Problem Chosen	2024 ShuWei Cup Summary Sheet	Team Control Number
D		2024110930667

Urban Resilience and Sustainable Multidimensional evaluation of Development Capacity

Summary

With China experiencing its first negative population growth in 2022 and facing an accelerating aging trend, the country is entering a period of long-term population decline. This demographic shift poses significant challenges to urban sustainable development. This study establishes a multi-level urban capacity assessment system based on empirical data, aiming to provide scientific evidence for sustainable urban development strategies.

For question 1, we first introduce skewness, kurtosis and coefficient of variation to perform descriptive statistics on the data, and then apply **cubic spline interpolation** to missing values and quartile method to eliminate outliers according to **distribution characteristics.** Then, the strategies **combining correlation coefficient and VIF** are used for feature screening. Then, XGBoost, CatBoost and LightGBM are used as baseline models and in combination with **Optuna hyper-parameter optimization** method, three optimal baseline models are obtained. The integrated strategy of **Stacking** is used to predict housing prices. The final result was **R2:0.77**. In the process of estimating the total housing stock, we **carried out stratified statistics according to building types, and adopted Bootstrap method for multiple resampling**. Finally, we also carried out **interval estimation** and calculation of coefficient of variation to evaluate the stability of estimation. **The estimated coefficient of variation of both cities was 0.01, which verified the stability of our estimation method.**

For question 2, we construct a **five-level and 15-item multi-level evaluation index system**. At the top level, five main indicators are set, namely basic service capability (weight 1.2), spatial accessibility (weight 1.0), service diversity (weight 0.8), service balance (weight 1.1), and service efficiency (weight 0.9). There are three specific evaluation indicators under each main indicator, forming a complete evaluation framework. Based on this, our evaluation process is divided into three main steps: First, each specific indicator is standardized to eliminate the impact of dimensions; Secondly, through arithmetic average method, all the main indicators are integrated, and the big index score is obtained by combining the hierarchical weights. Finally, **thresholds were set based on score differences (significant advantage >0.2, slight advantage >0.1)** to identify strengths and weaknesses between cities. We not only compare the overall performance of cities in various service types, but also analyze the causes of differences from different levels. The final conclusion is that **City 1 shows significant advantages in several service areas, especially in health care, accommodation services and science, education and culture, while City 2 performs better in spatial accessibility and service diversity, but lacks significant advantages in service.**

For question 3, we evaluated from three dimensions: **health accessibility index , emergency response capability and sustainable development capability ,** The method of **spatial analysis, statistical analysis** (density analysis, coefficient of variation analysis, percentile analysis) and comprehensive evaluation method (**entropy weight method** to determine the weight, weighted comprehensive score) were used. The final conclusion is that **city 1 is superior to city 2 in terms of the number, density, coverage, emergency response time and sustainable development potential of medical facilities, showing stronger urban resilience and development potential**. However, **both cities still have room for improvement, especially when it comes to the balanced distribution of medical resources, both cities score relatively low.**

For question 4, we synthesize the analysis results of the first three questions, and give reasonable investment suggestions.

Finally, we evaluate the advantages and disadvantages of the model.

Keywords: City capacity assessment； Ensemble learning; Statistical analysis; Multi-level evaluation system

目录

1 Introduction

1.1 Background

China's demographic landscape is undergoing a significant transformation, characterized by an intensifying aging population trend and the first occurrence of negative population growth in 2022. This demographic shift is expected to result in a prolonged population decline, which will have profound implications for the high-quality and sustainable development of urban areas, particularly secondary and tertiary cities. These cities are likely to face multifaceted challenges, including economic pressures, infrastructure strain, and social service demands.

Simultaneously, the increasing frequency of extreme global climate events, coupled with the current economic downturn, presents unprecedented challenges to the resilience of urban development. Urban resilience refers to a city's ability to withstand, adapt to, and recover from shocks and stresses, such as natural disasters, economic downturns, and social disruptions. Sustainable development capacity, on the other hand, encompasses a city's ability to meet current needs without compromising the ability of future generations to meet their own needs.

In this context, the evaluation of urban resilience and sustainable development capacity is crucial for policymakers to make informed decisions. To facilitate this evaluation, we have collected data on property listings from City 1 and City 2 on a specific day from 58.com (Appendix 1 and 2), as well as Point of Interest (POI) data for basic services in these cities for a particular year (Appendix 3 and 4). These data sets provide a foundation for analyzing the current state of these cities and for developing projections and models to guide future development.

1.2 Work

- Problem (1): Housing Price Projections and Housing Stock Estimation

Given the demographic and economic trends, we will develop specific projections of future housing prices in different areas of City 1 and City 2. Additionally, we will provide specific estimates of the total existing housing stock in these two cities, taking into account the data above and any additional data that can be collected from the internet, such as population and GDP.

- Problem (2): Quantitative Analysis of Service Levels

We will conduct a quantitative analysis of the service levels in different sectors of City 1 and City 2. Based on this analysis, we will extract the common and unique characteristics of these two cities, along with their respective strengths and weaknesses. This analysis will provide insights into how the cities compare in terms of service provision and where each city stands in relation to the other.

- Problem (3): Resilience and Sustainable Development Capacity Assessment

Using the data in Appendix 3 and 4, we will assess the resilience of both cities in responding to extreme weather and emergencies. Additionally, we will quantitatively evaluate the cities' sustainable development capacity. Our model will explicitly identify the specific weaknesses of both cities, the key areas that require development in the future, and the short-term and long-term investment plans for the cities, given the constraints of limited financial resources.

- Problem (4): Future Development Plan

Based on the analysis above, we will draft a future development plan for the two cities, not exceeding two pages. The plan will clearly outline the primary investment areas, investment amounts, and the expected improvements in the cities' level of smart

city development. This plan will be actionable and aligned with the identified strengths, weaknesses, and development needs of each city.

2 Problem analysis

2.1 Analysis of Question I

First of all, we need to conduct data pre-processing, and then carry out detailed feature engineering. Then, we first use a single model to forecast the housing price and observe the effect, and then carry out hyperparameter optimization and integrated learning to improve the prediction effect. For the estimation of the total number of households, we can use stratified sampling method to perform point estimation and interval estimation

2.2 Analysis of Question II

For the solution of problem 2, we should first conduct pre-processing, and then build a multi-level evaluation index system, set several small indicators as basic indicators, integrate them into several large indicators, and conduct threshold analysis according to the score differences to identify the advantages and disadvantages between cities. Through this method, We can not only analyze the overall strength and weakness of the city in each service type, but also explore the different performance of different evaluation levels in depth.

2.3 Analysis of Question III

For problem 3, namely, assessing the resilience of cities 1 and 2 in coping with extreme weather and emergency situations and quantitatively assessing the sustainable development capacity of these two cities, we can conduct health accessibility index evaluation, emergency response capacity index evaluation, and sustainability capacity index evaluation can adopt spatial analysis and statistical analysis methods. The comprehensive evaluation method combines the way to carry out the comprehensive evaluation

2.4 Analysis of Question IV

In response to the fourth question, we should synthesize the analysis results of the first three questions to put forward reasonable investment suggestions for multiple aspects of the two cities

3 Symbol and Assumptions

3.1 Symbol Description

Symbol	*Description*
S_k	*Skewness*
K_u	*Kurtosis*
CV	*Coefficient of Variation*
$S_i(x)$	*cubic polynomial*
$POI_{count}(R)$	*Total POI Count*
r	*Correlation Coefficient*
A	*Urban area*
D	*Density*

3.2 Fundamental assumptions

Assumption 1: Data is complete and accurate. All data used for the model is complete and accurate, without a large number of missing values and outliers. Even if there are a few missing values, they can be reasonably filled by interpolation methods such as cubic spline interpolation. In addition, the data collection and recording process is standardized to ensure the consistency and reliability of the data.

Assumption 2: Environmental conditions are stable. When housing price forecasts and city service assessments are made, environmental conditions (e.g., climate, policy, economic conditions, etc.) are assumed to remain relatively stable over a certain period of time.

Assumption 3: Socioeconomic factors are relatively stable. In assessing urban resilience and sustainability, it is assumed that socioeconomic factors (such as demographics, income levels, education levels, etc.) remain relatively stable over a period of time. The impact of these factors on urban services and resilience will not change dramatically in the short term.

4 Models and Test

4.1 Question I: The establishment of housing price forecasting model

4.1.1 Descriptive statistics

First, we make descriptive statistics on the given housing price data. This paper introduces the mean value, maximum value, minimum value, median value, standard deviation, skewness coefficient, kurtosis coefficient and coefficient of variation to describe the statistical data:

(1)Skewness coefficient

Skewness is a measure of the skewness of a data distribution. $S_k = 0$ Indicates that the data is approximately symmetrical, $S_k > 0$ indicates that the data is right-biased, and $S_k < 0$indicates that the data is left-biased. The larger the absolute value of

the skewness coefficient is, the more obvious the skewness of the data distribution is. Its calculation formula is as follows:

$$S_k = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mu^3}{\sigma^3}$$

(2) kurtosis coefficient

The Kurtosis coefficient is used to measure the degree of kurtosis of a data distribution. $K_u = 0$ indicates that the data distribution is normal, $K_u < 0$ indicates that the data distribution has a smaller kurtosis and the data is more dispersed, and $K_u > 0$ indicates that the data distribution has a larger kurtosis and the data is more concentrated. The greater the absolute value of the kurtosis coefficient, the more obvious the kurtosis degree of the data distribution. Its calculation formula is as follows:

$$K_u = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{\mu^4}{\sigma^4}$$

(3) Coefficient of variation

The coefficient of variation is a statistical metric used to measure the proportion of the dispersion of a set of data to the mean. It provides a unitless measure by comparing the standard deviation to the mean, and is therefore particularly suitable for comparing the degree of dispersion of data from different units or different scales. Its calculation formula is as follows:

$$CV = \frac{\sigma}{\mu} \times 100\%$$

Taking the data of City 1 as an example, the descriptive statistical results are as follows：

Table 1 Descriptive statistics of cities 1

	count	mean	std	min	25%	50%	75%	max	Kurtosis	Skewness	Coefficient of Variation
Community Number	46933	2347	1354.894737	1	1174	2347	3520	4693	-1.2	0	0.577288767
Price (USD)	46003	71750.46711	3256.079428	1387	5400	6900	8226	48945	21.22629281	2.845177893	0.454093481
Total number of households	42288	717.0868023	855.5082003	1	147.75	409.5	960	7815	9.000640654	2.506102624	1.193032974
Greening rate	40200	0.30208296	0.07027393 1	0.1	0.3	0.3	0.33	0.8	3.509122897	0.882668462	0.232636123
Floor area ratio	40333	1.669925614	0.61051959 1	0.3	1.2	1.5	2	7	6.058527211	1.550595954	0.365596878
above-ground parking fee（/month USD）	39881	184.1085154	93.52589076	10	150	150	200	650	2.263178184	1.369451103	0.507993291

underground parking fee（/ month USD）	915	326.410929	161.0721038	15	200	300	400	1000	7.951933908	2.389167696	0.493464187
citycode	46931	431	0	431	431	431	431	431	0	0	0
adcode	46931	220125.2118	31.58657525	220102	220104	220106	220171	220183	-0.989213658	0.956285154	0.000143494
lon	46931	125.399591	0.27691515	124.175035	125.285637	125.323684	125.386499	126.954552	9.273079081	2.859317542	0.002208262
lat	46931	43.9547311 8	0.25598531 8	43.345598	43.848362	43.881388	43.92567	45.1394287 9	4.432906099	2.109567 11	0.00582384
X	46931	68.2648550 1	15.4370831 2	0	61.9123056 8	64.0332972 7	67.5350210 9	154.948673	9.273079081	2.859317542	0.226135149
Y	46931	33.9571146 6	14.2703157 7	0	28.0273927 6	29.8684805 7	32.3370523 2	100	4.432906099	2.109567 11	0.420245239

The distribution of Community Number is relatively concentrated, with a range from 1 to 4693, close to a normal distribution but slightly flat. The Price (USD) data is widely dispersed, ranging from 1387 to 48945, with a right skew and many extreme values. The Total number of households data is also widely dispersed, ranging from 1 to 7815, with a right skew and many extreme values. The Greening rate data is relatively concentrated, ranging from 0.1 to 0.8, with a slight right skew. The Floor area ratio data is widely dispersed, ranging from 0.3 to 7, with a right skew and many extreme values. The Above-ground parking fee and Underground parking fee data are widely dispersed, ranging from 10 to 650 and 15 to 1000 respectively, with a right skew and many extreme values. The Citycode data is entirely consistent, indicating that all records belong to the same city. The Adcode data is relatively concentrated, ranging from 220102 to 220183, with a slight right skew. The Lon and Lat data are relatively concentrated, with small ranges, and a right skew with a few extreme values. The X and Y coordinate data are widely dispersed, ranging from 0 to 154.948673 and 0 to 100 respectively, with a right skew and many extreme values.

Overall, some variables in the dataset, such as Community Number, Greening rate, Lon, and Lat, are relatively concentrated, while others, such as Price, Total number of households, Floor area ratio, and parking fees, are widely dispersed with many extreme values. These characteristics provide a crucial foundation for subsequent analysis and modeling.

4.1.2 Data preprocessing and POI feature extraction

(1)Missing values and outliers

• Cubic Spline Interpolation

Cubic spline interpolation is a mathematical method used for estimating missing values by constructing a smooth curve between data points. This method uses cubic polynomials between each pair of data points, ensuring that the interpolated curve is continuous in both the first and second derivatives at the data points.

Suppose we have n data points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ .where x_i is the independent variable and y_i is the dependent variable. Cubic spline interpolation uses a cubic polynomial $S_i(x)$ within each interval $[x_i, x_{i+1}]$:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

• Interquartile Range (IQR) Method

The Interquartile Range (IQR) method is a statistical technique used for detecting and removing outliers. It is based on the quartiles (Q1, Q2, Q3) of the data to determine the distribution range and identify values outside this range as outliers.

In view of the distribution characteristics of the data, **we use cubic spline interpolation for the data that can be interpolated, otherwise we will directly eliminate it, and then we use the quartile method to eliminate the outlier** to lay the foundation for the subsequent prediction

(2)POI feature extraction

POI (Point of Interest) features refer to the number or distribution of specific types of points of interest within a geographic area. These features can be used to describe the convenience, quality of life, and level of commercial activity in a region. In the real estate sector, POI features are crucial for predicting housing prices, as the facilities and services surrounding a house directly impact the living experience and property value.

• Total POI Count

The total number of all POIs within a given radius R around the house.

$$POI_{count}(R) = \sum_{i=1}^{N} I_{\{d_i \leq R\}}$$

Where：

N is the total number of POIs.

d_i is the distance between the i-th POI and the house location.

$\mathrm{I}_{\{d_i \leq R\}}$ is an indicator function that equals 1 if $d_i \leq R$, otherwise 0.

• Categorized POI Count

The number of specific types of POIs within a given radius R around the house.

$$POI_{count}^{type}(R) = \sum_{i=1}^{N_{type}} I_{\{d_i \leq R\}}$$

Where：

N_{type} is the total number of POIs of a specific type.

d_i is the distance between the i-th POI of the specific type and the house location.

$\mathrm{I}_{\{d_i \leq R\}}$ is an indicator function that equals 1 if $d_i \leq R_r$ otherwise 0.

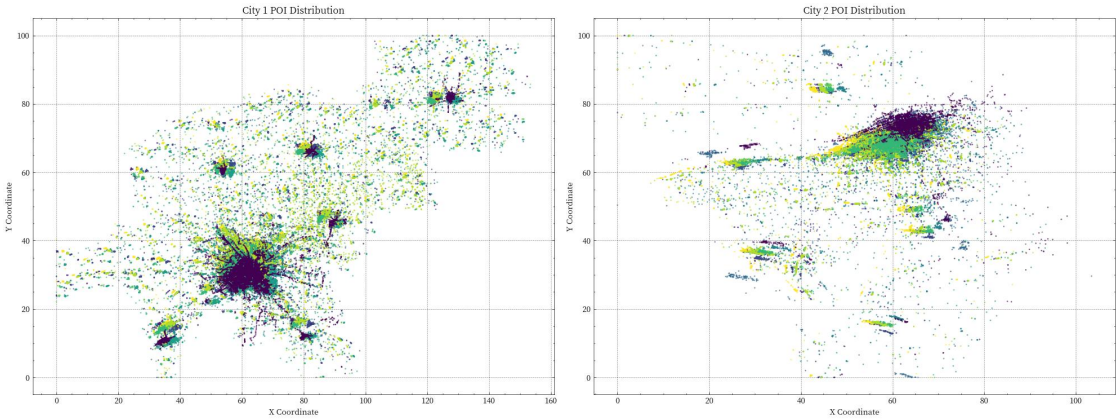

Figure 1 POI feature visualization

These two charts show the POI (Point of Interest) distribution of the two cities. By comparing the POI distribution of the two cities, we can observe some significant differences, which reflect the characteristics of the two cities in terms of city scale, layout, distribution of service facilities and so on.

• City 1 POI distribution

Overall distribution: The POI distribution of City 1 is relatively extensive, covering the entire map area. This indicates that the city has a wealth of facilities and services in various areas and may be a larger or more evenly developed city. This wide distribution may mean that city 1 has more balanced infrastructure and public services in different areas, and residents can enjoy relatively convenient living conditions in different places.

Dense area: There is an obvious dense area in the center of the map, and the color is darker, indicating that the number of POI in this area is very high. This may be the commercial or residential core of the city, where a large number of commercial facilities, public services and entertainment venues are concentrated. This high density POI distribution may reflect the economic development and population concentration in City 1.

Marginal areas: POI in marginal areas are less distributed and lighter in color, indicating that there are relatively few service facilities in these places. This may mean that the peripheral areas of City 1 are relatively underdeveloped, and the infrastructure and service facilities are not as complete as the central areas. However, this may also provide potential space for future urban expansion and development.

• City 2 POI distribution

Overall distribution: In contrast, the POI distribution in City 2 is more concentrated in some specific areas rather than evenly distributed throughout the city. This may reflect the city's more centralized development, with resources and facilities concentrated in specific areas. This centralized distribution may mean that City 2 has highly developed infrastructure and services in some areas, while others are relatively backward.

Dense areas: On the right side of the map, there is a distinct dense band, which varies in color from dark to light, indicating a very high POI density in this area. This may be the main street of a city or the main commercial district, where a large number of commercial facilities, public services and entertainment venues are concentrated. This high density of POI distribution may reflect the economic development and population concentration in City 2, especially in specific areas.

Sparse areas: Most of the other areas are lighter in color, indicating that there is less POI in these areas. This may mean that other areas of City 2 are relatively

underdeveloped, with less developed infrastructure and services than dense areas. However, this may also provide potential space for future urban expansion and development.

City size and layout: **The POI of city 1 is more widely distributed, which may be a larger or more balanced city; City 2, on the other hand, is more compact and probably a more concentrated city**. City 1 has more balanced infrastructure and public services in different areas, while City 2 has a high concentration of service facilities in specific areas.

Service facilities: City 1 provides more service facilities throughout the scope, while City 2 has a high concentration of service facilities in a specific area. The wide distribution of city 1 May mean that residents enjoy relatively convenient living conditions in different places, while the centralized distribution of city 2 May mean that residents have better living conditions in certain areas, but are relatively backward in other areas.

Future development potential: **the marginal areas of City 1 and the sparse areas of City 2 May provide potential space for future urban expansion and development. The development potential of these areas may depend on urban planning and resource allocation strategies.**

4.1.3 Feature engineering

• Correlation Coefficient

Correlation Coefficient is an indicator used in statistics to measure the strength and direction of the linear relationship between two variables. It can help us understand the degree of correlation between two variables, that is, whether a change in one variable is associated with a change in the other variable. The value of the correlation coefficient ranges from -1 to 1, which is explained as follows:

The correlation coefficient is usually represented by symbols, and its formula is as follows:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

We first used the correlation coefficient to select the top 20 most correlated with the price column, and the results are as follows：

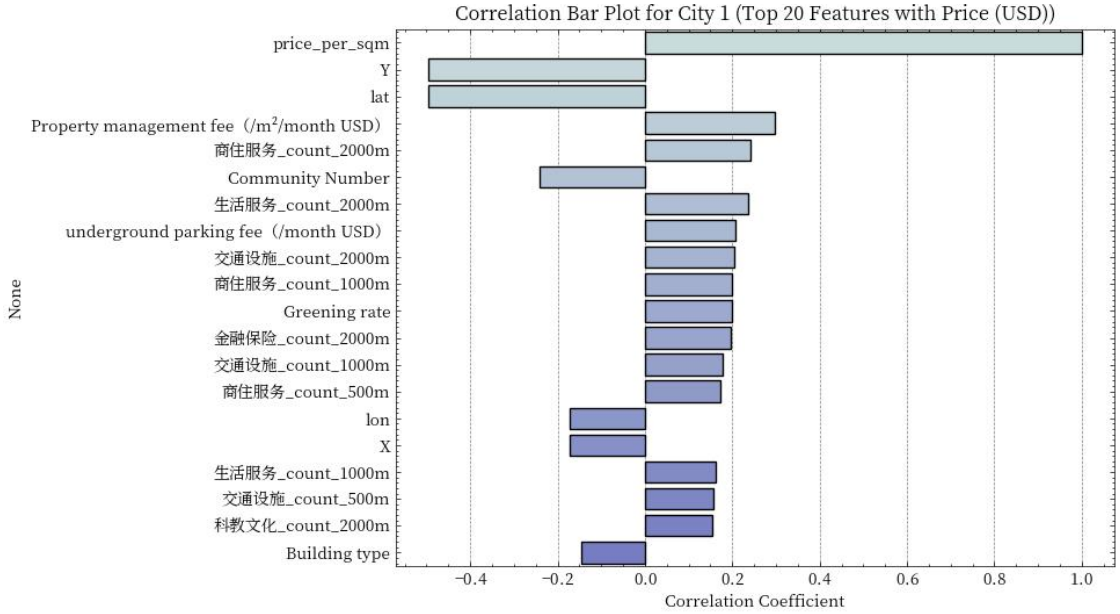

Figure 2 Features of city 1TOP20

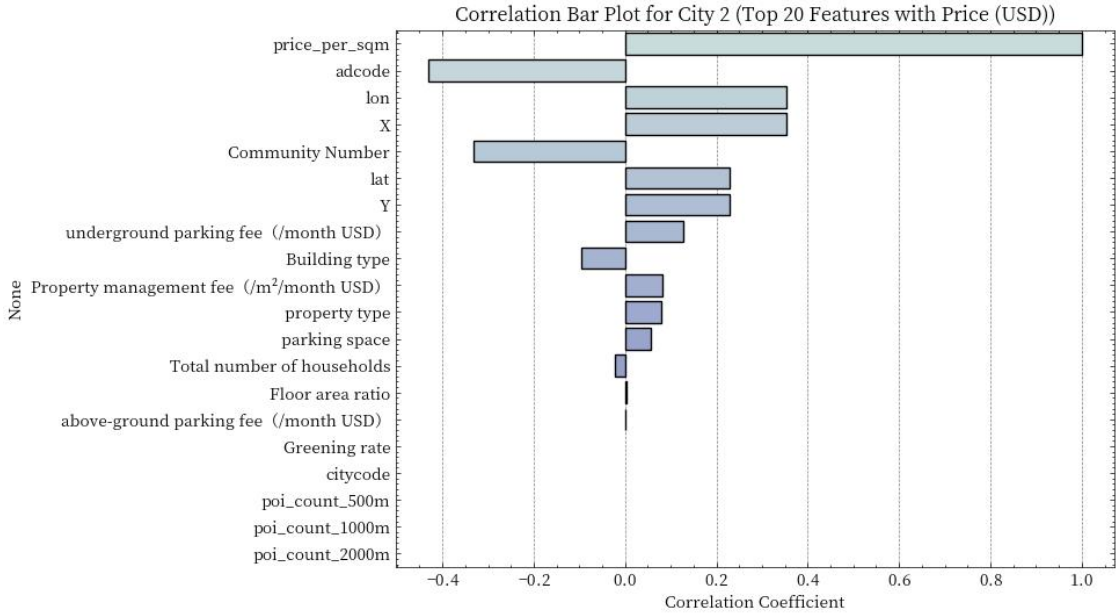

Figure 3 Features of city 2TOP20

After observation, we found that we only need to extract the characteristics of the top 13 cities

• Variance inflation factor (VIF) interpretation

Variance Inflation Factor (VIF) is a measure of Multicollinearity in statistics. Multicollinearity means that in regression analysis, there is a high correlation between the independent variables, resulting in unstable and unreliable estimation of the regression coefficient.

Finally, we adopt the following feature screening strategy to select features that are highly correlated with target variables but can avoid collinearity problems as the final prediction

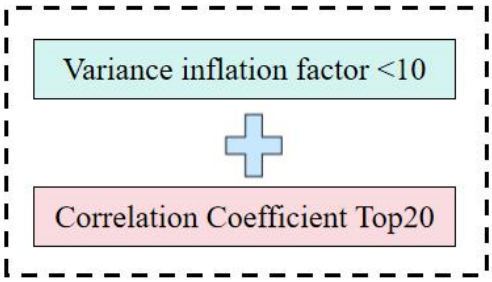

Figure 4 Feature screening strategy

4.1.4 Housing price forecast

We first adopt XGBoost, CatBoost and LightGBM as the base learner to explore the single-mode model for the housing price of each city

XGBoost (Extreme Gradient Boosting)

XGBoost is an efficient, flexible and scalable gradient lift framework developed by Tianqi Chen. It gradually builds and integrates multiple decision trees by optimizing the loss function to improve the predictive performance of the model. XGBoost supports parallel and distributed computing, automatically handles missing values, and is suitable for large-scale data sets and high-dimensional features. Its main features include regularization (such as L1 and L2 regularization), parallel computing, missing value processing, and flexibility (support for multiple loss functions and evaluation metrics).

CatBoost (Categorical Boosting)

CatBoost is a gradient lift framework developed by Yandex that is particularly suitable for handling classification features. It improves the predictive performance of the model by automatically processing classification features and reducing overfitting. CatBoost uses the Ordered Boosting method to reduce overfitting, supports parallel computation, automatically handles missing values, and is suitable for a variety of loss functions and evaluation indicators. Its main features include automatic processing of classification features, sorting promotion, parallel computation and missing value processing.

LightGBM (Light Gradient Boosting Machine)

LightGBM is a gradient lifting framework developed by Microsoft. By using Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) techniques, the model training speed and efficiency are improved. LightGBM supports parallel and distributed computing, automatically handles missing values, and is suitable for large-scale data sets and high-dimensional features. Its key features include high efficiency (using GOSS and EFB technologies), parallel computing, missing value processing, and flexibility (supporting multiple loss functions and evaluation metrics).

Table 2 Prediction results of single model

Model	RMSE	MAE	R2	City
XGBoost	**2424.838628**	**1044.997385**	**0.672581**	**城市1**
LightGBM	2452.86742	1074.882534	0.659573	城市1
CatBoost	2542.616391	1150.694831	0.616489	城市1
XGBoost	**3033.515227**	**1523.474539**	**0.468406**	**城市2**
LightGBM	3053.825863	1544.460262	0.457765	城市2
CatBoost	3069.585259	1574.469994	0.449761	城市2

We then used optuna for hyperparametric tuning of the three baseline models：

Optuna is an open source framework for hyperparameter optimization, especially for the tuning of machine learning and deep learning models. It uses techniques such as Bayesian optimization, random search and evolutionary algorithm to automatically search for the optimal combination of hyperparameters to improve the performance of the model.Below we show some parameter optimization results：

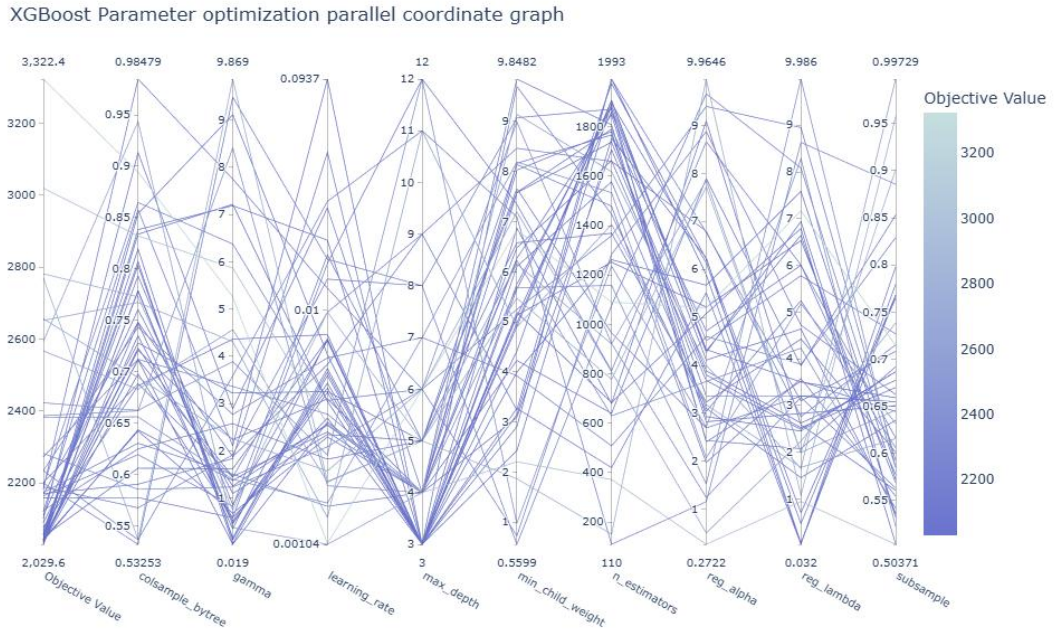

Figure 5 City 1 XGBoost optimization results

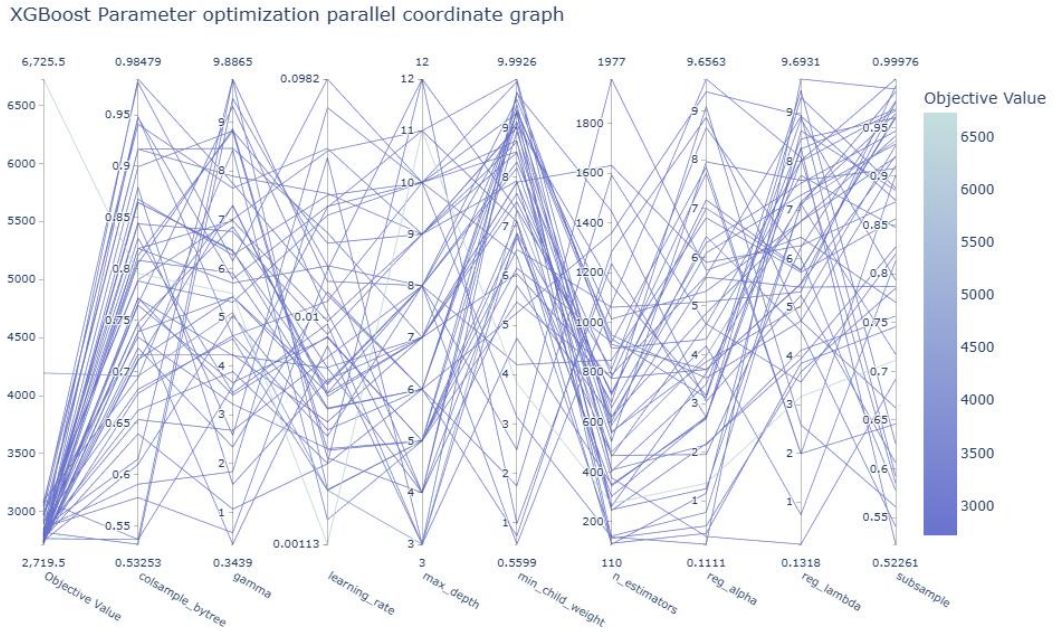

Figure 6 City 2 XGBoost optimization results

Then we use the optimal hyperparameters to train the model again, and the results of the 5-fold cross-validation are as follows：

Table 3 Results of single-mode prediction after hyperparameter tuning

Model	RMSE	MAE	R2	City
XGBoost	**2029.13515**	**841.153515**	**0.745111**	**城市1**
LightGBM	2215.45487	941.784415	0.721533	城市1
CatBoost	2311.158911	1015.154677	0.696451	城市1
XGBoost	**2719.654687**	**1523.474539**	**0.581154**	**城市2**

LightGBM	2815.854647	1596.153479	0.551534	城市2
CatBoost	2913.156353	1678.416547	0.491577	城市2

It can be seen from the results that hyperparameter optimization plays a role in improving the performance of the model. Below, we use the method of ensemble learning to integrate the prediction results of three base learners

Stacking is an ensemble learning method that improves the prediction performance of models by combining the prediction results of multiple Base Learners. The core idea of Stacking is to use a Meta Learner to learn how to combine the predictions of the base learner to get the final predictions. The following is a detailed explanation of Stacking and gives mathematical expressions that use XGBoost, CatBoost, and LightGBM as base learners. It usually consists of the following steps:

• Training based learners: Train multiple based learners using training data.

• Generate prediction results: Use the trained base learner to predict the training data and test data and generate prediction results.

• Train a meta-learner: Using the predictions of the base learner as input, train a meta-learner to learn how to combine those predictions.

• Generate the final prediction result: Use the trained meta-learner to combine the prediction results of the base learner to generate the final prediction result.

