**1. 分析数据**[**¶**](#bookmark1) **(1) 导入必要库**

In [2]:

import pandas as pd

In [1]:

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

#%matplotlib inline

import warnings

warnings.filterwarnings("ignore")

**(2) 导入数据**

|  |
| --- |
| df\_raw = pd.read\_csv("datasets/原始数据.csv") |

**(3) 观察数据**

In [3]:

|  |
| --- |
| df\_raw |

Out[3]:

浦 东 新 区 闵 行 区 宝 山 区 松 江 区 嘉 定 区

金 山 区 松 江 区 青 浦 区 奉 贤 区

**区 县**

**常住人**

**口（万**

**人）**

**出生**

**率**

**(‰)**

**GDP （亿 元）**

**义务教育 投入**

**年份**

**教育财政 拨款情况 （万元）**

**高中在校**

**学生人均**

**经费情况**

**（元）**

**中等职业**

**学校在校**

**学生人均**

**经费情况**

**（元）**

**初中在校**

**学生人均**

**经费情况**

**（元）**

**小学在校**

**学生人均**

**经费情况**

**（元）**

**幼儿园在 园幼儿人 均经费情 况（元）**

**大学**

**高中**

**生 （万 人）**

**生**

**（万**

**人）**

**小学**

**初中**

**生 （万 人）**

**生 （万 人）**

**在园**

**幼儿**

**（万**

**人）**

**0**

**1**

**2**

**3**

**4**

**...**

**120**

**121**

**122**

**123**

568.15

268.88

227.19

195.45

186.34

...

80.51

176.48

121.49

116.74

NaN

NaN

4.23

5.44

NaN

...

NaN

10.83

8.66

7.40

16013.00

2880.11

1771.20

1750.12

2768.30

...

801.56

1040.45

940.01

729.30

NaN

NaN

NaN

NaN

NaN

...

109555.5

151387.6

124050.0

143078.8

2022

2022

2022

2022

2022

...

2016

2016

2016

2016

NaN

NaN

NaN

NaN

NaN

...

183658.3

270169.0

205813.7

214711.5

NaN

NaN

NaN

NaN

NaN

...

45700.37

47451.67

43136.66

42102.77

NaN

NaN

NaN

NaN

NaN

...

35174.54

35627.77

46450.95

28808.14

NaN

NaN

NaN

NaN

NaN

...

35921.45

33183.82

33681.62

36366.51

NaN

NaN

NaN

NaN

NaN

...

27643.80

22898.00

30738.77

27035.03

NaN

NaN

NaN

NaN

NaN

...

29227.79

25242.19

31361.33

28818.61

NaN

NaN

NaN

NaN

0.71

NaN

1.11

NaN

0.94

NaN

...

...

NaN

NaN

NaN

NaN

0.57

NaN

0.60

NaN

NaN

NaN

8.28

NaN

7.77

1.88

7.52

3.63

6.24

3.05

...

...

NaN

NaN

6.06

NaN

3.57

1.67

3.79

2.07

NaN

6.19

5.01

4.96

4.29

...

NaN

4.78

2.53

2.68



**区 县**

**常住人**

**口（万**

**人）**

**出生**

**率**

**(‰)**

**GDP （亿 元）**

**义务教育 投入**

**年份**

**教育财政 拨款情况 （万元）**

**高中在校**

**学生人均**

**经费情况**

**（元）**

**中等职业**

**学校在校**

**学生人均**

**经费情况**

**（元）**

**初中在校**

**学生人均**

**经费情况**

**（元）**

**小学在校**

**学生人均**

**经费情况**

**（元）**

**幼儿园在 园幼儿人 均经费情 况（元）**

**大学**

**高中**

**生 （万 人）**

**生**

**（万**

**人）**

**小学**

**初中**

**生 （万 人）**

**生 （万 人）**

**在园**

**幼儿**

**（万**

**人）**

崇

**124** 明 69.89 5.35 295.29 118952.5 2016 187441.8 55308.80 41990.52 53513.03 41663.93 32269.26 NaN 0.49 1.14 1.64 0.99

区

125 rows × 17 columns

我们可以用到“常住人口（万人） ”、“ 出生率(‰)”、“GDP （亿元） ”、“在园幼儿（万人） ”、“小学生（万人） ”、“初中生（万人） ”这六个特征，同时 通过计算获得“义务教育经费（万元） ”数据作为模型样本标签。

**(4) 计算义务教育经费**

先计算各个阶段的教育经费。

df\_raw["幼儿园教育经费（万元）"] = df\_raw["幼儿园在园幼儿人均经费情况（元）"] \* df\_raw["在园幼儿（万人）"]

