```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import os
print(os.listdir('/content/drive/MyDrive'))
```

```
['抗生素含量高和抗生素抗性基因含量高有关系吗？具有什么关系.gsheet', '无标题文档.gdoc', 'Google AI Studio', 'Colab Notebooks', 'Animals',
```

```
%pip install lime
```

```
Requirement already satisfied: lime in /usr/local/lib/python3.12/dist-packages (0.2.0.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (from lime) (3.10.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from lime) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.12/dist-packages (from lime) (1.16.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from lime) (4.67.3)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.12/dist-packages (from lime) (1.6.1)
Requirement already satisfied: scikit-image>=0.12 in /usr/local/lib/python3.12/dist-packages (from lime) (0.25.2)
Requirement already satisfied: networkx>=3.0 in /usr/local/lib/python3.12/dist-packages (from scikit-image>=0.12->lime) (3.6.1)
Requirement already satisfied: pillow>=10.1 in /usr/local/lib/python3.12/dist-packages (from scikit-image>=0.12->lime) (11.3.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in /usr/local/lib/python3.12/dist-packages (from scikit-image>=0.12->lime) (2.37.3
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.12/dist-packages (from scikit-image>=0.12->lime) (2026.3.3)
Requirement already satisfied: packaging>=21 in /usr/local/lib/python3.12/dist-packages (from scikit-image>=0.12->lime) (26.0)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.12/dist-packages (from scikit-image>=0.12->lime) (0.5)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn>=0.18->lime) (1.5.3)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn>=0.18->lime) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->lime) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib->lime) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib->lime) (4.62.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->lime) (1.5.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib->lime) (3.3.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib->lime) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib->lime) (1.17.0
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import shap
import lime
import lime.lime_tabular
import warnings
import lightgbm as lgb

from sklearn.model_selection import train_test_split, GridSearchCV, KFold
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, BaggingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.inspection import PartialDependenceDisplay
from sklearn.preprocessing import StandardScaler

# --- Matplotlib 和 Seaborn 美化设置 ---
plt.rcParams['font.family'] = 'sans-serif'
plt.style.use('seaborn-v0_8-whitegrid')
warnings.filterwarnings("ignore")
```

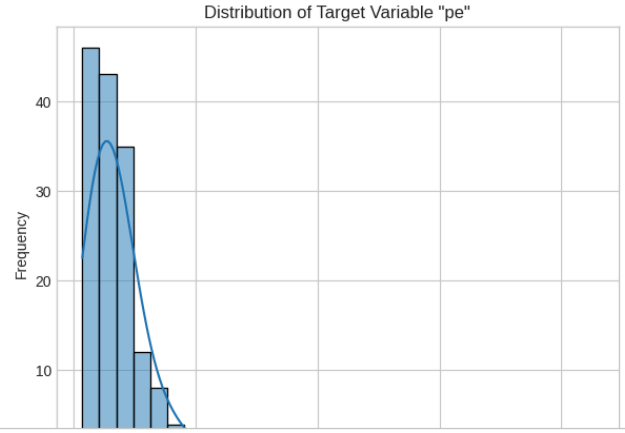
```
# ==========================================================
# 1. 数据加载与初步探查
# ==========================================================
print("--- 1. 数据加载与初步探查 ---")
data = pd.read_csv('/content/drive/MyDrive/公众号Python机器学习ml-2025-7-26数据.csv')
pd.set_option('display.max_columns', None)

print("数据信息 (data.info()):")
data.info()
print("\n描述性统计 (data.describe()):")
print(data.describe())
print("\n数据前5行 (data.head()):")
print(data.head())
```

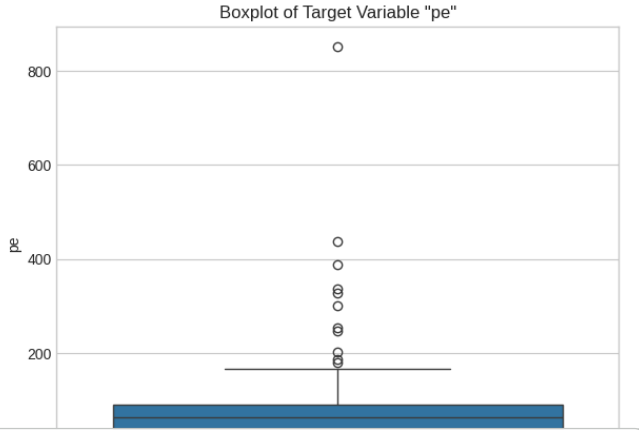
```
 9   incomegrow       158 non-null    float64
 10  netprofitgrow    158 non-null    float64
 11  cashflowgrow     158 non-null    float64
dtypes: float64(12)
memory usage: 14.9 KB

描述性统计 (data.describe()):
               pe         roe        debt  assetturnover      rdgrow  \
count  158.000000  158.000000  158.000000     158.000000  158.000000
mean    83.307215   10.075633   30.121456       0.557532   32.605506
std     90.878100    6.143545   15.653299       0.230418   54.377280
min     14.600000    1.130000    2.260000       0.120000  -41.270000
25%     38.730000    5.952500   18.200000       0.410000    9.215000
50%     62.795000    9.545000   28.515000       0.530000   20.990000
75%     89.875000   12.427500   41.987500       0.670000   39.632500
max    851.710000   36.970000   71.670000       1.330000  499.270000

              roic         rop  netincomeprofit  quickratio  incomegrow  \
count  158.000000  158.000000       158.000000  158.000000  158.000000
mean     7.671835   68.638354        65.176329    3.743797   21.026646
std      4.695364   66.580491        61.833241    5.299191   33.253435
min      0.270000   -3.340000      -277.400000    0.410000  -46.710000
25%      4.887500   28.605000        58.642500    1.310000    7.765000
50%      6.910000   47.730000        75.415000    2.130000   17.510000
75%      9.672500   81.770000        86.762500    3.707500   27.322500
max     27.670000  432.530000       418.400000   42.940000  331.700000

       netprofitgrow  cashflowgrow
count     158.000000    158.000000
mean       44.651139     95.075127
std       136.656692    530.459536
min       -94.500000  -1340.550000
25%        -4.975000    -39.407500
50%        19.195000     28.875000
75%        43.647500    124.252500
max      1045.390000   4669.230000

数据前5行 (data.head()):
       pe    roe   debt  assetturnover  rdgrow   roic     rop  \
0  851.71  36.97  24.31           0.91   84.37  22.71   35.01
1  437.93  32.77  22.84           0.90   69.15  21.72   79.62
2  387.54  30.60  59.05           0.94   59.88   1.81   10.93
3  337.62  27.28  23.80           0.37   52.33  22.50  116.83
4  328.84  24.49  41.46           0.53   51.39  27.67   72.40

   netincomeprofit  quickratio  incomegrow  netprofitgrow  cashflowgrow
0           418.40        2.28       69.32        1045.39         66.19
1           312.92        3.75       62.96         194.31        723.18
2           143.97        1.17       59.49          10.19        648.22
3            98.70        3.33       49.30          41.60        130.80
4            94.07        1.25       48.25         108.65         14.27
```

```
# ============================================================
# 2. 探索性数据分析 (EDA)
# ============================================================
print("\n--- 2. 探索性数据分析 (EDA) ---")

# --- 2.1 目标变量分布 ---
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.histplot(data.iloc[:, 0], kde=True, bins=30)
plt.title(f'Distribution of Target Variable "{data.columns[0]}"')
plt.xlabel("Value")
plt.ylabel("Frequency")

plt.subplot(1, 2, 2)
sns.boxplot(y=data.iloc[:, 0])
plt.title(f'Boxplot of Target Variable "{data.columns[0]}"')
plt.tight_layout()
plt.savefig('EDA_目标变量分布.png')
plt.show()

# --- 2.2 特征与目标变量的相关性热力图 ---
plt.figure(figsize=(12, 10))
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Correlation Heatmap of Features and Target')
plt.savefig('EDA_相关性热力图.png')
plt.show()

# --- 2.3 查看与目标变量最相关的几个特征的散点图 ---
target_corr = correlation_matrix.iloc[1:, 0].abs().sort_values(ascending=False)
top_features = target_corr.head(3).index

print(f"\n与目标变量最相关的3个特征是: {list(top_features)}")
plt.figure(figsize=(15, 5))
for i, feature in enumerate(top_features):
```

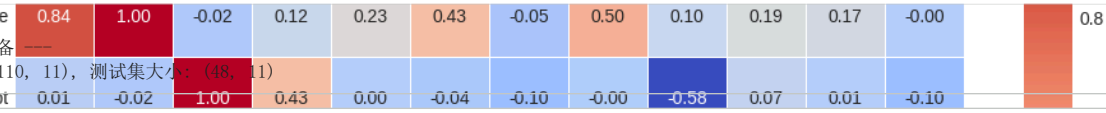
```
            plt.subplot(1,  3,  i  +  1)
            sns.scatterplot(x=data[feature],  y=data.iloc[:,  0])
            plt.title(f'"{feature}"  vs  "{data.columns[0]}"')
            plt.xlabel(feature)
            plt.ylabel(data.columns[0])
    plt.tight_layout()
    plt.savefig('EDA_高相关特征散点图.png')
    plt.show()
```

```
--- 2. 探索性数据分析 (EDA) ---
```





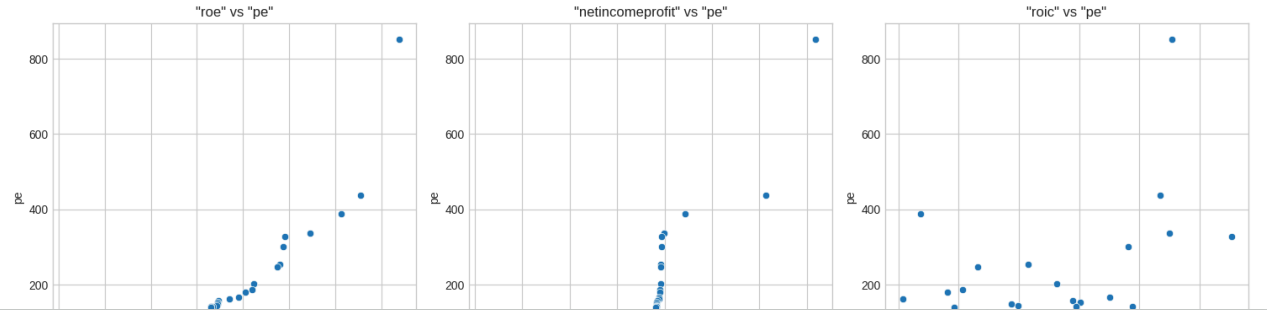

```
# ============================================================
# 3. 数据准备
# ============================================================
print("\n--- 3. 数据准备 ---")
X = data.iloc[:, 1:]
y = data.iloc[:, 0]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
print(f"训练集大小: {X_train.shape}, 测试集大小: {X_test.shape}")
```

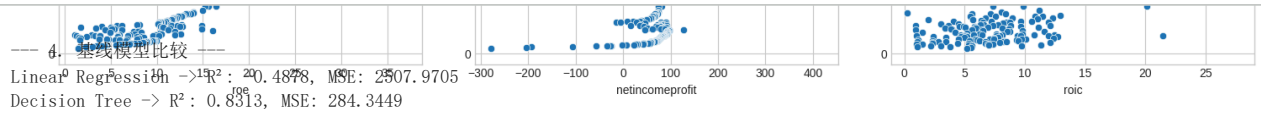



```
--- 3. 数据准备 ---
训练集大小: (110, 11), 测试集大小: (48, 11)
```

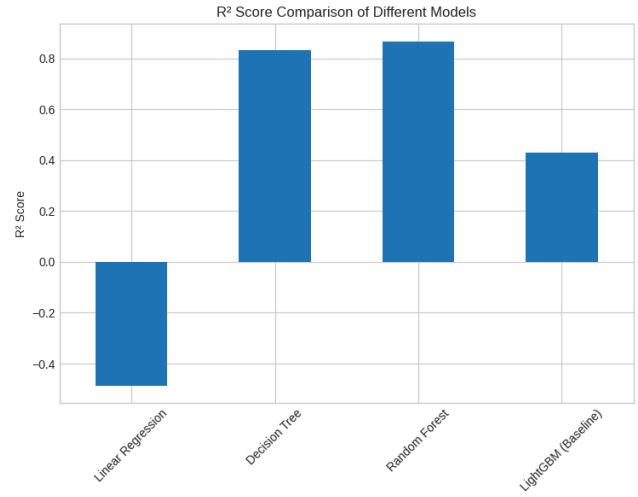
```
# ============================================================
# 4. 基线模型比较（包含LightGBM）
# ============================================================
print("\n--- 4. 基线模型比较 ---")

models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(random_state=42),
    "Random Forest": RandomForestRegressor(random_state=42, n_estimators=100),
    "LightGBM (Baseline)": lgb.LGBMRegressor(random_state=42, verbose=-1)
}

results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    results[name] = {'R²': r2, 'MSE': mse}
    print(f"{name} -> R²: {r2:.4f}, MSE: {mse:.4f}")

results_df = pd.DataFrame(results).T
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))
results_df['R²'].plot(kind='bar', ax=ax1, title='R² Score Comparison of Different Models')
ax1.set_ylabel('R² Score')
ax1.tick_params(axis='x', rotation=45)

results_df['MSE'].plot(kind='bar', ax=ax2, title='MSE Comparison of Different Models', color='orange')
ax2.set_ylabel('MSE')
ax2.tick_params(axis='x', rotation=45)
plt.tight_layout()
plt.savefig('模型比较_性能对比.png')
plt.show()
```

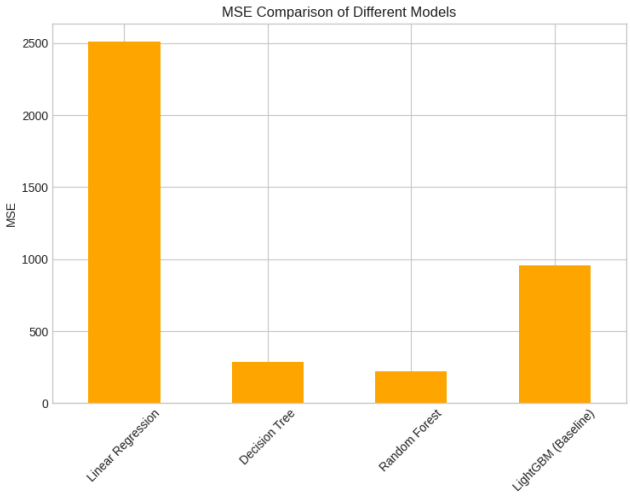

```
与目标变量最相关的3个特征是: ['roe', 'netincomeprofit', 'roic']
```






```
--- 4. 基线模型比较 ---
Linear Regression -> R²: 0.4878, MSE: 2507.9705
Decision Tree -> R²: 0.8313, MSE: 284.3449
Random Forest -> R²: 0.8677, MSE: 222.9395
LightGBM (Baseline) -> R²: 0.4308, MSE: 959.4920
```



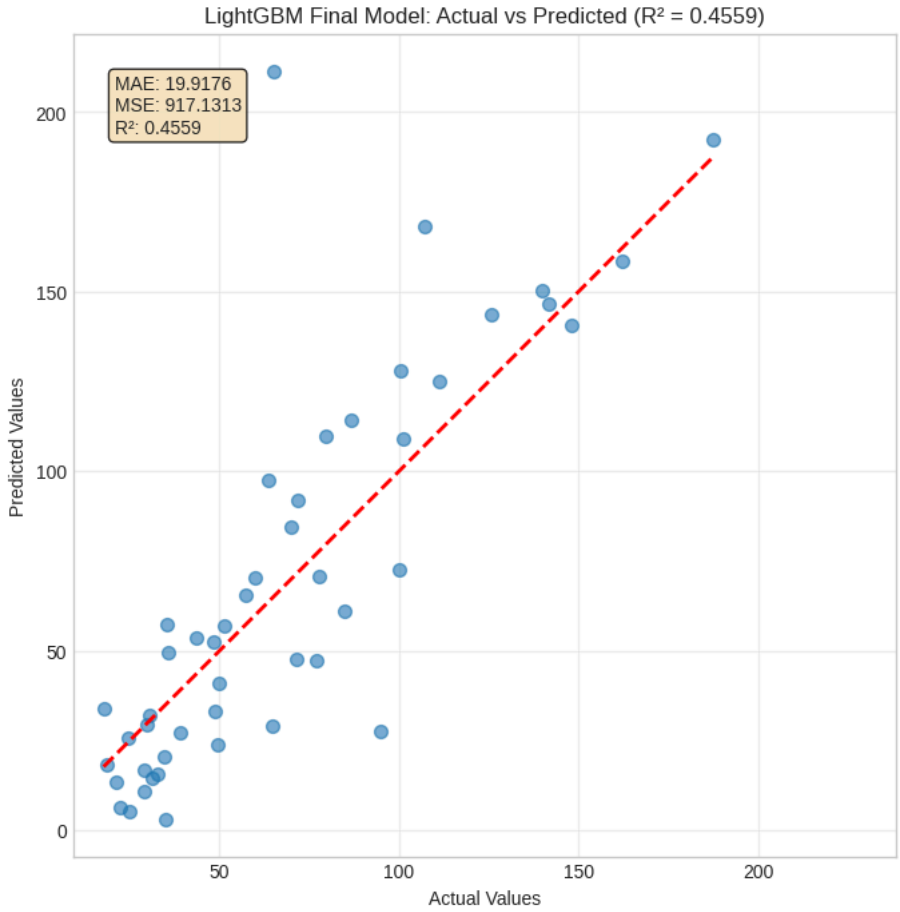


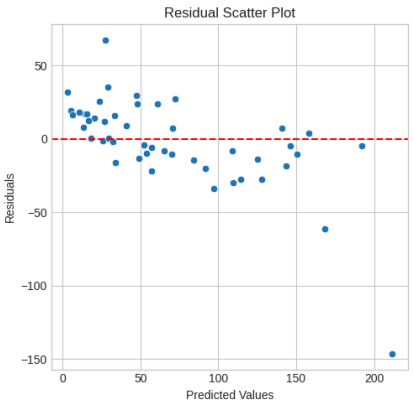

```
# ============================================================
# 5. LightGBM超参数调优
# ============================================================
print("\n--- 5. LightGBM超参数调优 ---")

# LightGBM参数网格
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [-1, 10],
    'num_leaves': [31, 50],
    'learning_rate': [0.05, 0.1],
    'feature_fraction': [0.8, 0.9],
    'bagging_fraction': [0.8, 0.9]
}

# 初始化LightGBM模型
lgb_model = lgb.LGBMRegressor(random_state=42, verbose=-1)

# 网格搜索
grid_search = GridSearchCV(
    estimator=lgb_model,
    param_grid=param_grid,
    cv=5,
    n_jobs=-1,
    verbose=1,
    scoring='r2'
)

print("开始LightGBM超参数调优...")
grid_search.fit(X_train, y_train)

# 获取最佳模型
final_model = grid_search.best_estimator_
print(f"\n最佳参数组合: {grid_search.best_params_}")
print(f"交叉验证最佳R²得分: {grid_search.best_score_:.4f}")
```

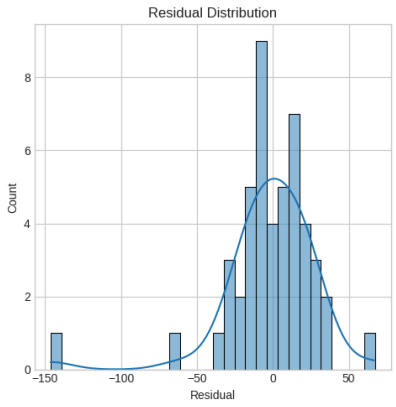
```
--- 5. LightGBM超参数调优 ---
开始LightGBM超参数调优...
Fitting 5 folds for each of 64 candidates, totalling 320 fits

最佳参数组合: {'bagging_fraction': 0.8, 'feature_fraction': 0.8, 'learning_rate': 0.1, 'max_depth': -1, 'n_estimators': 100, 'num_leaves
交叉验证最佳R²得分: 0.4686
```

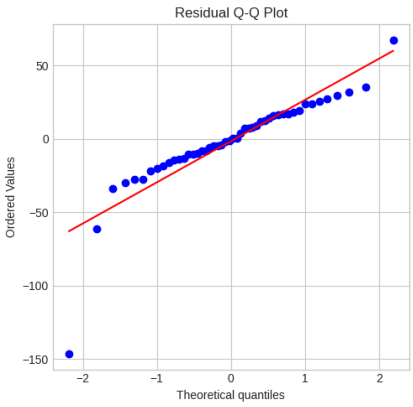
```
# ============================================================
# 6. 最终模型评估
# ============================================================
print("\n--- 6. 最终模型评估 ---")
y_pred = final_model.predict(X_test)

# --- 6.1 性能指标 ---
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"测试集 MAE: {mae:.4f}")
print(f"测试集 MSE: {mse:.4f}")
print(f"测试集 R²: {r2:.4f}")

# --- 6.2 预测值 vs 真实值散点图 ---
plt.figure(figsize=(10, 8))
plt.scatter(y_test, y_pred, alpha=0.6, s=50)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title(f'LightGBM Final Model: Actual vs Predicted (R² = {r2:.4f})')
plt.axis('equal')
plt.axis('square')

plt.text(0.05, 0.95, f'MAE: {mae:.4f}\nMSE: {mse:.4f}\nR²: {r2:.4f}',
                  transform=plt.gca().transAxes, verticalalignment='top',
                  bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.8))
plt.grid(True, alpha=0.3)
plt.savefig('LightGBM_真实值vs预测值.png', dpi=300)
plt.show()

# --- 6.3 残差分析 ---
residuals = y_test - y_pred
plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)
sns.scatterplot(x=y_pred, y=residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residual Scatter Plot')

plt.subplot(1, 3, 2)
sns.histplot(residuals, kde=True, bins=30)
plt.title('Residual Distribution')
plt.xlabel('Residual')

plt.subplot(1, 3, 3)
from scipy import stats
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Residual Q-Q Plot')

plt.tight_layout()
plt.savefig('LightGBM_残差分析.png', dpi=300)
plt.show()
```

```
--- 6. 最终模型评估 ---
测试集 MAE: 19.9176
测试集 MSE: 917.1313
测试集 R²: 0.4559
```







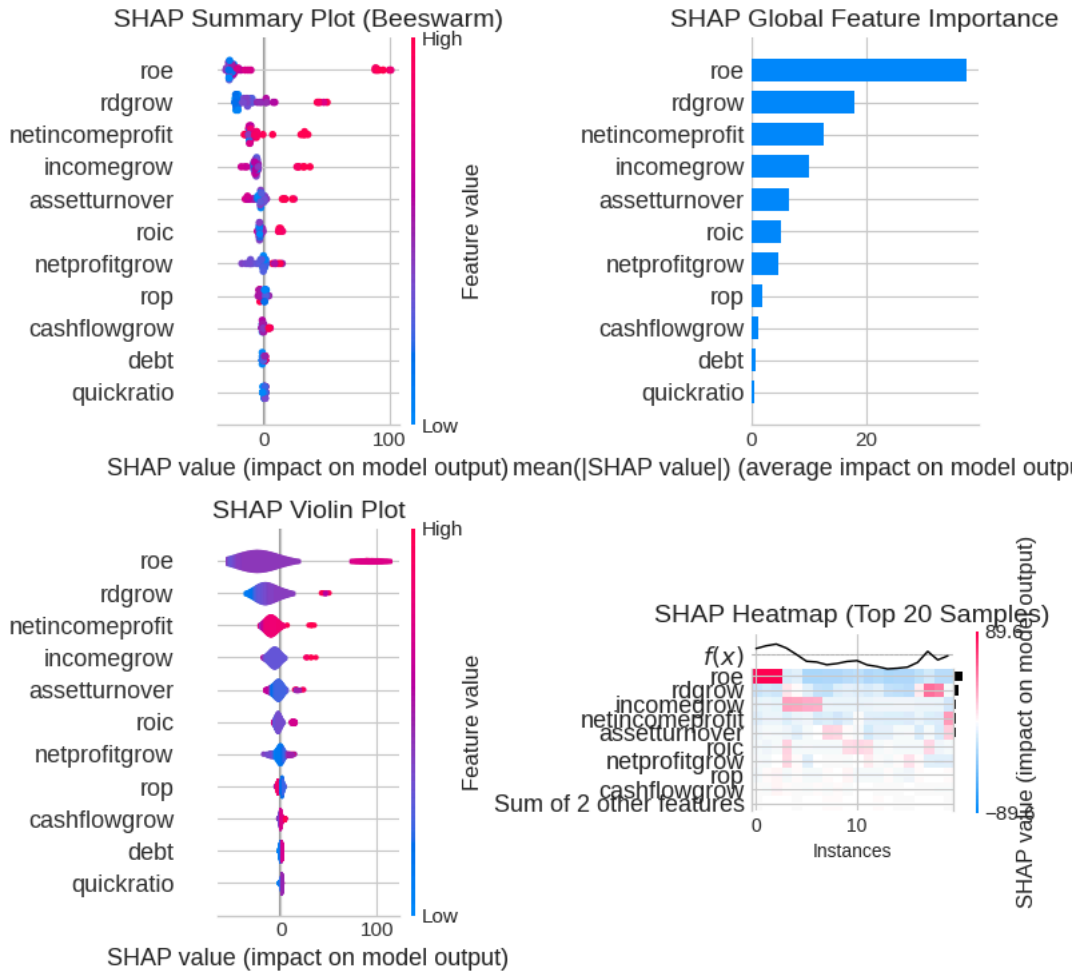



```
# --- 7.1 计算SHAP值 ---
print("正在计算SHAP值...")
explainer = shap.TreeExplainer(final_model)
shap_values = explainer.shap_values(X_test)

print(f"SHAP值形状: {shap_values.shape}")
print(f"基准值(expected_value): {explainer.expected_value:.4f}")
```

```
正在计算SHAP值...
SHAP值形状: (48, 11)
基准值(expected_value): 90.2898
```

```
# --- 7.2 SHAP摘要图系列 ---
fig, axes = plt.subplots(2, 2, figsize=(20, 16))
```

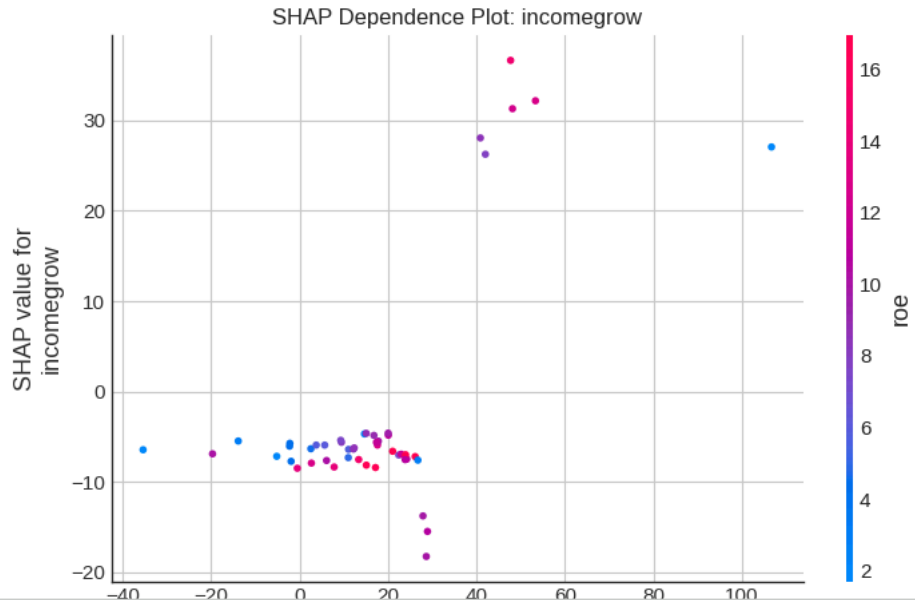
```
# Beeswarm plot (蜂群图)
plt.subplot(2, 2, 1)
shap.summary_plot(shap_values, X_test, plot_type="dot", show=False)
plt.title("SHAP Summary Plot (Beeswarm)", fontsize=14)

# Bar plot (条形图)
plt.subplot(2, 2, 2)
shap.summary_plot(shap_values, X_test, plot_type="bar", show=False)
plt.title("SHAP Global Feature Importance", fontsize=14)

# Violin plot (小提琴图)
plt.subplot(2, 2, 3)
shap.summary_plot(shap_values, X_test, plot_type="violin", show=False)
plt.title("SHAP Violin Plot", fontsize=14)

# Heatmap for top samples (热力图)
plt.subplot(2, 2, 4)
shap.plots.heatmap(shap.Explanation(
        values=shap_values[:20],
        base_values=explainer.expected_value,
        data=X_test.values[:20],
        feature_names=X_test.columns.tolist()
), show=False)
plt.title("SHAP Heatmap (Top 20 Samples)", fontsize=14)

plt.tight_layout()
plt.savefig('SHAP_综合分析.png', dpi=300, bbox_inches='tight')
plt.show()
```




```
# --- 7.3 SHAP依赖图（交互效应分析）---
print("正在生成SHAP依赖图...")
# 选择最重要的特征进行依赖图分析
importances = np.abs(shap_values).mean(0)
top_features_indices = np.argsort(importances)[-4:]
top_features = X_test.columns[top_features_indices]

for feature in top_features:
        plt.figure(figsize=(10, 6))
        shap.dependence_plot(feature, shap_values, X_test, interaction_index="auto", show=False)
        plt.title(f'SHAP Dependence Plot: {feature}')
        plt.tight_layout()
```

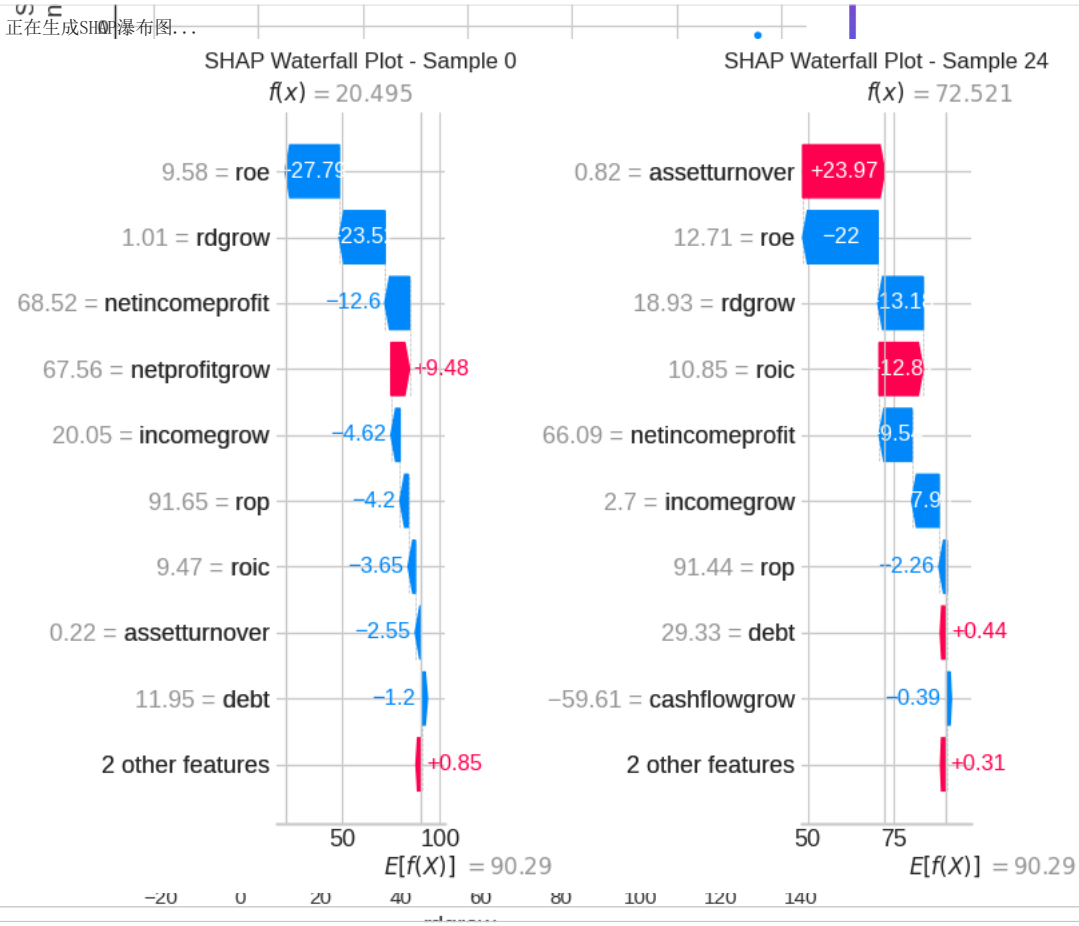
```
    plt.savefig(f'SHAP_依赖图_{feature}.png',  dpi=300,  bbox_inches='tight')
    plt.show()
```

```
正在生成SHAP依赖图...
<Figure size 1000x600 with 0 Axes>
```

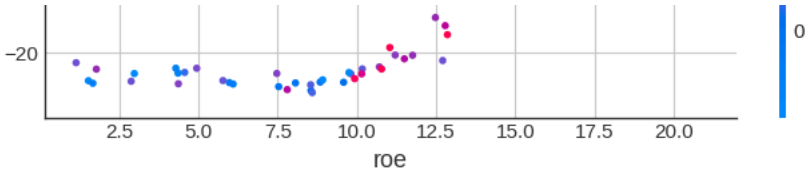



```
# --- 7.4 SHAP瀑布图（单个预测详细分解）---
print("正在生成SHAP瀑布图...")
fig, axes = plt.subplots(1, 2, figsize=(24, 8))

for i, idx in enumerate([0, len(X_test) // 2]):
        plt.subplot(1, 2, i + 1)
        shap.plots.waterfall(shap.Explanation(
                values=shap_values[idx],
                base_values=explainer.expected_value,
                data=X_test.iloc[idx].values,
                feature_names=X_test.columns.tolist()
        ), show=False)
        plt.title(f'SHAP Waterfall Plot - Sample {idx if idx >= 0 else len(X_test) + idx}')

plt.tight_layout()
plt.savefig('SHAP_瀑布图.png', dpi=500, bbox_inches='tight')
plt.show()
```

```
正在生成SHAP瀑布图...
```



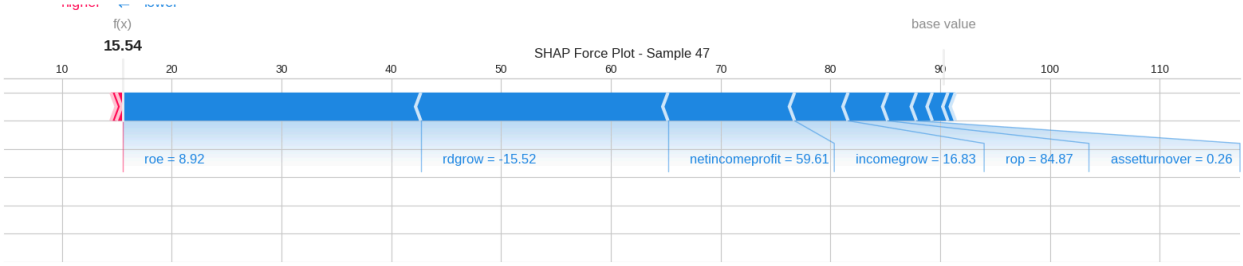
```
# --- 7.5 SHAP力图 ---
print("正在生成SHAP力图...")
shap.initjs()

# 选择几个有代表性的样本
sample_indices = [0, len(X_test) // 4, len(X_test) // 2, 3 * len(X_test) // 4, -1]

for i, idx in enumerate(sample_indices):
    force_plot = shap.force_plot(
        explainer.expected_value,
        shap_values[idx],
        X_test.iloc[idx],
        matplotlib=True,
        show=False
    )
    plt.title(f"SHAP Force Plot - Sample {idx if idx >= 0 else len(X_test) + idx}")
    plt.savefig(f'SHAP_力图_样本_{i}.png', bbox_inches='tight', dpi=300)
    plt.show()
```



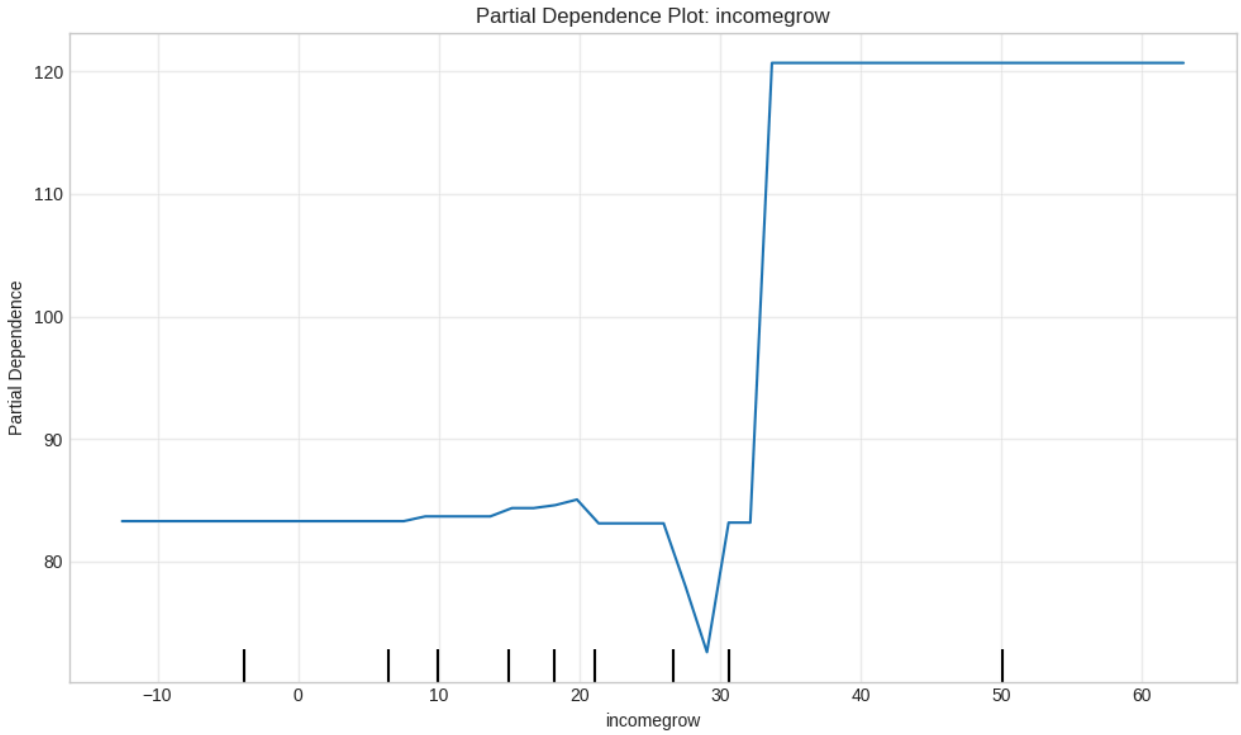
正在生成SHAP力图...

