МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**ТРОНЬ ДЕНИС ВІТАЛІЙОВИЧ**

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
д-р. техн. наук, професор
_____Наталія ВЕСЕЛОВСЬКА
«____»_____ 2026 р.

# РЕАЛІЗАЦІЯ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ АНАЛІЗУ МАКРОЕКОНОМІЧНИХ ІНДИКАТОРІВ ДЕРЖАВИ

Спеціальність 122 Комп’ютерні науки

**Кваліфікаційна (бакалаврська) робота**

Керівник:
Надія ПОТАПОВА, доцент кафедри
інформаційних технологій,
к. е. н., доцент
_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____

Вінниця – 2026

# АНОТАЦІЯ

**Тронь Д. В. Реалізація алгоритмів машинного навчання для аналізу макроекономічних індикаторів держави.** Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця 2026.

У кваліфікаційній (бакалаврській) роботі досліджено основні макроекономічні індикатори держави (ВВП, інфляція, безробіття, державний борг, показники експорту та імпорту) та проаналізовано сучасні методи машинного навчання для їх прогнозування. Розроблено Програмний застосунок для аналізу та прогнозування макроекономічних показників із використанням алгоритмів лінійної регресії та Random Forest. Засіб реалізовано мовою Python із застосуванням бібліотек Pandas, NumPy, Scikit-learn, Matplotlib та Streamlit.

**Ключові слова:** машинне навчання, прогнозування, макроекономічні індикатори, лінійна регресія, Random Forest, Python, Streamlit, ВВП, інфляція, World Bank.

59 ст., 28 рис., 5 табл., 1 дод., 43 джерел.

# ABSTRACT

**Tron D.V. Implementation of Machine Learning Algorithms for the Analysis of State Macroeconomic Indicators.** Specialty 122 «Computer Science», educational program «Computer Science». Vasyl Stus Donetsk National University, Vinnytsia 2026.

In the qualification (bachelor's) thesis, the main macroeconomic indicators of the state (GDP, inflation, unemployment, public debt, exports and imports) are studied, and modern machine learning methods for their forecasting are analyzed. A software tool for the analysis and forecasting of macroeconomic indicators using linear regression and Random Forest algorithms is developed. The tool is implemented in Python using the Pandas, NumPy, Scikit-learn, Matplotlib and Streamlit libraries.

**Keywords:** machine learning, forecasting, macroeconomic indicators, linear regression, Random Forest, Python, Streamlit, GDP, inflation, World Bank.

# ЗМІСТ

# ВСТУП

*Актуальність дослідження.* У сучасних умовах розвитку світової економіки, аналіз макроекономічних показників є важливим інструментом для оцінки економічного стану країни та прогнозування її подальшого розвитку. До таких показників входять: валовий внутрішній продукт (ВВП), рівень інфляції, безробіття, державний борг та зовнішньоторгівельний баланс. Знання цих показників, дають можливість оцінювати ефективність економічної політики та виявляти можливі майбутні ризики для економіки.

Зі збільшенням обсягів економічних даних, традиційні статистичні методи аналізу не завжди дають змогу забезпечити достатню точність та швидкість обробки інформації. У зв’язку з цим, останніми роками, набувають поширення методи машинного навчання, які дозволяють автоматизувати процес аналізу даних, виявляти приховані закономірності та будувати прогнози на основі уже відомих, історичних даних.

Використання алгоритмів машинного навчання у сфері аналізу макроекономічних показників, відкриває можливості для більш точного та швидшого прогнозування та прийняття аналітичних рішень. Це є особливо актуальним в умовах економічної нестабільності, кризових явищ та швидких змін економічного середовища.

*Метою роботи* є розробка програмного застосунку для аналізу та прогнозування макроекономічних індикаторів держави із використанням алгоритмів машинного навчання.

Для досягнення поставленої мети необхідно виконати наступні *завдання:*

1. Проаналізувати сучасні методи машинного навчання для задач прогнозування макроекономічних індикаторів.
2. Обрати технології та інструменти для реалізації програмного засобу.
3. Реалізувати алгоритми аналізу та прогнозування макроекономічних показників.

4. Провести аналіз та попередню обробку даних для проведення оцінки розробленого програмного застосунку.

5. Реалізувати користувацький інтерфейс програмного засобу.

6. Виконати оцінку якості побудованих моделей та порівняти результати роботи.

*Об'єктом дослідження* є процес реалізації алгоритмів машинного навчання для аналізу та прогнозування макроекономічних показників держави.

*Предметом дослідження* є методи та алгоритми машинного навчання для аналізу та прогнозування макроекономічних індикаторів на основі статистичних даних.

У роботі використано методи аналізу даних, машинного навчання, статистичної обробки інформації, регресійного аналізу та візуалізації результатів.

*Практичне значення роботи* полягає у створенні програмного продукту, який наочно продемонструє автоматизацію процесів аналізу макроекономічних даних, призначений для прогнозування економічних показників та візуалізації результатів для подальшого аналізу.

*Апробація результатів* даного дослідження була здійснена на VII Всеукраїнській науково-практичній конференції здобувачів, аспірантів та молодих вчених «Прикладні інформаційні технології» (м. Вінниця, 15 травня 2026 року).

# РОЗДІЛ 1
# ТЕОРЕТИЧНІ ОСНОВИ МАШИННОГО НАВЧАННЯ ПРИ ПРОГНОЗУВАННІ МАКРОЕКОНОМІЧНИХ ІНДИКАТОРІВ ДЕРЖАВИ

## 1.1 Сутність макроекономічних індикаторів держави

Макроекономічні індикатори є важливими показниками, що характеризують стан економіки держави, її рівень розвитку та ефективність економічної політики. Ці показники використовуються для аналізу економічної ситуації, оцінки фінансової стабільності та прогнозування подальшого розвитку економіки та її тенденції. До макроекономічних показників належать [1]:

- Валовий внутрішній продукт.
- Рівень інфляції.
- Рівень безробіття.
- Державний борг.
- Показники експорту та імпорту.
- Валютний курс.
- Індекс споживчих цін.

Одним із найважливіших показників є валовий внутрішній продукт (ВВП) – який відображає загальну вартість товарів та послуг, вироблених у межах держави за певний період часу. Зростання ВВП зазвичай свідчить про розвиток економіки та збільшення економічної активності. На обсяг ВВП впливають: рівень виробництва, інвестиційна активність, споживчий попит, розвиток промисловості, торгівля та економічна стабільність.

Окремо потрібно згадати про показник ВВП на душу населення, який визначається як співвідношення валового внутрішнього продукту до кількості населення країни. Даний індикатор дозволяє оцінити середній рівень економічного розвитку та добробуту населення. У подальшому, в цій роботі буде використовуватись лише показник ВВП.

Рівень інфляції характеризує тривале зростання загального рівня цін у

державі, що відображає зниження купівельної спроможності грошової одиниці. За темпами зростання цін, виділяють такі види інфляції [3]:

1. Помірна – коли ціни зростають поступово й не більше ніж на 10% протягом року. Є нормальним для ринкової економіки.

2. Галопуюча – темпи росту цін становлять від 10 до 50% на рік, що вказує на серйозні проблеми в економіці.

3. Гіперінфляція – зростання цін на понад 50% на місяць, за якого гроші повністю втрачають свою цінність, що призводить до руйнування економіки.

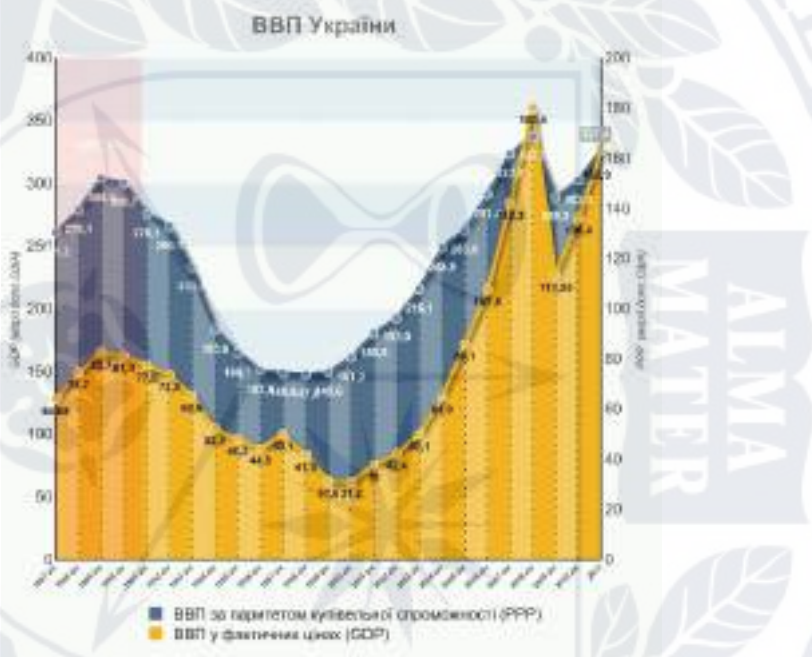


Рисунок 1.1 – ВВП України за 1987-2011 роки

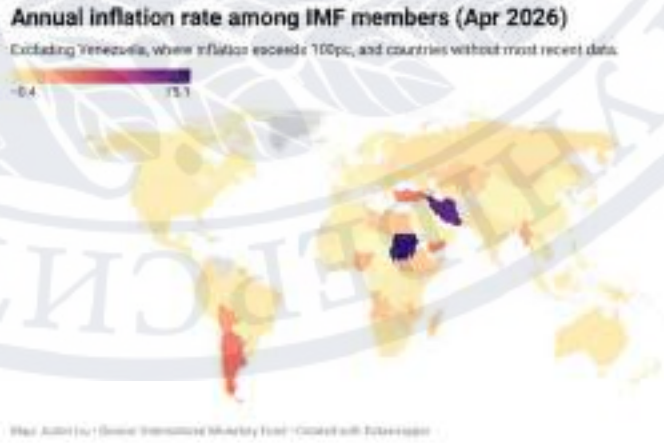


Рисунок 1.2 – Теплова карта рівня інфляції у світі станом на квітень 2026 року

Рівень інфляції залежить від грошової політики держави, співвідношення попиту та пропозиції, змін валютного курсу, вартості енергоресурсів та економічної ситуації на внутрішньому та зовнішньому ринках.

Важливим показником є рівень безробіття, який демонструє частку працездатного населення, що не має роботи. Високий рівень безробіття може свідчити про економічні труднощі.

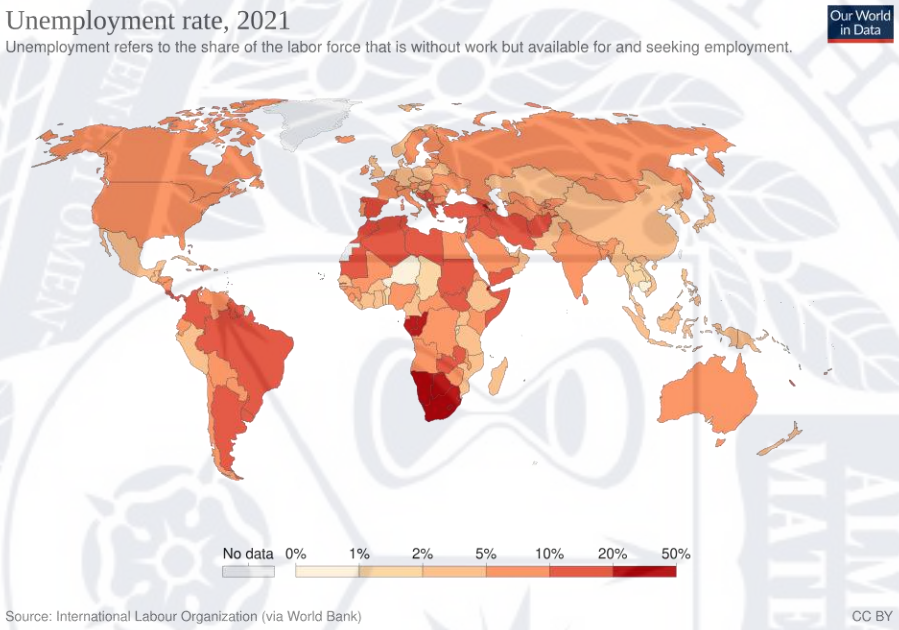


Рисунок 1.3 – Рівень безробіття у світі станом на 2021 рік

На рівень безробіття впливають темпи економічного розвитку, стан ринку праці, рівень інвестицій, автоматизація виробництва та загальна економічна ситуація в країні.

Державний борг є показником що характеризує обсяг фінансових зобов'язань держави перед внутрішніми та зовнішніми кредиторами. Надмірне зростання державного боргу може створювати ризики для економічної стабільності та фінансової безпеки країни. Обсяг державного боргу формується під впливом дефіциту бюджету, економічних криз, державних витрат, рівня податкових надходжень та необхідності зовнішнього фінансування [5].

Показники імпорту та експорту дозволяють оцінити зовнішню економічну діяльність держави та її участь у міжнародній торгівлі. Аналіз

співвідношення між імпортом та експортом дає змогу оцінити торгівельний баланс країни.

Макроекономічні показники тісно пов’язані між собою, тому для якісного аналізу економічної ситуації необхідно враховувати їх комплексний вплив. У сучасних умовах, через великі обсяги економічних даних, потребують застосування автоматизованих методів аналізу, в тому числі й машинного навчання.

### 1.2 Аналіз методів прогнозування економічних показників

Прогнозування макроекономічних показників є важливим елементом аналізу та планування економіки. Це дозволяє оцінювати можливі зміни економічного стану держави, визначати тенденції розвитку та приймати обгрунтовані управлінські рішення.

Зазвичай, для прогнозування економічних показників, використовуються статистичні та економетричні методи. Серед них – методи часових рядів, кореляційний аналіз, регресійні моделі та інші математичні підходи. Такі методи дають змогу аналізувати історичні дані та визначати залежності між економічними показниками.

Одним з найбільш поширених підходів є регресійний аналіз, який використовується для встановлення залежності між незалежними змінними та прогнозованим показником. Лінійна регресія є одним із найпростіших та найпоширеніших методів прогнозування.

Проте традиційні методи мають певні обмеження, які полягають у тому, що вони часто погано працюють із великими обсягами даних, складними нелінійними залежностями та нестабільними економічними умовами. Крім того, сучасні економічні процеси характеризуються значною кількістю факторів, які не просто врахувати за допомогою класичних статистичних моделей.

У зв’язку з цим, дедалі ширшого поширення набувають методи машинного навчання. На відміну від звичних підходів, алгоритми машинного

навчання здатні автоматично виявляти закономірності у даних, адаптуватися до складних залежностей та працювати з великими наборами інформації.


Рисунок 1.4 – Процес машинного навчання з учителем

Найбільш популярними алгоритмами машинного навчання для задач прогнозування є лінійна регресія, дерева рішень, Random Forest, градієнтний бустинг та нейронні мережі. Вибір певного алгоритму залежить від структури даних, складності завдань та необхідної точності прогнозування.

Застосування методів машинного навчання у сфері макроекономічного аналізу дозволяє покращити ефективність прогнозування та забезпечити більш глибокий аналіз.