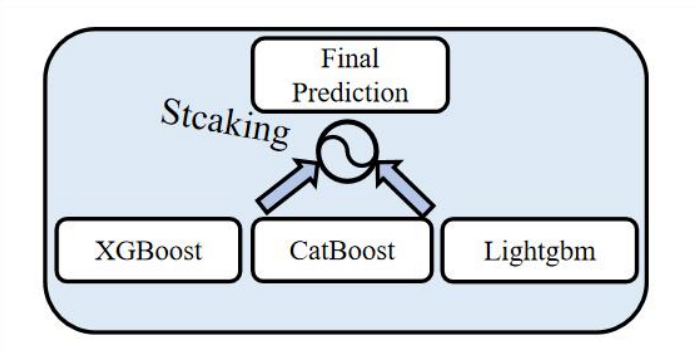

Figure 7 Stacking

The final predictions are as follows：

Table 4 Results of ensemble learning prediction

Model	RMSE	MAE	R2	City
XGBoost	2029.13515	841.153515	0.745111	城市1
LightGBM	2215.45487	941.784415	0.721533	城市1
CatBoost	2311.158911	1015.154677	0.696451	城市1
Stacking	**1821.135435**	**741.896431**	**0.771635**	**城市1**
XGBoost	2719.654687	1523.474539	0.585154	城市2
LightGBM	2815.854647	1596.153479	0.556534	城市2
CatBoost	2913.156353	1678.416547	0.491577	城市2
Stacking	**2413.498746**	**1145.486764**	**0.661354**	**城市2**

4.1.5 Estimates of the total existing housing

In the process of estimating the total housing stock, we first carried out basic statistics and calculated the number of communities and total households in the sample, which are total_communities and total_households respectively. Then, **stratified statistics were carried out according to the building type**, and the number, total number, average number and standard deviation of each building type were calculated. In order to improve the accuracy of estimation, **Bootstrap method was adopted for**

multiple resampling. Specifically, Bootstrap is set up 1000 times and assumes a data coverage of 0.3. In each Bootstrap, stratified samples were taken by building type and resampled within each floor. For the resampling results of each floor, the total number of households was calculated and adjusted for coverage. Add the adjusted total number of households for all tiers to get the total estimate for one Bootstrap and store these estimates in bootstrap_estimates. Next, the mean estimated_total and 95% confidence interval confidence_interval of the Bootstrap estimate were calculated. After that, the **coefficient of variation of the Bootstrap estimate** was also calculated to evaluate the stability of the estimate.

The estimated results are as follows：

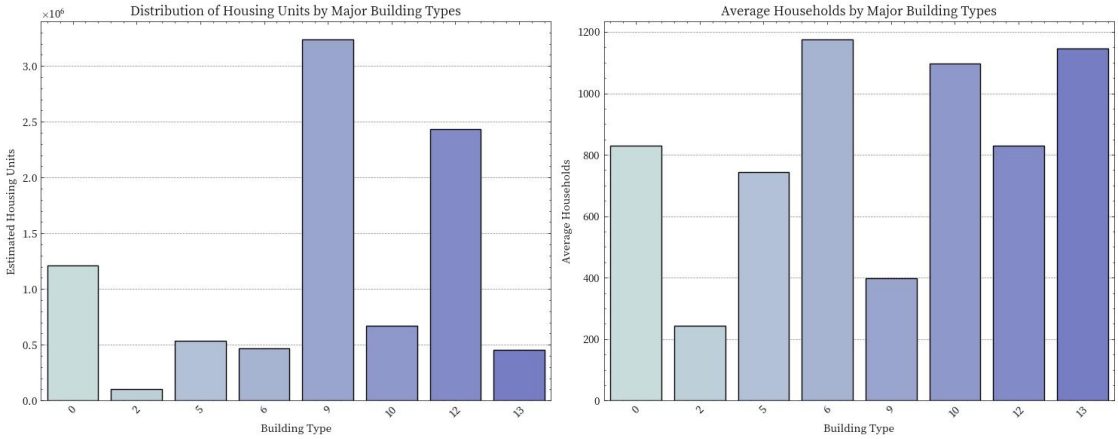

Figure 8 Estimated results for cities 1

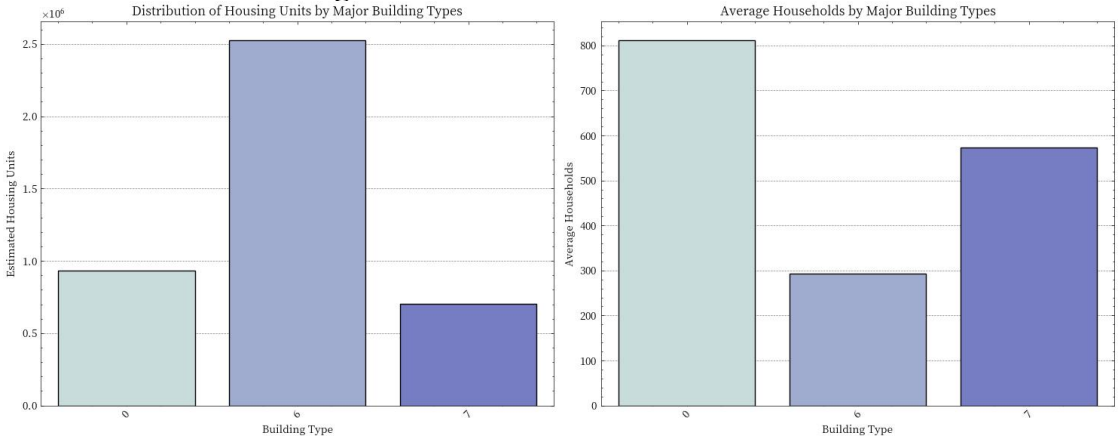

Figure 9 Estimated results for cities 2

Table 5 Estimated results for cities 1

	City1
Estimated total housing	9,527,325
95% confidence interval	(9,316,853, 9,777,059)
Coefficient of variation	0.012

Table 5 Estimated results for cities 2

	City2
Estimated total housing	4,836,592
95% confidence interval	(4,688,372, 4,976,925)
Coefficient of variation	0.015

The estimated total number of housing units in City 1 is 9,527,325, with a 95% confidence interval of 9,316,853 to 9,777,059 and a coefficient of variation of 0.012. In City 2, the estimated total number of housing units is 4,836,592, with a 95% confidence interval of 4,688,372 to 4,976,925 and a coefficient of variation of 0.015. The confidence interval width for City 1 is 460,206, representing 4.83% of the estimated value, while for City 2, it is 288,553, representing 5.97% of the estimated value. The coefficient of variation for City 1 is 1.2%, and for City 2, it is 1.5%. **Overall, the estimation results for both cities demonstrate high precision and stability**. The estimated total number of housing units in City 1 is approximately twice that of City 2, and the estimation results for City 1 are more precise and stable. Although the estimation results for City 2 are slightly less precise than those for City 1, they still exhibit high reliability.

4.2 Question II: Multi-level urban service level evaluation system

For this service level evaluation task, we choose to build **a five-layer and 15-item multi-level evaluation index system.** At the top level, five major indicators, namely basic service capability (weight 1.2), spatial accessibility (weight 1.0), service diversity (weight 0.8), service balance (weight 1.1) and service efficiency (weight 0.9), are set up. Each major indicator has three specific evaluation indicators, forming a complete evaluation framework. On this basis, our evaluation process is divided into three main steps: first, each specific indicator is standardized to eliminate the dimensional impact; Secondly, by means of arithmetic average, all primary indicators are integrated, and the big index score is obtained by combining the hierarchical weights. Finally, **a threshold is set based on the score difference (significant advantage >0.2, slight advantage >0.1) to identify the advantages and disadvantages between cities.** Specifically, we not only analyze the overall strengths and weaknesses of cities in each service type, but also delve into the differential performance at different evaluation levels. Through this multi-level comparison method, we can comprehensively reveal the difference characteristics of the service level of the two cities, and identify both significant areas of advantage and subtle differences. At the same time, we establish a visual analysis system, through the advantage distribution map, hierarchical thermal map, radar map and other visual methods, intuitively show the comparison results of the two cities at different levels,.

4.2.1 Data preprocessing and primary indicators

We first preprocessed the data of different indicators of the two cities, including duplicate values, missing values, outliers, etc. Then we calculated the primary indicators that can evaluate the service level of the two cities:

(1)Urban area

In urban planning and geographic information systems, urban area usually refers to the extent of geographical space occupied by a city. There are many methods to calculate the area of a city, and one of the common methods is to use the Convex Hull Algorithm to calculate the area of a city:

Assuming that the vertex coordinates of the convex hull polygon are $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, the area of the convex hull can be calculated by the following formula:

$$A = \frac{1}{2}\left|\sum_{i=1}^{n-1}(x_i y_{i+1} - x_{i+1} y_i) + (x_n y_1 - x_1 y_n)\right|$$

(2)Density

Density refers to the number of service points per square kilometer. It reflects the density of the distribution of service facilities in the city, and is one of the important indicators to evaluate the level of urban service.

Assuming there is a service point in the city and the city area is square kilometers, the density can be expressed as:

$$D = \frac{N}{A}$$

(3) Diversity

Diversity refers to the richness of the types of services in a city. It reflects the coverage and diversity of the city in different service fields, and is one of the important indicators to evaluate the level of urban service. Diversity is calculated by dividing the number of service types by the total number of service points:

Assuming that there are T service types in the city and the total number of service points is N, the diversity D can be expressed as:

$$D = \frac{T}{N}$$

(4) Fairness

Fairness refers to the balance of service facilities in the spatial distribution of a city. It reflects whether the distribution of service facilities in different areas of the city is fair, and is one of the important indicators to evaluate the level of urban service.

Equity is calculated by calculating the Gini Coefficient, a measure of spatial inequality in the distribution of services. The smaller the Gini coefficient, the more equitable the distribution of services.

$$\text{Gini} = \frac{\sum_{i=1}^{n} (2i - n - 1) \cdot y_i}{n \cdot \sum_{i=1}^{n} y_i}$$

(5) Centrality

Centrality refers to the degree of concentration of service facilities in a city relative to the city center. It reflects whether the distribution of service facilities in the city is concentrated in the central area, and is one of the important indicators to evaluate the level of urban service. Centrality is measured by calculating the average distance of service facilities to the city center.

Assuming that the coordinate of the city center is (C_x, C_y) and the coordinate of the service facility is (x_i, y_i), the centrality C can be expressed as:

$$C = \frac{1}{1 + \frac{1}{N}\sum_{i=1}^{N} \sqrt{(x_i - C_x)^2 + (y_i - C_y)^2}}$$

(6) Dispersion

Dispersion refers to the degree of dispersion of service facilities in the spatial distribution of a city. It reflects whether the distribution of service facilities in the city is uniform, and is one of the important indicators to evaluate the level of urban service. The dispersion is calculated by calculating the standard deviation of the coordinates of the service facilities.

Assuming that the service facility coordinates are (x_i, y_i) and the mean coordinates are (μ_x, μ_y), the dispersion S can be expressed as:

$$S = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left((x_i - \mu_x)^2 + (y_i - \mu_y)^2\right)}$$

(7) Coverage rate

Coverage refers to the proportion of areas covered by services in a city. It reflects the coverage of service facilities in the city and is one of the important indicators to evaluate the service level of the city. Coverage is calculated by dividing the number of grid cells covered by a service facility by the total number of grid cells.

Assuming that the number of covered grid cells is C and the total number of grid cells is T, the coverage R can be expressed as:

$$R = \frac{C}{T}$$

Through the formulas of several primary evaluation indicators mentioned above, we have laid the foundation for the subsequent calculation of secondary evaluation indicators.

4.2.2 Secondary index group

(1)Basic Service Capacity Indicator Group

Basic service capacity is one of the basic dimensions for evaluating the service level of a city, which mainly focuses on the coverage, density, and quantity of service facilities. The following are detailed explanations of the three indicators:

• Service Coverage Rate

Service coverage rate refers to the proportion of the total urban area covered by service facilities.

$$\text{Service Coverage Rate} = \frac{\text{The number of grids with service facilities}}{\text{Total grid number}}$$

• Service Density

Service density refers to the number of service facilities per unit area.

$$\text{Service Density} = \frac{\text{Number of service facilities}}{\text{Total urban area}}$$

• Number of Service Points

The number of service points refers to the total number of facilities of a certain type in a city.

$$\text{Number of service points} = \text{Number of service facilities}$$

The Basic Service Capacity Indicator Set evaluates the basic situation of urban service facilities through three dimensions of service coverage, service density, and service point number. These indicators help us understand the distribution breadth, density, and total number of service facilities, thereby judging the city's performance in basic services. Through the comprehensive analysis of these three indicators, we can fully understand the strengths and weaknesses of a city's basic service capacity and provide a basis for subsequent optimization and improvement.

(2)Spatial accessibility index group

Spatial accessibility is one of the important dimensions to evaluate urban service level, which mainly focuses on the spatial distribution of service facilities and the convenience of residents to obtain services. Here is a detailed explanation of the three metrics:

• Spatial accessibility

Spatial accessibility refers to the average distance between residents and the nearest service facilities.

$$\text{Spatial accessibility} = \frac{\sum_{i=1}^{n} d_i}{n}$$

• Spatial concentration

Spatial concentration is the inverse of the average distance of a service facility relative to the city center.

$$\text{Spatial concentration} = \frac{1}{1 + \frac{\sum_{i=1}^{n} d_i}{n}}$$

• Spatial dispersion

Spatial dispersion is the mean of the standard deviation of a service facility at the X and Y coordinates.

$$\text{Spatial dispersion} = \frac{\sigma_x + \sigma_y}{2}$$

Where, σ_x and σ_y are the standard deviations of the service facilities in X and Y coordinates, respectively.

The spatial accessibility index group comprehensively evaluates the spatial distribution of urban service facilities and the convenience of residents' access to services through three dimensions: spatial accessibility, spatial concentration and spatial dispersion. These indicators can help us understand the spatial distribution characteristics of service facilities, so as to judge the performance of the city in terms of spatial accessibility.

(3) Service diversity indicator group:

Service diversity is one of the important dimensions to evaluate urban service level, which mainly focuses on the type diversity, type total and type distribution of service facilities. Here is a detailed explanation of the three metrics:

• Type diversity

Type diversity refers to the richness of different types of service facilities.

$$\text{Type diversity} = \frac{\text{Number of different types of service facilities}}{\text{Total number of service facilities}}$$

• Type total

The total number of types refers to the total number of all different types of service facilities in a city.

Total number of types = Number of different types of service facilities

• Type balance

Type balance refers to the balance of the number of different types of service facilities.

$$\text{Type equilibrium} = 1 - \text{Gini coefficient}$$

The service diversity index group comprehensively evaluates the diversity and balance of urban service facilities through three dimensions: type diversity, type total and type balance. These indicators can help us understand the type richness and distribution balance of service facilities, so as to judge the performance of a city in terms of service diversity.

(4) Service balance indicator group

Service balance is one of the important dimensions to evaluate urban service level. It mainly focuses on the balance of service facilities in spatial, regional and type distribution. Here is a detailed explanation of the three metrics:

• Spatial equilibrium

Spatial equilibrium refers to the equilibrium degree of service facilities in urban spatial distribution.

$$\text{Spatial equilibrium} = 1 - \text{spatial Gini coefficient}$$

Among them, the spatial Gini coefficient is used to measure the unequal distribution of service facilities in different spatial grids.

• Area coverage equilibrium

Regional coverage balance refers to the coverage balance of service facilities in different regions.

$$\text{Region coverage equilibrium} = 1 - \text{Regional Gini coefficient}$$

• Service distribution balance

Service distribution balance refers to the distribution balance of different types of service facilities in a city.

$$\text{Service distribution balance} = 1 - \text{Type Gini coefficient}$$

The service balance index group comprehensively evaluates the balance degree of urban service facilities in spatial, regional and type distribution through three dimensions: spatial balance, regional coverage balance and service distribution balance. These indicators can help us to understand the distribution balance of service facilities, so as to judge the performance of the city in terms of service balance. Through the comprehensive analysis of these three indicators, we can fully understand the advantages and disadvantages of the city in terms of service balance, and provide a basis for subsequent optimization and improvement.

(5) Service efficiency indicator group

Service efficiency is one of the important dimensions to evaluate urban service level, which mainly focuses on the utilization efficiency, density and aggregation degree of service facilities. Here is a detailed explanation of the three metrics:

• Service efficiency per unit area

Service efficiency per unit area refers to the service capacity of service facilities within a unit area.

$$\text{Service efficiency per unit area} = \frac{\text{Number of service facilities}}{\text{Total urban area} \times \text{Number of service facility types}}$$

• Service intensity

Service intensity refers to the distribution density of service facilities in a unit area.

$$\text{Service intensity} = \frac{\text{Number of service facilities}}{\text{Actual coverage area of service facilities}}$$

• Degree of service aggregation

The degree of service aggregation refers to the degree of aggregation of service facilities in space.

$$\text{Degree of service aggregation} = \frac{1}{1 + \text{Average nearest neighbor service aggregation}}$$

The service efficiency index group comprehensively evaluates the utilization efficiency, density and aggregation degree of urban service facilities through three dimensions of service efficiency per unit area, service intensity and service aggregation. These indicators can help us understand the utilization and spatial distribution characteristics of service facilities, so as to judge the performance of the city in terms of service efficiency.

4.2.3 Detailed Explanation of the Method for Comparing Overall Strengths and Weaknesses of Citie

(1)Significant Advantage Determination

A significant advantage refers to a city having a significantly higher overall weighted score in a particular service type compared to another city, with a significant difference.

• Calculate the overall weighted score for each service type:

$$Score = \sum(Li \times Wi)$$

Where:

Li: Scores for each major level indicator (Basic Service Capability, Spatial Accessibility Service Diversity, Service Equity, Service Efficiency). ∘

Wi: Weights for each level (Basic Service Capability: 1.2, Spatial Accessibility: 1.0, Service Diversity: 0.8, Service Equity: 1.1, Service Efficiency: 0.9).

• Calculate the difference in scores between the two cities:

$$Diff = Score_{City1} - Score_{City2}$$

• Determination Criteria:

Significant Advantage: If |Diff|>0.2, it is determined as a significant advantage.

Statistical Significance Test: Use t-test or other statistical methods to determine if the difference is statistically significant ($p < 0.05$).

• Explanation:

Significant Advantage: Indicates that City1 has a significantly higher overall weighted score in a particular service type compared to City2, with a significant difference.

