Bihan Qian

Linear and Robust Regression with Assumption Tests

Consider the Boston dataset, in R library MASS, on Housing Values in Suburbs of Boston

Q1

Fit a multiple linear regression model to predict medv (median value of owner-occupied homes in $1000s) using the following set of predictors:

- crim per capita crime rate by town.
- zn proportion of residential land zoned for lots over 25,000 sq.ft.
- indus proportion of non-retail business acres per town.
- nox nitrogen oxides concentration (parts per 10 million).
- rm average number of rooms per dwelling.
- age proportion of owner-occupied units built prior to 1940.
- tax full-value property-tax rate per $10,000.

Import dataset Boston

In [1]:
```
import statsmodels.api as sm
import statsmodels.stats.api as sms
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import scipy.stats as stats
from statsmodels.stats.diagnostic import het_breuschpagan
```

In [2]:
```
Boston = sm.datasets.get_rdataset('Boston','MASS')
```

In [3]:
```
Boston.data
```

Out[3]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

In [4]:

```
boston = pd.DataFrame(Boston.data)
boston.shape
```

Out[4]:

```
(506, 14)
```

Fit mutiple linear regression

In [5]:

```
X = boston[['crim', 'zn', 'indus', 'nox', 'rm', 'age', 'tax']]
y = boston['medv']
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
residuals = model.resid
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.582
Model:                            OLS   Adj. R-squared:                  0.576
Method:                 Least Squares   F-statistic:                     98.99
Date:                Wed, 27 Sep 2023   Prob (F-statistic):           4.02e-90
Time:                        09:17:00   Log-Likelihood:                -1619.6
No. Observations:                 506   AIC:                             3255.
Df Residuals:                     498   BIC:                             3289.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -19.6153      3.221     -6.089      0.000     -25.945     -13.286
crim          -0.1325      0.038     -3.444      0.001      -0.208      -0.057
zn             0.0221      0.015      1.491      0.137      -0.007       0.051
indus         -0.0150      0.072     -0.207      0.836      -0.157       0.127
nox            0.0106      4.230      0.003      0.998      -8.301       8.322
rm             7.6065      0.418     18.179      0.000       6.784       8.429
age           -0.0232      0.015     -1.558      0.120      -0.052       0.006
tax           -0.0090      0.003     -3.384      0.001      -0.014      -0.004
==============================================================================
Omnibus:                      280.244   Durbin-Watson:                   0.729
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2717.977
Skew:                           2.238   Prob(JB):                         0.00
Kurtosis:                      13.435   Cond. No.                     7.72e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 7.72e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

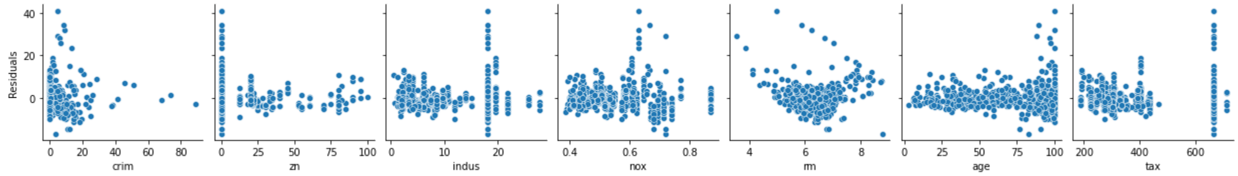
Q2

State and assess the validity of the underlying assumptions, and suggest remedial measures in case of violations of any of the underlying assumptions

- Linearity/functional form
- Normality
- Homoscedasticity
- Uncorrelated error

Linearity

In [6]:
```
# Plot residual vs X
residuals_df = pd.concat([pd.Series(residuals, name='Residuals'), X], axis=1)
sns.pairplot(residuals_df, y_vars=['Residuals'], x_vars=['crim', 'zn', 'indus', 'nox',
plt.show()
```



In [ ]:
```
# Plot residual vs fitted value
centered aroud y=0, same distribution
```

If the relationship between X and Y is linear, we would expect the residuals to be randomly scattered around the zero line in the residual plot.

Report outliers for all explantory variables in our regression. Ignoring the outlier problem, nox is suggested little linearity. For all other variables, use transformations.

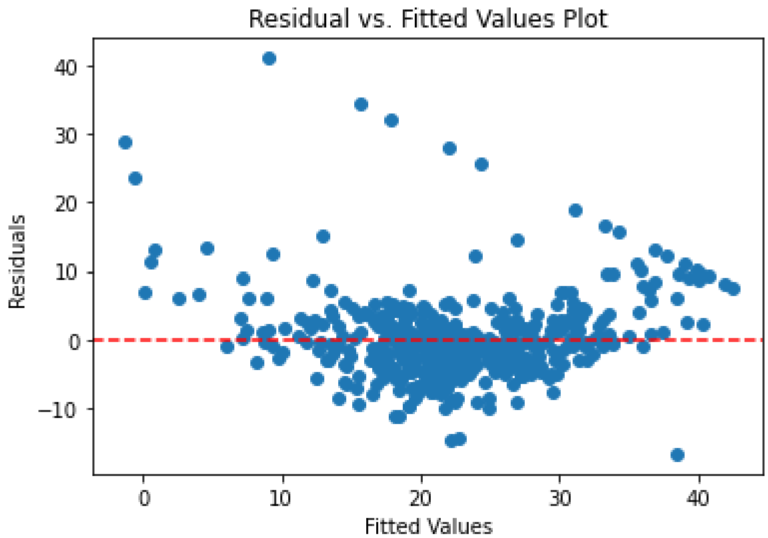
In [7]:
```
# dictionary to R-squared
r_squared_values = {}

X1 = boston[['crim', 'zn', 'indus', 'nox', 'rm', 'age', 'tax']]
for column in X1.columns:
    if column != 'medv':
        array = X1[[column]]
        model = LinearRegression()
        model.fit(array, y)
        r_squared = model.score(array, y)
        r_squared_values[column] = r_squared
for column, r_squared in r_squared_values.items():
    print(f"R-squared for {column}: {r_squared}")
```

```
R-squared for crim: 0.15078046904975717
R-squared for zn: 0.12992084489428946
R-squared for indus: 0.2339900304444752
R-squared for nox: 0.182603042501699
R-squared for rm: 0.48352545599133423
R-squared for age: 0.14209474407780465
R-squared for tax: 0.2195259210442192
```

R-squared suggests linear is only a moderate good fit for rm and non-linearity for all other explantory variables.

In [26]:
```
fitted_values = model.fittedvalues
plt.scatter(fitted_values, residuals)
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.title("Residual vs. Fitted Values Plot")
plt.axhline(y=0, color='r', linestyle='--')
plt.show()
```



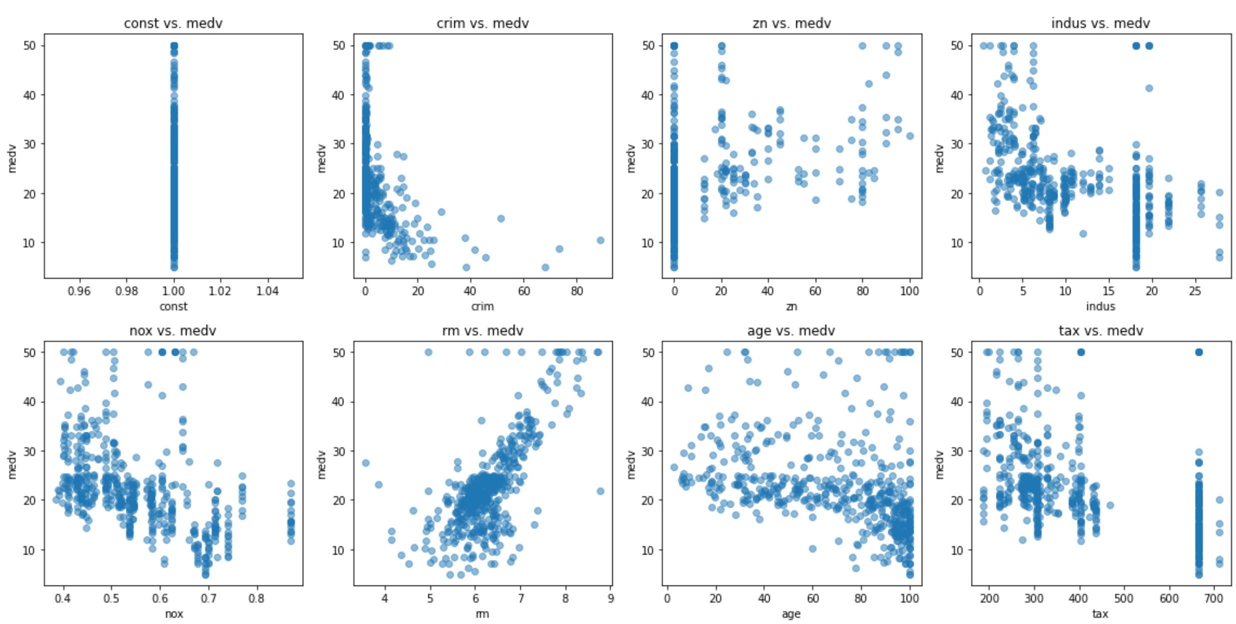
In [8]:
```
import matplotlib.pyplot as plt
import pandas as pd

fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(16, 8))

axes = axes.flatten()

for i, predictor in enumerate(X):
    ax = axes[i]
    ax.scatter(X[predictor], y, alpha=0.5)
    ax.set_title(f'{predictor} vs. medv')
    ax.set_xlabel(predictor)
    ax.set_ylabel('medv')

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



State:

- We assume that there exist linear relationships between each explantory variables(crim, zn, indus, nox, rm, age, tax) and response variable (medv).

Access:

- Overall, no linearity.
- crim: Transform by log(crim) because variability increases as crim increases. Report influential points and outliers.
- indus: Use non-linear to fit with.
- nox: Transform by adding square.
- age: Transform by log(age).

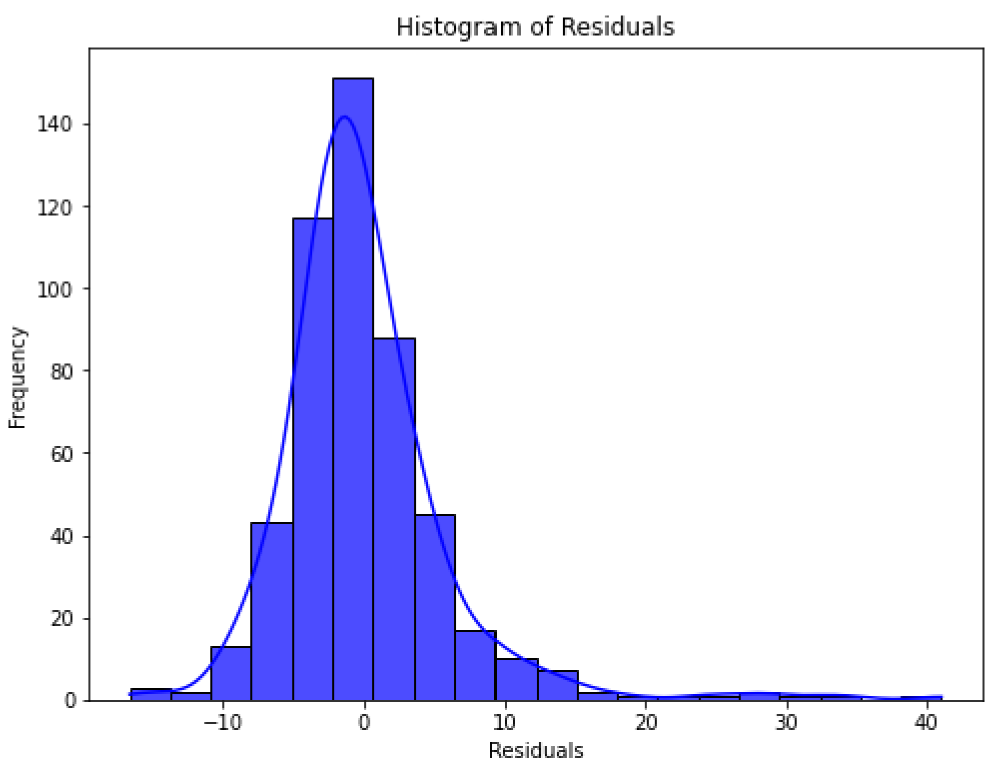
Suggest:

- Take transformations (log, square, inverse, square root), use non-linear model fit, try adding other explantory variables.

Normality

In [9]:

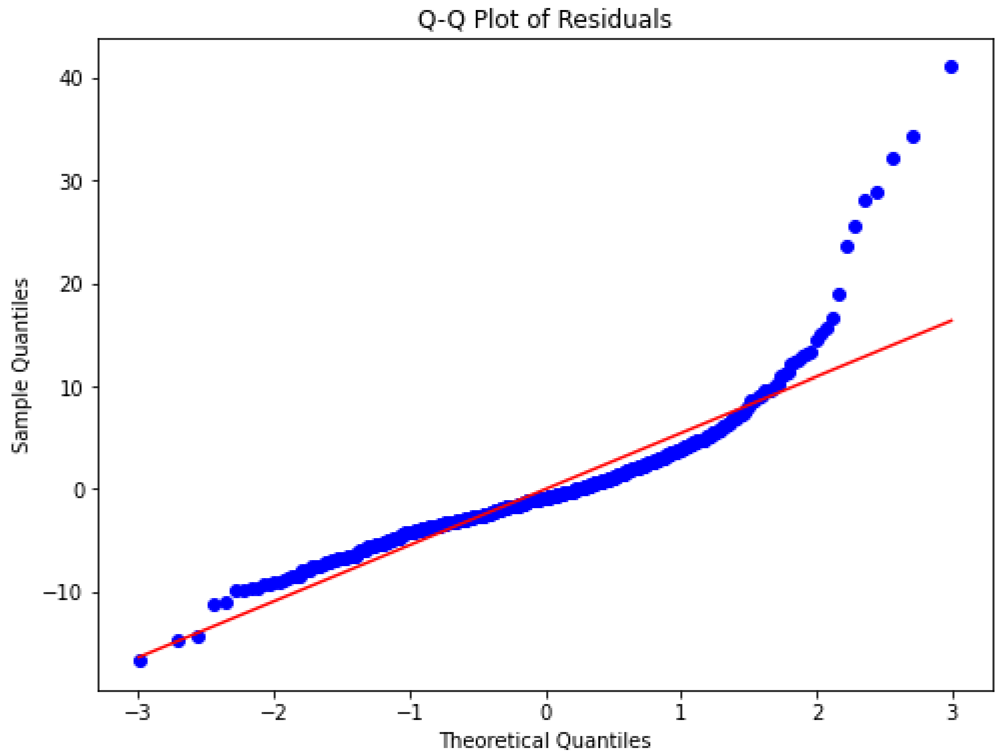
```python
# histogram of residual
plt.figure(figsize=(8, 6))
sns.histplot(residuals, kde=True, bins=20, color='blue', alpha=0.7)
plt.title('Histogram of Residuals')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.show()
```



In [10]:

```python
# qqnorm
plt.figure(figsize=(8, 6))
stats.probplot(residuals, dist="norm", plot=plt)
plt.title('Q-Q Plot of Residuals')
plt.xlabel('Theoretical Quantiles')
```

```
plt.ylabel('Sample Quantiles')
plt.show()
```



In [11]:

```
# Kolmogorov-Smirnov test
ks_statistic, ks_p_value = stats.kstest(residuals, 'norm')

# Shapiro-Wilk test
shapiro_statistic, shapiro_p_value = stats.shapiro(residuals)


print("Kolmogorov-Smirnov Test:")
print("KS Statistic:", ks_statistic)
print("KS p-value:", ks_p_value)
print("\nShapiro-Wilk Test:")
print("Shapiro Statistic:", shapiro_statistic)
print("Shapiro p-value:", shapiro_p_value)
```

```
Kolmogorov-Smirnov Test:
KS Statistic: 0.3619649071450999
KS p-value: 6.761831453311945e-60

Shapiro-Wilk Test:
Shapiro Statistic: 0.8394485712051392
Shapiro p-value: 3.2835149995200612e-22
```

State:

- We assume that the residual are normally distributed.

Access:

- From residual histogram and qqnorm plot, we can observe a long right tail and points that do not fall on the 45-degree reference line of qqnorm plot. Furthermore, the two normality tests with extremely small p-values provide convicing evidence that our data does not follow a normal distribution.

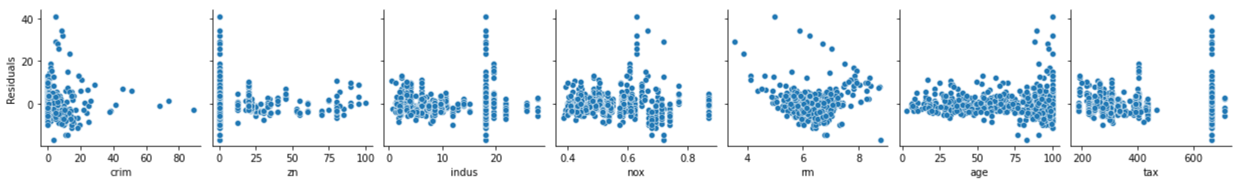
Suggest:

- The remedies are trying transformations for each variables as I described in the result of linearity part, or use other robust regression methods toward outliers, like L1 regression, LMS regression, LTS regression, and M estimate of robust regression.

Homoscedasticity

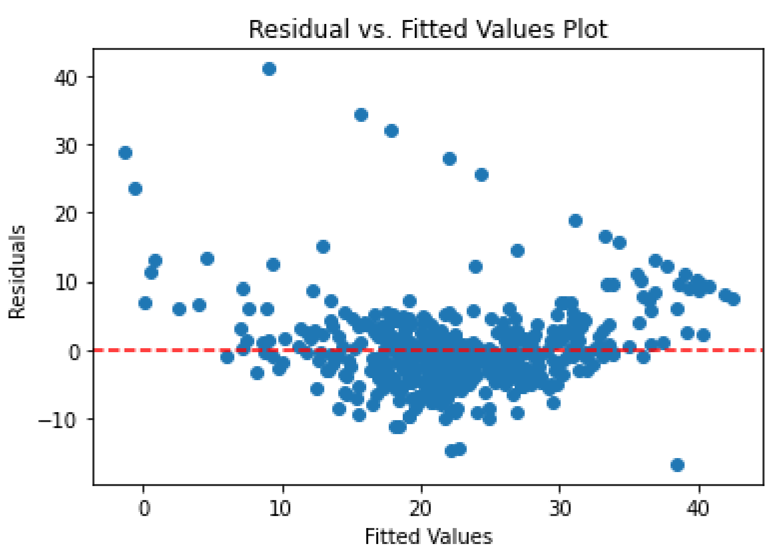
In [12]:

```
# Plot residual vs X
residuals_df = pd.concat([pd.Series(residuals, name='Residuals'), X], axis=1)
sns.pairplot(residuals_df, y_vars=['Residuals'], x_vars=['crim', 'zn', 'indus', 'nox',
plt.show()
```



In [27]:

```
fitted_values = model.fittedvalues
plt.scatter(fitted_values, residuals)
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.title("Residual vs. Fitted Values Plot")
plt.axhline(y=0, color='r', linestyle='--')
plt.show()
```



In [13]:

```
# variance test
test_result = het_breuschpagan(residuals, X)
LM_statistic = test_result[0]
LM_p_value = test_result[1]
F_statistic = test_result[2]
F_p_value = test_result[3]

# Print the results
```

```
print("Breusch-Pagan Test Results:")
print(f"LM Statistic: {LM_statistic}")
print(f"LM p-value: {LM_p_value}")
print(f"F Statistic: {F_statistic}")
print(f"F p-value: {F_p_value}")
```

```
Breusch-Pagan Test Results:
LM Statistic: 31.24209049360534
LM p-value: 5.609580584933l3e-05
F Statistic: 4.681652556650759
F p-value: 4.244806010810502e-05
```

State:

- We assume that the variances of the error terms (residuals) are the same for all values of the independent variables.

Assess:

- Homoscedasticity assumption is not satisfied.
- In resiudal plot, we cannot see the spread or variability of residuals is consistent across the range of values of each predictor.
- In the variance test, the null hypothesis is assumed the variances of the residuals are equal. Our variance test provides strong evidence that the variances of the resiudal are not equal(two-sided p-value=5.60958058493313e-05).

Suggest:

- The remedies are transformations or building WLS incorporating the variance structure to the model.

Uncorrelated error

In [18]:
```
durbin_watson_test = sms.durbin_watson(residuals)
print("Durbin-Watson Statistic:", durbin_watson_test)
```

```
Durbin-Watson Statistic: 0.7288349004473201
```

State:

- We assume that errors (residuals) resulting from the linear regression model are not correlated with each other.

Access:

- We obtain a Durbin-Watson statistic of 0.729 is significantly less than 2, indicating the presence of positive serial correlation in the residuals. Errors in our model are positively correlated with each other, violating the assumption of independent errors.
- By the design of study, taking clustering housing from one city and collecting data cross geographic locations of differnt town could introduce correlation in the data.

Suggest:

- The solution are tranformations by Cochrane-Orcutt Procedure or using GEE generalized estimating equation to incorporate correlation structure.

Q3

Repeat (1) using Least Median of Squares Regression and compare the results with those obtained in (1).

In [25]:

```
# model2 = sm.RLM(y, X, M=sm.robust.norms.TrimmedMean()).fit()
# print(model2.summary())
```

In [24]:

```
# because no median least square norm in statmodel package of python, I run it in R in
from IPython.display import Image
image_file_path = f"C:/Users/11139/Desktop/STAT5391/hw3_pict.png"
Image(filename=image_file_path)
```

Out[24]:

```
```{r}
library(MASS) #bihan Q
summary(Boston)
lqs(formula = medv ~ crim+zn+indus+nox+rm+age+tax, data = Boston, method='lms')
```
```

```
Call:
lqs.formula(formula = medv ~ crim + zn + indus + nox + rm + age +
    tax, data = Boston, method = "lms")

Coefficients:
(Intercept)         crim           zn        indus          nox           rm
 -14.321593    -0.290920    -0.019262     0.157838    -4.965621     6.792077
        age          tax
  -0.049526    -0.005307

Scale estimates 3.647 3.611
```

In [16]:

```
model = sm.OLS(y, X).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.582
Model:                            OLS   Adj. R-squared:                  0.576
Method:                 Least Squares   F-statistic:                     98.99
Date:                Wed, 27 Sep 2023   Prob (F-statistic):           4.02e-90
Time:                        09:17:04   Log-Likelihood:                -1619.6
No. Observations:                 506   AIC:                             3255.
Df Residuals:                     498   BIC:                             3289.
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -19.6153      3.221     -6.089      0.000     -25.945     -13.286
crim          -0.1325      0.038     -3.444      0.001      -0.208      -0.057
zn             0.0221      0.015      1.491      0.137      -0.007       0.051
indus         -0.0150      0.072     -0.207      0.836      -0.157       0.127
nox            0.0106      4.230      0.003      0.998      -8.301       8.322
rm             7.6065      0.418     18.179      0.000       6.784       8.429
age           -0.0232      0.015     -1.558      0.120      -0.052       0.006
tax           -0.0090      0.003     -3.384      0.001      -0.014      -0.004
==============================================================================
Omnibus:                      280.244   Durbin-Watson:                   0.729
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2717.977
Skew:                           2.238   Prob(JB):                         0.00
Kurtosis:                      13.435   Cond. No.                     7.72e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 7.72e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

From the assumptions: LMS regression, being a robust method, is less sensitive to violations of these assumptions, especially outliers. It produces a better outcome than the multi-linear regression we performed at the beginning.

The results of LMS are different in every time we run it. The screenshot we have above are a regression result with the intercept closest to the MLR. However, there is a common place.

Even we get multiple LMS regression results, **nox** always have the largest difference from the coefficient in MLR. **nox** changed from 0.0106 (MLR) to -4.965621 (LMS).

This change can be attributed to the fact that, as we can see from the residual plot, outliers are a major concern for nitrogen oxides concentration (nox). When the regression is more robust to outliers, the nitrogen oxides concentration (nox) begins to show larger impacts on the median value of owner-occupied homes (medv).