Практична робота №4

Побудова та діагностика регресійних моделей для аналізу ринку нерухомості

Мета роботи: Отримати практичні навички побудови моделей лінійної регресії, опанувати методи діагностики вхідних даних (мультиколінеарність, розподіл, гетероскедастичність), навчитися інтерпретувати статистичні звіти та порівнювати регресійні моделі з різними методами регуляризації.

Стек технологій:

Pandas *- маніпуляція табличними даними.*

NumPy *- робота з багатовимірними масивами та математичними функціями.*

Statsmodels *- побудова OLS-регресії, розрахунок p-values, перевірка мультиколінеарності (VIF) та гетероскедастичності.*

SciPy *(stats) - статистичні тести на нормальність (Shapiro-Wilk, Normaltest) та побудова Q-Q графіків.*

Scikit-learn *- розділення даних (Train/Test split), побудова моделей регресії (Linear, Ridge, Lasso) та розрахунок метрик помилок.*

Matplotlib *- базові графіки та діаграми.*

Seaborn *- розширені статистичні графіки (теплові карти, гістограми з KDE).*

Зміст роботи

Практична робота спрямована на дослідження та побудову інтелектуальних моделей прогнозування ціни нерухомості на основі регресійного аналізу. Об'єктом дослідження є набір даних ***USA_Housing.csv,*** що містить статистичну інформацію про доходи населення, вік будинків, кількість кімнат та демографічні показники в регіонах США.

Основні змінні:

- *Avg. Area Income - середній дохід населення в районі*
- *Avg. Area House Age - середній вік будинків*
- *Avg. Area Number of Rooms - середня кількість кімнат*
- *Avg. Area Number of Bedrooms - середня кількість спалень*
- *Area Population - чисельність населення*
- *Price - ціна будинку*

– *Address - адреса (текстовий стовпець, видаляється перед навчанням)*

Для дослідження будуть використані наступні бібліотеки:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import statsmodels.api as sm
import statsmodels.formula.api as smf
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.stats.diagnostic import het_breuschpagan
from scipy.stats import normaltest, shapiro

from sklearn.model_selection import train_test_split, cross_val_score,
GridSearchCV, KFold
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import RFE
```

Завдання 1. Етап підготовки даних і первинний аналіз:

– Завантажити данні і проаналізувати структуру.

– Побудувати описову статистику (describe()).

– Виявити пропущені значення.

– Проаналізувати типи змінних.

– Відповісти на питання:

1. Чи є в даних змінні, що мають нульову дисперсію (однакові значення для всіх рядків)?
2. Чи виявлено аномалії в розрізі кімнат (Avg. Area Number of Rooms)?
3. Які змінні мають найбільший діапазон значень і як це може вплинути на коефіцієнти?
4. Чому ми видаляємо текстові стовпці перед навчанням моделі.

Методичні рекомендації

Первинний аналіз (*Exploratory Data Analysis, EDA*) є фундаментом інтелектуального аналізу. На цьому етапі ви маєте підтвердити, що дані придатні для лінійної регресії та не містять критичних помилок.

Перш ніж будувати модель, необхідно зрозуміти типи даних, з якими ви працюєте. Зверніть увагу на типи стовпців, для регресійного аналізу предиктори повинні бути числовими (*int64* або *float64*). Якщо числовий за змістом стовпець позначений як *object*, це свідчить про наявність некоректних символів (пробіли, текстові позначки). У такому разі потрібне приведення типів.

Функція *describe*() надає ключову інформацію про розподіл даних: середнє значення, стандартне відхилення, медіану та квартилі. Порівняйте максимальні (*max*) та мінімальні (*min*) значення із середнім. Наприклад, якщо *Avg. Area House Age* має від'ємне значення або понад 150 років , то це є аномалія, яку слід видалити або скоригувати.

```
housingDF = pd.read_csv('USA_Housing.csv')

housingDF.info()
housingDF.describe()
print('\nПропущені значення:')
print(housingDF.isnull().sum())
print(f'\nДублікати: {housingDF.duplicated().sum()}')
```

Зверніть увагу на масштаб. *Avg. Area Income* вимірюється десятками тисяч, тоді як *Number of Bedrooms* - одиницями. Різні масштаби критичні для моделей (Ridge/Lasso), де змінні з більшим масштабом будуть домінувати.

Пропуски в даних (NaN) можуть призвести до помилок під час навчання моделі або до зміщення оцінок. Використовуйте *housingDF.isnull().sum()*. Якщо пропусків < 5%, рядки можна видалити. Якщо більше - використайте стратегію імпутації (заповнення медіаною для числових даних). !!! Порада. Уникайте заповнення середнім значенням, якщо дані мають сильні викиди.

Завдання 2. Провести аналіз розподілу цільової змінної

– Побудувати розподіл цільової змінної (*Price*) та перевірити відповідності нормальному закону за допомогою графіків *Q-Q Plot*.

– Обчислити Skewness та Kurtosis.

– Обчислити т ест Шапіро-Вілка (для вибірок до 5000 спостережень)

– Провести тест на нормальність (D'Agostino – Pearson).

– Провести інтерпретацію результатів.

Методичні рекомендації

Дослідження розподілу цільової змінної є фундаментальним. Попри те, що МНК-регресія (Метод Найменших Квадратів, англ. Ordinary Least Squares, OLS) припускає нормальність лише для залишків, значна асиметрія часто спричиняє нестабільність прогнозів та знижує точність моделі.

Поєднання гістограми з KDE (kde=True) дає змогу верифікувати гіпотезу про нормальність розподілу.

Наявність лінії KDE дозволяє візуально порівняти розподіл із теоретичною «дзвоноподібною» кривою Гаусса. Наявність декількох піків (мультимодальність) або «важких» хвостів, може свідчити про неоднорідність вибірки.