In [4]:

df\_raw["小学教育经费（万元）"] = df\_raw["小学在校学生人均经费情况（元）"] \* df\_raw["小学生（万人）"]

df\_raw["初中教育经费（万元）"] = df\_raw["初中在校学生人均经费情况（元）"] \* df\_raw["初中生（万人）"]

df\_raw["高中教育经费（万元）"] = df\_raw["高中在校学生人均经费情况（元）"] \* df\_raw["高中生（万人）"]

df\_raw[[ "教育财政拨款情况（万元）", "幼儿园教育经费（万元）", "小学教育经费（万元）", "初中教育经费（万元）", \ "高中教育经费（万元）"]] \

[df\_raw["小学教育经费（万元）"].notna() & df\_raw["初中教育经费（万元）"].notna()].head()

Out[4]:

**教育财政拨款情况（万元） 幼儿园教育经费（万元） 小学教育经费（万元） 初中教育经费（万元） 高中教育经费（万元）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **37** | 506638.42 | 181431.9820 | 248979.0204 | 80959.3610 | 35115.3344 |
| **38** | 449801.73 | 188178.0660 | 211574.7356 | 149880.5361 | 57347.5320 |
| **39** | 405118.14 | 188121.2619 | 196946.4840 | 143436.0460 | 48611.8182 |
| **40** | 347942.26 | 97909.1764 | 166036.1505 | 131238.7266 | 71632.0938 |
| **42** | 314102.29 | 98829.7236 | 123575.5812 | 89196.3840 | 36411.3035 |

再将小学和初中的教育经费相加，获得义务教育经费；并去除缺数据的样本，重置行号，最终用df\_curated变量表示。

In [5]:

|  |
| --- |
| df\_curated = df\_raw[["区县","年份","常住人口（万人）","出生率(‰)","GDP（亿元）","在园幼儿（万人）","小学生（万人）", "初中生（万人）","小学教育经费（万元）","初中教育经费（万元）"]]  [df\_raw["小学教育经费（万元）"].notna() & df\_raw["初中教育经费（万元）"].notna()]  df\_curated["义务教育经费（万元）"] = df\_curated["小学教育经费（万元）"] + df\_curated["初中教育经费（万元）"] df\_curated.dropna(inplace=True)  df\_curated.reset\_index(drop=True, inplace=True)  df\_curated.head() |

Out[5]:

**小学教育经费（万**

**小学生（万 人）**

**在园幼儿 （万人）**

**初中教育经费（万 元）**

**义务教育经费（万 元）**

**GDP （亿 元）**

**初中生（万 人）**

**常住人口（万 人）**

**出生率 (‰)**

**年份**

**区县**

**元）**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 宝山 区 | 2021 | 225.01 | 4.43 | 1725.56 | 5.35 | 7.59 | 1.91 | 248979.0204 | 80959.3610 | 329938.3814 |
| **1** | 松江 区 | 2021 | 193.88 | 5.59 | 1782.28 | 5.10 | 7.22 | 3.33 | 211574.7356 | 149880.5361 | 361455.2717 |
| **2** | 奉贤 区 | 2021 | 114.71 | 4.14 | 1300.00 | 2.54 | 3.66 | 1.92 | 123575.5812 | 89196.3840 | 212771.9652 |
| **3** | 徐汇 区 | 2020 | 111.31 | 5.17 | 2176.73 | 2.49 | 4.78 | 2.97 | 147392.4472 | 142103.5110 | 289495.9582 |
| **4** | 长宁 区 | 2020 | 69.31 | 4.69 | 1561.17 | 1.25 | 2.29 | 1.34 | 74358.4984 | 70785.9690 | 145144.4674 |

选择合适的列用于构建模型：

df\_train = df\_curated[["常住人口（万人）","出生率(‰)","GDP（亿元）","在园幼儿（万人）","小学生（万人）", "初中生（万人）","义务教育经费（万元）"]].dropna().reset\_index(drop=True)

In [6]:

In [7]:

df\_train.head()

Out[6]:

**常住人口（万人） 出生率(‰) GDP （亿元） 在园幼儿（万人） 小学生（万人） 初中生（万人） 义务教育经费（万元）**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 225.01 | 4.43 | 1725.56 | 5.35 | 7.59 | 1.91 | 329938.3814 |
| **1** | 193.88 | 5.59 | 1782.28 | 5.10 | 7.22 | 3.33 | 361455.2717 |
| **2** | 114.71 | 4.14 | 1300.00 | 2.54 | 3.66 | 1.92 | 212771.9652 |
| **3** | 111.31 | 5.17 | 2176.73 | 2.49 | 4.78 | 2.97 | 289495.9582 |
| **4** | 69.31 | 4.69 | 1561.17 | 1.25 | 2.29 | 1.34 | 145144.4674 |

|  |
| --- |
| df\_train.describe() |

Out[7]:

**常住人口（万人） 出生率(‰) GDP （亿元） 在园幼儿（万人） 小学生（万人） 初中生（万人） 义务教育经费（万元）**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 46.000000 | 46.000000 | 46.000000 | 46.000000 | 46.000000 | 46.000000 | 46.000000 |
| **mean** | 148.631087 | 6.560435 | 1543.412174 | 3.587391 | 4.865435 | 1.944130 | 240355.275343 |
| **std** | 57.010882 | 1.645705 | 659.335288 | 1.882391 | 2.331285 | 0.570643 | 79729.986469 |
| **min** | 63.790000 | 3.620000 | 295.290000 | 0.900000 | 1.550000 | 1.100000 | 129333.699400 |
| **25%** | 114.245000 | 5.275000 | 1167.987500 | 2.532500 | 3.415000 | 1.620000 | 179084.907200 |
| **50%** | 128.815000 | 6.410000 | 1576.210000 | 2.685000 | 4.225000 | 1.825000 | 217514.156050 |
| **75%** | 193.152500 | 7.712500 | 2082.882500 | 5.082500 | 6.555000 | 2.317500 | 279529.017625 |
| **max** | 265.350000 | 10.750000 | 2608.120000 | 7.490000 | 10.300000 | 3.330000 | 451621.206000 |

|  |
| --- |
| df\_train.info()  In [8]: |

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 46 entries, 0 to 45

Data columns (total 7 columns):

# Column Non-Null Count Dtype

0 常住人口（万人） 46 non-null float64

1 出生率(‰) 46 non-null float64

2 GDP（亿元） 46 non-null float64

3 在园幼儿（万人） 46 non-null float64

4 小学生（万人） 46 non-null float64

5 初中生（万人） 46 non-null float64

6 义务教育经费（万元） 46 non-null float64

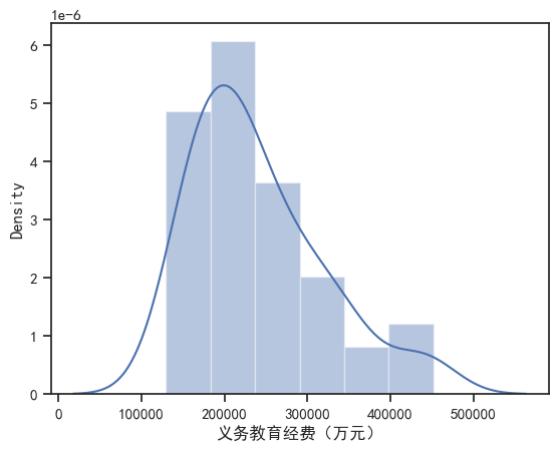
dtypes: float64(7)

memory usage: 2.6 KB

查看目标字段分布：

|  |
| --- |
| sns.set(font\_scale=1.25)  In [9]:  rc = {'font.sans-serif': 'SimHei',  'axes.unicode\_minus': False}  sns.set(context='notebook', style='ticks', rc=rc)  sns.distplot(df\_train['义务教育经费（万元）']) |

Out[9]: <AxesSubplot:xlabel='义务教育经费（万元）', ylabel='Density'>



分布偏左，后续会通过自然对数函数进行转换，以尽量服从正态分布。

使用热力图查看字段间的相关性：

cols = df\_train.corr ().nlargest(10, '义务教育经费（万元）')['义务教育经费（万元）'].index

In [10]:

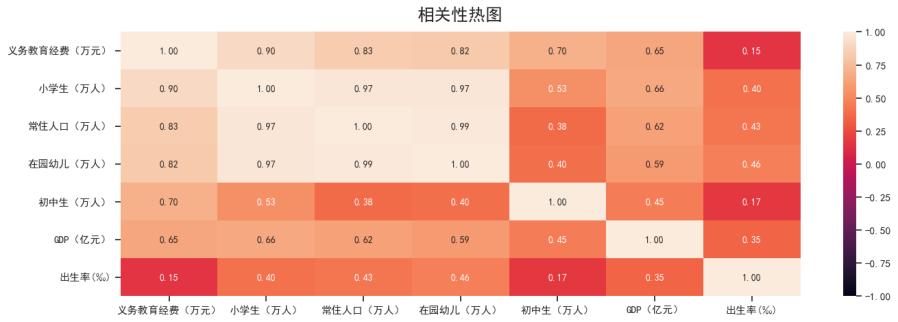
cm = np.corrcoef(df\_train[cols].values.T)

plt.figure(figsize = (16, 5))

heatmap = sns.heatmap(cm, cbar=True, annot=True, fmt='.2f', annot\_kws= {'size':10}, vmin=-1, \

vmax=1, yticklabels=cols.values, xticklabels=cols.values)

heatmap.set\_title('相关性热图', fontdict= {'fontsize':18}, pad=12);



颜色越浅表示相关性越强。该热力图显示“义务教育经费（万元） ”与其它六个特征都具有较强的相关性。

**2. 特征工程**

**(1) 读取数据**

In [11]:

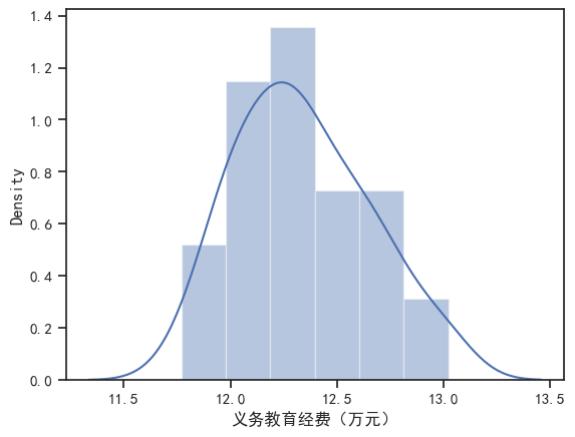
In [12]:

|  |
| --- |
| raw\_df = df\_train |

**(2) 尝试对标签列进行转换 ，使之尽量呈正态分布**

|  |
| --- |
| y\_train = np.log1p(raw\_df['义务教育经费（万元）'])  sns.distplot(y\_train) |

Out[12]: <AxesSubplot:xlabel='义务教育经费（万元）', ylabel='Density'>



新增log结果列，并取名为label：

In [14]:

In [15]:

In [13]: raw\_df["label"] = y\_train

raw\_df.head()

Out[13]:

**常住人口（万人） 出生率(‰) GDP （亿元） 在园幼儿（万人） 小学生（万人） 初中生（万人） 义务教育经费（万元） label**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 225.01 | 4.43 | 1725.56 | 5.35 | 7.59 | 1.91 | 329938.3814 | 12.706664 |
| **1** | 193.88 | 5.59 | 1782.28 | 5.10 | 7.22 | 3.33 | 361455.2717 | 12.797896 |
| **2** | 114.71 | 4.14 | 1300.00 | 2.54 | 3.66 | 1.92 | 212771.9652 | 12.267981 |
| **3** | 111.31 | 5.17 | 2176.73 | 2.49 | 4.78 | 2.97 | 289495.9582 | 12.575900 |
| **4** | 69.31 | 4.69 | 1561.17 | 1.25 | 2.29 | 1.34 | 145144.4674 | 11.885492 |

**(2) 复制出新dataframe ，用于特征工程**

|  |
| --- |
| raw\_train\_df = raw\_df.copy () |

**(3) 标准化数据**

仅对非标签列进行标准化，使用z-score标准分数。

|  |
| --- |
| numeric\_cols = raw\_train\_df.columns[(raw\_train\_df.dtypes != 'object') \  & (~raw\_train\_df.columns.isin(['义务教育经费（万元）', 'label']))]  numeric\_cols |

Out[15]: Index(['常住人口（万人）', '出生率(‰)', 'GDP（亿元）', '在园幼儿（万人）', '小学生（万人）', '初中生（万人）'], dtype='object')

|  |
| --- |
| numeric\_col\_mean = raw\_train\_df.loc[:, numeric\_cols].mean ()  In [16]:  In [17]:  numeric\_col\_std = raw\_train\_df.loc[:, numeric\_cols].std() |

|  |
| --- |
| numeric\_col\_mean |

Out[17]: 常住人口（万人） 148.631087

出生率(‰) 6.560435

GDP（亿元） 1543.412174

在园幼儿（万人） 3.587391

小学生（万人） 4.865435

初中生（万人） 1.944130

dtype: float64

In [18]: numeric\_col\_std

Out[18]: 常住人口（万人） 57.010882

出生率(‰) 1.645705

GDP（亿元） 659.335288

在园幼儿（万人） 1.882391

小学生（万人） 2.331285

初中生（万人） 0.570643

dtype: float64

In [19]: raw\_train\_df.loc[:, numeric\_cols] = (raw\_train\_df.loc[:, numeric\_cols] - numeric\_col\_mean)/numeric\_col\_std raw\_train\_df["label"] = y\_train

去除原来的“义务教育经费（万元） ”列：

|  |
| --- |
| raw\_train\_df.drop(columns= ["义务教育经费（万元）"], inplace=True)  In [20]:  raw\_train\_df.head() |

Out[20]:

**常住人口（万人） 出生率(‰) GDP （亿元） 在园幼儿（万人） 小学生（万人） 初中生（万人） label**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 1.339725 -1.294542 | 0.276260 | 0.936367 | 1.168697 | -0.059810 | 12.706664 |
| **1** | 0.793689 -0.589677 | 0.362286 | 0.803557 | 1.009986 | 2.428610 | 12.797896 |
| **2** | -0.594993 -1.470758 | -0.369178 | -0.556415 | -0.517069 | -0.042286 | 12.267981 |
| **3** | -0.654631 -0.844887 | 0.960540 | -0.582977 | -0.036647 | 1.797743 | 12.575900 |
| **4** | -1.391332 -1.136555 | 0.026933 | -1.241714 | -1.104728 | -1.058683 | 11.885492 |

**3. 建立模型**

**(1) 准备样本**

构建验证样本集，用于后续验证模型效果：

|  |
| --- |
| revised\_train\_df = raw\_train\_df.copy ()  In [21]:  from sklearn.model\_selection import train\_test\_split  #revised\_train\_df, revised\_test\_df = train\_test\_split(verify\_df, test\_size=0.2)  \_, revised\_test\_df = train\_test\_split(raw\_train\_df, test\_size=0.2)  y\_train = revised\_train\_df.pop ("label")  revised\_train\_df.head() |

Out[21]:

**常住人口（万人） 出生率(‰) GDP （亿元） 在园幼儿（万人） 小学生（万人） 初中生（万人）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0** | 1.339725 -1.294542 | 0.276260 | 0.936367 | 1.168697 | -0.059810 |
| **1** | 0.793689 -0.589677 | 0.362286 | 0.803557 | 1.009986 | 2.428610 |
| **2** | -0.594993 -1.470758 | -0.369178 | -0.556415 | -0.517069 | -0.042286 |
| **3** | -0.654631 -0.844887 | 0.960540 | -0.582977 | -0.036647 | 1.797743 |
| **4** | -1.391332 -1.136555 | 0.026933 | -1.241714 | -1.104728 | -1.058683 |

输出测试样本，便于直观观察模型计算结果：

|  |
| --- |
| y\_test = revised\_test\_df.pop ("label")  In [22]:  In [23]:  revised\_test\_df |

Out[22]:

**常住人口（万人） 出生率(‰) GDP （亿元） 在园幼儿（万人） 小学生（万人） 初中生（万人）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **44** | -0.559386 0.510155 | -1.234747 | -0.482042 | -0.461306 | 0.220575 |
| **5** | -0.427657 -1.245931 | 0.854221 | -0.487354 | -0.285437 | 0.711249 |
| **13** | -0.687081 -0.170404 | 0.860864 | -0.593602 | -0.126726 | 1.534881 |
| **31** | -0.303295 0.364321 | 0.349728 | -0.492667 | -0.564253 | -0.007238 |
| **10** | -0.376965 -0.796275 | -0.529931 | -0.407668 | -0.615727 | -0.287624 |
| **7** | 1.313590 -0.996797 | 0.053187 | 1.069177 | 1.061460 | -0.550485 |
| **43** | -0.476069 1.275784 | -0.915167 | -0.561728 | -0.555674 | -0.480388 |
| **3** | -0.654631 -0.844887 | 0.960540 | -0.582977 | -0.036647 | 1.797743 |
| **28** | -0.586398 -0.522836 | -1.064575 | -0.529853 | -0.632885 | -0.235051 |
| **11** | -0.605868 -1.331000 | -0.535725 | -0.545790 | -0.572832 | -0.235051 |

|  |
| --- |
| y\_train.head() |

|  |  |
| --- | --- |
| Out[23]: 0 | 12.706664 |
| 1 | 12.797896 |
| 2 | 12.267981 |
| 3 | 12.575900 |

4 11.885492

Name: label, dtype: float64

**(2) 线性回归**

我们选取岭回归（Ridge Regression）。

|  |
| --- |
| from sklearn.linear\_model import Ridge  In [24]:  In [25]:  In [26]:  from sklearn.model\_selection import cross\_val\_score |

通过cross\_val\_score选择超参alpha。

|  |
| --- |
| alphas = np.logspace(-2, 1, 20)  alphas |

Out[25]: array([ 0.01 , 0.0143845 , 0.02069138, 0.02976351, 0.04281332,

0.06158482, 0.08858668, 0.1274275 , 0.18329807, 0.26366509,

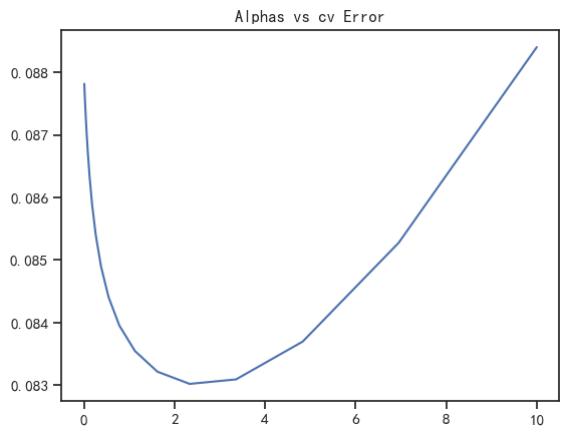
0.37926902, 0.54555948, 0.78475997, 1.12883789, 1.62377674,

2.33572147, 3.35981829, 4.83293024, 6.95192796, 10. ])

|  |
| --- |
| X\_train = revised\_train\_df.values  test\_scores= []  for alpha in alphas:  ridge\_es = Ridge(alpha)  test\_score = np.sqrt(-cross\_val\_score (ridge\_es, X\_train, y\_train, cv=10, scoring='neg\_mean\_squared\_error')) test\_scores.append(np.mean (test\_score)) |

|  |
| --- |
| plt.plot(alphas, test\_scores)  In [27]:  In [28]:  plt.title('Alphas vs cv Error') |

Out[27]: Text(0.5, 1.0, 'Alphas vs cv Error')



|  |
| --- |
| minpos = test\_scores.index(min(test\_scores))  print(alphas[minpos]) |

2.3357214690901213

由上图可知超参alphas取2.34时错误率最低。

**(3) 非线性回归**

我们选择随机森林（Random Forest）。

In [29]:

In [30]:

|  |
| --- |
| from sklearn.ensemble import RandomForestRegressor |

随机森林由多个超参，我们通过GridSearchCV进行网格搜索。

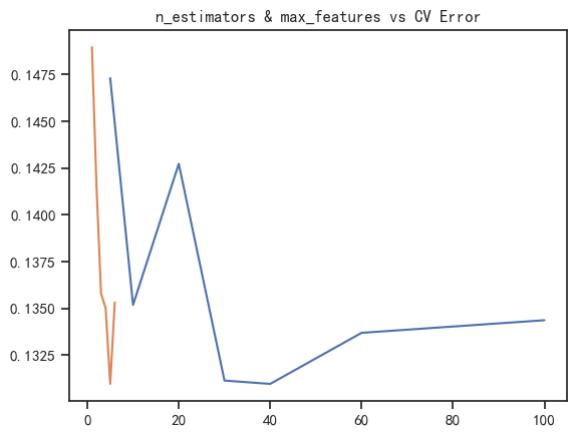
|  |
| --- |
| from sklearn.model\_selection import GridSearchCV  parameters = [  {'n\_estimators': [5, 10, 20, 30, 40, 60, 100], 'max\_features': [1, 2, 3, 4, 5, 6]}  ]  rf\_es = RandomForestRegressor(random\_state=20)  grid\_search = GridSearchCV(rf\_es, parameters, cv=5,  scoring='neg\_mean\_squared\_error',  return\_train\_score=True)  grid\_search.fit(X\_train, y\_train)  grid\_search.best\_params\_ |

Out[30]: {'max\_features': 5, 'n\_estimators': 40}

由结果可知，超参n\_estimators=40, max\_features=5时错误率最低。画图确认下。

|  |
| --- |
| n\_estimators = [5, 10, 20, 30, 40, 60, 100]  In [31]:  test\_scores = []  for n in n\_estimators:  rf\_es = RandomForestRegressor(n\_estimators=n, max\_features=5, random\_state=20)  test\_score = np.sqrt(-cross\_val\_score (rf\_es, X\_train, y\_train, cv=3, scoring='neg\_mean\_squared\_error')) test\_scores.append(np.mean (test\_score))  plt.plot(n\_estimators, test\_scores)  plt.title("n\_estimator vs CV Error")  max\_features = [1, 2, 3, 4, 5, 6]  test\_scores = []  for m in max\_features:  rf\_es = RandomForestRegressor(n\_estimators=40, max\_features=m, random\_state=20)  test\_score = np.sqrt(-cross\_val\_score (rf\_es, X\_train, y\_train, cv=3, scoring='neg\_mean\_squared\_error')) test\_scores.append(np.mean (test\_score))  plt.plot(max\_features, test\_scores)  plt.title("n\_estimators & max\_features vs CV Error") |

