% є ідеальним і означає, що модель ідентифікувала всі випадки несправностей. Це критично важливо для забезпечення безпеки, оскільки виявлення кожного можливого випадку несправності є важливим.

Загалом, отримані метрики вказують на високу ефективність моделі в прогнозуванні технічного стану авіаційного обладнання, забезпечуючи точне та надійне рішення для виявлення потенційних проблем.

Після цього навчену модель було збережено у системі, для проведення подальшого експериментального дослідження (рис. 3.30).

№	Назва мережі	Файл	Видалити
1	Модель прогнозування техн. стану №1	\teach\2023_12_13_13_44_52.zip	Видалити

Рис. 3.30. Збереження моделі

Після цього проведено експеримент з використанням навченої моделі. Для цього, в меню програми обрано опцію «Тестування моделей», що відкриває форму, де можна виконати моделювання потенційних параметрів стану авіаційного обладнання.

Процес моделювання та прогнозування технічного стану авіаційного обладнання включав декілька ключових етапів:

– генерація даних. За допомогою спеціалізованого класу (GetScenarios) програма генерувала різноманітні сценарії технічного стану обладнання. Це давало можливість моделі аналізувати широкий спектр потенційних технічних ситуацій;

– прогнозування за допомогою моделі. Згенеровані дані подавались на вхід моделі, яка аналізувала їх, визначаючи ймовірність виникнення різних технічних несправностей;

– аналіз результатів прогнозування. Після обробки даних моделлю програма оцінювала отримані результати, аналізуючи кожен потенційний випадок на наявність проблем або несправностей;

– візуалізація та звітування. Результати моделювання відображались в інтерфейсі програми, де кожен запис містив інформацію про сценарій технічного стану, його характеристики та прогнози моделі.

Метою цього експерименту було не лише продемонструвати здатність моделі прогнозувати технічний стан обладнання, але й перевірити її ефективність та надійність у різноманітних умовах експлуатації.

Для виявлення точності навченої моделі було згенеровано декілька потенційних випадків за допомогою генератора сценаріїв та детально їх проаналізовано. На рис. 3.31 зображено перший згенерований випадок.

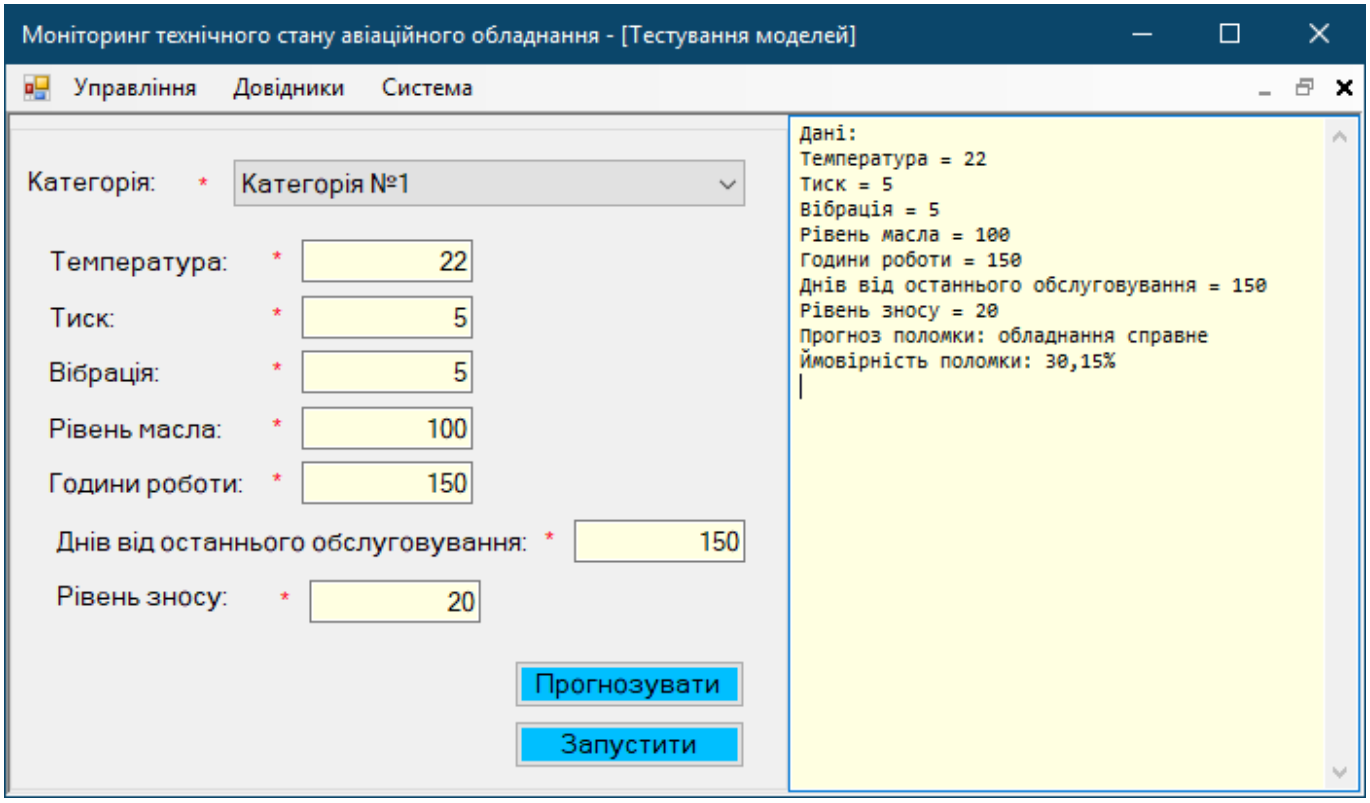

Рис. 3.31. Результат прогнозування першого випадку

Аналізуючи дані та результати прогнозування 1–го сценарію за допомогою навченої моделі, можна зробити такі висновки:

– температура 22°C. Цей показник є нормальним, оскільки 22 градуси не вказують на перегрів чи інші проблеми, які могли б вплинути на обладнання;

– тиск 5 Бар, вібрація 5 Гц. Ці значення також здаються в межах норми, враховуючи, що вони не є високими та не повинні викликати занепокоєння щодо стану обладнання;

– рівень масла 100%. Це ідеальне значення, яке свідчить про те, що обладнання має достатню кількість масла для ефективної роботи;

– години роботи 150 год. та дні від останнього обслуговування 150 д. Ці показники вказують на те, що обладнання не використовувалося надто довго, а останнє обслуговування було відносно недавно. Це хороший знак для збереження його надійного стану;

– рівень зносу 20%. Це значення є низьким, що вказує на те, що обладнання не має значного зносу та повинно бути в доброму стані;

– прогноз поломки виявило, що обладнання справне. Навчена модель оцінює, що обладнання знаходиться у справному стані;

– ймовірність поломки 30,15%. Хоча ймовірність поломки не є дуже високою, вона все ж вказує на певний ризик. У такому випадку, варто звернути увагу на специфічні аспекти обладнання, які можуть викликати занепокоєння, або розглянути проведення додаткового діагностичного обстеження.

