Machine learning - Assignment 2 + 3 - Simple and multiple linear regression (| + ||)

Author: Kemal Cikota

Course: Machine learning

Introduction

In this assignment, we were tasked with performing simple linear regression on different variables in the Boston housing market dataset. We would then extrapolate this linear regression method on several variables and in the end combine all of the variables in the same linear regression model. Conclusions and interpretations of the outputs will also be made in this notebook to make it clearer as to how and why certain calculations are computed and what their results indicate.

A. Conceptual Questions

1. Which answer is correct and why?

In this part of the assignment, we have to decide which of the following statements is true. The first statement is as follows:

- For a fixed value of IQ and GPA, high school graduates earn more, on average, than college graduates.

For this task, i use a linear regression in order to fit the predictors to their coefficients and from this deduce the response variable which will be the starting salary after graduation in thousands of dollars.

The raw model (simplified) for the starting salary for High school grads is as follows:

$$\hat{Y}_{HighSchool} = 50 + 20X_1 + 0.07X_2 + 35 * X_3 + 0.01X_4 - 10 * X_5 = 50 + 20X_1 + 0.07$$

This model is given by multiplying the coefficients to their predictor variables. From this, notice that the X_3 term is eliminated because the 'level' is 0 and the X_5 term is eliminated because the interaction between GPA and Level can be expressed as $X_1 * X_3$ which is also 0. This is the only simplification that has been done just to avoid confusion

The raw model (simplified) for the starting salary for college grads is as follows:

$$\hat{Y}_{College} = 50 + 20X_1 + 0.07X_2 + 35 * X_3 + 0.01X_4 - 10 * X_5 = 50 + 20X_1 + 0.07X_2$$

For this model i only simplified by substituting the X_4 and X_5 as the interactions between their variables which will give me a cleaner funciton to evaluate since now we

only have the response variable as an expression of GPA and IQ which is nice (except for X_3 which is 1 so it's easy to deal with)!

Now, even though the statement is clear it is still up to interpretation what constitutes earning more "on average". I interpreted this as taking an average IQ and GPA as the predictor variables for the model. For this i used $GPA = X_1 = 3$ and $IQ = X_2 = 100$ as this felt "average" enough for this task.

This gave me the following result when inputted in the models:

$$\hat{Y}_{HighSchool} = 50 + 20X_1 + 0.07X_2 + 0.01(X_1 * X_2) = 50 + 20(3) + 0.07(100) + 0.01($$

$$\hat{Y}_{College} = 50 + 20X_1 + 0.07X_2 + 35 + 0.01(X_1 * X_2) - 10 * (X_1 * X_3) = 50 + 20(3) +$$

This concludes that on average (average IQ and GPA), college students do have a higher entering salary then HS grads. So the statement is **NOT CORRECT**.

- For a fixed value of IQ and GPA, college graduates earn more, on average, than high school graduates.

This is essentially the reverse statement as the last one. So we can instantly see that this statement is **CORRECT**.

- For a fixed value of IQ and GPA, high school graduates earn more, on average, than college graduates provided that the GPA is high enough.

For this task. I wanted to find if there exists a threshold for the GPA that would make it so that for a fixed GPA and IQ for a given GPA level, the HS grads would have a higher entering salary than the college grads. For this i found a pretty cool mathematicall expression where this threshold can be found easily with simple algebraic rules. This expression was found when trying to find the difference between the models of $\hat{Y}_{HighSchool}$ and $\hat{Y}_{College}$. The derivation of this is as follows:

$$\hat{Y}_{HighSchool} - \hat{Y}_{College} = 50 + 20X_1 + 0.07X_2 + 0.01(X_1 * X_2) - (50 + 20X_1 + 0.07X_2$$

This expression will tell us by how much the entering salary between the HS and college grads is. If the result from this equation is positive. it means that the HS grads have a higher entering salary but if its negative, it means that the college grads would have a higher entering salary. We can test this out for the same GPA as we did for task *i* and *ii* which would give $X_1 = 3$.

$$\hat{Y}_{HighSchool} - \hat{Y}_{College} = -35 + 10 * X_1 = -35 + (10 * 3) = -5$$

This yielded -5 which confirms that for this GPA level, a college student would make $5000 more since the difference is negative (because it's in 1000's of dollars).

Because most of the terms cancel out, we can from that equation also find the threshold where the high school grads would have a higher entering salary. This can be done by solving for X_1 when the difference is 0.

$$\hat{Y}_{HighSchool} - \hat{Y}_{College} = -35 + 10 * X_1 = 0 \leftrightarrow X_1 = 3.5$$

This means that the threshold for when HS grads would have a higher entering salary is $GPA = X_1 = 3.5$. So for all GPA's that are over 3.5. the entering salary would be higher for HS grads then college grads. To answer the statement. **YES, IT IS CORRECT**.

- For a fixed value of IQ and GPA, college graduates earn more, on average, than high school graduates provided that the GPA is high enough.

This is the reverse of statement *iii*. So for GPA's that are smaller than 3.5. the starting salary is **HIGHER** for college grads. This depends a bit on how one would interpret the question but for GPA's that are "high enough", a college grad would actually earn less. This means that the statement **IS NOT CORRECT**.

2. Predict the salary of a college graduate with IQ of 110 and GPA of 4.0

In order to predict the salary of a college student with IQ of 110 and GPA of 4 we can use the raw regression model from statement *i* and just substitute the variables in order to get the salary.

$$\hat{Y}_{College} = 50 + 20X_1 + 0.07X_2 + 35 * X_3 + 0.01X_4 - 10 * X_5 = 50 + 20X_1 + 0.07X_2$$

This gives us a starting salary for a college student with a GPA of 4 and IQ of 110 of $137.100.

3. True or false: Since the coefficient for the GPA/IQ interaction term is very small, there is very little evidence of an interaction effect. Justify your answer.

The coefficient for the GPA/IQ interaction ($\hat{\beta}_4 = 0.01$) is small when comparead to the other coefficients but in order to be certain about it's statistical significance we would have to compute the P-value or confidence interval for that coefficient and in order to do that we would have to have more information available about the standard errors, sample size, variance and degrees of the dataset that theese coefficients have been taken from. Since this data isn't given to us we can't draw a conclusion of it's statistical significance. We can only draw some vague conclusions and nuances from its practical implication.

For example, if we take the interaction between GPA/IQ from task 2 where $0.01 * X_4 = 0.01X_1 * X_2 = 4.4$. This means that the interaction between GPA/IQ in our model contributes $4.400 to the total salary of$171.100 which also means that the interaction between GPA/IQ contributes roughly 2.6% to the overall salary in our model. Depending on who you might ask, this could be seen as a significant amount of money from a practical perspective. But from a statistical perspective, we can not make a definitive conclusion of whether or not this is true or false.

ASSIGNMENT 2

Load the data and get an overview of the data

Like in the last assignment, we need to import all of our libraries, load the dataset and call some functions/commands in order to get an overview of the data. At this stage, a

lot of code was reused from my A1 submission so thats why some commments and lines of code will look similar.

In [74]:
```
import pandas as pd # Never coded in R before but this seems to be the equivalen
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import statsmodels.api as sm
from statsmodels.stats.anova import anova_lm


# load boston.csv
boston = pd.read_csv('Boston.csv')

# Set pandas option to display all columns
pd.set_option('display.max_columns', None)
```

Once the dataset is loaded, we can display the number of predictors (variables/columns) and their names.

In [75]:
```
# This will print '15' and not '14' because it counts the first 'empty' column
numFeatures = boston.shape[1]
print(numFeatures)

featureNames = boston.columns.tolist()
print(featureNames, end="\n\n")
```

```
15
['Unnamed: 0', 'crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad',
'tax', 'ptratio', 'black', 'lstat', 'medv']
```

We can now print a statistic summary of the whole dataset, using the 'describe' function which is similar to 'summary' in R.

In [76]:
```
print(boston.describe(), end="\n\n")
```

```
       Unnamed: 0        crim          zn       indus        chas         nox  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean   253.500000    3.613524   11.363636   11.136779    0.069170    0.554695
std    146.213884    8.601545   23.322453    6.860353    0.253994    0.115878
min      1.000000    0.006320    0.000000    0.460000    0.000000    0.385000
25%    127.250000    0.082045    0.000000    5.190000    0.000000    0.449000
50%    253.500000    0.256510    0.000000    9.690000    0.000000    0.538000
75%    379.750000    3.677083   12.500000   18.100000    0.000000    0.624000
max    506.000000   88.976200  100.000000   27.740000    1.000000    0.871000

              rm         age         dis         rad         tax     ptratio  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     6.284634   68.574901    3.795043    9.549407  408.237154   18.455534
std      0.702617   28.148861    2.105710    8.707259  168.537116    2.164946
min      3.561000    2.900000    1.129600    1.000000  187.000000   12.600000
25%      5.885500   45.025000    2.100175    4.000000  279.000000   17.400000
50%      6.208500   77.500000    3.207450    5.000000  330.000000   19.050000
75%      6.623500   94.075000    5.188425   24.000000  666.000000   20.200000
max      8.780000  100.000000   12.126500   24.000000  711.000000   22.000000

            black       lstat        medv
count  506.000000  506.000000  506.000000
mean   356.674032   12.653063   22.532806
std     91.294864    7.141062    9.197104
min      0.320000    1.730000    5.000000
25%    375.377500    6.950000   17.025000
50%    391.440000   11.360000   21.200000
75%    396.225000   16.955000   25.000000
max    396.900000   37.970000   50.000000
```

We can now also print all of the datapoints in this dataset.

In [77]:
```
print("total amount of datapoints: ", boston.shape[0], end="\n\n")
```

```
total amount of datapoints:  506
```

At this stage, we have quite a good idea at how the data looks like. I did also find a good source online that explains the dataset in further detail. The most important part for me was just to get an idea of what the columns actually mean which is important for the interpretation later.

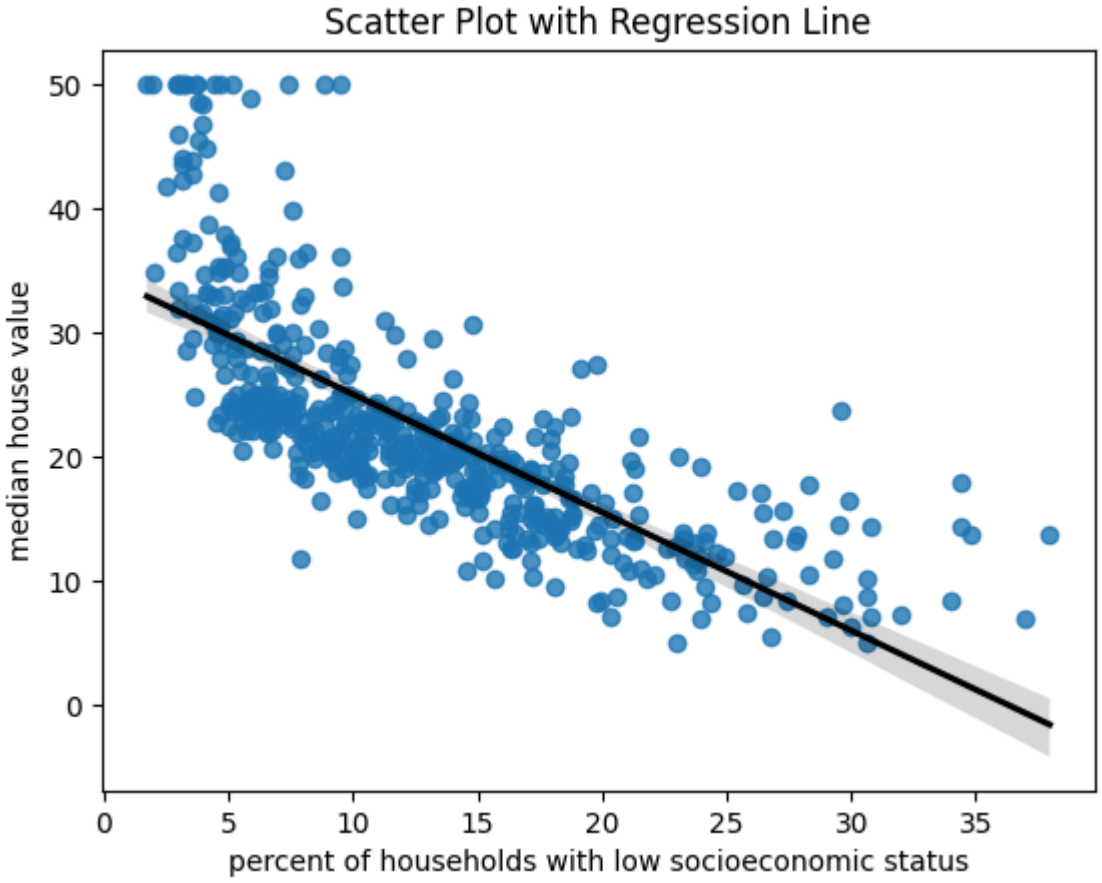
However, now when we have a good overview, we can now plot some predictors against some response values using linear regression. For this, i decided to use the same variables as the example, which was lstat, rm and age.

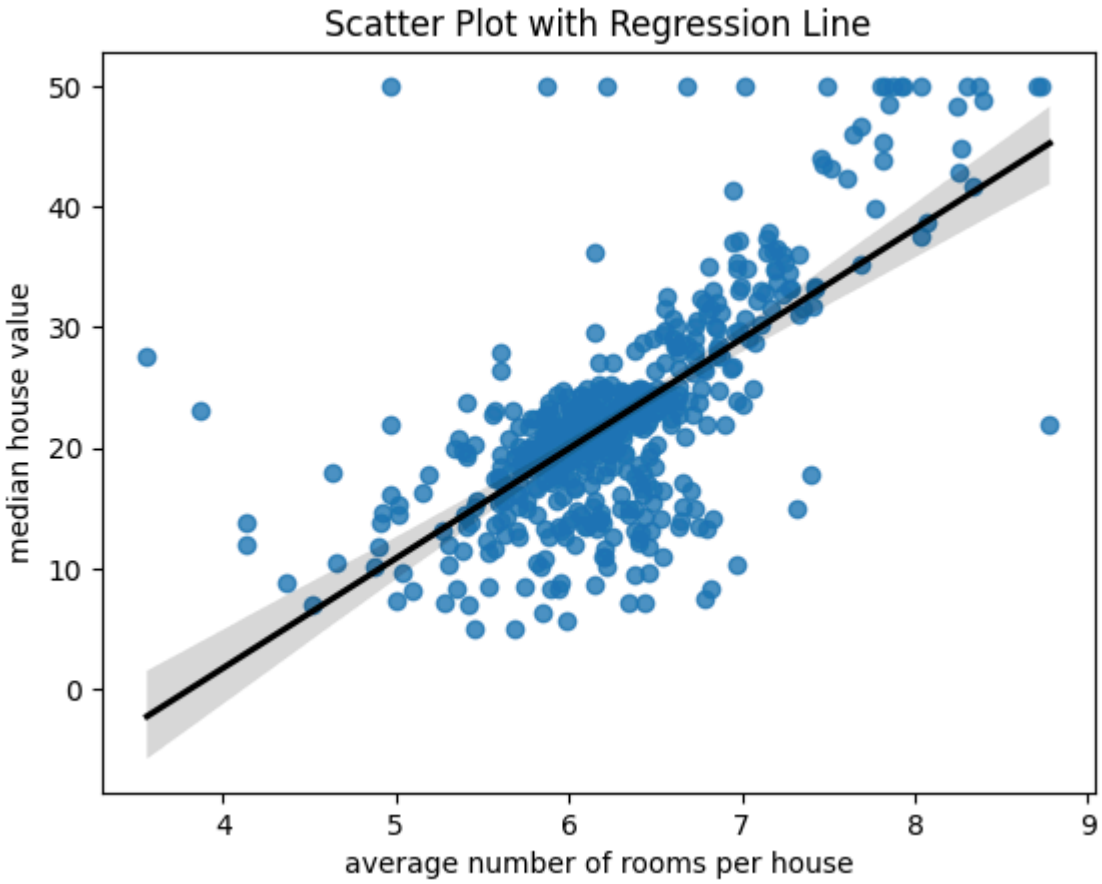
At this stage i will plot the scatter plots with a linear regression line, confidence interval and correlation coefficient.

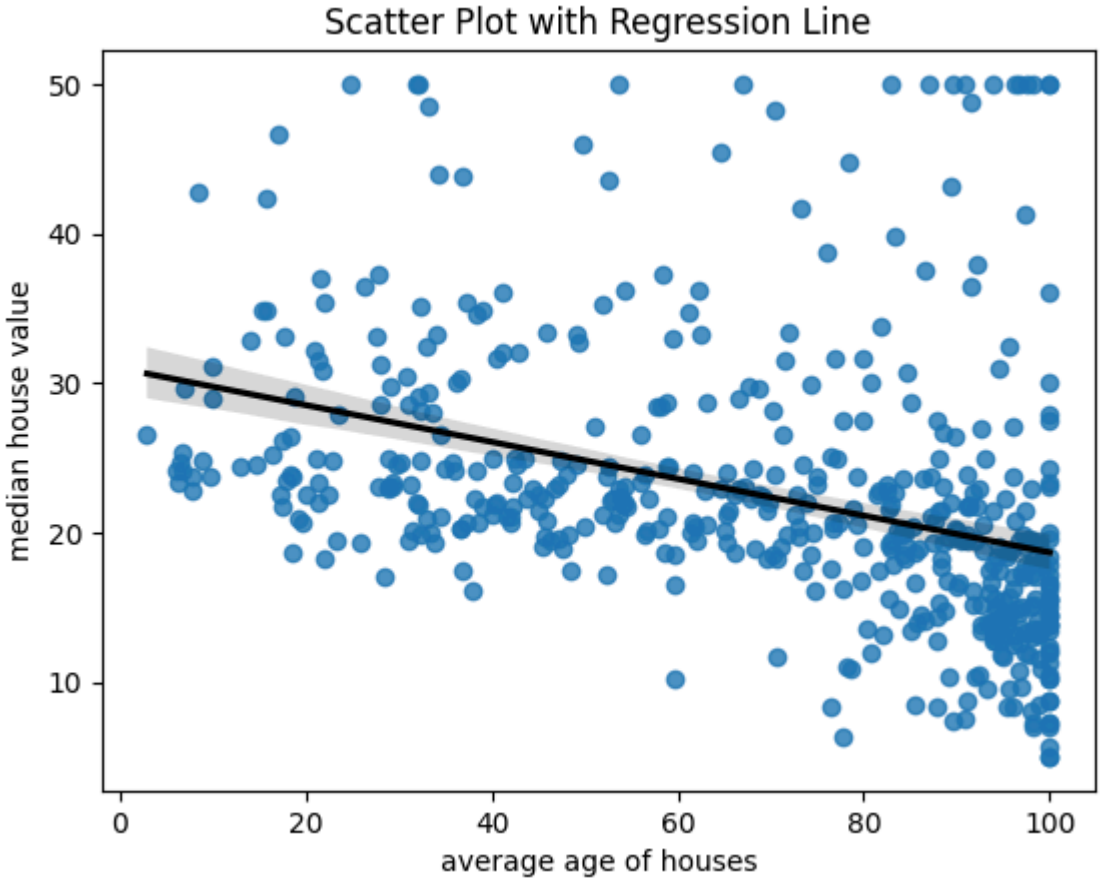
In [78]:
```
# correlation coefficients between our variables for the 3 plots
corCoef_medv_lstat, pValue_medv_lstat = stats.pearsonr(boston['medv'], boston['l
print("Correlation coefficient between medv and lstat: ", corCoef_medv_lstat, ",

corCoef_medv_lstat, pValue_medv_rm = stats.pearsonr(boston['medv'], boston['rm']
print("Correlation coefficient between medv and rm: ", corCoef_medv_lstat, ", wi

corCoef_medv_lstat, pValue_medv_age = stats.pearsonr(boston['medv'], boston['age
```

```
print("Correlation coefficient between medv and age: ", corCoef_medv_lstat, ", w

# Scatter plot with regression line between lstat and medv
sns.regplot(x=boston['lstat'], y=boston['medv'],  line_kws={'color': 'black'})

# Add labels to the plot
plt.xlabel("percent of households with low socioeconomic status")
plt.ylabel("median house value")
plt.title(f"Scatter Plot with Regression Line")
plt.show() # Remember to make the window bigger to see the plot

# Scatter plot with regression line between rm and medv
sns.regplot(x=boston['rm'], y=boston['medv'],  line_kws={'color': 'black'})

# Add labels to the plot
plt.xlabel("average number of rooms per house")
plt.ylabel("median house value")
plt.title(f"Scatter Plot with Regression Line")
plt.show()

# Scatter plot with regression line between age and medv
sns.regplot(x=boston['age'], y=boston['medv'],  line_kws={'color': 'black'})

# Add labels to the plot
plt.xlabel("average age of houses")
plt.ylabel("median house value")
plt.title(f"Scatter Plot with Regression Line")
plt.show()
```

```
Correlation coefficient between medv and lstat:  -0.7376627261740147 , with p-val
ue:  5.081103394387547e-88
Correlation coefficient between medv and rm:  0.6953599470715394 , with p-value:
2.4872288710071593e-74
Correlation coefficient between medv and age:  -0.37695456500459623 , with p-valu
e:  1.5699822091877261e-18
```


Scatter Plot with Regression Line
median house value
percent of households with low socioeconomic status
0
5
10
15
20
25
30
35
0
10
20
30
40
50


Scatter Plot with Regression Line
median house value
average number of rooms per house
4
5
6
7
8
9
0
10
20
30
40
50



Perform simple linear regressions

Now when we have a good overview of the data, and also plotted some regression plots with correlation coefficients. We can not go ahead and fit theese simple linear regression models with medv as the response value with some other variables. I will keep using the lstat, rm and age variables as predictors.

I can fit the models using the statsmodels.OLS package and from that print the statistical summary. However, the built in summary doesent include the residuals so i used the .resid function from the same package in order to print the residuals to make it as similar as possible to the example.

In [79]:
```
model_lstat_medv = sm.OLS(boston['medv'], sm.add_constant(boston['lstat'])).fit(
residuals_model_lstat_medv = model_lstat_medv.resid

print("Residuals: ", residuals_model_lstat_medv.describe(), end="\n\n") # we nee
print(model_lstat_medv.summary())



model_rm_medv = sm.OLS(boston['medv'], sm.add_constant(boston['rm'])).fit()
residuals_model_rm_medv = model_rm_medv.resid

print("Residuals: ", residuals_model_rm_medv.describe(), end="\n\n") # we need t
print(model_rm_medv.summary())



model_age_medv = sm.OLS(boston['medv'], sm.add_constant(boston['age'])).fit()
```

```
residuals_model_age_medv = model_age_medv.resid

print("Residuals: ", residuals_model_age_medv.describe(), end="\n\n") # we need
print(model_age_medv.summary())
```

```
Residuals:  count    5.060000e+02
mean     3.521821e-14
std      6.209603e+00
min     -1.516745e+01
25%     -3.989612e+00
50%     -1.318186e+00
75%      2.033701e+00
max      2.450013e+01
dtype: float64

                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.544
Model:                            OLS   Adj. R-squared:                  0.543
Method:                 Least Squares   F-statistic:                     601.6
Date:                Mon, 10 Feb 2025   Prob (F-statistic):           5.08e-88
Time:                        17:30:57   Log-Likelihood:                -1641.5
No. Observations:                 506   AIC:                             3287.
Df Residuals:                     504   BIC:                             3295.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         34.5538      0.563     61.415      0.000      33.448      35.659
lstat         -0.9500      0.039    -24.528      0.000      -1.026      -0.874
==============================================================================
Omnibus:                      137.043   Durbin-Watson:                   0.892
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              291.373
Skew:                           1.453   Prob(JB):                     5.36e-64
Kurtosis:                       5.319   Cond. No.                         29.7
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
Residuals:  count    5.060000e+02
mean     2.359114e-14
std      6.609606e+00
min     -2.334590e+01
25%     -2.547477e+00
50%      8.976267e-02
75%      2.985532e+00
max      3.943314e+01
dtype: float64

                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.484
Model:                            OLS   Adj. R-squared:                  0.483
Method:                 Least Squares   F-statistic:                     471.8
Date:                Mon, 10 Feb 2025   Prob (F-statistic):           2.49e-74
Time:                        17:30:57   Log-Likelihood:                -1673.1
No. Observations:                 506   AIC:                             3350.
Df Residuals:                     504   BIC:                             3359.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
const        -34.6706      2.650    -13.084      0.000     -39.877     -29.465
rm             9.1021      0.419     21.722      0.000       8.279       9.925
==============================================================================
Omnibus:                      102.585   Durbin-Watson:                   0.684
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              612.449
Skew:                           0.726   Prob(JB):                    1.02e-133
Kurtosis:                       8.190   Cond. No.                         58.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
Residuals:  count    5.060000e+02
mean    -1.246960e-14
std      8.518650e+00
min     -1.509662e+01
25%     -5.138002e+00
50%     -1.957464e+00
75%      2.397527e+00
max      3.133759e+01
dtype: float64

                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.142
Model:                            OLS   Adj. R-squared:                  0.140
Method:                 Least Squares   F-statistic:                     83.48
Date:                Mon, 10 Feb 2025   Prob (F-statistic):           1.57e-18
Time:                        17:30:57   Log-Likelihood:                -1801.5
No. Observations:                 506   AIC:                             3607.
Df Residuals:                     504   BIC:                             3615.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         30.9787      0.999     31.006      0.000      29.016      32.942
age           -0.1232      0.013     -9.137      0.000      -0.150      -0.097
==============================================================================
Omnibus:                      170.034   Durbin-Watson:                   0.613
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              456.983
Skew:                           1.671   Prob(JB):                    5.85e-100
Kurtosis:                       6.240   Cond. No.                         195.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Interpretation of results

Lets first try to understand what is actually being calculated at this stage and explain some of the terminologies, this is where the source i cited earlier will be very helpfull.

A *response variable*, in this case **medv**, is the dependent variable in the regression mode and it represents the outcome that we are trying to predict or explain using our desired model. In this case, **medv** means, the median house value in 1000's of dollars.

A *predictor variable*, in this case **lstat (percentege of households with low socioeconomic status)**, **rm (average number of rooms per house)** and **age (Proportion of owner occupied units built before 1940)**, are independent variables used to predict the response. In this analysis, i am essentially trying to understand how each predictor variable individually influences **medv**.

From the data that was gathered, i will create my own summary that extracts the key findings from the experiment. I have ranked theese based on the highest R-squared values.

Variables	R^2	Coefficient	P-value
medv ~ lstat	0.544	-0.950	P < 0.001
medv ~ rm	0.484	9.102	P < 0.001
medv ~ age	0.142	-0.123	P < 0.001

So, as we can see, **lstat** explains 54.4% of the variability in medv. We can also conclude from the coefficient that for each 0.95% increase in the percentage of low socioeconomic households, the median house value decreases by approximately $950.

rm explains 48.4% of the variability in medv. From the coefficient we can see that for each additional room in the house, the median house value increases by about $9102.

age explains 14.2% of the variability in medv. From the coefficient we can see that for every 1% increase in the proportion of older houses, the median house value decreases by about 123$.

Because **lstat** has the highest R^2 value, we can conclude that **lstat** has the strongest negative influence on house prices. This is followed by **rm** which has a strong positive influence. **age** also negatively affects house prices but has a much weaker relationship. All of theese relationships have a P-value of P < 0.001 which indicates that all of the predictors are statistically significant to the response variable.

From a practical standpoint, this could indicate that stakeholders that have an interest in housing policies such as governmental bodies, banks and credit unions should prioritize socioeconomic factors and house sizes over the age of homes when estimating or influencing house prices when doing market analysies.

Up next, we now have to obtain a confidence interval for the coefficient estimates for the individual models.

In [80]:
```
# Confidence intervals for the coefficients from the model with lstat
conf_lstat_medv = model_lstat_medv.conf_int()
conf_lstat_medv.columns = ['2.5%', '97.5%'] # trying to make it look like the ex
conf_lstat_medv.index = ['(Intercept)', 'lstat (slope)']
print("Confidence interval for lstat: ")
print(conf_lstat_medv, end="\n\n")

# Confidence intervals for the coefficients from the model with rm
conf_rm_medv = model_rm_medv.conf_int()
```

```
conf_rm_medv.columns = ['2.5%', '97.5%'] # trying to make it look like the examp
conf_rm_medv.index = ['(Intercept)', 'rm (slope)']
print("Confidence interval for rm: ")
print(conf_rm_medv, end="\n\n")

# Confidence intervals for the coefficients from the model with age
conf_age_medv = model_age_medv.conf_int()
conf_age_medv.columns = ['2.5%', '97.5%'] # trying to make it look like the exam
conf_age_medv.index = ['(Intercept)', 'age (slope)']
print("Confidence interval for age: ")
print(conf_age_medv, end="\n\n")
```

```
Confidence interval for lstat:
                   2.5%      97.5%
(Intercept)    33.448457  35.659225
lstat (slope)  -1.026148  -0.873951

Confidence interval for rm:
                  2.5%      97.5%
(Intercept) -39.876641 -29.464601
rm (slope)    8.278855   9.925363

Confidence interval for age:
                  2.5%      97.5%
(Intercept)  29.015752  32.941604
age (slope)  -0.149647  -0.096679
```

A confidence interval provides a range of values for the coefficient estimates where we are 95% confident that the true population parameter lies. If the range doesent include 0, the predictor is statistically significant.

The confidence interval for **lstat** show that we have a range for the intercept of [33.448 - 35.659] which means that if **lstat** would be 0 (meaning we have no low socioeconomic households in a given area), the house value is predicted to be between $33448 and$ 35659. The slope for lstat falls in the range [-1.026 - -0.874] which means that for every 1% increse in low socioeconomic households in a given neighborhood, the median house value will decrease by $874 - 1026$

The confidence interval for **rm** show that we have a range for the intercept of [-39.877 - -29.465] which means that if a room would have 0 rooms, the median house value would be negative. This specific piece of information isnt that meaningfull but could reflect extrapolation. The slope, however, is more meaningfull as it falls in the range [8.279 - 9.925] which means that for every additional room, the median house value increases by $8279 - 9925$.

The confidence interval for **age** show that we have a range for the intercept of [29.016 - 32.942] which means that if **age** would be 0 (brand new), the median house value would be predicted to be between $29016 - 32942$. The slope for **age** falls in the range [-0.150 - -0.097] which means that for every 1% increase in the proportion of older houses, the median house value decreases by $97 - 150$.

All of theese confidence intervals dont include 0 in the ranges so we know that all of them are statistically signifcant.

Use the simple linear regression models

For the next part of the assignment, we have to predict the medv response values for the selected predictor values. Calculate the prediction intervals for theese values.

In [81]:
```
# Prediction interval for the model with lstat

new_lstat = pd.DataFrame({'lstat': [5, 10, 15]})  # The three levels

new_lstat_with_const = sm.add_constant(new_lstat)

pred_lstat_medv = model_lstat_medv.get_prediction(new_lstat_with_const)
pred_lstat_medv_summary = pred_lstat_medv.summary_frame(alpha=0.05)
print("Prediction interval for lstat: ")
print(pred_lstat_medv_summary, end="\n\n")

# Prediction interval for the model with rm

new_rm = pd.DataFrame({'rm': [5, 6.5, 8]})  # The three levels

new_rm_with_const = sm.add_constant(new_rm)

pred_rm_medv = model_rm_medv.get_prediction(new_rm_with_const)
pred_rm_medv_summary = pred_rm_medv.summary_frame(alpha=0.05)
print("Prediction interval for rm: ")
print(pred_rm_medv_summary, end="\n\n")

# Prediction interval for the model with age

new_age = pd.DataFrame({'age': [25, 50, 75]})  # The three levels

new_age_with_const = sm.add_constant(new_age)

pred_age_medv = model_age_medv.get_prediction(new_age_with_const)
pred_age_medv_summary = pred_age_medv.summary_frame(alpha=0.05)
print("Prediction interval for age: ")
print(pred_age_medv_summary, end="\n\n")
```

```
Prediction interval for lstat:
        mean   mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower  \
0  29.803594  0.405247      29.007412      30.599776     17.565675
1  25.053347  0.294814      24.474132      25.632563     12.827626
2  20.303101  0.290893      19.731588      20.874613      8.077742

   obs_ci_upper
0     42.041513
1     37.279068
2     32.528459

Prediction interval for rm:
        mean   mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower  \
0  10.839924  0.613410       9.634769      12.045079     -2.214474
1  24.493088  0.307657      23.888639      25.097536     11.480391
2  38.146251  0.776633      36.620414      39.672088     25.058353

   obs_ci_upper
0     23.894322
1     37.505784
2     51.234149

Prediction interval for age:
        mean   mean_se  mean_ci_lower  mean_ci_upper  obs_ci_lower  \
0  27.899610  0.699094      26.526112      29.273107     11.090368
1  24.820542  0.454307      23.927973      25.713110      8.043748
2  21.741474  0.388844      20.977518      22.505429      4.971031

   obs_ci_upper
0     44.708852
1     41.597335
2     38.511917
```

A prediction interval provides a range where an individual prediction is expected to fall, taking the uncertainty of the model and variability of data in to account. Prediction intervals will be much wider than confidence intervals because they include variability for individual predictions, not just the mean.

For **lstat** we have predictions of the mean of predicted median house values as follows:

- For lstat = 5, predicted medv is $29,804.
- For lstat = 10, predicted medv is $25,053.
- For lstat = 15, predicted medv is $20,303.

the prediction interval for lstat = 5 can be found if we look at the 'obs_ci_lower' and 'obs_ci_upper' columns of the first row. For lstat = 5, the median house value for an individual house is expected to be between $17,566 and $42,042. Similarily, wider ranges for higher lstat values which means it increases uncertainty. The key takeaway from this is that as the percentage of low socioeconomic houses increase in a neighborhood (lstat value increases), the median house values drop.

For **rm** we have predictions of the mean of predicted median house as follows:

- For rm = 5, predicted medv is $10,840.
- For rm = 6.5, predicted medv is $24,493.

- For rm = 8, predicted medv is $38,146.

Prediction intevals:

- For rm = 5, individual predictions range from $-2,214 to 23,894$.
- For rm = 6.5, predictions range from $11,480 to 37,506$.
- For rm = 8, predictions range from $25,058 to 51,234$.

For **age** we have predictions of the mean of predicted median house as follows:

- For age = 25, predicted medv is $27,900.
- For age = 50, predicted medv is $24,821.
- For age = 75, predicted medv is $21,741.

prediction intervals:

- For age = 25, individual predictions range from $11,090 to 44,709$.
- For age = 50, predictions range from $8,044 to 41,597$.
- For age = 75, predictions range from $4,971 to 38,512$.

Use the simple linear regression models

Now we fit **medv** as a response with the predictors selected before altogether.

In [82]:
```
model_multivariate = sm.OLS(boston['medv'], sm.add_constant(boston[['lstat', 'rm

residuals_model_multivariate = model_multivariate.resid
print("Residuals: ", residuals_model_multivariate.describe(), end="\n\n") # we n
print(model_multivariate.summary())
```

```
Residuals:  count    5.060000e+02
mean     1.685082e-15
std      5.525660e+00
min     -1.820992e+01
25%     -3.467402e+00
50%     -1.053282e+00
75%      1.957443e+00
max      2.750044e+01
dtype: float64
```

OLS Regression Results

Dep. Variable:	medv	R-squared:	0.639
Model:	OLS	Adj. R-squared:	0.637
Method:	Least Squares	F-statistic:	296.2
Date:	Mon, 10 Feb 2025	Prob (F-statistic):	1.20e-110
Time:	17:30:57	Log-Likelihood:	-1582.4
No. Observations:	506	AIC:	3173.
Df Residuals:	502	BIC:	3190.
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-1.1753	3.182	-0.369	0.712	-7.427	5.076
lstat	-0.6685	0.054	-12.298	0.000	-0.775	-0.562
rm	5.0191	0.454	11.048	0.000	4.127	5.912
age	0.0091	0.011	0.811	0.418	-0.013	0.031

Omnibus:	138.819	Durbin-Watson:	0.851
Prob(Omnibus):	0.000	Jarque-Bera (JB):	415.436
Skew:	1.296	Prob(JB):	6.15e-91
Kurtosis:	6.603	Cond. No.	985.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

From the residuals we can see that we have a mean residual that is very close to 0 which means that the model predictions on average align well with the observed values.

We have an R^2 value of 0.639 which means that our model that contains all three predictors explain 63.9% of the variability in **medv**. This is an improvement over the individual models that had at most 54.4%. We have a P value P < 0.001 which means that at least one of our predictors is statistically signifcant in explaining medv.

We can also deduce some information from the table containing our predictor variables.

the 'const' has an intercept coefficient of -1.1753 which means that this is the predicted value of **medv** when all predictors are 0. However, because 0 can be an unrealistic number, especially **rm**, this has little to no meaning.

lstat has intercept of -0.6685 with p < 0.001 which means that a 1% increase in **lstat** the **medv** will decrease by $668.5.

rm (intercept: 5.0191, P < 0.001) shows that for every additional room, the house values increase by $5019, holding all other predictors constant.

age (intercept: 0.0091, P = 0.418) the coefficient is small and significantly insignificant which means that age has little to no impact on predicting medv.

Conclusively, we can say that this model, including multiple predictors can improve the models explanatory power (given that R^2 = 0.639) as opposed to using each variable individually as a predictor.

One small detail, which is easy to miss but very important when taking multiple variables as predictors for one response variable, is the condition number. This can be seen as a rating of multicollinearity where everything over 30 can signal that multicollinearity should be investigated. In my case, i have 985 which is extremely high. Multicollinearity is when predictors are strongly correlated with each other, this can make it so that the model struggles to determine which predictor is responsible for output variations in the response variable which can lead to higher errors and unstable estimates.

Perform multiple linear regressions

For this part we have to predict the medv response values for all of the predictors, meaning all of the variables in the dataset. Calculate the prediction intervals for these values.

In [83]:
```
model_all = sm.OLS(boston['medv'], sm.add_constant(boston.drop(columns=['medv'])
residuals_model_all = model_all.resid
print("Residuals: ", residuals_model_all.describe(), end="\n\n") # we need to aa
print(model_all.summary())
```

```
Residuals:  count    5.060000e+02
mean     2.237508e-13
std      4.676799e+00
min     -1.589479e+01
25%     -2.758540e+00
50%     -4.662679e-01
75%      1.796326e+00
max      2.609108e+01
dtype: float64
```

OLS Regression Results

Dep. Variable:	medv	R-squared:	0.741
Model:	OLS	Adj. R-squared:	0.734
Method:	Least Squares	F-statistic:	100.6
Date:	Mon, 10 Feb 2025	Prob (F-statistic):	3.44e-134
Time:	17:30:57	Log-Likelihood:	-1498.0
No. Observations:	506	AIC:	3026.
Df Residuals:	491	BIC:	3089.
Df Model:	14		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	36.4614	5.101	7.148	0.000	26.439	46.484
Unnamed: 0	-0.0025	0.002	-1.215	0.225	-0.007	0.002
crim	-0.1088	0.033	-3.310	0.001	-0.173	-0.044
zn	0.0480	0.014	3.484	0.001	0.021	0.075
indus	0.0199	0.061	0.324	0.746	-0.101	0.141
chas	2.7052	0.861	3.141	0.002	1.013	4.398
nox	-17.5416	3.822	-4.589	0.000	-25.052	-10.031
rm	3.8392	0.418	9.175	0.000	3.017	4.661
age	-0.0019	0.013	-0.145	0.885	-0.028	0.024
dis	-1.4933	0.200	-7.471	0.000	-1.886	-1.101
rad	0.3249	0.068	4.771	0.000	0.191	0.459
tax	-0.0116	0.004	-3.046	0.002	-0.019	-0.004
ptratio	-0.9480	0.131	-7.246	0.000	-1.205	-0.691
black	0.0094	0.003	3.485	0.001	0.004	0.015
lstat	-0.5262	0.051	-10.377	0.000	-0.626	-0.427

Omnibus:	175.545	Durbin-Watson:	1.084
Prob(Omnibus):	0.000	Jarque-Bera (JB):	760.925
Skew:	1.502	Prob(JB):	5.85e-166
Kurtosis:	8.202	Cond. No.	1.68e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.68e+04. This might indicate that there are strong multicollinearity or other numerical problems.

From the residuals we can see that the *mean residual* being close to 0 suggests that the model predictions align well with the actual data on average. the *std* of 4.68 indicates the average distance of predictions from the observed values, showing reasonable variability. The range (from min to max) is [-15.89 - 26.09] show that the model has some outliers with large prediction errors which can cause the model to be make big mistakes both on the upside and downside.

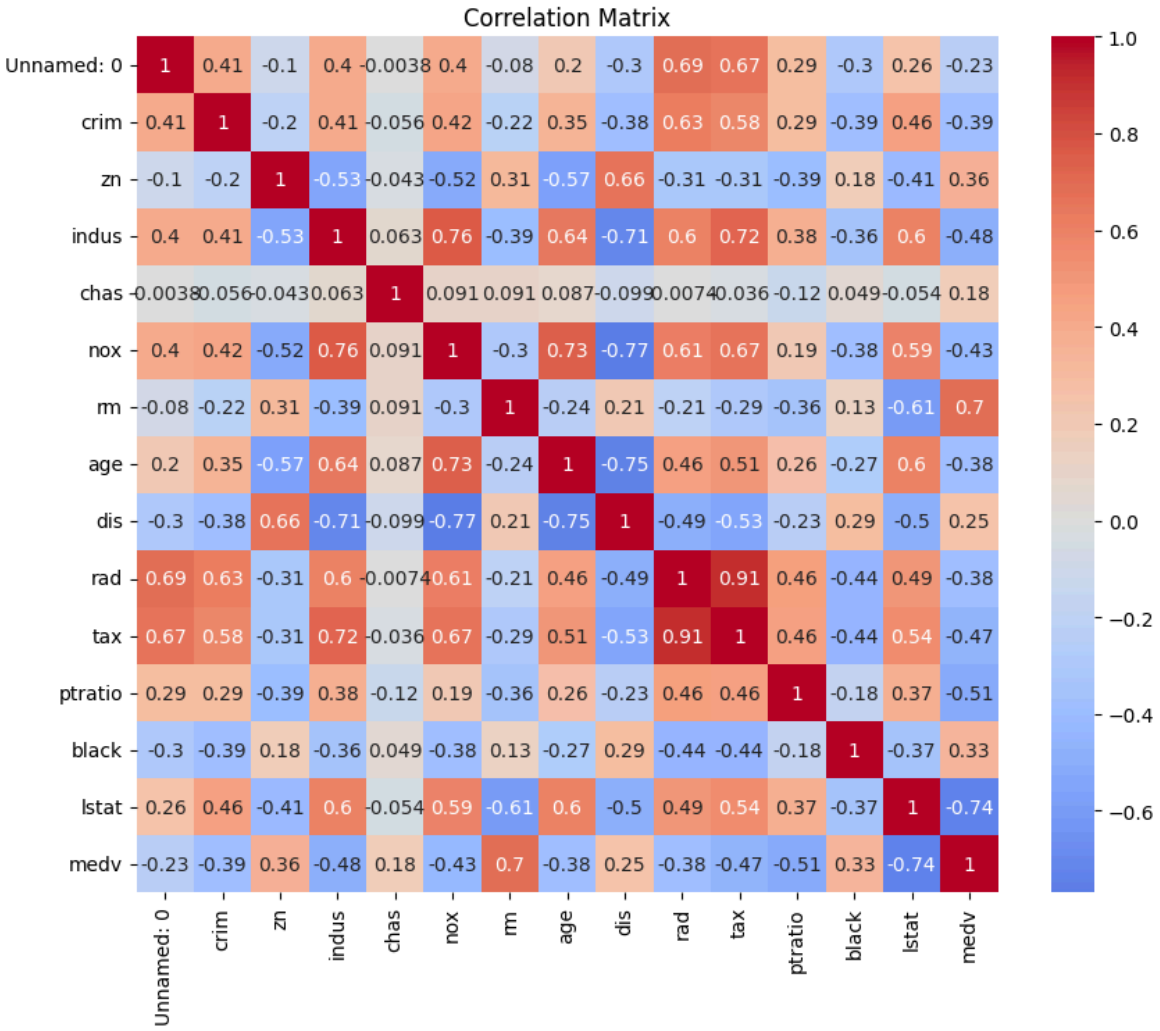
R^2 = 0.741 which means that the model explains 74.1% of the variability in **medv**. This is a significant improvement compared to models with fewer predictors. We have P < 0.001 which means that the model as a whole is highly significant.

I could explain every single variables impact on **medv** from the table but because i have demonstrated how to do that before, i will not do that for all of the 13 predictors as that would make the report too long. Instead, i will write a list of some key takeways from the data.

- lstat has a strong negative impact (-0.5262) which confirms that higher proportions of low income households lower house values.
- **rm** has a strong positive impact (3.8392), showing that larger homes are valued higher.
- **nox** has a strong negative impact (-17.5416), indicating that pollution reduces house values significantly.
- **ptratio** has a negative impact (-0.9480) which means that worse school/education quality is associated with lower **medv** values.
- **indus** and **age** are not statistically significant and has no meaningfull effect on **medv** because P > 0.05.
- The condition number is 1680 is very high which means that there are variables that have big correlation.

In conclusion and in practice, we can say that socioeconomic factors, house sizes and environmental quality are the leading factors that drives house prices.

In [84]:
```
correlation_matrix = boston.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title("Correlation Matrix")
plt.show()
```



The correlation matrix shows the pairwise correlations between all predictors and the response variable **medv**.

theese are some of the conclusions we can draw from the graph:

- **lstat** has a strong negative correlation with **medv** (-0.74): A higher percentage of low socioeconomic households strongly reduces house values.
- **rm** has a strong positive correlation with **medv** (0.7): Larger houses with more rooms are associated with higher house values.
- **ptratio** (-0.51), **tax** (-0.47), and **nox** (-0.43) also have notable negative correlations with **medv**, indicating that poor school quality, higher taxes, and pollution reduce house values.

we have multicollinearity between the following:

- Tax ~ rad (0.91) --> high property taxes near highways
- nox ~ indus (0.76) --> pollution is strong in industrial areas
- dis ~ nox (-0.77) --> proximity to employment centers are negatively correlated with pollution levels
- age ~ nox (0.73) --> Older neighborhoods tend to have higher pollution levels

In [85]:
```
lstatC = [5, 10, 15]
rmC = [5, 6.5, 8]
selected_predictor_values = pd.DataFrame(
```

```
    [(lstat, rm) for lstat in lstatC for rm in rmC], columns=["lstat", "rm"]
)

# Add a constant for the intercept
selected_predictor_values_with_const = sm.add_constant(selected_predictor_values

# Fit the regression model with lstat and rm
model_lstat_rm_medv = sm.OLS(boston['medv'], sm.add_constant(boston[['lstat', 'r

# Predict `medv` and calculate prediction intervals
predictions = model_lstat_rm_medv.get_prediction(selected_predictor_values_with_
prediction_intervals = predictions.summary_frame(alpha=0.05)  # 95% intervals

# Display results
result_df = pd.concat([selected_predictor_values, prediction_intervals], axis=1)
print(result_df, end="\n\n")
```

```
   lstat   rm       mean   mean_se  mean_ci_lower  mean_ci_upper  \
0      5  5.0  20.903875  0.856315      19.221481      22.586269
1      5  6.5  28.546057  0.377499      27.804387      29.287727
2      5  8.0  36.188239  0.663860      34.883959      37.492519
3     10  5.0  17.692084  0.693873      16.328837      19.055330
4     10  6.5  25.334266  0.263915      24.815754      25.852777
5     10  8.0  32.976448  0.739470      31.523618      34.429277
6     15  5.0  14.480292  0.570322      13.359785      15.600799
7     15  6.5  22.122474  0.304004      21.525200      22.719748
8     15  8.0  29.764656  0.865184      28.064837      31.464475

   obs_ci_lower  obs_ci_upper
0      9.889729     31.918021
1     17.635923     39.456192
2     25.225479     47.150999
3      6.722152     28.662016
4     14.437027     36.231505
5     21.995024     43.957872
6      3.537875     25.422709
7     11.221204     33.023745
8     18.747835     40.781477
```

Here we have predicted **medv** using specific values for the predictors variables. Here, again, i will not explain every single row but i will give some examples of interpretations of the key artifacts observed from the output and the rest of the interpretation can be done by the reader.

The *mean* row predicts the median house value **medv** for the given values of **lstat** and **rm**. *mean_ci_lower* and *mean_ci_upper* span the confidence interval for the mean prediction. *obs_ci_lower* and *obs_ci_upper* span the prediction interval for individual observations.

This would be how one would interpret Row 1:

Predicted medv (mean): $20,904.

Mean Confidence Interval: [19, 221,22,586].The average house value for neighborhoods with lstat=5% and rm=5 is expected to fall within this range.

Prediction Interval: $[9,890, 31,918]$. The house price for an individual house in this type of neighborhood can range between these values.

The conclusion that can be made is that neighborhoods with higher **lstat** (lower socioeconomic status) and lower **rm** (fewer rooms) tend to have lower house prices, with considerable variability. Neighborhoods with lower **lstat** and more rooms tend to have higher and more predictable house prices.

ASSIGNMENT 3

This assignment follows up on the previous one. We will start by checking the accuracy of the linear regression model once again with some predictors.

In [86]:
```
# Check again for accuracy in the linear regression

model_multivariate_2 = sm.OLS(boston['medv'], sm.add_constant(boston[['lstat', '

residuals_model_multivariate_2 = model_multivariate_2.resid
print("Residuals: ", residuals_model_multivariate_2.describe(), end="\n\n") # we
print(model_multivariate_2.summary())
```

```
Residuals:  count    5.060000e+02
mean    -4.937289e-14
std      4.969081e+00
min     -1.277651e+01
25%     -3.018557e+00
50%     -6.480976e-01
75%      1.975195e+00
max      2.776253e+01
dtype: float64
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.708
Model:                            OLS   Adj. R-squared:                  0.705
Method:                 Least Squares   F-statistic:                     242.6
Date:                Mon, 10 Feb 2025   Prob (F-statistic):          3.67e-131
Time:                        17:30:58   Log-Likelihood:                -1528.7
No. Observations:                 506   AIC:                             3069.
Df Residuals:                     500   BIC:                             3095.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         37.4992      4.613      8.129      0.000      28.436      46.562
lstat         -0.5811      0.048    -12.122      0.000      -0.675      -0.487
rm             4.1633      0.412     10.104      0.000       3.354       4.973
nox          -17.9966      3.261     -5.519      0.000     -24.403     -11.590
dis           -1.1847      0.168     -7.034      0.000      -1.516      -0.854
ptratio       -1.0458      0.114     -9.212      0.000      -1.269      -0.823
==============================================================================
Omnibus:                      187.456   Durbin-Watson:                   0.971
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              885.498
Skew:                           1.584   Prob(JB):                    5.21e-193
Kurtosis:                       8.654   Cond. No.                         545.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Now, we want to add an interaction term **lstat*rm** while also including them as plain predictors. This was simply done by creating a new placeholder column and within it multiply the two columns and then adding them in my list of predictors from the boston dataset. An interaction term basically captures how the effect of one predictor can change depending on another predictor. So in this specific scenario, which is actually quite realistic, a houses price might be affected differently by **lstat** depending on how many rooms **rm** it has. This allows theese two variables to affect the **medv** response variable together by scaling instead of acting separately.

In [87]:
```
# Fit a model with interaction terms. Don't forget to also include the include t

# Check again for accuracy in the linear regression

# This is the new 'interaction' column
boston['lstat_rm'] = boston['lstat'] * boston['rm'] # lowkey cringe that i have
```

```
model_interaction = sm.OLS(boston['medv'], sm.add_constant(boston[['lstat', 'rm'

residuals_model_interaction = model_interaction.resid
print("Residuals: ", residuals_model_interaction.describe(), end="\n\n")
print(model_interaction.summary())
```

```
Residuals:  count    5.060000e+02
mean     1.565441e-13
std      4.337354e+00
min     -1.930609e+01
25%     -2.472038e+00
50%     -3.606848e-01
75%      1.819178e+00
max      2.990857e+01
dtype: float64

                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.778
Model:                            OLS   Adj. R-squared:                  0.775
Method:                 Least Squares   F-statistic:                     290.8
Date:                Mon, 10 Feb 2025   Prob (F-statistic):          2.48e-159
Time:                        17:30:58   Log-Likelihood:                -1459.9
No. Observations:                 506   AIC:                             2934.
Df Residuals:                     499   BIC:                             2963.
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.1518      4.880      0.646      0.519      -6.435      12.739
lstat          1.8115      0.196      9.237      0.000       1.426       2.197
rm             8.3344      0.491     16.971      0.000       7.370       9.299
nox          -12.3651      2.885     -4.286      0.000     -18.033      -6.697
dis           -1.0184      0.148     -6.893      0.000      -1.309      -0.728
ptratio       -0.7152      0.103     -6.967      0.000      -0.917      -0.514
lstat_rm      -0.4185      0.034    -12.488      0.000      -0.484      -0.353
==============================================================================
Omnibus:                      246.928   Durbin-Watson:                   1.079
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2792.613
Skew:                           1.836   Prob(JB):                         0.00
Kurtosis:                      13.908   Cond. No.                     2.36e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.36e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

So for this interpretation, i will only interpret how this interaction changes the performance of the models. I will not discuss the coefficients as they are the same as the example and i have already explained the meaning of theese coefficients before and the logic is the same for theese ones.

When comparing this model with the interaction terms to the model without the interaction terms we see that the the R^2 and $AdjustedR^2$ are higher for the interaction model then for the non-interaction model which is an almost 10% boost. The higher F-

statistic also tells us that this interaction term also is a statistically significant improvement. The residual standard error is also lower which means that the model makes more accurate predictions even though the range is a bit wider.

Up next, we want to apply some non-linear transformation to a predictor. In this example, we do that by just including a new predictor which is the squared of the interaction term that we defined for the previous model. I do this also simply by just creating a new placeholder column in the dataset and then just defining it as the square of the **lstat_rm** interaction variable.

In [88]:
```
# Fit a model with non-linear transformations of the predictor terms. Don't forg

boston['lstat_rm^2'] = boston['lstat_rm']**2


model_interaction = sm.OLS(boston['medv'], sm.add_constant(boston[['lstat', 'rm'

residuals_model_interaction = model_interaction.resid
print("Residuals: ", residuals_model_interaction.describe(), end="\n\n")
print(model_interaction.summary())
```

```
Residuals:  count    5.060000e+02
mean     4.956948e-13
std      4.302529e+00
min     -1.841490e+01
25%     -2.433926e+00
50%     -2.956008e-01
75%      1.942593e+00
max      2.731066e+01
dtype: float64
```

OLS Regression Results

Dep. Variable:	medv	R-squared:	0.781
Model:	OLS	Adj. R-squared:	0.778
Method:	Least Squares	F-statistic:	253.9
Date:	Mon, 10 Feb 2025	Prob (F-statistic):	8.05e-160
Time:	17:30:58	Log-Likelihood:	-1455.8
No. Observations:	506	AIC:	2928.
Df Residuals:	498	BIC:	2961.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	10.5522	5.499	1.919	0.056	-0.253	21.357
lstat	1.5468	0.216	7.167	0.000	1.123	1.971
rm	7.6004	0.552	13.777	0.000	6.516	8.684
nox	-12.2898	2.865	-4.290	0.000	-17.918	-6.662
dis	-1.0641	0.148	-7.209	0.000	-1.354	-0.774
ptratio	-0.7112	0.102	-6.977	0.000	-0.912	-0.511
lstat_rm	-0.4468	0.035	-12.864	0.000	-0.515	-0.379
lstat_rm^2	0.0004	0.000	2.845	0.005	0.000	0.001

Omnibus:	217.415	Durbin-Watson:	1.059
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2007.945
Skew:	1.622	Prob(JB):	0.00
Kurtosis:	12.204	Cond. No.	3.02e+05

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.02e+05. This might indicate that there are strong multicollinearity or other numerical problems.

In [89]:
```
anova_linear_interaction = sm.add_constant(boston[['lstat', 'rm', 'nox', 'dis',
anova_linear_interaction_model = sm.OLS(boston['medv'], anova_linear_interaction

anova_nonlinear_interaction = sm.add_constant(boston[['lstat', 'rm', 'nox', 'dis
anova_nonlinear_interaction_model = sm.OLS(boston['medv'], anova_nonlinear_inter

anova_results = anova_lm(anova_linear_interaction_model, anova_nonlinear_interac
print(anova_results, end="\n\n")

# INTERPRET THE RESULT
```

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	499.0	9500.381881	0.0	NaN	NaN	NaN
1	498.0	9348.435955	1.0	151.945925	8.094303	0.004623

When comparing the previous linear model with the new transformed non-linear model we can see from the summary that we get a slightly better R^2 score for the non-linear model and we also get lower residual errors. But we also want to use the ANOVA test in order to see if this improvement is statistically significant or not. And from the output from the ANOVA test we can see that pr(>F) is 0.004623 whici is smaller than 0.05 which measnt that the test is statistically significant which also mean that the new proposed model improvement is statistically significant.

It is probably worth to mention that i chose 0.05 as my significance level (threshold) because that is the level that i am most used to from my statistics courses that i have taken prior to this. This means that if this improvement posed to real positive effect, we would make a mistake on that assumption 5 out of 100 tests. of corse, this could be tested on other levels but i think that 0.05 is a good threshold for this context.

In [90]:
```
# This is very hardcoded but i think this is clearer than the OLS built in funct

# Like what is that??? absolutely crazy.

# OLS.from_formula('... ~ ...' + '+'.join(['np.power(lstat,' + str(i) + ')' for

boston['lstat^2'] = boston['lstat']**2 # Much easier to just do it like this. Cr
boston['lstat^3'] = boston['lstat']**3
boston['lstat^4'] = boston['lstat']**4
boston['lstat^5'] = boston['lstat']**5


model_poly = sm.OLS(boston['medv'], sm.add_constant(boston[['lstat', 'rm', 'nox'

residuals_model_poly = model_poly.resid
print("Residuals: ", residuals_model_poly.describe(), end="\n\n")
print(model_poly.summary()) # Gives the same output as the example so the built-

# INTERPRET THE RESULT
```

```
Residuals:  count    5.060000e+02
mean     7.566364e-09
std      4.198956e+00
min     -1.630619e+01
25%     -2.256228e+00
50%     -3.016073e-01
75%      1.854274e+00
max      2.826980e+01
dtype: float64
```

OLS Regression Results

Dep. Variable:	medv	R-squared:	0.792
Model:	OLS	Adj. R-squared:	0.787
Method:	Least Squares	F-statistic:	188.0
Date:	Mon, 10 Feb 2025	Prob (F-statistic):	1.80e-161
Time:	17:30:58	Log-Likelihood:	-1443.5
No. Observations:	506	AIC:	2909.
Df Residuals:	495	BIC:	2956.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	33.9246	7.663	4.427	0.000	18.869	48.981
lstat	-5.6422	1.426	-3.957	0.000	-8.444	-2.841
rm	6.5291	0.711	9.183	0.000	5.132	7.926
nox	-13.7513	2.823	-4.871	0.000	-19.298	-8.204
dis	-1.0326	0.145	-7.127	0.000	-1.317	-0.748
ptratio	-0.7407	0.101	-7.324	0.000	-0.939	-0.542
lstat_rm	-0.3055	0.052	-5.878	0.000	-0.408	-0.203
lstat^2	0.8633	0.187	4.622	0.000	0.496	1.230
lstat^3	-0.0495	0.012	-4.153	0.000	-0.073	-0.026
lstat^4	0.0013	0.000	3.795	0.000	0.001	0.002
lstat^5	-1.279e-05	3.64e-06	-3.514	0.000	-1.99e-05	-5.64e-06

Omnibus:	232.049	Durbin-Watson:	1.116
Prob(Omnibus):	0.000	Jarque-Bera (JB):	2275.267
Skew:	1.742	Prob(JB):	0.00
Kurtosis:	12.787	Cond. No.	3.38e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.38e+08. This might indicate that there are strong multicollinearity or other numerical problems.

In [91]:
```
anova_linear_interaction = sm.add_constant(boston[['lstat', 'rm', 'nox', 'dis',
anova_linear_interaction_model = sm.OLS(boston['medv'], anova_linear_interaction

anova_poly = sm.add_constant(boston[['lstat', 'rm', 'nox', 'dis', 'ptratio', 'ls
anova_poly_model = sm.OLS(boston['medv'], anova_poly).fit()

anova_results = anova_lm(anova_linear_interaction_model, anova_poly_model)
print(anova_results, end="\n\n")

# INTERPRET THE RESULT
```

```
   df_resid          ssr  df_diff     ss_diff         F    Pr(>F)
0     499.0  9500.381881      0.0         NaN       NaN       NaN
1     495.0  8903.772453      4.0  596.609428  8.292038  0.000002
```

For this part, we want to check if several polynomials as predictors make for an even better model and then compare it to the linearly fitted mdoel.

For this model, i tried using a built in function that i could translate from the R-code in the example and i did find something in the documentation but it felt way too complex for this problem so i hardcoded my own polynomial in a similar way as the other new predictors for the previous models. I just created new columns for the new columns and mulitplied them with each other 1-5 times to get the new predictors and then append them to the list of predictors manually.

As we can see, we once again got an even better R-squared score and a better adjusted R-squared to take the newly added variables in to account but a much lower F-statistic which means that the polynomial introduces more complexity to the model. similarily to before, we want to check if this improvement is statistically significant or not with the ANOVA test.

From the ANOVA test we got that the P-value which is the same as Pr(>F) is 0.000002 which is much lower than 0.05 which means that on this significance level. This improvement is statistically significant.

In [92]:
```
# Trying to implement logarithmic transformation

# log(rm)
boston['log_rm'] = np.log(boston['rm']) # this is also hardcoded but i think thi

model_poly_log = sm.OLS(boston['medv'], sm.add_constant(boston[['lstat', 'rm', '

residuals_model_poly_log = model_poly_log.resid
print("Residuals: ", residuals_model_poly_log.describe(), end="\n\n")
print(model_poly_log.summary())


anova_linear_interaction = sm.add_constant(boston[['lstat', 'rm', 'nox', 'dis',
anova_linear_interaction_model = sm.OLS(boston['medv'], anova_linear_interaction

anova_poly_log = sm.add_constant(boston[['lstat', 'rm', 'nox', 'dis', 'ptratio',
anova_poly_log_model = sm.OLS(boston['medv'], anova_poly_log).fit()

anova_results = anova_lm(anova_linear_interaction_model, anova_poly_log_model)
print(anova_results, end="\n\n")


# INTERPRET THE RESULT
```

```
Residuals:  count    5.060000e+02
mean     6.434405e-09
std      4.075222e+00
min     -1.961512e+01
25%     -2.197839e+00
50%     -1.420940e-01
75%      1.863879e+00
max      2.891670e+01
dtype: float64
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.804
Model:                            OLS   Adj. R-squared:                  0.800
Method:                 Least Squares   F-statistic:                     202.6
Date:                Mon, 10 Feb 2025   Prob (F-statistic):          7.10e-168
Time:                        17:30:58   Log-Likelihood:                -1428.4
No. Observations:                 506   AIC:                             2879.
Df Residuals:                     495   BIC:                             2925.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        172.9866     13.954     12.397      0.000     145.571     200.402
lstat         -8.5527      1.227     -6.969      0.000     -10.964      -6.141
rm            25.1967      2.732      9.224      0.000      19.830      30.564
nox          -16.6408      2.734     -6.087      0.000     -22.012     -11.270
dis           -0.9709      0.141     -6.885      0.000      -1.248      -0.694
ptratio       -0.7843      0.097     -8.116      0.000      -0.974      -0.594
lstat^2        1.0064      0.178      5.654      0.000       0.657       1.356
lstat^3       -0.0582      0.011     -5.087      0.000      -0.081      -0.036
lstat^4        0.0015      0.000      4.672      0.000       0.001       0.002
lstat^5    -1.521e-05   3.52e-06     -4.323      0.000   -2.21e-05    -8.3e-06
log_rm      -137.4038     16.761     -8.198      0.000    -170.336    -104.472
==============================================================================
Omnibus:                      221.958   Durbin-Watson:                   1.064
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2718.500
Skew:                           1.567   Prob(JB):                         0.00
Kurtosis:                      13.914   Cond. No.                     9.63e+08
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 9.63e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
```

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	499.0	9500.381881	0.0	NaN	NaN	NaN
1	495.0	8386.756361	4.0	1113.62552	16.431997	1.190966e-12

For this model i took the logarithmic transformation of **rm** (log(rm)) which i could simply do by using the log() function from Numpy. I first tried including the **lstat_rm** predictor in the log-model but i didn't get the same results as the example but when i tried droping it from my list of predictors i did get a good result so this is what i used.

As we can see, we get further improvements on the R^2 and adjusted R^2 scores, finally breaking that 0.8 barrier. We can again see from the ANOVA test that this performance

improvement is statistically significant.

Use categorical predictors

We will now examine the carseats.csv data that contains a good portion of categorical data. Categorical data are columns in the CSV files that are non-numerical. For example, the 'ShelveLoc' indicates the quality of the shelving location for the carseats. This is, atleast in this dataset, not quantifiable on a numerical scale (for example 0-10 scale) so instead we have the categories *bad*, *good* and *medium* to indicate it's quality. This is what we would call a categorical variable or predictor. There is a nice source here that explains the whole dataset.

We start by just loading the dataset and getting an overview of the predictors.

In [93]:
```
# load boston.csv
carseats = pd.read_csv('Carseats.csv')

# Set pandas option to display all columns
pd.set_option('display.max_columns', None)

print(carseats.describe(), end="\n\n")

# I count the categorical separate instances manually as i couldnt find a good f
categorical_columns = carseats.select_dtypes(include=['object', 'category']).col

for col in categorical_columns:
    print(f"Summary for {col}:")
    print(carseats[col].value_counts(), end="\n\n")
```

```
       Unnamed: 0       Sales   CompPrice      Income  Advertising  \
count  400.000000  400.000000  400.000000  400.000000   400.000000
mean   200.500000    7.496325  124.975000   68.657500     6.635000
std    115.614301    2.824115   15.334512   27.986037     6.650364
min      1.000000    0.000000   77.000000   21.000000     0.000000
25%    100.750000    5.390000  115.000000   42.750000     0.000000
50%    200.500000    7.490000  125.000000   69.000000     5.000000
75%    300.250000    9.320000  135.000000   91.000000    12.000000
max    400.000000   16.270000  175.000000  120.000000    29.000000

       Population       Price         Age   Education
count  400.000000  400.000000  400.000000  400.000000
mean   264.840000  115.795000   53.322500   13.900000
std    147.376436   23.676664   16.200297    2.620528
min     10.000000   24.000000   25.000000   10.000000
25%    139.000000  100.000000   39.750000   12.000000
50%    272.000000  117.000000   54.500000   14.000000
75%    398.500000  131.000000   66.000000   16.000000
max    509.000000  191.000000   80.000000   18.000000

Summary for ShelveLoc:
ShelveLoc
Medium    219
Bad        96
Good       85
Name: count, dtype: int64

Summary for Urban:
Urban
Yes    282
No     118
Name: count, dtype: int64

Summary for US:
US
Yes    258
No     142
Name: count, dtype: int64



Summary for ShelveLoc:
ShelveLoc
Medium    219
Bad        96
Good       85
Name: count, dtype: int64

Summary for Urban:
Urban
Yes    282
No     118
Name: count, dtype: int64

Summary for US:
US
Yes    258
No     142
Name: count, dtype: int64
```

In [94]:

```
# Convert the categorical data to fit on numerical data model
carseats_all_predictors = pd.get_dummies(carseats.drop(columns=['Sales']), drop_
carseats_all_predictors = sm.add_constant(carseats_all_predictors)
carseats_all_predictors = carseats_all_predictors.astype(float)


y = pd.to_numeric(carseats['Sales'], errors='coerce') # combine the predictors a

model_carseats_sales_all = sm.OLS(y, carseats_all_predictors).fit()

# we get the residuals and print the summmary similar to before.
residuals_carseats_sales_all = model_carseats_sales_all.resid
print("Residuals Summary:")
print(residuals_carseats_sales_all.describe(), "\n")

print(model_carseats_sales_all.summary())

# This is how i would implement it if python could handle categorical data like

#model_carseats_sales_all = sm.OLS(carseats['Sales'], sm.add_constant(carseats.d

#residuals_carseats_sales_all = model_carseats_sales_all.resid
#print("Residuals: ", residuals_carseats_sales_all.describe(), end="\n\n")
#print(model_carseats_sales_all.summary()) # Gives the same output as the exampl
```

```
Residuals Summary:
count    4.000000e+02
mean    -4.252154e-16
std      1.004113e+00
min     -2.840893e+00
25%     -6.816797e-01
50%      1.268090e-02
75%      6.468280e-01
max      3.468381e+00
dtype: float64
```

OLS Regression Results

Dep. Variable:	Sales	R-squared:	0.874
Model:	OLS	Adj. R-squared:	0.870
Method:	Least Squares	F-statistic:	222.9
Date:	Mon, 10 Feb 2025	Prob (F-statistic):	1.97e-165
Time:	17:30:59	Log-Likelihood:	-568.72
No. Observations:	400	AIC:	1163.
Df Residuals:	387	BIC:	1215.
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.7286	0.611	9.375	0.000	4.527	6.930
Unnamed: 0	-0.0003	0.000	-0.724	0.470	-0.001	0.001
CompPrice	0.0930	0.004	22.366	0.000	0.085	0.101
Income	0.0157	0.002	8.422	0.000	0.012	0.019
Advertising	0.1239	0.011	11.078	0.000	0.102	0.146
Population	0.0002	0.000	0.582	0.561	-0.001	0.001
Price	-0.0954	0.003	-35.678	0.000	-0.101	-0.090
Age	-0.0462	0.003	-14.480	0.000	-0.052	-0.040
Education	-0.0225	0.020	-1.133	0.258	-0.061	0.017
ShelveLoc_Good	4.8520	0.153	31.666	0.000	4.551	5.153
ShelveLoc_Medium	1.9579	0.126	15.515	0.000	1.710	2.206
Urban_Yes	0.1278	0.113	1.129	0.260	-0.095	0.351
US_Yes	-0.1854	0.150	-1.236	0.217	-0.480	0.109

Omnibus:	1.027	Durbin-Watson:	2.017
Prob(Omnibus):	0.598	Jarque-Bera (JB):	0.923
Skew:	0.117	Prob(JB):	0.630
Kurtosis:	3.032	Cond. No.	4.84e+03

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.84e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

As we can see, because im dealing with categorical data here i couldnt just fit the data, print the summary and then print my residuals. R-coding allows for handling categorical data with minimum code but in python we have to handle this differently. We have to convert the categorical data in to binary numerical data using dummy coding. I can then fit the data similarily as before.

As we can see i got the same residuals, coefficients and performance metrics as the example. The R^2 is 0.873 which means that our model explain 87.3% of the variability in our result which is a pretty good result. We also got an F-statistic and a F-probability that strengthen my assumption that this a pretty well fit model.

In [95]:
```
predictor_df = carseats.drop(columns=['Sales'])

predictor_df['Income:Advertising'] = predictor_df['Income'] * predictor_df['Adve

predictor_df['Price:Age'] = predictor_df['Price'] * predictor_df['Age'] # Intera

predictor_df['Price_sq'] = predictor_df['Price'] ** 2 # Add a polynomial term.

predictors = pd.get_dummies(predictor_df, drop_first=True)
predictors = sm.add_constant(predictors)
predictors = predictors.astype(float)
y = pd.to_numeric(carseats['Sales'], errors='coerce')

model_full = sm.OLS(y, predictors).fit() # fit the model

print(model_full.resid.describe(), "\n") # realised i could just output everythi
print(model_full.summary())
```

```
count    4.000000e+02
mean     1.421085e-15
std      9.925373e-01
min     -2.885304e+00
25%     -7.618723e-01
50%      2.807875e-02
75%      6.541576e-01
max      3.325621e+00
dtype: float64
```

OLS Regression Results

Dep. Variable:	Sales	R-squared:	0.876
Model:	OLS	Adj. R-squared:	0.872
Method:	Least Squares	F-statistic:	181.7
Date:	Mon, 10 Feb 2025	Prob (F-statistic):	7.34e-164
Time:	17:30:59	Log-Likelihood:	-564.08
No. Observations:	400	AIC:	1160.
Df Residuals:	384	BIC:	1224.
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	7.3174	1.273	5.749	0.000	4.815	9.820
Unnamed: 0	-0.0003	0.000	-0.603	0.547	-0.001	0.001
CompPrice	0.0933	0.004	22.551	0.000	0.085	0.101
Income	0.0108	0.003	4.121	0.000	0.006	0.016
Advertising	0.0712	0.023	3.133	0.002	0.027	0.116
Population	0.0002	0.000	0.467	0.641	-0.001	0.001
Price	-0.1132	0.016	-7.252	0.000	-0.144	-0.082
Age	-0.0581	0.016	-3.639	0.000	-0.090	-0.027
Education	-0.0241	0.020	-1.212	0.226	-0.063	0.015
Income:Advertising	0.0007	0.000	2.679	0.008	0.000	0.001
Price:Age	0.0001	0.000	0.811	0.418	-0.000	0.000
Price_sq	5.35e-05	5.91e-05	0.905	0.366	-6.27e-05	0.000
ShelveLoc_Good	4.8442	0.153	31.627	0.000	4.543	5.145
ShelveLoc_Medium	1.9548	0.126	15.524	0.000	1.707	2.202
Urban_Yes	0.1405	0.113	1.246	0.213	-0.081	0.362
US_Yes	-0.1531	0.149	-1.026	0.306	-0.447	0.140

```
Omnibus:                        1.385   Durbin-Watson:                   2.051
Prob(Omnibus):                  0.500   Jarque-Bera (JB):                1.252
Skew:                           0.135   Prob(JB):                        0.535
Kurtosis:                       3.048   Cond. No.                     4.08e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.08e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.876
Model:                            OLS   Adj. R-squared:                  0.872
Method:                 Least Squares   F-statistic:                     181.7
Date:                Mon, 10 Feb 2025   Prob (F-statistic):          7.34e-164
Time:                        17:30:59   Log-Likelihood:                -564.08
No. Observations:                 400   AIC:                             1160.
Df Residuals:                     384   BIC:                             1224.
Df Model:                          15
Covariance Type:            nonrobust
==============================================================================
=====
                         coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-----
const                  7.3174      1.273      5.749      0.000       4.815
9.820
Unnamed: 0            -0.0003      0.000     -0.603      0.547      -0.001
0.001
CompPrice              0.0933      0.004     22.551      0.000       0.085
0.101
Income                 0.0108      0.003      4.121      0.000       0.006
0.016
Advertising            0.0712      0.023      3.133      0.002       0.027
0.116
Population             0.0002      0.000      0.467      0.641      -0.001
0.001
Price                 -0.1132      0.016     -7.252      0.000      -0.144      -
0.082
Age                   -0.0581      0.016     -3.639      0.000      -0.090      -
0.027
Education             -0.0241      0.020     -1.212      0.226      -0.063
0.015
Income:Advertising     0.0007      0.000      2.679      0.008       0.000
0.001
Price:Age              0.0001      0.000      0.811      0.418      -0.000
0.000
Price_sq            5.35e-05   5.91e-05      0.905      0.366   -6.27e-05
0.000
ShelveLoc_Good         4.8442      0.153     31.627      0.000       4.543
5.145
ShelveLoc_Medium       1.9548      0.126     15.524      0.000       1.707
2.202
Urban_Yes              0.1405      0.113      1.246      0.213      -0.081
0.362
US_Yes                -0.1531      0.149     -1.026      0.306      -0.447
0.140
```

```
==============================================================================
Omnibus:                        1.385   Durbin-Watson:                   2.051
Prob(Omnibus):                  0.500   Jarque-Bera (JB):                1.252
Skew:                           0.135   Prob(JB):                        0.535
Kurtosis:                       3.048   Cond. No.                     4.08e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.08e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

So for this task i didn't manage to build a model that performs better on *all* of the metrics. INstead i just decided to focus on one of the metrics, which in this case is the $R^2 score$ and the adjusted $R^2 score$. I managed to build a model that gets a adjusted R-squared of 0.872 which is slightly better then the model that was proposed by the teacher which got an R-squared of 0.8698. However, the F-statistic that the teacher got was higher which means that my model has a smaller explanatory power than the teachers. This is an indicator that my model actually might be overfitted compared to the teachers.

I don't have a good explanation as to how i actually found this model though. I found this by trial and error. I tried using the stuff that i learnt from this assignment so what i did created an interaction between income and advertising and price and age.

I used the interaction between income and age because this is a relationship that the teacher actually discussed on a lecture but instead of carseats it was TV's so i felt like that relationship was relevant.

I then also included the price column to the power of two and then included all other plain predictors.