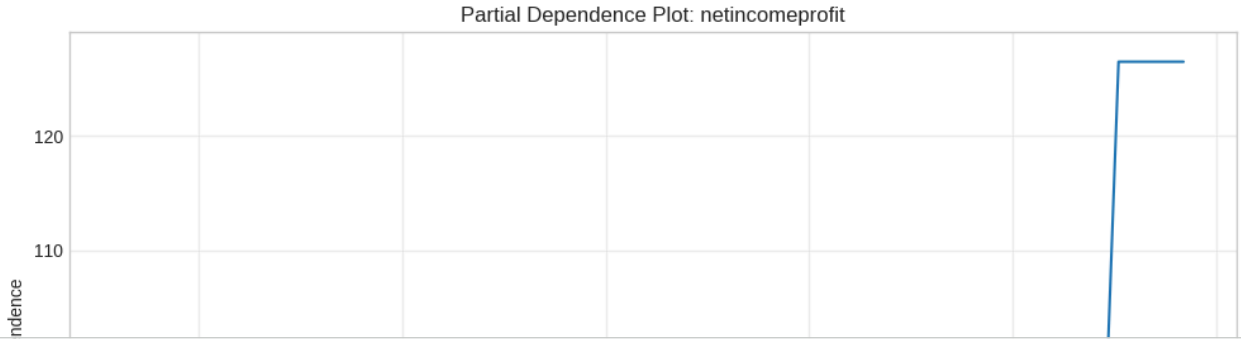
higher ⇄ lower

```
#  ---  7.6  SHAP部分依赖图  ---
print("正在生成SHAP部分依赖图...")
#  选择最重要的特征进行PDP分析
#  注意：这部分我们复用之前计算出的  top_features
from  sklearn.inspection  import  PartialDependenceDisplay

top_pdp_features  =  top_features[:2]
for  feature  in  top_pdp_features:
        plt.figure(figsize=(10,  6))
        PartialDependenceDisplay.from_estimator(
                final_model,
                X_train,
                features=[feature],
                kind='average',
                grid_resolution=50,
                ax=plt.gca()
        )
        plt.title(f'Partial  Dependence  Plot:  {feature}')
        plt.xlabel(feature)
        plt.ylabel('Partial  Dependence')
        plt.grid(True,  alpha=0.3)
        plt.tight_layout()
        plt.savefig(f'SHAP_PDP_{feature}.png',  dpi=300,  bbox_inches='tight')
        plt.show()

#  同时绘制部分依赖图和个体条件期望图
print("正在绘制部分依赖图和个体条件期望图...")
for  feature  in  top_pdp_features:
        fig,  ax  =  plt.subplots(figsize=(10,  6))
        PartialDependenceDisplay.from_estimator(
                final_model,
                X_train,
                features=[feature],
                kind='both',
                grid_resolution=50,
                ax=ax
        )
        ax.set_title(f'Partial  Dependence  and  ICE  Plot:  {feature}')
        ax.set_xlabel(feature)
        ax.set_ylabel('Partial  Dependence  /  ICE')
        ax.grid(True,  alpha=0.3)
        plt.tight_layout()
        plt.savefig(f'SHAP_PDP_ICE_{feature}.png',  dpi=300,  bbox_inches='tight')
        plt.show()
```



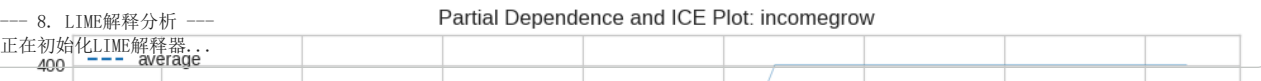
```
正在生成SHAP部分依赖图...
```






```
# ============================================================
# 8. LIME解释分析
# ============================================================
print("\n--- 8. LIME解释分析 ---")

# --- 8.1 初始化LIME解释器 ---
print("正在初始化LIME解释器...")
lime_explainer = lime.lime_tabular.LimeTabularExplainer(
    X_train.values,
    feature_names=X_train.columns,
    class_names=['target'],
    mode='regression',
    discretize_continuous=True,
    random_state=42
)
```

```
正在绘制部分依赖图和个体条件期望图...
```




```
--- 8. LIME解释分析 ---
正在初始化LIME解释器...
```

```
# --- 8.2 LIME单个样本分析 ---
def analyze_sample_with_lime(sample_idx, title_suffix=""):
    """使用LIME分析单个样本"""
    exp = lime_explainer.explain_instance(
        X_test.iloc[sample_idx].values,
        final_model.predict,
        num_features=len(X_test.columns)
    )

    # 获取解释结果
    explanation = exp.as_list()
    features = [item[0] for item in explanation]
    contributions = [item[1] for item in explanation]

    # 可视化
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))
```