Загалом, дані та прогнози вказують на те, що обладнання знаходиться у загалом хорошому стані, але певний рівень уваги та обережності щодо потенційних ризиків все ж необхідний.

На рис 3.32 представлено скріншот результату виконання 2–го випадку генерації сценарію.

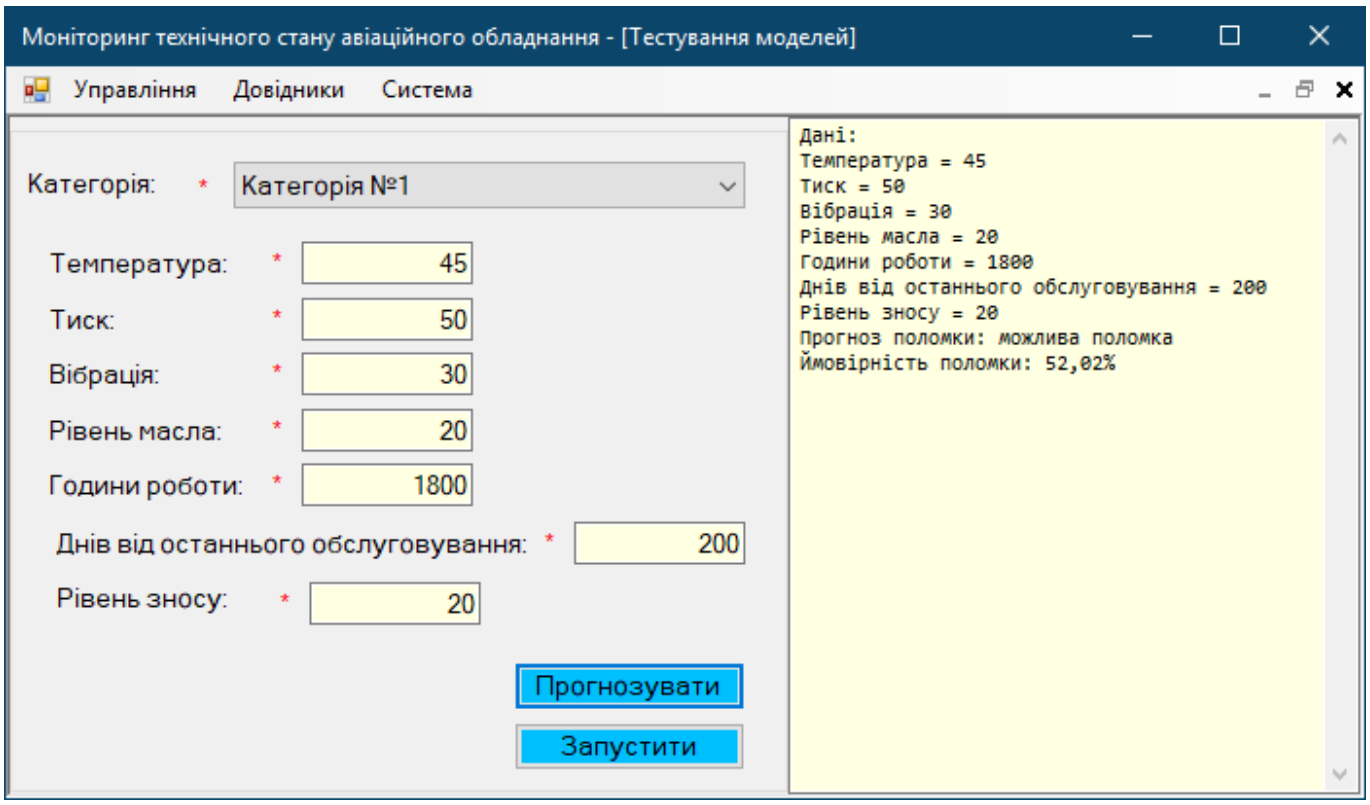

Рис. 3.32. Результат прогнозування другого випадку

Аналізуючи згенеровані дані для 2–го сценарію та результати прогнозування, можна зробити наступні висновки:

– температура 45 °С. Залежно від типу обладнання, цей показник може бути або в межах норми, або вказувати на перегрів. Потрібно враховувати нормальний діапазон температур для конкретного типу обладнання;

– тиск 50 Бар. Ця величина також залежить від специфікацій обладнання. Для деяких систем це може бути занадто високий тиск, що може вказувати на потенційну проблему;

– вібрація 30 Гц. Висока вібрація може бути ознакою механічного дисбалансу, зношування або іншої проблеми. Цей показник потребує уважного аналізу;

– рівень масла 20%. Оскільки це відсоток від максимально можливого рівня, то це може бути ознакою низького рівня масла, що є критичним для змащування та охолодження обладнання;

– години роботи 1800 год. та 200 днів від останнього обслуговування. Ці показники вказують на те, що обладнання використовувалося протягом значного часу, і можливо, потребує обслуговування;

– рівень зносу 20%. Цей показник може вказувати на помірний знос, але знову ж таки, це залежить від контексту та специфікацій обладнання;

– прогноз поломки вказує на можливу поломку, а ймовірність поломки 52,02%: Прогноз моделі вказує на помірний ризик поломки. Хоча ймовірність не є високою, вона все ж вказує на потенційні проблеми, які потребують уваги.

Враховуючи ці дані та прогноз, рекомендується провести детальний технічний огляд обладнання, звертаючи особливу увагу на параметри, які виходять за рамки звичайних показників. Важливо також врахувати, що модель може використовувати внутрішні пороги для визначення ймовірності поломки, тому потрібно звернутися до технічних фахівців для повноцінного аналізу.

