# Linear Regression and Polynomial Regression

## *advertising.csv* dataset

In [1]:
```
#This homework assignment will build three models on the advertising data and evaluate their performanc
# You can use tools from sklearn to complete this task.
```

In [2]:
```
# Source of data: https://www.statlearning.com/s/Advertising.csv
```

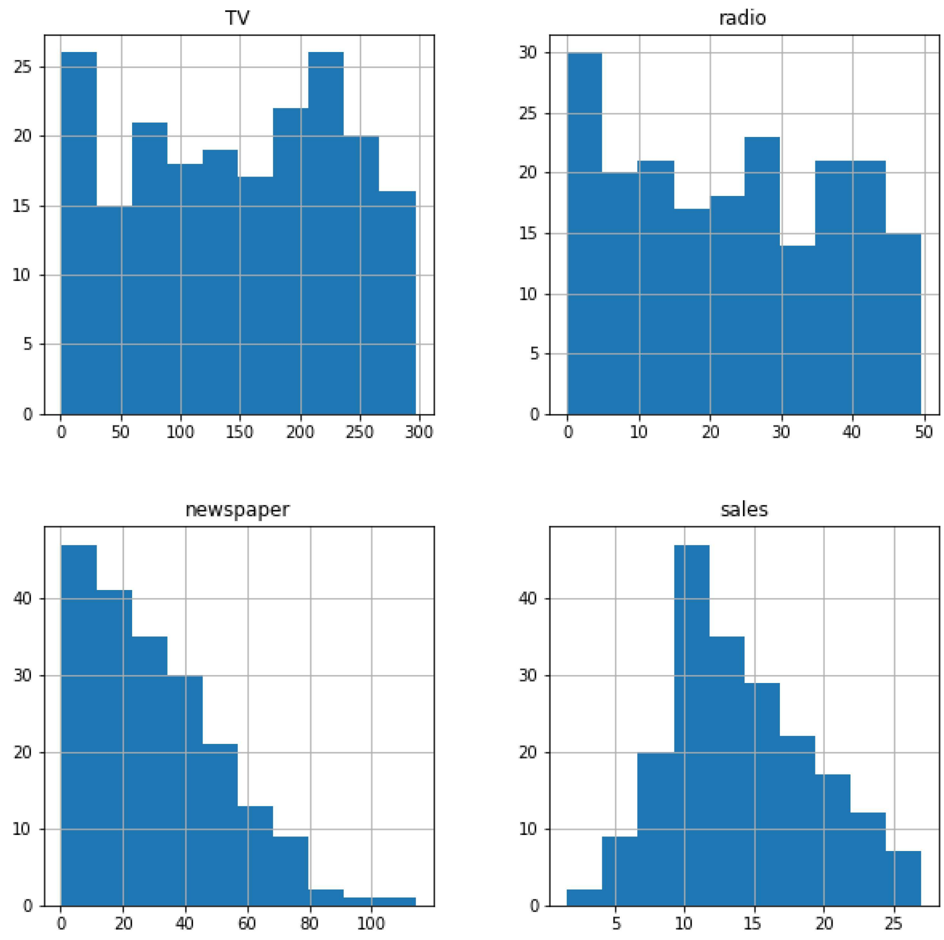
In [3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [7]:
```
advertising = pd.read_csv("Advertising.csv")
advertising.head()
```

| | 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 |

In [3]:

```
fig = advertising.hist(figsize=(10, 10))
```




In [4]:

