本项目使用 XGBoost 机器学习模型，探讨国企与民营企业在估值上的差异及其影响因素。以市净率（PB）作为估值指标，考察企业性质（EquityNature）、资本支出（Capexp）、第一大股东持股比例（LargestHolderRate）、前十股东持股比例（TopTenHoldersRate），以及公司规模（size）、固定资产占比（ppe_ratio）、无形资产占比（intangible_ratio）、资产负债率（lev）、毛利率（Gross_Margin）和公司年龄（age）等因素对估值的影响，旨在识别国企与民营企业在市场定价中的表现差异。

环境检查

In [1]:
```
! python --version
! pip list | findstr "pandas matplotlib scikit-learn"
```

```
Python 3.13.2
matplotlib              3.10.3
matplotlib-inline       0.1.7
pandas                  2.3.0
scikit-learn            1.7.0
```

In [2]:
```
import numpy as np
import pandas as pd
import xgboost as xgb
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split, cross_val_score, GridSearc
from bayes_opt import BayesianOptimization
from IPython.core.interactiveshell import InteractiveShell
from sklearn.metrics import (
    median_absolute_error,
    mean_absolute_error,
    mean_squared_error,
    r2_score,
)
import pandas as pd
from pprint import pprint
import shap
import numpy as np
```

```
c:\Users\Lenovo\AppData\Local\Programs\Python\Python313\Lib\site-packages\tqdm\au
to.py:21: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets.
See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

In [3]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

目 录

数据整理

导入数据

X的相关数据

In [72]:
```
df_cas=pd.read_excel("rawdata\X\FS_Combas.xlsx",skiprows=[1,2],header=0)
df_far=pd.read_excel("rawdata\X\FAR_Finidx.xlsx",skiprows=[1,2],header=0)
df_en=pd.read_excel("rawdata\X\EN_EquityNatureAll.xlsx",skiprows=[1,2],header=0)
df_stk=pd.read_excel("rawdata\X\STK_LISTEDCOINFOANL.xlsx",skiprows=[1,2],header=
df_T10=pd.read_excel("rawdata\Y\FI_T10.xlsx",skiprows=[1,2],header=0)

df_cas.head()
df_far.head()
df_en.head()
df_stk.head()
df_T10.head()
```

```
<>:1: SyntaxWarning: invalid escape sequence '\X'
<>:2: SyntaxWarning: invalid escape sequence '\X'
<>:3: SyntaxWarning: invalid escape sequence '\X'
<>:4: SyntaxWarning: invalid escape sequence '\X'
<>:5: SyntaxWarning: invalid escape sequence '\Y'
<>:1: SyntaxWarning: invalid escape sequence '\X'
<>:2: SyntaxWarning: invalid escape sequence '\X'
<>:3: SyntaxWarning: invalid escape sequence '\X'
<>:4: SyntaxWarning: invalid escape sequence '\X'
<>:5: SyntaxWarning: invalid escape sequence '\Y'
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\2676626486.py:1: SyntaxWarnin
g: invalid escape sequence '\X'
  df_cas=pd.read_excel("rawdata\X\FS_Combas.xlsx",skiprows=[1,2],header=0)
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\2676626486.py:2: SyntaxWarnin
g: invalid escape sequence '\X'
  df_far=pd.read_excel("rawdata\X\FAR_Finidx.xlsx",skiprows=[1,2],header=0)
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\2676626486.py:3: SyntaxWarnin
g: invalid escape sequence '\X'
  df_en=pd.read_excel("rawdata\X\EN_EquityNatureAll.xlsx",skiprows=[1,2],header=
0)
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\2676626486.py:4: SyntaxWarnin
g: invalid escape sequence '\X'
  df_stk=pd.read_excel("rawdata\X\STK_LISTEDCOINFOANL.xlsx",skiprows=[1,2],header
=0)
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\2676626486.py:5: SyntaxWarnin
g: invalid escape sequence '\Y'
  df_T10=pd.read_excel("rawdata\Y\FI_T10.xlsx",skiprows=[1,2],header=0)
c:\Users\Lenovo\AppData\Local\Programs\Python\Python313\Lib\site-packages\openpyx
l\styles\stylesheet.py:237: UserWarning: Workbook contains no default style, appl
y openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
c:\Users\Lenovo\AppData\Local\Programs\Python\Python313\Lib\site-packages\openpyx
l\styles\stylesheet.py:237: UserWarning: Workbook contains no default style, appl
y openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
c:\Users\Lenovo\AppData\Local\Programs\Python\Python313\Lib\site-packages\openpyx
l\styles\stylesheet.py:237: UserWarning: Workbook contains no default style, appl
y openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
c:\Users\Lenovo\AppData\Local\Programs\Python\Python313\Lib\site-packages\openpyx
l\styles\stylesheet.py:237: UserWarning: Workbook contains no default style, appl
y openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
c:\Users\Lenovo\AppData\Local\Programs\Python\Python313\Lib\site-packages\openpyx
l\styles\stylesheet.py:237: UserWarning: Workbook contains no default style, appl
y openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
```

Out[72]:

	Stkcd	ShortName	Accper	Typrep	A001212000	A001218000	A001000000
0	2	万科A	2007-12-31	A	5.752056e+08	0.000000e+00	1.000945e+11
1	2	万科A	2008-12-31	A	1.265333e+09	0.000000e+00	1.192366e+11
2	2	万科A	2009-12-31	A	1.355977e+09	8.196633e+07	1.376086e+11
3	2	万科A	2010-12-31	A	1.219582e+09	3.739519e+08	2.156376e+11
4	2	万科A	2011-12-31	A	1.595863e+09	4.354743e+08	2.962084e+11

Out[72]:

	Stkcd	Accper	T30100	T40100	Capexp
0	2	2007-12-31	0.661125	0.419946	2.578978e+08
1	2	2008-12-31	0.674441	0.389993	2.152837e+08
2	2	2009-12-31	0.670017	0.293903	8.060622e+08
3	2	2010-12-31	0.746861	0.406996	2.619386e+08
4	2	2011-12-31	0.770997	0.397792	2.615609e+08

Out[72]:

	Symbol	ShortName	EndDate	LargestHolderRate	TopTenHoldersRate	EquityNa
0	2	万科A	2007-12-31	14.63	22.71	
1	2	万科A	2008-12-31	14.73	22.01	
2	2	万科A	2009-12-31	14.73	22.91	
3	2	万科A	2010-12-31	14.73	22.77	
4	2	万科A	2011-12-31	14.73	23.15	

Out[72]:

	Symbol	ShortName	EndDate	EstablishDate
0	2	万科A	2007-12-31	1984-05-30
1	2	万科A	2008-12-31	1984-05-30
2	2	万科A	2009-12-31	1984-05-30
3	2	万科A	2010-12-31	1984-05-30
4	2	万科A	2011-12-31	1984-05-30

Out[72]:

	Stkcd	ShortName	Accper	Source	Indcd1	Indnme1	F100101B	F100401A
0	2	万科A	2007-12-31	0	K70	房地产业	37.271017	5.842908
1	2	万科A	2008-12-31	0	K70	房地产业	15.284721	1.826939
2	2	万科A	2009-12-31	0	K70	房地产业	18.484927	2.617532
3	2	万科A	2010-12-31	0	K70	房地产业	10.224503	1.655741
4	2	万科A	2011-12-31	0	K70	房地产业	7.080777	1.210838

数据合并

Y

In [73]:
```
df_T10["year"]=df_T10["Accper"].str.slice(0,4).astype("int64")

df_T10 = df_T10.rename(columns={"F100101B": "pe"})
df_T10 = df_T10.rename(columns={"F100401A": "pb"})
```

X

In [74]:
```
df_11=pd.merge(df_cas,df_far,how="outer",on=["Stkcd","Accper"])
df_22=pd.merge(df_en,df_stk,how="outer",on=["Symbol","EndDate"])
```

In [75]:
```
df_11.columns
```

Out[75]:
```
Index(['Stkcd', 'ShortName', 'Accper', 'Typrep', 'A001212000', 'A001218000',
       'A001000000', 'A002101000', 'A002201000', 'A002000000', 'T30100',
       'T40100', 'Capexp'],
      dtype='object')
```

In [76]:
```
df_11["year"]=df_11["Accper"].str.slice(0,4).astype("int64")
df_1=df_11[['Stkcd', 'ShortName', 'year',
            'A001212000', 'A001218000','A001000000',
            'A002101000', 'A002201000', 'A002000000',
            'T30100','T40100', 'Capexp']]
```

In [77]:
```
df_22.columns
```

Out[77]:
```
Index(['Symbol', 'ShortName_x', 'EndDate', 'LargestHolderRate',
       'TopTenHoldersRate', 'EquityNature', 'EquityNatureID', 'Seperation',
       'ShortName_y', 'EstablishDate'],
      dtype='object')
```

In [78]:
```
df_22["year"]=df_22["EndDate"].str.slice(0,4).astype("int64")
df_22 = df_22.rename(columns={"Symbol": "Stkcd"})
df_2=df_22[['Stkcd', 'year',
            'LargestHolderRate','TopTenHoldersRate', 'EquityNature',
            'EquityNatureID', 'Seperation', 'EstablishDate']]
```

In [79]:
```
df=pd.merge(df_1,df_2,how="outer",on=["Stkcd","year"])
df=pd.merge(df,df_T10,how="outer",on=["Stkcd","year"])
```

In [80]:
```
df.head()
```

Out[80]:

	Stkcd	ShortName_x	year	A001212000	A001218000	A001000000	A0021
0	2	万科A	2007	5.752056e+08	0.000000e+00	1.000945e+11	1.10485
1	2	万科A	2008	1.265333e+09	0.000000e+00	1.192366e+11	4.60196
2	2	万科A	2009	1.355977e+09	8.196633e+07	1.376086e+11	1.18825
3	2	万科A	2010	1.219582e+09	3.739519e+08	2.156376e+11	1.47800
4	2	万科A	2011	1.595863e+09	4.354743e+08	2.962084e+11	1.72444

5 rows × 25 columns

数据清洗

In [81]:
```
df.columns
```

Out[81]:
```
Index(['Stkcd', 'ShortName_x', 'year', 'A001212000', 'A001218000',
       'A001000000', 'A002101000', 'A002201000', 'A002000000', 'T30100',
       'T40100', 'Capexp', 'LargestHolderRate', 'TopTenHoldersRate',
       'EquityNature', 'EquityNatureID', 'Seperation', 'EstablishDate',
       'ShortName_y', 'Accper', 'Source', 'Indcd1', 'Indnme1', 'pe', 'pb'],
      dtype='object')
```

In [82]:
```
df.shape
```

Out[82]:
```
(55194, 25)
```

In [83]:
```
df.isnull().sum()
```

Out[83]:
```
Stkcd                  0
ShortName_x            1
year                   0
A001212000             4
A001218000           237
A001000000             1
A002101000          8095
A002201000         17077
A002000000             1
T30100                30
T40100                71
Capexp                58
LargestHolderRate   2204
TopTenHoldersRate   2204
EquityNature        3269
EquityNatureID      3269
Seperation          4599
EstablishDate        822
ShortName_y         3169
Accper              3169
Source              3169
Indcd1              3169
Indnme1             3169
pe                 10185
pb                  3488
dtype: int64
```

In [84]:
```
# 剔除资产总计缺失的样本
df=df.dropna(axis=0,subset="A001000000")