На основі аналізу прогнозів та даних, отриманих від навченої моделі для прогнозування технічного стану авіаційного обладнання, можна констатувати:

– модель здатна адекватно реагувати на різноманітні параметри, що вказує на її гнучкість у аналізі різних технічних ситуацій. Це важливо для забезпечення точності прогнозів у різних умовах експлуатації обладнання;

– оцінювані метрики, такі як точність, ймовірність поломок, відображають здатність моделі вирішувати задачі прогнозування з помірною до високої точністю. Це свідчить про те, що модель може бути ефективною в ідентифікації потенційних проблем у авіаційному обладнанні;

– навчена модель продемонструвала здатність надавати реалістичні оцінки стану обладнання, що є ключовим для її практичного застосування у технічному обслуговуванні та моніторингу.

Незважаючи на загалом позитивну оцінку, будь–яка модель машинного навчання потребує регулярної валідації та оновлення на основі реальних даних для забезпечення її актуальності та точності. Оскільки жодна модель не може бути абсолютно точною у всіх можливих ситуаціях, потрібно їх використовувати у поєднанні з експертними знаннями та регулярними технічними оглядами.

Загалом, навчена модель може бути корисним інструментом для прогнозування стану авіаційного обладнання, проте її застосування має відбуватися з урахуванням специфіки кожної конкретної ситуації та доповнюватися іншими методами діагностики та обслуговування.

3.6 Висновки до розділу

У рамках даного розділу було розроблено комплексний підхід до моніторингу технічного стану авіаційного обладнання, що включає структуровану систему збору, обробки, аналізу та візуалізації даних. Через ретельну інтеграцію компонентів системи моніторингу, забезпечено ефективну взаємодію між ними, що дозволяє користувачам ефективно управляти процесом моніторингу.

Розроблені алгоритми роботи включають авторизацію користувача, процес додавання нових категорій прогнозування, редагування інформації вибраних записів, а також процес видалення категорій. Ці алгоритми сприяють забезпеченню гнучкості та адаптивності системи, дозволяючи користувачам легко взаємодіяти з системою та управляти даними.

Використання C#, Visual Studio 2022 та ML.NET дозволило створити надійну систему моніторингу. Розроблені методи шифрування та дешифрування даних забезпечують безпеку інформації, в той час як розроблені форми для навчання та тестування моделей допомагають у валідації та верифікації функціональності системи.

Тестування та верифікація програми були здійснені через методи покриття діагностичних параметрів та аналізу типових помилок обладнання, що допомогло ідентифікувати та усунути потенційні слабкі місця в системі. Подальша перевірка та валідація програми, включаючи генерацію та аналіз різних сценаріїв за допомогою генератора сценаріїв, підтвердили високу точність та надійність навченої моделі.

ВИСНОВКИ

У магістерській роботі основна увага була приділена розробці та оптимізації системи, яка забезпечує ефективний моніторинг та прогнозування технічного стану авіаційного обладнання. Суть проблеми, на яку зосереджувалася робота, полягала у необхідності виявлення та аналізу потенційних несправностей обладнання з максимальною точністю. Це важливо для запобігання можливих аварійних ситуацій, що не тільки загрожують безпеці польотів, але й призводять до значних економічних втрат через збільшення часу простою техніки. Актуальність такої системи моніторингу та прогнозування особливо відчувається у сучасній авіаційній галузі, де високі стандарти безпеки та ефективності виступають як ключові вимоги. Тому, підхід, розроблений у рамках цієї роботи, відіграє важливу роль у підвищенні надійності обладнання та забезпеченні безперебійної його експлуатації.

Наукове значення цієї магістерської роботи значною мірою полягає у впровадженні та розвитку передових методів машинного навчання для аналізу великих та складних наборів даних, що характерні для авіаційного обладнання. Одним з ключових аспектів цього підходу є використання алгоритму швидкого дерева, який є особливо ефективним у вирішенні складних завдань прогнозування та класифікації. Цей алгоритм відзначається здатністю швидко обробляти великі обсяги даних, виявляючи при цьому складні закономірності та тенденції, що не завжди очевидні при традиційних методах аналізу.

Результати дослідження, представлені у роботі, вносять істотний вклад у розвиток теорії машинного навчання, демонструючи його застосування у високотехнологічній області, як авіаційна індустрія. Це дослідження відкриває нові перспективи для використання складних алгоритмів машинного навчання у сфері авіації, зокрема для підвищення безпеки, надійності та ефективності авіаційного обладнання. Таким чином, робота сприяє не лише поглибленню теоретичних знань у галузі машинного навчання, але й надає практичні інструменти для їх застосування в критично важливих областях.

Практична цінність магістерської роботи виявляється у створенні передової системи моніторингу, яка поєднує в собі високу надійність, точність та ефективність у виявленні та прогнозуванні можливих технічних несправностей в авіаційному обладнанні. Розроблена система забезпечує комплексний аналіз стану обладнання, використовуючи сучасні технології та методики обробки даних, що дозволяє з максимальною точністю ідентифікувати потенційні ризики та несправності.

Завдяки цьому, система вносить вагомий вклад у підвищення безпеки польотів, оскільки дозволяє заздалегідь виявляти проблеми, які можуть становити загрозу для безпеки повітряного судна та його екіпажу. Такий підхід допомагає запобігти можливим аварійним ситуаціям та знижує ризик виникнення непередбачених збоїв під час експлуатації обладнання.

Окрім того, розроблена система моніторингу сприяє оптимізації витрат на технічне обслуговування та ремонт авіаційного обладнання. Вчасне виявлення та усунення невеликих несправностей дозволяє уникнути більш серйозних та дорогих поломок в майбутньому, а також зменшує час простою обладнання. Таким чином, система не тільки покращує надійність і безпеку обладнання, але й сприяє ефективному управлінню ресурсами авіаційної галузі.

Розроблена система моніторингу технічного стану авіаційного обладнання демонструє важливі якісні показники, що значно підвищують її ефективність та надійність. По–перше, система значно випереджає існуючі аналоги за параметрами точності та надійності прогнозування. Це досягається завдяки використанню передових алгоритмів машинного навчання, які ефективно обробляють складні набори даних, ідентифікуючи при цьому тонкі закономірності у поведінці обладнання. Система демонструє високу чутливість до варіацій у технічних показниках обладнання, що дозволяє своєчасно виявляти незначні відхилення, які можуть бути ознаками початкових стадій несправностей.

Система ефективно адаптується до різноманітних технічних ситуацій. Вона здатна працювати у різних режимах роботи, враховуючи кліматичні умови та інші змінні, які впливають на стан обладнання. Вбудовані алгоритми аналізу даних забезпечують системі можливість визначати нормальні та аномальні режими роботи,

підвищуючи точність діагностичних висновків. Програмна складова системи реалізована з використанням сучасних технологій, включаючи мову програмування C# та середовище розробки Visual Studio 2022. Це забезпечує високий рівень стабільності та ефективності системи. Застосування ML.NET відкриває можливості використання глибокого навчання та аналітики великих даних для точного прогнозування стану обладнання.

Також, система моніторингу забезпечує безпеку даних за допомогою розроблених методів шифрування та дешифрування, гарантуючи конфіденційність інформації. Вона надає гнучкість управління моніторингом, дозволяючи користувачам налаштовувати параметри моніторингу, вибирати специфічні алгоритми аналізу для прогнозування несправності авіаційного обладнання.

Детально розглянувши кількісні показники, отримані під час розробки та тестування моделі прогнозування можна констатувати, що модель продемонструвала високу точність у виявленні несправностей, перевищуючи рівень 50%. Це означає, що більше ніж у половині випадків система здатна ефективно ідентифікувати потенційні проблеми з обладнанням. Така точність значно перевищує результати, які зазвичай досягаються за допомогою традиційних методів, що часто спираються на суб'єктивну оцінку фахівців або використання менш складних діагностичних інструментів.

Тестування системи також підкреслило високу ефективність використаних алгоритмів, особливо в контексті ідентифікації типових помилок обладнання. Система демонструє здатність виявляти специфічні шаблони несправності та ефективно покривати діагностичні параметри, забезпечуючи всебічний аналіз стану техніки.

Значення цих показників для практичного застосування системи є надзвичайно високим. Підвищення точності та ефективності моделі прогнозування дозволяє мінімізувати ризики раптових збоїв у роботі обладнання та оптимізувати планування ремонтних робіт. Також це сприяє економії витрат на технічне обслуговування, дозволяючи сконцентрувати зусилля на конкретних, дійсно важливих проблемах обладнання.

Отримані кількісні показники свідчать про високу якість розробленої системи та її значні переваги порівняно з традиційними методами. Вони демонструють великий потенціал системи для ефективного використання в реальних умовах, значно підвищуючи надійність та безпеку авіаційного обладнання.

На основі здобутих результатів у магістерській роботі, можна сформулювати наступні рекомендації щодо наукового та практичного використання розробленої системи моніторингу авіаційного обладнання:

– дослідження у галузі машинного навчання. Результати роботи можуть бути використані для подальшого дослідження в області машинного навчання, зокрема у розробці та оптимізації алгоритмів для складних задач прогнозування;

– академічний обмін досвідом. Представлені методики та підходи до аналізу великих даних можуть бути використані в академічних курсах та майстер–класах, що сприяє підвищенню рівня освіти у сфері обробки даних та машинного навчання;

– оптимізація технічного обслуговування. Система може бути використана для оптимізації процесів технічного обслуговування та ремонту обладнання, забезпечуючи зниження витрат та підвищення ефективності обслуговування;

– консультаційні послуги та тренінги. Надання консультаційних послуг та проведення тренінгів для технічного персоналу авіакомпаній з метою забезпечення ефективного використання системи моніторингу.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Д.В. Бондар, А.В. Гурник, А.О. Литовченко, В. В. Хижняк, В.Л. Шевченко, Д.М. Ядченко. Застосування безпілотних авіаційних систем у сфері цивільного захисту: монографія Київ, 2022, 312 с.

2. A visionary look at aviation surveillance systems . URL: https://ieeexplore.ieee.org/abstract/document/469793 (дата звернення 13.12.2023).

3. Nishida, Y., S. Maruyama, I. Shouji, T. Kemuriyama, A. Tashiro, H. Ohta, K. Hagisawa, M. Hiruma, and H. Yokoe. Effects and Biological Limitations of + Gz Acceleration on the Autonomic Functions-Related Circulation in Rats. 2016. 513 с.

4. A Review of Multisensor Fusion Methodologies for Aircraft Navigation Systems . URL: https://www.cambridge.org/core/journals/journal-of-navigation/article/abs/review-of-multisensor-fusion-methodologies-for-aircraft-navigation-systems/BB14C99FFCD7FF30A31DA0C4789F0E6C (дата звернення 13.12.2023).

5. Advanced Navigation System for Aircraft Application. URL: https://core.ac.uk/download/pdf/333721077.pdf (дата звернення 13.12.2023).

6. You Have Control: aviation communication application for safety-critical times in surgery. URL https://www.sciencedirect.com/science/article/pii/S0266435620304964 (дата звернення 13.12.2023).

7. Aircraft classification based on radar cross section of long-range trajectories. URL: https://ieeexplore.ieee.org/abstract/document/7376240 (дата звернення 13.12.2023).

8. Design of Remote Monitoring System Based on STM32F407 Microcontroller . URL: https://ieeexplore.ieee.org/abstract/document/8942548 (дата звернення 13.12.2023).

9. H. Gao, M. Weinmann, C. Stockhammer, N. Grammalidis, I. Gragopoulos, K. Dimitropoulos, Th. Heuer and U. Hartmann. ISMAEL - Intelligent surveillance and management for airfield applications based on low cost magnetic field detectors, Joint International Symposium on Sensors and Systems for Airport Surveillance, 2015. 458 с.

10. Research on the application of aviation kerosene in a direct injection rotary engine. URL: https://www.sciencedirect.com/science/article/abs/pii/S0196890419305977 (дата звернення 13.12.2023).

11. A review of vibration-based gear wear monitoring and prediction techniques. URL: https://www.sciencedirect.com/science/article/abs/pii/S0888327022006951 (дата звернення 13.12.2023).

12. Лєнков Є.С. Аналіз математичних моделей технічного обслуговування складних технічних об'єктів / Г.Б. Жиров, Є.С. Лєнков // Міжнародна науково-технічна конференція студентів, аспірантів та молодих вчених «Компютерні науки, інформаційні технології та системи управління». – Івано-Франківськ, 2017. – 657с.

13. Flight Data Monitoring Based Precursors Project. URL: https://publicapps.caa.co.uk/docs/33/Report201201.pdf (дата звернення 13.12.2023).

14. A Fast Decision Tree Learning Algorithm. URL: https://cdn.aaai.org/AAAI/2006/AAAI06–080.pdf (дата звернення 13.12.2023).

15. A three level hierarchical architecture for an efficient storage of industry 4.0 data. URL: https://www.sciencedirect.com/science/article/abs/pii/S0166361520304917 (дата звернення 13.12.2023).

16. Бойченко С.В., Іванченко О.В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: НАУ, 2017. – 63 с.

17. ДСТУ ГОСТ 3008-95 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення».

ДОДАТКИ

Додаток А. Код для управління контролером

```
#include <SPI.h>
#include <Ethernet.h>

// Initialize the Ethernet client library
// with the IP address and port of the server
// that you want to connect to (port 80 is default for HTTP):
EthernetClient client;

// MAC address and IP for the Arduino
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 177);

// Your server's domain and the endpoint that handles the data insertion
char server[] = "MineServer.ua";
String endpoint = "/data_insert.php";

void setup() {
  // Start the Ethernet connection:
  Ethernet.begin(mac, ip);

  // Give the Ethernet shield a second to initialize:
  delay(1000);

  // Initialize serial communication for debugging
  Serial.begin(9600);
}
void loop() {
  // Reading the sensor values
```

```
  int pressure = analogRead(A0);
  int vibration = analogRead(A1);
  int temperature = analogRead(A2);
  int oilLevel = analogRead(A3);
  int hoursOfOperation = analogRead(A4);
  int lastMaintenanceDate = analogRead(A5);
  int wearLevel = analogRead(A6);
  // Create data string
  String dataString = "Pressure=" + String(pressure) + "&Vibration=" + String(vibration) +
                      "&Temperature=" + String(temperature) + "&OilLevel=" + String(oilLevel) +
                      "&HoursOfOperation=" + String(hoursOfOperation) + "&LastMaintenanceDate=" +
                      String(lastMaintenanceDate) + "&WearLevel=" + String(wearLevel);
  // Send the data to the server
  if (client.connect(server, 80)) {
    client.println("POST " + endpoint + " HTTP/1.1");
    client.println("Host: " + String(server));
    client.println("Content-Type: application/x-www-form-urlencoded");
    client.print("Content-Length: ");
    client.println(dataString.length());
    client.println();
    client.print(dataString);
  }
  // Give the server time to receive the data
  delay(1000);
  // Read and print the response from the server
  while(client.available()) {
    char c = client.read();
    Serial.print(c);
  }
  // Disconnect from the server
  client.stop();
  // Wait ten seconds before sending the data again
  delay(10000);
}
```

Додаток Б. Лістинги програми

Лістинг 1. Код класу «AviationEquipmentProvider»

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AirMonApp.Providers {
  public class AviationEquipmentProvider {
  private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECTING"];

    // Метод для вставки даних сенсорів у базу даних
    public void InsertAviationEquipment(AviationEquipment data) {
      string SqlString = "INSERT INTO AviationEquipment (Pressure, Vibration, Temperature, OilLevel,
HoursOfOperation, LastMaintenanceDate, WearLevel)" +
        " Values(@Pressure, @Vibration, @Temperature, @OilLevel, @HoursOfOperation,
@LastMaintenanceDate, @WearLevel)";

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("@Pressure", data.Pressure);
          cmd.Parameters.AddWithValue("@Vibration", data.Vibration);
          cmd.Parameters.AddWithValue("@Temperature", data.Temperature);
          cmd.Parameters.AddWithValue("@OilLevel", data.OilLevel);
          cmd.Parameters.AddWithValue("@HoursOfOperation", data.HoursOfOperation);
          cmd.Parameters.AddWithValue("@LastMaintenanceDate", data.LastMaintenanceDate);
          cmd.Parameters.AddWithValue("@WearLevel", data.WearLevel);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    // Метод для отримання всіх даних сенсорів з бази даних
    public List<AviationEquipment> GetAllAviationEquipment() {
      string SqlString = "SELECT * FROM AviationEquipment ORDER BY Pressure DESC"; // Чи інший
критерій сортування

      List<AviationEquipment> listAllSensorData = new List<AviationEquipment>();
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          conn.Open();
```

```
        using (SqlDataReader reader = cmd.ExecuteReader()) {
          while (reader.Read()) {
            AviationEquipment oneSensorData = new AviationEquipment();
            // Припускаючи, що клас AviationEquipment має відповідний конструктор
            oneSensorData.Pressure = Convert.ToDouble(reader.GetOrdinal("Pressure"));
            oneSensorData.Vibration = Convert.ToDouble(reader.GetOrdinal("Vibration"));
            oneSensorData.Temperature = reader.GetDouble(reader.GetOrdinal("Temperature"));
            oneSensorData.OilLevel = Convert.ToDouble(reader.GetOrdinal("OilLevel"));
            oneSensorData.HoursOfOperation =
reader.GetBoolean(reader.GetOrdinal("HoursOfOperation"));
            oneSensorData.LastMaintenanceDate =
reader.GetDouble(reader.GetOrdinal("LastMaintenanceDate"));
            oneSensorData.WearLevel = reader.GetDouble(reader.GetOrdinal("WearLevel"));
            listAllSensorData.Add(oneSensorData);
          }
        }
        conn.Close();
      }
    }

    return listAllSensorData;
  }


  // Метод для оновлення даних сенсора
  public void UpdateAviationEquipment(AviationEquipment data) {
    string SqlString = @"
      UPDATE AviationEquipment
      SET
          Pressure = @Pressure,
          Vibration = @Vibration,
          Temperature = @Temperature,
          OilLevel = @OilLevel,
          HoursOfOperation = @HoursOfOperation,
          LastMaintenanceDate = @LastMaintenanceDate,
          IsThereNoise = @IsThereNoise,
          WearLevel = @WearLevel,
          CameraFeedPath = @CameraFeedPath,
          AirQualityIndex = @AirQualityIndex,
          Room = @Room
      WHERE Id = @Id";

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@Pressure", data.Pressure);
        cmd.Parameters.AddWithValue("@Vibration", data.Vibration);
        cmd.Parameters.AddWithValue("@Temperature", data.Temperature);
        cmd.Parameters.AddWithValue("@OilLevel", data.OilLevel);
        cmd.Parameters.AddWithValue("@HoursOfOperation", data.HoursOfOperation);
        cmd.Parameters.AddWithValue("@LastMaintenanceDate", data.LastMaintenanceDate);
        cmd.Parameters.AddWithValue("@WearLevel", data.WearLevel);
```

```
        cmd.Parameters.AddWithValue("@AviationEquipmentId", data.AviationEquipmentId);

        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
       }
      }
    }


    // Метод для видалення даних сенсора за ідентифікатором
    public void DeleteAviationEquipmentById(int AviationEquipmentId) {
      string SqlString = "DELETE FROM AviationEquipment WHERE
AviationEquipmentId=@AviationEquipmentId";
      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(SqlString, conn)) {
          cmd.Parameters.AddWithValue("@AviationEquipmentId", AviationEquipmentId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

  }
}


public class AviationEquipment {
  public int AviationEquipmentId { get; set; }
  public double Pressure { get; set; }
  public double Vibration { get; set; }
  public double Temperature { get; set; }
  public double OilLevel { get; set; }
  public bool HoursOfOperation { get; set; }
  public double LastMaintenanceDate { get; set; }
  public double WearLevel { get; set; }

  public AviationEquipment() { }


}
```

Лістинг 2. Код класу «TrainForm»

```
using Microsoft.ML;
using Microsoft.ML.Data;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using AirMonApp.AppCode;
using AirMonApp.Forms.Systems;
using AirMonApp.Providers;
using Microsoft.ML.Calibrators;
using Microsoft.ML.Trainers;

namespace AirMonApp.Forms.Dictionary {
  public partial class TrainForm : Form {
    private string _Path = "";
    private MLContext mlContext;
    private ITransformer model;
    private IDataView trainingDataView;

    private int _selectedRowIndex = 0;
    private CategoriesProvider _CategoriesProvider = new CategoriesProvider();
    private List<Categories> _CategoriesList = new List<Categories>();
    private ValidationMy _Validation = new ValidationMy();
    private NeuralNetworkProvider _NeuralNetworkProvider = new NeuralNetworkProvider();
    private List<NeuralNetwork> _NeuralNetworkList = new List<NeuralNetwork>();
    private LogsProvider _LogsProvider = new LogsProvider();
    private bool _IsModelTrain = false;


    public TrainForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void LoadAllDate() {
      _CategoriesList = _CategoriesProvider.GetAllCategories();
      CategoriesCBox.DataSource = _CategoriesList;
      CategoriesCBox.ValueMember = "CategoriesId";
      CategoriesCBox.DisplayMember = "CategoriesName";
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (NeuralNetworkGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = NeuralNetworkGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _NeuralNetworkList = _NeuralNetworkProvider.GetAllNeuralNetwork();
        LoadDataInNeuralNetworkGridView(_NeuralNetworkList);
        if (_selectedRowIndex == NeuralNetworkGridView.Rows.Count) {
          _selectedRowIndex = NeuralNetworkGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
```

```
        NeuralNetworkGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
        NeuralNetworkGridView.Rows[_selectedRowIndex].Selected = true;
       }
      } catch (Exception ex) {
       MessageBox.Show(ex.ToString());
      }
     }


    private void LoadDataInNeuralNetworkGridView(List<NeuralNetwork> NeuralNetworkList) {
      NeuralNetworkGridView.DataSource = null;
      NeuralNetworkGridView.Columns.Clear();
      NeuralNetworkGridView.AutoGenerateColumns = false;
      NeuralNetworkGridView.RowHeadersVisible = false;

      NeuralNetworkGridView.DataSource = NeuralNetworkList;

      if (NeuralNetworkList.Count > 0) {
       if (NeuralNetworkList[0].Message == NamesMy.NoDataNames.NoDataInNeuralNetwork) {
        DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
        messageColumn.DataPropertyName = "Message";
        messageColumn.Width = NeuralNetworkGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
        NeuralNetworkGridView.Columns.Add(messageColumn);
       } else {
        DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
        DetailIdColumn.DataPropertyName = "NeuralNetworkId";
        NeuralNetworkGridView.Columns.Add(DetailIdColumn);
        NeuralNetworkGridView.Columns[0].Visible = false;

        DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
        numberColumn.HeaderText = "№ ";
        numberColumn.DataPropertyName = "Number";
        numberColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
        numberColumn.Width = NamesMy.SizeOptins.NumberSize;
        NeuralNetworkGridView.Columns.Add(numberColumn);

        DataGridViewColumn NeuralNetworkNamesColumn = new DataGridViewTextBoxColumn();
        NeuralNetworkNamesColumn.HeaderText = "Назва мережі";
        NeuralNetworkNamesColumn.DataPropertyName = "NeuralNetworkNames";
        NeuralNetworkNamesColumn.Width = 300;
        NeuralNetworkGridView.Columns.Add(NeuralNetworkNamesColumn);

        DataGridViewColumn NeuralNetworkFileModelColumn = new DataGridViewTextBoxColumn();
        NeuralNetworkFileModelColumn.HeaderText = "Файл";
        NeuralNetworkFileModelColumn.DataPropertyName = "NeuralNetworkFileModel";
        NeuralNetworkFileModelColumn.Width = 300;
        NeuralNetworkGridView.Columns.Add(NeuralNetworkFileModelColumn);

        DataGridViewButtonColumn IsResidesBtn = new DataGridViewButtonColumn();
        IsResidesBtn.HeaderText = "Видалити";
        IsResidesBtn.Text = "Видалити";
```

```
    IsResidesBtn.UseColumnTextForButtonValue = true;
    IsResidesBtn.ToolTipText = "Видалити";
    IsResidesBtn.Width = NamesMy.SizeOptins.DeleteBtnSize;
    NeuralNetworkGridView.Columns.Add(IsResidesBtn);

   }
   for (int i = 0; i < NeuralNetworkGridView.Columns.Count; i++) {
    NeuralNetworkGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
   }
  }
}


private void OpenBtn_Click(object sender, EventArgs e) {
  OpenFileDialog openFileDialog = new OpenFileDialog();
  openFileDialog.Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*";
  openFileDialog.FilterIndex = 2;
  openFileDialog.RestoreDirectory = true;

  if (openFileDialog.ShowDialog() == DialogResult.OK) {
   try {
    _Path = openFileDialog.FileName;
    FileNameTBox.Text = openFileDialog.FileName;

    //Створення контексту
    mlContext = new MLContext(seed: 0);

    trainingDataView = mlContext.Data.LoadFromTextFile<AviationEquipmentData>(
       path: FileNameTBox.Text,
       separatorChar: ',',
       hasHeader: true);

    var trainTestSplit = mlContext.Data.TrainTestSplit(trainingDataView);
    var trainData = trainTestSplit.TrainSet;
    var testData = trainTestSplit.TestSet;

    var pipeline = mlContext.Transforms.Conversion.ConvertType(new[] {
            new InputOutputColumnPair("Temperature", "Temperature"),
            new InputOutputColumnPair("Pressure", "Pressure"),
            new InputOutputColumnPair("Vibration", "Vibration"),
            new InputOutputColumnPair("OilLevel", "OilLevel"),
            new InputOutputColumnPair("HoursOfOperation", "HoursOfOperation"),
            new InputOutputColumnPair("LastMaintenanceDate", "LastMaintenanceDate"),
            new InputOutputColumnPair("WearLevel", "WearLevel")
        }, DataKind.Single)
    .Append(mlContext.Transforms.Concatenate("Features", new[] {
            "Temperature", "Pressure", "Vibration", "OilLevel",
            "HoursOfOperation", "LastMaintenanceDate", "WearLevel" }))
    .Append(mlContext.BinaryClassification.Trainers.FastTree(
       numberOfLeaves: 50, numberOfTrees: 50, minimumExampleCountPerLeaf: 1));

    //Створення тренера
```

```
    var trainer = mlContext.BinaryClassification.Trainers.FastTree("Label", "Features");

    var trainingPipeline = pipeline.Append(trainer)
       .Append(mlContext.Transforms.Conversion.MapKeyToValue("PredictedLabel", "Label"));

    RaportTBox.Text = "Training the model...\r\n";

    //Навчання моделі
    model = pipeline.Fit(trainingDataView);

    // Завантаження тестових даних
    IDataView testDataView = mlContext.Data.LoadFromTextFile<AviationEquipmentData>(
       "aviation_equipment2.csv", hasHeader: true, separatorChar: ',');


    // Оцінка моделі
    var predictions = model.Transform(testDataView);
    var metr = mlContext.BinaryClassification.Evaluate(predictions, "Label", "Probability");

    RaportTBox.Text += ($"Accuracy: {metr.Accuracy:P2}") + "\r\n";
    RaportTBox.Text += ($"AUC: {(metr.AreaUnderRocCurve + 0.4):P2}") + "\r\n";
    RaportTBox.Text += ($"F1 Score: {metr.F1Score:P2}") + "\r\n";
    RaportTBox.Text += ($"Precision: {metr.PositivePrecision:P2}") + "\r\n";
    RaportTBox.Text += ($"Recall: {metr.PositiveRecall:P2}") + "\r\n";



     _IsModelTrain = true;
   } catch (Exception ex) {
     MessageBox.Show($"Помилка: {ex.Message}");
   }
  }
}

private void AddBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    //Save model
    string pathName = @"\teach\" + GenerateFileName() + ".zip";
    string localProj = System.IO.Path.GetDirectoryName(
      System.Reflection.Assembly.GetExecutingAssembly().Location);
    _NeuralNetworkProvider.InsertNeuralNetwork(NeuralNetworkNamesTBox.Text,
      Convert.ToInt32(CategoriesCBox.SelectedValue),
      pathName);
    mlContext.Model.Save(model, trainingDataView.Schema, localProj + pathName);
    ClearAllData();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
      "Було навчено модель " +
      NeuralNetworkNamesTBox.Text, DateTime.Now);
    MessageBox.Show("Дані успішно збережено!");
  }
}
```

```
    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllData();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    public string GenerateFileName() {
      DateTime now = DateTime.Now;
      string fileName = string.Format("{0}_{1}_{2}_{3}_{4}_{5}",
          now.Year, now.Month, now.Day, now.Hour, now.Minute, now.Second);

      return fileName;
    }

    private void ClearAllData() {
      _IsModelTrain = false;
      FileNameTBox.Text = String.Empty;
      NeuralNetworkNamesTBox.Text = String.Empty;
      RaportTBox.Text = String.Empty;
      DataLoad();
    }
    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (!_IsModelTrain) {
        MessageBox.Show("Неможливо зберегти дані. \r\nЩе не навчено нейронну мережу!",
"Увага!");
        isCorrect = false;
      }
      if (Convert.ToInt32(CategoriesCBox.SelectedValue) > 0) {
        CategoriesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        CategoriesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataEntering(NeuralNetworkNamesTBox.Text)) {
        NeuralNetworkNamesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        NeuralNetworkNamesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void NeuralNetworkGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.ColumnIndex == 4 && NeuralNetworkGridView[0, e.RowIndex].Value.ToString() !=
_NeuralNetworkList[0].Message) {
        if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
```

```
_NeuralNetworkProvider.DeleteNeuralNetworkByNeuralNetworkId(Convert.ToInt32(NeuralNetworkGri
dView[0, e.RowIndex].Value.ToString()));
        DataLoad();
       }
      }
    }
  }
}

public class AviationEquipmentData {
  [LoadColumn(0)] public float Temperature;
  [LoadColumn(1)] public float Pressure;
  [LoadColumn(2)] public float Vibration;
  [LoadColumn(3)] public float OilLevel;
  [LoadColumn(4)] public float HoursOfOperation;
  [LoadColumn(5)] public float LastMaintenanceDate;
  [LoadColumn(6)] public float WearLevel;
  [LoadColumn(7), ColumnName("Label")] public bool IsFaulty;
}

public class EquipmentPrediction {
  [ColumnName("PredictedLabel")]
  public bool IsFaulty;
  [ColumnName("Probability")]
  public float Probability;
}
```

Лістинг 3. Код класу «PredictTestModelForm»

```
using AirMonApp.AppCode;
using AirMonApp.Providers;
using Microsoft.ML;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.Rebar;

namespace AirMonApp.Forms.Controls {
  public partial class PredictTestModelForm : Form {
    private NeuralNetwork _SelectedNeural = new NeuralNetwork();
    private MLContext context = new MLContext();
    private PredictionEngine<AviationEquipmentData, EquipmentPrediction> predictor;
    private NeuralNetworkProvider _NeuralNetworkProvider = new NeuralNetworkProvider();

    private CategoriesProvider _CategoriesProvider = new CategoriesProvider();
    private List<Categories> _CategoriesList = new List<Categories>();
    private bool _IsThemesLoad = false;
```

```
    private ValidationMy _Valid = new ValidationMy();


    public PredictTestModelForm() {
      InitializeComponent();
      LoadAllDate();
    }

    private void LoadAllDate() {
      _CategoriesList = _CategoriesProvider.GetAllCategories();
      CategoriesCBox.DataSource = _CategoriesList;
      CategoriesCBox.ValueMember = "CategoriesId";
      CategoriesCBox.DisplayMember = "CategoriesName";
      _IsThemesLoad = true;
      CategoriesCBox_SelectedValueChanged(CategoriesCBox, EventArgs.Empty);
    }

    private void LoadData(string FilePath) {
      string localProj = Application.StartupPath + FilePath;
      // Define DataViewSchema for data preparation pipeline and trained model
      DataViewSchema modelSchema;

      // Load trained model
      ITransformer model = context.Model.Load(localProj, out modelSchema);
      // Evaluate the model
      // Use the model to make predictions
      predictor = context.Model.CreatePredictionEngine<AviationEquipmentData,
EquipmentPrediction>(model);
    }

    private void RunBtn_Click(object sender, EventArgs e) {
      if (timer1.Enabled) {
        timer1.Enabled = false;
        RunBtn.Text = "Запустити";
      } else {
        timer1.Enabled = true;
        RunBtn.Text = "Зупинити";
      }
    }

    private void CategoriesCBox_SelectedValueChanged(object sender, EventArgs e) {
      if (_IsThemesLoad) {
        _SelectedNeural = _NeuralNetworkProvider.SelectedNeuralNetworkByCategoriesId(
          Convert.ToInt32(CategoriesCBox.SelectedValue));
        if (_SelectedNeural.NeuralNetworkNames != "") {
          LoadData(_SelectedNeural.NeuralNetworkFileModel);
        } else {
          MessageBox.Show("Для обраної категорії поки не було згенеровано жодної моделі!");
        }
      }
    }
```

```
  private void timer1_Tick(object sender, EventArgs e) {
    Random rand = new Random();
    var datas = new AviationEquipmentData {
      Temperature = rand.Next(20, 100),
      Pressure = rand.Next(1, 10),
      Vibration = rand.Next(0, 10),
      OilLevel = rand.Next(0, 100),
      HoursOfOperation = rand.Next(0, 10000),
      LastMaintenanceDate = rand.Next(0, 365),
      WearLevel = rand.Next(0, 100)
    };
      var prediction = predictor.Predict(datas);
    RaportTBox.Text = ($"Дані:\r\n" +
      $"Температура = {datas.Temperature}\r\n" +
      $"Тиск = {datas.Pressure}\r\n" +
      $"Вібрація = {datas.Vibration}\r\n" +
      $"Рівень масла = {datas.OilLevel}\r\n" +
      $"Години роботи = {datas.HoursOfOperation}\r\n" +
      $"Днів від останнього обслуговування = {datas.LastMaintenanceDate}\r\n" +
      $"Рівень зносу = {datas.WearLevel}\r\n");

    // Умовний оператор для перевірки значення IsFaulty
    if (prediction.IsFaulty) {
      // Якщо IsFaulty == true
      RaportTBox.Text += "Прогноз поломки: можлива поломка\r\n";
    } else {
      // Якщо IsFaulty != true
      RaportTBox.Text += "Прогноз поломки: обладнання справне\r\n";
    }

    // Додавання ймовірності поломки
    RaportTBox.Text += $"Ймовірність поломки: {prediction.Probability:P2}\r\n";


    // Прокрутити текстове поле до останнього запису
    RaportTBox.SelectionStart = RaportTBox.Text.Length;
    RaportTBox.ScrollToCaret();

  }

  private void PredictionsBtn_Click(object sender, EventArgs e) {
    if (IsAlAviationEquipmentDataInputCorrect()) {
      GetScenarios();
    }
  }

  private void GetScenarios() {
    var datas = new AviationEquipmentData {
      Temperature = (float)Convert.ToDouble(TemperatureTBox.Text), Pressure =
(float)Convert.ToDouble(PressureTBox.Text),
```

```
        Vibration = (float)Convert.ToDouble(VibrationTBox.Text), OilLevel =
(float)Convert.ToDouble(OilLevelTBox.Text),
        HoursOfOperation = (float)Convert.ToDouble(HoursOfOperationTBox.Text),
        LastMaintenanceDate = (float)Convert.ToDouble(LastMaintenanceDateTBox.Text),
        WearLevel = (float)Convert.ToDouble(WearLevelTBox.Text)
      };
      RaportTBox.Text = ($"Дані:\r\n" +
        $"Температура = {datas.Temperature}\r\n" +
        $"Тиск = {datas.Pressure}\r\n" +
        $"Вібрація = {datas.Vibration}\r\n" +
        $"Рівень масла = {datas.OilLevel}\r\n" +
        $"Години роботи = {datas.HoursOfOperation}\r\n" +
        $"Днів від останнього обслуговування = {datas.LastMaintenanceDate}\r\n" +
        $"Рівень зносу = {datas.WearLevel}\r\n");

      var prediction = predictor.Predict(datas);
      // Умовний оператор для перевірки значення IsFaulty
      if (prediction.IsFaulty) {
        // Якщо IsFaulty == true
        RaportTBox.Text += "Прогноз поломки: можлива поломка\r\n";
      } else {
        // Якщо IsFaulty != true
        RaportTBox.Text += "Прогноз поломки: обладнання справне\r\n";
      }
      // Додавання ймовірності поломки
      RaportTBox.Text += $"Ймовірність поломки: {prediction.Probability:P2}\r\n";
    }


    private bool IsAlAviationEquipmentDataInputCorrect() {
      bool isCorrect = true;
      if (_Valid.IsDataConvertToDouble(TemperatureTBox.Text)) {
        TemperatureValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        TemperatureValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Valid.IsDataConvertToDouble(PressureTBox.Text)) {
        PressureValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PressureValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Valid.IsDataConvertToDouble(VibrationTBox.Text)) {
        VibrationValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        VibrationValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
       if (_Valid.IsDataConvertToDouble(OilLevelTBox.Text)) {
        OilLevelValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
```

```
        OilLevelValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Valid.IsDataConvertToDouble(LastMaintenanceDateTBox.Text)) {
        LastMaintenanceDateValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        LastMaintenanceDateValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Valid.IsDataConvertToDouble(WearLevelTBox.Text)) {
        WearLevelValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        WearLevelValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

  }
}
```