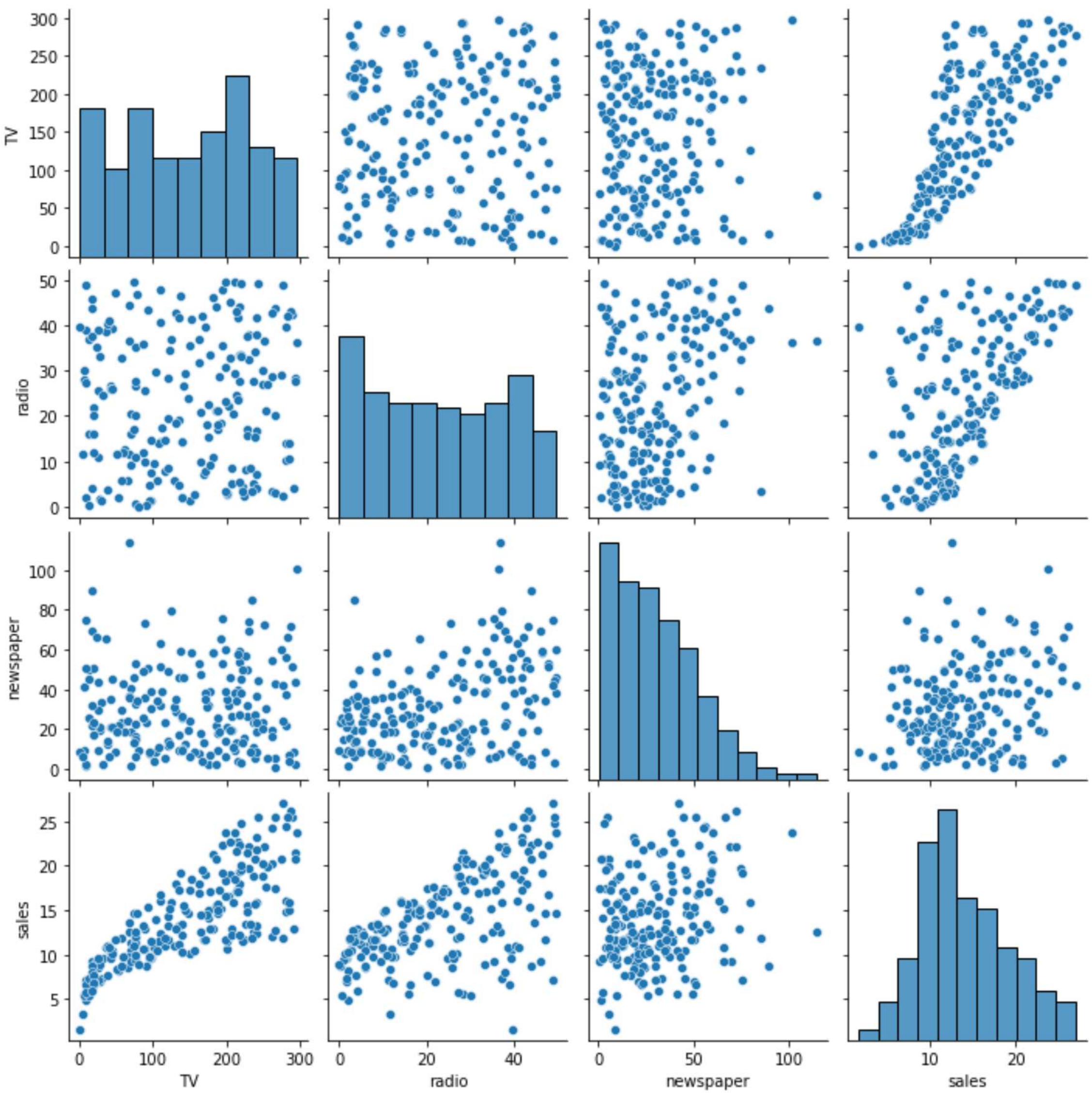
```
advertising.describe()
```

| | TV | radio | newspaper | sales |
|---|---|---|---|---|
| **count** | 200.000000 | 200.000000 | 200.000000 | 200.000000 |
| **mean** | 147.042500 | 23.264000 | 30.554000 | 14.022500 |
| **std** | 85.854236 | 14.846809 | 21.778621 | 5.217457 |
| **min** | 0.700000 | 0.000000 | 0.300000 | 1.600000 |
| **25%** | 74.375000 | 9.975000 | 12.750000 | 10.375000 |
| **50%** | 149.750000 | 22.900000 | 25.750000 | 12.900000 |
| **75%** | 218.825000 | 36.525000 | 45.100000 | 17.400000 |
| **max** | 296.400000 | 49.600000 | 114.000000 | 27.000000 |

In [5]:

```
sns.pairplot(advertising)
```

<seaborn.axisgrid.PairGrid at 0x260b78ea670>



In [6]:

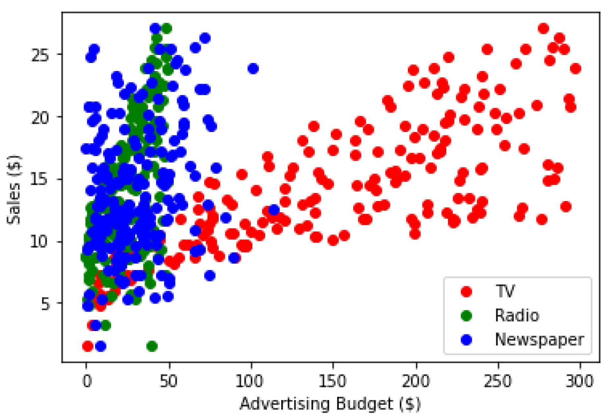
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset into a pandas dataframe
url = "https://www.statlearning.com/s/Advertising.csv"
data = pd.read_csv(url)

# Create a scatter plot of TV, radio, and newspaper against sales
plt.scatter(data["TV"], data["sales"], color="red", label="TV")
plt.scatter(data["radio"], data["sales"], color="green", label="Radio")
plt.scatter(data["newspaper"], data["sales"], color="blue", label="Newspaper")

# Add axis labels and legend
plt.xlabel("Advertising Budget ($)")
plt.ylabel("Sales ($)")
plt.legend()

# Show the plot
plt.show()
```




In [7]:

```
# 1. Use train_test_split to split the data into training set (80%) and test set (20%).
```

In [8]:

```
from sklearn.model_selection import train_test_split
training_data, test_data = train_test_split(advertising, test_size=0.2)
```

In [9]:

```
# 2. Build a multilinear regression model with 'TV', 'Radio', and 'newspaper' as input variables and 's
#    Name the model model_lr. Train the model on the training set and obtain model predictions on the t
```

In [10]:

```
from sklearn.linear_model import LinearRegression

input_cols = ["TV", "radio", "newspaper"]

model_lr = LinearRegression()

model_lr.fit(training_data[input_cols], training_data[["sales"]])
```

```
LinearRegression()
```

In [11]:

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

```
Theta 0: [3.10846438]
Theta 1 and Theta 2: [[ 0.04488643  0.19149168 -0.00338586]]
```

In [12]:

```
# Apply the model to provide prediction for Fred
test_data['prediction'] = model_lr.predict(test_data[input_cols])
test_data.head()
```

| | TV | radio | newspaper | sales | prediction |
|---|---|---|---|---|---|
| **25** | 62.3 | 12.6 | 18.3 | 9.7 | 8.255723 |
| **169** | 215.4 | 23.6 | 57.6 | 17.1 | 17.101180 |
| **42** | 177.0 | 33.4 | 38.7 | 17.1 | 17.318152 |
| **88** | 110.7 | 40.6 | 63.2 | 16.0 | 15.637968 |
| **44** | 206.9 | 8.4 | 26.4 | 12.9 | 13.914611 |

In [13]:

```
# 3. Build a degree 2 polynomial regression model with 'TV', 'Radio', and 'newspaper' as input variable
#    variable. Name the model model_pr2. Train the model on the training set and obtain model predicti
```

In [14]:

```
# use .to_numpy() function in order to plot pandas data
```

In [15]:

```
input_cols
```

```
['TV', 'radio', 'newspaper']
```

In [16]:

```
training_data
```

| | TV | radio | newspaper | sales |
|---|---|---|---|---|
| **166** | 234.5 | 3.4 | 84.8 | 11.9 |
| **81** | 76.4 | 26.7 | 22.3 | 11.8 |
| **13** | 23.8 | 35.1 | 65.9 | 9.2 |
| **61** | 53.5 | 2.0 | 21.4 | 8.1 |
| **46** | 175.1 | 22.5 | 31.5 | 14.9 |
| **...** | ... | ... | ... | ... |
| **120** | 19.4 | 16.0 | 22.3 | 6.6 |
| **157** | 93.9 | 43.5 | 50.5 | 15.3 |
| **167** | 17.9 | 37.6 | 21.6 | 8.0 |
| **191** | 39.5 | 41.1 | 5.8 | 10.8 |
| **35** | 95.7 | 1.4 | 7.4 | 9.5 |

160 rows × 4 columns

In [17]:

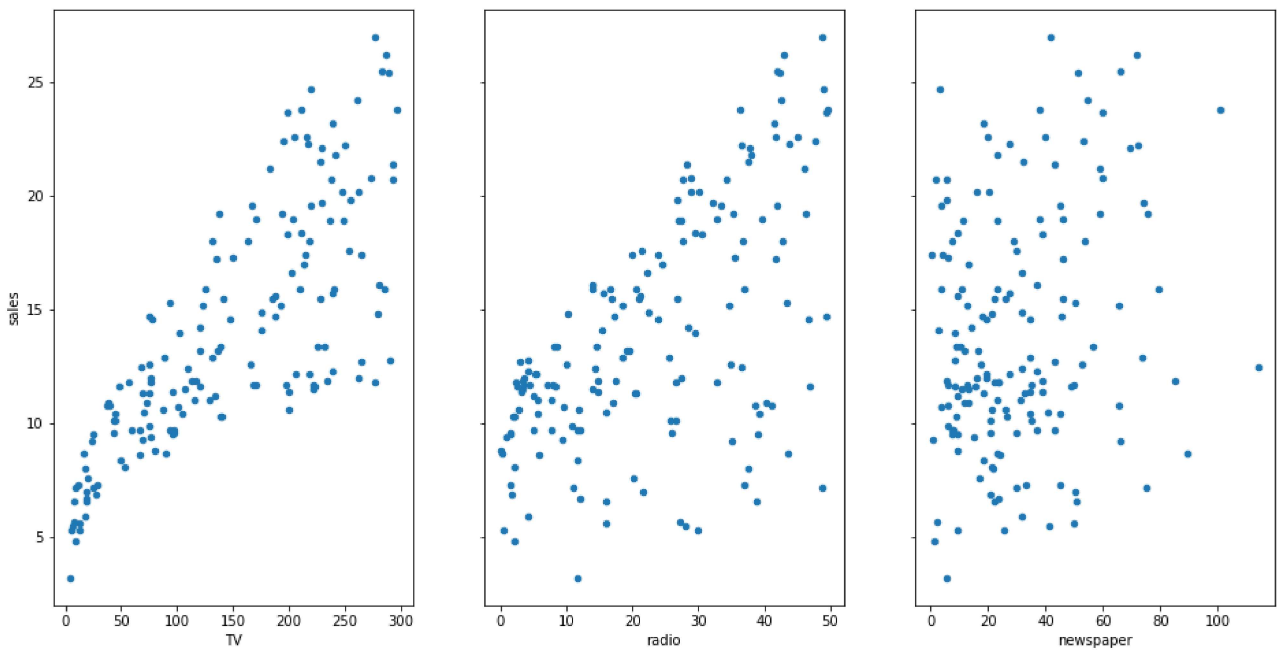
```
training_data[input_cols]
```

| | TV | radio | newspaper |
|---|---|---|---|
| **166** | 234.5 | 3.4 | 84.8 |
| **81** | 76.4 | 26.7 | 22.3 |
| **13** | 23.8 | 35.1 | 65.9 |
| **61** | 53.5 | 2.0 | 21.4 |
| **46** | 175.1 | 22.5 | 31.5 |
| **...** | ... | ... | ... |
| **120** | 19.4 | 16.0 | 22.3 |
| **157** | 93.9 | 43.5 | 50.5 |
| **167** | 17.9 | 37.6 | 21.6 |
| **191** | 39.5 | 41.1 | 5.8 |
| **35** | 95.7 | 1.4 | 7.4 |

160 rows × 3 columns

In [18]:
```python
fig,axs= plt.subplots(1,3,sharey=True) # sharey : share same y axis across the plot
training_data.plot(kind="scatter",x='TV',y='sales',ax=axs[0],figsize=(16,8))
training_data.plot(kind="scatter",x='radio',y='sales',ax=axs[1],figsize=(16,8))
training_data.plot(kind="scatter",x='newspaper',y='sales',ax=axs[2],figsize=(16,8))
```

```
<AxesSubplot:xlabel='newspaper', ylabel='sales'>
```




In [19]:
```python
training_data['sales']
```

```
166    11.9
81     11.8
13      9.2
61      8.1
46     14.9
       ...
120     6.6
157    15.3
167     8.0
191    10.8
35      9.5
Name: sales, Length: 160, dtype: float64
```

In [20]:

```
training_data
```

| | TV | radio | newspaper | sales |
|---|---|---|---|---|
| **166** | 234.5 | 3.4 | 84.8 | 11.9 |
| **81** | 76.4 | 26.7 | 22.3 | 11.8 |
| **13** | 23.8 | 35.1 | 65.9 | 9.2 |
| **61** | 53.5 | 2.0 | 21.4 | 8.1 |
| **46** | 175.1 | 22.5 | 31.5 | 14.9 |
| **...** | ... | ... | ... | ... |
| **120** | 19.4 | 16.0 | 22.3 | 6.6 |
| **157** | 93.9 | 43.5 | 50.5 | 15.3 |
| **167** | 17.9 | 37.6 | 21.6 | 8.0 |
| **191** | 39.5 | 41.1 | 5.8 | 10.8 |
| **35** | 95.7 | 1.4 | 7.4 | 9.5 |

160 rows × 4 columns

In [21]:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
```

In [22]:

```
poly_features1 = PolynomialFeatures(degree=2, include_bias=False)
poly_features1.fit(training_data[input_cols])
X_poly = poly_features1.transform(training_data[input_cols])
model_pr2 = LinearRegression()
model_pr2.fit(X_poly, training_data[["sales"]])
print(model_pr2.coef_, model_pr2.intercept_)
```

```
[[ 5.24158201e-02  3.36901470e-02  4.44480335e-03 -1.11856878e-04
   1.05004811e-03 -9.70745288e-06  1.78961711e-04 -6.72082521e-05
   1.42266115e-05]] [4.95708809]
```

In [23]:

```
test_data['prediction'] = model_pr2.predict(poly_features1.transform(test_data[input_cols]))
test_data.head()
```

| | TV | radio | newspaper | sales | prediction |
|---|---|---|---|---|---|
| **25** | 62.3 | 12.6 | 18.3 | 9.7 | 9.125159 |
| **169** | 215.4 | 23.6 | 57.6 | 17.1 | 17.381653 |
| **42** | 177.0 | 33.4 | 38.7 | 17.1 | 18.302846 |
| **88** | 110.7 | 40.6 | 63.2 | 16.0 | 15.868310 |
| **44** | 206.9 | 8.4 | 26.4 | 12.9 | 13.193492 |

In [24]:

```
# 4. Build a degree 10 polynomial regression model with 'TV', 'Radio', and 'newspaper' as input variabl
#    variable. Name the model model_pr10. Train the model on the training set and obtain model predict
```

In [1]:

```
poly_features2 = PolynomialFeatures(degree=10, include_bias=False)
poly_features2.fit(training_data[input_cols])
X_poly = poly_features2.fit_transform(training_data[input_cols])
model_pr10 = LinearRegression()
model_pr10.fit(X_poly, training_data[["sales"]])
print(model_pr10.coef_, model_pr10.intercept_)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [1], in <cell line: 1>()
----> 1 poly_features2 = PolynomialFeatures(degree=10, include_bias=False)
      2 poly_features2.fit(training_data[input_cols])
      3 X_poly = poly_features2.fit_transform(training_data[input_cols])

NameError: name 'PolynomialFeatures' is not defined
```

In [26]:

```
test_data['prediction'] = model_pr10.predict(poly_features2.transform(test_data[input_cols]))
test_data.head()
```

| | TV | radio | newspaper | sales | prediction |
|---|---|---|---|---|---|
| **25** | 62.3 | 12.6 | 18.3 | 9.7 | 7.413632 |
| **169** | 215.4 | 23.6 | 57.6 | 17.1 | -747.235038 |
| **42** | 177.0 | 33.4 | 38.7 | 17.1 | 11.604793 |
| **88** | 110.7 | 40.6 | 63.2 | 16.0 | 338.580949 |
| **44** | 206.9 | 8.4 | 26.4 | 12.9 | 94.519694 |

In [27]:

```
# 5. Calculate the test MSE of each model using the mean_squared_error function. Which model gives the
```

In [28]:

```python
theta = np.array([3.0516, 0.0443, 0.1876, 0.0037])

list_errors = []

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

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