Out[31]: Text(0.5, 1.0, 'n\_estimators & max\_features vs CV Error')



从上图可知，通过GridSearchCV进行网格搜索的确可以获得最有的超参。

**(4) 集成学习**

通过集成两个模型的预测值取平均值，以提高泛化能力。

ridge = Ridge(alpha=2.34)

In [32]:

In [33]:

rf = RandomForestRegressor(n\_estimators=40, max\_features=5)

ridge.fit(X\_train, y\_train)

rf.fit(X\_train, y\_train)

Out[32]: RandomForestRegressor(max\_features=5, n\_estimators=40)

验证测试样本：

|  |
| --- |
| X\_test = revised\_test\_df.values  train\_score\_ridge = ridge.score (X\_train, y\_train)  test\_score\_ridge = ridge.score (X\_test, y\_test)  print(f"岭回归训练集预测分数{train\_score\_ridge}，测试集预测分数{test\_score\_ridge}")  train\_score\_rf = rf.score (X\_train, y\_train)  test\_score\_rf = rf.score (X\_test, y\_test)  print(f'随机森林训练集预测分数{train\_score\_rf}，测试集预测分数{test\_score\_rf}')  ridge\_predict = ridge.predict(X\_test)  rf\_predict = rf.predict(X\_test)  y\_ridge = np.expm1(ridge\_predict)  y\_rf = np.expm1(rf\_predict) |

岭回归训练集预测分数0.940256241614267，测试集预测分数0.9216924553679973

随机森林训练集预测分数0.988094553782405，测试集预测分数0.9845917677390525

使用两模型预测值的均值作为最终预测值，并计算R^2分值：

y\_final = (y\_ridge+y\_rf)/2

In [34]:

y\_test\_ori = np.expm1(y\_test.values)

from sklearn.metrics import mean\_squared\_error, r2\_score

print("根均方误差(RMSE)：{}".format(np.sqrt(mean\_squared\_error (y\_test\_ori, y\_final))))

print("R^2 score：{}".format(r2\_score (y\_test\_ori, y\_final)))

根均方误差(RMSE)：9192.753877462712

R^2 score：0.9645709679639592

**(5) 比对预测结果**

In [35]:

构建验证集：

|  |
| --- |
| df\_final = df\_curated.copy ()  verify\_df = raw\_train\_df.copy ()  verify\_df.drop(columns= ["label"], inplace=True)  X\_verify = verify\_df.values  ridge\_predict = ridge.predict(X\_verify)  rf\_predict = rf.predict(X\_verify)  y\_ridge = np.expm1(ridge\_predict)  y\_rf = np.expm1(rf\_predict)  y\_final = (y\_ridge+y\_rf)/2 |

计算预测差额及百分比：

In [36]: df\_final['预测义务教育经费（万元）'] = y\_final.tolist()

df\_final['预测差额（万元）'] = df\_final['义务教育经费（万元）'] - df\_final['预测义务教育经费（万元）']

df\_final['预测差额百分比'] = df\_final['预测差额（万元）'] / df\_final['义务教育经费（万元）'] \* 100

df\_final.head()

Out[36]:

**区 县**

**年份**

**常住人**

**口（万**

**人）**

**出生**

**率**

**(‰)**

**GDP**

**（亿**

**元）**

**在园**

**幼儿**

**（万**

**人）**

**小学**

**初中**

**生 （万 人）**

**生**

**（万**

**人）**

**小学教育经费 （万元）**

**初中教育经费 （万元）**

**义务教育经费 （万元）**

**预测义务教育经 费（万元）**

**预测差额（万 元）**

**预测差额百**

**分比**

宝

**0** 山 2021 225.01 4.43 1725.56 5.35 7.59 1.91 248979.0204 80959.3610 329938.3814 326810.930101 3127.451299 0.947890

区

松

**1** 江 2021 193.88 5.59 1782.28 5.10 7.22 3.33 211574.7356 149880.5361 361455.2717 371949.086858 -10493.815158 -2.903213

区

奉

**2** 贤 2021 114.71 4.14 1300.00 2.54 3.66 1.92 123575.5812 89196.3840 212771.9652 216687.684054 -3915.718854 -1.840336

区

徐

**3** 汇 2020 111.31 5.17 2176.73 2.49 4.78 2.97 147392.4472 142103.5110 289495.9582 283671.085549 5824.872651 2.012074

区

长

**4** 宁 2020 69.31 4.69 1561.17 1.25 2.29 1.34 74358.4984 70785.9690 145144.4674 161499.934337 -16355.466937 -11.268405

区

**(5) 各区各年份义务教育经费多寡走势分析**

year\_min = df\_final['年份'].min()

In [37]:

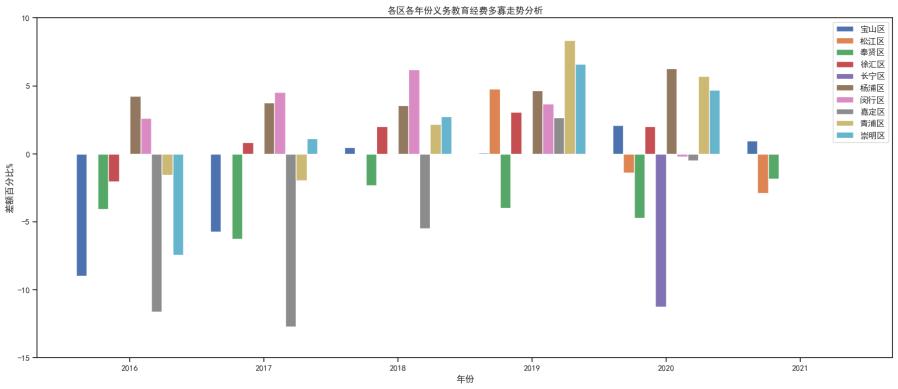
year\_max = df\_final['年份'].max ()

years = year\_max - year\_min + 1

districts = df\_final['区县'].unique()

districts\_count = len(districts)

|  |
| --- |
| total\_width = 0.8  In [38]:  width = total\_width / districts\_count  x = np.arange (years)  x = x - (total\_width - width) / 2 + year\_min  fig, ax = plt.subplots(figsize= (20,8))  ax.set(ylabel='差额百分比%',xlabel='年份', title='各区各年份义务教育经费多寡走势分析', ylim= (-15, 10))  for idx, dist in enumerate(districts):  dist\_values = [0] \* years  for y\_idx in range (years):  year = year\_min + y\_idx  row = df\_final[(df\_final['区县'] == dist) & (df\_final['年份'] == year)]['预测差额百分比'].values dist\_values[y\_idx] = row[0] if len(row) > 0 else 0  bar\_container = ax.bar(x + width \* idx, dist\_values, width=width, label=dist)  plt.legend()  plt.show() |



**分析结论**

1. 杨浦、闵行的义务教育经费持续较高。嘉定、奉贤经费持续较低。徐汇、青浦、崇明经费近年有所提升。松江经费近年有所降低。 2. 中心城区的数据偏少。最近两年的数据缺失。可以进一步完善。

**参考链接**

皮尔逊积矩相关系数： <https://zh.wikipedia.org/zh>-

hans/%E7%9A%AE%E5%B0%94%E9%80%8A%E7%A7%AF%E7%9F%A9%E7%9B%B8%E5%85%B3%E7%B3%BB%E6%95%B0 (<https://zh.wikipedia.org/zh>-

hans/%E7%9A%AE%E5%B0%94%E9%80%8A%E7%A7%AF%E7%9F%A9%E7%9B%B8%E5%85%B3%E7%B3%BB%E6%95%B0)

z-score标准化分数： <https://zh.wikipedia.org/zh-hans/%E6%A8%99%E6%BA%96%E5%88%86%E6%95%B8> (<https://zh.wikipedia.org/zh>- hans/%E6%A8%99%E6%BA%96%E5%88%86%E6%95%B8)

numpy.logspace: https://numpy.org/doc/stable/reference/generated/numpy.logspace.html

(<https://numpy.org/doc/stable/reference/generated/numpy.logspace.html>)

cross\_val\_score: <https://scikit-learn.org/stable/modules/generated/sklearn.model>\_selection.cross\_val\_score.html (https://scikit- learn.org/stable/modules/generated/sklearn.model\_selection.cross\_val\_score.html)

GridSearchCV: <https://scikit-learn.org/stable/modules/generated/sklearn.model>\_selection.GridSearchCV.html (https://scikit- learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html)

Ridge： <https://scikit-learn.org/stable/modules/generated/sklearn.linear>\_model.Ridge.html (https://scikit-

learn.org/stable/modules/generated/sklearn.linear\_model.Ridge.html)

RandomForestRegressor ：<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor>.html (https://scikit- learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html)

R2\_score： <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2>\_score.html (https://scikit-

learn.org/stable/modules/generated/sklearn.metrics.r2\_score.html)

In [ ]: