In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import scale
import sklearn.linear_model as lm
import statsmodels.formula.api as smf
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

In [ ]:
```
adv = pd.read_csv('/content/gdrive/MyDrive/ISLR/Advertising.csv',usecols=[1,2,3,4])
```

In [ ]:
```
adv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   radio      200 non-null    float64
 2   newspaper  200 non-null    float64
 3   sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

In [ ]:
```
credit = pd.read_csv('/content/gdrive/MyDrive/ISLR/Credit.csv')
```

In [ ]:

```
credit.head()
```

Out[17]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance
0	1	14.891	3606	283	2	34	11	Male	No	Yes	Caucasian	333
1	2	106.025	6645	483	3	82	15	Female	Yes	Yes	Asian	903
2	3	104.593	7075	514	4	71	11	Male	No	No	Asian	580
3	4	148.924	9504	681	3	36	11	Female	No	No	Asian	964
4	5	55.882	4897	357	2	68	16	Male	No	Yes	Caucasian	331

In [ ]:

```
credit['Student2']=credit.Student.map({'No':0,'Yes':1})
credit.head()
```

Out[18]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance	Student2
0	1	14.891	3606	283	2	34	11	Male	No	Yes	Caucasian	333	0
1	2	106.025	6645	483	3	82	15	Female	Yes	Yes	Asian	903	1
2	3	104.593	7075	514	4	71	11	Male	No	No	Asian	580	0
3	4	148.924	9504	681	3	36	11	Female	No	No	Asian	964	0
4	5	55.882	4897	357	2	68	16	Male	No	Yes	Caucasian	331	0

In [ ]:

```python
Auto = pd.read_csv('/content/gdrive/MyDrive/ISLR/Auto.csv')
Auto.isna().sum()
Auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 397 entries, 0 to 396
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           397 non-null    float64
 1   cylinders     397 non-null    int64  
 2   displacement  397 non-null    float64
 3   horsepower    397 non-null    object 
 4   weight        397 non-null    int64  
 5   acceleration  397 non-null    float64
 6   year          397 non-null    int64  
 7   origin        397 non-null    int64  
 8   name          397 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.0+ KB
```

In [ ]:

```python
print(Auto.to_markdown())
```

```
|     |   mpg |   cylinders |   displacement | horsepower   |   weight |   acceleration |   year |   origin | name
|
|----:|------:|------------:|---------------:|:-------------|---------:|---------------:|-------:|---------:|:---------
---------------------------|
|   0 |  18   |           8 |          307   | 130          |     3504 |           12   |     70 |        1 | chevrolet
chevelle malibu           |
|   1 |  15   |           8 |          350   | 165          |     3693 |           11.5 |     70 |        1 | buick sky
lark 320                    |
|   2 |  18   |           8 |          318   | 150          |     3436 |           11   |     70 |        1 | plymouth
satellite                   |
|   3 |  16   |           8 |          304   | 150          |     3433 |           12   |     70 |        1 | amc rebel
sst                       |
|   4 |  17   |           8 |          302   | 140          |     3449 |           10.5 |     70 |        1 | ford tori
no                          |
|   5 |  15   |           8 |          429   | 198          |     4341 |           10   |     70 |        1 | ford gala
xie 500                     |
|   6 |  14   |           8 |          454   | 220          |     4354 |            9   |     70 |        1 | chevrolet
impala                    |
|   7 |  14   |           8 |          440   | 215          |     4312 |            8.5 |     70 |        1 | plymouth
fury iii                    |
```

In [ ]:

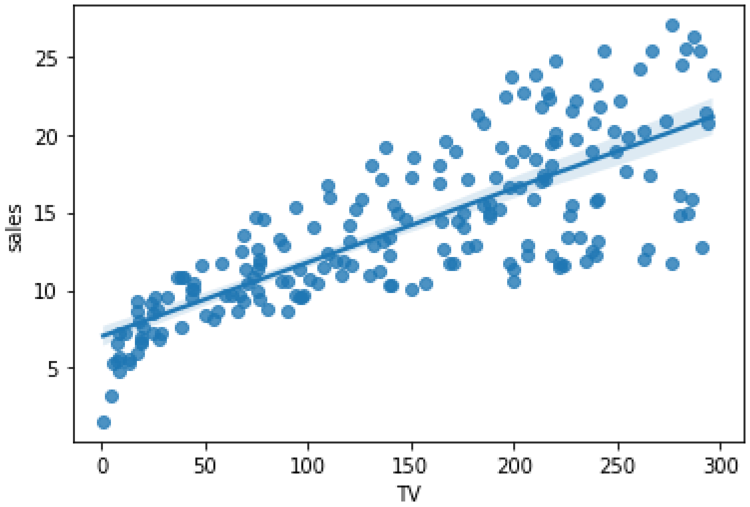
```python
Auto = pd.read_csv('/content/gdrive/MyDrive/ISLR/Auto.csv',na_values='?').dropna()
print(Auto.to_markdown())
```

```
|     |   mpg |   cylinders |   displacement |   horsepower |   weight |   acceleration |   year |   origin | name
|
|----:|------:|------------:|---------------:|-------------:|---------:|---------------:|-------:|---------:|:---------
----------------------------|
|   0 |  18   |           8 |          307   |          130 |     3504 |           12   |     70 |        1 | chevrolet
chevelle malibu              |
|   1 |  15   |           8 |          350   |          165 |     3693 |           11.5 |     70 |        1 | buick sky
lark 320                      |
|   2 |  18   |           8 |          318   |          150 |     3436 |           11   |     70 |        1 | plymouth
satellite                     |
|   3 |  16   |           8 |          304   |          150 |     3433 |           12   |     70 |        1 | amc rebel
sst                          |
|   4 |  17   |           8 |          302   |          140 |     3449 |           10.5 |     70 |        1 | ford tori
no                            |
|   5 |  15   |           8 |          429   |          198 |     4341 |           10   |     70 |        1 | ford gala
xie 500                       |
|   6 |  14   |           8 |          454   |          220 |     4354 |            9   |     70 |        1 | chevrolet
impala                       |
|   7 |  14   |           8 |          440   |          215 |     4312 |            8.5 |     70 |        1 | plymouth
```

In [ ]:
```python
##Least squares fit
sns.regplot(adv.TV,adv.sales,order=1)   ##check ploynomila or not
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x7f52ccf1f0a0>



In [ ]:
```python
lr = lm.LinearRegression().
X = scale(adv.TV,with_mean=True,with_std=False).reshape(-1,1)
Y = adv.sales
lr.fit(X,Y)
print(lr.intercept_)
print(lr.coef_)
```

```
14.0225
[0.04753664]
```

In [ ]:
```python
RSS = np.sum((lr.intercept_+lr.coef_*X-Y.values.reshape(-1,1))**2)
```

In [ ]:
```
RSS
```

Out[25]: 2102.5305831313512

In [ ]:
```
lm_fit1 = smf.ols('sales ~ TV', adv).fit()
lm_fit1.summary().tables[1]
```

Out[26]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	7.0326	0.458	15.360	0.000	6.130	7.935
TV	0.0475	0.003	17.668	0.000	0.042	0.053

In [ ]:
```
adv.head(3)
```

Out[27]:

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3

In [ ]:
```
lm_fit2 = smf.ols('sales ~ newspaper', adv).fit()
lm_fit2.summary().tables[1]
```

Out[28]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	12.3514	0.621	19.876	0.000	11.126	13.577
newspaper	0.0547	0.017	3.300	0.001	0.022	0.087

In [ ]:
```
##Multiple Linear Regression
lm_fit3 = smf.ols('sales ~ TV+radio+newspaper', adv).fit()
lm_fit3.summary().tables[1]
```

Out[29]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.9389	0.312	9.422	0.000	2.324	3.554
TV	0.0458	0.001	32.809	0.000	0.043	0.049
radio	0.1885	0.009	21.893	0.000	0.172	0.206
newspaper	-0.0010	0.006	-0.177	0.860	-0.013	0.011

In [ ]:
```
##Correlation Matrix
adv.corr()
```

Out[30]:

	TV	radio	newspaper	sales
TV	1.000000	0.054809	0.056648	0.782224
radio	0.054809	1.000000	0.354104	0.576223
newspaper	0.056648	0.354104	1.000000	0.228299
sales	0.782224	0.576223	0.228299	1.000000

In [ ]:
```
##RSS for Multiple regression
##https://github.com/JWarmenhoven/ISLR-python/blob/master/Notebooks/Chapter%203.ipynb

lr = lm.LinearRegression()

X = adv[['radio', 'TV']]
Y = adv.sales

lr.fit(X,Y)
print(lr.intercept_)
print(lr.coef_)
```

```
2.9210999124051398
[0.18799423 0.04575482]
```

In [ ]:
```
##Other Considerations in the Regression Model
credit.head(2)
```

Out[32]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance	Student2
0	1	14.891	3606	283	2	34	11	Male	No	Yes	Caucasian	333	0
1	2	106.025	6645	483	3	82	15	Female	Yes	Yes	Asian	903	1

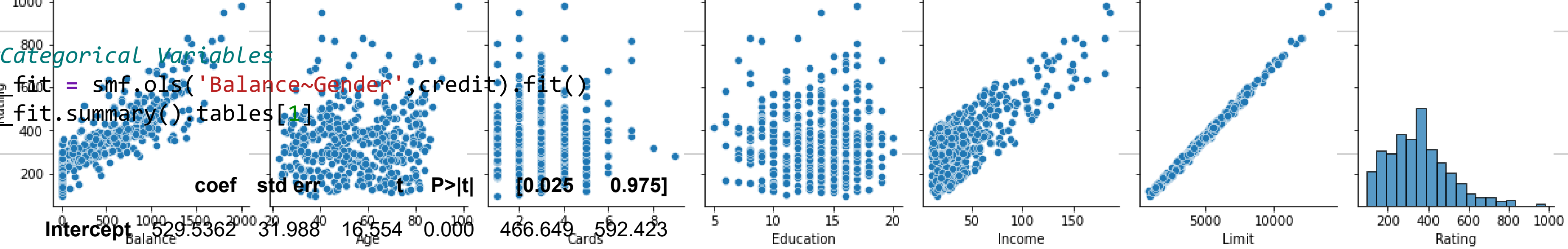
In [ ]:

```
sns.pairplot(credit[['Balance','Age','Cards','Education','Income','Limit','Rating']])
```

Out[33]:

```
<seaborn.axisgrid.PairGrid at 0x7f52cd3e14f0>
```

Balance
Age
Cards
Education
Income
Limit



In [ ]:
```
##Categorical Variables
lm_fit = smf.ols('Balance~Gender',credit).fit()
lm_fit.summary().tables[1]
```

Out[34]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	529.5362	31.988	16.554	0.000	466.649	592.423
Gender[T.Male]	-19.7331	46.051	-0.429	0.669	-110.267	70.801

In [ ]:
```
lm_fit1 = smf.ols('Balance~Ethnicity',credit).fit()
lm_fit1.summary().tables[1]
```

Out[35]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	531.0000	46.319	11.464	0.000	439.939	622.061
Ethnicity[T.Asian]	-18.6863	65.021	-0.287	0.774	-146.515	109.142
Ethnicity[T.Caucasian]	-12.5025	56.681	-0.221	0.826	-123.935	98.930

In [ ]:
```
adv.head(2)
```

Out[36]:

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4

In [ ]:
```
##Interaction effect
lm_fit2 = smf.ols('sales~TV+radio+TV*radio',adv).fit()
lm_fit2.summary().tables[1]
```

Out[37]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	6.7502	0.248	27.233	0.000	6.261	7.239
TV	0.0191	0.002	12.699	0.000	0.016	0.022
radio	0.0289	0.009	3.241	0.001	0.011	0.046
TV:radio	0.0011	5.24e-05	20.727	0.000	0.001	0.001

In [ ]:

```
credit.head(2)
```

Out[38]:

	Unnamed: 0	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance	Student2
0	1	14.891	3606	283	2	34	11	Male	No	Yes	Caucasian	333	0
1	2	106.025	6645	483	3	82	15	Female	Yes	Yes	Asian	903	1

In [ ]:

```
##Interaction between qualitative and quantative variables
lm_fit3 = smf.ols('Balance~Income',credit).fit()
print(lm_fit3.summary())
lm_fit4 = smf.ols('Balance~Income+Student2+Income*Student2',credit).fit()
print(lm_fit4.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                Balance   R-squared:                       0.215
Model:                            OLS   Adj. R-squared:                  0.213
Method:                 Least Squares   F-statistic:                     109.0
Date:                Thu, 29 Dec 2022   Prob (F-statistic):           1.03e-22
Time:                        05:50:06   Log-Likelihood:                -2970.9
No. Observations:                 400   AIC:                             5946.
Df Residuals:                     398   BIC:                             5954.
Df Model:                           1                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    246.5148     33.199      7.425      0.000     181.247     311.783
Income         6.0484      0.579     10.440      0.000       4.909       7.187
==============================================================================
Omnibus:                       42.505   Durbin-Watson:                   1.951
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               20.975
Skew:                           0.384   Prob(JB):                     2.79e-05
Kurtosis:                       2.182   Cond. No.                         93.3
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                Balance   R-squared:                       0.280
Model:                            OLS   Adj. R-squared:                  0.274
Method:                 Least Squares   F-statistic:                     51.30
Date:                Thu, 29 Dec 2022   Prob (F-statistic):           4.94e-28
Time:                        05:50:06   Log-Likelihood:                -2953.7
No. Observations:                 400   AIC:                             5915.
Df Residuals:                     396   BIC:                             5931.
Df Model:                           3                                         
Covariance Type:            nonrobust                                         
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
Intercept         200.6232     33.698      5.953      0.000     134.373     266.873
Income              6.2182      0.592     10.502      0.000       5.054       7.382
Student2          476.6758    104.351      4.568      0.000     271.524     681.827
Income:Student2    -1.9992      1.731     -1.155      0.249      -5.403       1.404
==============================================================================
Omnibus:                      107.788   Durbin-Watson:                   1.952
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               22.158
```

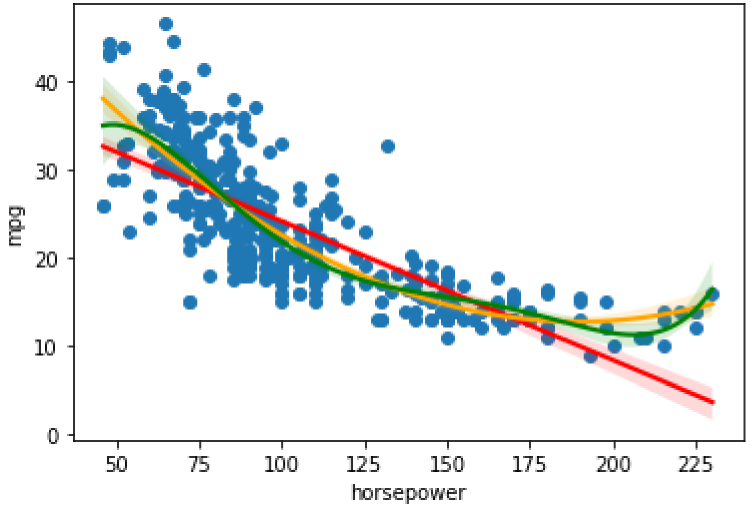
```
Skew:                           0.228   Prob(JB):                     1.54e-05
Kurtosis:                       1.941   Cond. No.                         309.
==============================================================================

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

In [ ]:

```
##Non-linear relationships
plt.scatter(Auto.horsepower, Auto.mpg)
sns.regplot(Auto.horsepower, Auto.mpg,order=1,label='linear',scatter=False,color='red')
sns.regplot(Auto.horsepower, Auto.mpg,order=2,label='order2',scatter=False,color='orange')
sns.regplot(Auto.horsepower, Auto.mpg,order=5,label='order5',scatter=False,color='green')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [ ]:

```
Auto['horsepower2'] = Auto.horsepower**2
Auto.head(2)
```

Out[41]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name	horsepower2
0	18.0	8	307.0	130.0	3504	12.0	70	1	chevrolet chevelle malibu	16900.0
1	15.0	8	350.0	165.0	3693	11.5	70	1	buick skylark 320	27225.0

In [ ]:

```
lm_fit4 = smf.ols('mpg~horsepower+horsepower2',Auto).fit()
lm_fit4.summary().tables[1]
```

Out[42]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	56.9001	1.800	31.604	0.000	53.360	60.440
horsepower	-0.4662	0.031	-14.978	0.000	-0.527	-0.405
horsepower2	0.0012	0.000	10.080	0.000	0.001	0.001

In [ ]:

```
##find out the oder
lr = lm.LinearRegression()
X = Auto.horsepower.values.reshape(-1,1)
Y = Auto.mpg
lr.fit(X,Y)
Auto['mpg_pred'] = lr.predict(X)
Auto['res'] = Auto.mpg-Auto.mpg_pred

lr1 = lm.LinearRegression()
X1 = Auto[['horsepower','horsepower2']]
Y = Auto.mpg
lr1.fit(X1,Y)
Auto['mpg_pred1'] = lr1.predict(X1)
Auto['res1'] = Auto.mpg-Auto.mpg_pred1
```

In [ ]:

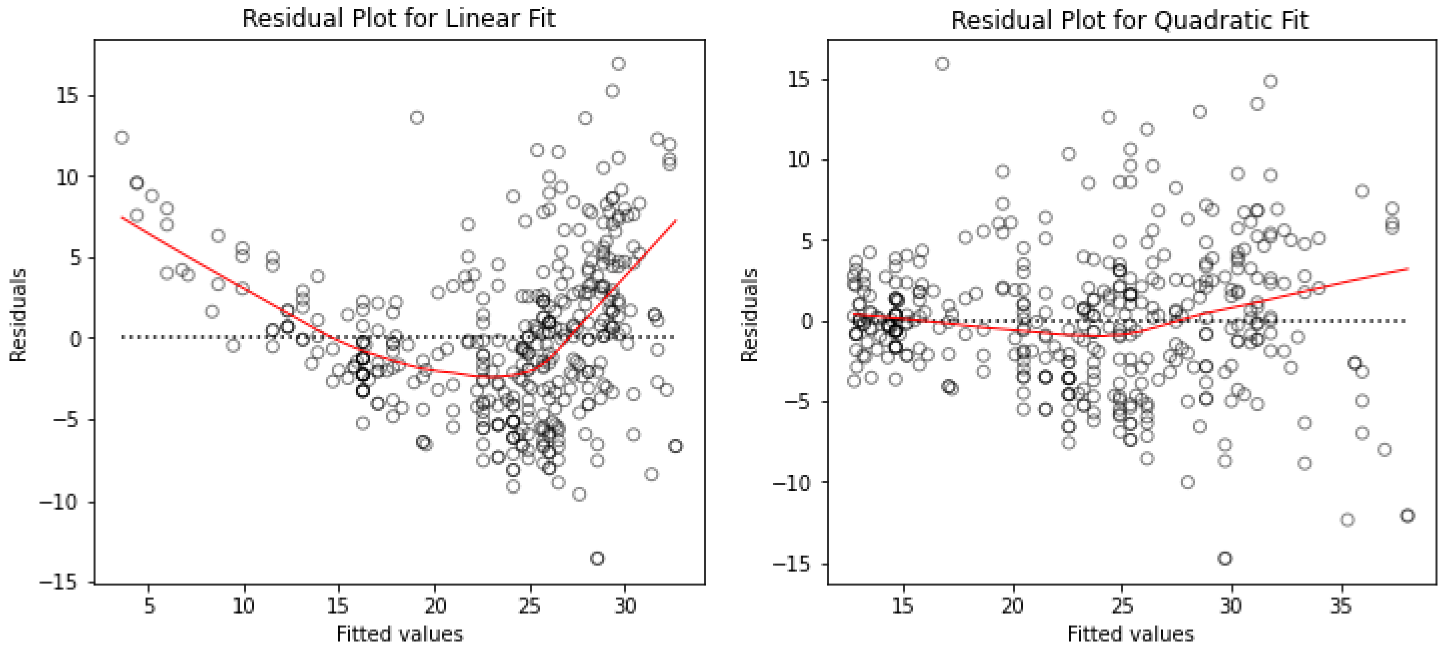
```
fig, (ax1,ax2) = plt.subplots(1,2, figsize=(12,5))

# Left plot
sns.regplot(Auto.mpg_pred, Auto.res,lowess=True,
            ax=ax1, line_kws={'color':'r', 'lw':1},
            scatter_kws={'facecolors':'None', 'edgecolors':'k', 'alpha':0.5})
ax1.hlines(0,xmin=ax1.xaxis.get_data_interval()[0],
           xmax=ax1.xaxis.get_data_interval()[1], linestyles='dotted')
ax1.set_title('Residual Plot for Linear Fit')

# Right plot
sns.regplot(Auto.mpg_pred1, Auto.res1, lowess=True,
            line_kws={'color':'r', 'lw':1}, ax=ax2,
            scatter_kws={'facecolors':'None', 'edgecolors':'k', 'alpha':0.5})
ax2.hlines(0,xmin=ax2.xaxis.get_data_interval()[0],
           xmax=ax2.xaxis.get_data_interval()[1], linestyles='dotted')
ax2.set_title('Residual Plot for Quadratic Fit')

for ax in fig.axes:
    ax.set_xlabel('Fitted values')
    ax.set_ylabel('Residuals')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [ ]:

```
##VIF-Variance Inflation Factor
est_Age = smf.ols('Age ~ Rating + Limit', credit).fit()
est_Rating = smf.ols('Rating ~ Age + Limit', credit).fit()
est_Limit = smf.ols('Limit ~ Age + Rating', credit).fit()

print(1/(1-est_Age.rsquared))
print(1/(1-est_Rating.rsquared))
print(1/(1-est_Limit.rsquared))
```

```
1.0113846860681328
160.66830095856935
160.59287978597942
```

In [ ]:

```
##LAB Assinments
df = pd.read_csv('/content/gdrive/MyDrive/ISLR/Boston.csv')
df.head(2)
```

Out[4]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.9	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6

In [ ]:
```
lm_fit = smf.ols('medv~lstat',df).fit()
print(lm_fit.params)
print()
print(lm_fit.conf_int())
print()
print(lm_fit.summary().tables[1])
```

```
Intercept    34.553841
lstat        -0.950049
dtype: float64

                   0          1
Intercept  33.448457  35.659225
lstat      -1.026148  -0.873951

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

In [ ]:
```
lm_fit.predict(pd.DataFrame({'lstat':[5,10,15]}))
```

Out[15]:
```
0    29.803594
1    25.053347
2    20.303101
dtype: float64
```

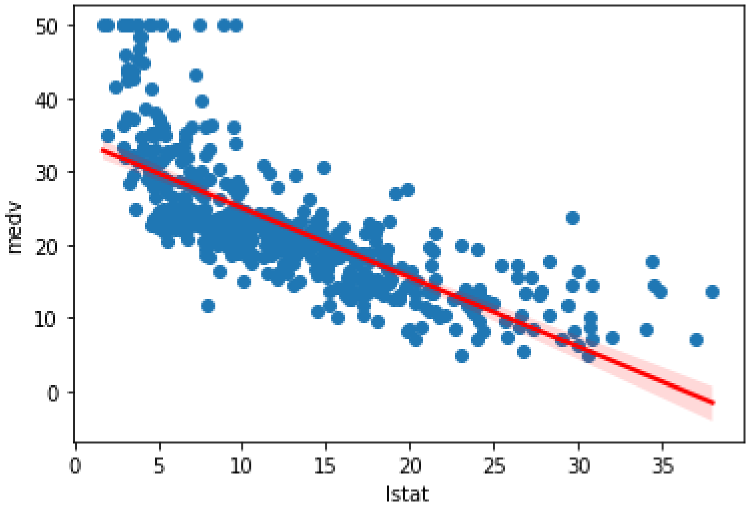
In [ ]:
```
##OR
lr = lm.LinearRegression()
X = df.lstat.values.reshape(-1,1)
y = df.medv
lr.fit(X,y)
x_test = pd.DataFrame([5,10,15])
y_pred = lr.predict(x_test)
print(y_pred)
```

```
[29.80359411 25.05334734 20.30310057]
```

In [ ]:
```
plt.scatter(df.lstat, df.medv)
sns.regplot(df.lstat, df.medv,order=1,label='linear',scatter=False,color='red')
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an ex
plicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8f65c0deb0>



In [ ]:
```
influence = lm_fit.get_influence()
leverage = influence.hat_matrix_diag
np.argmax(leverage)
```

Out[17]: 374

In [ ]:
```
lm_fit = smf.ols('medv~lstat+age',df).fit()
lm_fit.summary().tables[1]
```

Out[7]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	33.2228	0.731	45.458	0.000	31.787	34.659
lstat	-1.0321	0.048	-21.416	0.000	-1.127	-0.937
age	0.0345	0.012	2.826	0.005	0.011	0.059

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  506 non-null    int64  
 1   crim        506 non-null    float64
 2   zn          506 non-null    float64
 3   indus       506 non-null    float64
 4   chas        506 non-null    int64  
 5   nox         506 non-null    float64
 6   rm          506 non-null    float64
 7   age         506 non-null    float64
 8   dis         506 non-null    float64
 9   rad         506 non-null    int64  
 10  tax         506 non-null    int64  
 11  ptratio     506 non-null    float64
 12  black       506 non-null    float64
 13  lstat       506 non-null    float64
 14  medv        506 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 59.4 KB
```

In [ ]:

```
#all_columns = "+".join(Boston.columns.difference(["medv"]))
#my_formula = "medv~" + all_columns
#lm = smf.ols(my_formula, data=Boston).fit()
lm_fit = smf.ols('medv~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+lstat',df).fit()
lm_fit.summary().tables[1]
```

Out[10]:

	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

In [ ]:

```python
print(df.iloc[:,1:])
```

```
        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio   black  lstat  medv  intercept
0       15.3  396.90   4.98  24.0          1
1       17.8  396.90   9.14  21.6          1
2       17.8  392.83   4.03  34.7          1
3       18.7  394.63   2.94  33.4          1
4       18.7  396.90   5.33  36.2          1
..       ...     ...    ...   ...        ...
501     21.0  391.99   9.67  22.4          1
502     21.0  396.90   9.08  20.6          1
503     21.0  396.90   5.64  23.9          1
504     21.0  393.45   6.48  22.0          1
505     21.0  396.90   7.88  11.9          1

[506 rows x 15 columns]
```

In [ ]:

```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
X = add_constant(df)
#X = X.iloc[:,1:]

vif_df = pd.DataFrame()
vif_df["feature"] = X.columns

# calculating VIF for each feature
vif_df["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]

#Print the dataframe with VIF values
print(vif_df)
```

```
      feature         VIF
0        crim    1.767486
1          zn    2.298459
2       indus    3.987181
3        chas    1.071168
4         nox    4.369093
5          rm    1.912532
6         age    3.088232
7         dis    3.954037
8         rad    7.445301
9         tax    9.002158
10    ptratio    1.797060
11      lstat    2.870777
12  intercept  535.526619

/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/tsatools.py:142: FutureWarning: In a future version of pandas all a
rguments of concat except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

In [ ]:

```
lm_fit = smf.ols('medv~lstat+age+lstat*age',df).fit()
lm_fit.summary().tables[1]
```

Out[55]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	36.0885	1.470	24.553	0.000	33.201	38.976
lstat	-1.3921	0.167	-8.313	0.000	-1.721	-1.063
age	-0.0007	0.020	-0.036	0.971	-0.040	0.038
lstat:age	0.0042	0.002	2.244	0.025	0.001	0.008

In [ ]:

```
lm_order1 = smf.ols('medv~ lstat', data=df).fit()
lm_order2 = smf.ols('medv~ lstat+ I(lstat ** 2.0)', data=df).fit()
print(lm_order2.summary().tables[1])
```

```
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
Intercept          42.8620      0.872     49.149      0.000      41.149      44.575
lstat              -2.3328      0.124    -18.843      0.000      -2.576      -2.090
I(lstat ** 2.0)     0.0435      0.004     11.628      0.000       0.036       0.051
===================================================================================
```

In [ ]:

```
import statsmodels.api as sm
table = sm.stats.anova_lm(lm_order1, lm_order2)
print(table)  ##order2 is a superior model
```

```
   df_resid           ssr  df_diff     ss_diff           F        Pr(>F)
0     504.0  19472.381418      0.0         NaN         NaN           NaN
1     503.0  15347.243158      1.0  4125.13826  135.199822  7.630116e-28
```

In [ ]:

```python
##OR
df['lstat2']=df.lstat**2
lm_fit = smf.ols('medv~lstat+lstat2',df).fit()
lm_fit.summary().tables[1]
```

Out[58]:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	42.8620	0.872	49.149	0.000	41.149	44.575
lstat	-2.3328	0.124	-18.843	0.000	-2.576	-2.090
lstat2	0.0435	0.004	11.628	0.000	0.036	0.051

In [ ]:

```python
df = pd.read_csv('/content/gdrive/MyDrive/ISLR/carseats.csv')
df.head(2)
```

Out[23]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	9.50	138	73	11	276	120	Bad	42	17	Yes	Yes
1	11.22	111	48	16	260	83	Good	65	10	Yes	Yes

In [ ]:

```python
lm_fit = smf.ols('Sales ~ Income + Advertising + Price + Age',df).fit()
print(lm_fit.summary().tables[1])
```

```
================================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept      15.1829      0.777     19.542      0.000      13.656      16.710
Income          0.0108      0.004      2.664      0.008       0.003       0.019
Advertising     0.1203      0.017      7.078      0.000       0.087       0.154
Price          -0.0573      0.005    -11.932      0.000      -0.067      -0.048
Age            -0.0486      0.007     -6.956      0.000      -0.062      -0.035
================================================================================
```

In [ ]:
```
ShelveLoc_dummies = pd.get_dummies(df.ShelveLoc,prefix='ShelveLoc').iloc[:,1:]
ShelveLoc_dummies
df_dummy = pd.concat([df, ShelveLoc_dummies], axis=1)
df_dummy.head(2)
```

Out[31]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US	ShelveLoc_Good	ShelveLoc_Medium
0	9.50	138	73	11	276	120	Bad	42	17	Yes	Yes	0	0
1	11.22	111	48	16	260	83	Good	65	10	Yes	Yes	1	0

In [ ]:
```
#https://github.com/qx0731/Sharing_ISL_python/blob/master/Chapter_3_sec_6.1_6.7.ipynb
```

In [ ]:
```
lm_df_dummy = smf.ols('Sales ~ Income + Advertising + Price + Age + ShelveLoc_Good + ShelveLoc_Medium', data = df_dummy).fi
print(lm_df_dummy.summary().tables[1])
```

```
====================================================================================
                       coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------
Intercept           13.4006      0.545     24.575      0.000      12.329      14.473
Income               0.0136      0.003      4.891      0.000       0.008       0.019
Advertising          0.1057      0.012      9.076      0.000       0.083       0.129
Price               -0.0606      0.003    -18.436      0.000      -0.067      -0.054
Age                 -0.0498      0.005    -10.401      0.000      -0.059      -0.040
ShelveLoc_Good       4.8756      0.230     21.175      0.000       4.423       5.328
ShelveLoc_Medium     2.0046      0.189     10.590      0.000       1.632       2.377
====================================================================================
```

In [ ]:
```
##OR
lm_df_w_dummy = smf.ols('Sales ~ Income + Advertising + Price + Age + C(ShelveLoc)', data = df).fit()
print(lm_df_w_dummy.summary().tables[1])
```

```
=====================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept                13.4006      0.545     24.575      0.000      12.329      14.473
C(ShelveLoc)[T.Good]      4.8756      0.230     21.175      0.000       4.423       5.328
C(ShelveLoc)[T.Medium]    2.0046      0.189     10.590      0.000       1.632       2.377
Income                    0.0136      0.003      4.891      0.000       0.008       0.019
Advertising               0.1057      0.012      9.076      0.000       0.083       0.129
Price                    -0.0606      0.003    -18.436      0.000      -0.067      -0.054
Age                      -0.0498      0.005    -10.401      0.000      -0.059      -0.040
=====================================================================================
```

In [ ]:
```
one_hot_encoded_data = pd.get_dummies(credit, columns = ['Married', 'Student','Gender'])
print(one_hot_encoded_data)
```

In [ ]:
```
##1. NA value, replace with mean,median or mode,quantitative then mean, qualitative mode or median
##2. covert X and Y into scale
###3. df.info( ), factor variable and continuous variable
##4. VIF and check which variables i should drop
##5. check linear or non linear
##6. do simple LR model then findout the factor variable whcih is significant
##7. those  factor which is significant do onehotencode
###8. TRY to findout best model where all variables p values less than 0.05, R sq
##and adj R sq high and AIC and BIC low.
## this is only for traning model
```