```
[2.273008522500005, 4.6764062499999985, 11.429876256100002, 0.5969416643999997, 0.16257024, 32.35505042249999, 0.001527246399999878
80653955999992, 0.940434457599999, 3.4929367236000077, 2.0809505024999995, 0.10239360009999815, 3.0085943209000043, 0.7686905624999
186941209000015, 2.396087284900003, 0.7108176099999995, 1.5549092416000034, 1.6111986489, 0.21967969000000012, 0.014597472400000465
093136900002, 1.4465353984000007, 0.8698733289000031, 2.122965561599998, 11.743095312399996, 0.005569636899999864, 1.01139226239999
1705293690000307, 1.3648314276000002, 0.00921408010000009, 0.19348561690000077, 3.4226480016, 1.3707492241000088, 3.68206207690000
46629248399988, 5.241123422499995, 1.204155075600002, 7.83999999990377e-08, 1.3388341264000012, 0.0768675624999926, 0.040694992899
0.31501278760000084, 0.981664824100001, 0.41610240359999884, 0.0605553663999996, 2.513366329599999, 2.7788223203999958, 2.147016172
1.8323683225, 1.4697197824000046, 1.9221049599999953, 3.9698171536000078, 1.3804135081000013, 0.002408846400000001, 5.5116483361, 1
8072900003, 0.20376195999999963, 3.8632295600999935, 0.19871980839999925, 4.946042560899998, 1.8493280099999991, 0.9242899600000022
8175625000001, 1.0089398916000034, 2.084240816100003, 0.18477102250000005, 2.0148950809000015, 0.017392334399999706, 1.710550094400
2965020304000023, 2.8368328041000024, 2.8935711024999953, 1.0626517225000018, 0.02844282249999984, 13.170874888900014, 5.0778115600
0.1690114321000006, 13.212061825599998, 1.6374017520999997, 0.07422900249999956, 5.200360984900004, 0.968413446400001, 0.9247514896
0.9969822801000007, 0.011085984100000022, 0.12257701210000056, 0.037632120099999784, 0.7767425689000003, 0.13927824000000047, 1.550
6, 6.645671526399998, 0.040557932099999985, 0.8438627044000012, 1.0482688225000025, 0.24041370239999704, 0.6142013641000016, 0.0558
999944, 1.9314606528999962, 0.027652364100000017, 4.817410419600009, 0.18893800890000356, 6.961154559999986, 0.000990990400000011,
1422500002, 1.2465499201000054, 0.7524001080999992, 2.254682433599999, 2.244932856099999, 0.13497541210000025, 1.9653516481000024,
9602500002, 0.13466698090000018, 0.11630146090000118, 0.6776417760999975, 0.2910818303999995, 0.04179571359999998, 7.61230172159999
414864810000068, 0.1561277169, 0.9804762361000002, 1.3037985856000005, 3.5447852176000043, 0.02503673289999989, 0.0217651008999999
93183364900001, 18.335524, 4.672082250000004, 7.2085269169, 2.5518465024999952, 79.99084293760002, 7.857369610000004, 8.03274632409
0.129448844100002, 1.877859122500005, 5.9533072036, 4.147291520100002, 0.0003625216000000148, 0.0847916161000016, 1.47719715999
1.849192022499994, 0.41825969290000203, 0.8311968899999995, 2.3043847204, 1.3603256689000007, 0.4139249569000114, 3.5709660899999
20987988900007, 2.0716132760999972, 0.023070572099999957, 4.541800322499992, 2.0508244849000037, 0.13771521000000148, 0.59691075999
0.05538962249999935, 4.9750410304, 0.06698779239999911, 0.006157540899999901, 11.286844968099986, 0.18855569290000107, 0.0390062499
6, 0.1994515599999985, 0.0001439999999999683, 0.5995250041000009, 0.772183987600001, 6.207821402499992, 8.869973062499994, 1.123896
05, 0.018036489999999905, 7.0667183888999965, 0.7906055056000006, 0.004202928900000079, 0.02357453160000014, 0.036496281600000374,
1452900008, 5.538632764900014, 0.16293332249999914, 0.2352056004000002, 16.229456816399985, 0.07354944000000017, 5.550250000000597e
707900624899998, 3.9185390208999946, 4.306704067599986, 0.6429153123999989, 3.738886304399991, 0.3319718688999993, 0.10970006410000
966929852899996, 0.21478590250000074, 3.0042248928999973, 2.1183929209000003, 1.4408641296000018, 1.6043768896000061, 0.83892776490
4.456152121600015, 2.329072776899995, 0.01930432360000045, 3.1003462084000017, 2.493809472399997]
MSE: 2.8087825949885
```

In [29]:

```python
predictions_pr2 = model_pr2.predict(poly_features1.transform(test_data[input_cols]))
MSE_pr2 = mean_squared_error(test_data[["sales"]], predictions_pr2)
print("MSE for degree-2 polynomial regression:", MSE_pr2)
```

```
MSE for degree-2 polynomial regression: 0.8883903174037929
```

In [30]:

```python
predictions_pr10 = model_pr10.predict(poly_features2.transform(test_data[input_cols]))
mse = mean_squared_error(test_data[["sales"]], predictions_pr10)
print("MSE:", mse)
```

```
MSE: 759079.6228268797
```

In [31]:

```
print("model_pr2 has the best MSE (0.3148)")
```

```
model_pr2 has the best MSE (0.3148)
```

In [ ]: