This homework assignment will build three models on the advertising data and evaluate their performance. You can use tools from sklearn to complete this task.

Source of data: https://www.statlearning.com/s/Advertising.csv (https://www.statlearning.com/s/Advertising.csv)

1. Use train_test_split to split the data into training set (80%) and test set (20%).
2. Build a multilinear regression model with 'TV', 'Radio', and 'newspaper' as input variables and 'sales' as output variable. Name the model model_lr. Train the model on the training set and obtain model predictions on the test set.
3. Build a degree 2 polynomial regression model with 'TV', 'Radio', and 'newspaper' as input variables and 'sales' as output variable. Name the model model_pr2. Train the model on the training set and obtain model predictions on the test set.
4. Build a degree 4 polynomial regression model with 'TV', 'Radio', and 'newspaper' as input variables and 'sales' as output variable. Name the model model_pr4. Train the model on the training set and obtain model predictions on the test set.
5. Calculate the test MSE of each model using the mean_squared_error function. Which model gives the best MSE?

In [4]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [5]:

```
url = "https://www.statlearning.com/s/Advertising.csv"
df = pd.read_csv(url)
df
```

Out[5]:

	Unnamed: 0	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
...	...	...	...	...	...
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

200 rows × 5 columns

In [6]:

```
from sklearn.linear_model import LinearRegression

model_lr = LinearRegression()

model_lr.fit(df[["TV", "radio", "newspaper"]], df[["sales"]])
```

Out[6]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [7]:

```

# Retrieve the estimated parameter values.
print("Theta 0:", model_lr.intercept_)
print("Theta 1 and Theta 2:", model_lr.coef_)
```

```
Theta 0: [2.93888937]
Theta 1 and Theta 2: [[ 0.04576465  0.18853002 -0.00103749]]
```

In [8]:

```
# Remember prediction = theta0 + theta1 * tv + theta2 * radio + theta3 * newspaper
theta0 = 2.93888937
theta1 = 0.04576465
theta2 = 0.18853002
theta3 = -0.00103749

prediction_lr2 = theta0 + theta1 * 230.1 + theta2 * 37.8 + theta3 + 69.2
print(prediction_lr2)
```

```
89.79473260099999
```

In [9]:

```
theta = np.array([2.93888937, 0.04576465, 0.18853002,-0.00103749 ])

list_errors = []

for i in df.index:
    # print(i)
    x = np.array([1, df.loc[i, "TV"], df.loc[i, "radio"], df.loc[i, "newspaper"]])
    theta_dot_x = theta.dot(x)
    y = df.loc[i, "sales"]
    squared_error = (theta_dot_x - y) ** 2
    list_errors.append(squared_error)

print(list_errors)
print("MSE:", np.mean(list_errors))
```

```
[2.4838523325704758, 3.7552830939752964, 9.046086113478205, 0.8139097959042205, 0.08333203742095156, 27.860958
168608885, 0.004933603374869262, 1.1600284997016992, 1.1505975190689908, 3.8058145675968853, 2.457684566107966
3, 0.013195060613760221, 1.896462664531567, 0.7633499967703377, 0.31994006509921036, 2.4986102359223024, 0.104
75418507402931, 1.3807219862032158, 1.8179600996826975, 0.18829200432470147, 0.010154294219916095, 5.020016655
076762, 0.7905887254615187, 1.0939780683274185, 2.413302440620558, 13.032387862989614, 0.00010993382001450475,
1.3263544948459203, 0.26065038807635343, 1.8386698603387743, 0.05472574938323446, 0.3068123121469075, 3.84597
72205703437, 2.144085353128217, 3.7062757671187763, 17.697397968129724, 3.976427308199339, 0.85281239220569 7,
0.0366027990532167, 1.1075212518443787, 0.049432515610382526, 0.038409816448849284, 0.802460282931173, 1.1317
936037321032, 0.1504511316439902, 0.0683948548808095, 2.981191484865793, 2.1826186598555575, 2.14218764644659
83, 2.346515228926929, 1.5158838458676722, 1.8501074290379156, 3.752058644895477, 1.5757762798579174, 0.030426
316794587, 5.795517738463856, 9.167044744785015, 0.18097473162656233, 3.617401595795199, 0.07102861522861192,
5.559428804620496, 1.7143357107711539, 1.1756239718807509, 0.6230061701161455, 1.045022908775505 2, 2.105268482
2552936, 0.23422516150299597, 1.8577471729821111, 0.005864563790593714, 1.4186867169780493, 0.2709213597071414
3, 3.143746043768295 4, 2.455016080234923, 1.2034468314393205, 0.1084470868761784, 9.97499264295656, 5.8680739
19232705, 0.15061935359504983, 12.343372957862123, 1.7548197441082496, 0.12536971675919623, 5.512832297483659,
1.257967501844985, 0.6754253840948398, 0.8438887144070999, 0.0003456243514506398, 0.1610353695870151 7, 0.16501
01639526541, 1.4130749498237813, 0.049397689529928644, 1.4418478580288534, 7.8606427382350885, 0.0593422096745
79115, 0.9456098788537688, 1.036070698396088, 0.34231611864435935, 0.8755683420818117, 0.02654331903865259, 1.
641963322865619, 0.06741755490629413, 4.734800241759029, 0.3108189444046239, 8.088877597674085, 0.003870379976
414872, 0.15911324087558207, 1.5966454777984802, 1.1534279132251803, 2.5328628028648104, 2.9334832779735867,
0.011463487458176299, 1.8492608832253479, 0.4352354038834598, 0.04813166188304018, 0.25379154683663974, 0.4971
6111802823476, 0.10220141605175112, 0.04895236254326881, 8.004498290923511, 0.11149294637230348, 0.04842986872
674436, 1.1878669410903937, 0.7023726020151843, 4.106532730971415, 0.013737808796143388, 0.06031227071871106,
2.1686977426965064, 15.817916398919726, 4.841454784501653, 5.975615563299165, 3.297467638042237, 77.9280622544
3407, 8.281730180913012, 7.55782706958355, 0.11093941460282543, 1.0749633286080065, 5.766790352074783, 3.81610
9634612358, 0.0026269212622497787, 0.028368089882894392, 1.049646762815287, 1.9891113834146785, 0.641552077454
6323, 0.7227201871733621, 2.67386851586316, 1.712605037550582, 0.3498256496328107, 4.385777588327136, 4.575909
573333429, 1.858749794286466, 0.07438086362147399, 5.140226551537065, 2.5295680215514778, 0.05759936736173859
6, 0.602327584066254, 0.009679036223060235, 4.441804828887928, 0.007199968817916652, 0.007342440370417233, 9.5
1228325570708, 0.25085173521972387, 0.03465781157656008, 0.06715682242563885, 0.0021886535801701372, 0.4203746
38121672, 0.6917328776698606, 5.399687237617183, 7.97721035278199, 1.353145416286896, 0.007420793560064724, 8.
652761355325481, 1.0100646255286443, 0.020086064344948543, 5.917581782672063e-05, 0.07340832103608977, 5.03714
4982259188, 4.897936203896179, 0.04867583761227635, 0.2134877148300459, 17.73250885547805, 0.0464162093460776
6, 0.007603917608197815, 2.9863305673674807, 4.602439259971866, 4.272068218768789, 0.8808240437777741, 3.22916
6885645315, 0.3704257868120665, 0.049977624494584504, 7.531225493242362, 0.4203910470537395, 2.853259563066329
6, 2.1785177839969174, 2.0556971355201448, 1.2388605659760128, 0.6475400830797612, 4.971371621931512, 2.355264
712831983, 0.00019819937558890704, 3.0021691982093657, 3.144226338292858]
MSE: 2.7841263145117954
```

In [10]:

```
X = np.hstack([np.ones([200, 1]), df[["TV", "radio", "newspaper"]].values])
print(X)
```

```
[[  1.  230.1  37.8  69.2]
 [  1.   44.5  39.3  45.1]
 [  1.   17.2  45.9  69.3]
 [  1.  151.5  41.3  58.5]
 [  1.  180.8  10.8  58.4]
 [  1.    8.7  48.9  75. ]
 [  1.   57.5  32.8  23.5]
 [  1.  120.2  19.6  11.6]
 [  1.    8.6   2.1   1. ]
 [  1.  199.8   2.6  21.2]
 [  1.   66.1   5.8  24.2]
 [  1.  214.7  24.    4. ]
 [  1.   23.8  35.1  65.9]
 [  1.   97.5   7.6   7.2]
 [  1.  204.1  32.9  46. ]
 [  1.  195.4  47.7  52.9]
 [  1.   67.8  36.6 114. ]
 [  1.  281.4  39.6  55.8]
 [  1.   69.2  20.5  18.3]
```

In [12]:

```
y = df[["sales"]].values
print(y)
```

```
 [12.8]
 [25.4]
 [14.7]
 [10.1]

 [21.5]
 [16.6]
 [17.1]
 [20.7]
 [12.9]
 [ 8.5]
 [14.9]
 [10.6]
 [23.2]
 [14.8]
 [ 9.7]
 [11.4]
 [10.7]
 [22.6]
 [21.2]
```

In [13]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

model_lr2 = LinearRegression()
model_lr2.fit(X, y)
predictions_lr2 = model_lr2.predict(X)
MSE_lr2 = mean_squared_error(y, predictions_lr2)
print("MSE of the linear regression model:", MSE_lr2)
```

```
MSE of the linear regression model: 2.784126314510936
```

In [ ]:

In [14]:

```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=2, include_bias=False)
poly_features.fit(X)
X_poly = poly_features.transform(X)

print(X_poly[:5])
```

```
[[1.000000e+00 2.301000e+02 3.780000e+01 6.920000e+01 1.000000e+00
  2.301000e+02 3.780000e+01 6.920000e+01 5.294601e+04 8.697780e+03
  1.592292e+04 1.428840e+03 2.615760e+03 4.788640e+03]
 [1.000000e+00 4.450000e+01 3.930000e+01 4.510000e+01 1.000000e+00
  4.450000e+01 3.930000e+01 4.510000e+01 1.980250e+03 1.748850e+03
  2.006950e+03 1.544490e+03 1.772430e+03 2.034010e+03]
 [1.000000e+00 1.720000e+01 4.590000e+01 6.930000e+01 1.000000e+00
  1.720000e+01 4.590000e+01 6.930000e+01 2.958400e+02 7.894800e+02
  1.191960e+03 2.106810e+03 3.180870e+03 4.802490e+03]
 [1.000000e+00 1.515000e+02 4.130000e+01 5.850000e+01 1.000000e+00
  1.515000e+02 4.130000e+01 5.850000e+01 2.295225e+04 6.256950e+03
  8.862750e+03 1.705690e+03 2.416050e+03 3.422250e+03]
 [1.000000e+00 1.808000e+02 1.080000e+01 5.840000e+01 1.000000e+00
  1.808000e+02 1.080000e+01 5.840000e+01 3.268864e+04 1.952640e+03
  1.055872e+04 1.166400e+02 6.307200e+02 3.410560e+03]]
```

In [16]:

```
from sklearn.linear_model import LinearRegression
model_pr2 = LinearRegression()
model_pr2.fit(X_poly, y)
print(model_pr2.coef_, model_pr2.intercept_)
```

```
[[ 0.00000000e+00  2.58262743e-02  1.05371485e-02  3.44186766e-03
  -2.60208521e-18  2.58262743e-02  1.05371485e-02  3.44186766e-03
  -1.09702663e-04  1.10525949e-03 -4.55155391e-05  1.11997015e-04
   8.26605896e-05  1.19125650e-05]] [5.08478167]
```

In [17]:

