In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.linear_model import LinearRegression
sns.set()
```

In [ ]:
```
df = pd.read_csv("Auto.csv")

#remove any non-numerical data (? values)
df[df.columns[:-2]] = df[df.columns[:-2]].apply(pd.to_numeric, errors='coerce')
df = df.dropna()
df = df.reset_index(drop=True)
```

QUESTION 1:

a)

In [ ]:
```
x = df['horsepower'].values.reshape(-1, 1)
y = df['mpg'].values.reshape(-1, 1)
reg = LinearRegression().fit(x, y)
# print("Coeff of determination (R^2): {}".format(reg.score(x, y)))
X = sm.add_constant(x)
model = sm.OLS(y,X).fit()
print(model.summary())


print("Predicted mpg with a horsepower of 95: {}".format(reg.predict([[95]])[0][0]))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.606
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     599.7
Date:                Tue, 15 Nov 2022   Prob (F-statistic):           7.03e-81
Time:                        18:40:06   Log-Likelihood:                -1178.7
No. Observations:                 392   AIC:                             2361.
Df Residuals:                     390   BIC:                             2369.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         39.9359      0.717     55.660      0.000      38.525      41.347
x1            -0.1578      0.006    -24.489      0.000      -0.171      -0.145
==============================================================================
Omnibus:                       16.432   Durbin-Watson:                   0.920
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               17.305
Skew:                           0.492   Prob(JB):                     0.000175
Kurtosis:                       3.299   Cond. No.                         322.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
Predicted mpg with a horsepower of 95: 24.94061135257337
```

i) As the coeff of determination (R^2) is relatively close to 1, there is a relationship between the predictor and the response. The p-value is also very small for the linear regression with mpg and horsepower

ii) The relationship between the predictor and response is strong. The p value is small when performing a t-test, and the coefficient of determination is also .606, which indicates that almost 60.6% of the variability in mpg can be explained by horsepower.

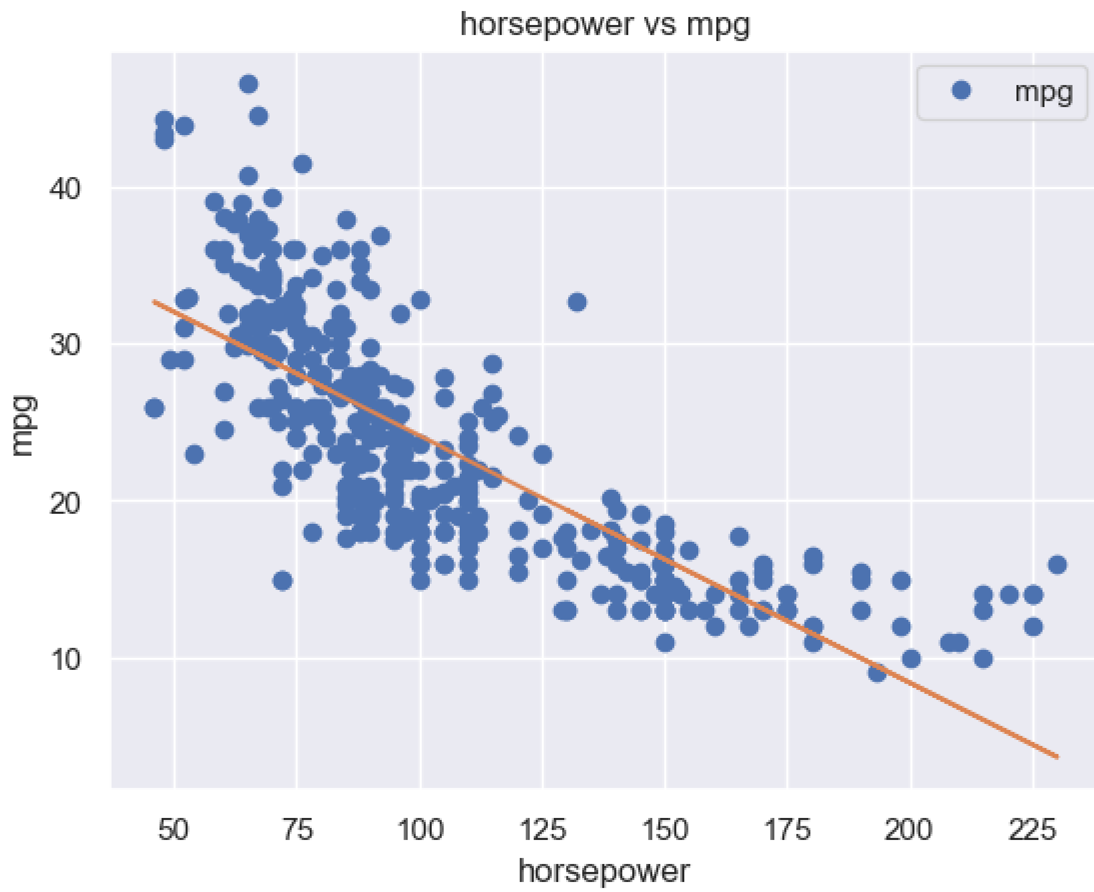
iii) The relationship between the predictor and response is negative. We can see this, as the coefficient of x1 (our predictor) is negative.

iv) The predicted mpg with a horsepower of 95 is 24.94061135257337

b)

In [ ]:

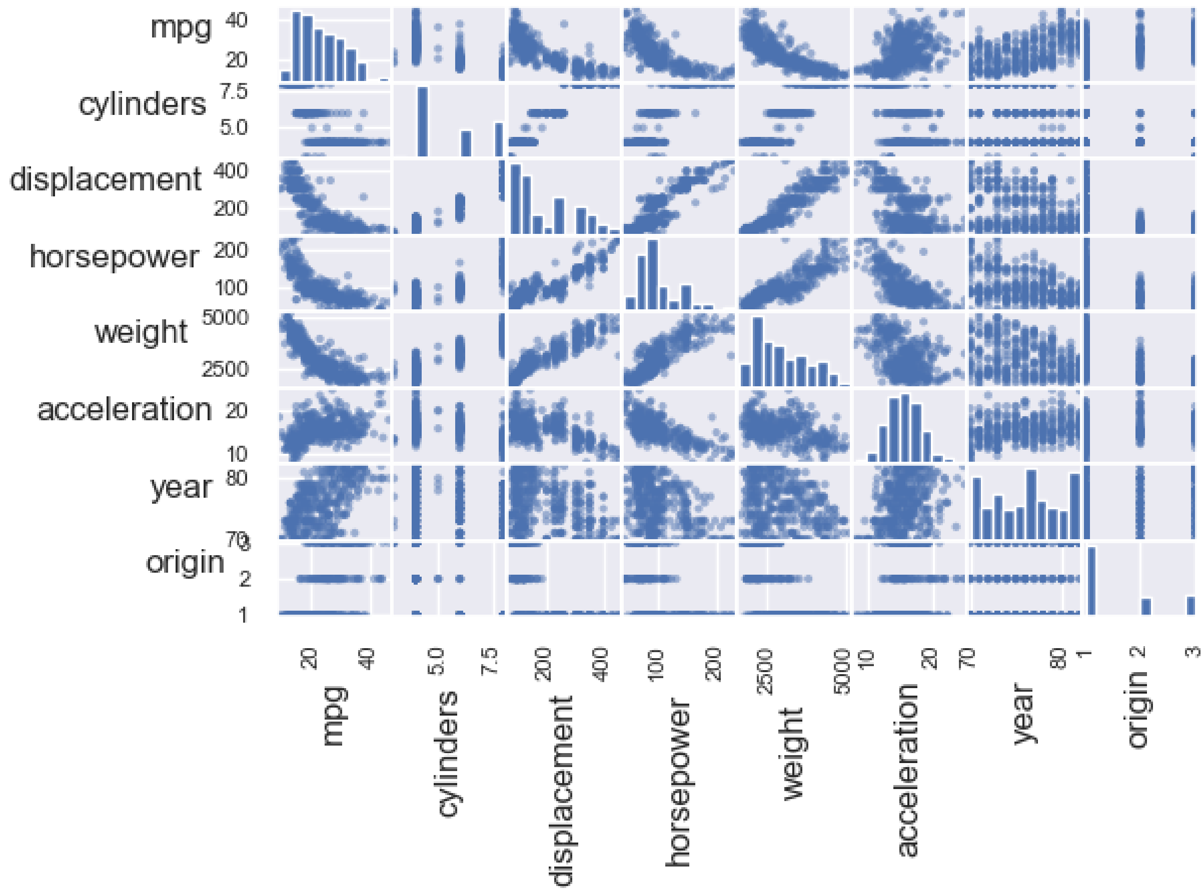
```
ypred = reg.predict(x)
df.plot(x='horsepower', y='mpg', style='o')
plt.title("horsepower vs mpg")
plt.ylabel('mpg')
plt.plot(x, ypred)
plt.show()
```



2.

a)

In [ ]:
```
axes = pd.plotting.scatter_matrix(df)
for ax in axes.flatten():
    ax.xaxis.label.set_rotation(90)
    ax.yaxis.label.set_rotation(0)
    ax.yaxis.label.set_ha('right')
plt.tight_layout()
plt.gcf().subplots_adjust(wspace=0, hspace=0)
plt.show()
```



b)

```
In [ ]: corrM = df.corr() #no names
        print(corrM)
```

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

              acceleration      year    origin
mpg               0.423329  0.580541  0.565209
cylinders        -0.504683 -0.345647 -0.568932
displacement     -0.543800 -0.369855 -0.614535
horsepower       -0.689196 -0.416361 -0.455171
weight           -0.416839 -0.309120 -0.585005
acceleration      1.000000  0.290316  0.212746
year              0.290316  1.000000  0.181528
origin            0.212746  0.181528  1.000000
C:\Users\Bernhard\AppData\Local\Temp\ipykernel_41592\2646264494.py:1: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future versio
n, it will default to False. Select only valid columns or specify the value of numeri
c_only to silence this warning.
  corrM = df.corr() #no names
```

c)

In [ ]:
```
dfplot = df.drop(["mpg","name"], axis = 1)

x = dfplot
y = df['mpg'].values.reshape(-1, 1)
reg = LinearRegression().fit(x, y)
print("Coeff of determination (R^2): {}".format(reg.score(x, y)))
X = sm.add_constant(x)
model = sm.OLS(y,X).fit()
print(model.summary())
```

```
Coeff of determination (R^2): 0.8214780764810599
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.821
Model:                            OLS   Adj. R-squared:                  0.818
Method:                 Least Squares   F-statistic:                     252.4
Date:                Tue, 15 Nov 2022   Prob (F-statistic):          2.04e-139
Time:                        18:40:08   Log-Likelihood:                -1023.5
No. Observations:                 392   AIC:                             2063.
Df Residuals:                     384   BIC:                             2095.
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const          -17.2184      4.644     -3.707      0.000     -26.350      -8.087
cylinders       -0.4934      0.323     -1.526      0.128      -1.129       0.142
displacement     0.0199      0.008      2.647      0.008       0.005       0.035
horsepower      -0.0170      0.014     -1.230      0.220      -0.044       0.010
weight          -0.0065      0.001     -9.929      0.000      -0.008      -0.005
acceleration     0.0806      0.099      0.815      0.415      -0.114       0.275
year             0.7508      0.051     14.729      0.000       0.651       0.851
origin           1.4261      0.278      5.127      0.000       0.879       1.973
==============================================================================
Omnibus:                       31.906   Durbin-Watson:                   1.309
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               53.100
Skew:                           0.529   Prob(JB):                     2.95e-12
Kurtosis:                       4.460   Cond. No.                     8.59e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 8.59e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

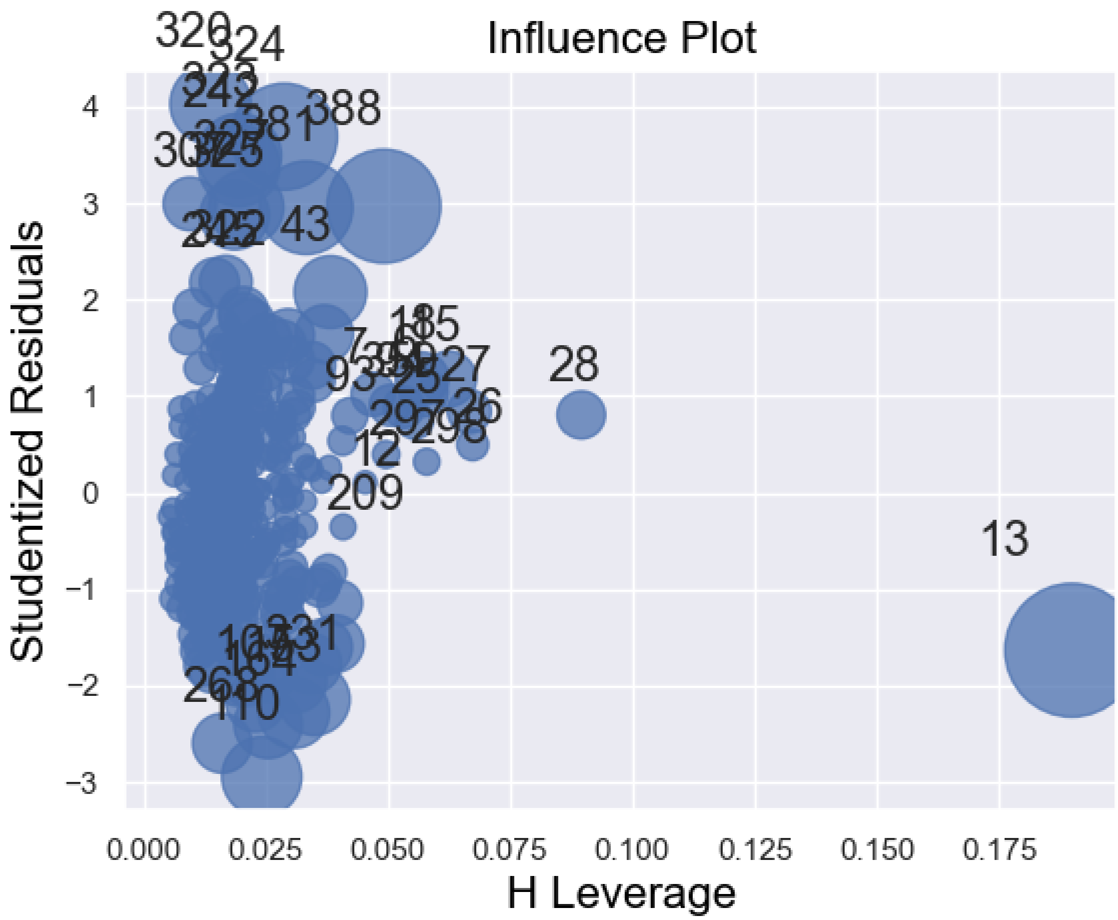
i) there is a strong relationship between the predictors and the response

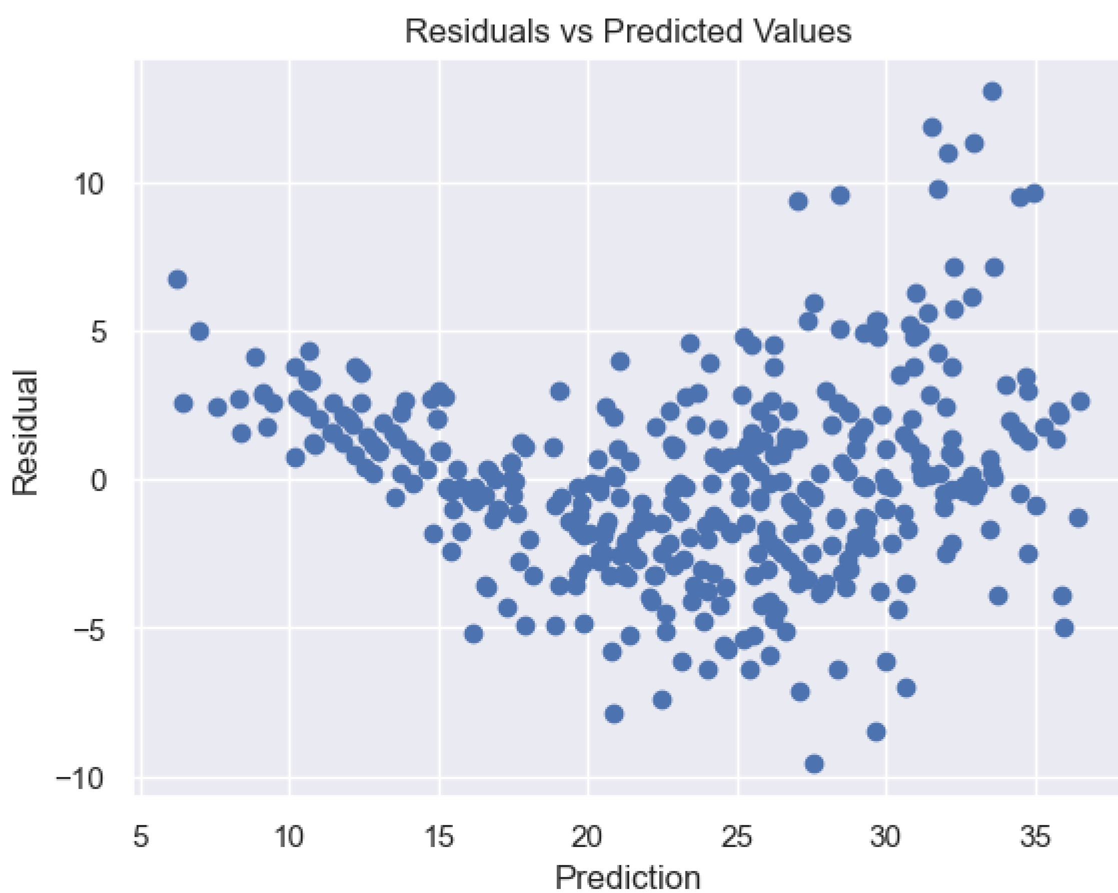
ii) weight, year, origin, and displacement seem to have statistically significant relationships to the response.

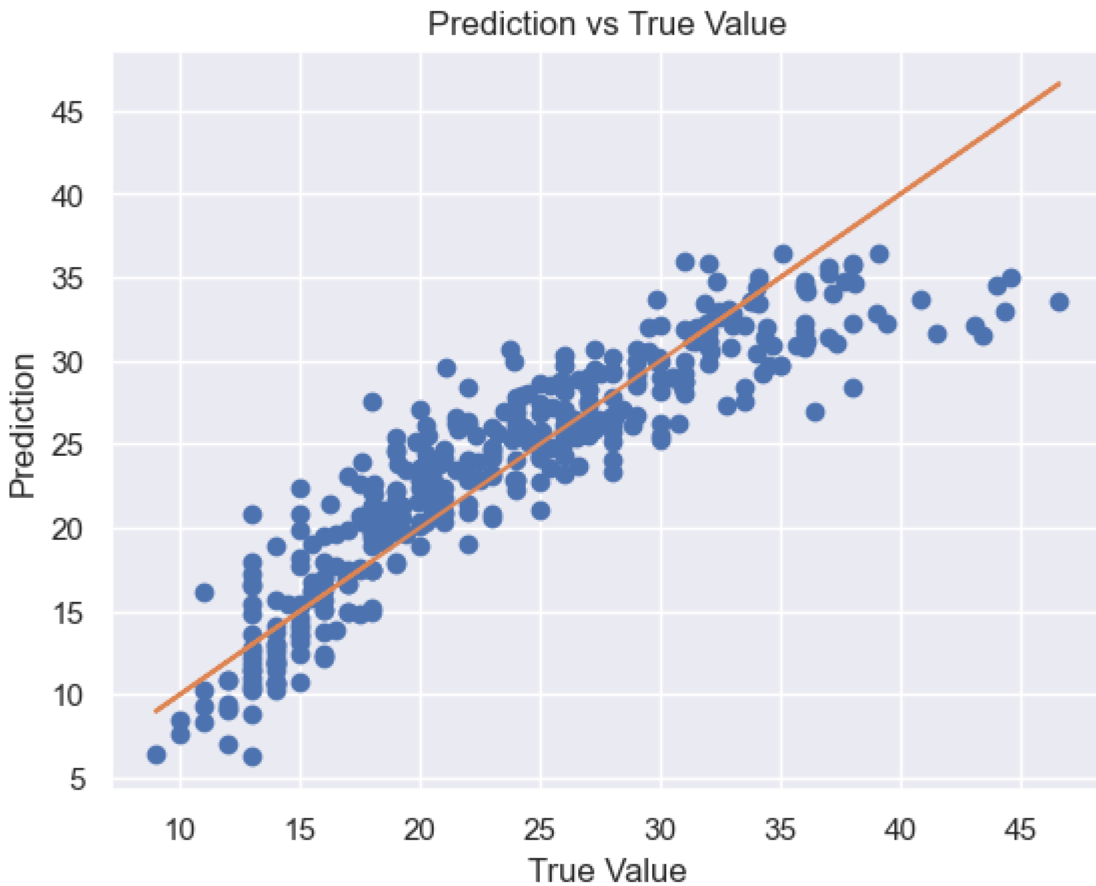
iii) the coefficient for 'year' suggests that as the year increases, the mpg of the vehicle increases as well by a factor of .75 assuming that all other predictors are constant.

d)

In [ ]:
```
# ypred = reg.predict(x)
# df.plot(x='horsepower', y='mpg', style='o')
# plt.title("horsepower vs mpg")
# plt.ylabel('mpg')
# plt.plot(x, ypred)
# plt.show()

y_pred=reg.predict(x)
resid = y-y_pred


sm.graphics.influence_plot(model)
plt.show()

plt.plot(y_pred,resid,'o')
plt.title('Residuals vs Predicted Values')
plt.ylabel('Residual')
plt.xlabel('Prediction')
plt.show()

# Input vs Error
plt.plot(y,y_pred,'o')
plt.plot(y,y)
plt.title('Prediction vs True Value')
plt.xlabel('True Value')
plt.ylabel('Prediction')
plt.show()
```


Influence Plot
Studentized Residuals
H Leverage
0.000 0.025 0.050 0.075 0.100 0.125 0.150 0.175
−3 −2 −1 0 1 2 3 4
320
324
388
43
28
13
209
110


Residuals vs Predicted Values
Residual
Prediction
5 10 15 20 25 30 35
−10 −5 0 5 10



The error vs predicted values chart seems to have a slight curve, which means that the relationship of the data is non-linarly associated. We might use non-linear transformations of the predictors such as log(x). The error terms also show a non-constant variance, which could be solved by using a non-linear transformation. The Influence plot reveals no real outliers, but reveals a very high leverage data point, point 13 (starting count from 0). Observation 28 also has a relatively high leverage. We can assess that observations 320, 324, 28 and 13 are likely outliers in our data along with other observations with a studentized residual of 3 or more. Most of our observations have a leverage lesser than 0.075 with the exception of two observations: 28, and 13.

e1)

In [ ]:
```
model_interaction = smf.ols(formula='mpg ~ weight + cylinders + weight:cylinders', dat
summary = model_interaction.summary()
print(summary.tables[1])

model_interaction = smf.ols(formula='mpg ~ horsepower + acceleration + horsepower:acce
summary = model_interaction.summary()
print(summary.tables[1])

model_interaction = smf.ols(formula='mpg ~ displacement + year + displacement:year', d
summary = model_interaction.summary()
print(summary.tables[1])

model_interaction = smf.ols(formula='mpg ~ horsepower + displacement + horsepower:disp
```

```python
summary = model_interaction.summary()
print(summary.tables[1])
```

```
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept            65.3865      3.733     17.514      0.000      58.046      72.727
weight               -0.0128      0.001     -9.418      0.000      -0.016      -0.010
cylinders            -4.2098      0.724     -5.816      0.000      -5.633      -2.787
weight:cylinders      0.0011      0.000      5.226      0.000       0.001       0.002
=====================================================================================
=======================================================================================
======
                               coef    std err          t      P>|t|      [0.025
0.975]
---------------------------------------------------------------------------------------
------
Intercept                   33.5124      3.420      9.798      0.000      26.788
40.237
horsepower                   0.0176      0.027      0.641      0.522      -0.036
0.072
acceleration                 0.8003      0.212      3.777      0.000       0.384
1.217
horsepower:acceleration     -0.0157      0.002     -7.838      0.000      -0.020
-0.012
=======================================================================================
======
=======================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
Intercept           -72.8784      8.368     -8.709      0.000     -89.330     -56.427
displacement          0.2523      0.041      6.216      0.000       0.173       0.332
year                  1.4077      0.110     12.779      0.000       1.191       1.624
displacement:year    -0.0041      0.001     -7.482      0.000      -0.005      -0.003
=======================================================================================
=======================================================================================
======
                               coef    std err          t      P>|t|      [0.025
0.975]
---------------------------------------------------------------------------------------
------
Intercept                   53.0511      1.526     34.765      0.000      50.051
56.051
horsepower                  -0.2343      0.020    -11.960      0.000      -0.273
-0.196
displacement                -0.0980      0.007    -14.674      0.000      -0.111
-0.085
horsepower:displacement      0.0006   5.19e-05     11.222      0.000       0.000
0.001
=======================================================================================
======
```

Interactions between weight and cylinders, and interactions between horsepower and acceleration seemed to be statistically significant (low p value). Interactions between displacement and year, and horsepower and displacement also seemed to be significant.

e2)

In [ ]:
```
model_interaction = smf.ols(formula='mpg ~ weight:cylinders', data = df).fit()
summary = model_interaction.summary()
print(summary.tables[1])

model_interaction = smf.ols(formula='mpg ~ horsepower:acceleration', data = df).fit()
summary = model_interaction.summary()
print(summary.tables[1])


model_interaction = smf.ols(formula='mpg ~ displacement:weight', data = df).fit()
summary = model_interaction.summary()
print(summary.tables[1])
```

```
====================================================================================
                       coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------
Intercept           34.2948      0.464     73.906      0.000      33.382      35.207
weight:cylinders    -0.0006   2.28e-05    -27.029      0.000      -0.001      -0.001
====================================================================================
=======================================================================================
======
                              coef    std err          t      P>|t|      [0.025      
0.975]
---------------------------------------------------------------------------------------
------
Intercept                  47.7299      0.978     48.789      0.000      45.807      
49.653
horsepower:acceleration    -0.0157      0.001    -25.612      0.000      -0.017      
-0.014
=======================================================================================
======
=======================================================================================
==
                          coef    std err          t      P>|t|      [0.025      0.97
5]
---------------------------------------------------------------------------------------
--
Intercept              31.2634      0.388     80.588      0.000      30.501      32.0
26
displacement:weight -1.182e-05    4.6e-07    -25.687      0.000   -1.27e-05   -1.09e-
05
=======================================================================================
==
```

There is a significance in the interaction between weight and cylinders, horsepower and acceleration, and displacement and weight.

f)

In [ ]:
```
#log
model = smf.ols(formula='mpg ~ cylinders + displacement + np.log(horsepower) + weight
summary = model.summary()
print("F value: {}".format(model.fvalue) )
print(model.summary().tables[1])
```

```
#quadratic
model = smf.ols(formula='mpg ~ cylinders + displacement + np.power(horsepower, 2) + we
summary = model.summary()
print("F value: {}".format(model.fvalue) )
print(model.summary().tables[1])
```

```
F value: 286.8516549018523
==================================================================================
===
                          coef    std err          t      P>|t|      [0.025      0.9
75]
----------------------------------------------------------------------------------
---
Intercept              42.9742     10.630      4.043      0.000      22.075      63.
874
cylinders              -0.4384      0.305     -1.439      0.151      -1.037       0.
161
displacement            0.0156      0.007      2.231      0.026       0.002       0.
029
np.log(horsepower)    -10.4455      1.522     -6.863      0.000     -13.438      -7.
453
weight                 -0.0038      0.001     -5.359      0.000      -0.005      -0.
002
np.log(acceleration)   -6.1583      1.645     -3.744      0.000      -9.392      -2.
925
year                    0.7050      0.048     14.683      0.000       0.611       0.
799
origin                  1.4379      0.258      5.574      0.000       0.931       1.
945
==================================================================================
===
F value: 256.6597155187827
==================================================================================
========
                               coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------------------
--------
Intercept                  -20.3992      4.117     -4.955      0.000     -28.494
-12.304
cylinders                   -0.2820      0.327     -0.862      0.389      -0.925
0.361
displacement                 0.0106      0.008      1.355      0.176      -0.005
0.026
np.power(horsepower, 2)    8.48e-05   4.14e-05      2.049      0.041    3.42e-06
0.000
weight                      -0.0071      0.001    -12.208      0.000      -0.008
-0.006
np.power(acceleration, 2)    0.0080      0.003      3.194      0.002       0.003
0.013
year                         0.7845      0.050     15.587      0.000       0.686
0.883
origin                       1.1926      0.280      4.267      0.000       0.643
1.742
==================================================================================
========
```

I tried performing a log transformation, and a quadratic transformation on displacement and horsepower, as they seemed to have a non-linear relationship to mpg in the scatterplot matrix. The previous F value for the linear regression is 256.

The log transformation yielded an F value of 305, which means that we can reject the null hypothesis that hte coefficients are equal to zero. The log transformation on displacement increased the p value of the coefficient, and on horsepower significantly decreased the p value of the coefficient. All of the log terms were significant with a 0.05 significance level.

The quadratic transformation yielded an F value of 256, so we can reject the null hypothesis that the coefficients are equal to zero. The quadratic transformation on displacement, horsepower, and weight decreased the p value on the displacement coefficient, slightly increased the p value on the horsepower coefficient. All of the squared terms were significant with a 0.05 significance level.

3.

a)

In [ ]:

```
df = pd.read_csv("Carseats.csv")

#convert to 0 and 1
df['Urban'] = df['Urban'].eq('Yes').mul(1)
df['US'] = df['US'].eq('Yes').mul(1)

x = df[['Price', 'Urban', 'US']]

y = df['Sales'].values.reshape(-1, 1)
X = sm.add_constant(x)
model = sm.OLS(y,X).fit()


# model = smf.ols(formula='Sales ~ Price + Urban + US', data = df).fit()
# summary = model.summary()
print(model.summary())
print("RSE: {}".format(np.sqrt(model.scale)))
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       0.239
Model:                            OLS   Adj. R-squared:                  0.234
Method:                 Least Squares   F-statistic:                     41.52
Date:                Tue, 15 Nov 2022   Prob (F-statistic):           2.39e-23
Time:                        18:40:09   Log-Likelihood:                -927.66
No. Observations:                 400   AIC:                             1863.
Df Residuals:                     396   BIC:                             1879.
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         13.0435      0.651     20.036      0.000      11.764      14.323
Price         -0.0545      0.005    -10.389      0.000      -0.065      -0.044
Urban         -0.0219      0.272     -0.081      0.936      -0.556       0.512
US             1.2006      0.259      4.635      0.000       0.691       1.710
==============================================================================
Omnibus:                        0.676   Durbin-Watson:                   1.912
Prob(Omnibus):                  0.713   Jarque-Bera (JB):                0.758
Skew:                           0.093   Prob(JB):                        0.684
Kurtosis:                       2.897   Cond. No.                         628.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
RSE: 2.4724924402701642
```

b)

For each one unit increase in price, sales decrease by -0.0545 on average while US and Urban are constant. Sales are -0.0219 units lower for observations that are Urban while US and Price are constant, and 1.2006 units higher for observations that are US while Urban and Price are constant

c)

for non-urban, non-us: Sales = 13.0435 - 0.0545(Price)

for urban, non-us: Sales = 13.0435 - 0.0545(Price) - 0.0219 + e

for non-urban, us: Sales = 13.0435 - 0.0545(Price) + 1.2006 + e

for urban and us: Sales = 13.0435 - 0.0545(Price) - 0.0219 + 1.2006 + e

d)

we can reject the null hypothesis for predictors Price and US, as their p value is below our alpha of 0.05. We cannot reject the null hypothesis of Urban, as the p value is very high (above our alpha of 0.05).

e)

In [ ]:
```
x = df[['Price', 'US']]
```

```
y = df['Sales'].values.reshape(-1, 1)
X = sm.add_constant(x)
model = sm.OLS(y,X).fit()

print(model.summary())
print("RSE: {}".format(np.sqrt(model.scale)))
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.239
Model:                            OLS   Adj. R-squared:                  0.235
Method:                 Least Squares   F-statistic:                     62.43
Date:                Tue, 15 Nov 2022   Prob (F-statistic):           2.66e-24
Time:                        18:40:09   Log-Likelihood:                -927.66
No. Observations:                 400   AIC:                             1861.
Df Residuals:                     397   BIC:                             1873.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         13.0308      0.631     20.652      0.000      11.790      14.271
Price         -0.0545      0.005    -10.416      0.000      -0.065      -0.044
US             1.1996      0.258      4.641      0.000       0.692       1.708
==============================================================================
Omnibus:                        0.666   Durbin-Watson:                   1.912
Prob(Omnibus):                  0.717   Jarque-Bera (JB):                0.749
Skew:                           0.092   Prob(JB):                        0.688
Kurtosis:                       2.895   Cond. No.                         607.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
RSE: 2.469396800574444
```

f) The models in a and e likely fit the data okay, as they have a low R-squared value of around 0.24 for both. The RSE of both models are above 1, but relatively close to 1. The model in e has a slightly better RSE, and R squared value, so it fits the data a bit better than the model in a.

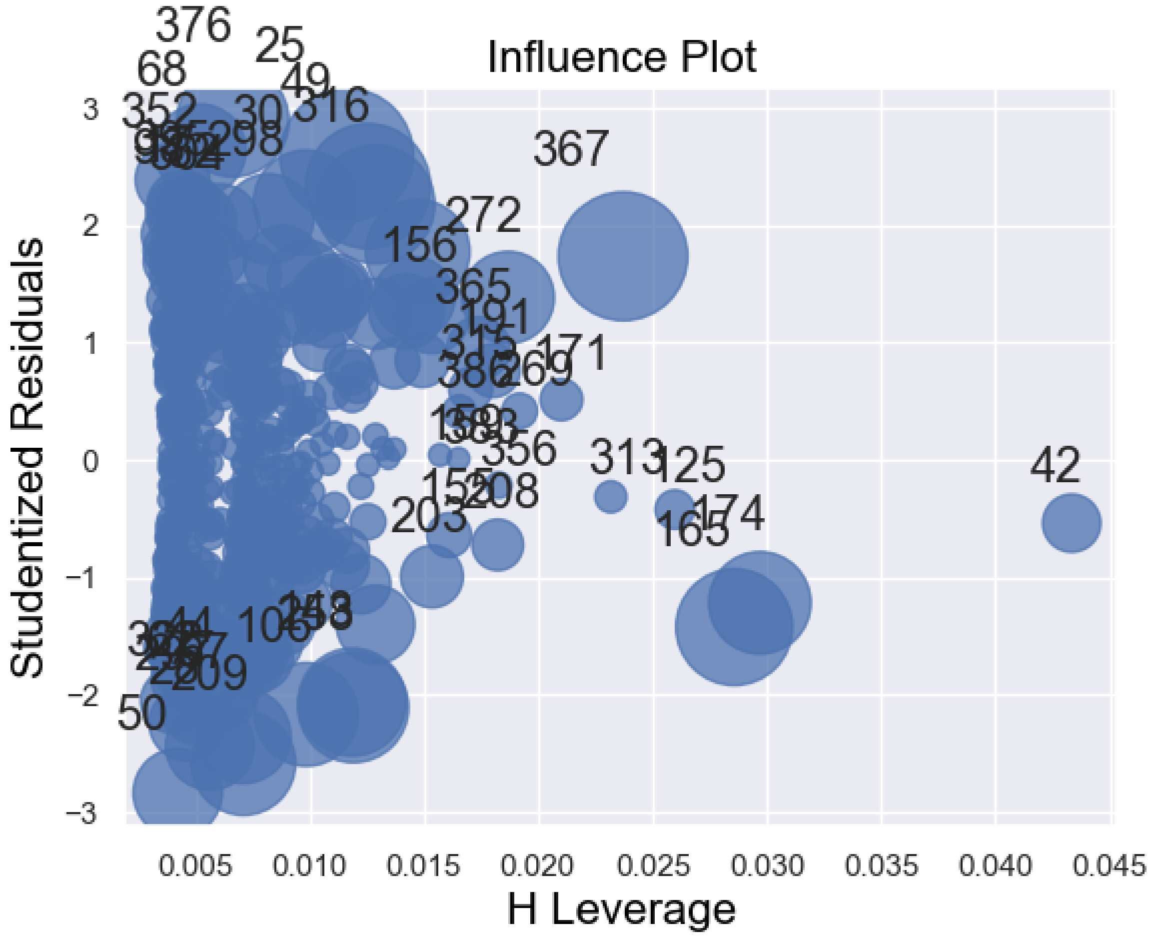
g)

95% confidence interval

coefficient	> 0.025	< 0.975
Price	-0.065	-0.044
US	0.692	1.708

h)

In [ ]:
```
sm.graphics.influence_plot(model)
plt.show()
```



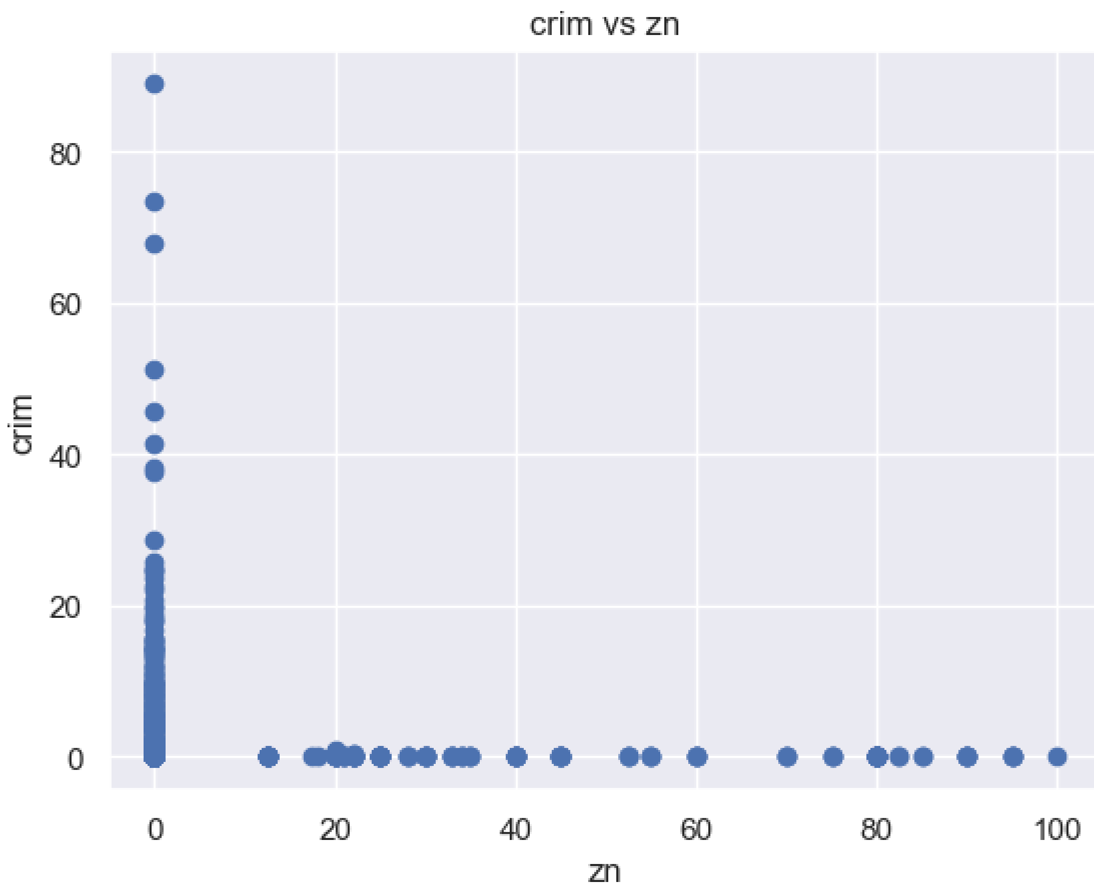
There is evidence of both outliers and high leverage points in the model. We can see that observation 376 is possibly an outlier, becuase it has a high studentized residual. We can also see that some observations have very high leverage, such as observation 42, as they are far outside the normal amount of leverage for this model, which hovers around about 0.010.

4.

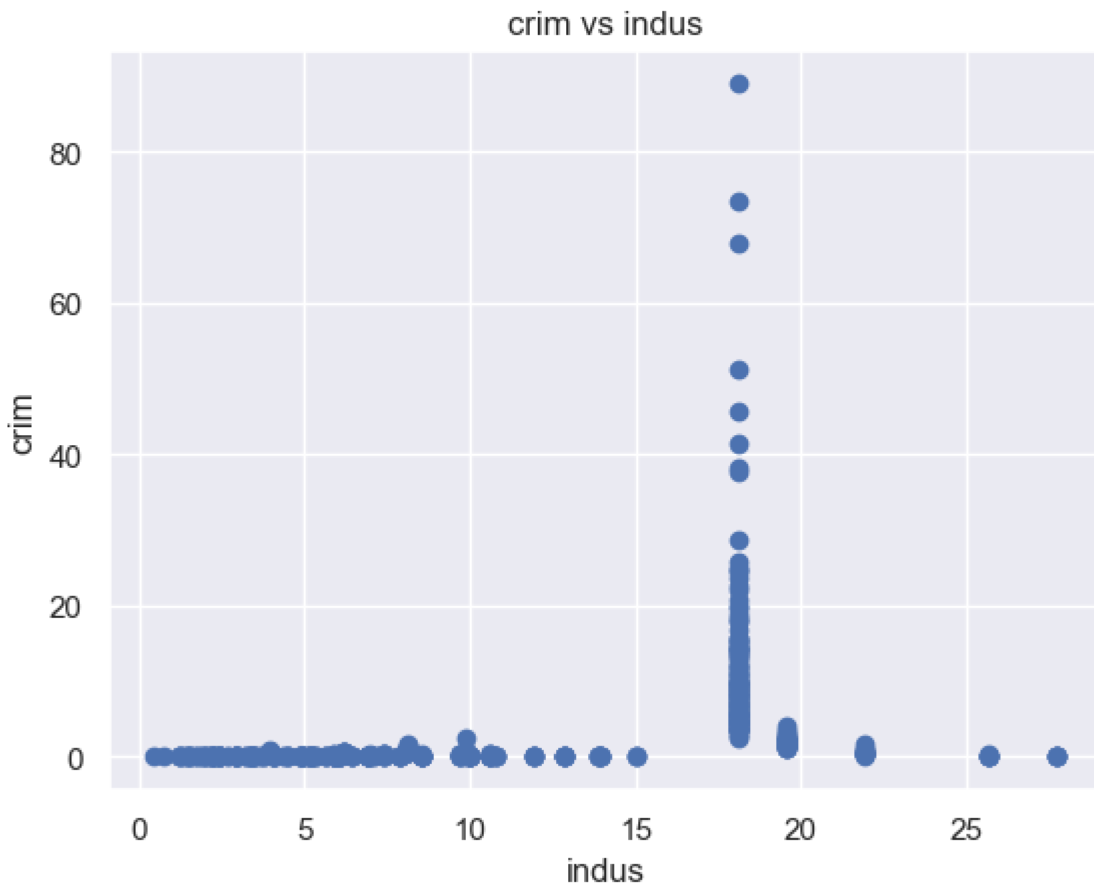
a)

In [ ]:
```
#we want to predict per capita crime rate.
df = pd.read_csv("Boston.csv", index_col=0)
y = df['crim'].values.reshape(-1, 1)
all_x = df.drop(['crim'], axis=1)


univar_coefficients = []
# y = df['Sales'].values.reshape(-1, 1)
for col in all_x:
    x = df[col]
    X = sm.add_constant(x)
    model = sm.OLS(y,X).fit()
    plt.scatter(x, y)
    plt.xlabel(col)
    plt.ylabel('crim')
    plt.title('crim vs {}'.format(col))
```

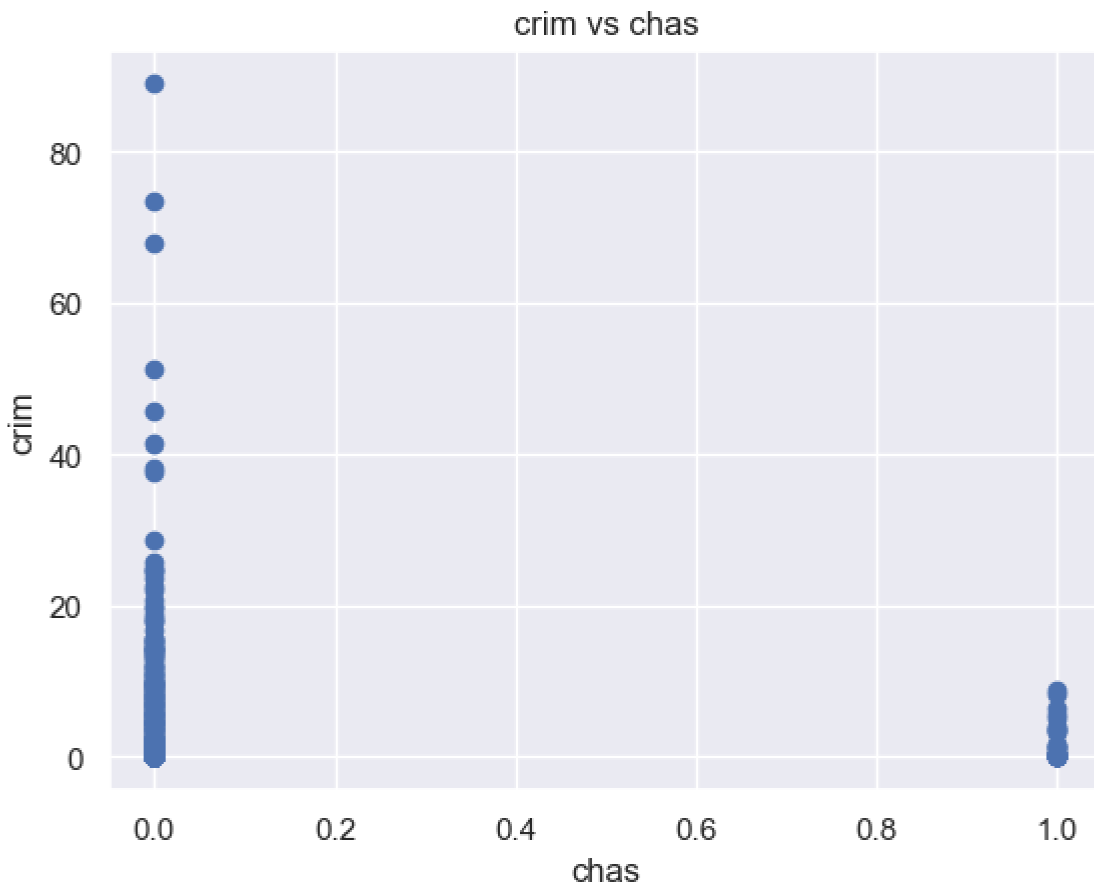
```
    plt.show()
    print(model.summary().tables[1])
    univar_coefficients.append(model.params[1])
```



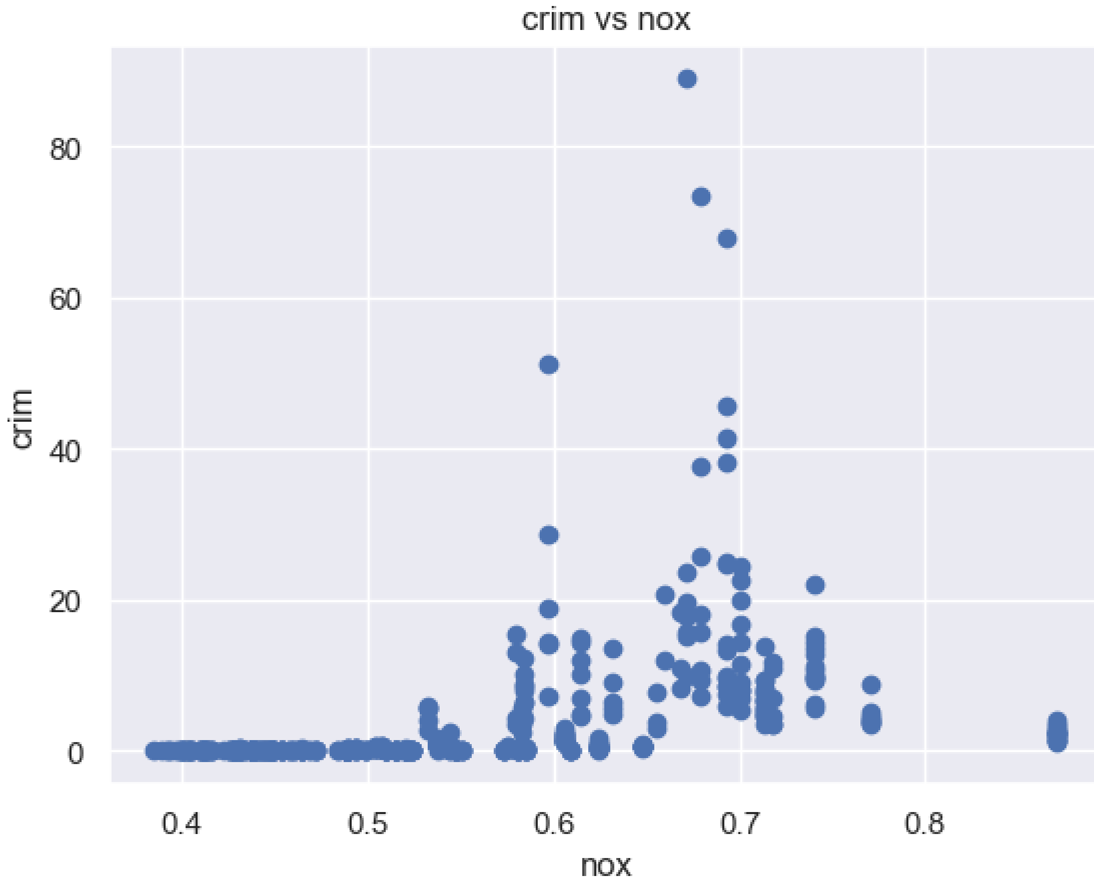

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          4.4537      0.417     10.675      0.000       3.634       5.273
zn            -0.0739      0.016     -4.594      0.000      -0.106      -0.042
==============================================================================
```



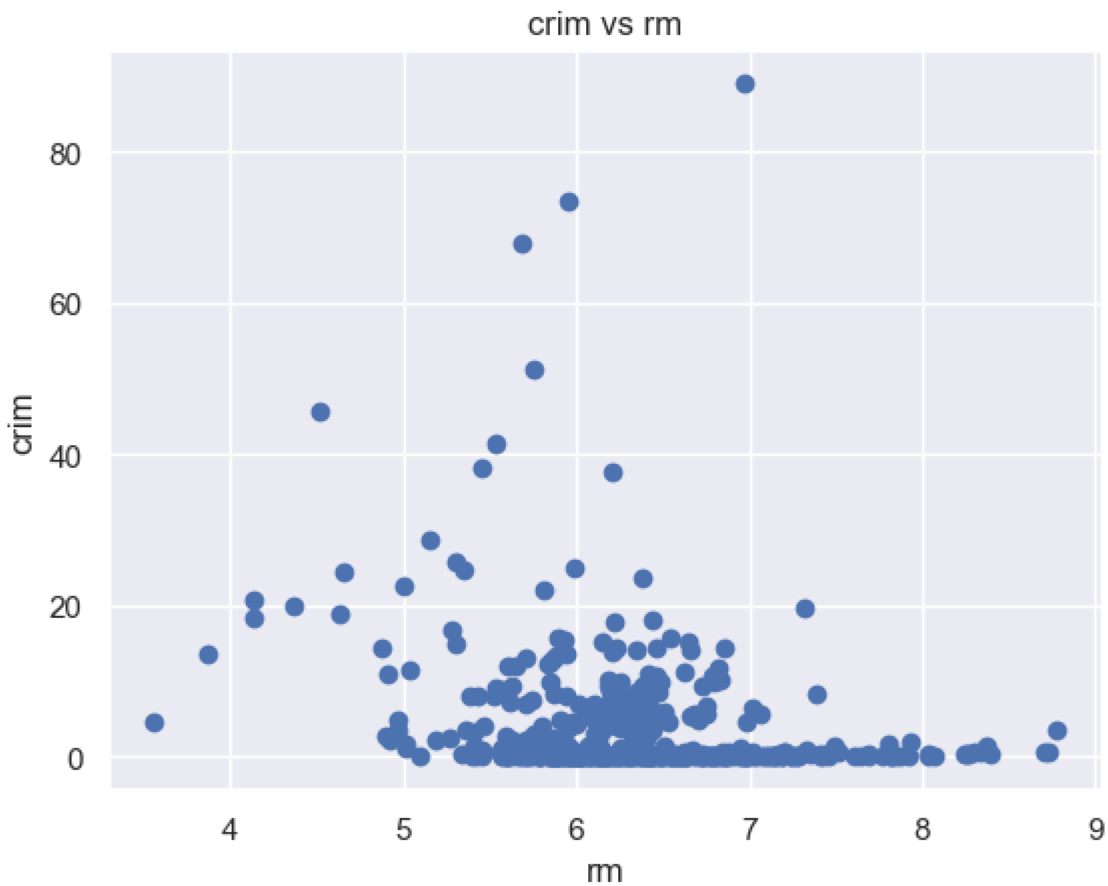
	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.0637	0.667	-3.093	0.002	-3.375	-0.753
indus	0.5098	0.051	9.991	0.000	0.410	0.610



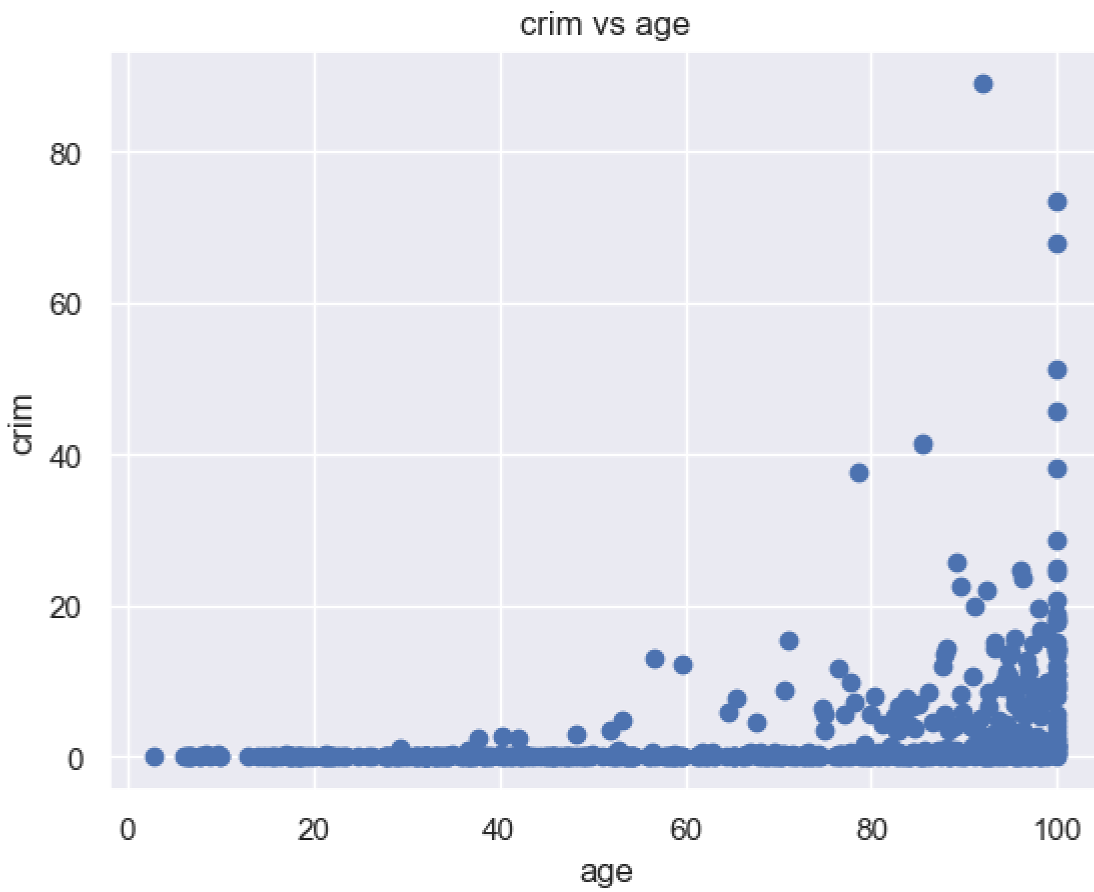

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          3.7444      0.396      9.453      0.000       2.966       4.523
chas          -1.8928      1.506     -1.257      0.209      -4.852       1.066
==============================================================================
```



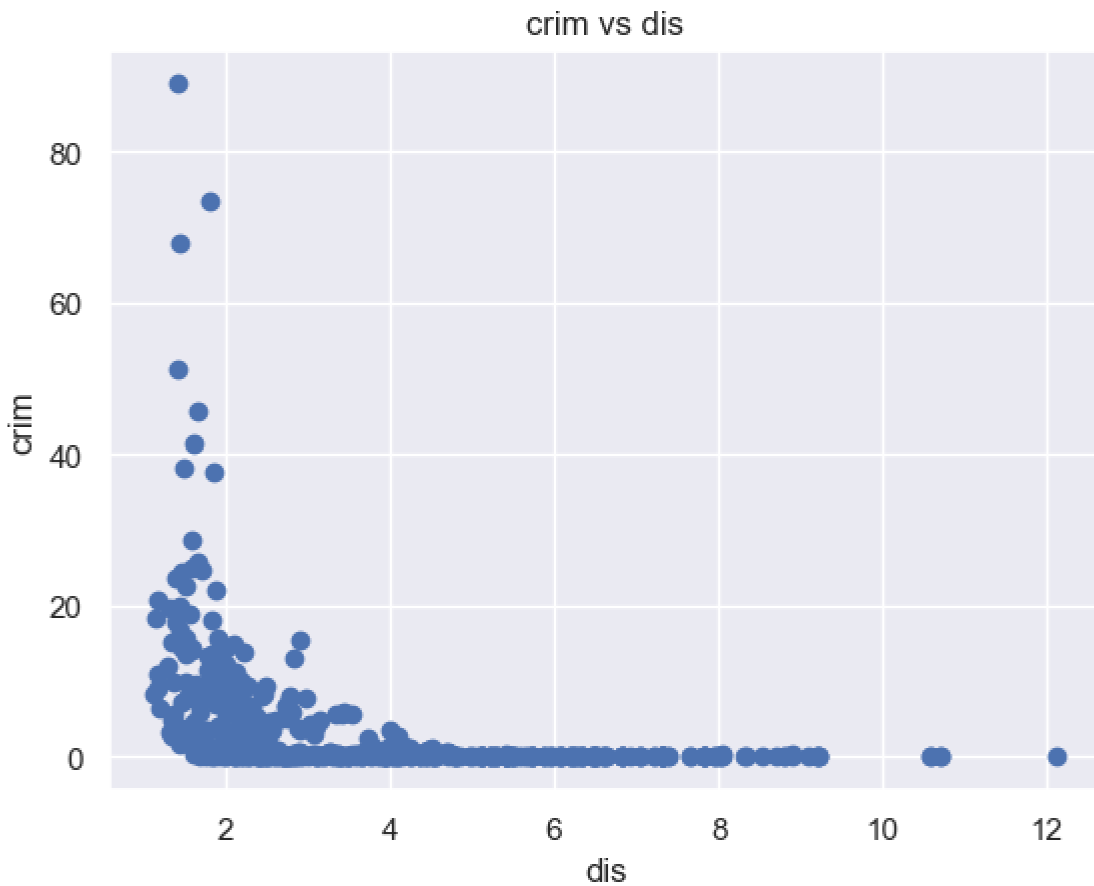

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -13.7199      1.699     -8.073      0.000     -17.059     -10.381
nox           31.2485      2.999     10.419      0.000      25.356      37.141
==============================================================================
```



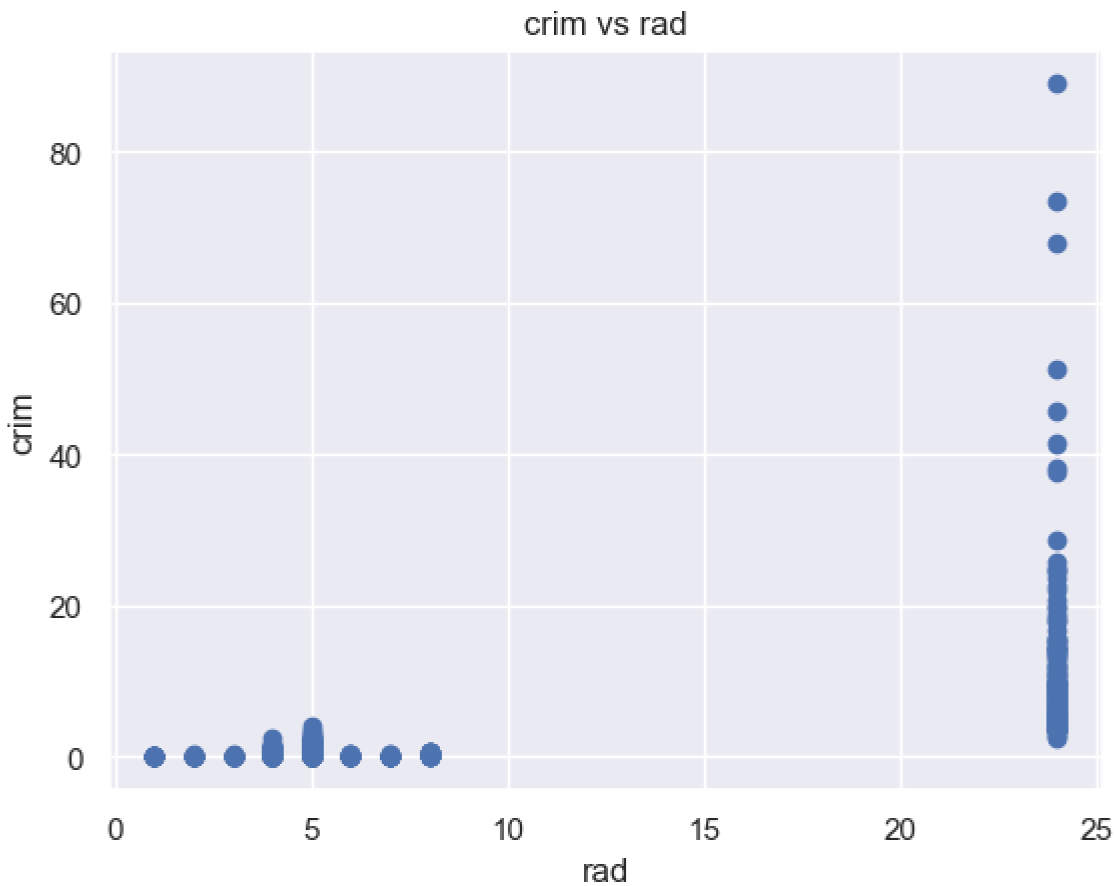

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         20.4818      3.364      6.088      0.000      13.872      27.092
rm            -2.6841      0.532     -5.045      0.000      -3.729      -1.639
==============================================================================
```



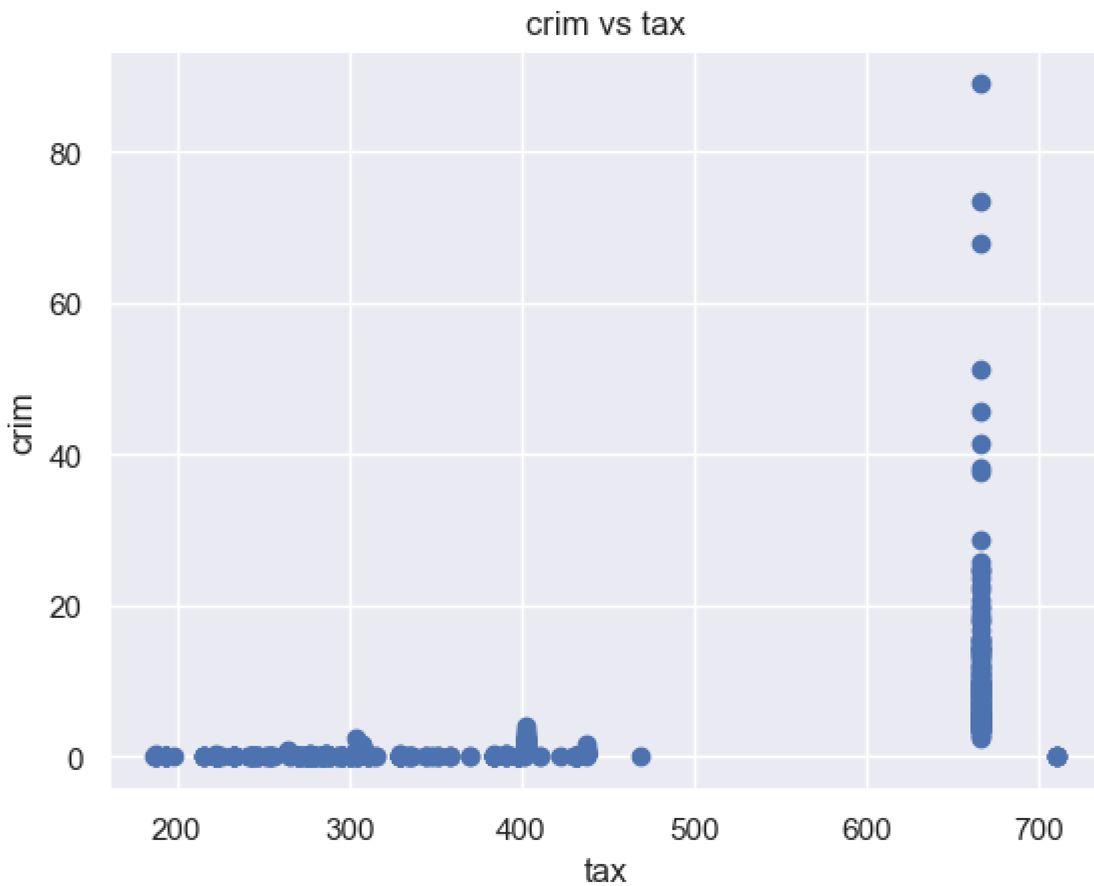

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -3.7779      0.944     -4.002      0.000      -5.633      -1.923
age            0.1078      0.013      8.463      0.000       0.083       0.133
==============================================================================
```



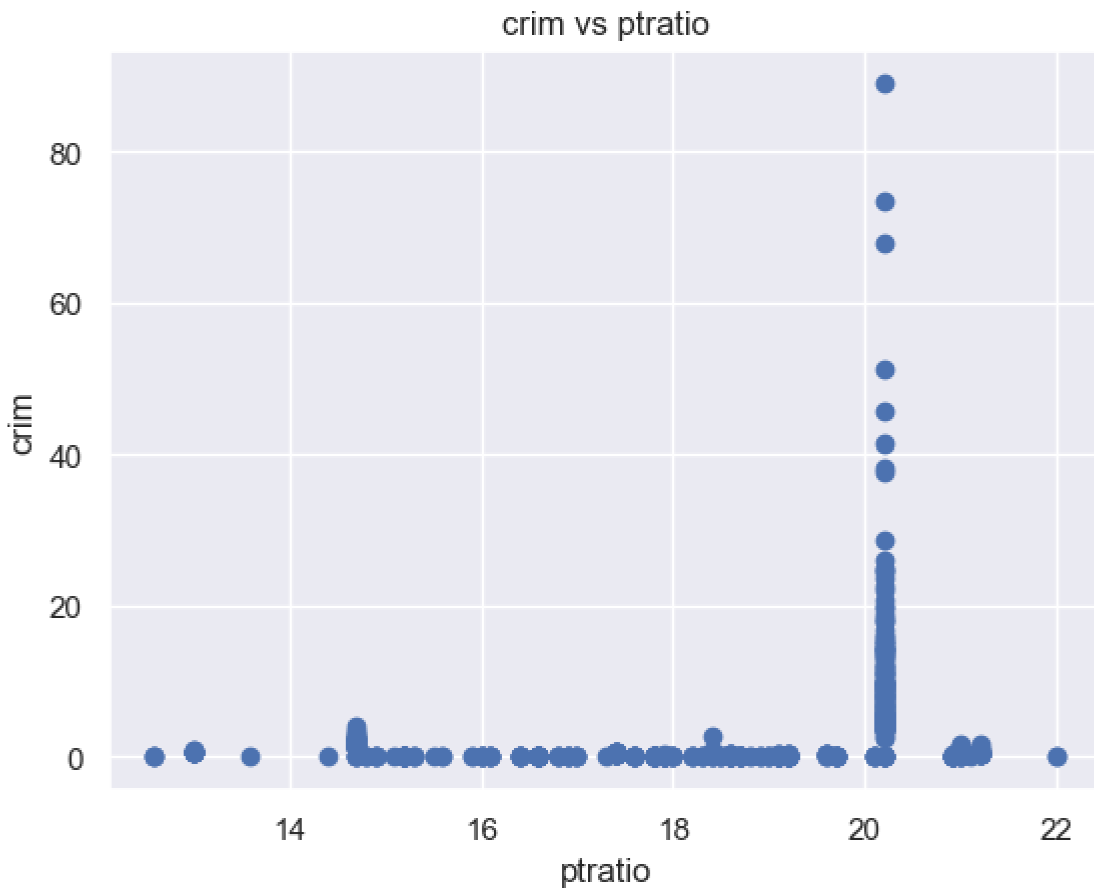

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.4993      0.730     13.006      0.000       8.064      10.934
dis           -1.5509      0.168     -9.213      0.000      -1.882      -1.220
==============================================================================
```



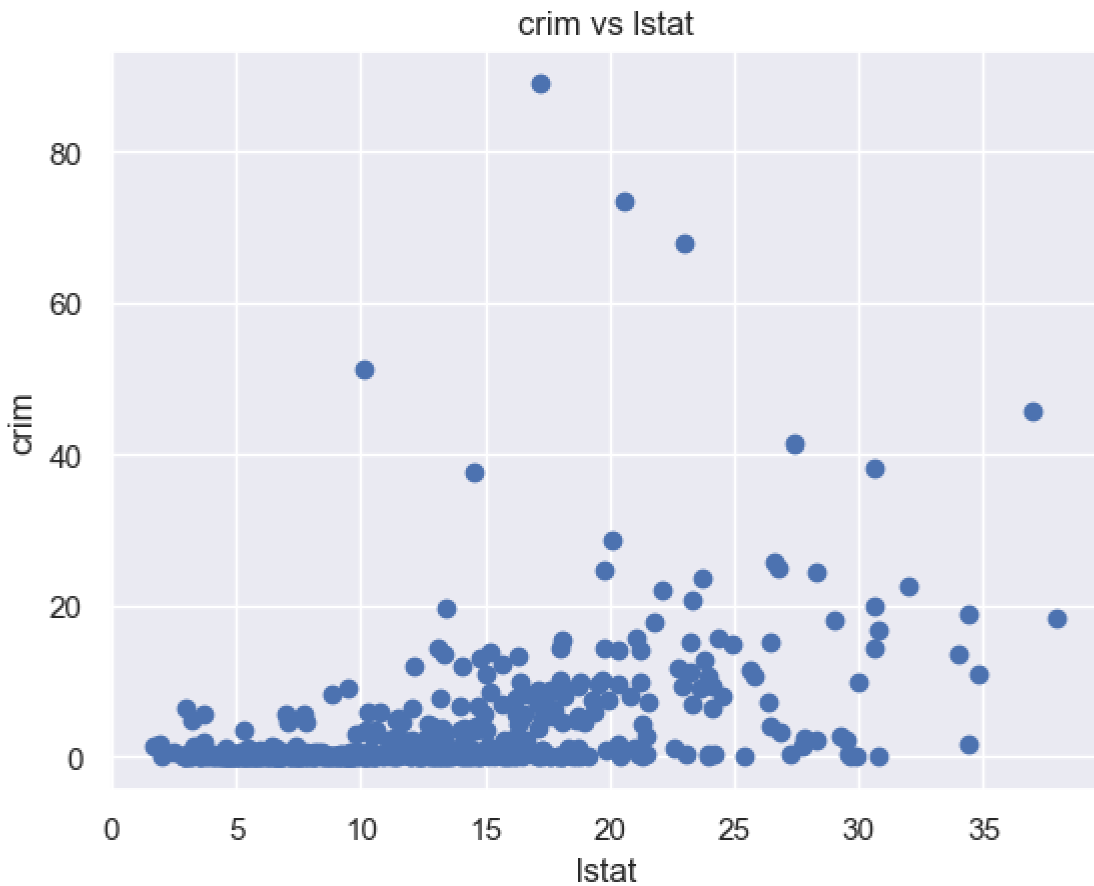

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -2.2872      0.443     -5.157      0.000      -3.158      -1.416
rad            0.6179      0.034     17.998      0.000       0.550       0.685
==============================================================================
```



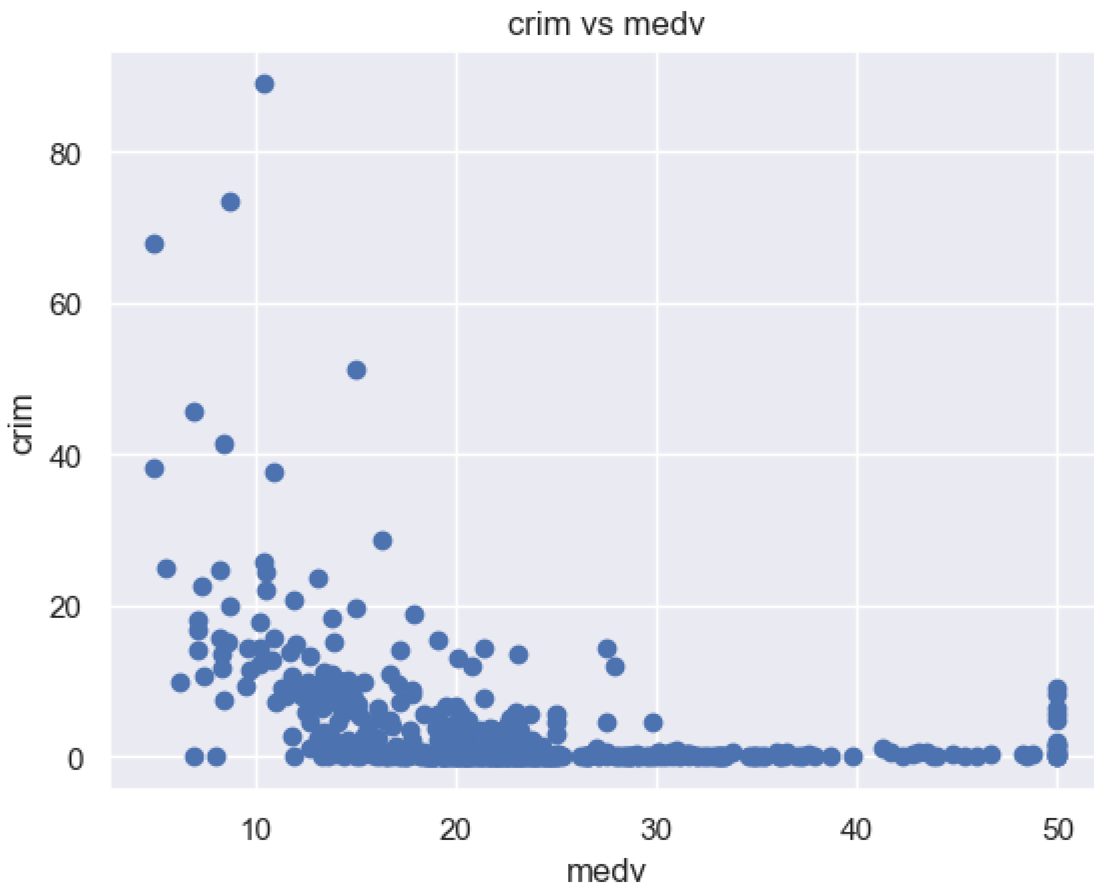

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -8.5284      0.816    -10.454      0.000     -10.131      -6.926
tax            0.0297      0.002     16.099      0.000       0.026       0.033
==============================================================================
```




```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -17.6469      3.147     -5.607      0.000     -23.830     -11.464
ptratio        1.1520      0.169      6.801      0.000       0.819       1.485
==============================================================================
```



	coef	std err	t	P>|t|	[0.025	0.975]
const	-3.3305	0.694	-4.801	0.000	-4.694	-1.968
lstat	0.5488	0.048	11.491	0.000	0.455	0.643




```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         11.7965      0.934     12.628      0.000       9.961      13.632
medv          -0.3632      0.038     -9.460      0.000      -0.439      -0.288
==============================================================================
```

all predictors have a p value of < 0.05 except for 'chas' so we acn determine that there is a statistically significant association between the predictors and response except for in 'chas'.

b)

In [ ]:
```
X = sm.add_constant(all_x)
model = sm.OLS(y,X).fit()

print(model.summary())
```

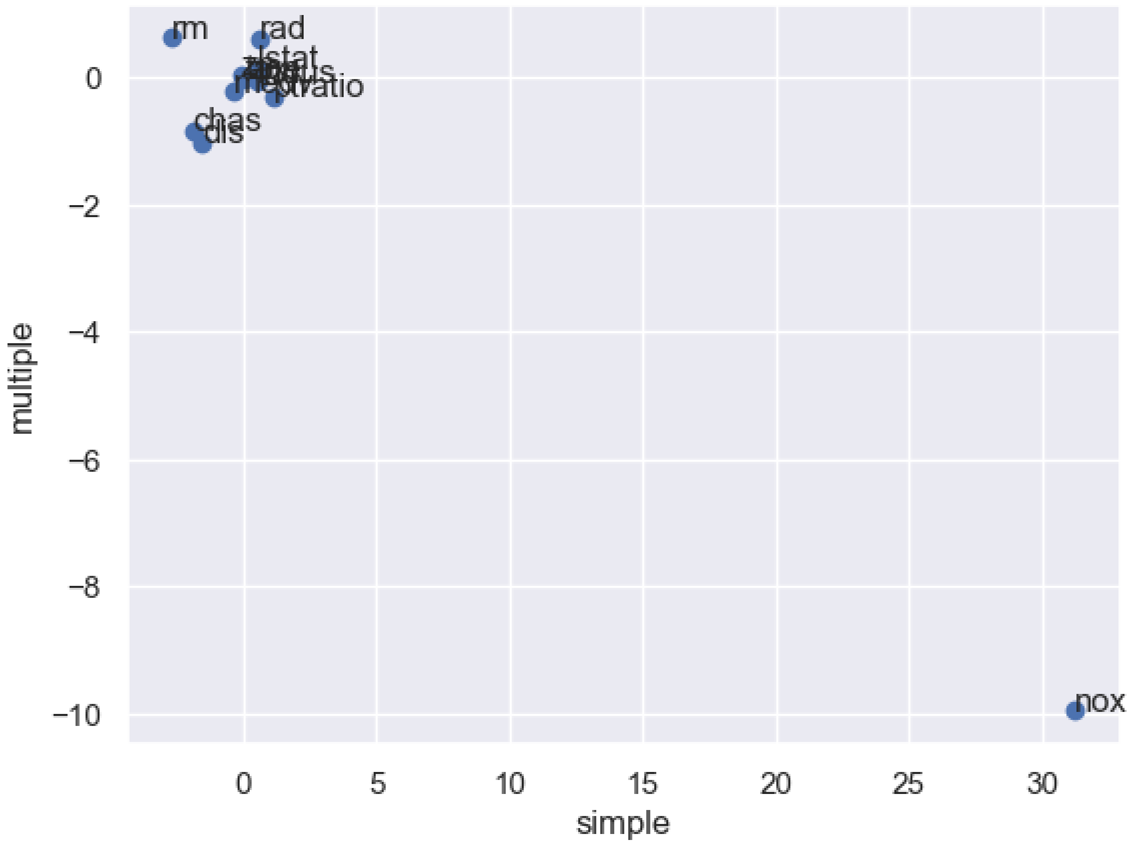
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.449
Model:                            OLS   Adj. R-squared:                  0.436
Method:                 Least Squares   F-statistic:                     33.52
Date:                Tue, 15 Nov 2022   Prob (F-statistic):           2.03e-56
Time:                        18:40:11   Log-Likelihood:                -1655.4
No. Observations:                 506   AIC:                             3337.
Df Residuals:                     493   BIC:                             3392.
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         13.7784      7.082      1.946      0.052      -0.136      27.693
zn             0.0457      0.019      2.433      0.015       0.009       0.083
indus         -0.0584      0.084     -0.698      0.486      -0.223       0.106
chas          -0.8254      1.183     -0.697      0.486      -3.150       1.500
nox           -9.9576      5.290     -1.882      0.060     -20.351       0.436
rm             0.6289      0.607      1.036      0.301      -0.564       1.822
age           -0.0008      0.018     -0.047      0.962      -0.036       0.034
dis           -1.0122      0.282     -3.584      0.000      -1.567      -0.457
rad            0.6125      0.088      6.997      0.000       0.440       0.784
tax           -0.0038      0.005     -0.730      0.466      -0.014       0.006
ptratio       -0.3041      0.186     -1.632      0.103      -0.670       0.062
lstat          0.1388      0.076      1.833      0.067      -0.010       0.288
medv          -0.2201      0.060     -3.678      0.000      -0.338      -0.103
==============================================================================
Omnibus:                      663.436   Durbin-Watson:                   1.516
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            80856.852
Skew:                           6.579   Prob(JB):                         0.00
Kurtosis:                      63.514   Cond. No.                     1.24e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
ified.
[2] The condition number is large, 1.24e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The r squared is relatively high, which means that the model is a relatively good fit for the data. We can reject the null hypothesis for zn, dis, rad, and medv, as their p values are below our alpha of 0.05.

c)

In [ ]:
```
multiple_reg = model.params[1:]
data = pd.DataFrame(multiple_reg, columns = ['multiple'] )
data['univar'] = univar_coefficents
plt.scatter(data['univar'], data['multiple'])
plt.xlabel('simple')
plt.ylabel('multiple')

for index, row in data.iterrows():
    plt.annotate(index, (row['univar'], row['multiple']))
```

```
plt.show()
```



The results showed that there is a difference for the simple and multiple regression coefficients. This is due to the fact that in the simple regression, the slope represents the average effect of the predictor, ignoring other factors. In a multiple regression, the slop represents the average effect of the predictor holding other factors constant.

d)

In [ ]:
```
#we want to predict per capita crime rate.
df = pd.read_csv("Boston.csv", index_col=0)
y = df['crim'].values.reshape(-1, 1)
all_x = df.drop(['crim'], axis=1)


univar_coefficients = []
# y = df['Sales'].values.reshape(-1, 1)
for col in all_x:
    x = df[col]
    model = smf.ols(formula='crim  ~ {} + np.power({}, 2) + np.power({}, 3)'.format(cc
    print(model.summary().tables[1])
```

```
==============================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept           4.8461      0.433     11.192      0.000       3.995       5.697
zn                 -0.3322      0.110     -3.025      0.003      -0.548      -0.116
np.power(zn, 2)     0.0065      0.004      1.679      0.094      -0.001       0.014
np.power(zn, 3) -3.776e-05   3.14e-05     -1.203      0.230   -9.94e-05    2.39e-05
==============================================================================
================================================================================
=
                         coef    std err          t      P>|t|      [0.025      0.97
5]
--------------------------------------------------------------------------------
-
Intercept              3.6626      1.574      2.327      0.020       0.570       6.75
5
indus                 -1.9652      0.482     -4.077      0.000      -2.912      -1.01
8
np.power(indus, 2)     0.2519      0.039      6.407      0.000       0.175       0.32
9
np.power(indus, 3)    -0.0070      0.001     -7.292      0.000      -0.009      -0.00
5
================================================================================
=
================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept             3.7444      0.397      9.444      0.000       2.965       4.523
chas               1.114e+14   2.71e+14      0.411      0.681   -4.21e+14    6.44e+14
np.power(chas, 2)  -5.61e+13   1.37e+14     -0.411      0.681   -3.24e+14    2.12e+14
np.power(chas, 3) -5.532e+13   1.35e+14     -0.411      0.681    -3.2e+14    2.09e+14
================================================================================
===============================================================================
                       coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept          233.0866     33.643      6.928      0.000     166.988     299.185
nox              -1279.3713    170.397     -7.508      0.000   -1614.151    -944.591
np.power(nox, 2)  2248.5441    279.899      8.033      0.000    1698.626    2798.462
np.power(nox, 3) -1245.7029    149.282     -8.345      0.000   -1538.997    -952.409
===============================================================================
==============================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept         112.6246     64.517      1.746      0.081     -14.132     239.382
rm                -39.1501     31.311     -1.250      0.212    -100.668      22.368
np.power(rm, 2)     4.5509      5.010      0.908      0.364      -5.292      14.394
np.power(rm, 3)    -0.1745      0.264     -0.662      0.509      -0.693       0.344
==============================================================================
===============================================================================
                       coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept           -2.5488      2.769     -0.920      0.358      -7.989       2.892
age                  0.2737      0.186      1.468      0.143      -0.093       0.640
np.power(age, 2)    -0.0072      0.004     -1.988      0.047      -0.014    -8.4e-05
np.power(age, 3)  5.745e-05   2.11e-05      2.724      0.007      1.6e-05    9.89e-05
===============================================================================
===============================================================================
```

```
                         coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------
Intercept             30.0476      2.446     12.285      0.000      25.242      34.853
dis                  -15.5544      1.736     -8.960      0.000     -18.965     -12.144
np.power(dis, 2)       2.4521      0.346      7.078      0.000       1.771       3.133
np.power(dis, 3)      -0.1186      0.020     -5.814      0.000      -0.159      -0.079
====================================================================================
====================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------
Intercept             -0.6055      2.050     -0.295      0.768      -4.633       3.422
rad                    0.5127      1.044      0.491      0.623      -1.538       2.563
np.power(rad, 2)      -0.0752      0.149     -0.506      0.613      -0.367       0.217
np.power(rad, 3)       0.0032      0.005      0.703      0.482      -0.006       0.012
====================================================================================
====================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------
Intercept             19.1836     11.796      1.626      0.105      -3.991      42.358
tax                   -0.1533      0.096     -1.602      0.110      -0.341       0.035
np.power(tax, 2)       0.0004      0.000      1.488      0.137      -0.000       0.001
np.power(tax, 3) -2.204e-07   1.89e-07     -1.167      0.244   -5.91e-07    1.51e-07
====================================================================================
=====================================================================================
===
                             coef    std err          t      P>|t|      [0.025      0.9
75]
-------------------------------------------------------------------------------------
---
Intercept                477.1840    156.795      3.043      0.002     169.129     785.
239
ptratio                  -82.3605     27.644     -2.979      0.003    -136.673     -28.
048
np.power(ptratio, 2)       4.6353      1.608      2.882      0.004       1.475       7.
795
np.power(ptratio, 3)      -0.0848      0.031     -2.743      0.006      -0.145      -0.
024
=====================================================================================
===
=====================================================================================
=
                           coef    std err          t      P>|t|      [0.025      0.97
5]
-------------------------------------------------------------------------------------
-
Intercept                1.2010      2.029      0.592      0.554      -2.785       5.18
7
lstat                   -0.4491      0.465     -0.966      0.335      -1.362       0.46
4
np.power(lstat, 2)       0.0558      0.030      1.852      0.065      -0.003       0.11
5
np.power(lstat, 3)      -0.0009      0.001     -1.517      0.130      -0.002       0.00
0
=====================================================================================
=
=====================================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
```

```
------------------------------------------------------------------------------------
Intercept             53.1655      3.356     15.840      0.000      46.571      59.760
medv                  -5.0948      0.434    -11.744      0.000      -5.947      -4.242
np.power(medv, 2)      0.1555      0.017      9.046      0.000       0.122       0.189
np.power(medv, 3)     -0.0015      0.000     -7.312      0.000      -0.002      -0.001
====================================================================================
```

For zn, chas, rm, rad, tax, and lstat there was no statistical significance pointing to a non-linear association. For indus, nox, age, dis, ptratio, and medv there was a statistical significance pointing to a non-linear association.