ЛАБОРАТОРНА РОБОТА 6

Тема: Технології інтелектуального аналізу даних (Data Mining)

Data Mining є певним етапом процесу KDD (Knowledge Discovery in Databases), на якому застосовуються алгоритмічні методи для автоматичного виявлення прихованих закономірностей, залежностей, структур і тенденцій у великих наборах даних. Головною метою Data Mining є перетворення даних у знання, що придатні для прийняття рішень, прогнозування та пояснення процесів. Основними задачами Data Mining є:

- класифікація;
- регресія;
- кластеризація;
- пошук асоціативних правил;
- виявлення аномалій;
- прогнозування.

Класифікація (Classification) — це одна з основних технік інтелектуального аналізу даних (Data Mining), де алгоритм розподіляє об'єкти за заздалегідь визначеними категоріями на основі їхніх характеристик. Наприклад, в управлінні ризиками та фінансами класифікація допомагає мінімізувати збитки та автоматизувати прийняття рішень. Так, при виконанні кредитного скорингу позичальники класифікуються на «надійних» та «ризикованих» і на основі цього банк автоматично схвалює або відхиляє кредит. Також класифікація широко використовується щодо виявлення шахрайства (Fraud Detection) — під час аналізу транзакцій, система має виконати класифікацію операції як «легітимної» або «підозрілої» (шахрайської). Основними методами класифікації є: дерево рішень, наївний байєс, k-NN, нейронні мережі та SVM (метод опорних векторів).

Регресія (Regression) або регресійний аналіз — це статистичний метод, який дозволяє визначити, як зміна значення однієї або кількох незалежних змінних впливає на залежну зміну (результат). На відміну від класифікації, де виконується пошук відповідності певній категорії, регресія передбачає конкретне числове значення. Наприклад, якщо класифікація дозволяє визначати категорію надійності клієнта і відповісти «чи відмовиться клієнт від послуг компанії», то регресія повинна відповісти на терміни цього процесу або певні кількісні характеристики — через скільки днів або скільки грошей буде витрачено клієнтом перед тим, як він відмовиться від послуг компанії. З безлічі прикладів застосування регресії найбільш класичним є її використання у плануванні бюджетів: розрахунок майбутнього обсягу продажів на основі витрат на маркетинг, сезонності, середнього чека та цін конкурентів; визначення того, як зміна ціни на певний відсоток вплине на кількість проданих одиниць товару, тощо. Найчастіше використовуються такі типи регресії: лінійна (linear), коли є пряма залежність між двома змінними; множинна (multiple), якщо на результат впливають два або більше факторів; поліноміальна, коли залежність не є прямою лінією, має вигини.

Кластеризація — це групування об'єктів у кластери без наперед заданих класів, або іншими словами навчання без учителя (unsupervised learning), де алгоритм самостійно групує об'єкти на основі їхньої схожості. Головна відмінність від класифікації полягає в тому, що немає заздалегідь визначених міток або категорій. Часто кластеризація застосовується для аналізу асортименту та кошиків, тому що допомагає зрозуміти, які товари «живуть» разом або мають схожий життєвий цикл. Наприклад, формування рекомендацій системи: якщо клієнт купив товар з кластера А, то йому можна запропонувати інший товар з цього ж кластера, або виявлення взаємозамінних товарів: якщо в одному кластері опинилися різні бренди молока, магазин розуміє, що відсутність одного бренду змусить покупця просто взяти інший з цієї ж групи. Також кластеризація застосовується для виявлення аномалій (Outlier Detection), тобто для пошуку того, що «не вписується» в норму. Наприклад, при виконанні фінансового моніторингу, якщо операція клієнта потрапляє в ізольовану зону — не схожа на жодну типову модель поведінки цього користувача, вона маркується як підозріла.

Популярними алгоритмами кластеризації для бізнесу є: K-Means (K-середніх), який вимагає заздалегідь вказати кількість груп *k*; ієрархічна кластеризація, яка будує структуру, схожу на дерево із вкладеними групами; DBSCAN (Density-Based Spatial Clustering of Applications with Noise), який автоматично визначає кількість кластерів, групує щільно розташовані точки та ефективно знаходить аномалії (шум), що робить його стійким до викидів.

Пошук асоціативних правил (Association Rule Mining) — це техніка пошуку прихованих взаємозв’язків між об'єктами у великих масивах даних. У бізнесі це часто називають аналізом ринкового кошика (Market Basket Analysis), оскільки класичний приклад — виявлення товарів, які клієнти зазвичай купують разом. Наприклад, якщо правило «Принтер» → «Картридж» має високу достовірність, магазин може зробити велику знижку на принтер, тому що прибуток буде отримано з подальшого продажу витратних матеріалів. Ще одним яскравим прикладом є розміщення товарів на полицях: товари із сильним зв'язком ставлять або максимально поруч для зручності, або навпаки в протилежних кінцях залу, щоб клієнт пройшов через весь магазин, здійснюючи імпульсивні покупки.

Для визначення корисності правила, аналітики використовують такі ключові метрики, як: підтримка (support) — частота появи певного набору товарів у всіх транзакціях; достовірність (confidence) - ймовірність, що при покупці товару А буде куплений товар Б; покращення (lift) - збільшення ймовірності покупки товару Б після покупки товару А у порівнянні із звичайною частотою покупки Б. Одним із найвідоміших алгоритмів є алгоритм пріоритету (Apriori Algorithm), який працює за принципом: якщо набір товарів популярний, то і всі його підмножини також популярні. Це дозволяє швидко обробляти мільйони чеків.

Процес пошуку точок у даних, які суттєво відхиляються від звичайної поведінки більшості об'єктів називається виявленням аномалій (Anomaly Detection). У бізнесі аномалія — це або тривожний сигнал (шахрайство, збій), або прихована можливість (несподіваний сплеск попиту). У такій сфері, як фінанси та безпека платежів швидкість виявлення аномалії напряму конвертується в збережені ресурси. Наприклад, система фрод-моніторингу (Banking Fraud) аналізує типові витрати клієнта і визначає, що він купує продукти в Києві, а через годину з картки намагаються зняти велику суму в банкоматі Сінгапура — це аномалія, яка призведе до блокування транзакції. Виявлення аномалій є важливим при визначені факту відмивання коштів (Anti-Money Laundering): велика кількість транзакцій із сумами трохи меншими за поріг обов'язкового фінмоніторингу, але відбуваються підозріло часто між пов'язаними рахунками.

Для пошуку аномалій використовують різні підходи залежно від наявності розмічених даних. Достатньо потужними та ефективними в бізнес-аналізі даних виступають статистичні методи: правило трьох сигм (3-sigma rule) та міжквартильний розмах (IQR). Цими методами в якості аномалій визначають усе, що виходить за певні межі на діаграмах розподілу. Методи ізоляційний ліс (Isolation Forest) та коефіцієнт локального відхилення (Local Outlier Factor) відносяться до категорії методів навчання без учителя (Unsupervised). Метод ізоляційний ліс спеціально розроблено для виявлення аномалій шляхом ізоляції спостережень і його алгоритм працює на двох основних припущеннях, що аномалій мало, і вони мають інші значення атрибутів, ніж звичайні випадки. Створюється ансамбль «ізоляційних дерев» (iTrees) шляхом випадкового вибору ознаки, а потім випадкового вибору значення розділення між мінімумом і максимумом ознаки. Оскільки аномалії є окремими та далекими від щільних кластерів, вони відокремлюються від решти даних меншою кількістю розділень, тому коротші довжини шляху від кореня до кінцевого вузла вказують на високу ймовірність того, що це аномалія. Метод коефіцієнту локального відхилення базується на концепції локальної щільності, де локальність визначається *k* найближчими сусідами, відстань до яких використовується для оцінки щільності. Об'єкт вважається аномалією, якщо густина інших об’єктів навколо нього значно менша, ніж густина навколо його сусідів. Густина визначається як обернена величина дистанції до сусідів, тому цей метод враховує не тільки оточення певної точки, а також щільність кластеру — густина, за якою визначаються аномалії у різних кластерів, може суттєво відрізнятись. Ефективним методом виявлення аномалій є автокодувальник (Autoencoder) — спеціальна нейронна мережа, яка навчається відтворювати вхідні дані на вихідні через стискання лише типової структури

нормальних даних. Нейронна мережа навчається кодуванню і відновленню тільки нормальних даних, тому появлення у вхідних даних аномалій не вписуються в цю структуру і не відновлюються. Autoencoder навчається стискати нормальні дані у компактне латентне представлення, наприклад, 100 варіацій значень після кодування зменшується до 10 значень, і після декодування відновлюється знову до 100 значень. Таке відновлення не дає 100% точність значень, але різниця між відповідними вхідними та вихідними даними знаходиться у певних межах. Аномальні дані не можуть бути коректно представлені у цьому просторі кодувальних значень, тому після декодування виникає велика розбіжність відновлення, яка і використовується для виявлення цих аномалій.

Часто повторювані впорядковані послідовності подій у часі визначають таким методом Data Mining, як пошук послідовних закономірностей (Sequential Pattern Mining). Шаблон для пошуку та визначення виглядає наступним чином: «Якщо відбулася подія А, то через певний час з високою ймовірністю відбудеться подія Б». Головна відмінність від асоціативних правил полягає в тому, що для асоціативних правил порядок неважливий, важливий тільки зв’язок між подією А та В, що вони зв’язані, вони разом, а для послідовних закономірностей порядок є основою методу — подія А передує подію В. Суть методу полягає в пошуку закономірності вигляду «A → B → C» (спочатку відбувається подія А, потім В, потім С) і така послідовність часто повторюється. Наприклад, для клієнтів є типовою покупка: смартфон (день 1) → чохол (наступний день) → навушники (7-мий день) → нова модель смартфона (через 2 роки). Також можна навести приклад застосування методу для прогнозу інвестиційної поведінки, а саме як трейдери реагують на ринкові події. Наприклад, «різке падіння індексу» → «продаж технологічних акцій» → «купівля золота» протягом одного торгового дня.

У цьому методи важливою метрикою є support. Так, якщо деякий шаблон <A B> зустрічається у 3 з 4 знайдених послідовних закономірностей, тоді support = 0.75.

Популярними алгоритмами цього методу є: AprioriAll — поступова побудува довших послідовностей з коротших; GSP (Generalized Sequential Patterns) — покращення AprioriAll через встановлення додаткових параметрів з обмеження часу, max/min інтервалів, тощо; SPADE (Sequential Pattern Discovery using Equivalence classes) використовує вертикальне представлення даних, що дозволяє зменшити кількість сканувань баз даних та пришвидшити отримання результату; PrefixSpan оснований на алгоритмах рекурсивного проєктування бази на основі префіксів, що робить його самим швидким з наведених вище алгоритмів.

Зменшення розмірності (Dimensionality Reduction) — це процес спрощення складних наборів даних шляхом зменшення кількості вхідних змінних (ознак), зберігаючи при цьому найважливішу інформацію. У бізнесі це критично важливо, коли даних занадто багато, вони дублюють один одного або створюють «шум», який заважає моделям працювати точно. Так, у фінансовому аналізі багато показників корелюють між собою, наприклад, різні коефіцієнти ліквідності або прибутковості часто рухаються в одному напрямку. Тому аналітик не вводить до моделі прогнозування банкрутства 30 різних фінансових коефіцієнтів, а використовує зменшення розмірності, щоб створити 5 інтегральних показників, які описують загальний стан компанії без надмірності. Інший приклад - обробка природної мови у відгуках. Кожне унікальне слово у відгуках клієнтів можна вважати окремою ознакою і отримати велику базу з безліччю колонок. Використання LSA (Latent Semantic Analysis) для зменшення розмірності текстових даних дозволяє згрупувати тисячі слів у декілька основних тем, наприклад, «якість обслуговування», «ціна», «швидкість доставки».

Зменшення розмірності також часто використовується як етап підготовки і очищення даних для Machine Learning, з метою уникнення «прокляття розмірності» (Curse of Dimensionality). Наприклад, якщо модель прогнозує ціну акції на основі 100 технічних індикаторів, багато з яких показують майже одне й те саме, зменшення розмірності допоможе прибрати мультиколінеарність і зробить модель швидшою, стабільнішою та менш схильною до перенавчання (overfitting).

Основними методами зменшення розмірності є: метод головних компонентів (PCA) - лінійна комбінація ознак, що зберігає максимум дисперсії і використовується для швидкого стиснення та очищення від шуму; t-SNE / UMAP - нелінійне стиснення, що зберігає локальні

зв'язки і дозволяє надати красиву візуалізацію складних структур; лінійний дискримінаційний аналіз (LDA), головною метою якого є спроектувати дані з багатовимірного простору на простір меншої розмірності (лінію, площину) так, щоб відстань між середніми значеннями класів була максимальною, а варіативність всередині кожного класу була мінімальною.

Приклад реалізації методу регресії та розрахунок основних метрик якості регресії

```
import numpy as np
import pandas as pd
import yfinance as yf
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, mean_absolute_percentage_error

# ====================== 1. ЗАВАНТАЖЕННЯ ДАНИХ ======================
print("1. Завантаження даних з Yahoo Finance...")
ticker = "AAPL"
data = yf.download(ticker, start="2020-01-01", end="2026-04-12", progress=False)
print(f"   Завантажено {len(data)} днів даних")

# ====================== 2. ПІДГОТОВКА ОЗНАК ======================
print("\n2. Підготовка ознак для регресії...")

# Лагові змінні (ціна за попередні дні)
data['Lag1'] = data['Close'].shift(1)
data['Lag2'] = data['Close'].shift(2)
data['Lag3'] = data['Close'].shift(3)

# Технічні індикатори
data['MA5'] = data['Close'].rolling(window=5).mean()   # 5-денне ковзне середнє
data['MA10'] = data['Close'].rolling(window=10).mean() # 10-денне ковзне середнє

# Волатильність (розмах цін)
data['Range'] = data['High'] - data['Low']

# Об'єм торгів (логарифмічне перетворення для нормалізації)
data['Log_Volume'] = np.log(data['Volume'] + 1)

# Видаляємо рядки з пропусками (перші 10 днів через зсуви та ковзні середні)
data.dropna(inplace=True)
print(f"   Після видалення пропусків: {len(data)} спостережень")

# Визначаємо ознаки (X) та цільову змінну (y)
feature_columns = ['Lag1', 'Lag2', 'Lag3', 'MA5', 'MA10', 'Range', 'Log_Volume']
X = data[feature_columns]
y = data['Close']

print(f"   Кількість ознак: {X.shape[1]}")
print(f"   Ознаки: {', '.join(feature_columns)}")

# ====================== 3. РОЗДІЛЕННЯ НА НАВЧАЛЬНУ ТА ТЕСТОВУ ВИБІРКИ ======================
print("\n3. Розділення даних (80% навчання, 20% тест)...")
# Важливо: shuffle=False для збереження часової структури
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, shuffle=False, random_state=42
)
print(f"   Навчальна вибірка: {len(X_train)} спостережень")
print(f"   Тестова вибірка: {len(X_test)} спостережень")

# ====================== 4. НАВЧАННЯ МОДЕЛІ ======================
print("\n4. Навчання моделі лінійної регресії...")
model = LinearRegression()
model.fit(X_train, y_train)
print("   Модель успішно навчена")

# ====================== 5. ПРОГНОЗУВАННЯ ======================
print("\n5. Прогнозування на тестовій вибірці...")
y_pred = model.predict(X_test)

# Переконуємось, що y_test та y_pred є 1D масивами
if hasattr(y_test, 'values'):
    y_test = y_test.values.flatten()
else:
    y_test = np.array(y_test).flatten()

y_pred = y_pred.flatten() if hasattr(y_pred, 'flatten') else np.array(y_pred).flatten()

# ====================== 6. ОСНОВНІ МЕТРИКИ ЯКОСТІ РЕГРЕСІЇ ======================
print("\n" + "="*60)
print("            ОСНОВНІ МЕТРИКИ ЯКОСТІ РЕГРЕСІЇ")
print("="*60)

# R-squared (коефіцієнт детермінації)
```

```
r2 = r2_score(y_test, y_pred)
print(f"\n📊 R² (R-squared): {r2:.4f}")
print(f"   → Пояснює {r2*100:.2f}% варіації цільової змінної")

# Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print(f"\n📊 MSE (Mean Squared Error): {mse:.2f}")
print(f"   → Середній квадрат помилки (чутливий до викидів)")

# Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print(f"\n📊 RMSE (Root Mean Squared Error): {rmse:.2f}")
print(f"   → Типова помилка прогнозу: ${rmse:.2f}")

# Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, y_pred)
print(f"\n📊 MAE (Mean Absolute Error): {mae:.2f}")
print(f"   → Середня абсолютна помилка: ${mae:.2f}")

# Mean Absolute Percentage Error (MAPE)
mape = mean_absolute_percentage_error(y_test, y_pred) * 100
print(f"\n📊 MAPE (Mean Absolute Percentage Error): {mape:.2f}%")
print(f"   → Типова відносна помилка: {mape:.2f}%")

# Adjusted R² (штрафує за кількість ознак)
n = len(y_test)  # кількість спостережень у тестовій вибірці
p = X.shape[1]   # кількість ознак
adjusted_r2 = 1 - (1 - r2) * (n - 1) / (n - p - 1)
print(f"\n📊 Adjusted R²: {adjusted_r2:.4f}")
print(f"   → З урахуванням {p} ознак та {n} спостережень")
if adjusted_r2 < r2 - 0.1:
    print(f"   ⚠️  Значне падіння Adjusted R² вказує на наявність зайвих ознак")

# ====================== 7. ДОДАТКОВІ МЕТРИКИ ДЛЯ АНАЛІЗУ ======================
print("\n" + "="*60)
print("            ДОДАТКОВИЙ АНАЛІЗ ПОМИЛОК")
print("="*60)

# Аналіз залишків (residuals)
residuals = y_test - y_pred

print(f"\n📊 Аналіз залишків:")
print(f"   Середнє залишків: {np.mean(residuals):.4f} (має бути близько 0)")
print(f"   Стандартне відхилення залишків: {np.std(residuals):.4f}")
print(f"   Медіана залишків: {np.median(residuals):.4f}")

# Квартилі помилок
abs_residuals = np.abs(residuals)
percentiles = np.percentile(abs_residuals, [25, 50, 75, 90, 95])
print(f"\n📊 Квантилі абсолютних помилок:")
print(f"   25% помилок менші за: ${percentiles[0]:.2f}")
print(f"   50% помилок менші за: ${percentiles[1]:.2f}")
print(f"   75% помилок менші за: ${percentiles[2]:.2f}")
print(f"   90% помилок менші за: ${percentiles[3]:.2f}")
print(f"   95% помилок менші за: ${percentiles[4]:.2f}")

# Співвідношення RMSE/MAE
rmse_mae_ratio = rmse / mae
print(f"\n📊 Співвідношення RMSE/MAE: {rmse_mae_ratio:.3f}")
if rmse_mae_ratio < 1.2:
    print("   → Помилки однорідні, немає великих викидів")
elif rmse_mae_ratio < 1.5:
    print("   → Помірна чутливість до викидів")
else:
    print("   → Є значні викиди в даних")

# Додатково: коефіцієнт кореляції між фактичними та передбаченими
correlation = np.corrcoef(y_test, y_pred)[0, 1]
print(f"\n📊 Кореляція фактичних та передбачених значень: {correlation:.4f}")

# ====================== 8. ВІЗУАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ======================
print("\n8. Створення візуалізацій...")

# Пригнічуємо попередження pandas
import warnings
warnings.filterwarnings('ignore', category=FutureWarning)

# Отримуємо індекси для візуалізації
if hasattr(y_test, 'index'):
    test_index = y_test.index
else:
    test_index = range(len(y_test))

# Створюємо фігуру з 4 графіками
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# Графік 1: Фактичні vs передбачені значення (часовий ряд)
axes[0, 0].plot(test_index, y_test, label='Фактичні', color='blue', linewidth=2)
axes[0, 0].plot(test_index, y_pred, label='Передбачені', color='red', linestyle='--', alpha=0.7)
axes[0, 0].set_title(f'{ticker}: Фактичні vs Передбачені ціни', fontsize=12, fontweight='bold')
axes[0, 0].set_xlabel('Дата' if hasattr(y_test, 'index') else 'Спостереження')
```

```
        axes[0, 0].set_ylabel('Ціна (USD)')
        axes[0, 0].legend()
        axes[0, 0].grid(True, alpha=0.3)

        # Графік 2: Scatter plot (фактичні vs передбачені)
        axes[0, 1].scatter(y_test, y_pred, alpha=0.5, s=20)
        axes[0, 1].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
                        'r--', linewidth=2, label='Ідеальна лінія')
        axes[0, 1].set_xlabel('Фактичні ціни (USD)')
        axes[0, 1].set_ylabel('Передбачені ціни (USD)')
        axes[0, 1].set_title(f'Діаграма розсіювання (R² = {r2:.4f})', fontsize=12, fontweight='bold')
        axes[0, 1].legend()
        axes[0, 1].grid(True, alpha=0.3)

        # Графік 3: Залишки (помилки) в часі
        axes[1, 0].plot(test_index, residuals, color='green', alpha=0.7)
        axes[1, 0].axhline(y=0, color='red', linestyle='-', linewidth=2)
        # Використовуємо where для fill_between, щоб уникнути помилки з розмірністю
        axes[1, 0].fill_between(test_index, residuals, 0, where=np.array(residuals) >= 0,
                                alpha=0.2, color='green', interpolate=True)
        axes[1, 0].fill_between(test_index, residuals, 0, where=np.array(residuals) < 0,
                                alpha=0.2, color='red', interpolate=True)
        axes[1, 0].set_title('Графік залишків (помилок прогнозу)', fontsize=12, fontweight='bold')
        axes[1, 0].set_xlabel('Дата' if hasattr(y_test, 'index') else 'Спостереження')
        axes[1, 0].set_ylabel('Помилка (USD)')
        axes[1, 0].grid(True, alpha=0.3)

        # Графік 4: Гістограма залишків
        axes[1, 1].hist(residuals, bins=30, edgecolor='black', alpha=0.7, color='purple')
        axes[1, 1].axvline(x=0, color='red', linestyle='-', linewidth=2)
        axes[1, 1].set_xlabel('Помилка (USD)')
        axes[1, 1].set_ylabel('Частота')
        axes[1, 1].set_title(f'Розподіл помилок (MAE = ${mae:.2f})', fontsize=12, fontweight='bold')
        axes[1, 1].grid(True, alpha=0.3)

        plt.tight_layout()
        plt.show()

        # ====================== 9. ВИСНОВКИ ТА ІНТЕРПРЕТАЦІЯ ======================
        print("\n" + "="*60)
        print("                    ВИСНОВКИ ТА ІНТЕРПРЕТАЦІЯ")
        print("="*60)

        print(f"\n✅ Модель пояснює {r2*100:.2f}% варіації ціни {ticker}")
        print(f"✅ Типова помилка прогнозу: ${rmse:.2f} ({mape:.2f}%)")

        if r2 > 0.7:
            print("✅ Висока пояснювальна здатність моделі")
        elif r2 > 0.4:
            print("⚠  Середня пояснювальна здатність моделі")
        else:
            print("❌ Низька пояснювальна здатність моделі")

        if adjusted_r2 < r2 - 0.1:
            print(f"⚠  Увага: Adjusted R² значно нижчий за R²")
            print(f"   → Варто видалити неінформативні ознаки або збільшити вибірку")

        if mape < 5:
            print(f"✅ Висока точність прогнозів (похибка {mape:.2f}%)")
        elif mape < 10:
            print(f"⚠  Прийнятна точність прогнозів (похибка {mape:.2f}%)")
        else:
            print(f"❌ Низька точність прогнозів (похибка {mape:.2f}%)")

        # Порівняння з базовою моделлю (прогноз за середнім)
        baseline_pred = np.full_like(y_test, np.mean(y_test))
        baseline_mape = mean_absolute_percentage_error(y_test, baseline_pred) * 100
        improvement = (baseline_mape - mape) / baseline_mape * 100
        print(f"\n📈 Порівняння з базовою моделлю (прогноз середнього):")
        print(f"   MAPE базової моделі: {baseline_mape:.2f}%")
        print(f"   Покращення: {improvement:.1f}%")

        # ====================== 10. ПІДСУМКОВА ТАБЛИЦЯ МЕТРИК ======================
        metrics_df = pd.DataFrame({
            'Метрика': ['R²', 'Adjusted R²', 'MSE', 'RMSE', 'MAE', 'MAPE (%)', 'Кореляція'],
              'Значення': [f'{r2:.4f}', f'{adjusted_r2:.4f}', f'{mse:.2f}', f'{rmse:.2f}', f'{mae:.2f}',
f'{mape:.2f}', f'{correlation:.4f}'],
            'Інтерпретація': [
                'Частка поясненої варіації',
                'R² зі штрафом за кількість ознак',
                'Середній квадрат помилки',
                'Типова помилка в оригінальних одиницях',
                'Середня абсолютна помилка',
                'Середня відносна помилка',
                'Кореляція між фактичними та передбаченими'
            ]
        })

        print("\n" + "="*60)
        print("                    ПІДСУМКОВА ТАБЛИЦЯ МЕТРИК")
        print("="*60)
```

```
print(metrics_df.to_string(index=False))

# ====================== 11. КОЕФІЦІЄНТИ МОДЕЛІ (ВИПРАВЛЕНО) ======================
print("\n" + "="*60)
print("                  КОЕФІЦІЄНТИ МОДЕЛІ")
print("="*60)

# Отримуємо коефіцієнти у правильному форматі
coefficients = model.coef_
if coefficients.ndim > 1:
    coefficients = coefficients.flatten()

# Створюємо DataFrame з коефіцієнтами
coef_df = pd.DataFrame({
    'Ознака': feature_columns,
    'Коефіцієнт': coefficients,
    'Вплив на ціну': ['позитивний' if c > 0 else 'негативний' for c in coefficients]
})
print(coef_df.to_string(index=False))

# Виправлено: перевіряємо тип intercept та конвертуємо у скаляр
intercept = model.intercept_
if isinstance(intercept, (np.ndarray, list)):
    intercept = intercept[0] if len(intercept) > 0 else 0
print(f"\n📊 Інтерсепт (вільний член): {intercept:.4f}")

# Додатково: сортуємо за важливістю (за абсолютним значенням коефіцієнта)
print("\n" + "="*60)
print("                  ВАЖЛИВІСТЬ ОЗНАК")
print("="*60)
coef_df['Абсолютне значення'] = np.abs(coefficients)
coef_df_sorted = coef_df.sort_values('Абсолютне значення', ascending=False)
print(coef_df_sorted[['Ознака', 'Коефіцієнт', 'Вплив на ціну']].to_string(index=False))

# Аналіз важливості
print(f"\n📊 Найважливіша ознака: {coef_df_sorted.iloc[0]['Ознака']} "
      f"(коефіцієнт: {coef_df_sorted.iloc[0]['Коефіцієнт']:.4f})")
print(f"📊 Найменш важлива ознака: {coef_df_sorted.iloc[-1]['Ознака']} "
      f"(коефіцієнт: {coef_df_sorted.iloc[-1]['Коефіцієнт']:.4f})")

# ====================== 12. ДОДАТКОВА СТАТИСТИКА ======================
print("\n" + "="*60)
print("                  ДОДАТКОВА СТАТИСТИКА")
print("="*60)

# Середнє значення цільової змінної
mean_y = np.mean(y_test)
print(f"\n📊 Середня ціна {ticker} в тестовій вибірці: ${mean_y:.2f}")
print(f"📊 RMSE у відсотках від середньої ціни: {(rmse/mean_y)*100:.2f}%")
print(f"📊 MAE у відсотках від середньої ціни: {(mae/mean_y)*100:.2f}%")

# Діапазон цін
print(f"\n📊 Діапазон цін в тестовій вибірці: ${y_test.min():.2f} - ${y_test.max():.2f}")
print(f"📊 Розмах цін: ${y_test.max() - y_test.min():.2f}")

# Якщо R² високий, але Adjusted R² низький
if r2 > 0.8 and adjusted_r2 < 0.6:
    print("\n⚠️  ПОПЕРЕДЖЕННЯ: Модель може бути перенавченою (overfitting)!")
    print("   Рекомендації:")
    print("   1. Зменшіть кількість ознак")
    print("   2. Використайте регуляризацію (Ridge, Lasso)")
    print("   3. Збільште розмір навчальної вибірки")
    print("   4. Видаліть мультиколінеарні ознаки")
```

Практичні завдання

1. Вивчіть теоретичний матеріал лабораторної роботи.
2. Відтворіть наведений приклад для тікерів свого варіанту. Отримайте результати для кожного тікеру свого варіанту.
3. В коді прикладу використовуються Lag1, Lag2, Lag3 та MA5. Ці ознаки сильно корелюють між собою, що може негативно впливати на стабільність коефіцієнтів лінійної регресії. Написати програму, що будує кореляційну матрицю (теплову карту) для всіх feature_columns. Автоматично визначати, які ознаки мають критичну кореляцію, що перевищує 0.9.
4. Підготуйте звіт з виконання практичних завдань.