# A002101000 [短期借款]、A002201000 [长期借款]、Seperation[两权分离率(%)]缺失较多
df.drop(["A002101000"],axis=1,inplace=True)
df.drop(["A002201000"],axis=1,inplace=True)
df.drop(["Seperation"],axis=1,inplace=True)
```

In [85]:
```
df.head()
```

Out[85]:

	Stkcd	ShortName_x	year	A001212000	A001218000	A001000000	A0020
0	2	万科A	2007	5.752056e+08	0.000000e+00	1.000945e+11	6.61749
1	2	万科A	2008	1.265333e+09	0.000000e+00	1.192366e+11	8.04180
2	2	万科A	2009	1.355977e+09	8.196633e+07	1.376086e+11	9.22000
3	2	万科A	2010	1.219582e+09	3.739519e+08	2.156376e+11	1.61051
4	2	万科A	2011	1.595863e+09	4.354743e+08	2.962084e+11	2.28375

5 rows × 22 columns

In [86]:
```
# 只保留国企和民营
df = df[(df['EquityNature'] == '国企') | (df['EquityNature'] == '民营')]
```

In [87]:
```
df.shape

df.isnull().sum()
```

Out[87]:
```
(48235, 22)
```

Out[87]:
```
Stkcd                  0
ShortName_x            0
year                   0
A001212000             1
A001218000           174
A001000000             0
A002000000             0
T30100                 2
T40100                32
Capexp                42
LargestHolderRate      1
TopTenHoldersRate      1
EquityNature           0
EquityNatureID         0
EstablishDate          0
ShortName_y         1429
Accper              1429
Source              1429
Indcd1              1429
Indnme1             1429
pe                  7533
pb                  1714
dtype: int64
```

In [88]:
```
df=df.dropna()
```

生成变量

In [89]:
```
df.columns
```

Out[89]:
```
Index(['Stkcd', 'ShortName_x', 'year', 'A001212000', 'A001218000',
       'A001000000', 'A002000000', 'T30100', 'T40100', 'Capexp',
       'LargestHolderRate', 'TopTenHoldersRate', 'EquityNature',
       'EquityNatureID', 'EstablishDate', 'ShortName_y', 'Accper', 'Source',
       'Indcd1', 'Indnme1', 'pe', 'pb'],
      dtype='object')
```

In [90]:
```
# 公司规模
df["size"] =np.log(df["A001000000"])
# 固定资产净额占比
df['ppe_ratio'] =df["A001212000"]/df["A001000000"]*100
# 无形资产净额占比
df['intangible_ratio'] =df["A001218000"]/df["A001000000"]*100
# 资产负债率
df['lev'] = (df['A002000000'] / df['A001000000'])*100
# 营业毛利率
df['Gross_Margin'] = df['T40100']
# 公司年龄
df["Establish_year"]=df["EstablishDate"].str.slice(0,4).astype("int64")
df["age"] = df["year"] - df["Establish_year"]
```

In [91]:
```
df.columns
```

Out[91]:
```
Index(['Stkcd', 'ShortName_x', 'year', 'A001212000', 'A001218000',
       'A001000000', 'A002000000', 'T30100', 'T40100', 'Capexp',
       'LargestHolderRate', 'TopTenHoldersRate', 'EquityNature',
       'EquityNatureID', 'EstablishDate', 'ShortName_y', 'Accper', 'Source',
       'Indcd1', 'Indnme1', 'pe', 'pb', 'size', 'ppe_ratio',
       'intangible_ratio', 'lev', 'Gross_Margin', 'Establish_year', 'age'],
      dtype='object')
```

In [92]:
```
df_clean=df[['Stkcd', 'year','Indcd1', 'Indnme1', 'EquityNature',
             'Capexp', 'LargestHolderRate','TopTenHoldersRate',
             'size', 'ppe_ratio', 'intangible_ratio',
              'lev', 'Gross_Margin','age','pe','pb']]
```

In [93]:
```
df_soe=df_clean[df_clean['EquityNature'] == "国企"]
df_po=df_clean[df_clean['EquityNature'] == "民营"]
```

In [94]:
```
df_soe.shape
df_po.shape
```

Out[94]: (15687, 16)

Out[94]: (24827, 16)

为后续估值比较，现对国企，民营样本随机抽取，保证样本数量一致

In [95]:
```
# 设置随机种子以确保结果可复现
np.random.seed(42)

# 从国企数据集中随机抽取13000个样本
df_soe_sample = df_soe.sample(n=13000, random_state=42)

# 从民企数据集中随机抽取13000个样本
df_po_sample = df_po.sample(n=13000, random_state=42)

# 检查新样本的形状
print(df_soe_sample.shape)
print(df_po_sample.shape)
```

```
(13000, 16)
(13000, 16)
```

数据导出

In [96]:
```
df_clean.to_excel("data/df_clean.xlsx",index=False)
df_soe.to_excel("data/df_soe.xlsx",index=False)
df_po.to_excel("data/df_po.xlsx",index=False)
df_soe_sample.to_excel("data/df_soe_sample.xlsx",index=False)
df_po_sample.to_excel("data/df_po_sample.xlsx",index=False)
```

描述性统计

In [97]:
```
df_soe = pd.read_excel('data\df_soe.xlsx')
df_po = pd.read_excel('data\df_po.xlsx')
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:2: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:2: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\2745604325.py:1: SyntaxWarnin
g: invalid escape sequence '\d'
  df_soe = pd.read_excel('data\df_soe.xlsx')
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\2745604325.py:2: SyntaxWarnin
g: invalid escape sequence '\d'
  df_po = pd.read_excel('data\df_po.xlsx')
```

In [98]:
```
columns_of_interest = ['pb','Capexp', 'LargestHolderRate', 'TopTenHoldersRate',
                       'intangible_ratio', 'lev', 'Gross_Margin', 'age',]
```

In [99]:
```
descriptive_soe = df_soe[columns_of_interest].describe().transpose().round(4)
descriptive_soe.insert(0, 'variable', columns_of_interest)
descriptive_soe

descriptive_po = df_po[columns_of_interest].describe().transpose().round(4)
descriptive_po.insert(0, 'variable', columns_of_interest)
descriptive_po
```

Out[99]:

	variable	count	mean	std	n
pb	pb	15687.0	3.065000e+00	4.151700e+00	0.10
Capexp	Capexp	15687.0	1.845876e+09	1.165562e+10	0.00
LargestHolderRate	LargestHolderRate	15687.0	3.962740e+01	1.548590e+01	3.62
TopTenHoldersRate	TopTenHoldersRate	15687.0	5.879050e+01	1.593320e+01	12.72
size	size	15687.0	2.280080e+01	1.490100e+00	17.66
ppe_ratio	ppe_ratio	15687.0	2.526530e+01	1.927640e+01	0.00
intangible_ratio	intangible_ratio	15687.0	5.388000e+00	8.379400e+00	0.00
lev	lev	15687.0	4.866140e+01	1.960780e+01	1.02
Gross_Margin	Gross_Margin	15687.0	2.546000e-01	1.604000e-01	-0.51
age	age	15687.0	1.948660e+01	7.161900e+00	0.00

Out[99]:

	variable	count	mean	std	m
pb	pb	24827.0	4.204600e+00	1.706560e+01	0.00
Capexp	Capexp	24827.0	3.870119e+08	1.925693e+09	0.00
LargestHolderRate	LargestHolderRate	24827.0	3.240710e+01	1.362090e+01	2.19
TopTenHoldersRate	TopTenHoldersRate	24827.0	6.030540e+01	1.472100e+01	3.58
size	size	24827.0	2.179430e+01	1.088900e+00	15.97
ppe_ratio	ppe_ratio	24827.0	1.864590e+01	1.295060e+01	0.00
intangible_ratio	intangible_ratio	24827.0	4.258700e+00	4.906400e+00	0.00
lev	lev	24827.0	3.557960e+01	1.870740e+01	0.70
Gross_Margin	Gross_Margin	24827.0	3.263000e-01	1.785000e-01	-1.21
age	age	24827.0	1.780180e+01	6.266200e+00	1.00

In [100…

```
descriptive_soe.to_excel("data/descriptive_soe.xlsx",index=False)
descriptive_po.to_excel("data/descriptive_po.xlsx",index=False)
```

构建模型

国民估值差异

In [101…

```
df_soe_sample = pd.read_excel('data\df_soe_sample.xlsx')
df_po_sample = pd.read_excel('data\df_po_sample.xlsx')

df_soe_sample.head()
df_po_sample.head()
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:2: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:2: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\3059238796.py:1: SyntaxWarnin
g: invalid escape sequence '\d'
  df_soe_sample = pd.read_excel('data\df_soe_sample.xlsx')
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\3059238796.py:2: SyntaxWarnin
g: invalid escape sequence '\d'
  df_po_sample = pd.read_excel('data\df_po_sample.xlsx')
```

Out[101…

	Stkcd	year	Indcd1	Indnme1	EquityNature	Capexp	LargestHolderRat
0	603123	2013	F52	零售业	国企	1.056187e+08	55.0
1	2125	2024	C26	化学原料和化学制品制造业	国企	1.199149e+08	29.5
2	600389	2020	C26	化学原料和化学制品制造业	国企	1.991432e+08	29.3
3	2302	2015	C30	非金属矿物制品业	国企	2.205167e+08	36.1
4	600861	2013	F52	零售业	国企	2.507833e+06	33.4

Out[101…

	Stkcd	year	Indcd1	Indnme1	EquityNature	Capexp	LargestHolderRat
0	2446	2020	C39	计算机、通信和其他电子设备制造业	民营	7.707588e+07	9.8
1	300067	2017	C26	化学原料和化学制品制造业	民营	4.317553e+07	38.1
2	2815	2023	C39	计算机、通信和其他电子设备制造业	民营	1.359295e+09	45.6
3	603057	2022	C13	农副食品加工业	民营	1.072157e+08	24.7
4	688622	2021	C40	仪器仪表制造业	民营	1.321843e+08	20.8

In [102…

```
def calculate_stats_pb(df):
    # 只计算 pb 的描述性统计信息
    stats_pb = df['pb'].describe(percentiles=[0.2, 0.4, 0.6, 0.8]).round(3)
    return stats_pb

# 国企 pb 统计
stats_pb_soe = calculate_stats_pb(df_soe_sample)
# 民营 pb 统计
stats_pb_po = calculate_stats_pb(df_po_sample)

# 将结果转换为 DataFrame
stats_soe = pd.DataFrame({'soe_pb': stats_pb_soe})
stats_po = pd.DataFrame({'po_pb': stats_pb_po})

# 合并两个 DataFrame
result_stats = pd.concat([stats_soe, stats_po], axis=1)

# 计算差值并新建一列
result_stats['diff'] = result_stats['soe_pb'] - result_stats['po_pb']
result_stats['indicator'] = result_stats['diff'].apply(lambda x: 1 if x > 0 else
```

```
print(result_stats)
```

```
          soe_pb      po_pb       diff  indicator
count  13000.000  13000.000      0.000         -1
mean       3.068      4.193     -1.125         -1
std        4.142     18.941    -14.799         -1
min        0.108      0.000      0.108          1
20%        1.198      1.798     -0.600         -1
40%        1.799      2.540     -0.741         -1
50%        2.160      2.946     -0.786         -1
60%        2.612      3.459     -0.847         -1
80%        4.107      5.170     -1.063         -1
max      244.631   2000.870  -1756.239         -1
```

性能函数

In [103…
```
def myscoring(y_true, y_pred):
    print("-" * 10, "\n")
    print("median_absolute_error:", median_absolute_error(y_true, y_pred))
    print("mean_absolute_error:", mean_absolute_error(y_true, y_pred))
    print("mean_squared_error:", mean_squared_error(y_true, y_pred))
    print("r2_score:", r2_score(y_true, y_pred))
```

XGBoost_pb

国企

In [36]:
```
df_soe = pd.read_excel('data\df_soe.xlsx')
df_soe.head()
df_soe.columns
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\8676071.py:1: SyntaxWarning: i
nvalid escape sequence '\d'
  df_soe = pd.read_excel('data\df_soe.xlsx')
```

Out[36]:

	Stkcd	year	Indcd1	Indnme1	EquityNature	Capexp	LargestHolderRate
0	2	2007	K70	房地产业	国企	2.578978e+08	14.63
1	2	2008	K70	房地产业	国企	2.152837e+08	14.73
2	2	2009	K70	房地产业	国企	8.060622e+08	14.73
3	2	2010	K70	房地产业	国企	2.619386e+08	14.73
4	2	2011	K70	房地产业	国企	2.615609e+08	14.73

Out[36]:
```
Index(['Stkcd', 'year', 'Indcd1', 'Indnme1', 'EquityNature', 'Capexp',
       'LargestHolderRate', 'TopTenHoldersRate', 'size', 'ppe_ratio',
       'intangible_ratio', 'lev', 'Gross_Margin', 'age', 'pe', 'pb'],
      dtype='object')
```

In [37]:
```
# 定义特征列和目标列名
features = ['Capexp','LargestHolderRate', 'TopTenHoldersRate', 'size',
```

```
            'ppe_ratio','intangible_ratio', 'lev', 'Gross_Margin', 'age']

# 获取特征数据（X）和目标变量数据（y）
X = df_soe[features]

y_pb = df_soe['pb']
```

In [38]:
```
from sklearn.model_selection import train_test_split

X_train_pb, X_test_pb, y_train_pb, y_test_pb = train_test_split(
    X, y_pb, test_size=0.3, random_state=42
)

print("PB 模型数据：")
print("X_train_pb shape:", X_train_pb.shape)
print("y_train_pb shape:", y_train_pb.shape)
print("X_test_pb shape:", X_test_pb.shape)
print("y_test_pb shape:", y_test_pb.shape)
```

```
PB 模型数据：
X_train_pb shape: (10980, 9)
y_train_pb shape: (10980,)
X_test_pb shape: (4707, 9)
y_test_pb shape: (4707,)
```

In [39]:
```
xgb_model_pb = xgb.XGBRegressor(objective="reg:squarederror", random_state=42)

xgb_model_pb.fit(X_train_pb, y_train_pb)

y_train_pred_pb = xgb_model_pb.predict(X_train_pb)
y_test_pred_pb = xgb_model_pb.predict(X_test_pb)


print("训练集评估：")
myscoring(y_train_pb, y_train_pred_pb)

print("测试集评估：")
myscoring(y_test_pb, y_test_pred_pb)
```

Out[39]:

▾ XGBRegressor

▶ Parameters

```
训练集评估：
----------

median_absolute_error: 0.49084503043746963
mean_absolute_error: 0.6914826948145606
mean_squared_error: 1.027504719691468
r2_score: 0.9416129741351332
测试集评估：
----------

median_absolute_error: 0.7982378918151856
mean_absolute_error: 1.3970954951954995
mean_squared_error: 11.356426512093265
r2_score: 0.306671750863294
```

调参

In [43]:

```python
def function_xgbreg(
    n_estimators,
    learning_rate,
    max_depth,
    min_child_weight,
    gamma,
    subsample,
    colsample_bytree,
    reg_alpha,      # 新增正则化参数
    reg_lambda      # 新增正则化参数
):
    xgbreg = XGBRegressor(
        objective="reg:squarederror",
        booster="gbtree",
        n_estimators=int(n_estimators),
        learning_rate=learning_rate,
        max_depth=int(max_depth),
        min_child_weight=min_child_weight,
        gamma=gamma,
        subsample=subsample,
        colsample_bytree=colsample_bytree,
        reg_alpha=reg_alpha,      # 添加正则化参数
        reg_lambda=reg_lambda     # 添加正则化参数
    )
    cvs = cross_val_score(estimator=xgbreg, X=X_train_pb, y=y_train_pb, scoring=
    return cvs.mean()

# 参数搜索空间调整（重点优化过拟合相关参数）
pds = {
    "n_estimators": (500, 2000),    # 扩大迭代次数范围
    "learning_rate": (0.001, 0.1),  # 降低学习率上限
    "max_depth": (3, 10),           # 降低树深度上限
    "min_child_weight": (1, 5),     # 降低最小样本权重上限
    "gamma": (1, 5),                # 提高分裂阈值下限
    "subsample": (0.5, 1),          # 限制子样本比例
    "colsample_bytree": (0.5, 1),   # 限制特征采样比例
    "reg_alpha": (0, 10),           # 添加L1正则化
    "reg_lambda": (0, 10)           # 添加L2正则化
}

optimizer = BayesianOptimization(function_xgbreg, pds, random_state=0)
optimizer.maximize(init_points=20, n_iter=50)
xgbpds = optimizer.max["params"]

# 最终模型构建（完整参数应用）
xgbreg_pb_gq = XGBRegressor(
    objective="reg:squarederror",
    booster="gbtree",
    n_estimators=int(xgbpds["n_estimators"]),
    learning_rate=xgbpds["learning_rate"],
    max_depth=int(xgbpds["max_depth"]),
    min_child_weight=xgbpds["min_child_weight"],
    gamma=xgbpds["gamma"],
    subsample=xgbpds["subsample"],
    colsample_bytree=xgbpds["colsample_bytree"],
    reg_alpha=xgbpds["reg_alpha"],      # 应用优化后的正则化参数
    reg_lambda=xgbpds["reg_lambda"]     # 应用优化后的正则化参数
)
```

```
xgbreg_pb_gq.fit(X_train_pb, y_train_pb)

y_train_pred_pb = xgbreg_pb_gq.predict(X_train_pb)
y_test_pred_pb = xgbreg_pb_gq.predict(X_test_pb)

myscoring(y_train_pb, y_train_pred_pb)
myscoring(y_test_pb, y_test_pred_pb)
```

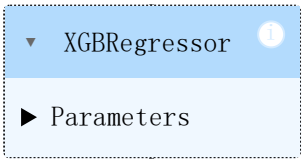
| iter | target | colsam... | gamma | learni... | max_depth | min_c
h... | n_esti... | reg_alpha | reg_la... | subsample |

| 1 | 0.361 | 0.7744 | 3.861 | 0.06067 | 6.814 | 2.695
| 1.469e+03 | 4.376 | 8.918 | 0.9818 |
| 2 | 0.388 | 0.6917 | 4.167 | 0.05336 | 6.976 | 4.702
| 606.6 | 0.8713 | 0.2022 | 0.9163 |
| 3 | 0.3714 | 0.8891 | 4.48 | 0.09788 | 8.594 | 2.846
| 1.671e+03 | 1.183 | 6.399 | 0.5717 |
| 4 | 0.3802 | 0.9723 | 3.087 | 0.04205 | 4.852 | 4.097
| 1.184e+03 | 5.684 | 0.1879 | 0.8088 |
| 5 | 0.441 | 0.806 | 3.468 | 0.09443 | 7.773 | 2.438
| 1.156e+03 | 6.976 | 0.6023 | 0.8334 |
| 6 | 0.4253 | 0.8353 | 1.842 | 0.01376 | 5.208 | 2.455
| 1.355e+03 | 4.386 | 9.884 | 0.551 |
| 7 | 0.4075 | 0.6044 | 1.645 | 0.06566 | 4.773 | 2.865
| 866.6 | 1.59 | 1.104 | 0.8282 |
| 8 | 0.4055 | 0.5691 | 1.786 | 0.0375 | 8.747 | 1.388
| 1.757e+03 | 0.961 | 9.765 | 0.7343 |
| 9 | 0.3488 | 0.9884 | 3.419 | 0.07419 | 3.274 | 2.131
| 680.3 | 2.961 | 1.187 | 0.659 |
| 10 | 0.3568 | 0.7071 | 1.257 | 0.06955 | 6.966 | 2.062
| 1.285e+03 | 0.9394 | 5.759 | 0.9646 |
| 11 | 0.418 | 0.6593 | 3.67 | 0.01405 | 8.014 | 2.158
| 774.8 | 5.865 | 0.2011 | 0.9145 |
| 12 | 0.4168 | 0.5023 | 3.711 | 0.02773 | 8.146 | 4.849
| 873.1 | 5.762 | 5.92 | 0.7861 |
| 13 | 0.3957 | 0.6115 | 4.811 | 0.04527 | 8.925 | 3.798
| 946.2 | 8.138 | 3.965 | 0.9406 |
| 14 | 0.3748 | 0.7906 | 4.527 | 0.06956 | 8.077 | 3.005
| 1.934e+03 | 6.44 | 4.239 | 0.8032 |
| 15 | 0.4071 | 0.5096 | 2.206 | 0.06636 | 5.031 | 3.472
| 1.143e+03 | 1.355 | 2.983 | 0.785 |
| 16 | 0.3638 | 0.7954 | 3.297 | 0.06567 | 7.565 | 2.726
| 1.845e+03 | 3.676 | 4.359 | 0.946 |
| 17 | 0.4227 | 0.9031 | 3.816 | 0.01092 | 9.436 | 3.857
| 1.998e+03 | 1.494 | 8.681 | 0.5812 |
| 18 | 0.3534 | 0.8078 | 1.495 | 0.08495 | 8.651 | 3.276
| 1.111e+03 | 0.6917 | 6.974 | 0.7268 |
| 19 | 0.3965 | 0.861 | 4.466 | 0.09758 | 8.991 | 1.047
| 1.04e+03 | 7.3 | 1.716 | 0.7605 |
| 20 | 0.3113 | 0.5272 | 1.8 | 0.002834 | 8.556 | 1.896
| 1.018e+03 | 9.281 | 7.044 | 0.5159 |
| 21 | 0.3901 | 0.812 | 2.314 | 0.08544 | 9.969 | 1.267
| 1.16e+03 | 5.981 | 9.481 | 0.6593 |
| 22 | 0.3798 | 1.0 | 5.0 | 0.1 | 3.0 | 5.0
| 1.155e+03 | 2.139 | 0.0 | 1.0 |
| 23 | 0.3936 | 0.7427 | 3.526 | 0.03991 | 6.75 | 3.804
| 912.6 | 7.341 | 3.601 | 0.9431 |
| 24 | 0.4013 | 0.6409 | 2.411 | 0.04466 | 4.251 | 3.641
| 1.127e+03 | 5.83 | 6.572 | 0.7809 |
| 25 | 0.3851 | 0.9628 | 4.123 | 0.05798 | 7.702 | 1.421
| 1.088e+03 | 8.925 | 7.968 | 0.7126 |
| 26 | 0.3524 | 0.8032 | 3.285 | 0.0812 | 7.613 | 3.329
| 1.463e+03 | 1.825 | 6.842 | 0.9468 |
| 27 | 0.385 | 0.9966 | 1.05 | 0.0913 | 8.3 | 2.533
| 1.153e+03 | 8.996 | 2.182 | 0.7109 |
| 28 | 0.3614 | 0.8227 | 4.337 | 0.02408 | 4.544 | 1.739
| 1.695e+03 | 3.108 | 8.403 | 0.9562 |

| 29 | 0.3907 | 0.9781 | 2.819 | 0.04431 | 8.908 | 3.403
| 1.296e+03 | 1.678 | 0.9366 | 0.8085 |
| 30 | 0.3825 | 0.9303 | 3.572 | 0.09915 | 7.059 | 1.008
| 687.5 | 5.161 | 5.16 | 0.5704 |
| 31 | 0.3797 | 0.7405 | 1.899 | 0.06215 | 4.612 | 4.402
| 1.387e+03 | 2.13 | 1.985 | 0.8184 |
| 32 | 0.4236 | 0.5286 | 4.644 | 0.07492 | 6.195 | 4.311
| 1.907e+03 | 5.982 | 3.569 | 0.8466 |
| 33 | 0.3723 | 0.8263 | 2.38 | 0.05286 | 3.568 | 2.752
| 1.154e+03 | 5.01 | 2.004 | 0.6375 |
| 34 | 0.3501 | 0.9644 | 3.613 | 0.04725 | 3.449 | 3.404
| 1.191e+03 | 8.224 | 7.101 | 0.755 |
| 35 | 0.4248 | 0.6988 | 4.729 | 0.003215 | 7.838 | 1.474
| 1.336e+03 | 6.881 | 0.2093 | 0.8806 |
| 36 | 0.4159 | 0.6587 | 2.963 | 0.04047 | 6.346 | 4.898
| 971.7 | 8.229 | 9.618 | 0.6229 |
| 37 | 0.3785 | 0.9983 | 2.992 | 0.09654 | 8.978 | 1.306
| 949.9 | 6.844 | 8.129 | 0.8197 |
| 38 | 0.3802 | 0.9707 | 2.157 | 0.07405 | 4.333 | 4.011
| 1.037e+03 | 9.579 | 5.18 | 0.5227 |
| 39 | 0.4016 | 0.73 | 3.887 | 0.04655 | 8.647 | 3.299
| 945.4 | 8.118 | 3.835 | 0.5428 |
| 40 | 0.3916 | 0.7999 | 4.48 | 0.04479 | 8.099 | 2.074
| 1.158e+03 | 6.821 | 1.412 | 0.7729 |
| 41 | 0.4427 | 0.9575 | 4.445 | 0.003271 | 9.823 | 3.087
| 1.156e+03 | 6.722 | 0.4112 | 0.5746 |
| 42 | 0.3926 | 0.5917 | 3.741 | 0.07389 | 9.317 | 3.199
| 1.156e+03 | 4.356 | 0.8916 | 0.8883 |
| 43 | 0.417 | 0.5077 | 2.329 | 0.01588 | 6.077 | 3.075
| 1.355e+03 | 4.178 | 9.081 | 0.8904 |
| 44 | 0.4018 | 0.6063 | 4.283 | 0.04041 | 7.71 | 3.185
| 1.155e+03 | 8.539 | 3.311 | 0.6518 |
| 45 | 0.3877 | 0.8827 | 4.753 | 0.05338 | 8.224 | 4.067
| 1.153e+03 | 6.015 | 0.5741 | 0.9317 |
| 46 | 0.3811 | 0.9149 | 1.948 | 0.01877 | 3.946 | 1.76
| 1.355e+03 | 4.299 | 8.289 | 0.5061 |
| 47 | 0.4269 | 0.6247 | 1.456 | 0.03129 | 7.135 | 3.046
| 1.157e+03 | 8.55 | 0.1817 | 0.8757 |
| 48 | 0.3761 | 0.9489 | 3.037 | 0.08433 | 9.54 | 4.016
| 1.157e+03 | 8.666 | 0.07724 | 0.6691 |
| 49 | 0.387 | 0.8114 | 4.723 | 0.04523 | 8.709 | 2.262
| 1.156e+03 | 6.087 | 0.7056 | 0.9729 |
| 50 | 0.3852 | 0.723 | 4.133 | 0.08006 | 6.273 | 4.062
| 1.906e+03 | 5.797 | 4.695 | 0.7237 |
| 51 | 0.4255 | 0.5516 | 3.984 | 0.03809 | 7.536 | 2.818
| 1.157e+03 | 6.767 | 1.105 | 0.9535 |
| 52 | 0.4522 | 0.6585 | 4.811 | 0.05453 | 8.669 | 1.404
| 1.156e+03 | 8.094 | 0.2741 | 0.8726 |
| 53 | 0.3848 | 0.7621 | 3.347 | 0.05451 | 5.429 | 3.98
| 971.6 | 7.935 | 8.701 | 0.7205 |
| 54 | 0.4261 | 0.5461 | 3.661 | 0.0623 | 8.746 | 2.526
| 1.157e+03 | 7.574 | 0.2177 | 0.5659 |
| 55 | 0.4226 | 0.7095 | 1.008 | 0.05615 | 6.447 | 2.787
| 1.157e+03 | 7.852 | 1.101 | 0.8772 |
| 56 | 0.3105 | 0.5304 | 4.812 | 0.001575 | 7.869 | 3.053
| 1.155e+03 | 8.767 | 1.358 | 0.7946 |
| 57 | 0.4267 | 0.7379 | 1.315 | 0.00722 | 6.944 | 4.585
| 1.156e+03 | 6.49 | 0.5259 | 0.6564 |
| 58 | 0.4231 | 0.6756 | 2.256 | 0.01259 | 8.574 | 2.312
| 1.158e+03 | 8.223 | 0.2954 | 0.7816 |

```
| 59          | 0.4253    | 0.6058    | 1.813     | 0.02846   | 6.057     | 3.578
| 1.157e+03 | 6.411     | 0.6889    | 0.7962    |
| 60          | 0.4254    | 0.5732    | 3.032     | 0.004982  | 7.783     | 1.869
| 1.156e+03 | 7.592     | 0.1918    | 0.6842    |
| 61          | 0.3955    | 0.593     | 1.238     | 0.08095   | 6.354     | 4.261
| 1.156e+03 | 6.025     | 2.449     | 0.6578    |
| 62          | 0.4048    | 0.97      | 1.01      | 0.06094   | 8.946     | 2.584
| 1.156e+03 | 6.662     | 1.941     | 0.9497    |
| 63          | 0.4211    | 0.8729    | 1.901     | 0.02917   | 6.795     | 2.837
| 1.155e+03 | 6.027     | 0.1386    | 0.739     |
| 64          | 0.3739    | 0.8       | 1.87      | 0.07425   | 5.673     | 3.223
| 1.157e+03 | 8.167     | 1.531     | 0.591     |
| 65          | 0.4138    | 0.9033    | 2.715     | 0.01092   | 9.46      | 2.018
| 1.156e+03 | 7.679     | 1.346     | 0.7532    |
| 66          | 0.4003    | 0.6941    | 2.706     | 0.02279   | 7.634     | 4.132
| 1.157e+03 | 7.421     | 2.206     | 0.8297    |
| 67          | 0.4296    | 0.7451    | 2.594     | 0.06569   | 7.491     | 2.992
| 1.156e+03 | 6.79      | 0.5319    | 0.8589    |
| 68          | 0.4267    | 0.9469    | 1.481     | 0.007131  | 7.69      | 2.113
| 1.156e+03 | 5.535     | 0.7025    | 0.8807    |
| 69          | 0.3906    | 0.9304    | 2.007     | 0.07809   | 6.507     | 4.105
| 1.156e+03 | 5.57      | 0.09368   | 0.9836    |
| 70          | 0.4393    | 0.5317    | 2.954     | 0.06121   | 8.184     | 1.573
| 1.158e+03 | 7.616     | 0.03646   | 0.9987    |
=============================================================================
=====================================================
```

Out[43]:




```
----------

median_absolute_error: 0.6345219159164428
mean_absolute_error: 0.9175571608217421
mean_squared_error: 1.8874593480804907
r2_score: 0.892746830585507
----------

median_absolute_error: 0.7947727151565551
mean_absolute_error: 1.3440422545743953
mean_squared_error: 7.318453858410396
r2_score: 0.553196527566472
```

模型解释

In [44]:
```
from matplotlib import rcParams
rcParams["font.family"] = ["Times New Roman", "SimSun"]

# model指向训练好的模型
model=xgbreg_pb_gq
```

In [45]:
```
featureimportance=pd.DataFrame(
    {"gain": model.get_booster().get_score(importance_type='gain'),
     "weight": model.get_booster().get_score(importance_type='weight'),
     "cover": model.get_booster().get_score(importance_type='cover')}
)
featureimportance
```

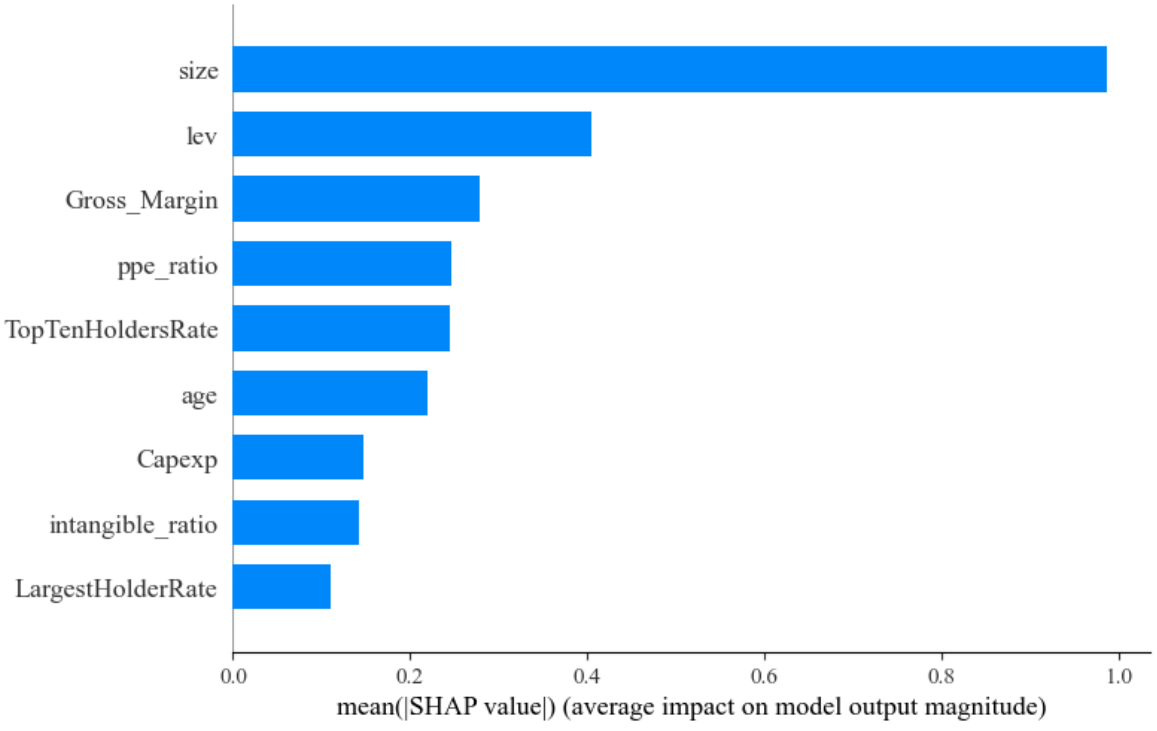
Out[45]:

	gain	weight	cover
Capexp	165.919373	1299.0	1322.929932
LargestHolderRate	52.716782	1377.0	803.522156
TopTenHoldersRate	81.243385	1398.0	856.545044
size	207.985443	1482.0	1951.215942
ppe_ratio	72.220818	1497.0	797.883118
intangible_ratio	107.979317	1384.0	804.452332
lev	187.637650	1330.0	1763.018066
Gross_Margin	79.723541	1369.0	1126.222046
age	83.576752	1068.0	996.573059

In [46]:
```
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X)
```

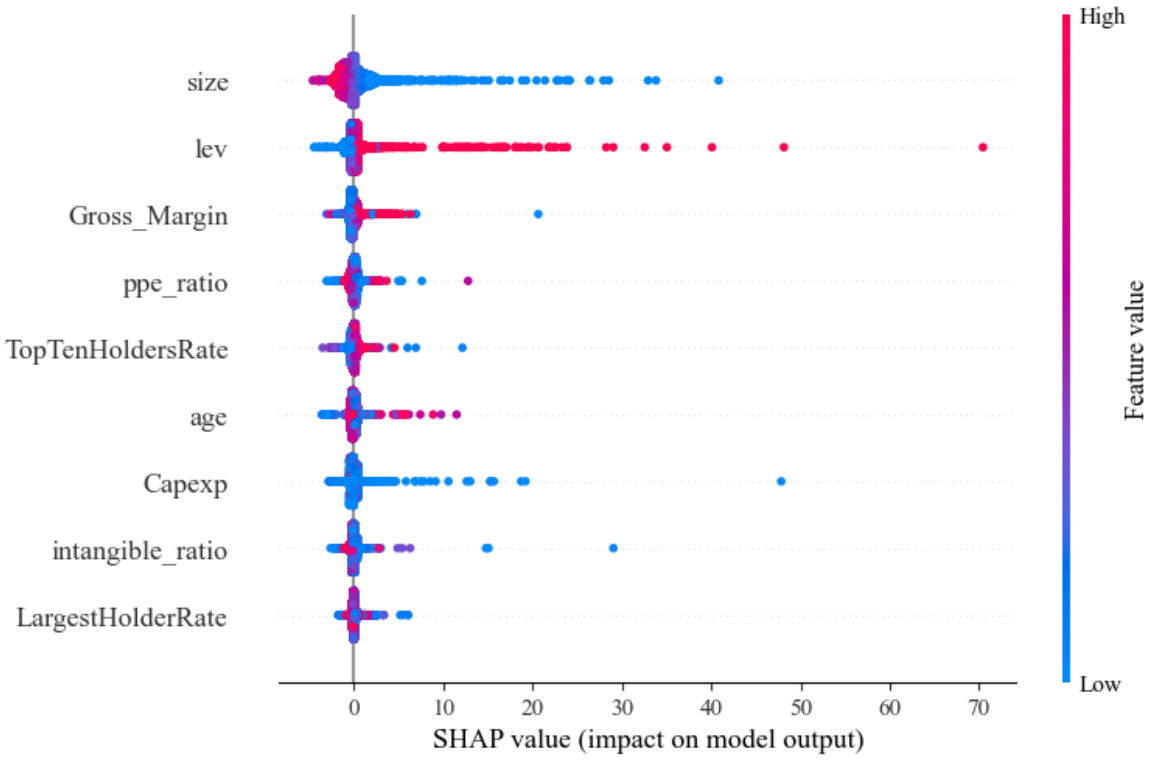
In [47]:
```
shap.summary_plot(shap_values, X, plot_type="bar")
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\1599793300.py:1: FutureWarnin
g: The NumPy global RNG was seeded by calling `np.random.seed`. In a future versi
on this function will no longer use the global RNG. Pass `rng` explicitly to opt-
in to the new behaviour and silence this warning.
  shap.summary_plot(shap_values, X, plot_type="bar")
```



In [48]:
```
shap.summary_plot(shap_values, X)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\3017445420.py:1: FutureWarnin
g: The NumPy global RNG was seeded by calling `np.random.seed`. In a future versi
on this function will no longer use the global RNG. Pass `rng` explicitly to opt-
in to the new behaviour and silence this warning.
  shap.summary_plot(shap_values, X)
```



民营

In [104…
```
df_po = pd.read_excel('data\df_po.xlsx')
df_po.head()
df_po.columns
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\3007340969.py:1: SyntaxWarnin
g: invalid escape sequence '\d'
  df_po = pd.read_excel('data\df_po.xlsx')
```

Out[104…

	Stkcd	year	Indcd1	Indnme1	EquityNature	Capexp	LargestHolderRate	T
0	7	2011	H61	住宿业	民营	723368.00	17.41	
1	7	2012	H61	住宿业	民营	1869274.04	17.41	
2	7	2013	H61	住宿业	民营	832043.82	17.41	
3	7	2015	H61	住宿业	民营	45829.00	15.17	
4	7	2016	H61	住宿业	民营	792581.11	13.53	

Out[104…
```
Index(['Stkcd', 'year', 'Indcd1', 'Indnme1', 'EquityNature', 'Capexp',
       'LargestHolderRate', 'TopTenHoldersRate', 'size', 'ppe_ratio',
       'intangible_ratio', 'lev', 'Gross_Margin', 'age', 'pe', 'pb'],
      dtype='object')
```

In [105…
```
# 定义特征列和目标列名
features = ['Capexp','LargestHolderRate', 'TopTenHoldersRate', 'size',
            'ppe_ratio','intangible_ratio', 'lev', 'Gross_Margin', 'age']

# 获取特征数据（X）和目标变量数据（y）
X = df_po[features]
```

```
y_pb = df_po['pb']
```

In [106…

```
from sklearn.model_selection import train_test_split

X_train_pb, X_test_pb, y_train_pb, y_test_pb = train_test_split(
    X, y_pb, test_size=0.3, random_state=42
)

print("PB 模型数据：")
print("X_train_pb shape:", X_train_pb.shape)
print("y_train_pb shape:", y_train_pb.shape)
print("X_test_pb shape:", X_test_pb.shape)
print("y_test_pb shape:", y_test_pb.shape)
```

```
PB 模型数据：
X_train_pb shape: (17378, 9)
y_train_pb shape: (17378,)
X_test_pb shape: (7449, 9)
y_test_pb shape: (7449,)
```

In [107…

```
xgb_model_pb = xgb.XGBRegressor(objective="reg:squarederror", random_state=42)

xgb_model_pb.fit(X_train_pb, y_train_pb)

y_train_pred_pb = xgb_model_pb.predict(X_train_pb)
y_test_pred_pb = xgb_model_pb.predict(X_test_pb)

print("训练集评估：")
myscoring(y_train_pb, y_train_pred_pb)

print("测试集评估：")
myscoring(y_test_pb, y_test_pred_pb)
```

Out[107…

▾ XGBRegressor

▶ Parameters

```
训练集评估：
----------

median_absolute_error: 0.8197997951126098
mean_absolute_error: 1.1341985475813798
mean_squared_error: 2.7301441029933113
r2_score: 0.983865463797295
测试集评估：
----------

median_absolute_error: 1.1095636865234377
mean_absolute_error: 2.093586804826088
mean_squared_error: 551.9456663493805
r2_score: 0.04152500534863912
```

调参

In [109…

```
def function_xgbreg(
    n_estimators,
    learning_rate,
    max_depth,
```

```
    min_child_weight,
    gamma,
    subsample,
    colsample_bytree,
    reg_alpha,      # 新增正则化参数
    reg_lambda      # 新增正则化参数
):
    xgbreg = XGBRegressor(
        objective="reg:squarederror",
        booster="gbtree",
        n_estimators=int(n_estimators),
        learning_rate=learning_rate,
        max_depth=int(max_depth),
        min_child_weight=min_child_weight,
        gamma=gamma,
        subsample=subsample,
        colsample_bytree=colsample_bytree,
        reg_alpha=reg_alpha,      # 添加正则化参数
        reg_lambda=reg_lambda     # 添加正则化参数
    )
    cvs = cross_val_score(estimator=xgbreg, X=X_train_pb, y=y_train_pb, scoring=
    return cvs.mean()

# 参数搜索空间调整（重点优化过拟合相关参数）
pds = {
    "n_estimators": (500, 1500),    # 扩大迭代次数范围
    "learning_rate": (0.001, 0.2),  # 降低学习率上限
    "max_depth": (3, 12),           # 降低树深度上限
    "min_child_weight": (1, 3),     # 降低最小样本权重上限
    "gamma": (1, 5),                # 提高分裂阈值下限
    "subsample": (0.6, 1),          # 限制子样本比例
    "colsample_bytree": (0.5, 1),   # 限制特征采样比例
    "reg_alpha": (0, 5),           # 添加L1正则化
    "reg_lambda": (0, 10)           # 添加L2正则化
}

optimizer = BayesianOptimization(function_xgbreg, pds, random_state=0)
optimizer.maximize(init_points=20, n_iter=50)
xgbpds = optimizer.max["params"]

# 最终模型构建（完整参数应用）
xgbreg_pb_my = XGBRegressor(
    objective="reg:squarederror",
    booster="gbtree",
    n_estimators=int(xgbpds["n_estimators"]),
    learning_rate=xgbpds["learning_rate"],
    max_depth=int(xgbpds["max_depth"]),
    min_child_weight=xgbpds["min_child_weight"],
    gamma=xgbpds["gamma"],
    subsample=xgbpds["subsample"],
    colsample_bytree=xgbpds["colsample_bytree"],
    reg_alpha=xgbpds["reg_alpha"],      # 应用优化后的正则化参数
    reg_lambda=xgbpds["reg_lambda"]     # 应用优化后的正则化参数
)

xgbreg_pb_my.fit(X_train_pb, y_train_pb)

y_train_pred_pb = xgbreg_pb_my.predict(X_train_pb)
y_test_pred_pb = xgbreg_pb_my.predict(X_test_pb)
```

```
myscoring(y_train_pb, y_train_pred_pb)
myscoring(y_test_pb, y_test_pred_pb)
```

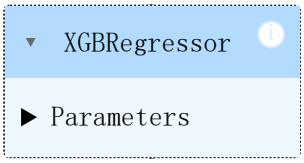
| iter | target | colsam... | gamma | learni... | max_depth | min_c
h... | n_esti... | reg_alpha | reg_la... | subsample |

| 1 | 0.1559 | 0.7744 | 3.861 | 0.1209 | 7.904 | 1.847
| 1.146e+03 | 2.188 | 8.918 | 0.9855 |
| 2 | 0.2853 | 0.6917 | 4.167 | 0.1063 | 8.112 | 2.851
| 571.0 | 0.4356 | 0.2022 | 0.933 |
| 3 | 0.1542 | 0.8891 | 4.48 | 0.1957 | 10.19 | 1.923
| 1.281e+03 | 0.5914 | 6.399 | 0.6573 |
| 4 | 0.329 | 0.9723 | 3.087 | 0.08352 | 5.381 | 2.548
| 956.2 | 2.842 | 0.1879 | 0.8471 |
| 5 | 0.1752 | 0.806 | 3.468 | 0.1888 | 9.136 | 1.719
| 937.0 | 3.488 | 0.6023 | 0.8667 |
| 6 | 0.2606 | 0.8353 | 1.842 | 0.02666 | 5.839 | 1.727
| 1.07e+03 | 2.193 | 9.884 | 0.6408 |
| 7 | 0.3023 | 0.6044 | 1.645 | 0.131 | 5.28 | 1.933
| 744.4 | 0.7948 | 1.104 | 0.8625 |
| 8 | 0.1603 | 0.5691 | 1.786 | 0.07438 | 10.39 | 1.194
| 1.338e+03 | 0.4805 | 9.765 | 0.7875 |
| 9 | 0.07942 | 0.9884 | 3.419 | 0.1481 | 3.353 | 1.566
| 620.2 | 1.481 | 1.187 | 0.7272 |
| 10 | 0.1521 | 0.7071 | 1.257 | 0.1388 | 8.099 | 1.531
| 1.023e+03 | 0.4697 | 5.759 | 0.9717 |
| 11 | 0.3337 | 0.6593 | 3.67 | 0.02723 | 9.447 | 1.579
| 683.2 | 2.933 | 0.2011 | 0.9316 |
| 12 | 0.1741 | 0.5023 | 3.711 | 0.05473 | 9.617 | 2.924
| 748.8 | 2.881 | 5.92 | 0.8289 |
| 13 | 0.1977 | 0.6115 | 4.811 | 0.08998 | 10.62 | 2.399
| 797.4 | 4.069 | 3.965 | 0.9524 |
| 14 | 0.1627 | 0.7906 | 4.527 | 0.1388 | 9.527 | 2.003
| 1.456e+03 | 3.22 | 4.239 | 0.8426 |
| 15 | 0.2015 | 0.5096 | 2.206 | 0.1324 | 5.611 | 2.236
| 928.8 | 0.6774 | 2.983 | 0.828 |
| 16 | 0.2426 | 0.7954 | 3.297 | 0.131 | 8.869 | 1.863
| 1.397e+03 | 1.838 | 4.359 | 0.9568 |
| 17 | 0.1341 | 0.9031 | 3.816 | 0.02095 | 11.28 | 2.428
| 1.499e+03 | 0.7472 | 8.681 | 0.665 |
| 18 | 0.0327 | 0.8078 | 1.495 | 0.1698 | 10.27 | 2.138
| 907.2 | 0.3458 | 6.974 | 0.7814 |
| 19 | 0.1158 | 0.861 | 4.466 | 0.1951 | 10.7 | 1.023
| 860.0 | 3.65 | 1.716 | 0.8084 |
| 20 | 0.3495 | 0.5272 | 1.8 | 0.004686 | 10.14 | 1.448
| 845.4 | 4.64 | 7.044 | 0.6127 |
| 21 | 0.1034 | 0.5396 | 1.34 | 0.1871 | 10.3 | 2.783
| 1.159e+03 | 1.094 | 8.178 | 0.9324 |
| 22 | 0.2183 | 0.6323 | 4.423 | 0.1095 | 4.66 | 1.143
| 1.341e+03 | 4.236 | 0.5911 | 0.8423 |
| 23 | 0.1364 | 0.7427 | 3.526 | 0.07921 | 7.821 | 2.402
| 775.1 | 3.67 | 3.601 | 0.9545 |
| 24 | 0.1154 | 0.6409 | 2.411 | 0.08877 | 4.608 | 2.32
| 918.0 | 2.915 | 6.572 | 0.8247 |
| 25 | 0.1626 | 0.9628 | 4.123 | 0.1155 | 9.045 | 1.211
| 892.2 | 4.462 | 7.968 | 0.7701 |
| 26 | 0.08203 | 0.8032 | 3.285 | 0.1622 | 8.93 | 2.165
| 1.142e+03 | 0.9124 | 6.842 | 0.9575 |
| 27 | 0.1751 | 0.5421 | 1.827 | 0.1392 | 10.82 | 1.534
| 844.7 | 1.46 | 9.432 | 0.8274 |
| 28 | 0.2144 | 0.8227 | 4.337 | 0.0474 | 4.985 | 1.37
| 1.297e+03 | 1.554 | 8.403 | 0.9649 |

| 29 | 0.321 | 0.9781 | 2.819 | 0.08805 | 10.6 | 2.202
| 1.031e+03 | 0.8392 | 0.9366 | 0.8468 |
| 30 | 0.1721 | 0.9303 | 3.572 | 0.1983 | 8.219 | 1.004
| 625.0 | 2.58 | 5.16 | 0.6563 |
| 31 | 0.2096 | 0.7405 | 1.899 | 0.1239 | 5.073 | 2.701
| 1.091e+03 | 1.065 | 1.985 | 0.8547 |
| 32 | 0.1799 | 0.5286 | 4.644 | 0.1496 | 7.108 | 2.656
| 1.438e+03 | 2.991 | 3.569 | 0.8773 |
| 33 | 0.142 | 0.8263 | 2.38 | 0.1052 | 3.731 | 1.876
| 935.9 | 2.505 | 2.004 | 0.71 |
| 34 | 0.02195 | 0.9644 | 3.613 | 0.09397 | 3.578 | 2.202
| 960.3 | 4.112 | 7.101 | 0.804 |
| 35 | 0.3704 | 0.6988 | 4.729 | 0.005452 | 9.22 | 1.237
| 1.058e+03 | 3.441 | 0.2093 | 0.9044 |
| 36 | 0.05729 | 0.6587 | 2.963 | 0.08034 | 7.302 | 2.949
| 814.4 | 4.115 | 9.618 | 0.6983 |
| 37 | 0.1819 | 0.9983 | 2.992 | 0.1931 | 10.69 | 1.153
| 800.0 | 3.422 | 8.129 | 0.8558 |
| 38 | 0.006878 | 0.9707 | 2.157 | 0.1478 | 4.714 | 2.506
| 857.7 | 4.79 | 5.18 | 0.6181 |
| 39 | 0.2852 | 0.6387 | 3.404 | 0.02886 | 9.123 | 1.816
| 681.9 | 3.197 | 0.2775 | 0.8564 |
| 40 | 0.2231 | 0.6865 | 3.869 | 0.04241 | 7.54 | 1.349
| 1.058e+03 | 3.751 | 0.6029 | 0.8353 |
| 41 | 0.1756 | 0.8931 | 1.303 | 0.02596 | 10.65 | 2.67
| 844.3 | 3.535 | 6.932 | 0.631 |
| 42 | 0.08819 | 0.5578 | 4.479 | 0.1354 | 8.156 | 1.466
| 682.4 | 3.233 | 0.1488 | 0.6225 |
| 43 | 0.3868 | 0.9089 | 2.495 | 0.003496 | 5.177 | 1.661
| 743.7 | 0.4578 | 1.041 | 0.6067 |
| 44 | 0.3664 | 0.9542 | 2.319 | 0.1075 | 9.298 | 2.84
| 1.031e+03 | 0.2912 | 0.7484 | 0.9949 |
| 45 | 0.1895 | 0.7673 | 4.723 | 0.107 | 9.189 | 1.878
| 1.057e+03 | 4.04 | 0.6215 | 0.8245 |
| 46 | 0.1904 | 0.9362 | 3.174 | 0.1214 | 10.0 | 2.782
| 683.4 | 3.712 | 0.6875 | 0.7611 |
| 47 | 0.2834 | 0.8973 | 3.597 | 0.07567 | 11.48 | 2.828
| 1.03e+03 | 0.96 | 0.8705 | 0.7985 |
| 48 | 0.3099 | 0.8492 | 4.1 | 0.01051 | 6.095 | 1.638
| 955.1 | 2.671 | 0.144 | 0.7643 |
| 49 | 0.3072 | 0.5998 | 3.128 | 0.03146 | 9.653 | 1.806
| 682.6 | 3.304 | 1.103 | 0.9313 |
| 50 | 0.1483 | 0.5 | 2.141 | 0.001 | 9.775 | 1.0
| 846.1 | 5.0 | 7.056 | 0.6 |
| 51 | 0.05865 | 0.7725 | 3.057 | 0.1175 | 10.92 | 1.216
| 1.031e+03 | 1.279 | 1.081 | 0.7347 |
| 52 | 0.2443 | 0.8631 | 3.49 | 0.03216 | 10.48 | 1.807
| 537.2 | 0.4562 | 0.2647 | 0.6816 |
| 53 | 0.1181 | 0.7102 | 1.706 | 0.1089 | 9.136 | 1.563
| 1.438e+03 | 1.506 | 9.395 | 0.9725 |
| 54 | 0.1799 | 0.9704 | 3.856 | 0.1547 | 7.949 | 1.826
| 1.413e+03 | 1.246 | 4.989 | 0.8423 |
| 55 | 0.2046 | 0.5098 | 4.371 | 0.1101 | 5.11 | 1.342
| 967.1 | 0.515 | 4.943 | 0.6764 |
| 56 | 0.3404 | 0.9412 | 4.433 | 0.03575 | 7.769 | 1.346
| 1.057e+03 | 3.118 | 0.02884 | 0.8037 |
| 57 | 0.1906 | 0.9377 | 4.685 | 0.1781 | 5.875 | 1.946
| 954.8 | 3.66 | 0.9687 | 0.7945 |
| 58 | 0.1624 | 0.5488 | 3.639 | 0.08229 | 7.046 | 2.348
| 955.4 | 2.409 | 0.5836 | 0.7309 |

```
| 59          | 0.2258    | 0.5357    | 1.706     | 0.06841   | 5.221     | 1.309
| 1.391e+03 | 1.571     | 2.037     | 0.6773    |
| 60          | 0.2336    | 0.5099    | 3.916     | 0.1693    | 8.146     | 2.787
| 1.366e+03 | 0.9445    | 2.907     | 0.8565    |
| 61          | 0.1802    | 0.9753    | 1.447     | 0.1631    | 4.053     | 1.355
| 506.5       | 2.168     | 8.399     | 0.8306    |
| 62          | 0.2024    | 0.8854    | 2.173     | 0.08992   | 11.7      | 2.135
| 548.6       | 3.87      | 2.217     | 0.8186    |
| 63          | 0.3082    | 0.9573    | 3.57      | 0.03765   | 5.567     | 2.051
| 955.7       | 2.776     | 0.08653   | 0.815     |
| 64          | 0.3065    | 0.7206    | 4.699     | 0.08722   | 8.011     | 1.856
| 572.1       | 0.1802    | 0.3555    | 0.9028    |
| 65          | 0.2534    | 0.8176    | 2.946     | 0.1758    | 5.577     | 1.918
| 744.2       | 0.8801    | 2.097     | 0.9031    |
| 66          | 0.06529   | 0.8481    | 2.874     | 0.1216    | 4.635     | 2.474
| 955.1       | 2.752     | 0.4702    | 0.7789    |
| 67          | 0.2229    | 0.6126    | 3.349     | 0.1923    | 9.361     | 1.476
| 1.057e+03 | 4.007     | 0.3266    | 0.9678    |
| 68          | 0.2921    | 0.6543    | 4.709     | 0.06362   | 8.268     | 1.04
| 1.057e+03 | 2.85      | 1.28      | 0.9345    |
| 69          | 0.1559    | 0.8337    | 3.0       | 0.1034    | 4.89      | 1.028
| 744.4       | 0.06196   | 1.359     | 0.7158    |
| 70          | 0.2642    | 0.9887    | 3.316     | 0.1748    | 11.8      | 1.636
| 810.8       | 3.653     | 6.02      | 0.8013    |
=============================================================================
=====================================================
```

Out[109…




```
----------

median_absolute_error: 1.3134588806304932
mean_absolute_error: 1.8570707546119096
mean_squared_error: 33.09265599435633
r2_score: 0.8044298630247384
----------

median_absolute_error: 1.3007100518188475
mean_absolute_error: 2.16383471079438
mean_squared_error: 540.3668043883429
r2_score: 0.061632146925776454
```

模型解释

In [110…

```
from matplotlib import rcParams
rcParams["font.family"] = ["Times New Roman", "SimSun"]

# model指向训练好的模型
model=xgbreg_pb_my
```

In [111…

```
featureimportance=pd.DataFrame(
    {"gain": model.get_booster().get_score(importance_type='gain'),
     "weight": model.get_booster().get_score(importance_type='weight'),
     "cover": model.get_booster().get_score(importance_type='cover')}
)
featureimportance
```

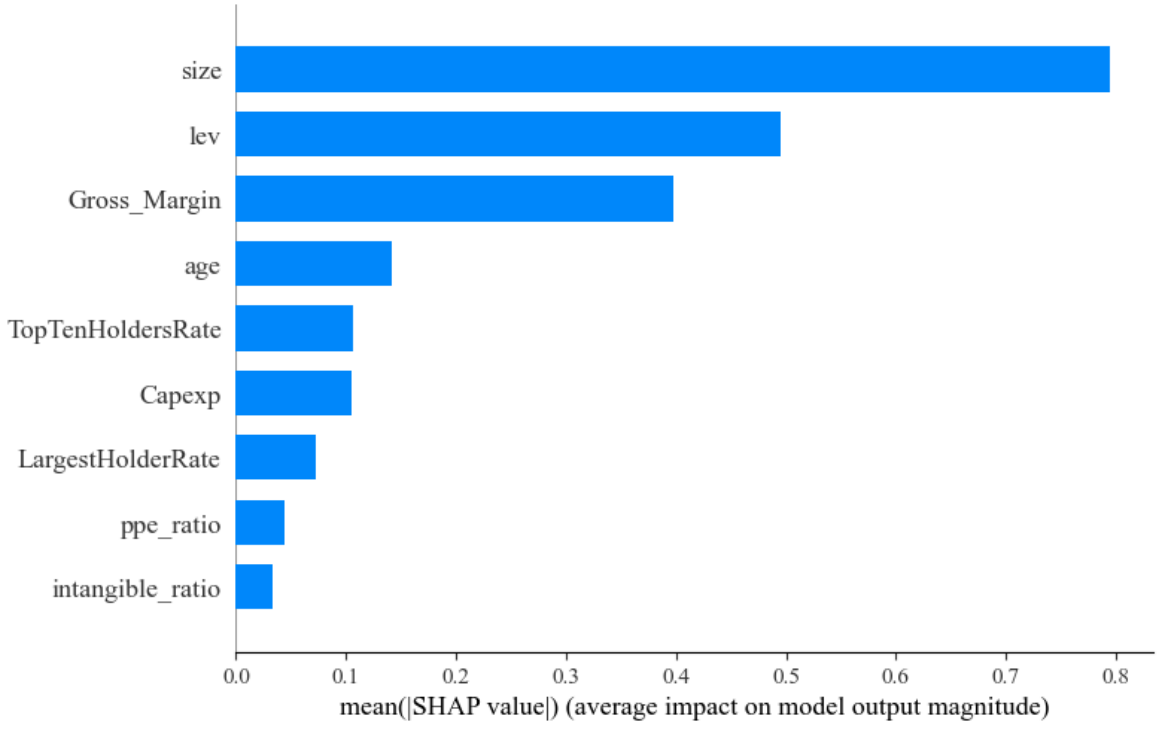
Out[111...

	gain	weight	cover
Capexp	33513.023438	1513.0	1034.339111
LargestHolderRate	47090.054688	1236.0	89.431229
TopTenHoldersRate	4576.290039	1643.0	1750.121094
size	20681.775391	2835.0	4951.875000
ppe_ratio	18550.898438	1071.0	473.787109
intangible_ratio	16709.542969	1084.0	326.362549
lev	22402.300781	2781.0	3976.354492
Gross_Margin	6433.604492	2131.0	3042.277832
age	2896.394043	969.0	2243.337402

In [112...
```
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X)
```

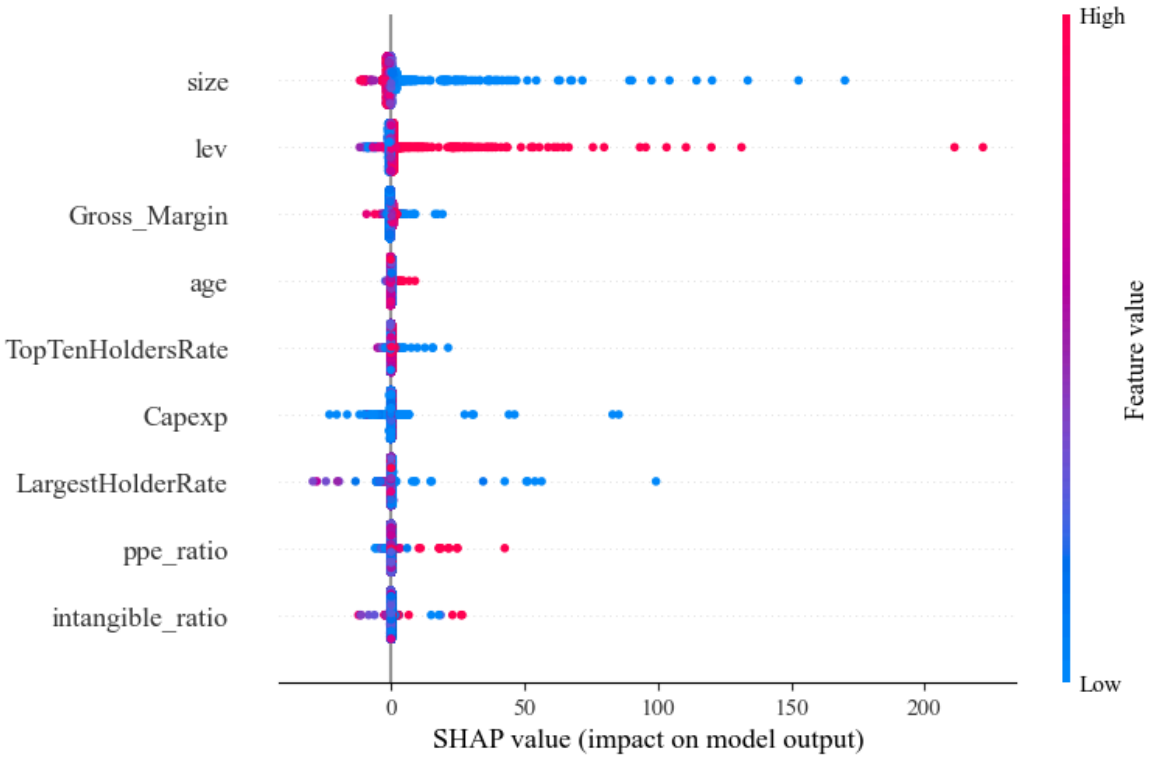
In [113...
```
shap.summary_plot(shap_values, X, plot_type="bar")
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\1599793300.py:1: FutureWarnin
g: The NumPy global RNG was seeded by calling `np.random.seed`. In a future versi
on this function will no longer use the global RNG. Pass `rng` explicitly to opt-
in to the new behaviour and silence this warning.
  shap.summary_plot(shap_values, X, plot_type="bar")
```



In [114...
```
shap.summary_plot(shap_values, X)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_19584\3017445420.py:1: FutureWarnin
g: The NumPy global RNG was seeded by calling `np.random.seed`. In a future versi
on this function will no longer use the global RNG. Pass `rng` explicitly to opt-
in to the new behaviour and silence this warning.
  shap.summary_plot(shap_values, X)
```



典型特征可视化

In [4]:
```
df_clean = pd.read_excel('data\df_clean.xlsx')
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_15752\171085527.py:1: SyntaxWarning:
invalid escape sequence '\d'
  df_clean = pd.read_excel('data\df_clean.xlsx')
```

营业毛利率

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

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


Gross_Margin_mean = df_clean.groupby(['year', 'EquityNature'])['Gross_Margin'].m

Gross_Margin_pivot = Gross_Margin_mean.pivot(index='year', columns='EquityNature

fig, ax = plt.subplots(figsize=(10, 6))


bar_width = 0.35
index = range(len(Gross_Margin_pivot))


ax.bar(index, Gross_Margin_pivot['国企'], bar_width, label='国企', alpha=0.7, co
ax.bar([i + bar_width for i in index], Gross_Margin_pivot['民营'], bar_width, la
```

```
plt.xlabel('年份')
plt.ylabel('营业毛利率均值')
plt.title('不同股权性质下各年份营业毛利率均值差异')
plt.xticks([i + bar_width / 2 for i in index], Gross_Margin_pivot['year'])
plt.legend(title="股权性质")
plt.grid(True)

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

Out[ ]:
```
<BarContainer object of 18 artists>
```

Out[ ]:
```
<BarContainer object of 18 artists>
```

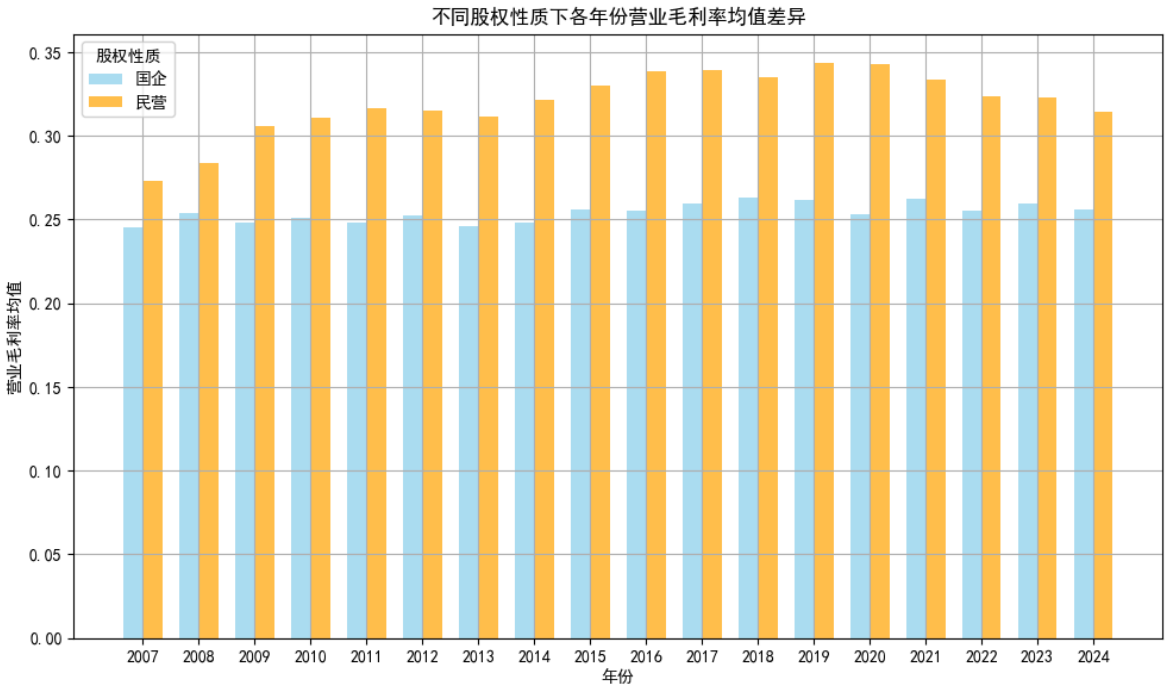
Out[ ]:
```
Text(0.5, 0, '年份')
```

Out[ ]:
```
Text(0, 0.5, '营业毛利率均值')
```

Out[ ]:
```
Text(0.5, 1.0, '不同股权性质下各年份营业毛利率均值差异')
```

Out[ ]:
```
([<matplotlib.axis.XTick at 0x24c0fd53390>,
  <matplotlib.axis.XTick at 0x24c0fd64410>,
  <matplotlib.axis.XTick at 0x24c0fd64b90>,
  <matplotlib.axis.XTick at 0x24c0fd65310>,
  <matplotlib.axis.XTick at 0x24c0fd65a90>,
  <matplotlib.axis.XTick at 0x24c0fd66210>,
  <matplotlib.axis.XTick at 0x24c0fd66990>,
  <matplotlib.axis.XTick at 0x24c0fd67110>,
  <matplotlib.axis.XTick at 0x24c0fd67890>,
  <matplotlib.axis.XTick at 0x24c0fd78050>,
  <matplotlib.axis.XTick at 0x24c0fd787d0>,
  <matplotlib.axis.XTick at 0x24c0fd78f50>,
  <matplotlib.axis.XTick at 0x24c0fd796d0>,
  <matplotlib.axis.XTick at 0x24c0fd79e50>,
  <matplotlib.axis.XTick at 0x24c0fd7a5d0>,
  <matplotlib.axis.XTick at 0x24c0fd7ad50>,
  <matplotlib.axis.XTick at 0x24c0fd7b4d0>,
  <matplotlib.axis.XTick at 0x24c0fd7bc50>],
 [Text(0.175, 0, '2007'),
  Text(1.175, 0, '2008'),
  Text(2.175, 0, '2009'),
  Text(3.175, 0, '2010'),
  Text(4.175, 0, '2011'),
  Text(5.175, 0, '2012'),
  Text(6.175, 0, '2013'),
  Text(7.175, 0, '2014'),
  Text(8.175, 0, '2015'),
  Text(9.175, 0, '2016'),
  Text(10.175, 0, '2017'),
  Text(11.175, 0, '2018'),
  Text(12.175, 0, '2019'),
  Text(13.175, 0, '2020'),
  Text(14.175, 0, '2021'),
  Text(15.175, 0, '2022'),
  Text(16.175, 0, '2023'),
  Text(17.175, 0, '2024')])
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x24c0fdd4550>
```



资产负债率

In [ ]:
```
import matplotlib.pyplot as plt

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

df_clean = pd.read_excel('data/df_clean.xlsx')

means = df_clean.groupby(['year', 'EquityNature'])['lev'].mean().reset_index()
yes_means = means[means['EquityNature'] == "国企"]
no_means = means[means['EquityNature'] == "民营"]


plt.plot(yes_means['year'], yes_means['lev'], label='国企', marker='p', color='c
plt.plot(no_means['year'], no_means['lev'], label='民营', marker='d', color='g')

plt.legend()
plt.xlabel('年份')
plt.ylabel('资产负债率均值')
plt.title('国企与民营资产负债率均值差异')
plt.grid(True)


plt.show()
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x24ba3649310>]
```

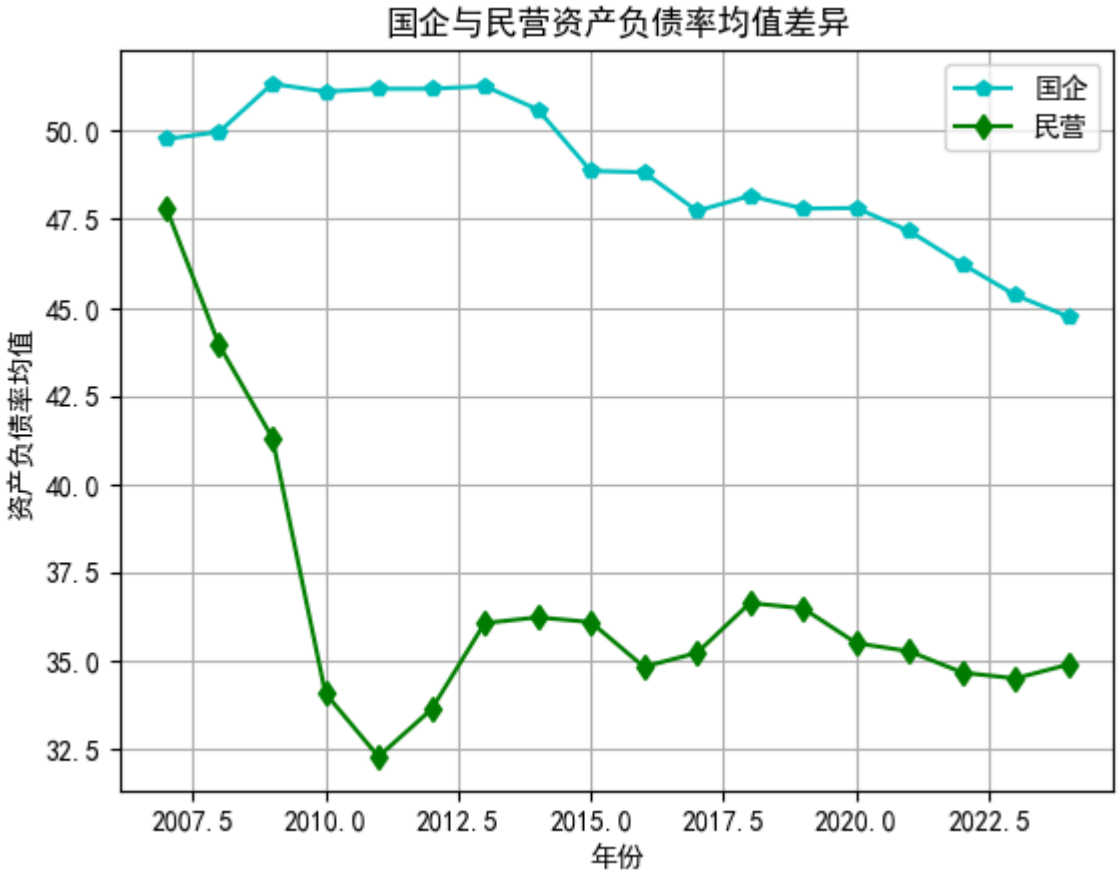
Out[ ]:
```
[<matplotlib.lines.Line2D at 0x24ba3649450>]
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x24ba3649590>
```

Out[ ]:
```
Text(0.5, 0, '年份')
```

Out[ ]:
```
Text(0, 0.5, '资产负债率均值')
```

Out[ ]:
```
Text(0.5, 1.0, '国企与民营资产负债率均值差异')
```


国企与民营资产负债率均值差异
国企
民营
50.0
47.5
45.0
42.5
40.0
37.5
35.0
32.5
资产负债率均值
2007.5
2010.0
2012.5
2015.0
2017.5
2020.0
2022.5
年份