Міністерство освіти і науки України
Національний університет «Запорізька політехніка»

# МЕТОДИЧНІ ВКАЗІВКИ
**до виконання лабораторної роботи**
**з дисципліни «Методи цифрового аналізу даних»**
**на тему «Розробка моделі для передбачення»**
для студентів спеціальності
F7 (123) Комп'ютерна інженерія
всіх форм навчання

2025

Методичні вказівки до виконання лабораторної роботи з дисципліни «Методи цифрового аналізу даних» на тему «Розробка моделі для передбачення» для студентів спеціальності F7 (123) Комп'ютерна інженерія всіх форм навчання / Укл. Т.В. Голуб, І.Я. Зеленьова. - Запоріжжя: НУ «Запорізька політехніка», 2025.- 20 с.

Укладачі: Т.В. Голуб, доцент, к.т.н.
І.Я.Зеленьова, доцент, к.т.н.

Рецензент: С.С. Грушко, доцент, к.т.н.

Відповідальний
за випуск: Т.В. Голуб, доцент, к.т.н.

Затверджено
на засіданні кафедри КСМ
Протокол № 2 від 29.08.2025

Рекомендовано до видання
на засіданні НМК факультету КНТ
Протокол 2 від 10.09.2025

# ЗМІСТ

# ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та накопичення великих обсягів даних особливої актуальності набувають методи цифрового аналізу даних. Одним із фундаментальних інструментів аналізу та моделювання залежностей між змінними є регресійний аналіз. Він дозволяє будувати математичні моделі, які описують зв'язок між незалежними (вхідними) та залежною (вихідною) змінними, а також здійснювати передбачення значень вихідної змінної на основі нових вхідних даних.

Дана лабораторна робота має на меті формування практичних навичок з побудови регресійних моделей, аналізу їхньої якості та інтерпретації результатів. У ході виконання роботи студенти ознайомляться з основними етапами регресійного аналізу: вибором типу моделі, навчанням моделі, оцінкою точності прогнозування та візуалізацією результатів.

Здобуті знання та навички є необхідними для розв'язання прикладних задач у різних галузях, включаючи економіку, інженерію, медицину, соціологію та інші сфери, де аналіз і передбачення на основі даних відіграють ключову роль.

Середовищем для проведення аналізу є будь-який інструмент, що підтримує програмування на мові Python [1], або онлайн-середовище Google Colaboratory [2]. Додаткові необхідні бібліотеки мови Python: Pandas [3], Sklearn [4], Seaborn [5] та інші за потреби.

# ЛАБОРАТОРНА РОБОТА
# РОЗРОБКА МОДЕЛІ ДЛЯ ПЕРЕДБАЧЕННЯ

**Мета роботи:** отримати навички розробки моделей для передбачення.

## 1 ТЕОРЕТИЧНІ ВІДОМОСТІ

Моделі для передбачення – це математичні або статистичні алгоритми, які використовуються для прогнозування майбутніх значень певних показників на основі аналізу даних про минулі значення цих показників.

Моделі для передбачення можуть бути простими або складними, залежно від кількості факторів, які враховуються в аналізі, та складності математичних формул, за якими розраховуються прогнози. Однією з найпоширеніших моделей прогнозування є регресійний аналіз.

Регресійний аналіз – це статистичний метод для виявлення та кількісної оцінки зв'язку між залежною змінною та однією або кількома незалежними змінними.

Залежна змінна – це змінна, яка аналізується або передбачається.

Незалежна змінна (змінна-предиктор) – це змінна, яка впливає на залежну змінну.

Метою регресійного аналізу є визначення найкращої лінії або кривої, яка відображає зв'язок між незалежними змінними та залежною змінною. Дана лінія генерується за допомогою статистичних методів, які зменшують розбіжності між очікуваними та реальними значеннями в процесі збору даних.

Основними типами регресійного аналізу є лінійна регресія; множинна регресія, поліноміальна регресія та їх модифікації.

Для проведення регресійного аналізу при моделюванні мовою Python в рамках роботи використовується пакет scikit-learn: https://scikit-learn.org/stable/install.html.

Scikit-learn – це бібліотека машинного навчання, написана на Python. Вона спеціалізується на алгоритмах машинного навчання для вирішення задач навчання з учителем: класифікації й регресії, а також для завдань навчання без учителя: кластеризації, зменшення розмірності й виявлення аномалій. Бібліотека розширює функціональність пакетів NumPy і SciPy за допомогою численних алгоритмів, а також використовує пакет Matplotlib для побудови графіків.

### 1.1 Лінійна регресія

В загальному сенсі лінійна регресія деякої залежної змінної Y, що спирається на набір незалежних змінних X = ($x_1$, …, $x_r$), де r – це кількість предикторних (незалежних) змінних, передбачає, що лінійне відношення між Y та X описується формулою 1.1.

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_r x_r + \varepsilon \tag{1.1}$$

де $\beta_0$, $\beta_1$, …, $\beta_r$ – коефіцієнти регресії;

$\varepsilon$ – випадкова помилка.

Формула 1.1 є **рівнянням регресії**.

Лінійна регресія обчислює **оцінні функції** коефіцієнтів регресії або просто **прогнозовані вагові коефіцієнти**, що позначаються як $b_0$, $b_1$, …, $b_r$. Вони визначають **оцінну функцію регресії** (формула 1.2).

$$f(x) = b_0 + b_1 x_1 + \cdots + b_r x_r. \tag{1.2}$$

де $b_0$, $b_1$, …, $b_r$ – прогнозовані ваги виміру;

$x_1$, …, $x_r$ – незалежна змінна.

Для кожного результату спостереження $i = 1, \ldots, n$ **оцінна** (або **передбачена**) **відповідь** $f(x_i)$ має бути якомога ближчою до відповідної фактичної відповіді $y_i$. Різниці $y_i - f(x_i)$ для всіх результатів спостережень називаються **залишками**. Регресія визначає найкращі прогнозовані ваги вимірювання, які відповідають найменшим залишкам.

Для отримання найкращих ваг потрібно мінімізувати суму залишкових квадратів (Sum of Squares due to Regression SSR) для всіх результатів спостережень (формула 1.3).

$$SSR = \Sigma_i(y_i - f(x_i))^2. \quad (1.3)$$

де $y_i$ – фактичне значення;

$f(x_i)$ – передбачене значення.

Цей підхід називається **методом найменших квадратів (МНК)**.

## 1.2 Проста лінійна регресія

**Проста лінійна регресія** – це метод, який допомагає зрозуміти зв'язок між двома змінними: незалежною змінною (**X**) та відповіддю/залежною змінною (яку необхідно передбачити) (**Y**).

Оцінна функція для простої лінійної регресії має вигляд, наведений і формулі 1.4.

$$f(x) = b_0 + b_1x. \quad (1.4)$$

де $b_0$ - коефіцієнт перетину функції;

$b_1$ - коефіцієнт нахилу функції.

Результатом лінійної регресії є лінійна функція, яка прогнозує змінну відповіді (залежну) як функцію предикторної (незалежної) змінної. Для створення простої лінійної регресії використовується функція *LinearRegression()* (лістинг 1.1).

Лістинг 1.1 – Підключення функції для реалізації лінійної регресії

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
```

При підборі моделі та для подальшої перевірки коректності її функціонування доречно виділити з існуючого dataset (датасет – набір вхідних даних) дві вибірки: для тренування та для тестування, які не будуть перетинатися та дозволять оцінити якість результатів.

З метою розбиття наявного dataset на частину для тренування (навчальну) та тестову вибірку можна скористатися функцією *train_test_split()* (лістинг 1.2).

Лістинг 1.2 – Розбиття вхідної вибірки на тестову та тренувальну

```
X = df[['Year']]
Y = df['Percent Growth']
from sklearn.model_selection import
     train_test_split
x_train, x_test, y_train, y_test =
     train_test_split(X, Y, test_size = 0.3,
     random_state=42)
```

Параметр *test_size* визначає відсотковий розподіл між тренувальною та тестовою частинами. Значення «*0.3*» означає, що 30% даних будуть використані для тестування, а 70% – для побудови моделі (або тренування).

Параметр *random_state* використовується для керування правилом випадкового розподілу між частинами. При використанні будь-якого цілочисельного значення випадкове розділення фіксується за певним принципом, щоб результати були відтворюваними (тобто кожного разу розділення буде однаковим, якщо використовувати однакові X і Y і те саме значення в параметрі random_state).

Далі на основі частини для тренування виконаємо розрахунок лінійної регресії (лістинг 1.3).

Лістинг 1.3 – Розрахунок лінійної регресії

```
lm.fit(x_train,y_train) # обчислюються значення ваг
     # функції, використовуючи X та Y як аргументи
r_sq = lm.score(x_train,y_train) # визначення (R²)
print('intercept:', lm.intercept_) # коефіцієнт b0
print('slope:', lm.coef_) # коефіцієнт b1
```

В результаті отримуються коефіцієнти оцінної функції для простої лінійної регресії.

**Використання моделі.** З метою отримання передбачуваної відповіді використовується функція *.predict()* для тестової частини набору даних (лістинг 1.4).

Лістинг 1.4 – Варіанти отримання передбачуваної відповіді для тестової частини вибірки

```
y_pred = lm.predict(x_test)
print('predicted response:', y_pred, sep='\n')
```

або

```
y_pred = lm. intercept_ + lm.coef_ * x_test
print('predicted response:', y_pred, sep='\n')
```

Отриману модель можна використовувати також і для довільної послідовності вхідних значень, тобто, фактично, для безпосереднього прогнозування результату використання моделі. Приклад використання наведений в лістингу 1.5.

Лістинг 1.5 - Варіанти отримання передбачуваної відповіді для довільного значення

```
x_new = np.arange(5).reshape((-1, 1))
y_new = lm.predict(x_new)
print(y_new)
```

### 1.3 Множинна лінійна регресія

**Множинна лінійна регресія** дозволяє виконувати передбачення параметру, використовуючи більше однієї змінної в моделі для прогнозування. Множинна лінійна регресія дуже схожа на просту лінійну регресію, але цей метод використовується для пояснення зв’язку між однією залежною і двома або більше предикторними (незалежними) змінними. Більшість реальних регресійних моделей включають кілька предикторів.

Рівняння множинної лінійної регресії описано формулою 1.5.

$$y = w_0 + w_1*x_1 + w_2*x_2 + ... + w_n*x_n \quad (1.5)$$

де $w_0$ – коефіцієнт перетину функції;

$w_1, w_2, ..., w_n$ – коефіцієнти при предикторних змінних;

$x_1, x_2, ..., x_n$ – предикторні змінні.

В цьому випадку для підготовки даних в dataset залишаються тільки необхідні для аналізу критерії, значення яких нормалізується відносно даних, що там зберігаються (функція *StandardScaler().fit_transform()*). Приклад нормалізації наведено в лістингу 1.6.

Лістинг 1.6 – Нормалізація вхідних даних для множинної регресії

```
lmr1 = LinearRegression() # параметр fit_intercept
     # відповідає за зсув w_0, зсув не дорівнює 0,
     # якщо fit_intercept = True
X1 = df[['Year','Jan Units','Active']]
     # обираються кілька критеріїв для аналізу
Y1 = df['Percent Growth']

from sklearn.preprocessing import StandardScaler
X_scal = StandardScaler().fit_transform(X1)
     # нормалізація вхідних значень за вмістом
```

Функція *LinearRegression()* має параметр *fit_intercept*, який визначає факт обчислення вільного члену *w_0* в моделі (фактично – зсуву моделі, коефіцієнт перетину функції). За замовчуванням *fit_intercept = True*, тобто обчислення вільного члену виконується.

Далі отриманий набір даних розбивається на навчальну та тестову вибірки і аналізуються за принципами простої лінійної регресії.

При використанні лінійної регресії взаємозв'язок між даними моделюється за допомогою лінійних функцій, а невідомі параметри моделі оцінюються за вхідними даними. Подібно до інших методів регресійного аналізу лінійна регресія повертає розподіл умовної

імовірності Y в залежності від X, а не розподіл спільної імовірності Y та X, що стосується області мультиваріативного аналізу.

При розрахунках параметрів моделі лінійної регресії зазвичай застосовується метод найменших квадратів (МНК), але також можуть бути використані інші методи. Їх вибір залежить від задачі та вимог до точності результатів.

### 1.4 Графік регресії

Для візуалізації відповідності моделі при роботі з простою лінійною регресією є використання **графіків регресії**. Цей графік показує комбінацію розсіяних точок даних (діаграма розсіювання), а також розраховану лінію лінійної регресії, що проходить через дані. Це дає обґрунтовану оцінку зв'язку між двома змінними, силу кореляції, а також напрям (позитивна чи негативна кореляція).

Для візуалізації дисперсії даних також використовується **графік залишку**. Різниця між спостережуваним значенням (y) і прогнозованим значенням (Y) називається **залишок** (e). На графіку регресії залишок – це відстань від точки даних до встановленої лінії регресії.

**Графік залишків** – це графік, який показує помилки на вертикальній осі Y і незалежну змінну на горизонтальній осі X. Для цього використовується функція *seaborn.residplot()* [6] (лістинг 1.7).

Лістинг 1.7 – Приклад побудови графіку залишків

```
width = 12
height = 10
plt.figure(figsize=(width, height))
sns.residplot (x="Year", y="Percent Growth",
     data=df, order = 3)
     # order - порядок полінома, що підбирається
plt.show()
```

Відображення моделі для множинної лінійної регресії є складнішим, тому що його не можна візуалізувати за допомогою регресії або залишкового графіка. Один із способів поглянути на підгонку моделі – це поглянути на графік розподілу. Можна порівняти розподіл підібраних значень, які є результатом моделі, з розподілом фактичних значень і зробити відповідні висновки.

У **статистиці лінійна регресія** – це метод моделювання залежності між скалярною змінною Y та векторною (у загальному випадку) змінною X. У разі, якщо змінна X також є скаляром, регресію називають **простою**.

### 1.5 Поліноміальна регресія

**Поліноміальна регресія** є окремим випадком моделі загальної лінійної регресії або моделей множинної лінійної регресії. В даному випадку отримуються нелінійні зв'язки шляхом піднесення в квадрат або встановлення членів вищого порядку змінних-предикторів.

Поліноміальна регресія є однією з форм регресійного аналізу, в якому залежність між незалежною змінною $X$ і залежною змінною $Y$ моделюється як поліном від $X$ ступеню $n$. Поліноміальна регресія відповідає нелінійній залежності між значенням $X$ та відповідним умовним математичним сподіванням $Y$, що позначається $E(Y|X)$. Хоча поліноміальна регресія відноситься до нелінійних моделей даних, з боку теорії оцінювання ця задача є лінійною, в тому сенсі, що функція регресії $E(Y|X)$ є лінійною за невідомих параметрів, які оцінюються за даними. З цього приводу поліноміальна регресія вважається окремим випадком множинної лінійної регресії.

Пояснювальні (незалежні) змінні, що є результатом поліноміального розширення «базових» змінних, відомі як **терміни вищого ступеня**. Такі змінні також використовуються в налаштуваннях класифікації.

Фактично поліноміальна регресія отримується як лінійна регресія шляхом додавання ступеня кожної ознаки (критерія) у вигляді

нових ознак з наступним аналізом набору даних. Такий підхід (**поліноміальна регресія**) дозволяє вловлювати лінійні зв'язки в багатовимірному просторі ознак.

Для перетворення ознак у поліном ступеня *n* в *scikit-learn* є клас *PolynomialFeatures,* який крім ступенів кожної ознаки ще додає їх комбінації до заданого ступеня. Наприклад, для ознак *a* та *b* зі ступенем *3* крім $a^2$, $a^3$ та $b^2$, $b^3$ будуть також додані їх комбінації у вигляді $ab$, $a^2b$ та $ab^2$. Приклад виконання даного процесу наведений на лістингу 1.8.

Лістинг 1.8 – Перетворення ознак у поліном певного ступеня

```
from sklearn.preprocessing import
     PolynomialFeatures
X_ = PolynomialFeatures(degree=2,
     include_bias=False).fit_transform(X)
```

За допомогою *.fit_transform()* вхідний масив перетворюється в сукупність нових ознак шляхом комбінування існуючих ознак. Параметр degree визначає ступінь поліноміальної комбінації ознак (при degree=2 в рамках однієї ознаки присутні виключно до двох змінних, наприклад: $x_1$, $x_1^2$, $x_1 \cdot x_2$). Параметр *include_bias* відповідає за додавання *(True)* або не додавання *(False)* стовпця вільних членів *(intercept)*. Цей метод повертає змінений масив вхідних даних.

Поліноміальна регресія залишається лінійною, проте на графіку у вихідному просторі ознак вона матиме вигляд кривої, оскільки отримана гіперплощина у багатовимірному просторі ознак буде відповідати складній кривій лінії у вихідному просторі.

Інтерпретувати окремі коефіцієнти в поліноміальній регресії часто буває важко, оскільки основні одночлени можуть бути високо корельованими. Наприклад, X та $X^2$ мають кореляцію близько 0.97 коли X рівномірно розподіляється на інтервалі (0, 1). Хоча кореляцію можна зменшити за допомогою ортогональних поліномів, загалом більш інформативно розглядати побудовану функцію регресії в цілому.

## 1.6 Pipeline конвеєри

Конвеєри даних спрощують етапи обробки даних. Для створення конвеєра використовується модуль *Pipeline*, а також *StandardScaler* як крок конвеєра.

Оцінюючи моделі варто отримати не тільки візуалізацію результатів, а й кількісний показник, щоб визначити, наскільки точна модель.

Показники, які часто використовуються в статистиці для визначення точності моделі: **середня квадратична помилка** (MSE) та **коефіцієнт детермінації ($R^2$) –** міра, яка вказує, наскільки близькі дані до підігнаної лінії регресії.

***Значення $R^2$*** – це відсоток варіації змінної відповіді ($y$), що пояснюється лінійною моделлю (формула 1.6).

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y}_i)^2}, \qquad (1.6)$$

де $y_i$ - представляє фактичні значення,

$\hat{y}_i$ - представляє передбачення,

$\overline{y}_i$ - середнє значення всіх значень

*Середня квадратична помилка (MSE)* вимірює середнє значення квадратів помилок. Тобто різниця між фактичним значенням ($y$) і розрахунковим (передбаченим) значенням ($\hat{y}$).

## 1.7 Прийняття рішень: оцінка відповідності моделі

Якщо порівнювати моделі, то для даних краще підходить:

- модель з більшим значенням R-квадрата.
- модель з найменшим значенням MSE

Підрахунок даних параметрів реалізується на основі спрогнозованих за допомогою моделі відповідей у відповідності із тестовою вибіркою (лістинг 1.9).

Лістинг 1.9 – Розрахунок коефіцієнтів для оцінки моделей

```
r_sq = lm.score(x_train,y_train) # R-квадрата
print('coefficient of determination:', r_sq)

from sklearn.metrics import mean_squared_error,
sk_lm_mse = mean_squared_error(y_test, y_pred)
    # MSE
print(f'Linear regression MSE: {sk_lm_mse}', '\n')
```

Наприклад, в результаті аналізу були отримані наступні значення різних моделей.

Проста лінійна регресія:

- R-квадрат: 0,49659118843391759;
- MSE: 3,16 x $10^7$.

Множина лінійна регресія:

- R-квадрат: 0,80896354913783497;
- MSE: 1,2 x$10^7$.

Поліноміальна підгонка:

- R-квадрат: 0,6741946663906514;
- MSE: 2,05 x $10^7$.

Проведемо аналіз отриманих результатів.

### 1.7.1 Проста модель лінійної регресії (SLR) в порівнянні з моделлю множинної лінійної регресії (MLR)

Зазвичай, чим більше змінних приймає участь в аналізі, тим краще модель прогнозує, але це не завжди вірно. Іноді може в якості документу бути недостатньо даних, і можна зіткнутися з чисельними проблемами, або багато змінних можуть бути некорисними і навіть діяти як шум. У результаті завжди треба перевіряти MSE та $R^2$.

Щоб порівняти результати моделей MLR і SLR, розглянемо комбінацію як R-квадрата, так і MSE, щоб зробити найкращий висновок щодо відповідності моделі.

MSE: MSE SLR становить $3{,}16 \times 10^7$, тоді як MLR має MSE $1{,}2 \times 10^7$. MSE MLR набагато менший.

R-квадрат: у цьому випадку також видно, що існує велика різниця між R-квадратом SLR та R-квадратом MLR. R-квадрат для SLR (~0,497) дуже малий порівняно з R-квадрат для MLR (~0,809).

Цей R-квадрат у поєднанні з MSE показує, що MLR здається кращою моделлю в цьому випадку в порівнянні з SLR.

### 1.7.2 Проста лінійна модель (SLR) в порівнянні з поліноміальною регресією (Polynomial Fit)

MSE: Очевидно, що Polynomial Fit краще MSE, оскільки ця MSE менша, ніж у SLR.

R-квадрат: R-квадрат для Polynomial Fit більше, ніж R-квадрат для SLR, тому Polynomial Fit також значно збільшив R-квадрат.

Оскільки поліноміальна підгонка призвела до нижчого MSE і більшого R-квадрата, ми можемо зробити висновок, що це була краща модель, ніж проста лінійна регресія.

### 1.7.3 Множинна лінійна регресія (MLR) в порівнянні з поліноміальною регресією

MSE: MSE для MLR менший, ніж MSE для поліноміальної моделі.

R-квадрат: R-квадрат для MLR також набагато більший, ніж для Polynomial Fit.

Таким чином, за результатами моделювання найкращий опис даних надає модель множинної лінійної регресії.

# 2 ЛАБОРАТОРНЕ ЗАВДАННЯ

Виконати аналіз даних з набору dataset, підготовленого в попередніх лабораторних роботах або обрати довільний інший dataset по узгодженню з викладачем.

В рамках завдання побудувати та проаналізувати наступні моделі, обираючи змінні в залежності від предметної області та поставленого завдання:

- лінійно-регресійну модель. Чи вдала ця модель для даного випадку?
- модель множинної лінійної регресії;
- поліноміальну модель 11 порядку зі змінними x і y.

Побудувати для деяких змінних на ваш розсуд усі розглянуті моделі.

За результатами аналізу отриманих результатів зробити висновки в рамках предметної області.

# 3 ЗМІСТ ЗВІТУ

Звіт з лабораторної роботи може бути представлений у вигляді документу, підготовленому за допомогою текстового редактору, або у вигляді документа з розширенням .ipynb (файл / завантажити / завантажити ipynb для середовища Google Colaboratory).

Звіт має містити:

- назва роботи;
- виконавець роботи;
- мета роботи;
- хід роботи;
- результати перевірки та обробки вхідних даних;
- висновки;
- відповіді на контрольні питання.

# 4 КОНТРОЛЬНІ ПИТАННЯ

1. Що таке лінійна регресія?
2. Що таке множинна регресія?
3. Що таке $R^2$? Що таке MSE?
4. Як зрозуміти, яка модель краще робить передбачення?
5. Як використати побудовану модель? І для чого потрібно її використовувати?

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Офіційний сайт Python. URL: https://www.python.org/
2. Google Colaboratory. URL: https://colab.research.google.com/
3. Pandas API reference. URL: https://pandas.pydata.org/docs/reference/index.html
4. Sklearn API reference. URL: https://scikit-learn.org/stable/api/index.html
5. Seaborn.residplot. URL: https://seaborn.pydata.org/generated/seaborn.residplot.html