Problem Chosen	2024 ShuWei Cup Summary Sheet	Team Control Number
D		2024110430340

Modeling and Evaluation of Urban Resilience and Sustainable Development Capacity

Abstract

China's population is aging increasingly, the future population will be in a long-term downward trend. This will have a profound impact on the sustainable development of second - and third-tier cities and pose serious challenges to urban resilience. Based on the attached data, this paper establishes a mathematical model to evaluate urban resilience and sustainable development ability.

For problem 1, missing values and outliers are first processed on the original data, and then three regression models (linear regression, random forest and gradient lifting) are used to train the data to learn the relationship between features and targets, so as to predict housing prices. The results show that the random forest model and the gradient lifting model have better prediction effect. As a result, the housing stock estimate of City1 is 1459933.97, and the housing stock estimate of City2 is 834249.01.

For problem 2, the data in the attachment is cleaned and converted first, and then the K-means clustering method is adopted to analyze the geographical distribution and clustering characteristics of various facilities in City1 and City2, and the results are visualized. The clustering results revealed their common and unique characteristics, and an analysis of their strengths and weaknesses was conducted. The findings show that City1 has significantly higher quantities and densities of facilities such as geographical name and address information, car services, medical and health services, retail services, motorbike facilities, and science education, compared to City2.

In view of problem 3, a multi-index weighted urban resilience assessment model was established according to service index data. The results showed that City1 had a better overall resilience score of 0.74, while City2 had a score of 0.67. The analysis shows that City1 has outstanding performance in the allocation of emergency management resources and the coverage rate of government agencies, but its weaknesses lie in the diversity of social support systems and the mobilization ability of grass-roots organizations. It is suggested to strengthen the construction of community emergency facilities. The distribution and accessibility of grass-roots emergency resources in City2 is limited, and the weakness lies in the low participation of social groups. It is suggested to strengthen the emergency support capacity.

In response to question 4, based on the analysis and results of the first three questions, and based on the comprehensive analysis and research on the advantages and disadvantages of City1 and City2, the paper puts forward the main investment fields, capital investment plans and expected effects according to the specific development needs of each city. Investment in the city's existing advantages will be increased to further consolidate and expand its leading position in the construction of smart cities; For the disadvantages, it is committed to optimizing its shortcomings through precise capital investment and promoting the comprehensive improvement of the development level of smart cities.

Keywords: random forest, gradient lifting, K-means clustering, toughness assessment model

Content

1. Introduction

1.1 Background

As China's population ages and economic downward pressure increases, second- and third-tier cities face challenges in resource allocation, infrastructure, and social services. Achieving high-quality and sustainable development has become crucial. This study utilizes big data statistical analysis and machine learning models (such as random forests and decision trees) to evaluate urban service levels, analyze strengths and weaknesses, and identify key factors that drive urban resilience and sustainable development. At the same time, by examining the response capabilities to extreme weather and emergencies, the study proposes strategies and investment directions to optimize efforts, promote smart city construction, and enhance cities' ability to withstand risks. This research provides theoretical support for urban sustainable development policies and improving residents' quality of life.

2. Problem analysis

2.1 Analysis of question one

For question 1, the goal is to predict the future housing prices and existing housing stock for City 1 and City 2, based on the provided property sales information and other relevant data that can be obtained (such as population and GDP data). By combining property sales data scraped from the 58.com website, basic service POI data, and relevant economic indicators, a specific optimization scheme and modeling method are proposed. Through trend analysis of housing prices in different regions and an assessment of the impact of factors such as population and GDP, the aim is to provide a comprehensive forecast of the housing market dynamics for the two cities in the future.

2.2 Analysis of question two

For question 2, the task requires us to quantitatively analyze the services of two cities, extract common and unique characteristics, and identify their respective strengths and weaknesses. First, we will clean and standardize the data provided in the attachment. Then, we will apply the K-means clustering method to analyze the distribution, aggregation, quantity, coverage, and service type proportions of these facilities. By using visualization techniques and tabular data, we aim to compare the two cities.

2.3 Analysis of question three

For question 3, the objective is to assess the resilience and sustainability of the two cities under extreme weather and emergency scenarios. By analyzing the data of 15

service indicators provided, this paper makes a preliminary determination of the correlation between the various indicators, so as to identify the indicators that are crucial to measuring urban resilience and sustainability. This analysis will provide theoretical support and data basis for further assessment models of urban resilience.

2.3 Analysis of question four

Based on the analysis conclusions of questions 1 to 3, this section aims at the specific advantages and disadvantages of Changchun and Hohhot, defines the main investment directions and capital allocation, and predicts the improvement of the development level of smart cities brought by these investments. On the basis of maintaining the existing advantages of the city, it focuses on the optimization and improvement of the short board areas, so as to realize the comprehensive progress of the two cities in the construction of smart cities. The following contents systematically sort out and analyze the main investment areas, budget allocation and development expectations.

3. Symbol And Assumptions

3.1 Symbol Description

Symbol Definition	Symbol Description
D	the input feature matrix
y	the target variable
d_i, j	The j-th feature of the i-th sample
D_{num}	the numerical feature matrix
D_{cat}	the categorical feature matrix
K	Number of clusters
f_i	Each data point
c_j	Centroid of cluster C_j
C_j	Cluster assigned to data point x_i based on the closest centroid

X	A collection of four critical facility categories
X_1	Medical and Health Data
X_2	Public Facilities Data
X_3	Transportation Facilities Data
X_4	Government and Social Organizations Data
ω_i	Category of facilities The weight of X_i
$x_{i,j}$	express X_j the j the original value of the data point
$x_{i,min}, x_{i,max}$	express X_i the minimum and maximum values of the data
$x'_{i,j}$	express X_i the j normalized values for data points
S_i	express X_i the weighted score
S_{total}	the city's overall resilience score

3.2 Fundamental Assumptions

1. Data Integrity and Accuracy: It is assumed that the data used is accurate, complete, and representative of the actual conditions in the study area.
2. Consistency of Economic Development Trends: It is assumed that the city's economic development follows current trends and that no significant fluctuations or unforeseen policy changes will occur in the short term.

4. Model and Test the Models

4.1 Solution and Analysis of Problem 1

4.1.1 Modeling for Problem 1

Mathematical Model and Algorithm — Urban Housing Market Prediction Based on Regression Models

In the prediction of urban housing markets, multiple key factors influence the future trends of housing prices and existing housing stock. A regression analysis model is used to predict the urban housing market by incorporating different features. Through the training and evaluation of various regression models, the performance of different models in different cities and regions is comprehensively considered, providing a forecast for the future housing market.

1) Model Framework

Model Selection: Three regression models are used for prediction: Linear Regression, Random Forest Regressor, and Gradient Boosting Regressor. These models learn the relationship between features and the target variable from the training data, and then predict housing prices.

Preprocessing of missing and anomalous data.

2) Symbol Definition and Explanation

Features and Target Variable

D：Represents the input feature matrix (containing the values of all features), where each row represents a sample (a property or a region), and each column represents a feature.

y：Represents the target variable, i.e., housing price.

The elements of the feature matrix DDD are represented as:

d_i, j：The j-th feature of the i-th sample.

Numerical Features and Categorical Features

D_{num} ：Represents the numerical feature matrix, which includes numerical features such as "total number of households", "greening rate", and "floor area ratio".

D_{cat}：Represents the categorical feature matrix, which includes categorical features such as "building type," "property type," and "region code."。

3）Data Processing and Model Training

Feature Selection: Eleven features related to housing prices were selected from the dataset: total number of households, greening rate, floor area ratio, building type, property management fee (/m²/month USD), above-ground parking fee (/month USD), underground parking fee (/month USD), property type, longitude, latitude, and region code.

Data Standardization: The Standard-Scaler was used to standardize the numerical features, ensuring that the feature values are on the same scale.

Categorical Feature Encoding: The One-Hot-Encoder was used to encode categorical features (such as building type, property type, and region code), converting them into a format that can be accepted by machine learning models.

4）Regression Models

Linear Regression: The data is fitted using the least squares method to obtain the linear relationship between features and housing prices.

Random Forest Regression: Housing prices are predicted non-linearly by constructing an ensemble of decision trees.

Gradient Boosting Regression: The prediction accuracy is improved by progressively optimizing the loss function using the gradient boosting algorithm.

5）Model Evaluation

The performance of each model is evaluated using the Root Mean Squared Error (RMSE). A smaller RMSE indicates better prediction performance of the model. The RMSE is calculated using the following formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{\text{true},i} - y_{\text{pred},i})^2} \quad 4.2$$

Where:

$y_{true,i}$ is the actual values，

$y_{Pred,i}$ is the predicted value，

n is the number of samples.

6）Model Training and Prediction Process

First, the data is split using train-test-split, with 80% of the data used for training and 20% used for testing. For each region in each city, the following steps are performed:

- The features are processed by selecting valid features and performing standardization and encoding.
- The three regression models mentioned above are used for training, and the housing prices in the test set are predicted.
- The RMSE for each model on the test set is calculated, and the prediction results are saved.

Mathematical Model and Algorithm — Housing Stock Estimation Based on Formula Derivation

Model Establishment

For the housing data of the two cities (City1 and City2), a systematic estimation of the existing housing stock was performed based on multi-dimensional data cleaning and calculation methods. The specific steps are as follows: First, two data files were read and merged into a complete dataset. Data preprocessing was conducted by filling missing values and parsing key fields. Missing values for the "total number of households" and "floor area ratio" fields were filled with the mean value, and the "parking spaces" field was parsed into the ratio of above-ground and underground parking, then normalized to calculate the total parking ratio, ensuring the integrity and reliability of the data. After data cleaning, the total building area, average housing area per household, and housing stock were calculated based on formulas. The total building area was calculated using the inverse relationship between the total number of

households and the floor area ratio. The average housing area per household was obtained by dividing the total building area by the total number of households. Finally, by combining the parking ratio, the housing stock for each community was estimated. Based on this, housing stock for each of the two cities was calculated by summing the results for each city grouped by city code. Construct the formula based on the available data:

a) Calculation of Total Building Area:
Based on the floor area ratio, the formula for calculating the total building area is:
Total floor area=Total number of households×（1/Floor area ratio）4.2

b) Calculation of Average Housing Area per Household：
Calculate the average housing area per household based on the total building area and total number of households:
Average household area= Total floor area /Total number of households4.2

c) Calculation of Parking Space Ratio：
By adding the above-ground and underground parking space ratios, the total parking space ratio for the community is calculated:
Total parking ratio=Parking ratio ground+Parking ratio underground4.2

d) Calculation of Total Housing Stock：
By combining the total number of households, average housing area per household, and the total parking space ratio, the housing stock is obtained:
Total housing stock=Total number of households × Average household area × Total parking ratio4.2

4.1.2 Results Presentation for Problem 1

Basic Data Visualization

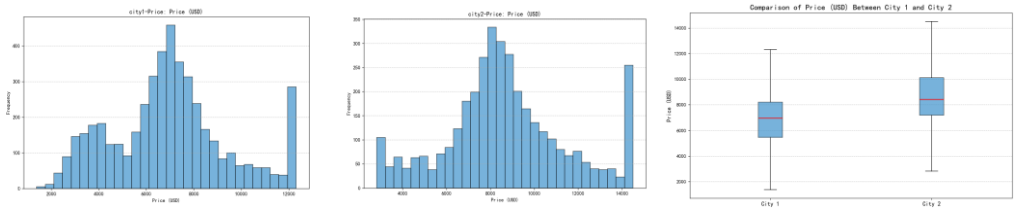


Figure 1 Housing Price Distribution for City 1 and City 2

The housing price distributions in City 1 and City 2 show a significant concentration, with prices primarily ranging from 6000 to 9000, indicating the relative stability of housing prices in both cities. This concentration trend suggests that there is a dominant price range, reflecting an adequate supply of mid-priced properties in the market. The box plots further reveal the interquartile range of housing prices and potential outliers. The price volatility in City 1 is notably higher than in City 2, with a few peaks in the high-price ranges of both cities: high-priced properties in City 1 are concentrated around 12000, while those in City 2 are around 14000. These high-priced properties likely represent high-end residential areas or luxury communities, further highlighting the presence of price segmentation in the housing market.

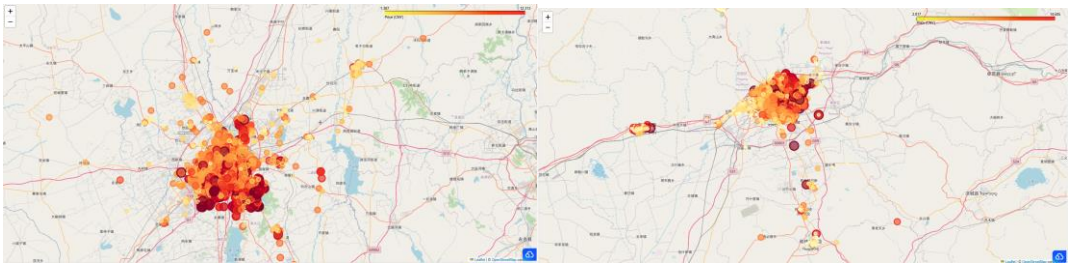


Figure 2 Housing Price Distribution for City 1 and City 2

From the geographical distribution map, it is clear that housing prices exhibit a significant clustering effect, which is observed in both City 1 (Changchun) and City 2 (Hohhot). In City 1, high housing prices are mainly concentrated in the city center, while areas farther from the city center have relatively lower prices. This is consistent with the fact that urban centers typically offer better public resources, such as commercial, educational, and healthcare facilities. In contrast, in City 2, high-priced properties are primarily located in the northern areas, especially near major transportation hubs, indicating that accessibility to transportation may be an important factor influencing housing prices in City 2. This spatial clustering effect of housing prices suggests that geographic location plays a significant role in shaping housing prices in both cities. Therefore, geographic location can be considered a key feature in the construction of housing price prediction models. Specifically, variables such as the distance from the property to the city center, transportation hubs, or key landmarks can be introduced to quantify the impact of geographic location on housing prices. In summary, the housing price distribution and geographic clustering characteristics in both City 1 and City 2 reveal the segmentation of the real estate market, as well as the significant impact of geographic location and accessibility to public facilities on housing prices. This not only helps to understand the current market structure but also provides clear feature directions for model construction.

Correlation Analysis Results Presentation

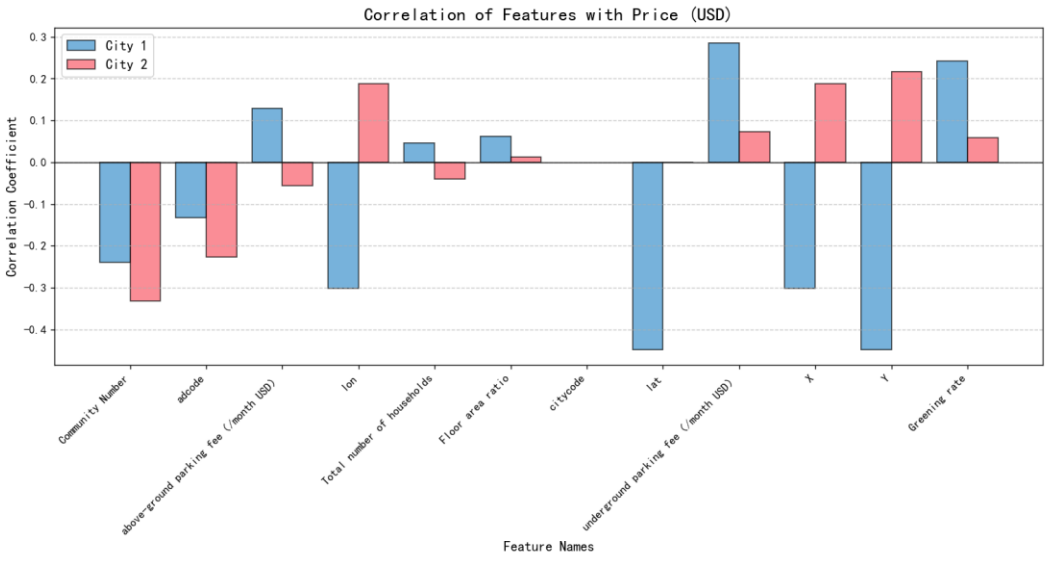


Figure 3 Correlation Analysis of Various Indicators and Housing Prices for City 1 and City 2

This figure displays the correlation coefficients between various features and housing prices (in USD) for City 1 (blue) and City 2 (red), showing the similarities and differences between the two cities in terms of factors influencing housing prices. For City 2, greening rate and floor area ratio show a strong positive correlation with housing prices, indicating that areas with better greening conditions or higher land use efficiency tend to have higher housing prices. This trend is relatively weaker in City 1. In contrast, latitude and total number of households show a negative correlation with housing prices in both cities, with the negative correlation between and housing prices being more significant in City 1. This suggests that in City 1, areas located further north or regions with a higher number of households tend to have lower housing prices. Additionally, underground parking fees have a similar impact on housing prices in both cities, indicating that parking-related costs play a certain role in the housing price structure. Overall, housing prices in City 2 are more influenced by environmental and planning factors (such as greening rate and floor area ratio), while housing prices in City 1 are more dependent on geographical location (such as latitude and longitude) and population distribution (such as the number of households).

Housing Price Prediction Results

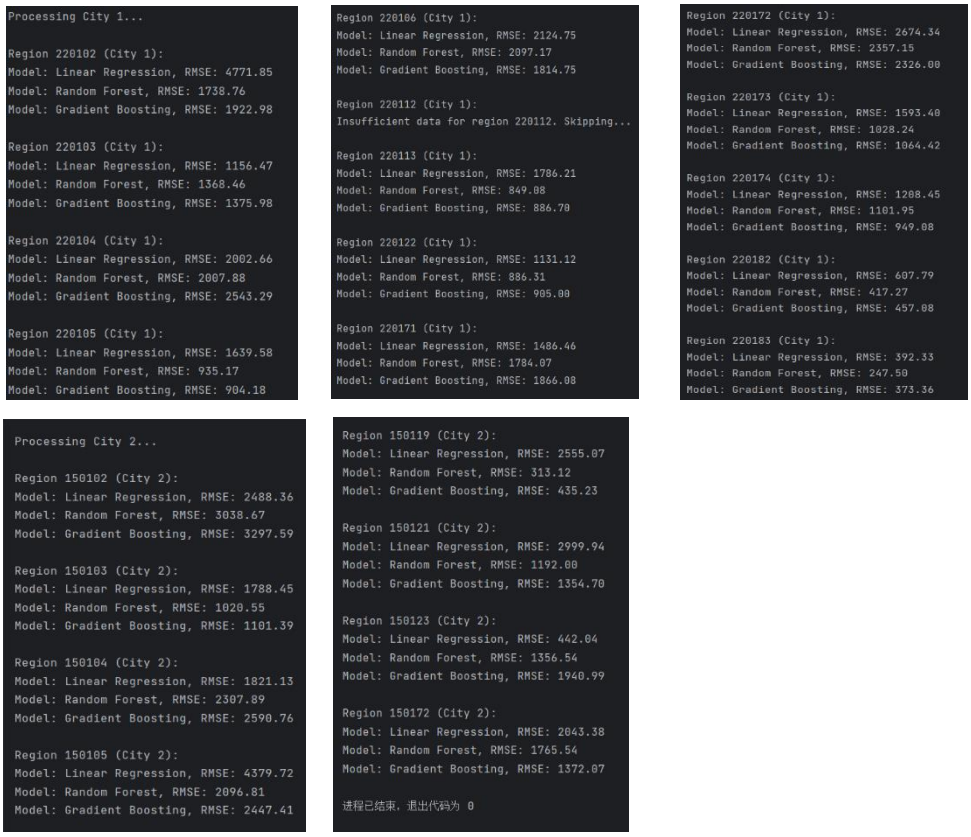


Figure 4 Display of Housing Price Prediction Results for City 1 and City 2

Comparison of Housing Price Prediction Performance Across Multiple Models

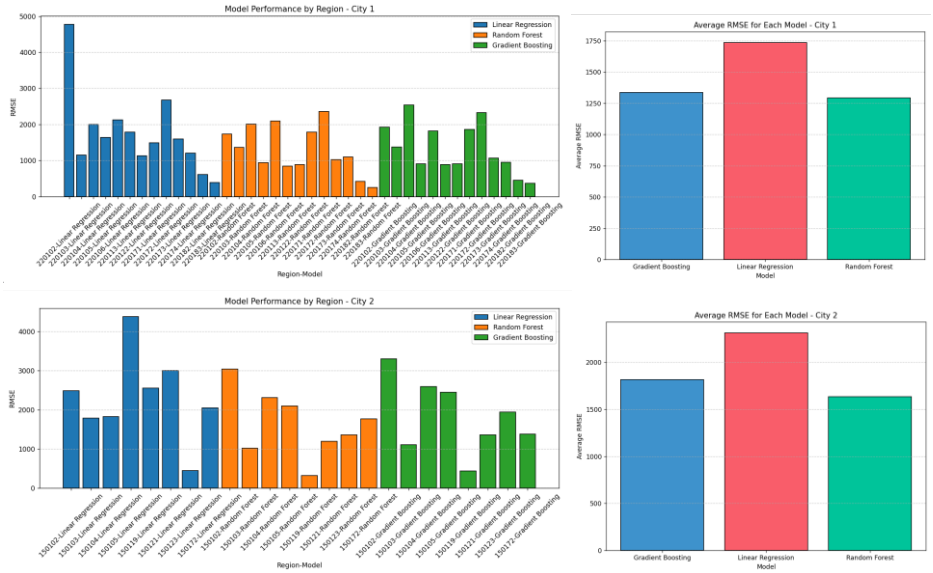


Figure 5 RMSE and Average RMSE of Housing Price Predictions for Each Model and Region in City 1 and City 2

Visualization of the Difference Between Actual and Predicted Housing Prices (Partial)

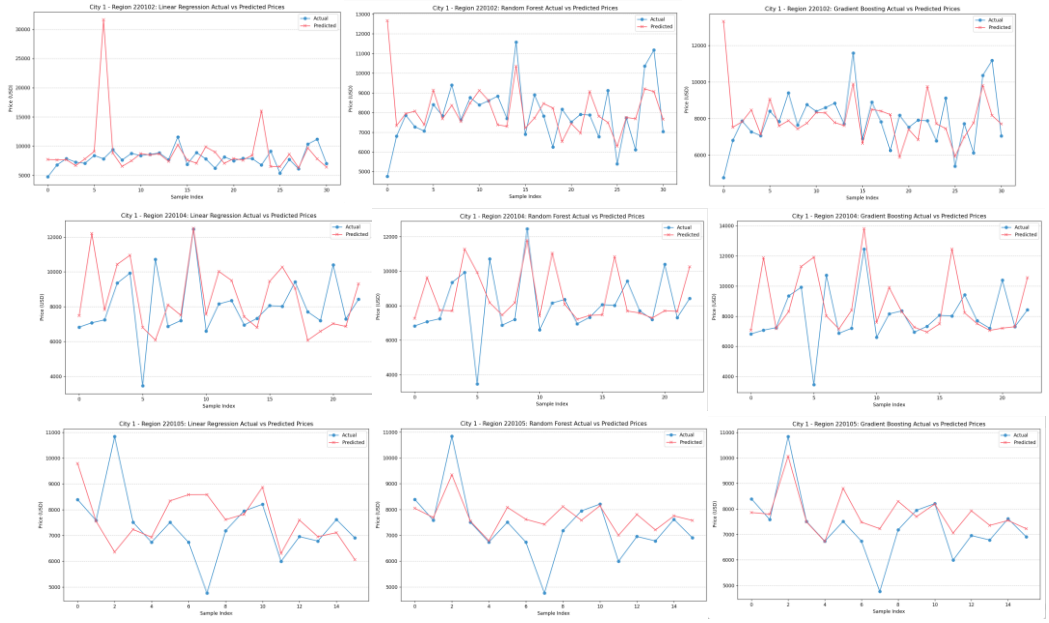


Figure 6 Comparison of Predicted and Actual Values

The results show that in the regional model performance bar chart for City 1, there is a significant variation in model performance across different regions. In some regions, the linear regression model performs poorly, with RMSE values even exceeding 4000, while the Random Forest and Gradient Boosting models show relatively stable performance in most regions, with small fluctuations in RMSE. From the average RMSE bar chart for City 1, it can be seen that the average RMSE for both Gradient Boosting and Random Forest models is very close, and significantly lower than that of the Linear Regression model. This indicates that these two nonlinear models outperform Linear Regression in overall prediction performance for City 1. In the regional model performance bar chart for City 2, a similar trend is observed, but Random Forest and Gradient Boosting models show significantly better performance

in certain regions compared to City 1, with their RMSE values much lower than that of Linear Regression. From the average RMSE bar chart for City 2, it can be seen that the Random Forest model slightly outperforms the Gradient Boosting model, with both models significantly outperforming Linear Regression. This further illustrates that nonlinear models are better able to capture the characteristic relationships in City 2, making them more suitable for housing price prediction in City 2.

Comprehensive analysis shows that both the Random Forest and Gradient Boosting models demonstrate strong performance in housing price prediction for City 1 and City 2, effectively adapting to the feature differences and complex relationships across different regions. In contrast, the Linear Regression model, due to its stricter assumptions, struggles to accurately capture complex feature relationships and performs poorly in most regions. In the comparison between City 1 and City 2, the model performance in City 2 exhibits more significant fluctuations between regions, suggesting that the feature data for City 2 may be more complex or the regional differences more pronounced. In future housing price prediction tasks, it would be more appropriate to prioritize the Random Forest or Gradient Boosting models. These two models not only exhibit lower error rates but also effectively capture the complex feature differences between regions, providing a more reliable basis for policy planning, market analysis, and housing price forecasting.

```
City1 : 1459933.97
City2 : 834249.01

进程已结束，退出代码为 0
```

Figure 7 Housing Stock Estimation Results

The results show that the estimated housing stock for City 1 is 1,459,933.97, while the estimated housing stock for City 2 is 834,249.01. This result indicates a certain disparity in the existing housing resources between the two cities, providing important data support for subsequent urban planning, resource allocation, and housing supply policies.

4.2 Solution and Analysis of Problem 2

In Problem 2, the goal is to conduct a quantitative analysis of the service levels across different industries in City 1 and City 2. For such data, the quantitative analysis can be approached by examining:

- The number of services or facilities provided in each domain (e.g., healthcare, education, etc.).
- Service quality metrics, such as customer satisfaction, service efficiency, or response time.

From Attachments 3 and 4, the data includes information about various facilities in both cities, such as administrative region distributions, facility types, and geographical coordinates. However, no data regarding service quality ratings or metrics was collected. Therefore, this analysis will focus on the distribution of service facilities.

We propose using the K-means clustering method to analyze and visualize the facilities of City 1 and City 2 based on geographical distribution and type dimensions. This approach helps identify spatial distribution patterns and clustering characteristics of service facilities. Furthermore, combining density analysis and spatial accessibility analysis, the number of service facilities in each administrative region can be used to calculate service coverage rates and service type proportions. By visualizing and comparing the facilities in both cities, we aim to extract commonalities and unique features, as well as identify their respective strengths and weaknesses.

4.2.1 K-means Clustering Model

1. Objective

Using K-means clustering, group all service facilities in the table based on their geographical coordinates and type codes to reveal potential spatial distribution patterns and clustering characteristics of service types within the data.

Formula：

$$\min \sum_{j=1}^{K} \sum_{x_i \in c_j} \left\| f_i - c_j \right\|^2 \tag{4.2}$$

Where:

K: Number of clusters.

f_i: Each data point.

c_j: Centroid of cluster C_j.

C_j: Cluster assigned to data point x_i based on the closest centroid.

$\left\| f_i - c_j \right\|^2$: Squared Euclidean distance between data point f_i and centroid c_j.

2. Data Preparation

（1）Data Preprocessing: Clean and transform the data from Attachments 3 and 4 Ensure column formats are correct and handle missing values (fill or remove as appropriate).

（2）Feature Selection: Select three features from the data: lon_gcj02 (longitude), lat_gcj02 (latitude), and typecode (service type).

（3）Data Standardization: Standardize the data to adjust all features to the same scale and prevent discrepancies in feature values from affecting clustering results.

3. Model Construction

（1） Determining the Number of Clusters (K): Initially set K to be equal to the number of administrative regions.

（2） Model Training:

a) Initialization: Randomly select K points based on the number of administrative regions as initial cluster centers.

b) Data Point Assignment: Assign each data point to the nearest cluster based on Euclidean distance.

c) Update Cluster Centers: Compute the average position of all points within each cluster to update the cluster centers.

d) Iterative Updates: Repeat the assignment and update steps until the cluster centers stabilize or the maximum number of iterations is

reached.

4. Model Output

   Cluster Labels: Each data point is assigned a cluster label (cluster), indicating the group to which it belongs.

5. Cluster Analysis and Visualization

4.2.2 Results Presentation for Problem 2

Using Python code, we visualize the clustering results, mapping the distribution of service facilities as shown below. The left map represents the clustering results from Attachment 3, and the right map represents the clustering results from Attachment 4. (The following is an example diagram; the complete version can be found in Appendix A.)

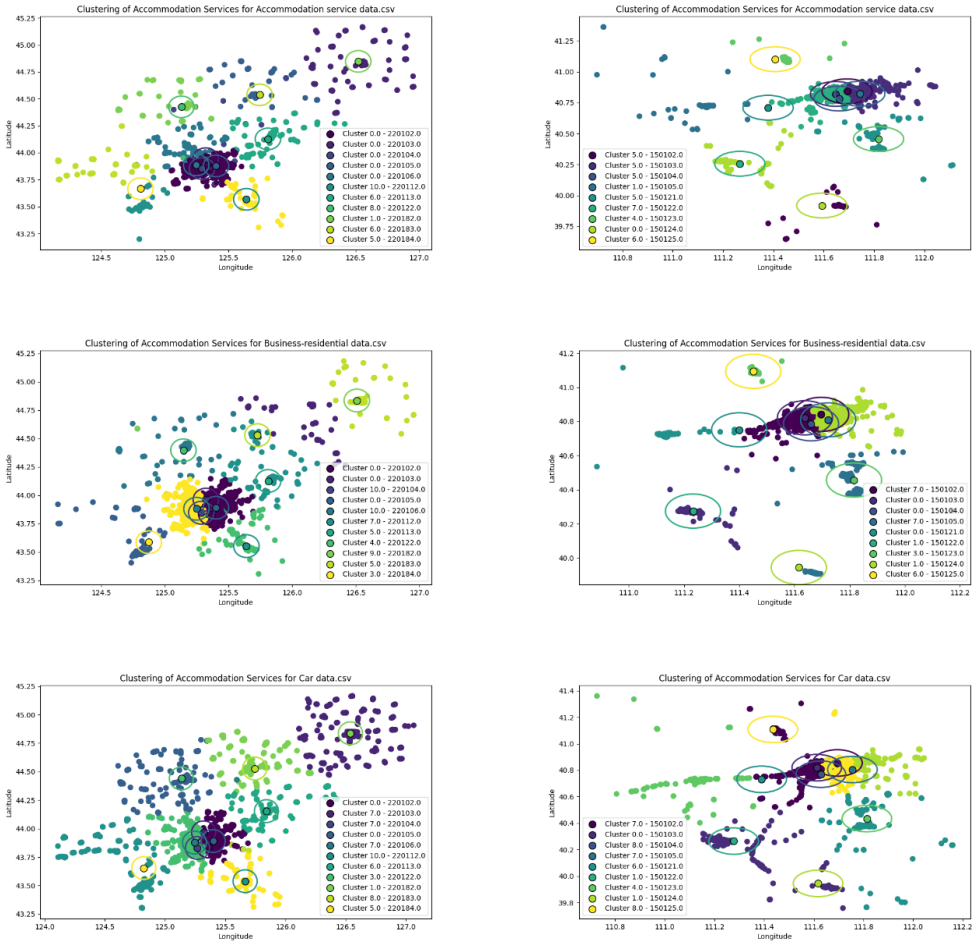


Figure 8 City 1、2 Facility Distribution

By analyzing the services included in both cities from the attachments, we obtained the clustering results above. Observing the results, it can be seen that the distribution of various facilities in City 1 is relatively uniform compared to City 2. In contrast, City 2 exhibits high-density clusters in many areas but has a relatively low overall citywide coverage, especially for categories such as Accommodation and Business.

The clustering results effectively reflect the density of facility distributions, but

they are not very intuitive in terms of facility quantities. Therefore, we performed a comparison of the quantities and densities of different facilities, which provides a clearer visualization of the differences between the two cities. The table below shows the facility quantity and density analysis for the two cities.

Table1 Quantity and Density Comparison

file_name	num_facilities（city1）	num_facilities（city2）	facility_density（city1）	facility_density（city2）
Accommodation service	4399	2457	0.178097	0.142849
Business-residential	7523	4056	0.304575	0.235814
Car	10135	4284	0.410324	0.24907
Finance and insurance	4792	2199	0.194008	0.127849
Food and beverage service	46049	22339	1.864332	1.298779
Geographical name and address information	57339	22656	2.321417	1.317209
Government and social organizations	13235	5957	0.53583	0.346337
Lifestyle service	38533	19398	1.56004	1.127791
Medical and health	16647	6024	0.673968	0.350233
Motorbike	691	159	0.027976	0.009244
Public facilities	1915	2153	0.07753	0.125174
Retail service	76444	41380	3.094899	2.405814
Science, education, and culture	15731	5487	0.636883	0.319012
Transportation facilities	14502	4553	0.587126	0.264709

From the table, it can be observed that in categories such as Geographical name and address information, Car, Medical and health, Retail service, Motorbike, and Science education and culture, City 1 has significantly higher quantities and densities compared to City 2.

4.2.3 Common and Unique Characteristics of the Two Cities

Common Characteristics:

1. Diverse Range of Services:

Both cities provide a wide variety of facilities, including accommodation, retail services, medical and health services, education, and government organizations.

2. Clustering Patterns:

The clustering results indicate that both cities have areas with dense service distributions, mainly concentrated in urban centers.

Unique Characteristics:

1. City 1:

Even Distribution:

Service facilities in City 1 are more evenly distributed, resulting in better overall coverage.

Higher Facility Density:

City 1 has significantly higher quantities and densities in the following key categories:

- Geographical Name and Address Information
- Car Services
- Medical and Health
- Retail Services
- Motorbike Services
- Science, Education, and Culture

Better Accessibility:

The even distribution ensures that facilities in City 1 are more accessible to residents across different regions.

2. City 2:

High-Density Clusters:

Service facilities in City 2 are highly concentrated in certain areas, such as accommodation and business services.

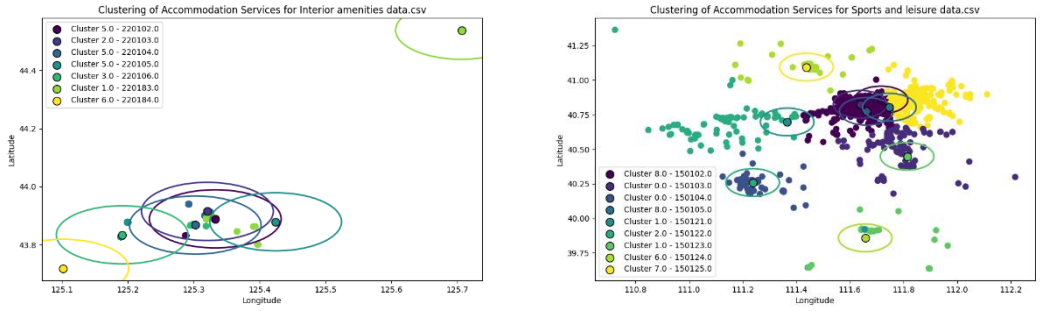
Lower Coverage:

Despite high-density clusters in specific regions, the overall citywide coverage of facilities in City 2 is relatively low, leaving some areas underserved.

In addition, the two cities have distinct service categories:

City 1 offers Interior Amenities, which focus on enhancing the convenience of residential living spaces.

City 2 features Sports and Leisure facilities, providing residents with more options for recreation and fitness.



City 1: Interior Amenities City 2: Sports and Ieisure facilities

Figure 9 Unique services of the two cities

4.2.4 Strengths and Weaknesses of the Two Cities

1. City 1:

Strengths:

Service facilities are evenly distributed, ensuring equitable access across all areas of the city.

Higher density in key sectors supports a diverse range of needs citywide.

Weaknesses:

Lacks prominent high-density zones in specific service sectors, such as business or tourism, which might reduce its appeal to certain industries.

Due to limited data, City 1 may offer fewer options for recreational and fitness facilities for residents.

2. City 2:

Strengths:

High-density clusters in specific sectors, such as accommodation and business services, attract related industries and traffic.

Weaknesses:

Uneven distribution of facilities leaves peripheral areas underserved.

In several key categories, facility quantity and density are lower than in City 1, limiting its service capacity.

In conclusion, City 1 excels in the balance and coverage of service facilities, ensuring accessibility and equitable resource distribution. On the other hand, City 2 demonstrates strengths in the high-density development of specific sectors, which can attract targeted industries. However, City 2's overall facility coverage is low, with underdeveloped areas that require improved resource allocation to enhance service levels.

4.3 Solution and analysis of problem 3

4.3.1 Problem 3 Analysis

For question 3, the objective is to assess the resilience and sustainability of the two cities under extreme weather and emergency scenarios. By analyzing the data of 15 service indicators provided, this paper makes a preliminary determination of the correlation between the various indicators, so as to identify the indicators that are crucial to measuring urban resilience and sustainability. This analysis will provide

theoretical support and data basis for further assessment models of urban resilience.

4.3.2 Problem 3 Modeling

Mathematical model and algorithm: urban resilience assessment method based on multi-index weighting. This paper proposes an urban resilience assessment model based on multi-index weighting. By calculating the weighted scores of four types of key facilities, the city's emergency response and resilience can be comprehensively evaluated.

1) Model framework

Weight allocation: Weight allocation of various facilities based on expert opinions or data analysis.

Standardized processing: The original data of each type of facility was normalized using Min-Max standardization method.

Weighted score calculation: According to the standardized values of facilities and their corresponding weights, the weighted scores of various facilities are calculated, and the overall resilience score of the city is obtained comprehensively.

2) Symbol definition and description

$X = \{X_1, X_2, X_3, X_4\}$ A collection of four critical facility categories, including:

3) Weight allocation

Data analysis was used to determine the relative importance of each type of facility in resilience assessment. Pearson correlation coefficient was used to measure the correlation between facility types and resilience indicators. The formula is as follows:

$$\rho(X_i, Y) = \frac{\mathrm{cov}(X_i, Y)}{\sigma_{X_i}\sigma_Y} \tag{4.3}$$

Among them:

X_i Data for the facility category;

Y Target variables for resilience assessment (e.g. speed of recovery after disaster, public safety index, etc.);

The powers of establishment are:

4) Data standardization

In order to eliminate the impact of data magnitude differences between different facility categories, the Min-Max standardization method is used to combine raw data $x_{i,j}$ Convert to standardized values within a uniform [0, 1] range $x_{i,j}^{'}$， The formula is as follows:

$$x_{i,j}^{'} = \frac{x_{i,j} - x_{i,\min}}{x_{i,\max} - x_{i,\min}}, \forall j \in \{1, 2, ..., n_i\} \tag{4.4}$$

Among them:

$x_{i,j}$ is X_j the j the original value of the data point， $i \in \{1,2,3,4\}$, $j \in \{1,2,...,n_i\}$ 。

$x_{i,\min}, x_{i,\max}$ Category of facilities X_i the minimum and maximum values of the data in,

respectively, are used for standardization processing.

5) Weighted score calculation

For each category X_i, Weighted score S_i The calculation formula is as follows:

$$S_i = \omega_i \cdot \frac{1}{n_i} \sum_{j=1}^{n_i} x_{i,j}^{'} \tag{4.5}$$

Among them:

ω_i express X_i weight;

n_i express X_i The total number of data points in;

6) Overall resilience score

The city's overall resilience score S_{total} is the sum of the weighted scores of all facilities categories, calculated as follows:

$$S_{total} = \sum_{i=1}^{4} S_i = \sum_{i=1}^{4} \left(\omega_i \cdot \frac{1}{n_i} \sum_{j=1}^{n_i} x_{i,j}^{'} \right) \tag{4.6}$$

Among them:

S_{total} The final urban resilience score indicates the city's overall ability to cope with extreme weather and unexpected events.

S_i Weighted scores for each category.

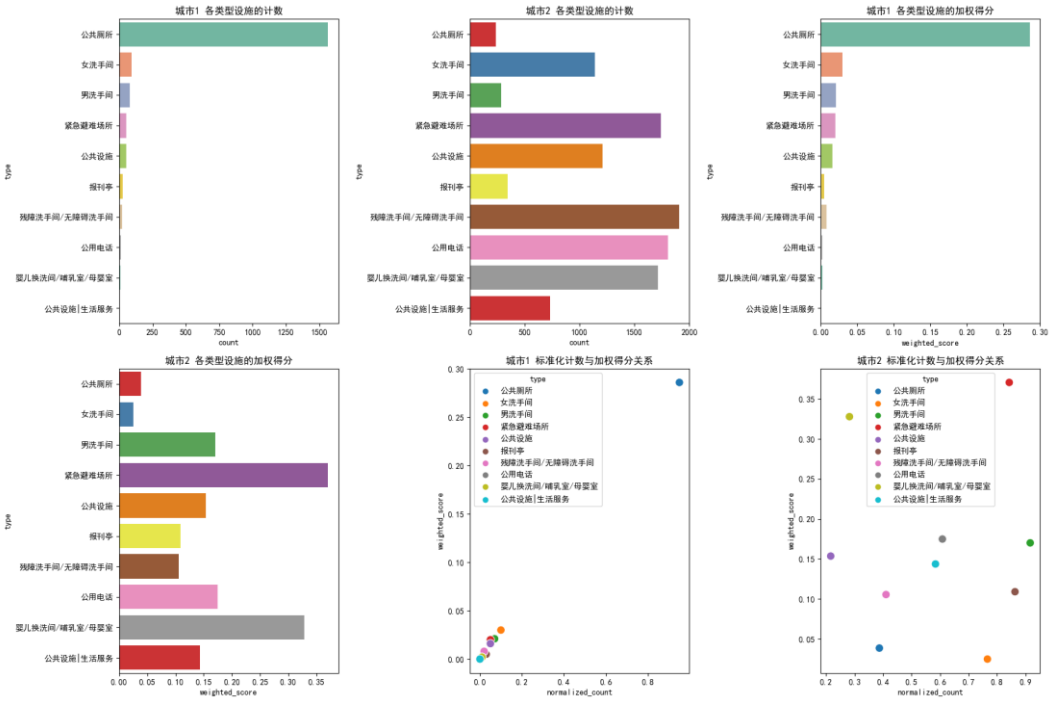


Figure 10 Comparison of urban resilience indicators

4.3.3 Question 3 Result presentation

City 1, 2 government and social organization comparison

The quantitative analysis results of the data of city 1 and 2 governments and social organizations show that: based on the importance of various organizations in emergency and emergency management, the following weights can be divided. Some organizations with very small amounts of data (only 1) were not considered in this analysis.

1. Core government departments: Due to their direct involvement in the management and response to emergencies, these departments have a weight of 0.4.

2. Auxiliary government agencies: They play a significant role in organizational coordination and management support, with a weight of 0.3.

3. Social organizations and other groups: These organizations played an important role in the recovery and assistance after the incident, with a weight of 0.2.

4. Other institutions and affiliated organizations: their influence in emergencies is relatively limited, and the weight is set at 0.1.

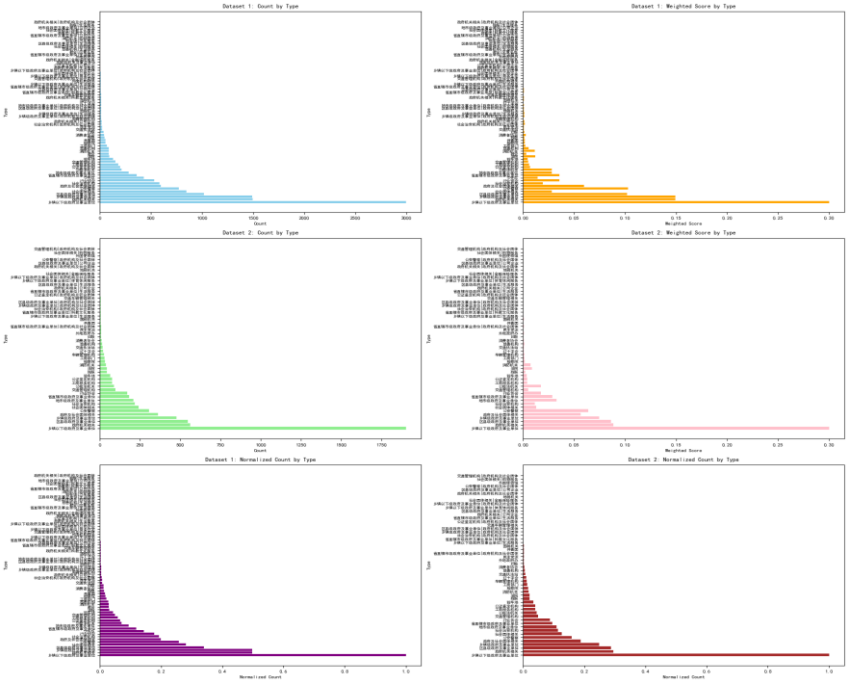


Figure 11 Comparison of government and social organizations in cities 1 and 2

Comparison of medical service data in cities 1 and 2

The role of medical services in public health emergencies is critical, and the following are weighted based on the importance of their functions:

1. Core medical institutions with a weight of 0.4.
2. Specialized medical institutions with a weight of 0.3.
3. Basic medical facilities with a weight of 0.2.
4. Other medical related institution with a weight of 0.1.

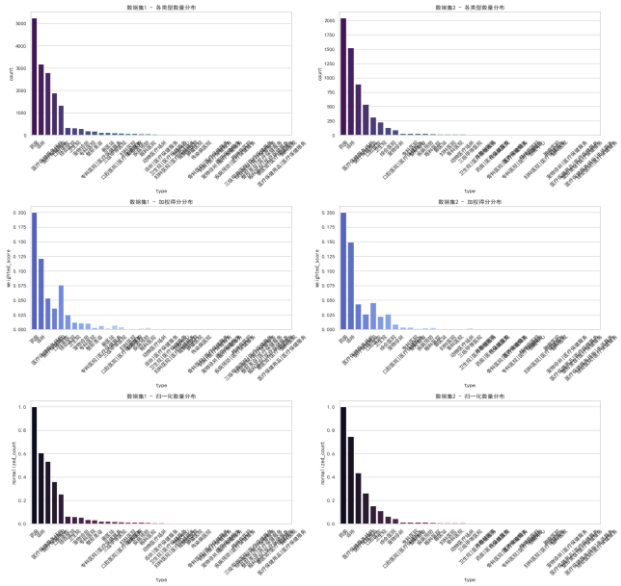


Figure 12 Comparison of city 1 and 2 medical service data

City 1, 2 public facilities data comparison

The role of public facilities in cities 1 and 2 in emergencies can be divided into the following weights according to the importance of their functions:

Emergency shelters with a weight of 0.4.

Basic sanitation facilities with a weight of 0.3.

Other services with a weight of 0.2.

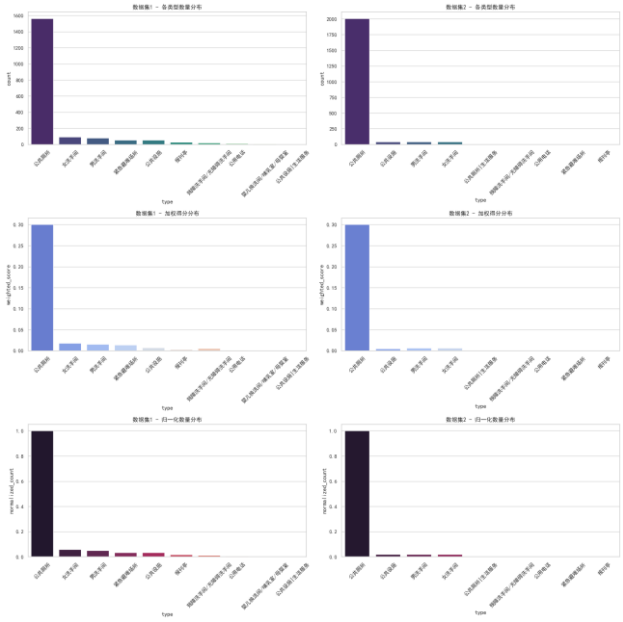


Figure 13 Comparison of urban 1 and 2 public facilities data

Comparison of urban 1 and 2 traffic facilities data

In response to emergencies, the functional importance of various types of facilities is different. The following is the importance weight division scheme for different types of facilities:

1. Public transport facilities with a weight of 0.4.

2. Parking facilities with a weight of 0.3.
3. Other transportation service facilities with a weight of 0.2.

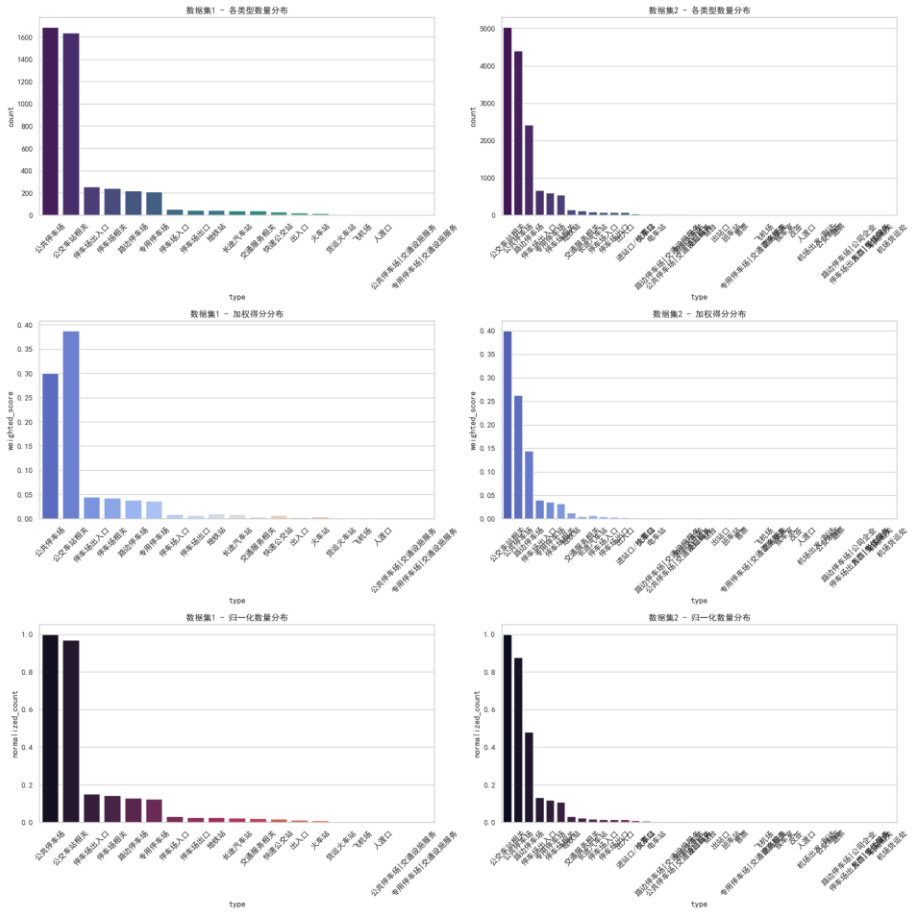


Figure 14 Comparison of city 1 and 2 traffic facilities data

4.3.4 Result analysis of problem 3

According to the weighted calculation, Changchun scored 0.74, slightly higher than Hohhot's 0.67, indicating that Changchun performed better in terms of overall resilience.

1. Result analysis

Changchun's score advantage: Changchun has outstanding performance in the allocation of emergency management resources and the coverage rate of government agencies. In particular, the number and importance of core emergency departments (such as public security, fire protection, etc.) are superior, which enables Changchun to quickly organize effective response measures in the face of emergencies. This systematic and balanced distribution of resources provides the basis for its high score.

Hohhot's challenges: Hohhot's slightly lower score reflects its shortcomings in a number of key areas. For example, the distribution and accessibility of emergency resources at the grass-roots level is limited, particularly in terms of traffic management facilities or community-level emergency response capacity. In addition, the small number of social good organizations (e.g., Red Cross, charities, etc.) also reduces their emergency support and mobilization capacity.

2. Short board analysis

Changchun's weaknesses: Despite Changchun's overall high score, there is still room for improvement in the diversity of social support systems and the mobilization capacity of grass-roots organizations. For example, the coverage of community volunteer networks and special emergency facilities remains inadequate. These improvements will further enhance the flexibility and overall resilience of its emergency response.

Hohhot's weaknesses: Hohhot has weaknesses in core emergency management infrastructure, such as traffic dispersal and community shelter facilities. At the same time, the participation of social groups is low, such as the insufficient coverage of industry associations and charitable organizations, resulting in limited mobilization of volunteer forces in emergencies, weakening their overall resilience.

3. Key areas for future development

Changchun: Strengthen grass-roots emergency facilities: increase community-level shelters and small medical stations, and improve grass-roots emergency response capacity. Enhance the participation of social welfare: Promote the development of non-governmental organizations through incentive measures to increase the depth of participation of social forces in emergencies.

Hohhot: Improve the emergency management network: expand the coverage of core facilities such as fire stations and traffic evacuation centers. Promote the diversity of social groups: support the development of different types of social welfare organizations through policies to improve the effectiveness of volunteer forces in emergencies.

4. Investment Plan

Changchun, Short-term plan: Strengthen community emergency facilities, such as emergency shelters and community first aid centers. Provide financial incentives for social projects and attract more ngos to participate in emergency management.

Long-term plan: Establish a comprehensive public health surveillance network to improve the ability to identify and respond to public health crises. Develop an intelligent traffic management system to ensure efficient traffic scheduling in emergencies.

Hohhot, Short-term plan: Increase grass-roots emergency coverage, especially in community disaster shelters and traffic management infrastructure. Support social groups with special funds to enhance their participation in emergency response.

Long-term plan: Promote the normalization of disaster prevention education in communities, and enhance residents' awareness of disaster prevention and self-rescue ability. Improve transportation infrastructure and increase the efficiency of material transportation and mass evacuation.

4.4 Solution and analysis of problem 4

Based on the analysis conclusions of questions 1 to 3, this section aims at the specific advantages and disadvantages of Changchun and Hohhot, defines the main investment directions and capital allocation, and predicts the improvement of the development level of smart cities brought by these investments. On the basis of maintaining the existing advantages of the city, it focuses on the optimization and

improvement of the short board areas, so as to realize the comprehensive progress of the two cities in the construction of smart cities. The following contents systematically sort out and analyze the main investment areas, budget allocation and development expectations.

4.4.1Main investment areas

Health care facilities

Investment content: Prioritize the establishment of general hospitals, emergency centers and community medical service points within the city, and improve the layout of medical facilities. At the same time, smart medical technologies, such as telemedicine and intelligent diagnosis, are introduced to alleviate the uneven distribution of medical resources and improve the city's medical service capabilities.

Budget ratio: about 30%-40% of the total budget.

Expected results: Changchun can significantly improve medical accessibility by optimizing the allocation of medical resources, especially expanding the medical service network to the marginal areas. Hohhot plans to fill gaps in some areas lacking medical resources and enhance its ability to respond to public health emergencies.

4.4.2. Investment and implementation phase

Short-term plan (1-3 years): Objective: To quickly address the shortage of existing facilities, focusing on increasing the number of health service points, public facilities and transportation infrastructure. Funding source: local financial appropriation and smart city special fund. Key projects: construction of community shelters, expansion of clinics and initial development of intelligent traffic management system.

Medium-term Plan (3-5 years): Objective: To comprehensively improve the smart city platform and promote the landing of real-time monitoring, prediction and emergency dispatch functions. Funding source: PPP model (public-private partnership) or government-enterprise partnership investment. Key projects: the improvement of smart medical platform, the expansion of smart traffic management system and the construction of government emergency center.

Long-term plan (5-10 years): Objective: To realize the digital and intelligent management of infrastructure, and comprehensively improve the management capability of smart cities. Source of funds: Rolling investment mechanism, installment allocation to ensure long-term capital needs. Key projects: Construction and data integration of intelligent emergency management system, covering intelligent management of transportation, medical care and public services.

4.4.3 Expected effect and smart city development

Health care: Through the addition of medical facilities and smart medical technology, the accessibility of urban medical services has been improved by more than 25%, and the rescue response time in emergencies has been reduced by 30%.

Public services: The coverage and utilization rate of public service facilities has increased significantly, and residents' satisfaction with facilities is expected to increase by 20%.

4.4.4 Monitoring and feedback mechanism for plan implementation

Project monitoring: Periodically evaluate each project every year, record the

completion of key performance indicators (KPIs), and adjust resource allocation if necessary.

Resident feedback: Collect public opinions on smart city services through regular surveys of resident satisfaction to optimize project implementation and service quality.

5.Strengths and Weakness

5.1Advantage

1. Standardized processing to eliminate data differences: The Min-Max standardized method is adopted to normalize the data of various facilities, which can effectively reduce the deviation of the original data and improve the accuracy and fairness of the evaluation results.

2. K-means clustering is a simple, efficient, and widely used method that works well with large datasets. It is computationally efficient, as its time complexity is linear with respect to the number of data points and clusters. K-means also converges quickly, especially with good initial centroid selection, and provides clear cluster boundaries, making it easy to interpret and analyze.

5.2Shortcoming

1. Limitations of the standardized method: Although the Min-Max standardized method can eliminate the difference in data magnitude, it also has certain limitations.

2. K-means clustering has several limitations. It is sensitive to the initial selection of centroids, which can lead to suboptimal results if poorly initialized. The number of clusters (K) must be predefined, which can be challenging without domain knowledge or appropriate methods. Additionally, K-means assumes that clusters are spherical and equally sized, which may not always hold true, and it is sensitive to outliers, which can distort the results. It also struggles with categorical data unless transformed properly.

6.Conclusion

This study provides a comprehensive analysis of City1 and City2 in terms of housing market predictions, infrastructure distribution, urban resilience, and investment strategies. The results show that City1 outperforms City2 in housing stock, facility distribution, and overall resilience, particularly in emergency management resources and government agency coverage. However, City1 faces challenges in the diversity of social support systems and grassroots mobilization. City2, while limited in grassroots emergency resources and social participation, has room for improvement. Based on these findings, targeted investment strategies are proposed to enhance City1's strengths and address City2's weaknesses, aiming to foster smart city development and improve urban resilience. In the futrue, it is essential to continuously monitor the progress of these investments and adjust strategies according to evolving urban needs.

References

[1] Zhu Chunli. Research on Multi-Factor Coupling Modeling and Collaborative Control of Resilient Transportation Systems [D]. Tsinghua University, 2021. DOI: 10.27266/d.cnki.gqhau.2021.000164.

[2] Xia Chenhong, Ma Donghui, Guo Xiaodong, et al. Urban Disaster Prevention Spatial Operation Logic and Planning Optimization Path for System Resilience [J]. Urban Development Research, 2024, 31(09): 49-55+108.

[3] Fu Yuhang. Geological Disaster Susceptibility and Regional Disaster Prevention Resilience Evaluation in the Southeastern Hilly Area [D]. Hunan University, 2023. DOI: 10.27135/d.cnki.ghudu.2023.000762.

[4] Xia Chenhong, Ma Donghui, Guo Xiaodong, et al. Urban Disaster Prevention Spatial Operation Logic and Planning Optimization Path for System Resilience [J]. Urban Development Research, 2024, 31(09): 49-55+108.

[5] Wang Jiwu, Shen Yuyan, Mao Danyi, et al. Study on the Network Resilience of Urban Medical Facilities System and Planning Countermeasures: A Case Study of the Main Urban Area of Zhengzhou [J]. Urban Planning, 2024, 48(08): 90-100.

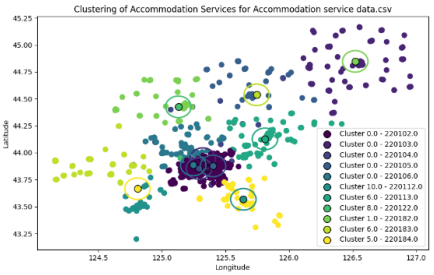
[6] Zhou Xiaoliang. Theoretical Interpretation, Formation System, and Path of Improving China's Economic Resilience [J]. Southeast Academic, 2024, (01): 127-140+248. DOI: 10.13658/j.cnki.sar.20240002.010.

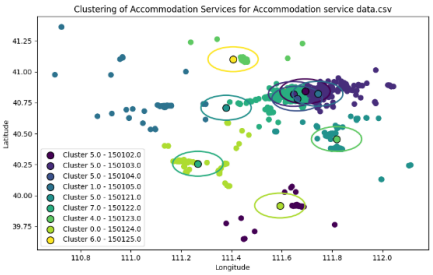
[7] Chen Hongmei, Cai Songlin. Resilience Evaluation and Spatiotemporal Evolution Analysis of Regional Digital Innovation Ecosystems [J]. Statistics and Decision, 2023, 39(20): 51-55. DOI: 10.13546/j.cnki.tjyjc.2023.20.009.

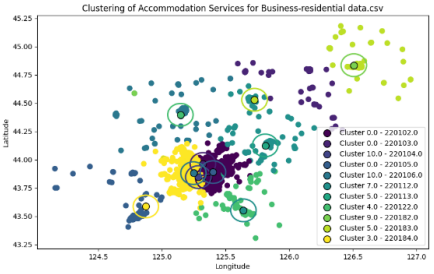
[8] Liu Hedong, Lu Chenxi. The Impact of Innovation Ecosystem Resilience on High-Quality Economic Development [J]. China Science and Technology Forum, 2023, (01): 48-57. DOI: 10.13580/j.cnki.fstc.2023.01.005.

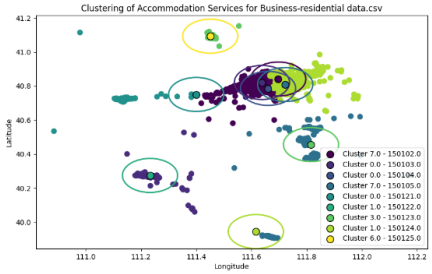
Appendix

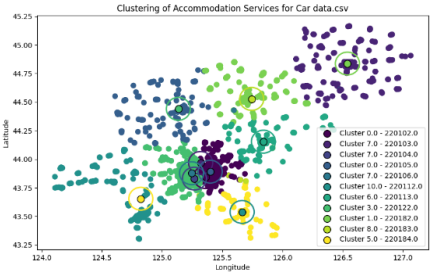
附录 A：

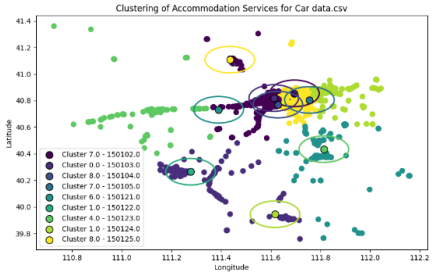


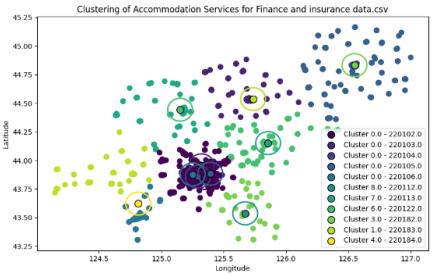


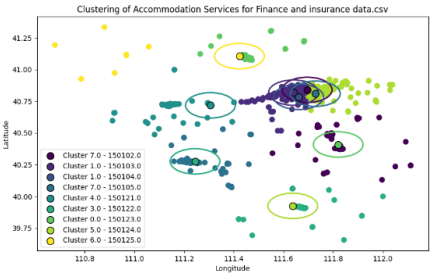


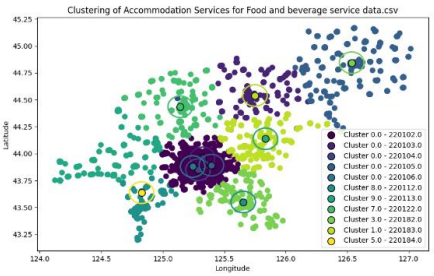


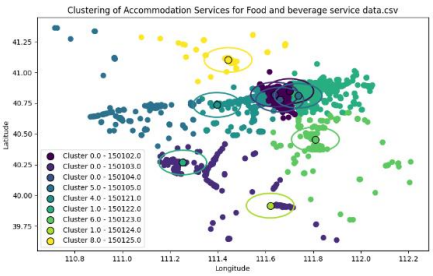


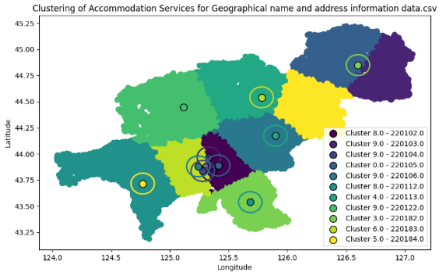


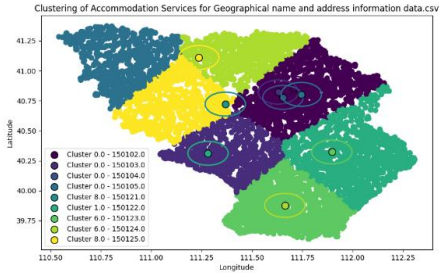


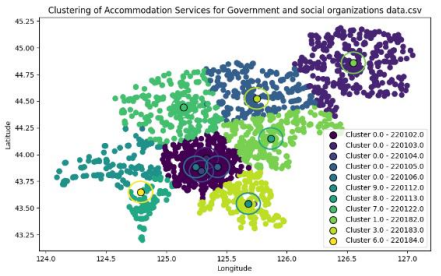


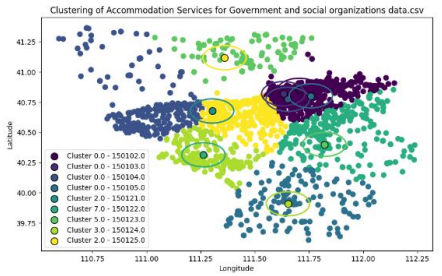


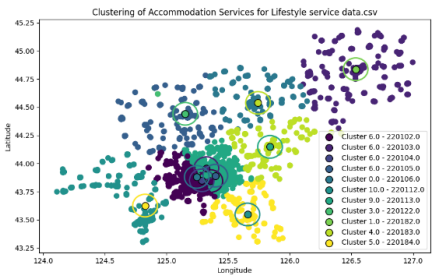


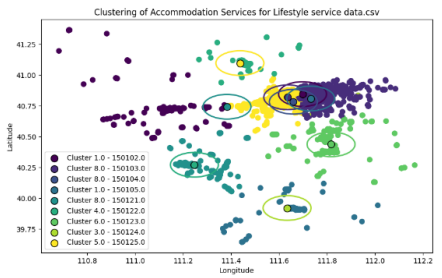

Clustering of Accommodation Services for Geographical name and address information data.csv
Latitude
Longitude
Cluster 8.0 - 220102.0
Cluster 9.0 - 220103.0
Cluster 9.0 - 220104.0
Cluster 0.0 - 220105.0
Cluster 9.0 - 220106.0
Cluster 8.0 - 220112.0
Cluster 4.0 - 220113.0
Cluster 9.0 - 220122.0
Cluster 3.0 - 220182.0
Cluster 6.0 - 220183.0
Cluster 5.0 - 220184.0

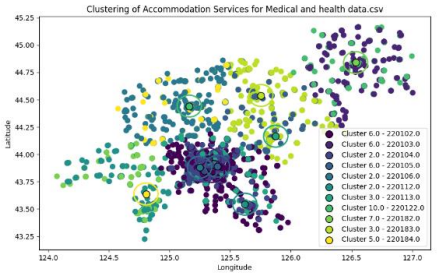

Clustering of Accommodation Services for Geographical name and address information data.csv
Latitude
Longitude
Cluster 0.0 - 150102.0
Cluster 0.0 - 150103.0
Cluster 0.0 - 150104.0
Cluster 0.0 - 150105.0
Cluster 8.0 - 150121.0
Cluster 1.0 - 150122.0
Cluster 6.0 - 150123.0
Cluster 6.0 - 150124.0
Cluster 8.0 - 150125.0

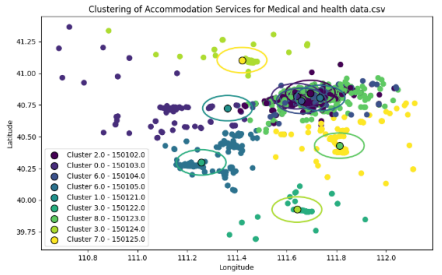

Clustering of Accommodation Services for Government and social organizations data.csv
Latitude
Longitude
Cluster 0.0 - 220102.0
Cluster 0.0 - 220103.0
Cluster 0.0 - 220104.0
Cluster 0.0 - 220105.0
Cluster 0.0 - 220106.0
Cluster 9.0 - 220112.0
Cluster 8.0 - 220113.0
Cluster 7.0 - 220122.0
Cluster 1.0 - 220182.0
Cluster 3.0 - 220183.0
Cluster 6.0 - 220184.0

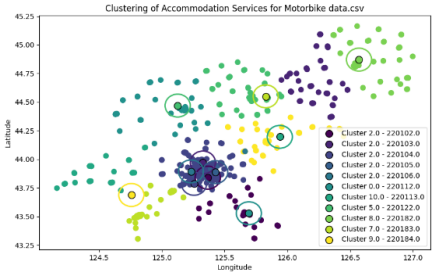

Clustering of Accommodation Services for Government and social organizations data.csv
Latitude
Longitude
Cluster 0.0 - 150102.0
Cluster 0.0 - 150103.0
Cluster 0.0 - 150104.0
Cluster 0.0 - 150105.0
Cluster 2.0 - 150121.0
Cluster 7.0 - 150122.0
Cluster 5.0 - 150123.0
Cluster 3.0 - 150124.0
Cluster 2.0 - 150125.0


Clustering of Accommodation Services for Lifestyle service data.csv
Latitude
Longitude
Cluster 6.0 - 220102.0
Cluster 6.0 - 220103.0
Cluster 6.0 - 220104.0
Cluster 6.0 - 220105.0
Cluster 0.0 - 220106.0
Cluster 10.0 - 220112.0
Cluster 9.0 - 220113.0
Cluster 3.0 - 220122.0
Cluster 1.0 - 220182.0
Cluster 4.0 - 220183.0
Cluster 5.0 - 220184.0


Clustering of Accommodation Services for Lifestyle service data.csv
Latitude
Longitude
Cluster 1.0 - 150102.0
Cluster 8.0 - 150103.0
Cluster 8.0 - 150104.0
Cluster 1.0 - 150105.0
Cluster 8.0 - 150121.0
Cluster 4.0 - 150122.0
Cluster 6.0 - 150123.0
Cluster 3.0 - 150124.0
Cluster 5.0 - 150125.0


Clustering of Accommodation Services for Medical and health data.csv
Latitude
Longitude
Cluster 6.0 - 220102.0
Cluster 6.0 - 220103.0
Cluster 2.0 - 220104.0
Cluster 6.0 - 220105.0
Cluster 2.0 - 220106.0
Cluster 2.0 - 220112.0
Cluster 3.0 - 220113.0
Cluster 10.0 - 220122.0
Cluster 7.0 - 220182.0
Cluster 3.0 - 220183.0
Cluster 5.0 - 220184.0


Clustering of Accommodation Services for Medical and health data.csv
Latitude
Longitude
Cluster 2.0 - 150102.0
Cluster 0.0 - 150103.0
Cluster 6.0 - 150104.0
Cluster 6.0 - 150105.0
Cluster 1.0 - 150121.0
Cluster 3.0 - 150122.0
Cluster 8.0 - 150123.0
Cluster 3.0 - 150124.0
Cluster 7.0 - 150125.0


Clustering of Accommodation Services for Motorbike data.csv
Latitude
Longitude
Cluster 2.0 - 220102.0
Cluster 2.0 - 220103.0
Cluster 2.0 - 220104.0
Cluster 2.0 - 220105.0
Cluster 2.0 - 220106.0
Cluster 0.0 - 220112.0
Cluster 10.0 - 220113.0
Cluster 5.0 - 220122.0
Cluster 8.0 - 220182.0
Cluster 7.0 - 220183.0
Cluster 9.0 - 220184.0

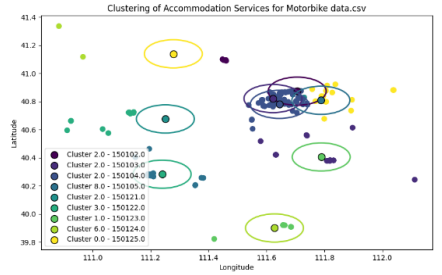

Clustering of Accommodation Services for Motorbike data.csv
Latitude
Longitude
Cluster 2.0 - 150102.0
Cluster 2.0 - 150103.0
Cluster 2.0 - 150104.0
Cluster 8.0 - 150105.0
Cluster 2.0 - 150121.0
Cluster 3.0 - 150122.0
Cluster 1.0 - 150123.0
Cluster 6.0 - 150124.0
Cluster 9.0 - 150125.0

Figure 8 City 1、2 Facility Distribution

附录 B:

附录
问题 1 代码

```
数据处理
import pandas as pd
import matplotlib.pyplot as plt
import folium
from geopy.distance import geodesic
import seaborn as sns
```

```
import matplotlib as mpl

mpl.rcParams['font.family'] = 'SimHei'  # Windows 用户
mpl.rcParams['axes.unicode_minus'] = False

# 文件路径配置
file_path_1 = r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 1.xlsx"
file_path_2 = r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 2.xlsx"

city1_files = {
    "public_facilities": r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 3\Public facilities data.csv",
    "transportation": r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 3\Transportation facilities data.csv",
    "medical_health": r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 3\Medical and health data.csv",
    "education_culture": r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 3\Science, education, and culture data.csv"
}

city2_files = {
    "public_facilities": r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 4\Public facilities data.csv",
    "transportation": r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 4\Transportation facilities data.csv",
    "medical_health": r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 4\Medical and health data.csv",
    "education_culture": r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 4\Science, education, and culture data.csv"
}

# 加载房产数据
```

```
def load_house_data(file_path):
    data = pd.read_excel(file_path, sheet_name='Sheet1')
    return data

house_data_1 = load_house_data(file_path_1)
house_data_2 = load_house_data(file_path_2)

# 处理缺失值和异常值
def process_missing_and_outliers(data, column_name):
    print(f"处理前{column_name}缺失值数量:",
data[column_name].isnull().sum())
    data[column_name] =
data[column_name].fillna(data[column_name].mean())  # 填充缺失值
    Q1, Q3 = data[column_name].quantile([0.25, 0.75])  # 四分位数
    IQR = Q3 - Q1  # 四分位距
    lower_bound, upper_bound = Q1 - 1.5 * IQR, Q3 + 1.5 * IQR
    data[column_name] = data[column_name].apply(lambda x: max(min(x,
upper_bound), lower_bound))
    print(f"处理后{column_name}缺失值数量:",
data[column_name].isnull().sum())
    return data

house_data_1 = process_missing_and_outliers(house_data_1, 'Price (USD)')
house_data_2 = process_missing_and_outliers(house_data_2, 'Price (USD)')

# 数据分布分析
def plot_distribution(data, column_name, title):
    plt.figure(figsize=(10, 6))
    plt.hist(data[column_name], bins=30, color='#3E92CC',
edgecolor='black', alpha=0.7)
    plt.title(f"{title}: {column_name}")
    plt.xlabel(column_name)
    plt.ylabel('Frequency')
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()

plot_distribution(house_data_1, 'Price (USD)', 'city1-Price')
plot_distribution(house_data_2, 'Price (USD)', 'city2-Price')

# 绘制箱型图
def plot_boxplot(data1, data2, column_name, city1_name, city2_name,
title):
    """
    绘制两个城市的数据箱型图
```

```
    :param data1: 城市 1 数据
    :param data2: 城市 2 数据
    :param column_name: 绘制的列名
    :param city1_name: 城市 1 名称
    :param city2_name: 城市 2 名称
    :param title: 图标题
    """
    plt.figure(figsize=(10, 6))

    # 合并数据用于绘制
    combined_data = [data1[column_name], data2[column_name]]

    # 绘制箱型图
    plt.boxplot(combined_data,
                patch_artist=True,
                boxprops=dict(facecolor='#3E92CC', color='black',
alpha=0.7),
                medianprops=dict(color='red', linewidth=1.5),
                whiskerprops=dict(color='black', linewidth=1),
                capprops=dict(color='black', linewidth=1))

    # 设置 x 轴和标题
    plt.xticks([1, 2], [city1_name, city2_name], fontsize=12)
    plt.title(title, fontsize=16)
    plt.ylabel(column_name, fontsize=12)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()


# 绘制城市 1 和城市 2 的房价箱型图
plot_boxplot(house_data_1, house_data_2, 'Price (USD)', 'City 1', 'City
2', 'Comparison of Price (USD) Between City 1 and City 2')

# 加载设施数据
def load_facilities(file_dict):
    datasets = {}
    for key, path in file_dict.items():
        try:
            data = pd.read_csv(path)
            if 'lon_gcj02' in data.columns and 'lat_gcj02' in
data.columns:
                data['lon_gcj02'] = pd.to_numeric(data['lon_gcj02'],
errors='coerce')
```

```
                data['lat_gcj02'] = pd.to_numeric(data['lat_gcj02'],
errors='coerce')
                datasets[key] = data.dropna(subset=['lon_gcj02',
'lat_gcj02'])
            else:
                print(f"{key} 跳过。")
        except Exception as e:
            print(f"加载 {key} {e}")
    return datasets

city1_data = load_facilities(city1_files)
city2_data = load_facilities(city2_files)

# 计算房产到设施的最近距离
def calculate_nearest_distance(house_data, facility_data, house_lat_col,
house_lon_col):
    distances = []
    for _, house in house_data.iterrows():
        house_location = (house[house_lat_col], house[house_lon_col])
        min_distance = facility_data.apply(
            lambda row: geodesic(house_location, (row['lat_gcj02'],
row['lon_gcj02'])).km, axis=1
        ).min()
        distances.append(min_distance)
    return distances

# 为房产添加设施距离特征
def add_facility_distance_features(house_data, facilities,
house_lat_col, house_lon_col):
    for facility_type, facility_data in facilities.items():
        print(f"计算房产到最近 {facility_type} 的距离...")
        house_data[f'nearest_{facility_type}_distance'] =
calculate_nearest_distance(
            house_data, facility_data, house_lat_col, house_lon_col
        )

add_facility_distance_features(house_data_1, city1_data, 'lat', 'lon')
add_facility_distance_features(house_data_2, city2_data, 'lat', 'lon')

# 综合相关性分析（城市 1 和城市 2 在同一张图中）
def analyze_feature_correlation_combined(data1, data2, target_col,
city1_name, city2_name):
    """
    综合分析目标列与其他特征的相关性，城市 1 和城市 2 在同一张图中绘制
```

```
    """
    # 选择数值型列
    numeric_cols1 = data1.select_dtypes(include=['float64',
'int64']).columns
    numeric_cols2 = data2.select_dtypes(include=['float64',
'int64']).columns

    # 计算相关性
    correlation1 =
data1[numeric_cols1].corr()[target_col].sort_values(ascending=False)[1:]
    correlation2 =
data2[numeric_cols2].corr()[target_col].sort_values(ascending=False)[1:]

    # 合并列索引
    combined_index = list(set(correlation1.index) |
set(correlation2.index))
    correlation1 = correlation1.reindex(combined_index, fill_value=0)
    correlation2 = correlation2.reindex(combined_index, fill_value=0)

    # 绘制相关性柱状图
    x = range(len(combined_index))
    plt.figure(figsize=(15, 8))

    # 城市1的柱状图
    plt.bar(x, correlation1, width=0.4, label=city1_name,
color='#3E92CC', edgecolor='black', alpha=0.7, align='center')

    # 城市2的柱状图
    plt.bar([i + 0.4 for i in x], correlation2, width=0.4,
label=city2_name, color='#F95D6A', edgecolor='black',
            alpha=0.7, align='center')

    # 图表美化
    plt.title(f"Correlation of Features with {target_col}", fontsize=16)
    plt.ylabel("Correlation Coefficient", fontsize=12)
    plt.xlabel("Feature Names", fontsize=12)
    plt.axhline(0, color='black', linewidth=0.8)  # 添加水平线
    plt.xticks([i + 0.2 for i in x], combined_index, rotation=45,
ha='right', fontsize=10)  # 调整x轴刻度和标签
    plt.legend(fontsize=12)
    plt.grid(axis='y', linestyle='--', alpha=0.7)  # 添加网格线，样式统一
    plt.tight_layout()  # 调整布局防止溢出
    plt.show()
```

```
# 调用函数进行分析
analyze_feature_correlation_combined(house_data_1, house_data_2, 'Price
(USD)', 'City 1', 'City 2')



预测
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# File path configuration
file_path_1 = r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 1.xlsx"
file_path_2 = r"C:\Users\11856\Desktop\2024“ShuWei
Cup”_Problem\2024“ShuWei Cup”_Problem\2024_“ShuWei
Cup”D_Problem\Appendix 2.xlsx"

# Load data from both files
df1 = pd.read_excel(file_path_1, sheet_name='Sheet1')
df2 = pd.read_excel(file_path_2, sheet_name='Sheet1')

# Merge the two datasets
df = pd.concat([df1, df2], ignore_index=True)

# Fill missing values (for 'Total number of households' and 'Floor area
ratio')
df['Total number of households'] = df['Total number of
households'].fillna(df['Total number of households'].mean())
df['Floor area ratio'] = df['Floor area ratio'].fillna(df['Floor area
ratio'].mean())

# Extract parking space ratios
def extract_parking_ratios(parking_str):
    try:
```

```
        if "(" in parking_str:
            ratio_part = parking_str.split('(')[1].strip(')')
            ground_ratio, underground_ratio = map(float,
ratio_part.split(':'))
            total_ratio = ground_ratio + underground_ratio
            return ground_ratio / total_ratio, underground_ratio /
total_ratio
        else:
            return 0, 0
    except Exception as e:
        return 0, 0

# Parse parking space ratios and calculate the total ratio
df[['ground_ratio', 'underground_ratio']] = df['parking
space'].apply(extract_parking_ratios).apply(pd.Series)
df['total_parking_ratio'] = df'ground_ratio'] + df['underground_ratio']

# Calculate housing stock
df['total_floor_area'] = df['Total number of households'] * (1 /
df['Floor area ratio'])
df['average_household_area'] = df['total_floor_area'] / df['Total number
of households']
df['total_housing_stock'] = df['Total number of households'] *
df['average_household_area'] * df['total_parking_ratio']

# Replace city codes with City1 and City2
city_mapping = {431: 'City1', 471: 'City2'}
df['city_name'] = df['citycode'].map(city_mapping)

# Calculate total housing stock by city
city_housing_stock =
df.groupby('city_name')['total_housing_stock'].sum()

# Output the estimated housing stock for each city
for city, housing_stock in city_housing_stock.items():
    print(f"{city} : {housing_stock:.2f}")

# Features and target
features = [
    'Total number of households', 'Greening rate', 'Floor area ratio',
    'Building type', 'Property management fee（/m²/month USD）',
    'above-ground parking fee（/month USD）', 'underground parking fee
（/month USD）',
    'property type', 'lon', 'lat', 'adcode'
```

```
]
target = 'Price (USD)'
categorical_features = ['Building type', 'property type', 'adcode']

# Dynamically validate features
def validate_features(data, features):
    existing_features = [col for col in features if col in data.columns]
    missing_features = [col for col in features if col not in
data.columns]
    if missing_features:
        print(f"Warning: Missing features {missing_features}")
    return existing_features

# Create dynamic preprocessor
def create_preprocessor(numerical_features, categorical_features):
    transformers = []
    if numerical_features:
        transformers.append(('num', StandardScaler(),
numerical_features))
    if categorical_features:
        transformers.append(('cat',
OneHotEncoder(handle_unknown='ignore'), categorical_features))
    return ColumnTransformer(transformers)

# Model definitions
models = {
    "Linear Regression": LinearRegression(),
    "Random Forest": RandomForestRegressor(n_estimators=100,
random_state=42),
    "Gradient Boosting": GradientBoostingRegressor(n_estimators=100,
random_state=42)
}

# Initialize result storage
all_results = []

# Process for City 1 and City 2
for city_name, house_data in {"City 1": df[df['city_name'] == 'City1'],
"City 2": df[df['city_name'] == 'City2']}.items():
    print(f"\nProcessing {city_name}...")

    # Filter valid features
    valid_features = validate_features(house_data, features)
```

```
    valid_categorical_features = validate_features(house_data,
categorical_features)
    valid_numerical_features = [col for col in valid_features if col not
in valid_categorical_features]

    # Data preprocessing
    preprocessor = create_preprocessor(valid_numerical_features,
valid_categorical_features)
    house_data = house_data.dropna(subset=valid_features + [target])

    # Predict by region
    for region, region_data in house_data.groupby('adcode'):
        print(f"\nRegion {region} ({city_name}):")

        # Check data size
        if len(region_data) < 2:  # Skip if data size is insufficient
            print(f"Insufficient data for region {region}. Skipping...")
            continue

        X = region_data[valid_features]
        y = region_data[target]

        # Split dataset
        X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

        for model_name, model in models.items():
            pipeline = Pipeline(steps=[
                ('preprocessor', preprocessor),
                ('model', model)
            ])
            pipeline.fit(X_train, y_train)
            y_pred = pipeline.predict(X_test)
            rmse = mean_squared_error(y_test, y_pred, squared=False)

            # Save results
            all_results.append({'City': city_name, 'Region': region,
'Model': model_name, 'RMSE': rmse})
            print(f"Model: {model_name}, RMSE: {rmse:.2f}")

# Convert to DataFrame
results_df = pd.DataFrame(all_results)

# Plot RMSE bar chart by city
```

```
for city_name in results_df['City'].unique():
    city_results = results_df[results_df['City'] == city_name]

    # Plot bar chart by region and model
    plt.figure(figsize=(12, 6))
    for model_name in city_results['Model'].unique():
        model_results = city_results[city_results['Model'] == model_name]
        plt.bar(
            model_results['Region'].astype(str) + "-" + model_name,
            model_results['RMSE'],
            label=model_name,
            edgecolor='black'  # Add border color
        )
    plt.xlabel("Region-Model")
    plt.ylabel("RMSE")
    plt.title(f"Model Performance by Region - {city_name}")
    plt.legend()
    plt.xticks(rotation=45)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()

    # Plot average RMSE
    city_avg_metrics =
city_results.groupby('Model')['RMSE'].mean().reset_index()
    plt.figure(figsize=(8, 6))
    plt.bar(
        city_avg_metrics['Model'],
        city_avg_metrics['RMSE'],
        color=['#3E92CC', '#F95D6A', '#00C49A'],
        edgecolor='black'  # Add border color
    )
    plt.xlabel("Model")
    plt.ylabel("Average RMSE")
    plt.title(f"Average RMSE for Each Model - {city_name}")
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.tight_layout()
    plt.show()

# Visualizing actual vs predicted prices
for city_name, house_data in {"City 1": df[df['city_name'] == 'City1'],
"City 2": df[df['city_name'] == 'City2']}.items():
    print(f"\nVisualizing Actual vs Predicted Prices for
{city_name}...")
```

```
    valid_features = validate_features(house_data, features)
    valid_categorical_features = validate_features(house_data,
categorical_features)
    valid_numerical_features = [col for col in valid_features if col not
in valid_categorical_features]

    preprocessor = create_preprocessor(valid_numerical_features,
valid_categorical_features)
    house_data = house_data.dropna(subset=valid_features + [target])

    for region, region_data in house_data.groupby('adcode'):
        print(f"\nRegion {region} ({city_name}):")

        if len(region_data) < 2:  # Skip if data size is insufficient
            print(f"Insufficient data for region {region}. Skipping...")
            continue

        X = region_data[valid_features]
        y = region_data[target]

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

        for model_name, model in models.items():
            pipeline = Pipeline(steps=[
                ('preprocessor', preprocessor),
                ('model', model)
            ])
            pipeline.fit(X_train, y_train)
            y_pred = pipeline.predict(X_test)

            # Plot actual vs predicted comparison
            plt.figure(figsize=(10, 6))
            plt.plot(range(len(y_test)), y_test.values, 'o-',
label='Actual', color='#3E92CC')  # Actual values
            plt.plot(range(len(y_pred)), y_pred, 'x-', label='Predicted',
color='#F95D6A')  # Predicted values
            plt.title(f"{city_name} - Region {region}: {model_name}
Actual vs Predicted Prices")
            plt.xlabel("Sample Index")
            plt.ylabel("Price (USD)")
            plt.legend()
            plt.tight_layout()
```

```
            plt.grid(axis='y', linestyle='--', alpha=0.7)
            plt.show()
```

附录

问题 2 代码

聚类

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import os

# 文件夹路径
folder_path = "Appendix 3"

# 列出文件夹中的CSV 文件
csv_files = [f for f in os.listdir(folder_path) if f.endswith('.csv')]

# 逐个处理文件
for file in csv_files:
    file_path = os.path.join(folder_path, file)
    print(f"正在处理文件: {file}")

    # 读取CSV 文件
    data = pd.read_csv(file_path)

    # 清理列名
    data.columns = data.columns.str.strip()

    # 检查是否有'typecode'列
    if 'typecode' in data.columns:
        # 处理'typecode'列
        data['typecode'] = data['typecode'].astype(str).fillna('unknown')

        # 提取所需特征
        features = data[['lon_gcj02', 'lat_gcj02']].copy()
        features['typecode'] = pd.factorize(data['typecode'])[0]  # 将
'typecode'编码为整数

        # 标准化特征
        scaler = StandardScaler()
        features_scaled = scaler.fit_transform(features)

        # 设置聚类数量
        n_clusters = 8
        print(f"使用 {n_clusters} 个聚类...")
```

```
        # 训练 KMeans 模型
        kmeans = KMeans(n_clusters=n_clusters, random_state=42)
        data['cluster'] = kmeans.fit_predict(features_scaled)

        # 绘制聚类结果
        plt.figure(figsize=(10, 6))
        plt.scatter(data['lon_gcj02'], data['lat_gcj02'],
c=data['cluster'], cmap='viridis', s=50, label='数据点')

        # 绘制聚类中心
        centers = kmeans.cluster_centers_
        centers_original = scaler.inverse_transform(centers)
        plt.scatter(centers_original[:, 0], centers_original[:, 1],
c='red', marker='x', s=200, label='聚类中心')

        # 添加标题和标签
        plt.title(f'K-means 聚类结果 ({file})')
        plt.xlabel('经度')
        plt.ylabel('纬度')
        plt.legend()

        # 显示图像
        plt.show()

    else:
        print(f"警告: 文件 {file} 中未找到'typecode'列")

print("所有文件处理完成! ")
```

```
import pandas as pd
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import numpy as np
import os
import chardet

# Folder path containing the CSV files
folder_path = "Appendix 4"  # Replace with your folder path

# Detect file encoding
def detect_encoding(file_path):
    with open(file_path, 'rb') as f:
```

```
        result = chardet.detect(f.read())
        return result['encoding']

# Read CSV file with encoding detection
def read_csv(file_path):
    try:
        encoding = detect_encoding(file_path)
        print(f"Detected encoding for {file_path}: {encoding}")
        return pd.read_csv(file_path, encoding=encoding)
    except Exception as e:
        print(f"Error reading {file_path}: {e}")
        return None

# Process each CSV file
csv_files = [f for f in os.listdir(folder_path) if f.endswith('.csv')]
for file in csv_files:
    file_path = os.path.join(folder_path, file)
    print(f"Processing {file}...")

    # Read the CSV file
    data = read_csv(file_path)
    if data is None:
        print(f"Skipping {file} due to read error.")
        continue

    # Clean column names (strip any extra spaces or newlines)
    data.columns = data.columns.str.strip()

    # Check if 'typecode' exists and contains valid data
    if 'typecode' in data.columns:
        # Convert 'typecode' to strings and handle any non-string values
        data['typecode'] = data['typecode'].astype(str)
        data['typecode'] = data['typecode'].replace(['nan', 'None'], '')

        # Split 'typecode' column by '|' and explode into multiple rows
        expanded_data = data.copy()
        expanded_data =
expanded_data.assign(typecode=expanded_data['typecode'].str.split('|'))
        expanded_data = expanded_data.explode('typecode')

        # Convert 'typecode' to numeric if possible
        expanded_data['typecode'] =
pd.to_numeric(expanded_data['typecode'], errors='coerce')
```

```
        # Extract necessary features (longitude, latitude, typecode)
        if {'lon_gcj02', 'lat_gcj02',
'typecode'}.issubset(expanded_data.columns):
            features = expanded_data[['lon_gcj02', 'lat_gcj02',
'typecode']]

            # Standardize features
            scaler = StandardScaler()
            features_scaled = scaler.fit_transform(features)

            # Get the number of clusters based on unique 'adcode'
            if 'adcode' in expanded_data.columns:
                n_clusters = expanded_data['adcode'].nunique()
                print(f"Number of clusters based on 'adcode':
{n_clusters}")

                # Train KMeans model and assign clusters
                kmeans = KMeans(n_clusters=n_clusters, random_state=42)
                expanded_data['cluster'] =
kmeans.fit_predict(features_scaled)

                # Save the cluster plot as an image
                plot_filename = os.path.join(folder_path,
f"cluster_{file.replace('.csv', '.png')}")
                plt.figure(figsize=(10, 6))

                # Plot data points with cluster colors
                plt.scatter(expanded_data['lon_gcj02'],
expanded_data['lat_gcj02'],
                            c=expanded_data['cluster'], cmap='viridis',
s=50)

                # Plot cluster centers
                grouped = expanded_data.groupby('adcode').agg({
                    'lon_gcj02': 'mean',
                    'lat_gcj02': 'mean',
                    'cluster': 'first'
                }).reset_index()

                colors = plt.cm.viridis(np.linspace(0, 1, len(grouped)))

                for i, row in grouped.iterrows():
                    plt.scatter(row['lon_gcj02'], row['lat_gcj02'],
color=colors[i],
```

```
                                s=100, edgecolor='black', label=f'Cluster
{row["cluster"]} - {row["adcode"]}')
                    circle = plt.Circle((row['lon_gcj02'],
row['lat_gcj02']), 0.1,
                                        color=colors[i], fill=False,
linewidth=2)
                    plt.gca().add_artist(circle)

                # Add title and labels
                plt.title(f'Clustering of Accommodation Services for
{file}')
                plt.xlabel('Longitude')
                plt.ylabel('Latitude')

                # Save the plot
                plt.legend()
                plt.savefig(plot_filename)
                plt.close()

                print(f"Cluster plot saved as {plot_filename}")
            else:
                print(f"Warning: 'adcode' column not found in {file}.
Skipping clustering.")
        else:
            print(f"Warning: Necessary columns not found in {file}.
Skipping clustering.")
    else:
        print(f"Warning: 'typecode' column not found in {file}.
Skipping.")

print("Processing complete.")
```

密度计算

```
import pandas as pd
import os

folder_path = "Appendix 3"
output_path = os.path.join(folder_path,
"facility_density_results_changchun.csv")

# 获取所有CSV文件
csv_files = [f for f in os.listdir(folder_path) if f.endswith('.csv')]
```

```
# 初始化结果列表
all_density_data = []

# 长春市的固定面积（单位：平方公里）
changchun_area_km2 = 24700

# 处理每个CSV文件
for file in csv_files:
    file_path = os.path.join(folder_path, file)
    print(f"Processing {file}...")

    # 读取CSV文件
    data = pd.read_csv(file_path)

    # 检查是否包含必要的列
    if {'lat_gcj02', 'lon_gcj02', 'adcode'}.issubset(data.columns):
        # 计算设施数量
        num_facilities = len(data)

        # 使用指定的长春市面积计算设施密度
        facility_density = num_facilities / changchun_area_km2  # 单位：设施/平方公里

        # 将结果添加到密度数据中
        density_data = {
            'file_name': file,
            'num_facilities': num_facilities,
            'area_km2': changchun_area_km2,
            'facility_density': facility_density
        }

        all_density_data.append(density_data)
    else:
        print(f"Warning: Missing required columns in {file}")

# 将所有文件的密度结果合并并保存
if all_density_data:
    density_df = pd.DataFrame(all_density_data)
    density_df.to_csv(output_path, index=False)
    print(f"Facility density results saved to {output_path}")
else:
    print("No valid data processed.")

print("Processing complete.")
```

```
import pandas as pd
import os
from charset_normalizer import from_path

# 定义函数自动检测文件编码
def detect_encoding(file_path):
    result = from_path(file_path).best()
    return result.encoding if result else 'utf-8'

# 定义输入文件夹和输出路径
folder_path = "Appendix 4"
output_path = os.path.join(folder_path,
"facility_density_results_changchun.csv")

# 获取所有CSV 文件
csv_files = [f for f in os.listdir(folder_path) if f.endswith('.csv')]

# 初始化结果列表
all_density_data = []

# 呼和浩特市的固定面积（单位：平方公里）
changchun_area_km2 = 17200

# 处理每个CSV 文件
for file in csv_files:
    file_path = os.path.join(folder_path, file)
    print(f"Processing {file}...")

    # 检测文件编码
    encoding = detect_encoding(file_path)
    print(f"Detected encoding for {file}: {encoding}")

    try:
        # 使用检测到的编码读取文件
        data = pd.read_csv(file_path, encoding=encoding)

        # 检查是否包含必要的列
        if {'lat_gcj02', 'lon_gcj02', 'adcode'}.issubset(data.columns):
            # 计算设施数量
            num_facilities = len(data)

            # 使用指定的呼和浩特市面积计算设施密度
```

```
            facility_density = num_facilities / changchun_area_km2  # 单
位：设施/平方公里

            # 将结果添加到密度数据中
            density_data = {
                'file_name': file,
                'num_facilities': num_facilities,
                'area_km2': changchun_area_km2,
                'facility_density': facility_density
            }

            all_density_data.append(density_data)
        else:
            print(f"Warning: Missing required columns in {file}")

    except Exception as e:
        print(f"Error processing {file}: {e}")

# 将所有文件的密度结果合并并保存
if all_density_data:
    density_df = pd.DataFrame(all_density_data)
    density_df.to_csv(output_path, index=False)
    print(f"Facility density results saved to {output_path}")
else:
    print("No valid data processed.")

print("Processing complete.")
```

附录

问题 3 代码

数据清洗

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor,
RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
import numpy as np
import warnings

warnings.filterwarnings("ignore")

property_data = pd.read_excel('Appendix 1.xlsx')
```

```
# 查看缺失值情况
print("各列缺失值情况: \n", property_data.isna().sum())

# 对每一列进行缺失值填补
for column in property_data.columns:
    if property_data[column].isna().sum() > 0:
        print(f"正在填补列 {column} 的缺失值...")

        # 检查是否为数值型列
        if property_data[column].dtype in ['float64', 'int64']:
            # 数值型列：使用回归模型填补
            known = property_data[property_data[column].notna()]
            unknown = property_data[property_data[column].isna()]

            X_known =
known.drop(columns=[column]).select_dtypes(include=['number'])  # 已知部
分的特征
            y_known = known[column]  # 已知部分的目标变量

            # 检查是否有足够的数值特征用于回归
            if X_known.empty:
                print(f"{column} 没有足够的数值特征用于回归，跳过此列")
                continue

            # 处理 X_known 和 X_unknown 中的 NaN 和 infinity 值
            X_known = X_known.replace([np.inf, -np.inf], np.nan).dropna()
            y_known = y_known[X_known.index]  # 保证 y_known 对应的行没有丢
失

            # 如果 X_known 为空，则跳过此列
            if X_known.empty or y_known.empty:
                print(f"{column} 没有足够的数据用于回归，跳过此列")
                continue

            # 训练模型填补缺失值
            model = RandomForestRegressor(n_estimators=100,
random_state=42)
            model.fit(X_known, y_known)

            X_unknown =
unknown.drop(columns=[column]).select_dtypes(include=['number'])

            # 处理 X_unknown 中的 NaN 和 infinity 值
```

```
            X_unknown = X_unknown.replace([np.inf, -np.inf],
np.nan).dropna()

            # 确保 X_unknown 中有数据，且行数与需要填补的行数一致
            if X_unknown.empty:
                print(f"{column} 没有足够的测试数据，跳过此列")
                continue

            # 检查预测结果的长度是否与缺失值的行数一致
            predicted_values = model.predict(X_unknown)

            # 确保预测结果的长度与缺失值的行数一致
            if len(predicted_values) == len(unknown):
                property_data.loc[property_data[column].isna(), column] =
predicted_values
            else:
                print(f"{column} 的预测结果长度与缺失值行数不匹配，跳过此列")

        else:
            # 对于非数值型列，使用众数填补缺失值
            most_frequent = property_data[column].mode()[0]
            property_data[column].fillna(most_frequent, inplace=True)
            print(f"{column} 列使用众数 '{most_frequent}' 填补了缺失值")

print("填补后的缺失值情况：\n", property_data.isna().sum())

property_data.to_excel('cleaned_Appendix_1.xlsx', index=False)
print("所有缺失值已处理并保存到 'cleaned_Appendix_1.xlsx'")

import pandas as pd
from sklearn.ensemble import RandomForestRegressor,
RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
import numpy as np
import warnings

warnings.filterwarnings("ignore")

property_data = pd.read_excel('Appendix 2.xlsx')

# 查看缺失值情况
print("各列缺失值情况：\n", property_data.isna().sum())

# 对每一列进行缺失值填补
```

```
for column in property_data.columns:
    if property_data[column].isna().sum() > 0:
        print(f"正在填补列 {column} 的缺失值...")

        # 检查是否为数值型列
        if property_data[column].dtype in ['float64', 'int64']:
            # 数值型列：使用回归模型填补
            known = property_data[property_data[column].notna()]
            unknown = property_data[property_data[column].isna()]

            X_known =
known.drop(columns=[column]).select_dtypes(include=['number'])  # 已知部
分的特征
            y_known = known[column]  # 已知部分的目标变量

            # 检查是否有足够的数值特征用于回归
            if X_known.empty:
                print(f"{column} 没有足够的数值特征用于回归，跳过此列")
                continue

            # 处理 X_known 和 X_unknown 中的 NaN 和 infinity 值
            X_known = X_known.replace([np.inf, -np.inf], np.nan).dropna()
            y_known = y_known[X_known.index]  # 保证 y_known 对应的行没有丢
失

            # 如果 X_known 为空，则跳过此列
            if X_known.empty or y_known.empty:
                print(f"{column} 没有足够的数据用于回归，跳过此列")
                continue

            # 训练模型填补缺失值
            model = RandomForestRegressor(n_estimators=100,
random_state=42)
            model.fit(X_known, y_known)

            X_unknown =
unknown.drop(columns=[column]).select_dtypes(include=['number'])

            # 处理 X_unknown 中的 NaN 和 infinity 值
            X_unknown = X_unknown.replace([np.inf, -np.inf],
np.nan).dropna()

            # 确保 X_unknown 中有数据，且行数与需要填补的行数一致
            if X_unknown.empty:
```

```
                print(f"{column} 没有足够的测试数据，跳过此列")
                continue

            # 检查预测结果的长度是否与缺失值的行数一致
            predicted_values = model.predict(X_unknown)

            # 确保预测结果的长度与缺失值的行数一致
            if len(predicted_values) == len(unknown):
                property_data.loc[property_data[column].isna(), column] =
predicted_values
            else:
                print(f"{column} 的预测结果长度与缺失值行数不匹配，跳过此列")

        else:
            # 对于非数值型列，使用众数填补缺失值
            most_frequent = property_data[column].mode()[0]
            property_data[column].fillna(most_frequent, inplace=True)
            print(f"{column} 列使用众数 '{most_frequent}' 填补了缺失值")

print("填补后的缺失值情况：\n", property_data.isna().sum())

property_data.to_excel('cleaned_Appendix_2.xlsx', index=False)
print("所有缺失值已处理并保存到 'cleaned_Appendix_2.xlsx'")
```

公共数据量化

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# 公共设施数据
data = pd.DataFrame({
    'type': [
        '公共厕所', '女洗手间', '男洗手间', '紧急避难场所', '公共设施',
        '报刊亭', '残障洗手间/无障碍洗手间', '公用电话', '婴儿换洗间/哺乳室/母
婴室',
        '公共设施|生活服务'
    ],
    'count': [1565, 92, 80, 55, 54, 27, 21, 12, 8, 1]
})

# 分配类别权重
weight_dict = {
    '紧急避难设施': 0.4,    # 包含紧急避难场所、残障洗手间/无障碍洗手间等
    '基本公共设施': 0.3,    # 包含公共厕所、男女洗手间等
    '辅助公共服务设施': 0.2  # 包含公用电话、报刊亭、婴儿换洗间等
```

```
}

# 定义每个设施类型的权重
data['weight'] = data['type'].apply(lambda x:
    weight_dict['紧急避难设施'] if '避难' in x or '残障' in x else
    weight_dict['基本公共设施'] if '厕所' in x or '洗手间' in x else
    weight_dict['辅助公共服务设施']
)

# 标准化计数
scaler = MinMaxScaler()
data['normalized_count'] = scaler.fit_transform(data[['count']])

# 计算加权得分
data['weighted_score'] = data['normalized_count'] * data['weight']

# 计算公共设施的总得分
total_score = data['weighted_score'].sum()

# 输出结果到CSV 文件
data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']].to_csv('Appendix_3_Public_facilities_data.csv',
index=False)

# 显示结果
print("公共设施数据量化结果: ")
print(data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']])
print(f"\n 城市 1 公共设施的总得分: {total_score:.2f}")
```

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# 公共设施数据
data = pd.DataFrame({
    'type': [
        '公共厕所', '公共设施', '男洗手间', '女洗手间', '公共厕所|生活服务',
        '残障洗手间/无障碍洗手间', '公用电话', '紧急避难场所', '报刊亭'
    ],
    'count': [2009, 45, 43, 41, 5, 5, 3, 1, 1]
})

# 分配类别权重
weight_dict = {
```

```
    '紧急避难设施': 0.4,  # 紧急避难场所
    '基础公共设施': 0.3, # 公共厕所、女洗手间、男洗手间等
    '其他公共服务设施': 0.2 # 公用电话、报刊亭等
}

# 定义每个设施类型的权重
data['weight'] = data['type'].apply(lambda x:
    weight_dict['紧急避难设施'] if '避难' in x else
    weight_dict['基础公共设施'] if '厕所' in x or '洗手间' in x else
    weight_dict['其他公共服务设施']
)

# 标准化计数
scaler = MinMaxScaler()
data['normalized_count'] = scaler.fit_transform(data[['count']])

# 计算加权得分
data['weighted_score'] = data['normalized_count'] * data['weight']

# 计算公共设施的总得分
total_score = data['weighted_score'].sum()

# 输出结果到CSV 文件
data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']].to_csv('Appendix_4_Public_facilities_data.csv',
index=False)

# 显示结果
print("公共设施数据量化结果: ")
print(data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']])
print(f"\n 城市 2 公共设施的总得分: {total_score:.2f}")
```

交通数据量化

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# 交通设施数据
data = pd.DataFrame({
    'type': [
        '公共停车场', '公交车站相关', '停车场出入口', '停车场相关', '路边停车
场',
        '专用停车场', '停车场入口', '停车场出口', '地铁站', '长途汽车站',
        '交通服务相关', '快速公交站', '出入口', '火车站', '货运火车站',
```

```
        '飞机场', '人渡口', '公共停车场|交通设施服务', '专用停车场|交通设施服
务'
    ],
    'count': [1689, 1639, 256, 244, 221, 208, 52, 43, 43, 40, 38, 30,
21, 17, 6, 3, 1, 1, 1]
})

# 分配类别权重
weight_dict = {
    '公共交通设施': 0.4,  # 包含公交车站相关、地铁站、火车站等
    '停车设施': 0.3,     # 包含公共停车场、路边停车场、专用停车场等
    '其他交通服务': 0.2  # 包含出入口、交通服务相关等
}

# 定义每个设施类型的权重
data['weight'] = data['type'].apply(lambda x:
    weight_dict['公共交通设施'] if '公交车站' in x or '地铁' in x or '火车
站' in x or '长途汽车站' in x or '快速公交' in x else
    weight_dict['停车设施'] if '停车场' in x else
    weight_dict['其他交通服务']
)

# 标准化计数
scaler = MinMaxScaler()
data['normalized_count'] = scaler.fit_transform(data[['count']])

# 计算加权得分
data['weighted_score'] = data['normalized_count'] * data['weight']

# 计算交通设施的总得分
total_score = data['weighted_score'].sum()

# 输出结果到CSV 文件
data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']].to_csv('Appendix_3_Transportation_facilities_data.csv
', index=False)

# 显示总得分
print("交通设施数据量化结果: ")
print(data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']])
print(f"\n 交通设施的总得分: {total_score:.2f}")

import pandas as pd
```

```
from sklearn.preprocessing import MinMaxScaler

data = pd.DataFrame({
    'type': [
        '公交车站相关', '公共停车场', '路边停车场', '停车场出入口', '专用停车场',
        '停车场相关', '地铁站', '交通服务相关', '长途汽车站', '停车场入口',
        '停车场出口', '出入口', '进站口/检票口', '火车站', '电车站',
        '路边停车场|交通设施服务', '公共停车场|交通设施服务', '货运火车站', '港口码头', '站台',
        '出站口', '班车站', '售票', '专用停车场|交通设施服务', '飞机场',
        '票务相关', '候车室', '改签', '人渡口', '机场出发/到达',
        '公安制证', '退票', '路边停车场|公司企业', '停车场出入口|餐饮服务', '售票|生活服务',
        '机场相关', '机场货运处'
    ],
    'count': [
        5036, 4410, 2415, 666, 595,
        543, 154, 122, 91, 83,
        77, 74, 44, 30, 25,
        19, 18, 14, 14, 14,
        10, 8, 7, 6, 4,
        4, 3, 3, 2, 2,
        2, 2, 1, 1, 1,
        1, 1
    ]
})

# 分配类别权重
weight_dict = {
    '公共交通设施': 0.4,  # 包含公交车站相关、地铁站、火车站等
    '停车设施': 0.3,     # 包含公共停车场、路边停车场、专用停车场等
    '其他交通服务': 0.2  # 包含出入口、售票等
}

# 定义每个设施类型的权重
data['weight'] = data['type'].apply(lambda x:
    weight_dict['公共交通设施'] if '公交车站' in x or '地铁' in x or '火车站' in x or '长途汽车站' in x or '电车站' in x else
    weight_dict['停车设施'] if '停车场' in x else
    weight_dict['其他交通服务']
)

# 标准化计数
```

```
scaler = MinMaxScaler()
data['normalized_count'] = scaler.fit_transform(data[['count']])

# 计算加权得分
data['weighted_score'] = data['normalized_count'] * data['weight']

# 计算交通设施的总得分
total_score = data['weighted_score'].sum()

# 输出结果到CSV 文件
data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']].to_csv('Appendix_4_Transportation_facilities_data.csv
', index=False)

# 显示总得分
print("交通设施数据量化结果: ")
print(data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']])
print(f"\n 城市 2 交通设施的总得分: {total_score:.2f}")
```

数据分析可视化

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

# 假设你已经加载了数据，以下是一个示例数据
# 我们为每个城市添加了一个 'city' 列，以便区分城市 1 和城市 2
data = {
    'type': ['公共厕所', '女洗手间', '男洗手间', '紧急避难场所', '公共设施',
             '报刊亭', '残障洗手间/无障碍洗手间', '公用电话', '婴儿换洗间/哺乳
室/母婴室',
             '公共设施|生活服务'],
    'count': [1565, 92, 80, 55, 54, 27, 21, 12, 8, 1],
    'weight': [0.3, 0.3, 0.3, 0.4, 0.3, 0.2, 0.4, 0.2, 0.2, 0.3],
    'normalized_count': [0.95, 0.10, 0.07, 0.05, 0.05, 0.03, 0.02, 0.01,
0.01, 0.00],
    'weighted_score': [0.286, 0.03, 0.021, 0.02, 0.016, 0.005, 0.008,
0.002, 0.002, 0.000],
    'city': ['城市 1', '城市 1', '城市 1', '城市 1', '城市 1', '城市 1', '城市
1', '城市 1', '城市 1', '城市 1']
}

# 为了对比，我们添加城市 2 的数据
```

```
data_city2 = {
    'type': ['公共厕所', '女洗手间', '男洗手间', '紧急避难场所', '公共设施',
             '报刊亭', '残障洗手间/无障碍洗手间', '公用电话', '婴儿换洗间/哺乳
室/母婴室',
             '公共设施|生活服务'],
    'count': np.random.randint(1, 2000, 10),  # 随机生成数据
    'weight': [0.3, 0.3, 0.3, 0.4, 0.3, 0.2, 0.4, 0.2, 0.2, 0.3],
    'normalized_count': np.random.rand(10),  # 随机生成标准化计数
    'weighted_score': np.random.rand(10) * 0.4,  # 随机生成加权得分
    'city': ['城市 2'] * 10  # 所有设施数据标记为城市 2
}

# 将城市 1 和城市 2 的数据合并
df = pd.DataFrame(data)
df_city2 = pd.DataFrame(data_city2)

# 合并数据
df = pd.concat([df, df_city2], ignore_index=True)

# 设置中文显示字体
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 创建一个 2 行 3 列的子图布局
fig, axes = plt.subplots(2, 3, figsize=(18, 12))
axes = axes.flatten()

# 绘制每个城市的柱状图 - 显示各设施的计数
sns.barplot(x='count', y='type', data=df[df['city'] == '城市 1'],
ax=axes[0], palette='Set2')
axes[0].set_title('城市 1 各类型设施的计数')
sns.barplot(x='count', y='type', data=df[df['city'] == '城市 2'],
ax=axes[1], palette='Set1')
axes[1].set_title('城市 2 各类型设施的计数')

# 绘制每个城市的加权得分柱状图
sns.barplot(x='weighted_score', y='type', data=df[df['city'] == '城市
1'], ax=axes[2], palette='Set2')
axes[2].set_title('城市 1 各类型设施的加权得分')
sns.barplot(x='weighted_score', y='type', data=df[df['city'] == '城市
2'], ax=axes[3], palette='Set1')
axes[3].set_title('城市 2 各类型设施的加权得分')

# 绘制散点图 - 显示每个城市的标准化计数与加权得分之间的关系
```

```
sns.scatterplot(x='normalized_count', y='weighted_score',
data=df[df['city'] == '城市 1'], hue='type', ax=axes[4], s=100)
axes[4].set_title('城市 1 标准化计数与加权得分关系')

sns.scatterplot(x='normalized_count', y='weighted_score',
data=df[df['city'] == '城市 2'], hue='type', ax=axes[5], s=100)
axes[5].set_title('城市 2 标准化计数与加权得分关系')

# 调整整体布局
plt.tight_layout()
plt.show()
```

医疗服务数据量化

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# 医疗服务数据
data = pd.DataFrame({
    'type': [
        '药房', '诊所', '医疗保健服务场所', '医疗保健用品', '口腔医院', '综合
医院',
        '卫生院', '宠物诊所', '专科医院', '整形美容', '专科医院|医疗保健服务
', '兽医站',
        '三级甲等医院', '口腔医院|医疗保健服务', '妇科医院', '骨科医院',
        '疾病预防', '眼科医院', '动物医疗场所', '诊所|医疗保健服务', '卫生院|
医疗保健服务',
        '妇科医院|医疗保健服务', '急救中心', '耳鼻喉医院', '肿瘤医院', '传染
病医院',
        '骨科医院|医疗保健服务', '药房|医疗保健服务', '宠物诊所|医疗保健服务',
        '医药保健相关', '疾病预防|医疗保健服务', '精神病医院', '三级甲等医院|
医疗保健服务',
        '综合医院|医疗保健服务', '整形美容|医疗保健服务', '眼科医院|医疗保健服
务',
        '兽医站|医疗保健服务', '医疗保健用品|医疗保健服务', '胸科医院'
    ],
    'count': [
        5240, 3172, 2802, 1895, 1331, 330, 320, 291, 190, 174, 113, 113,
        96, 81, 71, 69, 65, 61, 42, 36, 28, 17, 16, 13, 12, 10, 9, 7,
        7, 6, 5, 5, 5, 5, 4, 3, 3, 1, 1
    ]
})

# 分配类别权重
weight_dict = {
```

```
    '核心医疗设施': 0.4,  # 三级甲等医院、综合医院、急救中心等
    '专科医疗设施': 0.3,  # 专科医院、口腔医院、眼科医院等
    '基础医疗服务': 0.2, # 药房、诊所、卫生院、疾病预防等
    '其他医疗相关设施': 0.1  # 兽医站、医疗保健用品等
}

# 定义每个设施类型的权重
data['weight'] = data['type'].apply(lambda x:
    weight_dict['核心医疗设施'] if '三级甲等医院' in x or '综合医院' in x
or '急救中心' in x else
    weight_dict['专科医疗设施'] if '专科医院' in x or '口腔医院' in x or '
眼科医院' in x or '肿瘤医院' in x else
    weight_dict['基础医疗服务'] if '药房' in x or '诊所' in x or '卫生院'
in x or '疾病预防' in x else
    weight_dict['其他医疗相关设施']
)

# 标准化计数
scaler = MinMaxScaler()
data['normalized_count'] = scaler.fit_transform(data[['count']])

# 计算加权得分
data['weighted_score'] = data['normalized_count'] * data['weight']

# 计算医疗服务的总得分
total_score = data['weighted_score'].sum()

# 输出结果到CSV 文件
data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']].to_csv('Appendix_3_Medical_and_health_data.csv',
index=False)

# 显示综合得分
print(f"\n 城市 1 医疗服务的总得分: {total_score:.2f}")
```

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# 医疗服务数据
data = pd.DataFrame({
    'type': [
        '药房', '诊所', '医疗保健服务场所', '医疗保健用品', '口腔医院', '卫生
院',
```

```
        '综合医院', '宠物诊所', '口腔医院|医疗保健服务', '专科医院', '整形美容',
        '疾病预防', '眼科医院', '兽医站', '妇科医院', '骨科医院', '动物医疗场所',
        '三级甲等医院', '卫生院|医疗保健服务', '耳鼻喉医院', '药房|医疗保健服务',
        '传染病医院', '骨科医院|医疗保健服务', '医药保健相关', '专科医院|医疗保健服务',
        '精神病医院', '急救中心', '妇科医院|医疗保健服务', '肿瘤医院', '脑科医院',
        '宠物诊所|医疗保健服务', '医疗保健用品|医疗保健服务', '整形美容|医疗保健服务',
        '医疗保健服务场所|生活服务', '综合医院|医疗保健服务'
    ],
    'count': [
        2042, 1522, 891, 535, 314, 229,
        134, 94, 29, 29, 29,
        29, 22, 17, 15, 15, 15,
        14, 12, 7, 5,
        3, 3, 3, 2,
        2, 2, 2, 2, 1,
        1, 1, 1,
        1, 1
    ]
})

# 分配类别权重
weight_dict = {
    '核心医疗设施': 0.4,  # 三级甲等医院、综合医院、急救中心等
    '专科医疗设施': 0.3,  # 专科医院、口腔医院、眼科医院等
    '基础医疗服务': 0.2, # 药房、诊所、卫生院、疾病预防等
    '其他医疗相关设施': 0.1  # 兽医站、医疗保健用品等
}

# 定义每个设施类型的权重
data['weight'] = data['type'].apply(lambda x:
    weight_dict['核心医疗设施'] if '三级甲等医院' in x or '综合医院' in x or '急救中心' in x else
    weight_dict['专科医疗设施'] if '专科医院' in x or '口腔医院' in x or '眼科医院' in x or '肿瘤医院' in x else
    weight_dict['基础医疗服务'] if '药房' in x or '诊所' in x or '卫生院' in x or '疾病预防' in x else
    weight_dict['其他医疗相关设施']
)
```

```
# 标准化计数
scaler = MinMaxScaler()
data['normalized_count'] = scaler.fit_transform(data[['count']])

# 计算加权得分
data['weighted_score'] = data['normalized_count'] * data['weight']

# 计算医疗服务的总得分
total_score = data['weighted_score'].sum()

# 输出结果到CSV 文件
data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']].to_csv('Appendix_4_Medical_and_health_data.csv',
index=False)

# 显示综合得分
print(f"\n 城市 2 医疗服务的总得分: {total_score:.2f}")
```

政府及社会组织量化

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# 定义数据
data = pd.DataFrame({
    'type': [
        '乡镇以下级政府及事业单位', '政府机关相关', '乡镇级政府及事业单位', '
区县级政府及事业单位',
        '社会团体相关', '公安警察', '政府及社会团体相关', '社会治安机构', '行
业协会', '少先队',
        '省直辖市级政府及事业单位', '地市级政府及事业单位', '公检法机关', '公
证鉴定机构',
        '工商税务机构', '交通管理机构', '验车场', '法院', '教会', '消防机关',
        '慈善机构', '工商部门', '检察院', '共青团', '残联', '消费者协会', '妇
联',
        '交通执法站', '民主党派', '社会治安机构|政府机构及社会团体', '政府机关
相关|公司企业',
        '车辆管理机构', '乡镇级政府及事业单位|政府机构及社会团体', '乡镇级政府
及事业单位|生活服务',
        '区县级政府及事业单位|政府机构及社会团体', '地税机关', '地市级政府及事
业单位|政府机构及社会团体',
        '红十字会', '国税机关', '政府机关相关|科教文化服务', '交通车辆管理相关
','省直辖市级政府及事业单位|科教文化服务', '区县级政府及事业单位|公司企业', '
省直辖市级政府及事业单位|政府机构及社会团体',
```

```
        '乡镇以下级政府及事业单位|生活服务', '外国机构相关', '交通管理机构|政府机构及社会团体',
        '乡镇以下级政府及事业单位|商务住宅', '社会团体相关|公司企业', '乡镇以下级政府及事业单位|政府机构及社会团体',
        '政府机关相关|生活服务', '政府机关相关|金融保险服务', '国家级机关及事业单位', '公证鉴定机构|公司企业',
        '外地政府办', '省直辖市级政府及事业单位|生活服务', '教会|风景名胜', '慈善机构|生活服务',
        '社会团体相关|购物服务', '区县级政府及事业单位|生活服务', '验车场|汽车服务',
        '消防机关|购物服务', '省直辖市级政府及事业单位|公司企业', '共青团|科教文化服务',
        '社会团体相关|科教文化服务', '地市级政府及事业单位|公司企业', '残联|生活服务',
        '政府机关相关|政府机构及社会团体'
    ],
    'count': [
        2995, 1493, 1491, 1017, 845, 772, 598, 580, 532, 431,
        359, 283, 213, 199, 177, 149, 129, 92, 88, 88,
        85, 66, 54, 53, 43, 39, 36, 23, 21, 17, 15,
        12, 12, 12, 11, 11, 10, 10, 10, 10, 10, 8, 8, 7, 7, 7, 6, 6, 6, 5, 5, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3,
        3, 2, 2, 2, 2, 2, 2
    ]
})

# 分配类别权重
weight_dict = {
    '核心应急管理机构': 0.4,  # 公安警察、消防机关、公检法机关等
    '政府机关及管理部门': 0.3,  # 区县级政府、省直辖市级政府等
    '社会团体和公益机构': 0.2,  # 慈善机构、消费者协会等
    '其他组织和服务机构': 0.1  # 行业协会、少先队等
}

# 为每个类型分配相应的权重
data['weight'] = data['type'].apply(lambda x:
    weight_dict['核心应急管理机构'] if '公安' in x or '消防' in x or '法院' in x or '公检法' in x else
    weight_dict['政府机关及管理部门'] if '政府' in x or '机关' in x else
    weight_dict['社会团体和公益机构'] if '慈善' in x or '协会' in x else
    weight_dict['其他组织和服务机构']
)

# 标准化计数
```

```
scaler = MinMaxScaler()
data['normalized_count'] = scaler.fit_transform(data[['count']])

# 计算加权得分
data['weighted_score'] = data['normalized_count'] * data['weight']

# 计算综合得分
total_score = data['weighted_score'].sum()

# 输出结果
data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']].to_csv("Appendix_3_Government_and_social_organization
s_data.csv", index=False)

print(f"\n 城市的政府及社会组织的综合得分: {total_score:.2f}")

import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# 数据输入
data = pd.DataFrame({
    'type': [
        '乡镇以下级政府及事业单位', '政府机关相关', '区县级政府及事业单位', '
乡镇级政府及事业单位',
        '政府及社会团体相关', '公安警察', '社会团体相关', '社会治安机构',
        '地市级政府及事业单位', '省直辖市级政府及事业单位', '行业协会', '交通
管理机构',
        '公检法机关', '公证鉴定机构', '工商税务机构', '验车场', '残联', '法院
',
        '消防机关', '检察院', '工商部门', '车辆管理机构', '红十字会', '交通执
法站',
        '慈善机构', '消费者协会', '妇联', '外地政府办', '省直辖市级政府及事业
单位|政府机构及社会团体',
        '民主党派', '共青团', '国税机关', '乡镇以下级政府及事业单位|生活服务',
        '省直辖市级政府及事业单位|科教文化服务', '社会治安机构|政府机构及社会
团体',
        '乡镇级政府及事业单位|政府机构及社会团体', '区县级政府及事业单位|政府
机构及社会团体',
        '交通车辆管理相关', '区县级政府及事业单位|生活服务', '政府机关相关|公
司企业',
        '公证鉴定机构|政府机构及社会团体', '省直辖市级政府及事业单位|生活服务
',
        '乡镇以下级政府及事业单位|体育休闲服务', '乡镇以下级政府及事业单位|政
府机构及社会团体',
```

```
        '社会团体相关|金融保险服务', '地税机关', '政府机关相关|政府机构及社会团体',
        '区县级政府及事业单位|公司企业', '公安警察|政府机构及社会团体', '外国使领馆',
        '社会团体相关|购物服务', '交通管理机构|政府机构及社会团体'
    ],
    'count': [
        1902, 561, 547, 475, 360, 305, 241, 218, 209, 182, 169, 95,
        86, 76, 76, 65, 42, 42, 36, 32, 28, 24, 20, 16, 14, 11, 8, 7,
        7, 7, 6, 6, 5, 4, 4, 4, 4, 4, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2,
        2, 2, 2, 2
    ]
})

# 分配类别权重
weight_dict = {
    '核心应急管理机构': 0.4,  # 公安警察、消防机关、公检法机关等
    '政府机关及管理部门': 0.3,  # 区县级政府、省直辖市级政府等
    '社会团体和公益机构': 0.2,  # 慈善机构、消费者协会等
    '其他组织和服务机构': 0.1  # 行业协会、少先队等
}

# 定义权重规则
data['weight'] = data['type'].apply(lambda x:
    weight_dict['核心应急管理机构'] if '公安' in x or '消防' in x or '法院' in x or '公检法' in x else
    weight_dict['政府机关及管理部门'] if '政府' in x or '机关' in x else
    weight_dict['社会团体和公益机构'] if '慈善' in x or '协会' in x else
    weight_dict['其他组织和服务机构']
)

# 标准化计数
scaler = MinMaxScaler()
data['normalized_count'] = scaler.fit_transform(data[['count']])

# 计算加权得分
data['weighted_score'] = data['normalized_count'] * data['weight']

# 计算综合得分
total_score = data['weighted_score'].sum()

# 输出结果到CSV文件
```

```
data[['type', 'count', 'weight', 'normalized_count',
'weighted_score']].to_csv('Appendix_4_Government_and_social_organization
s_data.csv', index=False)

# 输出综合得分
print(f"\n 城市 2 政府及社会组织的综合得分: {total_score:.2f}")
```

公共设施对比

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

# 替换为你的文件路径
file1_path = 'Appendix_3_Public_facilities_data.csv'
file2_path = 'Appendix_4_Public_facilities_data.csv'

# 读取数据
data1 = pd.read_csv(file1_path)
data2 = pd.read_csv(file2_path)

# 设置颜色主题
sns.set_theme(style="whitegrid")

# 设置中文显示字体
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 创建一个大图窗口
fig, axes = plt.subplots(3, 2, figsize=(18, 18))

# 数据集 1 - 数量分布
sns.barplot(ax=axes[0, 0], x='type', y='count', data=data1,
palette="viridis")
axes[0, 0].set_title("数据集 1 - 各类型数量分布")
axes[0, 0].tick_params(axis='x', rotation=45)

# 数据集 1 - 加权得分分布
sns.barplot(ax=axes[1, 0], x='type', y='weighted_score', data=data1,
palette="coolwarm")
axes[1, 0].set_title("数据集 1 - 加权得分分布")
axes[1, 0].tick_params(axis='x', rotation=45)

# 数据集 1 - 归一化数量分布
```

```
sns.barplot(ax=axes[2, 0], x='type', y='normalized_count', data=data1,
palette="rocket")
axes[2, 0].set_title("数据集 1 - 归一化数量分布")
axes[2, 0].tick_params(axis='x', rotation=45)

# 数据集 2 - 数量分布
sns.barplot(ax=axes[0, 1], x='type', y='count', data=data2,
palette="viridis")
axes[0, 1].set_title("数据集 2 - 各类型数量分布")
axes[0, 1].tick_params(axis='x', rotation=45)

# 数据集 2 - 加权得分分布
sns.barplot(ax=axes[1, 1], x='type', y='weighted_score', data=data2,
palette="coolwarm")
axes[1, 1].set_title("数据集 2 - 加权得分分布")
axes[1, 1].tick_params(axis='x', rotation=45)

# 数据集 2 - 归一化数量分布
sns.barplot(ax=axes[2, 1], x='type', y='normalized_count', data=data2,
palette="rocket")
axes[2, 1].set_title("数据集 2 - 归一化数量分布")
axes[2, 1].tick_params(axis='x', rotation=45)

# 调整布局以防止重叠
plt.tight_layout()
plt.show()
```

交通设施对比

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

# 替换为你的文件路径
file1_path = 'Appendix_3_Transportation_facilities_data_translated.csv'
file2_path = 'Appendix_4_Transportation_facilities_data_translated.csv'

# 读取数据
data1 = pd.read_csv(file1_path)
data2 = pd.read_csv(file2_path)

# 设置颜色主题
sns.set_theme(style="whitegrid")

# 设置中文显示字体
```

```
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 创建一个大图窗口
fig, axes = plt.subplots(3, 2, figsize=(18, 18))

# 数据集1 - 数量分布
sns.barplot(ax=axes[0, 0], x='type', y='count', data=data1,
palette="viridis")
axes[0, 0].set_title("City1 - Quantity distribution of each type")
axes[0, 0].tick_params(axis='x', rotation=45)

# 数据集1 - 加权得分分布
sns.barplot(ax=axes[1, 0], x='type', y='weighted_score', data=data1,
palette="coolwarm")
axes[1, 0].set_title("City1 - Weighted score distribution")
axes[1, 0].tick_params(axis='x', rotation=45)

# 数据集1 - 归一化数量分布
sns.barplot(ax=axes[2, 0], x='type', y='normalized_count', data=data1,
palette="rocket")
axes[2, 0].set_title("City1 - Normalized quantity distribution")
axes[2, 0].tick_params(axis='x', rotation=45)

# 数据集2 - 数量分布
sns.barplot(ax=axes[0, 1], x='type', y='count', data=data2,
palette="viridis")
axes[0, 1].set_title("City2 - Quantity distribution of each type")
axes[0, 1].tick_params(axis='x', rotation=45)

# 数据集2 - 加权得分分布
sns.barplot(ax=axes[1, 1], x='type', y='weighted_score', data=data2,
palette="coolwarm")
axes[1, 1].set_title("City2 - Weighted score distribution")
axes[1, 1].tick_params(axis='x', rotation=45)

# 数据集2 - 归一化数量分布
sns.barplot(ax=axes[2, 1], x='type', y='normalized_count', data=data2,
palette="rocket")
axes[2, 1].set_title("City2 - Normalized quantity distribution")
axes[2, 1].tick_params(axis='x', rotation=45)

# 调整布局以防止重叠
plt.tight_layout()
```

```
plt.show()
```

医疗服务设施对比

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

file1_path = 'Appendix_3_Medical_and_health_data.csv'
file2_path = 'Appendix_4_Medical_and_health_data.csv'

# 读取数据
data1 = pd.read_csv(file1_path)
data2 = pd.read_csv(file2_path)

# 设置颜色主题
sns.set_theme(style="whitegrid")

# 设置中文显示字体
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 创建一个大图窗口
fig, axes = plt.subplots(3, 2, figsize=(18, 18))

# 数据集 1 - 数量分布
sns.barplot(ax=axes[0, 0], x='type', y='count', data=data1,
palette="viridis")
axes[0, 0].set_title("数据集 1 - 各类型数量分布")
axes[0, 0].tick_params(axis='x', rotation=45)

# 数据集 1 - 加权得分分布
sns.barplot(ax=axes[1, 0], x='type', y='weighted_score', data=data1,
palette="coolwarm")
axes[1, 0].set_title("数据集 1 - 加权得分分布")
axes[1, 0].tick_params(axis='x', rotation=45)

# 数据集 1 - 归一化数量分布
sns.barplot(ax=axes[2, 0], x='type', y='normalized_count', data=data1,
palette="rocket")
axes[2, 0].set_title("数据集 1 - 归一化数量分布")
axes[2, 0].tick_params(axis='x', rotation=45)

# 数据集 2 - 数量分布
```

```
sns.barplot(ax=axes[0, 1], x='type', y='count', data=data2,
palette="viridis")
axes[0, 1].set_title("数据集 2 - 各类型数量分布")
axes[0, 1].tick_params(axis='x', rotation=45)

# 数据集 2 - 加权得分分布
sns.barplot(ax=axes[1, 1], x='type', y='weighted_score', data=data2,
palette="coolwarm")
axes[1, 1].set_title("数据集 2 - 加权得分分布")
axes[1, 1].tick_params(axis='x', rotation=45)

# 数据集 2 - 归一化数量分布
sns.barplot(ax=axes[2, 1], x='type', y='normalized_count', data=data2,
palette="rocket")
axes[2, 1].set_title("数据集 2 - 归一化数量分布")
axes[2, 1].tick_params(axis='x', rotation=45)

# 调整布局以防止重叠
plt.tight_layout()
plt.show()
```

政府及社会组织对比

```
import matplotlib.pyplot as plt
import pandas as pd

# 设置中文显示字体
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# Replace these paths with the actual file paths in your environment
file1_path = 'Appendix_3_Government_and_social_organizations_data.csv'
file2_path = 'Appendix_4_Government_and_social_organizations_data.csv'

# Load the data
data1 = pd.read_csv(file1_path)
data2 = pd.read_csv(file2_path)

# Plot 1: Count by Type (Dataset 1)
data1_sorted = data1.sort_values(by='count', ascending=False)
plt.figure(figsize=(12, 8))
plt.barh(data1_sorted['type'], data1_sorted['count'], color='skyblue')
plt.title('Dataset 1: Count by Type')
plt.xlabel('Count')
plt.ylabel('Type')
```

```
plt.tight_layout()
plt.show()

# Plot 2: Weighted Score by Type (Dataset 1)
plt.figure(figsize=(12, 8))
plt.barh(data1_sorted['type'], data1_sorted['weighted_score'],
color='orange')
plt.title('Dataset 1: Weighted Score by Type')
plt.xlabel('Weighted Score')
plt.ylabel('Type')
plt.tight_layout()
plt.show()

# Plot 3: Count by Type (Dataset 2)
data2_sorted = data2.sort_values(by='count', ascending=False)
plt.figure(figsize=(12, 8))
plt.barh(data2_sorted['type'], data2_sorted['count'],
color='lightgreen')
plt.title('Dataset 2: Count by Type')
plt.xlabel('Count')
plt.ylabel('Type')
plt.tight_layout()
plt.show()

# Plot 4: Weighted Score by Type (Dataset 2)
plt.figure(figsize=(12, 8))
plt.barh(data2_sorted['type'], data2_sorted['weighted_score'],
color='pink')
plt.title('Dataset 2: Weighted Score by Type')
plt.xlabel('Weighted Score')
plt.ylabel('Type')
plt.tight_layout()
plt.show()

# Plot 5: Normalized Count (Dataset 1)
plt.figure(figsize=(12, 8))
plt.barh(data1_sorted['type'], data1_sorted['normalized_count'],
color='purple')
plt.title('Dataset 1: Normalized Count by Type')
plt.xlabel('Normalized Count')
plt.ylabel('Type')
plt.tight_layout()
plt.show()
```

```
# Plot 6: Normalized Count (Dataset 2)
plt.figure(figsize=(12, 8))
plt.barh(data2_sorted['type'], data2_sorted['normalized_count'],
color='brown')
plt.title('Dataset 2: Normalized Count by Type')
plt.xlabel('Normalized Count')
plt.ylabel('Type')
plt.tight_layout()
plt.show()
```