```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import seaborn as sns

import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.formula.api as smf
from statsmodels.stats.anova import anova_lm

!pip install ISLP
from ISLP import load_data
from ISLP.models import ( ModelSpec as MS ,summarize, poly)
```

Show hidden output

8a. Use the sm.OLS() function to perform a simple linear regression with mpg as the response and horsepower as the predictor. Use the summarize() function to print the results.

```
Auto = load_data("Auto")
Auto.columns
X =  X = pd.DataFrame({'intercept':np.ones(Auto.shape [0]) ,
'horsepower': Auto['horsepower']})
Y = Auto['mpg']
model = sm.OLS(Y, X)
results = model.fit()
summarize(results)
```

	coef	std err	t	P>|t|
intercept	39.9359	0.717	55.660	0.0
horsepower	-0.1578	0.006	-24.489	0.0

From the results summary we can see that there is a relationship between horsepower and mpg. There is a relationship between the two as the t critical values tells us that we can reject the null hypothesis of the predictor having a 0 magnitude effect on the outcome variable. However for every 10 unit increase in horsepower the mpg of the car is only predicted to decrease by 1.578 mpg which does show a weak affect that the predictor has on mpg.The relationship between the two is negative.

```
design = MS(['horsepower'])
design = design.fit(Auto)
X = design.transform(Auto)
model = sm.OLS(Y, X)
new_df = pd.DataFrame ({'horsepower':[98]})
newX = design. transform (new_df)
newX
new_predictions = results. get_prediction (newX);
print(new_predictions.predicted_mean)
print(new_predictions.conf_int(alpha =0.05))
```

```
[24.46707715]
[[23.97307896 24.96107534]]
```

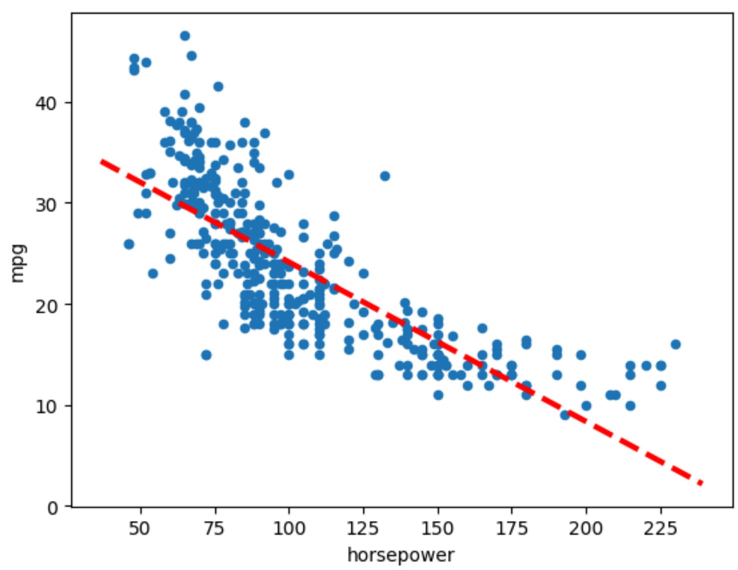
The predicted mpg at 98 horsepower is 24.467 mpg. With 95% confidence we can say the true value is between 23.973 and 24.961.

8b - Plot the response and the predictor in a new set of axesax. Use the ax.axline() method or the abline() function defined in the lab to display the least squares regression line.

```
def abline(ax , b, m, *args , ** kwargs):
  "Add a line with slope m and intercept b to ax"
  xlim = ax. get_xlim ()
  ylim = [m * xlim [0] + b, m * xlim [1] + b]
  ax.plot(xlim , ylim , *args , ** kwargs)
```

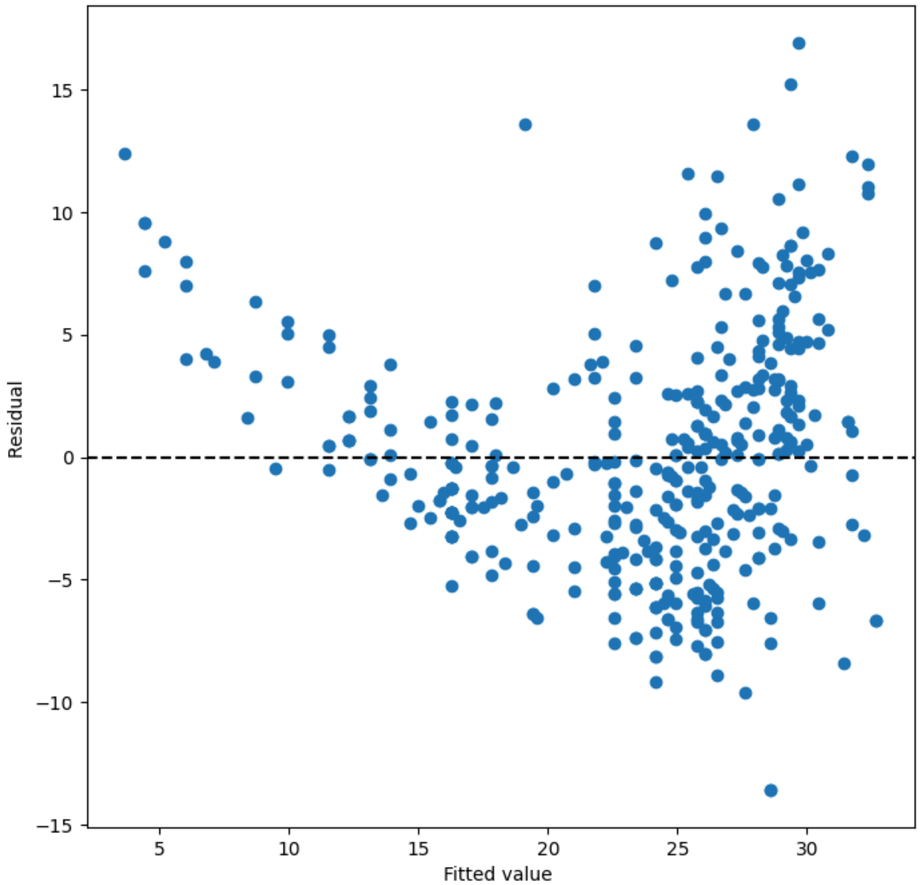
```
ax = Auto.plot.scatter('horsepower', 'mpg')
abline(ax ,
       results.params [0],
       results.params [1],
       'r--',
       linewidth =3)
```

```
<ipython-input-498-de519c427683>:11: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future
  results.params [0],
<ipython-input-498-de519c427683>:12: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future
  results.params [1],
```



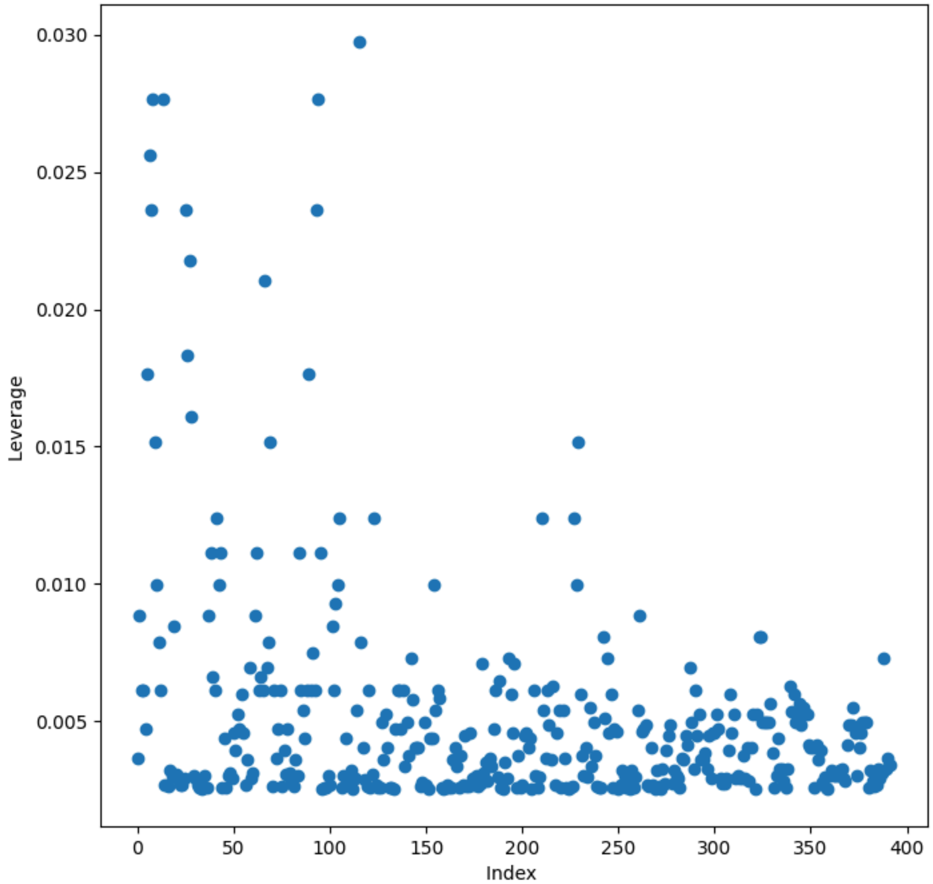
8c - plot diagnostic plots

```
ax = plt.subplots(figsize =(8 ,8))[1]
ax.scatter(results.fittedvalues , results.resid)
ax. set_xlabel ('Fitted value ')
ax. set_ylabel ('Residual ')
ax.axhline (0, c='k', ls='--');
```




```
infl = results. get_influence ()
ax = plt.subplots (figsize =(8 ,8))[1]
ax.scatter(np.arange(X.shape [0]) , infl. hat_matrix_diag )
ax. set_xlabel ('Index ')
ax. set_ylabel ('Leverage ')
np.argmax(infl. hat_matrix_diag )
```

115



Residual seems to grow at a higher rate for increase levels of fitted values which can indicate inaccuracy in the relationship and our productionb model. At lower index levels there is also higher leverage which could also signify more inaccuracy or undefitting of our line.

9a - Produce a scatterplot matrix which includes all of the variables in the data set.

```
sns.pairplot(Auto)
```

```
<seaborn.axisgrid.PairGrid at 0x79b17c76d110>
```

