In [21]:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import seaborn as sns
sns.set()
```

In [27]:

```
raw_data = pd.read_csv(r'C:\Users\kratikrathi\Downloads\Real estate.csv')
```

In [28]:

```
raw_data
```

Out[28]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1
...	...	...	...	...	...	...	...	...
409	410	2013.000	13.7	4082.01500	0	24.94155	121.50381	15.4
410	411	2012.667	5.6	90.45606	9	24.97433	121.54310	50.0
411	412	2013.250	18.8	390.96960	7	24.97923	121.53986	40.6
412	413	2013.000	8.1	104.81010	5	24.96674	121.54067	52.5
413	414	2013.500	6.5	90.45606	9	24.97433	121.54310	63.9

414 rows × 8 columns

In [29]:

```
raw_data.describe()
```

Out[29]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude
count	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000
mean	207.500000	2013.148971	17.712560	1083.885689	4.094203	24.969030	121.533361
std	119.655756	0.281967	11.392485	1262.109595	2.945562	0.012410	0.015347
min	1.000000	2012.667000	0.000000	23.382840	0.000000	24.932070	121.473530
25%	104.250000	2012.917000	9.025000	289.324800	1.000000	24.963000	121.528085
50%	207.500000	2013.167000	16.100000	492.231300	4.000000	24.971100	121.538630
75%	310.750000	2013.417000	28.150000	1454.279000	6.000000	24.977455	121.543305
max	414.000000	2013.583000	43.800000	6488.021000	10.000000	25.014590	121.566270

In [30]:

```
raw_data.head()
```

Out[30]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

In [31]:

```
data = raw_data.drop(['X1 transaction date','No'],axis = 1)
```

In [32]:

```
data
```

Out[32]:

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	32.0	84.87882	10	24.98298	121.54024	37.9
1	19.5	306.59470	9	24.98034	121.53951	42.2
2	13.3	561.98450	5	24.98746	121.54391	47.3
3	13.3	561.98450	5	24.98746	121.54391	54.8
4	5.0	390.56840	5	24.97937	121.54245	43.1
...	...	...	...	...	...	...
409	13.7	4082.01500	0	24.94155	121.50381	15.4
410	5.6	90.45606	9	24.97433	121.54310	50.0
411	18.8	390.96960	7	24.97923	121.53986	40.6
412	8.1	104.81010	5	24.96674	121.54067	52.5
413	6.5	90.45606	9	24.97433	121.54310	63.9

414 rows × 6 columns

In [33]:

```
data
```

Out[33]:

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	32.0	84.87882	10	24.98298	121.54024	37.9
1	19.5	306.59470	9	24.98034	121.53951	42.2
2	13.3	561.98450	5	24.98746	121.54391	47.3
3	13.3	561.98450	5	24.98746	121.54391	54.8
4	5.0	390.56840	5	24.97937	121.54245	43.1
...	...	...	...	...	...	...
409	13.7	4082.01500	0	24.94155	121.50381	15.4
410	5.6	90.45606	9	24.97433	121.54310	50.0
411	18.8	390.96960	7	24.97923	121.53986	40.6
412	8.1	104.81010	5	24.96674	121.54067	52.5
413	6.5	90.45606	9	24.97433	121.54310	63.9

414 rows × 6 columns

In [34]:

```
data.isnull()
```

Out[34]:

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
409	False	False	False	False	False	False
410	False	False	False	False	False	False
411	False	False	False	False	False	False
412	False	False	False	False	False	False
413	False	False	False	False	False	False

414 rows × 6 columns

In [35]:

```
data.isnull().sum()
```

Out[35]:

```
X2 house age                              0
X3 distance to the nearest MRT station    0
X4 number of convenience stores           0
X5 latitude                               0
X6 longitude                              0
Y house price of unit area                0
dtype: int64
```

In [36]:

```
data.describe(include = 'all')
```

Out[36]:

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
count	414.000000	414.000000	414.000000	414.000000	414.000000	414.000000
mean	17.712560	1083.885689	4.094203	24.969030	121.533361	37.980193
std	11.392485	1262.109595	2.945562	0.012410	0.015347	13.606488
min	0.000000	23.382840	0.000000	24.932070	121.473530	7.600000
25%	9.025000	289.324800	1.000000	24.963000	121.528085	27.700000
50%	16.100000	492.231300	4.000000	24.971100	121.538630	38.450000
75%	28.150000	1454.279000	6.000000	24.977455	121.543305	46.600000
max	43.800000	6488.021000	10.000000	25.014590	121.566270	117.500000

In [37]:

```
f, (ax1,ax2,ax3,ax4,ax5)  = plt.subplots(1,5, sharey = True, figsize = (15,3))
ax1.scatter(data['X6 longitude'], data['Y house price of unit area'])
ax1.set_title('Longitude and Price/area')
ax2.scatter(data['X3 distance to the nearest MRT station'], data['Y house price of unit are
ax2.set_title('Distance station and Price/Area')
ax3.scatter(data['X4 number of convenience stores'], data['Y house price of unit area'])
ax3.set_title(' Convience store and Price/Area')
ax4.scatter(data['X5 latitude'], data['Y house price of unit area'])
ax4.set_title('Laitude and Price/Area')
ax5.scatter(data['X2 house age'], data['Y house price of unit area'])
ax5.set_title('House age and Price/Area')

plt.show()
```

In [38]:

```
log_price = np.log(data['Y house price of unit area'])
```

In [39]:

```
data['log_price'] = log_price
```

In [40]:

```
data
```

Out[40]:

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area	log_price
0	32.0	84.87882	10	24.98298	121.54024	37.9	3.634951
1	19.5	306.59470	9	24.98034	121.53951	42.2	3.742420
2	13.3	561.98450	5	24.98746	121.54391	47.3	3.856510
3	13.3	561.98450	5	24.98746	121.54391	54.8	4.003690
4	5.0	390.56840	5	24.97937	121.54245	43.1	3.763523
...	...	...	...	...	...	...	...
409	13.7	4082.01500	0	24.94155	121.50381	15.4	2.734368
410	5.6	90.45606	9	24.97433	121.54310	50.0	3.912023
411	18.8	390.96960	7	24.97923	121.53986	40.6	3.703768
412	8.1	104.81010	5	24.96674	121.54067	52.5	3.960813
413	6.5	90.45606	9	24.97433	121.54310	63.9	4.157319

414 rows × 7 columns

In [41]:

```
f, (ax1,ax2,ax3,ax4,ax5)  = plt.subplots(1,5, sharey = True, figsize = (15,3))
ax1.scatter(data['X6 longitude'], data['log_price'])
ax1.set_title('Longitude and LogPrice')
ax2.scatter(data['X3 distance to the nearest MRT station'], data['log_price'])
ax2.set_title('Distance station and LogPrice')
ax3.scatter(data['X4 number of convenience stores'], data['log_price'])
ax3.set_title(' Convience store and LogPrice')
ax4.scatter(data['X5 latitude'], data['log_price'])
ax4.set_title('Laitude and LogPrice')
ax5.scatter(data['X2 house age'], data['log_price'])
ax5.set_title('House age and LogPrice')

plt.show()
```

In [63]:

```
data = data.drop(['Y house price of unit area'], axis = 1)
```

In [64]:

```
data
```

Out[64]:

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	log_price
0	32.0	84.87882	10	24.98298	121.54024	3.634951
1	19.5	306.59470	9	24.98034	121.53951	3.742420
2	13.3	561.98450	5	24.98746	121.54391	3.856510
3	13.3	561.98450	5	24.98746	121.54391	4.003690
4	5.0	390.56840	5	24.97937	121.54245	3.763523
...	...	...	...	...	...	...
409	13.7	4082.01500	0	24.94155	121.50381	2.734368
410	5.6	90.45606	9	24.97433	121.54310	3.912023
411	18.8	390.96960	7	24.97923	121.53986	3.703768
412	8.1	104.81010	5	24.96674	121.54067	3.960813
413	6.5	90.45606	9	24.97433	121.54310	4.157319

414 rows × 6 columns

In [65]:

```
data.columns.values
```

Out[65]:

```
array(['X2 house age', 'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
       'log_price'], dtype=object)
```

In [66]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
variables = data[['X2 house age', 'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude']]
vif = pd.DataFrame()
vif["VIF"] = [variance_inflation_factor(variables.values, i) for i in range(variables.shape
vif["features"] = variables.columns
```

In [67]:

```
vif
```

Out[67]:

	VIF	features
0	3.470796e+00	X2 house age
1	2.970976e+00	X3 distance to the nearest MRT station
2	4.732308e+00	X4 number of convenience stores
3	5.913399e+06	X5 latitude
4	5.913114e+06	X6 longitude

In [68]:

```
data_no_multicollinearity = data.drop(['X5 latitude', 'X6 longitude'], axis = 1)
```

In [69]:

```
data_no_multicollinearity
```

Out[69]:

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	log_price
0	32.0	84.87882	10	3.634951
1	19.5	306.59470	9	3.742420
2	13.3	561.98450	5	3.856510
3	13.3	561.98450	5	4.003690
4	5.0	390.56840	5	3.763523
...	...	...	...	...
409	13.7	4082.01500	0	2.734368
410	5.6	90.45606	9	3.912023
411	18.8	390.96960	7	3.703768
412	8.1	104.81010	5	3.960813
413	6.5	90.45606	9	4.157319

414 rows × 4 columns

In [74]:

```
data_1 = data_no_multicollinearity
```

In [144]:

```
data_1
```

Out[144]:

	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	log_price
0	32.00	84.88	10	3.63
1	19.50	306.59	9	3.74
2	13.30	561.98	5	3.86
3	13.30	561.98	5	4.00
4	5.00	390.57	5	3.76
...	...	...	...	...
409	13.70	4082.01	0	2.73
410	5.60	90.46	9	3.91
411	18.80	390.97	7	3.70
412	8.10	104.81	5	3.96
413	6.50	90.46	9	4.16

414 rows × 4 columns

In [145]:

```
y = data_1['log_price']
```

In [150]:

```
x1 = data_1[['X2 house age','X3 distance to the nearest MRT station','X4 number of convenie
```

In [155]:

```
x = sm.add_constant(x1)
```

```
C:\adobeTemp\Anaconda\lib\site-packages\statsmodels\tsa\tsatools.py:142: Fut
ureWarning: In a future version of pandas all arguments of concat except for
the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

In [156]:

```
results = sm.OLS(y,x).fit()
```

In [158]:

```
results.summary()
```

Out[158]:

OLS Regression Results

Dep. Variable:	log_price	**R-squared:**	0.634
Model:	OLS	**Adj. R-squared:**	0.631
Method:	Least Squares	**F-statistic:**	236.4
Date:	Thu, 03 Mar 2022	**Prob (F-statistic):**	5.05e-89
Time:	21:35:08	**Log-Likelihood:**	8.1469
No. Observations:	414	**AIC:**	-8.294
Df Residuals:	410	**BIC:**	7.810
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	3.7470	0.036	105.017	0.000	3.677	3.817
X2 house age	-0.0064	0.001	-6.206	0.000	-0.008	-0.004
X3 distance to the nearest MRT station	-0.0002	1.17e-05	-15.940	0.000	-0.000	-0.000
X4 number of convenience stores	0.0330	0.005	6.586	0.000	0.023	0.043

Omnibus:	94.173	**Durbin-Watson:**	2.092
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1270.796
Skew:	-0.519	**Prob(JB):**	1.12e-276
Kurtosis:	11.520	**Cond. No.**	5.09e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.09e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

In [139]:

```
targets = data_1['log_price']
inputs = data_1.drop(['log_price'], axis = 1)
```

In [83]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(inputs)
inputs_scaled = scaler.transform(inputs)
```

In [84]:

```
from sklearn.model_selection import train_test_split
```

In [85]:

```
x_train,x_test,y_train,y_test = train_test_split(inputs_scaled,targets, test_size = 0.2, ra
```

In [86]:

```
reg = LinearRegression()
```

In [88]:

```
reg.fit(x_train, y_train)
```

Out[88]:

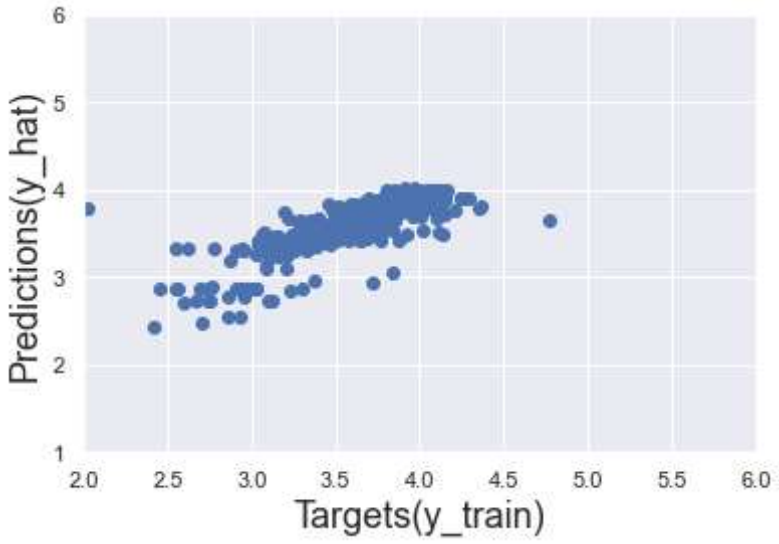
```
LinearRegression()
```

In [89]:

```
y_hat = reg.predict(x_train)
```

In [97]:

```
plt.scatter(y_train,y_hat)
plt.xlabel('Targets(y_train)', size = 18)
plt.ylabel('Predictions(y_hat)', size = 18)
plt.xlim(2,6)
plt.ylim(1,6)
plt.show()
```



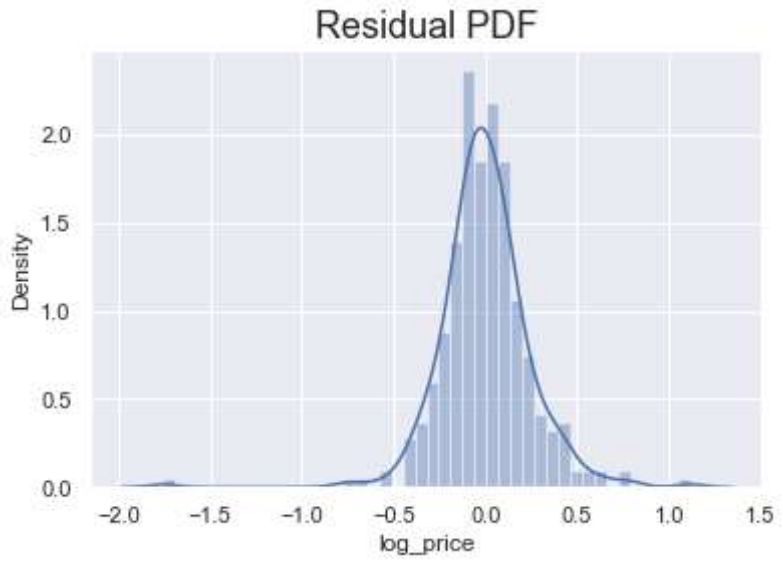
In [100]:

```
sns.distplot(y_train - y_hat)
plt.title("Residual PDF", size = 18)
reg.score(x_train,y_train)
```

```
C:\adobeTemp\Anaconda\lib\site-packages\seaborn\distributions.py:2619: Futur
eWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
```

Out[100]:

```
0.6366261480799151
```



In [101]:

```
reg.intercept_
```

Out[101]:

```
3.5663902543755923
```

In [102]:

```
reg.coef_
```

Out[102]:

```
array([-0.06753142, -0.23787347,  0.10756863])
```

In [104]:

```
reg_summary = pd.DataFrame(inputs.columns.values, columns = ["Features"])
```

In [105]:

```
reg_summary ["Weights"] = reg.coef_
```

In [106]:

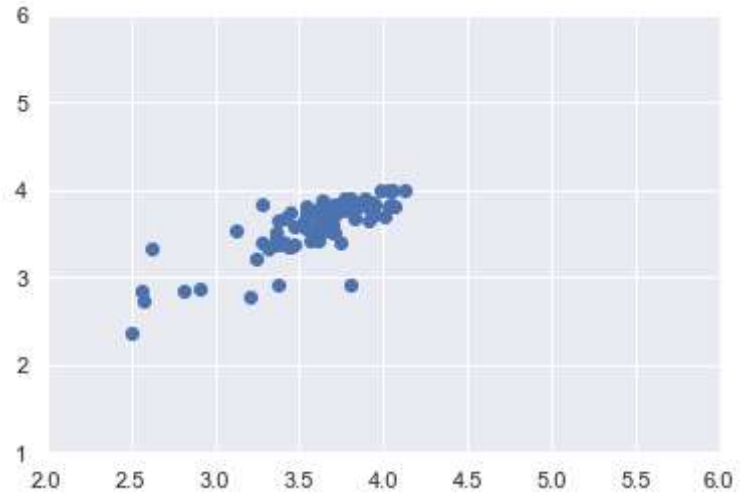
```
reg_summary
```

Out[106]:

	Features	Weights
0	X2 house age	-0.067531
1	X3 distance to the nearest MRT station	-0.237873
2	X4 number of convenience stores	0.107569

In [107]:

```
y_hat_test = reg.predict(x_test)
plt.scatter(y_test,y_hat_test)
plt.xlim(2,6)
plt.ylim(1,6)
plt.show()
```



In [115]:

```
df_pf = pd.DataFrame(np.exp(y_hat_test), columns = ['Predictions'])
```

In [116]:

```
df_pf
```

Out[116]:

	Predictions
0	27.753585
1	40.099154
2	29.612403
3	29.639264
4	49.534353
...	...
78	10.724558
79	49.770262
80	17.349917
81	35.961754
82	48.194329

83 rows × 1 columns

In [113]:

```
df_pf['Target'] = np.exp(y_test)
```

In [114]:

```
df_pf
```

Out[114]:

	Predictions	Target
0	27.753585	27.7
1	40.099154	37.9
2	29.612403	30.5
3	29.639264	26.5
4	49.534353	49.0
...	...	...
78	10.724558	12.2
79	49.770262	43.2
80	17.349917	13.0
81	35.961754	34.1
82	48.194329	45.2

83 rows × 2 columns

In [112]:

```
y_test = y_test.reset_index(drop = True)
```

In [117]:

```
df_pf['Target'] = np.exp(y_test)
```

In [118]:

```
df_pf
```

Out[118]:

	Predictions	Target
0	27.753585	27.7
1	40.099154	37.9
2	29.612403	30.5
3	29.639264	26.5
4	49.534353	49.0
...	...	...
78	10.724558	12.2
79	49.770262	43.2
80	17.349917	13.0
81	35.961754	34.1
82	48.194329	45.2

83 rows × 2 columns

In [120]:

```
df_pf['Residual'] = df_pf['Target'] - df_pf['Predictions']
```

In [121]:

```
df_pf['Difference %'] = np.absolute(df_pf['Residual']/df_pf['Target'])
```

In [122]:

```
df_pf
```

Out[122]:

	Predictions	Target	Residual	Difference %
0	27.753585	27.7	-0.053585	0.001934
1	40.099154	37.9	-2.199154	0.058025
2	29.612403	30.5	0.887597	0.029102
3	29.639264	26.5	-3.139264	0.118463
4	49.534353	49.0	-0.534353	0.010905
...	...	...	...	...
78	10.724558	12.2	1.475442	0.120938
79	49.770262	43.2	-6.570262	0.152089
80	17.349917	13.0	-4.349917	0.334609
81	35.961754	34.1	-1.861754	0.054597
82	48.194329	45.2	-2.994329	0.066246

83 rows × 4 columns

In [123]:

```
df_pf.describe()
```

Out[123]:

	Predictions	Target	Residual	Difference %
count	83.000000	83.000000	83.000000	83.000000
mean	37.675022	37.961446	0.286423	0.146888
std	9.792016	10.818456	7.057230	0.165541
min	10.724558	12.200000	-19.497071	0.000694
25%	32.147566	31.300000	-3.112769	0.043177
50%	39.632674	37.900000	-0.066895	0.094113
75%	44.543073	44.000000	3.502073	0.183951
max	54.845250	62.200000	26.639877	1.017384

In [124]:

```
df_pf.sort_values(by = ['Difference %'])
```

Out[124]:

	Predictions	Target	Residual	Difference %
69	40.971539	41.0	0.028461	0.000694
0	27.753585	27.7	-0.053585	0.001934
54	29.366895	29.3	-0.066895	0.002283
70	40.524190	40.3	-0.224190	0.005563
49	46.669527	47.0	0.330473	0.007031
...	...	...	...	...
37	18.622592	29.3	10.677408	0.364417
53	34.463343	22.8	-11.663343	0.511550
18	18.460123	45.1	26.639877	0.590685
67	45.997071	26.5	-19.497071	0.735739
51	27.638166	13.7	-13.938166	1.017384

83 rows × 4 columns

In [126]:

```
pd.options.display.max_rows = 83
```

In [127]:

```
pd.set_option('display.float_format',lambda x:'%.2f'%x)
```

In [128]:

```
df_pf.sort_values(by = ['Difference %'])
```

Out[128]:

	Predictions	Target	Residual	Difference %
69	40.97	41.00	0.03	0.00
0	27.75	27.70	-0.05	0.00
54	29.37	29.30	-0.07	0.00
70	40.52	40.30	-0.22	0.01
49	46.67	47.00	0.33	0.01
36	34.85	34.60	-0.25	0.01
4	49.53	49.00	-0.53	0.01
64	43.66	44.30	0.64	0.01
58	43.24	42.40	-0.84	0.02
46	35.83	36.70	0.87	0.02
41	25.08	25.70	0.62	0.02

In [133]:

```
print(df_pf)
```

```
    Predictions  Target  Residual  Difference %
0         27.75   27.70     -0.05          0.00
1         40.10   37.90     -2.20          0.06
2         29.61   30.50      0.89          0.03
3         29.64   26.50     -3.14          0.12
4         49.53   49.00     -0.53          0.01
5         45.53   56.20     10.67          0.19
6         54.78   56.30      1.52          0.03
7         40.10   55.30     15.20          0.27
8         17.41   16.70     -0.71          0.04
9         41.76   34.60     -7.16          0.21
10        45.87   51.70      5.83          0.11
11        36.29   37.40      1.11          0.03
12        38.30   50.20     11.90          0.24
13        30.27   37.00      6.73          0.18
14        43.78   40.90     -2.88          0.07
15        38.57   29.30     -9.27          0.32
16        33.52   40.60      7.08          0.17
17        41.99   38.90     -3.09          0.08
18        18.46   45.10     26.64          0.59
19        28.79   31.30      2.51          0.08
20        45.39   58.80     13.41          0.23
21        41.26   36.50     -4.76          0.13
22        35.44   36.50      1.06          0.03
23        46.35   40.90     -5.45          0.13
24        17.44   18.30      0.86          0.05
25        40.83   38.10     -2.73          0.07
26        38.28   34.10     -4.18          0.12
27        46.48   56.80     10.32          0.18
28        44.01   37.20     -6.81          0.18
29        33.34   36.80      3.46          0.09
30        31.03   28.90     -2.13          0.07
31        30.47   35.60      5.13          0.14
32        42.06   40.50     -1.56          0.04
33        54.65   58.10      3.45          0.06
34        36.76   40.30      3.54          0.09
35        30.20   42.30     12.10          0.29
36        34.85   34.60     -0.25          0.01
37        18.62   29.30     10.68          0.36
38        41.99   31.30    -10.69          0.34
39        39.55   30.50     -9.05          0.30
40        54.62   62.20      7.58          0.12
41        25.08   25.70      0.62          0.02
42        33.26   40.80      7.54          0.18
43        47.81   51.60      3.79          0.07
44        46.67   38.40     -8.27          0.22
45        42.31   40.20     -2.11          0.05
46        35.83   36.70      0.87          0.02
47        42.06   39.30     -2.76          0.07
48        28.63   31.10      2.47          0.08
49        46.67   47.00      0.33          0.01
50        33.50   28.90     -4.60          0.16
51        27.64   13.70    -13.94          1.02
52        39.63   46.00      6.37          0.14
53        34.46   22.80    -11.66          0.51
54        29.37   29.30     -0.07          0.00
55        29.47   32.10      2.63          0.08
```

56	42.58	41.50	-1.08	0.03
57	15.45	13.20	-2.25	0.17
58	43.24	42.40	-0.84	0.02
59	49.53	45.10	-4.43	0.10
60	54.85	53.50	-1.35	0.03
61	43.03	46.10	3.07	0.07
62	48.72	37.90	-10.82	0.29
63	35.70	32.20	-3.50	0.11
64	43.66	44.30	0.64	0.01
65	40.90	52.20	11.30	0.22
66	38.40	37.40	-1.00	0.03
67	46.00	26.50	-19.50	0.74
68	44.23	38.60	-5.63	0.15
69	40.97	41.00	0.03	0.00
70	40.52	40.30	-0.22	0.01
71	45.62	43.70	-1.92	0.04
72	16.26	24.70	8.44	0.34
73	34.70	38.80	4.10	0.11
74	36.24	37.70	1.46	0.04
75	35.86	37.80	1.94	0.05
76	45.65	34.40	-11.25	0.33
77	44.85	49.50	4.65	0.09
78	10.72	12.20	1.48	0.12
79	49.77	43.20	-6.57	0.15
80	17.35	13.00	-4.35	0.33
81	35.96	34.10	-1.86	0.05
82	48.19	45.20	-2.99	0.07