### 1.3 Особливості використання машинного навчання в прогнозуванні

Машинне навчання є одним із напрямів штучного інтелекту, який являє собою створення алгоритмів, що здатні навчатися на основі даних та виконувати прогнозування без явного програмування кожного правила. Основною особливістю машинного навчання є можливість автоматичного виявлення закономірностей у даних та використання цих закономірностей для прогнозування нових значень.

У сучасних умовах машинне навчання активно застосовується у фінансовій сфері, медицині, транспорті, маркетингу та економіці. Особливо

важливим є використання алгоритмів машинного навчання для аналізу макроекономічних показників, оскільки економічні процеси характеризуються великою кількістю взаємопов’язаних факторів та значними обсягами статистичних даних.

Для задач прогнозування макроекономічних показників найбільш поширеним є навчання з учителем. У цьому підході, модель навчається на наборі історичних даних, де для кожного прикладу відомі вхідні параметри та правильний результат. Під час навчання, алгоритм аналізує залежності між параметрами та формує математичну модель, яка надалі використовується для прогнозування нових значень.

Під час навчання використовуються [10]:

- Вхідні ознаки – параметри, на основі яких здійснюється прогнозування.
- Цільова змінна – показник, який необхідно передбачити.
- Модель машинного навчання.
- Навчальні та тестові вибірки.

До прикладу, для прогнозування ВВП, вхідними ознаками можуть бути: рівень інфляції, безробіття, державний борг та обсяг експорту.

Процес побудови моделі машинного навчання зазвичай виглядає так:

1. Збір та аналіз даних
2. Попередня обробка інформації
3. Розділення даних на тренувальну та тестову вибірки
4. Навчання моделі
5. Оцінка точності прогнозування
6. Аналіз результатів

Для перевірки ефективності моделі дані поділяються на дві частини:

- Тренувальні дані, які використовуються для тренування моделі
- Тестові дані, які використовуються для перевірки якості прогнозування

Для оцінки якості моделей, використовуються спеціальні метрики. Однією з найпоширеніших є середня абсолютна помилка (MAE), яка показує середнє відхилення прогнозованих значень від реальних. Також є такі

параметри як середньоквадратична помилка (MSE), корінь із середньоквадратичної помилки (RMSE) та коефіцієнт детермінації $R^2$.

Важливим етапом аналізу є візуалізація результатів роботи. Побудова графіків дозволяє порівнювати реальні та прогнозовані значення, аналізувати тенденції та оцінювати якість прогнозування. Візуальний аналіз значно спрощує інтерпретацію результатів та дозволяє виявляти можливі помилки моделі.

### 1.4 Аналіз алгоритмів машинного навчання

Лінійна регресія є одним із найпростіших та найбільш поширених алгоритмів машинного навчання для задач прогнозування числових значень. Основною метою даного алгоритму є встановлення математичної залежності між вхідними параметрами та цільовою змінною. У випадку одного параметра, модель описується рівнянням [12]:

$$y = b_0 + b_1 x \quad (1.1)$$

де $y$ – прогнозоване значень;

$x$ – вхідний параметр;

$b_0$ – вільний коефіцієнт;

$b_1$ – коефіцієнт впливу параметра.

У більшості задач зазвичай використовується множинна лінійна регресія, яка враховує декілька вхідних параметрів одночасно:

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n \quad (1.2)$$

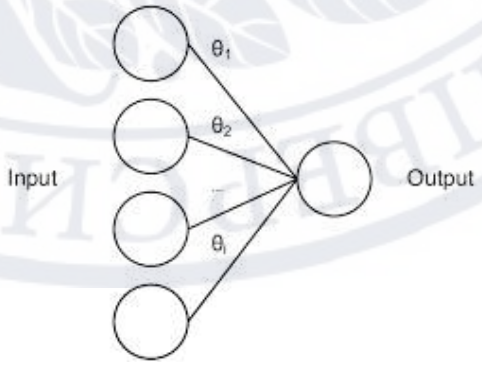


Рисунок 1.5 – Схема побудови лінійної регресії

Проте лінійна регресія має певні недоліки. Основними проблемами є припущення про лінійний характер залежності між параметрами. У реальних економічних процесах залежності часто є складними та нелінійними, тому точність лінійної регресії може бути недостатньою.

Також алгоритм чутливий до аномальних значень, шуму в даних та сильної кореляції між параметрами.

Незважаючи на це, лінійна регресія залишається важливим інструментом для аналізу економічних даних та використовується для побудови базових прогнозів і порівняння ефективності інших моделей машинного навчання.

Random Forest є одним із найбільш популярних алгоритмів машинного навчання, який використовується для задач класифікації та регресії. Даний алгоритм базується на використанні великої кількості дерев рішень, результати яких об’єднуються для отримання більш точного прогнозу.

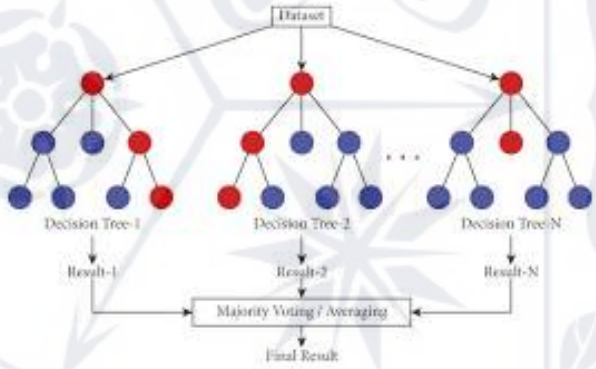


Рисунок 1.6 – Схема Random Forest

Кожне дерево навчається на випадковій частині даних та використовує випадковий набір параметрів. Завдяки цьому, окремі дерева формують різні прогнози а кінцевий результат визначається шляхом усереднення отриманих значень. Принцип роботи алгоритму можна представити таким чином:

- формування декількох підмножин даних;
- побудова окремих дерев рішень;
- отримання прогнозів від кожного дерева;
- об’єднання результатів у фінальний прогноз.

Кожне дерево рішень являє собою структуру з умовами та розгалуженнями, яка послідовно аналізує параметри та приймає рішення на основі певних правил. Наприклад, дерево може враховувати такі параметри як рівень інфляції, безробіття та державного боргу для прогнозування економічного зростання.

Однією з головних переваг Random Forest є здатність працювати зі складними нелінійними залежностями між парметрами. На відміну від лінійної регресії, алгоритм не обмежується лише лінійними зв'язками та може враховувати складні взаємодії між економічними показниками. Основними перевагами Random Forest є:

- висока точність прогнозування;
- стійкість до шуму та аномальних значень;
- здатність працювати з великими обсягами даних;
- ефективна робота із нелінійними залежностями;
- стійкість до перенавчань.

Завдяки цим властивостям, Random Forest широко використовується для аналізу фінансових та економічних даних.

Проте цей алгоритм також має і недоліки. Порівняно з лінійною регресією, Random Forest потребує більше обчислювальних ресурсів та часу для навчання. А також, результати роботи моделі складніше інтерпретувати, оскільки фінальний прогноз формується на основі великої кількості дерев рішень.

У задачах прогнозування макроекономічних показників, Random Forest дозволяє враховувати складні взаємозв'язки між економічними індикаторами та забезпечувати більш точніші результати прогнозування в умовах нестабільних або нелінійних даних.

Використання одночасно лінійної регресії та Random Forest, дозволяє провести порівняння між простішою лінійною моделлю та більш складнішим алгоритмом, щоб оцінити їх ефективність та визначити найбільш підходящий підхід для задач аналізу макроекономічних даних.

Під час побудови моделей машинного навчання, важливим етапом є оцінка якості прогнозування. Для оцінки якостей моделей використовують відповідні метрики. Навіть якщо модель успішно навчається на історичних даних, це не гарантує точності прогнозів для нових значень. Саме тому, для аналізу ефективності моделей використовуються спеціальні метрики оцінки якості.

Метрики дозволяють визначити рівень помилки між реальними та прогнозованими значеннями, а також оцінити здатність моделі описувати закономірності між даними. У задачах на прогнозування макроекономічних показників, найчастіше використовуються метрики регресійного аналізу, оскільки більшість економічних параметрів представлені у числових значеннях.

У даній роботі для оцінки ефективності моделей машинного навчання будуть використані наступні метрики [15]:

- Mean Absolute Error (MAE);
- Mean Squared Error (MSE);
- Root Mean Squared Error (RMSE);
- Коефіцієнт детермінації $R^2$.

Mean Absolute Error (MAE) є однією з найпростіших та найпоширеніших параметрів оцінки якості регресійних моделей. Вона показує середнє абсолютне відхилення між реальними та прогнозованими значеннями.

Її формула має наступний вигляд:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad (1.3)$$

де $y_i$ – реальне значення;

$\hat{y}_i$ – прогнозоване значення;

$n$ – кількість спостережень.

Чим менше значення MAE, тим точнішою є модель прогнозування. Перевагою цього параметра є простота інтерпретації, оскільки вона показує

середню помилку прогнозу у тих самих одиницях вимірювання що і показник.

Недоліком цього параметра є те, що він однаково враховує як малі, так і великі помилки, не надаючи більшої ваги значним відхиленням.

Mean Squared Error (MSE) використовується для оцінки середнього квадрата помилок прогнозування. На відміну від MAE, цей параметр підносить помилки до квадрату, що дозволяє сильніше враховувати великі відхилення.

Формула MSE виглядає наступним чином:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (1.4)$$

Використання квадратів помилок робить даний параметр чутливим до аномальних значень та великих похибок прогнозування. Це дає змогу ефективніше оцінювати стабільність роботи моделі.

Основною перевагою MSE є можливість виявлення значних помилок у прогнозуванні. Проте інтерпретація даного параметра є менш зручною, тому що результат виражається у квадраті одиниць вимірювання прогнозованого показника.

Root Mean Squared Error (RMSE) є модифікацією MSE та визначається як квадратний корінь із середньоквадратичної помилки.

Її формула виглядає наступним чином:

$$R\text{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \qquad (1.5)$$

Дана формула сильніше враховує великі помилки прогнозування, однак її результат повертається до початкових одиниць вимірювання. Завдяки цьому, RMSE є більш зручною для практичної інтерпретації, порівняно до MSE.

Чим менше значення RMSE, тим вищою є точність прогнозування моделі. Цей параметр широко використовується для оцінки моделей машинного навчання, включно із задачами прогнозування та аналізу економічних

показників.

Коефіцієнт детермінації $R^2$ використовується для оцінки того, наскільки добре модель пояснює залежність між параметрами та цільовою змінною.

ЇЇ формула виглядає наступним чином:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2} \quad (1.6)$$

де $y_i$ – реальне значення;

$\hat{y}_i$ – прогнозоване значення;

$\bar{y}_i$ – середнє значення цільової змінної.

Значення коефіцієнта $R^2$ знаходиться в межах від 0 до 1. Чим ближче значення до 1, тим краще модель пояснює закономірності у даних та наскільки точнішим є прогнозування. Наприклад:

1. $R^2 = 0.9$, що свідчить про дуже хорошу якість моделі

2. $R^2 = 0.5$, означає середню точність

3. Значення менше, свідчать про слабку ефективність.

Перевагою коефіцієнта детермінації є можливість оцінити загальну якість моделі та порівнювати ефективність різних алгоритмів для машинного навчання.

Для отримання об’єктивної оцінки якості моделей, доцільно використовувати декілька метрик одночасно. Це дозволяє більш ефективно та комплексно аналізувати результати прогнозування та враховувати різні аспекти роботи алгоритмів.

# РОЗДІЛ 2
# РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ МАКРОЕКОНОМІЧНИХ ПОКАЗНИКІВ

## 2.1 Технології та інструменти програмної реалізації алгоритмів машинного навчання

У сучасних умовах значного зростання обсягів економічної інформації, виникає потреба в автоматизованих засобах аналізу та прогнозування макроекономічних показників. Традиційні методи аналізу часто потребують значних витрат у часі та не завжди мають змогу забезпечити достатню точність прогнозування в умовах складних економічних процесів. У зв'язку з цим, актуальним стає використання методів машинного навчання для автоматизації аналізу економічних даних та побудови прогнозів на майбутнє.

Метою розробки програмного застосунку є створення демонстраційної системи для аналізу та прогнозування макроекономічних показників держави на основі алгоритмів машинного навчання. Програмний продукт повинен забезпечувати можливість роботи з економічними даними, навчання моделей машинного навчання, побудову прогнозів та візуалізацію результатів.

Основним завданням системи є прогнозування одного з макроекономічних показників на основі інших економічних індикаторів. Для реалізації даного завдання планується використання алгоритмів лінійної регресії та Random Forest з подальшим порівнянням ефективності їх роботи [20].

Програмний застосунок повинен забезпечувати виконання таких функцій:

- завантаження макроекономічних даних із файлів формату csv;
- аналіз структури датасету;
- попередня обробка даних;
- вибір цільового показника для прогнозування;
- навчання моделей машинного навчання;
- побудова прогнозів;

- оцінка точності моделей;
- візуалізація результатів прогнозування;
- порівняння результатів роботи різних алгоритмів.

Вхідними даними для системи є датасет із макроекономічними показниками держави за певний період часу. Набір даних може містити наступні параметри:

- валовий внутрішній продукт;
- рівень інфляції;
- рівень безробіття;
- обсяг імпорту та експорту;
- та інші економічні індикатори.

Результатом роботи системи є:

- прогнозовані значення економічного показника;
- графіки порівняння реальних та прогнозованих значень;
- метрики оцінки точності моделей;
- порівняльний аналіз алгоритмів машинного навчання.

Для реалізації програмного застосункупланується використання мови програмування Python та бібліотек для аналізу даних і машинного навчання. Вибір мови Python обумовлений широкими можливостями для роботи з даними, наявністю готових бібліотек для машинного навчання та зручністю реалізації аналітичних систем.

Однією з важливих вимог до системи є забезпечення простоти використання та наочності результатів. З цією метою передбачається реалізація базового користувацького інтерфейсу, який дозволить завантажувати датасети, запускати процес прогнозування та переглядати результати аналізу у графічному вигляді.

Під час розробки системи, особлива увага приділятиметься коректній підготовці даних, оскільки якість вхідної інформації безпосередньо впливає на точність прогнозування. Для оцінки ефективності моделей,

використовуватимуться такі параметри як MAE, MSE, RMSE та коефіцієнт детермінації $R^2$, згадані раніше.

Таким чином, програмний продукт повинен забезпечити автоматизацію аналізу макроекономічних даних, побудову прогнозів на основі алгоритмів машинного навчання та надання результатів у зручному для аналізу вигляді.

Для розробки програмного застосункуаналізу та прогнозування макроекономічних показників було обрано сучасні інструменти та бібліотеки, які забезпечують ефективну роботу з даними, реалізацію алгоритмів машинного навчання та побудову користувацького інтерфейсі.

Основною мовою програмування для реалізації системи, як було зазначено раніше, обрано Python. Це обумовлено тим, що дана мова широко використовується у сфері аналізу даних, машинного навчання та розробки аналітичних систем. Популярність Python обумовлена простотою синтаксису, великою кількістю спеціалізованих бібліотек та підтримкою сучасних методів машинного навчання [25].

Рисунок 2.1 – Логотип Python

Однією з головних переваг Python є наявність великої екосистеми бібліотек для роботи з даними та реалізації алгоритмів машинного навчання. Це дозволяє значно спростити процес розробки та забезпечити високу ефективність програмного засобу.

Для обробки та аналізу даних, використовуватиметься бібліотека Pandas. Вона забезпечує роботу з табличними даними, дозволяє завантажувати набори інформації у форматах CSV та Excel, виконувати фільтрацію, сортування, очищення та попередню обробку інформації. Використання Pandas, у свою чергу, є особливо важливим для аналізу макроекономічних показників, оскільки

економічні дані зазвичай представлені у вигляді таблиць.


Рисунок 2.2 – Бібліотека Pandas

Для виконання математичних операцій та роботи з багатовимірними масивами, використовується бібліотека NumPy. Дана бібліотека забезпечує швидкі математичні обчислення та є основною для багатьох інструментів аналізу даних у мові програмування Python.


Рисунок 2.3 – Основні функції бібліотеки NumPy

Реалізація алгоритмів машинного навчання, здійснюється за допомогою бібліотеки Scikit-learn. Вона містить велику кількість готових алгоритмів для задач класифікації, регресії та аналізу даних. У даній роботі, ця бібліотека використовуватиметься для реалізації алгоритмів лінійної регресії та Random Forest, а також для оцінки точності моделей, за допомогою вище перечислених метрик MAE, MSE, RMSE та $R^2$.

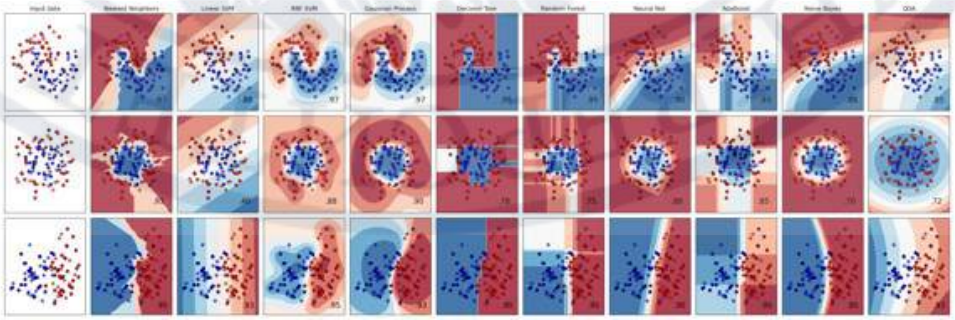

Рисунок 2.4 – Приклад класифікації категорії за допомогою Scikit-learn

Однією з переваг Scikit-learn є простота використання та зручний інтерфейс для навчання моделей машинного навчання. Бібліотека також містить інструменти для розділення даних на тренувальні та тестові вибірки, попередньої обробки інформації та автоматизації процесу аналізу даних.

Для побудови графіків та візуалізації результатів використовуватиметься бібліотека Matplotlib. Візуалізаця даних є важливим етапом аналізу, оскільки дозволяє наочно порівняти реальні та прогнозовані значення, оцінювати точність моделей та аналізувати тенденції зміни економічних показників.

Рисунок 2.5 – Приклад можливостей бібліотеки Matplotlib

Для створення базового користувацького інтерфейсу програмного засобу, використовуватиметься середовище Streamlit. Дани інструмент дозволяє швидко створювати інтерактивні вебзастосунки для аналізу даних та машинного навчання без необхідності використання складних вебтехнологій. Використання Streamlit дозволяє реалізувати [30]:

- завантаження датасетів користувачем;
- вибір параметрів прогнозування;
- запуск навчання моделей;
- відображення результатів аналізу;
- побудова графіків та таблиць.


Рисунок 2.6 – Приклад можливостей бібліотеки Streamlit

Для збереження навчених моделей машинного навчання може використовуватись бібліотека Joblib, яка дозволяє серіалізувати моделі та повторно використовувати їх без необхідності повторного навчання. У таблиці 2.1 нав едено основні технології та їх призначення у програмному застосунку.

Таблиця 2.1 Технології та призначення

| Технологія | Призначення |
|---|---|
| Python | Основна мова програмування |
| Pandas | Обробка та аналіз даних |
| NumPy | Математичні обчислення |
| Scikit-learn | Реалізація алгоритмів машинного навчання |
| Matplotlib | Візуалізація результатів |
| Streamlit | Створення користувацького інтерфейсу |
| Joblib | Збереження моделей |

Під час вибору технологій також враховувалась складність задачі та тип даних. Для даної роботи не планується використовувати складні середовища глибокого навчання, такі як TensorFlow або PyTorch, оскільки прогнозування макроекономічних показників виконується на основі табличних даних та не потребує використання нейронних мереж. У даному випадку, бібліотека Scikit-

learn забезпечує достатній функціонал та ефективність для реалізації поставлених задач.

Таким чином, обраний набір бібліотек та технологій, дозволяє реалізувати програмний застосунок для аналізу та прогнозування макроекономічних показників, забезпечити ефективну роботу з даними, реалізувати алгоритми машинного навчання та представити результати у зручному для користувача вигляді.

Як джерело макроекономічних даних обрано базу World Development Indicators Світового банку. Порівняно з альтернативними джерелами вона має суттєві переваги для цілей даного дослідження. База даних МВФ містить переважно фінансові показники, до яких входять: платіжний баланс, валютні резерви, зовнішній борг, що менш повно охоплює реальний сектор економіки. Євростат надає детальніші квартальні та місячні дані але лише для країн Європейського Союзу що обмежує можливості глобального порівняльного аналізу. Національні статистичні служби забезпечують найточніші дані для конкретної країни але кожна у власному форматі що унеможливлює автоматизоване завантаження для різних країн в єдиній системі. World Development Indicators натомість забезпечує уніфіковану методологію підрахунку для понад 200 країн світу в єдиному форматі CSV що є принциповою перевагою для порівняльного аналізу та автоматизованого завантаження даних без необхідності ручного редагування файлів.

### 2.2 Архітектура програмного застосунку

Програмний застосунок реалізовано з використанням модульної архітектури, що забезпечує чіткий розподіл відповідальності між компонентами системи, спрощує підтримку коду та дозволяє незалежно модифікувати окремі частини застосунку без впливу на інші. Такий підхід є загальноприйнятою практикою при розробці аналітичних систем, оскільки дозволяє легко розширювати функціонал – наприклад, додати новий алгоритм машинного навчання або змінити спосіб візуалізації без необхідності переписувати всю

систему.

Проєкт має наступну структуру файлів і каталогів:

```
macro_forecast/
app.py
requirements.txt
modules/
     data_loader.py
     preprocessor.py
     models.py
     evaluator.py
     forecaster.py
     visualizer.py
models/
```

Точкою входу до застосунку є файл app.py, який реалізує користувацький інтерфейс засобами бібліотеки Streamlit та координує взаємодію між модулями. Важливим архітектурним рішенням було те, що app.py не містить жодної бізнес-логіки – він лише приймає дані від користувача, передає їх відповідним модулям та відображає отримані результати. Вся логіка обробки даних, навчання моделей та побудови прогнозів зосереджена у відповідних модулях каталогу modules/. Це відповідає принципу єдиної відповідальності, згідно з яким, кожен компонент системи повинен виконувати лише одну чітко визначену функцію. Каталог models/ використовується для збереження навчених моделей у форматі .pkl засобами бібліотеки Joblib. Це дозволяє повторно використовувати навчені моделі без необхідності повторного навчання, що є особливо корисним при роботі з великими обсягами даних [32].

У таблиці 2.2 наведено призначення кожного модуля системи та перелік основних функцій які він реалізовує. Окремої уваги заслуговує підхід до роботи з вхідними даними. Файли CSV з бази даних World Bank мають специфічний формат – перші кілька рядків містять службову інформацію, а роки представлені як окремі стовпці замість рядків. Модуль data_loader.py автоматично визначає рядок із заголовками шляхом пошуку ключового слова «Country Name» та перетворює дані з широкого формату у довгий, де кожний

рядок відповідає одному спостереженню для конкретної країни та року. Це дозволяє користувачеві завантажувати файли безпосередньо із сайту World Bank без будь-якого попереднього ручного редагування.

Таблиця 2.2 – Модулі програмного застосунку

| Модуль | Призначення | Основні функції |
|---|---|---|
| data_loader.py | Завантаження та злиття даних | Визначення заголовка CSV, перетворення wide=>long формату, злиття індикаторів |
| preprocessor.py | Підготовка даних | Сортування, видалення пропусків цільового показника, заповнення медіаною, створення лагів |
| models.py | Навчання моделей | Часове розбиття даних, навчання лінійної регресії та Random Forest, збереження моделей |
| evaluator.py | Оцінка точності | Обчислення MAE, MSE, RMSE, $R^2$ |
| forecaster.py | Прогнозування | Рекурсивний прогноз на N кроків вперед, формування датафрейму з історією та прогнозом |
| visualizer.py | Візуалізація | Графіки реальних та прогнозованих значень, важливість ознак, порівняння моделей, прогноз |

Важливим архітектурним рішенням є також забезпечення відповідності між даними та роками на етапі обробки. Оскільки модуль preprocessor.py видаляє певні рядки під час очищення даних, зокрема перші кілька рядків при створення лагових змінних, поряд з основним датафреймом підтримується окремий об’єкт actual_years, який зберігає реальні роки що відповідають кожному рядку після обробки. Це забезпечує коректне відображення років на

графіках, навіть для країн з великою кількістю пропусків у даних, таких як Україна, де більшість даних починаються лише з 1988 року.

Потік даних між модулями системи має послідовний характер і може бути представлений наступним чином: користувач завантажує CSV файли через інтерфейс => data_loader зливає їх в єдиний датасет => користувач обирає країну та цільовий показник => preprocessor виконує очищення та підготовку даних => models навчає обидві моделі з часовим розбиттям => evaluator обчислює метрики точності => forecaster будує прогноз => visualizer формує графіки для відображення в інтерфейсі. На кожному етапі, результат одного модуля є вхідними даними для наступного, що забезпечує чітку та передбачувану поведінку системи.

### 2.3 Реалізація основних модулів програмного застосунку

Реалізація програмного застосунку передбачає, як зазначалось раніше, створення шести окремих модулів, кожен з яких відповідає за певний етап обробки даних та побудови прогнозу. У даному підрозділі розглядає реалізація кожного модуля з описом ключових алгоритмічних рішень.

Модуль завантаження даних: модуль data_loader.py відповідає за завантаження CSV файлів з бази даних World Bank та їх злиття в єдиний датасет. Основною складністю при роботі з даними World Bank є специфічний формат файлів – службові рядки на початку документу та предсталення років як окремих стовпців замість рядків.

Для автоматичного визначення рядка із заголовками реалізовано допоміжну функцію _find_header_row, яка послідовно зчитує рядки файлу та шукає той, що містить ключове слово «Country Name»:

```
    def _find_header_row(file) -> int:
        content = file.read()
        file.seek(0)
        for i, line in enumerate(io.StringIO(content.decode("utf-
8", errors="ignore"))):
            if "Country Name" in line:
```

```
            return i
    return 4
```

Після визначення рядка заголовка, файл зчитується з пропуском службових рядків. Оскільки World Bank представляє роки як окремі стовпці, виконується перетворення з широкого формату у довгий за допомогою операції melt. Колонки-роки визначаються динамічно – як стовпці що місять лише цифри, що забезпечує сумісніть з файлами різних індикаторів незалежно від діапазону років. Назва індикатора зчитується автоматично з відповідної колонки та використовується як назва стовпця у результуючому датафреймі.

Якщо користувач завантажує кілька файлів одночасно, модуль зливає їх в єдиний датасет шляхом послідовного об'єднання по колонках «Country Name», «Country Code» та «Year» з використанням внутрішнього з'єднання. Це означає що у результуючому датасеті залишаються лише ті роки для яких наявні дані в усіх завантажених даних. Повний лістинг модуля наведено у Додатку А

Модуль для попередньої обробки даних: модуль preprocessor.py підготовку даних до навчання моделей та містить дві функції: add_lags для створення лагових змінних та preprocess для повної обробки датафрейму.

Також, важливою особливістю реалізації є те, що функція preprocess повертає два об'єкти – очищений датафрейм та окремий об'єкт years з реальними роками що відповідають кожному рядку після обробки даних. Це необхідно тому, що в процесі очищення деякі рядки видаляються, і без явного відстеження років, подальша візуалізація відображала б некоректні роки на графіках. Особливо критично це для країн з великою кількістю пропусків у даних.

Обробка виконується у наступній послідовності: вибір числових колонок, сортування по рокам, видалення рядків де відсутній цільовий показник, заповнення пропусків медіаною та створення лагових змінних. Для заповнення пропусків використовується медіана:

```
df.fillna(df.median(), inplace=True)
```

Медіана обрана замість середнього значення через її стійкість до викидів, які часто присутні в економічних даних – різких стрибків показників під час

криз та економічних потрясінь.

Після заповнення пропусків створюються лагові змінні цільового показника. Функція add_lags додає три нові колонки – значення цільового показника за один, два та три попередні роки:

```
    def add_lags(df: pd.DataFrame, target: str, n_lags: int = 3)
-> pd.DataFrame:
        df = df.copy()
        for lag in range(1, n_lags + 1):
            df[f"{target}_lag{lag}"] = df[target].shift(lag)
        return df
```

Лагові змінні є ключовим інструментом для прогнозування часових рядів – вони дозволяють моделі спиратись на реальну економічну динаміку попередніх років. Після створення лагів, перші три рядки містять NaN значення і видаляються разом з відповідними записами в об’єкті years, що забезпечує синзронність між даними та роками на всіх подальших етапах. Повний лістинг модуля наведено у додатку А.

Модуль навчання моделей: модуль models.py реалізує навчання двох алгоритмів машинного навчання та містить функцію часового розбиття даних temporal_train_test_split.

На відміну від стандартного випадкового розбиття, часове розбиття враховує послідовний характер економічних даних. Перші 80% спостережень за часом використовуються для навчання, решта 20% - для тестування. Це усуває проблему витоку даних, коли модель при випадковому розбитті могла б використовувати майбутні значення під час навчання:

```
    def temporal_train_test_split(df: pd.DataFrame, target: str,
test_size: float = 0.2):
        n = len(df)
        split_idx = int(n * (1 - test_size))
        if split_idx < 5:
            split_idx = max(2, n - 3)
```

```
    train = df.iloc[:split_idx]
    test = df.iloc[split_idx:]
    X_train = train.drop(columns=[target])
    y_train = train[target]
    X_test = test.drop(columns=[target])
    y_test = test[target]
    return X_train, X_test, y_train, y_test
```

Додатково реалізовано захист від малих датасетів – якщо після розбиття тренувальна вибірка містить менше п’яти спостережень, межа зсувається щоб гарантувати мінімально достатній обсяг навчальних даних.

Навчання виконується для обох алгоритмів з однаковими вхідними даними, що забезпечує коректне порівняння їх ефективності. Після навчання кожна модель зберігається коректне порівняння їх ефективності. Після навчання кожна модель зберігається на диск у форматі .pkl засобами бібліотеки Joblib. На початку роботи модуля автоматично створюється каталог models/ якщо він не існує:

```
os.makedirs("models", exist_ok=True)
```

Повний лістинг модуля наведено у Додатку А.

Модуль оцінювання моделей: модуль evaluator.py обчислює чотири метрики точності для кожної навченої моделі та повертає результати у вигляді датафрейму для відображення в інтерфейсі. Для обчислення використовується відповідні функції бібліотеки Scikit-learn, а RMSE обчислюється як квадратний корінь з MSE:

```
def evaluate(results: dict) -> pd.DataFrame:
    rows = []
    for name, res in results.items():
        y_test, y_pred = res["y_test"], res["y_pred"]
        rows.append({
         "Модель": name,
```

```
        "MAE":  round(mean_absolute_error(y_test, y_pred), 4),
        "MSE":  round(mean_squared_error(y_test, y_pred), 4),
        "RMSE": round(np.sqrt(mean_squared_error(y_test, y_pred)),
4),
        "R²":   round(r2_score(y_test, y_pred), 4)
        })
    return pd.DataFrame(rows)
```

Використання одразу чотирьох метрик дозволяє отримати комплексну оцінку якості моделі – MAE показує середню абсолютну похибку, MSE підсилює вплив великих помилок, RMSE повертає результат до початкових одиниць вимірювання, а $R^2$ відображає загальну якість апроксимації. Повний лістинг модуля наведено у Додатку А.

Модуль прогнозування: модуль forecaster.py реалізує рекурсивний прогноз на задану кількість років вперед. На початку роботи, функція перевіряє наявність лагових колонок та генерує виняток з описовим повідомленням якщо їх не знайдено:

```
lag_cols = [c for c in df.columns if c.startswith(f"{target}_lag")]
if not lag_cols:
    raise ValueError(
        f"Лагові колонки для '{target}' не знайдені. "
        f"Переконайтесь що preprocess() викликано з
target='{target}'."
    )
```

Рекурсивний підхід означає що прогноз кожного наступного року використовуватиме як вхідні дані прогнози попередніх років. На кожному кроці лагові значення оновлюються – lag1 отримує значення попереднього року, lag2 – два роки тому і так далі. Для всіх інших економічних показників використовуються їх останні відомі значення, що є обгрунтованим спрощенням для системи, оскільки реальні майбутні значення для цих показників є

невідомими.

Функція build_forecast_df формує єдиний датафрейм що містить як історичні дані так і прогнозовані значення з відповідною міткою за типом – «Історія» або «Прогноз». Це спрощує подальшу побудову графіка де обидві частини відображатимуться різними кольорами. Повний лістинг модуля наведено у додатку А.

Модуль візуалізації: Ммодуль visualizer.py містить чотири функції для побудови графіків засобами бібліотеки Matplotlib з подальшим відображенням в інтерфейсі Streamlit через виклик st.pyplot().

Функція plot_predictions будує графіки реальних та прогнозованих значень на тестовій вибірці для обох моделей паралельно. Ключовою особливістю є використання реальних років на осі X - для кожної моделі роки визначаються за індексами тестової вибірки з об’єкту actual_years:

```
if actual_years is not None:
    x_vals = actual_years.iloc[y_test.index].values
else:
    x_vals = range(len(y_test))
```

Функція plot_feature_importance відображає важливість ознак моделей у вигляді горизонтальної діаграми, що дозволяє начоно оцінити внесок кожного показника у прогноз. Функція plot_comparison будує порівняльні барчарти метрик RMSE та $R^2$ для обох моделей. Функція plot_forecast будує графік що поєднує повну історію показника та прогноз на майбутнє з вертикальною пунктирною лінією що позначає межу між відомими даними та прогнозом. Повний лістинг модуля наведено у додатку А.

### 2.4 Реалізація користувацького інтерфейсу

Користувацький інтерфейс програмного застосункуреалізовано засобами бібліотеки Streamlit у файлі app.py. Streamlit дозволяє створювати інтерактивні

веб-застосунки для аналізу даних без необхідності використовувати складні веб-технології, тому що весь інтерфейс описується безпосередньо у Python коді. Застосунок запускається локально та відкривається у браузері за адресою “http://localhost:8501”.

Файл app.py є точкою входу в систему та координує послідовний виклик усіх модулів відповідно до дій користувача. На початку роботи виконується налаштування сторінки та відображається головний заголовок:
st.set_page_config(page_title="Макроекономічний аналіз", layout="wide")

st.title("Аналіз та прогнозування макроекономічних показників")

Інтерфейс побудований за принципом послідовних кроків, де кожен наступний елемент стає доступним лише після виконання попереднього [36]. Це досягається шляхом вкладення блоків коду всередині умовних операторів.

Завантаження даних: першим елементом інтерфейсу є компонент завантаження файлів, який дозволяє користувачу обрати один або кілька CSV файлів з локального диску:

```
uploaded_files = st.file_uploader(
    "Завантажте CSV файли з World Bank (кожен індикатор - окремий файл)",
    accept_multiple_files=True,
    type=["csv", "xlsx"]
)
```

Параметр accept_multiple_files=True дозволяє завантажувати кілька файлів одночасно, що є необхідним оскільки кожен макроекономічний індикатор на сайті World Bank зберігається в окремому файлі. Після завантаження файлів автоматично викликається модуль data_loader який зливає їх в єдиний датасет.

Попередній перегляд та аналіз структури даних: після злиття даних, користувачу пропонується обрати країну для аналізу зі списку всіх країн що містяться у завантажених файлах. Далі відображається блок аналізу структури

датасету що містить три ключові метрики, такі як кількість рядків, кількість стовпців та діапазон років:

```
col1, col2, col3 = st.columns(3)
col1.metric("Кількість рядків", df_country.shape[0])
col2.metric("Кількість стовпців", df_country.shape[1])
col3.metric("Діапазон років",
    f"{int(df_country['Year'].min())}-
{int(df_country['Year'].max())}")
```

Використання st.columns дозволяє розмістити метрики горизонтально, що робить інтерфейс більш компактним та зручним для сприйняття. Нижче відображається таблиця з кількістю пропусків для кожного показника або повідомлення про їх відсутність. Додатково у розгортуваному блоці st.expander, доступна детальна описова статистика серед якої можна знайти: мінімум, максимум, середнє, медіана та стандартне відхилення кожного показника.

На рис. 2.7 та 2.8 зображено вікно завантаження даних та блок попереднього аналізу структури датасету.

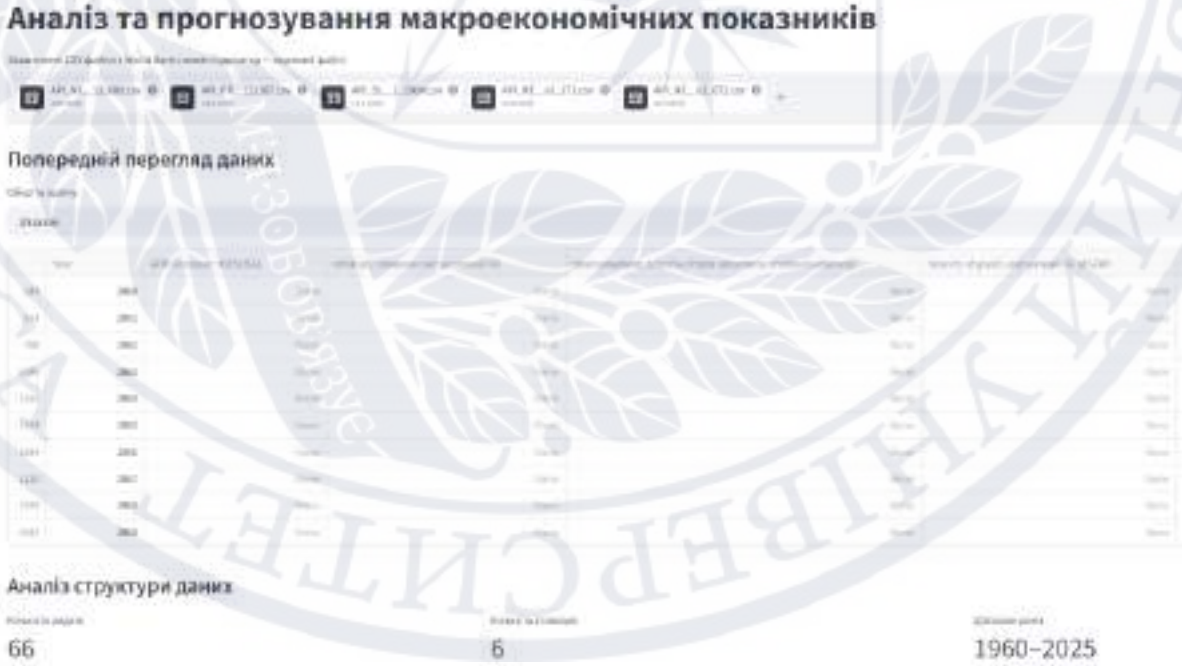


Рисунок 2.7 – Інтерфейс завантаження даних та аналізу структури датасету ч.1

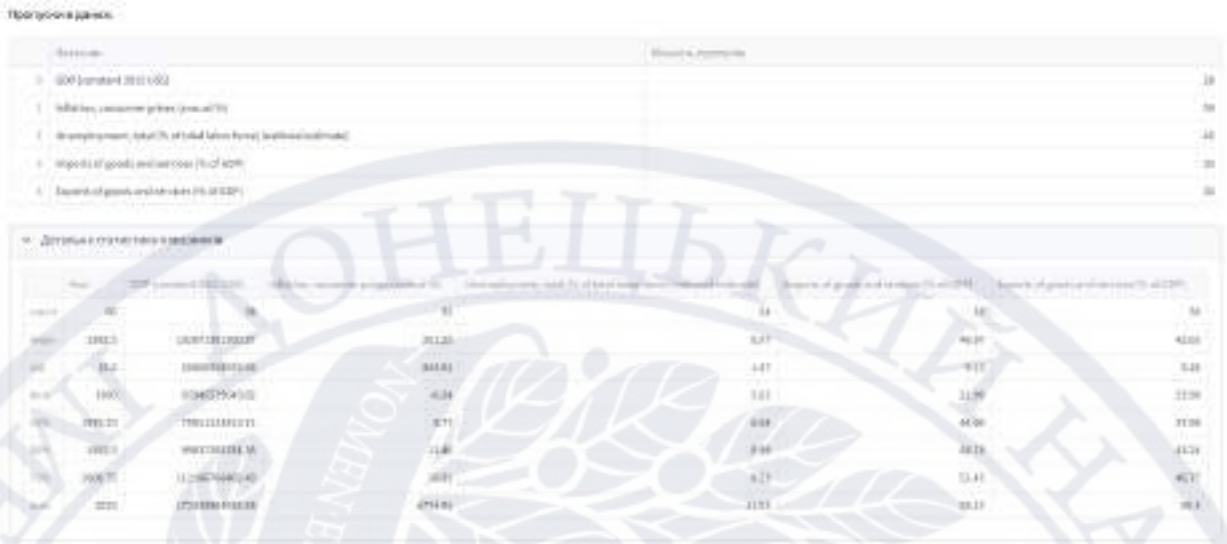

Рисунок 2.8 – Інтерфейс завантаження даних та аналізу структури датасету ч.2

Налаштування прогнозування: після аналіз структури, користувач обирає цільовий показник для прогнозування зі списку доступних індикаторів та задає кількість років прогнозу:

```
target = st.selectbox("Оберіть цільовий показник для прогнозування", indicator_cols)
n_forecast = st.number_input(
    "Кількість років прогнозу", min_value=1, max_value=20, value=5, step=1
)
```

Компонент st.number_input обмежує введення значень у діапазоні від 1 до 20 років, що запобігає побудові нереалістично довгих прогнозів з надмірним накопиченням похибки.

Навчання моделей та відображення результатів: запуск процесу навчання здійснюється натисканням кнопки. Перед навчанням виконується перевірка обсягу даних, де у випадку якщо датасет містить менше десяти спостережень, користувачу відображатиметься наступне попередження:

```
if len(df_country) < 10:
```

```
st.warning(
    f"Датасет містить лише {len(df_country)} спостережень - "
    f"точність моделей може бути низькою."
)
```

Під час виконання тривалих операцій, таких як завантаження даних, обробки та навчання, буде відображатись індикатор завантаження st.spinner з відповідним текстовим повідомленням, що інформує користувача про поточний етап роботи системи.

Результати відображаються послідовно у наступному порядку: спочатку виводиться таблиця метрик точності для обох моделей та тестовій вибірці. Для відображаються графіки реальних та прогнозованих значень, діаграма важдивості ознак та порівняльні барчарти метрик RMSE і $R^2$. На рис. 2.9 зображено блок результатів навчання моделей.

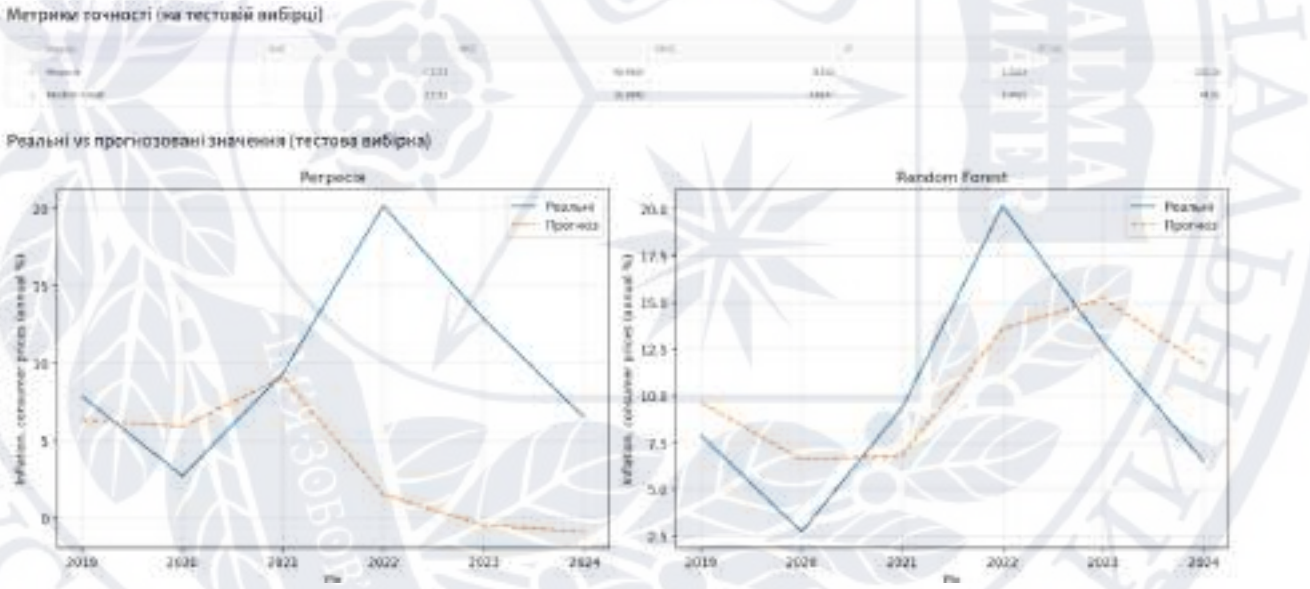


Рисунок 2.9 – Метрики точності та графіки реальних і прогнозованих значень (Інфляція України)

Завершальним блоком інтерфейсу є секція прогнозу на майбутнє. Для кожної окремої моделі відображається графік що поєднує повну історію показника та прогнозовані значення, а також зведена таблиця прогнозованих значень з реальними роками у вигляді індексу:

```
forecast_table.index = [
    str(last_known_year + i + 1) for i in range(int(n_forecast))
]
forecast_table.index.name = "Рік"
st.dataframe(forecast_table, use_container_width=True)
```

На рис. 2.10, 2.11 та 2.12 зображено графіки прогнозу та таблицю прогнозованих значень.


Рисунок 2.10 – Прогноз методом Регресії

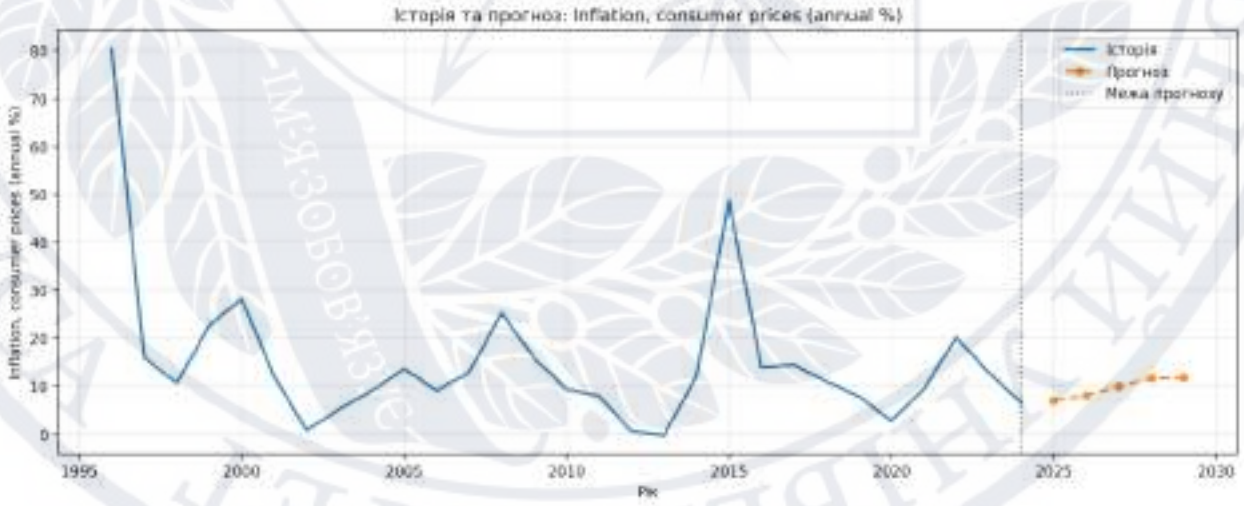


Рисунок 2.11 – Прогноз методом Random Forest


Рисунок 2.12 – Прогнозовані значення на прикладі Інфляції України

Повний лістинг файлу app.py наведено у Додатку А.

Запуск програмного застосунку: для спрощення запуску програмного застосунку розроблено командний файл start_program.bat, який автоматизує всі необхідні кроки для старту застосунку. Файл послідовно виконує перехід до каталогу проєкту, активацію віртуального середовища Python та запуск Streamlit застосунку.

# РОЗДІЛ 3
# ОЦІНКА МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ ПРИ ПРОГНОЗУВАННІ МАКРОЕКОНОМІЧНИХ ІНДИКАТОРІВ

## 3.1 Тестування програмного застосунку на наборах даних

Для тестування програмного застосунку було виділено дані чотирьох країн з різними економічними характеристиками: Німеччини та Швеції, як представників стабільних розвинених економік, Польщі, як економіки що розвивається, та України, як країни з нестабільною економікою що зазнала зовнішніх шоків. Такий вибір дозволяє оцінити поведінку моделей в різних економічних умовах. Для кожної країни аналізувались два макроекономічні показники, такі як ВВП у постійних цінах 2015 року та інфляція споживчих цін. Дані завантажуються з бази World Development Indicators Світового банку у форматі CSV окремим файлом для кожного індикатора та автоматично зливались системою.

При виборі показника ВВП для аналізу перевагу надано значенням у постійних цінах 2015 року замість поточних цін. Це обумовлено двома факторами. По-перше усуваються спотворення спричинені інфляцією, одне з яких це номінальне зростання ВВП у поточних цінах яке може відображати не реальне збільшення виробництва а лише підвищення загального рівня цін. По-друге фіксований курс базового 2015 року усуває вплив коливань валютних курсів для країн з національною валютою відмінною від долара США ВВП у поточних цінах суттєво змінюється при зміні обмінного курсу навіть якщо реальна економічна активність залишається незмінною. Практична перевірка підтвердила важливість цього вибору, де при використанні поточних цін графік ВВП Німеччини показував деякі падіння спричинені знеціненням євро відносно долара, тоді як у постійних цінах відображається реальна динаміка економіки без курсових спотворень.

У таблиці 3.1 наведено характеристики датасету для кожної країни після обробки даних.

Таблиця 3.1 – Характеристики датасетів

| Країна | Показник | Діапазон років | Кількість спостережень |
|---|---|---|---|
| Німеччина | ВВП | 1963-2024 | 62 |
| Німеччина | Інфляція | 1963-2024 | 62 |
| Швеція | ВВП | 1963-2024 | 62 |
| Швеція | Інфляція | 1963-2024 | 62 |
| Польща | ВВП | 1993-2024 | 32 |
| Польща | Інфляція | 1974-2024 | 51 |
| Україна | ВВП | 1990-2024 | 35 |
| Україна | Інфляція | 1996-2024 | 29 |

Різниця в кількості спостережень між країнами пояснюється доступністю даних у базі Світового банку, де як приклад, для Німеччини та Швеції дані доступні з початку 1960-их років, тоді як для Польщі та України повний набір індикаторів з’явився лише після переходу до ринкової економіки.

Показник державного боргу був виключений з набору ознак через критичну кількість пропущених значень у базі Світового банку, що призводить до того що для більшості досліджуваних країн відсутня значна частина річних спостережень що унеможливлює коректне використання цього індикатора в якості ознаки для навчання моделей. У таблиці 3.2 наведено результати оцінки точності обох моделей для кожної країни та показника.

Таблиця 3.2 – Метрики точності моделей

| Країна | Показник | Модель | MAE | RMSE | $R^2$ | $R^2$(%) |
|---|---|---|---|---|---|---|
| Німеччина | ВВП | Регресія | 58.6 млрд | 83.2 млрд | 0.6933 | 69.33% |
| Німеччина | ВВП | Random Forest | 344.7 млрд | 374.1 млрд | -5.205 | -520.5% |
| Німеччина | Інфляція | Регресія | 1.14 | 1.38 | 0.503 | 50.3% |
| Німеччина | Інфляція | Random Forest | 1.38 | 1.65 | 0.2952 | 29.52% |

Продовження табл. 3.2

| Країна | Показник | Модель | MAE | RMSE | $R^2$ | $R^2$(%) |
|---|---|---|---|---|---|---|
| Швеція | ВВП | Регресія | 12.2 млрд | 15.1 млрд | 0.8298 | 82.98% |
| Швеція | ВВП | Random Forest | 76.7 млрд | 82.9 млрд | -4.1029 | -410.29% |
| Швеція | Інфляція | Регресія | 1.79 | 2.04 | 0.4624 | 46.24% |
| Швеція | Інфляція | Random Forest | 1.47 | 2.26 | 0.3439 | 34.39% |
| Польща | ВВП | Регресія | 17.7 млрд | 19.3 млрд | 0.7393 | 73.93% |
| Польща | ВВП | Random Forest | 100.2 млрд | 107.5 млрд | -7.1073 | -710.73% |
| Польща | Інфляція | Регресія | 53.48 | 75.64 | -261.67 | -26166% |
| Польща | Інфляція | Random Forest | 4.15 | 5.12 | -0.2054 | -20.54% |
| Україна | ВВП | Регресія | 7.7 млрд | 12.5 млрд | -0.0406 | -4.06% |
| Україна | ВВП | Random Forest | 9.1 млрд | 14.1 млрд | -0.334 | -33.4% |
| Україна | Інфляція | Регресія | 7.38 | 9.92 | -2.2513 | -225.13% |
| Україна | Інфляція | Random Forest | 3.72 | 4.08 | 0.4491 | 44.91% |

Аналіз результатів дозволяє виділити кілька характерних закономірностей. Регресія стабільно показує кращі результати по ВВП для країн зі стабільною економікою, як Швеція 82.98%, Польща 73.93%, Німеччина 69.33%. Random Forest в усіх цих випадках демонструє від'ємний $R^2$ що свідчить про перетренування на обмеженій вибірці.

Виняток становить інфляція України де Random Forest показав $R^2$=44.91% проти -225.13% у регресії. Це єдиний випадок серед вибраних країн, де

складніша модель перевершила просту.

Окремої уваги заслуговує інфляція Польщі де обидві моделі показали катастрофічні результати. Це пояснюється гіперінфляцією початку 1990-их років, де інфляція досягала тисячі відсотків на рік, що неможливо коректно апроксимувати поряд із сучасними значеннями у 2-3%. Random Forest впорався значно краще (-20.54% проти -26166%) але обидва результати є неприйнятними для практичного використання.

### 3.2 Порівняльний аналіз алгоритмів машинного навчання

За результатами тестування системи на даних чотирьох країн можна виділити декілька ключових закономірностей, що характеризують поведінку обох алгоритмів в різних економічних умовах.

Проведемо оцінку моделі регресії на показнику ВВП. Регресія стабільно перевершує Random Forest при прогнозі ВВП для всіх досліджувальних країн. Коефіцієнт детермінації складає 82.98% для Швеції, 73.93% для Польщі та 69.33% для Німеччини. Random Forest у всіх трьох випадках демонструє від'ємний $R^2$, як від -410.29% для Швеції до -710.73% для Польщі – це свідчить про те що модель прогнозує гірше ніж просте передбачення середнього значення. Причиною такої різниці є обмежений обсяг навчальної вибірки. ВВП має виражений монотонний тренд зростання який регресія апроксимує доволі ефективно завдяки своїй простоті. Random Forest натомість потребує більшої кількості спостережень для стабільного навчання, тому що при 30-60 спостереженнях він схильний до перетренування коли модель запам'ятовує тренувальні дані замість виявлення загальних закономірностей.

На рис. 3.1 наведено графіки реальних та прогнозованих значень ВВП Швеції для обох моделей. Регресія відтворює загальну тенденцію тоді як Random Forest демонструє майже горизонтальну лінію, що є характерною ознакою перетренування на малій вибірці.

Оцінка Random Forest на нелінійних показниках показала, що єдиним випадком, серед вибраних країн, де Random Forest перевершив регресію є

інфляція України, що має значення $R^2$ в 44.91% проти -225.13% регресії. Це принципово відрізняється від результатів по ВВП і потребує окремого пояснення. Інфляція України має виражений нелінійний характер – різкі стрибки у 1993-1994 роках, стабілізація у 2000-их, новий стрибок у 2015-2016 роках та у 2022-2023.

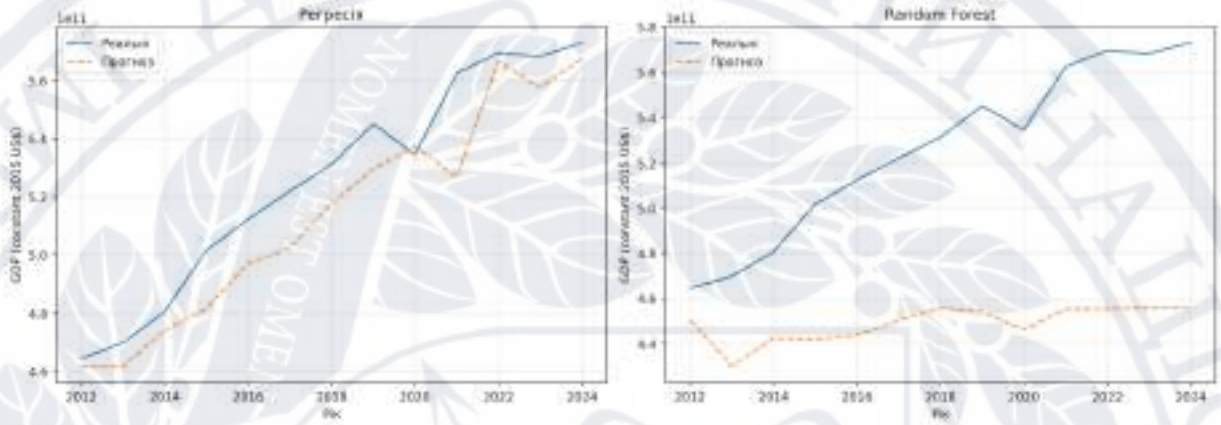


Рисунок 3.1 – Реальні та прогнозовані значення ВВП Швеції

Регресія шукає лінійну залежність між ознаками і не здатна коректно апроксимувати такі різні нелінійні зміни, звідки випливає і такий катастрофічний результат. Random Forest, через механізм розбиття простору ознак деревами рішень, краще вловлює нелінійні патерни і повторювані стрибки.

На рис. 3.2 наведено графіки реальних та прогнозованих значень інфляції України. Можна спостерігати, що Random Forest значно точніше відтворює характерні піки порівняно з регресією.

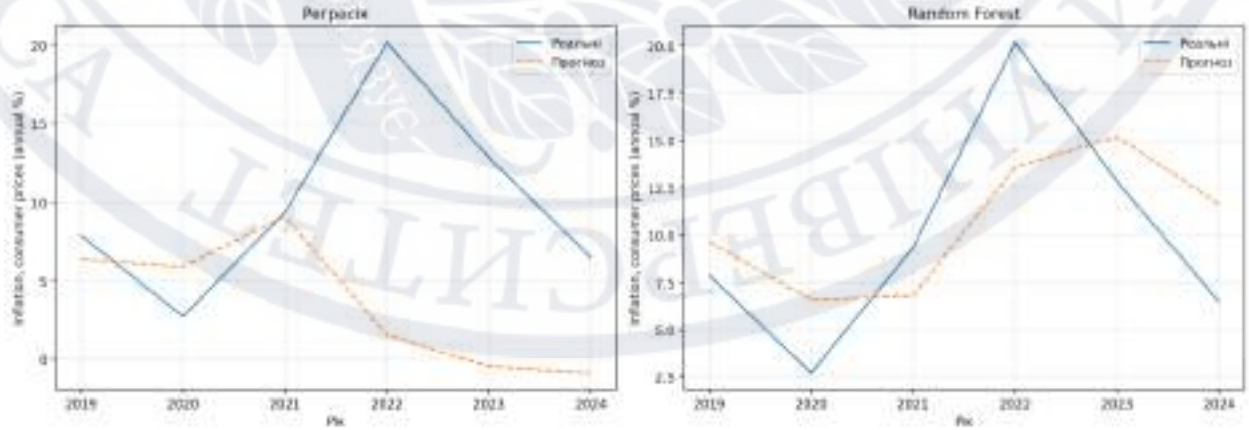


Рисунок 3.2 – Реальні та прогнозовані значення інфляції України

Оцінка аномалій показала, що аномальний результат інфляції властивий для характеристик Польщі. Окремої уваги заслуговує інфляція Польщі, де обидві моделі показали неприйнятні результати, такі як $R^2$ регресії -26166% та Random Forest -20.54%. Така різниця між двома моделями, як коротко зазначалось раніше, пояснюється гіперінфляцією початку 1990-их років, коли річна інфляція сягала кількох тисяч відсотків. Регресія намагається знайти єдину лінійну залежність, що охоплює одночасно значення 1000% та сучасні 2-3%, що є методологічно неможливим. Random Forest, завдяки локальному характеру розбиття деревами, частково ізолює екстремальні значення від сучасних і показує значно кращий, хоча і все одно незадовільний, результат.

Цей випадок демонструє важливе обмеження системи – що при наявності екстремальних історичних значень, якість прогнозу суттєво знижується для обох алгоритмів. Потенційним рішенням могло би бути обмеження діапазону аналізованих років або логарифмічне перетворення показника.

Оцінка результатів ВВП по Україні показала, що ВВП України є окремим складним випадком, де обидві моделі показали від'ємні результати у вигляді $R^2$ -4.06% для регресії та -33.4% для Random Forest. Незважаючи на від'ємний $R^2$, графік рис. 3.3 свідчить що регресія частково вловлює загальну тенденцію показника до 2021 року – прогнозована крива відносно близька до реальної в період стабільності.


Рисунок 3.3 – Графік реальних та прогнозованих значень, ВВП України

На відміну від інфляції, де нелінійність є закономірним шаблоном, падіння ВВП у 2014-2015 та 2022 спричинені зовнішніми шоками, такі як збройні конфлікти та війна, які не мають аналогів у тренувальних даних. Жодна модель машинного навчання не здатна передбачити події, які принципово відрізняються від усього що вона бачила під час навчання.

На рис. 3.4 наведено графік важливості ознак для Random Forest при прогнозуванні ВВП Німеччини.

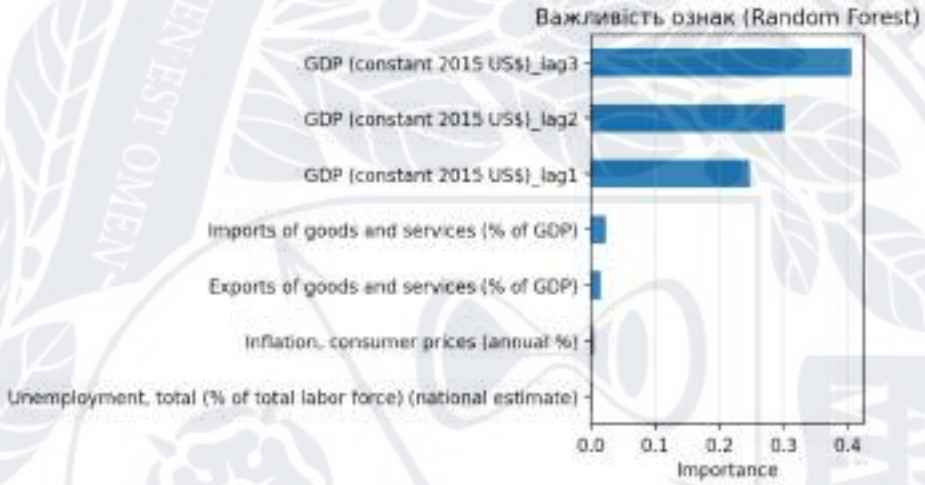


Рисунок 3.4 – Важливість ознак для Random Forest, ВВП Німеччини

Найбільший внесок мають лагові змінні цільового показника, такі як lag3, lag2, lag1, що підтверджує попередні значення ВВП є найкращим предиктором майбутнього. Показники імпорту та експорту займають четверте та п’яте місце що узгоджується з економічною теорією, яка полягає у тому, що Німеччина є однією з найбільших експортних економік світу, для якої зовнішня торгівля є прямою складовою ВВП.

У випадку України, аналіз важливості ознак для інфляції (рис. 3.5) виявив що найбільший вплив на прогноз має lag2 – значення інфляції два роки тому, тоді як у більшості інших країн lag1 або lag3 є домінуючими. Показник експорту займає четверте місце що узгоджується з експортно орієнтованим характером української економіки. Показово, що імпорт має мінімальний вплив – на відміну від більшості інших країн де імпорт та експорт мають схожу важливість. Це може пояснюватись тим що в умовах економічних криз імпорт різко

скорочується і втрачає предикторну силу.

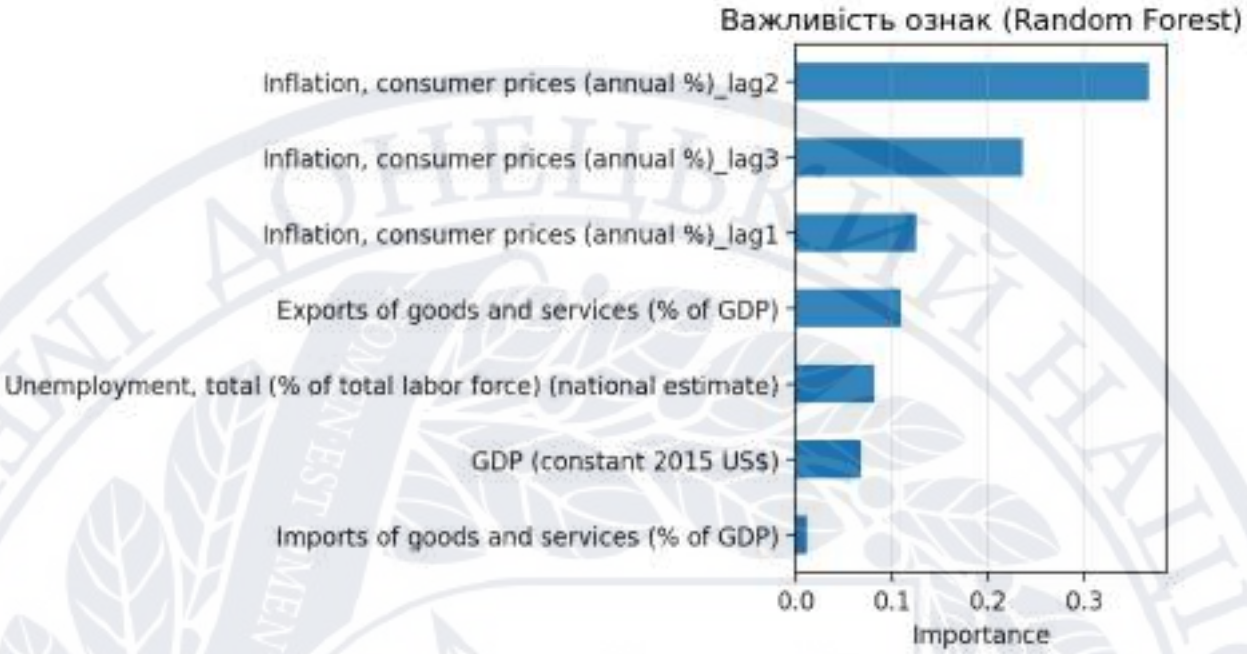


Рисунок 3.5 – Важливість ознак для Random Forest, Інфляція України

Для порівняння на рис. 3.6 наведено графік важливості ознак для Random Forest при прогнозуванні ВВП України.

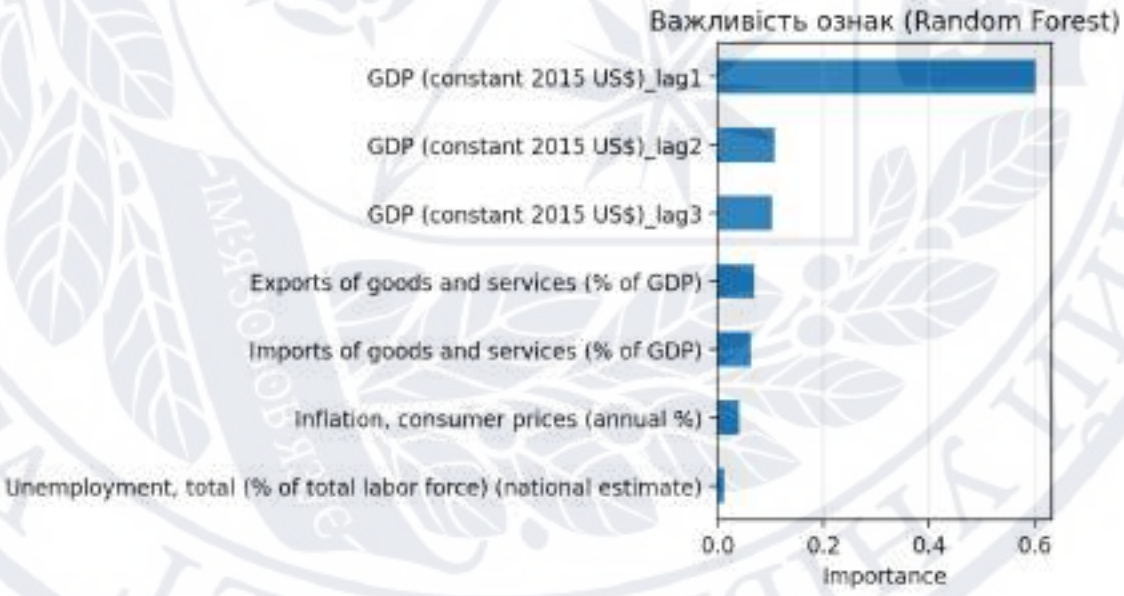


Рисунок 3.6 - Важливість ознак для Random Forest, ВВП України

На відміну від Німеччини де найбільший вплив мав lag3, для України домінуючою ознакою є lag1 з важливістю близько 0.6. Це пояснюється

нестабільним характером української економіки – через різкі стрибки ВВП у кризові роки найсвіжіше попереднє значення несе значно більше інформації про поточний стан ніж більш віддалені лаги. Цікаво що для ВВП Польщі спостерігається протилежна ситуація, де lag3 є домінуючою ознакою з важливістю близько 0.33, що свідчить про більш стабільний характер польської економіки де довгострокові тенденції є кращим предиктором ніж короткострокові коливання. Таким чином розподіл важливості лагових змінних може слугувати непрямим індикатором економічної стабільності країни.

Додатково проведено аналіз прогнозування рівня безробіття України. На рис. 3.7 наведено графіки реальних та прогнозованих значень безробіття на тестовій вибірці. Random Forest показав кращі результати з RMSE 0.78 проти 1.16 у регресії та $R^2$=-73.67% проти -290.69%. Хоча обидві моделі демонструють від'ємний $R^2$ що свідчить про складність прогнозування цього показника в умовах нестабільної економіки, Random Forest знову виявився стійкішим що узгоджується з результатами прогнозування інфляції.


Рисунок 3.7 – Графік реальних та прогнозованих значень, показник Безробіття України

Особливо цікавим є графік важливості ознак за прогнозом безробіття (рис. 3.8) де ВВП у постійних цінах домінує з важливістю близько 0.75 і значно перевищуючи власні лагові змінні безробіття. Це відповідає відомому в

економічній теорії закону Оукена який описує обернену залежність між темпами зростання ВВП та рівнем безробіття. Той факт що модель самостійно виявила цю фундаментальну економічну залежність без будь-яких попередніх знань підтверджує здатність алгоритмів машинного навчання ідентифікувати реальні економічні зв'язки з даних.


Рисунок 3.8 – Важливість ознак для Random Forest, показник Безробіття України

У таблиці 3.3 наведено узагальнені висновки щодо доцільності застосування кожного алгоритму залежно від характеру даних.

Таблиця 3.3 – Порівняння алгоритмів за умовами застосування

| Умова | Регресія | Random Forest |
|---|---|---|
| Стабільна економіка, ВВП | Краща | Перетренування |
| Нелінійний показник (інфляція) | Гірша | Краща |
| Мала вибірка | Стійка | Перетренування |
| Зовнішні шоки в даних | Обидві моделі неефективні | Обидві моделі неефективні |
| Гіперінфляція в історії | Критично погана | Частково справляється |

Слід також зазначити що якість результатів суттєво залежить від обсягу доступних даних, серед яких країни з коротшою історією спостережень як Польща (32 роки по ВВП) або Україна (35 років по ВВП) демонструють нижчу стабільність моделей порівняно з Німеччиною та Швецією де доступні дані з початку 1960-х.

Таким чином, експериментально підтверджено що не існує універсального алгоритму який би показав найкращі результати в усіх умовах. Вибір методі прогнозування має базуватись на характері конкретного економічного показника, обсязі доступних даних та економічній стабільності досліджувальної країни.

### 3.3 Оцінка якості прогнозних характеристик

За результатами проведення експериментального дослідження на реальних макроекономічних даних чотирьох країн можна сформулювати розгорнуті висновки щодо застосування алгоритмів машинного навчання для прогнозування макроекономічних показників.

Регресія продемонструвала високу якість оцінок при прогнозуванні ВВП у постійних цінах для країн зі стабільною економікою. Найкращий результат серед досліджених країн, отримано для Швеції з показником $R^2$ у 82.98%, що означає що модель пояснює майже 83% варіації показника. Для Польщі та Німеччини склали 73.93% та 69.33% відповідно. Ці результати є цілком прийнятними для поточної системи і свідчать про те що регресія здатна ефективно вловлювати довгострокові тенденції зростання ВВП.

Спільною рисою для всіх трьох країн є те що Random Forest показав від'ємний $R^2$ при прогнозуванні ВВП незважаючи на те що теоретично він є складнішим і потенційно кращим алгоритмом. Це підтверджує відому в машинному навчанні закономірність яка полягає у тому що складніші моделі не завжди перевершують прості на малих вибірках. При обсязі від 32 до 62 спостережень Random Forest не має достатньо даних для виявлення стабільних закономірностей і схильний до перетренування.

Окремим складним випадком є ВВП України, де обидві моделі показали ві'дємний $R^2$ у -4.06% для регресії та -33.4% для Random Forest. Це не є свідченням помилки в реалізації системи, а відображає об'єктивну неможливість прогнозування різких структурних змін спричинених зовнішніми чинниками. Падіння ВВП України у 2014-2015 роках внаслідок анексії Криму та збройного конфлікту на сході країни, а також катастрофічне падіння у 2022 році через повномаштабне вторгнення є подіями що принципово виходять за межі будь яких економічних закономірностей які модель могла б виявити в тренувальних даних. Навіть найсучасніші економічні моделі МВФ та Світового банку не в змозі предебачати такі падіння внаслідок таких обставин.

На рахунок якості прогнозування інфляції, результати виявились більш неоднозначними і різняться залежно від країни та характеру самого показника.