```
url2 = "https://www.statlearning.com/s/Advertising.csv"
df2 = pd.read_csv(url2)
df2
```

Out[17]:

	Unnamed: 0	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
...	...	...	...	...	...
195	196	38.2	3.7	13.8	7.6
196	197	94.2	4.9	8.1	9.7
197	198	177.0	9.3	6.4	12.8
198	199	283.6	42.0	66.2	25.5
199	200	232.1	8.6	8.7	13.4

200 rows × 5 columns

In [18]:

```
from sklearn.linear_model import LinearRegression

model_lr4 = LinearRegression()

model_lr4.fit(df2[["TV", "radio", "newspaper"]], df2[["sales"]])
```

Out[18]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [19]:

```
print("Theta 0:", model_lr4.intercept_)
print("Theta 1 and Theta 2:", model_lr4.coef_)
```

```
Theta 0: [2.93888937]
Theta 1 and Theta 2: [[ 0.04576465  0.18853002 -0.00103749]]
```

In [20]:

```
# Remember prediction = theta0 + theta1 * tv + theta2 * radio + theta3 * newspaper
theta0 = 2.93888937
theta1 = 0.04576465
theta2 = 0.18853002
theta3 = -0.00103749

prediction_lr4 = theta0 + theta1 * 230.1 + theta2 * 37.8 + theta3 + 69.2
print(prediction_lr4)
```

```
89.79473260099999
```

In [22]:
```
theta = np.array([2.93888937, 0.04576465, 0.18853002,-0.00103749 ])

list_errors = []

for i in df.index:
    # print(i)
    x = np.array([1, df2.loc[i, "TV"], df2.loc[i, "radio"], df2.loc[i, "newspaper"]])
    theta_dot_x = theta.dot(x)
    y = df2.loc[i, "sales"]
    squared_error = (theta_dot_x - y) ** 2
    list_errors.append(squared_error)

print(list_errors)
print("MSE:", np.mean(list_errors))
```

```
[2.4838523325704758, 3.7552830939752964, 9.046086113478205, 0.8139097959042205, 0.08333203742095156, 27.860958
168608885, 0.004933603374869262, 1.1600284997016992, 1.1505975190689908, 3.8058145675968853, 2.457684566107966
3, 0.013195060613760221, 1.896462664531567, 0.7633499967703377, 0.31994006509921036, 2.4986102359223024, 0.104
75418507402931, 1.3807219862032158, 1.8179600996826975, 0.18829200432470147, 0.010154294219916095, 5.020016655
076762, 0.7905887254615187, 1.0939780683274185, 2.413302440620558, 13.032387862989614, 0.00010993382001450475,
1.3263544948459203, 0.26065038807635343, 1.8386698603387743, 0.05472574938323446, 0.30681231214690075, 3.84597
72205703437, 2.144085353128217, 3.7062757671187763, 17.697397968129724, 3.976427308199339, 0.852812392205697,
0.0366027990532167, 1.1075212518443787, 0.049432515610382526, 0.038409816448849284, 0.8024660282931173, 1.1317
936037321032, 0.15045111316439902, 0.0683948548808095, 2.981191484865793, 2.1826186598555575, 2.14218764644659
83, 2.346515228926929, 1.5158838458676722, 1.8501074290379156, 3.752058644895477, 1.5757762798579174, 0.030426
316794587, 5.795517738463856, 9.167044744785015, 0.18097473162656233, 3.617401595795199, 0.07102861522861192,
5.559428804620496, 1.7143357107711539, 1.1756239718807509, 0.6230061701161455, 1.0450229087755052, 2.105268482
2552936, 0.23422516150299597, 1.8577471729821111, 0.005864563790593714, 1.4186867169780493, 0.2709213597071414
3, 3.1437460437682954, 2.455016080234923, 1.2034468314393205, 0.1084470868761784, 9.97499264295656, 5.8680739
19232705, 0.15061935359504983, 12.343372957862123, 1.7548197441082496, 0.12536971675919623, 5.512832297483659,
1.257967501844985, 0.6754253840948398, 0.8438887144070999, 0.0003456243514506398, 0.16103536958701517, 0.16501
01639526541, 1.4130749498237813, 0.049397689529928644, 1.4418478580288534, 7.8606427382350885, 0.0593422096745
79115, 0.9456098788537688, 1.036070698396088, 0.34231611864435935, 0.8755683420818117, 0.02654331903865259, 1.
641963322865619, 0.06741755490629413, 4.734800241759029, 0.3108189444046239, 8.088877597674085, 0.003870379976
414872, 0.15911324087558207, 1.5966454777984802, 1.1534279132251803, 2.5328628028648104, 2.9334832779735867,
0.011463487458176299, 1.8492608832253479, 0.4352354038834598, 0.04813166188304018, 0.25379154683663974, 0.4971
6111802823476, 0.10220141605175112, 0.04895236254326881, 8.004498290923511, 0.11149294637230348, 0.04842986872
674436, 1.1878669410903937, 0.7023726020151843, 4.106532730971415, 0.013737808796143388, 0.06031227071871106,
2.1686977426965064, 15.817916398919726, 4.841454784501653, 5.975615563299165, 3.297467638042237, 77.9280622544
3407, 8.281730180913012, 7.55782706958355, 0.11093941460282543, 1.0749633286080065, 5.766790352074783, 3.81610
9634612358, 0.0026269212622497787, 0.028368089882894392, 1.049646762815287, 1.9891113834146785, 0.641552077454
6323, 0.7227201871733621, 2.67386851586316, 1.712605037550582, 0.3498256496328107, 4.385777588327136, 4.575909
573333429, 1.858749794286466, 0.07438086362147399, 5.140226551537065, 2.5295680215514778, 0.05759936736173859
6, 0.602327584066254, 0.009679036223060235, 4.441804828887928, 0.007199968817916652, 0.007342440370417233, 9.5
1228325570708, 0.25085173521972387, 0.03465781157656008, 0.06715682242563885, 0.0021886535801701372, 0.4203746
38121672, 0.6917328776698606, 5.399687237617183, 7.97721035278199, 1.353145416286896, 0.007420793560064724, 8.
652761355325481, 1.0100646255286443, 0.020086064344948543, 5.917581782672063e-05, 0.07340832103608977, 5.03714
4982259188, 4.897936203896179, 0.04867583761227635, 0.2134877148300459, 17.73250885547805, 0.0464162093460776
6, 0.007603917608197815, 2.9863305673674807, 4.602439259971866, 4.272068218768789, 0.8808240437777741, 3.22916
6885645315, 0.3704257868120665, 0.049977624494584504, 7.531225493242362, 0.4203910470537395, 2.853259563066329
6, 2.1785177839969174, 2.0556971355201448, 1.2388605659760128, 0.6475400830797612, 4.971371621931512, 2.355264
712831983, 0.00019819937558890704, 3.0021691982093657, 3.144226338292858]
MSE: 2.7841263145117954
```

In [23]:

```python
X = np.hstack([np.ones([200, 1]), df2[["TV", "radio", "newspaper"]].values])
print(X)
```

```
[[  1.   230.1  37.8  69.2]
 [  1.    44.5  39.3  45.1]
 [  1.    17.2  45.9  69.3]
 [  1.   151.5  41.3  58.5]
 [  1.   180.8  10.8  58.4]
 [  1.     8.7  48.9  75. ]
 [  1.    57.5  32.8  23.5]
 [  1.   120.2  19.6  11.6]
 [  1.     8.6   2.1   1. ]
 [  1.   199.8   2.6  21.2]
 [  1.    66.1   5.8  24.2]
 [  1.   214.7  24.    4. ]
 [  1.    23.8  35.1  65.9]
 [  1.    97.5   7.6   7.2]
 [  1.   204.1  32.9  46. ]
 [  1.   195.4  47.7  52.9]
 [  1.    67.8  36.6 114. ]
 [  1.   281.4  39.6  55.8]
 [  1.    69.2  20.5  18.3]
```

In [24]:

```
y = df2[["sales"]].values
print(y)
```

```
[[22.1]
 [10.4]
 [ 9.3]
 [18.5]
 [12.9]
 [ 7.2]
 [11.8]
 [13.2]
 [ 4.8]
 [10.6]
 [ 8.6]
 [17.4]
 [ 9.2]
 [ 9.7]
 [19. ]
 [22.4]
 [12.5]
 [24.4]
 [11.3]
 [14.6]
```

In [25]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

model_lr4 = LinearRegression()
model_lr4.fit(X, y)
predictions_lr4 = model_lr4.predict(X)
MSE_lr4 = mean_squared_error(y, predictions_lr4)
print("MSE of the linear regression model:", MSE_lr4)
```

```
MSE of the linear regression model: 2.784126314510936
```

In [28]:

```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=10, include_bias=False)
poly_features.fit(X)
X_poly = poly_features.transform(X)

print(X_poly[:5])
```

```
[[1.00000000e+00 2.30100000e+02 3.78000000e+01 ... 7.51332811e+17
  1.37545583e+18 2.51803026e+18]
 [1.00000000e+00 4.45000000e+01 3.93000000e+01 ... 2.64361030e+16
  3.03376144e+16 3.48149214e+16]
 [1.00000000e+00 1.72000000e+01 4.59000000e+01 ... 1.12070484e+18
  1.69204456e+18 2.55465552e+18]
 [1.00000000e+00 1.51500000e+02 4.13000000e+01 ... 2.33962919e+17
  3.31400261e+17 4.69416835e+17]
 [1.00000000e+00 1.80800000e+02 1.08000000e+01 ... 1.57815730e+16
  8.53373946e+16 4.61454060e+17]]
```

In [29]:

```
from sklearn.linear_model import LinearRegression
model_pr4 = LinearRegression()
model_pr4.fit(X_poly, y)
print(model_pr4.coef_, model_pr4.intercept_)
```

```
  7.97497147e-14  6.39414790e-14  1.22055006e-12  6.93201630e-13
 -1.96884950e-13 -2.79734165e-14 -1.69658957e-13 -1.30883447e-13
 -3.27597769e-12 -3.79036948e-12 -3.10652350e-12  7.37684006e-12
 -4.38559188e-12  1.47506811e-12 -4.66879702e-14  5.97625621e-12
  4.39304022e-12  2.54520860e-11 -1.56218613e-11 -1.76701169e-11
  2.17446308e-11 -8.73072153e-12  1.25895482e-12 -4.83943490e-12
 -1.17314170e-12 -1.95575034e-11 -4.53769549e-11  8.13199909e-11
 -3.64372503e-11 -9.22802105e-12  1.10635740e-11 -2.18692614e-12
  1.45100702e-12 -4.27440912e-12  2.76003144e-11 -1.61960911e-11
  6.34560882e-12 -1.56523387e-11  2.59034571e-11 -1.17229246e-11
  1.70843826e-12 -2.89672519e-14  1.71342838e-20 -3.95837257e-19
  2.62963809e-19  9.05572743e-18  2.16503785e-18 -6.17157463e-18
 -1.11680602e-16 -8.25573121e-18 -5.93689100e-18  2.89071341e-17
  5.92364208e-16 -3.55612537e-16  4.52818348e-17  3.45464208e-16
  5.97979711e-18  1.03249855e-15  5.12707091e-16  8.34053658e-15
 -6.47298194e-15 -1.68452067e-15 -4.87000099e-16 -2.67969313e-14
  6.26313161e-15 -5.00308510e-14 -1.76119461e-14  5.47631632e-14
 -7.94088242e-15  3.83130029e-15  1.17907034e-13  4.14203063e-14
  1.89085813e-14  3.90517762e-13 -3.65303329e-13  2.06232504e-14
  9.61979351e-15 -1.00712173e-14 -2.07520601e-13 -3.54445709e-13
```

In [ ]: