МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н. Каразіна

Факультет: **ННІ Каразінський банківський інститут**

Кафедра: **Інформаційних технологій та математичного моделювання**

Спеціальність: **125 Кібербезпека**

Освітня програма: **Кібербезпека у фінансових технологіях**

Група: АБ-41Б **денна форма навчання**

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему:

ОПТИМІЗАЦІЯ ПРОЦЕСУ РЕКРУТИНГУ ЗА ДОПОМОГОЮ ПРЕДИКТИВНОЇ АНАЛІТИКИ

ЗА НАКАЗОМ № 4601-5/335 ВІД 07 ЛЮТОГО 2025 РОКУ

здобувача вищої освіти **Юженко Єви Олександрівни**

Робота допущена до захисту в ЕК

протокол кафедри ІТММ № 13 від 31.05.2025р.

Завідувач кафедри ІТММ

к.п.н., доцент

_____ Н.І. Стяглик

Науковий керівник

к.ф.-м.н., доцент

_____ Г. В. Макарова__

м. Харків 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В. Н. Каразіна

Факультет навчально-науковий інститут "Каразінський банківський інститут"

Кафедра інформаційних технологій та математичного моделювання

Рівень вищої освіти перший (бакалаврський)

Спеціальність 125 Кібербезпека

Освітня програма Кібербезпека у фінансових технологіях

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ **Н. І. Стяглик**

Підпис ініціали, прізвище

"08" лютого 2025 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЄКТ)

Юженко Єва Олександрівна

(прізвище, ім'я, по батькові студента)

1. Тема роботи Оптимізація процесу рекрутингу за допомогою предиктивної аналітики

керівник роботи Макарова Ганна Валеріївна, к.ф.-м.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від "08" лютого 2025 року № 4601-5/335

2. Строк подання студентом роботи 15 травня 2025 року

3. Перелік питань, які потрібно розробити:

У розділі 1: Сутність і доцільність предиктивної аналітики в рекрутингу, принципи та етапи побудови моделей, джерела даних і виявлення прихованих закономірностей.

У розділі 2: Обґрунтування вибору інструментів реалізації (ML.NET, C#, Visual Studio), аналіз архітектури системи, порівняння з аналогами та оцінка ефективності розробленого рішення.

У розділі 3: Реалізація програмного продукту SmartRecruit, опис функціоналу системи, побудова моделей машинного навчання, розробка архітектури та ключових алгоритмів.

4. План роботи

№ з/п	Назви етапів роботи
1	Вибір здобувачем теми кваліфікаційної бакалаврської роботи
2	Затвердження плану і завдання кваліфікаційної бакалаврської роботи
3	Здача кваліфікаційної бакалаврської роботи керівнику
4	Підпис кваліфікаційної бакалаврської роботи керівника
5	Підпис кваліфікаційної бакалаврської роботи у нормоконтролера
6	Допуск завідувачем кафедри до захисту кваліфікаційної бакалаврської роботи
7	Захист кваліфікаційної бакалаврської роботи

5. Дата видачі завдання 08 лютого 2025 року

Студент _____ Є. О. Юженко

підпис ініціали, прізвище

Керівник роботи _____ Г. В. Макарова

підпис ініціали, прізвище

РЕФЕРАТ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ
«ОПТИМІЗАЦІЯ ПРОЦЕСУ РЕКРУТИНГУ ЗА ДОПОМОГОЮ ПРЕДИКТИВНОЇ АНАЛІТИКИ»
Юженко Єви Олександрівни

Дипломна робота містить: 89 сторінок, 2 таблиці, 27 рисунків, список літератури з 26 найменувань.

Об'єктом дослідження є процес рекрутингу та підбору персоналу на основі аналізу даних.

Предмет дослідження є застосування методів машинного навчання для прогнозування плинності кадрів у процесі рекрутингу.

Метою роботи є створення програмного засобу для прогнозування ймовірності звільнення працівника з використанням предиктивної аналітики.

Завданнями роботи є:

– у першому розділі розглянути теоретичні основи предиктивної аналітики, її значення в сучасному рекрутингу, а також визначити джерела даних для побудови моделей прогнозування;

– у другому розділі проаналізувати інструменти збору та обробки даних, порівняти існуючі програмні рішення у сфері інтелектуального рекрутингу, виділити переваги й недоліки та обґрунтувати створення власного застосунку;

– у третьому розділі описати функціональні можливості створеної системи, архітектуру програмного продукту, структуру інтерфейсу, механізми навчання та прогнозування, а також провести оцінювання її точності.

Актуальність дослідження обумовлена необхідністю використання інтелектуальних інструментів для покращення якості прийняття рішень у сфері найму персоналу.

За результатами дослідження сформовано програмний продукт SmartRecruit, який реалізує побудову, навчання, оцінювання та застосування моделі машинного навчання для прогнозування ймовірності звільнення кандидата.

Практична новизна: створення застосунку, що забезпечує гнучке управління моделями, має простий графічний інтерфейс і дозволяє здійснювати прогнозування ймовірність звільнення нового працівника.

Одержані результати можуть бути використані в HR-відділах компаній, кадрових агентствах, навчальних закладах для демонстрації застосування штучного інтелекту в управлінні персоналом, а також як основа для подальшого розвитку інтелектуальних систем у сфері рекрутингу.

КЛЮЧОВІ СЛОВА: РЕКРУТИНГ, МАШИННЕ НАВЧАННЯ, ПРОГНОЗУВАННЯ, ПЛИННІСТЬ КАДРІВ, ML.NET, C#, ПРЕДИКТИВНА АНАЛІТИКА, КЛАСИФІКАЦІЯ.

ABSTRACT
AT QUALIFICATION BACHELOR WORK
«OPTIMIZATION OF THE RECRUITMENT PROCESS USING PREDICTIONAL ANALYTICS»
Yuzhenko Yeva Oleksandrivna

The thesis contains: 89 pages, 2 tables, 27 figures, a list of references of 26 titles.

The object of the research is the process of recruiting and selecting personnel based on data analysis.

The subject of the research is the application of machine learning methods to predict staff turnover in the recruitment process.

The purpose of the work is to create a software tool for predicting the probability of employee dismissal using predictive analytics.

The tasks of a bachelor's degree are:

– in the first section, to consider the theoretical foundations of predictive analytics, its importance in modern recruiting, and to identify data sources for building forecasting models;

– in the second section, analyze data collection and processing tools, compare existing software solutions in the field of intelligent recruiting, highlight advantages and disadvantages, and justify the creation of your own application;

– in the third section, describe the functionality of the created system, the architecture of the software product, the interface structure, learning and prediction mechanisms, and evaluate its accuracy.

The relevance of the study is due to the need to use intelligent tools to improve the quality of decision-making in the field of personnel recruitment.

According to the results of the research, a software product called SmartRecruit was created, which implements the construction, training, evaluation and application of a machine learning model to predict the probability of a candidate's dismissal.

Practical novelty: creating an application that provides flexible model management, has a simple graphical interface, and allows you to predict the probability of a new employee being fired.

The results obtained can be used in HR departments of companies, recruitment agencies, and educational institutions to demonstrate the application of artificial intelligence in personnel management, as well as as a basis for the further development of intelligent systems in the field of recruiting.

KEYWORDS: RECRUITMENT, MACHINE LEARNING, FORECASTING, HUMAN RESOURCE FLOW, ML.NET, C#, PREDICTIVE ANALYTICS, CLASSIFICATION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧОК, СИМВОЛІВ І ТЕРМІНІВ

API (Application Programming Interface) – інтерфейс прикладного програмування, набір функцій для взаємодії між програмними модулями.

ARIMA (AutoRegressive Integrated Moving Average) – авторегресивна інтегрована модель ковзного середнього, що використовується в прогнозуванні часових рядів.

BTP (Business Travel Program) – програма службових відряджень або параметр категорії мобільності працівника.

C# (C-Sharp) – мова програмування загального призначення, яка використовується для створення прикладних програм на платформі .NET.

CSV (Comma-Separated Values) – формат текстових файлів, у якому значення розділені комами (або іншими роздільниками).

HR (Human Resources) – управління людськими ресурсами, кадрова служба.

KPI (Key Performance Indicator) – ключовий показник ефективності діяльності працівника чи процесу.

ML.NET – фреймворк машинного навчання для платформи .NET, розроблений компанією Microsoft.

NLP (Natural Language Processing) – технологія обробки природної мови, застосовується для аналізу текстів.

ВСТУП

У сучасному світі ринок праці зазнає динамічних змін, зумовлених глобальними економічними трансформаціями, стрімким розвитком цифрових технологій і зміною підходів до управління персоналом. Зростання конкуренції між компаніями за кваліфіковані кадри та постійна потреба в адаптації до нових умов змушують підприємства шукати ефективні методи оптимізації процесів найму. Традиційні підходи до рекрутингу, що ґрунтуються на ручному аналізі резюме, інтуїтивних рішеннях менеджерів з підбору персоналу та класичних тестових оцінюваннях, все частіше виявляються малоефективними, не лише через значні часові витрати, а й через високий ризик суб'єктивних помилок у прийнятті рішень [1].

Інтеграція інструментів штучного інтелекту та машинного навчання у сферу управління людськими ресурсами відкриває нові перспективи для підвищення точності оцінки кандидатів, зменшення витрат на процес найму та покращення прогнозованості кадрових рішень. Використання методів предиктивної аналітики дозволяє аналізувати великі обсяги даних про кандидатів, виявляти закономірності в успішних і неуспішних кар'єрних сценаріях та прогнозувати можливість адаптації нового співробітника до корпоративної культури компанії [2]. Це особливо актуально для ринків із високою динамікою звільнень та працевлаштування, де кожна помилка у підборі персоналу може призвести до суттєвих фінансових втрат.

Світові корпорації, такі як Google, Amazon і IBM, уже активно впроваджують алгоритми машинного навчання у процеси рекрутингу, використовуючи великі обсяги даних для автоматизації аналізу резюме, проведення персоналізованих тестувань та прогнозування успішності потенційного кандидата на основі історичних даних. Такі підходи демонструють високу ефективність, проте часто є недоступними для середнього та малого бізнесу через високу вартість імплементації та необхідність спеціалізованих знань у сфері Data Science.

В умовах України проблема ефективного рекрутингу стає ще більш актуальною з огляду на нестабільну економічну ситуацію, трудову міграцію та необхідність адаптації до нових викликів глобального ринку. Багато українських підприємств досі використовують застарілі методи підбору персоналу, що знижує їхню конкурентоспроможність та призводить до значних витрат через високу плинність кадрів. У цьому контексті впровадження сучасних методів предиктивної аналітики може стати ключовим фактором підвищення ефективності управління людськими ресурсами, дозволяючи компаніям швидше знаходити та утримувати найкращих спеціалістів.

Метою роботи є створення програмного засобу для прогнозування ймовірності звільнення працівника з використанням предиктивної аналітики, обґрунтування вибору інструментів розробки, формалізація задачі та реалізація алгоритмів машинного навчання у системі підтримки прийняття кадрових рішень.

Для досягнення поставленої мети необхідно вирішити такі основні задачі:

– обґрунтувати сутність та значення предиктивної аналітики у сфері рекрутингу шляхом аналізу теоретичних підходів, визначення переваг застосування машинного навчання та великих даних у прогнозуванні кадрових рішень;

– дослідити сучасні інструменти для збору, обробки та аналізу даних у сфері рекрутингу з метою визначення їхньої ефективності, особливостей застосування та доцільності інтеграції в автоматизовані системи управління персоналом;

– розробити програмний застосунок, що використовує предиктивну аналітику для оцінки потенційних кандидатів з архітектурних принципів та необхідного алгоритмів машинного навчання;

– оцінити ефективність розробленого рішення шляхом порівняння його можливостей з існуючими аналогами, виявлення переваг, недоліків та

потенційних напрямів удосконалення системи автоматизованого підбору персоналу.

Об’єкт дослідження – процес рекрутингу та підбору персоналу на основі аналізу даних.

Предмет дослідження – є застосування методів машинного навчання для прогнозування плинності кадрів у процесі рекрутингу.

Методи дослідження включають:

– аналіз літературних джерел та існуючих рішень – для визначення сучасних підходів до використання предиктивної аналітики у рекрутингу;

– методи машинного навчання – для побудови та оцінки моделі прогнозування успішності кандидатів;

– експериментальне моделювання – для тестування та валідації розробленого алгоритму на реальних даних.

– порівняльний аналіз – для оцінки ефективності запропонованого рішення у порівнянні з традиційними методами відбору персоналу.

Таким чином, необхідність розробки ефективних інструментів для оптимізації рекрутингових процесів є не лише науково-технічним викликом, а й вагомим практичним завданням, розв'язання якого сприятиме підвищенню ефективності бізнесу та стійкості ринку праці України в умовах сучасних глобальних тенденцій.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ПРЕДИКТИВНОЇ АНАЛІТИКИ В РЕКРУТИНГУ

1.1 Сутність та значення предиктивної аналітики

Предиктивна аналітика – це розділ розширеної аналітики, що використовує історичні дані, статистичні алгоритми, методи обробки даних та машинного навчання для прогнозування ймовірних майбутніх результатів [3]. Іншими словами, на основі великого масиву наявних даних вона визначає закономірності і тенденції, які дозволяють передбачити події, що можуть відбутися в майбутньому. У ієрархії підходів до аналізу даних предиктивна аналітика відповідає на запитання «що може статися?», доповнюючи описову («що вже сталося?»), діагностичну («чому це сталося?») та прескриптивну («що робити далі?») аналітику. Метою предиктивної аналітики є не просто констатувати минулі факти, а надати найкращу оцінку майбутніх подій на основі наявних даних.

Хоча термін «предиктивна аналітика» набув популярності відносно недавно, сам підхід прогнозування на основі даних має давнє коріння. Одним із перших задокументованих випадків використання аналізу даних для прогнозів було страхування морських перевезень наприкінці XVII століття. Так, у 1689 році в лондонській кав’ярні Едварда Ллойда збиралися мореплавці, торговці і страховики, які обмінювалися достовірною інформацією про морські рейси і ризики [4]. Така колективна база даних про страхові випадки дала змогу оцінювати ризик плавань (звідси й походить термін андерайтинг, від практики підписуватися під описом ризику на страховому аркуші) та заклала основи для майбутнього аналізу ризиків. Надалі, протягом XIX–XX століть, розвиток статистичних методів (наприклад, регресійного аналізу) і поява електронно-обчислювальних машин суттєво розширили можливості аналітики. Зі збільшенням

обчислювальних потужностей предиктивна аналітика еволюціонувала від відносно простих моделей (лінійна чи логістична регресія) до значно складніших алгоритмів машинного навчання. Конвергенція величезних масивів даних, удосконалення алгоритмів та зростаючі обчислювальні ресурси привели до того, що прогнозні моделі стали високоточними та широко застосовуваними у бізнесі.

В останні десятиліття спостерігається вибуховий розвиток предиктивної аналітики, зумовлений кількома чинниками: стрімким зростанням обсягів і різновидів даних, швидшими і дешевшими обчисленнями, появою більш зручного програмного забезпечення для аналізу, а також жорсткішою конкуренцією на ринках. Якщо раніше такі прогнози були прерогативою статистиків, то тепер інтуїтивні програмні рішення дозволяють залучати до роботи з предиктивною аналітикою і бізнес-аналітиків, і фахівців галузі.

Для предиктивної аналітики характерні такі особливості:

– прогнозування і ймовірнісний характер. Результатом є прогноз (очікування майбутнього стану) з певною імовірністю, а не стовідсотково точне передбачення. Моделі видають оцінки ймовірності подій (наприклад, ймовірність того, що кандидат досягне успіху на посаді), отже, висновки завжди мають імовірнісну природу, а не гарантію;

– опора на дані та виявлення закономірностей. Предиктивна аналітика базується на великих масивах історичних даних і застосуванні до них статистичного моделювання та методів штучного інтелекту. Спеціальні алгоритми аналізують поточні та минулі факти, щоби виявити приховані патерни і взаємозв’язки, які неможливо знайти неозброєним оком [5]. Завдяки цьому організації можуть на основі даних ідентифікувати фактори, що впливають на майбутні результати (ризики або можливості);

– використання великих і різнорідних джерел. Для побудови прогнозів залучаються великі дані – інформація з багатьох джерел, часто розрізнених, у великих обсягах і різних форматах. Історичні записи,

транзакції, логи, зображення, текстові документи та інші типи даних можуть бути інтегровані в єдину модель. Сучасний бізнес генерує величезні масиви різнорідних даних, і предиктивна аналітика тісно пов’язана з технологіями Big Data та наукою про дані, які дозволяють ці масиви зберігати й опрацьовувати. Загальна закономірність – чим більше релевантних даних доступно для аналізу, тим точнішими можуть бути прогнози (звісно, за умови належної якості даних) [6];

– застосування складних алгоритмів. Для аналізу даних і побудови прогнозів використовуються сучасні методи Data Mining та машинного навчання. Поширеними підходами є побудова предиктивних моделей різних типів – регресійних (що прогнозують числові показники), класифікаційних (що відносять об’єкт до категорії), кластерних (що групують схожі об’єкти) тощо. Такі моделі можуть бути реалізовані за допомогою різноманітних алгоритмів: від класичних (лінійна та логістична регресія, дерева рішень) до більш складних ансамблевих методів (випадкові ліси, градієнтний бустинг) та нейронних мереж. Вибір алгоритму залежить від характеру задачі та даних, проте усі вони спрямовані на те, щоб навчитися наявним даним і зробити узагальнення для нових випадків;

– потреба у валідації та оновленні. Побудовані моделі потребують постійної перевірки адекватності та коригування. Експерти галузі повинні інтерпретувати результати моделювання та підтверджувати, що вони мають сенс з практичної точки зору, перш ніж покладатися на них у прийнятті рішень. Крім того, моделі необхідно регулярно оновлювати новими даними та переглядати їхні параметри, оскільки ринкові умови і поведінка людей змінюються. Предиктивна аналітика – це ітеративний процес: після початкового розгортання модель відстежують за показниками точності прогнозів і вдосконалюють, щоб підвищити її ефективність з часом.

Предиктивна аналітика задіює поєднання статистичних методів і алгоритмів штучного інтелекту для побудови прогнозних моделей. Процес

побудови такої моделі умовно поділений на кілька етапів та зображений на рис. 1.1.

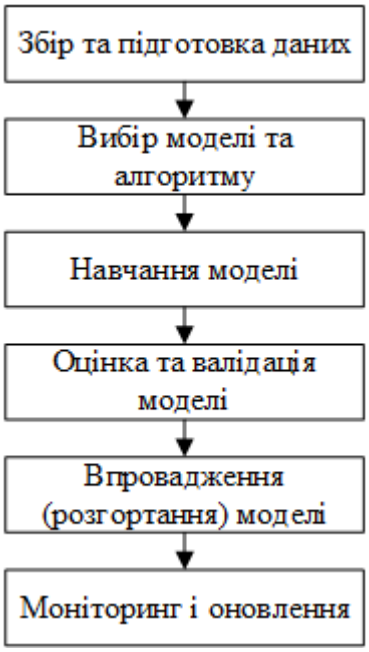

Рис. 1.1. Процес побудови моделі предиктивної аналітики

На початку організація збирає великий обсяг даних, релевантних до питання, на яке потрібно відповісти (наприклад, дані про кандидатів та їхні результати роботи для прогнозування успішності найму). Дані можуть надходити з різних джерел – внутрішніх баз даних, анкет кандидатів, результатів тестувань, соціальних мереж тощо. Важливим кроком є інтеграція цих різнорідних даних в єдине сховище та очистка даних: виправлення помилок, обробка пропущених значень, приведення форматів до єдиних стандартів. Якісна підготовка даних є критичною, адже неточності чи шуми в даних можуть призвести до хибних моделей.

На основі сформульованої задачі та наявних даних визначають тип моделі, що буде побудовано. Якщо потрібно передбачити певний кількісний показник (скажімо, ймовірність звільнення співробітника або бал результативності), обирають модель регресії. Якщо мета – класифікувати об’єкт (наприклад, чи варто наймати даного кандидата – так/ні), використовують моделі класифікації. Для виявлення сегментів або схожих груп серед даних (наприклад, типи кандидатів) застосовують методи

кластеризації. Іноді в рекрутингу використовують і аналіз часових рядів – наприклад, щоб прогнозувати, коли компанії знадобиться додатковий персонал, зважаючи на сезонні коливання бізнес-активності. Відповідно до обраного типу моделі підбирають алгоритм: це може бути класичний статистичний метод або алгоритм машинного навчання. Наприклад, для класифікації часто застосовують логістичну регресію, дерево рішень або нейронну мережу; для кластеризації – алгоритм k-середніх або ієрархічну кластеризацію; для прогнозування часових рядів – моделі типу ARIMA.

Обраний алгоритм застосовують до підготовлених даних. Модель «навчається» на історичних даних: алгоритм аналізує приклади, де відомий фактичний результат (наприклад, дані про попередніх кандидатів і те, чи стали вони успішними працівниками), і налаштовує свої параметри так, щоб відтворити ці результати. Цей процес називають тренуванням моделі. У ході навчання використовуються методи оптимізації, які мінімізують різницю між прогнозованими моделлю значеннями і реальними результатами в навчальних даних. Важливо уникнути перенавчання (overfitting), коли модель надто підлаштовується під навчальні дані і втрачає здатність узагальнювати на нові випадки. Тому часто дані ділять на навчальну і тестову вибірки: модель тренують на одних даних, а її якість перевіряють на інших, відкладених для оцінки.

Після навчання модель перевіряють на тестових даних або шляхом крос-валідації, щоб оцінити її точність і надійність. Використовуються метрики якості, наприклад, точність прогнозу, повнота, F-міра (для класифікації) або середня квадратична помилка (для регресії). Якщо модель демонструє незадовільні результати, виконують налаштування: змінюють алгоритм, додають більше даних, створюють нові ознаки (features) або параметри моделі. Процес навчання може повторюватися декілька разів, поки не буде досягнуто оптимальної продуктивності.

Коли модель пройшла перевірку, її впроваджують у реальний процес ухвалення рішень. У контексті рекрутингу це може означати інтеграцію

моделі у систему відбору персоналу: наприклад, модель оцінює нових кандидатів і виставляє їм певний рейтинг або рекомендацію (найняти/не наймати) на основі прогнозованої продуктивності. Впроваджена модель автоматизовано аналізує нові дані (наприклад, резюме, результати співбесід) і генерує прогноз для кожного нового випадку.

Після впровадження важливо стежити за тим, як модель працює на реальних даних. Час від часу якість прогнозів оцінюють на основі фактичних результатів (наприклад, відстежують, як справи у найнятих за рекомендацією моделі працівників: чи справді вони успішні). Якщо точність починає падати (скажімо, через зміну умов ринку чи появу нових типів кандидатів), модель перенавчають на актуальніших даних або коригують її алгоритми. Таким чином, предиктивна модель підтримується в актуальному стані протягом її життєвого циклу.

Отже, в основі роботи предиктивної аналітики лежить ідея, що патерни, виявлені у минулих даних, збережуть свою дію для майбутніх випадків при достатньо стабільних умовах. Алгоритми машинного навчання фактично узагальнюють досвід, наявний у даних, і застосовують його для передбачення нових ситуацій.

Важливо зауважити, що різні алгоритми мають свої переваги і сфери ефективного застосування: наприклад, нейронні мережі добре виявляють складні нелінійні залежності, тоді як регресійні моделі простіші в інтерпретації. Часто на практиці використовують ансамблі моделей (комбінації кількох алгоритмів), щоб підвищити надійність прогнозу. Окрім того, процес побудови моделей нерідко включає співпрацю аналітиків з експертами з управління персоналом: спільно визначають, які фактори слід врахувати (наприклад, стаж роботи, рівень освіти, результати тестів), і перевіряють інтуїтивну логічність отриманих моделей.

За останні роки предиктивна аналітика стала одним з найперспективніших інструментів у сфері управління персоналом і рекрутингу. Традиційний процес найму часто базувався на експертній

інтуїції рекрутерів і керівників, що підвищувало ризик суб'єктивних помилок. Натомість підхід, заснований на даних, дозволяє приймати більш обґрунтовані та об'єктивні рішення щодо відбору талантів. Особливо у великих компаніях, де щороку обробляються тисячі резюме, використання алгоритмів прогнозування допомагає виявити найперспективніших кандидатів і оптимізувати процес найму персоналу.

Переваги використання предиктивної аналітики в рекрутингу. Застосування аналізу даних і прогнозних моделей на різних етапах рекрутингової «воронки» забезпечує ряд суттєвих переваг:

Підвищення якості найму

Завдяки аналізу великих обсягів історичних даних про кандидатів і їхній подальший успіх (або неуспіх) на посаді, алгоритми можуть виділити характеристики, що статистично притаманні успішним працівникам. Це дозволяє більш точно оцінювати відповідність кандидата вимогам посади і культурі організації. У результаті компанія наймає більш кваліфікованих і придатних працівників, що позитивно впливає на продуктивність і командну динаміку.

Як зазначається в аналітичних оглядах, використання AI-інструментів для оцінки кандидатів корелює з вищою ефективністю співробітників на роботі. Практичний приклад: корпорація Xerox, проаналізувавши дані психометричних тестів своїх співробітників, побудувала модель ідеального кандидата на посаду оператора кол-центру; запровадження цієї моделі при відборі дозволило знизити плинність нових працівників на 20% лише за пів року [7].

Швидкість найму і зменшення витрат

Предиктивні моделі можуть автоматично ранжувати або фільтрувати резюме кандидатів за ступенем відповідності, що значно прискорює відбір. Рутинні етапи (перевірка резюме, первинний відбір за формальними критеріями) можуть виконуватися алгоритмами, звільняючи час рекрутерів для глибшої роботи з фіналістами. Це скорочує time-to-hire – середній час

закриття вакансії. Наприклад, компанія Google проаналізувала власний процес відбору і з'ясувала, що для прийняття якісного рішення достатньо провести з кандидатом близько чотирьох співбесід; додаткові раунди майже не підвищували точність прогнозу успішності (лише приблизно на 1%). На основі цих висновків Google зменшила кількість раундів інтерв'ю, що дозволило скоротити медіанний час найму з 180 до 47 днів [8].

Швидше закриття вакансій без втрати якості напряму впливає на витрати: компанія економить кошти, адже скорочується проста вакансії і потреба витрачати ресурси на додатковий пошук. Крім того, зменшується кількість неправильних наймів – коли взято на роботу невідповідну людину і доводиться незабаром шукати заміну.

Передбачення ймовірності успішності кандидата дає змогу уникнути прийому на роботу тих, хто, ймовірно, не впорається, що економить кошти на повторному рекрутингу і навчанні. Загалом досвід компаній свідчить про зниження витрат на найм завдяки аналітиці даних.

Зниження ризиків і плинності персоналу

Однією з суттєвих вигод є те, що рішення щодо найму стають менш ризикованими. Підкріплені даними прогнози зменшують ймовірність помилки – наприклад, виключають кандидатів, які з високою імовірністю виявляться непродуктивними або швидко звільняться. Аналізуючи характеристики співробітників, які довго пропрацювали і досягли успіхів, модель може виявити фактори, що сприяють довготривалій співпраці. Врахування таких факторів при відборі нових кандидатів підвищує retention rate (коефіцієнт утримання персоналу).

Навіть після найму HR-відділ може застосовувати предиктивну аналітику для прогнозування ймовірності звільнення кожного співробітника (наприклад, на основі рівня задоволеності, результатів оцінок, динаміки кар'єри тощо) і завчасно вживати заходів для утримання цінних кадрів [9]. Показовий кейс – банк Credit Suisse розробив модель, що прогнозує ймовірність звільнення співробітника протягом року з урахуванням різних

факторів (розмір команди, оцінки керівника, підвищення, життєві події тощо).

Використання цих прогнозів дозволило менеджерам вжити превентивних заходів і скоротити відтік кадрів, зекономивши близько $70 млн на рік на витратах, пов’язаних із наймом і навчанням нових працівників.

Об’єктивність та зменшення упередженості

Алгоритми, якщо вони правильно налаштовані, здатні оцінювати кандидатів більш неупереджено, ніж людина, оскільки вони спираються на чіткі дані і визначені метрики [10]. Це може допомогти знизити вплив людських стереотипів чи несвідомої упередженості на рішення про найм. Наприклад, модель може ігнорувати такі демографічні характеристики, як стать чи вік, фокусуючись на об’єктивних предикторах успіху (навички, досвід, результати тестів).

Таким чином, підбір персоналу стає справедливішим і різноманітнішим. Водночас, важливо стежити, щоб самі історичні дані не містили упередженості (якщо в минулому компанія надавала перевагу певним групам кандидатів, модель може непомітно успадкувати ці тенденції).

Тому впровадження предиктивної аналітики потребує постійного аудиту критеріїв моделі і коригування, щоб уникнути algorithmic bias. Загалом, при належному контролі, аналітика сприяє більш прозорим і обґрунтованим рішенням у наймі персоналу.

Стратегічне планування і прогноз потреб

Предиктивна аналітика корисна не лише на етапі відбору, а й для стратегічного HR-планування. Аналіз тенденцій дозволяє прогнозувати майбутні потреби в персоналі. Наприклад, врахувавши темпи зростання компанії, відкриття нових підрозділів або сезонні коливання бізнесу, модель може передбачити, які фахівці і коли знадобляться компанії [11]. Це дає змогу завчасно сформувати резерв кандидатів або програми найму, а також оптимально розподіляти бюджет на рекрутинг.

Прогнозування кадрових потреб особливо актуальне для великих організацій, де запізнення з наймом може призвести до втрати вигоди, а надлишковий найм – до зайвих витрат. Завдяки даним HR-аналітики керівництво може приймати проактивні рішення щодо розвитку персоналу, планувати навчання для закриття майбутніх компетентнісних прогалин тощо.

Покращення досвіду кандидатів

Даний аспект часто недооцінюють, але аналітика також впливає на те, як кандидати переживають процес найму. Коли процес відбору проходить швидше і більш прозоро (наприклад, кандидати отримують швидкий зворотний зв'язок або більш цільові запитання на співбесіді завдяки тому, що рекрутеру підказує система), задоволеність кандидатів зростає [12].

Компанія, що оперативно і професійно взаємодіє з кандидатами, формує позитивний бренд роботодавця. Предиктивні системи можуть підказати, на яких етапах виникають «вузькі місця» або довгі затримки, і HR-менеджер виправить процес для більшої ефективності. Врешті, кращий досвід кандидатів означає більшу ймовірність, що обраний кандидат прийме пропозицію роботи і новий співробітник з самого початку матиме позитивне враження про роботодавця.

HR-аналітика стає ключовим елементом сучасного управління персоналом, дозволяючи не лише аналізувати поточний стан робочої сили, а й прогнозувати майбутні тенденції та оптимізувати процеси найму. Згідно з даними Global Market Insights, глобальний ринок HR-аналітики зріс до 3,7 млрд доларів у 2023 році та, за прогнозами, досягне 11,1 млрд доларів до 2032 року, що підтверджує її зростаюче значення для бізнесу [13]. Використання предиктивної аналітики, яка є невід'ємною складовою сучасних HR-інструментів, дозволяє компаніям значно підвищити ефективність рекрутингу: наприклад, за даними Mike Pritula Academy, впровадження HR-аналітики може збільшити ефективність процесу підбору персоналу на 80%. Це доводить, що аналітичні підходи не лише спрощують

процес найму, а й забезпечують якісніші рішення щодо залучення та утримання талантів, формуючи конкурентні переваги для організацій.

1.2 Використання великих даних у рекрутингу

Сучасний ринок праці характеризується значною конкуренцією, високою динамічністю змін та зростаючими вимогами до кандидатів. У таких умовах традиційні методи рекрутингу, які базуються на ручному аналізі резюме та інтуїтивних рішеннях HR-фахівців, часто виявляються недостатньо ефективними. Збільшення обсягів інформації, яку роботодавці можуть використовувати для оцінки кандидатів, зумовило необхідність впровадження аналітичних підходів, що використовують великі дані.

Великі дані у сфері рекрутингу – це масиви різнорідної, структурованої та неструктурованої інформації, які дозволяють значно покращити процес підбору персоналу. Завдяки їх аналізу компанії отримують можливість не лише швидше знаходити кандидатів, що відповідають вимогам вакансії, а й прогнозувати їхню ефективність, адаптивність та лояльність до компанії.

Ключовою перевагою використання великих даних у рекрутингу є можливість автоматизованого аналізу значних обсягів інформації, що надходять із різних джерел. Це дозволяє мінімізувати вплив людського фактора, зменшити витрати часу на аналіз резюме, а також підвищити точність прогнозів щодо майбутньої продуктивності кандидатів. Наприклад, компанії, які активно використовують аналітику великих даних, можуть оцінювати відповідність претендента вакансії не лише за формальними критеріями (освіта, досвід роботи), але й на основі глибшого аналізу його поведінкових характеристик, кар'єрного шляху, рекомендацій та навіть особистісних якостей.

Умовно всі джерела великих даних, які можуть бути використані у рекрутингу, можна поділити на внутрішні та зовнішні. Внутрішні дані

формуються в межах компанії та містять інформацію про минулий досвід найму, ефективність співробітників, показники їхньої продуктивності та лояльності. Зовнішні дані надходять із відкритих онлайн-ресурсів, соціальних мереж, платформ для пошуку роботи, що дозволяє отримати більш повний портрет потенційного кандидата.

На рис. 1.2 представлено основні джерела великих даних у рекрутингу, які використовуються для покращення процесу підбору персоналу.

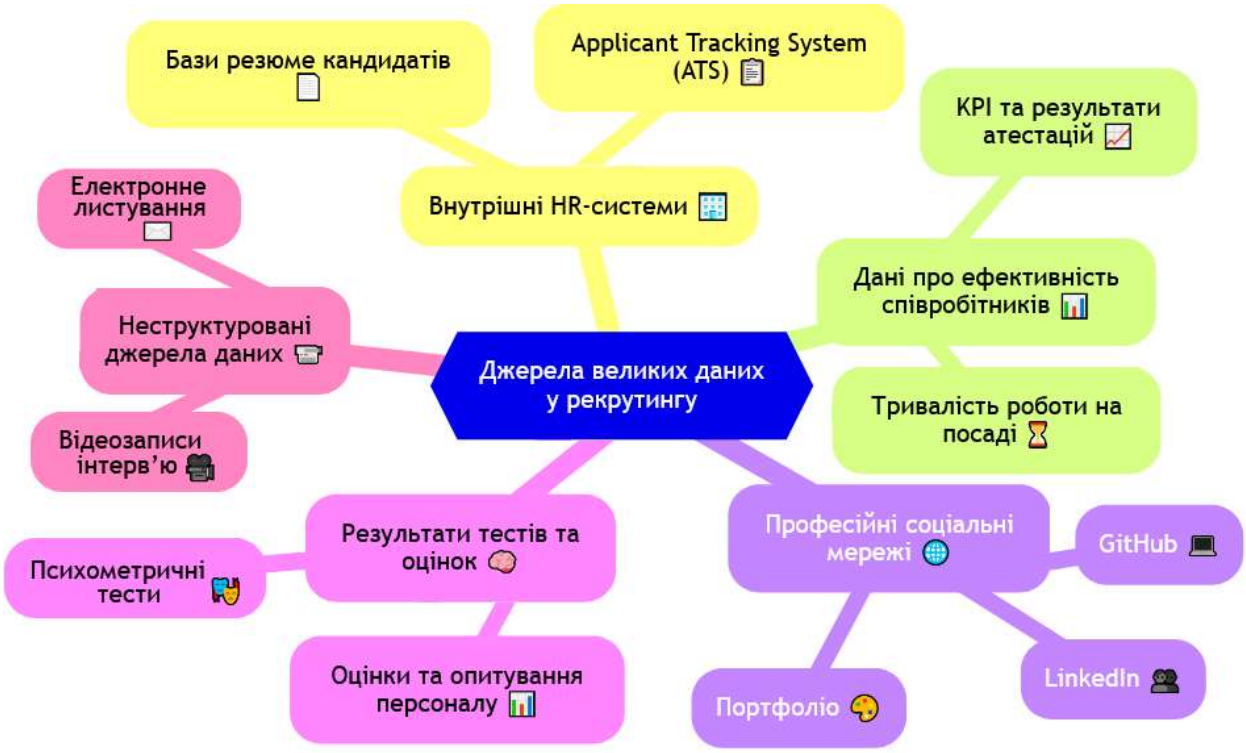

Рис. 1.2. Основні джерела великих даних для рекрутингу

Одним із найцінніших джерел є внутрішні HR-системи, що містять інформацію про попередні процеси відбору. Це можуть бути бази даних резюме, історія відгуків кандидатів на вакансії, записи про проходження співбесід, результати тестувань та інші параметри, що допомагають аналізувати ефективність рекрутингових кампаній [14]. Наприклад, якщо певна категорія працівників регулярно звільняється через невідповідність очікуванням, то аналіз попередніх даних дозволить виявити причини та скоригувати критерії відбору.

Також важливим джерелом є дані про ефективність і кар’єрний шлях наявних співробітників. У цю категорію входять ключові показники продуктивності (KPI), результати атестацій, частота підвищень, тривалість роботи на різних посадах. Наприклад, якщо аналіз виявляє, що співробітники, які раніше змінювали багато місць роботи, частіше звільняються у перший рік після найму, це може бути враховано при прийнятті рішень щодо нових кандидатів.

Значний обсяг корисної інформації надають професійні соціальні мережі та онлайн-профілі, такі як LinkedIn, GitHub, Behance або ResearchGate. Вони дозволяють аналізувати не лише формальні навички кандидатів, а й їхню професійну активність, участь у проєктах, наявність рекомендацій, а також тенденції кар’єрного зростання. Наприклад, дані з GitHub можуть показати реальний рівень навичок програміста на основі його внесків у відкриті проєкти, що є набагато об’єктивнішим критерієм, ніж просто зазначені навички в резюме.

Результати психометричних тестів та опитувань персоналу також можуть стати корисним джерелом великих даних у процесі рекрутингу. Аналіз поведінкових характеристик кандидатів, рівня стресостійкості, здатності до командної роботи та комунікативних навичок дозволяє не лише підбирати відповідних фахівців, але й прогнозувати їхню інтеграцію в команду та довготривалу продуктивність.

Окрім традиційних джерел, великі дані можуть містити і неструктуровану інформацію, наприклад, електронне листування, відеозаписи інтерв’ю або навіть поведінкові патерни під час спілкування. Використання штучного інтелекту дозволяє аналізувати тональність відповіді кандидата, його невербальні сигнали та навіть реакцію на певні питання, що може підвищити об'єктивність оцінки.

Для ефективного використання всіх цих джерел дані повинні бути консолідовані і підготовлені до аналізу. Часто інформація надходить у різних форматах – структуровані таблиці, текстові поля резюме, аудіо- та

відеозаписи співбесід. Тут вступають в дію методи обробки природної мови (NLP) та аналізу мультимедіа. Наприклад, щоб врахувати текстові резюме в моделі, система повинна перетворити текст у числові ознаки: виділити ключові навички, згаданий досвід, рівень освіти тощо [15]. NLP-алгоритми можуть автоматично парсити резюме та профілі, нормалізувати назви посад, навичок, компаній для уніфікації даних. Аналогічно, відеозаписи співбесід можуть бути проаналізовані за допомогою технологій розпізнавання мови (щоб отримати текст відповіді кандидата) і навіть комп'ютерного зору (для фіксації міміки чи жестів). Хоча такі підходи ще розвиваються, в перспективі вони дозволять включити в аналіз поведінкові особливості кандидатів.

Машинне навчання є тим інструментом, який дає змогу витягти знання з великих масивів даних. Класичні методи статистики могли використовувати обмежену кількість параметрів, тоді як сучасні ML-алгоритми здатні одночасно враховувати десятки і сотні різних показників кандидата. Наприклад, модель може брати до уваги сукупність факторів (освіта, досвід, результати тестів, активність у професійних спільнотах, ключові слова в резюме, рівень володіння мовами тощо) і на основі цього багатовимірного профілю прогнозувати, чи досягне кандидат успіху на конкретній посаді. Подібні прогнози будуються на навчанні моделей на великих наборах даних: якщо у нас є дані про тисячі кандидатів і відомо, хто з них став успішним працівником, алгоритм знайде нелінійні залежності між характеристиками кандидатів і ауткомами. Саме за рахунок обробки великих даних алгоритми ML нерідко перевершують людину – вони виявляють тонкі патерни, які неможливо вирахувати вручну. Як наслідок, підвищується і точність, і швидкість прогнозування.

Практичний приклад інтеграції великих даних та ML у рекрутингу – платформа LinkedIn. Володіючи однією з найбільших у світі баз даних професійних резюме і контактів, LinkedIn запровадила інтелектуальну систему Talent Solutions. Зокрема, інструмент LinkedIn Recruiter використовує алгоритми машинного навчання для ранжування потенційних

кандидатів під конкретну вакансію: коли рекрутер формує пошуковий запит, система аналізує профілі тисяч фахівців і на основі їхнього досвіду, навичок, активності та навіть географії автоматично виводить у топ тих, хто найбільше відповідає вимогам [16]. За публічними даними, LinkedIn застосовує градієнтні бустингові дерева рішень та інші складні ML-моделі для зіставлення профілів з вакансіями. Цей приклад демонструє силу великих даних: маючи в розпорядженні глобальну базу користувачів, алгоритм може навчитися прогнозувати збіги між кандидатом і посадою краще, ніж це зробив би рекрутер, прочитавши декілька резюме.

Великі дані також дозволяють врахувати зовнішні фактори при прогнозуванні кадрових рішень. Наприклад, можна інтегрувати в модель дані ринку праці: рівень безробіття, середню зарплату по галузі, активність конкурентів у наймі – і таким чином передбачати складність закриття тієї чи іншої вакансії. Такі моделі допомагають HR-відділам реалістично планувати терміни і стратегії рекрутингу (скажімо, розуміти, що на певну посаду знадобиться довший час пошуку або більш привабливий компенсаційний пакет у період високого попиту на відповідних спеціалістів).

На практиці, щоб реалізувати можливості машинного навчання і великих даних, компанії впроваджують спеціалізовані HR-tech рішення. Деякі використовують готові платформи для аналізу даних про кандидатів, інші розробляють власні моделі «на замовлення». Наприклад, існують системи, які інтегруються з ATS і в реальному часі підказують рекрутеру рейтинг кожного кандидата, прогнозуючи, хто пройде далі [17]. Інші системи фокусуються на прогнозуванні кар’єрного шляху: визначають, яких працівників варто підвищити, щоб утримати, або хто може стати лідером. Усі такі рішення працюють на перетині великих даних та ML, постійно вдосконалюючись по мірі надходження нової інформації.

Варто підкреслити, що успіх застосування предиктивної аналітики в рекрутингу залежить як від технологій, так і від людського чинника. Алгоритми машинного навчання – дуже потужний інструмент, але вони

потребують правильного налаштування і якісних даних. Великі дані можуть давати хибні кореляції, якщо не врахувати контекст. Тому найкращі результати досягаються при співпраці data science спеціалістів, які знаються на методах ML, та HR-експертів, які розуміють сутність рекрутингових процесів. Разом вони можуть сформулювати правильні гіпотези, інтерпретувати виходи моделей і вирішити, як саме інтегрувати прогнози в реальну практику найму.

Предиктивна аналітика являє собою потужний підхід, що поєднує математичні моделі, машинне навчання та великі дані для підтримки прийняття рішень. У сфері рекрутингу її сутність полягає в прогнозуванні успіху кандидатів і потреб бізнесу на основі доказових даних. Значення цього підходу важко переоцінити: він підвищує ефективність і об’єктивність процесу найму, знижує витрати і ризики, допомагає будувати довгострокову кадрову стратегію. Сучасні компанії, що використовують предиктивну аналітику, отримують конкурентну перевагу на ринку талантів, адже можуть проактивно залучати і утримувати потрібних фахівців. З розвитком технологій машинного навчання та зростанням обсягів даних роль предиктивної аналітики в управлінні персоналом буде ще більш зростати, відкриваючи нові горизонти оптимізації рекрутингових процесів.

РОЗДІЛ 2
ІНСТРУМЕНТИ ДЛЯ ДОСЛІДЖЕННЯ ТА АКТУАЛЬНІСТЬ ЇХ ЗАСТОСУВАННЯ

2.1 Опис інструментів для збору та аналізу даних

Сучасні тенденції у сфері підбору персоналу вимагають використання інтелектуальних технологій, здатних обробляти великі обсяги даних та забезпечувати точні прогнози щодо ефективності майбутніх працівників. Процес аналізу даних у рекрутингу ґрунтується на системному використанні низки інструментів, які забезпечують збирання, підготовку, обробку та інтерпретацію як структурованої, так і неструктурованої інформації.

Оскільки процес прийняття рішень у сфері найму значною мірою залежить від точності й повноти даних, першочергове значення має використання програмних засобів, які дозволяють здійснювати ефективний аналіз показників кандидатів та працівників. У даному дослідженні головним технологічним засобом є ML.NET, призначена для розробки моделей безпосередньо в середовищі .NET. ML.NET дає змогу будувати предиктивні моделі, не вдаючись до зовнішніх мов програмування, таких як Python або R, що забезпечує зручність для інтеграції з існуючими інформаційними системами на базі C#.

ML.NET – це програмний фреймворк машинного навчання від компанії Microsoft, створений спеціально для розробників .NET-платформи. Він дозволяє створювати, тренувати та застосовувати моделі машинного навчання без необхідності використовувати сторонні мови програмування. У сфері рекрутингу ML.NET є особливо цінним, оскільки дає змогу інтегрувати інтелектуальні механізми прогнозування безпосередньо в десктопні або вебзастосунки, які використовуються HR-відділами. За допомогою ML.NET можна створювати класифікаційні моделі, які прогнозують, наприклад, імовірність звільнення кандидата, ґрунтуючись на його демографічних,

поведінкових та професійних характеристиках. Перевага полягає в тому, що все навчання та прогнозування здійснюється на основі локальних даних, що забезпечує більший контроль над конфіденційністю. Такий підхід особливо актуальний для невеликих та середніх компаній, які не мають ресурсів для впровадження масштабних зовнішніх аналітичних платформ.

На рис. 2.1 зображено життєвий цикл створення та використання моделі машинного навчання на основі ML.NET, що відображає етапи повного циклу – від збирання даних до впровадження моделі в прикладну систему [18].

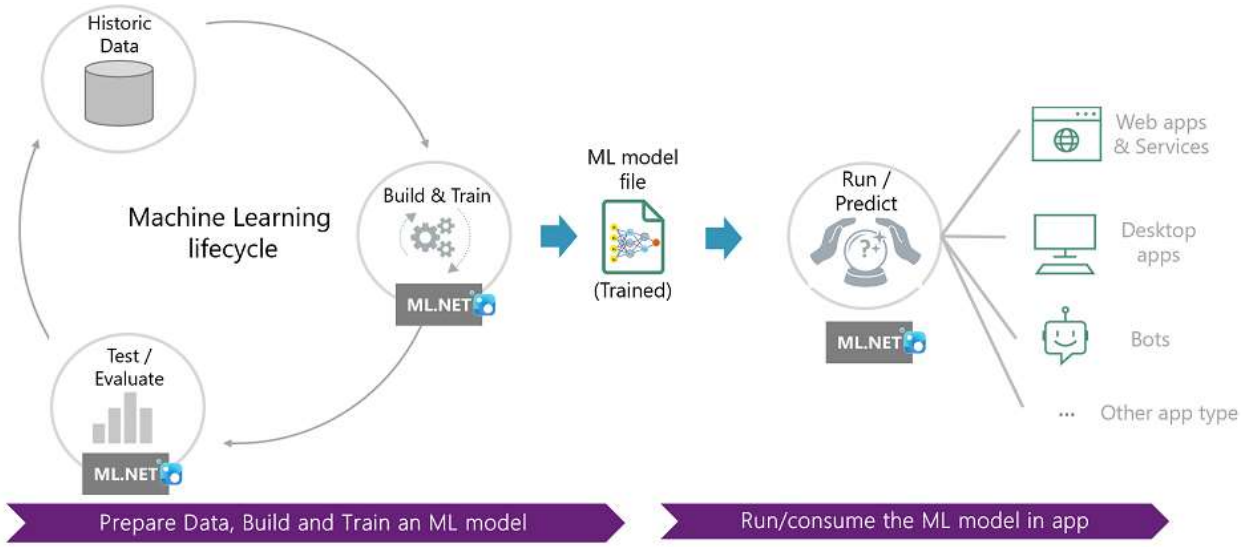

Рис. 2.1. Процес побудови та використання моделі машинного навчання

На початковому етапі використовується історична інформація, яка формує основу для навчання моделі. Ці дані проходять обробку, очищення та перетворення у відповідну форму. На етапі побудови та навчання модель створюється за допомогою обраного алгоритму, після чого тренується на підготовлених даних. Після завершення тренування формується файл із навченою моделлю, який зберігається в структурованому форматі. Наступним кроком є етап тестування й оцінювання моделі – це дозволяє перевірити точність передбачень та інші метрики якості. Після успішного проходження оцінки модель може бути завантажена до будь-якого

програмного середовища – вебзастосунку, настільної програми, чат-бота або іншого типу системи. У такому вигляді вона використовується для прогнозування нових результатів на основі даних, введених кінцевим користувачем.

Можна зазначити, що ML.NET забезпечує повний цикл роботи з машинним навчанням – від навчання до практичного застосування моделі. Його гнучкість, інтеграція в .NET-середовище та підтримка десктопних інтерфейсів роблять цей інструмент надзвичайно корисним для прикладних рішень у сфері оптимізації рекрутингу. Це дозволяє реалізувати аналітику без залежності від зовнішніх сервісів, що важливо як з технічної, так і з організаційної точки зору.

Для обробки історичних даних про кандидатів та працівників використовуються CSV-файли, які імпортуються у форматі таблиць з фіксованою структурою. У таких таблицях зберігається інформація про вік, посаду, рівень освіти, стаж роботи, задоволеність роботою, оцінки ефективності, наявність понаднормових годин тощо. Завдяки ML.NET ці дані перетворюються в числові вектори ознак за допомогою механізмів кодування (наприклад, OneHotEncoding) та об’єднуються у вектор характеристик, що подається на вхід моделі машинного навчання.

Щоб побудувати моделі класифікації у дослідженні використано алгоритм логістичної регресії зі стохастичним градієнтним спуском, який входить до складу засобів бінарної класифікації ML.NET. Цей алгоритм дозволяє визначити ймовірність належності кандидата до певного класу – зокрема, передбачити, чи залишиться він у компанії, чи існує ймовірність звільнення в найближчому періоді.

Графічна частина системи реалізована за допомогою Windows Forms – технології створення візуальних інтерфейсів для настільних додатків у середовищі .NET. Через інтерфейс користувач може вводити дані про кандидата, обирати модель для прогнозування, переглядати результати у вигляді текстового звіту, а також зберігати нові моделі після тренування.

Елементи управління, такі як текстові поля, комбіновані списки та кнопки, дозволяють користувачеві легко взаємодіяти з системою.

Для об’єктивної оцінки якості побудованої моделі використовуються стандартні метрики класифікації:

– точність (Accuracy) – відображає загальну правильність передбачень моделі;

– AUC (Area Under ROC Curve) – показує здатність моделі відрізняти позитивні та негативні класи;

– F1-метрика – гармонійне середнє між точністю та повнотою, особливо цінне при роботі з незбалансованими вибірками.

Процедура збереження моделей у форматі .zip дозволяє повторно використовувати навчений класифікатор без необхідності його перевчання. Моделі зберігаються у локальному каталозі проєкту, а їхні метадані реєструються у внутрішній базі даних за допомогою допоміжного класу ModelsProvider.

Для контролю активності користувача та зберігання історії дій передбачено реєстрацію подій, що дозволяє створювати записи про кожне тренування або прогнозування. Це, у свою чергу, відкриває можливості для подальшого аудиту, удосконалення моделі та виявлення помилкових рішень.

У рамках даного дослідження для створення програмної системи прогнозування в рекрутингу було використано мову програмування C#, яка є основною мовою платформи .NET. Її застосування дало змогу ефективно реалізувати як логіку обробки даних і побудови моделі машинного навчання за допомогою ML.NET, так і зручний графічний інтерфейс на базі Windows Forms. Завдяки інтеграції всіх компонентів у межах єдиного технологічного стеку вдалося досягти узгодженості між обробкою даних, тренуванням моделей, прогнозуванням і взаємодією з користувачем.

Вибір C# як основного інструмента для розробки був зумовлений низкою технічних і практичних переваг, а саме:

– повна сумісність із платформою .NET і ML.NET, що забезпечує просту інтеграцію з інструментами машинного навчання;

– строга типізація, яка зменшує кількість помилок під час компіляції та підвищує стабільність роботи системи;

– об’єктно-орієнтований підхід до програмування, що сприяє модульності, повторному використанню коду та зручності його супроводження;

– середовище розробки Visual Studio, яке включає інструменти для налагодження, тестування, профілювання та розгортання застосунків;

– велика кількість бібліотек та активна спільнота, що забезпечує доступ до готових рішень, документації та технічної підтримки [19];

– зручність роботи з файлами та інтерфейсом користувача, включно з підтримкою WinForms для створення настільних застосунків;

– можливість багатопотокової обробки даних, що дозволяє оптимізувати продуктивність під час роботи з великими обсягами інформації;

– безпека та стабільність виконання, завдяки механізмам керованого коду та автоматичного управління пам’яттю [20].

У тісному взаємозв’язку з мовою C# у процесі розробки системи використовувалося середовище розробки Visual Studio 2022 (VS2022), яке забезпечило зручну та функціонально насичену платформу для реалізації усіх компонентів програмного рішення. VS2022 дозволяє ефективно поєднувати модулі обробки даних, навчання моделей машинного навчання, інтерфейсу користувача та логіки прогнозування в межах одного проєкту. Завдяки широкому набору вбудованих інструментів, розробник має змогу зосередитися на логіці застосунку, не витрачаючи зайвого часу на вирішення інфраструктурних завдань.

Вибір VS2022 був зумовлений рядом практичних переваг, серед яких варто виокремити:

– інтегрована підтримка ML.NET, що дає змогу швидко створювати застосунки з використанням алгоритмів машинного навчання;

– потужні засоби налагодження, включаючи покрокове виконання, перегляд змінних, стеків викликів та інструменти профілювання продуктивності;

– підтримка роботи з базами даних, включно з візуальним редактором запитів, переглядом таблиць і прямим підключенням до джерел даних;

– інтеграція з системами контролю версій (Git, GitHub), що забезпечує керування історією змін і командну розробку;

– підтримка розширень і плагінів, які дозволяють розширювати функціональність середовища під конкретні потреби проєкту;

– висока стабільність і продуктивність, особливо при роботі з великими проєктами або складними багаторівневими структурами.

Таким чином, описані інструменти створюють єдину програмну екосистему для збирання, структуризації, аналізу та інтерпретації даних у задачах оптимізації рекрутингу. Вони забезпечують технічну основу для реалізації предиктивної аналітики на практиці, дозволяючи приймати обґрунтовані рішення щодо відбору персоналу.

2.2 Аналіз існуючих аналогів

У сучасних умовах цифрової трансформації бізнесу зростає попит на інтелектуальні інструменти, здатні автоматизувати та оптимізувати процес рекрутингу. Це зумовило появу широкого спектра програмних рішень, які застосовують методи машинного навчання та аналізу даних для прогнозування поведінки кандидатів, ранжування резюме, автоматичного скринінгу заявок та оцінювання ймовірності успішної адаптації нового співробітника. Такі системи вже активно використовуються провідними компаніями у світі, а також поступово впроваджуються у середній і малий бізнес.

Аналіз існуючих аналогів дозволяє не лише вивчити сильні та слабкі сторони подібних рішень, а й виявити характерні риси, які варто врахувати

при розробці власної системи предиктивної аналітики для рекрутингу. Також це дає змогу сформулювати вимоги до функціональності, інтерфейсу та алгоритмічної частини проєкту, орієнтуючись на кращі практики та досягнення галузі.

IBM Watson Recruitment – це інтелектуальна платформа, створена для автоматизації та вдосконалення процесу найму персоналу за допомогою технологій штучного інтелекту (рис. 2.2) [21]. Основне призначення даного рішення полягає у підвищенні точності та ефективності підбору кандидатів шляхом аналізу великих обсягів даних, виявлення шаблонів успішного працевлаштування та надання HR-фахівцям обґрунтованих рекомендацій щодо прийняття рішень [22].

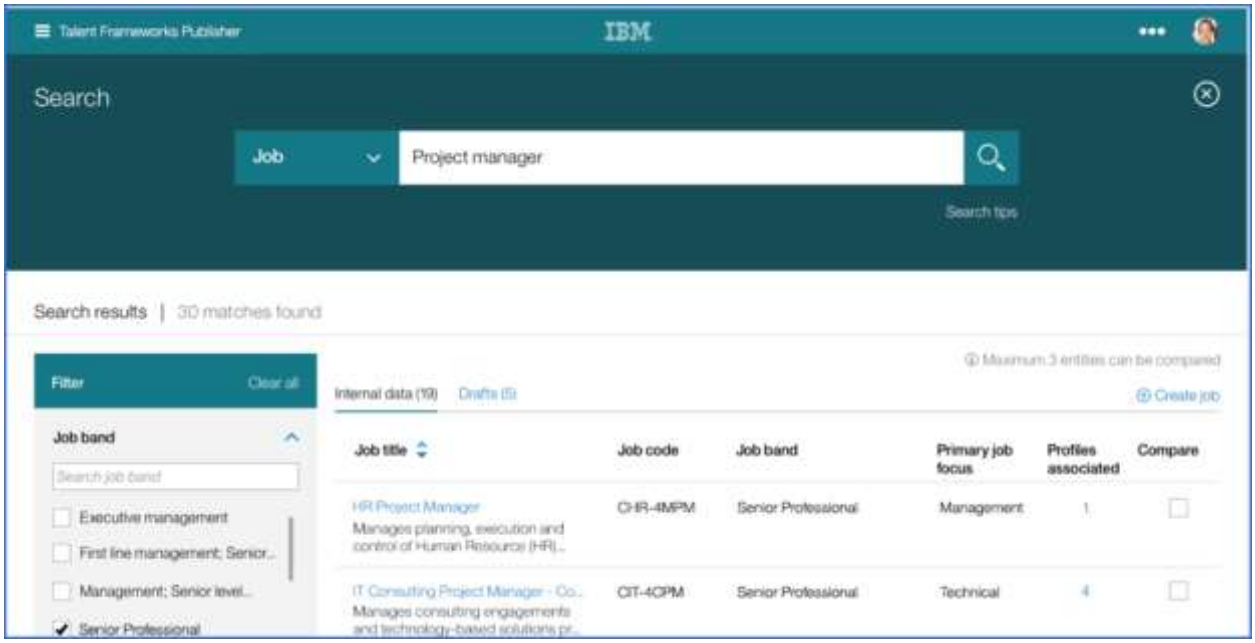

Рис. 2.2. Приклад інтерфейсу системи «IBM Watson Recruitment»

Застосунок використовує потужні алгоритми машинного навчання для аналізу як структурованої інформації з резюме, так і неструктурованих джерел, таких як мотиваційні листи, соціальні профілі та історія взаємодії кандидатів із системою. Watson Recruitment оцінює відповідність кожного кандидата до конкретної вакансії, враховуючи не лише технічні навички, але й контекстні фактори, такі як успішність подібних кандидатів у минулому.

Архітектура рішення базується на хмарних обчисленнях платформи IBM Cloud і включає модулі збору даних, обробки та збагачення інформації, модуль аналітики на основі Watson AI, інтерфейс користувача для рекрутерів та API для інтеграції з іншими HR-системами. Дані з різних джерел надходять у єдину аналітичну платформу, де проходять очищення, нормалізацію та подальший аналіз. Результати прогнозування відображаються у вигляді інтерактивних панелей та рекомендацій, які допомагають приймати кадрові рішення на основі доказів.

Переваги IBM Watson Recruitment:

– інтеграція штучного інтелекту та машинного навчання для глибокого аналізу кандидатів і підвищення точності прогнозування;

– здатність працювати з великими обсягами як структурованих, так і неструктурованих даних, включно з соціальними мережами, резюме та історією взаємодії;

– інтуїтивно зрозумілий інтерфейс та аналітичні панелі, які полегшують прийняття кадрових рішень на основі даних;

– можливість інтеграції з іншими HR-системами через API, що забезпечує високу адаптивність до внутрішньої інфраструктури компанії.

Недоліки IBM Watson Recruitment:

– висока вартість використання, що робить продукт недоступним для малого та середнього бізнесу;

– залежність від хмарної інфраструктури IBM Cloud, що може бути неприйнятним для компаній з жорсткими вимогами до зберігання персональних даних;

– обмежена локалізація – система орієнтована переважно на англомовне середовище, що ускладнює її використання в інших мовних контекстах;

– складність початкової конфігурації та навчання персоналу, що вимагає часу та залучення технічних фахівців.

Отже, IBM Watson Recruitment є потужним інструментом для корпоративного рекрутингу, що поєднує передові аналітичні технології з практичними потребами HR-відділів. Попри високу ефективність і точність прогнозів, його використання виправдане переважно у великих організаціях із достатніми ресурсами для адаптації системи до власної інфраструктури.

SAP SuccessFactors – це потужна хмарна платформа управління людським капіталом, розроблена для автоматизації всіх етапів життєвого циклу працівника – від рекрутингу до звільнення (рис. 2.3) [23]. Основне призначення застосунку полягає в оптимізації процесів підбору, адаптації, розвитку, мотивації та оцінювання персоналу шляхом поєднання інструментів HR-аналітики, предиктивного моделювання та стратегічного планування [24].

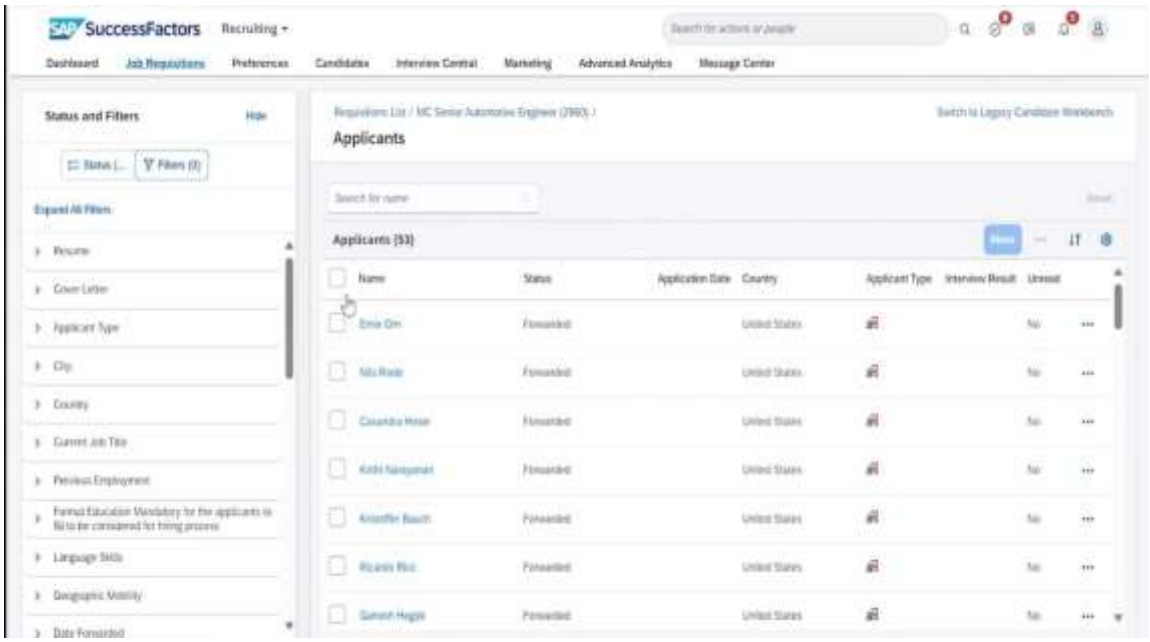

Рис. 2.3. Приклад інтерфейсу системи «SAP SuccessFactors»

Система використовує модульну архітектуру, де кожен модуль відповідає за окрему бізнес-функцію: рекрутинг, управління талантами, навчання, компенсації, продуктивність тощо. Архітектура базується на хмарній платформі SAP Business Technology Platform (BTP) і включає сховище даних, засоби інтеграції, аналітичний модуль, засоби конфігурації

бізнес-процесів та інтерфейс користувача. Усі компоненти об'єднані через централізовану модель даних, що дозволяє забезпечити безперервність інформаційних потоків і цілісність профілю працівника. Інтерфейс представлений у вигляді вебпанелі з адаптивним дизайном, а для аналітики використовуються вбудовані засоби прогнозування на основі машинного навчання та статистичних моделей.

Переваги SAP SuccessFactors:

– комплексний підхід до управління персоналом, що охоплює весь життєвий цикл працівника – від найму до звільнення;

– модульна архітектура, яка дозволяє масштабувати систему відповідно до потреб компанії та поступово додавати нові функції;

– потужні аналітичні можливості, включно з інструментами предиктивної аналітики та KPI-відстеженням у реальному часі;

– інтеграція з іншими рішеннями SAP, що забезпечує єдину екосистему для управління бізнес-процесами.

Недоліки SAP SuccessFactors:

– висока складність налаштування, що потребує залучення консультантів або фахівців з досвідом роботи з SAP;

– значна вартість ліцензування та впровадження, яка може бути бар’єром для малих організацій;

– інтерфейс потребує адаптації для користувачів без технічного досвіду, що ускладнює початкове навчання персоналу;

– обмежена гнучкість при локалізації, особливо для нестандартних організаційних структур або специфічних вимог на національному рівні.

Отже, SAP SuccessFactors – це високофункціональна система для управління людським капіталом, яка особливо ефективна у великих компаніях із комплексною структурою HR-процесів. Вона поєднує аналітичну глибину, інтегрованість та стратегічний підхід до розвитку персоналу, хоча вимагає серйозних ресурсів для повноцінного впровадження та налаштування.

Zoho Recruit – це хмарне програмне рішення, призначене для автоматизації процесів підбору персоналу як у кадрових агентствах, так і у внутрішніх HR-відділах компаній (рис. 2.4) [25]. Основне завдання застосунку полягає у спрощенні та прискоренні всіх етапів рекрутингу – від публікації вакансій до прийняття остаточного рішення щодо найму кандидата. Система забезпечує централізоване керування вакансіями, резюме, інтерв'ю, комунікацією з претендентами та аналітичним супроводом процесу [26].

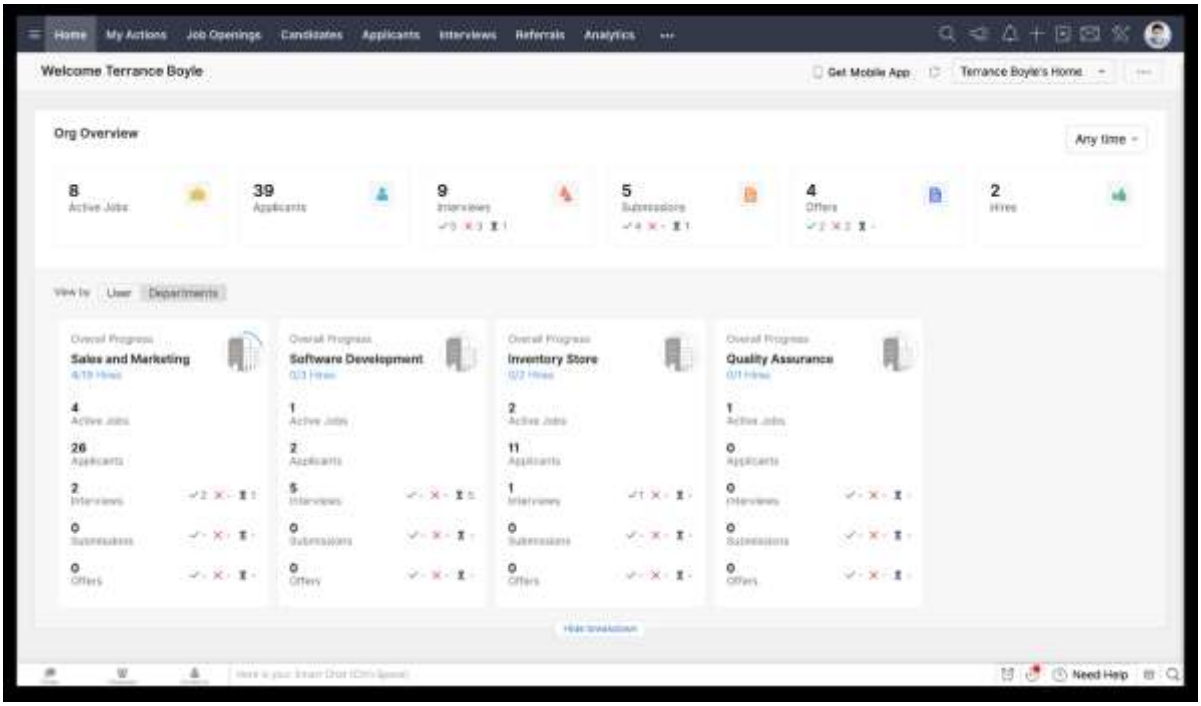

Рис. 2.4. Приклад інтерфейсу системи «Zoho Recruit»

Архітектура Zoho Recruit побудована на базі хмарної інфраструктури Zoho Cloud, що забезпечує доступність застосунку через браузер і мобільні платформи. Вона включає модулі для управління вакансіями, базою кандидатів, аналітикою, автоматизацією робочих процесів, а також засоби інтеграції з електронною поштою, сайтами пошуку роботи (наприклад, LinkedIn, Indeed), календарями та сторонніми сервісами через API. Дані з різних каналів збираються в єдиній базі, де обробляються за допомогою вбудованих фільтрів, правил автоматизації та аналітичних інструментів.

Застосунок також підтримує інструменти на базі штучного інтелекту для автоматичного ранжування кандидатів та аналізу відповідності їхніх навичок до вимог вакансії.

Переваги Zoho Recruit:

– доступність у хмарному середовищі з підтримкою веб- та мобільного доступу, що забезпечує зручну роботу з будь-якого місця;

– гнучке налаштування під потреби бізнесу, з можливістю адаптувати інтерфейс, автоматизацію та логіку процесів;

– інтеграція з популярними платформами пошуку роботи та засобами комунікації, включно з LinkedIn, Indeed, Gmail, Outlook тощо;

– наявність інструментів штучного інтелекту, які допомагають в автоматичному ранжуванні кандидатів і прискоренні прийняття рішень.

Недоліки Zoho Recruit:

– обмежена функціональність у безкоштовній версії, що змушує користувача переходити на платні тарифи для повного доступу;

– інтерфейс може здаватися перевантаженим, особливо для нових користувачів без досвіду роботи з подібними системами;

– часткова підтримка локалізації, яка може створювати труднощі для користувачів, що працюють поза англомовним середовищем;

– деякі можливості аналітики обмежені в стандартних тарифних планах, що потребує додаткових витрат на розширення функцій.

Отже, Zoho Recruit є функціонально багатим і доступним інструментом для автоматизації рекрутингу, який особливо підходить малим і середнім компаніям, а також кадровим агентствам. Завдяки підтримці інтеграцій, AI-механізмам і модульності, система забезпечує ефективну організацію процесу найму, хоча для повного використання її потенціалу можуть знадобитися розширені тарифні плани та певне налаштування.

Враховуючи аналіз існуючих програмних рішень, можна зробити висновок, що більшість з них орієнтовані на великі компанії та потребують значних фінансових і технічних ресурсів для налаштування та повноцінного

використання. Багато систем мають надлишкову функціональність або складні інтерфейси, що ускладнює їх адаптацію в середовищі малого та середнього бізнесу. Окрім того, обмеження безкоштовних версій суттєво знижують доступність таких рішень для широкого кола користувачів. У зв’язку з цим актуальним є створення нового програмного продукту з акцентом на використання штучного інтелекту, мінімалізм у функціоналі, легкість у впровадженні та можливість повноцінного використання у безкоштовній версії. Таке рішення дозволить зробити предиктивну аналітику у рекрутингу доступною для значно ширшої аудиторії.

2.3 Переваги та недоліки існуючих аналогів

Аналіз сучасних програмних рішень, що реалізують предиктивну аналітику у сфері рекрутингу, дозволяє виявити низку важливих закономірностей щодо їхньої функціональності, доступності та технологічної складності. Такі системи, як IBM Watson Recruitment, SAP SuccessFactors, Zoho Recruit та інші, демонструють високий рівень автоматизації процесів найму та широкі можливості інтеграції, однак здебільшого орієнтовані на великі корпорації. Висока вартість впровадження, складність конфігурації та потреба в спеціалізованому технічному супроводі робить їх малопридатними для малого та середнього бізнесу.

Деякі з аналізованих систем обмежені в локалізації та не передбачають повноцінної підтримки української мови, що ускладнює їхнє використання в реаліях національного ринку. Крім того, у більшості випадків безкоштовні версії або відсутні взагалі, або мають суттєво урізаний функціонал. Значна частина систем працює лише в хмарному середовищі, що не дозволяє повністю контролювати обробку персональних даних, що є критично важливим для компаній, які дотримуються вимог інформаційної безпеки.

Розроблена в рамках даного дослідження система вирізняється локальним характером роботи, повною підтримкою української мови,

простим інтерфейсом та відкритістю до налаштувань. Завдяки використанню ML.NET і мови C# створено легке у впровадженні рішення, яке не потребує потужних обчислювальних ресурсів і доступне для безкоштовного використання. Система забезпечує достатньо високі показники точності при тестуванні на реальних даних, що підтверджує її практичну придатність.

У табл. 2.1 представлено порівняльну таблицю, яка відображає ключові характеристики аналізованих систем і розробленого рішення.

Таблиця 2.1

Порівняльна характеристика програмних рішень

Характеристика	IBM Watson Recruitment	SAP SuccessFactors	Zoho Recruit	Розроблена система
Тип доступу	Комерційна (SaaS)	Комерційна (SaaS)	SaaS / Хмара	Локальна, безкоштовна
Простота інтерфейсу	Складний	Середній	Середній	Простий
Українізований інтерфейс	Обмежено	Частково	Частково	Повністю
Локалізація під український ринок	Обмежена	Часткова	Часткова	Повна
Безкоштовна версія	Відсутня	Відсутня	Обмежена	Так
Точність (Accuracy)	>90%	~85–88%	~80%	84,28%
AUC	Високий	Високий	Середній	63,39%
Потреба в технічному персоналі	Висока	Висока	Середня	Мінімальна
Інтеграція в існуючу IT-інфраструктуру	Складна	Середня	Проста	Гнучка

Як видно з таблиці, розроблена система має низку переваг, які роблять її доцільною альтернативою для цільових користувачів – зокрема, компаній, що потребують функціонального, україномовного, доступного та легкого у налаштуванні інструменту прогнозування. Простота використання, відсутність комерційної ліцензії та задовільні показники якості моделі

забезпечують конкурентоспроможність розробленого рішення на фоні складних корпоративних платформ.

РОЗДІЛ 3
РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Опис функціональних можливостей застосунку

Розроблений програмний застосунок призначений для підтримки процесу рекрутингу шляхом використання методів предиктивної аналітики з метою оцінювання ймовірності звільнення кандидата після його працевлаштування. Основна ідея полягає в створенні інструменту, що дає змогу кадровим спеціалістам, спираючись на аналіз історичних даних, оперативно й обґрунтовано ухвалювати рішення про доцільність найму конкретного претендента.

Програма реалізована як настільний застосунок із графічним інтерфейсом користувача на базі Windows Forms та побудована із використанням бібліотеки ML.NET, яка забезпечує інтеграцію алгоритмів машинного навчання у прикладне середовище C#.

Функціональні можливості застосунку охоплюють повний цикл роботи з моделлю машинного навчання: від завантаження та попередньої обробки вхідних даних до навчання моделі, її збереження, тестування та перегляду результатів. Окрім цього, система дозволяє здійснювати прогнозування на основі індивідуально введених характеристик кандидата, керувати декількома моделями, вести журнал дій користувача, переглядати історію прогнозів і формувати зрозумілі текстові звіти з поясненнями, адаптованими для користувачів без технічної підготовки. Інтерфейс програми побудований з орієнтацією на простоту, зручність та доступність, що сприяє легкому засвоєнню навіть нефахівцями у сфері аналітики.

Для кращого розуміння можливостей системи в таблиці 3.1 наведено сценарії використання, які відображають ключові функції застосунку та практичні ситуації, в яких вони застосовуються.

Таблиця 3.1

Опис варіантів використання системи

№	Ім’я	Опис
UC1	Реєстрація користувача	Дозволяє новим користувачам створити обліковий запис для доступу до функцій системи
UC2	Автентифікація користувача	Забезпечує перевірку облікових даних та вхід до системи зареєстрованого користувача
UC3	Перегляд користувачів	Дозволяє адміністраторам переглядати список усіх зареєстрованих користувачів
UC4	Додавання користувача	Надає адміністраторам можливість додавати нових користувачів до бази даних системи
UC5	Редагування даних користувача	Дозволяє змінювати персональні або рольові дані користувачів у системі
UC6	Перегляд доступних моделей	Дозволяє користувачам переглядати перелік створених моделей машинного навчання
UC7	Створення нової моделі	Забезпечує запуск процесу навчання нової моделі на основі обраного набору даних
UC8	Видалення моделі	Дозволяє видаляти моделі машинного навчання, що є застарілими або неефективними
UC9	Прогнозування кадрових рішень	Надає можливість здійснити оцінку ймовірності звільнення кандидата на основі введених даних
UC10	Перегляд історії дій користувачів	Дозволяє відстежувати події, що відбулися у системі, з метою аудиту та аналізу активності

Розроблений застосунок SmartRecruit підтримує дві основні ролі: адміністратор і користувач, кожна з яких має свій набір доступних функцій. На основі функціональних вимог було побудовано діаграму варіантів використання, яка відображає взаємодію між користувачами системи та її функціональністю. На рис. 3.1 наведено діаграму варіантів використання, яка демонструє, які саме дії можуть виконуватись адміністраторами та звичайними користувачами, а також логічні зв’язки між окремими сценаріями.

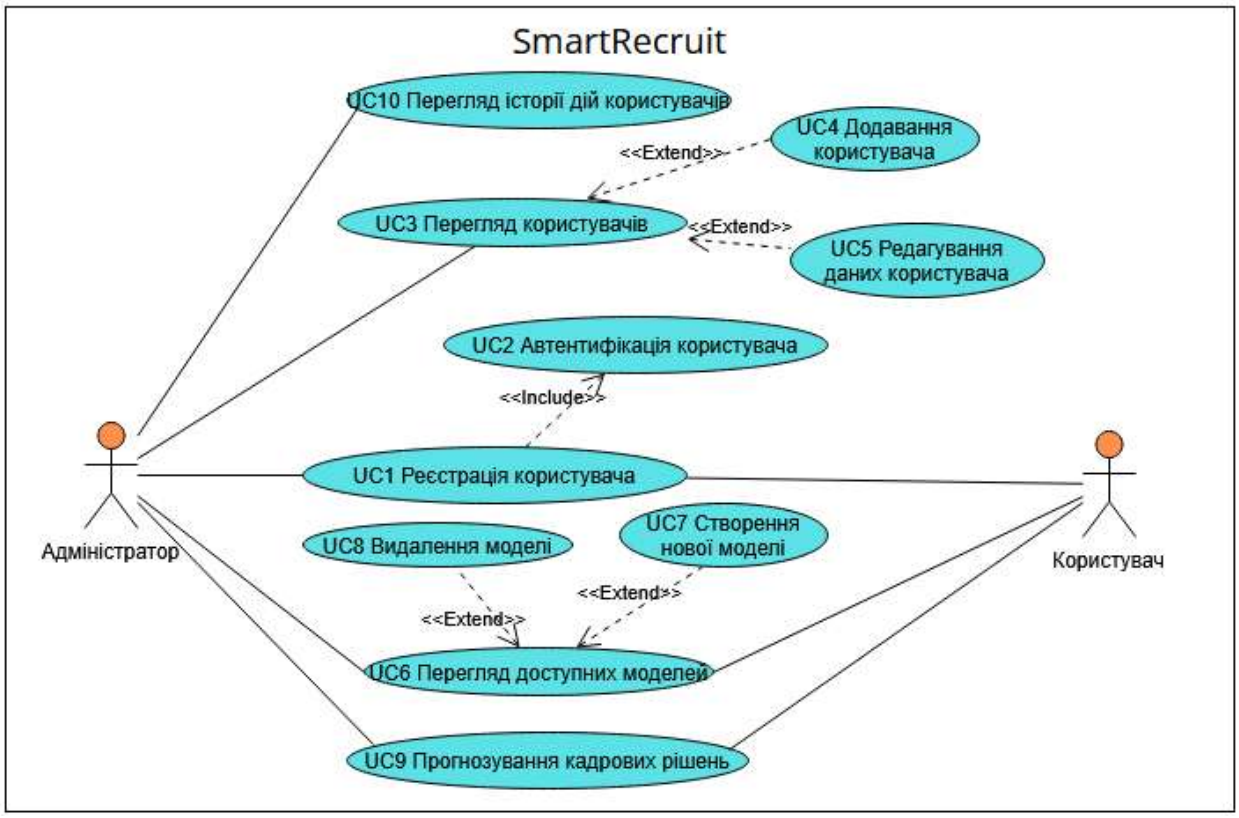

Рис. 3.1. Діаграма варіантів використання системи

Дана діаграма дає цілісне уявлення про структуру поведінки системи з погляду її користувачів. Вона є основою для подальшого проєктування архітектури та інтерфейсів, а також відіграє важливу роль у тестуванні й валідації функціональності.

У процесі побудови інтелектуального модуля для прогнозування плинності кадрів важливим етапом є формування алгоритму тренування моделей машинного навчання. Такий алгоритм повинен враховувати не лише послідовність обробки вхідних даних, а й механізми перевірки їхньої коректності, оцінювання якості моделі та подальшого збереження отриманих результатів у системі. Чітке визначення етапів дозволяє стандартизувати підхід до навчання, забезпечити контроль за помилками, а також зробити процес створення моделі прозорим і керованим.

На рис. 3.2 представлено алгоритм тренування моделей для оцінки ймовірності звільнення кандидата, який реалізовано в межах функціональних можливостей системи SmartRecruit.

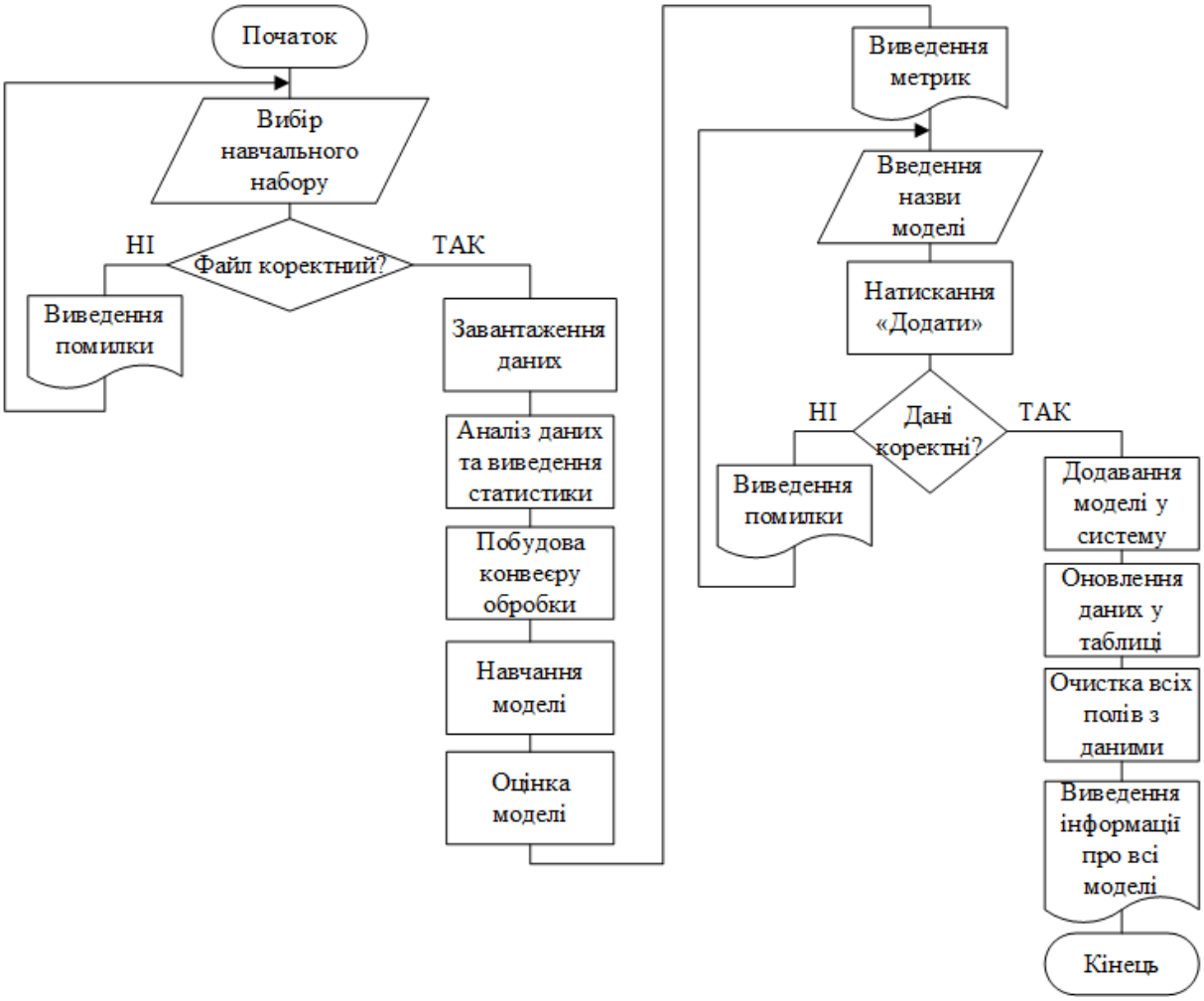

Рис. 3.2. Алгоритм тренування моделей для оцінки ймовірності звільнення кандидата

Алгоритм починається з вибору навчального набору даних, після чого система перевіряє коректність обраного файлу. Якщо дані некоректні, користувач отримує повідомлення про помилку й процес переривається. У разі, якщо файл відповідає вимогам, система переходить до завантаження даних та побудови конвеєра обробки. Наступним етапом є нормалізація даних, яка забезпечує узгодженість масштабів числових ознак для подальшої обробки. Після цього виконується навчання моделі та оцінка її якості на тестовому наборі, з подальшим виведенням метрик.

Паралельно реалізовано механізм додавання моделі до системи. Користувач вводить назву моделі, натискає кнопку «Додати», після чого система перевіряє правильність заповнення всіх полів. У разі помилки

виводиться повідомлення про її суть. Якщо дані вірні, модель додається до списку доступних, таблиця оновлюється, поля очищуються, а користувач отримує вивід інформації про всі наявні моделі у системі. Завершується процес формальним виходом із процедури, що означає завершення сеансу тренування та реєстрації моделі.

Наступним ключовим етапом функціонування системи SmartRecruit є процес здійснення прогнозування на основі попередньо навченої моделі. Алгоритм повинен забезпечити не лише коректне підключення моделі, а й стабільну обробку введених користувачем даних, що використовуються для генерації результату. Важливими складовими цього процесу є перевірка вхідних значень, створення механізму прогнозування (двигуна), виведення результату та оновлення відповідних елементів інтерфейсу. На рис. 3.3 наведено алгоритм виконання прогнозування ймовірності звільнення кандидата на основі введених даних.

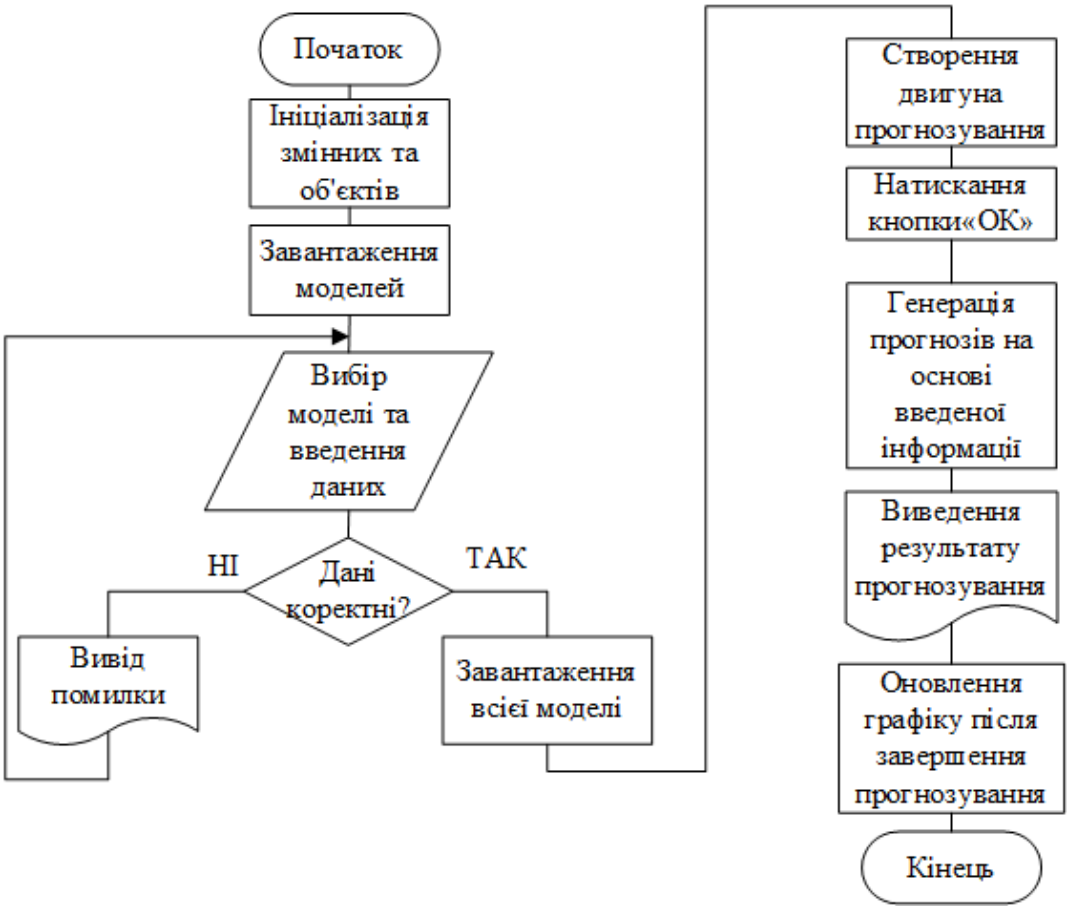

Рис. 3.3. Алгоритм прогнозування ймовірності звільнення кандидата

Алгоритм починається з ініціалізації змінних і об’єктів, які потрібні для роботи інтерфейсу та логіки обчислень. Далі відбувається завантаження переліку доступних моделей, після чого користувач обирає одну з них та вводить необхідні дані для прогнозування. Система перевіряє коректність введених значень. Якщо дані є некоректними, виводиться повідомлення про помилку, і процес призупиняється. Якщо ж дані відповідають вимогам, система завантажує повну модель і переходить до створення спеціального об’єкта – двигуна прогнозування.

Після натискання кнопки «ОК» активується прогнозування, у ході якого здійснюється передача введених значень до моделі машинного навчання. На основі цих даних генерується прогноз, результат якого відображається у вікні програми у зручному для користувача форматі. Після завершення обчислень відбувається оновлення графіків і візуальних елементів, що дозволяє швидко інтерпретувати отриманий результат. Завершується алгоритм переходом до фінального стану, що свідчить про успішне виконання процедури прогнозування.

3.2 Архітектура застосунку та його компоненти

Проєктування архітектури програмного застосунку є одним із найважливіших етапів у розробці інформаційних систем, оскільки саме архітектурна структура визначає логіку взаємодії між компонентами, масштабованість, зручність супроводження та надійність функціонування всієї системи. З урахуванням особливостей поставленої задачі – побудови системи для прогнозування кадрових рішень на основі предиктивної аналітики – було прийнято рішення реалізувати застосунок на основі трьохрівневої архітектури (three-tier architecture).

Такий підхід дозволяє чітко розмежувати функціональні обов’язки між основними частинами системи: рівнем представлення (Presentation Layer), рівнем логіки застосунку (Business Logic Layer) та рівнем доступу до даних

(Data Access Layer). Ізоляція користувацького інтерфейсу від обчислювальної логіки дає змогу розвивати або змінювати інтерфейс без необхідності втручання в алгоритми обробки даних.

Логіка навчання моделей, їх оцінювання, прогнозування та керування зберігається в окремому шарі, що підвищує гнучкість і дозволяє багаторазово використовувати ключові методи в різних сценаріях. Така структурованість полегшує тестування та модифікацію алгоритмів, не порушуючи роботу всієї системи.

Виділення окремого рівня доступу до даних спрощує централізоване керування джерелами інформації, дозволяє зручно працювати з файлами CSV, забезпечує підготовку до переходу на інші типи сховищ та баз даних. Це особливо важливо в умовах подальшого розвитку системи або розширення її функціональності.

Таким чином, обрана архітектурна модель забезпечує логічну цілісність, масштабованість і простоту супроводу, що відповідає вимогам до сучасного програмного забезпечення у сфері аналітики рекрутингових процесів.

Для реалізації доступу до даних у системі SmartRecruit було спроєктовано окремий рівень, який відповідає за взаємодію з інформаційними структурами, що використовуються для збереження, обробки та передачі даних між компонентами застосунку. Такий підхід дозволяє централізовано керувати всіма даними, ізолювати логіку зберігання від бізнес-логіки та забезпечити гнучкість у розширенні системи. Класи рівня даних охоплюють як структури для зберігання користувацької інформації та прогнозів, так і класи-постачальники, які реалізують відповідні CRUD-операції.

На рис. 3.4 наведено діаграму класів рівня даних, яка демонструє структуру об’єктів і логіку взаємозв’язків між ними у програмному застосунку.

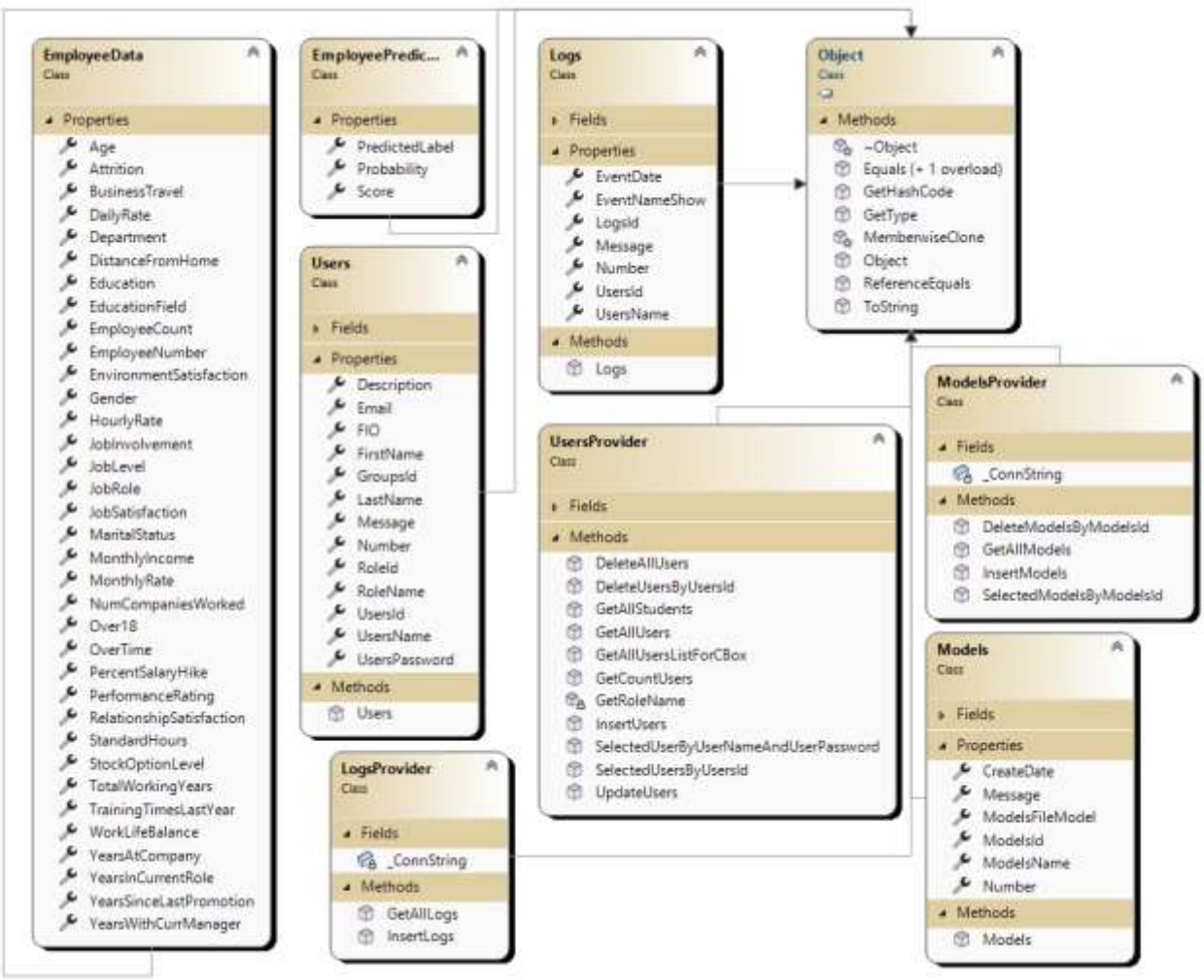

Рис. 3.4. Діаграма класів рівня даних

Діаграма класів рівня даних складається з:

– класу UsersProvider, який відповідає за доступ до даних про користувачів. Він містить методи для додавання, оновлення, видалення та отримання інформації про всіх зареєстрованих користувачів, а також для автентифікації, фільтрації користувачів за ролями та авторизації за іменем і паролем;

– класу LogsProvider, що призначений для збереження та отримання даних про події в системі. Клас забезпечує логування дій користувачів, а також зберігає службову інформацію, пов’язану з активністю у застосунку;

– класу ModelsProvider, який реалізує доступ до даних про моделі машинного навчання. Він дозволяє додавати нові моделі, видаляти їх, а також здійснювати пошук моделей за унікальним ідентифікатором;

– класу Users, який містить основні властивості користувача – ім’я, прізвище, логін, пароль, роль та опис. Його екземпляри відображають кожного зареєстрованого користувача системи;

– класу Logs, що реалізує структуру зберігання даних про події, зокрема повідомлення, дату події, тип події, ім’я користувача, який її виконав, та інші деталі;

– класу Models, який описує структуру моделі: дату створення, назву файлу, шлях до збереженого файлу, назву моделі, її ідентифікаційний номер. Цей клас забезпечує зберігання метаданих про моделі, що використовуються для прогнозування;

– класу EmployeeData, який являє собою структуру для опису повного набору вхідних характеристик працівника, необхідних для навчання та прогнозування: демографічні, професійні, поведінкові та інші ознаки;

– класу EmployeePrediction, що використовується для збереження результатів прогнозування. Він включає бінарну мітку (PredictedLabel), імовірність (Probability) та оціночний бал (Score);

– класу Object, який є базовим класом системи та надає базові методи (порівняння, клонування, перетворення в текст), успадковані іншими класами.

Ця структура рівня даних забезпечує повноцінне управління інформаційними потоками в системі, гарантує збереження цілісності даних та дозволяє легко масштабувати систему в разі додавання нових функціональних можливостей.

Після визначення структури даних та реалізації бізнес-логіки наступним важливим етапом стала побудова рівня користувацького інтерфейсу, який забезпечує безпосередню взаємодію користувача із системою. Усі вікна, форми та функціональні елементи були реалізовані з використанням Windows Forms – платформи для розробки десктопних інтерфейсів у середовищі .NET. Кожен інтерфейсний клас містить у собі набір методів, пов’язаних із подіями, керуванням введенням даних,

завантаженням вмісту, викликом бізнес-логіки та перевіркою правильності введення.

На рис. 3.5 зображено діаграму класів рівня користувацького інтерфейсу, що демонструє структуру та функціональне призначення основних форм застосунку SmartRecruit.

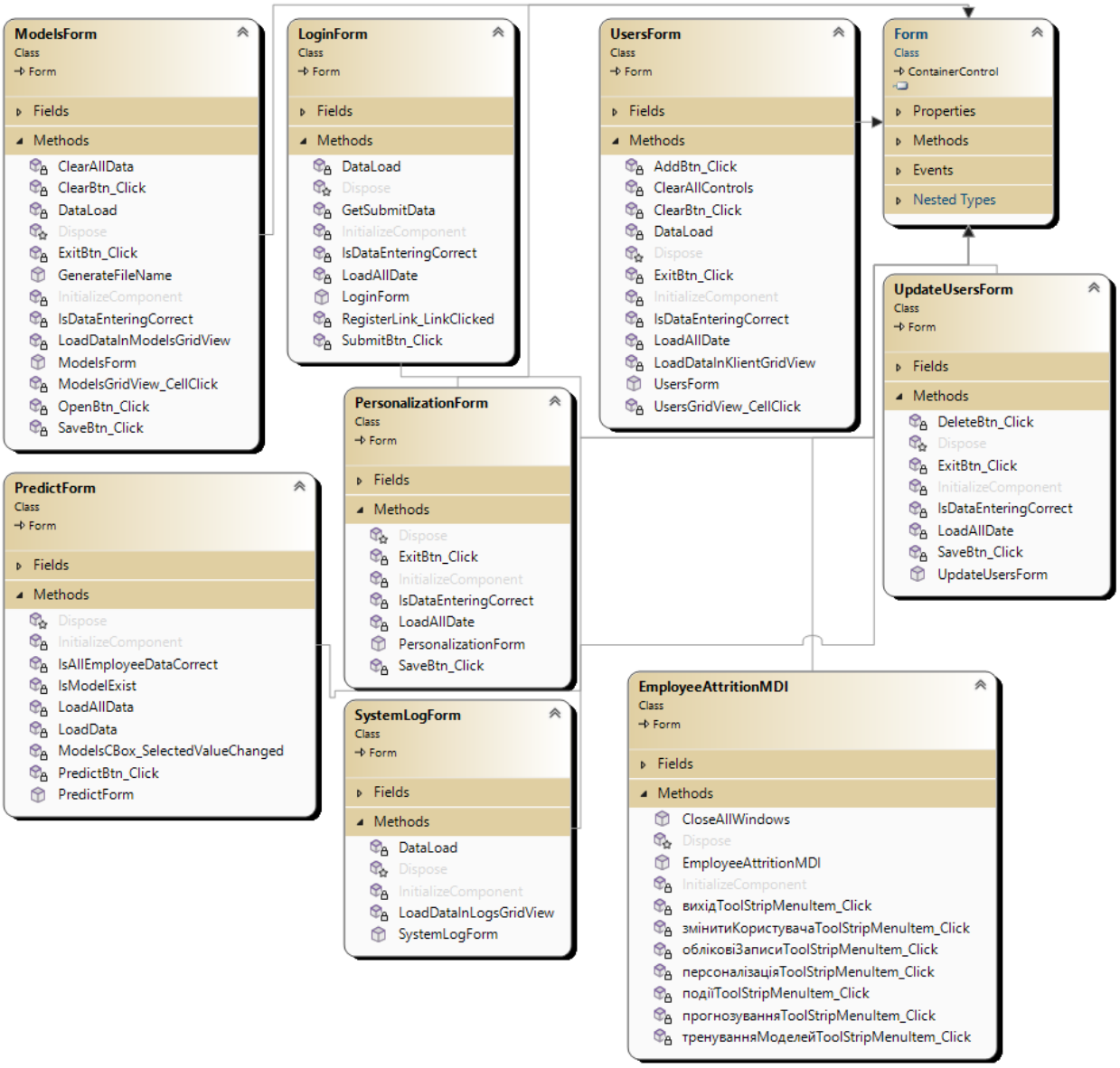

Рис. 3.5. Діаграма класів користувацького інтерфейсу

Діаграма класів рівня користувацького інтерфейсу включає:

– клас EmployeeAttritionMDI, який є головною контейнерною формою застосунку. Він реалізує основне меню програми, через яке

користувач може переходити до різних підрозділів: тренування моделей, прогнозування, обліку користувачів, перегляду логів, персоналізації інтерфейсу тощо. Містить методи обробки подій натискання на пункти меню;

– клас LoginForm, який відповідає за автентифікацію користувачів. Він містить методи для завантаження даних, перевірки введених облікових даних, очищення полів і обробки подій натискання кнопок входу та реєстрації;

– клас UsersForm використовується для перегляду та адміністрування зареєстрованих користувачів. Реалізує функції додавання нового користувача, редагування, фільтрації, а також відображення даних у вигляді таблиці;

– клас UpdateUsersForm призначений для редагування даних користувача. Містить методи перевірки правильності введення, збереження змін, скасування редагування та видалення користувача з бази;

– клас ModelsForm реалізує інтерфейс керування моделями машинного навчання. Дає змогу завантажувати навчальні дані, очищувати поля, зберігати модель, переглядати список моделей у таблиці, а також відкривати моделі для перегляду або редагування;

– клас PredictForm відповідає за процес прогнозування. Він містить методи для перевірки коректності введених даних, вибору моделі для прогнозування, ініціалізації, обчислення та виведення результату;

– клас SystemLogForm забезпечує відображення системних подій і дій користувачів у вигляді таблиці журналу. Містить методи для завантаження та виведення логів системи;

– клас PersonalizationForm надає можливість змінювати параметри персоналізації, такі як мова інтерфейсу або кольорове оформлення. Містить методи перевірки введення, збереження налаштувань та завантаження попередніх значень.

Ці класи формують логічну структуру представлення користувацького рівня та забезпечують взаємодію з усіма функціональними модулями

системи. Такий підхід дозволяє не лише ефективно організувати інтерфейс, але й підтримувати принципи модульності, повторного використання коду та зрозумілості для кінцевого користувача.

У розробці програмної системи SmartRecruit важливу роль відіграє рівень бізнес-логіки, який реалізує основні прикладні правила, перевірки, обробку вхідних даних, а також забезпечує підтримку роботи інтерфейсу та взаємодію з даними. На цьому рівні зосереджено функціональність, що не належить до візуального представлення чи збереження інформації, але є критичною для правильної поведінки системи та контролю введення.

На рис. 3.6 зображено діаграму класів рівня бізнес-логіки системи, яка демонструє структуру допоміжних класів, що реалізують перевірку, шифрування, стандартизацію даних і підтримку повідомлень у межах системи.

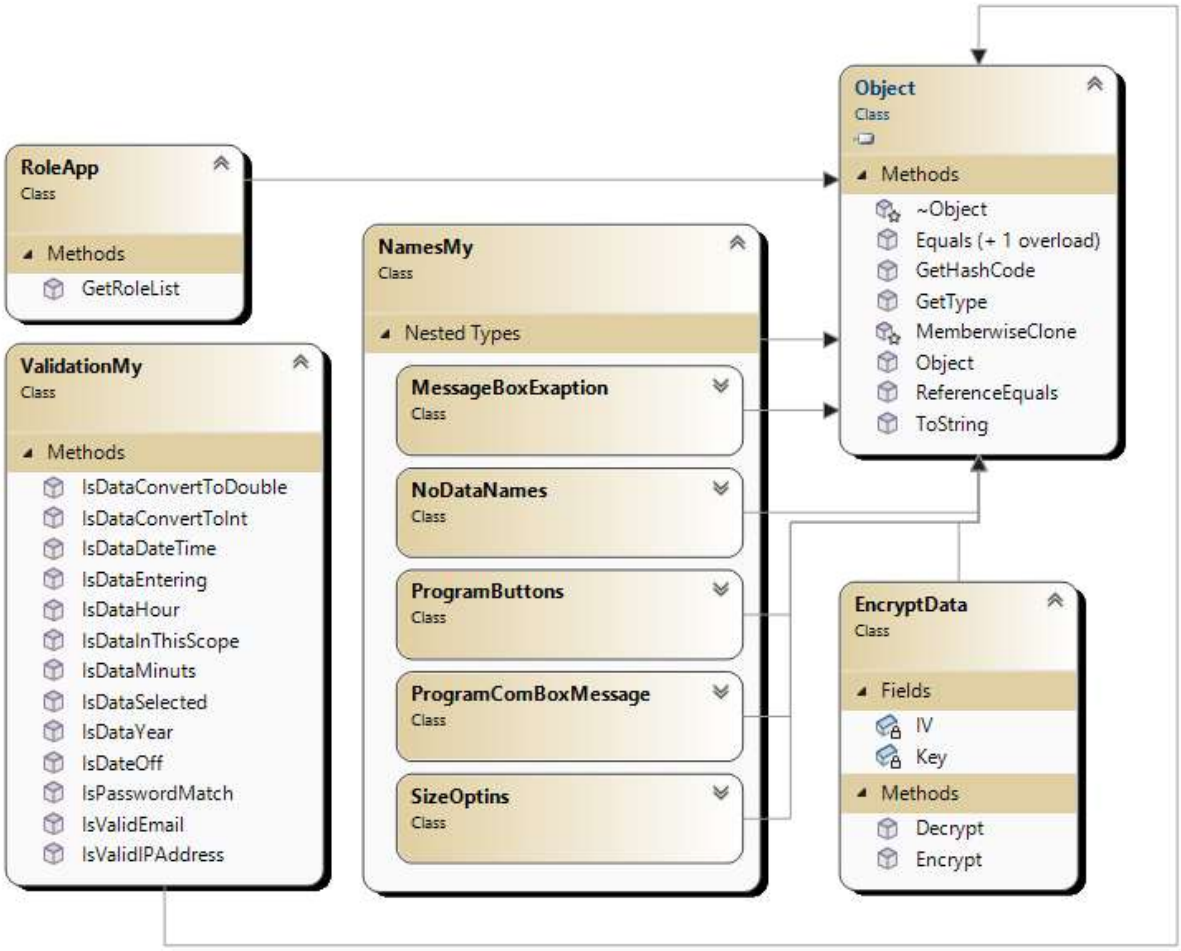

Рис. 3.6. Діаграма класів рівня бізнес-логіки

Діаграма даного рівня містить наступні класи:

– клас ValidationMy, який виконує функції перевірки та валідації вхідних даних. Він містить методи для перевірки правильності введення чисел, дати, часу, електронної пошти, пароля та ін. Завдяки цьому класу забезпечується попередній контроль якості введених значень до обробки та збереження, що знижує ризик виникнення помилок у роботі системи;

– клас RoleApp, який містить метод GetRoleList, що забезпечує отримання переліку доступних ролей користувачів у системі. Клас використовується під час створення або редагування користувачів, а також при автентифікації;

– клас EncryptData, який реалізує механізми симетричного шифрування. Він містить поля Key і IV (ініціалізуючий вектор), а також методи Encrypt і Decrypt, які використовуються для захисту чутливих даних – зокрема паролів користувачів під час збереження у базі даних;

– клас NamesMy, що виконує роль контейнера для вкладених типів даних, які містять константи або службові ідентифікатори, що використовуються в інтерфейсі та бізнес-логіці. Включає вкладені класи: повідомлення про помилки, мітки про відсутність даних, назви елементів керування (кнопок), службові повідомлення, пов’язані з елементами списків, параметри розмірів інтерфейсних елементів або форм.

Цей рівень бізнес-логіки забезпечує стабільність і цілісність усієї системи, концентруючи в собі повторно використовувані перевірки, правила обробки даних та допоміжні функції. Така структура дає змогу зберігати чіткий поділ обов’язків між частинами системи, знижуючи складність її підтримки та розширення.

Після визначення архітектури та структури системи, а також реалізації взаємодії між рівнями, наступним етапом стало поетапне проєктування та розробка функціональних компонентів. Кожен компонент виконує окрему логічну функцію в межах загальної архітектури – від обробки введення

користувача, до навчання моделі машинного навчання та формування прогнозу. Особлива увага була зосереджена на модулі навчання, який є ключовим для реалізації предиктивної аналітики у системі SmartRecruit, оскільки саме він забезпечує побудову моделі прогнозування на основі історичних даних про працівників.

Компонент навчання моделей починається з ініціалізації процесу вибору файлу з навчальними даними. Для цього в застосунку реалізовано метод, що викликається при натисканні кнопки відкриття файлу. У межах цього методу створюється діалогове вікно, яке дозволяє користувачу обрати файл формату .csv зі структурованими даними. Для зручності передбачено фільтрацію за типами файлів: першочергово відображаються текстові файли CSV, але користувач також має можливість переглядати всі доступні файли. Встановлено параметр повернення до попереднього каталогу після завершення дії, що полегшує повторне використання шляху.

```
private void OpenBtn_Click(object sender, EventArgs e) {
  // Створення діалогового вікна для відкриття файлу
  OpenFileDialog openFileDialog = new OpenFileDialog {
    Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*",
    FilterIndex = 2,
    RestoreDirectory = true
  };
  if (openFileDialog.ShowDialog() == DialogResult.OK) {
    _Path = openFileDialog.FileName;
    FileNameTBox.Text = openFileDialog.FileName;
```

Рис. 3.7. Відкриття файлу із тренувальними даними

Якщо користувач обирає файл і підтверджує вибір, система зчитує шлях до файлу і зберігає його у змінну, призначену для подальшої обробки. Паралельно текстове поле у формі заповнюється повним шляхом до обраного файлу, щоб користувач мав змогу переконатися у правильності вибору. Цей етап є критично важливим, оскільки саме з цього моменту розпочинається обробка вхідних даних, які надалі будуть передані до алгоритмів попередньої обробки, нормалізації, навчання та оцінювання моделі. Такий підхід дозволяє

мінімізувати помилки на ранніх етапах і зробити процес роботи з даними інтуїтивно зрозумілим.

Наступним кроком після вибору та підтвердження файлу навчальних даних є ініціалізація середовища машинного навчання, що виконується за допомогою створення об'єкта типу MLContext (рис. 3.8). Цей об’єкт є базовим компонентом бібліотеки ML.NET і забезпечує доступ до всіх основних функцій, пов’язаних із побудовою та виконанням моделей машинного навчання. Для забезпечення повторюваності результатів під час експериментів у конструкторі MLContext задається фіксоване зерно випадковості (seed), що гарантує однакові результати при кожному запуску.

```
mlContext = new MLContext(seed: 123);
// Завантаження даних
try {
  // Тепер CSV має в другій колонці (Attrition) значення 1 або 0
  dataView = mlContext.Data.LoadFromTextFile<EmployeeData>(
      _Path,
      hasHeader: true,
      separatorChar: ',',
      allowQuoting: true
  );
} catch (Exception ex) {
  RaportTBox.Text +=($"Помилка зчитування CSV: {ex.Message}");
  return;
}
```

Рис. 3.8. Ініціалізація середовища та завантаження даних

Далі виконується завантаження навчального набору даних у вигляді IDataView з CSV-файлу, шлях до якого було збережено раніше. Для цього застосовується метод LoadFromTextFile, який використовує узагальнений тип EmployeeData, що описує структуру вхідних ознак працівника. Завантаження здійснюється з урахуванням заголовків у першому рядку файлу, з розділювачем-комою, а також дозволом на використання лапок для обрамлення значень. Процедура завантаження обгорнута в конструкцію try-catch, яка дозволяє перехопити винятки у разі помилки – наприклад, якщо файл пошкоджений, має неправильний формат або відсутній доступ до нього. У разі виникнення помилки її опис додається до текстового звіту в інтерфейсі

користувача, що дозволяє швидко локалізувати проблему та зупиняє подальше виконання процедури. Така реалізація є прикладом захищеного завантаження даних із вбудованим механізмом повідомлення про помилки.

Наступним етапом є розділення вибірки на навчальну та тестову підмножини, що є обов'язковим кроком у процесі побудови моделі машинного навчання (рис. 3.9). Такий поділ дає змогу здійснити навчання моделі на одній частині даних, а оцінювання її якості – на іншій, незалежній частині, що дозволяє уникнути перенавчання і дає об'єктивну оцінку здатності моделі до узагальнення.

```
var split =
  mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2, seed: 123);
var trainData = split.TrainSet;
var testData = split.TestSet;
```

Рис. 3.9. Розділення вибірки на навчальну та тестову підмножини

Для реалізації цього кроку використовується вбудований метод TrainTestSplit, який викликається через об'єкт mlContext.Data. Як параметр задається пропорція для тестової вибірки, яка у даному випадку становить 20% від загального обсягу даних. Решта 80% використовується для навчання. Додатково, як і в попередніх етапах, встановлюється фіксоване зерно випадковості, що забезпечує стабільність результатів при повторному запуску. Результатом виклику функції є структура, яка містить окремо сформовані навчальний (TrainSet) та тестовий (TestSet) піднабори, що надалі використовуються у процесі побудови та оцінки моделі відповідно.

На рис. 3.10 зображено процес побудова трансформаційного конвеєра – послідовності операцій попередньої обробки даних, які будуть виконані перед навчанням моделі. Конвеєр формується на основі методів API ML.NET і складається з кількох етапів, що послідовно додаються один до одного.

```
var pipeline = mlContext.Transforms.Conversion.ConvertType(
        inputColumnName: nameof(EmployeeData.Attrition),  // "Attrition"
        outputColumnName: "Label",                        // створюємо нову колонку Label типу bool
        outputKind: DataKind.Boolean
    )
    // Далі кодуємо категоріальні стовпці
    .Append(mlContext.Transforms.Categorical.OneHotEncoding(new[]
    {
new InputOutputColumnPair(nameof(EmployeeData.BusinessTravel)),
new InputOutputColumnPair(nameof(EmployeeData.Department)),
new InputOutputColumnPair(nameof(EmployeeData.EducationField)),
new InputOutputColumnPair(nameof(EmployeeData.Gender)),
new InputOutputColumnPair(nameof(EmployeeData.JobRole)),
new InputOutputColumnPair(nameof(EmployeeData.MaritalStatus)),
new InputOutputColumnPair(nameof(EmployeeData.Over18)),
new InputOutputColumnPair(nameof(EmployeeData.OverTime))
    }))
    // Далі об'єднуємо усі фічі
    .Append(mlContext.Transforms.Concatenate("Features",
        nameof(EmployeeData.Age),
        nameof(EmployeeData.BusinessTravel), nameof(EmployeeData.Department),
        nameof(EmployeeData.DailyRate), nameof(EmployeeData.DistanceFromHome),
        nameof(EmployeeData.Education), nameof(EmployeeData.EducationField),
        nameof(EmployeeData.EmployeeCount), // EmployeeNumber - ідентифікатор
        nameof(EmployeeData.EnvironmentSatisfaction), nameof(EmployeeData.Gender),
        nameof(EmployeeData.HourlyRate), nameof(EmployeeData.JobInvolvement),
        nameof(EmployeeData.JobLevel), nameof(EmployeeData.JobRole),
        nameof(EmployeeData.JobSatisfaction), nameof(EmployeeData.MaritalStatus),
        nameof(EmployeeData.MonthlyIncome), nameof(EmployeeData.MonthlyRate),
        nameof(EmployeeData.NumCompaniesWorked), nameof(EmployeeData.Over18),
        nameof(EmployeeData.OverTime), nameof(EmployeeData.PercentSalaryHike),
        nameof(EmployeeData.PerformanceRating), nameof(EmployeeData.RelationshipSatisfaction),
        nameof(EmployeeData.StandardHours), nameof(EmployeeData.StockOptionLevel),
        nameof(EmployeeData.TotalWorkingYears), nameof(EmployeeData.TrainingTimesLastYear),
        nameof(EmployeeData.WorkLifeBalance), nameof(EmployeeData.YearsAtCompany),
        nameof(EmployeeData.YearsInCurrentRole), nameof(EmployeeData.YearsSinceLastPromotion),
        nameof(EmployeeData.YearsWithCurrManager)
```

Рис. 3.10. Побудова конвеєра

Першим кроком у конвеєрі є перетворення цільової змінної Attrition у логічний тип Boolean, що відповідає вимогам для класифікаційних задач. Створюється нова колонка з ім'ям Label, яка буде інтерпретуватися моделлю як мітка класу. Така операція є стандартною для підготовки вихідного атрибуту у ML.NET. Далі здійснюється кодування категоріальних ознак у числові вектори за допомогою техніки one-hot encoding. До цієї категорії належать такі ознаки, як вид службових подорожей, підрозділ, спеціальність освіти, стать, посада, сімейний стан, а також значення полів Over18 і OverTime. Кодування дозволяє перетворити номінативні значення у формат, придатний для обробки алгоритмами машинного навчання.

Наступним кроком є формування єдиного вектору ознак з назвою Features. Для цього виконується об'єднання всіх вхідних полів, як числових,

так і попередньо закодованих категоріальних. Серед них – вік, освітній рівень, щоденна ставка, кількість відпрацьованих років, оцінка задоволеності, доходи, кількість компаній, де працював працівник, кількість навчань за рік, тривалість роботи в компанії тощо. Колонка EmployeeNumber, що є ідентифікатором, виключається з переліку, оскільки не містить інформативного навантаження.

У результаті формується послідовний трансформаційний конвеєр, який автоматизує підготовку даних до навчання моделі та забезпечує уніфіковану структуру вхідного простору ознак. Такий підхід гарантує сталість і відтворюваність усіх трансформацій при наступному використанні моделі.

Завершальним етапом побудови трансформаційного конвеєра є додавання алгоритму машинного навчання, який виконуватиме класифікацію на основі підготовлених ознак (рис. 3.11). У цьому випадку використовується тренер логістичної регресії, реалізований у бібліотеці ML.NET через метод SdcaLogisticRegression. Це один із стандартних бінарних класифікаторів, який базується на стохастичному градієнтному спуску з координатним оновленням (SDCA – Stochastic Dual Coordinate Ascent), що добре підходить для задач передбачення двійкової змінної, зокрема, такої як Attrition.

```
.Append(mlContext.BinaryClassification.Trainers.SdcaLogisticRegression(
    labelColumnName: "Label",
    featureColumnName: "Features"
));
```

Рис. 3.11. Додавання алгоритму машинного навчання

У параметрах тренера явно вказуються назви колонок: Label – цільова змінна, яку система намагається передбачити, та Features – об’єднаний вектор ознак, що слугує входом для навчання. Додавання цього кроку до конвеєра завершує його формування, внаслідок чого створюється повна послідовність дій – від перетворення та кодування даних до застосування обраного алгоритму класифікації. Такий підхід дозволяє здійснити повний цикл

навчання моделі в єдиній структурі, яка зберігає узгодженість і спрощує подальше тестування та розгортання моделі.

Після завершення навчання моделі виконується етап її оцінювання на тестових даних, що не використовувались під час тренування. Цей етап є важливим для перевірки здатності моделі до узагальнення – тобто, наскільки добре вона працює з новими, раніше невідомими прикладами. Спочатку у текстовому полі звіту виводиться повідомлення про початок процесу оцінювання, що інформує користувача про перебіг виконання.

```
RaportTBox.Text += ("Оцінювання моделі на тестових даних...\r\n");
var predictions = model.Transform(testData);
var metrics = mlContext.BinaryClassification.Evaluate(
    predictions,
    labelColumnName: nameof(EmployeeData.Attrition)
);
RaportTBox.Text += ($"Точність (Accuracy): {metrics.Accuracy:P2}\r\n");
RaportTBox.Text += ($"AUC: {metrics.AreaUnderRocCurve:P2}\r\n");
RaportTBox.Text += ($"F1 Score: {metrics.F1Score:P2} \r\n");
```

Рис. 3.12. Оцінювання моделі на тестових даних

Модель застосовується до тестової вибірки за допомогою методу Transform, у результаті чого формується прогнозована інформація – включно з імовірностями та передбаченими класами для кожного прикладу. Отримані результати передаються на вхід методу Evaluate, який обчислює набір стандартних метрик для бінарної класифікації. У параметрах методу явно зазначається, що еталонною змінною є поле Attrition, яке вказує на факт звільнення працівника.

Після обчислення метрик система виводить у звіт значення точності (Accuracy), яке показує відсоток правильно класифікованих прикладів, площу під ROC-кривою (AUC), що відображає якість ранжування позитивних прикладів, та F1-метрику, яка враховує баланс між повнотою та точністю моделі. Кожне з цих значень виводиться з форматуванням у вигляді відсотків з двома знаками після коми, що робить звіт зрозумілим та наочним для кінцевого користувача. Такий підхід дозволяє швидко оцінити ефективність

моделі й прийняти рішення про її подальше використання або необхідність повторного навчання.

Метод, який зображено на рис. 3.13 виконується при натисканні кнопки збереження моделі, реалізує повний цикл перевірки, створення і збереження навченої моделі у файловій системі, а також її реєстрацію в базі даних із супровідним логуванням дій користувача. Перед початком збереження перевіряється правильність заповнення вхідних полів за допомогою методу IsDataEnteringCorrect. Якщо введені дані відповідають вимогам, система переходить до основної частини процедури.

```
private void SaveBtn_Click(object sender, EventArgs e) {
  if (IsDataEnteringCorrect()) {
    //Зберігання моделі
    string pathName = @"\teach\" + GenerateFileName() + ".zip";
    string localProj =
      System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location);
    _ModelsProvider.InsertModels(ModelsNamesTBox.Text, pathName);
    mlContext.Model.Save(model, dataView.Schema, localProj + pathName);
    ClearAllData();
    _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
      "Було навчено модель " +
      ModelsNamesTBox.Text, DateTime.Now);
    MessageBox.Show("Дані успішно збережено!");
  }
}
```

Рис. 3.13. Збереження моделі

На початку формується шлях для збереження моделі у вигляді ZIP-архіву, де назва файлу генерується автоматично за допомогою методу GenerateFileName. Цей шлях доповнюється до кореневої директорії виконуваного застосунку, що отримується динамічно через відображення місця розташування збірки. Далі до бази даних додається запис про нову модель – її назва та шлях збереження – за допомогою відповідного методу об’єкта _ModelsProvider.

Модель, яка була попередньо навчена, зберігається на диск за допомогою методу Save, де вказується не лише сама модель, але й схема даних, яка використовувалась під час навчання. Після завершення збереження всі поля у формі очищуються, щоб забезпечити підготовку до

можливого наступного сеансу. Паралельно до системного журналу вноситься запис про створення нової моделі із зазначенням користувача, який здійснив дію, назви моделі та поточної дати. Наприкінці метод інформує користувача про успішне завершення операції за допомогою стандартного вікна повідомлення. Така послідовність дій забезпечує збереження моделі як на файловому рівні, так і на рівні метаданих у системі.

На рис. 3.14 зображено метод, який призначений для завантаження раніше збереженої моделі машинного навчання та створення на її основі об’єкта прогнозування. На початку формується повний шлях до файлу моделі, який комбінується з директорією запуску застосунку за допомогою властивості StartupPath та відносного шляху, переданого у параметрі FilePath. Це дозволяє локалізувати файл моделі незалежно від абсолютного розташування програми на комп’ютері користувача.

```
private void LoadData(string FilePath) {
  string localProj = Application.StartupPath + FilePath;
  // Визначення DataViewSchema для конвеєра
  // підготовки даних і навченої моделі
  DataViewSchema modelSchema;
  // Завантаження моделі
  ITransformer model = mlContext.Model.Load(localProj,
    out modelSchema);
  // Створення механізму прогнозування
  predictionEngine =
      mlContext.Model.CreatePredictionEngine<EmployeeData,
      EmployeePrediction>(model);
}
```

Рис. 3.14. Збереження моделі

В методі оголошується змінна типу DataViewSchema, яка використовується для отримання схеми вхідних даних, що були застосовані під час навчання моделі. Цей об’єкт потрібен для правильного зчитування структури ознак, із якими буде працювати модель у процесі прогнозування. За допомогою методу Load завантажується файл моделі у вигляді об’єкта ITransformer, а отримана схема зберігається у змінній modelSchema.

Після завантаження моделі створюється механізм прогнозування – PredictionEngine, який поєднує типи вхідних даних (EmployeeData) і результатів прогнозу. Цей об’єкт дозволяє застосовувати модель до одиничних прикладів у реальному часі, тобто вводити конкретні характеристики кандидата вручну та отримувати миттєвий результат прогнозу. Метод виконує підготовку системи до прогнозування й забезпечує можливість інтерактивного використання навченої моделі без необхідності повторного тренування.

На рис. 315 зображено фрагмент методу, що активується під час натискання кнопки прогнозування та виконує ініціацію процесу оцінювання ймовірності звільнення працівника на основі введених користувачем даних. Насамперед здійснюється перевірка коректності заповнення всіх обов’язкових полів, а також перевірка наявності завантаженої моделі. Для цього використовуються допоміжні методи IsAllEmployeeDataCorrect та IsModelExist. У разі позитивного результату виконання обох перевірок система переходить до формування структури вхідних даних.

```
private void PredictBtn_Click(object sender, EventArgs e) {
  // Перевірка, чи введені всі необхідні дані та чи існує модель
  if (IsAllEmployeeDataCorrect() && IsModelExist()) {
    // Створення об'єкта EmployeeData з введених значень
    EmployeeData employeeData = new EmployeeData {
      Age = float.Parse(AgeTBox.Text),
      Attrition = 0, // Значення буде прогнозуватися моделлю
      BusinessTravel = BusinessTravelCBox.Text,
      DailyRate = float.Parse(DailyRateTBox.Text),
      Department = DepartmentCBox.Text,
      DistanceFromHome = float.Parse(DistanceFromHomeTBox.Text),
      Education = float.Parse(EducationTBox.Text),
      EducationField = EducationFieldCBox.Text,
      EmployeeCount = float.Parse(EmployeeCountTBox.Text),
      EmployeeNumber = float.Parse(EmployeeNumberTBox.Text),
```

Рис. 3.15. Ініціація процесу оцінювання ймовірності звільнення працівника

Створюється новий об’єкт типу EmployeeData, який заповнюється значеннями з відповідних елементів інтерфейсу користувача: текстових полів та випадаючих списків. Поле Attrition, яке є цільовим, задається значенням 0,

оскільки воно буде передбачене безпосередньо моделлю. Кожне з числових значень попередньо перетворюється до формату float, що відповідає типу ознак у класі EmployeeData.

Заповнюються ключові характеристики працівника, включаючи вік, рівень освіти, відстань до роботи, посаду, заробітну плату, рівень задоволеності, стать, рівень залученості, відділ та інші. Таким чином, формується повний набір вхідних ознак, на основі яких модель зможе здійснити прогноз щодо ймовірності звільнення конкретного кандидата або працівника. Цей об'єкт передається у прогнозуючий механізм у наступних кроках методу.

Після формування об’єкта з вхідними даними employeeData здійснюється прогнозування за допомогою раніше створеного механізму predictionEngine (рис. 3.16). Метод Predict застосовує навчений алгоритм до переданого прикладу і повертає результат прогнозу у вигляді об’єкта EmployeePrediction. Цей об’єкт містить імовірність звільнення та бінарне значення PredictedLabel, яке інтерпретується як остаточне рішення моделі.

```
var prediction = predictionEngine.Predict(employeeData);
// Формуємо результат прогнозування
var answer = new StringBuilder();
answer.AppendLine("\r\n--- Прогнозування ---");
answer.AppendLine($"\r\nЙмовірність звільнення: " +
  $"{prediction.Probability:P2}");
answer.AppendLine($"\r\nПрогнозований статус:" +
  $" {(prediction.PredictedLabel ? "Звільнення" : "Залишається")}");
```

Рис. 3.16. Прогнозування ймовірності звільнення працівника

Також створюється об’єкт типу StringBuilder, що використовується для форматування тексту виводу. У результативний блок додається заголовок секції прогнозування, після чого виводиться значення ймовірності звільнення у відсотках із двома знаками після коми. Потім формується текстовий опис передбаченого статусу: якщо значення PredictedLabel є істинним (true), система інтерпретує це як очікуване звільнення; у протилежному випадку – як імовірне залишення працівника на посаді. Такий підхід дозволяє наочно

відобразити як числову ймовірність, так і текстовий висновок, зрозумілий кінцевому користувачу.

3.3 Технічні вимоги

Програмна система SmartRecruit, розроблена для підтримки процесу прийняття кадрових рішень із використанням предиктивної аналітики, має низку технічних вимог, що забезпечують її стабільну роботу, зручність використання та можливість подальшої підтримки й розвитку. Основна функціональність реалізована як настільний застосунок з графічним інтерфейсом користувача на основі Windows Forms з використанням середовища Visual Studio 2022 та мови програмування C#. Для реалізації алгоритмів машинного навчання використовується бібліотека ML.NET, яка дозволяє інтегрувати повноцінну модель класифікації без потреби у зовнішніх сервісах чи хмарній інфраструктурі.

Функціональність системи охоплює можливість завантаження вхідних даних із CSV-файлів, побудову моделі машинного навчання, оцінювання її якості, збереження моделі у форматі .zip, подальше її завантаження, а також здійснення індивідуального прогнозування на основі введених характеристик кандидата. Застосунок підтримує багаторівневу систему ролей користувачів (адміністратор, користувач), забезпечує ведення журналу подій, а також дозволяє управляти доступними моделями. Усі дії супроводжуються повідомленнями у зрозумілому текстовому вигляді, що робить інтерфейс доступним навіть для користувачів без спеціальної технічної підготовки.

Застосунок функціонує в автономному режимі без потреби у постійному підключенні до інтернету, а дані зберігаються локально, що забезпечує дотримання вимог конфіденційності.

Для коректної роботи програмного продукту на машині розробника або у середовищі експлуатації необхідна наявність апаратного забезпечення з такими мінімальними характеристиками:

– процесор: двоядерний CPU із частотою не менше 2.0 ГГц (рекомендовано – Intel Core i5 або еквівалент);

– оперативна пам’ять: не менше 4 ГБ (рекомендовано 8 ГБ і більше для роботи з більшими наборами даних);

– жорсткий диск: 1 ГБ вільного місця для збереження моделей, логів і користувацьких даних;

– монітор: мінімальна роздільна здатність 1280×768 пікселів для коректного відображення інтерфейсу;

– наявність клавіатури та миші як основних пристроїв введення.

Кінцевий користувач системи SmartRecruit повинен мати доступ до персонального комп’ютера або ноутбука, на якому встановлена операційна система Windows 10 або новіша. Для забезпечення повноцінної роботи застосунку необхідно попередньо встановити MSSQLLocalDB – локальну версію Microsoft SQL Server, яка використовується для збереження даних користувачів, моделей та журналів подій. MSSQLLocalDB не потребує окремого конфігурування та запускається автоматично під час звернення з боку застосунку. Інсталяція програми не вимагає спеціальних технічних навичок, а запуск здійснюється безпосередньо через виконуваний файл .exe.

Для комфортної роботи користувача рекомендується мати доступ до дисплею зі стандартним масштабуванням і стандартні технічні параметри пристрою, аналогічні до мінімальних вимог системи. Додаткове програмне забезпечення – зокрема, табличний редактор (наприклад, Microsoft Excel або інший CSV-редактор) – може використовуватись для формування вхідних наборів даних, однак не є обов’язковим для функціонування системи.

Завдяки локальній роботі, автономності та простому інтерфейсу система може використовуватись навіть у середовищах із обмеженим технічним забезпеченням, що робить її доступною для широкого кола користувачів.

3.4 Інструкція користувача

Для коректного використання програмної системи SmartRecruit користувачеві необхідно пройти процедуру авторизації, яка забезпечує контроль доступу та розмежування прав між звичайними користувачами й адміністраторами. Вхід до системи реалізовано через спеціальну форму, що відкривається автоматично при запуску застосунку.

На рис. 3.17 представлено форму авторизації в системі, яка є першим етапом взаємодії користувача з програмним продуктом.

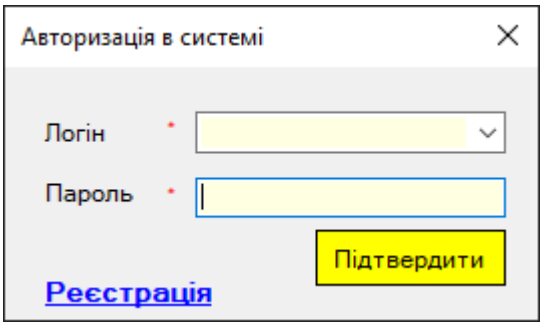

Рис. 3.17. Авторизація користувача

У формі передбачено два обов'язкові поля для введення – Логін і Пароль. Після їх заповнення користувач натискає кнопку «Підтвердити», що ініціює перевірку облікових даних. У випадку, якщо користувач ще не зареєстрований, передбачено гіперпосилання «Реєстрація», яке відкриває відповідну форму створення нового облікового запису.

У разі відсутності облікового запису, є процедура створення нового користувача. Вона дає змогу зареєструватися в системі з подальшим використанням персонального профілю. Важливо зазначити, що перший зареєстрований обліковий запис автоматично отримує роль адміністратора, усі наступні – роль звичайного користувача.

На рис. 3.18 наведено форму реєстрації нового користувача, яка відкривається після натискання відповідного посилання у вікні авторизації.

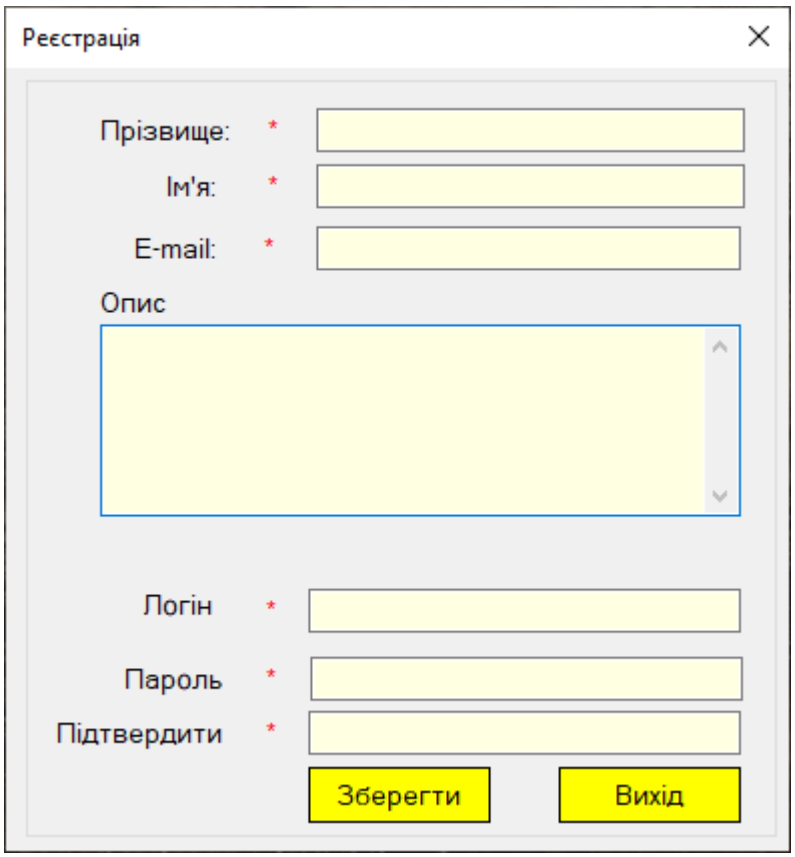

Рис. 3.18. Реєстрація нового користувача

Форма містить обов’язкові поля для введення прізвища, імені, електронної адреси, логіна та пароля (з підтвердженням). Окрім того, користувач може додатково вказати опис – наприклад, посаду, підрозділ або іншу довільну інформацію. Після заповнення всіх полів потрібно натиснути кнопку «Зберегти», після чого відбудеться реєстрація в системі. Кнопка «Вихід» дозволяє повернутись до попереднього вікна без збереження даних.

Після успішної авторизації або реєстрації користувач має можливість перейти до одного з основних розділів системи – тренування моделі, яка використовується для прогнозування ймовірності звільнення працівника. У цьому режимі адміністратор або авторизований користувач може створити нову модель машинного навчання на основі власного набору даних у форматі CSV.

На рис. 3.19 зображено форму для навчання моделей, що включає всі необхідні елементи для побудови, оцінювання та збереження моделі.

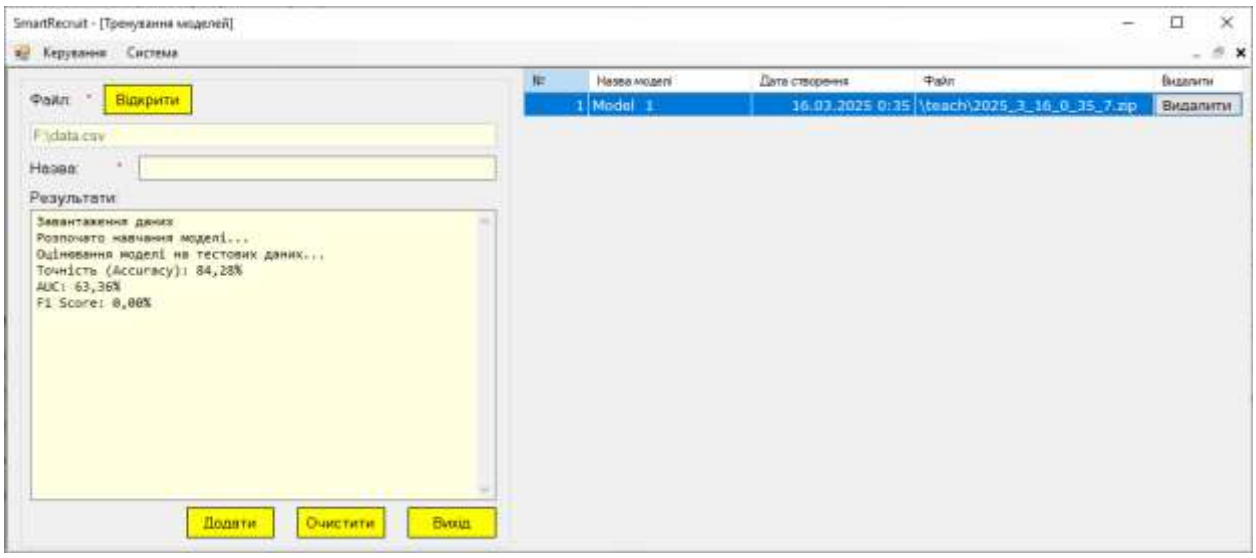

Рис. 3.19. Тренування моделі

У лівій частині вікна знаходиться панель завантаження файлу навчальних даних через кнопку «Відкрити», після чого шлях до обраного файлу автоматично відображається в полі. Користувач повинен ввести назву нової моделі у відповідне текстове поле. Після завантаження даних система проводить навчання, оцінює її на тестовій вибірці та виводить метрики точності, AUC і F1 Score у полі «Результати». Для збереження моделі необхідно натиснути кнопку «Додати», після чого нова модель з’являється у таблиці праворуч із зазначенням дати створення, назви, шляху до файлу та можливістю її видалення. Кнопки «Очистити» та «Вихід» призначені для скидання введених даних та повернення до головного меню відповідно.

Ще одним ключовим функціональним компонентом системи SmartRecruit є модуль прогнозування співпраці, який дозволяє оцінити ймовірність звільнення працівника або кандидата на основі введених характеристик. Цей інструмент дає змогу кадровим спеціалістам швидко отримати обґрунтований результат без необхідності глибокого занурення в технічні деталі побудови моделі.

На рис. 3.20 показано форму прогнозування, яка надає зручний інтерфейс для введення понад 30 параметрів, що охоплюють демографічні, фінансові, поведінкові та професійні характеристики працівника. Користувач

обирає попередньо натреновану модель зі списку, заповнює вхідні поля та натискає кнопку «Прогнозувати», після чого в правій частині вікна відображається деталізований результат аналізу, включно з ймовірністю звільнення у відсотках та текстовим висновком (наприклад, «залишається» або «звільнення»).

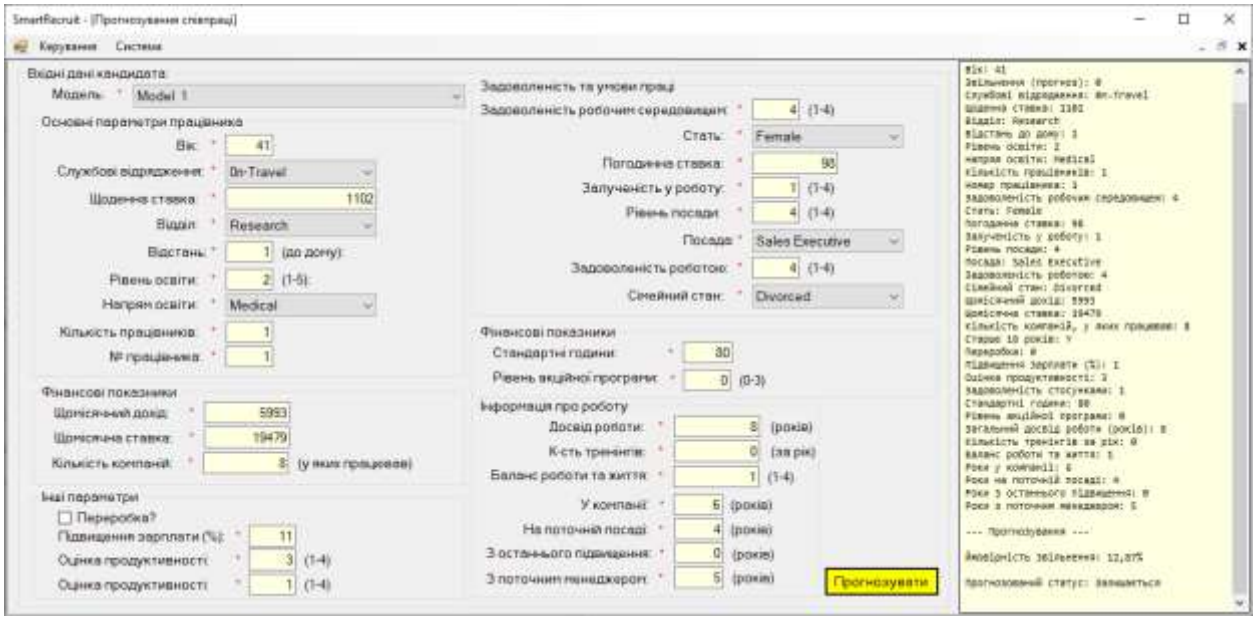

Рис. 3.20. Прогнозування співпраці

Форма дає змогу персоналізовано оцінити ризики плинності кадрів до моменту прийняття кандидата на роботу, що дозволяє значно підвищити якість підбору персоналу та зменшити витрати на повторний рекрутинг.

Адміністратор системи SmartRecruit має доступ до функціоналу управління обліковими записами користувачів, що забезпечує контроль доступу, реєстрацію нових співробітників, а також редагування чи видалення наявних записів. Така можливість є критично важливою для підтримки актуальності користувацької бази та забезпечення відповідного рівня безпеки.

На рис. 3.21 представлено форму облікових даних користувачів, яка призначена для створення нових записів та перегляду вже зареєстрованих користувачів системи.

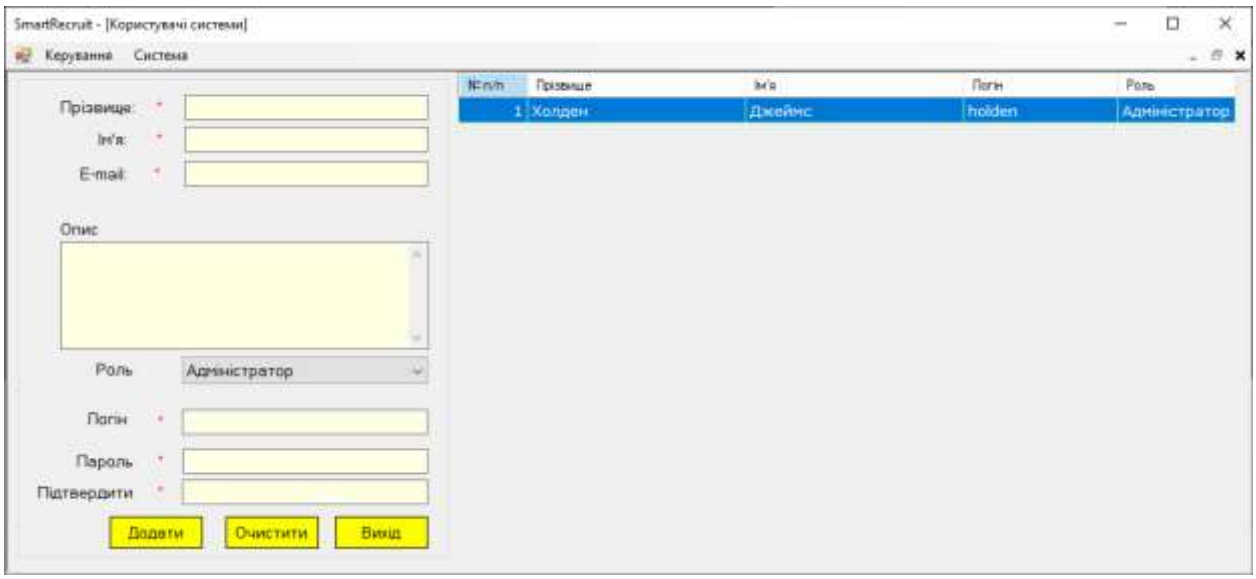

Рис. 3.21. Форма облікових даних користувачів

У лівій частині вікна знаходиться блок введення даних, де адміністратор може вказати прізвище, ім’я, електронну адресу, опис, логін, пароль, а також обрати роль – «Користувач» або «Адміністратор». Після заповнення полів потрібно натиснути кнопку «Додати», щоб зареєструвати нового користувача. Всі збережені записи відображаються у вигляді таблиці праворуч, де зручно переглядати перелік зареєстрованих осіб із зазначенням логіна та ролі. Кнопки «Очистити» та «Вихід» дозволяють скинути введені значення або повернутися до головного вікна відповідно.

ВИСНОВКИ

Дана робота присвячена розробці програмної системи SmartRecruit, що реалізує предиктивну аналітику для оптимізації процесу рекрутингу. Метою дослідження стало створення ефективного та доступного інструменту, здатного прогнозувати ймовірність звільнення працівника на основі вхідних характеристик кандидата. Поставлені задачі було вирішено шляхом детального аналізу теоретичних основ, порівняння існуючих рішень і реалізації власного програмного продукту з використанням методів машинного навчання.

У першому розділі була обґрунтована доцільність використання предиктивної аналітики в сфері підбору персоналу, визначено її сутність, ключові характеристики, принципи формування моделей та типові етапи побудови аналітичного прогнозу. Розглянуто джерела великих даних, які можуть бути використані для навчання моделей у рекрутингу, зокрема внутрішні HR-системи, онлайн-профілі, результати тестувань та неструктуровані джерела інформації. Було показано, що успішне використання таких даних дозволяє виявити приховані закономірності, які неможливо ідентифікувати за допомогою традиційних підходів.

Другий розділ висвітлює вибір інструментів, необхідних для реалізації задачі. Вибір ML.NET обґрунтовано з погляду інтеграції алгоритмів машинного навчання безпосередньо у середовище C# без потреби у зовнішніх бібліотеках або сервісах. Детально описано можливості середовища Visual Studio 2022, мови програмування C#, а також обґрунтовано вибір трьохрівневої архітектури. Проведено порівняльний аналіз трьох популярних систем – IBM Watson Recruitment, SAP SuccessFactors і Zoho Recruit – на основі низки характеристик: точність, доступність, простота використання, українізований інтерфейс, адаптованість до локального ринку, потреба в технічному супроводі та можливість автономного застосування. На основі порівняння зроблено висновок, що

розроблена система є ефективною альтернативою для малого та середнього бізнесу в Україні, оскільки поєднує достатню точність (84,28%) з простотою інтерфейсу, локальною реалізацією та повною українізацією.

Третій розділ був присвячений безпосередньо розробці програмного продукту. Описано функціональні можливості системи, зокрема – управління користувачами, створення, збереження, перегляд та використання моделей машинного навчання. Наведено діаграму варіантів використання та блок-схеми ключових алгоритмів – побудови моделі та прогнозування на її основі. Описано реалізацію трьохрівневої архітектури з поділом на рівень представлення, бізнес-логіки та доступу до даних.

Проведена мною робота демонструє не лише застосування сучасних технологій машинного навчання та програмної інженерії для вирішення прикладної задачі оптимізації рекрутингових процесів, а й мій особистий внесок у реалізацію повноцінного програмного рішення. Усі етапи – від формалізації задачі, моделювання архітектури, вибору технологій до реалізації ключових модулів і перевірки працездатності системи – виконані самостійно, з урахуванням актуальних вимог бізнес-середовища та особливостей локального ринку. Реалізована система дозволяє на основі історичних даних прогнозувати ймовірність звільнення нового працівника, що може стати основою для підвищення обґрунтованості прийняття рішень у сфері найму. Завдяки локальній архітектурі, україномовному інтерфейсу та простоті розгортання система є доступною для використання у вітчизняному бізнес-середовищі без потреби у значних технічних ресурсах.

Перспективи подальшого розвитку полягають у вдосконаленні моделі прогнозування шляхом розширення обсягу і типів вхідних даних, реалізації підтримки багатокласової класифікації, автоматичної побудови звітів, а також інтеграції з зовнішніми базами резюме чи HR-системами. Результати роботи можуть бути використані як основа для впровадження системи в практику управління персоналом у малих і середніх організаціях України.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bilan Y., Mishchuk H., Roshchyk I., Joshi O. Hiring and retaining skilled employees in SMEs: problems in human resource practices and links with organizational success. Business. 2020. Vol. 21, No. 2, pp. 780-791.

2. Cho W., Choi S., Choi H. Human resources analytics for public personnel management: Concepts, cases, and caveats. Administrative Sciences. 2023. Vol. 13, No. 2, 41 p.

3. What is predictive analytics? URL: https://www.ibm.com/think/topics/predictive-analytics (дата звернення 16.03.2025).

4. Reynolds E. Coffeehouse culture in the Atlantic World. Bloomsbury Publishing. 1650-1789. 2022. Vol. 13, No. 2, pp. 1650-1789.

5. Bringing Predictive Analytics in Recruitment: 6 Interesting Case Studies. URL: https://benchpoint.com/blog/predictive-analytics-in-recruitment/ (дата звернення 16.03.2025).

6. What Is Predictive Analytics? Definition, Types and Examples. URL: https://www.datamation.com/big-data/what-is-predictive-analytics (дата звернення 16.03.2025).

7. Штучний інтелект для рекрутерів і HR: Революція в підборі персоналу. URL: https://ua.learntoearn.global/blog/shtuchnyi-intelekt-dlia-rekruteriv-i-hr (дата звернення 16.03.2025).

8. Nath G., Harfouche A., Coursey A., Saha K., Prabhu S., Sengupta S. Integration of a machine learning model into a decision support tool to predict absenteeism at work of prospective employees. 2022. 14 p.

9. Mohiuddin K., Alam M., Alam M., Welke P., Martin M., Lehmann J., Vahdati S. Retention Is All You Need. 2023. 7 p.

10. Як штучний інтелект використовується в HR? URL: https://fillin.ua/stati/yak-shtuchniy-intelekt-vikoristovuetsya-v-hr/ (дата звернення 16.03.2025).

11. Зелінська Д. О., Зянько В. В. Інноваційні методи прогнозування потреби в персоналі. Матеріали Всеукраїнської науково-практичної конференції «Сучасні тенденції розвитку економіки та підприємництва», 2021. С. 45-49.

12. Пучкова С. HR-аналітика як сучасний підхід до прийняття кадрових рішень в бізнесі. Економічний аналіз, Т. 30, № 1. 2020. С. 68-75.

13. HR-аналітика: що вам потрібно знати, щоб почати роботу. URL: https://ukr.pritula.academy/tpost/3tx2gbxlg1-hr-analtika-scho-vam-potrbno-znati-schob (дата звернення 16.03.2025).

14. Unlocking the future of recruitment: The power of predictive analytics. URL: https://hireez.com/blog/predictive-analytics-in-recruitment/ (дата звернення 16.03.2025).

15. Jurafsky D., Martin J. H. Speech and Language Processing. – 3rd ed. – Upper Saddle River: Prentice Hall, 2021. – 1024 p.

16. Smeulders A. W. M., Worring M., Santini S., Gupta A., Jain R. Content-Based Image Retrieval at the End of the Early Years // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2020. Vol. 22, No. 12. pp. 1349–1380.

17. Pease G., Byerly B., Fitz-enz J. Human Capital Analytics: How to Harness the Potential of Your Organization's Greatest Asset. Hoboken: Wiley, 2013. 224 p.

18. Creating/training and running ML.NET models. URL: https://devblogs.microsoft.com/cesardelatorre/what-is-ml-net-1-0-machine-learning-for-net/ (дата звернення 16.03.2025).

19. Безменов М.І., Безменова О.М., Калінін Д.В. Основи візуального програмування мовою C#: навч. посіб. для студентів навчально-наукового інституту комп’ютерних наук та інформаційних технологій. – Харків: ФОП Панов А. М., 2023. 648 с.

20. Троелсен Е. C# 7.0 та платформа .NET: Посібник для професіоналів. - Львів: Видавництво Львівської політехніки, 2018. - 918 с.

21. IBM Watson Talent Frameworks Reviews & Provider Details. URL: https://www.g2.com/products/ibm-watson-talent-frameworks/reviews (дата звернення 16.03.2025).

22. IBM Watson reviews. URL: https://www.glassdoor.com/Reviews/IBM-Watson-Reviews-EI_IE354.0,3_KO4,10.htm (дата звернення 16.03.2025).

23. SAP SuccessFactors 1H 2024 Release Analysis: Recruiting - Rizing. URL: https://rizing.com/human-capital-management/sap-successfactors-1h-2024-release-analysis-recruiting/ (дата звернення 16.03.2025).

24. SAP SuccessFactors HXM Suite Reviews. URL: https://www.softwareadvice.com/hr/sap-successfactors-hxm-suite-profile/reviews/ (дата звернення 16.03.2025).

25. Zoho Recruit | Overview Dashboard. URL: https://help.zoho.com/portal/en/kb/recruit/module-set-up/home-tab/overview-dashboard/articles/overview-dashboard (дата звернення 16.03.2025).

26. Zoho Recruit Ratings Overview. URL: https://www.gartner.com/reviews/market/applicant-tracking-systems/vendor/zoho/product/zoho-recruit (дата звернення 16.03.2025).

ДОДАТКИ

Додаток А – Лістинги програмного коду

Лістинг 1. Код класу «EmployeeAttritionProvider»

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

namespace EmployeeAttritionPredictionApp.Providers {
  internal class EmployeeAttritionProvider {
  }
}

// Клас для зчитування рядків із CSV
public class EmployeeData {
  // Перша колонка CSV (індекс 0)
  [LoadColumn(0)]
  public float Age { get; set; }

  // Друга колонка CSV (індекс 1) – ТЕПЕР 0 або 1, замість "No"/"Yes"
  [LoadColumn(1)]
  public float Attrition { get; set; }

  // Решта полів (індекси 2..34)
  [LoadColumn(2)] public string BusinessTravel { get; set; }
  [LoadColumn(3)] public float DailyRate { get; set; }
  [LoadColumn(4)] public string Department { get; set; }
  [LoadColumn(5)] public float DistanceFromHome { get; set; }
  [LoadColumn(6)] public float Education { get; set; }
  [LoadColumn(7)] public string EducationField { get; set; }
  [LoadColumn(8)] public float EmployeeCount { get; set; }
  [LoadColumn(9)] public float EmployeeNumber { get; set; }
  [LoadColumn(10)] public float EnvironmentSatisfaction { get; set; }
  [LoadColumn(11)] public string Gender { get; set; }
  [LoadColumn(12)] public float HourlyRate { get; set; }
  [LoadColumn(13)] public float JobInvolvement { get; set; }
  [LoadColumn(14)] public float JobLevel { get; set; }
  [LoadColumn(15)] public string JobRole { get; set; }
  [LoadColumn(16)] public float JobSatisfaction { get; set; }
  [LoadColumn(17)] public string MaritalStatus { get; set; }
  [LoadColumn(18)] public float MonthlyIncome { get; set; }
  [LoadColumn(19)] public float MonthlyRate { get; set; }
  [LoadColumn(20)] public float NumCompaniesWorked { get; set; }
  [LoadColumn(21)] public string Over18 { get; set; }
```

```
  [LoadColumn(22)] public string OverTime { get; set; }
  [LoadColumn(23)] public float PercentSalaryHike { get; set; }
  [LoadColumn(24)] public float PerformanceRating { get; set; }
  [LoadColumn(25)] public float RelationshipSatisfaction { get; set; }
  [LoadColumn(26)] public float StandardHours { get; set; }
  [LoadColumn(27)] public float StockOptionLevel { get; set; }
  [LoadColumn(28)] public float TotalWorkingYears { get; set; }
  [LoadColumn(29)] public float TrainingTimesLastYear { get; set; }
  [LoadColumn(30)] public float WorkLifeBalance { get; set; }
  [LoadColumn(31)] public float YearsAtCompany { get; set; }
  [LoadColumn(32)] public float YearsInCurrentRole { get; set; }
  [LoadColumn(33)] public float YearsSinceLastPromotion { get; set; }
  [LoadColumn(34)] public float YearsWithCurrManager { get; set; }
}

// Результат прогнозу
public class EmployeePrediction {
  // Для бінарної класифікації ML.NET повертатиме True / False
  [ColumnName("PredictedLabel")]
  public bool PredictedLabel { get; set; }

  // Ймовірність звільнення, якщо PredictedLabel = true
  [ColumnName("Probability")]
  public float Probability { get; set; }

  // Сирий вихід
  [ColumnName("Score")]
  public float Score { get; set; }
}
```

Лістинг 2. Код класу «ModelsForm»

```
using EmployeeAttritionPredictionApp.AppCode;
using EmployeeAttritionPredictionApp.Forms.Systems;
using EmployeeAttritionPredictionApp.Providers;
using Microsoft.ML;
using Microsoft.ML.Data;
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 System.Globalization;
using System.IO;
using System.Diagnostics;

namespace EmployeeAttritionPredictionApp.Forms.SysMS {
  public partial class ModelsForm : Form {
    private MLContext mlContext;
    ITransformer model;
```

```
private IDataView dataView;
private string _Path = "";

private int _selectedRowIndex = 0;
private ValidationMy _Validation = new ValidationMy();
private ModelsProvider _ModelsProvider = new ModelsProvider();
private List<Models> _ModelsList = new List<Models>();
private LogsProvider _LogsProvider = new LogsProvider();
private bool _IsModelTrain = false;
private Random _rand = new Random();

public ModelsForm() {
  InitializeComponent();
  DataLoad();
}

private void OpenBtn_Click(object sender, EventArgs e) {
  // Створення діалогового вікна для відкриття файлу
  OpenFileDialog openFileDialog = new OpenFileDialog {
    Filter = "Text files (*.csv)|*.csv|All files (*.*)|*.*",
    FilterIndex = 2,
    RestoreDirectory = true
  };
  if (openFileDialog.ShowDialog() == DialogResult.OK) {
    _Path = openFileDialog.FileName;
    FileNameTBox.Text = openFileDialog.FileName;


    // Ініціалізація MLContext
    mlContext = new MLContext(seed: 123);
    // Завантаження даних
    try {
      // Тепер CSV має в другій колонці (Attrition) значення 1 або 0
      dataView = mlContext.Data.LoadFromTextFile<EmployeeData>(
          _Path,
          hasHeader: true,
          separatorChar: ',',
          allowQuoting: true
      );
    } catch (Exception ex) {
      RaportTBox.Text +=($"Помилка зчитування CSV: {ex.Message}");
      return;
    }


    RaportTBox.Text = "Завантаження даних\r\n";
    Application.DoEvents();

    // Розділяємо на train/test
    var split =
      mlContext.Data.TrainTestSplit(dataView, testFraction: 0.2, seed: 123);
    var trainData = split.TrainSet;
```

```
        var testData = split.TestSet;

        // Конвеєр побудови
        var pipeline = mlContext.Transforms.Conversion.ConvertType(
                inputColumnName: nameof(EmployeeData.Attrition),  // "Attrition"
                outputColumnName: "Label",                        // створюємо нову колонку Label типу
bool
                outputKind: DataKind.Boolean
            )
            // Далі кодуємо категоріальні стовпці
            .Append(mlContext.Transforms.Categorical.OneHotEncoding(new[]
            {
        new InputOutputColumnPair(nameof(EmployeeData.BusinessTravel)),
        new InputOutputColumnPair(nameof(EmployeeData.Department)),
        new InputOutputColumnPair(nameof(EmployeeData.EducationField)),
        new InputOutputColumnPair(nameof(EmployeeData.Gender)),
        new InputOutputColumnPair(nameof(EmployeeData.JobRole)),
        new InputOutputColumnPair(nameof(EmployeeData.MaritalStatus)),
        new InputOutputColumnPair(nameof(EmployeeData.Over18)),
        new InputOutputColumnPair(nameof(EmployeeData.OverTime))
            }))
            // Далі об’єднуємо усі фічі
            .Append(mlContext.Transforms.Concatenate("Features",
                nameof(EmployeeData.Age),
                nameof(EmployeeData.BusinessTravel), nameof(EmployeeData.Department),
                nameof(EmployeeData.DailyRate), nameof(EmployeeData.DistanceFromHome),
                nameof(EmployeeData.Education), nameof(EmployeeData.EducationField),
                nameof(EmployeeData.EmployeeCount), // EmployeeNumber - ідентифікатор
                nameof(EmployeeData.EnvironmentSatisfaction), nameof(EmployeeData.Gender),
                nameof(EmployeeData.HourlyRate), nameof(EmployeeData.JobInvolvement),
                nameof(EmployeeData.JobLevel), nameof(EmployeeData.JobRole),
                nameof(EmployeeData.JobSatisfaction), nameof(EmployeeData.MaritalStatus),
                nameof(EmployeeData.MonthlyIncome), nameof(EmployeeData.MonthlyRate),
                nameof(EmployeeData.NumCompaniesWorked), nameof(EmployeeData.Over18),
                nameof(EmployeeData.OverTime), nameof(EmployeeData.PercentSalaryHike),
                nameof(EmployeeData.PerformanceRating),
nameof(EmployeeData.RelationshipSatisfaction),
                nameof(EmployeeData.StandardHours), nameof(EmployeeData.StockOptionLevel),
                nameof(EmployeeData.TotalWorkingYears),
nameof(EmployeeData.TrainingTimesLastYear),
                nameof(EmployeeData.WorkLifeBalance), nameof(EmployeeData.YearsAtCompany),
                nameof(EmployeeData.YearsInCurrentRole),
nameof(EmployeeData.YearsSinceLastPromotion),
                nameof(EmployeeData.YearsWithCurrManager)
            ))
            // 3) Вказуємо бінарний тренер, labelColumnName = "Label" (типу bool)
            .Append(mlContext.BinaryClassification.Trainers.SdcaLogisticRegression(
                labelColumnName: "Label",
                featureColumnName: "Features"
            ));
        // Навчаємо
        RaportTBox.Text += ("Розпочато навчання моделі...\r\n");
```

```
        model = pipeline.Fit(trainData);

        // Оцінимо
        RaportTBox.Text += ("Оцінювання моделі на тестових даних...\r\n");
        var predictions = model.Transform(testData);
        var metrics = mlContext.BinaryClassification.Evaluate(
            predictions,
            labelColumnName: nameof(EmployeeData.Attrition)
        );
        RaportTBox.Text += ($"Точність (Accuracy): {metrics.Accuracy:P2}\r\n");
        RaportTBox.Text += ($"AUC: {metrics.AreaUnderRocCurve:P2}\r\n");
        RaportTBox.Text += ($"F1 Score: {metrics.F1Score:P2} \r\n");


        RaportTBox.SelectionStart = RaportTBox.Text.Length;
        RaportTBox.ScrollToCaret();

        _IsModelTrain = true;
      }
    }



    private void ModelsGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.ColumnIndex == 5 && ModelsGridView[0, e.RowIndex].Value.ToString() !=
_ModelsList[0].Message) {
        if (MessageBox.Show("Ви дійсно хочете видалити цю модель?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
          _ModelsProvider.DeleteModelsByModelsId(Convert.ToInt32(ModelsGridView[0,
e.RowIndex].Value.ToString()));
          DataLoad();
        }
      }
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        //Зберігання моделі
        string pathName = @"\teach\" + GenerateFileName() + ".zip";
        string localProj =

System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Loca
tion);
        _ModelsProvider.InsertModels(ModelsNamesTBox.Text, pathName);
        mlContext.Model.Save(model, dataView.Schema, localProj + pathName);
        ClearAllData();
        _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
          "Було навчено модель " +
          ModelsNamesTBox.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;
  ModelsNamesTBox.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 модель!", "Увага!");
    isCorrect = false;
  }
  if (_Validation.IsDataEntering(ModelsNamesTBox.Text)) {
    ModelsNamesValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    ModelsNamesValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  return isCorrect;
}

private void DataLoad() {
  int firstRowIndex = 0;
  if (ModelsGridView.FirstDisplayedScrollingRowIndex > 0) {
    firstRowIndex = ModelsGridView.FirstDisplayedScrollingRowIndex;
  }
  try {
    _ModelsList = _ModelsProvider.GetAllModels();
    LoadDataInModelsGridView(_ModelsList);
    if (_selectedRowIndex == ModelsGridView.Rows.Count) {
      _selectedRowIndex = ModelsGridView.Rows.Count - 1;
    }
    if (_selectedRowIndex >= 0) {
```

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


  }
}
```

Лістинг 3. Код класу «PredictForm»

```
using EmployeeAttritionPredictionApp.AppCode;
using EmployeeAttritionPredictionApp.Forms.Systems;
using EmployeeAttritionPredictionApp.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;

namespace EmployeeAttritionPredictionApp.Forms.Controls {
  public partial class PredictForm : Form {
    private ValidationMy _Validation = new ValidationMy();

    private Models _SelectedModels = new Models();
    private MLContext mlContext = new MLContext();
    private PredictionEngine<EmployeeData, EmployeePrediction> predictionEngine;
    private ModelsProvider _ModelsProvider = new ModelsProvider();
    private List<Models> _ModelsList = new List<Models>();
    private bool _IsModelsLoad = false;
    private LogsProvider _LogsProvider = new LogsProvider();


    string filePath = "data.csv";
    private List<EmployeeData> _UserActivityData = new List<EmployeeData>();

    public PredictForm() {
      InitializeComponent();
      LoadAllData();
    }

    private void ModelsCBox_SelectedValueChanged(object sender, EventArgs e) {
      if (_IsModelsLoad && IsModelExist()) {
        _SelectedModels = _ModelsProvider.SelectedModelsByModelsId(
          Convert.ToInt32(ModelsCBox.SelectedValue));
```

```
    LoadData(_SelectedModels.ModelsFileModel);
  }
}

private void LoadAllData() {
  BusinessTravelCBox.SelectedIndex = 0;
  DepartmentCBox.SelectedIndex = 0;
  EducationFieldCBox.SelectedIndex = 0;
  GenderCBox.SelectedIndex = 0;
  JobRoleCBox.SelectedIndex = 0;
  MaritalStatusCBox.SelectedIndex = 0;
  _ModelsList = _ModelsProvider.GetAllModels();
  ModelsCBox.DataSource = _ModelsList;
  ModelsCBox.ValueMember = "ModelsId";
  ModelsCBox.DisplayMember = "ModelsName";
  _IsModelsLoad = true;
  ModelsCBox_SelectedValueChanged(ModelsCBox, EventArgs.Empty);
}

private void LoadData(string FilePath) {
  string localProj = Application.StartupPath + FilePath;
  // Визначення DataViewSchema для конвеєра
  // підготовки даних і навченої моделі
  DataViewSchema modelSchema;
  // Завантаження моделі
  ITransformer model = mlContext.Model.Load(localProj,
    out modelSchema);
  // Створення механізму прогнозування
  predictionEngine =
      mlContext.Model.CreatePredictionEngine<EmployeeData,
      EmployeePrediction>(model);
}



private bool IsModelExist() {
  bool isCorrect = true;
  if (Convert.ToInt32(ModelsCBox.SelectedValue) > 0) {
    ModelsValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
  } else {
    ModelsValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
    isCorrect = false;
  }
  return isCorrect;
}

private void PredictBtn_Click(object sender, EventArgs e) {
  // Перевірка, чи введені всі необхідні дані та чи існує модель
  if (IsAllEmployeeDataCorrect() && IsModelExist()) {
    // Створення об'єкта EmployeeData з введених значень
    EmployeeData employeeData = new EmployeeData {
      Age = float.Parse(AgeTBox.Text),
```

```
  Attrition = 0, // Значення буде прогнозуватися моделлю
  BusinessTravel = BusinessTravelCBox.Text,
  DailyRate = float.Parse(DailyRateTBox.Text),
  Department = DepartmentCBox.Text,
  DistanceFromHome = float.Parse(DistanceFromHomeTBox.Text),
  Education = float.Parse(EducationTBox.Text),
  EducationField = EducationFieldCBox.Text,
  EmployeeCount = float.Parse(EmployeeCountTBox.Text),
  EmployeeNumber = float.Parse(EmployeeNumberTBox.Text),
  EnvironmentSatisfaction = float.Parse(EnvironmentSatisfactionTBox.Text),
  Gender = GenderCBox.Text,
  HourlyRate = float.Parse(HourlyRateTBox.Text),
  JobInvolvement = float.Parse(JobInvolvementTBox.Text),
  JobLevel = float.Parse(JobLevelTBox.Text),
  JobRole = JobRoleCBox.Text,
  JobSatisfaction = float.Parse(JobSatisfactionTBox.Text),
  MaritalStatus = MaritalStatusCBox.Text,
  MonthlyIncome = float.Parse(MonthlyIncomeTBox.Text),
  MonthlyRate = float.Parse(MonthlyRateTBox.Text),
  NumCompaniesWorked = float.Parse(NumCompaniesWorkedTBox.Text),
  Over18 = "Y",
  OverTime = OverTimeChBox.Checked ? "1" : "0",
  PercentSalaryHike = float.Parse(RelationshipSatisfactionTBox.Text),
  PerformanceRating = float.Parse(PerformanceRatingTBox.Text),
  RelationshipSatisfaction = float.Parse(RelationshipSatisfactionTBox.Text),
  StandardHours = float.Parse(StandardHoursTBox.Text),
  StockOptionLevel = float.Parse(StockOptionLevelTBox.Text),
  TotalWorkingYears = float.Parse(TotalWorkingYearsTBox.Text),
  TrainingTimesLastYear = float.Parse(TrainingTimesLastYearTBox.Text),
  WorkLifeBalance = float.Parse(WorkLifeBalanceTbox.Text),
  YearsAtCompany = float.Parse(YearsAtCompanyTBox.Text),
  YearsInCurrentRole = float.Parse(YearsInCurrentRoleTBox.Text),
  YearsSinceLastPromotion = float.Parse(YearsSinceLastPromotionTBox.Text),
  YearsWithCurrManager = float.Parse(YearsWithCurrManagerTBox.Text)
};

MonitoringTBox.Clear();

// Відображення всіх даних з поясненнями українською мовою
var dataInfo = new StringBuilder();
dataInfo.AppendLine("Введені дані для прогнозування плинності кадрів:");

var fieldTranslations = new Dictionary<string, string>
{
        { "Age", "Вік" },
        { "Attrition", "Звільнення (прогноз)" },
        { "BusinessTravel", "Службові відрядження" },
        { "DailyRate", "Щоденна ставка" },
        { "Department", "Відділ" },
        { "DistanceFromHome", "Відстань до дому" },
        { "Education", "Рівень освіти" },
        { "EducationField", "Напрям освіти" },
```

```
            { "EmployeeCount", "Кількість працівників" },
            { "EmployeeNumber", "Номер працівника" },
            { "EnvironmentSatisfaction", "Задоволеність робочим середовищем" },
            { "Gender", "Стать" },
            { "HourlyRate", "Погодинна ставка" },
            { "JobInvolvement", "Залученість у роботу" },
            { "JobLevel", "Рівень посади" },
            { "JobRole", "Посада" },
            { "JobSatisfaction", "Задоволеність роботою" },
            { "MaritalStatus", "Сімейний стан" },
            { "MonthlyIncome", "Щомісячний дохід" },
            { "MonthlyRate", "Щомісячна ставка" },
            { "NumCompaniesWorked", "Кількість компаній, у яких працював" },
            { "Over18", "Старше 18 років" },
            { "OverTime", "Переробка" },
            { "PercentSalaryHike", "Підвищення зарплати (%)" },
            { "PerformanceRating", "Оцінка продуктивності" },
            { "RelationshipSatisfaction", "Задоволеність стосунками" },
            { "StandardHours", "Стандартні години" },
            { "StockOptionLevel", "Рівень акційної програми" },
            { "TotalWorkingYears", "Загальний досвід роботи (років)" },
            { "TrainingTimesLastYear", "Кількість тренінгів за рік" },
            { "WorkLifeBalance", "Баланс роботи та життя" },
            { "YearsAtCompany", "Роки у компанії" },
            { "YearsInCurrentRole", "Роки на поточній посаді" },
            { "YearsSinceLastPromotion", "Роки з останнього підвищення" },
            { "YearsWithCurrManager", "Роки з поточним менеджером" }
        };

      foreach (var property in typeof(EmployeeData).GetProperties()) {
        string propertyName = property.Name;
        string displayName = fieldTranslations.ContainsKey(propertyName) ?
fieldTranslations[propertyName] : propertyName;
        object value = property.GetValue(employeeData) ?? "(не вказано)";
        dataInfo.AppendLine($"{displayName}: {value}");
      }

      MonitoringTBox.Text = dataInfo.ToString();

      // Прогнозування на основі введених даних
      var prediction = predictionEngine.Predict(employeeData);
      // Формуємо результат прогнозування
      var answer = new StringBuilder();
      answer.AppendLine("\r\n--- Прогнозування ---");
      answer.AppendLine($"\r\nЙмовірність звільнення: " +
        $"{prediction.Probability:P2}");
      answer.AppendLine($"\r\nПрогнозований статус:" +
        $" {(prediction.PredictedLabel ? "Звільнення" : "Залишається")}");



      // Логування прогнозу
```

```
      _LogsProvider.InsertLogs(LoginForm.CurrentUser.UsersId,
         "Було проведено прогнозування плинності кадрів для моделі " + ModelsCBox.Text,
DateTime.Now);

      // Додавання результату прогнозування до текстового поля
      MonitoringTBox.Text += answer.ToString();
    }
  }


  private bool IsAllEmployeeDataCorrect() {
    bool isCorrect = true;
    if (_Validation.IsDataConvertToInt(AgeTBox.Text)) {
      AgeValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      AgeValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_Validation.IsDataConvertToDouble(DailyRateTBox.Text)) {
      DailyRateValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      DailyRateValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_Validation.IsDataConvertToDouble(DistanceFromHomeTBox.Text)) {
      DistanceFromHomeValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      DistanceFromHomeValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_Validation.IsDataInThisScope(1,5, EducationTBox.Text)) {
      EducationValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      EducationValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_Validation.IsDataConvertToInt(EmployeeCountTBox.Text)) {
      EmployeeCountValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      EmployeeCountValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_Validation.IsDataConvertToInt(EmployeeNumberTBox.Text)) {
      EmployeeNumberValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      EmployeeNumberValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    if (_Validation.IsDataConvertToInt(EmployeeNumberTBox.Text)) {
      EmployeeNumberValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
    } else {
      EmployeeNumberValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
```

```
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(MonthlyIncomeTBox.Text)) {
        MonthlyIncomeValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        MonthlyIncomeValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(MonthlyRateTBox.Text)) {
        MonthlyRateValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        MonthlyRateValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToInt(NumCompaniesWorkedTBox.Text)) {
        NumCompaniesWorkedValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        NumCompaniesWorkedValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(PercentSalaryHikeTBox.Text)) {
        PercentSalaryHikeValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PercentSalaryHikeValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataInThisScope(1,4, PerformanceRatingTBox.Text)) {
        PerformanceRatingValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PerformanceRatingValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataInThisScope(1,4, RelationshipSatisfactionTBox.Text)) {
        RelationshipSatisfactionValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        RelationshipSatisfactionValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataInThisScope(1, 4, EnvironmentSatisfactionTBox.Text)) {
        EnvironmentSatisfactionValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        EnvironmentSatisfactionValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToDouble(HourlyRateTBox.Text)) {
        HourlyRateValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        HourlyRateValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
```

```
        }
        if (_Validation.IsDataInThisScope(1, 4, JobInvolvementTBox.Text)) {
          JobInvolvementValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          EnvironmentSatisfactionValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_Validation.IsDataInThisScope(1, 4, JobLevelTBox.Text)) {
          JobLevelValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          JobLevelValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_Validation.IsDataInThisScope(1, 4, JobSatisfactionTBox.Text)) {
          JobSatisfactionValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          JobSatisfactionValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_Validation.IsDataConvertToDouble(StandardHoursTBox.Text)) {
          StandardHoursValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          StandardHoursValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_Validation.IsDataInThisScope(0, 3, StockOptionLevelTBox.Text)) {
          StockOptionLevelValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          StockOptionLevelValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_Validation.IsDataConvertToDouble(TotalWorkingYearsTBox.Text)) {
          TotalWorkingYearsValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          TotalWorkingYearsValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_Validation.IsDataConvertToDouble(TrainingTimesLastYearTBox.Text)) {
          TrainingTimesLastYearValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
        } else {
          TrainingTimesLastYearValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_Validation.IsDataInThisScope(0, 3, WorkLifeBalanceTbox.Text)) {
          WorkLifeBalanceValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
        } else {
          WorkLifeBalanceValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
          isCorrect = false;
        }
        if (_Validation.IsDataConvertToInt(YearsAtCompanyTBox.Text)) {
          YearsAtCompanyValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
```

```
      } else {
        YearsAtCompanyValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToInt(YearsInCurrentRoleTBox.Text)) {
        YearsInCurrentRoleValidationLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        YearsInCurrentRoleValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToInt(YearsSinceLastPromotionTBox.Text)) {
        YearsSinceLastPromotionValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        YearsSinceLastPromotionValidationLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToInt(YearsWithCurrManagerTBox.Text)) {
        YearsWithCurrManagerValidationLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        YearsWithCurrManagerValidationLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

  }
}
```