```
from scipy import stats

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
sns.histplot(housingDF['Price'], kde=True, color='teal')
plt.title('Histogram of Housing Prices')
plt.xlabel('Price ($)')
plt.ylabel('Frequency')

plt.subplot(1, 2, 2)
stats.probplot(housingDF['Price'], plot=plt)
plt.title('Probability Plot (Q-Q)')
plt.tight_layout()
plt.show()
```

!!! Порада. При аналізі результатів фокусуйтеся на «важких хвостах» та мультимодальності, оскільки ці фактори прямо впливають на стабільність майбутньої регресійної моделі.

Q-Q Plot (*Quantile-Quantile*) - графік зіставляє квантилі даних із квантилями ідеального нормального розподілу. Якщо точки лежать на діагональній лінії - розподіл нормальний. Відхилення точок на кінцях лінії свідчать про аномалії в «хвостах» (викиди).

Skewness та *Kurtosis* дозволяють кількісно оцінити відхилення від нормальності:

– Skewness (S) - коефіцієнт асиметрії (S = 0 - розподіл симетричний, S > 0 - правобічна асиметрія (довгий хвіст справа). Типово для цін на нерухомість, S < 0 -лівобічна асиметрія, |S| > 1 вказує на суттєву асиметрію.

– Kurtosis (K) - коефіцієнт ексцесу, який визначає «гостроту» піка та товщину хвостів. Для нормального розподілу K ≈ 3 (або 0 для excess kurtosis).

5. Leptokurtic (K > 3) - гострий пік і товсті хвости (багато викидів).
6. Platykurtic (K < 3) - плоский пік і тонкі хвости.

```
print(f'Skewness: {housingDF["Price"].skew():.4f}')
print(f'Kurtosis: {housingDF["Price"].kurtosis():.4f}')
```

Тест Д'Агостіно–Пірсона базується на поєднанні показників асиметрії та ексцесу. У бібліотеці *scipy* він реалізований як stats.*normaltest*(). Гіпотези**:**

– H0: Дані розподілені нормально.

– H1: Дані не підпорядковуються нормальному закону.

```
stat, p = stats.normaltest(housingDF['Price'])
print("--- Тест Д'Агостіно-Пірсона ---")
print(f'Статистика: {stat:.4f}, p-value: {p:.4e}')
if p > 0.05:
    print('Розподіл виглядає нормальним (не відхиляємо H0)')
else:
    print('Відхиляємо H0 — розподіл не є нормальним')

if len(housingDF) <= 5000:
    sw_stat, sw_p = shapiro(housingDF['Price'])
    print(f'\nТест Шапіро-Вілка: stat={sw_stat:.4f}, p={sw_p:.4e}')
```

Інтерпретація. Якщо *p-value* < 0.05, відхиляємо H0, дані не є нормальними. На великих вибірках (n > 5000) цей тест майже завжди відхиляє H0 навіть при незначних відхиленнях, тому візуальний аналіз (*Q-Q plot*) має пріоритет.

Якщо Skewness суттєво більше 1, а тест нормальності відхилив H0, рекомендується застосувати логарифмування: y' = ln(y).

Попередження! Після логарифмування інтерпретація коефіцієнтів змінюється - вони тепер показують відсоткову зміну ціни, а не абсолютну.

Завдання 3. Провести кореляційний аналіз

– Побудувати матрицю кореляції Пірсона.

– Створити теплову карту.

– Виявити сильні зв’язки.

Методичні рекомендації

У контексті регресії ми шукаємо високу кореляцію між незалежними змінними та цільовою міткою.

```
corr_matrix = housingDF.drop('Address', axis=1).corr()

# Маска для верхнього трикутника (зменшує візуальне навантаження)
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, mask=mask, annot=True, fmt='.2f',
            cmap='coolwarm', vmin=-1, vmax=1,
            square=True, linewidths=0.5)
plt.title('Кореляційна матриця (нижній трикутник)')
plt.tight_layout()
plt.show()
```

!!! Порада. Використовуйте розбіжну палітру кольорів (наприклад, *coolwarm* або *RdBu*), де нейтральний колір відповідає нулю. Це дозволяє миттєво відрізнити позитивний зв'язок від негативного. Оскільки матриця кореляції симетрична відносно головної діагоналі, рекомендується накладати маску на верхній трикутник (*np.triu*), щоб зменшити візуальне навантаження.

Завдання 4. Провести перевірку мультиколінеарності

– Розрахувати коефіцієнта інфляції дисперсії (Variance Inflation Factor, VIF) для виявлення та усунення мультиколінеарності.

Методичні рекомендації

Мультиколінеарність - це явище, коли дві або більше незалежних змінних у моделі регресії мають сильний лінійний зв'язок між собою. Це призводить до необмеженого зростання стандартних помилок коефіцієнтів, через що результати стають статистично незначущими, навіть якщо модель має високий *R^2*.

VIF (*Variance Inflation Factor*) - коефіцієнт інфляції дисперсії. Для кожної незалежної змінної x_i розраховується окремий показник *VIF* шляхом побудови допоміжної регресії, де x_i виступає як залежна змінна, а всі інші ознаки - як незалежні.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
```

```
X = housingDF.drop(['Price', 'Address'], axis=1)
X_with_const = add_constant(X)

vif_data = pd.DataFrame()
vif_data['Feature'] = X_with_const.columns
vif_data['VIF'] = [variance_inflation_factor(X_with_const.values, i)
                   for i in range(len(X_with_const.columns))]

print(vif_data[vif_data['Feature'] != 'const'].sort_values('VIF',
ascending=False))
```

Таблиця 2.- Шкала оцінювання

Значення	Стан	Опис та наслідки
VIF < 5	Норма	Мультиколінеарність відсутня або незначна. Коефіцієнти стабільні.
5 ≤ VIF < 10	Потенційна проблема	Присутня помірна колінеарність. Необхідно проаналізувати кореляційну матрицю.
VIF ≥ 10	Критичний рівень	Сильна мультиколінеарність. Довіра до значень коефіцієнтів β практично втрачена.

!!! Порада. Якщо *Avg. Area Number of Rooms* та *Avg. Area Number of Bedrooms* сильно корелюють, залиште ту ознаку, яка має вищу кореляцію з цільовою змінною (Price).

Завдання 5. Виявлення та обробка викидів

– Провести аналіз впливових спостережень, які можуть суттєво зміщувати коефіцієнти регресії. Для цього використовуються три взаємодоповнюючих інструменти: відстань *Кука*, *leverage-значення* та графік *Residuals vs Leverage*.

– Проведіть інтерпретацію графіка *Residuals vs Leverage*.

– За необхідності зробіть видалення впливових спостережень.

Методичні рекомендації

Відстань *Кука Di* вимірює, наскільки суттєво змінилися б усі регресійні коефіцієнти, якби видалити і-те спостереження. Спостереження з Di > 4/n вважається потенційно впливовим.

```
# Побудова моделі для аналізу залишків
tempDF = housingDF.rename(columns={
    'Avg. Area Income': 'Income',
    'Avg. Area House Age': 'HouseAge',
    'Avg. Area Number of Rooms': 'Rooms',
```

```
    'Avg. Area Number of Bedrooms': 'Bedrooms',
    'Area Population': 'Population'
})

model_full = smf.ols('Price ~ Income + HouseAge + Rooms + Population',
                     data=tempDF).fit()

# Відстань Кука
influence = model_full.get_influence()
cooks_d = influence.cooks_distance[0]

# Поріг: 4/n
threshold = 4 / len(tempDF)
influential = np.where(cooks_d > threshold)[0]

print(f'Кількість впливових спостережень: {len(influential)}')
print(f'Поріг Кука: {threshold:.4f}')
```

Візуалізація відстані *Кука* (Cook's Distance) є критично важливою частиною побудови моделей регресії (особливо лінійної). Простими словами, вона допомагає знайти дані, які «занадто сильно» тягнуть ковдру на себе, викривляючи результати прогнозування. Візуалізація дозволяє миттєво побачити такі точки як високі «піки», що виділяються над загальною масою.

```
plt.figure(figsize=(12, 4))
plt.stem(range(len(cooks_d)), cooks_d, markerfmt=',')

plt.axhline(y=threshold, color='red', linestyle='--', label=f'Поріг =
{threshold:.4f}')
plt.xlabel('Індекс спостереження')
plt.ylabel("Cook's Distance")
plt.title("Cook's Distance — виявлення впливових спостережень")
plt.legend()
plt.tight_layout()
plt.show()
```

Значення вище порогу – це точки, які потребують ручної перевірки. Можливо, там сталася помилка при введенні даних або це унікальний випадок, який не має описуватися загальною моделлю.

У контексті машинного навчання та регресійного аналізу, *Leverage* (важіль) та графік *Residuals vs Leverage* є критично важливими інструментами для діагностики якості моделі та виявлення аномалій. Leverage — це показник того, наскільки значення незалежних змінних конкретного спостереження є «віддаленим» від середнього значення вибірки.

```
# Leverage values (hat matrix diagonal)
```

```
leverage = influence.hat_matrix_diag
p = model_full.df_model + 1  # кількість параметрів включно з константою
leverage_threshold = 2 * p / len(tempDF)

# Стандартизовані залишки
standardized_resid = influence.resid_studentized_internal

# Графік Residuals vs Leverage
plt.figure(figsize=(8, 6))
plt.scatter(leverage, standardized_resid, alpha=0.5)
plt.axhline(y=0, color='black', linestyle='-', linewidth=0.5)
plt.axhline(y=2, color='red', linestyle='--', alpha=0.7, label='|resid| =
2')
plt.axhline(y=-2, color='red', linestyle='--', alpha=0.7)
plt.axvline(x=leverage_threshold, color='orange', linestyle='--',
            label=f'leverage = {leverage_threshold:.3f}')
plt.xlabel('Leverage')
plt.ylabel('Стандартизовані залишки')
plt.title('Residuals vs Leverage')
plt.legend()
plt.tight_layout()
plt.show()
```

Інтерпретація графіка Residuals vs Leverage:

- Точки у правому верхньому/нижньому куті: HIGH leverage + HIGH residual = найнебезпечніші викиди.

- High leverage + low residual: точка 'тягне' пряму, але лежить близько до неї.

- Low leverage + high residual: помітний, але не суттєво впливає на коефіцієнти.

```
clean_mask = cooks_d <= threshold
tempDF_clean = tempDF[clean_mask].copy()
print(f'Залишилось спостережень: {len(tempDF_clean)} з {len(tempDF)}')

# Перебудова моделі після очищення
model_clean = smf.ols('Price ~ Income + HouseAge + Rooms + Population',
                      data=tempDF_clean).fit()
print(f'R² до очищення: {model_full.rsquared:.4f}')
print(f'R² після очищення: {model_clean.rsquared:.4f}')
```

Завдання 6. Побудова простої лінійної регресії OLS

– Побудувати модель OLS.

– Оцінити коефіцієнти.

– Побудувати прогноз.

Методичні рекомендації

Проста лінійна регресія: $Price = \beta 0 + \beta 1 Income + \varepsilon$

де:

– $\beta 0$ (*Intercept*) - вільний член, точка перетину з віссю ординат.

– $\beta 1$ (*Slope*) - коефіцієнт нахилу, що визначає силу впливу доходу на ціну.

– ε - випадкова помилка.

```
df_model = housingDF.rename(columns={'Avg. Area Income': 'Income'})

ols_model = smf.ols(formula='Price ~ Income', data=df_model).fit()
print(ols_model.summary())

b0, b1 = ols_model.params
print(f'\nМодель: Price = {b0:.2f} + {b1:.2f} * Income')
print(f'R²: {ols_model.rsquared:.4f}')
```

Інтерпретація результатів МНК (OLS) повинна мати прикладний характер. Коефіцієнт β1 показує, на скільки одиниць зміниться ціна будинку при зміні середнього доходу в районі на одну одиницю. Якщо $\beta 1 > 0$, маємо пряму залежність. Коефіцієнт β0 - базовий рівень ціни, коли дохід дорівнює нулю. У даному датасеті він часто від'ємний, що є математично необхідним для апроксимації прямої в межах існуючих даних, але не має прямого фізичного змісту (оскільки дохід не може бути нульовим).

R^2 (R-squared) - це показник того, наскільки добре незалежна змінна пояснює варіацію залежної.

Значення R^2 = 0.64 означає, що 64% змін ціни на будинки можна пояснити зміною рівня доходу населення. Решта 36% припадають на чинники, не враховані в цій простій моделі (кількість кімнат, площа тощо).

Для перевірки статистичної значущості кожного коефіцієнта аналізуємо p-value (стовпчик P > |t|):

– Нульова гіпотеза (H0) $\beta 1 = 0$ - дохід не впливає на ціну.

– Альтернативна гіпотеза (H1) $\beta 1 <> 0$ - вплив є суттєвим.

– Якщо p-value < 0.05, ми відхиляємо H0. Це означає, що зв'язок статистично підтверджений.

Побудова прогнозу. Наприклад, прогноз ціни для району з середнім доходом $80 000.

```
new_data = pd.DataFrame({'Income': [80000]})
prediction = ols_model.predict(new_data)
print(f'\nПрогноз для доходу $80,000: ${prediction[0]:,.2f}')
```

Завдання 7. Побудувати множинну регресію та провести діагностику факторів

– Побудова множинної регресії з використанням усіх значущих факторів.

– Оцінити модель.

– Порівняти R^2 та Adjusted R^2.

– Проаналізувати p-value коефіцієнтів.

– Інтерпретувати F-статистику.

Питання для аналізу:

– Визначте, на скільки зросла точність моделі після переходу від одного фактора (Income) до чотирьох.

– Перерахуйте змінні, які виявилися найбільш впливовими.

– Підтвердьте, що зростання моделі є обґрунтованим (Adjusted R^2 має бути дуже близьким до звичайного R^2).

– *Проблема мультиколінеарності*. Як сильна кореляція між *Avg. Area Number of Rooms* та *Avg. Area Number of Bedrooms* впливає на інтерпретацію коефіцієнтів β?

– *Гетероскедастичність*. Що відбувається з довірчими інтервалами прогнозів, якщо дисперсія похибок залежить від рівня доходу (*Income*)?

– *Регуляризація*. У яких випадках варто перейти від LinearRegression до *Ridge* або *Lasso* регресії?

– Чому високий R^2 не гарантує високу прогностичну здатність моделі на практиці?

Методичні рекомендації

Множинна лінійна регресія дозволяє оцінити внесок кожної змінної в ціну, утримуючи інші чинники сталими. Модель має вигляд:

$$Price=\beta 0 + \beta 1 Income + \beta 2\ HouseAge + \beta 3\ Rooms + \beta 4\ Population + \in$$

```
tempDF = housingDF.rename(columns={
    'Avg. Area Income': 'Income',
    'Avg. Area House Age': 'HouseAge',
    'Avg. Area Number of Rooms': 'Rooms',
    'Avg. Area Number of Bedrooms': 'Bedrooms',
    'Area Population': 'Population'
})

multi_model = smf.ols(
    formula='Price ~ Income + Rooms + HouseAge + Population',
    data=tempDF
).fit()

print(multi_model.summary())
print(f'\nR²: {multi_model.rsquared:.4f}')
print(f'Adjusted R²: {multi_model.rsquared_adj:.4f}')
print(f'F-statistic p-value: {multi_model.f_pvalue:.4e}')
```

Порівняльний аналіз R^2 та Adjusted R^2 - це ключовий момент для розуміння перенавчання (overfitting):

– R^2 (Коефіцієнт детермінації) показує частку поясненої дисперсії. Його головний недолік - він ніколи не зменшується при додаванні нових предикторів, навіть якщо вони є статистичним шумом.

– Adjusted R^2 (Скоригований коефіцієнт) вводить «штраф» за кількість параметрів у моделі.

Попередження. Якщо при додаванні нової змінної R^2 зростає, а Adjusted R^2 падає - змінна є зайвою і погіршує прогнозну здатність моделі.

Аналіз p-value та обґрунтування включення змінних

Важливо не просто включити всі дані, а провести селекцію ознак:

– Кожне p-value перевіряє гіпотезу H0: βi = 0.

– Якщо p-value < 0.05, ми відхиляємо H0 і вважаємо змінну значущою.

Якщо ви помітили, що після включення Rooms значення p для Bedrooms стало високим (наприклад, 0.45), це свідчить про надлишковість

(мультиколінеарність). У фінальну модель слід включати лише ті змінні, які мають низьке p-value та високий економічний сенс.

Якщо *t-статистика* перевіряє окремі коефіцієнти, то *F-статистика* перевіряє модель в цілому.

– Гіпотеза: H0: $\beta1 = \beta2 = \ldots = \beta k = 0$.

– Якщо *Prob (F-statistic)* дуже мале (зазвичай < 0.05 або навіть 10^-100), це означає, що принаймні один із ваших предикторів має суттєвий зв'язок із ціною. Модель є «статистично кращою», ніж просте вгадування за середнім значенням.

Завдання 8. Повна діагностика залишків (OLS Residual Diagnostics)

– Виконати візуальну діагностику залишків побудованої багатофакторної регресійної моделі для перевірки виконання основних передумов методу найменших квадратів (МНК) та Гаусса-Маркова. Необхідні кроки:

1. Аналіз гомоскедастичності (Residuals vs Fitted).
2. Перевірка нормальності розподілу (Normal Q-Q Plot).
3. Виявлення гетероскедастичності (Scale-Location Plot).
4. Гістограма помилок.
5. Тест Бройша-Пагана на гетероскедастичність.
6. Тест Дарбіна-Вотсона на автокореляцію залишків.
7. Тест нормальності залишків (Jarque-Bera та Shapiro-Wilk)

Методичні рекомендації

Умови Гаусса-Маркова є фундаментальними для коректності OLS-оцінок: $E(\varepsilon) = 0$, $Var(\varepsilon) = \sigma^2$ (гомоскедастичність), $Cov(\varepsilon_i, \varepsilon_j) = 0$ (відсутність автокореляції), $\varepsilon \sim N(0, \sigma^2)$. Порушення будь-якої з цих умов призводить до ненадійних висновків.

Набір із чотирьох діагностичних графіків, об’єднаних у єдину область (2x2), що дозволяють зробити комплексний висновок про відсутність автокореляції, гетероскедастичності та нормальність залишків.

```
fig, axes = plt.subplots(2, 2, figsize=(14, 10))

fitted = multi_model.fittedvalues
```

```
residuals = multi_model.resid
std_resid = (residuals - residuals.mean()) / residuals.std()

#Residuals vs Fitted
axes[0, 0].scatter(fitted, residuals, alpha=0.4, s=10)
axes[0, 0].axhline(y=0, color='red', linestyle='--')
axes[0, 0].set_xlabel('Fitted values')
axes[0, 0].set_ylabel('Residuals')
axes[0, 0].set_title('Residuals vs Fitted (гомоскедастичність)')

#Q-Q Plot залишків
stats.probplot(residuals, plot=axes[0, 1])
axes[0, 1].set_title('Normal Q-Q (нормальність залишків)')

#Scale-Location Plot
axes[1, 0].scatter(fitted, np.sqrt(np.abs(std_resid)), alpha=0.4, s=10)
axes[1, 0].axhline(y=np.sqrt(0.5*np.pi/2), color='red', linestyle='--')
axes[1, 0].set_xlabel('Fitted values')
axes[1, 0].set_ylabel('√|Standardized Residuals|')
axes[1, 0].set_title('Scale-Location (виявлення гетероскедастичності)')

#Гістограма залишків
axes[1, 1].hist(residuals, bins=50, edgecolor='black', color='steelblue',
alpha=0.7)
axes[1, 1].set_xlabel('Residuals')
axes[1, 1].set_ylabel('Frequency')
axes[1, 1].set_title('Розподіл залишків')

plt.suptitle('Діагностика залишків регресійної моделі', fontsize=14,
fontweight='bold')
plt.tight_layout()
plt.show()
```

Тест Бройша-Пагана на гетероскедастичність. Гетероскедастичність — дисперсія похибок непостійна і залежить від значень предикторів. Це порушує третю умову Гаусса-Маркова та робить стандартні помилки некоректними.

```
from statsmodels.stats.diagnostic import het_breuschpagan

# Тест Бройша-Пагана
X_bp = sm.add_constant(tempDF[['Income', 'HouseAge', 'Rooms',
'Population']])
bp_test = het_breuschpagan(multi_model.resid, X_bp)

print('=== Тест Бройша-Пагана ===',
      f'LM statistic: {bp_test[0]:.4f}',
      f'p-value:      {bp_test[1]:.4e}',
      f'F-statistic:  {bp_test[2]:.4f}',
      f'F p-value:    {bp_test[3]:.4e}', sep='\n')

alpha = 0.05
```

```
if bp_test[1] < alpha:
    print('\nВиявлено гетероскедастичність (p < 0.05)!')
    print('Рекомендації: використати HC3 стандартні помилки або WLS')
else:
    print('\nГомоскедастичність не порушена (p >= 0.05)')

# Якщо гетероскедастичність виявлена — використайте робастні SE
model_robust = smf.ols(
    'Price ~ Income + Rooms + HouseAge + Population',
    data=tempDF).fit(cov_type='HC3')
print('\nМодель з робастними стандартними помилками HC3:')
print(model_robust.summary().tables[1])
```

Тест Дарбіна-Вотсона на автокореляцію залишків. Автокореляція залишків типова для часових рядів, але може виникати і в крос-секційних даних при наявності просторової залежності. Статистика DW ∈ [0, 4] значення ≈ 2 означає відсутність автокореляції, DW < 1.5 - позитивна автокореляція (недооцінка стандартних помилок, завищення t-статистик), DW > 2.5 - негативна автокореляція.

```
from statsmodels.stats.stattools import durbin_watson

dw_stat = durbin_watson(multi_model.resid)
print(f'Статистика Дарбіна-Вотсона: {dw_stat:.4f}')

if 1.5 < dw_stat < 2.5:
    print('DW ≈ 2: автокореляція залишків відсутня')
elif dw_stat < 1.5:
    print('DW < 1.5: позитивна автокореляція — залишки не незалежні')
else:
    print('DW > 2.5: негативна автокореляція')
```

Тест нормальності залишків (Jarque-Bera та Shapiro-Wilk).

```
import statsmodels.stats.api as sms
from scipy import stats

#Виклик тесту Харке-Бера
#Повертає: [JB_stat, p_value, skewness, kurtosis]
jb_test = sms.jarque_bera(multi_model.resid)
jb_stat = jb_test[0]
jb_p    = jb_test[1]

print(f'Тест Харке-Бера: stat={jb_stat:.4f}, p={jb_p:.4e}')

#Тест Д'Агостіно (нормальність)
da_stat, da_p = stats.normaltest(multi_model.resid)
print(f"Тест Д'Агостіно для залишків: stat={da_stat:.4f}, p={da_p:.4e}")
```

```
if jb_p > 0.05:
    print('Залишки мають нормальний розподіл (H₀ не відхиляємо)')
else:
    print('Залишки не є нормальними — перевірте Q-Q plot на важкі хвости')
```

Завдання 9. Машинне навчання та валідація (Scikit-Learn)

Етапи:

- Підготовка даних та Pipeline.
- Підбір гіперпараметрів Ridge та Lasso (GridSearchCV).
- K-Fold крос-валідація та порівняння всіх моделей.
- Аналіз залишків sklearn-моделі.

Методичні рекомендації

Pipeline гарантує коректне застосування *StandardScaler* окремо на *train* і *test* частинах, унеможливлюючи *data leakage* - типову помилку при масштабуванні всього датасету до розбивки.

```
X = housingDF.drop(['Price', 'Address'], axis=1)
y = housingDF['Price']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

- ПРАВИЛЬНИЙ спосіб масштабування всередині *Pipeline*. *Scaler* навчається ЛИШЕ на *train* - не бачить *test*-даних

```
pipeline_lr = Pipeline([
    ('scaler', StandardScaler()),
    ('model',  LinearRegression())
])

pipeline_ridge = Pipeline([
    ('scaler', StandardScaler()),
    ('model',  Ridge())
])

pipeline_lasso = Pipeline([
    ('scaler', StandardScaler()),
    ('model',  Lasso(max_iter=10000))
])
```

Попередження. Ніколи не застосовуйте *StandardScaler* до повного датасету перед *split*. *scaler.fit_transform(X)* - неправильно, тест-дані впливають на scaler. Використовуйте *Pipeline: scaler* навчається лише на X_train.

Підбір гіперпараметрів Ridge та Lasso (GridSearchCV). Завдяки впровадженню L1 (*Lasso*) або L2 (*Ridge*) через *scikit-learn*, можна автоматично

обмежувати вплив колінеарних предикторів. Метод «штрафує» великі значення коефіцієнтів, роблячи модель стійкішою до перенавчання.

```
from sklearn.model_selection import GridSearchCV

# Простір пошуку для alpha
param_grid = {'model__alpha': [0.001, 0.01, 0.1, 1, 10, 50, 100, 500, 1000]}

# Ridge
gs_ridge = GridSearchCV(
    pipeline_ridge, param_grid, cv=5,
    scoring='neg_root_mean_squared_error', n_jobs=-1
)
gs_ridge.fit(X_train, y_train)
print(f'Ridge: найкращий alpha = {gs_ridge.best_params_["model__alpha"]}')
print(f'Ridge: CV RMSE = {-gs_ridge.best_score_:,.2f}')

# Lasso
gs_lasso = GridSearchCV(
    pipeline_lasso, param_grid, cv=5,
    scoring='neg_root_mean_squared_error', n_jobs=-1
)
gs_lasso.fit(X_train, y_train)
print(f'Lasso: найкращий alpha = {gs_lasso.best_params_["model__alpha"]}')
print(f'Lasso: CV RMSE = {-gs_lasso.best_score_:,.2f}')
```

K-Fold крос-валідація та порівняння всіх моделей.

```
from sklearn.model_selection import KFold, cross_val_score

kf = KFold(n_splits=5, shuffle=True, random_state=42)

models = {
    'Linear Regression': pipeline_lr,
    'Ridge (optimal)':   gs_ridge.best_estimator_,
    'Lasso (optimal)':   gs_lasso.best_estimator_,
}

comparison = []
for name, pipe in models.items():
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)

    cv_rmse = cross_val_score(pipe, X_train, y_train, cv=kf,
                              scoring='neg_root_mean_squared_error')

    comparison.append({
        'Модель':   name,
        'Train R²': round(pipe.score(X_train, y_train), 4),
        'Test R²':  round(pipe.score(X_test, y_test), 4),
        'MAE':      round(mean_absolute_error(y_test, y_pred), 0),
        'RMSE':     round(np.sqrt(mean_squared_error(y_test, y_pred)), 0),
```

```
        'CV RMSE':  round(-cv_rmse.mean(), 0),
        'CV std':   round(cv_rmse.std(), 0),
    })

df_comparison = pd.DataFrame(comparison)
print(df_comparison.to_string(index=False))
```

Зведена таблиця порівняння моделей є обов'язковою при захисті роботи. Вона дозволяє обґрунтовано обрати фінальну модель на основі об'єктивних метрик, а не лише R^2.

Ситуація	Рекомендована модель	Обґрунтування
Немає мультиколінеарності	Linear Regression	Простота, інтерпретованість
Мультиколінеарність	Ridge (L2)	Стабілізує коефіцієнти, не обнуляє
Потрібен відбір ознак	Lasso (L1)	Обнуляє незначущі коефіцієнти

Аналіз залишків sklearn-моделі.

```
# Використовуємо кращу модель для аналізу залишків
best_pipe = gs_ridge.best_estimator_
best_pipe.fit(X_train, y_train)
y_pred_best = best_pipe.predict(X_test)

residuals_sk = y_test - y_pred_best

fig, axes = plt.subplots(1, 3, figsize=(16, 5))

# Scatter: Actual vs Predicted
axes[0].scatter(y_test, y_pred_best, alpha=0.3, s=10)
mn, mx = y_test.min(), y_test.max()
axes[0].plot([mn, mx], [mn, mx], 'r--', lw=1)
axes[0].set_xlabel('Actual')
axes[0].set_ylabel('Predicted')
axes[0].set_title('Actual vs Predicted')

# Residuals vs Predicted
axes[1].scatter(y_pred_best, residuals_sk, alpha=0.3, s=10)
axes[1].axhline(y=0, color='red', linestyle='--')
axes[1].set_xlabel('Predicted')
axes[1].set_ylabel('Residuals')
axes[1].set_title('Residuals vs Predicted')

# Histogram residuals
axes[2].hist(residuals_sk, bins=50, color='steelblue', edgecolor='black')
axes[2].set_xlabel('Residuals')
axes[2].set_title('Гістограма залишків')

plt.suptitle(f'Діагностика: {gs_ridge.best_params_}', fontsize=12)
plt.tight_layout()
plt.show()
```

```
# Метрики
print(f'MAE:  {mean_absolute_error(y_test, y_pred_best):,.2f}')
print(f'RMSE: {np.sqrt(mean_squared_error(y_test, y_pred_best)):,.2f}')
print(f'R²:   {r2_score(y_test, y_pred_best):.4f}')
```

Завдання 10. Інтерпретованість моделі (SHAP Values)

– Пояснити, чому модель робить саме такий прогноз. SHAP (SHapley Additive exPlanations) базується на теорії ігор Шеплі та дозволяє інтерпретувати внесок кожної ознаки в конкретний прогноз.

– Встановлення та базовий аналіз SHAP

```
#Встановлення: pip install shap
import shap

#LinearRegression (SHAP підтримує LinearExplainer)
lr_model = LinearRegression()
scaler_shap = StandardScaler()

X_train_sc = scaler_shap.fit_transform(X_train)
X_test_sc  = scaler_shap.transform(X_test)

lr_model.fit(X_train_sc, y_train)

#LinearExplainer — ефективний для лінійних моделей
explainer = shap.LinearExplainer(lr_model, X_train_sc)
shap_values = explainer(X_test_sc)
```

– Візуалізація SHAP

```
#Summary Plot — глобальна важливість ознак
plt.figure(figsize=(10, 6))
shap.summary_plot(shap_values, X_test_sc,
                  feature_names=X.columns.tolist())
plt.title('SHAP Summary Plot')
plt.tight_layout()
plt.show()

#Waterfall Plot — інтерпретація одного прогнозу
shap.plots.waterfall(shap_values[0])
```

Порада. *SHAP Summary Plot* - вісь *X* = *SHAP value* (вплив на прогноз), колір = значення ознаки. Позитивний *SHAP* → ознака підвищила прогноз. Негативний *SHAP* → знизила. *Waterfall Plot* показує як ознаки 'будують' конкретний прогноз від *baseline*.

Завдання 11. Машинне навчання та валідація (Scikit-Learn)

– Прогнозування та підсумкове порівняння моделей.

```
 Прогноз для об'єкта з заданими характеристиками
new_house = pd.DataFrame({
    'Avg. Area Income':          [65000],
    'Avg. Area House Age':        [7],
    'Avg. Area Number of Rooms':  [7],
    'Avg. Area Number of Bedrooms': [2],
    'Area Population':            [35000]
})

# Прогнози від кожної моделі
for name, pipe in models.items():
    pred = pipe.predict(new_house)[0]
    print(f'{name:<25}: ${pred:>12,.2f}')
```

– Сформуйте повну зведену таблицю для порівняння всіх побудованих моделей.

Завдання 12. У ході виконання практичної роботи необхідно зробити висновки за наступними пунктами:

– Визначити якість даних (наявність пропусків, викидів, аномалій; вжиті заходи з їх усунення).

– Який розподіл має цільова змінна (чи є необхідність у log-трансформації; значення Skewness та Kurtosis).

– Що дала перевірка взаємозв'язку ознак (мультиколінеарність). Вкажіть показники VIF для кожної змінної та поясніть, які з них ви видалили, щоб уникнути дублювання інформації).

– Що дала діагностика залишків? Поясніть результати тестів Бройша-Пагана, Дарбіна-Вотсона, нормальності. Чи виконуються умови Гаусса-Маркова?

– Заповніть зведену таблицю: Train R^2, Test R^2, CV RMSE, MAE для Linear/Ridge/Lasso. Яка модель обрана фінальною та чому?

– Визначити на основі стандартизованих β-коефіцієнтів та SHAP values які 3 ознаки мають найбільший вплив на Price?

– Наскільки модель придатна для реального прогнозування цін на нерухомість у США?

Контрольні питання

1. Яке основне призначення регресійного аналізу?

2. Які типи регресійних моделей виділяють?

3. Який вигляд має лінійна регресійна модель?

4. Які висуваються основні вимоги до специфікації моделі та відомих експериментальних даних?

5. У чому полягає сутність методу найменших квадратів?

6. Як оцінити якість отриманої моделі?

7. Перелічіть п'ять умов Гаусса-Маркова та поясніть, що відбувається при порушенні кожної з них.

8. Які графіки використовуються для візуальної перевірки гомоскедастичності? Що є ознакою її порушення?

9. Що таке тест Бройша-Пагана? Сформулюйте H_0 та H_1. Що робити при відхиленні H_0?

10. Як інтерпретувати статистику Дарбіна-Вотсона? За яких умов виникає автокореляція залишків?

11. Чим відрізняються тест Харке-Бера, Шапіро-Вілка та D'Agostino для перевірки нормальності залишків?

12. Що таке мультиколінеарність? Як вона впливає на стандартні помилки коефіцієнтів?

13. Що вимірює VIF? Поясніть, чому $VIF = 1/(1 - R^2_i)$, де R^2_i — коефіцієнт допоміжної регресії.

14. Порівняйте методи відбору ознак: RFE, Forward/Backward Stepwise, LASSO. У яких ситуаціях кожен з них є переважним?

15. Що таке стандартизований β-коефіцієнт і чому він необхідний для порівняння впливу змінних різного масштабу?

16. У чому фундаментальна різниця між L1 (Lasso) та L2 (Ridge) регуляризацією? Чому Lasso обнуляє коефіцієнти, а Ridge - ні?

17. Що таке bias-variance tradeoff? Як параметр alpha впливає на компроміс між зміщенням та дисперсією моделі?

18. Чому StandardScaler необхідно застосовувати всередині Pipeline, а не до повного датасету? Що таке data leakage?

19. Поясніть різницю між prediction interval та confidence interval для прогнозу. Який з них ширший і чому?

20. Чому GridSearchCV з CV=5 є кращим за простий train/test split для підбору гіперпараметрів?

21. Що таке *SHAP values*? На якій математичній основі вони базуються?

22. Чому Waterfall Plot корисний для пояснення конкретного прогнозу? Як його інтерпретувати?

23. Чому високий R^2 не гарантує якість прогнозу на нових даних? Наведіть конкретний приклад.

24. Що таке *Cook's Distance* та *leverage*? Як виявити найбільш впливові спостереження?