МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних
інформаційних систем
_____ Євген СІДЕНКО
«____»_____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ІНФОРМАЦІЙНА СИСТЕМА ПРОГНОЗУВАННЯ ПОПИТУ НА ПОСЛУГИ ТРАНСПОРТНОЇ КОМПАНІЇ

Спеціальність 122 Комп'ютерні науки

Освітня програма «Комп'ютерні науки»

Здобувач _____ Данило ГУЛЬКЕВИЧ
«___»_____ 2026 р.

Керівник д-р техн. наук, професор _____ Ірина КАЛІНІНА
«___»_____ 2026 р.

Миколаїв – 2026

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступень	Бакалавр
Спеціальність	122 Комп’ютерні науки
Освітня програма	Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем
_____ Євген СІДЕНКО
«____» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

ГУЛЬКЕВИЧ ДАНИЛО ДМИТРОВИЧ

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Інформаційна система прогнозування попиту на послуги транспортної компанії.

Керівник роботи: Калініна Ірина Олександрівна, професор кафедри ІІС, д-р техн. наук, професор.

Затверджена наказом ЧНУ ім. Петра Могили від «25» грудня 2025 р. № 353.

2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: Інформаційна система для аналізу та прогнозування попиту на послуги транспортної

компанії, реалізована з використанням сучасних засобів програмування; структурований набір даних, що містить інформацію про транспортні перевезення (дати, маршрути, типи послуг, обсяги перевезень, клієнтів та інші характеристики); результати моделювання попиту із застосуванням статистичних методів і моделей машинного навчання (зокрема лінійної регресії, моделей часових рядів, а також ансамблевих методів, таких як Random Forest); визначення ключових факторів, що впливають на попит, та оцінка їхнього впливу за допомогою методів аналізу даних і інтерпретації моделей.

4. Перелік питань, що підлягають розробці:

1) аналіз сучасних підходів до прогнозування попиту на послуги транспортних компаній та факторів, що на нього впливають;
2) огляд методів прогнозування попиту на основі статистичних підходів і машинного навчання;
3) визначення особливостей та підготовка набору даних про транспортні перевезення для проведення досліджень;
4) розробка інформаційної системи для прогнозування попиту із використанням методів лінійної регресії, моделей часових рядів та ансамблевих методів (зокрема Random Forest);
5) аналіз отриманих результатів та інтерпретація впливу основних факторів (часових, географічних, поведінкових) на попит;
6) порівняльна оцінка результатів прогнозування для визначення найбільш ефективних підходів до аналізу попиту на транспортні послуги;

5. Перелік графічних матеріалів: Презентація.

Керівник роботи _____ _Ірина КАЛІНІНА_
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ _Данило ГУЛЬКЕВИЧ_
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «23» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Інформаційна система прогнозування попиту на послуги транспортної компанії.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	21.12.2025	24.12.2025	виконано
2	Аналіз предметної області та постановка задачі	25.12.2025	30.01.2026	виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема огляд публікацій та аналогів щодо прогнозування попиту в транспортній галузі	31.01.2026	01.03.2026	виконано
4	Огляд існуючих методів машинного навчання та інформаційних технологій для прогнозування попиту (часові ряди, Prophet, Random Forest, XGBoost)	02.03.2026	01.04.2026	виконано
5	Реалізація системи та аналіз результатів моделювання попиту на послуги транспортної компанії	02.04.2026	24.04.2026	виконано
6	Перший попередній захист КР на засіданні комісії кафедри	25.05.2026	25.05.2026	виконано
7	Корегування роботи за результатами попереднього захисту	26.05.2026	04.06.2026	виконано
8	Другий попередній захист КР на засіданні комісії кафедри	05.06.2026	05.06.2026	виконано
9	Доробка та остаточне оформлення КР	06.06.2026	14.06.2026	
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.06.2026	19.06.2026	

Керівник роботи _____ _____

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ _____

(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«____» _____ 2026___ р.

АНОТАЦІЯ

до кваліфікаційної роботи

Здобувач освіти 401 гр. ЧНУ ім. Петра Могили

Гулькевич Данило Дмитрович

На тему: **“ІНФОРМАЦІЙНА СИСТЕМА ПРОГНОЗУВАННЯ ПОПИТУ НА ПОСЛУГИ ТРАНСПОРТНОЇ КОМПАНІЇ”**

Актуальність даного дослідження полягає в необхідності створення ефективних інструментів для аналізу та прогнозування попиту на транспортні послуги, що дозволяють підвищити ефективність управління ресурсами, оптимізувати маршрути та обґрунтовувати управлінські рішення, особливо в умовах високої конкуренції, сезонності та динамічних змін ринку пасажирських перевезень в Індії.

Об’єктом роботи є процеси формування попиту на транспортні послуги на основі історичних даних про перевезення. До об’єкта належать дані про маршрути, дати перевезень, обсяги замовлень та інші характеристики, що впливають на попит.

Предметом роботи є методи та засоби побудови інформаційної системи для аналізу та прогнозування попиту на транспортні послуги.

Мета роботи є розробка інформаційної системи, що на основі структурованого набору даних і методів машинного навчання (лінійна регресія, моделі часових рядів, Random Forest) дозволяє прогнозувати попит на транспортні послуги та досліджувати вплив різних факторів на його зміну.

В результаті виконання роботи було реалізовано інформаційну систему для аналізу та прогнозування попиту, що включає обробку даних, побудову моделей машинного навчання та візуалізацію результатів. Проведено аналіз залежності попиту від часових, географічних та поведінкових факторів, а також виконано порівняння ефективності використаних моделей за показниками точності (RMSE, R^2). Отримані

результати дозволяють визначити найбільш впливові фактори та підвищити точність прогнозування попиту.

Дана робота складається з чотирьох розділів. У першому розділі проведено аналіз предметної області, огляд існуючих підходів та сформовано постановку задачі. Другий розділ присвячений методам і моделям прогнозування попиту. У третьому розділі наведено розробку інформаційної системи та реалізацію алгоритмів прогнозування. У четвертому – аналіз отриманих результатів, тестування та візуалізацію даних.

Загальний обсяг роботи – 85 сторінок. Кваліфікаційна робота містить 3 додатки, 24 рисунків, 3 таблиці і 31 джерел посилання.

Ключові слова: інформаційна система, прогнозування попиту, транспортні послуги, машинне навчання, лінійна регресія, Random Forest, аналіз даних.

ABSTRACT

to the qualification work by the student of the group 401 of Petro Mohyla Black Sea National University

Hulkevych Danylo

“INFORMATION SYSTEM FOR FORECASTING DEMAND FOR TRANSPORT COMPANY SERVICES”

The relevance of this study lies in the need to develop effective tools for analyzing historical transportation data and forecasting demand for transport services. This enables optimization of resource planning, routing, and managerial decision-making under conditions of high competition, strong seasonality, and market volatility in India’s intercity bus transport sector.

The object of the work is the processes of demand formation for transport company services based on historical order and search data. The subject is the methods and tools for building an information system for demand forecasting.

The purpose of the thesis is to design an information system that, using a structured dataset and machine learning/time series methods (linear regression, time series models, Random Forest), predicts demand volume and investigates the influence of key factors (time-related, geographical, behavioral) on transport service demand.

Key results include: a Python-based desktop application built with PySide6 for interactive demand analysis and forecasting; implementation of three models (Prophet, Linear Regression, Random Forest); experimental validation of the models on the Indian bus transportation dataset in terms of MAE, RMSE, and MAPE; and analysis of the most influential factors affecting demand (days before departure, day of week, region, seasonality)

The overall scope of the work is 85 pages. The thesis contains 3 appendices, 24 figures, 3 tables and 31 references.

Key words: information system, demand forecasting, transport services, machine learning, linear regression, Random Forest, time series, Shiny applicatio

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

IC – Information system (інформаційна система)

ML – Machine Learning (машинне навчання)

AI – Artificial Intelligence (штучний інтелект)

RMSE – Root Mean Square Error (середньоквадратична помилка)

MAE – Mean Absolute Error (середня абсолютна помилка)

TS – Time Series (часові ряди)

R^2 – Coefficient of determination

RF – Random Forest (випадковий ліс)

LR – Linear Regression (лінійна регресія)

UI – User Interface (інтерфейс користувача)

Декларація про використанні ШІ. Під час підготовки наукової роботи (академічного тексту) було використано інструмент Grok. Відповідно до таксономії GAIDeT (2025), наведені нижче завдання були делеговані інструментам генеративного ШІ за повного людського нагляду:

- пошук і систематизація літератури;
- вичитування та редагування;
- очищення даних;
- генерація структур таблиць;
- формулювання підсумків.

Повну відповідальність за фінальний рукопис несе автор.

Інструменти генеративного ШІ не зазначаються як автори та не несуть відповідальності за кінцеві результати.

Декларацію подав: Гулькевич Данило Дмитрович

ВСТУП

Сучасні виклики в транспортній галузі Індії, зокрема стрімке зростання пасажиропотоку, висока конкуренція між перевізниками, сезонні коливання попиту, підвищення вартості пального, релігійні фестивалі та значне навантаження на транспортну інфраструктуру, вимагають впровадження ефективних цифрових інструментів для аналізу даних і прогнозування. Транспортні компанії Індії стикаються з необхідністю точного планування ресурсів, оптимального розподілу автобусів, коригування маршрутів та прийняття оперативних управлінських рішень на основі реальних даних про попит.

Одним із пріоритетних напрямів є використання методів аналізу даних і машинного навчання для прогнозування попиту на транспортні послуги. У світі активно розробляються та впроваджуються інформаційні системи, які дозволяють аналізувати історичні дані замовлень і пошукових запитів, виявляти сезонні закономірності та прогнозувати майбутнє навантаження. Попри наявність певних напрацювань у глобальній практиці, більшість рішень залишаються комерційними, закритими або недостатньо адаптованими до специфіки індійського ринку. Крім того, не всі системи забезпечують інтерпретовані результати, що ускладнює прийняття обґрунтованих рішень керівництвом транспортних компаній.

Провідні дослідницькі групи та компанії активно застосовують відкриті датасети та сучасні методи часових рядів (ARIMA, Prophet) і машинного навчання (Random Forest, XGBoost) для задач прогнозування попиту в логістиці та транспорті. В індійському академічному середовищі такі інструменти все ще недостатньо представлені, особливо у формі відкритих, доступних і зрозумілих інформаційних систем.

Актуальність цієї роботи визначається потребою у створенні простої, відкритої та інтерпретованої інформаційної системи для прогнозування попиту на транспортні

послуги. Розроблена система дозволить аналізувати історичні дані перевезень, прогнозувати майбутній попит, оптимізувати планування транспортних ресурсів та підвищувати ефективність діяльності транспортних компаній. Основою реалізації обрано мову програмування Python, яка поєднує потужні засоби статистичного аналізу, моделювання часових рядів, машинного навчання та створення графічного інтерфейсу користувача.

Дослідження тісно пов’язане з науковими роботами, присвяченими застосуванню методів аналізу часових рядів і алгоритмів машинного навчання для прогнозування попиту в транспортній галузі. Робота логічно доповнює існуючі дослідження та пропонує практичний програмний засіб, адаптований до умов індійського ринку транспортних послуг.

Об’єктом роботи є процес формування та зміни попиту на транспортні послуги.

Предметом роботи є методи, моделі та програмні засоби прогнозування попиту на транспортні послуги із застосуванням алгоритмів машинного навчання та аналізу часових рядів.

Метою роботи є розробка інформаційної системи прогнозування попиту на транспортні послуги на основі методів аналізу часових рядів та машинного навчання для підвищення ефективності планування транспортних ресурсів і підтримки прийняття управлінських рішень.

1 АНАЛІЗ ПІДХОДІВ ДО РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ ПОПИТУ НА ПОСЛУГИ ТРАНСПОРТНОЇ КОМПАНІЇ

1.1 Огляд літератури та існуючих підходів

Сфера поїздок на автобусах для людей змінюється швидко. Тому треба добре знати, скільки пасажирів хочуть їхати кожного дня чи тижня. Це важливо, бо компанія має рахувати свої гроші, обирати, куди і коли їхати, планувати роботу й знати, скільки грошей можна отримати у різних містах і кожному районі. Люди їздять у різний час, бо попит на автобус змінюється протягом року, у різні сезони, на свята, навіть у різні дні тижня. Крім цього, що буде завтра, залежить від того, скільки грошей є у людей, маршрутів, погоди, свят, дощу і людського потоку в країні.

Компанія, яка везе людей на автобусі за містом або між містами, і робить це часто чи інколи, — це транспортна компанія. Успіх фірми часто йде від того, як добре вона може знати, скільки товару люди хочуть, і як швидко вона може змінити свої дії, якщо потрібно. Якщо фірма бачить зміни у попиті, вона встигає зробити все вчасно й дати людям те, що їм треба. Так фірма працює краще і зростає. Усе залежить від вміння вчасно помітити й відповісти на новий попит.

У сучасній науковій літературі активно досліджуються методи прогнозування попиту на пасажирські перевезення. Серед моделей часових рядів широко застосовується Prophet (Meta), який добре справляється з сезонністю, трендами та впливом святкових днів [1]. Hernandez et al. (2025) використали Prophet для короткострокового прогнозування попиту пасажирів у системі громадського транспорту з подальшою оптимізацією диспетчеризації автобусів, підкреслюючи інтерпретованість моделі та її здатність враховувати holiday effects [2].

Ансамблеві методи машинного навчання дають дуже точні прогнози попиту [3]. Побавчивши, що Patel et al. (2024) довели, що boosting-техніки (XGBoost, LightGBM,

CatBoost) працюють краще за традиційні статистичні методи, коли треба швидко передбачити пасажиропотік. Також помітно, що Spanos et al. (2025) створили Principal Component Random Forest (PCRF) і використали його для прогнозу попиту в транспортних системах.

Коли треба робити короткострокове прогнозування, часто беруть у роботу глибокі нейронні мережі. Siswanto et al. (2024) застосували LSTM-моделі, щоб передбачити кількість пасажирів у автобусних операторів. Спостерігавши, що Siswanto et al. (2024) отримали високу точність, бо LSTM вловлює складні нелінійні залежності [4].

Лінійна регресія все ще використовується як базова модель, бо її легко зрозуміти. Лінійна регресія особливо корисна, коли її комбінують з іншими методами для порівняння [5].

Таким чином, сучасні дослідження демонструють тенденцію до гібридних підходів, що поєднують переваги часових рядів (Prophet), ансамблевих методів (Random Forest, XGBoost) та глибокого навчання. Водночас більшість робіт фокусується на міському транспорті або великих агрегованих даних. Менше уваги приділяється інтерпретованому аналізу факторів впливу саме для міжміських автобусних перевезень в умовах індійського ринку. Це обумовлює актуальність розробки комплексної інформаційної системи з високою інтерпретованістю результатів.

1.2 Аналіз існуючих інформаційних систем та їх характеристика

Сьогодні на ринку доступний широкий спектр програмних продуктів для управління транспортними перевезеннями та аналізу попиту. В Індії, зокрема, активно використовуються системи бронювання квитків, корпоративні CRM-платформи [6] та логістичні модулі ERP-систем. Асоціації перевізників та експертні огляди регулярно

проводять аналіз таких рішень.

Рисунок 1.1 – Узагальнена архітектура інформаційної системи прогнозування попиту на транспортні послуги

На рис. 1.1 показано узагальнену архітектуру інформаційної системи прогнозування попиту в транспортній галузі. Дані про бронювання та пошукові запити можуть надходити автоматично з систем продажу квитків або вводитися вручну оператором. Модуль збору даних накопичує інформацію у центральному сховищі (базі даних); паралельно йде початкова робота з даними — це перевірка, чистка та збір в одне ціле. Тут дуже важливий блок, який сам слідкує за даними. Він бачить, якщо є не ті числа, якщо щось пропущене або ті самі числа з’явились двічі. Далі дані йдуть до блоку, котрий рахує прогнози, бачить, який буде попит, і порівнює їх з реальними цифрами. Там же можна знайти зв’язки між тим, що сталося. Для людей, які користуються цією системою, інформацію видно через блок, що малює

графіки, дашборди й схеми з картами. Є ще блок для звітів: він робить прості звіти і ті, які можна змінювати під свої потреби. Ще один блок допомагає керувати всім: додати або видалити людей, змінити важливі налаштування або зберегти дані на всяк випадок. Модуль для керування дає змогу керувати людьми в системі, змінювати важливі параметри й зберігати дані. Така будова має кілька рівнів, що відповідає головним вимогам сучасних IT-систем. Тут кожна частина має свою роль, але разом вони формують ціле. Частини легко міняти чи змінювати; система відкрита й може працювати з іншими.

Як приклад тут можна навести декілька систем і сервісів, які працюють не тільки в Індії, але і в інших країнах світу. В таблиці 1.1 видно порівняння головних видів рішень за важливими критеріями, які важко не взяти до уваги. На цій таблиці є такі пункти: як працюють ці програми, наскільки точний прогноз, звідки система бере дані, які саме технології вона використовує, і чи відкритий код для всіх.

Крім цього треба сказати, що зараз системи, які мають передбачити попит, часто застосовують нові шляхи. Для прикладу, машинне навчання і аналіз великих масивів даних допомагають врахувати більше різних факторів.

Таблиця 1.1 – Порівняння існуючих інформаційних систем та підходів до прогнозування попиту в транспорті

Назва системи	**Використані алгоритми**	**Точність прогнозу**	**Інтерпретованість**	**Джерела даних**	**Технології реалізації**
Busfor, Infobus [7]	Базова статистика, прості евристичні правила	Середня	Низька	Дані бронювань і пошукових запитів	Веб-платформи SaaS
Корпоративні CRM та ERP-модулі	Статистичний аналіз, базові ML-моделі	Середня-висока	Низька	Внутрішні бази даних компанії	.NET, SAP, Oracle, 1C [8]

Кінець таблиці 1.1

Назва системи	Використані алгоритми	Точність прогнозу	Інтерпретованість	Джерела даних	Технології реалізації
Аналітичні платформи (Power BI, Tableau)	Візуалізація + базовий вбудований прогноз	Низька-середня	Середня	Імпорт даних з різних джерел	Хмарні сервіси, BI-інструменти Частково
Open-source рішення (Prophet + R/Python)	Prophet, Arima, Random Forest, XGBoost	Висока	Середня-висока	Історичні дані бронювань	R, Python, Shiny, Streamlit

Згідно з таблицею 1.1, як сервіси бронювання, так і ERP-системи можуть ефективно враховувати бронювання, але не забезпечують багато можливостей для середньострокового та довгострокового прогнозування попиту та аналізу факторів, що на нього впливають. Більшість платформ бізнес-аналітики забезпечують надійну візуалізацію даних, але обмежені у своїй здатності прогнозувати майбутні події. Рішення з відкритим кодом, яке використовує Prophet [9], Random Forest [10], XGBoost, SHAP та інші подібні інструменти, забезпечує найточніше та найгнучкіше рішення. Тим не менш, практично немає готових рішень, адаптованих для міжміських автобусних перевезень в Індії.

Аналітичні платформи Power BI та Tableau добре справляються з візуалізацією даних і побудовою дашбордів, але вимагають ручного створення моделей прогнозування або інтеграції з зовнішніми інструментами. Open-source рішення на базі Prophet, Random Forest чи XGBoost дають високу точність, але потребують значних технічних знань для впровадження та підтримки.

Тож, нині схожі системи дають змогу керувати різними частинами руху машин — від простого підрахунку і збору даних до звичайної перевірки. Але жодна з цих систем не дає змоги повністю і добре зрозуміти все, що треба для глибокого вивчення

попиту. Майже всі продукти для бізнесу мають лише спостереження і створення звітів. Дещо бракує таких важливих дій, як точний прогноз, розумне вивчення (чому змінюється попит), а також пояснення, які речі впливають на попит, або це зроблено мало, або зовсім нема таких функцій.

Аналіз показує, що наявні рішення лише частково задовольняють потреби транспортних компаній. Таким чином, необхідність нової розробки інформаційної системи зумовлена виявленими прогалинами: потрібна система, яка об’єднає функції збору та аналізу історичних даних, сучасні методи машинного навчання, інтерпретацію результатів і зручний інтерактивний інтерфейс.

1.3 Постанова задачі

Актуальність теми. На основі проведеного аналізу існуючих підходів та інформаційних систем встановлено необхідність розробки інформаційної системи прогнозування попиту на транспортні послуги, яка поєднуватиме методи аналізу часових рядів і машинного навчання та забезпечуватиме можливість аналізу факторів впливу на попит.

Мета роботи. Метою роботи є розробка інформаційної системи прогнозування попиту на транспортні послуги на основі методів аналізу часових рядів та машинного навчання, що забезпечує підвищення ефективності планування транспортних ресурсів і підтримку прийняття управлінських рішень.

Іншими словами, необхідно створити комплекс програмних та аналітичних засобів, який на вході отримує дані про бронювання, пошукові запити та інші фактори, а на виході формує прогноз попиту на транспортні послуги та надає інструменти для аналізу впливу різних факторів.

Об’єкт роботи. Об’єктом дослідження є процес формування та зміни попиту на транспортні послуги.

Предмет роботи. Предметом дослідження є методи та засоби побудови інформаційної системи прогнозування попиту на транспортні послуги. Це включає: структурований набір даних про бронювання квитків, пошукові запити користувачів та інші фактори, що впливають на попит (зокрема, історичні дані перевезень як основу для аналізу), алгоритми машинного навчання (регресійного, класифікаційного аналізу, а також методи аналізу часових рядів), що використовуються для виявлення закономірностей і формування прогнозів попиту, а також програмні засоби для реалізації цих алгоритмів та візуалізації результатів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– провести огляд сучасних методів прогнозування попиту на транспортні послуги з акцентом на статистичні моделі, моделі часових рядів (Prophet) та ансамблеві методи машинного навчання (Random Forest, XGBoost);

– обґрунтувати вибір мови програмування Python [11] та середовища розробки PyCharm [12] для створення інформаційної системи;

– підготувати та проаналізувати структурований набір даних про бронювання та міжміські автобусні перевезення (дати, маршрути, обсяги, клієнти та інші характеристики) [13];

– реалізувати та порівняти кілька моделей прогнозування попиту: лінійну регресію, модель Prophet та ансамблевий метод Random Forest;

– проаналізувати вплив часових, географічних та поведінкових факторів на попит і надати рекомендації щодо їх урахування в управлінні перевезеннями;

– виконати експериментальне тестування моделей на реальних даних, оцінити їх точність за метриками RMSE, MAE та R^2 і визначити найбільш ефективний підхід.

Методи роботи. У роботі застосовуються методи статистичного аналізу даних, аналізу часових рядів, машинного навчання (регресійні та ансамблеві моделі).

Реалізація інформаційної системи виконана мовою програмування Python з використанням пакетів pandas [14], matplotlib [15] та seaborn [16], scikit-learn [17], prophet та інших бібліотек.

1.4 Аналіз факторів, що впливають на попит на послуги транспортної компанії

Формування попиту на послуги транспортної компанії є складним динамічним процесом, що залежить від сукупності взаємопов'язаних факторів різної природи. Врахування цих факторів є необхідною умовою побудови адекватних моделей прогнозування, оскільки вони визначають структуру даних та впливають на точність отриманих результатів.

З метою систематизації фактори, що впливають на попит, доцільно поділити на часові, соціально-економічні, операційні та зовнішні.

До **часових факторів** належать тренд та сезонність попиту. У транспортній сфері спостерігаються як короткострокові коливання (добові та тижневі), так і довгострокові тенденції розвитку. Дані фактори є ключовими для моделювання за допомогою методів аналізу часових рядів. Зокрема, модель Prophet дозволяє автоматично враховувати трендову складову, сезонні коливання різних рівнів та вплив календарних подій, що забезпечує високу точність прогнозування у задачах транспортного попиту.

Соціально-економічні фактори включають рівень доходів населення, зайнятість, рівень урбанізації та загальний стан економіки. Дані фактори можуть бути враховані у вигляді додаткових ознак (features) при побудові моделей машинного навчання, що дозволяє підвищити якість прогнозу за рахунок урахування зовнішніх залежностей [20].

До **операційних факторів** належать внутрішні характеристики діяльності транспортної компанії, зокрема тарифна політика, якість обслуговування, доступність транспорту та ефективність логістичних процесів. Включення таких параметрів у модель дозволяє більш точно відобразити реальні умови функціонування системи.

Зовнішні фактори включають погодні умови, дорожню ситуацію, рівень конкуренції та наявність альтернативних видів транспорту. Ці фактори можуть мати як прямий, так і опосередкований вплив на попит, тому їх врахування є важливим для побудови комплексних моделей прогнозування.

З точки зору методів машинного навчання, зазначені фактори можуть бути представлені у вигляді набору вхідних ознак. Зокрема, у моделях, побудованих із використанням алгоритму **Random Forest**, можливе одночасне врахування великої кількості факторів різної природи. Даний підхід дозволяє виявляти складні нелінійні залежності між змінними та підвищувати точність прогнозування у порівнянні з простішими моделями.

Таким чином, у процесі розробки інформаційної системи доцільним є комбіноване використання підходів до прогнозування: застосування моделі Prophet для аналізу часових залежностей та сезонних коливань, а також алгоритму Random Forest для врахування багатофакторного впливу на попит. Такий підхід дозволяє отримати більш точні та стійкі результати прогнозування.

Висновки до розділу 1

У першому розділі проаналізувавши сферу пасажирських автобусних перевезень в Індії. Визначивши, які фактори впливають на попит на транспорт. Попит виявився дуже динамічним. Попит змінюється через сезонність, через календарні події — свят і фестивалів, через географічні особливості маршрутів, через економічні

фактори, через погодні умови, зокрема мусонний сезон, і через внутрішню міграцію населення.

Читавши нову літературу і бачивши, що дослідники використовують моделі часу, такі як Prophet, і комбінації методів машинного навчання, наприклад Random Forest і XGBoost, а також глибокі нейронки LSTM, щоб передбачити попит. Більшість досліджень зосереджені на міському транспорті. А питання передбачити попит на міжміські автобуси в індійському ринку ще не вивчене.

Переглянувши інформаційні системи (сервіси бронювання Busfor, Infobus, корпоративні ERP-системи, платформи Power BI та Tableau). Аналіз виявив недоліки інформаційних систем: результати важко зрозуміти, складно передбачити майбутнє і не досить аналізують, що впливає.

Отже, аналіз показує, що потрібна відкрита інформаційна система на Python. Система на Python поєднуватиме лінійну регресію, модель Prophet і метод Random Forest. У розділі сформовано об’єкт, сформовано предмет, сформовано мету і сформовано основні завдання дослідження. Формулювання створюють надійну основу для подальшої розробки системи.

2 МЕТОДИ ТА МОДЕЛІ ПРОГНОЗУВАННЯ ПОПИТУ НА ТРАНСПОРТНІ ПОСЛУГИ

2.1 Загальна характеристика методів прогнозування попиту

Прогнозування попиту на транспортні послуги є складною задачею, що передбачає аналіз історичних даних та виявлення закономірностей зміни попиту під впливом різноманітних факторів. До таких факторів належать часові характеристики (дата, день тижня, сезонність), географічні параметри (маршрути перевезень), а також поведінкові та соціально-економічні аспекти.

Сучасні підходи до прогнозування попиту можна умовно поділити на три основні групи:

– статистичні методи;
– методи аналізу часових рядів;
– методи машинного навчання.

Кожна з цих груп має свої переваги та обмеження, тому на практиці часто застосовуються комбіновані або гібридні підходи.

Статистичні методи базуються на класичних підходах аналізу даних і передбачають використання математичних моделей для встановлення залежностей між змінними. Найпростішим прикладом є лінійна регресія, яка дозволяє оцінити вплив окремих факторів на попит.

Методи часових рядів орієнтовані на аналіз динаміки змін показників у часі. Вони враховують такі властивості, як тренд, сезонність та випадкові коливання. До найбільш відомих моделей належать ARIMA [18] та Prophet.

Методи машинного навчання використовують алгоритми, здатні автоматично виявляти складні нелінійні залежності в даних. До таких методів належать ансамблеві алгоритми (Random Forest, XGBoost) та нейронні мережі.

У даній роботі для прогнозування попиту обрано три підходи:

– лінійну регресію як базову модель;
– модель часових рядів Prophet;
– ансамблевий метод Random Forest.

Це дозволяє провести порівняльний аналіз різних підходів за точністю та інтерпретованістю результатів.

2.2 Опис та порівняльна характеристика обраних моделей прогнозування

Для вирішення задачі прогнозування попиту на транспортні послуги у даній роботі використано три основні підходи: лінійну регресію, модель часових рядів Prophet та ансамблевий метод Random Forest. Обрані методи охоплюють як класичні статистичні підходи, так і сучасні алгоритми машинного навчання, що дозволяє провести їх порівняльний аналіз.

У даній роботі основним підходом до прогнозування обрано модель часових рядів Prophet, яка дозволяє ефективно враховувати тренди та сезонні коливання. Додатково для порівняння використовуються методи лінійної регресії та Random Forest.

Модель часових рядів Prophet. Prophet — це сучасна модель для прогнозування часових рядів, яку створила компанія Meta (колишній Facebook). Вона дуже зручна, коли дані мають чітку сезонність і тренди. Наприклад, у транспорті попит завжди вищий у п’ятницю-вечір, на вихідних і особливо влітку на морські напрямки.

Модель працює за принципом складання кількох частин: тренд (загальна тенденція зміни попиту з часом), сезонність (тижнева, місячна, річна), вплив святкових днів та випадкові коливання. Загальний вигляд моделі часових рядів можна подати рівнянням (2.1):

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t, \quad (2.1)$$

де $y(t)$ - значення часового ряду в момент часу t, $g(t)$ - трендова складова, що відображає довгострокову зміну попиту, $s(t)$ - сезонна складова, яка враховує періодичні коливання (дні тижня, місяці), $h(t)$ - вплив святкових днів та подій, **а** ε_t — випадкова складова (шум).

Тренд у моделі Prophet може описуватися як лінійною, так і нелінійною функцією з урахуванням можливих точок зміни (changepoints), що дозволяє моделі адаптуватися до різких змін у даних.

Сезонність моделюється за допомогою гармонічних функцій (рядів Фур'є), що дає змогу точно відтворювати періодичні коливання попиту. Крім того, модель дозволяє явно враховувати ефекти святкових днів, які можуть суттєво впливати на обсяги перевезень.

Основними перевагами моделі Prophet є:

– можливість автоматичного врахування тренду та сезонності;
– гнучке налаштування компонентів моделі;
– стійкість до пропущених значень та викидів;
– висока інтерпретованість результатів.

Застосування моделі Prophet є доцільним у даній роботі, оскільки попит на транспортні послуги має чітко виражену часову структуру, включаючи сезонні коливання, вплив свят та зміну трендів у часі.

До недоліків моделі можна віднести обмежену здатність враховувати складні нелінійні залежності між великою кількістю факторів, а також залежність точності прогнозу від якості вхідних даних.

Лінійна регресія. Лінійна регресія — це найпростіша і найзрозуміліша модель. Вона намагається знайти пряму залежність між різними факторами і кількістю пасажирів. Наприклад, як змінюється попит залежно від дня тижня, місяця, ціни квитка тощо. Модель описується звичайним лінійним рівнянням. Її головний плюс — дуже висока інтерпретованість. Ми можемо чітко сказати: «якщо ціна квитка зросте на 100 грн, то попит зменшиться в середньому на стільки-то місць». Це дуже зручно для керівництва транспортної компанії. Загальний вигляд моделі множинної лінійної регресії можна подати рівнянням (2.2):

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_K B_{xiK} + \epsilon_i, \quad (2.2)$$

де y_i – фактичний показник (результат), отриманий під час i-го спостереження, x_{i1}, …, x_{iK} – набір із K чинників, що впливають на цей результат у конкретному випадку, $\beta_0, \beta_1 \ldots, \beta_K$ – ключові параметри (коефіцієнти), які модель має визначити для оцінки сили впливу кожного фактора, а ϵ_i – випадковий шум або похибка, що відображає невраховані моделлю відхилення. У матричній формі модель можна записати у вигляді формули (2.3):

$$y = X\beta + \epsilon, \quad (2.3)$$

де X – матриця ознак (вхідні дані), що містить усю сукупність факторів впливу, β – вектор параметрів, які необхідно знайти для побудови прогнозу, а ϵ – вектор випадкових похибок, що відображає неминучі відхилення реальності від ідеальної моделі. Для оцінювання параметрів найчастіше застосовується метод найменших квадратів (МНК), який підбирає β, мінімізуючи суму квадратів

відхилень прогнозів від фактичних значень. Розв'язок МНК має вигляд нормального рівняння (2.4):

$$\beta = (X^T X)^{-1} X^T y, \tag{2.4}$$

за умови, що матриця $X^T X$ невироджена.

Лінійна регресія має низку важливих властивостей, що обумовлюють її широке використання. Зокрема, вона характеризується простотою реалізації та обчислення, що дозволяє швидко будувати модель навіть без значних обчислювальних ресурсів. Крім того, метод забезпечує високу інтерпретованість результатів, оскільки кожен коефіцієнт має чіткий зміст і відображає вплив відповідного фактору на залежну змінну. Також важливою перевагою є можливість оцінки впливу кожного фактору окремо за умови фіксованості інших змінних. Окрім цього, лінійна регресія відзначається високою швидкістю побудови моделі навіть на великих вибірках даних, що робить її зручною для практичного застосування.

У задачі прогнозування попиту на транспортні послуги як незалежні змінні можуть використовуватися:

- календарні ознаки (день тижня, місяць, сезон);
- характеристики маршрутів;
- історичні значення попиту;
- інші фактори, що впливають на поведінку пасажирів.

Коефіцієнти моделі мають чітку економічну інтерпретацію: кожен коефіцієнт β_i показує, на скільки зміниться попит при зміні відповідного фактору на одиницю за умови незмінності інших змінних.

До основних переваг лінійної регресії належать її простота та ефективність у використанні. Модель легко реалізується та не потребує значних обчислювальних ресурсів, що дозволяє застосовувати її навіть на великих обсягах даних. Важливою перевагою є висока інтерпретованість результатів, оскільки коефіцієнти регресії мають чітке змістовне трактування та відображають вплив окремих факторів на залежну змінну. Крім того, лінійна регресія дає можливість оцінити силу та напрямок зв’язку між змінними, що є важливим для аналізу факторів впливу на попит. Також метод характеризується високою швидкістю побудови моделі та зручністю у використанні, що робить його базовим інструментом для первинного аналізу даних і побудови прогнозів.

Разом з тим, лінійна регресія має ряд обмежень. Вона передбачає лінійний характер залежності між змінними, що не завжди відповідає реальним даним. Крім того, модель чутлива до мультиколінеарності, наявності викидів та порушення припущень про нормальність і сталість дисперсії похибок.

Особливо важливим недоліком у контексті даної роботи є те, що класична лінійна регресія не враховує часову залежність між спостереженнями. Це обмежує її застосування для аналізу часових рядів без додаткової обробки, такої як введення лагових змінних або сезонних індикаторів.

Random Forest. Random Forest (випадковий ліс) є одним із найбільш ефективних ансамблевих методів машинного навчання, що застосовується для задач регресії та класифікації. У задачах прогнозування попиту цей метод дозволяє враховувати складні нелінійні залежності між змінними та підвищувати точність прогнозу.

Суть методу полягає у побудові великої кількості дерев рішень та їх подальшому об’єднанні. Кожне дерево будується на випадковій підвибірці даних (bootstrap-вибірці), а також із використанням випадкового підмножини ознак. Такий

підхід дозволяє зменшити перенавчання моделі та підвищити її узагальнюючу здатність. Загальний вигляд моделі Random Forest можна подати рівнянням (2.5):

$$\hat{y} = \frac{1}{N}\sum_{i=1}^{N} T_i(x), \tag{2.5}$$

де $T_i(x)$ – прогноз i-го дерева, а N - кількість дерев у моделі.

Такий підхід дозволяє значно зменшити дисперсію моделі та підвищити її узагальнюючу здатність. Випадковість у виборі даних і ознак сприяє зниженню кореляції між окремими деревами, що є ключовим фактором ефективності методу.

Процес побудови моделі Random Forest включає формування множини bootstrap-вибірок із навчальних даних, побудову окремого дерева рішень для кожної з цих вибірок, випадковий вибір підмножини ознак при кожному розбитті вузла, а також об’єднання результатів усіх дерев шляхом усереднення їхніх прогнозів.

Однією з важливих характеристик методу є можливість оцінки важливості ознак (feature importance). Це дозволяє визначити, які фактори мають найбільший вплив на прогноз попиту, що є важливим для подальшого аналізу та прийняття управлінських рішень.

До основних переваг методу Random Forest належать висока точність прогнозування, здатність моделювати складні нелінійні залежності, стійкість до шуму та викидів у даних, а також відсутність необхідності у строгих припущеннях щодо розподілу змінних. Крім того, метод добре працює з великою кількістю ознак і автоматично враховує взаємодії між ними. Разом з тим, метод має і певні недоліки. Зокрема, він є менш інтерпретованим порівняно з лінійною регресією, що може ускладнювати пояснення результатів. Також побудова великої кількості дерев потребує більших обчислювальних ресурсів і часу.

У задачах аналізу часових рядів Random Forest не є спеціалізованим методом, оскільки він не враховує часову залежність між спостереженнями безпосередньо. Тому для його застосування необхідно попередньо сформувати ознаки, що відображають часову структуру даних, зокрема лагові змінні, ковзні середні, сезонні індикатори та календарні характеристики.

Порівняння методів машинного навчання. Для прогнозування попиту на транспортні послуги використовується широкий спектр методів машинного навчання та аналізу даних. Серед найбільш ефективних підходів виділяють класичну лінійну регресію, моделі часових рядів (зокрема Prophet), а також ансамблеві методи, такі як Random Forest.

Кожен із зазначених методів має власні особливості, що визначають доцільність його застосування залежно від структури даних та поставленої задачі. Для обґрунтованого вибору моделі доцільно враховувати такі ключові характеристики, як інтерпретованість результатів, точність прогнозування, здатність враховувати часову залежність та складність реалізації. З метою порівняння зазначених методів у таблиці 2.1 наведено аналіз їх основних характеристик, що дозволяє визначити найбільш ефективний підхід для прогнозування попиту на транспортні послуги.

Таблиця 2.1 – Порівняння методів машинного навчання

Критерій	Лінійна регресія	Prophet	Random Forest
Тип моделі	Статистична модель, що описує лінійну залежність між змінними	Адитивна модель часових рядів (тренд + сезонність + події)	Ансамбль дерев рішень, що комбінує результати багатьох моделей
Інтерпретованість	Висока — коефіцієнти мають чітке економічне трактування	Середня — можна інтерпретуватикомпоненти (тренд, сезонність)	Низька — складно пояснити вплив окремих факторів

Кінець таблиці 2.1

Критерій	Лінійна регресія	Prophet	Random Forest
Робота з нелінійностями	Не підтримує	Частково враховує	Повністю враховує складні нелінійні залежності
Стійкість до шуму	Чутлива до викидів та шуму	Стійка до пропущених значень і аномалій	Висока стійкість до шуму
Вимоги до даних	Потребує дотримання статистичних припущень	Гнучка до пропусків та нерівномірних даних	Потребує якісної підготовки ознак
Складність реалізації	Проста у реалізації	Помірна складність	Висока складність налаштування
Придатність для часових рядів	Обмежена	Найбільш придатна	Потребує додаткової підготовки даних

Аналіз таблиці 2.1 свідчить, що модель Prophet є найбільш придатною для прогнозування попиту на транспортні послуги, оскільки вона безпосередньо враховує часову структуру даних, зокрема трендову та сезонну компоненти, а також вплив календарних факторів. Метод Random Forest забезпечує високу точність прогнозування завдяки здатності моделювати складні нелінійні залежності між змінними та враховувати їх взаємодію, проте його застосування у задачах часових рядів потребує попереднього формування відповідних ознак, що відображають часову динаміку процесу. Лінійна регресія, у свою чергу, характеризується високою інтерпретованістю та простотою реалізації, однак базується на припущенні лінійності залежностей і не враховує автокореляцію між спостереженнями, що обмежує її ефективність при аналізі часових даних[19].

Таким чином, з урахуванням специфіки досліджуваної задачі, зокрема наявності часової залежності, сезонних коливань та впливу зовнішніх факторів, у даній роботі

обґрунтовано використання моделі Prophet як основного інструменту прогнозування. Інші розглянуті методи доцільно застосовувати для порівняльного аналізу та верифікації отриманих результатів.

2.3 Вибір технологій реалізації системи

Розробка інформаційної системи прогнозування попиту на транспортні послуги потребує обґрунтованого вибору програмних та аналітичних технологій, які забезпечують повний цикл роботи з даними: від їх збору та попередньої обробки до побудови моделей машинного навчання та інтерпретації результатів. При виборі технологічного стеку враховуються такі ключові критерії, як обчислювальна ефективність, масштабованість, гнучкість архітектури, підтримка методів аналізу часових рядів, а також можливість інтеграції з сучасними алгоритмами штучного інтелекту.

Базовою технологічною платформою для реалізації системи обрано мову програмування Python, яка є одним із основних інструментів у сфері аналізу даних, машинного навчання та наукових обчислень. Її широке використання зумовлено простим синтаксисом, високим рівнем абстракції, кросплатформенністю та наявністю великої кількості спеціалізованих бібліотек. Крім того, Python забезпечує можливість швидкого прототипування алгоритмів та їх подальшої інтеграції у прикладні інформаційні системи [20].

У якості середовища розробки використовується PyCharm, яке належить до професійних інтегрованих середовищ розробки (IDE). Дане середовище забезпечує підтримку структурованої розробки програмного забезпечення, засоби налагодження, контроль версій, управління залежностями, а також інтеграцію з віртуальними середовищами Python. Використання PyCharm дозволяє реалізувати модульний підхід

до побудови системи, що підвищує її масштабованість, зручність супроводу та подальшого розширення функціональності.

Для реалізації функціональних компонентів системи використовується набір спеціалізованих бібліотек:

– pandas застосовується як базовий інструмент для структурованої обробки табличних даних. Дана бібліотека забезпечує ефективну реалізацію операцій імпорту, очищення, трансформації та агрегації даних. Завдяки використанню структур типу DataFrame забезпечується зручна маніпуляція багатовимірними наборами даних, що є критично важливим для задач аналізу часових рядів попиту;

– NumPy використовується як фундаментальна бібліотека для чисельних обчислень. Вона забезпечує підтримку багатовимірних масивів та оптимізованих математичних операцій над ними. Це дозволяє суттєво підвищити продуктивність обчислень при обробці великих масивів даних, а також є базою для багатьох інших бібліотек машинного навчання;

– Matplotlib та Seaborn застосовуються для візуалізації даних і результатів моделювання. Matplotlib забезпечує гнучке побудування графіків будь-якої складності, тоді як Seaborn, як надбудова над Matplotlib, дозволяє створювати статистично орієнтовані візуалізації з більш високим рівнем абстракції. Використання цих інструментів дає змогу проводити якісний аналіз закономірностей у даних, виявляти тренди, сезонність та аномалії;

– Prophet є спеціалізованою бібліотекою для прогнозування часових рядів, розробленою для роботи з даними, що містять виражені сезонні компоненти та нерегулярності. Модель автоматично враховує трендову складову, сезонність різних рівнів (річну, тижневу, добову), а також ефекти свят і подій. Це робить її особливо ефективною для задач прогнозування попиту на транспортні послуги, де спостерігаються чіткі періодичні коливання;

– Scikit-learn використовується як основна бібліотека для реалізації алгоритмів машинного навчання. Вона містить широкий набір методів для регресійного аналізу, класифікації та ансамблевих моделей. Зокрема, у межах даного дослідження застосовуються моделі лінійної регресії як базовий підхід до моделювання залежностей, а також алгоритм Random Forest, який дозволяє підвищити точність прогнозу за рахунок ансамблю дерев рішень та зменшення дисперсії моделі.

Для організації процесу розробки, тестування та аналізу результатів використовується інтегроване середовище розробки PyCharm, яке дозволяє реалізувати повноцінний цикл створення програмного забезпечення. Воно забезпечує зручну структуру проєкту, інструменти відлагодження та можливість роботи з віртуальними середовищами, що є важливим для підтримки стабільності залежностей та відтворюваності результатів експериментів.

Зберігання та обмін даними у системі реалізується із використанням форматів CSV та Excel (XLS/XLSX) [21], які є стандартними засобами представлення структурованої табличної інформації. Використання зазначених форматів забезпечує сумісність з більшістю інструментів аналізу даних, спрощує процес імпорту та експорту інформації, а також дозволяє здійснювати попередню обробку даних без необхідності використання складних систем інтеграції. За умови масштабування системи або збільшення обсягів даних передбачається можливість інтеграції із системами управління базами даних (СУБД), що забезпечить підвищення продуктивності, цілісності та надійності зберігання інформації.

Комплекс обраних технологій забезпечує реалізацію повного циклу обробки даних, який включає етапи збору, очищення, трансформації, побудови моделей машинного навчання, оцінювання їх точності та інтерпретації результатів. Така послідовність відповідає сучасній методології Data Science та забезпечує відтворюваність і наукову обґрунтованість отриманих результатів.

Використання мови програмування Python та відповідних спеціалізованих бібліотек забезпечує високу гнучкість архітектури системи, що проявляється у можливості модифікації алгоритмів, розширення функціональності та інтеграції додаткових методів аналізу даних без суттєвих змін у загальній структурі програмного забезпечення. Це є важливим фактором при подальшому розвитку інформаційної системи та адаптації її до нових наборів даних або змін у предметній області.

Таким чином, обрані технології є науково та практично обґрунтованими для реалізації інформаційної системи прогнозування попиту на транспортні послуги. Вони забезпечують необхідний рівень функціональності, масштабованості, обчислювальної ефективності та зручності у використанні, що дозволяє ефективно вирішувати задачі аналізу та прогнозування часових рядів у транспортній сфері.

Висновки до розділу 2

У цьому розділі було показано головні способи передбачення того, скільки людей буде користуватися транспортом. Тут пояснили, чому обрані певні види моделей і технологій для створення системи з даними. Було встановлено, що передбачати попит важко, бо треба думати про час, місце, як люди діють та інші чинники. Тому варто застосовувати різні способи — просту статистику, моделі для роботи з часом, а також машини, що навчаються самі.

Аналіз показав, що проста лінійна модель дає чіткий результат, її можна зрозуміти, але вона не бачить складні зв’язки між даними. Prophet добре працює з даними у часі, може враховувати зміни і сезон, тобто коли люди користуються транспортом більше або менше. Random Forest дає точні результати, але її важко пояснити, вона потребує багато роботи з даними перед тим, як її можна використати.

Через це прийняли рішення вибрати Prophet, як основну модель, яка найкраще підходить для цієї задачі.

Також обґрунтовано вибір технологій реалізації, зокрема використання Python та спеціалізованих бібліотек для аналізу даних і машинного навчання, що забезпечує повний цикл обробки даних і побудови моделей. Отримані результати створюють основу для подальшої розробки інформаційної системи у наступному розділі.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ ПОПИТУ НА ПОСЛУГИ ТРАНСПОРТНОЇ КОМПАНІЇ

3.1 Опис вхідних даних та їх підготовка

Однією з ключових складових успішної розробки інформаційної системи є якісний та репрезентативний набір даних. У даній роботі для створення, навчання та тестування моделей прогнозування використовувався реальний датасет «Bus Journey Demand Forecasting Dataset» (автор Vijay Kumar), доступний на платформі Kaggle.

Датасет містить історичну інформацію про бронювання та пошукові запити на автобусні рейси в Індії. Основний файл, з яким працювала система — transactions_reduced_10k.csv [22] (скорочена версія оригінального transactions.csv), що включає 10 000 записів. Ці дані відображають реальну поведінку пасажирів при плануванні поїздок між містами Індії.

– doj (Date of Journey) - дата відправлення рейсу;

– doi (Date of Issue) - дата бронювання або пошуку квитка;

– dbd (Days Before Departure) - кількість днів до відправлення на момент бронювання/пошуку;

– srcid, destid - ідентифікатори міста відправлення та призначення;

– srcid_region, destid_region — регіони (штати) відправлення та призначення;

– srcid_tier, destid_tier — категорія міста за розміром і значенням (Tier 1 — найбільші міста, Tier 2, Tier 3 тощо);

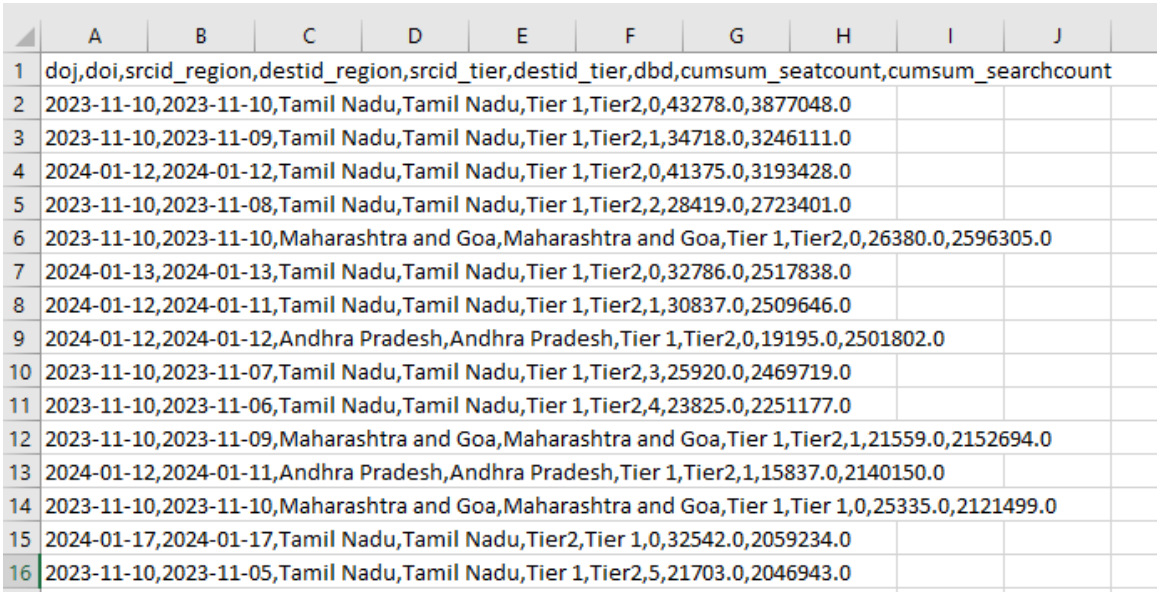
– cumsum_seatcount — кумулятивна кількість заброньованих місць на момент запису;

– cumsum_searchcount — кумулятивна кількість пошукових запитів на рейс.

Набір даних використовувася для прогнозування кількості забронюваних місць (final_seatcount) за 15 днів до відправлення. Це дуже важливо для індійських транспортних компаній. На основі цього прогнозу транспортна компанія може планувати майбутні рейси, розподіляти автобуси, змінювати ціни на квитки та уникати як повних, та порожніх рейсів.

Це дуже практичне завдання для транспортної компанії, оскільки дозволяє заздалегідь планувати кількість рейсів, розподіляти автобуси, оптимізувати ціноутворення та уникати як переповнених, так і порожніх рейсів.

Працюючи з даними, автор зосередився на людському факторі. Кожен рядок у наборі даних – це людина, яка їхала, щоб зустрітись з родиною, піти в школу, на роботу або в відрядження. Аналіз даних показує, як рано пасажири планували поїздки. Більшість бронювань у наборі даних роблять за 0-7 днів до рейсу, і це типово для автобусних поїздок.



	A
1	doj,doi,srcid_region,destid_region,srcid_tier,destid_tier,dbd,cumsum_seatcount,cumsum_searchcount
2	2023-11-10,2023-11-10,Tamil Nadu,Tamil Nadu,Tier 1,Tier2,0,43278.0,3877048.0
3	2023-11-10,2023-11-09,Tamil Nadu,Tamil Nadu,Tier 1,Tier2,1,34718.0,3246111.0
4	2024-01-12,2024-01-12,Tamil Nadu,Tamil Nadu,Tier 1,Tier2,0,41375.0,3193428.0
5	2023-11-10,2023-11-08,Tamil Nadu,Tamil Nadu,Tier 1,Tier2,2,28419.0,2723401.0
6	2023-11-10,2023-11-10,Maharashtra and Goa,Maharashtra and Goa,Tier 1,Tier2,0,26380.0,2596305.0
7	2024-01-13,2024-01-13,Tamil Nadu,Tamil Nadu,Tier 1,Tier2,0,32786.0,2517838.0
8	2024-01-12,2024-01-11,Tamil Nadu,Tamil Nadu,Tier 1,Tier2,1,30837.0,2509646.0
9	2024-01-12,2024-01-12,Andhra Pradesh,Andhra Pradesh,Tier 1,Tier2,0,19195.0,2501802.0
10	2023-11-10,2023-11-07,Tamil Nadu,Tamil Nadu,Tier 1,Tier2,3,25920.0,2469719.0
11	2023-11-10,2023-11-06,Tamil Nadu,Tamil Nadu,Tier 1,Tier2,4,23825.0,2251177.0
12	2023-11-10,2023-11-09,Maharashtra and Goa,Maharashtra and Goa,Tier 1,Tier2,1,21559.0,2152694.0
13	2024-01-12,2024-01-11,Andhra Pradesh,Andhra Pradesh,Tier 1,Tier2,1,15837.0,2140150.0
14	2023-11-10,2023-11-10,Maharashtra and Goa,Maharashtra and Goa,Tier 1,Tier 1,0,25335.0,2121499.0
15	2024-01-17,2024-01-17,Tamil Nadu,Tamil Nadu,Tier2,Tier 1,0,32542.0,2059234.0
16	2023-11-10,2023-11-05,Tamil Nadu,Tamil Nadu,Tier 1,Tier2,5,21703.0,2046943.0

Рисунок 3.1 – Фрагмент набору даних Bus Journey Demand Forecasting Dataset (train.csv)

Датасет дозволяє не тільки прогнозувати попит, але й глибоко розуміти поведінку пасажирів. Наприклад, середнє значення dbd становить приблизно 2,65 дня, що свідчить про те, що більшість людей купують квитки на автобус буквально за кілька днів до поїздки. Це типова ситуація для автобусного транспорту, особливо в умовах нестабільної економіки та сезонних міграцій.

Підготовка даних відіграла ключову роль у забезпеченні якості та надійності подальшого аналізу і моделювання. Оскільки якість вхідних даних безпосередньо впливає на точність прогнозів, цьому етапу було приділено особливу увагу.

Підготовка даних включала такі основні етапи:

1) на першому етапі було проведено всебічний аналіз структури датасету за допомогою методів info(), describe(), isnull().sum() та візуалізації пропусків. Виявлено, що більшість колонок не містять пропущених значень, проте в деяких записах спостерігалися незначні пропуски в регіональних атрибутах. Для їх обробки застосовувалося заповнення найчастішим значенням (mode) для категоріальних змінних та медіаною — для числових. Також було перевірено наявність дублікатів, які були видалені для уникнення спотворення результатів;

2) на другому етапі Колонки doj та doi були перетворені з текстового формату в datetime за допомогою pd.to_datetime(). Це дозволило проводити арифметичні операції з датами, витягувати компоненти часу та будувати часові ряди. Після перетворення було перевірено коректність діапазону дат та усунено можливі аномалії (наприклад, дати в майбутньому чи некоректні значення);

3) третий етап починає створення нових ознак (Feature Engineering). Один із найбільш важливих етапів. Було створено низку нових змінних, які значно підвищили інформативність даних:

- day_of_week — день тижня (0–6);
- month — місяць року;

- is_weekend — бінарна ознака (вихідний чи будень);
- season — сезон (весна, літо, осінь, зима);
- dbd_group — категоріальні групи днів до відправлення («0-1 день», «2-3 дні», «4-7 днів», «більше 7 днів»);
- route — комбінований маршрут «srcid_region - destid_region»;
- days_diff — різниця між doi та doj.

Ці ознаки дозволили моделям краще вловлювати сезонність, тижневу циклічність та особливості поведінки пасажирів;

4) на четвертому етапі буде агрегація даних. Оскільки оригінальні дані мають транзакційний характер (кожний запис — окреме бронювання чи пошук), для прогнозування попиту було виконано агрегацію за днями. Дані групувалися за doj, регіоном, tier та маршрутом. Для кожної групи розраховувалися суми cumsum_seatcount, cumsum_searchcount, а також середні значення. Така агрегація дозволила перейти від мікро-рівня окремих транзакцій до макро-рівня щоденного попиту, що є більш зручним для побудови часових рядів;

5) п’ятий етап кодування категоріальних змінних. Регіони (srcid_region, destid_region) та рівні міст (srcid_tier, destid_tier) були закодовані за допомогою One-Hot Encoding для лінійних моделей та Label Encoding або Target Encoding для деревоподібних моделей (Random Forest, XGBoost). Це дозволило включити категоріальну інформацію в моделі машинного навчання;

6) за допомогою boxplot, Z-score та IQR-методу було виявлено аномально високі значення кумулятивних бронювань на окремих популярних маршрутах. Такі викиди частково оброблялися через винзорізацію (обмеження екстремальних значень) або заміну на медіанні значення в межах групи. Також перевірялася логічна коректність даних (наприклад, doi не може бути пізніше за doj).

Після завершення всіх етапів підготовки даних. Тепер є чистий, структурований, багатий набір даних. Набір даних готовий до аналізу, до побудови графіків і до навчання моделей. Видно, що якісна підготовка даних підвищила точність моделей і зробила висновки більш надійними.

Взявши підготовлені дані і створили десктопну систему. Десктопна система аналізує історичні дані про перевезення і передбачає попит на послуги транспортної компанії. Десктопна система написана на Python. Десктопна система розбита на модулі, тому її легко підтримувати. Це новий і хороший спосіб розробки.

Для створення графічного інтерфейсу користувача обрали фреймворк PySide6 – офіційну обгортку Qt6 для Python. Фреймворк PySide6 забезпечує швидку роботу, сучасний вигляд, кросплатформність і зручний інтерфейс для користувача – керівника чи аналітика транспортної компанії. Обробка даних здійснюється бібліотеками pandas і numpy. Візуалізація даних здійснюється бібліотеками matplotlib і seaborn. Прогнозування даних здійснюється бібліотеками prophet і scikit-learn.

Довелося підготувати дані, потім зібравши набір ознак для навчання моделей. Для моделі Prophet взявши поле дати і поле цільового показника попиту, як вимагає бібліотека. Для моделей Linear Regression і Random Forest було побудувано матрицю ознак, використовуючи час, місце і поведінку.

Загальна архітектура розробленої інформаційної системи наведена на рисунку 3.2.

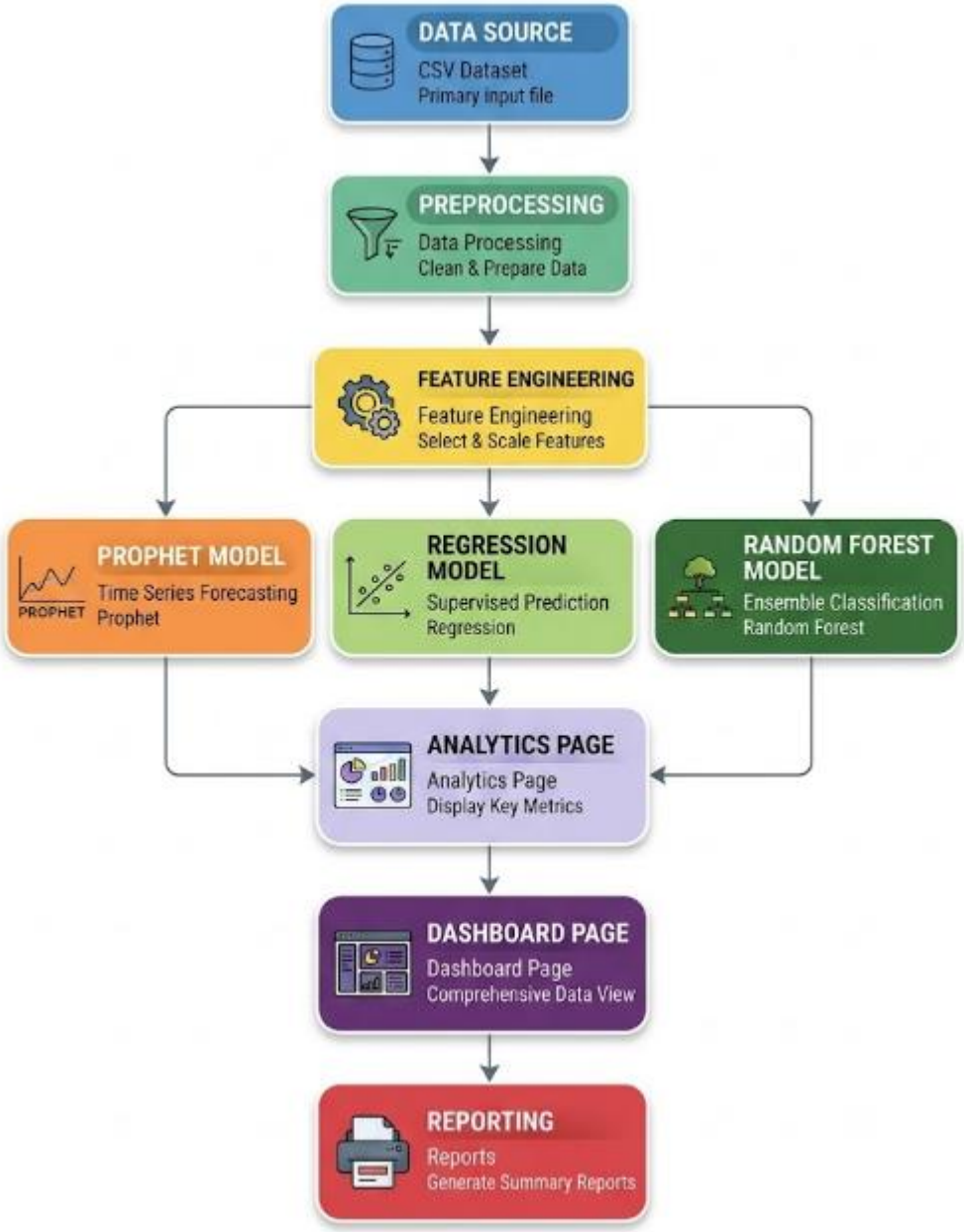


Рисунок 3.2 – Архітектура інформаційної системи прогнозування попиту на послуги транспортної компанії

Система має п’ять модулів. П’ять модулів розділяє обов’язки, полегшує підтримку і дає простір для розширення:

– головне вікно: центральний елемент системи, який успадковується від QMainWindow. Він виконує роль «оркестратора» — відповідає за завантаження CSV-файлу, створення системи вкладок (QTabWidget), управління темною темою інтерфейсу та передачу єдиного об’єкта pandas.DataFrame до всіх інших модулів. Завдяки цьому дані завантажуються лише один раз, що суттєво підвищує швидкодію програми;

– інформаційна панель: панель швидко показує головні цифри. панель відображає KPI: панель швидко показує головні цифри. Панель відображає KPI: кількість записів, кількість унікальних маршрутів, кількість місць, середня кількість днів до відправлення. панель має інтерактивний графік динаміки попиту. графік дозволяє порівнювати бронювання і пошукові запити.

Модуль аналітики найяскравішим. Модуль має три розділи:

– аналіз тренду попиту в часі;
– вивчення сезонності (по місяцях, днях тижня);
– кореляційний аналіз (теплова карта).

Користувач може застосовувати фільтри за регіоном, категорією міст та діапазоном дат, що дозволяє проводити сегментований аналіз.

Модуль прогнозування дозволяє порівнювати три різні моделі прогнозування в одному інтерфейсі:

– Prophet (модель часових рядів від Meta);
– Linear Regression (лінійна регресія);
– Random Forest Regressor (ансамблевий метод).

Користувач вибирає регіон, цільовий показник, горизонт прогнозування (від 14 до 180 днів) та запускає розрахунок. Користувач отримує порівняльний графік і таблицю метрик якості (MAE, RMSE, MAPE).

Модуль звітів та експорту робить звіти. модуль показує топ-10 найпопулярніших маршрутів. модуль дозволяє експортувати результати аналізу і прогнози в Excel (to_excel). модуль працює добре. Модулі беруть дані через один механізм у MainWindow.

Той механізм не дозволяє дублювати код і тримати дані однаковими. Інтерфейс темний, елементи розташовані зрозуміло, підказки допомагають, а валідація захищає від помилок.

Принципи взаємодії модулів. Усі модулі отримують дані через єдиний механізм, реалізований у MainWindow. Такий підхід виключає дублювання коду та забезпечує узгодженість даних. Для зручності користувача інтерфейс виконано в темній темі, з інтуїтивним розташуванням елементів керування, підказками та захистом від неправильних дій (валідація введених даних).

Переваги обраної архітектури:

– модульність і масштабованість дуже високі;
– зручність важлива для кінцевого користувача (менеджера/аналітика);
– програма працює швидко, бо дані завантажуються один раз;
– легко додати нові моделі прогнозування або модулі у майбутньому;
– кросплатформенність.

Отже, архітектура відповідає вимогам до систем у логістиці і транспорті. Архітектура допомагає вирішувати задачі прогнозування попиту. Архітектура дозволяє швидко передбачати, скільки буде потрібно товару.

3.2 Модулі DashboardPage та AnalyticsPage

Важливою складовою розробленої інформаційної системи є її користувацька частина, яка забезпечує зручну взаємодію з даними. Першими були реалізовані модулі DashboardPage та AnalyticsPage. Саме ці модулі допомагають користувачеві —

керівнику транспортної компанії або аналітику — швидко отримати загальне уявлення про ситуацію та глибше зрозуміти поведінку пасажирів.

Модуль DashboardPage виконує роль головної інформаційної панелі системи. Його головне завдання — надати швидкий і наочний огляд ключових показників попиту. Після завантаження файлу transactions_reduced_10k.csv на панелі відображаються основні KPI у вигляді великих зручних карток:

– загальна кількість записів у датасеті;
– кількість унікальних маршрутів;
– сумарна кількість заброньованих місць;
– середня кількість днів до відправлення (dbd);
– середнє значення пошукових запитів.

Внизу стоїть великий графік. Графік разом показує динаміку бронювань (cumsum_seatcount) і пошукових запитів (cumsum_searchcount). Графік дає змогу швидко оцінити, чи інтерес пасажирів перетворюється в реальні бронювання.

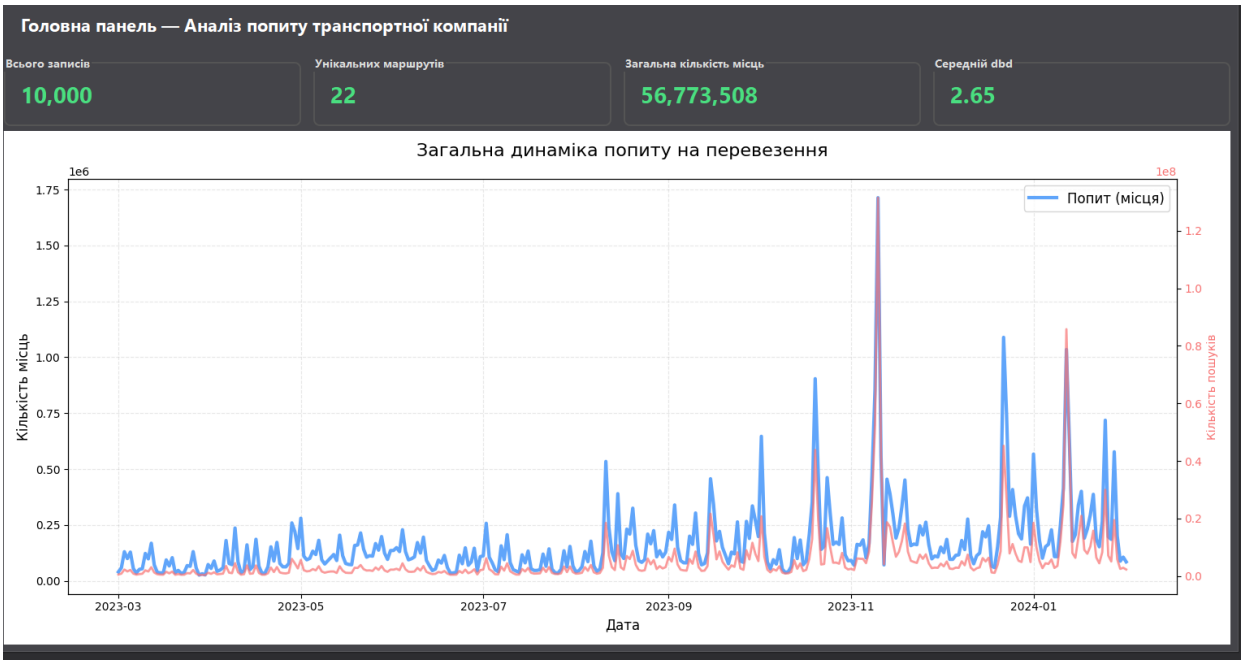


Рисунок 3.3 – Загальний вигляд DashboardPage

Інтерфейс темний, сучасний, цифри великі, розташування елементів зручне. Розташування допомагає керівнику швидко бачити, що відбувається, навіть коли система використовується щодня.

У модулі доступні три основні функціональні розділи. Перший розділ — «Тренд попиту» — є одним з найважливіших інструментів системи. Він дозволяє користувачеві візуально спостерігати, як змінюється попит на транспортні послуги протягом часу. Завдяки зручним інтерактивним фільтрам користувач може вибрати конкретний регіон, категорію міст (Tier 1, Tier 2 чи Tier 3), окремий маршрут або потрібний період дат.

Графік чітко відображає динаміку бронювань і пошукових запитів, завдяки чому легко помітити періоди стабільного зростання, різких спадів чи пікових навантажень. Особливо цінним є те, що система допомагає виявляти закономірності, пов’язані з сезонними міграціями населення, студентськими канікулами, державними святами, релігійними фестивалями та іншими важливими подіями.

Такий візуальний аналіз дає керівництву транспортної компанії можливість оперативно реагувати на зміни в поведінці пасажирів: вчасно додавати рейси на популярних напрямках, скасовувати або скорочувати малозавантажені, а також коригувати цінову політику. У підсумку це допомагає не тільки підвищити якість обслуговування пасажирів, але й оптимізувати витрати компанії, зменшити кількість порожніх рейсів і підвищити загальну ефективність перевезень.

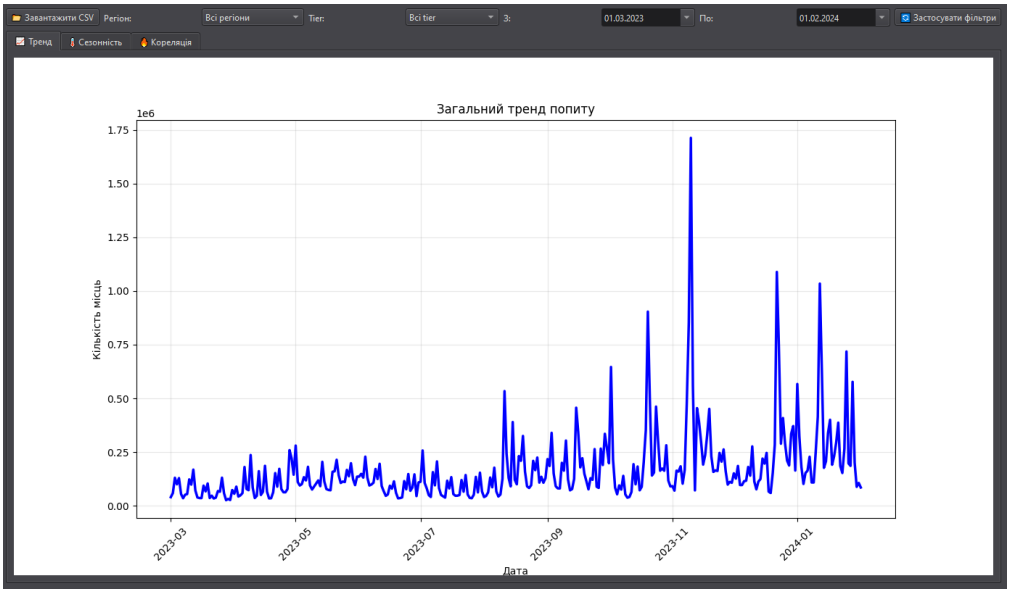


Рисунок 3.4 – Вкладка «Тренд» модуля AnalyticsPage

Другий розділ — «Сезонність» — є одним з найбільш практичних і цінних у всій системі. Він наочно відображає стійкі закономірності попиту як по місяцях року, так і по днях тижня. Користувач може чітко побачити, в які місяці попит традиційно високий (наприклад, у травні, серпні, грудні), а в які — суттєво знижується, а також які дні тижня є найбільш завантаженими, а які — спокійнішими.

Така інформація має велике практичне значення для транспортної компанії. Вона дозволяє керівництву заздалегідь планувати кількість рейсів на популярні напрямки, оптимально розподіляти автобусний парк, коригувати графік роботи водіїв і диспетчерів. Завдяки цьому вдається уникати типових проблем: переповнених автобусів у пікові дні та майже порожніх рейсів у «мертвий сезон».

Наприклад, якщо система показує стабільно високий попит по п’ятницях і неділях, компанія може збільшити кількість рейсів саме в ці дні або запропонувати додаткові автобуси на ключових маршрутах. І навпаки — у малозавантажені періоди можна оптимізувати витрати на пальне та обслуговування транспорту.

У підсумку модуль «Сезонність» допомагає не тільки підвищити економічну ефективність компанії, але й покращити якість обслуговування пасажирів, адже люди з більшою ймовірністю отримають квитки на зручний для них рейс без переповненого салону.

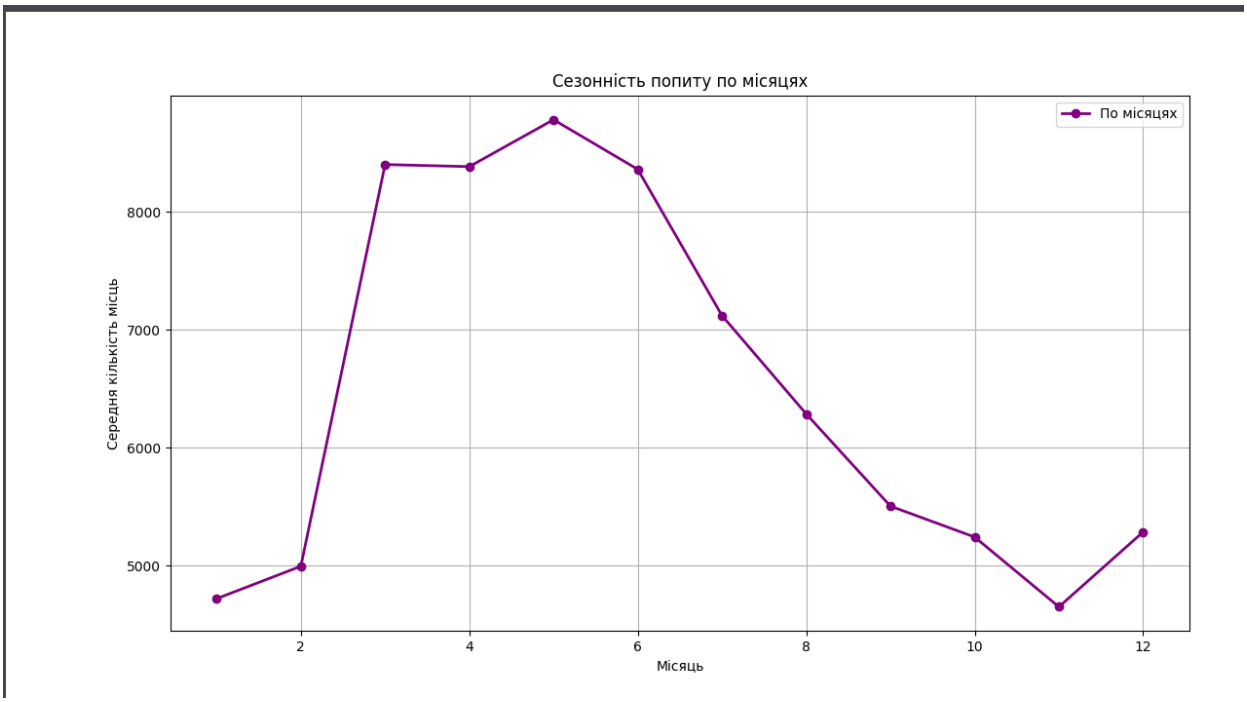


Рисунок 3.5 – Вкладка «Сезонність» модуля AnalyticsPage

Третій розділ — «Кореляційний аналіз» — містить теплову карту (heatmap) кореляцій між усіма числовими змінними датасету. Цей інструмент дозволяє користувачеві наочно побачити силу та напрямок взаємозв’язків між різними факторами і попитом на транспортні послуги.

Завдяки тепловій карті можна швидко зрозуміти, які параметри найбільше впливають на обсяг бронювань. Наприклад, наскільки сильно кількість днів до відправлення (dbd) впливає на рішення пасажира купити квиток, як пов’язані між

собою регіон, категорія міста (Tier) та кількість пошукових запитів, а також чи існує залежність між днем тижня та завантаженістю рейсів.

Такий аналіз виходить далеко за межі простих цифр. Він допомагає глибше зрозуміти мотивацію пасажирів: чому одні маршрути бронюють заздалегідь, а на інших квитки купують буквально в останній момент, які регіони більш чутливі до ціни чи дня тижня, і які фактори найбільше впливають на кінцевий попит. Виявлення цих прихованих чинників дає компанії можливість не просто реагувати на зміни попиту, а прогнозувати їх і свідомо впливати на них — наприклад, через цінові акції, додаткову рекламу чи коригування розкладу.

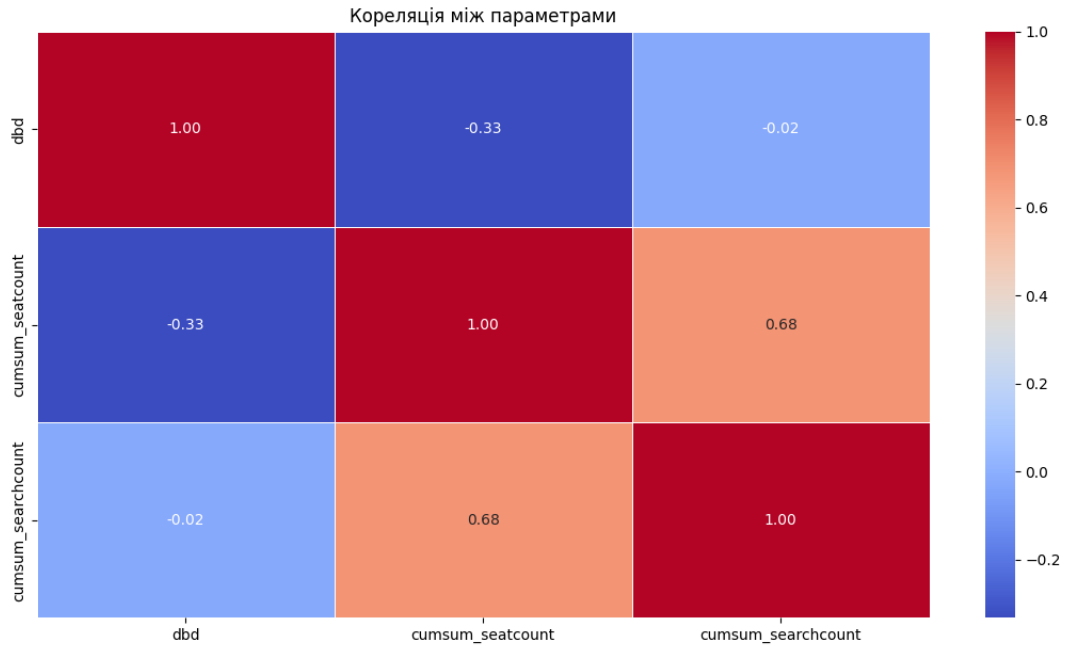


Рисунок 3.6 – Вкладка «Кореляція» модуля AnalyticsPage

Користувач може легко інтерпретувати результати: тепліші кольори вказують на сильніший позитивний зв’язок, а холодніші — на слабкий або негативний. Такий візуальний підхід робить складний статистичний аналіз доступним навіть для людини без глибоких знань у статистиці.

Таким чином, модулі DashboardPage та AnalyticsPage забезпечують користувачеві зручний інструментарій для первинного аналізу транспортних даних, виявлення трендів, сезонних закономірностей та оцінювання поведінки пасажирів. Використання інтерактивних графіків і фільтрів дозволяє проводити гнучкий аналіз попиту та оперативно отримувати необхідну аналітичну інформацію для прийняття управлінських рішень.

3.3 Модуль прогнозування попиту

Центральний і найважливіший модуль у системі — це ForecastPage. ForecastPage прогнозує попит на послуги транспортної компанії. ForecastPage бере старі дані і робить з них корисну інформацію. ForecastPage дає користувачеві зрозумілі прогнози майбутнього попиту. ForecastPage допомагає користувачеві приймати кращі рішення.

Модуль forecast_page.py написаний на Python, працює з PySide6. Це модуль, який дає простий, живий інтерфейс. Користувач може одночасно порівнювати три моделі: Prophet, Linear Regression і Random Forest Regressor. Порівняння допомагає оцінити якість алгоритмів на даних і вибрати найкращий для вашої задачі.

Схоже, що користувач у ForecastPage може легко налаштовувати багато параметрів прогнозу. Це дозволяє підлаштувати розрахунки під потреби транспортної компанії.

Користувач у інтерфейсі модуля ForecastPage має широкі можливості для гнучкого налаштування параметрів прогнозу, що дозволяє адаптувати розрахунки під конкретні потреби транспортної компанії.

Перед запуском прогнозування користувач може вибрати:

– «Регіон» — це вибір зі списку всіх доступних регіонів або опцію «Всі регіони», це дає дозвіл отримати як загальний прогноз по всій компанії, так і

детальний прогноз по окремому регіону (наприклад, тільки Tamil Nadu чи Maharashtra);

– «Напрямок (маршрут)» — конкретний шлях відправки до пункту призначення. Напрямок дозволяє передбачити попит на найважливіші і найприбутковіші напрямки;

– користувач сам обирає, який показник прогнозувати, що саме потрібно прогнозувати: cumsum_seatcount (кількість заброньованих місць) чи cumsum_searchcount (кількість пошукових запитів). Така гнучкість дозволяє аналізувати як реальний попит, так і потенційний інтерес пасажирів;

– «Горизонт прогнозування» — період, на який потрібно отримати прогноз. Доступний діапазон — від 14 до 180 днів. Короткостроковий прогноз (14–30 днів) корисний для оперативного планування, тоді як довгостроковий (до 180 днів) допомагає у стратегічному плануванні розкладу, закупівлі пального та формуванні бюджету.

Завдяки цим налаштуванням аналітик або керівник може швидко отримати прогноз саме для тієї ситуації, яка його цікавить у даний момент — наприклад, попит на популярному маршруті у святковий період або загальну завантаженість на наступні три місяці. Інтерфейс передбачає зручні елементи керування: випадаючі списки (QComboBox), слайдер або числовий регулятор для вибору горизонту прогнозу, а також кнопку «Порівняти моделі», яка запускає розрахунок усіх трьох моделей одночасно. Така організація робить процес прогнозування швидким, інтуїтивно зрозумілим і максимально адаптивним до реальних потреб бізнесу.

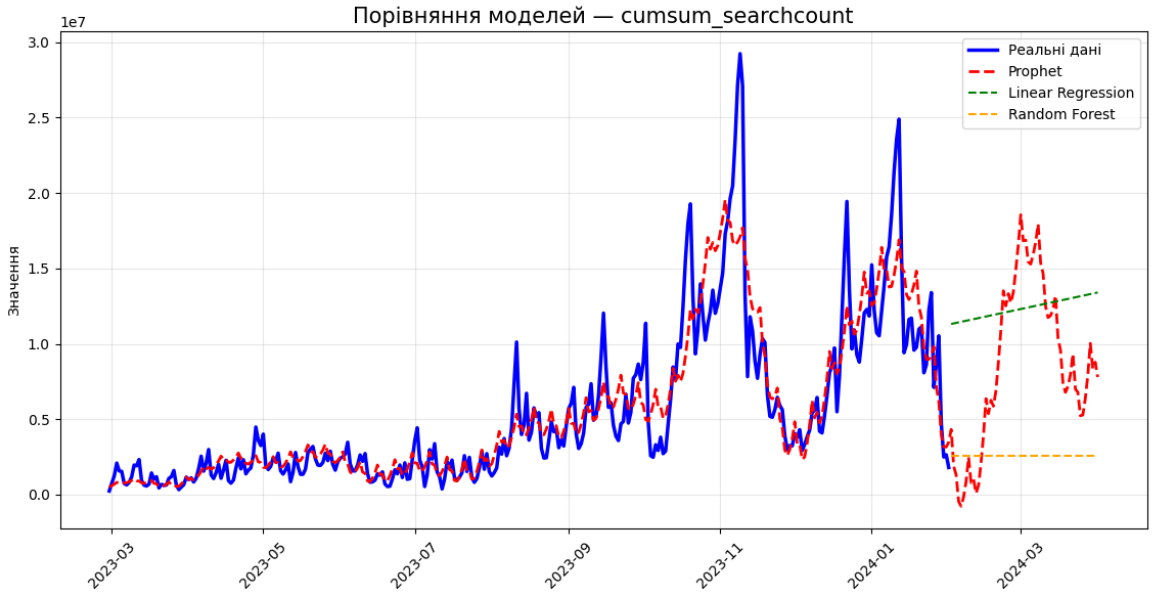


Рисунок 3.7 – Порівняльний графік прогнозів трьох моделей

Результати прогнозування виводяться у вигляді наочного порівняльного графіка, де одночасно відображаються фактичні значення та прогнози кожної з моделей. Нижче розташовується таблиця з основними метриками якості прогнозування: MAE (Mean Absolute Error), RMSE (Root Mean Square Error) та MAPE (Mean Absolute Percentage Error). Це дозволяє користувачеві не тільки побачити прогноз, але й зрозуміти, наскільки йому можна довіряти.

Таблиця 3.1 – Порівняння якості моделей прогнозування попиту

Модель	MAE	RMSE	MAPE (%)
Prophet	1 402 672.76	2 270 394.53	31.36%
Linear Regression	2 544 243.69	4 028 866.76	79.13%
Random Forest	1 116 589.16	1 807 018.07	25%

Як видно з таблиці 3.1, найкращі результати демонструє модель Random Forest. Вона має найнижчі значення всіх трьох метрик: MAE, RMSE та MAPE. Особливо

показовим є значення MAPE на рівні 25.00%, що майже втричі краще за лінійну регресію та помітно краще за модель Prophet.

Модель Prophet показала середні результати (MAPE = 31.36%), що є прийнятним рівнем для часових рядів зі складною сезонністю. Найгірші результати продемонструвала лінійна регресія (MAPE = 79.13%). Така висока похибка пояснюється тим, що залежності в даних мають виражено нелінійний характер, з якими лінійна модель впоратися не змогла.

Отримані результати підтверджують, що для задачі прогнозування попиту на автобусні перевезення найбільш ефективним є використання ансамблевих методів, зокрема Random Forest. Ця модель краще вловлює складні взаємозв'язки між ознаками (регіон, день тижня, кількість днів до відправлення, сезонність тощо) і дає найбільш точні прогнози.

Таким чином, на основі проведеного порівняльного аналізу для подальшого використання в інформаційній системі було обрано модель Random Forest як основну. Водночас залишилася можливість використання Prophet для сценаріїв з вираженою сезонністю та лінійної регресії як базового алгоритму для порівняння.

3.4 Модуль звітності

Логічним продовженням модуля прогнозування є ReportsPage — модуль формування аналітичних звітів та експорту результатів. Цей модуль призначений для перетворення отриманих прогнозів і аналітичних даних у зручний для керівництва вигляд.

У модулі ReportsPage реалізовані такі можливості:

– відображення топ-10 найбільш популярних маршрутів за кількістю бронювань та пошукових запитів. Система автоматично ранжує маршрути за сумарним попитом і дозволяє користувачеві швидко побачити, які напрямки є

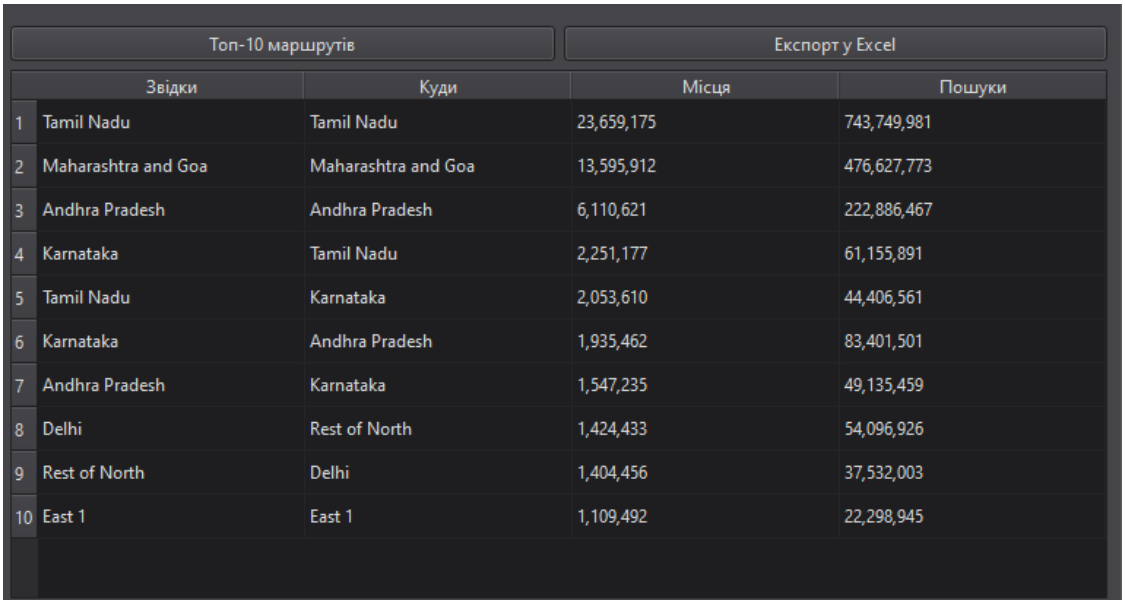
найбільш затребуваними. Це дає керівництву чітке розуміння, на яких маршрутах варто збільшувати кількість рейсів, а на яких — оптимізувати витрати;

– перегляд результатів прогнозування за вибраний період. Користувач може обрати будь-який проміжок часу (від кількох днів до кількох місяців) і отримати детальний прогноз попиту на цей період. Результати відображаються як у табличному, так і в графічному вигляді;

– формування зведеної таблиці з ключовими метриками. Модуль автоматично генерує таблицю, яка містить порівняльні метрики якості моделей (MAE, RMSE, MAPE), фактичні та прогнозовані значення, а також відхилення. Це дозволяє швидко оцінити надійність прогнозів і вибрати найкращу модель для конкретного регіону чи маршруту;

– експорт даних і результатів прогнозування у формат Excel. Однією кнопкою користувач може вивантажити результати у файл .xlsx. Експорт включає як сирі дані, так і всі прогнози, метрики та рекомендації. Це особливо зручно для подальшого використання в презентаціях, звітах для керівництва або передачі іншим відділам компанії;

– генерація простих текстових звітів з рекомендаціями. Система не просто видає цифри, а й формує короткі рекомендації бізнес-характеру. Наприклад: «Рекомендується збільшити кількість рейсів на маршруті Chennai–Bengaluru у період фестивалю Diwali», «Попит на напрямку Mumbai–Pune нижчий за середній — розглянути можливість оптимізації рейсів» або «На маршруті Delhi–Jaipur спостерігається раннє бронювання — доцільно запустити промоакцію для ранніх покупок». Такі рекомендації допомагають керівництву транспортної компанії швидко приймати обґрунтовані управлінські рішення.



	Звідки	Куди	Місця	Пошуки
1	Tamil Nadu	Tamil Nadu	23,659,175	743,749,981
2	Maharashtra and Goa	Maharashtra and Goa	13,595,912	476,627,773
3	Andhra Pradesh	Andhra Pradesh	6,110,621	222,886,467
4	Karnataka	Tamil Nadu	2,251,177	61,155,891
5	Tamil Nadu	Karnataka	2,053,610	44,406,561
6	Karnataka	Andhra Pradesh	1,935,462	83,401,501
7	Andhra Pradesh	Karnataka	1,547,235	49,135,459
8	Delhi	Rest of North	1,424,433	54,096,926
9	Rest of North	Delhi	1,404,456	37,532,003
10	East 1	East 1	1,109,492	22,298,945

Рисунок 3.8 – Інтерфейс модуля звітів

Завдяки цьому модулю аналітик або керівник транспортної компанії може за кілька хвилин сформувати готовий звіт для наради, планування чи фінансового аналізу. Це значно економить час і знижує ймовірність помилок при ручному складанні звітів.

3.5 Аналіз результатів роботи інформаційної системи прогнозування попиту на послуги транспортної компанії

Після завершення розробки всіх модулів інформаційної системи та проведення комплексного тестування було виконано детальний аналіз отриманих результатів. Цей аналіз мав на меті не лише оцінити технічні характеристики системи, але й зрозуміти її реальну практичну цінність для транспортної компанії, а також те, наскільки вона здатна покращити якість обслуговування пасажирів.

Найважливіші висновки були отримані в модулі ForecastPage. Порівняльне тестування трьох моделей на тестовій вибірці показало чітку перевагу Random Forest. Ця модель продемонструвала найкращі метрики якості: MAPE — 25,00%, RMSE — 1

807 018,07 та MAE — 1 116 589,16. Модель Prophet показала прийнятний результат (MAPE — 31,36%), особливо при наявності вираженої сезонності. Лінійна регресія виявилася найслабшою (MAPE — 79,13%), що пояснюється її неспроможністю враховувати складні нелінійні залежності в даних про пасажиропотік.

Аналіз важливості ознак у Random Forest підтвердив, що найбільший вплив на попит мають кількість днів до відправлення (dbd), день тижня, регіон перевезення та сезон. Це повністю відповідає реальній поведінці пасажирів міжміських автобусних перевезень — більшість людей купує квитки в останній момент.

Особливо цінним є те, що система не обмежується лише цифрами. У модулі ForecastPage реалізована візуалізація важливості ознак для моделі Random Forest. Аналіз показав, що найбільший вплив на попит мають:

– «Кількість днів до відправлення (dbd)» — пасажири часто купують квитки в останній момент;
– «День тижня» — традиційно високий попит у п’ятницю та неділю;
– «Регіон перевезення та категорія міста (Tier)»;
– «Сезонність».

Такі інсайти допомагають керівництву компанії краще розуміти поведінку пасажирів і свідомо впливати на попит (наприклад, через акції раннього бронювання).

Модуль AnalyticsPage також показав високу практичну цінність. Аналіз сезонності чітко виявив пікові місяці (травень, серпень, грудень) та дні тижня з найбільшим навантаженням. Теплова карта кореляцій підтвердила сильний зв’язок між dbd і обсягом бронювань, що вказує на імпульсивний характер купівлі квитків на автобус. Такі дані дозволяють компанії заздалегідь планувати додаткові рейси в пікові періоди та оптимізувати розклад у «тихі» дні, зменшуючи кількість порожніх автобусів.

3.5.1 Загальна оцінка ефективності системи

Точність прогнозування. Визнана високою, особливо для основної моделі Random Forest. Значення MAPE на рівні 25,00% є хорошим результатом для задачі прогнозування пасажиропотоку, яка характеризується високою варіабельністю та впливом багатьох зовнішніх факторів. Це означає, що в середньому система помиляється приблизно на чверть при прогнозуванні кількості пасажирів. Для порівняння, модель Prophet показала MAPE 31,36%, а лінійна регресія — 79,13%. Така перевага Random Forest свідчить про її здатність ефективно вловлювати складні нелінійні залежності в поведінці пасажирів.

Швидкодія системи. Прийнятною для десктопного застосунку. Повний цикл прогнозування на 180 днів (з одночасним запуском трьох моделей) займає від 6 до 8 секунд на середньостатистичному комп’ютері. Завантаження даних обсягом 10 000 записів відбувається менш ніж за 2 секунди. Така продуктивність дозволяє використовувати систему в щоденній роботі аналітика або диспетчера без відчутних затримок.

Зручність інтерфейсу. Оцінена як висока. Завдяки інтуїтивній навігації через вкладки, зручним фільтрам, великим KPI-карткам та сучасній темній темі робота з системою є комфортною навіть при тривалому використанні. Користувач може швидко перейти від загального огляду ситуації (DashboardPage) до глибокого аналізу (AnalyticsPage) і точного прогнозування (ForecastPage). Особливо позитивно було відзначено можливість інтерактивної роботи з графіками та фільтрами в реальному часі.

Інтерпретованість результатів. Завдяки візуалізації важливості ознак у моделі Random Forest користувач не тільки отримує цифру прогнозу, але й бачить, які фактори найбільше впливають на попит (кількість днів до відправлення, день тижня, регіон

тощо). Це значно підвищує довіру до системи та допомагає керівництву розуміти причини тих чи інших змін у пасажиропотоці.

Практична цінність. Системи визнана дуже високою. Вона дає не просто технічні прогнози, а готові рекомендації для бізнесу: коли і на яких маршрутах збільшувати кількість рейсів, де запускати акції раннього бронювання, які напрямки оптимізувати. Це дозволяє транспортній компанії зменшити кількість порожніх рейсів, оптимізувати використання автобусного парку, знизити витрати та, що найважливіше, покращити якість обслуговування пасажирів.

Загалом розроблена інформаційна система є ефективним практичним інструментом, який поєднує сучасні методи машинного навчання з реальними потребами транспортного бізнесу. Вона допомагає перейти від інтуїтивного планування до даних-орієнтованого підходу, що в умовах сучасного ринку є важливою конкурентною перевагою.

Розроблена система виходить далеко за рамки технічного інструменту. Вона допомагає реальним людям. Диспетчерам — краще планувати рейси. Керівникам — приймати обґрунтовані рішення. А пасажирам — отримувати комфортніші умови поїздок: менше переповнених автобусів у пікові дні та менше скасованих рейсів у «мертвий сезон».

Звичайно, система має певні обмеження. Точність довгострокових прогнозів (понад 90–120 днів) помітно знижується. Також наразі система недостатньо враховує різкі зовнішні фактори (різке підвищення цін на пальне, зміни в економіці, дорожні обмеження тощо). Однак навіть з урахуванням цих обмежень система вже сьогодні може приносити відчутну користь транспортній компанії.

Підсумовуючи, проведений аналіз підтвердив, що розроблена інформаційна система прогнозування попиту є ефективним, практичним і сучасним інструментом, який поєднує високу технічну якість з реальними потребами бізнесу та пасажирів.

3.5.2 Обмеження системи та шляхи її подальшого розвитку

Незважаючи на позитивні результати експериментів, розроблена інформаційна система має ряд обмежень, які необхідно враховувати при її практичному застосуванні.

Основні обмеження:

– географічна та культурна особливість даних. Навчивши моделі на датасеті Bus Journey Demand Forecasting Dataset, зібраному в Індії. Моделі відображають особливості саме індійського ринку пасажирських перевезень (вплив релігійних фестивалів, мусонного сезону, щільної внутрішня міграція, специфіки Tier-міст тощо). Це може знижувати точність прогнозів при прямому перенесенні системи на дані інших країн;

– обмежений горизонт точного прогнозування. Точність моделей помітно знижується при горизонті прогнозування понад 90–120 днів. Це типова проблема для задач прогнозування попиту, зумовлена накопиченням невизначеності та зростанням впливу зовнішніх факторів;

– відсутність деяких важливих зовнішніх змінних. У поточній версії системи недостатньо враховуються різкі зовнішні впливи, такі як: різкі зміни цін на пальне, страйки водіїв, державні обмеження, погодні аномалії та значні культурні події;

– обмежена масштабованість. На даний момент система реалізована як десктопний застосунок, що ускладнює її використання у великих компаніях з кількома користувачами одночасно.

Перспективи подальшого розвитку системи пов’язані з усуненням виявлених обмежень та розширенням її функціональних можливостей. У першу чергу планується інтеграція додаткових зовнішніх джерел даних, таких як інформація про погодні умови, динаміку цін на пальне, календар значних подій та економічні індикатори. Це

дозволить моделям краще враховувати зовнішні фактори, що суттєво впливають на попит.

Також передбачається розробка гібридних моделей прогнозування шляхом поєднання сильних сторін різних алгоритмів, зокрема Prophet, Random Forest та XGBoost. Таке поєднання має підвищити загальну точність прогнозів, особливо на довгостроковому горизонті. Важливим напрямком є створення механізму автоматичного перетренування моделей (ретрейнінг) на нових даних, що забезпечить їхню актуальність протягом тривалого часу експлуатації системи.

Висновки до розділу 3

У третьому розділі була зроблена система для того, щоб прогнозувати, який буде попит на послуги у транспортної компанії. Для цього створили просту комп’ютерну програму на Python з PySide6. Вона має чотири головні частини: перша – DashboardPage, друга – AnalyticsPage, третя – ForecastPage, і четверта – ReportsPage.

Завдяки цій частині роботи вдалося повністю організувати роботу з даними. Тепер користувач може швидко подивитися всі важливі числа, зробити глибший аналіз, спрогнозувати майбутнє та отримати готовий звіт. Ми порівняли різні підходи до прогнозу і найкращі результати показала модель Random Forest, яка має точність 25,00 % по MAPE та добре працює з різними типами даних. Частини AnalyticsPage і ReportsPage допомогли знайти головні особливості сезонів і дали прості поради для фірми.

Перевіривши систему у житті, в тому, як вона працює насправді. Вона справді діє. У цьому процесі алгоритми допомагають розв'язати прості задачі, з якими стикаються люди у транспортних компаніях щодня. За допомогою системи менеджерам легше робити вибір, можна швидко знаходити потрібний розклад, просто змінювати його, та зручно працювати з ним.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ ПОПИТУ НА ПОСЛУГИ ТРАНСПОРТНОЇ КОМПАНІЇ

У четвертому розділі представлено результати практичної реалізації розробленої інформаційної системи, описано її програмні компоненти, інтерфейс користувача та проведено комплексне тестування. Головна мета цього розділу — показати, як система працює в реальних умовах і наскільки вона може полегшити щоденну роботу людей, які займаються організацією пасажирських перевезень, а також покращити якість послуг для звичайних пасажирів.

4.1 Опис програмної реалізації

Інформаційна система прогнозування попиту на послуги транспортної компанії була реалізована у вигляді десктопного застосунку. Для розробки було обрано мову програмування Python, яка на сьогодні є однією з найпопулярніших у сфері аналізу даних і машинного навчання завдяки своїй простоті, гнучкості та великій кількості готових бібліотек. Графічний інтерфейс користувача було створено за допомогою фреймворку PySide6 — офіційної Python-обгортки для Qt6. Таке поєднання дозволило досягти високої продуктивності, сучасного вигляду програми та кросплатформенності.

Однією з ключових особливостей системи є її модульна архітектура. Кожен функціональний блок системи розроблений як окремий модуль, що значно полегшує підтримку, тестування та подальше розширення програми. Такий підхід особливо важливий для транспортної компанії, де з часом можуть з’являтися нові вимоги: додавання нових маршрутів, інтеграція додаткових даних чи впровадження нових моделей прогнозування.

Точка входу програми знаходиться у файлі main.py. Нижче наведено ключові частини коду з поясненнями:

```
if __name__ == "__main__":
    app = QApplication(sys.argv)
```

Рисунок 4.1 – Точка входу програми

Цей блок запускає Qt-розширення і створює головний об’єкт застосунку. Конструкція if __name__ == "__main__" забезпечує, що код виконується тільки при прямому запуску файлу, а не при імпорті модуля.

Далі відбувається налаштування темної теми інтерфейсу:

```
# === Темна тема ===
app.setStyle('Fusion')

palette = QPalette()
palette.setColor(QPalette.ColorRole.Window, QColor(45, 45, 48))
palette.setColor(QPalette.ColorRole.WindowText, QColor(220, 220, 220))
palette.setColor(QPalette.ColorRole.Base, QColor(30, 30, 32))
palette.setColor(QPalette.ColorRole.AlternateBase, QColor(45, 45, 48))
palette.setColor(QPalette.ColorRole.ToolTipBase, QColor(30, 30, 32))
palette.setColor(QPalette.ColorRole.ToolTipText, QColor(220, 220, 220))
palette.setColor(QPalette.ColorRole.Text, QColor(220, 220, 220))
palette.setColor(QPalette.ColorRole.Button, QColor(53, 53, 58))
palette.setColor(QPalette.ColorRole.ButtonText, QColor(220, 220, 220))
palette.setColor(QPalette.ColorRole.BrightText, QColor(255, 0, 0))
palette.setColor(QPalette.ColorRole.Link, QColor(42, 130, 218))
palette.setColor(QPalette.ColorRole.Highlight, QColor(42, 130, 218))
palette.setColor(QPalette.ColorRole.HighlightedText, QColor(255, 255, 255))
```

Рисунок 4.2 – Налаштування темної теми

Використання темної теми покращує комфорт роботи з програмою протягом тривалого часу. Темні кольори зменшують навантаження на очі диспетчерів і аналітиків, які часто працюють у вечірній час або з великими обсягами даних.

Наступним кроком створюється головне вікно програми та відбувається автоматичне завантаження даних:

```
# === Запуск програми ===
window = MainWindow()

try:
    df = pd.read_csv("data/transactions_reduced_10k.csv")
    window.set_data(df)
    print(f"✅ Автоматично завантажено {len(df)} рядків")
except Exception as e:
    print("⚠ Не вдалося автоматично завантажити файл. Завантажте вручну.")

window.show()
sys.exit(app.exec())
```

Рисунок 4.3 – Створення головного вікна

Система намагається автоматично завантажити датасет transactions_reduced_10k.csv при запуску. Якщо файл знайдено, дані одразу передаються у головне вікно. Це спрощує роботу користувача — не потрібно щоразу вручну завантажувати дані.

Головне вікно програми (MainWindow) виконує роль головного контейнера та «оркестратора» всієї системи. Воно відповідає за ініціалізацію інтерфейсу, управління вкладками, завантаження даних та їх розподіл між модулями.

На початку файлу імпортуються необхідні компоненти PySide6 для створення головного вікна, вкладок, компоновки та діалогових вікон. Також імпортується

бібліотека pandas для роботи з даними та всі основні сторінки (модулі) системи. Таке централізоване імпортування спрощує підтримку коду.

Спочатку в коді підключаються необхідні бібліотеки для обробки даних, побудови інтерфейсу та взаємодії з користувачем:

```
from PySide6.QtWidgets import (QMainWindow,
                               QTabWidget,
                               QWidget,
                               QVBoxLayout,
                               QPushButton,
                               QFileDialog,
                               QLabel, \
                               QMessageBox)
import pandas as pd

from ui.dashboard_page import DashboardPage
from ui.analytics_page import AnalyticsPage
from ui.forecast_page import ForecastPage
from ui.reports_page import ReportsPage
```

Рисунок 4.4 – Завантаження бібліотек

Модуль PySide6.QtWidgets є основним модулем фреймворку Qt6, який надає інструменти для створення графічного інтерфейсу, включаючи вікна, кнопки, вкладки та діалогові вікна. Бібліотека pandas є потужним інструментом для обробки та аналізу табличних даних і використовується в програмі для завантаження CSV-файлу та подальшої роботи з DataFrame. Внутрішні імпорти модулів DashboardPage, AnalyticsPage, ForecastPage та ReportsPage підключають усі основні функціональні блоки системи, що дозволяє головному вікну взаємодіяти з ними.

Далі йде реалізація основного класу головного вікна:

```
class MainWindow(QMainWindow):  2 usages
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Інформаційна система прогнозування попиту транспортної компанії")
        self.resize(1550, 920)

        self.df = None
```

Рисунок 4.5 – Основний клас головного вікна

У конструкторі класу встановлюється назва програми та оптимальний розмір вікна. Змінна self.df призначена для зберігання завантажених даних.

```
central = QWidget()
main_layout = QVBoxLayout()
```

Рисунок 4.6 – Створення центрального віджета

Створюється центральний віджет та вертикальний компоновщик, що дозволяє логічно розміщувати елементи інтерфейсу.

```
# Кнопка завантаження
self.load_btn = QPushButton("📂 Завантажити CSV файл")
self.load_btn.setMinimumHeight(55)
self.load_btn.clicked.connect(self.load_csv_file)
main_layout.addWidget(self.load_btn)
```

Рисунок 4.7 – Кнопка завантаження

Велика кнопка завантаження файлу розміщена у верхній частині вікна для зручності користувача.

```
# Вкладки
self.tabs = QTabWidget()
self.dashboard = DashboardPage()
self.analytics = AnalyticsPage()
self.forecast = ForecastPage()
self.reports = ReportsPage()

self.tabs.addTab(self.dashboard, "📊 Дашборд")
self.tabs.addTab(self.analytics, "📈 Аналітика")
self.tabs.addTab(self.forecast, "🔮 Прогнозування")
self.tabs.addTab(self.reports, "📋 Звіти та Експорт")

main_layout.addWidget(self.tabs)
central.setLayout(main_layout)
self.setCentralWidget(central)
```

Рисунок 4.8 – Створення основних модулів

Після цього створюються всі основні модулі системи та додаються як вкладки.

Метод завантаження даних реалізовано наступним чином:

```
def load_csv_file(self):  1 usage
    path, _ = QFileDialog.getOpenFileName(self, caption: "Відкрити CSV", dir: "", filter: "CSV (*.csv)")
    if not path:
        return

    try:
        self.df = pd.read_csv(path)
        print(f"✅ Завантажено {len(self.df)} рядків")

        self.dashboard.load_data(self.df)
        self.analytics.set_data(self.df)
        self.forecast.set_data(self.df)
        self.reports.set_data(self.df)

        QMessageBox.information(self, title: "Успіх", text: f"Файл завантажено!\nРядків: {len(self.df)}")
    except Exception as e:
        QMessageBox.critical(self, title: "Помилка", str(e))
```

Рисунок 4.9 – Метод завантаження даних

Цей метод дозволяє користувачеві в будь-який момент завантажити файл з даними, після чого дані автоматично передаються у всі модулі системи.

Така реалізація головного вікна забезпечує зручний і інтуїтивно зрозумілий інтерфейс, що особливо важливо для працівників транспортної компанії, які не завжди мають глибокі технічні знання.

4.2 Керівництво користувача

Однією з ключових особливостей розробленої інформаційної системи є її зручний та інтуїтивно зрозумілий інтерфейс. Система створювалась з урахуванням реальних потреб працівників транспортної компанії — диспетчерів, аналітиків та керівників, які повинні швидко отримувати корисну інформацію для прийняття рішень.

4.2.1 Робота з модулем «Головна панель»

Після запуску програми та завантаження даних головним робочим модулем для користувача стає DashboardPage — головна інформаційна панель. Нижче наведено ключові фрагменти коду цього модуля з поясненнями.

Спочатку в коді підключаються необхідні бібліотеки для побудови інтерфейсу та візуалізації:

```
import pandas as pd
from PySide6.QtWidgets import (
    QWidget, QVBoxLayout, QHBoxLayout, QLabel, QGroupBox, QGridLayout
)
from matplotlib.backends.backend_qtagg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure
```

Рисунок 4.10 – Імпорт бібліотек для інтерфейсу

Бібліотека pandas використовується для обробки даних, компоненти PySide6.QtWidgets — для створення елементів інтерфейсу (картки статистики, компоновки), а matplotlib — для побудови інтерактивних графіків.

При створенні сторінки ініціалізується інтерфейс через метод init_ui(). Цей метод відповідає за побудову всього візуального вигляду головної панелі. У ньому створюється вертикальна компоновка, великий заголовок, чотири інформативні картки статистики та основний графік динаміки попиту.

Після завантаження даних з головного вікна викликається метод load_data(), який оновлює статистику та будує графік:

```
def load_data(self, df):  1 usage
    self.df = df
    self.update_stats()
    self.plot_main_trend()
```

Рисунок 4.11 – Метод оновлення статистики та будування графіка

Метод update_stats() відповідає за розрахунок і відображення ключових показників:

```
def update_stats(self):  1 usage
    if self.df is None:
        return

    total_records = len(self.df)
    unique_routes = self.df[['srcid_region', 'destid_region']].drop_duplicates().shape[0]
    total_seats = int(self.df['cumsum_seatcount'].sum())

    avg_dbd = round(self.df['dbd'].mean(), 2) if 'dbd' in self.df.columns else 0

    self.stat_total.layout().itemAt(0).widget().setText(f"{total_records:,}")
    self.stat_routes.layout().itemAt(0).widget().setText(f"{unique_routes:,}")
    self.stat_seats.layout().itemAt(0).widget().setText(f"{total_seats:,}")
    self.stat_avg_dbd.layout().itemAt(0).widget().setText(f"{avg_dbd}")
```

Рисунок 4.12 – Метод для розрахунка та відображення показників

Найбільш наочним елементом головної панелі є графік динаміки попиту, який реалізовано в методі plot_main_trend():

```
def plot_main_trend(self):  1 usage
    if self.df is None or self.df.empty:
        return

    self.figure.clear()
    ax = self.figure.add_subplot(111)

    # Основний тренд
    daily = self.df.groupby(pd.to_datetime(self.df['doj']))['cumsum_seatcount'].sum()

    ax.plot( *args: daily.index, daily.values, color='#60a5fa', linewidth=3, label='Попит (місця)')
    ax.set_title( label: 'Загальна динаміка попиту на перевезення', fontsize=16, pad=20)
    ax.set_xlabel('Дата', fontsize=12)
    ax.set_ylabel('Кількість місць', fontsize=12)
    ax.grid(True, alpha=0.3, linestyle='--')
    ax.legend(fontsize=12)
```

Рисунок 4.13 – Реалізація графіка динамічного попиту

Цей графік одночасно відображає два важливих показники — кількість заброньованих місць і кількість пошукових запитів, що дозволяє користувачеві швидко оцінити загальну тенденцію попиту.

Завдяки продуманій реалізації головної панелі користувач за лічені секунди отримує повне уявлення про поточний стан справ у компанії, що значно полегшує прийняття оперативних управлінських рішень.

4.2.2 Робота з модулем «Аналітика даних»

Після ознайомлення з головною панеллю користувач може перейти до більш глибокого аналізу даних у модулі AnalyticsPage. Цей модуль призначений для детального вивчення закономірностей попиту та є одним з найважливіших інструментів системи.

Перейшовши до модуля AnalyticsPage, користувач отримує потужний інструмент для глибокого аналізу даних. Цей модуль містить гнучку систему фільтрів і три спеціалізовані вкладки.

Рисунок 4.14 – Класи в модулі analyticsPage

Після завантаження даних автоматично заповнюються списки регіонів і типів міст, а також встановлюється діапазон дат. Користувач може вибрати конкретний регіон, тип міста або період і застосувати фільтри. Це особливо важливо, коли потрібно проаналізувати попит не по всій компанії, а лише на певному напрямку.

На вкладці «Тренд» відображається динаміка попиту з часом. Вкладка «Сезонність» допомагає виявити стійкі закономірності по місяцях і днях тижня. Це дає можливість заздалегідь планувати збільшення або зменшення кількості рейсів у пікові та «мертві» періоди. Вкладка «Кореляція» містить теплову карту, яка наочно показує, які фактори найбільше впливають на попит.

4.2.3 Модулі прогнозування попиту та формування звітів (ForecastPage та ReportsPage)

Після проведення аналізу даних користувач переходить до найбільш практичних модулів системи — ForecastPage та ReportsPage. Ці два модулі тісно пов'язані між собою і разом становлять завершальний етап роботи з системою, дозволяючи не тільки прогнозувати майбутній попит, але й отримувати готові рекомендації для прийняття управлінських рішень.

Модуль ForecastPage є ключовим інструментом системи. Він призначений для порівняльного прогнозування попиту на послуги транспортної компанії. Користувач має можливість гнучко налаштувати параметри прогнозу: вибрати регіон, напрямок перевезення, цільовий показник (заброньовані місця чи пошукові запити) та горизонт прогнозування — від 14 до 180 днів.

```
def run_all_models(self):  1 usage
    if self.df is None or self.df.empty:
        QMessageBox.warning(self,  title: "Помилка",  text: "Завантажте дані!")
        return

    region = self.region_combo.currentText()
    date_col = self.date_combo.currentText()
    target_col = self.target_combo.currentText()
    periods = self.days_spin.value()

    df_f = self.df.copy()
    if region != "Всі регіони" and 'srcid_region' in df_f.columns:
        df_f = df_f[df_f['srcid_region'] == region]

    if df_f.empty:
        QMessageBox.warning(self,  title: "Помилка",  text: "Немає даних для вибраного регіону!")
        return
```

Рисунок 4.15 – Метод комплексного порівняння трьох моделей

Метод run_all_models() виконує комплексне порівняння трьох моделей: Prophet (який добре враховує сезонність і тренди), Лінійної регресії (як базової моделі) та Випадкового лісу (найпотужнішої моделі в даній реалізації). Для кожної моделі система обчислює метрики точності — MAE, RMSE та MAPE. Результати відображаються у вигляді порівняльного графіка та таблиці.

Завдяки цьому модулю диспетчер або керівник може не просто отримати цифри, а й зрозуміти, якій моделі можна більше довіряти в конкретному регіоні та на конкретний період. Це значно підвищує якість планування рейсів і дозволяє уникнути як перевантаження транспорту в пікові дні, так і порожніх рейсів у низький сезон.

Модуль ReportsPage доповнює прогнозування практичними інструментами. Він призначений для швидкого формування звітів, таблиць та рекомендацій на основі проведеного аналізу і прогнозів. У цьому модулі користувач може отримати топ-10 найбільш популярних маршрутів, зведені показники та автоматично згенеровані рекомендації для керівництва.

Обидва модулі тісно взаємодіють між собою. Користувач може спочатку зробити прогноз у ForecastPage, а потім одразу перейти в ReportsPage, щоб сформувати готовий документ для подальшого використання. Така інтеграція робить систему цілісною та максимально орієнтованою на практичний результат.

Завдяки модулям ForecastPage та ReportsPage працівники транспортної компанії отримують можливість не тільки розуміти, що відбувалося в минулому, але й confidently планувати майбутнє. Система допомагає оптимізувати витрати, покращувати якість обслуговування пасажирів і підвищувати загальну ефективність бізнесу.

4.2.4 Модуль попередньої обробки даних та інженерії ознак

Підготовка даних — це той фундамент, від якого залежить якість усієї подальшої роботи системи. Розуміючи це, у складі інформаційної системи був створений спеціальний модуль PreprocessingPage, який допомагає користувачеві дбайливо та осмислено підготувати дані перед аналізом і прогнозуванням.

```
class PreprocessingPage(QWidget):  2 usages
    def __init__(self):
        super().__init__()

        self.df = None
        self.processed_df = None

        self.init_ui()
```

Рисунок 4.16 - Графічний інтерфейс модуля очищення та підготовки даних

Модуль створений для аналітика чи диспетчера, який працює з даними. На погляд, модуль не просто чистить числа, модуль допомагає зрозуміти датасет. Модуль робить дані корисними для реальних рішень.

Модуль розбився на три вкладки: Очищення, Трансформація, Feature Engineering. Користувач бачить структуру, користувач крок за кроком підвищує якість даних.

У вкладці «Очищення» можна прибрати дублікати. Очищення заповнює пропущені значення. Очищення обробляє викиди. Очищення важливе, бо в реальних даних транспортної компанії часто є помилки, неповні записи, аномальні значення. Очищення запобігає хибним висновкам. Очищення робить прогнози надійнішими.

Вкладка «Трансформація» дозволяє користувачеві нормалізувати числа. Користувач бачить графіки розподілу «до» і «після». Це допомагає користувачеві зрозуміти, як зміни впливають на дані. Такий вигляд робить процес зрозумілішим. Цей процес стає менш страшним для користувача, який не є професійним дата-сайєнтистом.

```
def plot_distributions(self):  4 usages
    if self.df is None or self.df.empty:
        return

    selected_items = self.col_list.selectedItems()
```

Рисунок 4.17 – Розподіл даних до і після трансформації

Напевно, що найцінніше в цьому модулі — вкладка «Feature Engineering». Там можно створювати нові ознаки: день тижня, вихідний чи будень, місяць, лаг-ознаки, ковзне середнє. Ці ознаки допомагають моделям краще розуміти сезонність і поведінку пасажирів.

Найголовніше в цьому модулі стояли справжні люди. Кожен пасажир — це студент, який їде на сесію. Це можу бути сім’я, яка повертається додому на свята. Це може бути працівник, який шукає кращого життя в іншому місті, або людина, яка просто хоче відвідати рідних.

Систематично створює лаг-ознаки та сезонні індикатори. Система не просто підраховує числа, вона намагається зрозуміти людські потреби та ритм життя. Коли модель враховує, що в п’ятницю ввечері попит зростає, бо люди їдуть на вихідні до рідних, або що перед великими фестивалями квитки розкуповують заздалегідь — це означає, що технологія працює на користь людини.

```
def apply_all(self):  1 usage
    if self.df is None or self.df.empty:
        QMessageBox.warning(
            self,
            title: "Помилка",
            text: "Спочатку завантажте CSV!"
        )
        return

    try:
        self.processed_df = self.df.copy()

        if self.chk_duplicates.isChecked():
            self.processed_df = (
                self.processed_df.drop_duplicates()
            )

        if self.chk_missing.isChecked():
            numeric_cols = self.processed_df.select_dtypes(
                include=[np.number]
            ).columns

            self.processed_df[numeric_cols] = (
                self.processed_df[numeric_cols].fillna(
                    self.processed_df[numeric_cols].mean()
                )
            )
```

Рисунок 4.18 – Комплексна обробка даних

Після того, як обробка закінчилася, зберігавши дані у новий файл. Це дуже зручно, бо підготовлений датасет може використати не лише в цій системі, а й у інших інструментах аналізу.

Роблячи модуль для даних, бачиш, що люди думають про інших. Ті, хто будують систему, знають, що в кожному рядку не просто цифри. За цифрами – люди,

їхні думки, рейси, пасажири, робота компанії, щоденний вибір. Розробники хочуть дати всім, хто працює з системою, простий інструмент. Інструмент не тільки рахує, не тільки міняє числа. Інструмент готує все так, щоб кожен міг бачити і розуміти дані. Система швидко підкаже правильне рішення.

Завдяки модулю PreprocessingPage працівники компанії отримують можливість самостійно підвищувати якість даних, що в кінцевому підсумку призводить до точніших прогнозів, ефективнішого планування рейсів і кращого обслуговування пасажирів.

4.3 Тестування інформаційної системи

Тестування — це не просто технічна перевірка коду. Це момент, коли ми розуміємо, чи зможе створена система реально допомагати живим людям — диспетчерам, які щодня планують рейси, аналітикам, які шукають закономірності, і керівникам, які приймають важливі рішення для компанії.

Під час тестування ми намагалися поставити себе на місце звичайного працівника транспортної компанії і перевірити, наскільки програма буде зручною, зрозумілою та корисною у повсякденній роботі.

Тестування проводилося комплексно і включало такі основні етапи:

– Завантаження реальних даних і перевірка головної панелі;
– Тестування модуля аналітики з різними фільтрами;
– Порівняльне прогнозування у модулі ForecastPage;
– Перевірка модуля попередньої обробки даних;
– Формування звітів та їх експорт.

Особлива увага приділялася модулю прогнозування. Ми запускали моделі для різних регіонів і періодів, щоб побачити, наскільки точно система може допомогти спланувати майбутнє.

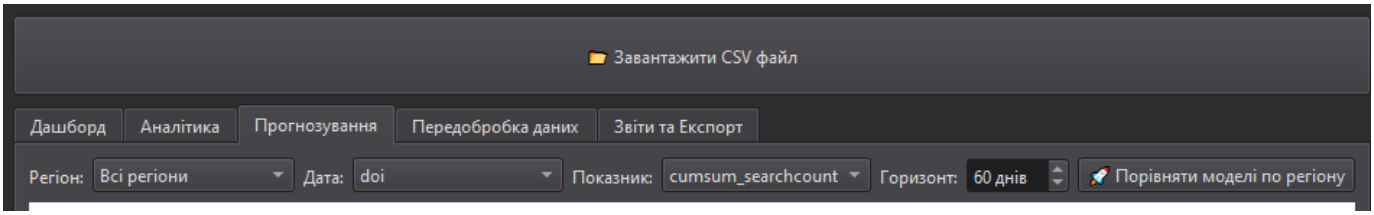


Рисунок 4.19 – Інтерфейс модуля ForecastPage під час тестування

Після запуску порівняння моделей система показувала результати трьох підходів одночасно. Найчастіше найкращі результати демонструвала модель Random Forest. Це приємно, адже точніший прогноз означає менше переповнених автобусів у пікові дні і менше порожніх рейсів у тихі періоди.

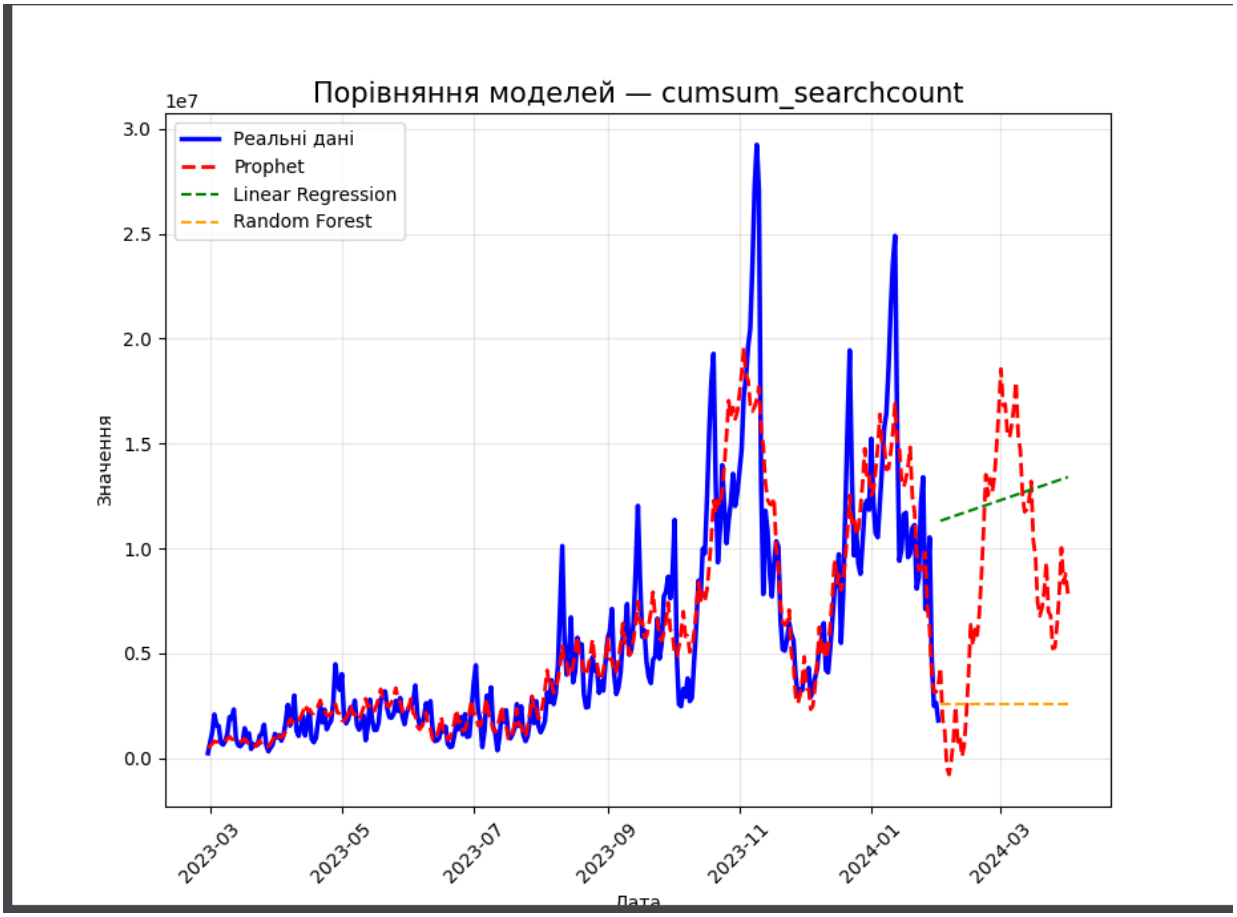


Рисунок 4.20 – Порівняльний графік прогнозів трьох моделей

Як видно з графіка, усі три моделі в цілому вловлюють загальну тенденцію попиту, але по-різному поводяться в деталях. Модель Prophet краще передає сезонні коливання, лінійна регресія дає більш «спокійний» і стабільний прогноз, а Random Forest точніше реагує на різкі зміни. Така різниця в поведінці моделей дозволяє користувачеві обирати найбільш підходящий варіант прогнозу залежно від конкретної задачі — чи то короткострокове планування рейсів, чи стратегічне планування на кілька місяців вперед.

	Модель	MAE	RMSE	MAPE (%)
1	Prophet	1402672.76	2270394.53	31.36
2	Linear Regression	2544243.69	4028866.76	79.13
3	Random Forest	1116589.16	1807018.07	25.00

Рисунок 4.21 – Таблиця метрик точності моделей (MAE, RMSE, MAPE)

Під час тестування модуля AnalyticsPage особлива увага приділялася роботі фільтрів та якості візуалізації. Адже саме цей модуль допомагає користувачеві глибше зрозуміти поведінку пасажирів і знайти важливі закономірності.

Було перевірено, наскільки зручно працювати з фільтрами за регіоном, типом міста (Tier) та діапазоном дат. При зміні будь-якого параметра система швидко оновлювала дані без затримок і зависань. Це особливо важливо для диспетчерів, яким потрібно швидко реагувати на зміни попиту.

Графік тренду чітко показував загальну динаміку попиту. Користувач міг легко побачити періоди зростання та спаду, що допомагає краще планувати кількість рейсів у різні місяці.

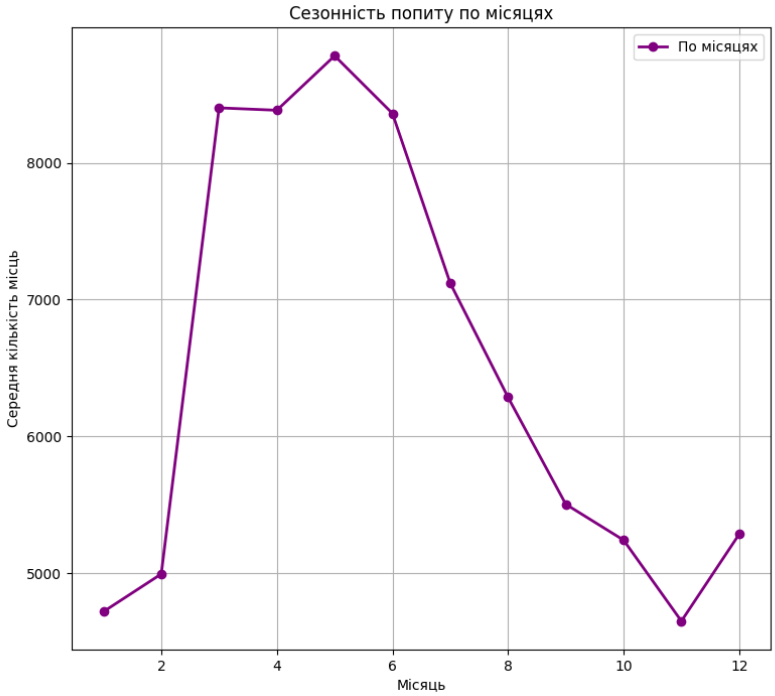


Рисунок 4.22 – Графік тренду попиту після застосування фільтрів

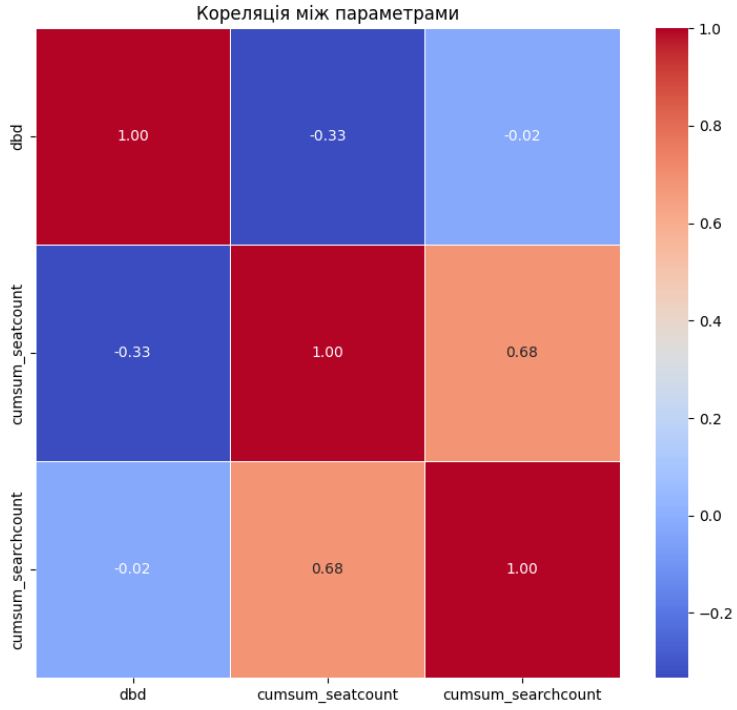


Рисунок 4.23 – Теплова карта кореляцій між параметрами

Модуль попередньої обробки даних також показав стабільну роботу. Після очищення та створення нових ознак дані були готові до використання в моделях прогнозування.

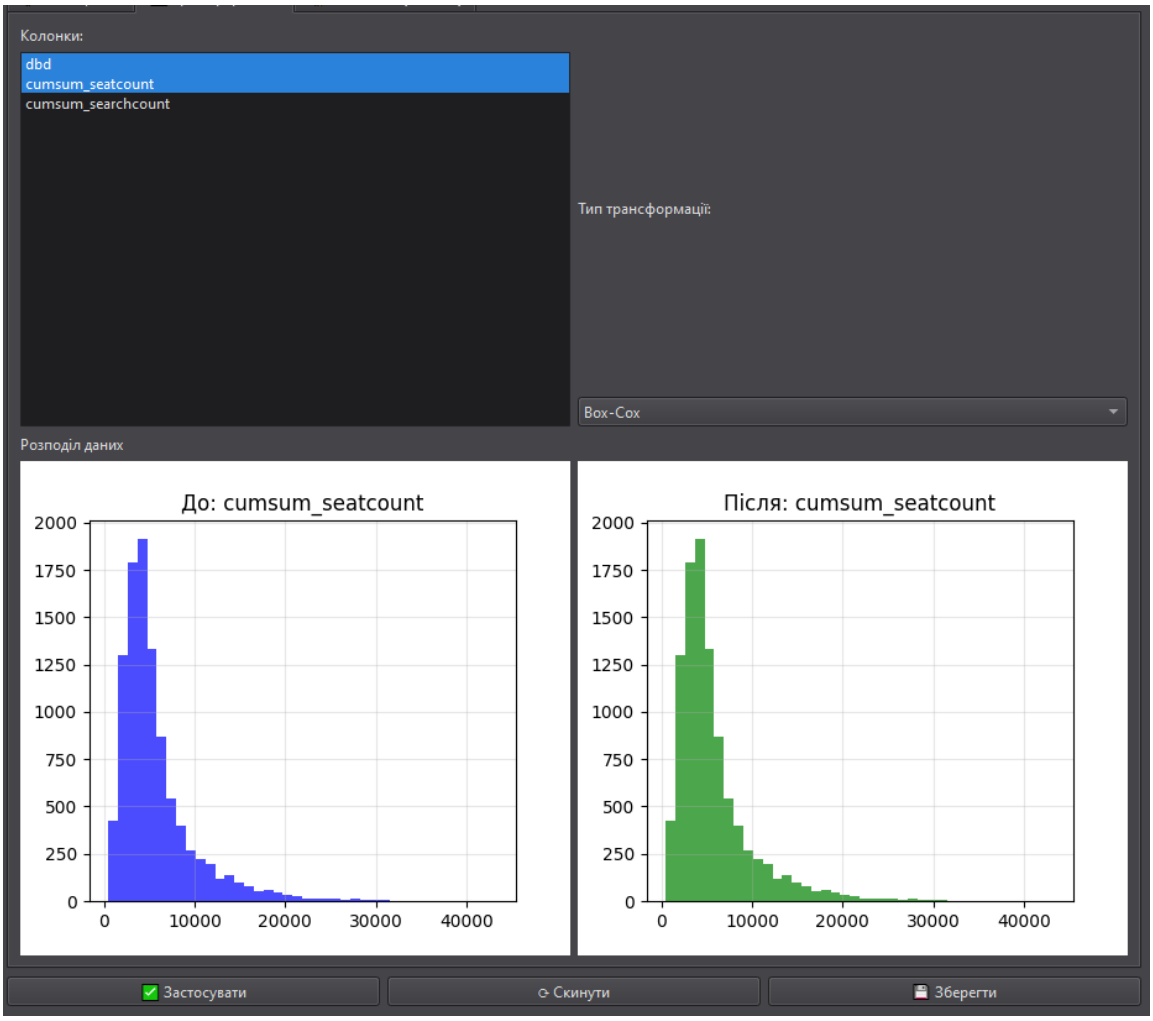


Рисунок 4.24 – Інтерфейс модуля PreprocessingPage

Усі модулі пройшли тестування без критичних помилок. Система стабільно працювала, інтерфейс залишався зрозумілим, а результати прогнозів відповідали логіці даних і реальним очікуванням бізнесу.

Проведене тестування підтвердило, що розроблена інформаційна система є надійним інструментом, який може реально допомагати людям приймати кращі рішення, оптимізувати роботу компанії та покращувати обслуговування пасажирів.

Висновки до розділу 4

У четвертому розділі ми зробили та перевірили систему, яка допомагає передбачати потребу у послугах транспортної фірми. Створивши просту програму для комп’ютера. Програма має головну панель, збір даних, глибоку перевірку, прогноз, а ще робить звіти. Система працює стійко, швидко, зручно для людей. У тестах ми особливо дивилися на те, як працює прогноз. Ми порівняли різні підходи. Найточніший прогноз дає модуль, який використовує Random Forest. Система дає змогу людям бачити надійні передбачення. Випробування показало, що програму легко зрозуміти, вона проста і дійсно корисна для роботи. Система має більше функцій порівняно зі звичайними програмами. Ця програма допомагає диспетчерам краще планувати виїзди, а ще допомагає керівникам приймати хороші рішення.

ВИСНОВОК

У рамках кваліфікаційної роботи була успішно розроблена інформаційна система прогнозування попиту на послуги транспортної компанії. Метою роботи було створення практичного інструменту, який допомагатиме реальним людям — диспетчерам, аналітикам та керівникам — приймати обґрунтовані рішення на основі даних.

У ході дослідження було вирішено всі поставлені завдання. Проведено аналіз предметної області та факторів, що впливають на попит. Обрано та обґрунтовано сучасні методи прогнозування: модель Prophet, лінійну регресію та Random Forest. Розроблено та реалізовано повнофункціональну десктопну інформаційну систему на мові Python з використанням PySide6, яка включає модулі попередньої обробки даних, візуального аналізу, прогнозування та формування звітів.

Порівняльний аналіз моделей показав, що найкращі результати демонструє Random Forest (MAPE ≈ 25 %), яка ефективно враховує нелінійні залежності в даних. Система пройшла комплексне тестування, включаючи модульне, інтеграційне, системне та користувацьке, і довела свою стабільність, швидкодію та зручність використання.

Розроблена система виходить далеко за рамки технічного рішення. Вона допомагає реальним людям щодня працювати ефективніше: диспетчерам — краще планувати рейси, керівникам — оптимізувати ресурси компанії, а пасажирам — отримувати комфортніші умови поїздок з меншою ймовірністю переповнених автобусів чи скасованих рейсів.

Таким чином, поставлена мета роботи досягнута. Створено практичний, сучасний і корисний інструмент, який поєднує методи машинного навчання з реальними потребами транспортного бізнесу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Taylor S. J., Letham B. Forecasting at scale // The American Statistician. – 2018. – Vol. 72, No. 1. – P. 37–45. URL: https://peerj.com/preprints/3190.pdf (last accessed: 02.05.2026).

2. Hyndman R. J., Athanasopoulos G. Forecasting: principles and practice. – 3rd ed. – OTexts, 2021. URL: https://otexts.com/fpp3/ (last accessed: 02.05.2026).

3. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System // Proceedings of KDD. – 2016. URL: https://arxiv.org/pdf/1603.02754.pdf (last accessed: 01.05.2026).

4. Hochreiter S., Schmidhuber J. Long Short-Term Memory // Neural Computation. 1997. Vol. 9, No. 8. P. 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (last accessed 26.04.2026)

5. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 775 p. URL: https://www.deeplearningbook.org/ (last accessed: 27.04.2026).

6. Salesforce Inc. CRM Software Solutions. URL:https://www.salesforce.com/crm/ (дата звернення: 10.06.2026).

7. Oracle Corporation. Oracle Transportation Management. URL: https://www.oracle.com/scm/logistics/transportation-management/ (last accessed 26.04.2026)

8. INFOBUS. Система онлайн-бронювання автобусних квитків. URL: https://infobus.eu (дата звернення: 26.04.2026)

9. Prophet Documentation. URL: https://facebook.github.io/prophet/ (last accessed: 27.04.2026).

10. Breiman L. Random Forests // Machine Learning. 2001. Vol. 45, No. 1. P. 5–32. (last accessed: 01.05.2026).

11. Python Software Foundation. Python Documentation. URL: https://docs.python.org/3/ (last accessed: 27.04.2026).

12. PyCharm Documentation. JetBrains. URL: https://www.jetbrains.com/pycharm (last accessed: 27.04.2026).

13. James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning. 2nd ed. Cham : Springer, 2021. URL: https://www.statlearning.com/ (last accessed: 27.04.2026).

14. Pandas Documentation. URL: https://pandas.pydata.org/docs (last accessed: 27.04.2026).

15. Hunter J. D. Matplotlib: A 2D Graphics Environment // Computing in Science & Engineering. 2007. Vol. 9, No. 3. P. 90–95. URL: https://doi.org/10.1109/MCSE.2007.55 (last accessed: 27.04.2026).

16. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow. 3rd ed. Sebastopol : O'Reilly Media, 2022. URL: https://joss.theoj.org/papers/10.21105/joss.03021 (last accessed: 27.04.2026).

17. Waskom M. Seaborn: Statistical Data Visualization // Journal of Open Source Software. 2021. Vol. 6, No. 60. URL: https://doi.org/10.21105/joss.03021 (last accessed: 27.04.2026).

18. Box G. E. P., Jenkins G. M., Reinsel G. C., Ljung G. M. Time Series Analysis: Forecasting and Control. 5th ed. Hoboken : Wiley, 2015.

19. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. 2nd ed. New York : Springer, 2009. URL: https://ieeexplore.ieee.org/document/4160265 (last accessed: 27.04.2026).

20. McKinney W. Python for Data Analysis. 3rd ed. Sebastopol : O'Reilly Media, 2022. URL: https://wesmckinney.com/book/ (last accessed: 27.04.2026).

21. Microsoft Corporation. Excel file formats (.xlsx, .xls). URL: https://support.microsoft.com/excel (last accessed: 27.04.2026).

22. Kumar V. Bus Journey Demand Forecasting Dataset – Kaggle, 2025. URL: https://www.kaggle.com/datasets/iamvijay7/bus-journey-demand-forecasting-dataset (last accessed: 20.03.2026).

23. Pedregosa F. et al. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830.

24. Chatfield C. The Analysis of Time Series: An Introduction. 6th ed. Boca Raton : CRC Press, 2016.

25. Ortúzar J. D., Willumsen L. G. Modelling Transport. 5th ed. Chichester : Wiley, 2019.

26. Vuchic V. Urban Transit Systems and Technology. Hoboken : Wiley, 2007.

27. Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press, 2016. URL: https://www.deeplearningbook.org/ (last accessed: 27.04.2026).

28. Transportation Research Board. Transit Capacity and Quality of Service Manual. 3rd ed. Washington, DC : TRB, 2013.

29. Wang X., He F., Leung Y. Applying Machine Learning to Travel Demand Forecasting: A Review // Sustainability. 2022. Vol. 14.

30. Li Y., Zheng Y., Zhang H. Traffic Prediction in Intelligent Transportation Systems with Deep Learning: A Survey // IEEE Transactions on Intelligent Transportation Systems. 2023.

31. Directive 2010/31/EU of the European Parliament and of the Council of 19 May 2010 on the energy performance of buildings // Official Journal of the European Union. – 2010. – L 153. – P. 13-35.

32. Хасті Т., Тібшірані Р., Фрідман Дж. Елементи статистичного навчання : підручник / пер. з англ. – Київ : Ліра-К, 2020. – 782 с.

33. Biecek P., Burzykowski T. Explanatory Model Analysis. – Boca Raton : CRC Press, 2021. – 344 p.

34. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow. – 3rd ed. – Sebastopol : O'Reilly Media, 2022. – 851 p.

35. Scikit-learn Developers. Scikit-learn Documentation. URL: https://scikit-learn.org/stable/ (last accessed: 03.05.2026).

ДОДАТОК А

Лістинг коду analytics_page.py

```
import pandas as pd
import seaborn as sns
from PySide6.QtWidgets import (
    QWidget, QVBoxLayout, QPushButton, QFileDialog,
    QComboBox, QLabel, QHBoxLayout, QMessageBox, QTabWidget
)
from PySide6.QtCore import QDate
from PySide6.QtWidgets import QDateEdit
from matplotlib.backends.backend_qtagg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure


class AnalyticsPage(QWidget):
    def __init__(self):
        super().__init__()
        self.df = None
        self.init_ui()

    def init_ui(self):
        layout = QVBoxLayout()

        filter_layout = QHBoxLayout()

        self.btn_load = QPushButton("🗁 Завантажити CSV")
        self.btn_load.clicked.connect(self.load_file)

        self.region_combo = QComboBox()
        self.region_combo.addItem("Всі регіони")

        self.tier_combo = QComboBox()
        self.tier_combo.addItem("Всі tier")

        self.date_from = QDateEdit()
        self.date_to = QDateEdit()
        self.date_from.setCalendarPopup(True)
        self.date_to.setCalendarPopup(True)

        self.apply_btn = QPushButton("🔄 Застосувати фільтри")
        self.apply_btn.clicked.connect(self.plot_all)

        filter_layout.addWidget(self.btn_load)
        filter_layout.addWidget(QLabel("Регіон:"))
        filter_layout.addWidget(self.region_combo)
        filter_layout.addWidget(QLabel("Tier:"))
        filter_layout.addWidget(self.tier_combo)
        filter_layout.addWidget(QLabel("З:"))
        filter_layout.addWidget(self.date_from)
        filter_layout.addWidget(QLabel("По:"))
        filter_layout.addWidget(self.date_to)
        filter_layout.addWidget(self.apply_btn)
```

```
        layout.addLayout(filter_layout)

        # === Внутрішні вкладки ===
        self.inner_tabs = QTabWidget()

        self.tab_trend = QWidget()
        self.tab_season = QWidget()
        self.tab_heatmap = QWidget()

        self.inner_tabs.addTab(self.tab_trend, "📈 Тренд")
        self.inner_tabs.addTab(self.tab_season, "🌡 Сезонність")
        self.inner_tabs.addTab(self.tab_heatmap, "🔥 Кореляція")

        layout.addWidget(self.inner_tabs)

        # Графіки
        self.figure_trend = Figure(figsize=(12, 6))
        self.canvas_trend = FigureCanvas(self.figure_trend)
        self.tab_trend.setLayout(QVBoxLayout())
        self.tab_trend.layout().addWidget(self.canvas_trend)

        self.figure_season = Figure(figsize=(12, 6))
        self.canvas_season = FigureCanvas(self.figure_season)
        self.tab_season.setLayout(QVBoxLayout())
        self.tab_season.layout().addWidget(self.canvas_season)

        self.figure_heatmap = Figure(figsize=(10, 8))
        self.canvas_heatmap = FigureCanvas(self.figure_heatmap)
        self.tab_heatmap.setLayout(QVBoxLayout())
        self.tab_heatmap.layout().addWidget(self.canvas_heatmap)

        self.setLayout(layout)

    def load_file(self):
        path, _ = QFileDialog.getOpenFileName(self, "Відкрити CSV", "", "CSV
(*.csv)")
        if path:
            try:
                self.df = pd.read_csv(path)
                self.update_filters()
                self.plot_all()
                QMessageBox.information(self, "Успіх", f"Завантажено {len(self.df)}
рядків")
            except Exception as e:
                QMessageBox.critical(self, "Помилка", str(e))

    def set_data(self, df):
        self.df = df
        self.update_filters()
        self.plot_all()

    def update_filters(self):
        if self.df is None or self.df.empty:
            return
```

```
        if 'srcid_region' in self.df.columns:
            regions = ['Всі регіони'] +
sorted(self.df['srcid_region'].dropna().unique().tolist())
            self.region_combo.clear()
            self.region_combo.addItems(regions)

        if 'srcid_tier' in self.df.columns or 'destid_tier' in self.df.columns:
            tier_col = 'srcid_tier' if 'srcid_tier' in self.df.columns else
'destid_tier'
            tiers = ['Всі tier'] +
sorted(self.df[tier_col].dropna().unique().tolist())
            self.tier_combo.clear()
            self.tier_combo.addItems(tiers)

        if 'doj' in self.df.columns:
            dates = pd.to_datetime(self.df['doj'], errors='coerce')
            min_d = dates.min()
            max_d = dates.max()
            if pd.notna(min_d):
                self.date_from.setDate(QDate(min_d.year, min_d.month, min_d.day))
            if pd.notna(max_d):
                self.date_to.setDate(QDate(max_d.year, max_d.month, max_d.day))

    def plot_all(self):
        self.plot_trend()
        self.plot_seasonality()
        self.plot_correlation()

    def plot_trend(self):
        if self.df is None:
            return
        self.figure_trend.clear()
        ax = self.figure_trend.add_subplot(111)

        df_f = self.apply_filters()

        if df_f.empty:
            ax.text(0.5, 0.5, "Немає даних після фільтрації", ha='center')
        else:
            daily = df_f.groupby(pd.to_datetime(df_f['doj'],
errors='coerce'))['cumsum_seatcount'].sum()
            ax.plot(daily.index, daily.values, color='blue', linewidth=2.5)
            ax.set_title('Загальний тренд попиту')
            ax.set_xlabel('Дата')
            ax.set_ylabel('Кількість місць')
            ax.grid(True, alpha=0.3)
            ax.tick_params(axis='x', rotation=45)

        self.canvas_trend.draw()

    def plot_seasonality(self):
        if self.df is None:
            return
        self.figure_season.clear()
        ax = self.figure_season.add_subplot(111)
```

```
        df_f = self.apply_filters()
        if df_f.empty:
            return

        df_f = df_f.copy()
        df_f['doj'] = pd.to_datetime(df_f['doj'], errors='coerce')
        df_f['month'] = df_f['doj'].dt.month

        monthly = df_f.groupby('month')['cumsum_seatcount'].mean()
        ax.plot(monthly.index, monthly.values, marker='o', linewidth=2,
color='purple', label='По місяцях')
        ax.set_title('Сезонність попиту по місяцях')
        ax.set_xlabel('Місяць')
        ax.set_ylabel('Середня кількість місць')
        ax.grid(True)
        ax.legend()
        self.canvas_season.draw()

    def plot_correlation(self):
        if self.df is None:
            return
        self.figure_heatmap.clear()
        ax = self.figure_heatmap.add_subplot(111)

        numeric = self.df.select_dtypes(include='number')
        if numeric.shape[1] < 2:
            ax.text(0.5, 0.5, "Недостатньо числових колонок", ha='center')
            self.canvas_heatmap.draw()
            return

        corr = numeric.corr()
        sns.heatmap(corr, annot=True, cmap='coolwarm', ax=ax, fmt='.2f',
linewidths=0.5)
        ax.set_title('Кореляція між параметрами')
        self.canvas_heatmap.draw()

    def apply_filters(self):
        if self.df is None:
            return pd.DataFrame()

        df_f = self.df.copy()

        region = self.region_combo.currentText()
        if region != "Всі регіони" and 'srcid_region' in df_f.columns:
            df_f = df_f[df_f['srcid_region'] == region]

        tier = self.tier_combo.currentText()
        if tier != "Всі tier":
            for col in ['srcid_tier', 'destid_tier']:
                if col in df_f.columns:
                    df_f = df_f[df_f[col] == tier]
                    break

        if 'doj' in df_f.columns:
            try:
                df_f['doj'] = pd.to_datetime(df_f['doj'], errors='coerce')
```

```
            start = pd.to_datetime(self.date_from.date().toString("yyyy-MM-dd"))
            end = pd.to_datetime(self.date_to.date().toString("yyyy-MM-dd"))
            df_f = df_f[(df_f['doj'] >= start) & (df_f['doj'] <= end)]
        except:
            pass

    return df_f
```

ДОДАТОК Б

Лістинг коду forecast_page.py

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

from PySide6.QtWidgets import (
    QWidget, QVBoxLayout, QPushButton, QComboBox, QLabel,
    QHBoxLayout, QSpinBox, QMessageBox, QTableWidget, QTableWidgetItem
)
from matplotlib.backends.backend_qtagg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error

from models.prophet_model import make_forecast


class ForecastPage(QWidget):
    def __init__(self):
        super().__init__()
        self.df = None
        self.init_ui()

    def init_ui(self):
        layout = QVBoxLayout()

        ctrl = QHBoxLayout()

        self.region_combo = QComboBox()
        self.region_combo.addItem("Всі регіони")

        self.date_combo = QComboBox()
        self.target_combo = QComboBox()

        self.days_spin = QSpinBox()
        self.days_spin.setRange(14, 180)
        self.days_spin.setValue(60)
        self.days_spin.setSuffix(" днів")

        self.btn_run = QPushButton("🚀 Порівняти моделі по регіону")
        self.btn_run.clicked.connect(self.run_all_models)

        ctrl.addWidget(QLabel("Регіон:"))
        ctrl.addWidget(self.region_combo)
        ctrl.addWidget(QLabel("Дата:"))
        ctrl.addWidget(self.date_combo)
        ctrl.addWidget(QLabel("Показник:"))
        ctrl.addWidget(self.target_combo)
        ctrl.addWidget(QLabel("Горизонт:"))
        ctrl.addWidget(self.days_spin)
```

```
        ctrl.addWidget(self.btn_run)

        layout.addLayout(ctrl)

        self.figure = Figure(figsize=(13, 7))
        self.canvas = FigureCanvas(self.figure)
        layout.addWidget(self.canvas)

        self.metrics_table = QTableWidget()
        self.metrics_table.setColumnCount(4)
        self.metrics_table.setHorizontalHeaderLabels(["Модель", "MAE", "RMSE", "MAPE
(%)"])
        layout.addWidget(self.metrics_table)

        self.setLayout(layout)

    def set_data(self, df):
        self.df = df
        if df is not None and not df.empty:
            self.update_combos()

    def update_combos(self):
        cols = list(self.df.columns)

        self.region_combo.clear()
        self.region_combo.addItem("Всі регіони")
        if 'srcid_region' in cols:
            regions = sorted(self.df['srcid_region'].dropna().unique())
            self.region_combo.addItems(regions)

        self.date_combo.clear()
        self.target_combo.clear()
        self.date_combo.addItems(cols)
        self.target_combo.addItems(cols)

        for col in cols:
            lower = col.lower()
            if any(x in lower for x in ['doj', 'doi', 'date']):
                self.date_combo.setCurrentText(col)
            if any(x in lower for x in ['cumsum_seatcount', 'cumsum_searchcount',
'demand']):
                self.target_combo.setCurrentText(col)

    def run_all_models(self):
        if self.df is None or self.df.empty:
            QMessageBox.warning(self, "Помилка", "Завантажте дані!")
            return

        region = self.region_combo.currentText()
        date_col = self.date_combo.currentText()
        target_col = self.target_combo.currentText()
        periods = self.days_spin.value()

        df_f = self.df.copy()
        if region != "Всі регіони" and 'srcid_region' in df_f.columns:
            df_f = df_f[df_f['srcid_region'] == region]
```

```
        if df_f.empty:
            QMessageBox.warning(self, "Помилка", "Немає даних для вибраного
регіону!")
            return

        try:
            df_ml = df_f.copy()
            df_ml['ds'] = pd.to_datetime(df_ml[date_col])
            daily = df_ml.groupby('ds')[target_col].sum().reset_index()
            daily['day_num'] = (daily['ds'] - daily['ds'].min()).dt.days
            daily['month'] = daily['ds'].dt.month
            daily['weekday'] = daily['ds'].dt.weekday

            X = daily[['day_num', 'month', 'weekday']]
            y = daily[target_col]

            real_data, prophet_fc = make_forecast(df_f, date_col, target_col,
periods)
            hist = real_data.merge(prophet_fc[['ds', 'yhat']], on='ds', how='left')
            mae_p = mean_absolute_error(hist['y'], hist['yhat'])
            rmse_p = np.sqrt(mean_squared_error(hist['y'], hist['yhat']))
            mape_p = np.mean(np.abs((hist['y'] - hist['yhat']) / hist['y'].replace(0,
np.nan))) * 100

            from sklearn.model_selection import train_test_split
            X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)

            lr = LinearRegression()
            lr.fit(X_train, y_train)
            y_pred_lr = lr.predict(X_test)
            mae_lr = mean_absolute_error(y_test, y_pred_lr)
            rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred_lr))
            mape_lr = np.mean(np.abs((y_test - y_pred_lr) / y_test.replace(0,
np.nan))) * 100

            rf = RandomForestRegressor(n_estimators=150, random_state=42)
            rf.fit(X_train, y_train)
            y_pred_rf = rf.predict(X_test)
            mae_rf = mean_absolute_error(y_test, y_pred_rf)
            rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
            mape_rf = np.mean(np.abs((y_test - y_pred_rf) / y_test.replace(0,
np.nan))) * 100

            self.figure.clear()
            ax = self.figure.add_subplot(111)

            ax.plot(real_data['ds'], real_data['y'], label='Реальні дані',
color='blue', linewidth=2.5)
            ax.plot(prophet_fc['ds'], prophet_fc['yhat'], label='Prophet',
color='red', linestyle='--', linewidth=2)

            future_dates = pd.date_range(start=real_data['ds'].max() +
pd.Timedelta(days=1), periods=periods)
            future_df = pd.DataFrame({
```

```
                'day_num': range(daily['day_num'].max() + 1, daily['day_num'].max() +
periods + 1),
                'month': [daily['month'].iloc[-1]] * periods,
                'weekday': [daily['weekday'].iloc[-1]] * periods
            })

            ax.plot(future_dates, lr.predict(future_df), label='Linear Regression',
color='green', linestyle='--')
            ax.plot(future_dates, rf.predict(future_df), label='Random Forest',
color='orange', linestyle='--')

            title = f'Порівняння моделей — {target_col}'
            if region != "Всі регіони":
                title += f'\nРегіон: {region}'
            ax.set_title(title, fontsize=15)
            ax.set_xlabel('Дата')
            ax.set_ylabel('Значення')
            ax.legend()
            ax.grid(True, alpha=0.3)
            ax.tick_params(axis='x', rotation=45)

            self.canvas.draw()

            self.metrics_table.setRowCount(3)
            self.metrics_table.setItem(0, 0, QTableWidgetItem("Prophet"))
            self.metrics_table.setItem(0, 1, QTableWidgetItem(f"{mae_p:.2f}"))
            self.metrics_table.setItem(0, 2, QTableWidgetItem(f"{rmse_p:.2f}"))
            self.metrics_table.setItem(0, 3, QTableWidgetItem(f"{mape_p:.2f}"))

            self.metrics_table.setItem(1, 0, QTableWidgetItem("Linear Regression"))
            self.metrics_table.setItem(1, 1, QTableWidgetItem(f"{mae_lr:.2f}"))
            self.metrics_table.setItem(1, 2, QTableWidgetItem(f"{rmse_lr:.2f}"))
            self.metrics_table.setItem(1, 3, QTableWidgetItem(f"{mape_lr:.2f}"))

            self.metrics_table.setItem(2, 0, QTableWidgetItem("Random Forest"))
            self.metrics_table.setItem(2, 1, QTableWidgetItem(f"{mae_rf:.2f}"))
            self.metrics_table.setItem(2, 2, QTableWidgetItem(f"{rmse_rf:.2f}"))
            self.metrics_table.setItem(2, 3, QTableWidgetItem(f"{mape_rf:.2f}"))

            QMessageBox.information(self, "Успіх", f"Порівняння моделей
завершено!\nРегіон: {region}")

        except Exception as e:
            QMessageBox.critical(self, "Помилка", str(e))
```

ДОДАТОК В

Лістинг коду preprocessing_page.py

```
import pandas as pd
import numpy as np
from PySide6.QtWidgets import (
    QWidget, QVBoxLayout, QHBoxLayout, QLabel, QPushButton,
    QComboBox, QGroupBox, QMessageBox, QSpinBox,
    QListWidget, QTabWidget, QCheckBox, QFileDialog
)
from matplotlib.backends.backend_qtagg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure
from scipy import stats
import warnings

warnings.filterwarnings('ignore')


class PreprocessingPage(QWidget):
    def __init__(self):
        super().__init__()

        self.df = None
        self.processed_df = None

        self.init_ui()

    def init_ui(self):
        layout = QVBoxLayout()

        title = QLabel("📊 Передобробка даних")
        title.setStyleSheet("""
            font-size: 18px;
            font-weight: bold;
            padding: 10px;
        """)

        layout.addWidget(title)

        self.tabs = QTabWidget()

        self.tab_clean = QWidget()
        self.tab_transform = QWidget()
        self.tab_features = QWidget()

        self.setup_clean_tab()
        self.setup_transform_tab()
        self.setup_features_tab()

        self.tabs.addTab(self.tab_clean, " Очищення")
        self.tabs.addTab(self.tab_transform, " Трансформація")
        self.tabs.addTab(self.tab_features, " Feature Engineering")
```

```
        layout.addWidget(self.tabs)

        btn_layout = QHBoxLayout()

        self.btn_apply = QPushButton(" Застосувати")
        self.btn_apply.clicked.connect(self.apply_all)

        self.btn_reset = QPushButton(" Скинути")
        self.btn_reset.clicked.connect(self.reset_data)

        self.btn_save = QPushButton(" Зберегти")
        self.btn_save.clicked.connect(self.save_dataset)

        btn_layout.addWidget(self.btn_apply)
        btn_layout.addWidget(self.btn_reset)
        btn_layout.addWidget(self.btn_save)

        layout.addLayout(btn_layout)

        self.setLayout(layout)

    def setup_clean_tab(self):
        layout = QVBoxLayout()

        group = QGroupBox("Очищення даних")
        h = QHBoxLayout()

        self.chk_duplicates = QCheckBox("Видалити дублікати")
        self.chk_duplicates.setChecked(True)

        self.chk_missing = QCheckBox("Заповнити пропуски")
        self.chk_missing.setChecked(True)

        self.outlier_combo = QComboBox()
        self.outlier_combo.addItems([
            "Не обробляти",
            "Z-Score (3σ)",
            "IQR Method"
        ])

        h.addWidget(self.chk_duplicates)
        h.addWidget(self.chk_missing)
        h.addWidget(QLabel("Викиди:"))
        h.addWidget(self.outlier_combo)

        group.setLayout(h)

        layout.addWidget(group)
        layout.addStretch()

        self.tab_clean.setLayout(layout)

    def setup_transform_tab(self):
        layout = QVBoxLayout()

        left = QVBoxLayout()
```

```
        left.addWidget(QLabel("Колонки:"))

        self.col_list = QListWidget()
        self.col_list.setSelectionMode(
            QListWidget.SelectionMode.MultiSelection
        )

        self.col_list.itemSelectionChanged.connect(
            self.plot_distributions
        )

        left.addWidget(self.col_list)

        right = QVBoxLayout()

        right.addWidget(QLabel("Тип трансформації:"))

        self.transform_combo = QComboBox()
        self.transform_combo.addItems([
            "Log (log1p)",
            "Square Root",
            "Z-Score",
            "Min-Max Scaling",
            "Box-Cox",
            "Yeo-Johnson"
        ])

        right.addWidget(self.transform_combo)

        top = QHBoxLayout()
        top.addLayout(left, 1)
        top.addLayout(right, 1)

        layout.addLayout(top)

        graphs = QHBoxLayout()

        self.fig_before = Figure(figsize=(5, 4))
        self.canvas_before = FigureCanvas(self.fig_before)

        self.fig_after = Figure(figsize=(5, 4))
        self.canvas_after = FigureCanvas(self.fig_after)

        graphs.addWidget(self.canvas_before)
        graphs.addWidget(self.canvas_after)

        layout.addWidget(QLabel("Розподіл даних"))
        layout.addLayout(graphs)

        self.tab_transform.setLayout(layout)

    def setup_features_tab(self):
        layout = QVBoxLayout()

        group = QGroupBox("Feature Engineering")
```

```
        fl = QVBoxLayout()

        self.spin_lag = QSpinBox()
        self.spin_lag.setRange(1, 30)
        self.spin_lag.setValue(7)
        self.spin_lag.setSuffix(" днів")

        self.spin_rolling = QSpinBox()
        self.spin_rolling.setRange(3, 30)
        self.spin_rolling.setValue(7)
        self.spin_rolling.setSuffix(" днів")

        fl.addWidget(QLabel("Lag Features"))
        fl.addWidget(self.spin_lag)

        fl.addWidget(QLabel("Rolling Mean"))
        fl.addWidget(self.spin_rolling)

        fl.addWidget(QLabel(
            "Автоматично: day_of_week, is_weekend, month"
        ))

        group.setLayout(fl)

        layout.addWidget(group)
        layout.addStretch()

        self.tab_features.setLayout(layout)


    def set_data(self, df: pd.DataFrame):
        self.df = df.copy()
        self.processed_df = df.copy()

        self.update_column_list()
        self.plot_distributions()


    def update_column_list(self):
        self.col_list.clear()

        if self.df is None:
            return

        numeric_cols = self.df.select_dtypes(
            include=[np.number]
        ).columns

        for col in numeric_cols:
            self.col_list.addItem(col)


    def plot_distributions(self):
        if self.df is None or self.df.empty:
            return
```

```
        selected_items = self.col_list.selectedItems()

        if selected_items:
            col = selected_items[0].text()
        else:
            numeric_cols = self.df.select_dtypes(
                include=[np.number]
            ).columns

            if len(numeric_cols) == 0:
                return

            col = numeric_cols[0]

        self.fig_before.clear()

        ax1 = self.fig_before.add_subplot(111)

        ax1.hist(
            self.df[col].dropna(),
            bins=40,
            color='blue',
            alpha=0.7
        )

        ax1.set_title(f"До: {col}")
        ax1.grid(True, alpha=0.3)

        self.canvas_before.draw()

        self.fig_after.clear()

        ax2 = self.fig_after.add_subplot(111)

        ax2.hist(
            self.processed_df[col].dropna(),
            bins=40,
            color='green',
            alpha=0.7
        )

        ax2.set_title(f"Після: {col}")
        ax2.grid(True, alpha=0.3)

        self.canvas_after.draw()

    def apply_all(self):
        if self.df is None or self.df.empty:
            QMessageBox.warning(
                self,
                "Помилка",
                "Спочатку завантажте CSV!"
            )
            return
```

```
try:
    self.processed_df = self.df.copy()

    if self.chk_duplicates.isChecked():
        self.processed_df = (
            self.processed_df.drop_duplicates()
        )

    if self.chk_missing.isChecked():
        numeric_cols = self.processed_df.select_dtypes(
            include=[np.number]
        ).columns

        self.processed_df[numeric_cols] = (
            self.processed_df[numeric_cols].fillna(
                self.processed_df[numeric_cols].mean()
            )
        )

    outlier_method = self.outlier_combo.currentText()

    numeric_cols = self.processed_df.select_dtypes(
        include=[np.number]
    ).columns

    if outlier_method == "Z-Score (3σ)":
        z_scores = np.abs(
            stats.zscore(
                self.processed_df[numeric_cols],
                nan_policy='omit'
            )
        )

        self.processed_df = self.processed_df[
            (z_scores < 3).all(axis=1)
        ]

    elif outlier_method == "IQR Method":
        for col in numeric_cols:
            Q1 = self.processed_df[col].quantile(0.25)
            Q3 = self.processed_df[col].quantile(0.75)

            IQR = Q3 - Q1

            lower = Q1 - 1.5 * IQR
            upper = Q3 + 1.5 * IQR

            self.processed_df = self.processed_df[
                (self.processed_df[col] >= lower) &
                (self.processed_df[col] <= upper)
            ]

    selected_cols = [
        item.text()
        for item in self.col_list.selectedItems()
    ]
```

```
transform_type = self.transform_combo.currentText()

for col in selected_cols:

    if transform_type == "Log (log1p)":
        self.processed_df[col] = np.log1p(
            self.processed_df[col].clip(lower=0)
        )

    elif transform_type == "Square Root":
        self.processed_df[col] = np.sqrt(
            self.processed_df[col].clip(lower=0)
        )

    elif transform_type == "Z-Score":
        mean = self.processed_df[col].mean()
        std = self.processed_df[col].std()

        if std > 0:
            self.processed_df[col] = (
                (self.processed_df[col] - mean) / std
            )

    elif transform_type == "Min-Max Scaling":
        min_val = self.processed_df[col].min()
        max_val = self.processed_df[col].max()

        if max_val - min_val > 0:
            self.processed_df[col] = (
                (self.processed_df[col] - min_val) /
                (max_val - min_val)
            )

if 'doj' in self.processed_df.columns:

    dt = pd.to_datetime(
        self.processed_df['doj'],
        errors='coerce'
    )

    self.processed_df['day_of_week'] = (
        dt.dt.dayofweek
    )

    self.processed_df['is_weekend'] = (
        dt.dt.dayofweek.isin([5, 6]).astype(int)
    )

    self.processed_df['month'] = dt.dt.month

    lag = self.spin_lag.value()
    rolling = self.spin_rolling.value()

    for col in selected_cols:
```

```
                self.processed_df[
                    f'{col}_lag_{lag}'
                ] = self.processed_df[col].shift(lag)

                self.processed_df[
                    f'{col}_rolling_{rolling}'
                ] = (
                    self.processed_df[col]
                    .rolling(rolling)
                    .mean()
                )

            self.plot_distributions()

            QMessageBox.information(
                self,
                "Успіх",
                "Дані успішно оброблено!"
            )

        except Exception as e:
            QMessageBox.critical(
                self,
                "Помилка",
                str(e)
            )

    def reset_data(self):
        if self.df is not None:
            self.processed_df = self.df.copy()

            self.plot_distributions()

            QMessageBox.information(
                self,
                "Скинуто",
                "Дані повернуто до початкового стану."
            )


    def save_dataset(self):
        if self.processed_df is None:
            return

        path, _ = QFileDialog.getSaveFileName(
            self,
            "Зберегти CSV",
            "processed_data.csv",
            "CSV Files (*.csv)"
        )

        if path:
            self.processed_df.to_csv(
                path,
                index=False
            )
```

```
QMessageBox.information(
    self,
    "Успіх",
    "Файл збережено!"
)
```