Для Німеччини та Швеції регресія показала помірні але прийнятні результати які становлять 50.3% та 46.24% відповідно. Random Forest поступився регресії в обох випадках у 29.52% та 34.39%. Інфляція цих країн є відносно стабільною і не має різких стрибків що пояснює чому простіша модель справляється краще.

Принципово інша ситуація склалась для інфляції України, що є єдиним випадком серед вибраних країн де Random Forest перевершив регресію з результатом 44.91% проти -255.13%. Інфляція України має виражений нелінійний характер з різкими стрибками у кризові роки, що є типовим прикладом даних де ансамблеві методи мають перевагу над лінійними. Цей результат підтверджує теоретичне положення про те що Random Forest ефективніший на нелінійних залежностях.

Найпроблемнішим виявився прогноз інфляції Польщі, де обидві моделі показали неприйнятні результати через наявність гіперінфляції початку 1990-их у тренувальних даних. Регресія отримала катастрофічний результат у -26166% оскільки намагалась знайти єдину лінійну залежність що охоплює одночасно значення кількох тисяч відсотків та сучасні 2-3%. Random Forest частково впорався з цим завданням показавши -20.54%, що є значно кращим результатом

хоча і все одно неприйнятний для практичного використання. Цей випадок демонструє що наявність екстремальних історичних значень у тренувальних даних є серйозним викликом для будь-якого алгоритму машинного навчання і потребує додаткової попередньої обробки даних.

Аналіз важливості ознак для моделі Random Forest підтвердив ключову роль лагових змінних цільового показника у прогнозуванні. Для всіх досліджуваних країн та показників, лаги lag1, lag2, lag3 займають перші позиції за важливістю що свідчить по те що попередні значення показника є найкращим предиктором його майбутніх значень.

Особливо цікавим є результат для ВВП Німеччини, де показники імпорту та експорту посідають четверте та п’яте місця за важливістю після лагових змінних. Це повністю узгоджується з економічною теорією, яка полягає у тому що Німеччина є однією з найбільших експортних економік світу і зовнішня торгівля є прямою складовою ВВП. Той факт що модель самостійно виявила дану залежність без будь-яких апріорних знань про економіку Німеччини є свідченням того що алгоритм коректно ідентифікував реальні економічні зв’язки.

Для Швеції спостерігається подібна закономірність, де показник експорту займає вищу позицію над одним лаговим показником серед важливості ознак, що також узгоджується з економічною роллю зовнішньою торгівлі для шведської економіки.

Щодо практичної застосовності системи, результати тестування довзоляють окреслити умови за яких розроблена система може бути практично корисною та умови де її застосування є обмеженим.

Система демонструє найкращі результати для країн зі стабільною економікою та достатнім обсягом історичних даних у щонайменше 50-60 річних спостережень. За таких умов регресія здатна забезпечити прийнятну точність прогнозування ВВП з $R^2$ понад 70%. Застосування системи є обмеженим для країн що пережили різкі структурні зміни або зовнішні шоки такі як – збройні конфлікти, гіперінфляцію, системні економічні кризи. У таких випадках якість

прогнозу суттєво знижується незалежно від обраного алгоритму, оскільки модель не може передбачити події що не мають аналогів у тренувальних даних.

На рис. 3.9 наведено прогноз інфляції України на 5 років вперед побудований моделлю Random Forest. Прогноз демонструє помірне зростання інфляції в межах 7-12% що відповідає загальній тенденції стабілізації після кризових піків. Поступове вирівнювання кривої прогнозу є характерною поведінкою рекурсивного підходу – з кожним кроком модель спирається на власні прогнози а не на реальні значення що призводить до згладжування коливань на довшому горизонті.


Рисунок 3.9 – Графік прогнозу інфляції України на 5 років

Для порівняння, на рис. 3.10 наведено прогноз ВВП України на 5 років вперед побудований моделлю регресії. На відміну від інфляції де прогноз демонструє помірне зростання, прогноз ВВП показує поступове відновлення після падіння 2022 року що відповідає загальній тенденції відновлення економіки. Результати слід інтерпретувати як орієнтовну тенденцію за умови відсутності нових зовнішніх шоків.

Окремим обмеженням є використання виключно річних даних з бази Світового банку. Збільшення частоти спостережень, як приклад – використання квартальних даних з Євростату для країн ЄС, могло би суттєво покращити результати Random Forest який потребує більшого обсягу даних для стабільного

навчання. Однак це лише охоплювало б країни Європейського Союзу та призвело б до втрати уніфікованості даних між країнами що є принциповою перевагою обраного джерела для порівняльного аналізу.


Рисунок 3.10 - Графік прогнозу ВВП України на 5 років

Щодо вибору алгоритму, одним з ключових висновків дослідження є підтвердження того що не існує універсального алгоритму машинного навчання який би показав найкращі результати в усіх умовах. Вибір методу прогнозування має базуватись на комплексному аналізі характеру показника, обсягу доступних даних та економічної стабільності досліджуваної країни.

Отже, регресія є кращим вибором для прогнозування показників з монотонним трендом на обмежених вибірках завдяки своїй простоті та стійкості до перетренування. Random Forest доцільніше застосовувати для показників з вираженою нелінійністю та частими стрибками за умови наявності достатнього обсягу навчальних даних.

# ВИСНОВКИ

У даній роботі вирішено комплекс завдань пов’язаних з розробкою програмного застосунку для реалізації алгоритмів аналізу та прогнозування макроекономічних індикаторів держави на основі машинного навчання. За результатами роботи можна зробити наступні висновки:

1. Досліджено основні макроекономічні індикатори та їх значення. Розглянуто сутність та економічний зміст ключових показників – валового внутрішнього продукту, рівня інфляції, безробіття, а також показників зовнішньої торгівлі. Встановлено що дані показники є взаємопов’язаними і комплексно характеризують стан національної економіки. Визначено що для коректного міжнародного порівняння доцільно використовувати ВВП у постійних цінах базового року що усуває вплив інфляції та коливань валютних курсів. Проаналізовано сучасні методи машинного навчання для задач прогнозування. Розглянуто алгоритми регресії та Random Forest, досліджено їх теоретичні основи, переваги та обмеження. Встановлено що регресія є ефективною для даних з монотонним трендом та обмеженим обсягом спостережень, тоді як Random Forest краще справляється з нелінійними залежностями за наявності достатньої кількості навчальних даних. Визначено метрики оцінювання якості моделей – MAE, MSE, RMSE та коефіцієнт детермінації $R^2$.

2. Обґрунтовано вибір технологій та інструменти для реалізації програмного застосунку, зокрема: Python та бібліотеки Pandas, NumPy, Scikit-learn, Matplotlib та Streamlit. Як джерело даних обрано базу World Development Indicators Світового банку що забезпечує уніфіковану методологію для всіх країн світу в єдиному форматі CSV.

3. Реалізовано алгоритми аналізу та прогнозування макроекономічних індикаторів. Розроблено модуль рекурсивного прогнозування, що будує прогноз на задану кількість років вперед на основі лагових змінних цільового

показника. Реалізовано часове розбиття даних що усуває проблему витоку даних характерну для випадкового розбиття часових рядів.

4. Виконано аналіз та попередню обробку даних. Розроблено модуль автоматичного завантаження та злиття файлів World Bank що враховує специфічний формат цього джерела даних. Реалізовано обробку пропусків – видалення рядків де відсутній цільовий показник та заповнення медіаною для решти показників. Створено механізм лагових змінних що дозволяє моделі спиратись на реальну економічну динаміку попередніх років.

5. Реалізовано користувацький інтерфейс програмного застосунку засобами бібліотеки Streamlit. Інтерфейс забезпечує завантаження даних, аналіз структури датасету, вибір цільового показника та кількості років прогнозу, запуск навчання моделей та відображення результатів у вигляді таблиць і графіків.

6. Проведено оцінку точності побудованих моделей та результатів роботи алгоритмів на даних чотирьох країн. Експериментально підтверджено що регресія показує кращі результати при прогнозуванні ВВП для стабільних економік – $R^2$ до 82.98% для Швеції. Random Forest перевершує регресію лише на нелінійних показниках з частими стрибками, зокрема на інфляції України де отримано $R^2$=44.91%. Встановлено, що низькі значення $R^2$ для обох моделей при прогнозуванні показників України пояснюються зовнішніми економічними шоками які не мають аналогів у тренувальних даних і не можуть бути передбачені жодним алгоритмом машинного навчання.

Розроблений програмний застосунок повністю реалізує визначені функції та може бути використаний як демонстраційний і навчальний інструмент у сфері аналітики макроекономічних показників. Перспективами подальшого розвитку є розширення набору алгоритмів машинного навчання, використання квартальних даних для збільшення обсягу навчальної вибірки та реалізація механізму автоматичного вибору алгоритму на основі характеру вхідних даних.

# СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd Edition. O'Reilly Media, 2022. 861 p.

2. McKinney W. Python for Data Analysis. 3rd Edition. O'Reilly Media, 2022. 579 p.

3. James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning with Applications in Python. Springer, 2023. 617 p.

4. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. 2nd Edition. Springer, 2009. 745 p.

5. Wooldridge J.M. Introductory Econometrics: A Modern Approach. 7th Edition. Cengage Learning, 2019. 789 p.

6. Hyndman R.J., Athanasopoulos G. Forecasting: Principles and Practice. 3rd Edition. OTexts, 2021. 442 p.

7. Breiman L., Friedman J., Olshen R., Stone C. Classification and Regression Trees. Chapman and Hall, 1984. 368 p.

8. VanderPlas J. Python Data Science Handbook. O'Reilly Media, 2016. 548 p.

9. Бідюк П.І., Романенко В.Д., Тимощук О.Л. Аналіз часових рядів. Київ: Політехніка, 2013. 600 с.

10. Лук'яненко І.Г., Краснікова Л.І. Економетрика. Київ: Товариство "Знання", КОО, 1998. 494 с.

11. Breiman L. Random Forests. Machine Learning. 2001. № 45. С. 5-32.

12. Box G.E.P., Jenkins G.M. Time Series Analysis: Forecasting and Control. Journal of Time Series Analysis. 1976. № 31. С. 238-242.

13. Varian H.R. Big Data: New Tricks for Econometrics. Journal of Economic Perspectives. 2014. № 28(2). С. 3-28.

14. Mullainathan S., Spiess J. Machine Learning: An Applied Econometric Approach. Journal of Economic Perspectives. 2017. № 31(2). С. 87-106.

15. Stock J.H., Watson M.W. Forecasting Using Principal Components from a Large Number of Predictors. Journal of the American Statistical Association. 2002. № 97(460). С. 1167-1179.

16. Coulombe P.G., Leroux M., Stevanovic D., Surprenant S. How is Machine Learning Useful for Macroeconomic Forecasting? Journal of Applied Econometrics. 2022. № 37(5). С. 920-964.

17. Medeiros M.C., Vasconcelos G.F., Veiga Á., Zilberman E. Forecasting Inflation in a Data-Rich Environment: The Benefits of Machine Learning Methods. Journal of Business & Economic Statistics. 2021. № 39(1). С. 98-119.

18. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. № 12. С. 2825-2830.

19. McKinney W. Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference. 2010. С. 56-61.

20. Hunter J.D. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering. 2007. № 9(3). С. 90-95.

21. Enhancing inflation nowcasting with online search data: a random forest application for Colombia// Borradores de Economía, Banco de la República (Central Bank of Colombia). – 2025.

22. Machine learning methods in evaluating the impact of economic factors on the consumer price index in Albania. Basha L., Puka L. // CRJ. – 2024.

23. Explainable inflation forecasts by machine learning models// Expert Systems with Applications. – 2022.

24. Тронь Д. В., Потапова Н. А. Огляд методів машинного навчання для прогнозування ВВП на основі макроекономічних індикаторів.

25. Офіційний сайт Світового банку. World Development Indicators. URL: https://data.worldbank.org/indicator

26. Офіційна документація бібліотеки Scikit-learn. URL: https://scikit-learn.org/stable/documentation.html

27. Офіційна документація бібліотеки Pandas. URL: https://pandas.pydata.org/docs/

28. Офіційна документація бібліотеки NumPy. URL: https://numpy.org/doc/

29. Офіційна документація бібліотеки Matplotlib. URL: https://matplotlib.org/stable/contents.html

30. Офіційна документація бібліотеки Streamlit. URL: https://docs.streamlit.io/

31. Офіційна документація бібліотеки Joblib. URL: https://joblib.readthedocs.io/

32. Офіційний сайт Python. URL: https://www.python.org/

33. Офіційний сайт МВФ. World Economic Outlook Database. URL: https://www.imf.org/en/Publications/WEO

34. Офіційний сайт Євростату. URL: https://ec.europa.eu/eurostat

35. Kaggle. Machine Learning for Economic Forecasting. URL: https://www.kaggle.com/

36. Towards Data Science. Random Forest Explained. URL: https://towardsdatascience.com/random-forest-explained

37. Towards Data Science. Time Series Forecasting with Machine Learning. URL: https://towardsdatascience.com/time-series-forecasting-with-machine-learning

38. Real Python. Pandas Tutorial. URL: https://realpython.com/pandas-python-explore-dataset/

39. Real Python. Scikit-learn Tutorial. URL: https://realpython.com/train-test-split-python-data/

40. Офіційний репозиторій Scikit-learn на GitHub. URL: https://github.com/scikit-learn/scikit-learn

41. World Bank Open Data. GDP (constant 2015 US$). URL: https://data.worldbank.org/indicator/NY.GDP.MKTP.KD

42. World Bank Open Data. Inflation, consumer prices (annual %). URL: https://data.worldbank.org/indicator/FP.CPI.TOTL.ZG