9b - Compute the matrix of correlations between the variables using the DataFrame.corr() method.

```
Auto.corr()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
mpg	1.000000	-0.777618	-0.805127	-0.778427	-0.832244	0.423329	0.580541	0.565209
cylinders	-0.777618	1.000000	0.950823	0.842983	0.897527	-0.504683	-0.345647	-0.568932
displacement	-0.805127	0.950823	1.000000	0.897257	0.932994	-0.543800	-0.369855	-0.614535
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361	-0.455171
weight	-0.832244	0.897527	0.932994	0.864538	1.000000	-0.416839	-0.309120	-0.585005
acceleration	0.423329	-0.504683	-0.543800	-0.689196	-0.416839	1.000000	0.290316	0.212746
year	0.580541	-0.345647	-0.369855	-0.416361	-0.309120	0.290316	1.000000	0.181528
origin	0.565209	-0.568932	-0.614535	-0.455171	-0.585005	0.212746	0.181528	1.000000

9c - Use the sm.OLS() function to perform a multiple linear regression with mpg as the response and all other variables except name as the predictors.

I tried doing this method the lab did in the book but have an issue as Pandas and this method seem incompatible even though the book says it is possible. So I am doing the following method I found online.

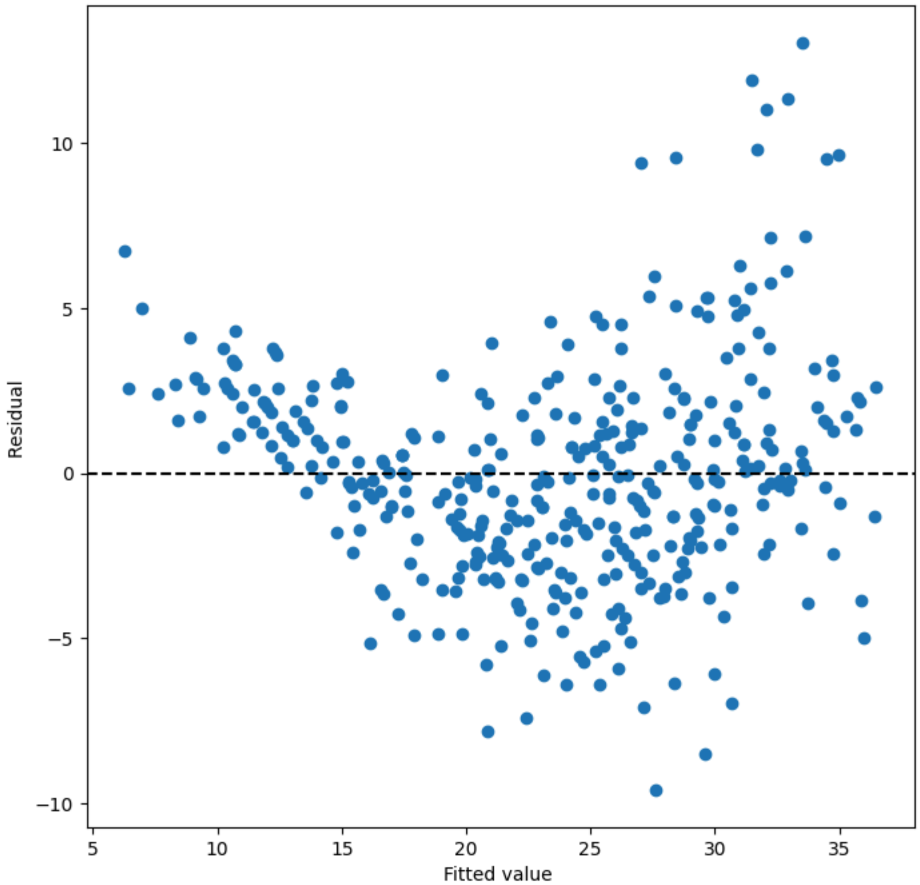
```
formula = 'mpg ~ cylinders + displacement + horsepower + weight + acceleration + year + origin'
multi_model = smf.ols(formula, data=Auto)
results_multi = multi_model.fit()
anova_lm(results_multi)
```

	df	sum_sq	mean_sq	F	PR(>F)
cylinders	1.0	14403.083079	14403.083079	1300.683788	2.319511e-125
displacement	1.0	1073.344025	1073.344025	96.929329	1.530906e-20
horsepower	1.0	403.408069	403.408069	36.430140	3.731128e-09
weight	1.0	975.724953	975.724953	88.113748	5.544461e-19
acceleration	1.0	0.966071	0.966071	0.087242	7.678728e-01
year	1.0	2419.120249	2419.120249	218.460900	1.875281e-39
origin	1.0	291.134494	291.134494	26.291171	4.665681e-07
Residual	384.0	4252.212530	11.073470	NaN	NaN

From the data we can see that most of the variables besides accelration seem to have a strong correlation to the multi model. we fail to reject a 0 magnitude effect from acceleration in this model. Cylinders, Displacment, Weight, Horsepower adn Year seemed to have the most statistical significance in impacting mpg. The coefficient means that for a car that is newer by a year (has an 1 increase in the year value) the mpg saw an increase by about 0.58 mpg.

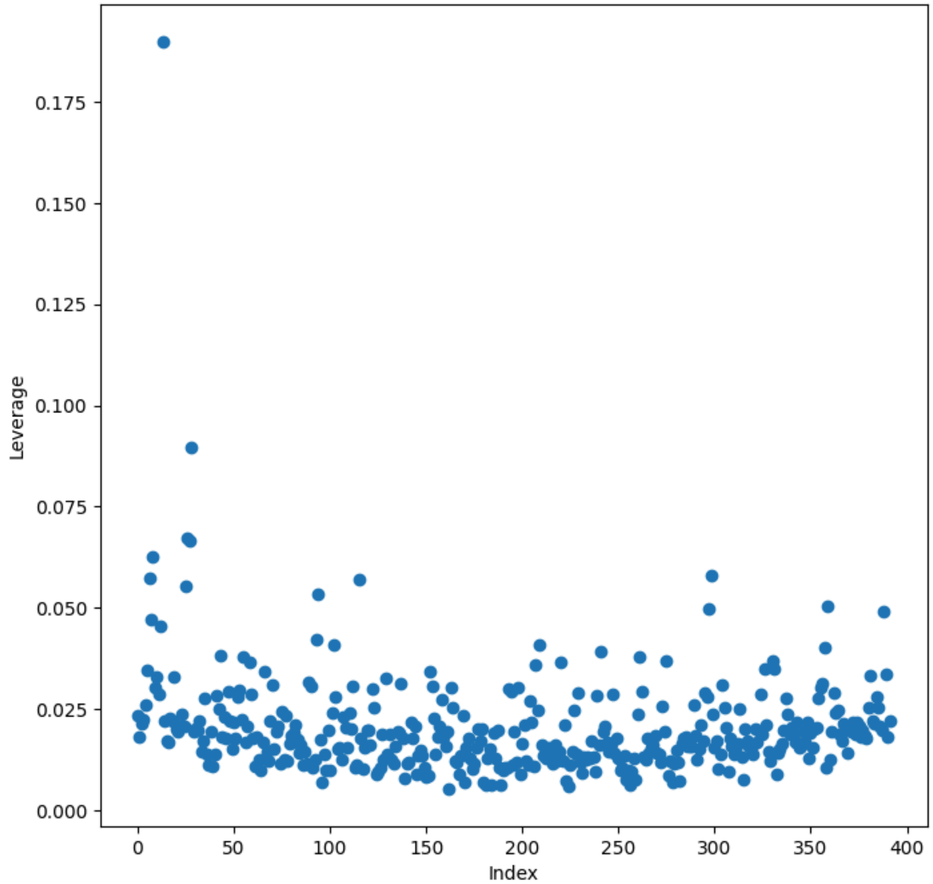
9d - Produce some of diagnostic plots of the linear regression fit as described in the lab. Comment on any problems you see with the fit. Do the residual plots suggest any unusually large outliers? Does the leverage plot identify any observations with unusually high leverage?

```
ax = plt.subplots(figsize =(8 ,8))[1]
ax.scatter(results_multi.fittedvalues , results_multi.resid)
ax.set_xlabel ('Fitted value ')
ax.set_ylabel ('Residual ')
ax.axhline (0, c='k', ls='--');
```




```
infl = results_multi.get_influence()
ax = plt.subplots(figsize =(8 ,8))[1]
ax.scatter(np.arange(X.shape[0]), infl.hat_matrix_diag)
ax.set_xlabel('Index')
ax.set_ylabel('Leverage')
np.argmax(infl.hat_matrix_diag)
```

13



There was high residulas at lower and higher fitted values aswell as higher leverage for the lower indexes which seemed like a big outlier. However the leverage seeems rather consistent and not out of normal levels except for one outlier which had abnormally higher leverage than other values. This could influence the trend our model visualizes which could stray away from the true model.

9e - Fit some models with interactions as described in the lab. Do any interactions appear to be statistically significant?

```
formula_interaction = 'mpg ~ cylinders * displacement + horsepower * weight + acceleration * year + origin'
interaction_model = smf.ols(formula_interaction, data=Auto)
results_interaction = interaction_model.fit()
print(results_interaction.summary())
anova_lm(results_interaction,results_multi)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.875
Model:                            OLS   Adj. R-squared:                  0.871
Method:                 Least Squares   F-statistic:                     266.1
Date:                Thu, 06 Feb 2025   Prob (F-statistic):          4.42e-165
Time:                        04:14:39   Log-Likelihood:                -953.98
No. Observations:                 392   AIC:                             1930.
Df Residuals:                     381   BIC:                             1974.
Df Model:                          10
Covariance Type:            nonrobust
============================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
Intercept                111.4537     18.485      6.030      0.000      75.109     147.798
cylinders                 -0.0415      0.477     -0.087      0.931      -0.980       0.897
displacement              -0.0143      0.015     -0.939      0.348      -0.044       0.016
cylinders:displacement     0.0015      0.002      0.707      0.480      -0.003       0.006
horsepower                -0.2295      0.026     -8.848      0.000      -0.280      -0.178
weight                    -0.0102      0.001    -11.380      0.000      -0.012      -0.008
horsepower:weight       5.151e-05   6.68e-06      7.714      0.000    3.84e-05    6.46e-05
acceleration              -6.9334      1.151     -6.023      0.000      -9.197      -4.670
year                      -0.6434      0.240     -2.679      0.008      -1.116      -0.171
acceleration:year          0.0892      0.015      5.978      0.000       0.060       0.119
origin                     0.6422      0.248      2.588      0.010       0.154       1.130
==============================================================================
Omnibus:                       40.489   Durbin-Watson:                   1.603
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               86.168
Skew:                           0.563   Prob(JB):                     1.94e-19
Kurtosis:                       5.002   Cond. No.                     5.30e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.3e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	381.0	2982.899441	0.0	NaN	NaN	NaN
1	384.0	4252.212530	-3.0	-1269.313089	38.208832	NaN

Horsepower x Weight and Acceleration x Year were the only two that seemed to be statistically significant with cylinders x displacement having a low t value meaning we can't rule out the possibility of a 0 coefficient effect.

9f - Try a few different transformations of variables and comment findings.

```
Auto_transformed = Auto.copy()


Auto_transformed['log_horsepower'] = np.log(Auto_transformed['horsepower'])
Auto_transformed['sqrt_weight'] = np.sqrt(Auto_transformed['weight'])
Auto_transformed['displacement_sq'] = Auto_transformed['displacement'] ** 2

# Define formulas with transformed variables
transform_formulas = [
    'mpg ~ log_horsepower + weight + acceleration + year + origin',
    'mpg ~ horsepower + sqrt_weight + acceleration + year + origin',
    'mpg ~ cylinders + displacement_sq + horsepower + weight + acceleration + year + origin',

]

for formula in transform_formulas:
    model = smf.ols(formula, data=Auto_transformed)
    results = model.fit()
    print(f"Formula: {formula}")
    print(results.summary())
    print("\n" + "=" * 50 + "\n")
```

```
sqrt_weight     -0.6892      0.053    -13.004      0.000      -0.793      -0.585
acceleration     0.0948      0.094      1.012      0.312      -0.089       0.279
year             0.7595      0.049     15.424      0.000       0.663       0.856
origin           0.9309      0.256      3.631      0.000       0.427       1.435
==============================================================================
Omnibus:                       41.349   Durbin-Watson:                   1.295
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               81.065
Skew:                           0.602   Prob(JB):                     2.49e-18
Kurtosis:                       4.874   Cond. No.                     3.91e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.91e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

==================================================

Formula: mpg ~ cylinders + displacement_sq + horsepower + weight + acceleration + year + origin
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.836
Model:                            OLS   Adj. R-squared:                  0.833
Method:                 Least Squares   F-statistic:                     279.9
Date:                Thu, 06 Feb 2025   Prob (F-statistic):          1.57e-146
Time:                        04:14:39   Log-Likelihood:                -1006.7
No. Observations:                 392   AIC:                             2029.
Df Residuals:                     384   BIC:                             2061.
Df Model:                           7
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
Intercept         -13.7793      4.488     -3.070      0.002     -22.604      -4.954
cylinders          -0.7083      0.261     -2.710      0.007      -1.222      -0.194
displacement_sq  6.951e-05   1.07e-05      6.475      0.000    4.84e-05    9.06e-05
horsepower         -0.0425      0.014     -3.075      0.002      -0.070      -0.015
weight             -0.0064      0.001    -11.024      0.000      -0.008      -0.005
acceleration        0.0747      0.094      0.792      0.429      -0.111       0.260
year                0.7644      0.049     15.646      0.000       0.668       0.860
origin              1.3374      0.254      5.271      0.000       0.838       1.836
==============================================================================
Omnibus:                       28.628   Durbin-Watson:                   1.392
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               48.813
Skew:                           0.473   Prob(JB):                     2.51e-11
Kurtosis:                       4.447   Cond. No.                     1.93e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.93e+06. This might indicate that there are
strong multicollinearity or other numerical problems.

==================================================
```

The transformed variables log_horsepower and sqrt_weight appear to provide a slightly improved fit without drastically changin the model's interpretation. The log transformation of horsepower shows that there may be a non linear relationship where an increase in horsepower has a diminishing effect on mpg while meaning smaller engines experience greater efficiency losses than larger ones. The square root transformation of weight showed a diminishing marginal impact of weight on fuel efficiency with the reinforcing the theory that heavier cars experience decreasing fuel efficiency losses as weight increases. These transformations help stabilize variance and may reduce multicollinearity.

13a - Using the normal() method of your random number generator create a vector, x, containing 100 observations drawn from a N(0,1) distribution this represents a feature X.

```
x = np.random.normal(0,1,100)
print(x)
len(x)
```

```
[-0.71896528 -0.34378447 -1.90126314 -0.7166678  -1.09313582 -0.53335378
  0.36020486 -1.19199502 -0.46485476  0.12300601 -0.41268306  2.07004512
 -1.85527234 -0.38975872 -0.14004807  0.31852385 -0.27715562 -0.74594951
  0.50877153 -0.12250746  1.56864964 -0.49433578  0.91988573  0.95886672
  1.44460813 -0.09978292 -0.00258054 -0.1969089  -0.14392027  0.94025683
  0.62119071  0.03390489 -0.04505755 -0.40027573 -0.80040501 -0.25120053
 -0.05990054  1.23542633 -0.73200465  1.62820874 -0.4633571   0.14388312
 -0.66630152 -0.10244666  0.35033646  1.39209807 -0.10236811 -0.105457
 -0.13125279  0.71040822  0.1587125   0.21055242 -0.26544538 -1.63809408
  0.69822819  1.12406972 -0.00695001  1.71591774 -0.72964932 -0.17485043
 -0.70993145 -0.7463968   0.26591789 -1.53825153 -1.52359683 -0.57863163
 -1.03810944  0.95843485  0.53822113  1.26119733  0.70161703  1.6376215
 -0.43924641  1.48685961 -0.50640188  1.27062862  1.23771299  0.74896209
  1.13169073  0.54482615  0.48997149 -0.87356548  0.26994964  0.56545755
  1.13469213  0.41607992  0.3431138   0.50452094 -0.14874546 -1.13711573
 -0.35623618  0.18688936  0.01947741  0.88116399 -0.0863606   1.07334577
 -0.166613   -1.4202214   0.81922492 -1.0800552 ]
100
```

13b - Using normal method create a vectore, eps, contining 100 obs drawn from a N(0,0.25) dist.

```
eps = np.random.normal(0,0.5,100)
print(eps)
```

```
[ 0.16371882  0.14657842 -1.42959459  0.64502854 -0.38839181 -0.01002392
 -0.18145081  0.62021593  0.40304532 -0.37946853 -0.06142914  0.73286436
  0.27015559  0.03842355  0.18149251  0.26064104  0.49729375  0.31701974
 -0.20824801  0.94967055  1.10949837  0.41602695 -0.32726152  0.40577725
 -0.42074245 -0.33570135  0.45977608  0.18559324 -0.24101175 -0.56326801
 -0.14770939  0.38827517  0.2843148   0.15414563 -0.2368309  -0.68696089
  0.11772286  0.10077489  1.01509441 -0.1970753  -0.9119012   0.10638436
  0.4113634  -0.57063902 -1.15049578 -0.22045401 -0.33900044  0.45784901
 -0.10998022 -0.94086333 -0.70753865 -0.09717885 -0.28995288 -0.88726563
 -0.30499893  0.16120043  0.1872182  -0.18922199 -0.58117743 -0.93478573
  0.63918291 -0.7934551   0.11752694  0.26534985  0.02456851  0.07633705
  0.611251   -0.20234985  0.61669537  0.24871942  0.41560693 -0.48441315
 -0.54254096 -0.55736905 -0.79325342  0.53289726 -0.18547117  0.88306492
  0.89730986 -0.23323246 -0.62885952  0.88866358  0.66386174  0.41153587
 -0.19094446  0.48706131 -0.1288325   0.28864111 -0.37275201 -0.05064927
  0.78188331  0.54705889 -0.52219712 -0.42788504  0.35393427  0.81940282
 -1.08450814 -0.12125089  0.18746051 -0.03691782]
```

13c - using x and eps, generate a vector y according to model:

Y = -1 + 0.5X +e

what is the lenght of the vector? what are the values of b0 and b1 in this linear model

```
y = -1 + (x/2) + eps
print(y)
len(y)
```

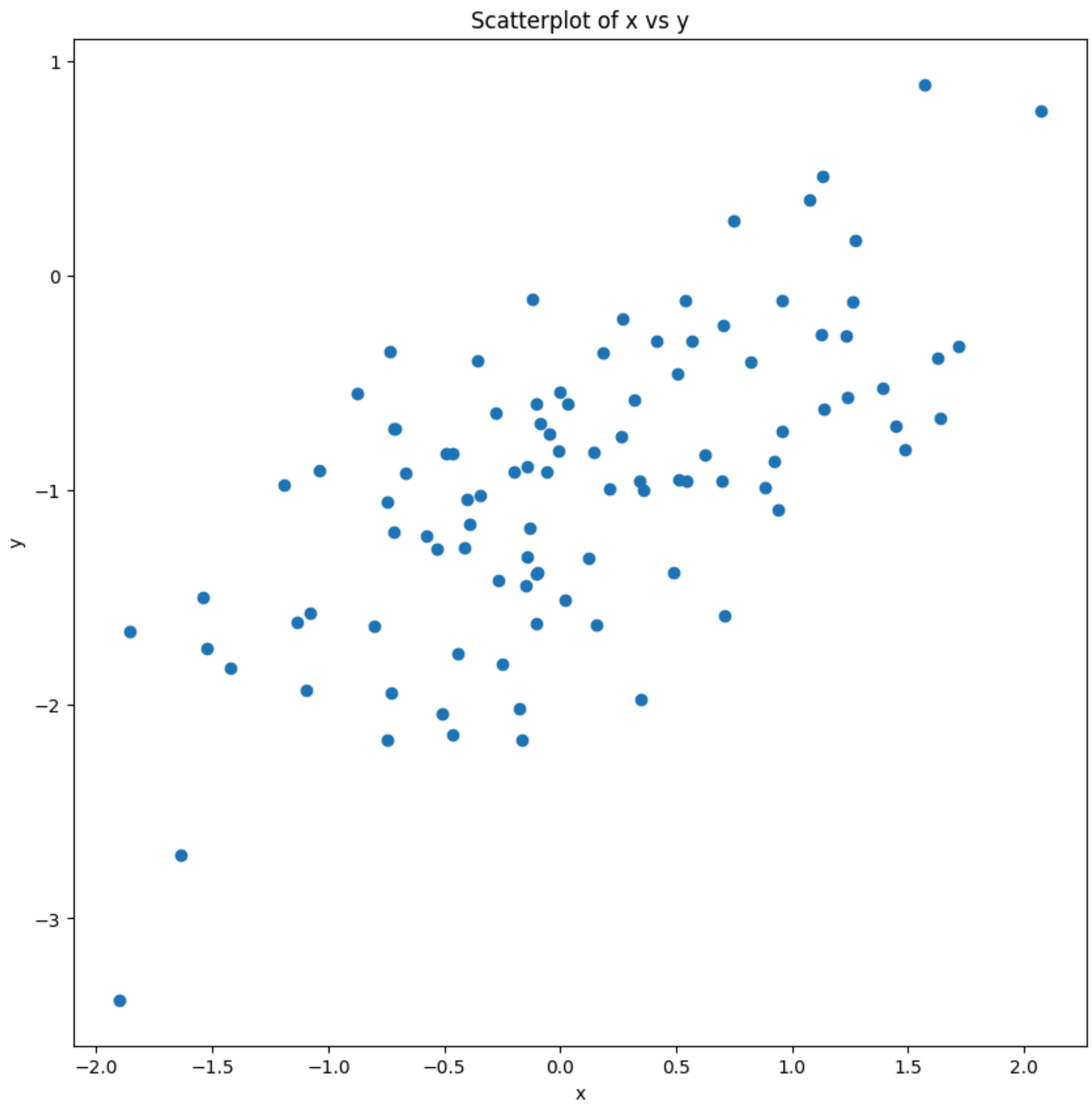
```
[-1.19576382 -1.02531381 -3.38022616 -0.71330536 -1.93495972 -1.27670081
 -1.00134838 -0.97578158 -0.82938206 -1.31796552 -1.26777067  0.76788692
 -1.65748058 -1.15645581 -0.88853153 -0.58009703 -0.64128407 -1.05595502
 -0.95386225 -0.11158318  0.89382319 -0.83114094 -0.86731866 -0.11478939
 -0.69843839 -1.38559281 -0.54151419 -0.91286121 -1.31297188 -1.09313959
 -0.83711404 -0.59477238 -0.73821398 -1.04599223 -1.63703341 -1.81256116
```

```
 -0.9122274  -0.28151195 -0.35090792 -0.38297093 -2.14357975 -0.82167408
 -0.92178736 -1.62186234 -1.97532755 -0.52440498 -1.3901845  -0.59487949
 -1.17560662 -1.58565922 -1.6281824  -0.99190264 -1.42267557 -2.70631267
 -0.95588484 -0.27676471 -0.81625681 -0.33126312 -1.94600209 -2.02221095
 -0.71578282 -2.1666535  -0.74951411 -1.50377591 -1.7372299  -1.21297876
 -0.90780372 -0.72313242 -0.11419407 -0.12068191 -0.23358455 -0.6656024
 -1.76216417 -0.81393924 -2.04645435  0.16821157 -0.56661467  0.25754596
  0.46315523 -0.96081939 -1.38387378 -0.54811916 -0.20116344 -0.30573535
 -0.62359839 -0.30489873 -0.9572756  -0.45909842 -1.44712474 -1.61920713
 -0.39623478 -0.35949643 -1.51245842 -0.98730305 -0.68924603  0.3560757
 -2.16781464 -1.83136159 -0.40292703 -1.57694541]
100
```

The lenght of the vector is 100 meaning it has 100 values within it. B0 is equal to -1 from the model we made above and b1 is 1/2

13d - Create a scatterplot displaying the relationship betweenx and y. Comment on what you observe.

```
plt.figure(figsize=(10, 10))
plt.scatter(x ,y )
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatterplot of x vs y')
plt.show()
```



This is a routine linear relationship with some noise around it. We can observe that the relationship looks almost perfectly linear and the data starts in the negative quadrant of the plot.

13e - Fit a least squares linear model to predict y using x. Comment on the model obtained. How do ^β0 and ^β1 compare to β0 and β1?

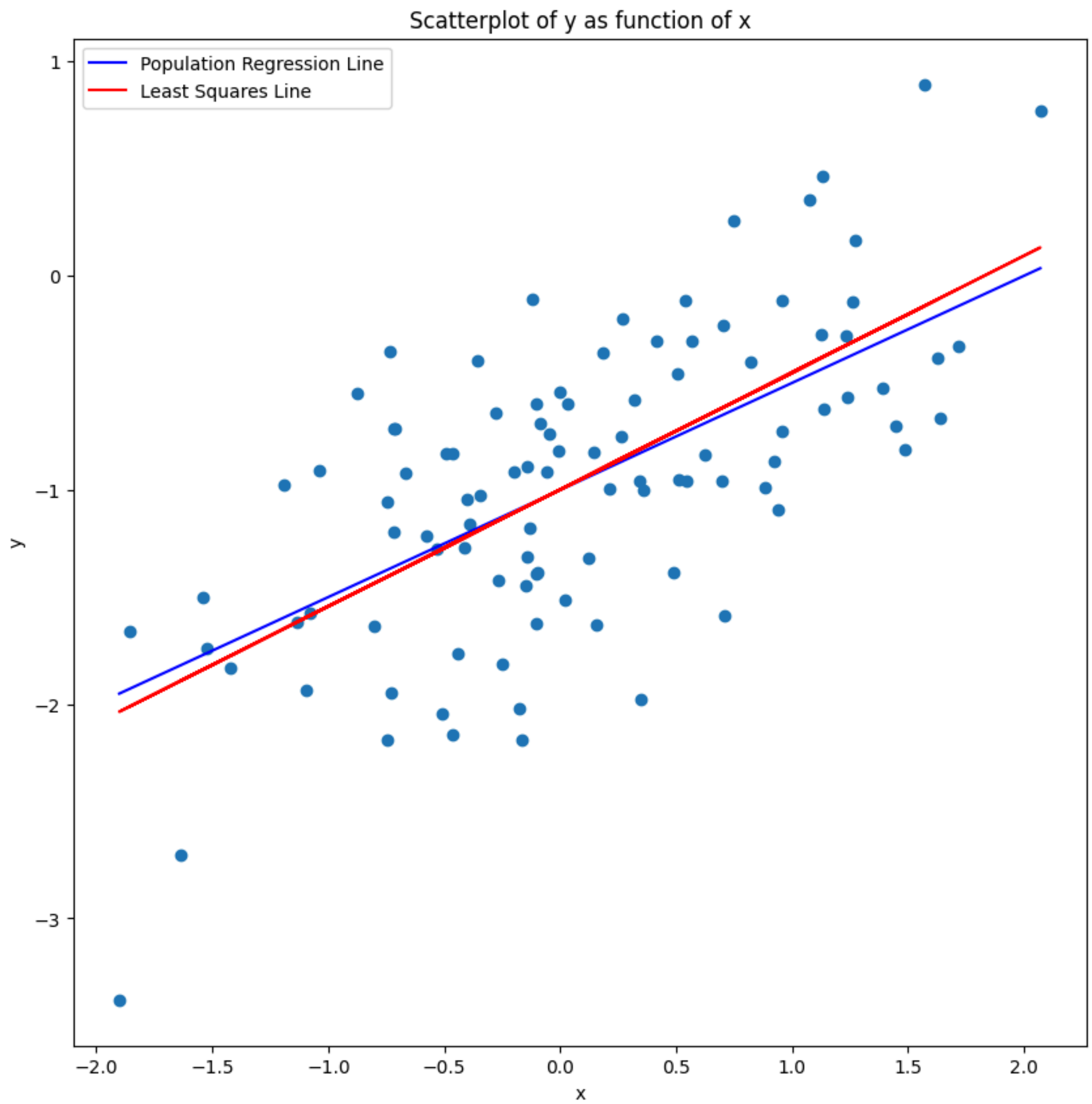
```
x = sm.add_constant(x)
model = sm.OLS(y,x)
results = model.fit()
print(results.summary())

b0_pred = results.params[0]
b1_pred = results.params[1]
print(f"Predicted b0: {b0_pred}")
print(f"Predicted b1: {b1_pred}")

print(f"Actual b0: -1")
print(f"Actual b1: 1/2")
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.440
Model:                            OLS   Adj. R-squared:                  0.434
Method:                 Least Squares   F-statistic:                     77.02
Date:                Thu, 06 Feb 2025   Prob (F-statistic):           5.46e-14
Time:                        04:14:39   Log-Likelihood:                -77.747
No. Observations:                 100   AIC:                             159.5
Df Residuals:                      98   BIC:                             164.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.9978      0.053    -18.700      0.000      -1.104      -0.892
x1             0.5457      0.062      8.776      0.000       0.422       0.669
==============================================================================
Omnibus:                        1.916   Durbin-Watson:                   1.842
Prob(Omnibus):                  0.384   Jarque-Bera (JB):                1.671
Skew:                          -0.187   Prob(JB):                        0.434
Kurtosis:                       2.488   Cond. No.                         1.19
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Predicted b0: -0.9977553842471643
Predicted b1: 0.5456980683513764
Actual b0: -1
Actual b1: 1/2
```

The actual and predicted coefficients were very close however not equal. there was an extremely small amount of error in both terms however they did capture the general trend of the data while be very clsoe to the true values.

13f - Display the least squares line on the scatterplot obtained in (d). Draw the population regression line on the plot, in a different color. Use the legend() method of the axes to create an **appropriate** legend.

```
pred_y = results.predict(x)
plt.figure(figsize=(10, 10))
plt.scatter(x[:,1] ,y )
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatterplot of y as function of x')

x_pop = np.linspace(x[:, 1].min(), x[:, 1].max(), 100)
y_pop = -1 + 0.5 * x_pop


plt.plot(x_pop, y_pop, color='blue', label='Population Regression Line')
plt.plot(x[:,1], pred_y, color='red', label='Least Squares Line')
plt.legend()
plt.show()
```



13g - Now fit a polynomial regression model that predicts y using x and x2. Is there evidence that the quadratic term improves the model fit? Explain your answer.

```
x_squared = x[:, 1]**2
X_poly = np.column_stack((x, x_squared))
model_poly = sm.OLS(y, X_poly)
results_poly = model_poly.fit()
print(results_poly.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.441
Model:                            OLS   Adj. R-squared:                  0.429
Method:                 Least Squares   F-statistic:                     38.20
Date:                Thu, 06 Feb 2025   Prob (F-statistic):           5.82e-13
Time:                        04:14:40   Log-Likelihood:                -77.699
No. Observations:                 100   AIC:                             161.4
Df Residuals:                      97   BIC:                             169.2
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.9850      0.068    -14.453      0.000      -1.120      -0.850
x1             0.5484      0.063      8.689      0.000       0.423       0.674
x2            -0.0176      0.058     -0.304      0.762      -0.133       0.097
==============================================================================
```

```
Omnibus:                        1.990   Durbin-Watson:                   1.835
Prob(Omnibus):                  0.370   Jarque-Bera (JB):                1.660
Skew:                          -0.168   Prob(JB):                        0.436
Kurtosis:                       2.466   Cond. No.                         2.14
==============================================================================

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

According to the summary statistics it seems that there is vert little to assume that the x^2 variable contributed to our model with a t value that is very low not ruling out that it has a 0 magnitude effect on the y value.

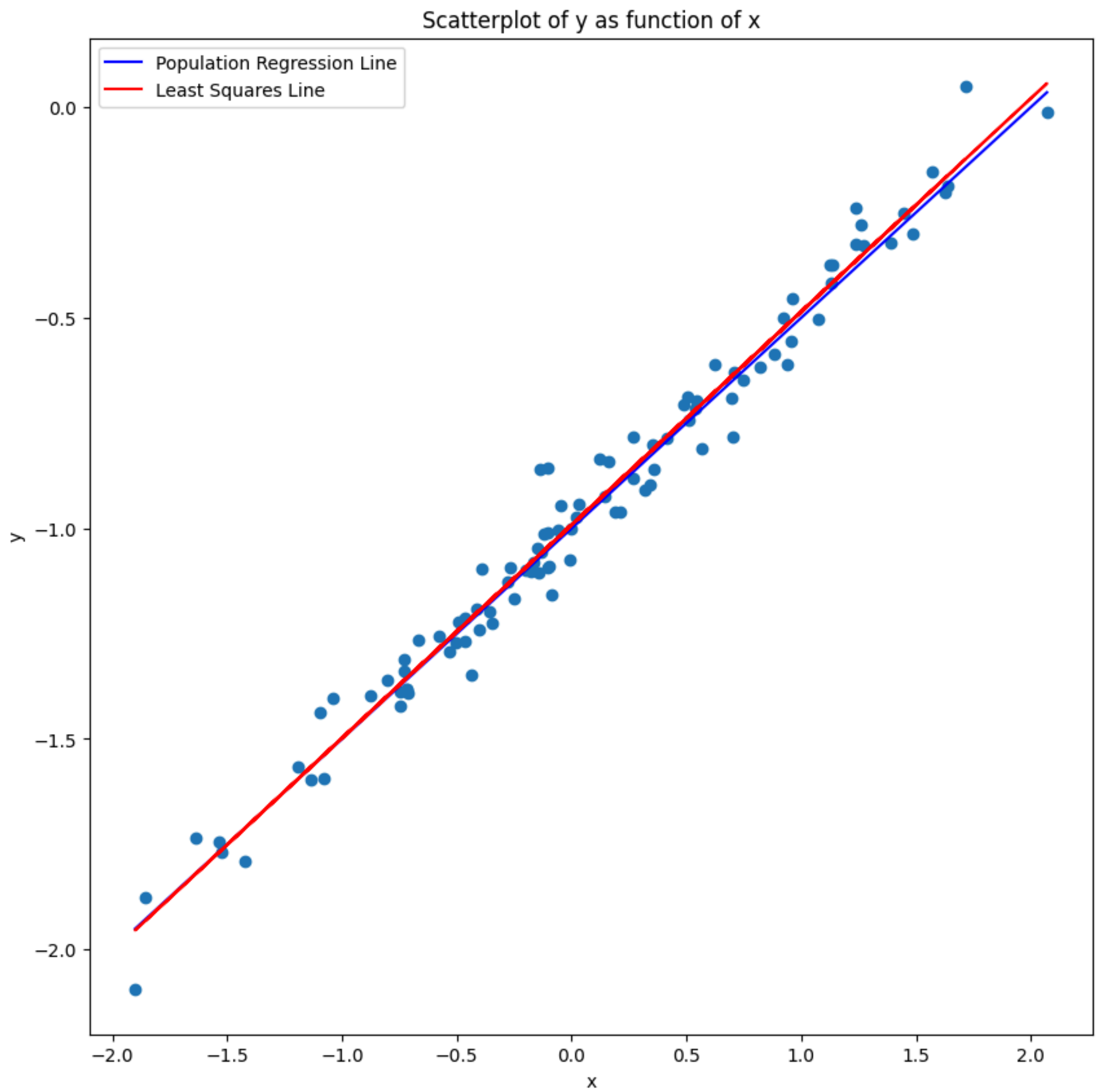
13h - Repeat (a)–(f) after modifying the data generation process in such a way that there is less noise in the data. The model (3.39) should remain the same. You can do this by decreasing the variance of the normal distribution used to generate the error term ϵ in (b). Describe your results.

```
eps_reduced = np.random.normal(0,0.0625,100)
y = -1 + (x[:, 1]/2) + eps_reduced
model = sm.OLS(y,x)
results = model.fit()
print(results.summary())

x = sm.add_constant(x)
model = sm.OLS(y,x)
results = model.fit()
print(results.summary())

b0_pred = results.params[0]
b1_pred = results.params[1]
print(f"Predicted b0: {b0_pred}")
print(f"Predicted b1: {b1_pred}")

print(f"Actual b0: -1")
print(f"Actual b1: 1/2")
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.978
Model:                            OLS   Adj. R-squared:                  0.978
Method:                 Least Squares   F-statistic:                     4393.
Date:                Thu, 06 Feb 2025   Prob (F-statistic):           3.29e-83
Time:                        04:14:40   Log-Likelihood:                 131.94
No. Observations:                 100   AIC:                            -259.9
Df Residuals:                      98   BIC:                            -254.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.9916      0.007   -151.289      0.000      -1.005      -0.979
x1             0.5062      0.008     66.276      0.000       0.491       0.521
==============================================================================
Omnibus:                        6.682   Durbin-Watson:                   1.863
Prob(Omnibus):                  0.035   Jarque-Bera (JB):                6.428
Skew:                           0.474   Prob(JB):                       0.0402
Kurtosis:                       3.802   Cond. No.                         1.19
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.978
Model:                            OLS   Adj. R-squared:                  0.978
Method:                 Least Squares   F-statistic:                     4393.
Date:                Thu, 06 Feb 2025   Prob (F-statistic):           3.29e-83
```

```
Time:                        04:14:40   Log-Likelihood:                 131.94
No. Observations:                 100   AIC:                            -259.9
Df Residuals:                      98   BIC:                            -254.7
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.9916      0.007   -151.289      0.000      -1.005      -0.979
x1             0.5062      0.008     66.276      0.000       0.491       0.521
==============================================================================
Omnibus:                        6.682   Durbin-Watson:                   1.863
Prob(Omnibus):                  0.035   Jarque-Bera (JB):                6.428
Skew:                           0.474   Prob(JB):                       0.0402
Kurtosis:                       3.802   Cond. No.                         1.19
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Predicted b0: -0.9916142361101516
Predicted b1: 0.5062396638194264
Actual b0: -1
Actual b1: 1/2
```

```
pred_y = results.predict(x)
plt.figure(figsize=(10, 10))
plt.scatter(x[:,1] ,y )
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatterplot of y as function of x')

x_pop = np.linspace(x[:, 1].min(), x[:, 1].max(), 100)
y_pop = -1 + 0.5 * x_pop


plt.plot(x_pop, y_pop, color='blue', label='Population Regression Line')
plt.plot(x[:,1], pred_y, color='red', label='Least Squares Line')
plt.legend()
plt.show()
```



The reduced noise did make the ols line get closer to the true population line but that was due to lesser noise and data that strayed away from the true population line. This is the closest the OLS line has been to replacing the population line.

13i - Repeat (a)–(f) after modifying the data generation process in such a way that there is more noise in the data. The model (3.39) should remain the same. You can do this by increasing the variance of the normal distribution used to generate the error term ϵ in (b). Describe your results.

```
eps_increased = np.random.normal(0,0.75,100)
y = -1 + (x[:, 1]/2) + eps_increased
model = sm.OLS(y,x)
results = model.fit()
print(results.summary())

x = sm.add_constant(x)
model = sm.OLS(y,x)
results = model.fit()
print(results.summary())
```

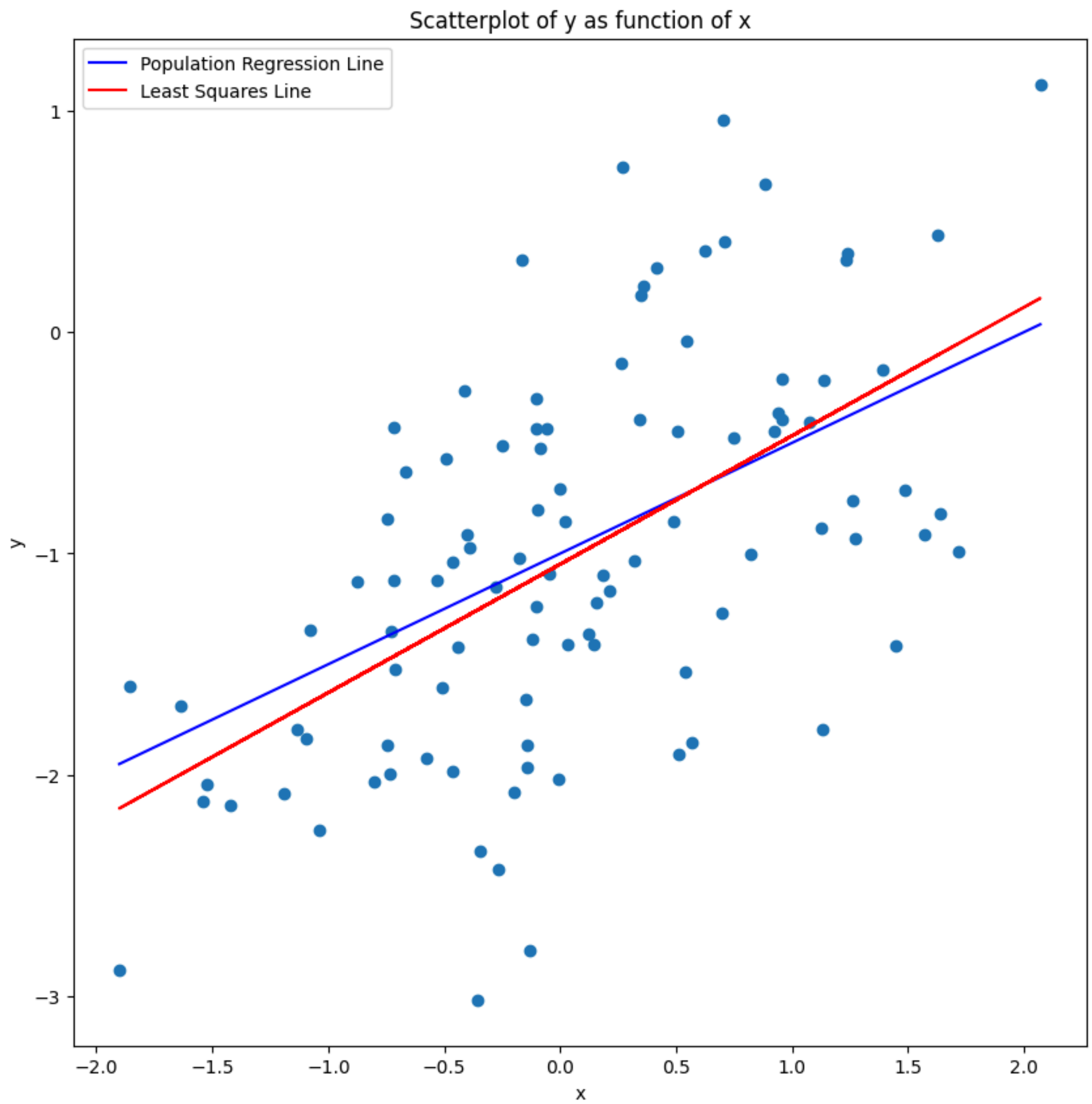
```
b0_pred = results.params[0]
b1_pred = results.params[1]
print(f"Predicted b0: {b0_pred}")
print(f"Predicted b1: {b1_pred}")

print(f"Actual b0: -1")
print(f"Actual b1: 1/2")
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.325
Model:                            OLS   Adj. R-squared:                  0.318
Method:                 Least Squares   F-statistic:                     47.13
Date:                Thu, 06 Feb 2025   Prob (F-statistic):           6.09e-10
Time:                        04:14:40   Log-Likelihood:                -108.43
No. Observations:                 100   AIC:                             220.9
Df Residuals:                      98   BIC:                             226.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.0481      0.073    -14.453      0.000      -1.192      -0.904
x1             0.5802      0.085      6.865      0.000       0.412       0.748
==============================================================================
Omnibus:                        0.363   Durbin-Watson:                   2.151
Prob(Omnibus):                  0.834   Jarque-Bera (JB):                0.528
Skew:                          -0.030   Prob(JB):                        0.768
Kurtosis:                       2.649   Cond. No.                         1.19
==============================================================================

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


```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.325
Model:                            OLS   Adj. R-squared:                  0.318
Method:                 Least Squares   F-statistic:                     47.13
Date:                Thu, 06 Feb 2025   Prob (F-statistic):           6.09e-10
Time:                        04:14:40   Log-Likelihood:                -108.43
No. Observations:                 100   AIC:                             220.9
Df Residuals:                      98   BIC:                             226.1
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.0481      0.073    -14.453      0.000      -1.192      -0.904
x1             0.5802      0.085      6.865      0.000       0.412       0.748
==============================================================================
Omnibus:                        0.363   Durbin-Watson:                   2.151
Prob(Omnibus):                  0.834   Jarque-Bera (JB):                0.528
Skew:                          -0.030   Prob(JB):                        0.768
Kurtosis:                       2.649   Cond. No.                         1.19
==============================================================================

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


```
Predicted b0: -1.0481144669873066
Predicted b1: 0.5801988470511434
Actual b0: -1
Actual b1: 1/2
```

```
pred_y = results.predict(x)
plt.figure(figsize=(10, 10))
plt.scatter(x[:,1] ,y )
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatterplot of y as function of x')

x_pop = np.linspace(x[:, 1].min(), x[:, 1].max(), 100)
y_pop = -1 + 0.5 * x_pop


plt.plot(x_pop, y_pop, color='blue', label='Population Regression Line')
plt.plot(x[:,1], pred_y, color='red', label='Least Squares Line')
plt.legend()
plt.show()
```



The increased noise made the ols line less accurate to use to predict the population line. while the slopes were still close the intercept ended up moving lower. the increased noise has definetly made the ols underfit the model which will make residual errors between predictions and trye values larger.

13j - What are the confidence intervals for β0 and β1 based on the original data set, the noisier data set, and the less noisy data set? Comment on your result?

```
x = sm.add_constant(x)
model_original = sm.OLS(y, x)
results_original = model_original.fit()

conf_int_original = results_original.conf_int(alpha=0.05)


print(conf_int_original)
```

```
[[-1.19202563 -0.90420331]
 [ 0.41248953  0.74790816]]
```

```
y_more_noise = -1 + (x[:, 1] / 2) + eps_increased
model_more_noise = sm.OLS(y_more_noise, x)
results_more_noise = model_more_noise.fit()

conf_int_more_noise = results_more_noise.conf_int(alpha=0.05)


print(conf_int_more_noise)
```

```
[[-1.19202563 -0.90420331]
 [ 0.41248953  0.74790816]]
```

```
y_less_noise = -1 + (x[:, 1] / 2) + eps_reduced
model_less_noise = sm.OLS(y_less_noise, x)
results_less_noise = model_less_noise.fit()

conf_int_less_noise = results_less_noise.conf_int(alpha=0.05)

print(conf_int_less_noise)
```

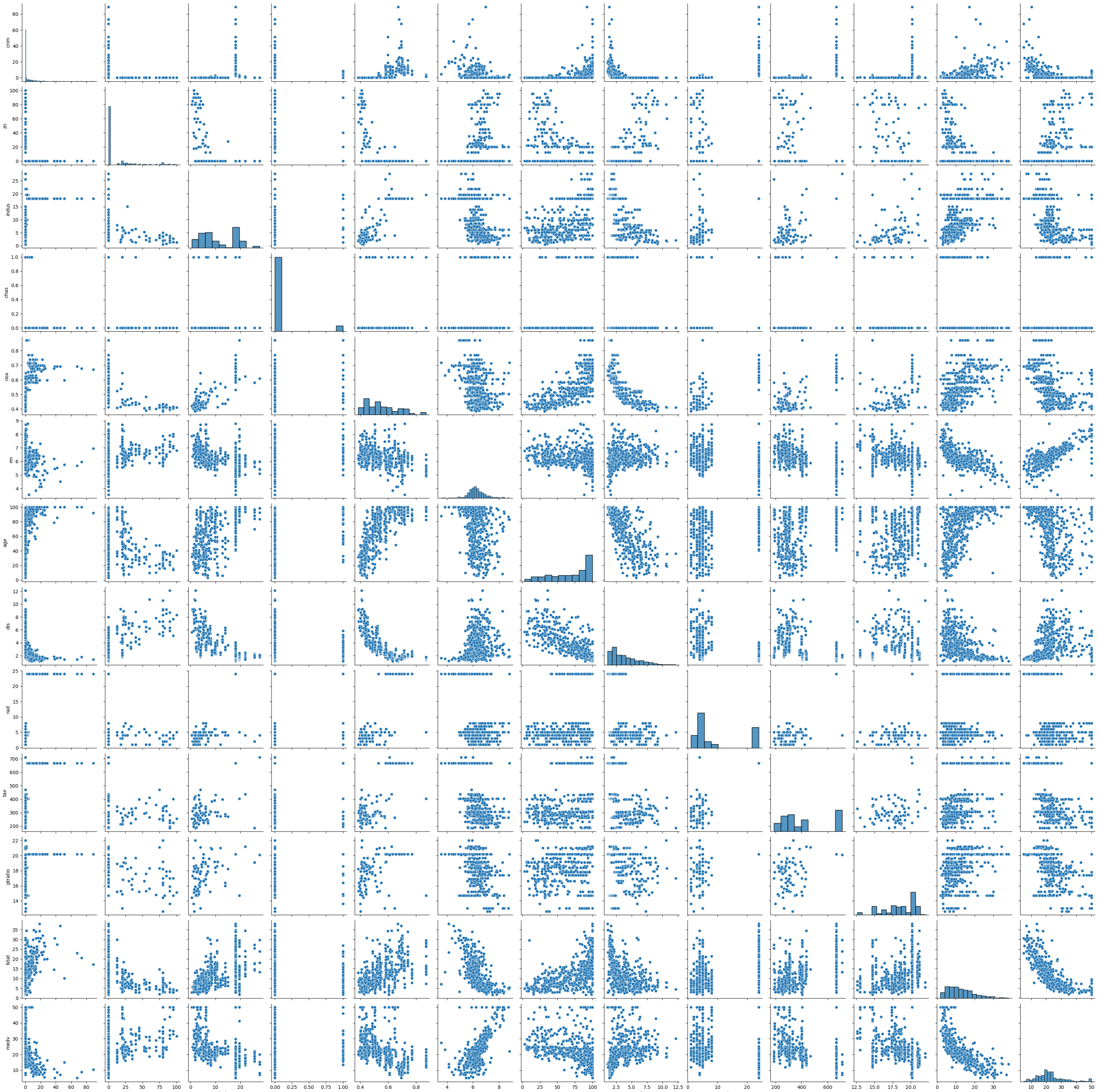
```
[[-1.00462128 -0.9786072 ]
 [ 0.49108169  0.52139764]]
```

From the confidence intervals we can see that the least varied interval was the least noisy while the one with the most variance was the data with more noise. both coeficients of b0 and b1 were closer to the true population values in the less noisy data and the least accurate or larger confidence intervals were from the noisy data.

15a - using the boston data set, For each predictor, fit a simple linear regression model to predict the response. Describe your results. In which of the models is there a statistically significant association between the predictor and the response? Create some plots to back up your assertions

```
from ISLP import load_data
boston_data = load_data('Boston')
boston = pd.DataFrame(boston_data)
sns.pairplot(boston)
```

```
<seaborn.axisgrid.PairGrid at 0x79b17043af90>
```



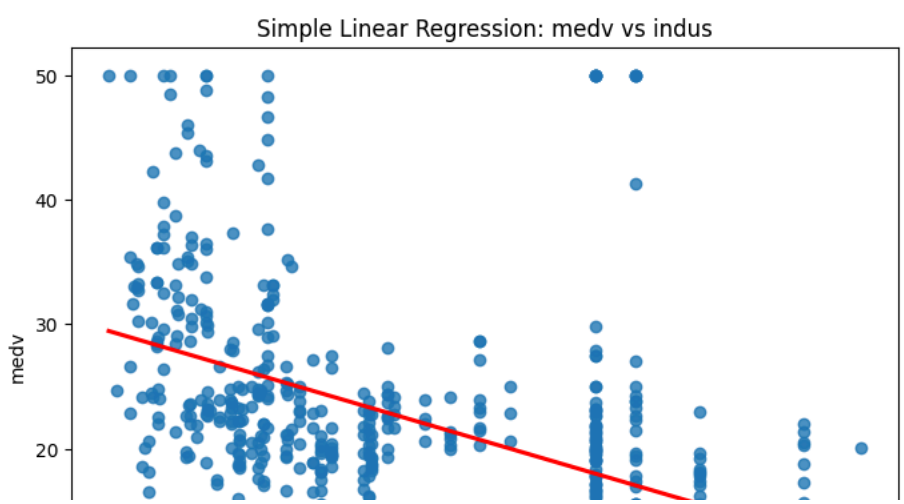
```
predictors = boston.columns.drop('medv')
results = {}

for predictor in predictors:
    formula = f'medv ~ {predictor}'
    model = smf.ols(formula, data=boston).fit()
    results[predictor] = model

for predictor, model in results.items():
  print(f"Predictor: {predictor}")
  print(model.summary())
  print("-" * 50)
```

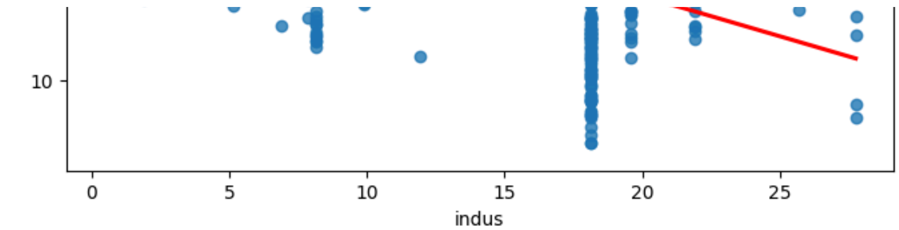
```
Dep. Variable:                   medv   R-squared:                       0.544
Model:                            OLS   Adj. R-squared:                  0.543
Method:                 Least Squares   F-statistic:                     601.6
Date:                Thu, 06 Feb 2025   Prob (F-statistic):           5.08e-88
Time:                        04:15:08   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]
------------------------------------------------------------------------------
Intercept     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.
--------------------------------------------------
```

```python
for predictor in predictors:
    plt.figure(figsize=(8, 6))
    sns.regplot(x=predictor, y='medv', data=boston, ci=None, line_kws={'color': 'red'})
    plt.title(f'Simple Linear Regression: medv vs {predictor}')
    plt.xlabel(predictor)
    plt.ylabel('medv')
    plt.show()
```

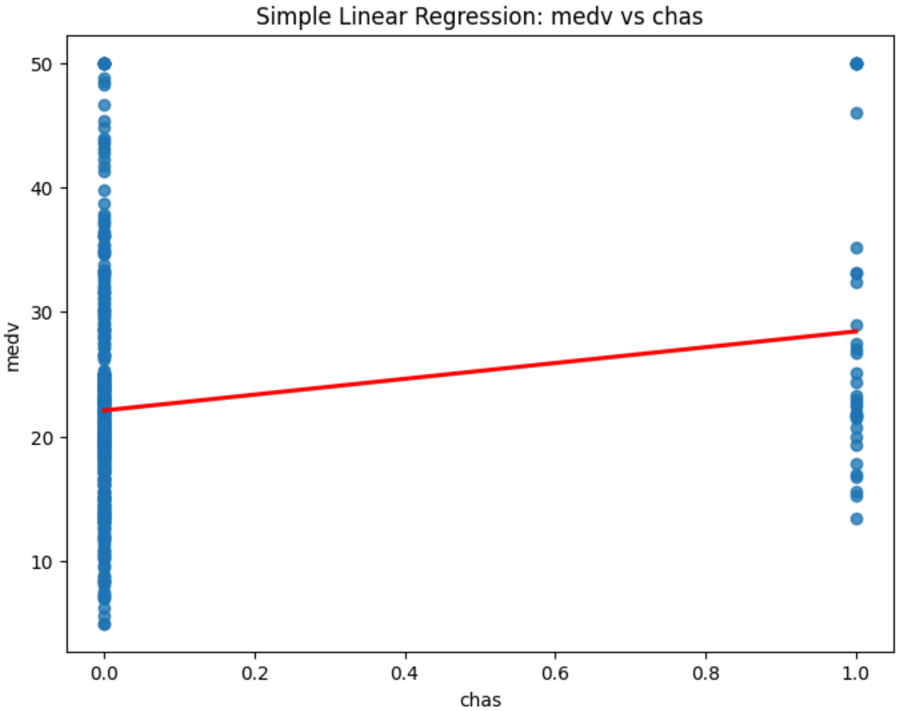

Simple Linear Regression: medv vs crim
medv
crim
50
40
30
20
10
0
−10
0
20
40
60
80


Simple Linear Regression: medv vs zn
medv
zn
50
40
30
20
10
0
20
40
60
80
100

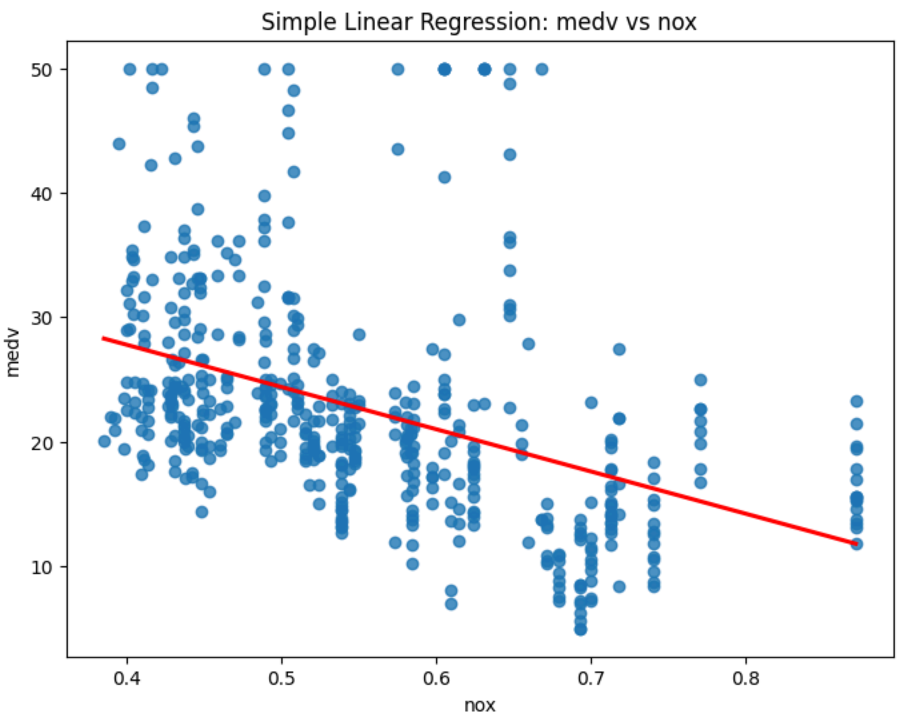

Simple Linear Regression: medv vs indus
medv
50
40
30
20



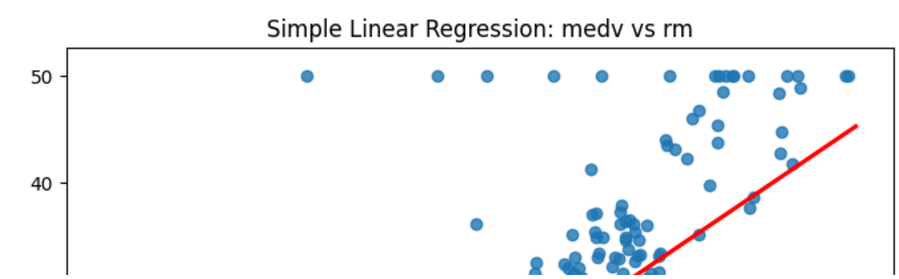
10
0
5
10
15
20
25
indus



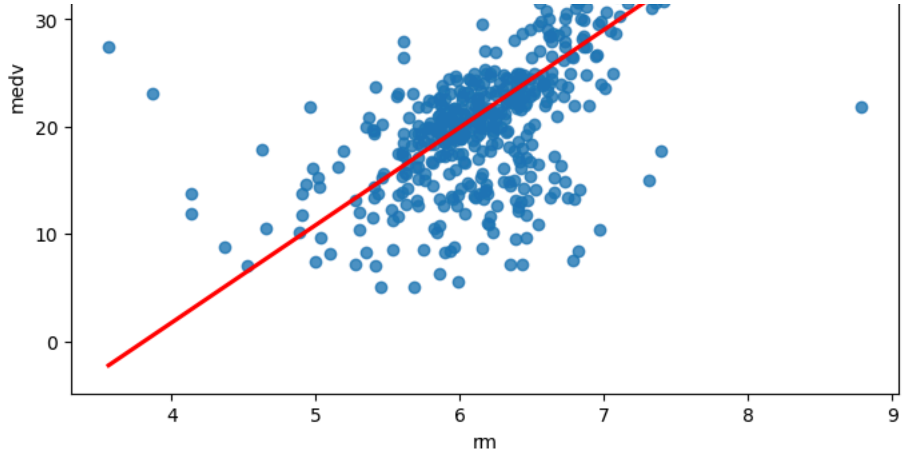
Simple Linear Regression: medv vs chas
medv
50
40
30
20
10
0.0
0.2
0.4
0.6
0.8
1.0
chas



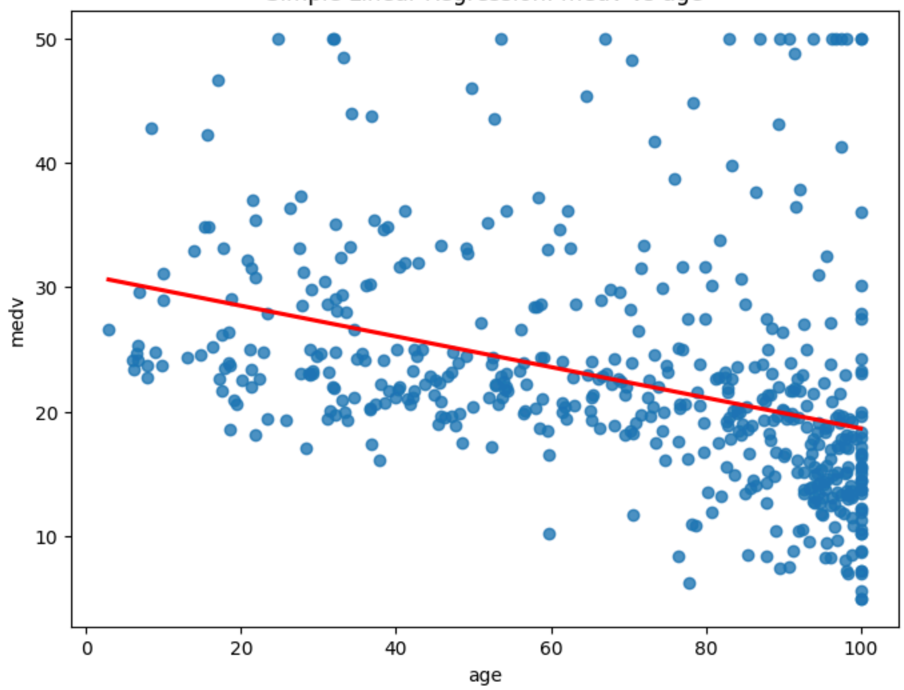
Simple Linear Regression: medv vs nox
medv
50
40
30
20
10
0.4
0.5
0.6
0.7
0.8
nox



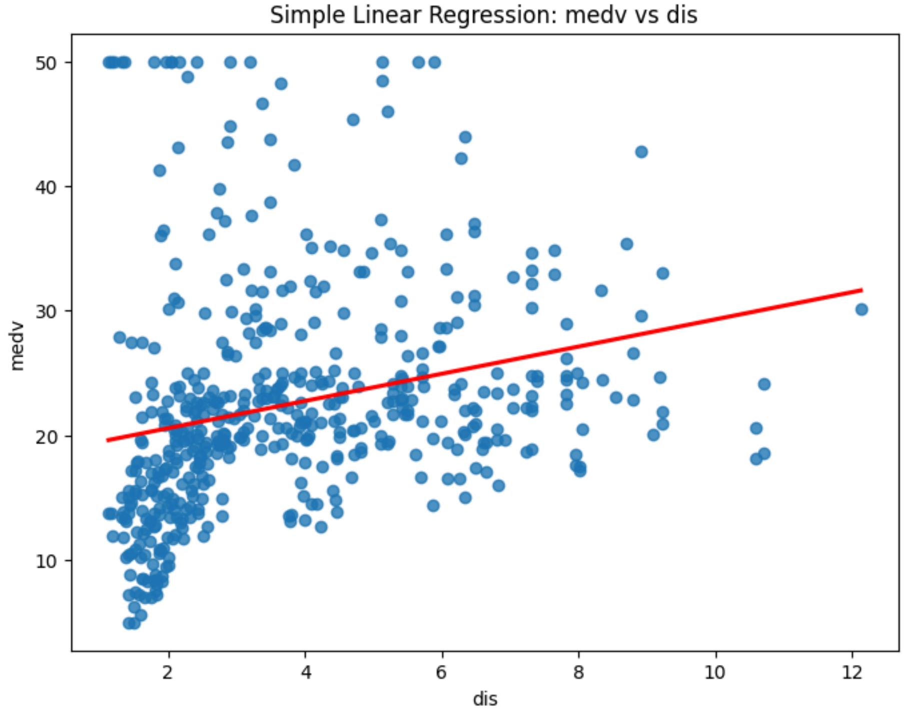
Simple Linear Regression: medv vs rm
50
40



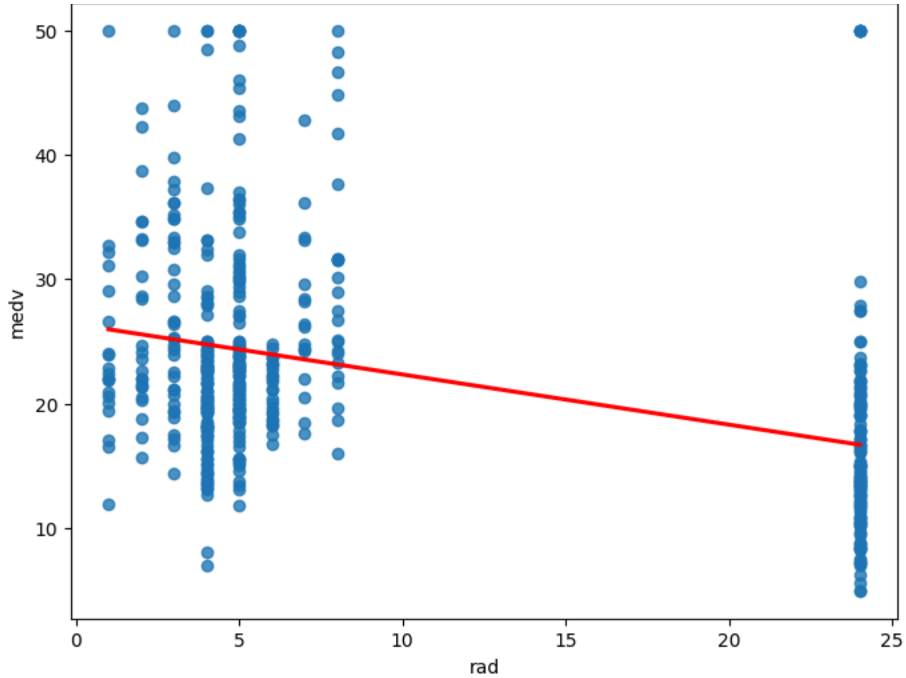
Simple Linear Regression: medv vs age



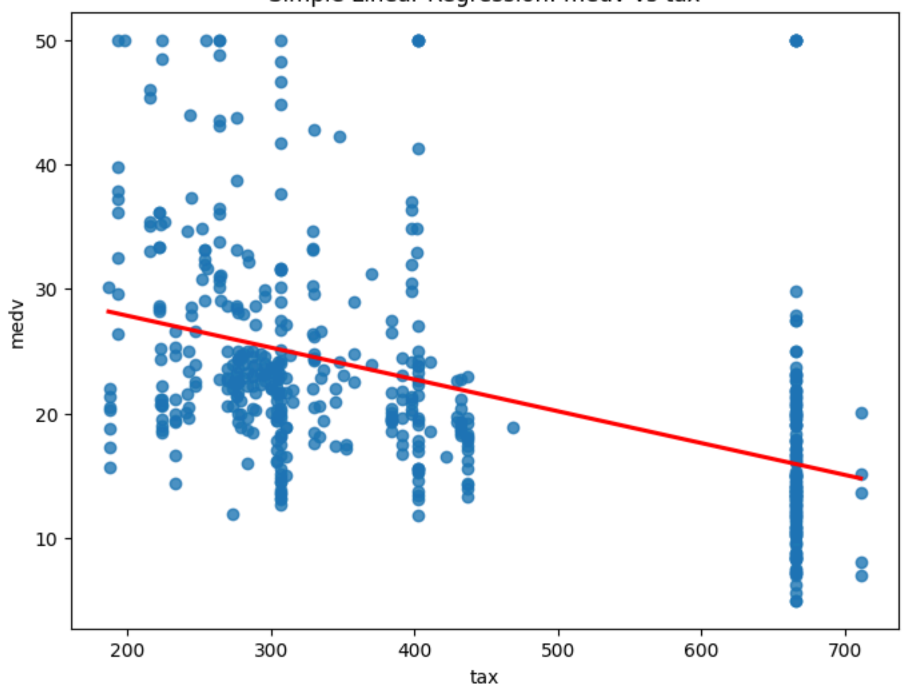
Simple Linear Regression: medv vs dis



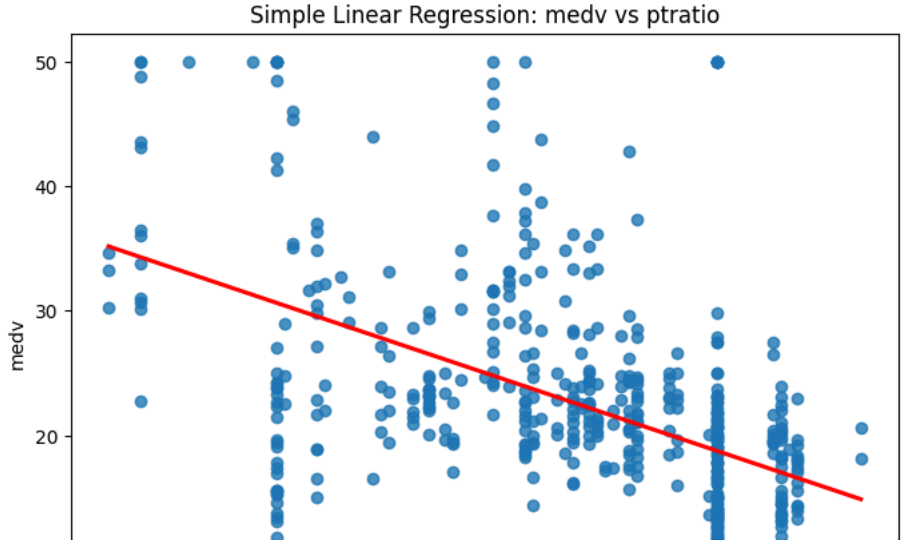
Simple Linear Regression: medv vs rad

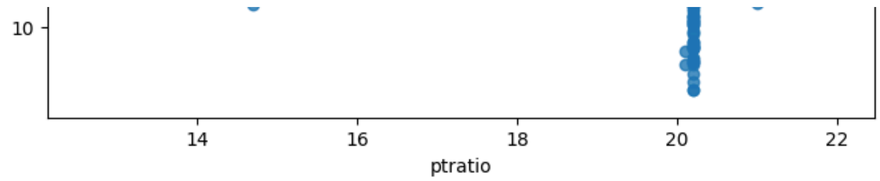


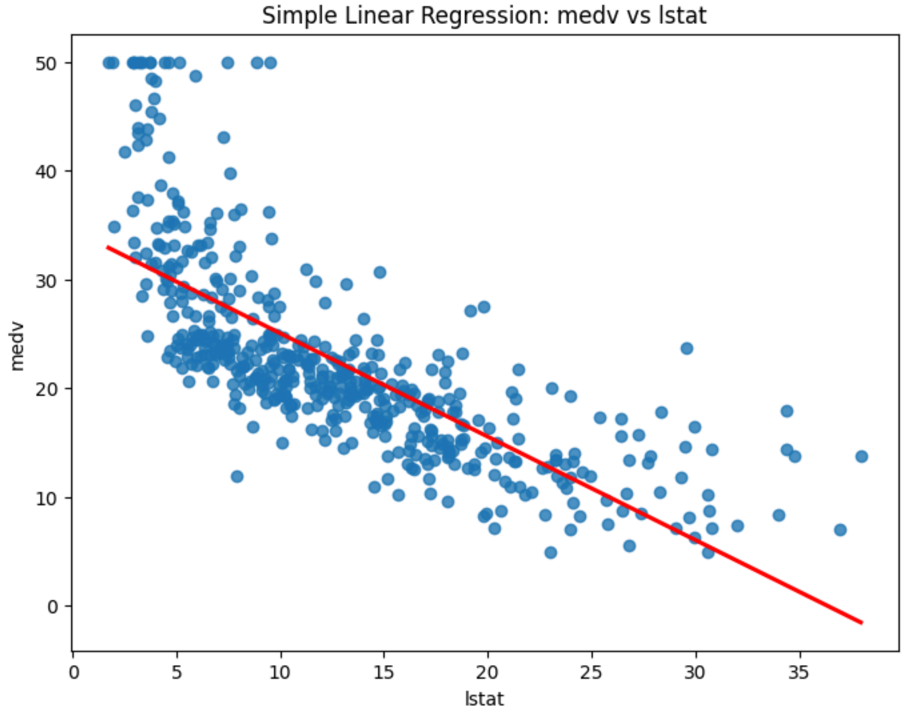
Simple Linear Regression: medv vs tax



Simple Linear Regression: medv vs ptratio




10
14
16
18
20
22
ptratio


Simple Linear Regression: medv vs lstat
medv
lstat
0
10
20
30
40
50
0
5
10
15
20
25
30
35

Some realtionships that surfaced from the regression tables and the pairplots above came from predictors like Crm, Lstat, Age, Rm. These variables related to the crime rate within the suburb (crm), The percentage of people who are considered to be of lower socioeconomic status in the neighborhood (lstat), Age of the owner of the home (age), Avearge number of rooms in a dwelling. all these varaible had some predictive relatiosbho whe setting median owner occupied house value as a dependent variable (medv).

15b - Fit a multiple regression model to predict the response using all of the predictors. Describe your results. For which predictors can we reject the null hypothesis H0 : βj = 0?

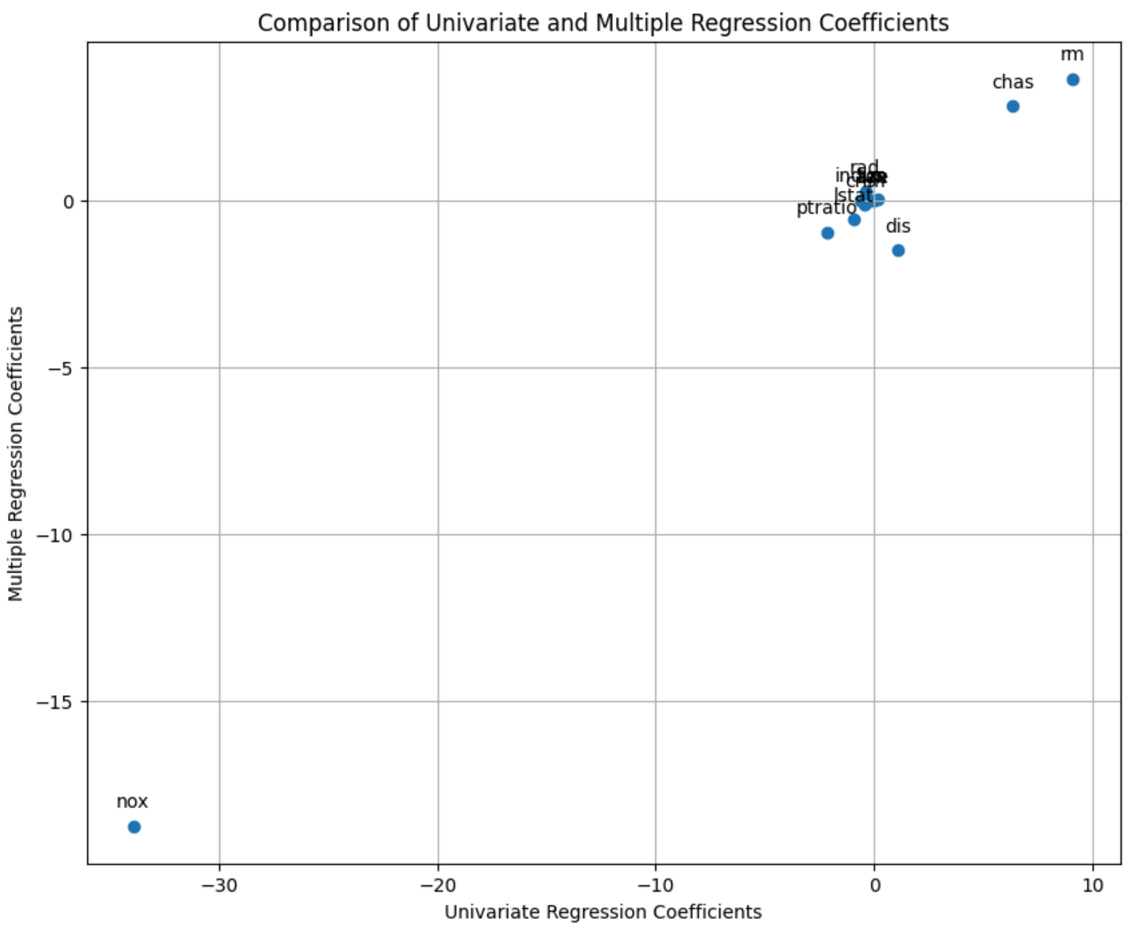
```
formula = 'medv ~ ' + ' + '.join(boston.columns.drop('medv'))
model = smf.ols(formula, data=boston).fit()
print(model.summary())
significant_predictors = [
    predictor for predictor in model.pvalues.index[1:]
    if model.pvalues[predictor] < 0.05 ]
print("\nSignificant Predictors (p-value < 0.05):")
print(significant_predictors)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.734
Model:                            OLS   Adj. R-squared:                  0.728
Method:                 Least Squares   F-statistic:                     113.5
Date:                Thu, 06 Feb 2025   Prob (F-statistic):          2.23e-133
Time:                        04:15:11   Log-Likelihood:                -1504.9
No. Observations:                 506   AIC:                             3036.
Df Residuals:                     493   BIC:                             3091.
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     41.6173      4.936      8.431      0.000      31.919      51.316
crim          -0.1214      0.033     -3.678      0.000      -0.186      -0.057
zn             0.0470      0.014      3.384      0.001       0.020       0.074
indus          0.0135      0.062      0.217      0.829      -0.109       0.136
chas           2.8400      0.870      3.264      0.001       1.131       4.549
nox          -18.7580      3.851     -4.870      0.000     -26.325     -11.191
rm             3.6581      0.420      8.705      0.000       2.832       4.484
age            0.0036      0.013      0.271      0.787      -0.023       0.030
dis           -1.4908      0.202     -7.394      0.000      -1.887      -1.095
rad            0.2894      0.067      4.325      0.000       0.158       0.421
tax           -0.0127      0.004     -3.337      0.001      -0.020      -0.005
ptratio       -0.9375      0.132     -7.091      0.000      -1.197      -0.678
lstat         -0.5520      0.051    -10.897      0.000      -0.652      -0.452
==============================================================================
Omnibus:                      171.096   Durbin-Watson:                   1.077
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              709.937
Skew:                           1.477   Prob(JB):                    6.90e-155
Kurtosis:                       7.995   Cond. No.                     1.17e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.17e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Significant Predictors (p-value < 0.05):
['crim', 'zn', 'chas', 'nox', 'rm', 'dis', 'rad', 'tax', 'ptratio', 'lstat']
```

How do your results from (a) compare to your results from (b)? Create a plot displaying the univariate regression coefficients from (a) on the x-axis, and the multiple regression coefficients from (b) on the y-axis. That is, each predictor is displayed as a single point in the plot. Its coefficient in a simple linear regression model is shown on the x-axis, and its coefficient estimate in the multiple linear regression model is shown on the y-axis

```
univariate_coeffs = {predictor: model.params[1] for predictor, model in results.items()}
multivariate_coeffs = {predictor: model.params[predictor] for predictor in predictors}

x_coords = list(univariate_coeffs.values())
y_coords = list(multivariate_coeffs.values())
labels = list(univariate_coeffs.keys())

plt.figure(figsize=(10, 8))
plt.scatter(x_coords, y_coords)

for i, label in enumerate(labels):
    plt.annotate(label, (x_coords[i], y_coords[i]), textcoords="offset points", xytext=(0, 10), ha='center')

plt.xlabel("Univariate Regression Coefficients")
plt.ylabel("Multiple Regression Coefficients")
plt.title("Comparison of Univariate and Multiple Regression Coefficients")
plt.grid(True)
plt.show()
```

```
<ipython-input-526-0ef02e6e4b17>:1: FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a future
  univariate_coeffs = {predictor: model.params[1] for predictor, model in results.items()}
```



In part A i disregarded some valid coefficients due to my interpretation from the graphs and tables as single predictors looked increadibly inconsistent. but in the multi reg those variable did prove to be useful in addition to the other predictors.

15d- Is there evidence of non-linear association between any of the predictors and the response? To answer this question, for each predictor X, fit a model of the form Y= β0 + β1X + β2X2 + β3X3 + ϵ.

```
for predictor in predictors:
    formula = f'medv ~ {predictor} + I({predictor}**2) + I({predictor}**3)'
    model = smf.ols(formula, data=boston).fit()
    print(f"Predictor: {predictor}")
    print(model.summary())
    print("-" * 50)
```

```
Model:                            OLS   Adj. R-squared:                  0.262
Method:                 Least Squares   F-statistic:                     60.91
Date:                Thu, 06 Feb 2025   Prob (F-statistic):           1.35e-33
Time:                        04:15:11   Log-Likelihood:                -1761.7
No. Observations:                 506   AIC:                             3531.
Df Residuals:                     502   BIC:                             3548.
Df Model:                           3
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept        312.2864    152.487      2.048      0.041      12.695     611.878
ptratio          -48.6911     26.884     -1.811      0.071    -101.511       4.129
I(ptratio ** 2)    2.8400      1.564      1.816      0.070      -0.233       5.913
I(ptratio ** 3)   -0.0569      0.030     -1.892      0.059      -0.116       0.002
==============================================================================
Omnibus:                       98.389   Durbin-Watson:                   0.735
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              200.314
Skew:                           1.062   Prob(JB):                     3.18e-44
Kurtosis:                       5.235   Cond. No.                     3.02e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.02e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
--------------------------------------------------
Predictor: lstat
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.658
Model:                            OLS   Adj. R-squared:                  0.656
Method:                 Least Squares   F-statistic:                     321.7
Date:                Thu, 06 Feb 2025   Prob (F-statistic):          1.78e-116
Time:                        04:15:11   Log-Likelihood:                -1568.9
No. Observations:                 506   AIC:                             3146.
Df Residuals:                     502   BIC:                             3163.
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       48.6496      1.435     33.909      0.000      45.831      51.468
lstat           -3.8656      0.329    -11.757      0.000      -4.512      -3.220
I(lstat ** 2)    0.1487      0.021      6.983      0.000       0.107       0.191
I(lstat ** 3)   -0.0020      0.000     -5.013      0.000      -0.003      -0.001
==============================================================================
Omnibus:                      107.925   Durbin-Watson:                   0.906
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              258.171
Skew:                           1.088   Prob(JB):                     8.69e-57
Kurtosis:                       5.741   Cond. No.                     5.20e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.2e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
--------------------------------------------------
```

Crim, zn, indus, chas, rm, dis, rad, tax, lstat were the only predictors that had consitent low pvalues on all terms which allowed for them to fit into non linear regression. This intuitively makes sense for a lot of the variables where crime rates or even the number of rooms can have an accelerating of even diminishing returns for every increase which lowers the magnitude of the effect they have on medv. There is strong eveidence both intutive and in the models to prove the variables above have some non linear relationship to the explanatory variable.

Part 3

1. Plot some relationships from the Homes data set and tell a story

```
homes_data = load_data('/homes2004')
homes = pd.DataFrame(homes_data)
sns.pairplot(homes)
```

```
<seaborn.axisgrid.PairGrid at 0x79b169afc150>
```