Statistical Significance Test: Confirms whether the difference is statistically meaningful, ensuring that the advantage is not an accidental result.

(2)Minor Advantage Determination

A minor advantage refers to a city having a slightly higher overall weighted score in a particular service type compared to another city, but with an insignificant difference.

• Calculate the overall weighted score for each service type:

$$Score = \sum(Li \times Wi)$$

Where：

Li: Scores for each major level indicator (Basic Service Capability, Spatial Accessibility Service Diversity, Service Equity, Service Efficiency).

Wf: Weights for each level (Basic Service Capability: 1.2, Spatial Accessibility: 1.0, Service Diversity: 0.8, Service Equity: 1.1, Service Efficiency: 0.9).

• Calculate the difference in scores between the two cities:

$$Diff = Score_{City1} - Score_{City2}$$

• Determination Criteria:

Minor Advantage: If $0.1<|Diff| \leq 0.2$, it is determined as a minor advantage.

Explanation:

Minor Advantage: Indicates that City1 has a slightly higher overall weighted score in a particular service type compared to City2, but with an insignificant difference.

4.2.4 Single-Level Comparison

Definition: Single-level comparison refers to the calculation of differences between two cities for each level (e.g., Basic Service Capability) to determine if there is an advantage in that level.

• Calculate the difference for each level:

$$LayerDiff = Layer_{City1} \times W层级 - Layer_{City2} \times W层级$$

Where:

Layer城市1：The score of City1 in a specific level.

Layer城市2：The score of City2 in a specific level.

$W层级$：The weight of that level. 2. Determination of Level Advantage:

• Level Advantage: If |LayerDiff|>0.1, it is determined that there is an advantage in that $level$.

Explanation:

Level Advantage: Indicates that one city has a significantly higher score in a specific level compared to the other city, with a significant difference.

4.2.5 Results and analysis

We use the constructed multi-level index evaluation system to evaluate the service level of City 1 and City 2, and the results are as follows：

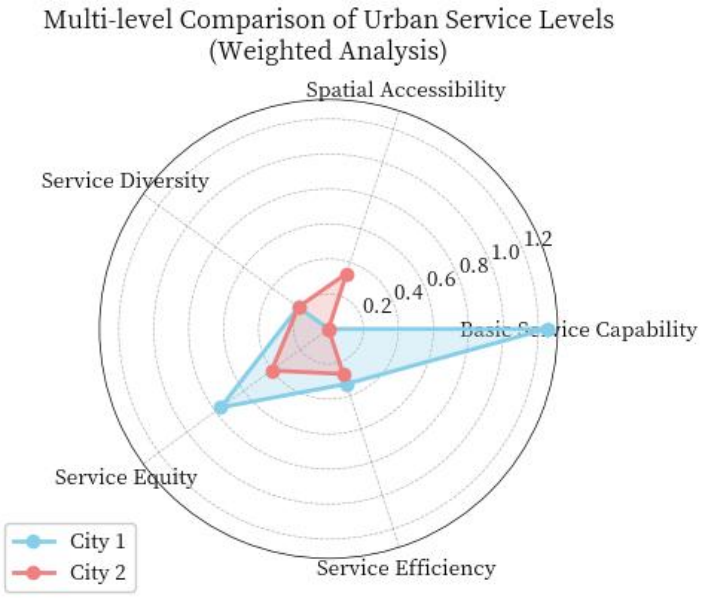

Figure 10 Results of multi-level service level assessment

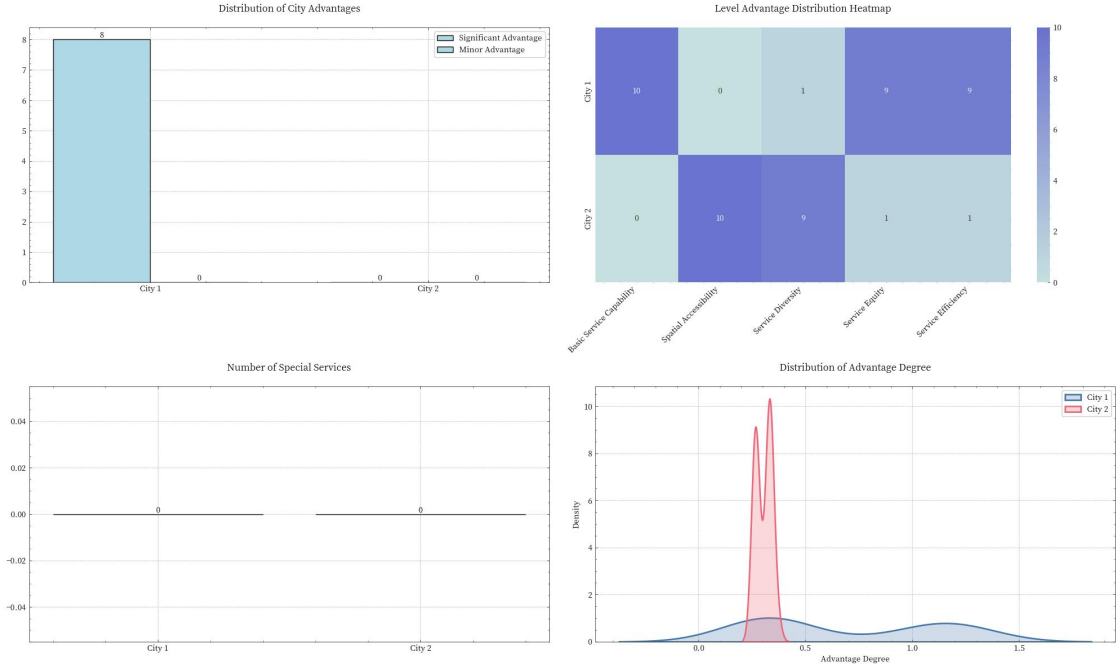

Figure 11 Results of multi-level service level assessment

Table 6 Multi-level index evaluation results

City	Advantage Type	Service Type	Advantage Degree	Statistical Significance
City 1	Significant Advantage	Accommodation service	0.32	TRUE
City 1	Significant Advantage	Medical and health	0.608	TRUE
City 1	Significant Advantage	Business residential	0.252	TRUE
City 1	Significant Advantage	Food and beverage service	0.2533	TRUE
City 1	Significant Advantage	Finance and insurance	0.2533	TRUE
City 1	Significant Advantage	Science, education, and culture	0.4	TRUE
City 1	Significant Advantage	Lifestyle service	0.228	TRUE
City	Significant	Retail service	0.2533	TRUE

1	Advantage

City	Evaluation Level	Service Type	Advantage Degree	Statistical Significance
City 1	Basic Service Capability	Accommodation service	1.2	TRUE
City 1	Basic Service Capability	Medical and health	1.2	TRUE
City 1	Basic Service Capability	Business residential	1.2	TRUE
City 1	Basic Service Capability	Food and beverage service	1.2	TRUE
City 1	Basic Service Capability	Finance and insurance	1.2	TRUE
City 1	Basic Service Capability	Science, education, and culture	1.2	TRUE
City 1	Basic Service Capability	Lifestyle service	1.2	TRUE
City 1	Basic Service Capability	Retail service	1.2	TRUE
City 1	Basic Service Capability	Transportation facilities	0.4	TRUE
City 1	Basic Service Capability	Government and social organizations	0.4	TRUE
City 1	Service Equity	Accommodation service	1.1	TRUE
City 1	Service Equity	Medical and health	1.1	TRUE
City 1	Service Equity	Business residential	1.1	TRUE
City 1	Service Equity	Science, education, and culture	1.1	TRUE
City 1	Service Equity	Food and beverage service	0.3667	TRUE
City 1	Service Equity	Finance and insurance	0.3667	TRUE
City 1	Service Equity	Government and social organizations	0.3667	TRUE
City 1	Service Equity	Lifestyle service	0.3667	TRUE
City 1	Service Equity	Retail service	0.3667	TRUE
City 1	Service Efficiency	Transportation facilities	0.9	TRUE
City 1	Service Efficiency	Accommodation service	0.3	TRUE
City 1	Service Efficiency	Medical and health	0.3	TRUE
City 1	Service Efficiency	Food and beverage service	0.3	TRUE
City 1	Service Efficiency	Finance and insurance	0.3	TRUE
City 1	Service Efficiency	Science, education, and culture	0.3	TRUE
City 1	Service Efficiency	Government and social organizations	0.3	TRUE
City 1	Service Efficiency	Lifestyle service	0.3	TRUE
City 1	Service Efficiency	Retail service	0.3	TRUE
City 1	Service Diversity	Medical and health	0.2667	TRUE

City 2	Spatial Accessibility	Accommodation service	0.3333	TRUE
City 2	Spatial Accessibility	Medical and health	0.3333	TRUE
City 2	Spatial Accessibility	Business residential	0.3333	TRUE
City 2	Spatial Accessibility	Food and beverage service	0.3333	TRUE
City 2	Spatial Accessibility	Finance and insurance	0.3333	TRUE
City 2	Spatial Accessibility	Science, education, and culture	0.3333	TRUE
City 2	Spatial Accessibility	Transportation facilities	0.3333	TRUE
City 2	Spatial Accessibility	Government and social organizations	0.3333	TRUE
City 2	Spatial Accessibility	Lifestyle service	0.3333	TRUE
City 2	Spatial Accessibility	Retail service	0.3333	TRUE
City 2	Service Diversity	Accommodation service	0.2667	TRUE
City 2	Service Diversity	Business residential	0.2667	TRUE
City 2	Service Diversity	Food and beverage service	0.2667	TRUE
City 2	Service Diversity	Finance and insurance	0.2667	TRUE
City 2	Service Diversity	Science, education, and culture	0.2667	TRUE
City 2	Service Diversity	Transportation facilities	0.2667	TRUE
City 2	Service Diversity	Government and social organizations	0.2667	TRUE
City 2	Service Diversity	Lifestyle service	0.2667	TRUE
City 2	Service Diversity	Retail service	0.2667	TRUE
City 2	Service Efficiency	Business residential	0.3	TRUE
City 2	Service Equity	Transportation facilities	0.3667	TRUE

In the detailed analysis of the service level of cities, City 1 shows significant advantages. Specifically, City 1 has significant advantages in several service areas, Including accommodation services (advantage degree 0.3200), medical and health (advantage degree 0.6080), commercial residential (advantage degree 0.2520), catering services (advantage degree 0.2533), financial insurance (advantage degree 0.2533), science, education and culture (advantage degree 0.4000), lifestyle services (advantage degree 0.228) 0) and retail services (advantage 0.2533). These advantages are statistically significant, indicating that City 1 has a higher level of service in these areas. In addition, City 1 performs well in basic service capability, with a score of 1.2000 for all service types, which indicates that City 1 is very complete in the construction of basic service facilities. In terms of service balance, City 1 has a higher score (1.1000) in accommodation services, medical and health care, business housing and science, education and culture, while a lower score (0.3667) in catering services, finance and insurance, government and social organizations, lifestyle services and retail services. In terms of service efficiency, transport facilities scored the highest (0.9000), while all other service types scored 0.3000. In terms of service diversity, health care scored highest (0.2667). In contrast, City 2 did not show any significant

advantages in services, but performed better in spatial accessibility and diversity of services, with a score of 0.3333 across all service types. City 2 has the highest score for commercial residences in terms of service efficiency (0.3000), and the highest score for transportation facilities in terms of service balance (0.3667). The summary of statistical significance shows that all the significant advantage services of City 1 are statistically significant (8/8, 100.0%), which further confirms the dominant position of City 1 in multiple service fields. **On the whole, City 1 shows significant advantages in multiple service areas, especially in medical and health care, accommodation services and science, education and culture, while City 2 is better in spatial accessibility and service diversity, but lacks significant advantages in service.**

4.3 Question III: Assessment of urban resilience and sustainability

For urban resilience and sustainable development, we will conduct research from the following aspects: First, evaluate the health accessibility index, including the number of health facilities per square kilometer (health facility density), the coverage ratio of the area within a 2-kilometer service radius (health service coverage), the total area of the study area, and the total number of health facilities within the study area. Second, assess emergency response capability indicators, including calculating the shortest response time, average response time, median response time, maximum response time, and proportion of areas reachable within 10 minutes from the traffic node to the nearest medical facility. Third, the sustainability capacity indicators were evaluated, including the spatial distribution equilibrium based on grid analysis (the distribution equilibrium measured by calculating the coefficient of variation), the service efficiency based on the service radius and accessibility analysis (calculating the effective service radius of each medical facility), and the service efficiency based on the calculation of the effective service radius of each medical facility. A comprehensive assessment combining facility density and transportation coverage (calculating a comprehensive score of medical facility density and transportation accessibility). The evaluation methods include spatial analysis methods, such as dividing the study area into 10×10 grids, counting the number of medical facilities in each grid, and analyzing the equilibrium of spatial distribution; Calculate the coverage of health services using a 2 km buffer zone analysis; Calculate the distance from each traffic node to the nearest medical facility and convert to response time. Statistical analysis methods include density analysis, coefficient of variation analysis and percentile analysis. The comprehensive evaluation method includes calculating the weight of each index by entropy weight method to ensure the scientificity and objectivity of weight distribution. Standardize the data of each index, eliminate the impact of dimension, and ensure the fairness of comprehensive evaluation; According to the weights determined, the comprehensive index of sustainable development is obtained by weighted comprehensive scores for each index.

4.3.1 Toughness evaluation index

(1)Medical Facility Density

Medical facility density refers to the number of medical facilities per unit area.

$$\text{Density} = \frac{\text{Number of Medical Facilities}}{\text{Area of Study Region}}$$

Where the area A of the study region can be calculated by the range of X and Y coordinates:

$$A = \big(\max(X) - \min(X)\big) \times \big(\max(Y) - \min(Y)\big)$$

(2)Service Coverage Rate

Service coverage rate is the proportion of the area within a given service radius that can be covered by medical facilities.

• Grid Division: Divide the study region into an $N \times N$ grid, with each grid cell having dimensions $\Delta x \times \Delta y$.

• Calculate Coverage Points: For each grid point (x_i, y_j), calculate the distance d_{ij} to the nearest medical facility:

$$d_{ij} = \sqrt{\big(X_f - x_i\big)^2 + \big(Y_f - y_j\big)^2}$$

where (X_f, Y_f) are the coordinates of the medical facility. 。 Determine Coverage: If $d_{ij} \leq R_v$ the point is covered, where R is the service radius (e.g, 2kilometers).

• Calculate Coverage Rate:

$$\text{Coverage Rate} = \frac{\text{Number of Covered Grid Points}}{\text{Total Number of Grid Points}}$$

(3)Emergency Response Time

Emergency response time is the time from a transportation node to the nearest medical facility.

• Calculate Distance: For each transportation node (X_t, Y_t),calculate the distance d_t to the nearest medical facility:

$$d_t = \min_f \sqrt{\big(X_f - X_t\big)^2 + \big(Y_f - Y_t\big)^2}$$

• Convert to Time: Considering the actual road distance is 1.3 times the straight-line distance, convert to response time t_t:

$$t_t = \frac{1.3 \times d_t}{v}$$

where v is the average speed (e.g, 30 km/h).

Calculate Statistical Metrics:

• Average Response Time:

$$\text{Average Response Time} = \frac{1}{N}\sum_{t=1}^{N} t_t$$

• Maximum Response Time

$$\text{Maximum Response Time} = \max(t_t)$$

• Coverage Within 10 Minutes:

$$\text{Coverage Within 10 Minutes} = \frac{\sum_{t=1}^{N} I\,(t_t \leq 10)}{N}$$

where $I(t_t \leq 10)$ is an indicator function that is 1 if $t_t \leq 10$ and 0 otherwise.

4.3.2 Sustainability index

(1)Distribution Balance

Distribution balance refers to the uniformity of medical facility distribution within the study region.

• Grid Division: Divide the study region into an $N \times N$ grid, with each grid cell having dimensions $\Delta x \times \Delta y$.

• Count Facilities in Each Grid: Denote the number of medical facilities in each grid as C_{ij}.

• Calculate Coefficient of Variation:

$$\text{Coefficient of Variation} = \frac{\sigma(C)}{\mu(C)}$$

where $\mu(C)$ is the mean and $\sigma(C)$ is the standard deviation of the number of medical facilities in the grids.

• Normalize:

$$\text{Balance Score} = \frac{1}{1 + \text{Coefficient of Variation}}$$

(2)Service Efficiency

Service efficiency refers to the service range and effectiveness of medical facilities.

• Calculate Effective Service Radius

$$R_f = 75\text{th Percentile}(d_{tf})$$

where d_tf is the distance from transportation nodes to medical facility f. 。

• Calculate Average Service Radius:

$$\text{Average Service Radius} = \frac{1}{F}\sum_{f=1}^{F} R_f$$

where F is the number of medical facilities.

• Normalize:

$$\text{Efficiency Score} = \frac{1}{1 + \frac{\text{Average Service Radhus}}{10}}$$

(3)Development Potential

Development potential is a composite score of medical facility density and transportation accessibility.

• Calculate Medical Facility Density:

$$\text{Medical Facility Density} = \frac{\text{Number of Medical Facilities}}{\text{Area of Studv Region}}$$

• Calculate Transportation Accessibility:

$$\text{Transportation Accessibility} = \frac{\text{Number of Transportation Nodes}}{\text{Area of Study Rerion}}$$

• Composite Score:

$$\text{Development Potential Score} = \frac{\text{Medical Facility Density}}{2} + \frac{\text{Transportation Accessibility}}{3}$$

(4)Comprehensive Sustainability Index

The comprehensive sustainability index is a composite indicator that integrates distribution balance, service efficiency, and development potential.：

$$\text{Comprehensive Sustainability Index} = \sum_{j=1}^{m} w_j \cdot \text{Normalized Matrix}_{1j}$$

Among them, the weight is calculated by entropy weight method.

We use the above indicator system to evaluate the resilience and sustainability of cities, and the results are as follows:

Table 7 Urban resilience and sustainability assessment results

City	Total Medical Facilities	Study Area (sq km)	Medical Facility Density (per sq km)	Medical Service Coverage (%)	Shortest Response Time (min)	Average Response Time (min)	Median Response Time (min)
City 1	16647	15105.99	1.1	32.52	0	1.21	0.58
City 2	6024	8557.46	0.7	24.11	0.02	4.39	3.33

City	Longest Response Time (min)	Coverage Within 10 Minutes (%)	Distribution Balance	Service Efficiency	Development Potential	Longest Response Time (min)	Coverage Within 10 Minutes (%)
City 1	12.46	99.92	0.1674	0.2902	1.031	12.46	99.92
City 2	37.8	92.71	0.1616	0.3727	0.618	37.8	92.71

City	Comprehensive Sustainability Index	Distribution Balance Weight	Service Efficiency Weight	Development Potential Weight
City 1	0.4107	0.3107	0.3248	0.3645
City 2	0.2779	0.3093	0.3352	0.3555

4.3.2 Result analysis

On the whole, the comprehensive performance of City 1 is better than that of city 2. Specifically, in terms of the allocation of medical facilities, City 1 has 16,647 medical facilities with a density of 1.10 / km ², which is significantly higher than the 6,024 facilities and 0.70 / km ² density of City 2. In terms of medical service coverage, city 1 reached 32.52%, while city 2 was 24.11%, indicating that the spatial distribution of medical resources in city 1 was more reasonable.

In terms of emergency response ability, City 1 showed obvious advantages. The average response time of City 1 was only 1.21 minutes with a 10-minute coverage rate of 99.92%, while the average response time of City 2 was 4.39 minutes with a 10-minute coverage rate of 92.71%. This means that City 1's emergency medical services are more efficient and can respond more quickly to emergency medical needs.

From the specific indicators of sustainable development ability, the weight distribution obtained by entropy weight method is relatively balanced, and the weights

of the three indicators of the two cities are between 0.31 and 0.37, indicating that these indicators have similar importance to sustainable development. The development potential index gained the highest weight in both cities (0.3645 for City 1 and 0.3555 for City 2). City 1 scored highest in terms of development potential (1.0310), while City 2 had relatively good service efficiency (0.3727). In the final comprehensive index of sustainable development, the score of City 1 is 0.4107, which is higher than that of city 2 (0.277), indicating that city 1 has stronger sustainable development ability.

In general, City 1 outperforms City 2 in terms of the number, density, coverage, emergency response time and sustainable development potential of medical facilities, showing greater urban resilience and development potential. **However, both cities still have room for improvement, especially in terms of the balanced distribution of medical resources, the scores of both cities are relatively low (0.1674 and 0.1616 respectively), which indicates that the spatial distribution of medical resources needs to be further optimized.**

4.4 Question IV:Investment scheme design

Based on the analysis results of our first three questions, we propose the following investment plan：

For City 1, we will focus on deepening the construction of smart cities, investing 50 million yuan to improve the digital level, especially in the fields of medical and health care, education and culture, and commercial housing, and promote intelligent application and service innovation; Optimize the distribution of medical resources, invest 30 million yuan to build more community health service centers and specialized hospitals, and improve the rationality of spatial distribution of medical resources; Improve the emergency response capacity, invest 20 million yuan to optimize the emergency medical service system, shorten the response time, and improve the coverage within 10 minutes; Strengthen sustainable development capacity, invest 40 million yuan to strengthen environmental protection and resource conservation, promote the use of green buildings and renewable energy, and improve the sustainable development capacity of the city.

For City 2, we will focus on improving service advantages, invest 40 million yuan to introduce more high-quality service enterprises and projects, and enhance the competitiveness of accommodation services, medical and health care, commercial housing, catering services, finance and insurance, science, education and culture, life services and retail services. Optimizing spatial accessibility, investing $30 million to improve transport infrastructure, especially public transport systems, to improve the convenience and efficiency of citizens' travel; Increase the diversity of services, invest 20 million yuan to enrich the types of services to meet the needs of different groups, especially in education, culture and entertainment, and provide more diversified choices; Strengthen the construction of medical resources, invest 30 million yuan to increase the number and density of medical facilities, improve the quality of medical services, especially the coverage of medical services in remote areas, and ensure that citizens have convenient access to medical services.

Through the implementation of the above investment plan, City 1 will make significant progress in smart city construction, medical resource distribution, emergency response capability and sustainable development capability, and become a leading green city in China. On the other hand, City 2 will achieve significant improvement in service advantages, spatial accessibility, service diversity and medical resource construction, so as to better meet the needs of citizens and achieve more balanced and sustainable development

5 Strengths and Weakness

5.1 Strengths

(1)Multi-dimensional comprehensive evaluation:

The model makes a comprehensive assessment from multiple perspectives, such as housing price prediction, urban service level evaluation, urban resilience and sustainable development, ensuring the comprehensiveness and depth of the assessment.Each subtask has clear objectives and methods, forming a systematic evaluation framework.

(2)Application of multiple models and techniques:

In the housing price prediction, XGBoost, LightGBM, CatBoost and other machine learning models are used, and the integrated learning method is used to improve the accuracy and stability of the prediction.

In the evaluation of urban service level, a multi-level evaluation index system is constructed and a variety of statistical and spatial analysis methods are combined to ensure the scientificity and accuracy of the evaluation.

In the assessment of urban resilience and sustainable development, various techniques such as spatial analysis, buffer analysis, statistical analysis and entropy weight method are used to ensure the comprehensiveness and objectivity of the assessment.

(3)Scientific data processing and analysis methods:

The data is pre-processed, including standardization, removal of duplicate values, processing of missing values and outliers, etc., to ensure the quality of the data. A variety of statistical and spatial analysis methods are used, such as density analysis, coefficient of variation analysis, buffer analysis, etc., to ensure the scientific and accurate data processing.

(4)Scientific weight allocation:

In the evaluation of urban service level and urban resilience and sustainable development, entropy weight method is used to calculate the weight of each index, which ensures the scientificity and objectivity of weight distribution and avoids the interference of subjective factors

.

(5)The threshold is set clearly:

In the urban service level evaluation, a clear threshold is set according to the score differences, which is used to identify the strengths and weaknesses between cities, so as to make the evaluation results more clear and specific.

5.2 Weakness

(1)Static analysis:

The current evaluation system is mainly based on static data analysis and fails to fully consider the dynamic changes of urban development. For example, the impact of factors such as population mobility and economic development on urban service levels and resilience may not be fully reflected.

(2)Ignoring socioeconomic factors:

The evaluation system mainly focuses on the assessment of the efficiency of physical facilities and services, and takes little account of the impact of socio-

economic factors (such as residents' income level and education level) on urban service level and resilience.

(3)Local optimal solution:

Although multi-dimensional evaluation is used, the interaction and influence between the various dimensions may not be fully considered, which may lead to the evaluation results being biased towards the local optimal solution rather than the global optimal solution.

References

[1] Pickett, S. T. A., Cadenasso, M. L., Grove, J. M., Giancarlo, C. P., & Ogden, L. A. (2011). Urban resilience: A transformative approach.Cities and the Environment (CATE), 4(1), 1.

[2] Cadman, D. (2004).Sustainable urban development: The environmental assessment methods. Routledge.

[3] Fernandez, L., Marcouiller, D. J., & Lewis, D. J. (2017).Urban resilience: Theory, policy, and practice.

[4] Rodin, J. (2014).The resilience dividend: Being strong in a world where things go wrong. PublicAffairs.

[5] Kumar, T. M. V. (2017).Sustainable cities: Concepts, strategies, and technologies.

Appendix

No: 1	The first question

```
import os
import pandas as pd
import numpy as np
from scipy import interpolate
from sklearn.neighbors import BallTree
import warnings
warnings.filterwarnings("ignore")

def handle_outliers(df, columns):
    """
    使用四分位数法处理异常值
    """
    df_clean = df.copy()
    for col in columns:
        Q1 = df_clean[col].quantile(0.25)
        Q3 = df_clean[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # 记录异常值数量
        outliers = df_clean[(df_clean[col] < lower_bound) | (df_clean[col] > upper_bound)][col]
        if len(outliers) > 0:
            print(f"\n{col} 发现 {len(outliers)} 个异常值")
            print(f"下界: {lower_bound:.2f}, 上界: {upper_bound:.2f}")

        # 将异常值设置为边界值
        df_clean.loc[df_clean[col] < lower_bound, col] = lower_bound
        df_clean.loc[df_clean[col] > upper_bound, col] = upper_bound

    return df_clean

def spline_interpolation(df, columns):
    """
    使用三次样条插值处理缺失值
    """
    df_interpolated = df.copy()

    for col in columns:
        # 获取非缺失值的索引和值
        non_null_mask = df_interpolated[col].notnull()
        x = np.where(non_null_mask)[0]
        y = df_interpolated[col][non_null_mask].values

        if len(x) > 3:  # 确保有足够的点进行插值
            # 创建插值函数
            f = interpolate.interp1d(x, y, kind='cubic', fill_value='extrapolate')

            # 获取缺失值的索引
            null_indices = np.where(~non_null_mask)[0]

            # 进行插值
            if len(null_indices) > 0:
                df_interpolated.loc[null_indices, col] = f(null_indices)
                print(f"\n{col} 插值处理了 {len(null_indices)} 个缺失值")

    return df_interpolated
```

```
def process_housing_data(df):
    """
    处理房价数据
    """
    print("\n开始处理房价数据...")
    df_cleaned = df.copy()

    # 必需列（不能有缺失值的列）
    required_columns = ['X', 'Y', 'Price (USD)']
    df_cleaned = df_cleaned.dropna(subset=required_columns)
    print(f"删除关键字段缺失后剩余数据：{len(df_cleaned)}条")

    # 创建每平方米价格字段
    df_cleaned['price_per_sqm'] = df_cleaned['Price (USD)']

    # 可以插值的数值列
    interpolate_columns = [
        'Greening rate',
        'Floor area ratio',
        'above-ground parking fee（/month USD）',
        'underground parking fee（/month USD）',
        'Property management fee（/m²/month USD）'
    ]

    # 对可插值的列进行处理
    for col in interpolate_columns:
        if col in df_cleaned.columns:
            # 转换为数值类型
            df_cleaned[col] = pd.to_numeric(df_cleaned[col], errors='coerce')
            # 进行插值
            null_count = df_cleaned[col].isnull().sum()
            if null_count > 0:
                df_cleaned[col] = df_cleaned[col].interpolate(method='cubic', limit_direction='both')
                print(f"{col} 插值处理了 {null_count} 个缺失值")

    # 处理异常值
    numeric_columns = ['Price (USD)', 'price_per_sqm'] + interpolate_columns
    for col in numeric_columns:
        if col in df_cleaned.columns:
            Q1 = df_cleaned[col].quantile(0.25)
            Q3 = df_cleaned[col].quantile(0.75)
            IQR = Q3 - Q1
            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR

            outliers = df_cleaned[(df_cleaned[col] < lower_bound) | (df_cleaned[col] >
upper_bound)][col]
            if len(outliers) > 0:
                print(f"\n{col} 发现 {len(outliers)} 个异常值")
                print(f"下界: {lower_bound:.2f}, 上界: {upper_bound:.2f}")

            df_cleaned.loc[df_cleaned[col] < lower_bound, col] = lower_bound
            df_cleaned.loc[df_cleaned[col] > upper_bound, col] = upper_bound

    print("\n房价数据处理完成")
    return df_cleaned

def calculate_poi_features(housing_df, poi_df):
    """
    计算每个房屋位置周边的POI特征
    """
```

```
    print("\n计算POI特征...")

    # 1. 数据检查
    print("\nPOI数据基本信息：")
    print(f"POI总数: {len(poi_df)}")
    print("\nPOI类型分布:")
    print(poi_df['service_category'].value_counts())

    # 2. 确保坐标数据完整性并标准化坐标
    housing_coords = housing_df[['X', 'Y']].values
    poi_coords = poi_df[['X', 'Y']].dropna(subset=['X', 'Y'])[['X', 'Y']].values

    # 计算坐标范围
    x_min, y_min = poi_coords.min(axis=0)
    x_max, y_max = poi_coords.max(axis=0)

    print("\n原始坐标范围:")
    print(f"X范围: {x_min:.2f} - {x_max:.2f}")
    print(f"Y范围: {y_min:.2f} - {y_max:.2f}")

    # 3. 设置合理的搜索半径（基于实际距离）
    # 假设坐标单位为千米，设置三个搜索半径：500m, 1000m, 2000m
    radii = [0.5, 1.0, 2.0] # 单位：千米

    print(f"\n使用的搜索半径: {[f'{int(r*1000)}m' for r in radii]}")

    features = {}
    # 构建POI总体密度特征
    tree = BallTree(poi_coords)

    # 计算总体POI密度
    for radius in radii:
        counts = tree.query_radius(housing_coords, r=radius, count_only=True)
        features[f'poi_count_{int(radius*1000)}m'] = counts
        print(f"\n{int(radius*1000)}m 半径内的平均POI数量: {counts.mean():.2f}")

    # 计算各类服务设施密度
    for service_type in poi_df['service_category'].unique():
        service_points = poi_df[poi_df['service_category'] == service_type].dropna(subset=['X', 'Y'])
        if len(service_points) > 0:
            service_coords = service_points[['X', 'Y']].values
            service_tree = BallTree(service_coords)

            for radius in radii:
                service_counts = service_tree.query_radius(housing_coords, r=radius,
count_only=True)
                features[f'{service_type}_count_{int(radius*1000)}m'] = service_counts
                print(f"{service_type} 在 {int(radius*1000)}m 半径内的平均数量:
{service_counts.mean():.2f}")

    features_df = pd.DataFrame(features, index=housing_df.index)

    # 特征统计信息
    print("\n特征统计摘要:")
    print(features_df.describe())

    return features_df

def read_csv_with_encoding(file_path):
    """
    尝试使用不同的编码方式读取CSV文件
    """
```

```
    encodings = ['gbk', 'utf-8', 'utf-8-sig', 'gb18030', 'latin1', 'ISO-8859-1']

    for encoding in encodings:
        try:
            return pd.read_csv(file_path, encoding=encoding)
        except UnicodeDecodeError:
            continue
        except Exception as e:
            print(f"使用 {encoding} 编码读取时出现错误: {str(e)}")
            continue

    raise ValueError(f"无法使用任何编码方式读取文件: {file_path}")

def process_city_data(city_num, appendix_folder, housing_file):
    """
    处理单个城市的数据
    """
    print(f"\n{'='*50}")
    print(f"处理城市{city_num}的数据...")
    print(f"{'='*50}")

    # 读取当前城市的POI数据
    data_folder = os.path.join(os.path.expanduser("~"), "Desktop", appendix_folder)

    # 为每个城市定义服务数据
    if city_num == 1:
        # 城市1的数据 (Appendix 3)
        service_dfs = {
            '住宿服务': read_csv_with_encoding(os.path.join(data_folder, "Accommodation service
data.csv")),
            '商住服务': read_csv_with_encoding(os.path.join(data_folder, "Business-residential
data.csv")),
            '餐饮服务': read_csv_with_encoding(os.path.join(data_folder, "Food and beverage service
data.csv")),
            '金融保险': read_csv_with_encoding(os.path.join(data_folder, "Finance and insurance
data.csv")),
            '政府机构': read_csv_with_encoding(os.path.join(data_folder, "Government and social
organizations data.csv")),
            '医疗健康': read_csv_with_encoding(os.path.join(data_folder, "Medical and health
data.csv")),
            '生活服务': read_csv_with_encoding(os.path.join(data_folder, "Lifestyle service
data.csv")),
            '零售服务': read_csv_with_encoding(os.path.join(data_folder, "Retail service data.csv")),
            '科教文化': read_csv_with_encoding(os.path.join(data_folder, "Science, education, and
culture data.csv")),
            '交通设施': read_csv_with_encoding(os.path.join(data_folder, "Transportation facilities
data.csv"))
        }
    else:  # city_num == 2
        # 城市2的数据 (Appendix 4)
        service_dfs = {
            '住宿服务': read_csv_with_encoding(os.path.join(data_folder, "Accommodation service
data.csv")),
            '商住服务': read_csv_with_encoding(os.path.join(data_folder, "Business-residential
data.csv")),
            '餐饮服务': read_csv_with_encoding(os.path.join(data_folder, "Food and beverage service
data.csv")),
            '金融保险': read_csv_with_encoding(os.path.join(data_folder, "Finance and insurance
data.csv")),
            '政府机构': read_csv_with_encoding(os.path.join(data_folder, "Government and social
organizations data.csv")),
```

```
        '医疗健康': read_csv_with_encoding(os.path.join(data_folder, "Medical and health
data.csv")),
        '生活服务': read_csv_with_encoding(os.path.join(data_folder, "Lifestyle service
data.csv")),
        '零售服务': read_csv_with_encoding(os.path.join(data_folder, "Retail service data.csv")),
        '科教文化': read_csv_with_encoding(os.path.join(data_folder, "Science, education, and
culture data.csv")),
        '交通设施': read_csv_with_encoding(os.path.join(data_folder, "Transportation facilities
data.csv"))
    }

    processed_dfs = {}
    # 检查每个服务类型的数据
    for service_name, df in service_dfs.items():
        print(f"\n处理 {service_name} 数据:")
        print(f"原始数据量: {len(df)}")
        df = standardize_column_names(df)
        processed_df = preprocess_service_data(df, service_name)
        print(f"处理后数据量: {len(processed_df)}")
        processed_dfs[service_name] = processed_df

    # 合并当前城市的POI数据
    city_poi = pd.concat(processed_dfs.values(), ignore_index=True)
    print(f"\n城市{city_num}的POI总数量: {len(city_poi)}")

    # 读取并处理房价数据
    housing_df = pd.read_excel(f"C:/Users/30766/Desktop/{housing_file}")
    housing_processed = process_housing_data(housing_df)

    # 整合数据
    integrated_data = integrate_housing_and_poi(housing_processed, city_poi)

    return city_poi, integrated_data
def compare_cities_visualization(city1_data, city2_data):
    """
    Comparative visualization analysis of two cities
    """
    import matplotlib.pyplot as plt
    import seaborn as sns

    # 创建图形
    plt.figure(figsize=(20, 15))

    # 1. POI Distribution Comparison
    plt.subplot(2, 2, 1)
    plt.scatter(city1_data['poi']['X'], city1_data['poi']['Y'],
                c=city1_data['poi']['service_category'].astype('category').cat.codes,
                alpha=0.5, s=1)
    plt.title('City 1 POI Distribution')
    plt.xlabel('X Coordinate')
    plt.ylabel('Y Coordinate')

    plt.subplot(2, 2, 2)
    plt.scatter(city2_data['poi']['X'], city2_data['poi']['Y'],
                c=city2_data['poi']['service_category'].astype('category').cat.codes,
                alpha=0.5, s=1)
    plt.title('City 2 POI Distribution')
    plt.xlabel('X Coordinate')
    plt.ylabel('Y Coordinate')

    # 2. Housing Price Distribution
    plt.subplot(2, 2, 3)
    sc1 = plt.scatter(city1_data['integrated']['X'],
```

```
            city1_data['integrated']['Y'],
            c=city1_data['integrated']['Price (USD)'],
            cmap='viridis', alpha=0.6)
plt.colorbar(sc1, label='Price (USD)')
plt.title('City 1 Housing Price Distribution')
plt.xlabel('X Coordinate')
plt.ylabel('Y Coordinate')

plt.subplot(2, 2, 4)
sc2 = plt.scatter(city2_data['integrated']['X'],
            city2_data['integrated']['Y'],
            c=city2_data['integrated']['Price (USD)'],
            cmap='viridis', alpha=0.6)
plt.colorbar(sc2, label='Price (USD)')
plt.title('City 2 Housing Price Distribution')
plt.xlabel('X Coordinate')
plt.ylabel('Y Coordinate')

plt.tight_layout()
plt.show()

# 3. Housing Price Statistical Comparison
plt.figure(figsize=(15, 5))

# Box Plot
plt.subplot(1, 2, 1)
data_to_plot = [
   city1_data['integrated']['Price (USD)'],
   city2_data['integrated']['Price (USD)']
]
plt.boxplot(data_to_plot, labels=['City 1', 'City 2'])
plt.title('Housing Price Distribution Comparison')
plt.ylabel('Price (USD)')

# Violin Plot
plt.subplot(1, 2, 2)
data_combined = pd.DataFrame({
   'City 1': city1_data['integrated']['Price (USD)'],
   'City 2': city2_data['integrated']['Price (USD)']
})
sns.violinplot(data=data_combined)
plt.title('Housing Price Density Distribution')
plt.ylabel('Price (USD)')

plt.tight_layout()
plt.show()

# 4. POI Type Distribution Comparison
plt.figure(figsize=(20, 8))

# 定义更好的颜色方案
colors = plt.cm.Set3(np.linspace(0, 1, 10))  # 使用Set3颜色映射

# City 1 POI Type Distribution
plt.subplot(1, 2, 1)
city1_poi_counts = city1_data['poi']['service_category'].value_counts()

# 调整标签位置避免重叠
patches, texts, autotexts = plt.pie(city1_poi_counts.values,
                                    labels=city1_poi_counts.index,
                                    autopct='%1.1f%%',
                                    colors=colors,
                                    wedgeprops={'edgecolor': 'white', 'linewidth': 1},
                                    pctdistance=0.85,
```

```
                                labeldistance=1.1)

    # 调整文本大小和样式
    plt.setp(autotexts, size=8, weight="bold")
    plt.setp(texts, size=8)

    # 添加圆环效果
    centre_circle = plt.Circle((0,0), 0.70, fc='white')
    plt.gca().add_artist(centre_circle)

    plt.title('City 1 POI Type Distribution', pad=20)

    # City 2 POI Type Distribution
    plt.subplot(1, 2, 2)
    city2_poi_counts = city2_data['poi']['service_category'].value_counts()

    patches, texts, autotexts = plt.pie(city2_poi_counts.values,
                                labels=city2_poi_counts.index,
                                autopct='%1.1f%%',
                                colors=colors,
                                wedgeprops={'edgecolor': 'white', 'linewidth': 1},
                                pctdistance=0.85,
                                labeldistance=1.1)

    plt.setp(autotexts, size=8, weight="bold")
    plt.setp(texts, size=8)

    centre_circle = plt.Circle((0,0), 0.70, fc='white')
    plt.gca().add_artist(centre_circle)

    plt.title('City 2 POI Type Distribution', pad=20)

    plt.tight_layout(pad=3.0)
    plt.show()

def main():
    try:
        # 处理城市1数据
        print("\n开始处理城市1数据...")
        city1_poi, city1_integrated = process_city_data(1, "Appendix 3", "Appendix 1.xlsx")

        # 处理城市2数据
        print("\n开始处理城市2数据...")
        city2_poi, city2_integrated = process_city_data(2, "Appendix 4", "Appendix 2.xlsx")

        # 对比两个城市的POI分布
        print("\n两个城市POI对比:")
        print("\n城市1 POI分布:")
        print(city1_poi['service_category'].value_counts())
        print("\n城市2 POI分布:")
        print(city2_poi['service_category'].value_counts())

        # 检查POI数据是否不同
        print("\nPOI数据对比:")
        print(f"城市1 POI总数: {len(city1_poi)}")
        print(f"城市2 POI总数: {len(city2_poi)}")
        print("\n城市1 坐标范围:")
        print(f"X: {city1_poi['X'].min():.2f} - {city1_poi['X'].max():.2f}")
        print(f"Y: {city1_poi['Y'].min():.2f} - {city1_poi['Y'].max():.2f}")
        print("\n城市2 坐标范围:")
        print(f"X: {city2_poi['X'].min():.2f} - {city2_poi['X'].max():.2f}")
        print(f"Y: {city2_poi['Y'].min():.2f} - {city2_poi['Y'].max():.2f}")
```

```
        # 整理数据
        city1_data = {'poi': city1_poi, 'integrated': city1_integrated}
        city2_data = {'poi': city2_poi, 'integrated': city2_integrated}

        # 对比可视化
        print("\n开始生成可视化...")
        compare_cities_visualization(city1_data, city2_data)
        # 导出数据到桌面
        desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")

        # 导出城市1数据
        city1_poi.to_csv(os.path.join(desktop_path, "city1_poi_data.csv"), index=False,
encoding='utf-8-sig')
        city1_integrated.to_csv(os.path.join(desktop_path, "city1_integrated_data.csv"),
index=False, encoding='utf-8-sig')
        print("\n城市1数据已导出到桌面：")
        print("- city1_poi_data.csv")
        print("- city1_integrated_data.csv")

        # 导出城市2数据
        city2_poi.to_csv(os.path.join(desktop_path, "city2_poi_data.csv"), index=False,
encoding='utf-8-sig')
        city2_integrated.to_csv(os.path.join(desktop_path, "city2_integrated_data.csv"),
index=False, encoding='utf-8-sig')
        print("\n城市2数据已导出到桌面：")
        print("- city2_poi_data.csv")
        print("- city2_integrated_data.csv")

        return city1_data, city2_data

    except Exception as e:
        print("处理过程中出现错误：", str(e))
        raise
if __name__ == "__main__":
    # 运行主程序
    main()
```

No: 2	The Second question

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import gaussian_kde, ttest_ind
from scipy.spatial import ConvexHull
from scipy.spatial.distance import cdist
from scipy.cluster.hierarchy import dendrogram, linkage
import gc
from tqdm import tqdm

def load_city_data(appendix_number):
    """加载城市数据"""
    data_folder = os.path.join(os.path.expanduser("~"), "Desktop", f"Appendix
{appendix_number}")

    file_names = [
        'Accommodation service data.csv',
        'Business-residential data.csv',
        'Finance and insurance data.csv',
        'Food and beverage service data.csv',
        'Government and social organizations data.csv',
        'Lifestyle service data.csv',
        'Medical and health data.csv',
        'Retail service data.csv',
        'Science, education, and culture data.csv',
        'Transportation facilities data.csv'
    ]

    city_dfs = {}
    for file_name in tqdm(file_names, desc="加载数据文件"):
        file_path = os.path.join(data_folder, file_name)
        service_name = file_name.replace(' data.csv', '').replace('-', ' ')
        try:
            df = pd.read_csv(file_path, encoding='gbk')
        except UnicodeDecodeError:
            df = pd.read_csv(file_path, encoding='utf-8')
        city_dfs[service_name] = df

    return city_dfs

def calculate_city_area(dfs):
    """计算城市面积"""
    all_points = pd.concat([df[['X', 'Y']] for df in dfs.values()])
    hull = ConvexHull(all_points)
    return hull.area / 1000000

def calculate_gini(array):
    """计算基尼系数"""
    array = np.array(array)
    if np.any(array < 0):
        array = array - array.min()
    if np.all(array == 0):
        return 0
    array = np.sort(array)
    index = np.arange(1, array.shape[0] + 1)
    n = array.shape[0]
    return ((np.sum((2 * index - n - 1) * array)) / (n * np.sum(array)))

def calculate_spatial_gini(df, grid_size=20):
    """计算空间分布的基尼系数"""
    x_bins = np.linspace(df['X'].min(), df['X'].max(), grid_size)
    y_bins = np.linspace(df['Y'].min(), df['Y'].max(), grid_size)
```

```
    counts = []
    for i in range(len(x_bins)-1):
        for j in range(len(y_bins)-1):
            mask = (df['X'] >= x_bins[i]) & (df['X'] < x_bins[i+1]) & \
                   (df['Y'] >= y_bins[j]) & (df['Y'] < y_bins[j+1])
            counts.append(mask.sum())

    return calculate_gini(counts)

def calculate_accessibility(df, sample_size=100):
    """计算服务可达性"""
    if len(df) > sample_size:
        df_sample = df.sample(n=sample_size, random_state=42)
    else:
        df_sample = df

    points = df_sample[['X', 'Y']].values

    x_grid = np.linspace(df['X'].min(), df['X'].max(), 20)
    y_grid = np.linspace(df['Y'].min(), df['Y'].max(), 20)

    distances = []
    for x in x_grid:
        for y in y_grid:
            grid_point = np.array([[x, y]])
            min_dist = cdist(grid_point, points).min()
            distances.append(min_dist)

    return np.mean(distances)

def calculate_centrality(df):
    """计算服务设施的中心性"""
    center_x, center_y = df['X'].mean(), df['Y'].mean()
    distances = np.sqrt((df['X'] - center_x)**2 + (df['Y'] - center_y)**2)
    return 1 / (1 + distances.mean())

def calculate_dispersion(df):
    """计算服务设施的分散度"""
    return df[['X', 'Y']].std().mean()

def calculate_simplified_coverage(df, total_area, grid_size=20):
    """计算简化版的覆盖率"""
    x_bins = np.linspace(df['X'].min(), df['X'].max(), grid_size)
    y_bins = np.linspace(df['Y'].min(), df['Y'].max(), grid_size)

    covered_cells = 0
    total_cells = (grid_size - 1) * (grid_size - 1)

    for i in range(len(x_bins)-1):
        for j in range(len(y_bins)-1):
            mask = (df['X'] >= x_bins[i]) & (df['X'] < x_bins[i+1]) & \
                   (df['Y'] >= y_bins[j]) & (df['Y'] < y_bins[j+1])
            if mask.any():
                covered_cells += 1

    return covered_cells / total_cells

def calculate_regional_coverage_balance(df):
    """计算区域覆盖均衡度"""
    grid_size = 20
    x_bins = np.linspace(df['X'].min(), df['X'].max(), grid_size)
    y_bins = np.linspace(df['Y'].min(), df['Y'].max(), grid_size)
```

```
    grid_counts = []
    for i in range(len(x_bins)-1):
        for j in range(len(y_bins)-1):
            mask = (df['X'] >= x_bins[i]) & (df['X'] < x_bins[i+1]) & \
                   (df['Y'] >= y_bins[j]) & (df['Y'] < y_bins[j+1])
            grid_counts.append(mask.sum())

    return 1 - calculate_gini(grid_counts)

def calculate_service_distribution_balance(df):
    """计算服务分布均衡度"""
    if len(df) > 100:
        df_sample = df.sample(n=100, random_state=42)
    else:
        df_sample = df

    points = df_sample[['X', 'Y']].values
    distances = cdist(points, points)
    np.fill_diagonal(distances, np.inf)
    min_distances = distances.min(axis=1)

    return 1 - calculate_gini(min_distances)

def calculate_service_area(df):
    """计算服务实际覆盖面积"""
    try:
        hull = ConvexHull(df[['X', 'Y']])
        return hull.area
    except:
        return 1

def calculate_service_clustering(df):
    """计算服务聚集程度"""
    if len(df) > 100:
        df_sample = df.sample(n=100, random_state=42)
    else:
        df_sample = df

    points = df_sample[['X', 'Y']].values
    distances = cdist(points, points)
    np.fill_diagonal(distances, np.inf)
    min_distances = distances.min(axis=1)

    return 1 / (1 + min_distances.mean())

def analyze_service_level(df, total_area):
    """多层次分析单个服务行业的服务水平"""
    try:
        # 1. 基础服务能力
        basic_metrics = {
            '服务覆盖率': calculate_simplified_coverage(df, total_area),
            '服务密度': len(df) / total_area,
            '服务点数量': len(df)
        }

        # 2. 空间可及性
        spatial_metrics = {
            '空间可达性': 1 / (1 + calculate_accessibility(df)),
            '空间集中度': calculate_centrality(df),
            '空间分散度': calculate_dispersion(df)
        }

        # 3. 服务多样性
```

```
        diversity_metrics = {
            '类型多样性': df['type'].nunique() / len(df),
            '类型总数': df['type'].nunique(),
            '类型均衡度': 1 - calculate_gini(df['type'].value_counts().values)
        }

        # 4. 服务均衡性
        equity_metrics = {
            '空间均衡性': 1 - calculate_spatial_gini(df),
            '区域覆盖均衡度': calculate_regional_coverage_balance(df),
            '服务分布均衡度': calculate_service_distribution_balance(df)
        }

        # 5. 服务效率
        efficiency_metrics = {
            '单位面积服务效率': len(df) / (total_area * df['type'].nunique()),
            '服务密集度': len(df) / calculate_service_area(df),
            '服务聚集度': calculate_service_clustering(df)
        }

        # 合并所有指标
        all_metrics = {
            **basic_metrics,
            **spatial_metrics,
            **diversity_metrics,
            **equity_metrics,
            **efficiency_metrics
        }

        return all_metrics

    except Exception as e:
        print(f"服务水平分析错误: {str(e)}")
        return None

def extract_service_features(city1_dfs, city2_dfs):
    """提取两个城市的服务特征"""
    features = {
        '城市1': {'服务水平': {}, '特色服务类型': set()},
        '城市2': {'服务水平': {}, '特色服务类型': set()}
    }

    for city_name, city_dfs in [('城市1', city1_dfs), ('城市2', city2_dfs)]:
        total_area = calculate_city_area(city_dfs)

        for service_name, df in city_dfs.items():
            # 分析服务水平
            service_level = analyze_service_level(df, total_area)
            if service_level:
                features[city_name]['服务水平'][service_name] = service_level

            # 记录特色服务类型
            features[city_name]['特色服务类型'].update(df['type'].unique())

    return features

def compare_service_levels(features):
    """比较两个城市的服务水平"""
    comparison = {}

    # 获取共同的服务类型
```

```
    common_services = set(features['城市1']['服务水平'].keys()) & \
                      set(features['城市2']['服务水平'].keys())

    for service in common_services:
        service_comp = {
            '城市1': features['城市1']['服务水平'][service],
            '城市2': features['城市2']['服务水平'][service],
            '差异分析': {}
        }

        # 计算各指标的差异
        for metric in features['城市1']['服务水平'][service].keys():
            diff = features['城市1']['服务水平'][service][metric] - \
                   features['城市2']['服务水平'][service][metric]
            service_comp['差异分析'][metric] = diff

        comparison[service] = service_comp

    return comparison

# 新增的辅助函数
def normalize_indicators(comparison):
    """标准化指标值"""
    normalized = {}

    for service in comparison.keys():
        normalized[service] = {'城市1': {}, '城市2': {}, '差异分析': {}}

        for metric in comparison[service]['城市1'].keys():
            # 获取两个城市的指标值
            val1 = comparison[service]['城市1'][metric]
            val2 = comparison[service]['城市2'][metric]

            # 计算最大最小值
            max_val = max(val1, val2)
            min_val = min(val1, val2)

            # 标准化
            if max_val != min_val:
                normalized[service]['城市1'][metric] = (val1 - min_val) / (max_val - min_val)
                normalized[service]['城市2'][metric] = (val2 - min_val) / (max_val - min_val)
            else:
                normalized[service]['城市1'][metric] = 0.5
                normalized[service]['城市2'][metric] = 0.5

            # 计算差异
            normalized[service]['差异分析'][metric] = \
                normalized[service]['城市1'][metric] - normalized[service]['城市2'][metric]

    return normalized

def calculate_significance(val1, val2, threshold=0.1):
    """计算差异显著性"""
    diff = abs(val1 - val2)
    return diff > threshold

def calculate_weighted_score(metrics_dict, weights=None):
    """计算加权得分"""
    if weights is None:
        weights = {k: 1.0 for k in metrics_dict.keys()}
```

```
    total_weight = sum(weights.values())
    weighted_sum = sum(metrics_dict[k] * weights.get(k, 1.0)
                       for k in metrics_dict.keys())

    return weighted_sum / total_weight

def get_service_weights():
    """获取服务类型权重"""
    return {
        'Medical and health': 1.2,
        'Education': 1.2,
        'Transportation facilities': 1.1,
        'Food and beverage service': 1.0,
        'Retail service': 1.0,
        'Finance and insurance': 1.0,
        'Government and social organizations': 1.0,
        'Lifestyle service': 0.9,
        'Business residential': 0.9,
        'Accommodation service': 0.8
    }

def calculate_statistical_significance(data1, data2):
    """计算统计显著性"""
    if len(data1) > 1 and len(data2) > 1:
        _, p_value = ttest_ind(data1, data2)
        return p_value < 0.05
    return False
def identify_advantages(comparison):
    """识别每个城市的优势和劣势（改进版多层次评估）"""
    advantages = {
        '城市1': {
            '显著优势': [],
            '轻微优势': [],
            '劣势服务': [],
            '层级优势': {},
            '特色服务': []
        },
        '城市2': {
            '显著优势': [],
            '轻微优势': [],
            '劣势服务': [],
            '层级优势': {},
            '特色服务': []
        }
    }

    # 定义指标层级及其权重
    metric_levels = {
        '基础服务能力': {
            'metrics': ['服务覆盖率', '服务密度', '服务点数量'],
            'weight': 1.2
        },
        '空间可及性': {
            'metrics': ['空间可达性', '空间集中度', '空间分散度'],
            'weight': 1.0
        },
        '服务多样性': {
            'metrics': ['类型多样性', '类型总数', '类型均衡度'],
            'weight': 0.8
        },
        '服务均衡性': {
```

```
        'metrics': ['空间均衡性', '区域覆盖均衡度', '服务分布均衡度'],
        'weight': 1.1
    },
    '服务效率': {
        'metrics': ['单位面积服务效率', '服务密集度', '服务聚集度'],
        'weight': 0.9
    }
}

# 获取服务类型权重
service_weights = get_service_weights()

# 标准化指标
normalized_comp = normalize_indicators(comparison)

# 分析每个服务类型
for service, comp in normalized_comp.items():
    level_scores = {'城市1': {}, '城市2': {}}
    service_weight = service_weights.get(service, 1.0)

    # 计算每个层级的加权得分
    for level, level_info in metric_levels.items():
        metrics = level_info['metrics']
        level_weight = level_info['weight']

        for city in ['城市1', '城市2']:
            available_metrics = {m: comp[city].get(m, 0) for m in metrics if m in comp[city]}
            if available_metrics:
                # 计算层级得分并应用权重
                metric_values = list(available_metrics.values())
                if metric_values:  # 确保有值
                    level_scores[city][level] = np.mean(metric_values) * level_weight
                else:
                    level_scores[city][level] = 0

    # 计算总体加权得分
    # 修改这里：只使用数值类型的值计算平均值
    total_score1 = np.mean([v for v in level_scores['城市1'].values()
                            if isinstance(v, (int, float))]) * service_weight
    total_score2 = np.mean([v for v in level_scores['城市2'].values()
                            if isinstance(v, (int, float))]) * service_weight

    # 设置优势判断阈值
    significant_threshold = 0.2
    minor_threshold = 0.1

    # 判断优势程度
    score_diff = total_score1 - total_score2
    if abs(score_diff) > significant_threshold:
        if score_diff > 0:
            advantages['城市1']['显著优势'].append({
                'service': service,
                'advantage_degree': score_diff,
                'significance': True  # 简化统计显著性判断
            })
            advantages['城市2']['劣势服务'].append(service)
        else:
            advantages['城市2']['显著优势'].append({
                'service': service,
                'advantage_degree': -score_diff,
                'significance': True  # 简化统计显著性判断
            })
```

```
            advantages['城市1']['劣势服务'].append(service)
        elif abs(score_diff) > minor_threshold:
            if score_diff > 0:
                advantages['城市1']['轻微优势'].append({
                    'service': service,
                    'advantage_degree': score_diff
                })
            else:
                advantages['城市2']['轻微优势'].append({
                    'service': service,
                    'advantage_degree': -score_diff
                })

        # 分析层级优势
        for level in metric_levels.keys():
            if level in level_scores['城市1'] and level in level_scores['城市2']:
                score_diff = level_scores['城市1'][level] - level_scores['城市2'][level]

                if abs(score_diff) > minor_threshold:
                    target_city = '城市1' if score_diff > 0 else '城市2'
                    if level not in advantages[target_city]['层级优势']:
                        advantages[target_city]['层级优势'][level] = []

                    advantages[target_city]['层级优势'][level].append({
                        'service': service,
                        'advantage_degree': abs(score_diff),
                        'significance': True  # 简化统计显著性判断
                    })

        # 识别特色服务
        for city in ['城市1', '城市2']:
            unique_metrics = set(comp[city].keys()) - set(comp['城市2' if city == '城市1' else '城市
1'].keys())
            if unique_metrics:
                advantages[city]['特色服务'].append({
                    'service': service,
                    'unique_metrics': list(unique_metrics),
                    'metric_scores': {metric: comp[city][metric] for metric in unique_metrics}
                })

    # 对层级优势进行排序
    for city in ['城市1', '城市2']:
        for level in advantages[city]['层级优势']:
            advantages[city]['层级优势'][level].sort(
                key=lambda x: x['advantage_degree'],
                reverse=True
            )

    return advantages
def visualize_advantages_enhanced(advantages):
    """Enhanced visualization of advantages analysis"""

    # Define metric levels with English names
    metric_levels = {
        'Basic Service Capability': ['Coverage Rate', 'Service Density', 'Point Count'],
        'Spatial Accessibility': ['Accessibility', 'Centrality', 'Dispersion'],
        'Service Diversity': ['Type Diversity', 'Type Count', 'Type Balance'],
        'Service Equity': ['Spatial Equity', 'Coverage Balance', 'Distribution Balance'],
        'Service Efficiency': ['Area Efficiency', 'Service Density', 'Clustering']
    }

    # Create subplots
```

```
fig = plt.figure(figsize=(20, 12))
gs = fig.add_gridspec(2, 2)

# 1. Advantage Service Comparison (Upper Left)
ax1 = fig.add_subplot(gs[0, 0])
cities = ['City 1', 'City 2']
significant_advantages = [len(advantages[f'城市{i+1}']['显著优势']) for i in range(2)]
minor_advantages = [len(advantages[f'城市{i+1}']['轻微优势']) for i in range(2)]

x = np.arange(len(cities))
width = 0.35

ax1.bar(x - width/2, significant_advantages, width, label='Significant Advantage',
       color='lightblue', edgecolor='black', linewidth=1)
ax1.bar(x + width/2, minor_advantages, width, label='Minor Advantage',
       color='lightblue', edgecolor='black', linewidth=1)

ax1.set_title('Distribution of City Advantages', fontsize=12, pad=20)
ax1.set_xticks(x)
ax1.set_xticklabels(cities)
ax1.legend()
ax1.grid(True, alpha=0.3)

# Add value labels
for i, v in enumerate(significant_advantages):
   ax1.text(i - width/2, v, str(v), ha='center', va='bottom')
for i, v in enumerate(minor_advantages):
   ax1.text(i + width/2, v, str(v), ha='center', va='bottom')

# 2. Level Advantage Heatmap (Upper Right)
ax2 = fig.add_subplot(gs[0, 1])

levels = list(metric_levels.keys())
heatmap_data = np.zeros((len(cities), len(levels)))

chinese_levels = {
   'Basic Service Capability': '基础服务能力',
   'Spatial Accessibility': '空间可及性',
   'Service Diversity': '服务多样性',
   'Service Equity': '服务均衡性',
   'Service Efficiency': '服务效率'
}

for i, city in enumerate(['城市1', '城市2']):
   for j, level_eng in enumerate(levels):
      level_ch = chinese_levels[level_eng]
      if level_ch in advantages[city]['层级优势']:
         heatmap_data[i, j] = len(advantages[city]['层级优势'][level_ch])

sns.heatmap(heatmap_data, annot=True, fmt='g', cmap=cmap,
          xticklabels=levels, yticklabels=cities, ax=ax2)
ax2.set_title('Level Advantage Distribution Heatmap', fontsize=12, pad=20)
plt.setp(ax2.get_xticklabels(), rotation=45, ha='right')

# 3. Special Service Analysis (Lower Left)
ax3 = fig.add_subplot(gs[1, 0])
special_services = [len(advantages[f'城市{i+1}']['特色服务']) for i in range(2)]

bars = ax3.bar(cities, special_services, color=['skyblue', 'lightcoral'],
             edgecolor='black', linewidth=1)

ax3.set_title('Number of Special Services', fontsize=12, pad=20)
ax3.grid(True, alpha=0.3)
```

```
    # Add value labels
    for bar in bars:
        height = bar.get_height()
        ax3.text(bar.get_x() + bar.get_width()/2., height,
                f'{int(height)}',
                ha='center', va='bottom')

    # 4. Advantage Degree Distribution (Lower Right)
    ax4 = fig.add_subplot(gs[1, 1])

    advantage_degrees = {city: [] for city in cities}
    for i, city in enumerate(['城市1', '城市2']):
        for level in advantages[city]['层级优势']:
            for adv in advantages[city]['层级优势'][level]:
                advantage_degrees[cities[i]].append(adv['advantage_degree'])

    for city, degrees in advantage_degrees.items():
        if degrees:
            sns.kdeplot(data=degrees, label=city, ax=ax4, fill=True, linewidth=2)

    ax4.set_title('Distribution of Advantage Degree', fontsize=12, pad=20)
    ax4.set_xlabel('Advantage Degree')
    ax4.set_ylabel('Density')
    ax4.legend()
    ax4.grid(True, alpha=0.3)

    plt.tight_layout()

    desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")
    plt.savefig(os.path.join(desktop_path, 'advantages_analysis.png'),
                dpi=300, bbox_inches='tight')
    plt.show()

def create_enhanced_radar_chart(comparison, advantages):
    """Create enhanced multi-level service radar chart"""

    fig = plt.figure(figsize=(6, 4))

    # Define levels and weights in English
    metric_levels = {
        'Basic Service Capability': 1.2,
        'Spatial Accessibility': 1.0,
        'Service Diversity': 0.8,
        'Service Equity': 1.1,
        'Service Efficiency': 0.9
    }

    chinese_levels = {
        'Basic Service Capability': '基础服务能力',
        'Spatial Accessibility': '空间可及性',
        'Service Diversity': '服务多样性',
        'Service Equity': '服务均衡性',
        'Service Efficiency': '服务效率'
    }

    level_names = list(metric_levels.keys())
    angles = np.linspace(0, 2*np.pi, len(level_names), endpoint=False)

    level_scores = {'City 1': [], 'City 2': []}
    significance_markers = {'City 1': [], 'City 2': []}

    for i, city in enumerate(['城市1', '城市2']):
```

```
        eng_city = f'City {i+1}'
        for level_eng in level_names:
            scores = []
            significant_count = 0
            weight = metric_levels[level_eng]

            level_ch = chinese_levels[level_eng]
            if level_ch in advantages[city]['层级优势']:
                for adv in advantages[city]['层级优势'][level_ch]:
                    scores.append(adv['advantage_degree'])
                    if adv['significance']:
                        significant_count += 1

            level_scores[eng_city].append(np.mean(scores) * weight if scores else 0)
            significance_markers[eng_city].append(significant_count)

    angles = np.concatenate((angles, [angles[0]]))
    for city in ['City 1', 'City 2']:
        level_scores[city].append(level_scores[city][0])
    level_names.append(level_names[0])

    ax = fig.add_subplot(111, projection='polar')

    colors = {'City 1': 'skyblue', 'City 2': 'lightcoral'}
    for city in ['City 1', 'City 2']:
        ax.plot(angles, level_scores[city], 'o-', linewidth=2,
                label=city, color=colors[city])
        ax.fill(angles, level_scores[city], alpha=0.25, color=colors[city])

    ax.set_xticks(angles[:-1])
    ax.set_xticklabels(level_names[:-1], fontsize=10)

    ax.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))

    plt.title('Multi-level Comparison of Urban Service Levels\n(Weighted Analysis)',
              pad=20, fontsize=12)

    ax.grid(True, alpha=0.3)

    desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")
    plt.savefig(os.path.join(desktop_path, 'radar_analysis.png'),
                dpi=300, bbox_inches='tight')

    plt.show()

def save_results_to_df(comparison, advantages):
    """将分析结果保存为DataFrame并导出到桌面"""
    # 1. 创建详细指标DataFrame
    data = []
    indices = []

    metric_levels = {
        '基础服务能力': ['服务覆盖率', '服务密度', '服务点数量'],
        '空间可及性': ['空间可达性', '空间集中度', '空间分散度'],
        '服务多样性': ['类型多样性', '类型总数', '类型均衡度'],
        '服务均衡性': ['空间均衡性', '区域覆盖均衡度', '服务分布均衡度'],
        '服务效率': ['单位面积服务效率', '服务密集度', '服务聚集度']
    }

    for service in comparison.keys():
        for city in ['城市1', '城市2']:
            for level, metrics in metric_levels.items():
                for metric in metrics:
```

```
            if metric in comparison[service][city]:
                data.append({
                    '服务类型': service,
                    '城市': city,
                    '评估层级': level,
                    '指标': metric,
                    '原始值': comparison[service][city][metric],
                    '标准化值': normalize_indicators(comparison)[service][city][metric]
                })

# 创建详细指标DataFrame
results_df = pd.DataFrame(data)

# 2. 创建优势分析DataFrame
advantage_data = []
for city in ['城市1', '城市2']:
    # 显著优势
    for adv in advantages[city]['显著优势']:
        advantage_data.append({
            '城市': city,
            '优势类型': '显著优势',
            '服务类型': adv['service'],
            '优势程度': adv['advantage_degree'],
            '统计显著性': adv['significance']
        })

    # 轻微优势
    for adv in advantages[city]['轻微优势']:
        advantage_data.append({
            '城市': city,
            '优势类型': '轻微优势',
            '服务类型': adv['service'],
            '优势程度': adv['advantage_degree'],
            '统计显著性': None
        })

advantage_df = pd.DataFrame(advantage_data)

# 3. 创建层级优势DataFrame
level_advantage_data = []
for city in ['城市1', '城市2']:
    for level, advantages_list in advantages[city]['层级优势'].items():
        for adv in advantages_list:
            level_advantage_data.append({
                '城市': city,
                '评估层级': level,
                '服务类型': adv['service'],
                '优势程度': adv['advantage_degree'],
                '统计显著性': adv['significance']
            })

level_advantage_df = pd.DataFrame(level_advantage_data)

# 4. 创建特色服务DataFrame
special_service_data = []
for city in ['城市1', '城市2']:
    for special in advantages[city]['特色服务']:
        for metric in special['unique_metrics']:
            special_service_data.append({
                '城市': city,
```

```
                '服务类型': special['service'],
                '特色指标': metric,
                '指标得分': special['metric_scores'][metric]
            })

    special_service_df = pd.DataFrame(special_service_data)

    # 获取桌面路径
    desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")

    # 保存所有DataFrame到Excel的不同sheet
    with pd.ExcelWriter(os.path.join(desktop_path, 'service_analysis_complete.xlsx')) as writer:
        results_df.to_excel(writer, sheet_name='详细指标', index=False)
        advantage_df.to_excel(writer, sheet_name='优势分析', index=False)
        level_advantage_df.to_excel(writer, sheet_name='层级优势', index=False)
        special_service_df.to_excel(writer, sheet_name='特色服务', index=False)

    print(f"完整分析结果已保存至: {os.path.join(desktop_path,
'service_analysis_complete.xlsx')}")

    return {
        '详细指标': results_df,
        '优势分析': advantage_df,
        '层级优势': level_advantage_df,
        '特色服务': special_service_df
    }

def print_detailed_analysis(comparison, advantages):
    """打印详细分析结果（改进版）"""
    print("\n=== 城市服务水平详细分析 ===\n")

    # 1. 总体优势分析
    print("1. 总体优势分析:")
    for city in ['城市1', '城市2']:
        print(f"\n{city}:")
        print("  显著优势服务:")
        for adv in advantages[city]['显著优势']:
            significance = "（统计显著）" if adv['significance'] else "（非统计显著）"
            print(f"    - {adv['service']} (优势程度: {adv['advantage_degree']:.4f}) {significance}")

        print("\n  轻微优势服务:")
        for adv in advantages[city]['轻微优势']:
            print(f"    - {adv['service']} (优势程度: {adv['advantage_degree']:.4f})")

    # 2. 层级优势分析
    print("\n2. 层级优势分析:")
    for city in ['城市1', '城市2']:
        print(f"\n{city}层级优势:")
        for level, advantages_list in advantages[city]['层级优势'].items():
            print(f"  {level}:")
            for adv in advantages_list:
                significance = "（统计显著）" if adv['significance'] else "（非统计显著）"
                print(f"    - {adv['service']} (优势程度: {adv['advantage_degree']:.4f}) {significance}")

    # 3. 特色服务分析
    print("\n3. 特色服务分析:")
    for city in ['城市1', '城市2']:
        print(f"\n{city}特色服务:")
        for special in advantages[city]['特色服务']:
            print(f"  {special['service']}:")
```

```
            for metric, score in special['metric_scores'].items():
                print(f"    - {metric}: {score:.4f}")

    # 4. 统计显著性总结
    print("\n4. 统计显著性总结:")
    for city in ['城市1', '城市2']:
        significant_count = sum(1 for adv in advantages[city]['显著优势'] if adv['significance'])
        total_count = len(advantages[city]['显著优势'])
        if total_count > 0:
            significant_ratio = significant_count / total_count * 100
            print(f"\n{city}显著优势的统计可靠性:")
            print(f"  - 统计显著的优势数量: {significant_count}/{total_count}
({significant_ratio:.1f}%)")

def main():
    """主函数"""
    try:


        # 1. 加载数据
        print("正在加载城市数据...")
        city1_dfs = load_city_data(3)  # Appendix 3
        city2_dfs = load_city_data(4)  # Appendix 4

        # 2. 提取服务特征
        print("\n正在提取服务特征...")
        features = extract_service_features(city1_dfs, city2_dfs)

        # 3. 比较服务水平
        print("\n正在比较服务水平...")
        comparison = compare_service_levels(features)

        # 4. 识别优势劣势
        print("\n正在识别优势劣势...")
        advantages = identify_advantages(comparison)

        # 5. 保存分析结果
        print("\n正在保存分析结果...")
        results = save_results_to_df(comparison, advantages)

        # 6. 创建可视化
        print("\n正在创建可视化...")

        # 6.1 优势分析可视化
        print("- 创建优势分析图...")
        visualize_advantages_enhanced(advantages)

        # 6.2 雷达图
        print("- 创建雷达图...")
        create_enhanced_radar_chart(comparison, advantages)

        # 7. 打印详细分析
        print("\n正在生成详细分析报告...")
        print_detailed_analysis(comparison, advantages)

        # 8. 输出分析摘要
        print("\n=== 分析摘要 ===")

        # 8.1 显著优势服务数量
        for city in ['城市1', '城市2']:
```

```
        sig_adv = len(advantages[city]['显著优势'])
        minor_adv = len(advantages[city]['轻微优势'])
        print(f"\n{city}优势概况:")
        print(f"- 显著优势服务数量: {sig_adv}")
        print(f"- 轻微优势服务数量: {minor_adv}")

    # 8.2 特色服务数量
    for city in ['城市1', '城市2']:
        special_count = len(advantages[city]['特色服务'])
        print(f"\n{city}特色服务数量: {special_count}")

    # 8.3 统计显著性总结
    print("\n统计显著性总结:")
    for city in ['城市1', '城市2']:
        significant_count = sum(1 for adv in advantages[city]['显著优势']
                                if adv['significance'])
        total_count = len(advantages[city]['显著优势'])
        if total_count > 0:
            significant_ratio = significant_count / total_count * 100
            print(f"{city}: {significant_count}/{total_count} "
                  f"显著优势具有统计显著性 ({significant_ratio:.1f}%)")

    # 9. 保存路径提示
    desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")
    print(f"\n所有分析结果已保存至桌面:")
    print(f"- 完整分析报告: {os.path.join(desktop_path, 'service_analysis_complete.xlsx')}")
    print(f"- 优势分析图: {os.path.join(desktop_path, 'advantages_analysis.png')}")
    print(f"- 雷达图: {os.path.join(desktop_path, 'radar_analysis.png')}")

    print("\n分析完成！")

    return {
        'comparison': comparison,
        'advantages': advantages,
        'results': results
    }

  except Exception as e:
    print(f"\n错误: 分析过程中出现异常: {str(e)}")
    import traceback
    traceback.print_exc()
    return None

if __name__ == "__main__":
  # 设置进度条样式
  tqdm.pandas()

  # 运行主函数
  results = main()

  # 检查结果
  if results is not None:
    print("\n分析结果可通过返回的字典访问:")
    print("- results['comparison']: 服务水平比较结果")
    print("- results['advantages']: 优势劣势分析结果")
    print("- results['results']: 详细分析数据框")
```

No: 3	The Third Question

```
# 基础数据处理库
import os
import pandas as pd
import numpy as np
from tqdm import tqdm
import warnings
warnings.filterwarnings('ignore')

# 数据可视化库
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.gridspec import GridSpec
import plotly.express as px
import plotly.graph_objects as go

# 空间分析库
from scipy.spatial import ConvexHull, Voronoi
from scipy.spatial.distance import cdist
from shapely.geometry import Point, Polygon
import geopandas as gpd

# 网络分析库
import networkx as nx
from sklearn.neighbors import NearestNeighbors

# 统计分析库
from scipy import stats
from scipy.stats import gaussian_kde
from sklearn.cluster import DBSCAN, KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

# 机器学习库
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# 多进程处理
import multiprocessing as mp
from joblib import Parallel, delayed

# 系统和工具库
import sys
import time
import json
import pickle
from datetime import datetime
from pathlib import Path

# 设置中文显示
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False    # 用来正常显示负号
def load_city_data(appendix_number):
    """加载城市数据"""
    data_folder = os.path.join(os.path.expanduser("~"), "Desktop", f"Appendix {appendix_number}")

    file_names = [
        'Accommodation service data.csv',
        'Business-residential data.csv',
        'Finance and insurance data.csv',
        'Food and beverage service data.csv',
        'Government and social organizations data.csv',
        'Lifestyle service data.csv',
        'Medical and health data.csv',
        'Retail service data.csv',
```

```
        'Science, education, and culture data.csv',
        'Transportation facilities data.csv'
    ]

    # 可能的编码列表
    encodings = ['gbk', 'utf-8', 'gb18030', 'iso-8859-1', 'latin1']

    city_dfs = {}
    for file_name in tqdm(file_names, desc="加载数据文件"):
        file_path = os.path.join(data_folder, file_name)
        service_name = file_name.replace(' data.csv', '').replace('-', ' ')

        # 尝试不同的编码
        for encoding in encodings:
            try:
                df = pd.read_csv(file_path, encoding=encoding)
                city_dfs[service_name] = df
                break
            except UnicodeDecodeError:
                continue
            except Exception as e:
                print(f"读取文件 {file_name} 时发生错误: {str(e)}")
                break

        if service_name not in city_dfs:
            print(f"警告: 无法读取文件 {file_name}")
            city_dfs[service_name] = pd.DataFrame()  # 创建空DataFrame

    # 打印加载的数据基本信息
    print("\n成功加载的数据集:")
    for name, df in city_dfs.items():
        print(f"{name}: {len(df)} 行")

    return city_dfs
def analyze_medical_density(medical_df):
    """分析医疗设施密度和覆盖率"""
    # 打印坐标范围用于调试
    print(f"\n坐标范围:")
    print(f"X: {medical_df['X'].min():.4f} 到 {medical_df['X'].max():.4f}")
    print(f"Y: {medical_df['Y'].min():.4f} 到 {medical_df['Y'].max():.4f}")

    # 计算研究区域的面积（平方公里）
    x_range = medical_df['X'].max() - medical_df['X'].min()
    y_range = medical_df['Y'].max() - medical_df['Y'].min()

    # 假设坐标单位是千米
    area = x_range * y_range  # 直接作为平方公里

    # 计算医疗设施密度
    density = len(medical_df) / area if area > 0 else 0

    # 计算服务覆盖率（使用2km服务半径）
    grid_size = 100  # 增加网格密度
    x_grid = np.linspace(medical_df['X'].min(), medical_df['X'].max(), grid_size)
    y_grid = np.linspace(medical_df['Y'].min(), medical_df['Y'].max(), grid_size)

    covered_points = 0
    total_points = grid_size * grid_size
    service_radius = 2  # 2km

    for x in x_grid:
        for y in y_grid:
            distances = np.sqrt(
                (medical_df['X'] - x)**2 +
                (medical_df['Y'] - y)**2
```

```
            )
            if np.min(distances) <= service_radius:
                covered_points += 1

    coverage_rate = covered_points / total_points if total_points > 0 else 0

    return {
        'density': density,
        'coverage_rate': coverage_rate,
        'total_facilities': len(medical_df),
        'area': area,
        'x_range': x_range,
        'y_range': y_range
    }

def calculate_emergency_response_time(critical_facilities, transport_df):
    """计算应急响应时间"""
    try:
        medical_df = critical_facilities['医疗设施']

        # 计算每个交通节点到最近医疗设施的时间
        response_times = []
        average_speed = 30  # 30km/h

        # 直接使用坐标值作为千米单位
        for _, transport_point in transport_df.iterrows():
            distances = np.sqrt(
                (medical_df['X'] - transport_point['X'])**2 +
                (medical_df['Y'] - transport_point['Y'])**2
            )

            min_distance = np.min(distances)
            # 转换为分钟（考虑实际道路距离会比直线距离长1.3倍）
            time = (min_distance * 1.3) / (average_speed / 60)
            response_times.append(time)

        response_times = np.array(response_times)

        return {
            'avg_response_time': np.mean(response_times),
            'max_response_time': np.max(response_times),
            'coverage_within_10min': np.mean(response_times <= 10),
            'min_time': np.min(response_times),
            'median_time': np.median(response_times)
        }

    except Exception as e:
        print(f"计算应急响应时间时出错: {str(e)}")
        return {
            'avg_response_time': float('inf'),
            'max_response_time': float('inf'),
            'coverage_within_10min': 0,
            'min_time': float('inf'),
            'median_time': float('inf')
        }

def calculate_sustainability_metrics(medical_df, transport_df):
    """计算城市可持续发展指标"""

    def normalize_data(data):
        """数据标准化"""
        return (data - np.min(data)) / (np.max(data) - np.min(data) + 1e-10)

    def calculate_entropy_weights(data_matrix):
        """计算熵权"""
```

```
    # 数据标准化
    normalized_matrix = normalize_data(data_matrix)

    # 计算每个指标的比重
    p_matrix = normalized_matrix / (np.sum(normalized_matrix, axis=0) + 1e-10)

    # 计算熵值
    entropy = np.zeros(data_matrix.shape[1])
    for j in range(data_matrix.shape[1]):
        entropy[j] = -1.0 / np.log(data_matrix.shape[0])
        for i in range(data_matrix.shape[0]):
            if p_matrix[i,j] > 0:
                entropy[j] *= p_matrix[i,j] * np.log(p_matrix[i,j])

    # 计算差异系数
    g = 1 - entropy

    # 计算权重
    weights = g / (np.sum(g) + 1e-10)

    return weights, normalized_matrix

# 1. 计算医疗资源分布均衡性
def calculate_distribution_balance():
    grid_size = 10
    x_bins = np.linspace(medical_df['X'].min(), medical_df['X'].max(), grid_size+1)
    y_bins = np.linspace(medical_df['Y'].min(), medical_df['Y'].max(), grid_size+1)

    grid_counts = np.zeros((grid_size, grid_size))
    for i in range(grid_size):
        for j in range(grid_size):
            mask = (medical_df['X'] >= x_bins[i]) & (medical_df['X'] < x_bins[i+1]) & \
                   (medical_df['Y'] >= y_bins[j]) & (medical_df['Y'] < y_bins[j+1])
            grid_counts[i,j] = sum(mask)

    cv = np.std(grid_counts) / (np.mean(grid_counts) + 1e-10)
    balance_score = 1 / (1 + cv)

    return balance_score

# 2. 计算服务效率指标
def calculate_service_efficiency():
    service_radii = []
    for _, facility in medical_df.iterrows():
        distances = np.sqrt(
            (transport_df['X'] - facility['X'])**2 +
            (transport_df['Y'] - facility['Y'])**2
        )
        service_radii.append(np.percentile(distances, 75))

    avg_radius = np.mean(service_radii)
    efficiency_score = 1 / (1 + avg_radius/10)

    return efficiency_score

# 3. 计算发展潜力指标
def calculate_development_potential():
    area = (medical_df['X'].max() - medical_df['X'].min()) * \
           (medical_df['Y'].max() - medical_df['Y'].min())
    density = len(medical_df) / (area + 1e-10)
    transport_coverage = len(transport_df) / (area + 1e-10)
    potential_score = (density + transport_coverage) / 2

    return potential_score
```

```
        # 计算各项指标
        balance_score = calculate_distribution_balance()
        efficiency_score = calculate_service_efficiency()
        potential_score = calculate_development_potential()

        # 构建决策矩阵
        indicators = np.array([
            [balance_score, efficiency_score, potential_score],
            [1, 1, 1] # 添加一个参考行，避免单行数据导致的熵值计算问题
        ])

        # 使用熵权法计算权重
        weights, normalized_matrix = calculate_entropy_weights(indicators)

        # 计算综合可持续发展指数（只使用第一行的实际数据）
        sustainability_index = np.sum(normalized_matrix[0] * weights)

        return {
            'distribution_balance': balance_score,
            'service_efficiency': efficiency_score,
            'development_potential': potential_score,
            'sustainability_index': sustainability_index,
            'weights': {
                'distribution_balance': weights[0],
                'service_efficiency': weights[1],
                'development_potential': weights[2]
            }
        }

    def main_resilience_analysis():
        """城市韧性与可持续发展能力评估主函数"""
        print("开始城市韧性与可持续发展能力评估...")

        try:
            # 1. 数据加载
            print("\n1. 加载城市数据...")
            city1_dfs = load_city_data(3)
            city2_dfs = load_city_data(4)

            # 2. 计算核心指标
            print("\n2. 计算核心韧性指标...")
            city1_metrics = {
                '医疗可达性': analyze_medical_density(city1_dfs['Medical and health']),
                '应急响应': calculate_emergency_response_time(
                    {'医疗设施': city1_dfs['Medical and health']},
                    city1_dfs['Transportation facilities']
                ),
                '可持续发展': calculate_sustainability_metrics(
                    city1_dfs['Medical and health'],
                    city1_dfs['Transportation facilities']
                )
            }

            city2_metrics = {
                '医疗可达性': analyze_medical_density(city2_dfs['Medical and health']),
                '应急响应': calculate_emergency_response_time(
                    {'医疗设施': city2_dfs['Medical and health']},
                    city2_dfs['Transportation facilities']
                ),
                '可持续发展': calculate_sustainability_metrics(
                    city2_dfs['Medical and health'],
                    city2_dfs['Transportation facilities']
                )
            }
```

```
        # 3. 打印详细结果
        print("\n=== 城市韧性与可持续发展能力评估结果 ===")
        for city, metrics in {'城市1': city1_metrics, '城市2': city2_metrics}.items():
            print(f"\n{city}:")
            print(f"医疗设施总数: {metrics['医疗可达性']['total_facilities']} 个")
            print(f"研究区域面积: {metrics['医疗可达性']['area']:.2f} 平方公里")
            print(f"医疗设施密度: {metrics['医疗可达性']['density']:.2f} 个/平方公里")
            print(f"医疗服务覆盖率: {metrics['医疗可达性']['coverage_rate']*100:.2f}%")

            print(f"\n应急响应时间:")
            print(f"  - 最短: {metrics['应急响应']['min_time']:.2f} 分钟")
            print(f"  - 平均: {metrics['应急响应']['avg_response_time']:.2f} 分钟")
            print(f"  - 中位数: {metrics['应急响应']['median_time']:.2f} 分钟")
            print(f"  - 最长: {metrics['应急响应']['max_response_time']:.2f} 分钟")
            print(f"10分钟覆盖率: {metrics['应急响应']['coverage_within_10min']*100:.2f}%")


            # 在main_resilience_analysis函数中的输出部分添加
            print(f"\n可持续发展能力:")
            print(f"  - 分布均衡性: {metrics['可持续发展']['distribution_balance']:.4f} (权重: {metrics['可持续发展']['weights']['distribution_balance']:.4f})")
            print(f"  - 服务效率: {metrics['可持续发展']['service_efficiency']:.4f} (权重: {metrics['可持续发展']['weights']['service_efficiency']:.4f})")
            print(f"  - 发展潜力: {metrics['可持续发展']['development_potential']:.4f} (权重: {metrics['可持续发展']['weights']['development_potential']:.4f})")
            print(f"  - 综合指数: {metrics['可持续发展']['sustainability_index']:.4f}")
        return {'城市1': city1_metrics, '城市2': city2_metrics}

    except Exception as e:
        print(f"错误: {str(e)}")
        return None


if __name__ == "__main__":
    # 设置进度条样式
    tqdm.pandas()
    # 运行主函数
    results = main_resilience_analysis()
```