# ДОДАТКИ

Додаток А

# ЛІСТИНГ ПРОГРАМНОГО КОДУ

apppy.py:

```
import pandas as pd
import streamlit as st
from modules.data_loader import load_and_merge
from modules.preprocessor import preprocess
from modules.models import train_models
from modules.evaluator import evaluate
from modules.forecaster import recursive_forecast, build_forecast_df
from modules.visualizer import (plot_predictions, plot_feature_importance,
                                plot_comparison, plot_forecast)


# Налаштування сторінки Streamlit
st.set_page_config(page_title="Макроекономічний аналіз", layout="wide")
st.title("Аналіз та прогнозування макроекономічних показників")


# Завантаження файлів
uploaded_files = st.file_uploader(
    "Завантажте CSV файли з World Bank (кожен індикатор — окремий файл)",
    accept_multiple_files=True,
    type=["csv", "xlsx"]
)


if uploaded_files:
    # Злиття завантажених даних
    with st.spinner("Завантаження та злиття даних..."):
```

```
    df_raw = load_and_merge(uploaded_files)


st.subheader("Попередній перегляд даних")


# Вибір країни
countries = df_raw["Country Name"].unique()
selected_country = st.selectbox("Оберіть країну", countries)
df_country = df_raw[df_raw["Country Name"] == selected_country].drop(
    columns=["Country Name", "Country Code"]
)


# Таблиця з першими рядками
st.dataframe(df_country.sort_values("Year").head(10))


st.subheader("Аналіз структури даних")


# Метрики розміру датасету
col1, col2, col3 = st.columns(3)
col1.metric("Кількість рядків", df_country.shape[0])
col2.metric("Кількість стовпців", df_country.shape[1])
col3.metric("Діапазон років", f"{int(df_country['Year'].min())}–{int(df_country['Year'].max())}")


# Перевірка пропусків
missing = df_country.isnull().sum()
if missing.sum() > 0:
    st.write("**Пропуски в даних:**")
    missing_df = pd.DataFrame({
        "Показник": missing[missing > 0].index,
```

```
        "Кількість пропусків": missing[missing > 0].values
    })
    st.dataframe(missing_df, use_container_width=True)
else:
    st.success("Пропуски відсутні")

# Детальна статистика
with st.expander("Детальна статистика показників"):
    numeric_cols = df_country.select_dtypes(include="number").columns
    if len(numeric_cols) > 0:
        st.dataframe(df_country[numeric_cols].describe().round(2))
    else:
        st.info("Немає числових стовпців для статистики")

# Вибір цільової змінної
indicator_cols = [c for c in df_country.columns if c != "Year"]
target = st.selectbox("Оберіть цільовий показник для прогнозування", indicator_cols)

# Глибина прогнозу
n_forecast = st.number_input(
    "Кількість років прогнозу", min_value=1, max_value=20, value=5, step=1
)

# Кнопка запуску навчання та прогнозу
if st.button("Навчити моделі та побудувати прогноз"):

    # Попередження про малу кількість даних
    if len(df_country) < 10:
```

```
        st.warning(
            f"Датасет містить лише {len(df_country)} спостережень — "
            f"точність моделей може бути низькою."
        )

    # Попередня обробка: лаги, видалення NaN
    with st.spinner("Обробка даних..."):
        before_len = len(df_country)
        df_clean, actual_years = preprocess(df_country, target=target, n_lags=3)
        removed = before_len - len(df_clean)
        if removed > 0:
            st.info(f"Видалено {removed} рядків після обробки даних")

        last_known_year = int(actual_years.iloc[-1])

    # Навчання кількох моделей
    with st.spinner("Навчання моделей..."):
        results = train_models(df_clean, target)
        metrics = evaluate(results)

    # Виведення метрик якості
    st.subheader("Метрики точності (на тестовій вибірці)")
    st.dataframe(metrics, use_container_width=True)

    # Графік реальних vs передбачених значень
    st.subheader("Реальні vs прогнозовані значення (тестова вибірка)")

    plot_predictions(results, target, actual_years=actual_years)
```

```
# Важливість ознак
st.subheader("Важливість ознак")
plot_feature_importance(results)

# Порівняння моделей за метриками
st.subheader("Порівняння моделей")
plot_comparison(metrics)

# Рекурсивний прогноз на n_forecast кроків
st.subheader(f"Прогноз на {n_forecast} років вперед")
forecast_rows = {}
for model_name, res in results.items():
    forecasted = recursive_forecast(
        res["model"], df_clean, target, n_steps=int(n_forecast)
    )
    forecast_rows[model_name] = forecasted
    combined = build_forecast_df(df_clean, target, forecasted)
    st.markdown(f"**{model_name}**")
    plot_forecast(combined, target, actual_years=actual_years)

# Таблиця з числовими значеннями прогнозу
st.subheader("Таблиця прогнозних значень")
forecast_table = pd.DataFrame(forecast_rows)
forecast_table.index = [
    str(last_known_year + i + 1) for i in range(int(n_forecast))
]
forecast_table.index.name = "Рік"
```

```
        st.dataframe(forecast_table, use_container_width=True)
```

data_loader.py:

```
import pandas as pd
import io


# Знаходить номер рядка, з якого починаються дані (де є "Country Name")
def _find_header_row(file) -> int:
    content = file.read()
    file.seek(0)  # Повертаємо вказівник на початок файлу
    for i, line in enumerate(io.StringIO(content.decode("utf-8", errors="ignore"))):
        if "Country Name" in line:
            return i
    return 4  # Якщо не знайдено стандартний відступ


# Завантажує кілька CSV файлів World Bank і зливає в одну таблицю
def load_and_merge(files) -> pd.DataFrame:
    dfs = []
    for file in files:
        header_row = _find_header_row(file)
        df = pd.read_csv(file, skiprows=header_row)


        # Видаляємо службовий стовпець Indicator Code якщо він є
        df = df.drop(columns=["Indicator Code"], errors="ignore")


        indicator = df["Indicator Name"].iloc[0]  # Назва показника
```

```
        # Вибір колонок які є роками (цілі числа)
        year_cols = [c for c in df.columns if str(c).strip().isdigit()]

        # Перетворення з широкого формату в довгий
        df_long = df.melt(
            id_vars=["Country Name", "Country Code"],
            value_vars=year_cols,
            var_name="Year",
            value_name=indicator
        )
        df_long["Year"] = df_long["Year"].astype(int)
        dfs.append(df_long)

    # Об’єднання всіх показників по країнах та роках (inner join)
    merged = dfs[0]
    for df in dfs[1:]:
        merged = merged.merge(df, on=["Country Name", "Country Code", "Year"], how="inner")

    return merged
```

evaluator.py:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np, pandas as pd

# Оцінює якість навчених моделей на тестовій вибірці
def evaluate(results: dict) -> pd.DataFrame:
    rows = []
```

```
    for name, res in results.items():
        y_test, y_pred = res["y_test"], res["y_pred"]  # Справжні та передбачені значення
        rows.append({
            "Модель": name,
            "MAE": round(mean_absolute_error(y_test, y_pred), 4),  # Середня абсолютна помилка
            "MSE": round(mean_squared_error(y_test, y_pred), 4),  # Середньоквадратична помилка
            "RMSE": round(np.sqrt(mean_squared_error(y_test, y_pred)), 4),  # Корінь із MSE
            "R²": round(r2_score(y_test, y_pred), 4),  # Коефіцієнт детермінації
            "R² (%)": round(r2_score(y_test, y_pred) * 100, 2)  # R^2 у відсотках

        })
    return pd.DataFrame(rows)  # Таблиця з метриками для всіх моделей
```

forecaster.py:

```
import pandas as pd


# Рекурсивний прогноз на n_steps років вперед.
# Використовує лаги цільового показника.
# Для інших ознак — бере останні відомі значення.
def recursive_forecast(model, df: pd.DataFrame, target: str, n_steps: int) -> list:


    # Знаходимо всі лагові колонки для цільового показника
    lag_cols = [c for c in df.columns if c.startswith(f"{target}_lag")]


    # Перевірка наявності лагів
    if not lag_cols:
        raise ValueError(
```

```
            f"Лагові колонки для '{target}' не знайдені. "
            f"Переконайтесь що preprocess() викликано з target='{target}'."
        )

    n_lags = len(lag_cols)
    last_target_values = list(df[target].values)  # Історія цільового показника
    last_row = df.drop(columns=[target]).iloc[-1].copy()  # Останній рядок без цільової змінної

    forecasted = []

    for _ in range(n_steps):
        row = last_row.copy()

        # Оновлюємо лаги на основі останніх відомих/прогнозованих значень
        for lag in range(1, n_lags + 1):
            col = f"{target}_lag{lag}"
            if col in row.index:
                row[col] = last_target_values[-lag]

        # Прогноз на один крок
        pred = model.predict(pd.DataFrame([row]))[0]
        forecasted.append(pred)
        last_target_values.append(pred)  # Додаємо прогноз для наступних кроків

    return forecasted

# Будує DataFrame з історичними даними та прогнозом для візуалізації
def build_forecast_df(df_original: pd.DataFrame, target: str,
```

```
                forecasted_values: list) -> pd.DataFrame:


    # Історична частина
    history = df_original[[target]].copy()
    history["Тип"] = "Історія"
    history.reset_index(drop=True, inplace=True)

    # Прогнозна частина
    forecast_df = pd.DataFrame({
        target: forecasted_values,
        "Тип": "Прогноз"
    }, index=range(len(history), len(history) + len(forecasted_values)))

    return pd.concat([history, forecast_df])  # Об'єднаний датафрейм
```

## models.py:

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
import pandas as pd
import joblib
import os

# Створення папки для збереження навчених моделей (якщо не існує)
os.makedirs("models", exist_ok=True)


# Часове розбиття
def temporal_train_test_split(df: pd.DataFrame, target: str, test_size: float = 0.2):
```

```
    n = len(df)
    split_idx = int(n * (1 - test_size))

    # Запобігаємо занадто малій тренувальній вибірці
    if split_idx < 5:
        split_idx = max(2, n - 3)

    train = df.iloc[:split_idx]
    test = df.iloc[split_idx:]

    # Відокремлюємо ознаки (X) від цільової змінної (y)
    X_train = train.drop(columns=[target])
    y_train = train[target]
    X_test = test.drop(columns=[target])
    y_test = test[target]

    return X_train, X_test, y_train, y_test

# Навчання моделей (регресія та Random Forest)
def train_models(df: pd.DataFrame, target: str) -> dict:
    X_train, X_test, y_train, y_test = temporal_train_test_split(df, target)

    models = {
        "Регресія": LinearRegression(),
        "Random Forest": RandomForestRegressor(n_estimators=100, random_state=42)
    }

    results = {}
```

```
    for name, model in models.items():
        model.fit(X_train, y_train)          # Навчання
        y_pred = model.predict(X_test)       # Прогноз на тесті
        results[name] = {
            "model": model,
            "y_test": y_test,
            "y_pred": y_pred,
            "features": X_train.columns,
            "X_train": X_train,
            "X_test": X_test,
        }
        # Збереження моделі у файл
        joblib.dump(model, f"models/{name.replace(' ', '_')}.pkl")

    return results
```

## preprocessor.py:

```
import pandas as pd


# Додає лагові колонки для цільового показника
def add_lags(df: pd.DataFrame, target: str, n_lags: int = 3) -> pd.DataFrame:
    df = df.copy()
    for lag in range(1, n_lags + 1):
        df[f"{target}_lag{lag}"] = df[target].shift(lag)
    return df


# Попередня обробка: числові колонки, сортування, видалення NaN, лаги
```

```
def preprocess(df: pd.DataFrame, target: str = None, n_lags: int = 3):
    # Зберігаємо роки окремо
    if "Year" in df.columns:
        years = df["Year"].copy()
    else:
        years = pd.Series(range(len(df)))

    # Вибір числових колонок
    numeric_cols = df.select_dtypes(include="number").columns
    df = df[numeric_cols].copy()

    # Сортуємо по рокові скидаємо індекс
    if "Year" in df.columns:
        df = df.sort_values("Year").reset_index(drop=True)
        years = years.sort_values().reset_index(drop=True)
        df = df.drop(columns=["Year"])

    # Видаляємо рядки де цільовий показник відсутній
    if target and target in df.columns:
        mask = df[target].notna()
        df = df[mask].reset_index(drop=True)
        years = years[mask].reset_index(drop=True)

    # Заповнення пропусків медіаною
    df.fillna(df.median(), inplace=True)

    # Додаємо лагові змінні (після цього перші n_lags рядків матимуть NaN)
    if target and target in df.columns:
```

```
        df = add_lags(df, target, n_lags)
        df.reset_index(drop=True, inplace=True)

    # Видаляємо рядки з будь-якими NaN (перші n_lags після створення лагів)
    valid_mask = df.notna().all(axis=1).values
    df = df[valid_mask].reset_index(drop=True)
    years = years[valid_mask].reset_index(drop=True)

    return df, years
```

visualizer.py:

```
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use("Agg")  # Використання backend без GUI для Streamlit
import streamlit as st
import pandas as pd

# Графік реальних vs прогнозованих значень на тестовій вибірці
def plot_predictions(results: dict, target: str, actual_years: pd.Series = None):
    n = len(results)
    fig, axes = plt.subplots(1, n, figsize=(7 * n, 5))
    if n == 1:
        axes = [axes]

    for ax, (name, res) in zip(axes, results.items()):
        y_test = res["y_test"]
        y_pred = pd.Series(res["y_pred"], index=y_test.index)
```

```
        # Визначення значень по осі X
        if actual_years is not None:
            x_vals = actual_years.iloc[y_test.index].values
        else:
            x_vals = range(len(y_test))

        ax.plot(x_vals, y_test.values, label="Реальні", color="#1f77b4", linewidth=1.5)
        ax.plot(x_vals, y_pred.values, label="Прогноз", color="#ff7f0e",
                linewidth=1.5, linestyle="--")
        ax.set_title(name)
        ax.set_xlabel("Рік")
        ax.set_ylabel(target)
        ax.legend()
        ax.grid(True, alpha=0.3)

    plt.tight_layout()
    st.pyplot(fig)  # Відображення у Streamlit
    plt.close()


# Важливість ознак для моделі Random Forest
def plot_feature_importance(results: dict):
    rf_result = results.get("Random Forest")
    if not rf_result:
        return

    model = rf_result["model"]
    features = rf_result["features"]
```

```
    importances = pd.Series(model.feature_importances_, index=features)
    importances = importances.sort_values(ascending=True)

    # Горизонтальна стовпчикова діаграма
    fig, ax = plt.subplots(figsize=(7, max(4, len(features) * 0.5)))
    importances.plot(kind="barh", ax=ax, color="#1f77b4")
    ax.set_title("Важливість ознак (Random Forest)")
    ax.set_xlabel("Importance")
    ax.grid(True, alpha=0.3, axis="x")
    plt.tight_layout()
    st.pyplot(fig)
    plt.close()

# Порівняння моделей за метриками RMSE та R^2
def plot_comparison(metrics_df):
    fig, axes = plt.subplots(1, 2, figsize=(10, 4))

    for ax, metric in zip(axes, ["RMSE", "R²"]):
        ax.bar(metrics_df["Модель"], metrics_df[metric],
               color=["#1f77b4", "#ff7f0e"])
        ax.set_title(f"Порівняння: {metric}")
        ax.set_ylabel(metric)
        ax.grid(True, alpha=0.3, axis="y")

    plt.tight_layout()
    st.pyplot(fig)
    plt.close()
```

```
# Графік історії + прогнозу (з вертикальною межею)
def plot_forecast(combined_df: pd.DataFrame, target: str, actual_years: pd.Series):
    fig, ax = plt.subplots(figsize=(12, 5))

    history = combined_df[combined_df["Тип"] == "Історія"]
    forecast = combined_df[combined_df["Тип"] == "Прогноз"]

    history_years = actual_years.values
    last_year = int(actual_years.iloc[-1])
    forecast_years = list(range(last_year + 1, last_year + 1 + len(forecast)))

    ax.plot(history_years, history[target].values,
            label="Історія", color="#1f77b4", linewidth=2)
    ax.plot(forecast_years, forecast[target].values,
            label="Прогноз", color="#ff7f0e",
            linewidth=2, linestyle="--", marker="o", markersize=5)

    # Вертикальна лінія в останній відомий рік
    ax.axvline(x=last_year, color="gray", linestyle=":",
               linewidth=1.5, label="Межа прогнозу")
    ax.set_title(f"Історія та прогноз: {target}")
    ax.set_xlabel("Рік")
    ax.set_ylabel(target)
    ax.legend()
    ax.grid(True, alpha=0.3)
    plt.tight_layout()
    st.pyplot(fig)
    plt.close()
```